

Machine Learning and Remote Healthcare Monitoring Applied to Dementia Care

I M P E R I A L

Alexander Capstick

Department of Brain Sciences
Imperial College London

This dissertation is submitted for the degree of
Doctor of Philosophy

March 2025

Abstract

With an ageing population, developing new ways to efficiently care for people living with dementia is a top priority. Passive remote healthcare monitoring, in combination with machine learning methods, can enable people living with dementia to be more independent and remain in their familiar home environments for longer, presenting an invaluable opportunity to improve their quality of life. These methods can ensure care is targeted, resources are well utilised, and unplanned hospital admissions are reduced.

This thesis focusses on improving methods for remote healthcare monitoring by providing technical contributions to machine learning and evaluating methods on a real-world case study. The Minder study has deployed passive infrared sensors and bed mats within the homes of ~ 160 people living with dementia, who have collectively generated almost 90,000 days of activity and nocturnal physiology data. In this thesis, we leverage the Minder study to inspire, develop, and evaluate clinically relevant machine learning systems for remote healthcare monitoring, with the goal of improving dementia care.

This work comprises of four main studies: The design and deployment of a risk prediction model for urinary tract infections, which are a significant cause of unplanned hospital admissions for people with dementia; the investigation of settings where data is generated by multiple sources which have differing data distributions or noise levels, and the implications for model training; the evaluation of large language models as knowledge bases within a prior elicitation framework, with the goal of learning predictive models with less data; and the use of language models as sentence encoders to perform representation learning, allowing for clustering, anomaly detection, vector searching, and similarity scoring of daily remote monitoring data.

We conclude by highlighting how the insights provided in this thesis translate to the improved delivery of care for people living with dementia.

Dedication

For my partner Elodie, my parents Tania and Jonathan, and my brothers George, Gerry, Harry, and Danny.

Thank you for your love, support, and time.

Acknowledgements

I express my sincere gratitude to my supervisor, Professor Payam Barnaghi, who has helped shape this project from the start. Payam has been instrumental in my research and has always been a source of new ideas, inspiration, and enthusiasm. I admire his extensive knowledge, creativity, and kindness and feel incredibly lucky to have conducted my doctoral studies with him in the Translational Machine Intelligence research group. I am indebted to the many members of my research group, both past and present, who provided insightful feedback, motivation, and friendship throughout my studies.

I am also grateful to Dr. Rahul G. Krishnan, who welcomed me into his lab at the Vector Institute of the University of Toronto and who supported and guided the research presented in Chapter 6. Rahul is a brilliant, thoughtful, and kind scientist driven by curiosity who is always generous with his time and ideas. The members of the Machine Learning and Computational Healthcare group were always supportive and eager to discuss research ideas, and I count many of them as friends.

I am grateful to Professor David Sharp and Dr Mazdak Ghajari for their help and support in the early stages of this research. I would also like to thank the people at the UK Dementia Research Institute and the Care Research and Technology Centre for their support, resources, and motivation. In particular, I am grateful to the participants in our study who have donated their valuable time and feedback for research.

I would also like to thank my sponsors: The UK Dementia Research Institute (UK DRI) Care Research and Technology Centre funded by the Medical Research Council (MRC), Alzheimer's Research UK, Alzheimer's Society (grant number: UKDRI-7002), and the UKRI Engineering and Physical Sciences Research Council (EPSRC) PROTECT Project (grant number: EP/W031892/1). In addition, infrastructure support for this research was provided by the NIHR Imperial Biomedical Research Centre (BRC) and the UKRI Medical Research Council (MRC). I am also grateful for the funding provided through the Turing scheme in the UK, which allowed me to collaborate with researchers at the University of Toronto. Whilst in Toronto, the resources used in the preparation of this research were provided in part by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute. Without all of their help, this work would not have been possible.

Lastly, I would like to thank my partner, Elodie, for her support, kindness, and love, and my family for their unwavering motivation and guidance. Elodie is an exceptionally determined, clever, and caring person who has been an inspiration throughout my research and an endless source of happiness. My family, who have provided encouragement throughout my education, have shaped me into the scientist that I am today. To them, I owe more than words could describe.

“If we knew what it was we were doing, it would not be called research”
Attributed to Albert Einstein

Statements

Originality:

All work presented in this thesis is my own, except as appropriately referenced.

Copyright:

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-Non Commercial 4.0 International Licence (CC BY-NC)¹.

Under this licence, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work. This is on the condition that you credit the author and do not use it, or any derivative works, for a commercial purpose.

When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes.

Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

¹<https://creativecommons.org/licenses/by-nc/4.0/>

Glossary

Adam Adaptive Moment Estimation

AI Artificial Intelligence

AUC PR Area Under the Precision-Recall Curve

BADL Bristol Activities of Daily Living

BERT Bidirectional Encoder Representations from Transformers

CMHT-OP Community Mental Health Teams for Older People

CR&T Care Research and Technology

FDA Food & Drugs Administration

GCP Good Clinical Practice

GDPR General Data Protection Regulation

GP General Practice

HR Heart Rate

ICL In-Context Learning

ID Identification

IRAS Integrated Research Application System

KDE Kernel Density Estimator

LAP Loss Adapted Plasticity

LLM Large Language Model

LSTM Long Short Term Memory

MAP Maximum a Posteriori

MAR Missing at random

MCAR Missing Completely at random

MCMC Markov Chain Monte Carlo

MLE Maximum Likelihood Estimation

MLP Multilayer Perceptron

MNAR Missing Not at Random

MSE Mean Squared Error

NHS National Health Service

NIHR National Institute for Health and Care Research

NLL Negative Log-Likelihood

NPI Neuropsychiatric Inventory

NREM Non-rapid Eye Movement

PIR Passive Infrared

PIS Participant Information Sheet

PLWD People Living with Dementia

QMS Quality Management System

ReLU Rectified Linear Unit

REM Rapid Eye Movement

RR Respiratory Rate

SGD Stochastic Gradient Descent

SHAP Shapley Additive Explanations

t-SNE t-distributed Stochastic Neighbour Embedding

TIHM Technology Integrated Health Management

UK United Kingdom

UKCA UK Conformity Assessed

UKDRI UK Dementia Research Institute

USA United States of America

UTI Urinary Tract Infection

WASO Wakefulness After Sleep Onset

XAI Explainable AI

Contents

Abstract	i
Acknowledgements	v
Statements	ix
1 Introduction	1
1.1 What is Dementia?	1
1.2 Caring for People Living With Dementia	2
1.3 Remote Healthcare Monitoring	3
1.4 Machine Learning	6
1.5 Unmet Clinical and Care Needs in Dementia	7
1.6 The Minder Study	8
1.7 Methodological Contributions	9
1.8 Thesis Outline	20
2 Background and Theory	21
2.1 The Minder Study	21
2.2 Additionally Studied Datasets	26
2.3 Relevant Methods in Machine Learning	27

2.4	Building Machine Learning Pipelines in This Study	37
2.5	Evaluation of Machine Learning Models in This Study	40
2.6	Mitigating the Effects of Noisy Data in Neural Networks	42
2.7	Large Language Models	44
2.8	Machine Learning with a Bayesian Perspective	45
2.9	Summary	47
3	Related Work	48
3.1	Remote Healthcare Monitoring for Dementia	48
3.2	Predictive Models for UTIs in Dementia Care	49
3.3	Noisy Data in Supervised Machine Learning	50
3.4	Using Language Models for Prior Elicitation	52
3.5	Time-Series Representation Learning	55
3.6	Summary	57
4	Risk Prediction of Urinary Tract Infections	58
4.1	Introduction to the Problem	59
4.2	The Minder UTI Dataset	60
4.3	UTI Risk Prediction Methods	63
4.4	UTI Risk Prediction Results	73
4.5	Discussion	89
4.6	Summary	91
5	Mitigating the Effects of Data Sources with Unknown Reliability	92
5.1	Introduction to the Problem	93
5.2	Noisy data	95

5.3	Why Do We Need New Methods?	96
5.4	Our Proposed Methods	97
5.5	Evaluation of LAP	107
5.6	Discussion	133
5.7	Summary	134
6	Using Large Language Models for Expert Prior Elicitation in Predictive Modelling	135
6.1	Introduction to the Problem	136
6.2	Prior Elicitation with Experts	138
6.3	Linear Models	138
6.4	In-Context Learning	138
6.5	AutoElicit: Using Language Models to Elicit Priors	139
6.6	Energy Distance and Statistic	140
6.7	In-Context Learning	141
6.8	Bayes Factor for Model Selection	143
6.9	Availability of Code	143
6.10	Evaluation of AutoElicit	145
6.11	Using AutoElicit on a New Dataset	173
6.12	Discussion	176
6.13	Summary	177
7	Time-series Representation Learning Using Fine-Tuned Language Models	178
7.1	Introduction to the Problem	179
7.2	The Unlabelled Minder Dataset	180
7.3	Language Modelling	181

7.4	Fine-Tuning Language Models	181
7.5	Large Language Models for Time-Series Representation Learning	182
7.6	The Embedding Space	184
7.7	Clustering	186
7.8	Vector Searching for Similar Days	188
7.9	Inferring Clinical and Care Needs	189
7.10	Discussion	190
7.11	Summary	192
8	Conclusions and Future Work	193
8.1	Extended Applications of This Research	194
8.2	Future Research Directions	197
8.3	A New Remote Healthcare Monitoring System	199
8.4	Lessons Learnt	200
8.5	Concluding Remarks	201
	Bibliography	202

List of Tables

2.1	The Minder study cohort demographics. For each characteristic, we show the number of participants and the corresponding percentage of the cohort. Some participants have requested not to share certain information outside the study and correspond to some of the N/A groups.	26
4.1	UTI prediction results. Mean (95% CI) % of the AUC Precision-Recall of the different models tested with the chosen imputation and scaling methods with 10 bootstrap repeats. The bold row shows the method achieving the best result on the “Date-ID Split” and the chosen model for our final pipeline. Here, LR: Logistic Regression; MLP: Multilayer Perceptron; NB: Naive Bayes; RF: Random Forest; S-Attn: Self-Attention Transformer Encoder; XGBoost: Extreme Gradient Boosting.	75
4.2	UTI prediction results, before and after risk stratification. Mean (95% CI) % of sensitivity, specificity, and area under the precision-recall curve of the UTI prediction model on the different data splits with 10 bootstrap repeats. These results are reported before and after risk stratification is performed.	76
4.3	UTI prediction results for subsets of devices. Mean (95% CI) % of sensitivity, specificity, and area under the precision-recall curve of the UTI prediction model on the different feature subsets with 10 bootstrap repeats.	77
4.4	UTI prediction results, split my sex. Mean (95% CI) % of the accuracy on the male and female participants on the test set of the different data splits with 10 bootstrap repeats. We also show the ratio of positive and negative labels in each demographic and the likelihood of a positive model prediction.	82

4.5	UTI prediction silent-mode testing results. Mean (95% CI) % of the metrics on the test set of the silent-mode testing data with 10 bootstrap repeats of the training set.	85
5.1	Comparison of LAP with the baselines. Mean $\pm$ standard deviation of the maximum test accuracy (%) of 5 repeats of the baselines and LAP on synthetic data with different noisy types. For CIFAR-100 these numbers represent the top 5 accuracy. For CIFAR-10, CIFAR-100, and F-MNIST, the number of noisy sources are 4, 2, and 6 out of 10 respectively. Unreliable sources are each 100% noisy. All values in bold are within 1 standard deviation of the maximum score. Note that IDPA and Co-teaching require almost twice the training time in comparison to Standard, ARFL, CDR, Label Smoothing (LS), and LAP.	118
5.2	Comparison of LAP with the baselines. Mean $\pm$ standard deviation of the percentage difference of the maximum test accuracy (%) of 5 repeats of the baselines and LAP on synthetic data with different noisy types. For CIFAR-100 these numbers represent the top 5 accuracy. For CIFAR-10, CIFAR-100, and F-MNIST, the number of noisy sources are 4, 2, and 6 out of 10 respectively. Unreliable sources are each 100% noisy. All values in bold are within 1 standard deviation of the maximum score.	119
5.3	RRL and RRL + LAP results. Mean $\pm$ standard deviation of the maximum test accuracy (%) of 10 repeats of RRL and LAP on CIFAR-10 data with different types of noise. Here, 4 out of 10 sources are 100% noisy.	120
5.4	Different noise level and number of sources. Mean $\pm$ standard deviation (%) of the <i>percentage difference</i> in maximum test accuracy between RRL + LAP and RRL (2021) over 5 repeats when training a model on CIFAR-10 for different noise levels and numbers of sources. Here a positive value represents an improvement in accuracy when using LAP. Here, U corresponds to the number of unreliable sources out of 10 sources in total. Batch label flipping was used to introduce noise.	120
5.5	Baselines and LAP results on a natural language task. Mean $\pm$ standard deviation of maximum test accuracy (%) of 5 repeats of the methods on the IMDB dataset with different types of noise. Here, 4 out of 10 sources are 100% noisy.	124

- 5.6 **Results on GoEmotions, an imbalanced sources dataset.** Mean $\pm$ standard deviation of maximum test top-5 accuracy (%) over 5 repeats of the methods on the GoEmotions data with noisy labels and random sentence permutation. Here, 30 out of 82 sources are 100% noisy. 126
- 5.7 **Comparison of LAP with the baselines with varied numbers of sources.** Mean $\pm$ standard deviation of maximum test accuracy (%) of 5 repeats of the baselines and LAP on synthetic data with different noisy types and numbers of sources. The number of noisy sources ranges from 10 to 50,000, corresponding to “LAP- n ”, where n denotes the number of sources in the training data. For all experiments, 40% of the sources were chosen as unreliable, with 100% noise rate. Since there are 50,000 training data points, LAP-50,000 corresponds to LAP acting over sources that have size 1: i.e the standard noisy data setting. 126
- 5.8 **Difficult data results.** Mean $\pm$ standard deviation of maximum test accuracy (%) over 5 repeats of standard training and LAP on a combination of MNIST and CIFAR-10 data with noisy labels. Here, 95 out of 100 sources are 100% noisy. 128
- 5.9 **Difficult data results on the CIFAR-10 classes.** Mean $\pm$ standard deviation of maximum test accuracy (%) on the CIFAR-10 classes over 5 repeats of standard training and LAP trained on a combination of MNIST and CIFAR-10 data with noisy labels. This is the accuracy on just the CIFAR-10 test set, which is a subset of the test set corresponding to Table 5.8. 128
- 5.10 **Tiny Imagenet results.** Mean $\pm$ standard deviation of maximum test top-5 accuracy (%) over 5 repeats of standard training and LAP on Tiny Imagenet data with noisy labels. Here, 40 out of 100 sources are 100% noisy. 129
- 5.11 **Imagenet results.** Mean $\pm$ standard deviation of maximum test top-5 accuracy (%) over 5 repeats of standard training and LAP on Imagenet data with noisy labels and noisy inputs. Here, 5 out of 10 sources are 100% noisy, with three of them containing label noise and 2 containing input noise. 130
- 5.12 **Imagenet late training results.** Mean $\pm$ standard deviation of test top-5 accuracy (%) over the last 10 epochs of standard training and LAP on Imagenet data with noisy labels and noisy inputs, repeated 5 times. Here, 5 out of 10 sources are 100% noisy, with three of them containing label noise and 2 containing input noise. 131

5.13	Cross-validation performance on the UTI prediction task, split by sex. An MLP model is trained on the Minder UTI dataset using a method for multi-task learning. This shows the mean $\pm$ standard deviation (%) of the accuracy of 5-fold cross-validation, where participants only appear in either the training or testing set.	132
6.1	GPT-3.5-turbo header completion test results. Normalised Levenshtein distance between the correct header completion and the language model’s completion for each of the datasets tested. A value of 0 indicates that the true completion and language model’s completion were equal and a value of 1 means that they were different at all positions. .	148
6.2	GPT-3.5-turbo row completion test results. Normalised Levenshtein distance between the correct row completion and the language model’s completion for each of the datasets tested. A value of 0 indicates that the true completion and language model’s completion were equal and a value of 1 means that they were different at all positions.	148
6.3	Approximated GPT-3.5-turbo in-context prior compared to the elicited prior. The energy statistic between the LLM’s approximate in-context prior, and the LLM-elicited prior. Since we extract a single prior distribution for both methods, no confidence intervals are given here. Lower values indicate that the two distributions are more similar. . . .	168
6.4	Difference between approximated posterior and MC on the approximated prior for GPT-3.5-turbo. The energy statistic between the language model’s internal posterior extracted through MLE sampling and the posterior samples from applying Monte Carlo techniques to the language model’s internal prior. Values are $\in [0, 1]$ with values closer to zero, indicating that the two distributions are more similar. Here, we show the mean and standard deviation of the energy statistic for the 5 different training sets tested as observed data.	171
7.1	Silhouette score for k-means clustering on the base and fine-tuned models.	186

List of Figures

- 2.1 **Illustration of activity and sleep sensor deployment within the home of a person living with dementia, from Capstick et al. (2024) [1].** Here, passive infrared sensors are located in each room for in-home activity monitoring. A sensor is located under the mattress for analysing physiology and movements in and out of the bed. 23
- 2.2 **Number of days recorded, updated from Capstick et al. (2024) [2].** Histogram showing the number of participants with a given number of days of data recorded. 23
- 2.3 **Dataset characteristics from 2021/07/01 to 2025/01/01.** (a) shows the number of participants who had activity or sleep data collected from them each week; (b) illustrates the length of time each participant produced activity or sleep data within the study. Here, the line joins the first and last days a given participant had data recorded for them; (c) shows how many days of UTI were labelled in this time. 24
- 2.4 **Time of location recordings, updated from Capstick et al. (2024) [2].** Histogram showing the number of sensor recordings by time of day, with locations that are ordered by their total frequency. 24
- 2.5 **Sleep staging by time of day.** This histogram shows the different sleep stages measured by the sleep mat by the time of day. 25
- 2.6 **Heart rate and respiratory rate between 2021/07/01 to 2025/01/01.** (a) shows a histogram of the heart rate measurements, whilst (b) shows the respiratory rate. 25
- 4.1 **Dataset characteristics.** (a) The cumulative number of sleep and activity data points recorded in total. (b) The cumulative number of clinically verified UTI labels over time. (c) The distribution of UTI labels by participant. We see that the vast majority of participants have few verified labels, which will reduce bias from overfitting in the model. 61

- 4.2 **The silent-mode dataset.** (a) Shows the amount of data collected each day, and whether it corresponds to the train or test set. (b) Illustrates the cumulative number of verified UTI labels (without extending them by 3 days either side) collected in the same time frame. 62
- 4.3 **Label distribution by day.** (a) The days for which we have verified UTI labels. (b) The distribution of these labels after they are extended by 3 days on either side. 62
- 4.4 **The Dataset.** (a): All PIR activations measured and the corresponding time of day. (b): The average proportions of PIR activations in a time period for days containing a verified positive and negative UTI label for a selected participant. The difference in Bathroom activity is of note, showing a significant increase in use for those days corresponding to a positive UTI. (c): All sleep state measurements within the dataset and the corresponding time of day. (d): The first of the graphs shows the percentage of a given time window a single participant spent in bed and awake; the second and third show the difference in heart rate and respiratory rate distributions for days labelled as positive and negative UTI for a single participant. The elements of the boxplots correspond to: centre line, median; box limits, upper and lower quartiles; whiskers, 1.5x interquartile range; points, outliers. Here, HR corresponds to Heart Rate and RR corresponds to Respiratory Rate. 63
- 4.5 **Patient disposition for the imputing, training, and evaluating stages of our analysis.** 65
- 4.6 **The evaluation procedures visualised.** (a) “Date Split”. (b) “Date ID Split” with the Leave-One-Out evaluation acting over the PLWD. 66
- 4.7 **The model reliability.** The top row shows the model confidence (on positive and negative UTI cases) against accuracy on the test set of “Date Split” and “Date-ID Split”. The gap shows the difference between the average accuracy and confidence of a bin, which would ideally be 0. The bottom row shows the histogram of confidences reported by the model on the test set of “Date Split” and “Date-ID Split”. 74
- 4.8 **The model calibration.** The mean UTI predicted risk against the proportion of positive UTI cases for the best model. This is plotted for the validation and test set from the “Date Split” and “Date-ID Split” data. The error bars represent the standard deviation of the values from the 10 bootstrap repeats. 76

- 4.9 **SHAP values.** (a): The feature importance for the top 10 most important features, as calculated by SHAP on the test set, and their corresponding feature values. The colour represents the normalised feature value, whilst the position in the x -axis represents the contribution that value made to the prediction. “MA” refers to the moving average, whilst “Delta” refers to the percentage change in the value from the previous day. These values are calculated on the test set of the “Date-ID Split”. (b): The breakdown of a single prediction shows how each feature contributed to a correct prediction of a positive UTI. Here, the values on the arrows correspond to the normalised feature value in units of standard deviations away from the mean. 78
- 4.10 **SHAP values for individual predictions.** In both figures, “MA” refers to the moving average, whilst “Delta” refers to the percentage change in the value from the previous day. Additionally, all figures show the breakdown of single predictions and how each feature contributed to that prediction based on SHAP values. Here, the values on the arrows correspond to the normalised feature value in units of standard deviations away from the mean. These were calculated on the “Date-ID Split” test set. 78
- 4.11 **SHAP values projected on the house layout for a single correct positive prediction.** The intensity of the colours shows the relevance of the sensory values to the UTI risk scores (orange representing positive correlations and blue representing negative correlations). . 79
- 4.12 **Sensitivity and specificity with different thresholds.** The variations in sensitivity and specificity that can be achieved on the validation set by changing the thresholds for defining the Green and Red groups. The line colours represent different threshold values. Sensitivity and specificity are calculated on the data in the validation set corresponding to Red and Green alerts. Here, to calculate the metrics on the Green group, the Red threshold was set at $> 50\%$, and when calculating the metrics on the Red group, the Green threshold was set at $\leq 50\%$. This figure shows the average results from the “Date Split” and “Date-ID Split”. 79
- 4.13 **Metrics for different thresholds.** The sensitivity and specificity of the model on the “Date Split” and “Date-ID Split” validation sets as the green and red thresholds are varied. Here, the results are coloured by the result rounded to the nearest 5% for ease of visualisation. 80

- 4.14 **UTI risk scores on “Date-ID Split” test set.** The risk scores as calculated by the best model on the test set of the “Date-ID Split”, with the thresholds corresponding to the horizontal coloured bands and the vertical coloured bands representing positive and negative UTI diagnoses (in Red and Green respectively). The error bands represent the 95% confidence interval (1000 bootstrap samples) of the mean of the values from the 10 bootstrap repeats. 81
- 4.15 **Performance at days prior to UTI label.** The performance of the model when tasked with analysing the risk of a UTI from the test set, at different numbers of days prior to the verified recorded date. Note that the features contain the number of previous UTIs, which are updated with a delay of 3 weeks to ensure no data leakage. The error bands represent the 95% confidence interval (1000 bootstrap samples) of the mean of the values from the 10 bootstrap repeats. 82
- 4.16 **Characteristics of UTIs in the test set. (a)** Histogram showing the positively labelled UTIs and the number of days they were measured before the start of the test set. **(b)** The same data as (a), presented as a minimum number of days for each participant. 83
- 4.17 **Performance by time since the most recent labelled UTI and number of UTIs. (a)** The AUC PR of the predictions on participants (on the test set of “Date Split”) that had their most recent UTI positive label less than the given number of weeks before the test set. The error bands represent the 95% confidence interval (1000 bootstrap samples) of the mean of the values from the 10 bootstrap repeats. **(b)** The accuracy (on the test set of “Date-ID Split”) split by the number of previous UTIs. The labels on the bars represent the number of positively and negatively labelled data points (in the test set) for the corresponding group. The error bars represent the 95% confidence interval (1000 bootstrap samples) of the mean of the values from the 10 bootstrap repeats. 84
- 4.18 **Learning curve on labelled data.** The cross-validation performance of the model on labelled data collected between 2021/06/28 and 2024/01/01, as the size of the training set increases. In each fold, we select the training and validation data in the ratio 4 : 1. We then randomly sample an increasing number of labelled data points from the training set to use for model training. Here, the horizontal axis represents the verified UTI days without extending their labels by three days on either side. We repeat this 10 times and report the standard deviation of results as error bands. 86

4.19	Metrics under different thresholds. The number of weekly red alerts, and their respective precision and sensitivity as the threshold for red alerts increases. The dashed vertical line represents the proposed threshold of 0.75.	86
4.20	The distribution of the predicted risk score on the labelled days in the test set. The risk score predicted on those verified UTI positive or negative days within the test set. The dashed vertical line represents the proposed threshold of 0.75.	87
4.21	The number of participants receiving a given number of alerts over 2023 - 2024. The number of participants that have received each number of green or red alerts between 2023 and 2024. This shows that the majority of participants receive very few red alerts, however a handful of participants receive many.	87
5.1	Loss Adapted Plasticity. Our method dynamically re-weights source contributions during training by comparing the historical loss values.	94
5.2	Comparing source noise estimation with and without source knowledge. This illustrates that knowing which source generated a data point provides important information about its probability of being noisy.	97
5.3	The python package. <code>pip install loss_adapted_plasticity.</code>	106
5.4	Visualisation of Equation 5.3. Each colour represents the loss values from a single source over a small number of steps, with its density weighted by its temperature, $w(s) = f(C_s)$. This shows how sources contribute to $\mu_{s'}$ and $\sigma_{s'}^2$ as their C_s changes during training and given the leniency λ . These values are synthetic and for demonstration.	113
5.5	Effect of the introduced parameters on training. Section 5.4, introduces three parameters that control the effects of LAP. $1 - d_s$ is multiplied by the gradient (equivalently, loss) contribution from a given source before the model is updated. Here, we show these values for each source (the different coloured lines) during model training on synthetic data. Unless stated in the title of a given plot, the parameters of LAP were set to $H = 25$, $\delta = 1.0$, $\lambda = 1.0$. We had 5 sources with noise levels of 0.0, 0.025, 0.05, 0.25, and 1.0 (a darker colour indicates a higher noise rate).	114
5.6	Sensitivity to Hyperparameters. Here, we demonstrate the sensitivity of the model performance based on the LAP hyperparameters introduced in Section 5.4 by varying their values whilst keeping the others fixed at $(\delta, H, \lambda) = (1.0, 50, 1.0)$	115

- 5.7 LAP results with a varied number of sources and noise levels.** In (a) we show the area under the precision-recall curve for standard training and using LAP on PTB-XL with label noise and simulated ECG interference noise for 12 total sources. In (b), we show the accuracy on CIFAR-10N with real human labelling noise when using RRL and RRL + LAP, with 10 sources. In both, the noise of the sources varies linearly from 25% to 100% for each number of noisy sources. The lines and error bands represent the mean and standard deviation of the maximum value for each of the 5 repeats. These figures illustrate that LAP maintains higher performance as noise rates increase. 121
- 5.8 Accuracy values on CIFAR-10N for an increasing number of noisy sources with a lower capacity model.** The lines and error bands represent the mean and standard deviation of the maximum test accuracy for each of the 5 repeats with random allocation of noisy sources. The noise of the sources increases linearly from 25% to 100% for each number of noisy sources. In total, there are 10 sources. Here, we test the low capacity CNN (Section 5.5.4). 123
- 5.9 Source and class distributions in GoEmotions.** In (a), we present the number of sources with a given size, plotted on a log scale. This demonstrates the imbalance in the size of the sources in the dataset, with some containing thousands of data points, whilst others contain just a few hundred. In (b), we additionally explore the number of classes within each data source, showing that around half of all sources do not contain all classes, with two sources containing less than 5 classes. 124
- 5.10 Performance on ImageNet.** In (a) we show the cross-entropy loss on the training data at each epoch where the loss using LAP represents the tempered cross-entropy loss. Note that in this figure, the training loss for LAP appears lower due to the weighting of sources we use. This figure shows that both training loss curves do not look unusual. In (b) we present the validation top-5 accuracy at each epoch. In (c) we present the test top-5 accuracy at each epoch to demonstrate the fitting to the noise that occurs when not using LAP during late training. Here, 5 out of a total of 10 sources are 100% noisy, with 2 containing input noise, and 3 containing label noise. The two steps in performance occur at the points at which we scale the learning rate using a scheduler (epoch 30 and 60). The lines and error bands represent the mean and standard deviation over the 5 repeats. The vertical black dashed line represents the epoch at which LAP achieved maximum top-5 accuracy on the test set. 131

- 6.1 **AutoElicit: Prior elicitation using a language model.** The left-hand figure demonstrates the process, whilst the right-hand figure illustrates the benefits of using AutoElicit, achieving the same peak accuracy after $n \approx 15$ labels; 220 days earlier in the study. . . . 137
- 6.2 **The AutoElicit python package: `pip install autoelicit`.** 143
- 6.3 **GPT-3.5-turbo-elicited priors for the synthetic task, $y = 2x_1 - x_2 + x_3$.** We sample 10,000 parameter values from the mixture generated from the Gaussian priors elicited from GPT-3.5-turbo when given complete task descriptions. This figure shows the histograms of those values in each feature dimension. 151
- 6.4 **Test performance of the posterior distribution for varied training data sizes.** The average and 95% confidence interval of the mean posterior accuracy or MSE of 10 splits of the dataset using AutoElicit priors and an uninformative prior ($\theta \sim \mathcal{N}(\theta|0, 1)$). These are calculated on a test set of each dataset, with the green arrow pointing in the direction of metric improvement. 151
- 6.5 **Posterior accuracy of the elicited priors on the UTI dataset considering label collection time.** The average mean posterior accuracy and 95% confidence interval on the UTI test set after observing the given number of data points against the time it took for the data points to be collected in the study. The horizontal axis shows the time in days to achieve the given performance and the number of labels collected in that time. 153
- 6.6 **Test accuracy of the posterior distribution over the model weights of a logistic and linear regression model for varied numbers of training data points.** We evaluate the accuracy and MSE of different prior distributions over the weights of a logistic and linear regression model. Each of the Gaussians in the mixture is extracted from a language model and is tested against an uninformative prior $\mathcal{N}(\theta|0, 1)$. These results are calculated on a test set of the datasets after each model is trained on the given number of data points. These values are the result of 5000 sampled parameters and 10 random splits of the training and testing data. The green arrows point in the direction of improvement in the metric. 154
- 6.7 **Further comparison with a “wider” uninformative prior** The average and 95% confidence interval of the mean posterior accuracy or MSE of 10 splits of the dataset using AutoElicit priors and two uninformative priors, $\theta \sim \mathcal{N}(\theta|0, 1)$ and $\theta \sim \mathcal{N}(\theta|0, 100)$. These are calculated on a test set of each dataset, with the green arrow pointing in the direction of metric improvement. 155

- 6.8 **Measuring how similar the GPT-3.5-turbo-elicited prior distributions are with fewer task descriptions.** We reduce the number of task descriptions and calculate the energy statistic between 10,000 samples of the prior elicited from the language model using the reduced set of descriptions and all of the 100 previous task descriptions. We repeat each experiment 10 times and show the mean and standard deviation. 156
- 6.9 **AutoElicit with different numbers of mixture components.** This figure presents the posterior accuracy or mean squared error of a linear model after observing the given number of training data points, over 10 repeats. The rows of the figure correspond to the different language models tested, and the colour of the lines corresponds to the number of mixture components in the prior. 156
- 6.10 **Test accuracy of the posterior distribution for some OpenAI language models.** Here we show the posterior results when using priors elicited from two other OpenAI language models and an uninformative prior $\mathcal{N}(\theta|0, 1)$. These results are calculated on a test set of the datasets after each model is trained on the given number of data points. The values are the result of 5000 sampled parameters for each of 5 chains, and 10 random splits of the training and testing data. For the regression tasks, we report the MSE on log-scale. This figure shows the line plot of the average mean posterior accuracy or MSE over the 10 splits, with the error bars representing the 95% confidence interval. The green arrows point in the direction of improvement in the metric. 158
- 6.11 **Test accuracy of the posterior distribution for open-source language models.** Here, we show the results when using priors elicited from two of the Llama language models and an uninformative prior $\mathcal{N}(\theta|0, 1)$ on a linear model. These results are calculated on a test set of the datasets after each model is trained on the given number of data points. The values are the result of 5000 sampled parameters for each of 5 chains, and 10 random splits of the training and testing data. For the regression tasks, we report the MSE on log-scale. This figure shows the line plot of the average mean posterior accuracy or MSE over the 10 splits, with the error bars representing the 95% confidence interval. The green arrows point in the direction of improvement in the metric. 160

- 6.12 MSE using the GPT-3.5-turbo-elicited priors for the task: $y = 2x_1 - x_2 + x_3$ where we have used increasingly more detailed task descriptions.** Here, feature values are generated using $X \sim \mathcal{U}(-5, 5)$ and the targets are calculated using $y = 2x_1 - x_2 + x_3$. Within each task description, we provide the language model with increasingly more information about the task to test how the descriptions affect the priors elicited. The lines depict the average mean posterior MSE over the 5 dataset splits, whilst the error bars represent the 95% confidence interval. 162
- 6.13 GPT-3.5-turbo-elicited parameter distribution as increasingly more information is included.** These histograms show the distribution of parameter values for the features as more expert information is provided. In each row, we provide information in the task description of an additional feature, with the green arrow indicating the effect this has on the parameter distribution. In particular, in the first row we provide information about the relationship between the target and the first feature, in the second row we provide information about the second feature, and the in the third row we provide information about the third feature. 164
- 6.14 UTI prior parameters with expert information.** The histograms in the three left hand plots show the distribution of parameter values for the features that we provide expert information for in the task description. In all three cases, we state that the features are positively correlated with UTI risk. In the right-hand plot, we show the posterior accuracy of these distributions for different numbers of observed samples. The average and 95% confidence interval of the mean posterior accuracy over the 10 test splits are shown. . . . 165
- 6.15 Extraction of GPT-3.5-turbo's in-context prior.** The bottom row shows the MSE between the LLM's in-context predictions and the MLE model's predictions, whilst the top row shows the parameter distribution of these MLE models, with each colour representing a single feature. For ease of visualisation, we exclude values outside of the (2.5%, 97.5%) percentiles and the bias term. 166
- 6.16 Extraction of GPT-3.5-turbo's in-context posterior.** The bottom row shows the MSE between the LLM's in-context predictions and the MLE model's predictions, whilst the top row shows the parameter distribution of these MLE models, with each colour representing a single feature. For ease of visualisation, we exclude values outside of the (2.5%, 97.5%) percentiles and the bias term. 166

- 6.17 GPT-3.5-turbo in-context predictions for the task: $y = 2x_1 - x_2 + x_3$ compared to the true values.** Here, feature values are generated using $X \sim \mathcal{U}(-5, 5)$ and the targets are calculated using $y = 2x_1 - x_2 + x_3$, which is specified fully in the in-context prompt. The language model is then asked to make predictions on X . This shows the true targets y against those predicted by the language model, as well as the binned root MSE. 167
- 6.18 Distribution of the prior parameters under elicitation and those approximated from the internal model for the different predictive tasks for GPT-3.5-turbo.** Here, we compare the prior distributions of the different predictive tasks, as defined by the internal model used in-context, and that which we elicit from the language model. Here, we draw the histogram separately in each dimension, but calculate the energy distance across all features combined. 170
- 6.19 The in-context posterior distribution for the tasks with low errors.** We compare the posterior extracted from GPT-3.5-turbo's in-context predictions against the distribution that we sample using Monte Carlo methods on the extracted prior distribution. Both posterior distributions are based on the same 25 data point sample from the corresponding dataset. We draw the histogram separately in each dimension, but calculate the energy distance across all features combined. For the diabetes dataset we have removed the long tails for improved visualisation. 171
- 6.20 The different posterior samples for GPT-3.5-turbo for the tasks with high errors.** Here, we compare the distribution of the posterior extracted from the language model's in-context predictions against the distribution that we sample using Monte Carlo methods on the extracted prior distribution. Both posterior distributions are based on the same 25 data point sample from the corresponding dataset. We draw the histogram separately in each dimension, but calculate the energy distance across all features combined. 172
- 6.21 The distribution of the prior marginal likelihood and the Bayes factor for GPT-3.5-turbo.** This shows the distribution of prior likelihoods, calculated using 25 data points from each dataset, over 5 test splits and 100 samples of prompts. The Bayes factor is calculated between prior elicitation and in-context learning, with a number greater than 1 favouring prior elicitation. For the regression tasks, we exclude values outside of the (2.5%, 97.5%) percentiles in the histograms. 173

7.1	Number of days recorded. Histogram showing the number of participants with a given number of days of data recorded.	180
7.2	Time of location recordings. Histogram showing the number of sensor recordings by time of day, with locations that are ordered by their total frequency.	180
7.3	Preprocessing of data. For a single participant, we present their data collected over a single day. The left graph shows the raw measurements; the central graph shows the measurements after the mode of each 20 minute window is taken and a token for no activity is assigned (“Nowhere”); and the right graph shows the day as a single text string that can be interpreted by a language model.	183
7.4	t-SNE of embeddings and journeys. The t-SNE embeddings of 1000 randomly sampled days of data (in grey) with the embeddings of a given participant’s data overlayed (in blue) with consecutive days connected by a line. Here, we show the 25 participants with the most recorded days of data.	185
7.5	Days of data in cluster 5. We present 8 randomly chosen days of data from cluster 5, which is characteristic of sparse recorded activity.	187
7.6	Days of data in cluster 1. We present 8 randomly chosen days of data from cluster 1, which is characteristic of frequent bedroom and kitchen activity.	188
7.7	t-SNE of embeddings. The left plot shows the t-SNE transformation of 5000 day-string embeddings, coloured by their cluster value. On the upper row of the right-hand figures, we show the t-SNE embeddings for the 4 participants with the most data, where a line indicates consecutive days. On the lower row, we show the daily cluster values in time for the same participants.	189
7.8	Cluster proportions. For each day in the dataset, we show the proportion of days clustered with each label.	190
7.9	Vector search over days. Given a day of data (shown in purple in the top left), we present the 9 most similar other days as measured by cosine similarity (shown in blue).	191
7.10	Day similarity. Each image shows the cosine similarity between every 20th recorded day for 10 participants. Blue, white, and red correspond to a cosine similarity of 1, 0, and -1 respectively. The range of the days is represented by each axis of the plots and is given by n .	192

Chapter 1

Introduction

1.1 What is Dementia?

One in three people born in the United Kingdom (UK) today will be diagnosed with dementia in their lifetime [3]. Furthermore, 982,000 people in the UK are currently estimated to have dementia, with this figure due to rise to 1.4 million by 2040 [4]. With an ageing population, this places a significant pressure on health services. In particular, the cost of dementia in the UK was forecast to be £42 billion in 2024, and is expected to increase to £90 billion in 2040. Worldwide, in 2015 approximately 5% of people aged 60 years and older were living with dementia, amounting to 47 million people. This is due to increase to 131.5 million people by 2050 [5].

Dementia is a group of symptoms caused by different diseases that affect the brain. These symptoms can include memory loss, confusion, difficulties in language and understanding, and changes in behaviour. Alzheimer's disease is the most common form of dementia and is estimated to account for 60 – 70% of cases [6]. Other major forms of the disease include vascular dementia, dementia with Lewy bodies, and frontotemporal dementia. Patients can often exhibit mixed forms of dementia, and around 95% of people with dementia have at least one of these four main forms.

There is currently no cure for dementia, making the delivery of effective care paramount to quality of life. However, due to the varied clinical presentation of dementia, care can be challenging to plan, requiring personalisation and frequent review.

1.2 Caring for People Living With Dementia

Dementia care is currently limited by the reactive way it is delivered. Due to resource constraints, support services often cannot intervene until crises occur and, as dementia progresses, lack the capacity to plan for predictable and preventable adverse health conditions. This includes mitigating the risk of common conditions such as infection, neuropsychiatric symptoms, social isolation, and functional decline [7, 8].

Dementia care can also vary considerably by geography. In the last 12 months, on average 84.9% of people in England diagnosed with dementia attending a National Health Service (NHS) general practice (GP) received a face-to-face care review or care interventions, however, in some regions this number can be as low as 17.4% [9].

In England, 25% of hospital beds are estimated to be occupied by people with dementia [10]. Many of these admissions are due to potentially preventable causes, such as Urinary Tract Infections (UTIs), chest infections, falls, and episodes of delirium. In particular, Sampson et al. (2009) [11] suggested that UTIs could account for as many as 9% of admissions alone.

Family members account for the majority of caregivers for people with dementia. In the United States of America (USA), 75% of people with dementia are cared for by their family or friends [12, 13]. In 2007, this translated to approximately 10 million people in the USA who were caring for a person with dementia. Providing care can be a significant burden on caregivers and often affects their mental wellbeing. In particular, Koyama et al. (2017) [14] found that those who provided care for a family member with dementia had significantly lower mental quality of life compared to those who did not. Furthermore, ensuring that people with dementia receive consistently good care presents its own challenge, as efforts can be difficult to coordinate between healthcare professionals and family members. Often, professional care or training can be expensive, preventing some families from seeking outside help.

New technologies for remote healthcare monitoring, utilising low-cost devices and machine learning, can facilitate the improvement of preventative methods and the early detection of adverse health conditions, as well as enable more efficient care delivery. These technologies provide a strong case for addressing the current challenges facing dementia care.

The Minder study, conducted by the Care Research and Technology Centre (CR&T) within the UK Dementia Research Institute (UK DRI), aims to combine knowledge from researchers working in data

analysis with clinicians and families providing care for those living with dementia. The goal of the study is to reduce the cost of care delivery and enable those with dementia to stay within their homes for longer, improving their quality of life.

1.3 Remote Healthcare Monitoring

Remote healthcare monitoring, often referred to as remote patient monitoring or connected health, is becoming an increasingly viable method of providing healthcare. This is attributed to innovations in wearables, low-cost monitoring devices, machine learning, mobile phones, and networking, which have all considerably reduced the cost of collecting, storing, and processing data [15]. These advances allow for scalable and continuous monitoring and analysis of patient wellbeing, activity, and physiology.

As the cost of these technologies reduces, the population ages [16], and the efficacy of data analysis methods improve, the demand for remote healthcare monitoring will increase considerably [17]. Additionally, the NHS reports that between November 2020 and January 2023, over 487,000 people in England had been supported with digital home care and remote monitoring technologies through national funding as part of the Regional Scale Programme [18]. These services were of particular importance during the COVID-19 pandemic, when this Regional Scale Programme facilitated the adoption of new care pathways supported by remote healthcare monitoring [19].

Remote healthcare monitoring has been evaluated for use at all stages of human life. Sasangohar et al. (2018) [20] discussed the uses of passive sensors in neonatal and paediatric settings. For example, studies have shown their effectiveness in monitoring vital signs within neonatal and paediatric intensive care units [21] and in the research and care of paediatric sepsis [22]. In contrast, Olmedo-Aguirre et al. (2022) [23] explored advances in methods of remote healthcare monitoring for older adults, which were shown to be useful for detecting chronic illness.

However, Peyroteo et al. (2021) [24] found that the adoption of wearable devices was difficult in practice, due to the challenges of adapting to the specific needs of patients and healthcare professionals, limiting the scalability of these studies. The use of passive sensors for remote healthcare monitoring, in place of wearables, offers an opportunity to resolve this issue, since they require little involvement from the study participant, which could provide greater adoption.

1.3.1 Remote Healthcare Monitoring Devices

There are many devices on the market for collecting behavioural and physiological data from study participants. Within the literature on remote healthcare monitoring, studies primarily focus on using either wearables, or passive sensors.

Wearables

Wearable devices are those that are designed to be worn on the body of the study participant to collect data. There are a wide range of wearable devices on the market that have been designed to be mounted on a variety of body locations, providing an array of options in the types of devices that can be used in a study. Some examples include smart-watches or fitness trackers, smart-hearing aids, smart-glasses, smart-clothing, and smart-patches. These devices are usually designed to collect activity or physiological data from the wearer, which can be analysed to provide insights into their health or fitness.

Many studies have focused on providing wearables to participants for the diagnosis of adverse health conditions, chronic illness, or for measuring biomarkers in the older adult population [23]. For example, Leirós-Rodríguez et al. (2019) [25] and Rucco et al. (2018) [26] studied the use of wearables for measuring accelerometer data to diagnose alterations in the balance of elderly people or detect falls, respectively. In contrast, Baig et al. (2017) [27] and Guo et al. (2021) [28] reviewed the advances and opportunities for using wearables to diagnose chronic diseases, whilst Schalkamp et al. (2023) [29] used accelerometer data to predict a diagnosis of Parkinson's Disease. These studies have demonstrated varied effectiveness in their use of wearables to provide an estimated risk of adverse health conditions remotely.

Wearable devices have an inherent limitation that makes their use challenging in study cohorts with cognitive impairment: Wearable devices require involvement from their wearer, either for charging, for interaction, or by simply needing to be worn. This suggests the use of a different type of remote monitoring device for study cohorts where the deployment of wearable devices might be ineffective.

Passive Sensors

Passive sensors are designed to require minimal or no interaction with the user and collect activity or physiological recordings from participants continuously, without being mounted to the subject's body.

These sensors exist in a variety of forms based on their purpose. Within in-home remote healthcare monitoring, they can be used to collect activity data, physiological data, or information about an environment.

Yue et al. (2025) [30] discussed the design of implantable passive sensors for healthcare applications. The authors describe a wide range of studies that use these sensors to collect information about participants, including devices for collecting physiological measurements and monitoring body fluids. However, such sensors are invasive and can be expensive to install. In contrast, Alemdar and Ersoy (2010) [31] discussed the use of wireless and passive sensor networks for in-home healthcare monitoring. The authors envisioned a future where low-cost passive sensors could be placed around the homes of older adults to measure activity. Based on recorded data, healthcare professionals would be able to plan the delivery of care and be alerted to adverse health conditions.

Earlier versions of our study setting put these ideas into practice and employed passive sensors for in-home remote healthcare monitoring. Enshaeifar et al. (2018) [32], and later Enshaeifar et al. (2019) [33], studied the use of in-home passive infrared (PIR) sensors for the monitoring of people living with dementia. The authors found initial evidence that using these activity sensors could allow for the detection of agitation events and UTIs. However, this work was limited to using passive sensors for the detection of activity only, as it required study participants to take their own physiological measurements. Despite this, the study demonstrated that sophisticated in-home monitoring systems and machine learning could empower people living with dementia by allowing them to remain within their homes for longer.

1.3.2 Data Collection

Due to the availability of low-cost passive sensing devices, the collection of in-home observational data is feasible on a large scale. At this level of data collection, automated methods are required for analysis, detection of adverse health conditions, and to provide meaningful insights for clinicians. For the effective detection of adverse health conditions, labelled examples are usually required. However, labelling data for adverse health conditions still provides a significant challenge. For this, clinicians must annotate collected data, which often requires the collection and analysis of biological samples.

Machine learning methods offer an opportunity to alleviate some of these issues by automating the processing of unlabelled data and allowing for more targeted clinical interventions.

1.4 Machine Learning

Machine learning is popularly defined as follows [34]:

A computer program is said to learn from experience E with respect to some class of tasks T , and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E .

Within our work, we mainly restrict T to represent tasks within healthcare and E to be data collected as part of the Minder study or publicly available datasets. With these restrictions, this definition describes a programme that, given historical healthcare data, can aid clinicians in producing insights or predictions on newly collected data.

This field, even within the duration of this doctoral study, has seen considerable advances and expansions in new areas¹. This rapid growth can be mainly attributed to two factors:

- Processing units continue to allow for faster compute [35] and specialised architectures are being included in products for more efficient machine learning workflows².
- As a population, due to the increasing number of connected devices [36], we are producing more data than ever before [37]. This scale of data requires the need for more sophisticated automated analysis systems.

Therefore, machine learning is an active area of research within many domains, including healthcare, as it allows for more sophisticated pattern analysis on large datasets. In healthcare, the study of machine learning methods in research is vast [38], however, barriers still remain for their frequent deployment [39]. Two of the key obstacles discussed by Ghassemi et al. (2020) [39] limiting the deployment of machine learning methods in healthcare are the presence of noise within datasets and the difficulty in producing explainable and reliable outcomes. Benjamens et al. (2020) [40] studied the US Food & Drugs Administration (FDA) approved medical devices and algorithms and find that of these technologies, 71.9% focused on either Radiology or Cardiology; demonstrating a current narrow application. They identified that some of the key challenges for the implementation of machine learning methods in daily clinical practice include the issues of transparency, inherent bias in training data, and security.

¹<https://www.nature.com/natmachintell/>

²<https://www.nvidia.com/en-gb/data-center/tensor-cores/>

1.5 Unmet Clinical and Care Needs in Dementia

Currently, major challenges limit the use of remote healthcare monitoring in the routine delivery of care for people living with dementia.

Wearable remote monitoring devices often cause challenges for people living with dementia by requiring effort for charging, causing discomfort, or needing involvement from the participant for data collection. To mitigate this issue, our study makes use of passive sensors for data collection, requiring no effort from participants to collect activity or physiological measures. These sensors are not in physical contact with the person living with dementia, allowing them to feel more independent.

Remote healthcare monitoring generates a considerable amount of data; especially when sensors that can continuously collect data are deployed. This scale of data collection necessitates the use of automated methods for analysis and the extraction of insights for care delivery. Therefore, remote healthcare systems should contain methods for encoding large amounts of time-series data to allow clinicians to view changes in patient behaviour, compare data between patients, and plan care needs.

Although this collection of observational data is scalable; the labelling of clinical events is more challenging. Currently, as clinicians are required to manually label data based on biological samples or conversations with participants, label collection is time-consuming and costly. Hence, available labelled data is sparse, requiring the use of machine learning techniques that can learn from little or no examples.

Remote healthcare monitoring data can contain a significant amount of noise, which affects the training of predictive models. This noise can be in the form of noisy features or labels, caused by corrupted measurements, faulty devices, or human error. In remote healthcare settings, datasets are often constructed by combining multiple data sources. Although many studies in the literature study noise in data as a whole, there is a lack of work understanding how to mitigate noisy data originating from multiple sources.

Further, deep learning architectures, which have seen success in computer vision and natural language processing, provide predictions that are difficult to explain. This limits their use in healthcare as they can be seen as untrustworthy to clinicians. Therefore, techniques to improve the explainability of predictions should be used when designing predictive models for the diagnosis of adverse health conditions. Additionally, simple and robust predictive models should be preferred over complex architectures as they can be more performant when trained on small datasets and their predictions are often easier to interpret.

1.6 The Minder Study

The Minder study, an initiative from the Care Research and Technology Centre (CR&T) within the UK Dementia Research Institute (UK DRI), aims to bring together researchers working in data analysis and the internet of things with clinicians providing care for those living with dementia. The goal of the study is to use these technologies to allow people living with dementia to stay in their homes for longer and feel more independent, reducing the burden on health services.

As part of this goal, The Minder platform has been developed and designed to allow caregivers and members of the clinical monitoring team to interact with collected data and outputs from data analysis methods. This platform enables clinicians to personalise their care delivery for each participant based on recommendations from predictive models and their own analysis. This clinical support is designed to promote independent living for those living with dementia and improve quality of life.

The Minder system³ uses a broad range of sensors to collect continuous data from participants and uses machine learning methods to extract clinical insights. This study includes around 85 currently active homes, and has over time included 163 total homes that have contributed to nearly 90,000 days of data, spread over multiple years. In parallel, a clinical monitoring team and general practitioners (GPs) work to label data and provide care interventions. This scale of data collection and the longitudinal nature of the dataset provide a unique opportunity to build machine learning methods for remote healthcare monitoring. Once methods are thoroughly tested and documented, the clinical monitoring team and engagement of participants allow for a real-time test bed of new innovations, providing a pipeline for UK Conformity Assessed (UKCA) marking and incorporation into a medical device. Therefore, the Minder system offers the opportunity to work closely with both those living with dementia and clinicians providing care, to design tools and predictive models to improve the lives of participants.

The data collected within the study varies in modality, with the majority of homes of people living with dementia containing Passive Infrared (PIR) sensors, environmental sensors, and smart plugs to measure activity, and a sleep sensor to measure nocturnal physiology. This allows researchers to build various data analysis methods and care interventions based on non-invasive sensor technology, requiring little effort from the study participant. This wealth of information leads to diverse solutions to a range of issues currently facing dementia care and remote healthcare monitoring.

³<https://www.ukdri.ac.uk/centres/care-research-technology/minder>

1.7 Methodological Contributions

This doctoral research has focused on developing technical and clinically applicable solutions to address some of the key technological challenges that limit dementia care supported by remote healthcare monitoring. In particular, this research makes the following contributions:

- We have developed an interpretable predictive model for urinary tract infections based on symptom-led engineered features, designed in collaboration with clinicians.
- We propose a method to mitigate the effects of noisy data when produced by multiple data sources and evaluate our solution on various public datasets commonly used within the noisy data literature.
- We propose an approach to overcome the challenges of labelling data, namely that it can be costly or noisy, by integrating Large Language Models (LLMs) into the prior elicitation framework. This improves the reliability of predictive models when trained with scarce or noisy data and allows for the estimation of predictive uncertainty. This combines recent advances in LLMs with predictive models that are already used and trusted by clinicians.
- Extending the use of LLMs, we propose a method for time-series representation learning that allows for human-centred analysis of large amounts of unlabelled activity data. We show that this method allows for effective clustering and similarity searching that can improve the planning of care delivery.

In this thesis, we will focus on data collected using a remote healthcare monitoring system, as part of the Minder study, to motivate the methods we develop and provide a real-world application for testing. In particular, we focus on the clinically validated labels of urinary tract infections, as well as the unlabelled daily activity and sleep data. Technological innovations are also tested on public datasets that are commonly used as benchmarks in the literature, allowing for reproducibility.

The background theory, as well as the benchmark datasets, are described in Chapter 2 and are presented in more detail within the chapter where they are applied. A description of the related work, which contextualises our research within the literature and discusses open problems, is given in Chapter 3.

1.7.1 Predicting Risk of Urinary Tract Infection

As part of studying methods for the delivery of remote healthcare, we choose to focus on a predictive problem that affects many People Living with Dementia (PLWD). Initially focussing on a single predictive task allows us to gain an understanding of the problems facing remote healthcare monitoring and develop methods that can translate to other problems.

Urinary Tract Infections (UTIs) are one of the most common bacterial infections in older adults, constituting around 25% of all infections [41–45] and clinical presentation ranges from self-limited illness to severe sepsis. UTIs account for approximately 9–31% of cases of severe sepsis, which itself has an estimated mortality of 20% - 40% [44, 46–49]. Currently, to differentiate between asymptomatic bacteriuria and UTIs, clinicians rely on positive findings of bacteriuria and genitourinary symptoms. Additionally, diagnosis is complicated by the presence of cognitive impairment or dementia, as PLWD can find it challenging to report their symptoms, which may lead to further health conditions [50–52]. As a result, acute infections may not be diagnosed until symptoms require hospitalisation [53]. In the United Kingdom, more than 20% of hospital beds are occupied by PLWD, with 9% of these attributed to UTIs [11, 33, 54, 55], which constitutes a serious problem facing both dementia care and hospital bed management.

Machine learning offers an opportunity to aid in the delivery of care for PLWD. Recent advances in predictive methods show promise for the development of advanced predictive models that incorporate patient data to improve diagnostic performance. Specifically, for UTI detection in PLWD, these methods could improve diagnostic performance and timeliness, resulting in earlier detection and treatment. In parallel, reliable and low-cost monitoring devices have been developed to offer complementary solutions to the typical delivery of care [56, 57], allowing the analysis of data that is not typically available to clinicians.

In Chapter 4, we assess the use of machine learning to identify the risk of UTI events in PLWD. This is done by analysing symptom-targeted features, engineered from continuous in-home activity and physiology data collected by low-cost and passive sensors. Then, through optimisation and consultations with clinicians, we determine thresholds for the stratification of the risk scores to improve the algorithm's clinical applicability. By providing tiered alerts with different required actions, our approach is flexible to the resources of a clinical monitoring team. This work aids clinicians in the early diagnosis of UTIs, and enables a better understanding of in-home behaviour at the point of clinical decision-making.

The use of high-resolution in-home observation and measurement data in combination with machine learning methods results in timely interventions that can have a significant impact on reducing preventable and unplanned hospital admissions in dementia patients. Such a tool allows for the precise collection of urine samples for culture analysis, improved clinical outcomes, and a reduction in the burden on healthcare services. It can also lead to decreased antibiotic overuse and misuse in PLWD by reducing UTI detection time and providing practitioners with more complete pictures of their patients.

In addition to tackling a significant problem, this study allows us to identify some of the current issues that limit the use of remote healthcare monitoring in the delivery of care, which we will target in the remainder of our research.

From the unmet clinical needs discussed in Section 1.5, this work targets the use of passive sensors in remote healthcare monitoring and techniques for the interpretation of the outputs of predictive models.

The contributions, as a result of this work, have been presented in the following publications:

- Alexander Capstick, Francesca Palermo, Kimberley Zakka, Nan Fletcher-Lloyd, et al. Digital remote monitoring for screening and early detection of urinary tract infections. *npj Digital Medicine*, 7(1), Jan 2024. ISSN 2398-6352. doi: 10.1038/s41746-023-00995-5. URL <https://www.nature.com/articles/s41746-023-00995-5>
- Marirena Bafaloukou, Ann-Kathrin Schalkamp, Nan Fletcher-Lloyd, Alexander Capstick, et al. An interpretable machine learning tool for in-home monitoring of agitation episodes in people living with dementia: A proof-of-concept study. *Lancet eClinicalMedicine*, 80, 2025. URL <https://doi.org/10.1016/j.eclinm.2024.103032>
- Maria R Lima, Alexander Capstick, Fatemeh Geranmayeh, Ramin Nilforooshan, et al. Evaluating spoken language as a biomarker for automated screening of cognitive impairment. *arXiv preprint arXiv:2501.18731*, 2025. URL <https://arxiv.org/abs/2501.18731> (under review)

In Bafaloukou et al. (2025) [58], we studied the use of the same feature engineering and post-processing pipeline for agitation detection in the Minder study. Predicting agitation is important as interventions can be made to reduce the stress and confusion of people living with dementia.

In Lima et al. (2025) [59], we explored the use of machine learning in the diagnosis of dementia, by analysing people’s descriptions of a standardised image. We find that our methods allow the accurate

prediction of dementia diagnosis and, remarkably, that our predicted risk scores correlate well with scores on cognitive tests without being explicitly trained for them. This work uses methods inspired by the post-processing pipeline discussed in our work on UTI detection and follows a similar evaluation strategy.

Regulatory Compliance

As part of the encompassing Minder system, the proposed UTI risk prediction model is currently in the process of obtaining the UK Conformity Assessed (UKCA) Class-I marking and has already met the Quality Management System (QMS) ISO 9001:2015 requirements. Once the UKCA marking is complete, the UTI algorithm will be a component of the Minder system, which will become a medical device.

During the process of acquiring these certifications, we performed retrospective silent testing on our Minder cohort with the trained UTI model, as it would appear in production. We present the results of this analysis in Chapter 4 alongside the work that was completed as part of our initial published study: Capstick et al. (2024) [1].

Although not within the scope of this thesis, significant work was undertaken to design and build production-ready code in collaboration with the software team and to develop a set of protocols that could be used to trigger UTI alerts in the Minder system and act accordingly. This involved a process for the clinical monitoring team to follow, including calling a participant or collecting a biological sample, based on the type of alert generated by the UTI model.

1.7.2 Learning from Multi-Source Data

When building a predictive model for the detection of UTIs using remote healthcare monitoring data, we identified an unexplored area of research that affects the deployment of machine learning models. In the remote monitoring context, and many other settings, training data is generated by multiple data sources. However, these data sources may have differing levels of data quality or noise, which is not considered when building predictive models. Currently, data sources are merged to form a single dataset, and methods to tackle noise assume that the probability that a data point is noisy is independent of its source.

In healthcare, for example, data sources might refer to different devices, sites, or human labellers, which are often combined [60, 61]. It is conceivable that these sources may not provide the same level of data quality and could contain noisy features, incorrect labelling, or missing values, especially in the case of

human labelling. If not addressed, these data quality issues can have adverse effects on the performance and robustness of trained machine learning models [62–64].

In the literature, there are many approaches designed to address the effects of noisy labels or features. These methods fall mainly into two categories: Those that use sample selection within preprocessing to remove out-of-distribution observations from datasets [65–67], and those methods that are designed to train neural networks on noisy data. However, in the first case, a user is required to define what is meant by “out-of-distribution” for the features and labels and assume that discarded data is not helpful for training [68]. In addition, the second approach considers the noise in the data as a whole and does not use the information that can be gained from incorporating the data source of an observation into learning.

In contrast, within federated learning, techniques for mitigating the effects of noisy features and labels from different data sources have been explored [69]. However, in federated learning, approaches assume that data cannot be transferred between sources, limiting performance when data sharing is permitted.

In Chapter 5, we propose Loss Adapted Plasticity (LAP), a method for training neural networks on noisy data originating from multiple sources. To achieve this, we use the source of a data point to condition our noisy data technique. Here, we evaluate our method in minimising the error on a noise-free test set after being trained on noisy multi-source data. Compared to baseline methods from the literature, we find that our approach provides considerably improved learning when sources have different data quality and, importantly, maintains consistent performance when data is free of noise.

LAP is designed to benefit from seeing data points from all sources during initial training but to reduce learning on noisy data later in training, when the model is prone to memorising data points [62, 63, 70, 71]. This is achieved through the scheduling of a temperature parameter that controls the effects on learning from each data source, taking inspiration from the literature on Bayesian learning with tempered likelihoods.

From the unmet clinical needs discussed in Section 1.5, this work aims to study the challenges posed by noisy data in healthcare, and specifically when it is generated by multiple data sources.

As part of this work, we have released the following code repositories:

- The code to reproduce the results in this chapter is available at:
 - <https://github.com/alexcapstick/unreliable-sources>
 - <https://github.com/tmi-lab/unreliable-sources>

- The code to implement the proposed methods on new datasets is available at:

- https://github.com/alexcapstick/loss_adapted_plasticity
- https://github.com/tmi-lab/loss_adapted_plasticity

The contributions, as a result of this work, have been presented in the following publications:

- Alexander Capstick and Payam Barnaghi. Predicting targets with data from non-conforming sources. *Tiny Papers Track at International Conference on Learning Representations (ICLR)*, 2023. URL <https://dblp.org/rec/conf/iclr/CapstickB23.html>
- Kexin Fan, Alexander Capstick, Ramin Nilforooshan, and Payam Barnaghi. Urinary tract infection detection in digital remote monitoring: Strategies for managing participant-specific prediction complexity. In *2024 IEEE International Conference on e-health Networking, Application & Services (HealthCom)*, pages 1–6, 2024. URL <https://ieeexplore.ieee.org/abstract/document/10880784>
- Alexander Capstick, Francesca Palermo, Tianyu Cui, and Payam Barnaghi. Training neural networks on data sources with unknown reliability. *Information Fusion*, 2025. URL <https://doi.org/10.1016/j.inffus.2025.103239>

In the above, in Capstick and Barnaghi (2023) [72], we studied a setting in which each data source produces contradictory labels that we would like to learn, without assuming that they are adversarial or that the sources contain noise.

In the above, in Fan et al. (2024) [73], we combined some of the insights in Capstick and Barnaghi (2023) [72] and Capstick et al. (2025) [74] to evaluate methods for improving the fairness of the prediction of UTI in our Minder study. This involved clustering participants by their empirical risk based on early training of a neural network. Using the average training loss, we grouped participants by how challenging they were to learn for a predictive model and then evaluated multi-task learning strategies to improve performance in each of these groups.

1.7.3 Improving Predictive Modelling with Prior Knowledge from Large Language Models

When studying the use of machine learning methods to predict the risk of UTI in people living with dementia, we identified a challenge limiting the construction of reliable predictive models. Observational data can be continuously collected in remote healthcare monitoring using passive sensors. However, labelling this data, for example with adverse health conditions such as urinary tract infection (UTI), can be challenging, time-consuming, and costly [39, 75]. Prior information of a task over model parameters, as privileged information [76], can reduce the number of samples needed to learn performant models while also enabling more accurate estimates of predictive uncertainty. However, this expert knowledge about model parameters can be difficult to obtain due to the challenges of performing expert prior elicitation, particularly in low-resource settings [77].

Linear models are widely used in healthcare and as part of our work on UTI prediction due to their interpretability, robustness, and clinical familiarity [1, 78, 79]. However, in many studies, models are trained without the use of prior distributions, possibly due to the challenges associated with elicitation.

Due to their diverse training data and rate of development [80, 81], Large Language Models (LLMs) present an opportunity to supplement or automate the prior elicitation process. LLMs can provide insights about a predictive model's prior by approximating expertise in a given field, acquired through their training data. This motivates the design of a probabilistic framework to extract prior task knowledge from language models in the form of distributions over the parameters of a linear predictive model. Our proposed approach, AutoElicit, combines LLMs with trusted and robust linear predictive models that are already widely understood and utilised in real-world applications.

AutoElicit, detailed in Chapter 6, is an algorithm used to produce LLM-elicited priors over predictive models, which we found to provide a higher posterior and lower mean square error than an uninformative prior on various numerical predictive tasks, improving models with fewer examples. In particular, when predicting UTIs in our study on dementia care, we found that AutoElicit priors result in performant accuracy over six months earlier than with uninformative priors.

From the unmet clinical needs discussed in Section 1.5, this work aims to resolve the limitations posed by the small number of labels available for adverse clinical events. Further, the focus on linear models ensures that priors elicited from large language models are interpretable, since these distributions are directly related to the risk factor of each feature.

As part of this work, we have released the following code repositories:

- The code to reproduce the results in this chapter is available at:
 - <https://github.com/alexcapstick/llm-elicited-priors>
 - <https://github.com/tmi-lab/llm-elicited-priors>
- The code to implement the proposed methods on new datasets is available at:
 - <https://github.com/alexcapstick/autoelicit>
 - <https://github.com/tmi-lab/autoelicit>

The contributions, as a result of this work, have been presented in the following publications:

- Alexander Capstick, Rahul G Krishnan, and Payam Barnaghi. AutoElicit: Using large language models for expert prior elicitation in predictive modelling. *International Conference on Machine Learning (ICML)*, 2025. URL <https://arxiv.org/abs/2411.17284>

1.7.4 Time-Series Representation Learning

Another identified challenge in remote healthcare monitoring is time-series representation learning. Time-series data is continuously collected in our study, using passive devices. Due to this scale of data collection, machine learning techniques are necessary for the effective analysis of data to obtain insights for care delivery.

Representation learning is a machine learning method used to transform data into embeddings useful for downstream tasks or further analysis and can be performed with few or no labels [83–86]. Further, since time-series data in healthcare can be costly, challenging, or even infeasible to label [87], representation learning techniques offer the extraction of insights from unlabelled data.

Therefore, in Chapter 7, we propose an approach for learning representations of time-series data containing irregularly appearing discrete values, such as a series of locations of in-home activity or electronic healthcare records. Here, each new measurement in time is a discrete or categorical value that can be understood intuitively. By fine-tuning an LLM using a triplet loss, we encode days of activity data that have been transformed into sentences, into vectors that can be used for clustering, finding similar clinical

cases, identifying changes in data patterns, and supporting the delivery of personalised care. In the initial results, this method produced intuitive data embeddings that allowed us to visualise and analyse in-home activity data more concisely.

From the unmet clinical needs discussed in Section 1.5, this study targets the challenges associated with studying and visualising the scale of data collected by continuously recording passive sensors. Further, we make use of the unlabelled data, avoiding the issues related to obtaining labelled examples.

As part of this work, we have released the following code repositories, which assume you have access to the underlying dataset:

- <https://github.com/alexcapstick/Text-Encoders-For-Daily-Movement-Data>
- <https://github.com/tmi-lab/Text-Encoders-For-Daily-Movement-Data>

The contributions, as a result of this work, have been presented in the following publications:

- Alexander Capstick, Tianyu Cui, Yu Chen, and Payam Barnaghi. Representation learning of daily movement data using text encoders. *International Conference on Learning Representations (ICLR) Workshop on Learning from Time Series For Health*, 2024. URL <https://arxiv.org/abs/2405.04494>
- Jin Cui, Alexander Capstick, Payam Barnaghi, and Gregory Scott. Two-stage representation learning for analyzing movement behavior dynamics in people living with dementia. *AAAI Workshop on Large Language Models and Generative AI for Health*, 2025. URL <https://arxiv.org/abs/2502.09173>

Cui et al. (2025) [88] used the embedded time-series data, along with dimension reduction techniques and clustering, to build transition matrices. From these transition matrices, PageRank [89] was used to calculate stationary distributions, which are shown to be predictive of scores on cognitive assessments.

1.7.5 Publications

Overall, as part of my PhD studies, I have contributed to both machine learning methods and applications to remote healthcare monitoring. The following papers are published or under review and listed in order of their most recent publication date.

- Francesca Palermo, Honglin Li, **Alexander Capstick**, Nan Fletcher-Lloyd, *et al.* Designing a clinically applicable deep recurrent model to identify neuropsychiatric symptoms in people living with dementia using in-home monitoring data. *NeurIPS Workshop-Bridging the Gap: From Machine Learning Research to Clinical Practice*, 2021. URL <https://arxiv.org/abs/2110.09868>
- **Alexander Capstick**, Francesca Palermo, Honglin Li, Payam Barnaghi, *et al.* Clinically relevant and explainable feature extraction from remote monitoring data for the prediction of urinary tract infections. *Alzheimer's & Dementia*, Wiley, 19:e064683, 2023. URL <https://alz-journals.onlinelibrary.wiley.com/doi/full/10.1002/alz.064683>
- Francesca Palermo, Yu Chen, **Alexander Capstick**, Nan Fletcher-Loyd, *et al.* TIHM: An open dataset for remote healthcare monitoring in dementia. *Nature Scientific Data*, 10(11):606, September 2023. ISSN 2052-4463. URL <https://www.nature.com/articles/s41597-023-02519-y>
- **Alexander Capstick** and Payam Barnaghi. Predicting targets with data from non-conforming sources. *Tiny Papers Track at International Conference on Learning Representations (ICLR)*, 2023. URL <https://dblp.org/rec/conf/iclr/CapstickB23.html>
- Yushan Huang, Yuchen Zhao, **Alexander Capstick**, Francesca Palermo, *et al.* Analyzing entropy features in time-series data for pattern recognition in neurological conditions. *Artificial Intelligence in Medicine*, Elsevier, page 102821, 2024. ISSN 0933-3657. URL <https://www.sciencedirect.com/science/article/pii/S0933365724000630>
- **Alexander Capstick**, Francesca Palermo, Kimberley Zakka, Nan Fletcher-Lloyd, *et al.* Digital remote monitoring for screening and early detection of urinary tract infections. *npj Digital Medicine*, 7(1), January 2024. ISSN 2398-6352. URL <https://www.nature.com/articles/s41746-023-00995-5>
- **Alexander Capstick**, Tianyu Cui, Yu Chen, and Payam Barnaghi. Representation learning of daily movement data using text encoders. *International Conference on Learning Representations (ICLR) Workshop on Learning from Time Series For Health*, 2024. URL <https://arxiv.org/abs/2405.04494>
- Kexin Fan, **Alexander Capstick**, Ramin Nilforooshan, and Payam Barnaghi. Urinary tract infection detection in digital remote monitoring: Strategies for managing participant-specific prediction complexity. *2024 IEEE International Conference on e-health Networking, Application & Services*

- (*HealthCom*), pages 1–6. IEEE, 2024. URL <https://ieeexplore.ieee.org/abstract/document/10880784>
- Louise Rigny, Nan Fletcher-Lloyd, **Alexander Capstick**, Ramin Nilforooshan, *et al.* Assessment of sleep patterns in dementia and general population cohorts using passive in-home monitoring technologies. *Nature Communications Medicine*, 4(1):222, 2024. URL <https://www.nature.com/articles/s43856-024-00646-0>
 - Chloe Walsh, **Alexander Capstick**, Nan Fletcher-Lloyd, Jessica True, *et al.* Longitudinal study of care needs and behavioural changes in people living with dementia using in-home assessment data. *Nature Communications Medicine*, 5(1):14, 2025. URL <https://www.nature.com/articles/s43856-024-00724-3>
 - Marirena Bafaloukou, Ann-Kathrin Schalkamp, Nan Fletcher-Lloyd, **Alexander Capstick**, *et al.* An interpretable machine learning tool for in-home monitoring of agitation episodes in people living with dementia: a proof-of-concept study. *Lancet eClinicalMedicine*, 80, 2025. URL <https://doi.org/10.1016/j.eclinm.2024.103032>
 - Maria R Lima, **Alexander Capstick**, Fatemeh Geranmayeh, Ramin Nilforooshan, *et al.* Evaluating spoken language as a biomarker for automated screening of cognitive impairment, 2025. URL <https://arxiv.org/abs/2501.18731> (under review)
 - Jin Cui, **Alexander Capstick**, Payam Barnaghi, and Gregory Scott. Two-stage representation learning for analyzing movement behavior dynamics in people living with dementia. *AAAI Workshop on Large Language Models and Generative AI for Health*, 2025. URL <https://arxiv.org/abs/2502.09173>
 - **Alexander Capstick**, Francesca Palermo, Tianyu Cui, and Payam Barnaghi. Training neural networks on data sources with unknown reliability. *Information Fusion*, 2025. URL <https://doi.org/10.1016/j.inffus.2025.103239>
 - **Alexander Capstick**, Rahul G Krishnan, and Payam Barnaghi. Autoelicit: Using large language models for expert prior elicitation in predictive modelling. *International Conference on Machine Learning (ICML)* 2025. URL <https://arxiv.org/abs/2411.17284>

1.8 Thesis Outline

This thesis consists of eight main chapters based on four main bodies of work that are either published or under review. These works explore the use of machine learning for remote healthcare monitoring with a focus on dementia care.

1. **Introduction:** The introduction describes the aims of this thesis and motivates the research conducted.
2. **Background and Theory:** In this section, we describe the methods required to understand our body of work.
3. **Related Work:** This section includes a description of the related work that is important to contextualise our research. This section also covers some of the data studied in this thesis.
4. **Risk Prediction of Urinary Tract Infections:** This chapter contains research conducted on the use of activity and physiological data to predict the risk of UTI in our study cohort.
5. **Mitigating the Effects of Data Sources with Unknown Reliability:** This chapter presents our work on training predictive models on multi-source data containing noise.
6. **Using Large Language Models for Expert Prior Elicitation in Predictive Modelling:** In this chapter, we present our research on studying the use of LLMs for prior elicitation of linear models. This includes a framework for supplementing current prior elicitation methods and a comparison to in-context learning.
7. **Time-series Representation Learning Using Fine-Tuned Language Models:** Here, we describe a method to learn representations of discrete time-series data using language models that are helpful for vector searching, clustering, and analysis of activity patterns.
8. **Conclusions and Future Work:** This chapter contains a final discussion of the work presented in this thesis, along with a description of additional work and future research directions.

Chapter 2

Background and Theory

2.1 The Minder Study

This thesis focusses on technical solutions that contribute to addressing some of the unmet clinical and care needs in remote healthcare monitoring for dementia. For this, we have analysed data collected as part of the ongoing Minder study. This Minder dataset, collected as part of a study organised by the UK Dementia Research Institute Centre for Research and Technology Centre¹, contains measurements of in-home activity and physiology from people living with dementia in the UK.

The Minder study is unique in the type of data recorded as well as in the length of time data has been collected. In this dataset, our most commonly labelled adverse health condition is Urinary Tract Infection (UTI).

An earlier version of this study, Technology Integrated Health Management (TIHM), led to a publicly released dataset described in Palermo et al. (2023) [90].

2.1.1 Minder Study Design and Population

The Minder study is conducted in collaboration with Imperial College London and Surrey and Borders Partnership NHS Trust. Participants were recruited from the following: (1) Health and social care partners within the primary care network and community NHS trusts; (2) urgent and acute care services within the NHS; (3) social services who oversee sheltered and extra care sheltered housing schemes;

¹<https://ukdri.ac.uk/centres/care-research-technology>

(4) NHS Community Mental Health Teams for Older People (CMHT-OP); and (5) specialist memory services at Surrey and Borders Partnership NHS Foundation Trust. All participants provided written informed consent. Capacity to consent was assessed according to Good Clinical Practice, as detailed in the Research Governance Framework for Health and Social Care (Department of Health 2005) and the Mental Capacity Act 2005. Participants were provided with a Participant Information Sheet (PIS) that includes information on how the study used their personal data collected in accordance with the General Data Protection Regulation (GDPR) requirements. If the participant was deemed to lack capacity, a personal or professional consultee was sought to provide written consent to the study. Additionally, capacity of both the participant and study partner is assessed at each research visit. Research staff conducting the assessment have completed the National Institute for Health and Care Research (NIHR) Good Clinical Practice (GCP) training and Valid Informed Consent training. If a participant is deemed to lack capacity but is willing to take part in the research, a personal consultee is sought in the first instance to sign a declaration of consent. If no personal consultee can be found, a professional consultee, such as a key worker, is sought. This process is included in the study protocol, and ethical panel approval is obtained.

Eligible study participants included adults older than 50 years with a clinically ascertained diagnosis of dementia or mild cognitive impairment and current or previous treatment at a psychiatric unit. Participants lacking capacity for informed consent were required to have a partner or caregiver who had known them for at least 6 months and was able to attend research assessments with them. Exclusion criteria were as follows: (1) Patients receiving treatment for terminal illness; (2) the presence of severe mental health conditions including depression, anxiety, psychosis, and agitation; or (3) the presence of active suicidal thoughts.

Demographic data was collected during the baseline assessment, whilst psychometric scales were used to collect various physical and cognitive data during regular visits. In-home observation and measurement data was obtained using low-cost, off-the-shelf monitoring technologies, including Passive Infrared (PIR) sensors (for measuring activity) and sleep monitoring devices. The PIR sensors can detect motion within 9 metres and with a maximum angle of 45° , whilst the sleep mat device can monitor breathing rates, heart rates, and sleep states.

As part of the data collection, a subset of participants regularly provided urine samples, which were analysed for suspicion of urinary tract infection by clinicians.

An example of the layout of a home with installed sensors is given in Figure 2.1, which illustrates the

breadth of information we collect. This figure shows an example of the location of the PIR sensors and the bed mat in the home of someone living with dementia. These sensors allow us to collect behavioural information about the occupants of the home, as well as physiological and sleep data from the participant. Using these sensors, we have collected millions of data points over multiple years.

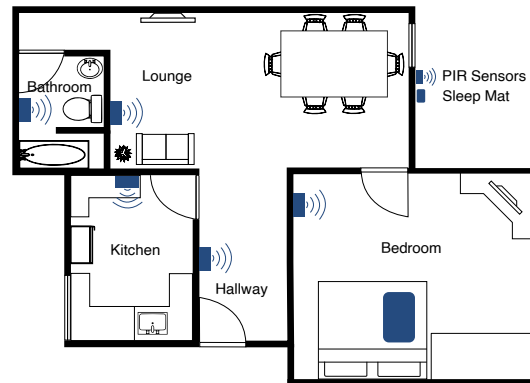


Figure 2.1: Illustration of activity and sleep sensor deployment within the home of a person living with dementia, from Capstick et al. (2024) [1]. Here, passive infrared sensors are located in each room for in-home activity monitoring. A sensor is located under the mattress for analysing physiology and movements in and out of the bed.

2.1.2 Dataset Characteristics

Focussing on data collected between 2021/06/28 and 2025/01/01, the Minder dataset contains 89,572 days of data from 163 participants. Figure 2.2 shows a histogram of the number of days of data recorded in each home, demonstrating the varied time participants have been contributing to the minder study. The average number of days recorded per participant is 581 (with a minimum of 2, a median of 560, and a maximum of 1282).

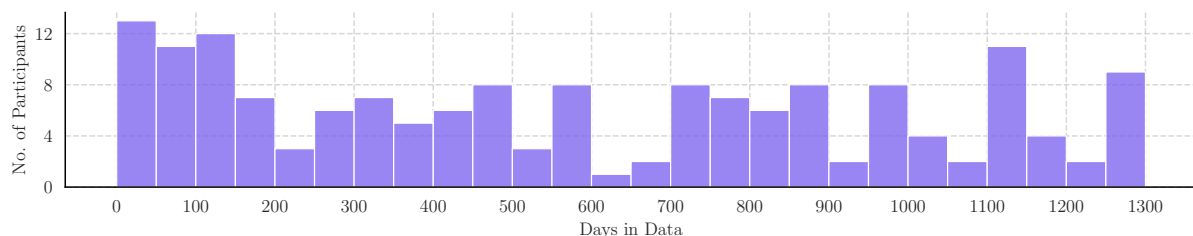


Figure 2.2: Number of days recorded, updated from Capstick et al. (2024) [2]. Histogram showing the number of participants with a given number of days of data recorded.

In Figure 2.3, we show general information about the number of participants in our study, as well as the length of time they contribute, and the number of verified urinary tract infection labels we collect. In total, the Minder dataset contains 700 negative days of UTI and 261 positive days of UTI, generated by

136 participants. Interestingly, the number of Positive and Negative days of UTI in our cohort rises at an approximately constant rate, especially once the number of active participants stabilises at approximately 80 – 85. This demonstrates the importance of building automated tools for their risk prediction.

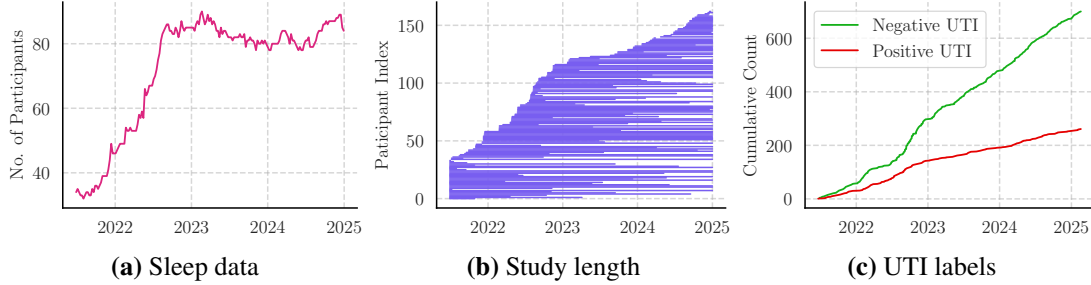


Figure 2.3: Dataset characteristics from 2021/07/01 to 2025/01/01. (a) shows the number of participants who had activity or sleep data collected from them each week; (b) illustrates the length of time each participant produced activity or sleep data within the study. Here, the line joins the first and last days a given participant had data recorded for them; (c) shows how many days of UTI were labelled in this time.

For the analysis in this thesis, we focus on information from the activity and sleep sensors. Each data point contains the participant ID, the time and date of a record (to the nearest second), and the location at which activity was detected or a sleep mat measurement.

Figure 7.2 shows the time of day participants trigger the activity sensors and two of the bed mat measurements, amounting to 34,545,025 data points. This scale of data, which is mostly unlabelled, necessitates the use of automated techniques for representation learning, clustering, and similarity searching.

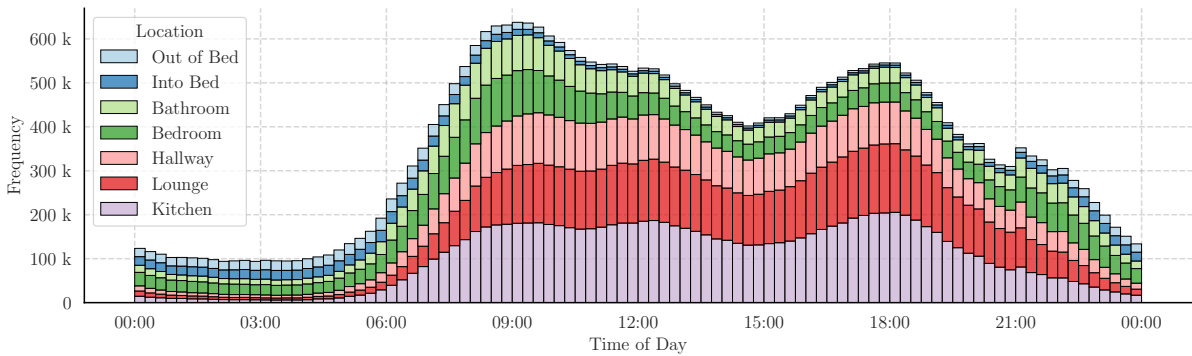


Figure 2.4: Time of location recordings, updated from Capstick et al. (2024) [2]. Histogram showing the number of sensor recordings by time of day, with locations that are ordered by their total frequency.

In addition, the sleep mat collects physiological information whilst the participant is in their bed. This includes sleep staging and measurements of heart rate and respiratory rate at minute intervals. Figure 2.5 shows the sleep stages for our cohort, representing 40,472,558 collected data points.

Further, Figures 2.6a and 2.6b show histograms of heart and respiratory rate in our cohort, respectively.

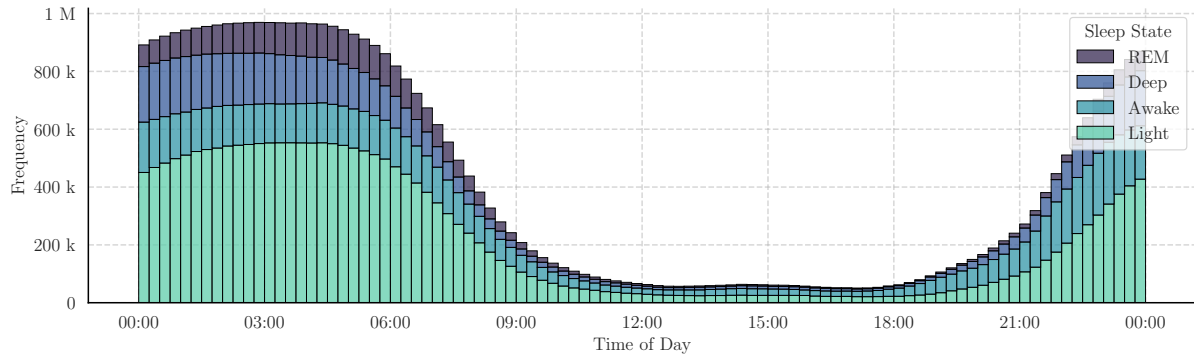


Figure 2.5: Sleep staging by time of day. This histogram shows the different sleep stages measured by the sleep mat by the time of day.

This information is unique to the participant, allowing us to gain important information about their nocturnal physiology. We hypothesise that these biomarkers are helpful in predicting infections and other adverse health conditions.

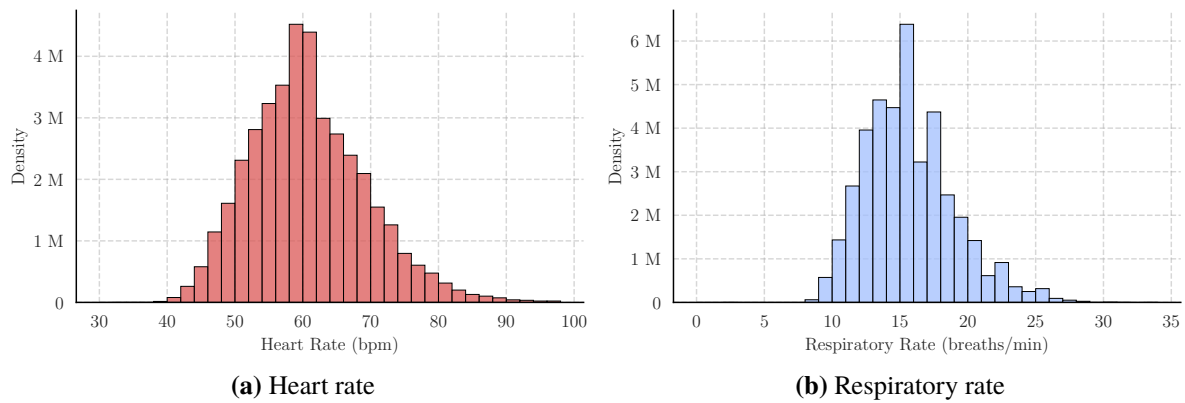


Figure 2.6: Heart rate and respiratory rate between 2021/07/01 to 2025/01/01. (a) shows a histogram of the heart rate measurements, whilst (b) shows the respiratory rate.

Since the Minder study is ongoing and research projects presented in this thesis were conducted sequentially, each chapter contains a subset of this data. In each chapter, we have specified the selection of this dataset used.

2.1.3 Demographics

Table 2.1 shows the characteristics of the cohort that contributes data to the Minder study. This study contains a balance of both male and female participants; however, we have many under-represented ethnic groups.

Table 2.1: The Minder study cohort demographics. For each characteristic, we show the number of participants and the corresponding percentage of the cohort. Some participants have requested not to share certain information outside the study and correspond to some of the N/A groups.

Sex			Primary Diagnosis			Birth Year			Ethnicity		
	No.	%		No.	%		No.	%		No.	%
Female	83	51	Alzheimer's	78	48	1920-1930	14	9	White	138	85
Male	79	48	Vascular Dementia	9	6	1930-1940	61	37	Asian	11	7
N/A	1	1	Frontotemporal Dementia	6	4	1940-1950	66	40	Black/African/Caribbean	3	2
			Parkinson's	5	3	1950-1960	18	11	Multiple Groups	1	1
			Other and Mixed	62	38	1960-1970	2	1	Other	1	1
			N/A	3	2	1970-1980	1	1	N/A	9	6
						N/A	1	1			

2.1.4 Ethics Approval

The Minder study received ethical approval from the London-Surrey Borders Research Ethics Committee; TIHM 1.5 REC: 19/LO/0102. The study is registered with NIHR in the United Kingdom under the Integrated Research Application System (IRAS) registration number 257561.

2.2 Additionally Studied Datasets

Within this thesis, we have also evaluated our methods on many public datasets to provide reproducible comparisons with baselines in the machine learning literature. These datasets are described in the following, with more details given in the relevant chapters:

- The **Breast Cancer** dataset [91] contains 569 samples of 30 features, calculated from images of breast masses that are labelled as either malignant or benign.
- **CIFAR-10** and **CIFAR-100** [92] are datasets containing 60,000 $32\text{px} \times 32\text{px}$ RGB images divided into 10 and 100 classes, with 6000 and 600 images in each class, respectively. The task is to predict the correct class of a given image.
- **CIFAR-10N** [93] contains all of the CIFAR-10 examples, but with human labels, categorised into different levels of quality.
- The **Diabetes** dataset [94], after preprocessing, contains physiological measurements and demographic information from 442 people with diabetes. Here, the task is to predict a quantitative measure of their disease progression one year after this baseline.
- **F-MNIST** [95] contains 70,000 greyscale images of items of clothing with size $28\text{px} \times 28\text{px}$,

divided into 10 classes with 7000 images in each class. The task of this dataset is to predict the correct clothing type for a given image.

- **GoEmotions** [96] is a dataset containing 171,820 natural language samples and their conveyed emotion from a possible 28 choices. These samples are annotated by 82 human raters.
- The **Heart Disease** dataset [97, 98] contains 10 physiological features and targets with a positive or negative diagnosis of heart disease for 299 people.
- The **Hypothyroid** dataset [99, 100], processed from the original, consists of 150 examples of patients' thyroid-related blood levels and whether they have a hypothyroidism diagnosis or not.
- **ImageNet** [101] contains 1,281,167 RGB images scaled to size $64\text{px} \times 64\text{px}$ from 1000 classes.
- **IMDB** [102] is a natural language-based sentiment analysis task, where the goal is to correctly classify a movie review as positive or negative based on its text. It contains 50,000 reviews divided equally between positive and negative sentiment.
- **MNIST** [103] contains 70,000 images of handwritten digits at a resolution of $28\text{px} \times 28\text{px}$ where the goal is to correctly identify the digit drawn (from 0 to 9).
- **PTB-XL** [104] consists of 21,837 ECG recordings from 18,885 patients, each of length 10 seconds and sampled at 100Hz. This dataset is labelled by 12 nurses, with each data point having a target of normal or abnormal cardiac rhythm.
- **Tiny-ImageNet** [101] contains 110,000 RGB images of size $64\text{px} \times 64\text{px}$ from 200 classes and is based on ImageNet.

2.3 Relevant Methods in Machine Learning

Machine learning is a field of research that studies algorithms to learn functions that estimate relationships between task inputs and outputs using collected data. Depending on the task, these functions can be learnt from labels, as in supervised learning or without labels, as in unsupervised learning.

Because of this flexibility, machine learning methods have been designed for a variety of uses and cover a broad range of innovations. Their use in real-world applications has recently increased rapidly, driven by developments in computing and specialised hardware, as well as theoretical developments in model architectures.

In this section, we will describe some of the methods required to understand the machine learning contributions made in this thesis.

2.3.1 Learning from Labelled Data

Supervised learning is the most common form of machine learning and is focused on learning a function f that can transform features $x \in \mathcal{X}$ to targets $y \in \mathcal{Y}$ based on collected data. Mathematically, if our function is parametrised by θ , then we wish to learn a function $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$ such that on new data, f_θ achieves some notion of maximum performance.

In classification problems, $\mathcal{Y}$ represents the set of possible ways of labelling samples of features from $\mathcal{X}$. For example, if we were to learn a model to classify UTIs based on in-home activity data, then $\mathcal{X}$ would represent the in-home device recordings, and $\mathcal{Y}$ would represent a positive or negative diagnosis of UTI. In this case, f_θ corresponds to a model that transforms feature values in $\mathcal{X}$ to a probability of each label $y \in \mathcal{Y}$ or the label itself.

Regression problems are similar in that the goal is to learn a parametrisation θ for a function f such that we minimise the error between the predicted and true targets. However, in this case, the targets correspond to values on the continuous real line. Given features $x \in \mathcal{X}$, we wish to learn a parametrisation of a function f_θ that models the relationship between the features and some targets $y \in \mathcal{Y} \subset \mathbb{R}$. For example, if we were trying to predict scores on a cognitive assessment based on features related to a patient's speech, we would wish to learn a function that can transform speech markers into a value corresponding to a possible cognitive score.

Supervised learning can be viewed through the lens of statistical decision theory, by considering that a model $p(y|\mathbf{x}, \theta)$ encodes our predictive belief about a target y given features $\mathbf{x}$. To act on this belief, such as by predicting a class or real-valued outcome, we choose to take an optimal action a^* from a set of actions $\mathcal{A}$ that minimises the expected loss, given a loss function $\mathcal{L}$:

$$a^*(\mathbf{x}) = \arg \min_{a \in \mathcal{A}} \mathbb{E}_{y \sim p(y|\mathbf{x}, \theta)} [\mathcal{L}(y, a)] \quad (2.1)$$

This does not assume that the loss on each (y, a) pair is constant. In fact, there could be different penalties associated with actions a given different true targets y . This is critical in areas such as healthcare, where incorrect predictions, such as false positives and false negatives, can cost differently.

Equation 2.1 tells us which action to take for a new observation of features $\mathbf{x}$, given knowledge of the distribution $p(y|\mathbf{x}, \theta)$. However, in machine learning, this distribution is often unknown and requires an approximation given an observed finite sample of previously observed data points.

To approximate the θ that provides this distribution, given a dataset of N features and target pairs $\mathcal{D} = \{x_n, y_n\}_{n=1}^N$, we can use maximum likelihood estimation. Here, we maximise the likelihood of generating a dataset $\mathcal{D}$ based on the parametrisation θ :

$$\hat{\theta} = \arg \max_{\theta} p(\mathcal{D}|\theta)$$

The empirical risk provides us with a method to calculate this $\hat{\theta}$ when we choose the loss function to be the Negative Log-Likelihood (NLL) of the probabilistic model we are using. To see this, consider maximum likelihood estimation:

$$\begin{aligned} \arg \max_{\theta} p(\mathcal{D}|\theta) &= \arg \max_{\theta} \prod_{i=1}^N p(y_i|x_i, \theta) \\ &= \arg \min_{\theta} \left[-\frac{1}{N} \sum_{i=1}^N \log p(y_i|x_i, \theta) \right] = \arg \min_{\theta} \text{NLL}(\mathcal{D}|\theta) \end{aligned}$$

Maximum likelihood estimation (red arrow from $\arg \max_{\theta} p(\mathcal{D}|\theta)$ to $\prod_{i=1}^N p(y_i|x_i, \theta)$)

Negative log-likelihood (purple arrow from $\text{NLL}(\mathcal{D}|\theta)$ to the expression)

log $ab = \log a + \log b$ (blue arrow from $\log p(y_i|x_i, \theta)$ to the sum)

Now consider the empirical risk $R_f(\mathcal{D}, \theta)$ using a measure of error $\mathcal{L}$ and given a predictive function f parametrised by θ :

$$\hat{\theta} = \arg \min_{\theta} R_f(\mathcal{D}, \theta) = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^N \mathcal{L}(y_i, f(x_i, \theta)) \quad (2.2)$$

If we choose $\mathcal{L}(y_i, f(x_i, \theta)) = -\log p(y_i|x_i, \theta)$, empirical risk minimisation is exactly maximum likelihood estimation:

$$\begin{aligned} \arg \min_{\theta} R_f(\mathcal{D}, \theta) &= \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^N \mathcal{L}(y_i, f(x_i, \theta)) \\ &= \arg \min_{\theta} \left[-\frac{1}{N} \sum_{i=1}^N \log p(y_i|x_i, \theta) \right] = \arg \max_{\theta} p(\mathcal{D}|\theta) \end{aligned}$$

Empirical risk minimisation (red arrow from $\arg \min_{\theta} R_f(\mathcal{D}, \theta)$ to the expression)

Maximum likelihood estimation (blue arrow from $\arg \max_{\theta} p(\mathcal{D}|\theta)$ to the expression)

For classification tasks, we usually require $f(x, \theta)$ to output the probability of the possible classes. As we will see in Section 2.3.3, for classification, this means $\mathcal{L}$ corresponds to the cross-entropy. When y

represents a class label distribution over M classes, this is defined as the following:

$$\text{cross_entropy}(y, f(x, \theta)) = - \sum_{i=1}^M y_i \log f(x, \theta)_i \quad (2.3)$$

Where y_i and $f(x|\theta)_i$ correspond to the true probability and predictive probability of class i , respectively. The cross-entropy measures how different the true probability distribution y is from the predicted probability distribution $f(x|\theta)$. Over a dataset or sample, the mean of this value is usually calculated.

For regression tasks, when assuming residuals are normally distributed, as will be discussed in Section 2.3.3, the squared error is used for our loss function $\mathcal{L}$:

$$\text{squared_error}(y, f(x, \theta)) = (y - f(x, \theta))^2 \quad (2.4)$$

The mean of this value over a dataset or sample is usually calculated to form the mean squared error.

2.3.2 Learning from Unlabelled Data

However, in many instances, we might wish to learn machine learning models without the use of labelled data. This is where unsupervised learning can be useful. In this case, the optimisation problem for calculating $\hat{\theta}$ is less defined since it depends on the task.

Unsupervised learning can be used for clustering or self-supervised learning where there are no true labels. Instead, given $x \in \mathcal{X}$, we wish to train a machine learning model to approximate the probability distribution $p(x)$.

In this case, we are modelling the feature space only. This can be particularly useful in healthcare tasks, where collecting labels can be expensive or infeasible.

Learning from Pseudo-Labels

As part of unsupervised learning, self-supervised learning makes use of theory from supervised learning by generating pseudo-labels from the features $x \in \mathcal{X}$.

This has become more popular recently and has been employed to train models on big datasets, where labelling all data is impractical, such as in language or generative modelling. In this approach, the idea is

to create pseudo-labels from unlabelled data so that supervised methods can be employed. For example, given image data, we could remove patches of the image and train a model to fill in the gaps. Similarly, with text data, we could mask words at random places in a sentence and train a model to predict the missing words. The aim is to train a model to predict $\tilde{x} = f(x|\theta)$ where x represents the observed input and $\tilde{x}$ is the label that is created from x . Here, the hope is that model f can learn information from this problem that is useful for downstream tasks.

These methods can form the initial training for a model used for transfer learning or fine-tuning, where a trained model is trained further on new data from a different task or similar data from a specialised task.

2.3.3 Machine Learning Model Types

In this thesis, we apply machine learning methods to learn patterns from data. We will primarily focus on linear models for their robustness and interpretability, and neural networks for their large capacity and ability to learn complex non-linear relationships. These models represent both limits of the spectrum of model capacity and are essential building blocks for other architectures. Additionally, depending on the task, we have evaluated other suitable machine learning methods. This section provides a description of these models and how to train them, based on the explanations given in Murphy (2022) [105].

In most cases, for a model f parametrised by θ the optimisation problem used to train on $\mathcal{D}$ involves maximum likelihood estimation:

$$\hat{\theta} = \arg \max_{\theta} p(\mathcal{D}|\theta) \quad (2.5)$$

Here, the goal is to find the parameters $\hat{\theta}$ that best describe the dataset $\mathcal{D}$. However, depending on the class of models that are being trained, the method to find $\hat{\theta}$ varies considerably.

Linear Models

Linear models are the simplest type of machine learning models. As the name suggests, they represent a class of models that make linear predictions.

First, we will look at binary logistic regression. This is a linear model used for classification, where outcomes can either be positively ($y = 1$) or negatively ($y = 0$) labelled. Given features $\mathbf{x} = (x_1, x_2, \dots, x_M)$

and a linear model f parametrised by $\theta = (\mathbf{w}, b)$ with $\mathbf{w}$ as weights and b as the bias, a prediction y is:

$$p(y = 1|\mathbf{x}, \theta) = \text{Ber}(y|\sigma(f(\mathbf{x}|\theta))) = \text{Ber}(y|\sigma(\mathbf{w}^T \mathbf{x} + b))$$

Where σ is the sigmoid function, transforming values in $(-\infty, \infty) \rightarrow (0, 1)$ and $\text{Ber}(y|q)$ represents the probability of observing y under the Bernoulli distribution, given the probability $q = p(y = 1)$.

Essentially, the logistic regression model uses a linear combination of features to calculate a logit, which is converted into the positive class probability using the sigmoid function. The prediction function $f(\mathbf{x}|\theta)$, which calculates the logit, can also be explicitly written as follows:

$$f(\mathbf{x}|\theta) = \mathbf{w}^T \mathbf{x} + b = b + \sum_{m=1}^M w_m x_m$$

This function defines a hyperplane with normal vector $\mathbf{w}$, consistent with its description as a linear model.

To train this model over a dataset $\mathcal{D}$ of N pairs of data points $(\mathbf{x}_n, y_n)$, we use the likelihood. By defining $a_n = f(\mathbf{x}_n|\theta)$, we have the optimisation problem:

$$\hat{\theta} = \arg \max_{\theta} p(\mathcal{D}|\theta) = \arg \max_{\theta} \prod_{n=1}^N \text{Ber}(y_n|\sigma(a_n))$$

Since maximising the likelihood is equivalent to minimising the Negative Log-Likelihood (NLL), we instead wish to calculate: $\arg \min_{\theta} \text{NLL}(\mathcal{D}|\theta)$. This NLL is given by the following:

$$\text{NLL}(\mathcal{D}|\theta) = -\log p(\mathcal{D}|\theta) = -\log \prod_{n=1}^N \text{Ber}(y_n|\sigma(a_n)) = -\sum_{n=1}^N \log \text{Ber}(y_n|\sigma(a_n))$$

↑ Since $\log ab = \log a + \log b$ ↑

By using the equation to calculate $\text{Ber}(y|q) = q^y \times (1 - q)^{1-y}$, this can be written as:

$$\text{NLL}(\mathcal{D}|\theta) = -\sum_{n=1}^N \log \text{Ber}(y_n|\sigma(a_n)) = -\sum_{n=1}^N [y_n \log \sigma(a_n) + (1 - y_n) \log(1 - \sigma(a_n))]$$

↑ The cross-entropy from Equation 2.3 naturally appears ↑

This shows that to find $\hat{\theta}$, we must minimise $\text{NLL}(\mathcal{D}|\theta)$, or equivalently, the cross-entropy. We can achieve this by using gradient-based optimisation methods. For logistic regression, this is commonly done using the limited-memory BFGS algorithm [106].

To extend this analysis to multi-class classification, the categorical distribution should be used in place of the Bernoulli distribution.

Linear *regression* models are similar in that they use linear combinations of features to make predictions, however, instead of providing class predictions, they return predictions in the continuous space. This makes them useful for the prediction of quantities. The definition of a linear regressor, therefore, uses a continuous probability distribution. Given features $\mathbf{x} = (x_1, x_2, \dots, x_M)$ and a linear model f parametrised by $\boldsymbol{\theta} = (\mathbf{w}, b, \sigma)$ with $\mathbf{w}$ as weights and b as the bias, a prediction y is:

$$p(y|\mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(y|\mathbf{w}^T \mathbf{x} + b, \sigma^2)$$

Where $\mathcal{N}(y|\mu, \sigma^2)$ defines a normal distribution over y given mean μ and standard deviation σ . Again, this function defines a hyperplane with normal vector $\mathbf{w}$.

To train this model using maximum likelihood estimation, we are required to calculate the likelihood analytically as before. Given a $\mathcal{D}$ of N pairs of data points $(\mathbf{x}_n, y_n)$ and defining $\tilde{y}_n = \mathbf{w}^T \mathbf{x}_n + b$ we write the likelihood:

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^N \mathcal{N}(y_n|\mathbf{w}^T \mathbf{x}_n + b, \sigma^2)$$

As before, we will use the NLL as our objective function. Using the probability density function of a normal distribution:

$$\mathcal{N}(y|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2}$$

We can write the NLL $\text{NLL}(\mathcal{D}|\boldsymbol{\theta})$ as:

$$\text{NLL}(\mathcal{D}|\boldsymbol{\theta}) = - \sum_{n=1}^N \log \mathcal{N}(y_n|\mathbf{w}^T \mathbf{x}_n + b, \sigma^2) = \frac{1}{2\sigma} \sum_{n=1}^N (y_n - \tilde{y}_n)^2 + \frac{N}{2} \log 2\pi\sigma^2$$



The squared error from Equation 2.4 naturally appears

For this case, when finding the optimal parameters $\hat{\boldsymbol{\theta}}$, we can calculate them analytically. This can be done by solving: $\nabla \text{NLL}(\mathcal{D}|\boldsymbol{\theta}) = 0$.

Linear models, as with all machine learning methods, make assumptions about the data that must be considered when using them for modelling.

For linear regression, in this thesis we make five main assumptions about the data: (1) Homoscedasticity of residuals: We assume that the residuals, $y_n - \tilde{y}_n$, have constant variance (σ^2 in the above) for all features $\mathbf{x}$. This does not necessarily have to be the case however, since weighted least squares can offer a solution to heteroscedastic regression [105]; (2) Independence: The residuals are assumed to be independent; (3) Linearity: In this thesis, the relationship between features and targets is assumed to be

linear. However, feature transformations could allow for non-linear relationships between features and targets to be learnt; (4) No multicollinearity: In this thesis, we have assumed that features are not highly correlated; (5) Normality of residuals: We will assume that residuals are normally distributed, however, other distributions could be used, corresponding to alternative loss functions.

For logistic regression, in this thesis we make four main assumptions: (1) Binary outcome: We assume that the targets are binary, in that each observation of features belongs to one of two classes; (2) Independence: All observed data points are assumed to be independent; (3) Linearity over the logit: The relationship between features and the logits of the target is assumed to be linear; (4) No multicollinearity: Features are assumed not to be highly correlated.

In practice, many of these assumptions are hard to satisfy, due to the realities of real-world data, and thought must be given to their effects on the modelling task.

Neural Networks

Linear models are robust and interpretable but are limited by the assumption that features and outputs are linearly related. In contrast, neural networks allow for the construction of highly non-linear machine learning models using far greater numbers of parameters. These can be constructed using a broad range of possible different layer architectures, with the general idea of stacking many layers sequentially.

To motivate the conception of neural networks, we start by modifying the linear architecture presented above, proposing to replace $\mathbf{x}$ with a non-linear transformation $\phi(\mathbf{x}, \boldsymbol{\theta}_2)$ [105]. Here, ϕ is a function parametrised by $\boldsymbol{\theta}_2$ that transforms $\mathbf{x}$ into another vector space. Our predictive model, f is parametrised by $\boldsymbol{\theta} = \boldsymbol{\theta}_1 \cup \boldsymbol{\theta}_2$, where $\boldsymbol{\theta}_1 = (\mathbf{w}, b)$.

$$f(\mathbf{x}|\boldsymbol{\theta}) = \mathbf{w}^T \phi(\mathbf{x}, \boldsymbol{\theta}_2) + b$$

The function $\phi(\mathbf{x}, \boldsymbol{\theta}_2)$, must be non-linear; otherwise f simply collapses back to $\mathbf{w}^T \mathbf{x} + b$, with a modified $\mathbf{w}$ and b . This non-linearity is often introduced by using an activation function, of which there are a variety to choose from. Common examples include ReLU, tanh, or the sigmoid function. This f is linearly related to $\phi(\mathbf{x}, \boldsymbol{\theta}_2)$, but a further non-linear function could be used on the output of f , for example, to produce outputs that sum to 1.

By further defining $\phi(\mathbf{x}, \boldsymbol{\theta}_2)$ as a composition of other non-linear functions, f can be made increasingly

more complex and non-linear. By specifying $\phi_{\theta_i}(\mathbf{x}) = \phi(\mathbf{x}, \theta_i)$, we can compose L layers in the following way:

$$f(\mathbf{x}|\boldsymbol{\theta}) = \mathbf{w}^T [\phi_{\theta_L} \circ \dots \circ \phi_{\theta_1}](\mathbf{x}) + b$$

The diagram illustrates the components of the equation $f(\mathbf{x}|\boldsymbol{\theta}) = \mathbf{w}^T [\phi_{\theta_L} \circ \dots \circ \phi_{\theta_1}](\mathbf{x}) + b$. A red arrow labeled "The final predictive layer" points to the $\mathbf{w}^T$ and $+b$ terms. A teal arrow labeled "The neural network layers transform features" points to the composition of functions $\phi_{\theta_L} \circ \dots \circ \phi_{\theta_1}$. A purple arrow labeled "The input features" points to the $(\mathbf{x})$ term.

Here $\boldsymbol{\theta} = (\mathbf{w}, b) \cup \theta_1 \cup \dots \cup \theta_L$ contains the parametrisation for all layers. We can then use $f(\mathbf{x}|\boldsymbol{\theta})$ in place of the linear models defined in Section 2.3.3 to define the maximum likelihood estimate:

For binary classification:

For regression:

$$\hat{\boldsymbol{\theta}} = \arg \max_{\boldsymbol{\theta}} \prod_{n=1}^N \text{Ber}(y_n | \sigma(f(\mathbf{x}_n | \boldsymbol{\theta}))) \quad \hat{\boldsymbol{\theta}} = \arg \max_{\boldsymbol{\theta}} \prod_{n=1}^N \mathcal{N}(y_n | f(\mathbf{x}_n | \boldsymbol{\theta}), \sigma^2)$$

These can be optimised by using gradient-based methods on the NLL. Typically, for neural networks, these include the stochastic gradient descent or Adaptive Moment Estimation (Adam) [107] algorithms.

Many types of functions have been proposed for neural network layers $\phi_{\theta_i}(\mathbf{x})$, with some notable methods including convolutions, pooling, Long and Short-Term Memory (LSTM) modules, and attention mechanisms.

One of the major benefits of using neural networks for modelling is that they make few assumptions about the mapping function between the data inputs and targets. Neural networks assume that the residuals are independently sampled from the same underlying distribution, allowing for the likelihood to be factorised. However, due to the considerable increase in the number of model parameters, they require significantly more data to train than linear models.

Specific architectures have been developed for different data modalities, each with its own set of assumptions about the data. For example, auto-regressive models assume that each data point contains a sequence of features, whilst 2-dimensional convolutional neural networks assume that each data point contains data with a 2-dimensional structure.

Other Machine Learning Models Evaluated in This Study

Within this thesis, we will also evaluate the following machine learning models:

Gaussian Naive Bayes The naive Bayes classifier is a probabilistic method based on the assumption that feature values are conditionally independent given a class label [105]. The Gaussian Naive Bayes model

is used for continuous-valued features and assumes that given a class label, values for each feature are Gaussian distributed.

Random Forest Random forests use an ensemble of decision trees for classification or regression tasks [108]. This model architecture uses bagging, where a bootstrap sample of data points is chosen at the creation of each new decision tree and is used for training. This is performed many times to produce many trained decision trees, aiming to reduce variance compared to using a single decision tree.

Extreme Gradient Boosting Extreme Gradient Boosting (XGBoost) [109] additionally uses bagging and ensembles of decision trees (by default) in its architecture for classification and regression tasks, but its training method differs. Boosting aims to build trees iteratively, such that each new tree reduces the errors made by the previous collection of trees. The objective is for each new tree to learn from the residuals of the previous trees and the true targets.

***k*-means** *k*-means is a simple unsupervised method for clustering unlabelled data [105]. Initially, centroids are randomly placed in the feature space. This method then updates centroids by clustering each data point based on its closest centroid, and then using the centre of mass of each cluster as the updated centroid. This method makes strict assumptions about the data being modelled: (1) Isotropic clusters: It assumes that data forms spherical clusters with equal variance and no overlaps; (2) Flat and fixed clusters: There is a fixed number of clusters with a flat structure (no hierarchy in the clusters); (3) Independence: All features are independent and hold equal importance; (4) Meaningful distance metric: The distance metric used to calculate the similarity between data points (most commonly, the Euclidean distance) is meaningful in the context of the data. Although these assumptions are restrictive, *k*-means is fast to run, making it a good candidate for large and unlabelled datasets.

All of these models, along with the assumptions mentioned above, assume that data points are independently sampled from the sample distribution.

2.3.4 Representation Learning

Representation learning, or feature learning, is an area of machine learning focused on building tools to transform inputs into features that are useful for downstream tasks or data analysis. In theory, with enough data, this can be used to replace feature engineering. Representation learning is usually conducted with a deep neural network containing many layers and can be trained using supervised and unsupervised tasks.

An example of a representation learning model that has found significant success is “word2vec” [110]. In this case, representation learning was used to encode text tokens into vectors that represented semantic meaning. Using these methods, words could be compared as if they were vectors in real space.

In supervised representation learning, we can think of a neural network, say f parametrised by θ , in two components. These two components can be parametrised by θ_1 and θ_2 with $\theta = \theta_1 \cup \theta_2$, where g describes a feature encoder and h describes a predictor:

$$f(x|\theta) = h(g(x|\theta_1), \theta_2)$$

The diagram illustrates the functional decomposition of the model $f(x|\theta)$. The input x is processed by the feature encoder $g(x|\theta_1)$, which is highlighted in a blue box. The output of g and the parameters θ_2 (highlighted in a red box) are then fed into the predictor h , which is highlighted in a red box. A blue arrow labeled 'The feature encoder' points from x to the blue box. A red arrow labeled 'The predictor' points from the blue box and the red box to the final output of the function f .

By training f on a dataset $\mathcal{D}$, we are learning a feature encoder g that best transforms data points x into values that are informative for a predictive task h . The general idea in representation learning is that the feature values outputted by g can also be useful for training models on other tasks related to the inputs x .

Further, when data is unlabelled, self-supervised learning can be used to generate pseudo-labels that are helpful for training feature encoders. Contrastive learning [111] is a common approach to this, where a feature encoder g is trained to transform data such that distances between similar pairs of samples are minimised and distances between dissimilar pairs of samples are maximised in the encoded space. In this case, pairs or triplets [112] of data points are labelled as similar or dissimilar based on a heuristic defined using knowledge of the data or task domain.

Another method for self-supervised representation learning is employed during the training of some natural language models. Bidirectional Encoder Representations from Transformers (BERT) [113], for example, is trained by masking tokens in sentences and using the language model to predict missing words. After the model is trained, the last layer, used to transform representations of sentences into probabilities over tokens, is removed to leave a feature encoder.

The goal in all cases is the same, to encode data into a space that is more meaningful for data analysis or downstream tasks.

2.4 Building Machine Learning Pipelines in This Study

When building a predictive system, the specific choice of model and pre-processing methods depends on the task. Having said this, there are some common methods and structures used in the majority of cases.

Typically, a machine learning pipeline will contain four steps: (1) feature engineering or data cleaning; (2) data imputation; (3) data normalisation; and (4) the predictive model.

2.4.1 Feature Engineering and Data Cleaning

The first of these steps involves the cleaning of a dataset, this might involve converting categorical features into one-hot-encoded features or converting free-text notes into labels. This can sometimes be the most time-consuming part of developing a machine learning system, as the quality of data has a significant effect on the performance of a trained predictive model.

Additionally, feature engineering is an important step in the process, allowing a practitioner to include some of their own prior knowledge of a task into the system. For example, when building a diagnostic tool to predict a given adverse health condition, feature engineering might include constructing features that correspond to its common symptoms.

Deep learning architectures containing large numbers of parameters might not require feature engineering, as it is hypothesised that the network should be able to learn important relationships from the data directly [114]. However, good feature engineering might reduce the parameter search space, aiding the training of models on small datasets.

This step might also include data augmentation, which can be important for training computer vision models.

2.4.2 Imputation

In healthcare, and many other fields, missing data is a common problem. Many methods have been proposed to address missing data [115, 116]. This includes categorising missing-ness into three groups:

- **Missing Completely at Random (MCAR):** The probability of missing an observation is independent of the observable and unobservable parameters, and occurs entirely at random.
- **Missing at Random (MAR):** The probability of an observation being missing is independent of the missing value, but not independent of the observed values.
- **Missing Not at Random (MNAR):** The probability of an observation being missing is dependent on the value itself.

It is not always clear which category should be assigned to missing values in a dataset, and this is an active area of research. Some machine learning models, such as decision tree-based models, naturally deal with missing values, allowing the model to learn the relationship between missingness and targets. When approaching missing data in our research, in Chapter 4, we tested multiple methods for imputation based on the MCAR and MAR assumptions. In this case, we chose the appropriate method based on the performance of predictions on the validation data.

2.4.3 Data Normalisation

Once the above have been considered, depending on the model, data should be normalised before being used for training. The majority of machine learning models benefit from training that has undergone normalisation. Tree-based models are an exception, in that data does not need to be normalised, since they consider each feature independently.

In the case of tabular data, and linear models or neural networks, normalising data can lead to considerably more robust training. This is because features on larger scales can dominate those on smaller scales during parameter updates, leading to their under-representation in a final model. Furthermore, many techniques for providing explainable predictions, assume that each feature is on the same scale. Additionally, when using distance-based models, scaling data is critical for their correct application.

The most common method for scaling features in continuous tabular data is z-score normalisation, where we remove the mean and scale the values by the standard deviation. This results in scaled features having a mean of zero and a standard deviation of one. This method is also computationally efficient.

For imaging datasets, z-score normalisation is also commonly applied, where the mean and standard deviation are calculated jointly across all pixels in a channel and the training samples.

2.4.4 The Predictive Model

In a machine learning pipeline, the predictive model learns the relationship between features and targets. The choice of which predictive model to use is one of the most important decisions when designing a machine learning system, as it allows the practitioner to impart their own knowledge of a predictive task.

Often, the structure of the data will eliminate many of the available options. For example, a convolutional neural network will commonly be used for tasks involving images, whereas a tree-based ensemble will

commonly be used for tabular datasets. However, there are still a considerable number of methods available to users that range in size from tens to billions of parameters [105, 117].

Within this thesis, we will explore a broad range of predictive models but will focus in detail on linear models, attention-based architectures (for language modelling), and convolutional neural networks.

2.5 Evaluation of Machine Learning Models in This Study

Evaluating machine learning models is important and requires careful thought to ensure that no data leakage takes place. Cross-validation is a typical candidate for hyper-parameter optimisation and evaluating models that involves separating data into training and holdout or validation sets.

The empirical risk of a model f with parameters θ on a dataset $\mathcal{D}$ made up of N pairs of features and targets $(\mathbf{x}_n, y_n)$ is as follows (Equation 2.2, in a slightly modified form):

$$R_f(\mathcal{D}; \theta) = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x}, y) \in \mathcal{D}} \mathcal{L}(f(\mathbf{x}|\theta), y)$$

Let $\hat{\theta}_f(\mathcal{D}) = \arg \min_{\theta} R_f(\mathcal{D}; \theta)$. By defining the training and validation sets of $\mathcal{D}$ as $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$, respectively, we can write the validation empirical risk as:

$$R_f(\mathcal{D}_{\text{val}}; \hat{\theta}_f(\mathcal{D}_{\text{train}})) = \frac{1}{|\mathcal{D}_{\text{val}}|} \sum_{(\mathbf{x}, y) \in \mathcal{D}_{\text{val}}} \mathcal{L}(f(\mathbf{x}|\hat{\theta}_f(\mathcal{D}_{\text{train}})), y)$$

This defines the empirical risk on the validation set $\mathcal{D}_{\text{val}}$ of the parameters obtained through training on the training set $\mathcal{D}_{\text{train}}$. By iterating over M mutually exclusive $\mathcal{D}_{\text{val}}$ sets in the dataset to perform M -fold cross-validation, the mean and standard deviation of the M results can be studied to understand variability in model performance. In this case, it is common to select $M = 5$ and therefore split the dataset such that $\mathcal{D}_{\text{train}} : \mathcal{D}_{\text{val}} = 4 : 1$, producing 5-fold cross-validation. By selecting $M = N$, we arrive at leave-one-out cross-validation, which is useful when datasets are small.

However, this can still lead to data leakage in certain scenarios. Consider the remote healthcare monitoring setting, in which participants contribute multiple data points. If cross-validation is performed in its basic form, individual data points will not appear in both the training and validation folds; however, data from the same participant might. This can lead to undetected bias in the trained models if a small group of participants generate a large proportion of the data. In these cases, it is important to perform participant-

level cross-validation by splitting the training and validation sets using the participants. In this way, in each fold, a single participant will only generate data that appears in either the training or validation split. However, this often makes it infeasible to specify a consistent training and validation set size, as varying numbers of data points are collected from each participant.

Further, using an additional test set, that is not used within the cross-validation folds, is important to avoid over-fitting; especially if decisions on the class of models or their hyper-parameters are made using the cross-validation results. In some cases, it might be beneficial to have multiple layers of cross-validation or use it alongside other data sampling techniques to ensure that hyper-parameters are tuned on a different subset of the data that the final test results are reported on.

2.5.1 Explainability of Machine Learning Outputs

Explainable AI (XAI) is an active area of research, concerned with developing methods to make machine learning models and predictions more interpretable.

In healthcare, where decisions can have long-term impacts on people's lives, this is of particular importance. Machine learning models, as complicated mathematical functions, have no mechanisms by which they can explain their own predictions. Therefore, using methods to understand how the model transforms features into predictions or to understand how important each feature is in a prediction can provide additional context to clinicians using models as tools to support decision making.

Doran et al. (2017) [118] categorised models into three levels of explainability: (1) Opaque models, which do not allow users to comprehend how the model transformed inputs into outputs; (2) interpretable models, where users can mathematically analyse the relationship between inputs and outputs; and (3) comprehensible models, where the model provides a prediction and a set of rules that support users to understand how the model works. To build robust systems for remote healthcare monitoring, predictive models with inherent explainability, such as linear models, or methods that allow for post hoc interpretability, are important.

Minh et al. (2022) [119] discussed the XAI landscape and presents a comprehensive review of recent methods. They argue that methods can be used to provide explainability in the pre-modelling, modelling, and post-modelling stages. Pre-modelling involves building an understanding of the dataset through visualisations or applying feature engineering techniques to allow for improved understanding of pre-

dictions. In the modelling stage, the decision on which model class to use can have an influence on the explainability of the system. Linear models, decision trees, and k -nearest neighbour models are examples of predictive models that are inherently explainable. Finally, post-modelling methods can be used to provide model-agnostic interpretations of predictions.

Shapley Additive Explanations (SHAP), proposed by Štrumbelj and Kononenko (2014) [120], is a common post-modelling choice since it provides a model-agnostic way of understanding feature contributions to predictions. Alongside feature engineering, this method allows the explanation of model predictions by quantifying how much each feature contributed to a prediction and in which direction. It does this by measuring how much each feature contributes to moving the prediction away from the average prediction. However, by using this method, we assume that data features are independent, feature contributions are additive, and that removing a feature from the dataset is equivalent to marginalising on it. In reality, features are frequently correlated and are not independent, which might lead to biased feature contributions.

Representation learning is also a useful technique to transform large-scale datasets into more interpretable vectors.

Within this thesis, where relevant, we will use methods from XAI to improve the interpretability of our data and methods.

2.6 Mitigating the Effects of Noisy Data in Neural Networks

In this thesis, “noisy data” refers to a multitude of uncertainties in the data being studied. Real-world data is often noisy due to measurement imprecision, whether due to the accuracy of sensors, missing data, or incorrect labelling. Within healthcare, noisy data can be a significant issue due to the scale of data collection. Noisy data can be introduced at any point during collection and is practically inevitable when humans are involved [121–123]: An expert may have insufficient information to provide the correct label [124, 125]; expert labels could be incorrect [124]; labelling might be subjective [126]; or there could be communication problems [127]. Noisy features can additionally be introduced during data collection or processing [128].

During the training of a machine learning model, noisy features and labels can have adverse effects on its performance, robustness, or generalisation.

This is illustrated when considering the use of gradient-based optimisation methods for training a model.

This is usually done by minimising the empirical risk of model parameters θ on a dataset of samples $x, y \in \mathcal{D}$ of size N (Equation 2.2):

$$R_f(\mathcal{D}; \theta) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(f(x_i|\theta), y_i)$$

Where $R(\mathcal{D}; \theta)$ is the empirical risk and f is the model with parameters θ . Updates to θ are guided by $\mathcal{L}(f(x_i|\theta), y_i)$, which defines the error between a prediction $f(x_i|\theta)$ and the true target y_i . When x_i or y_i (or both) are noisy, the model may be forced to learn relationships that do not exist in the clean data, as parameter updates might be misleading. This means that a model trained with data containing high levels of noise will likely not converge to the same parameters as a model trained on clean data, resulting in a reduction in accuracy on noise-free test examples.

Methods for dealing with noisy labels when training neural networks often fall into one of five research directions [121]:

- Sample selection: Identifying noisy examples in a dataset.
- Robust architectures: Architectural changes to a model to help with learning from noise, such as noise adaption layers to learn label transition matrices.
- Robust regularisation: Forcing a deep learning model to over-fit less on false-labelled data points.
- Robust loss functions: Improvements to the loss function.
- Loss adjustments: Using loss correction, re-weighting, or refurbishment to adjust the loss value before updating the neural network.

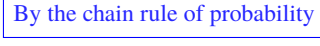
The type of noise present in a dataset can also be categorised in two ways. Noise is either instance-independent or instance-dependent, meaning that whether an observation is noisy, and in which way an observation is noisy, is either dependent on the true observation value or not. For example, a dataset with features that can be classed as one of ten labels: $c = \{0, \dots, 9\}$ has instance-dependent noise if some of the observations that should have class 5 are incorrectly labelled as class 0. The same dataset would contain instance-independent noise if an observation was equally likely to have an incorrect label, regardless of its true class.

2.7 Large Language Models

Large language models are neural networks that, at their lowest form, are autoregressive models, as they use previously generated states to predict new states.

Mathematically, given a starting state x_1 and previously generated states $\{x_2, \dots, x_{T-1}\}$, an autoregressive model generates the next state x_T by learning from the joint distribution over all states in a sequence [117]:

$$p(x_{1:T}) = p(x_1)p(x_2|x_1)p(x_3|x_2, x_1)p(x_4|x_3, x_2, x_1) \cdots = \prod_{t=1}^T p(x_t|x_{1:t-1})$$


 By the chain rule of probability

By approximating the conditional distribution $p(x_t|x_{1:t-1})$, autoregressive models can construct sequences. In this case, every new state is conditional on all of those that come before it.

Since this general idea is broad, many types of machine learning models can be used auto-regressively. For example, restricting the autoregression to order 1, so that new tokens only depend on the most recent token ($p(x_t|x_{t-1}) = p(x_t|x_{1:t-1})$), gives rise to Markov Models. However, this assumption is limiting, since a longer history of states can provide critical additional context for predicting the next state. For example, consider natural language, where the next word in a sentence depends on more than just the most recent word.

Large-capacity models are required to model long-range dependencies in sequences and look further into the past when making new state predictions. This is where LLMs, based on the self-attention architecture [129], offer a solution, since they often contain numbers of parameters on the order of billions and context lengths in the thousands. Within natural language processing, predictions over states are often referred to as predictions over tokens (which are usually words).

LLMs, trained on trillions of tokens, learn emergent behaviours for which they are not explicitly trained and are found to be useful few-shot predictors, as they encode a broad range of knowledge that can be extracted through prompting [130–135]. For example, LLMs have been evaluated for their use in healthcare [136] and have demonstrated a wide knowledge of clinical problems, with strong performance on medical questions; even without fine-tuning [137, 138]. However, using them directly in decision-making can be expensive and unreliable, and their capacity makes them difficult to fully explain.

This motivates us to consider whether their knowledge can be extracted in new ways, for use with more

cost-effective, robust, and explainable predictive models. This could be through the specification of predictive priors (Chapter 6) or the encoding of large-scale time-series data (Chapter 7).

2.8 Machine Learning with a Bayesian Perspective

Considering the optimisation objectives discussed in Section 2.3 through a probabilistic lens is important for understanding the use of Bayes' theorem in machine learning.

So far, we have considered the Maximum Likelihood Estimate (MLE), $p(\mathcal{D}, \theta)$, given some dataset $\mathcal{D}$ and parametrisation of predictive model f using θ . Using this allows us to maximise the probability that the observed data was generated using the model.

Using Bayes' theorem, we can also take a different direction. Instead, we aim to calculate the probability of the model parameters θ , given the observed data $\mathcal{D}$, written as $p(\theta|\mathcal{D})$ [105]. To see how this is different, consider a direct result of Bayes' theorem:

$$\overset{\text{The model posterior}}{p(\theta|\mathcal{D})} \propto \overset{\text{The likelihood as in Section 2.5}}{p(\mathcal{D}|\theta)} \times \overset{\text{The model prior}}{p(\theta)}$$

Bayesian inference relies on using Bayes' theorem to update the prior probability of a hypothesis based on available data. In our case, this can translate to updating our belief of the parameters θ based on observed data $\mathcal{D}$. Training machine learning models within a Bayesian framework has the added benefit of allowing us to specify a prior, $p(\theta)$, which defines our knowledge of the parameters before the data has been seen.

A well-specified prior allows models to achieve equivalent or better performance with fewer examples, as good priors prevent the model from over-fitting to limited or noisy data points and reduce the search space for the model parameters.

The key to making Bayesian predictions on new data is marginalisation. This is where parameters that we are not interested in are integrated out, or marginalised, to calculate a distribution over the parameters we are interested in. When making predictions, we use this to calculate the posterior predictive distribution over new observations. This allows us to estimate uncertainty in our predictions and provides more context when deployed. In the below, we marginalise over the posterior distribution of the parameters $p(\theta|\mathcal{D})$,

revealing the distribution over the target y , given features $\mathbf{x}$ and observed data $\mathcal{D}$.

$$p(y|\mathbf{x}, \mathcal{D}) = \int_{\Theta} p(y|\mathbf{x}, \theta) p(\theta|\mathcal{D}) d\theta$$

However, this is often challenging to calculate, since if θ is high-dimensional, the integral can be intractable. Further, in many non-linear cases, the posterior distribution does not have an analytical form. For this, we require the use of Markov Chain Monte Carlo (MCMC) or Variational Inference techniques, which can be computationally expensive.

Therefore, within the Bayesian framework, it is often easier to calculate the Maximum a Posteriori (MAP) estimate using the optimisation objective:

$$\theta_{\text{MAP}} = \arg \max_{\theta} p(\theta|\mathcal{D}) = \arg \max_{\theta} p(\mathcal{D}|\theta) \times p(\theta) = \arg \max_{\theta} [\log p(\mathcal{D}|\theta) + \log p(\theta)]$$

In this case, the problem reduces to one that can be solved using gradient-based methods as before, where the prior acts as a regularisation term.

In this thesis, when building Bayesian models, mainly in Chapter 6, we will focus on approximating the full posterior using MCMC methods, since we will restrict our analysis to linear models.

2.8.1 Prior Elicitation for Predictive Modelling

Prior elicitation is the process of specifying a prior distribution on a predictive model with the help of domain experts [139]. The procedure is standardised in Garthwaite et al. (2005) [140] and O’Hagan et al. (2006) [141], where human experts are trained in probability theory to provide judgments about a task which can be used to fit probability distributions. To simplify the process, Oakley and O’Hagan (2019) [142] later provided software and an e-learning course, which trains human experts in the knowledge required to describe informative priors.

However, prior elicitation can be costly or infeasible in low-resource settings, as it requires access to domain experts who are trained in prior elicitation [77]. Additionally, it can be challenging for experts to convert their intuitions into judgements of probability distributions that are useful for predictive models.

This motivates us to consider whether more efficient methods for prior elicitation are possible, particularly given the recent development of LLMs and their path towards becoming general knowledge bases.

2.9 Summary

This chapter summarised the background topics relevant to the research presented in this thesis. This included describing the datasets used for evaluation and the theoretical knowledge required to understand our contributions. In the next chapter, we will further contextualise our work within the broader research on remote healthcare monitoring and the areas of machine learning that we focus on.

Chapter 3

Related Work

In this chapter, we review the related work on remote healthcare monitoring for dementia care, including summarising the previous work that contributed to our Minder study. We then discuss the background literature for each of the main contributions we make within this thesis, including current methods for detecting Urinary Tract Infections (UTIs); mitigating noisy data; prior elicitation; and representation learning.

3.1 Remote Healthcare Monitoring for Dementia

Remote healthcare monitoring offers the opportunity to personalise care for people living with dementia through the use of passive sensors and wearables.

The use of wearables for the early detection of cognitive decline or for assessing the risk of dementia or related illnesses has recently found increased success. This is due to the ability of wearables to collect physiological measurements from participants directly. Schalkamp et al. (2023) [29] studied the use of accelerometer data collected using wearables, as a low-cost screening tool to determine those at risk of developing Parkinson’s disease (PD). They found that a reduction in measured acceleration prior to a disease diagnosis was unique to PD, suggesting the use of this biomarker for early screening of the condition. Furthermore, Godfrey et al. (2019) [143] described the use of inertial sensor-based wearables to quantify routine movements such as postural control and gait, which can be an early predictor of dementia or cognitive impairment [144, 145]. This study concluded that the deployment of wearables on a large scale currently remains technically and financially challenging, offering an opportunity for

low-cost passive sensing technologies instead. Furthermore, Nouredanesh et al. (2021) [146] studied the use of wearable sensors for fall detection in the literature, finding it to be a promising research direction. However, studies involving wearables are limited by the need for them to be attached directly to the participant, which presents a challenge for cohorts with cognitive decline.

Passive sensors provide an opportunity to resolve these issues by collecting data from participants without requiring their direct interaction. These sensors allow the person living with dementia to feel more independent and do not require a change in their behaviour. The authors in Alwan (2009) [147] and Alemdar and Ersoy (2010) [31] discussed a future remote healthcare monitoring system, where low-cost passive sensors are located within the homes of older adults to measure activity. Using this information, healthcare professionals would be able to plan care delivery more effectively and be alerted earlier to adverse health conditions. A predecessor of our ongoing Minder study, presented in Enshaeifar et al. (2020) [148] and later publicly released in Palermo et al. (2023) [90], described the use of passive in-home sensors for remote monitoring, avoiding many of the challenges facing the use of wearables in cohorts with cognitive impairment. An earlier version of this system was also used for the detection of adverse health conditions in Enshaeifar et al. (2018) [32] and Enshaeifar et al. (2019) [33], providing evidence for its use. However, alongside the use of passive sensors to collect activity data, this system required participants or caregivers to collect physiological measurements, which required changes in household behaviour. To overcome this, the current Minder study deploys sleep mat devices to passively monitor participants' nocturnal physiological data. As the technology advances, we envision future remote healthcare systems will also incorporate ultra-wideband radar technology for greater resolution in activity monitoring and passively collected physiological data during the day [149–151].

3.2 Predictive Models for UTIs in Dementia Care

UTIs are one of the most prevalent bacterial infections in older adults and a significant contributor to unplanned hospital admissions in People Living with Dementia (PLWD). Early detection is crucial due to the difficulty for PLWD to report symptoms and their limited help-seeking behaviour. The most common diagnostic tool is urine sample analysis, which can be time-consuming and is only employed where UTI clinical suspicion exists.

Several investigations have been conducted for the prediction of UTI risk in younger adult populations [152, 153], but these do not generalise to older adults. Further, existing methods developed for older adults rely

on typical symptoms as predictor variables, precluding their use in community-dwelling patients with dementia and atypical clinical manifestations, or those who may struggle to express symptoms [154].

For UTI detection in PLWD, machine learning offers the opportunity to improve diagnostic performance and timeliness. Due to recent developments in advanced and passive predictive models, these methods offer potential use as decision-support tools. Low-cost monitoring devices have been developed to offer complementary solutions to the typical diagnostic criteria [56, 57], showing promise by allowing the analysis of objective data not available to clinicians. With regards to this, Rantz et al. (2011) [155] use activity data collected from in-home Passive Infrared (PIR) sensors to detect UTIs in older adults, however, their work is limited to 37 participants and does not utilise ML techniques. This study is also limited to the use of activity data and does not consider physiological measurements. Extending this, Enshaeifar et al. (2019) [33] employed an unsupervised learning approach to predict UTIs based on in-home sensors and physiological measurements, however their work showed insufficient diagnostic performance and required the participant to record their own physiology measurements twice a day, a practical limitation for many people living with dementia.

This highlights several technical areas of study to improve the use of remote monitoring systems for detecting adverse health conditions. This includes the importance of using passive sensors, consulting with clinicians when developing models, providing explainable and calibrated predictions, and reducing predictive bias.

3.3 Noisy Data in Supervised Machine Learning

Healthcare data, often labelled and collected by humans, can contain unknown noise levels in features and targets. In this thesis, we study noise introduced by measurement imprecision, missing data, or incorrect labelling. This can be more challenging to mitigate in remote healthcare monitoring, as data often originates from multiple sources, each containing differing degrees of noise.

The majority of research on noisy data is focused on noisy *labels* where solutions usually target a combination of four aspects [121, 156]: sample selection, model architecture, regularisation, or training loss. Han et al. (2018) [157] proposed “Co-teaching”, a sample selection method in which two neural networks are trained simultaneously. Data points that are predicted with low error by one model are selected to “teach” the other network, as they are assumed to be more reliable. The authors exploit the

fact that neural networks learn clean data patterns and filter noisy instances in early training [62, 63]. Wang et al. (2021) [158] explored instance-dependent noise, where the noise of the label depends on the characteristics of the observation, and proposed IDPA, in which they estimate the true label of confusing instances using the confidence of the model during training. The authors Xia et al. (2021) [71] studied regularisation as a method to prevent important weights from being updated late in training. This preserves the clean data patterns learnt during the early training stages. In contrast, the authors Wei et al. (2022) [159] modified the loss during training to study how label smoothing, and negative label smoothing, can improve model accuracy when trained with noisy labels.

Weighted least squares [105], which associates a weight with each data point in a dataset when computing the likelihood, is often used in heteroskedastic regression. Here, the assumption of constant variance of residuals in regression is violated, and so the likelihood is weighted accordingly.

Noisy *inputs* are studied in the context of neural networks in Wright (1999) [160], which presents a Bayesian neural network framework that allows for training in the presence of noise given a model of the noise process. They are additionally studied in Li et al. (2021) [128], where they presented Robust Representation Learning (RRL), employing two contrastive losses and a noise cleaning method using model confidence during training. This approach requires modifications to the model architecture and a k -nearest neighbours search at each epoch.

However, these works on noisy data do not consider that data can originate from multiple sources before being combined. When multiple sources generate data, conventional training methods update models assuming equal reliability for each source and do not consider their individual data quality. However, in many applications, sources have varied levels of reliability, which can negatively affect the performance of a neural network. However, the above methods do not use the additional information that the source label can provide.

Sensor fusion [105] considers the setting where multiple observations from sensors with differing reliabilities are taken, with the goal to calculate the true observations. This is connected to our setting, in that sources of data contain different levels of imprecision. However, in sensor fusion, sources are assumed to measure the same data points rather than mutually exclusive data, as in our setup.

In federated learning, a server trains a global model using local updates on each source [161]. When sources are noisy, these algorithms often fail [162]. To address the varied data quality, Li et al. (2022) [69] proposed ARFL, which learns global and local weight updates simultaneously. The weighted sum of the

empirical risk of clients' loss is minimised, where the weights are distributed by comparing each client's empirical risk with the average empirical risk of the best k clients. The contribution to weight updates from clients with higher losses is minimised with respect to the updates from other clients, forcing the global model to learn more from clients that achieve lower losses. Here, the noise rates of individual data sources are considered during training, however the constraints of the federated learning setting limit this method's use when data can be shared.

Domain generalisation is an area of machine learning that studies methods to enable trained models to generalise to unseen data distributions [163]. In this context, many different but related domains are seen during training to produce a model that can continue to produce reliable predictions in a new test domain. To relate this field to our setting, we could consider the data sources in our context as domains, each producing the same data distribution but with different noise levels. Here, the test domain produces clean data from the same distribution as the non-noisy data in the training domains.

Current research methods for mitigating noisy data can be refined for the remote healthcare monitoring setting. This can be achieved by considering that noise might vary between different data sources, reducing the accuracy of trained models. Knowing the source that generated a data point provides important information for calculating its probability of being noisy.

3.4 Using Language Models for Prior Elicitation

In healthcare, specialised and interpretable linear models still hold considerable value. However, labelled data may be scarce or expensive to obtain. One way of addressing this issue is to use prior clinical knowledge to reduce the search space for model parameters. This can be done through feature engineering or by placing prior distributions over model parameters. Well-specified prior distributions over parameters can reduce the sample complexity of learning through Bayesian inference; however, eliciting these priors from experts can be time-consuming or infeasible. Large Language Models (LLMs) offer a solution since they acquire a breadth of information across various domains. However, their computational complexity, cost, and lack of transparency often hinder their direct application for predictive tasks in healthcare, where privacy and interpretability are paramount.

Garthwaite et al. (2005) [140] and O'Hagan et al. (2006) [141] discuss eliciting prior distributions from human experts using a standardised procedure: Experts are trained in probabilities to provide judgments

of a task, which can be used to fit probability distributions. Later, Oakley and O’Hagan (2019) [142] provided software and an e-learning course, which trains human experts in the knowledge required to describe informative priors. However, this can be challenging, costly, or infeasible in low-resource settings [77].

In parallel, research shows that LLMs encode varied domain knowledge, which could be extracted for predictive tasks, possibly in discussions with experts [164]. Recent works have used LLMs for prior elicitation in multiple ways.

For data pre-processing, Choi et al. (2022) [165] explored the use of an LLM to perform feature selection, causal discovery, and modify the reward function in a reinforcement learning task. In the first case, they observe that when listing features from two datasets, GPT-3 can correctly identify the features that correspond to a single specified predictive task. Additionally, when applied to causal discovery, the language model is often able to predict the directionality of causal relationships in the Tübingen Cause-Effect Pairs dataset [166]. Further, Hollmann et al. (2024) [167] show that LLMs can perform feature selection and engineering of a dataset. In this work, an LLM is used to suggest engineered features, which are automatically evaluated and iterated on. Similarly, Zhang et al. (2023) [168] and Zhang et al. (2024) [169] used LLMs to specify a modelling pipeline given a task description. These works show that language models have an understanding of machine learning for modelling, suggesting that we might also be able to use them to provide information about the parameters of predictive models directly.

In this direction, Gouk and Gao (2024) [170] used LLMs to generate synthetic data from public datasets, which define priors on linear models through a separate likelihood term. However, they do not discuss the memorisation of public datasets by LLMs, which is shown to be substantial [171], supported by the results in Chapter 6, making it challenging to evaluate. Related, Selby et al. (2024) [172] examined LLMs for estimating correlations between features and targets and showed that they contain informative knowledge. Additionally, Zhu and Griffiths (2024) [173] describe the use of iterated learning to elicit prior distributions from language models for causal learning, proportion estimation, and predicting everyday quantities. The authors employ a Markov chain Monte Carlo method to access an LLM’s prior knowledge by performing successive inference using previous model outputs that tend towards a limiting distribution. In doing this, the authors assume that in-context learning is a form of Bayesian inference. However, this knowledge is not used for predictive models.

LLMs can also be used to produce posterior predictive distributions. In-context learning is an active

research area, mainly focused on natural language tasks [132], with many works aiming to explain the underlying mechanisms.

In-context learning has proven to be effective for several natural language tasks, demonstrating the surprising ability of language models to perform outside of their intended use [131–135]. Brown et al. (2020) [131] studied GPT-3’s strong ability on natural language tasks after seeing a few examples, one example, or none at all. This work inspired others to study the extent to which language models can learn from demonstrations given as text alongside a task description. Further, Xie et al. (2021) [174] studied the mechanics of in-context learning and described it from a Bayesian Inference perspective. They found that in-context learning succeeds when language models can infer a shared concept across individual examples in a prompt. Min et al. (2022) [175] argued that in-context learning allows the language model to observe the label space, the distribution of input text, and the task format, as they find that swapping the real labels with random labels in demonstration examples does not significantly reduce accuracy.

Furthermore, Akyürek et al. (2022) [176] and Von Oswald et al. (2023) [177] discuss the types of learning algorithms that language models (and, more generally, transformers) can implement in-context. Focusing on linear models, Akyürek et al. (2022) [176] showed that, by construction, transformers are capable of performing gradient descent and closed-form ridge regression. Akyürek et al. (2022) [176] also demonstrated that trained in-context learners closely match predictors trained through gradient descent. Further, Von Oswald et al. (2023) [177] showed that transformers, by incorporating multilayer perceptrons into their architectures, can also solve non-linear regression tasks, and Dalal and Misra (2024) [178] proposed a Bayesian framework to reason about a language model’s internal state and suggest that in-context learning is consistent with Bayesian learning.

More closely related to our work, Requeima et al. (2024) [179] prompted language models with task descriptions and observed data points to generate posterior predictive distributions for regression tasks, which they found to outperform Gaussian processes in some tasks. This is done by repeatedly sampling a language model at a new feature value, given the previous input and output observations and a text description designed by a human expert. Additionally, Hollmann et al., Hollmann et al. (2022, 2025) [180, 181] trained a transformer to solve synthetically generated predictive tasks from a prior over tabular datasets and use in-context learning at prediction time. Given a new task and demonstrations, this transformer performs in-context learning to make predictions using inference steps only. However, since every prediction requires performing inference on a large transformer-based model, these methods can be resource-intensive and potentially unreliable.

LLMs have been evaluated for their use in healthcare [136] and have demonstrated a broad knowledge of clinical problems, with strong performance on medical questions; even without fine-tuning [137, 138]. Despite this, direct use of them for decision-making through in-context learning is not recommended as they sometimes produce inaccurate information. However, linear models have found wide and trusted uses in clinical applications [78, 79] since their model weights have intuitive descriptions. This motivates us to combine LLMs, which can reason about clinical problems and risk factors, with linear models, which are interpretable and familiar to clinicians.

In summary, predictive tasks in healthcare and remote healthcare monitoring can benefit from well-specified priors. However, prior elicitation from experts can be costly and sometimes infeasible. LLMs have demonstrated strong clinical knowledge and have been shown to be helpful in defining machine learning pipelines or making predictions directly. However, their use in specifying parameter distributions directly is unexplored, and their use as predictive models through in-context learning is not recommended. This reveals an open technical question: Can we elicit prior information from language models that can be translated into distributions over parameters of trusted and explainable linear models?

3.5 Time-Series Representation Learning

Time-series representation learning is a key area of research for remote healthcare monitoring applications and has broad uses for clinicians.

In representation learning, the goal is to encode raw data as vectors that are useful for further analysis, usually by harnessing deep learning. The literature on this topic is broad, including theoretical techniques concerned with few-shot prediction [182], continual learning [183], and multimodal learning [184, 185], as well as applications to healthcare [186], language modelling [129], and image generation [187]. In many ways, all neural networks utilise representation learning as part of their training.

For time-series data, various methods of learning data representations have been proposed. For example, Franceschi et al. (2019) [83], inspired by Mikolov et al. (2013) [188], trained an encoder-based model using a triplet loss by selecting random subsets of the same time-series recording as similar, and random subsets of other recordings as dissimilar; Eldele et al. (2021) [84] proposed training models to maximise the similarity between embeddings of weak and strong augmentations of data, which they show are useful for downstream tasks; and Zerveas et al. (2021) [85] use a transformer-encoder architecture [129] and a

linear projection over a time-series to learn representations by masking inputs and predicting the hidden values.

Nie et al. (2023) [189] propose PatchTST, adapting a language-based transformer architecture to perform representation learning on numerical-based time-series data by transforming sequences into patches of data. Chronos, a time-series forecasting method proposed by Ansari et al. (2024) [190], also transform numerical time-series into tokens that language-based transformer architectures can interpret, this time using scaling and quantisation. Both methods focus on numerical-based time-series where no qualitative information is assumed to be available about the values.

LLMs based on the self-attention architecture [129] are examples of time-series representation learning models that have found significant success due to their scalability and parallel design [191]. By viewing each word in a sequence as a point in time contextualised by the surrounding words, a transformer learns how to best encode text for a downstream task, such as next-word prediction.

Progress has been made in adapting LLMs for use outside of language modelling. Gruver et al. (2023) [192] experimented with GPT-3 [131] and LLaMA-2 [135] and found that they have a surprisingly good ability to zero-shot forecast time-series, when encoded as strings of numerical digits. Moreover, Jin et al. (2024) [193] used prompt engineering and reprogramming to adapt LLMs for classification tasks. In both, the authors tested numerical time-series data focused on predictive tasks.

Structured State Space Models (SSMs) [194, 195] have been proposed to address the computational inefficiency of transformers on long sequences. By relying on a linear scan, rather than the quadratic self-attention used in transformers, they significantly reduce the number of operations required to analyse a sequence. Despite these recent advancements, due to the success of LLMs, transformers have been studied in significantly more detail within the literature, and current open-source packages and models favour their use. This is particularly true for language modelling.

Fine-tuning LLMs allows the transfer of learning to specialised tasks initially outside of their intended uses [131, 196–198]. Additionally, since language models are trained on billions of text documents from various domains, they have been shown to encode relational knowledge [199] of text without being explicitly trained for these tasks.

The weights of pre-trained language models represent prior knowledge learnt from a large text corpus, which can be updated by fine-tuning the model on a specific time-series task. In this way, the learnt

representations capture qualitative notions of the values expressed in the time-series, as well as the data-specific knowledge acquired during fine-tuning. We consider employing this approach to encode discrete-valued activity data to evaluate the utility of pre-trained embeddings in remote healthcare monitoring.

3.6 Summary

There are still several research directions that require exploration before machine learning methods can be reliably applied to remote healthcare monitoring. UTIs are an important example of an adverse health condition that warrants the development of a predictive model, presented in Chapter 4. The methods discussed in this analysis are applicable to other remote healthcare monitoring tasks and have been extended in recent works. Further, Chapters 5, 6, and 7 extend this research by aiming to resolve some of the wider challenges in remote healthcare monitoring. As part of our methodological innovations, we will make contributions to the analysis of both labelled and unlabelled data; propose methods for mitigating the effects of noisy data; and suggest frameworks for incorporating language models into remote healthcare systems.

Chapter 4

Risk Prediction of Urinary Tract Infections

This chapter is based on Capstick et al. (2024) [1] and is licensed under a Creative Commons Attribution 4.0 International License¹.

Urinary Tract Infections (UTIs) are one of the most significant causes of unplanned hospital admissions for people living with dementia and are often preventable. In this chapter, we study the use of low-cost devices to passively measure activity, sleep, and nocturnal physiology from people living with dementia, with which we predict risk scores for UTI. For this, we engineer features in collaboration with clinicians that represent symptoms used to diagnose UTIs. In addition, we apply explainable machine learning techniques and perform stratification to support clinical translation and allow control over the balance between alert rate and sensitivity and specificity. Feature importance methods reveal that the largest contributions to the predictions were bathroom visit statistics, night-time respiratory rate, and the number of previous UTI events, in agreement with the literature.

This predictive task acts as a case study for using machine learning methods alongside remote healthcare monitoring for the prediction of adverse health conditions and will reveal some of the challenges still to be faced. Techniques presented in this chapter are task-agnostic and can be applied to other predictive tasks (as seen in Section 8.1).

¹<http://creativecommons.org/licenses/by/4.0/>

4.1 Introduction to the Problem

UTIs account for 25% of bacterial infections in older adults [41–45]. The clinical presentation of UTIs can range from self-limited illness to severe sepsis and account for 9%-31% of cases of severe sepsis, which itself has an estimated mortality of 20% - 40% [44, 46–49]. It is estimated that in the United Kingdom, over 20% of hospital beds are occupied by PLWD, with 9% of these attributed to UTIs [11, 33, 54, 55]. This is because diagnosis is often complicated by the presence of cognitive impairment due to dementia, as patients may find it challenging to report their symptoms [50–52]. As a result, acute infections might not be diagnosed until symptoms require hospitalisation [53].

Currently, a urine sample test and acute changes in baseline cognition are used to diagnose UTIs in PLWD [200]. To differentiate between asymptomatic bacteriuria and UTIs using a urine sample, clinicians rely on positive findings of bacteriuria and genitourinary symptoms. However, samples can be difficult to obtain due to urinary incontinence, cognitive impairment, sample contamination, or previous use of antibiotics [201]. These samples are currently taken on suspicion of an infection, which may cause a delay in treatment. Additionally, although they can be used as rapid detectors, dipstick tests have a high false positive rate for older adults and require action from the PLWD or their carer, which limits their effectiveness for diagnosis [43, 202]. Highlighting UTI risk by identifying early symptoms would allow for prompt diagnosis, improved health outcomes and effective allocation of healthcare resources.

This study presents a machine learning application to identify the risk of UTI events in PLWD by engineering symptom-targeted features from continuous in-home activity and physiology data collected by low-cost and passive sensors. With these features, we construct a predictive model, trained to calculate the risk of a person living with dementia having a UTI. Using optimisation and consultations with clinicians, we determine thresholds for the stratification of risk scores to improve the clinical applicability of our approach. This approach aids clinicians in the early diagnosis of UTIs and enables a better understanding of in-home behaviour at the point of clinical decision-making. The use of in-home activity and physiological data, in conjunction with machine learning methods, results in timely interventions that can have a significant impact on reducing unplanned hospital admissions for PLWD. This tool also allows for a more precise collection of urine samples for culture analysis which could result in improved clinical outcomes, a reduction in the burden on healthcare services, and decreased antibiotic overuse in PLWD.

As part of our initial testing, we evaluated the proposed UTI risk prediction algorithm in an observational

clinical study consisting of 117 participants living with dementia in their own homes, which generated 258 clinically validated labels.

As part of the ongoing work of the UK DRI Care Research and Technology Centre to acquire UKCA marking for the Minder system², we performed additional analysis using a greatly increased test dataset and a larger train dataset. We refer to this as silent-mode testing, where we use the larger test set to verify the performance reported in the initial analysis and select a Red threshold more applicable to the Minder clinical monitoring team's resources. This testing was performed on newly collected data from this cohort between 2023/01/01 and 2024/01/01, after training on data up to 2023/01/01. The silent-mode dataset, across the training and testing sets, consisted of 60,291 total days of data from 136 participants, containing 448 labelled days of positive or negative UTI from 82 participants.

4.2 The Minder UTI Dataset

Since this was an ongoing study, we evaluated our method on a subset of the Minder dataset with available UTI labels.

4.2.1 Data Collection and Definition of Outcome

As a reminder, in-home observation and measurement data was collected using low-cost off-the-shelf monitoring technologies, including PIR sensors (for measuring activity) and sleep monitoring devices. This allows us to collect behavioural information about the home as a whole, as well as physiological and sleep data from the participant only.

In total, 136 participants were selected for participation in our study, based on data collected between 2021/06/28 and 2024/01/01, with 117 contributing to the initial analysis and 19 new participants added to the silent-mode testing.

Urine samples were collected from enrolled participants and labelled by clinicians. Additionally, a baseline algorithm developed in our previous work [203] highlighted those patients that may need to be checked for symptoms of UTI by the study monitoring team. In those cases, sample collections were also organised. Once collected, a urine sample analysis was performed, and the results were sent to clinicians, who, with information from the monitoring team, determined a UTI.

²<https://www.ukdri.ac.uk/centres/care-research-technology/minder>

For our initial experimentation, we used data collected between 2021/06/28 and 2022/12/01 and trained models to predict whether a participant had a UTI on a given day (24-hour time window). In this initial analysis, 258 labelled urine samples were collected from 64 of the 117 participants, of which 81 samples were confirmed as positive cases of UTI. To increase the number of labelled days, if a single day has been labelled, we assume the preceding and following 3 days are also labelled the same, due to the nature of the infection. This extends the number of labelled days of data to 1752, consisting of 534 positives and 1218 negatives.

We further performed silent-mode testing, conducted at a later date. For this, we studied data collected between 2021/06/28 and 2024/01/01, containing 448 labelled days, from 82 participants, with 127 positive cases. In this analysis, similarly to the initial analysis, we extend the UTI labels by assuming the preceding and proceeding 3 days would also be labelled the same, producing 3058 labelled days. The test set represents the UTI labels collected between 2023/01/01 and 2024/01/01 from 104 participants with 29,539 days of collected data and 174 days labelled, whilst the training dataset includes data from 2021/06/28 to 2023/01/01 with 30,752 days of collected data from 117 participants with 274 verified UTI labels. When the labels are extended by 3 days on either side, the silent-mode dataset contains 3058 total UTI labels from 83 participants, with 838 positive days.

Later, when splitting the data into training and testing splits, we will ensure that days preceding and following a UTI label are not contained in both the training and testing splits to prevent data leakage.

The dataset composition for the initial analysis is given in Figure 4.1, which demonstrates the wealth of information available within the study. Notably, the vast majority of participants have few UTI labels, reducing the bias from overfitting during model training.

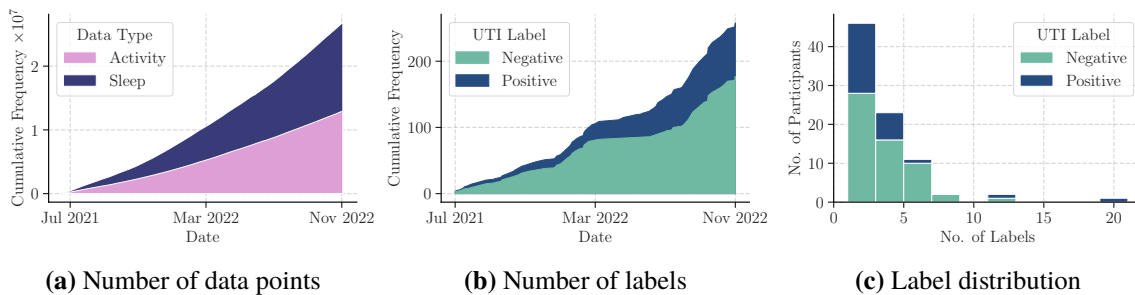


Figure 4.1: Dataset characteristics. (a) The cumulative number of sleep and activity data points recorded in total. (b) The cumulative number of clinically verified UTI labels over time. (c) The distribution of UTI labels by participant. We see that the vast majority of participants have few verified labels, which will reduce bias from overfitting in the model.

Figure 4.2 shows the scale of the silent-mode dataset and the extended size of the test set.

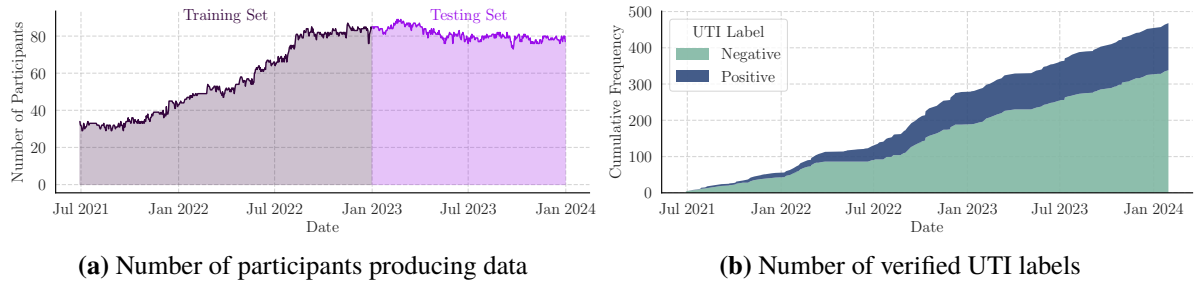


Figure 4.2: The silent-mode dataset. (a) Shows the amount of data collected each day, and whether it corresponds to the train or test set. (b) Illustrates the cumulative number of verified UTI labels (without extending them by 3 days either side) collected in the same time frame.

Assuming the 3 days preceding and proceeding a verified label have the same result, this increases the number of labelled days of data, and allows us to balance the distribution of days in the week that have a label. This is because the sample collection and testing in our study usually operates on a weekly basis, creating a distribution as shown in Figure 4.3a. If these labels were used directly, the trained machine learning model might show bias based on the recorded day. Therefore, extending the labels by 3 days on either side of the recorded test result balances the labels equally over all days of the week, allowing for more variation in the training data and providing a larger number of training and testing samples. The resulting distribution is shown in Figure 4.3b, which presents a more balanced label frequency by day.

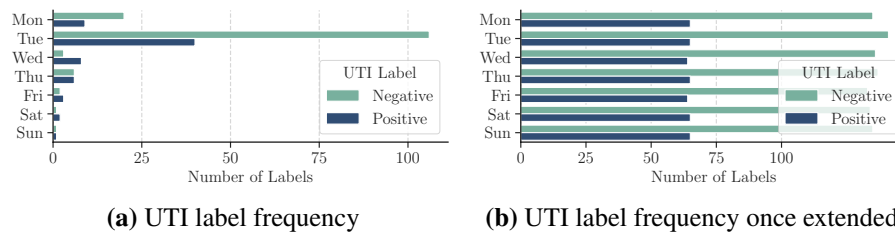


Figure 4.3: Label distribution by day. (a) The days for which we have verified UTI labels. (b) The distribution of these labels after they are extended by 3 days on either side.

To visualise the collected dataset, in Figure 4.4, we present the cohort-wide sleep and activity activations used in the initial experiments, as well as differences in sleep and activity for an example participant with both UTI positive and negative days. This figure demonstrates measurable differences between a participant's activity and sleep on days with and without a UTI, suggesting that it is possible to train a machine learning model to estimate their risk of developing a UTI.

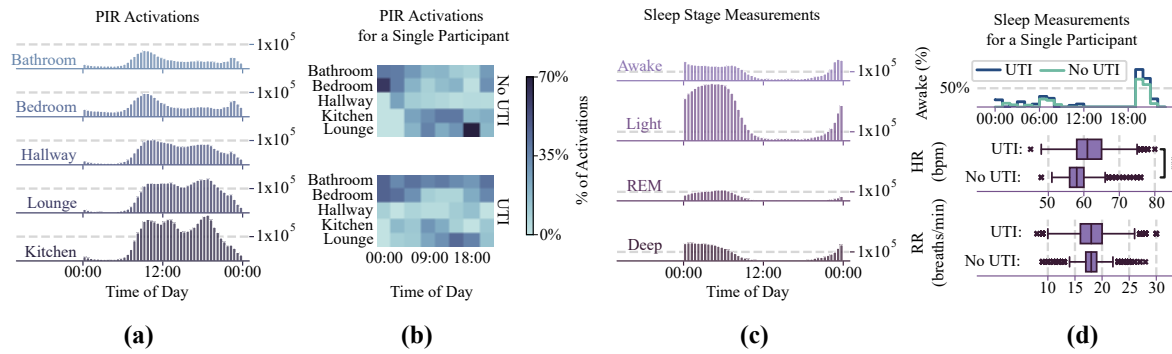


Figure 4.4: The Dataset. (a): All PIR activations measured and the corresponding time of day. (b): The average proportions of PIR activations in a time period for days containing a verified positive and negative UTI label for a selected participant. The difference in Bathroom activity is of note, showing a significant increase in use for those days corresponding to a positive UTI. (c): All sleep state measurements within the dataset and the corresponding time of day. (d): The first of the graphs shows the percentage of a given time window a single participant spent in bed and awake; the second and third show the difference in heart rate and respiratory rate distributions for days labelled as positive and negative UTI for a single participant. The elements of the boxplots correspond to: centre line, median; box limits, upper and lower quartiles; whiskers, 1.5x interquartile range; points, outliers. Here, HR corresponds to Heart Rate and RR corresponds to Respiratory Rate.

4.3 UTI Risk Prediction Methods

The UTI risk prediction model was developed in collaboration with clinicians to ensure prior knowledge was encoded into its design, its evaluation was robust, and its outputs were interpretable for healthcare professionals.

4.3.1 Data Pre-processing and Feature Engineering

In addition to using sensor firing frequencies, we perform feature engineering inspired by well-known symptoms of UTIs such as incontinence, urgency and increased frequency of urination, and behavioural changes³ to improve clinical interpretability of predictions and improve model performance and generalisability.

The raw features used in our predictive algorithm were:

- Frequency of Bathroom, Bedroom, Hallway, Kitchen, Lounge rooms: the number of times that a person enters a specific room over a day.
- Mean and Standard Deviation of Night-time Heart Rate and Respiratory Rate: mean and standard

³<https://www.nhs.uk/conditions/urinary-tract-infections-utis/>

deviation extracted from the HR and RR sensors within the bed mattress.

- Night-time Awake Occurrences: the number of times a person is awake on the bed mattress during the night.

The engineered features are:

- Bathroom Day and Night-time Frequencies, Moving Average, and Percentage Change: the number of times a person accesses the bathroom during the morning/afternoon and the night, as well as the moving average and percentage change.
- Daily Entropy: The entropy rate of a Markov chain, calculated on a transition matrix of all consecutive PIR sensor activations in a single day. This is a measure of the randomness of a person's activity patterns.
- Mean and Standard Deviation of the movement time from locations in the house to the Bathroom: mean and standard deviation of the time required to move to the bathroom from the previous location. We use this as a measure of the person's urgency to urinate.
- Number of Previous UTIs to Date: the number of times a person has suffered from a UTI before the investigated date.

Since data collection occurred outside controlled environments using in-home devices, missing measurements inevitably occurred. To limit the effects of incomplete data [204], we imputed missing values based on strategies depending on which features were being imputed.

Each raw PIR sensor data point contains a household identifier, a date and time, and the sensor that was triggered. To pre-process this data, we calculate the frequency of each sensor's activations during a day, for each participant. Since some sensor data is not complete, due to device or network issues, we perform data imputation. To do this, we assume that each of the PLWD does not have a given sensor installed until it has been active at least once before, at which point future values are imputed with 0 (instead of N/A). From these values, all of the activity-based features are then calculated.

Each night-time physiology sensor data point is collected with a PLWD identifier, a date and time, the sleep state (deep, REM, light, awake), and the heart and respiratory rate measurement. To pre-process this data, we calculate the frequency of each sleep state during a day, and the mean and standard deviations of the heart and respiratory rates.

4.3.2 Evaluating the UTI Risk Prediction Model

As part of the model development for the UTI risk prediction model, we perform two sets of evaluations.

- **Initial testing:** The first evaluation is conducted on data collected between 2021/06/28 and 2022/12/01 and is what is used for the method development. This evaluation is presented in Capstick et al. (2024) [1].
- **Silent-mode testing:** As part of the UKCA Marking testing and to verify our method, we use data collected between 2021/06/28 and 2024/01/01 and use a significantly extended testing set.

In the following sections, we will describe these two method evaluations in more detail.

Initial Analysis

To ensure generalisability, within the initial analysis we employed two evaluation strategies.

Firstly, the dataset was split into training and testing subsets in an 80:20 ratio. The data collected from 2021/06/28 to 2022/10/05 represented 80% ($n = 1394$ days in total, from $p = 54$ participants after extending the labels) of the dataset, whilst the data between 2022/10/05 and 2022/12/01 represented 20% ($n = 358$, $p = 39$ after extending the labels). In this analysis, no days from the same UTI event (within 3 days before and after a labelled event) appeared in both the training and testing sets. Figure 4.5 shows the number of participants and days of data that were used within each stage of the analysis.

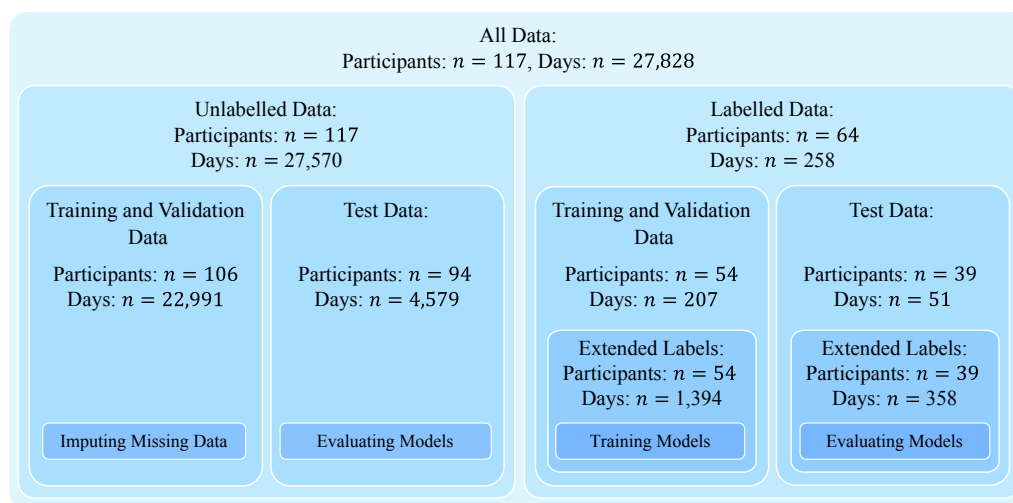


Figure 4.5: Patient disposition for the imputing, training, and evaluating stages of our analysis.

This straightforward data split formed the first analysis, evaluating the model in making predictions on future data from the same cohort on which it was trained. We will refer to this experimental setting as “Date Split”.

In the second analysis, we additionally used a leave-one-out cross-validation strategy [205]. Here, data was first split in the same way as in the first evaluation method. Then, training and testing of our machine learning models was repeated many times by including only a single PLWD in the test set at a time, whilst removing their past data from the training set. In this way, we test the model on data from participants outside of the cohort on which it is trained, and so we are evaluating its predictions on new participants. This means that no days from the same UTI event (within 3 days before and after a labelled event) appeared in both the training and testing sets. We will refer to this experimental setting as “Date-ID Split”.

During model development and whilst optimising model parameters, validation sets were produced by splitting the training data on the date 2022/09/11. All experiments were performed multiple times, with each run using a bootstrap sample [205] of the training set to ensure reproducibility. These strategies are illustrated in Figure 4.6, where the main differences occur during the final testing. Here, “Date-ID Split” uses a Leave-One-Out strategy on the IDs to measure performance and metrics are calculated over the combined predictions for all of the PLWD.

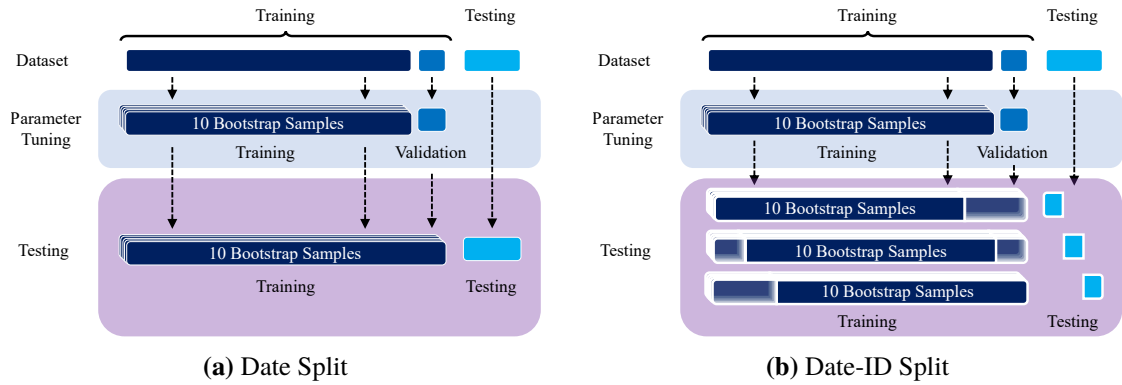


Figure 4.6: The evaluation procedures visualised. (a) “Date Split”. (b) “Date ID Split” with the Leave-One-Out evaluation acting over the PLWD.

Silent Mode Testing

To evaluate our model over a longer period, we additionally perform silent-mode testing. This was done after the original study, conducted using the data in the initial analysis, was published. For this evaluation, the data was split into a training and testing set based on data collected between 2021/06/28 to 2023/01/01 (training set: $n = 1861$ days in total, from $p = 69$ participants), and data collected between 2023/01/01

and 2024/01/01 (testing set: $n = 1197$ days in total, from $p = 55$ participants).

For this evaluation, model hyperparameters are tuned using a cross-validation approach on the training set, before being trained on the full training set. The model is then used to perform inference on the test set to measure performance metrics.

This is similar to the “Date-Split” approach used in the initial analysis, except that the hyperparameters were tuned using a shuffled cross-validation approach rather than using a fixed validation set. All experiments were performed multiple times, with each run using a bootstrap sample [205] of the training set to ensure reproducibility.

Evaluation Metrics

UTI is a common infection in older adults and, in particular, PLWD, and can result in serious complications and hospitalisation. In this study, we propose to use machine learning models to predict the risk of UTIs. However, evaluating the performance of these models is essential to ensure their reliability and usefulness in clinical practice.

In this study, we use three evaluation metrics: specificity, sensitivity, and the Area Under the Precision-Recall Curve (AUC PR). Each metric provides important information about the model’s performance, and their combined use helps provide a complete picture of the model’s predictive ability.

Sensitivity (and equivalently, recall) is the proportion of true positive predictions among all actual positive cases. It measures the model’s ability to correctly identify individuals who are at risk of developing a UTI. Specificity, on the other hand, measures the model’s ability to correctly identify individuals who are not at risk of a UTI. A high sensitivity indicates that the model is effective at identifying those who are at risk of a UTI. In contrast, a high specificity indicates that the model is effective at identifying those who are not at risk.

$$\text{Sensitivity} = \frac{TP}{TP + FN}, \quad \text{Specificity} = \frac{TN}{FP + TN}$$

$$\text{Recall} = \frac{TP}{TP + FN}, \quad \text{Precision} = \frac{TP}{TP + FP}$$

Where TP, TN, FP, and FN refer to True Positives, True Negatives, False Positives, and False Negatives, respectively.

The AUC PR is another commonly used metric that provides an overall measure of the performance of a model. It is defined as the area under the curve created by plotting the precision of a model against its recall as the thresholds on the positive prediction are varied. It is preferred over the Area Under the Receiver Operating Characteristic Curve (AUC ROC) (which plots the true positive rate against the false positive rate at various thresholds) when (1) the labels in the dataset are imbalanced [206] and (2) we aim to make reliable positive predictions, such as when predicting the risk of adverse health conditions. This is due to the formulation of both curves, where the difference stems from the use of precision in the AUC PR and the false positive rate in the AUC ROC. When the number of true positives in a dataset is low and we overestimate the positive class, the false positive rate can remain low, obscuring the model's issues. However, the precision will be reduced, making it clear that the model is overestimating the number of positives.

In summary, evaluating the performance of machine learning models for detecting the risk of UTI events using in-home monitoring data requires the use of multiple evaluation metrics. The sensitivity, specificity, and AUC PR provide important information about the model's ability to identify those at risk of a UTI and the overall performance. Understanding these metrics can help healthcare providers assess the reliability and usefulness of these models in clinical practice.

4.3.3 UTI Risk Prediction Model Development

As part of our model development, we evaluated a broad combination of predictive models, imputation strategies, feature scaling methods, and input window sizes to predict the risk of UTIs.

The predictive models' hyperparameters were tuned using Bayesian optimisation on train-validation splits, and the model producing the highest AUC PR (on validation data) was selected for the final analysis. AUC PR provides a performance measure calculated directly from the model output probability values and independent of any threshold used to define the positive and negative classes. This makes it ideal for assessing the performance of a classifier whose thresholds will be chosen later. For each of the model combinations, the number of days of data used as input to the model (input window) was tested, ranging from 1 day to 7 days.

Due to the number of hyperparameters and models we were optimising, Bayesian optimisation was used instead of a grid search to reduce the computational cost. As part of this, we used a Gaussian process to decide on the new parameters to examine at every iteration. Further, integer-valued features were transformed to continuous values before they were used in the optimisation process. For simplicity, we did not optimise over categorical hyperparameters.

All model selection was performed on the initial dataset, and then the best-performing pipeline was evaluated on the silent-mode data.

Data Imputation Strategy Candidates

After imputing values based on intuition (for example, missing activity implies zero activity), the dataset still contained some missing values that needed to be imputed before applying machine learning models. Tree-based predictive models often do not require the imputation of missing values as they can learn from them directly. However, we also test many other models that require missing data to be imputed, so we use a consistent missing data imputation approach across models. The following are the techniques that we tested:

- **Global Imputation** required using all of the data from the training sets (unlabelled and labelled data) to impute the missing values in the training, validation, and test data sets. When doing this, we imputed values using the mean of a given feature's values. Although we do not test the type of missing data, this method assumes that missingness is completely at random (MCAR described in Section 2.4.2).
- **Age Matched Imputation** was based on ensuring that data was imputed with age-matched background data. When imputing missing values from the training, validation, or testing data sets, unlabelled and labelled training data were filtered to ensure that a given point was being imputed based on the data from the most n similar ages present in the data. We set $n = 15$ in our experiments. This decision was based on the size and age distribution of the cohort to provide a sufficient and balanced number of similar matches. After the background data was chosen, we used three different approaches to perform the imputation:
 - Using the mean of features.
 - Based on a k -Nearest Neighbour strategy [207].

- Based on an iterative approach using a Random Forest Regression model [208].

These methods assume that data is missing at random (MAR described in Section 2.4.2) since the missingness is at least dependent on the age of the participant. Furthermore, using a k -nearest neighbour or random forest regression model for imputation also assumes that missing values depend on the other observed values. We do not test for the type of missing data in this work.

Ultimately, we found that applying age-matched imputation performed worse than global imputation and required higher computational costs, limiting its use in practice.

Model Input Scaling Candidates

After data imputation, before modelling, the data must be z-normalised to ensure that features are weighted equally when training. To do this, we tested two methods of data scaling:

- **Global Scaling** involved using the mean and standard deviation of each feature from training data across the cohort to z-normalise the training, validation, and testing data sets so that the training data had a mean of zero and standard deviation of one in each feature. The same scaling parameters were then used to transform the validation and testing sets to ensure no data leakage.
- **PLWD Scaling** involved calculating the mean and standard deviation on the training data for each feature and each of the PLWD separately and using these to z-normalise the training, validation, and testing data for the same PLWD. This was done in an effort to reduce the model's ability to learn to predict those PLWD who often had UTIs, rather than learning the representation of positive and negative days of UTI. When PLWD were present in the validation and testing sets but not in the training data, global statistics were used to scale their data. However, this proved to be infeasible in practice, as it meant that participants were not being consistently scaled in the same way, and some participants generated smaller portions of the data than others, leading to unreliable statistics.

Machine Learning Model Candidates

After the data was pre-processed, various machine learning models from deep learning and conventional machine learning methods were evaluated on their performance at predicting days labelled as positive or negative UTIs. We examined the following:

- Logistic Regression (LR): Using L2 regularisation with a penalty of $\lambda \in [0.01, 1]$, determined by hyperparameter optimisation. L2 regularisation is effective on datasets containing features with multicollinearity, as it encourages feature weights to be small and of a similar size for features that provide similar information. L1 regularisation, on the other hand, tends to select one of the correlated features in a group, which can make models unstable under changes in the feature values.
- Extreme Gradient Boosting (XGBoost) [109]: With a learning rate between $[0.01, 0.5]$, number of estimators between $[50, 500]$, max tree depth of $[1, 10]$, sub-sample ratio between $[0.01, 1]$, and L1 regularisation between $[0, 0.001]$ determined by hyperparameter optimisation.
- Multilayer Perceptron (MLP): With layer sizes of $(M, 50, 50, 2)$ (with M the number of features), ReLU activation function, batch size of 200, max number of epochs of 500, Adam optimiser [107] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, an L2 regularisation between $[0, 0.001]$, and a learning rate between $[0.001, 0.01]$ determined by hyperparameter optimisation.
- Self-Attention (S-Attn) model [129]: A transformer-encoder style model with 5 self-attention heads, fully connected layer sizes of 100, max number of epochs of 100, Adam optimiser [107] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, an L2 regularisation between $[0, 0.001]$, and a learning rate between $[0.001, 0.01]$ determined by hyperparameter optimisation.
- Random Forest (RF) [108]: With Gini entropy, number of estimators between $[50, 500]$ and a max depth between $[1, 100]$ determined by hyperparameter optimisation.
- Gaussian Naive Bayes (NB): With variance smoothing of 1×10^{-9} , preventing any feature's variance from becoming zero.

These models were chosen for their wide range of approaches to data modelling, including linear models, tree-based models, ensembles, probabilistic models, and deep learning. These candidates allow us to test models with varied assumptions on the data, described in Section 2.3.3.

4.3.4 Stratification of UTI Risk Scores for Improved Clinical Reporting

Previous work on predicting the occurrence of UTIs in populations has presented results as risk scores or predictions of positive or negative UTIs. However, since we plan for our model to be used as a tool for clinicians to better understand patients within their care and as an alerting system, we believe a more

advisory approach is required. Therefore, when reporting our predictions to clinicians, we stratify results into three levels of alert: Green, Amber, and Red; representing a minimal, medium, and high risk of UTI, respectively. We expect a Green alert to require no action, an Amber alert to possibly require further consideration from the clinician and analysis of in-home behaviour and medical history, and a Red alert to recommend that further analysis of symptoms or testing is performed. By using stratification, we can also control the average number of actionable alerts each week to align with a clinician's resources.

Once our chosen machine learning pipeline has been trained and the best-performing model is selected, risk scores are calculated on the validation data so that they can be used to determine the thresholds for the final model. We use the verified UTI labels on the validation set, along with the predicted risk scores, to calculate the sensitivity and specificity of stratified groups under different threshold values. These metrics need to be considered in combination with the number of Red alerts raised to ensure that the algorithm is aligned with the resources of the deployed environment. For example, thresholds that produce high sensitivity might not be viable within a given environment due to a high number of alerts (and possibly comparatively higher false positive rates).

When investigating the best thresholds, our goal was to maximise the sensitivity and specificity whilst ensuring that the number of generated alerts was realistic, considering the prevalence of UTIs and the available resources. To achieve this, we varied the stratification thresholds at a resolution of 10% and computed Youden's J statistic of the predictions on the Green and Red groups, assuming Green and Red refer to negative and positive UTI predictions, respectively. Since we are calculating the performance of different thresholds used to stratify the risk score, we require a single value that corresponds to the diagnostic performance of each stratification. Youden's J statistic provides this single value, based on both the positive and negative classes, and is commonly used in healthcare applications. We optimised these thresholds with the following criteria:

- Maximise Youden's J statistic [209], equivalent to jointly maximising sensitivity and specificity:

$$J = \text{Sensitivity} + \text{Specificity} - 1$$

- Ensure that the rate of Red alerts, R_r on the validation dataset is restricted in alignment with the availability of the care team to collect test samples and the prevalence of UTI in older adults [210]:

$$10\% \leq R_r \leq 15\%$$

Within the initial analysis, the optimal thresholds used were calculated using the algorithm’s predictions on the data collected between 2022/09/11 and 2022/10/05 (validation data) and with feedback from a clinical team.

For the silent-mode testing, we focused only on the Red threshold, which is the group for which the care monitoring team collects a urine sample for testing. For this analysis, we present only the binary classification performance, although the Amber group (using the same thresholds from the initial analysis) is used by clinicians to be alert to a participant’s changes in behaviour or physiology.

4.3.5 Analysis Platform

All analysis was performed on a secure computing environment at Imperial College London, and the code was implemented in python (v3.11.5), primarily using the following packages:

- numpy (v1.26.2) [211]: For manipulation of arrays and computing mathematical equations.
- pandas (v2.1.3) [212]: For manipulating the datasets and results.
- scikit-learn (v1.3.2) [213]: To pre-process the data and build predictive models.
- torch (v2.1.0) [214]: For building and manipulating neural networks.
- shap (v0.43.0) [215]: To calculate SHAP values.

4.4 UTI Risk Prediction Results

4.4.1 Initial Analysis

Within this section, we will present and discuss the results of the initial analysis, which includes the model development, assessment of calibration, stratification of results, and the exploration of model explainability and fairness.

Model Performance

We evaluated Logistic Regression (LR), Extreme Gradient Boosting Decision Trees (XGBoost) [109], Multilayer Perceptron (MLP), Self-Attention [129], Random Forest (RF) [108], and Naive Bayes (NB)

(Section 4.3.3) in their effectiveness to predict UTI events. We found that the best-performing classification model was LR with L2 regularisation, acting over 3 consecutive days of data.

Table 4.1 shows the AUC PR of all of the predictive models tested (with mean imputation and global scaling), whilst Table 4.2 presents the chosen model’s sensitivity, specificity, and AUC PR on the different data splits before risk stratification. The best model was selected using the AUC PR on the validation set of the “Date-ID Split”, which represents its performance in predicting UTIs on new participants. We used these criteria to ensure that our model was not biased towards those participants with recurrent UTIs which appear in the training and testing sets.

Reliability and Calibration

To motivate the use of risk stratification, we will study the calibration and reliability of our model on the validation and test sets.

Figure 4.7 shows the reliability plot for the model, which presents an agreement between the model risk scores and the observed accuracy [216–218] on the test set, calculated over both the positive and negative UTI predictions. This suggests that the risk scores can be considered reliable over both positive UTI and negative UTI predictions together. This is because we see small gaps between the average accuracy and average confidence in the lower plot, and small gaps between the true accuracy and the implied accuracy in the upper plot.

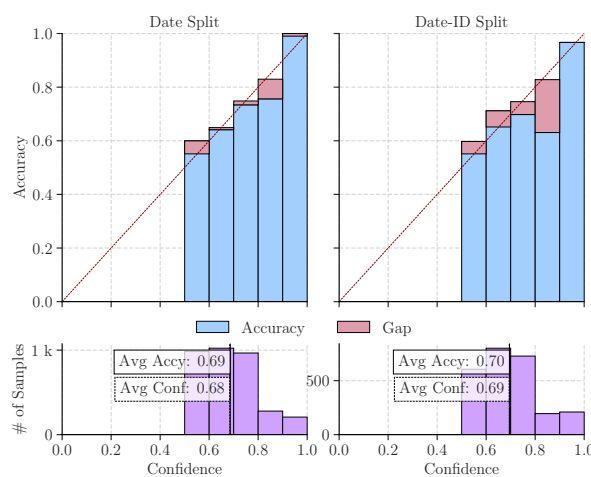


Figure 4.7: The model reliability. The top row shows the model confidence (on positive and negative UTI cases) against accuracy on the test set of “Date Split” and “Date-ID Split”. The gap shows the difference between the average accuracy and confidence of a bin, which would ideally be 0. The bottom row shows the histogram of confidences reported by the model on the test set of “Date Split” and “Date-ID Split”.

Table 4.1: UTI prediction results. Mean (95% CI) % of the AUC Precision-Recall of the different models tested with the chosen imputation and scaling methods with 10 bootstrap repeats. The bold row shows the method achieving the best result on the “Date-ID Split” and the chosen model for our final pipeline. Here, LR: Logistic Regression; MLP: Multilayer Perceptron; NB: Naive Bayes; RF: Random Forest; S-Attn: Self-Attention Transformer Encoder; XGBoost: Extreme Gradient Boosting.

Model	No. of Days	Date Split		Date-ID Split	
		Validation	Test	Validation	Test
LR	1	65.8 (64.0 - 67.7)	56.8 (55.7 - 57.9)	75.6 (73.6 - 77.5)	65.3 (63.6 - 67.0)
LR	2	67.6 (66.4 - 68.7)	55.2 (54.4 - 56.0)	76.0 (72.4 - 79.7)	61.9 (60.8 - 63.0)
LR	3	67.7 (66.4 - 69.0)	54.4 (53.4 - 55.4)	78.3 (76.8 - 79.7)	63.5 (61.8 - 65.2)
LR	4	64.2 (61.7 - 66.8)	55.9 (55.0 - 56.8)	76.0 (72.8 - 79.1)	63.5 (62.5 - 64.6)
LR	5	66.0 (63.8 - 68.3)	56.8 (55.8 - 57.9)	73.9 (67.2 - 80.6)	61.9 (61.1 - 62.6)
LR	6	73.1 (70.6 - 75.7)	57.0 (55.8 - 58.1)	72.6 (64.0 - 81.2)	63.5 (61.9 - 65.1)
LR	7	70.3 (67.8 - 72.7)	58.4 (57.4 - 59.5)	69.8 (62.6 - 77.0)	65.4 (64.5 - 66.3)
MLP	1	65.9 (63.8 - 68.1)	48.5 (45.5 - 51.5)	40.6 (30.1 - 51.0)	37.2 (28.2 - 46.2)
MLP	2	64.2 (62.7 - 65.7)	53.1 (49.8 - 56.3)	42.9 (29.6 - 56.2)	35.5 (28.0 - 43.0)
MLP	3	61.8 (58.4 - 65.2)	54.6 (52.2 - 56.9)	48.2 (33.3 - 63.1)	35.9 (26.0 - 45.7)
MLP	4	62.7 (60.5 - 64.8)	53.3 (50.2 - 56.3)	48.0 (36.3 - 59.7)	32.8 (24.3 - 41.3)
MLP	5	62.7 (60.5 - 65.0)	54.0 (50.3 - 57.7)	49.0 (40.3 - 57.6)	30.8 (23.7 - 37.9)
MLP	6	63.9 (59.5 - 68.4)	59.2 (54.9 - 63.5)	46.2 (34.2 - 58.2)	29.8 (20.0 - 39.5)
MLP	7	63.1 (57.7 - 68.5)	53.5 (49.6 - 57.5)	48.5 (37.0 - 60.1)	31.3 (18.7 - 43.9)
NB	1	63.9 (63.2 - 64.6)	59.0 (57.3 - 60.6)	69.7 (69.0 - 70.5)	68.1 (66.4 - 69.7)
NB	2	61.6 (59.5 - 63.8)	58.5 (57.1 - 59.9)	63.6 (60.4 - 66.8)	66.2 (65.0 - 67.5)
NB	3	54.5 (53.9 - 55.0)	58.6 (57.0 - 60.2)	58.4 (56.1 - 60.8)	60.7 (57.3 - 64.0)
NB	4	54.4 (53.0 - 55.8)	57.4 (55.1 - 59.7)	60.5 (58.4 - 62.6)	59.8 (58.3 - 61.4)
NB	5	58.1 (56.5 - 59.8)	56.8 (56.1 - 57.5)	64.6 (61.8 - 67.3)	58.9 (58.2 - 59.7)
NB	6	63.1 (61.1 - 65.1)	56.4 (55.5 - 57.3)	68.8 (65.2 - 72.4)	59.8 (58.6 - 61.0)
NB	7	68.3 (66.4 - 70.1)	55.0 (54.0 - 56.1)	72.3 (67.5 - 77.1)	59.4 (58.4 - 60.5)
RF	1	61.4 (60.3 - 62.5)	68.3 (66.7 - 69.9)	25.7 (21.4 - 30.0)	27.6 (26.0 - 29.1)
RF	2	61.6 (60.2 - 63.0)	70.2 (68.4 - 72.1)	26.3 (19.0 - 33.5)	30.6 (28.2 - 33.0)
RF	3	60.2 (58.4 - 61.9)	70.2 (68.5 - 72.0)	30.3 (27.2 - 33.4)	32.2 (28.3 - 36.0)
RF	4	57.7 (56.4 - 59.0)	72.5 (70.7 - 74.2)	25.1 (22.2 - 28.1)	32.2 (28.2 - 36.1)
RF	5	55.4 (53.5 - 57.2)	70.9 (68.7 - 73.1)	23.5 (21.2 - 25.8)	27.1 (25.3 - 28.9)
RF	6	55.8 (53.0 - 58.6)	73.0 (70.9 - 75.0)	23.3 (20.3 - 26.2)	25.6 (22.8 - 28.3)
RF	7	55.0 (53.1 - 57.0)	74.4 (72.0 - 76.8)	26.3 (22.5 - 30.1)	23.6 (21.9 - 25.3)
S-Attn	1	59.0 (54.7 - 63.3)	53.6 (50.7 - 56.4)	52.2 (42.9 - 61.4)	45.1 (38.4 - 51.8)
S-Attn	2	51.6 (45.0 - 58.2)	56.0 (52.2 - 59.7)	47.4 (34.0 - 60.7)	37.9 (29.8 - 46.0)
S-Attn	3	56.6 (53.8 - 59.4)	53.8 (50.9 - 56.6)	48.5 (37.7 - 59.4)	34.8 (25.8 - 43.8)
S-Attn	4	55.5 (51.6 - 59.4)	57.2 (53.4 - 61.1)	49.3 (36.6 - 62.0)	39.2 (31.4 - 46.9)
S-Attn	5	50.7 (44.4 - 57.0)	54.8 (49.3 - 60.2)	37.4 (28.2 - 46.6)	37.3 (33.0 - 41.6)
S-Attn	6	48.9 (42.1 - 55.7)	51.9 (47.0 - 56.8)	35.9 (25.7 - 46.0)	35.4 (27.9 - 42.8)
S-Attn	7	48.0 (38.7 - 57.3)	48.8 (42.8 - 54.7)	53.6 (35.0 - 72.2)	27.7 (22.6 - 32.9)
XGBoost	1	61.2 (59.5 - 62.8)	63.7 (62.4 - 65.1)	23.8 (21.5 - 26.1)	28.7 (26.0 - 31.5)
XGBoost	2	57.7 (53.8 - 61.6)	65.4 (63.4 - 67.5)	23.5 (20.6 - 26.4)	31.5 (27.3 - 35.8)
XGBoost	3	57.2 (53.8 - 60.6)	58.5 (55.7 - 61.3)	24.3 (18.5 - 30.1)	36.5 (33.2 - 39.8)
XGBoost	4	57.5 (56.7 - 58.3)	70.6 (68.3 - 72.8)	21.1 (19.0 - 23.1)	29.3 (27.8 - 30.7)
XGBoost	5	52.7 (49.8 - 55.7)	52.8 (47.8 - 57.9)	21.0 (19.1 - 22.8)	32.8 (29.2 - 36.4)
XGBoost	6	52.7 (49.8 - 55.6)	71.4 (68.7 - 74.1)	22.0 (20.2 - 23.7)	26.1 (23.5 - 28.8)
XGBoost	7	53.3 (51.3 - 55.4)	68.4 (66.2 - 70.7)	19.7 (17.6 - 21.7)	26.5 (24.8 - 28.2)

However, when considering the predicted probabilities separately, Figure 4.8 shows that the best-performing model was over-confident, a common characteristic of models trained within environments with significant class imbalance. This is because training a model on imbalanced data requires a trade-off between sample weighting in training and the true prevalence of the target. This is a key motivation for applying risk stratification, to ensure that the number of total actionable alerts and false positives can be controlled in alignment with the resources and requirements of the care team.

Table 4.2: UTI prediction results, before and after risk stratification. Mean (95% CI) % of sensitivity, specificity, and area under the precision-recall curve of the UTI prediction model on the different data splits with 10 bootstrap repeats. These results are reported before and after risk stratification is performed.

Before Risk Stratification

		Sensitivity	Specificity	AUC Precision-Recall
Date	Validation	67.3 (65.9 - 68.6)	69.1 (66.9 - 71.4)	67.7 (66.4 - 69.0)
	Test	54.5 (52.7 - 56.4)	73.0 (71.2 - 74.8)	54.4 (53.4 - 55.4)
Date-ID	Validation	87.7 (85.2 - 90.2)	66.0 (64.4 - 67.7)	78.3 (76.8 - 79.7)
	Test	65.2 (64.3 - 66.2)	70.9 (68.6 - 73.1)	63.5 (61.8 - 65.2)

After Risk Stratification

		Sensitivity	Specificity	Precision
Date	Validation	86.6 (80.9 - 92.3)	94.5 (91.7 - 97.3)	87.3 (82.2 - 92.4)
	Test	69.0 (64.4 - 73.5)	94.1 (92.0 - 96.2)	81.9 (75.5 - 88.2)
Date-ID	Validation	98.3 (95.5 - 101.1)	90.0 (85.5 - 94.5)	81.7 (74.4 - 89.1)
	Test	74.7 (67.9 - 81.5)	87.9 (85.0 - 90.9)	77.0 (71.9 - 82.1)

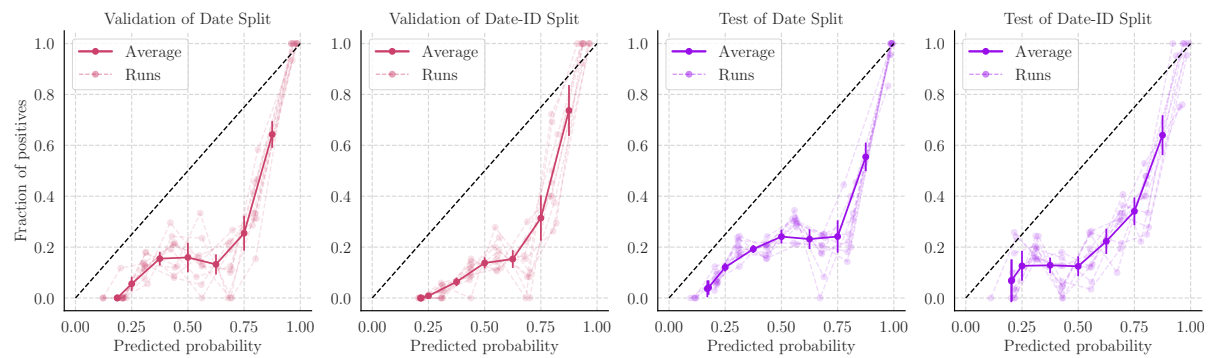


Figure 4.8: The model calibration. The mean UTI predicted risk against the proportion of positive UTI cases for the best model. This is plotted for the validation and test set from the “Date Split” and “Date-ID Split” data. The error bars represent the standard deviation of the values from the 10 bootstrap repeats.

Performance with Subsets of Sensors

We can also calculate the performance of the proposed algorithm on data collected using only a subset of the sensors. Here, we have used the same model and pre-processing but have separately trained and tested the model on data collected from the PIR and sleep sensors. Table 4.3 shows these results, in which we observe that using PIR sensors in addition to sleep sensors significantly improves the overall performance of the predictive model.

In particular, the results suggest that the PIR sensors aid in providing an improved sensitivity, whilst the sleep data helps to improve the specificity. This makes intuitive sense, since the sleep mat collects personalised data, which might help to convert some of the false positives generated by the PIR sensors, which collect household behaviour information, into true negatives based on the physiology of the PLWD.

These results also allow us to consider how the model might perform if sensors are omitted for cost-saving or specific deployment requirements. Without both sets of sensors, it might be necessary to have more

involvement from the caregiver or clinician to ensure urine testing is targeted.

Table 4.3: UTI prediction results for subsets of devices. Mean (95% CI) % of sensitivity, specificity, and area under the precision-recall curve of the UTI prediction model on the different feature subsets with 10 bootstrap repeats.

Feature Subset		Sensitivity	Specificity	AUC Precision-Recall
Date	Activity	62.6 (60.8 - 64.4)	64.8 (63.3 - 66.2)	41.4 (40.1 - 42.6)
	Sleep	12.4 (10.7 - 14.2)	66.4 (65.3 - 67.5)	21.8 (21.2 - 22.3)
	All	54.5 (52.7 - 56.4)	73.0 (71.2 - 74.8)	54.4 (53.4 - 55.4)
Date-ID	Activity	32.0 (29.0 - 34.9)	62.4 (60.8 - 64.0)	36.4 (35.0 - 37.8)
	Sleep	12.5 (11.6 - 13.3)	72.9 (71.7 - 74.1)	26.3 (25.8 - 26.9)
	All	65.3 (64.3 - 66.2)	70.9 (68.6 - 73.1)	63.5 (61.8 - 65.2)

Feature Importance

The most important features that influence the predictions were identified using Shapley Additive Explanations (SHAP) [219], a method for producing explainable predictions and calculating contributions from individual features to risk scores. However, by using this method, we assume that data features are independent, feature contributions are additive, and that removing a feature from the dataset is equivalent to marginalising on it. In reality, features are frequently correlated and are not independent, which might lead to biased feature contributions. Many features in the UTI dataset are likely correlated, such as heart rate and respiratory rate, and locations of activity around the homes. This means that some feature importance contributions found in the below analysis could be misleading. Therefore, we will aim to explain these results in Section 4.5 based on previous literature and intuition.

The results of this on the test set can be seen in Figure 4.9a and reveal that the number of previous confirmed UTI events, the standard deviation of the nocturnal respiratory rate, the nocturnal average heart rate, and the number of nocturnal awake states were positively correlated with a higher risk score. The importance of the number of previous UTIs in predicting future UTIs, in combination with current activity data, is akin to the model seeing a participant's medical history alongside their current symptoms when making a prediction.

We can also break down single predictions to understand contributions to a risk score, as shown in Figure 4.9b. Presenting predictions in this way enables clinicians to better understand the model's predictions and validate the provided risk scores.

For example, for the prediction given in Figure 4.9b, a large standard deviation of the night-time respiratory rate was the largest contributor to the positive prediction, whilst a negative relative change in the moving average of night-time bathroom use was characteristic of a negative prediction.

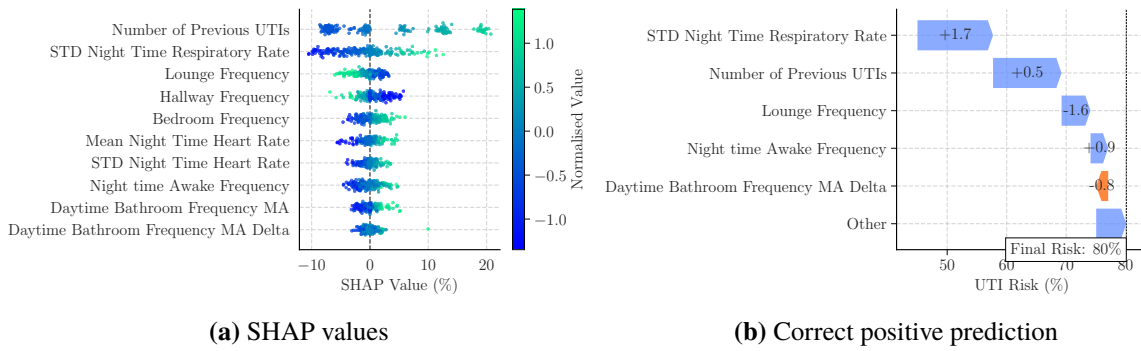


Figure 4.9: SHAP values. (a): The feature importance for the top 10 most important features, as calculated by SHAP on the test set, and their corresponding feature values. The colour represents the normalised feature value, whilst the position in the x -axis represents the contribution that value made to the prediction. “MA” refers to the moving average, whilst “Delta” refers to the percentage change in the value from the previous day. These values are calculated on the test set of the “Date-ID Split”. (b): The breakdown of a single prediction shows how each feature contributed to a correct prediction of a positive UTI. Here, the values on the arrows correspond to the normalised feature value in units of standard deviations away from the mean.

Figure 4.10 shows two other UTI predictions broken down by SHAP score contributions from each feature. In these figures, we present a correct negative prediction and a case where the model produced a correct positive prediction with low confidence.

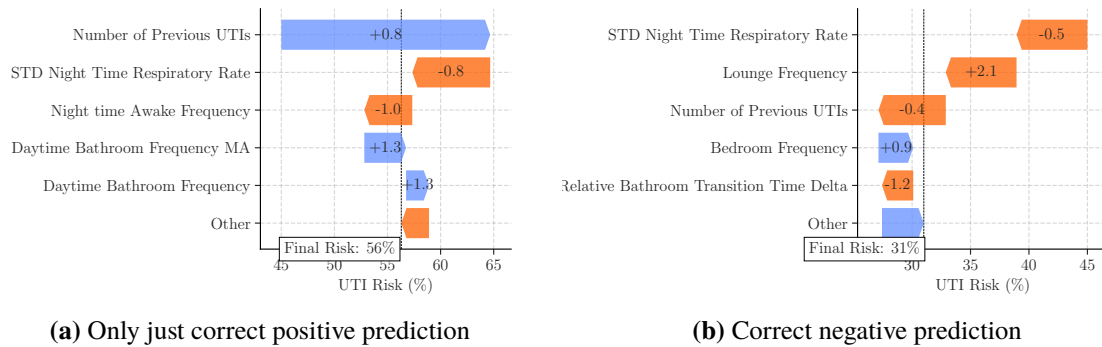


Figure 4.10: SHAP values for individual predictions. In both figures, “MA” refers to the moving average, whilst “Delta” refers to the percentage change in the value from the previous day. Additionally, all figures show the breakdown of single predictions and how each feature contributed to that prediction based on SHAP values. Here, the values on the arrows correspond to the normalised feature value in units of standard deviations away from the mean. These were calculated on the “Date-ID Split” test set.

Figure 4.11 also shows the contribution of the features to a correct prediction of positive UTI layered over a floor plan of a house for a more intuitive visualisation.

Risk Stratification

Using the method discussed in Section 4.3.4, we stratify the predicted risk scores to improve the flexibility of the model to varying clinical settings. Figure 4.12 shows the sensitivity and specificity that can be



Figure 4.11: SHAP values projected on the house layout for a single correct positive prediction. The intensity of the colours shows the relevance of the sensory values to the UTI risk scores (orange representing positive correlations and blue representing negative correlations).

achieved on the validation set by stratifying the results. By varying the stratification thresholds, sensitivity and specificity can be balanced with the number of people given Green and Red alerts. Here, we select thresholds $[0\%, 30\%]$, $(30\%, 80\%]$, and $(80\%, 100\%]$ (following interval notation) for groups Green, Amber, and Red, respectively. Table 4.2 shows the results after grouping the risk predictions on the Red and Green groups, compared to those before risk stratification.

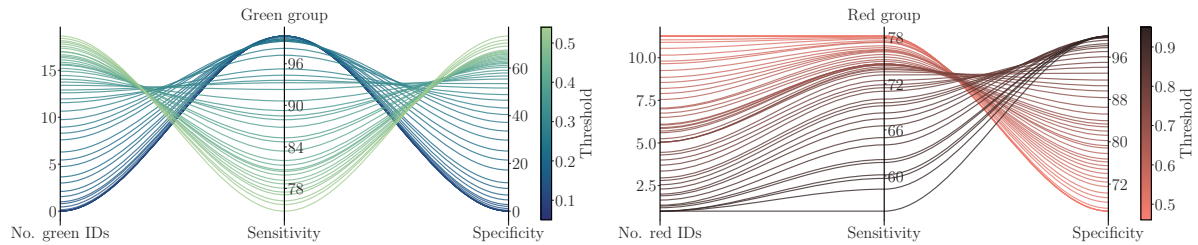


Figure 4.12: Sensitivity and specificity with different thresholds. The variations in sensitivity and specificity that can be achieved on the validation set by changing the thresholds for defining the Green and Red groups. The line colours represent different threshold values. Sensitivity and specificity are calculated on the data in the validation set corresponding to Red and Green alerts. Here, to calculate the metrics on the Green group, the Red threshold was set at $> 50\%$, and when calculating the metrics on the Red group, the Green threshold was set at $\leq 50\%$. This figure shows the average results from the “Date Split” and “Date-ID Split”.

The performance variation by threshold can also be analysed by jointly varying the Green and Red limits, as in Figure 4.13, which shows how the sensitivity and specificity on the validation set varies. By studying this figure, thresholds can be chosen to balance the metrics against the number of alerts raised in a cohort. We can see that the specificity is more sensitive to the thresholds than the sensitivity since the model predicts proportionally more false positives than false negatives.

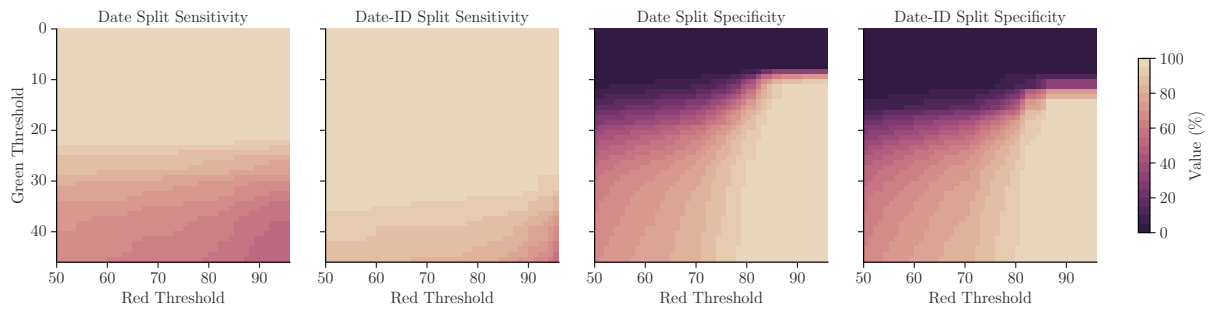


Figure 4.13: Metrics for different thresholds. The sensitivity and specificity of the model on the “Date Split” and “Date-ID Split” validation sets as the green and red thresholds are varied. Here, the results are coloured by the result rounded to the nearest 5% for ease of visualisation.

To understand the requirements of our model in a clinical setting, we also calculate the risk groups of each day of data in our test set, between 2022/10/05 and 2022/12/01, for each of the PLWD in our dataset. We find that on average, each of the PLWD will receive 0.25 Green alerts, 0.69 Amber alerts and 0.06 Red alerts each day.

Examples of Risk Score Over Time

Figure 4.14 shows how the predicted risk score varies over time on the test set of the “Date-ID Split” data. We have overlaid the stratification groups and UTI diagnoses for further context. This figure shows that the pattern of the risk score over time is highly dependent on the PLWD, with some participants showing strong increases, decreases, or stable values. PLWD 45 is an interesting case, in which the risk score was consistently and correctly high for the testing period, even though the model had never trained on this participant’s data.

Early Detection

By calculating the performance metrics for predictions made on days prior to the date of the recorded clinical urine tests, we can study the utility of the model in the early detection of UTIs. Figure 4.15 presents the specificity, sensitivity and the AUC PR for days prior to the recorded UTI events. This shows that 2 days prior to a sample test, our model achieved a sensitivity of 65.6% (95% CI = 63.0 - 68.1), specificity of 67.9 (95% CI = 63.9 - 71.9), and AUC PR of 64.9 (95% CI = 62.8 - 66.9), and 4 days prior, a sensitivity of 64.4 (95% CI = 64.4 - 64.4), specificity of 71.5 (95% CI = 67.7 - 75.2), and AUC PR of 67.1 (95% CI = 65.9 - 68.3).

These results suggest that our method can effectively alert clinicians to suspected UTIs days earlier than



Figure 4.14: UTI risk scores on “Date-ID Split” test set. The risk scores as calculated by the best model on the test set of the “Date-ID Split”, with the thresholds corresponding to the horizontal coloured bands and the vertical coloured bands representing positive and negative UTI diagnoses (in Red and Green respectively). The error bands represent the 95% confidence interval (1000 bootstrap samples) of the mean of the values from the 10 bootstrap repeats.

the collection date within our study. However, further research is needed to determine whether this is of clinical utility, since samples were not collected in our cohort at the point of presence of symptoms, but instead routinely.

Evaluation of Potential Bias

To better understand the model, we present its performance on the male and female participants within our dataset separately. To do this, we split the predictions made by the proposed model (with thresholds of $> 50\%$ = Positive and $< 50\%$ = Negative) on the test set into those made on PLWD from the male and female demographics and then calculate the mean and standard deviation of the accuracy on each of the participants. This is shown in Table 4.4 alongside the number of participants in each group and their UTI label ratios. Unfortunately, the small size of the dataset restricts the study of bias for all demographics.

This analysis highlights a discrepancy in the accuracy according to the sex of the participants. This is due to the difference in the rates of labelled positive and negative examples for each sex. We see that

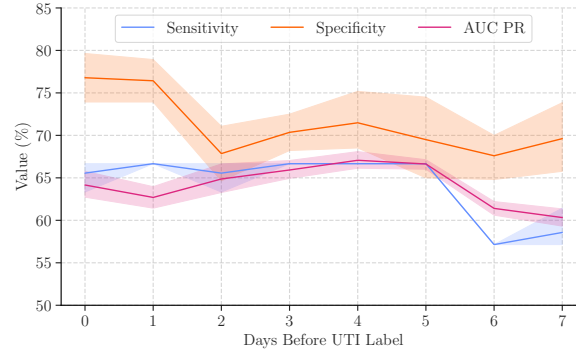


Figure 4.15: Performance at days prior to UTI label. The performance of the model when tasked with analysing the risk of a UTI from the test set, at different numbers of days prior to the verified recorded date. Note that the features contain the number of previous UTIs, which are updated with a delay of 3 weeks to ensure no data leakage. The error bands represent the 95% confidence interval (1000 bootstrap samples) of the mean of the values from the 10 bootstrap repeats.

Table 4.4: UTI prediction results, split by sex. Mean (95% CI) % of the accuracy on the male and female participants on the test set of the different data splits with 10 bootstrap repeats. We also show the ratio of positive and negative labels in each demographic and the likelihood of a positive model prediction.

Evaluation	Sex	Accuracy	No. of Participants	Positive : Negative	$p(\hat{Y} = 1 \text{Sex})$
Date	Female	52.6 (31.8 - 73.4)	16	1 : 1.7	35.5 (32.4 - 38.5)
	Male	86.5 (76.2 - 96.9)	25	1 : 5.5	32.3 (30.9 - 33.8)
Date-ID	Female	54.6 (26.8 - 82.4)	11	1 : 2.1	37.4 (33.6 - 41.1)
	Male	85.3 (72.9 - 97.7)	20	1 : 4.1	38.0 (36.4 - 39.7)

female participants are more likely to contract a UTI than male participants in our data, in agreement with clinical observations [220].

A disparity in metrics on different demographics is inevitable when their proportion of labels differs⁴, which should be considered when deploying this predictive tool. If there is sufficient data, a separate model could be trained to make predictions on each sex separately. However, for this proof-of-concept study, training a separate model for male and female participants is infeasible and would likely lead to overfitting. If a single model is desired, risk score thresholds could be adjusted individually, with a given fairness metric in mind [221].

Table 4.4 shows the likelihood of a positive UTI prediction ($\hat{Y} = 1$) by sex using our model, $p(\hat{Y} = 1|\text{Sex})$. This is referred to as demographic (or statistical) parity. We see that the model has a high demographic parity, as the likelihood of a positive prediction across demographics is close to being equal. Varying the prediction threshold would cause either increased false negatives for the female participants (when reducing the threshold) or increased false positives for the male participants (when increasing the threshold).

Given a context with a clear fairness objective, these thresholds can be optimised to suit the setting.

⁴<https://pair.withgoogle.com/explorables/measuring-fairness/>

Recurrent UTIs

Since some of the PLWD within our dataset have multiple UTI episodes, it is important to understand their effect on the performance under the “Date Split” evaluation. Recurrent UTIs will have minimal effects on the evaluation of “Date-ID Split” because, in this case, participants do not contribute data to both the training and testing splits. However, the number of previous UTIs is provided as a feature for the models, so this must still be considered. In this section, we will examine the performance of the model when making predictions on participants who had a UTI at varying numbers of days before the test set and how the number of previous UTIs affects the results.

Figure 4.16 shows the distribution of positive UTI labels in days leading up to the date of the test set. This demonstrates that the majority of positive UTI labels in our dataset were collected in the 6 months preceding the test set. Similarly, if we view the number of days between the most recent positive UTI label per participant (with at least one positive diagnosis) and the start of the test set, we see a greater skew towards recent positive UTI diagnosis. This aligns with Figure 4.1b, which shows an increase in the rate of collected labels towards the end date of the studied data. This demonstrates that when participants in the test set have had at least one previously labelled UTI, it has likely appeared recently.

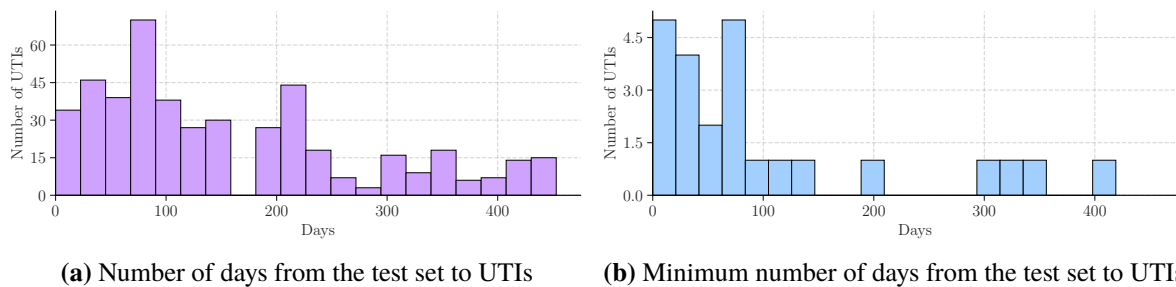


Figure 4.16: Characteristics of UTIs in the test set. (a) Histogram showing the positively labelled UTIs and the number of days they were measured before the start of the test set. (b) The same data as (a), presented as a minimum number of days for each participant.

To investigate the effect of this, in Figure 4.17a, we match each prediction in the test set of the “Date Split” evaluation with the number of days since their most recent UTI diagnosis. We then calculate the AUC PR of the model predictions on participants with varied durations of time to the most recent positive UTI. Note, to understand the performance of the model when predicting UTIs from participants that it has never seen an example for, we need to study the results of the “Date-ID Split” evaluation in Section 4.4.1. This figure shows that there is a small change in the AUC PR depending on the time between the predicted label and the last positive UTI for a given participant. This is except for those test predictions where the

participant has had a very recent positive UTI, in which case the AUC PR is considerably higher.

Figure 4.17b shows the results of the model split by participants with different numbers of previous positive UTIs on the “Date-ID Split” evaluation, demonstrating the performance based on this feature. This figure suggests that performance varies without a clear trend. This is likely because, within our dataset, the number of positive and negative labels in the test set varies significantly between groups, making it hard to interpret the performance of the model on each.

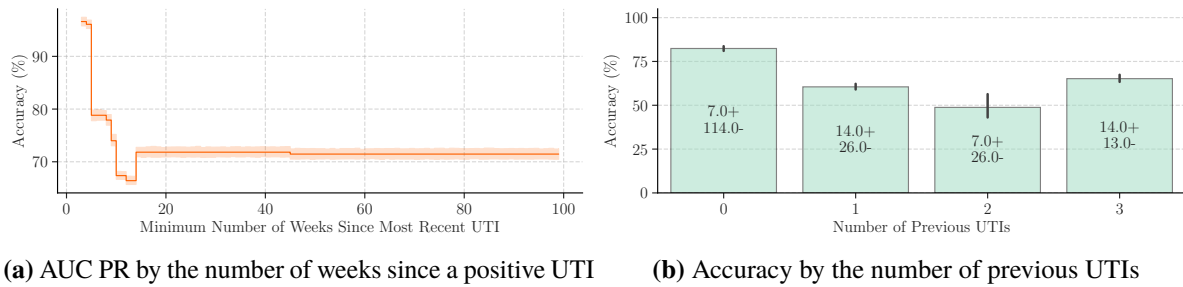


Figure 4.17: Performance by time since the most recent labelled UTI and number of UTIs. (a) The AUC PR of the predictions on participants (on the test set of “Date Split”) that had their most recent UTI positive label less than the given number of weeks before the test set. The error bands represent the 95% confidence interval (1000 bootstrap samples) of the mean of the values from the 10 bootstrap repeats. (b) The accuracy (on the test set of “Date-ID Split”) split by the number of previous UTIs. The labels on the bars represent the number of positively and negatively labelled data points (in the test set) for the corresponding group. The error bars represent the 95% confidence interval (1000 bootstrap samples) of the mean of the values from the 10 bootstrap repeats.

4.4.2 Silent-Mode Testing the UTI Risk Prediction Model

To perform the silent-mode testing, we train the UTI model on all data collected before 2023/01/01 and test it on all data between 2023/01/01 and 2024/01/01. This provides an extended test set to provide improved estimates of the algorithm’s performance in production.

Performance on Labelled Data from 2023 to 2024

For direct comparison, we evaluated the proposed method using the “Date Split” and “Date-ID Split” methodology, repeated 10 times by bootstrap sampling the training dataset. Given a training set, we use cross-validation (training : validation = 4 : 1) in combination with Bayesian optimisation to choose our hyperparameters. For these results, we use the binary classification of negative prediction: $< 50\%$ and positive prediction: $> 50\%$. The results of this are given in Table 4.5.

We found that our modelling pipeline continued to produce reliable UTI predictions, achieving a mean

Table 4.5: UTI prediction silent-mode testing results. Mean (95% CI) % of the metrics on the test set of the silent-mode testing data with 10 bootstrap repeats of the training set.

	Sensitivity	Specificity	AUC Precision-Recall
Date	72.0 (67.6 - 76.3)	81.6 (80.3 - 82.8)	59.1 (57.8 - 60.4)
Date-ID	70.4 (66.8 - 74.1)	81.5 (80.3 - 82.6)	59.0 (57.9 - 60.0)

sensitivity of 72.0% (95% CI = 67.6 - 76.3), a mean specificity of 81.6% (95% CI = 80.3 - 82.8), and a mean AUC PR of 59.1% (95% CI = 57.8 - 60.48) when using “Date Split”. We also found that on the test set of the “Date-ID Split” evaluation, we achieve a mean sensitivity of 70.4% (95% CI = 66.8 - 74.1), a mean specificity of 81.5% (95% CI = 80.3 - 82.6), and a mean AUC PR of 59.0% (95% CI = 57.9 - 60.0). The similarity in these metric values across evaluation types shows that our results are reliable.

Learning Curve

With the increased test and training set size, we evaluate the number of verified labelled examples required to achieve a given performance. This is useful for considering the deployment of the Minder system and the UTI model in different settings. For this result, we use the binary classification of negative prediction: $< 50\%$ and positive prediction: $> 50\%$. We use cross-validation (training : validation = 4 : 1) to ensure our results are reproducible, by repeatedly splitting data collected between 2021/06/28 and 2024/01/01. To construct the training data, we take random samples from each training dataset of sizes equal to 10%, 20%, ..., 100% and calculate the performance of the trained model on the validation set. This process was repeated 10 times. Hyperparameters are optimised using a cross-validation (training : validation = 4 : 1) Bayesian optimisation objective on the training dataset to maximise the AUC PR.

These results are presented in Figure 4.18, which demonstrates that the performance of the model saturated quickly, with small changes in AUC PR after ~ 100 verified UTI labels (~ 700 labelled days).

Results by Threshold

Since it was decided that in this analysis urine samples should only be collected after a Red alert is raised, we only focus on this threshold. Because of this, we treat the problem as a binary classification task, where we use the same Amber threshold as calculated in the initial analysis to notify clinicians of a participant’s changes in behaviour or physiology.

To calculate the metric values, we use the test labels collected between 2023 and 2024, and use the predictions on unlabelled data to estimate the number of raised alerts each week. This allows us to

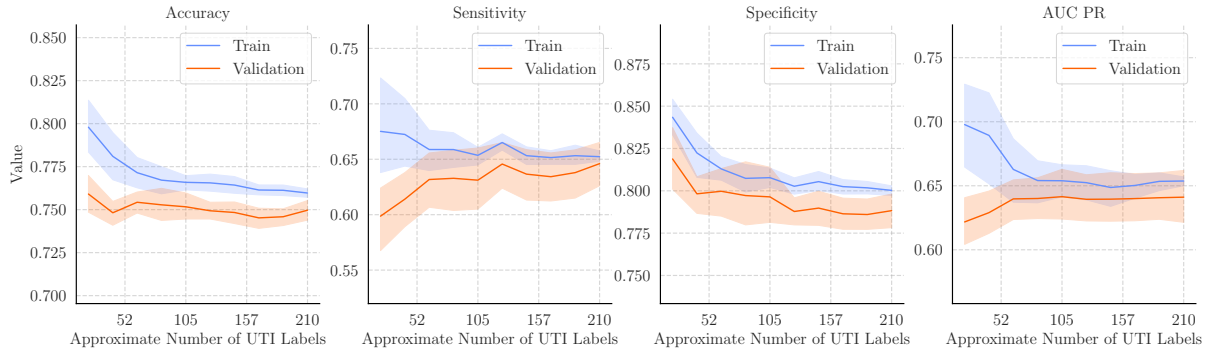


Figure 4.18: Learning curve on labelled data. The cross-validation performance of the model on labelled data collected between 2021/06/28 and 2024/01/01, as the size of the training set increases. In each fold, we select the training and validation data in the ratio 4 : 1. We then randomly sample an increasing number of labelled data points from the training set to use for model training. Here, the horizontal axis represents the verified UTI days without extending their labels by three days on either side. We repeat this 10 times and report the standard deviation of results as error bands.

maximise the model’s predictive performance whilst ensuring we are producing alerts within the resource limits of the monitoring team. To convert daily predictions of UTI risk to weekly scores, we take the maximum value over a 7-day period.

Figure 4.19 shows the number of raised alerts, precision, and sensitivity as the Red threshold increases. The number of alerts raised (30 alerts) is too great under the binary classification threshold of 50%, although this has the greatest sensitivity. Additionally, although the precision is higher for very strict thresholds, the sensitivity is significantly reduced. A threshold somewhere in the middle could set a good balance between the sensitivity and precision of predictions, whilst ensuring that the number of raised alerts is manageable.

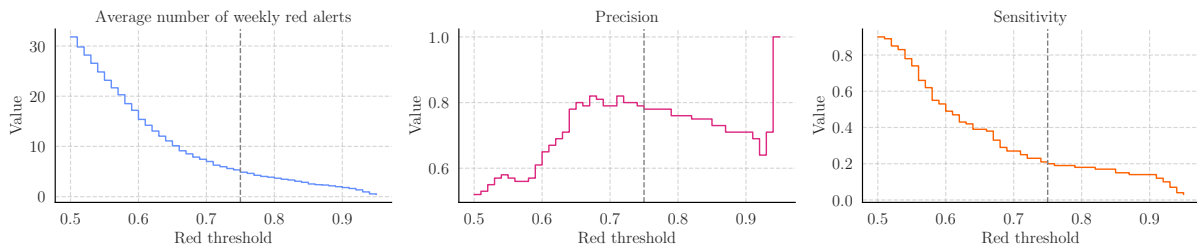


Figure 4.19: Metrics under different thresholds. The number of weekly red alerts, and their respective precision and sensitivity as the threshold for red alerts increases. The dashed vertical line represents the proposed threshold of 0.75.

Figure 4.20 shows the distribution of the predicted risk score for days with positive and negative UTI labels, which are used to calculate the metrics in Figure 4.19. Based on this and the average number of weekly alerts, we decided, in collaboration with the monitoring team, to choose a threshold of 0.75 when generating Red alerts. This is because it provides a local maximum in the precision, whilst ensuring that

the number of generated alerts is within the resource limits of the clinical monitoring team; on average 5 urine sample collections each week.

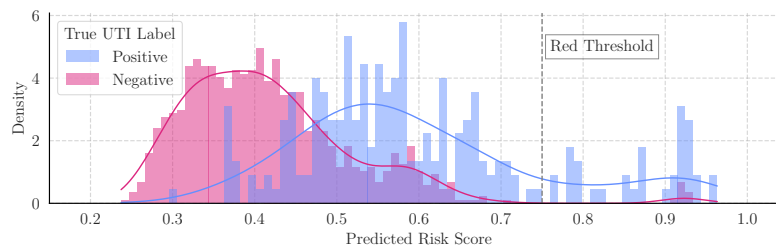


Figure 4.20: The distribution of the predicted risk score on the labelled days in the test set. The risk score predicted on those verified UTI positive or negative days within the test set. The dashed vertical line represents the proposed threshold of 0.75.

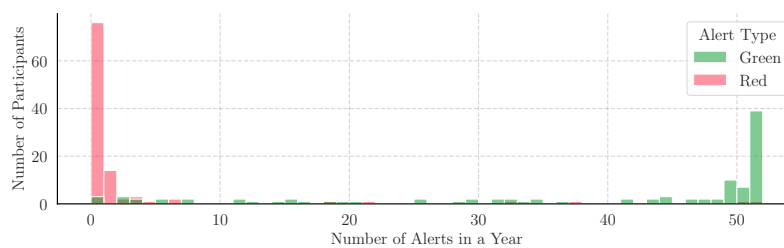


Figure 4.21: The number of participants receiving a given number of alerts over 2023 - 2024. The number of participants that have received each number of green or red alerts between 2023 and 2024. This shows that the majority of participants receive very few red alerts, however a handful of participants receive many.

The chosen threshold of 0.75, highlighted with a vertical dashed line in Figures 4.19 and 4.20, achieves the following:

- A sensitivity of 20%: We correctly predicted 20% of the cases of UTI in the cohort between 2023 and 2024.
- A precision of 77.7%: When we predict a positive UTI, we are right 77.7% of the time.
- A specificity of 98.5%: We correctly predicted 98.5% of negative cases of UTI in the cohort between 2023 and 2024.

This is all whilst generating an average of 4.86 weekly Red alerts.

For the selected threshold, Figure 4.21 shows the number of alerts that each participant could receive in a year. We observe that the vast majority of participants do not receive any Red alerts, with 14 people receiving a single Red alert over a period of twelve months. Four people received more than 30 weekly alerts, with two receiving an alert every week. For one of the participants who received an alert every

week, 16 weeks were labelled, of which 12 were labelled as UTI positive and 4 were labelled as UTI negative, a precision of 75%. For the other participants who received more than 30 alerts, no other weeks were labelled.

Threshold Performance on Unseen Participants

We also evaluated the model on participants that it had not seen during training. To do this, for each participant who had labelled data between 2023 and 2024, we removed their data from the training set (data before 2023), trained the model, and made predictions on their data between 2023 and 2024. We then calculated the metrics over all predictions on participants who have labelled data between 2023 and 2024.

With the chosen threshold of 0.75, we achieve a sensitivity of 17.6% (i.e. we catch 17.6% of UTI cases in the cohort between 2023 and 2024) and a precision of 75.9% (when we predict positive UTI, we are right 75.9% of the time). This is similar to the results achieved when the model was trained on all data between 2023 and 2024, showing good generalisation to new participants.

Compliance with Regulatory Requirements for the UTI Model

The silent-mode testing was conducted as part of our work to meet the Quality Management System (QMS) ISO 9001:2015 requirements for the encompassing Minder system. This has allowed us to progress with obtaining UK Conformity Assessed (UKCA) Class-I marking, which is currently in progress, allowing the UTI prediction model to be a part of the Minder medical device. The UKCA marking, once complete, will register the Minder system and the embedded UTI algorithm as a medical device.

In addition to the silent tests presented in this chapter, we, the care monitoring team and the software development team, made considerable effort to build and design production-ready code and a set of protocols that can be used to trigger UTI alerts in the Minder system and appropriate actions. This included processes to be followed by the clinical monitoring team based on the type of alert generated by the UTI model, including calling a participant or collecting a biological sample.

4.5 Discussion

The 2020 report of the Lancet Commission on dementia prevention, treatment, and care emphasises the significance of individualised interventions to address complex medical problems in dementia, which result in unnecessary hospital admissions, accelerated functional decline, and decreased quality of life [8]. An area of priority development is infection prevention and timely detection and treatment [222]. By conducting preliminary experiments into early identification of possible UTIs in remote healthcare settings, and verifying results with a follow-up analysis, we hope to contribute to directly addressing this priority by investigating more individualised, predictive, and preventative healthcare.

In this chapter, we present a machine learning pipeline for continuous screening of UTI risk through the analysis of passively collected in-home activity and physiological data.

We considered several models and found in the initial analysis that LR acting over 3 days attained the best performance. This model achieved a sensitivity of 65.2% (95% CI = 64.3 - 66.2) and 54.5% (95% CI = 52.7 - 56.4), and a specificity of 70.9% (95% CI = 68.6 - 73.1) and 73.0% (95% CI = 71.2 - 74.8) on “Date-ID Split” and “Date Split” respectively. The performance was higher on “Date-ID Split” than on “Date Split”, which we hypothesise is due to some PLWD who have opposing labels in the training and testing data. In this case, in “Date Split”, the model might overfit the training data of a PLWD. However, in “Date-ID Split” all the data seen by the model during testing is from participants who do not appear in training. In the initial analysis, the ratios of positive to negatives in the test sets of the “Date Split” and the “Date-ID Split” are 0.31 and 0.32, respectively and 0.47 and 0.47 for the training set of the “Date Split” and the “Date-ID Split” respectively. This difference in the proportions of the labels during validation and testing is a likely contributor to the difference in test and validation performance that we see in Table 4.2.

In the silent-mode testing, our model achieved a mean sensitivity of 72.0% (95% CI = 67.6 - 76.3), a mean specificity of 81.6% (95% CI = 80.3 - 82.8), and a mean AUC PR of 59.1% (95% CI = 57.8 - 60.48), on the test set of “Date Split”. On the “Date-ID Split”, it achieves an almost identical result, with a mean sensitivity of 70.4% (95% CI = 66.8 - 74.1), a mean specificity of 81.5% (95% CI = 80.3 - 82.6), and a mean AUC PR of 59.0% (95% CI = 57.9 - 60.0). These results demonstrate the effectiveness of our modelling pipeline and show consistent results across evaluation types, as well as between the silent-mode testing and the initial analysis.

Using stratification, risk scores were transformed into more accessible groups, allowing for flexible

management of the number and type of actionable alerts within a period of time. Following this, the performance of the Green and Red groups were significantly improved in the initial analysis, achieving a sensitivity of 74.7% (95% CI = 67.9 - 81.5) and 69.0% (95% CI = 64.4 - 73.5), and specificity of 87.9% (95% CI = 85.0 - 90.9) and 94.1% (95% CI = 92.0 - 96.2) on the “Date-ID Split” and “Date Split” respectively.

We chose a high Red group threshold for our production-ready UTI model, decided based on the silent-mode testing on the labels collected between 2023/01/01 and 2024/01/01, and removed the Amber group. This resulted in a sensitivity of 20%, but a precision of 77.7% and specificity of 98.5%, whilst producing 4.86 actionable weekly alerts on average.

Using SHAP, we highlight the features that are most strongly predictive of the risk score. Our analysis shows that an increase in the number of previously confirmed UTI events is associated with a positive prediction of UTI, which agrees with the literature [223]. This also highlights that the frequency of the lounge and hallway activations are negatively correlated with risk scores, whilst the bedroom frequency is positively correlated. We postulate that this results from participants spending more time in bed due to interrupted sleep, or due to the effects of co-morbidities. Additionally, increases in the standard deviation of the night-time respiratory rate and the night-time average heart rate were correlated with a higher risk of a UTI. This is significant, since the nocturnal respiratory rate has been associated with stress, reflects physiological and pathophysiological determinants, and has been suggested as a biomarker for impending hospitalisation [224–228]. Increased nocturnal awake occurrences were associated with a higher risk of UTI, suggesting that PLWD with UTI had more disturbed nights of sleep; in agreement with the literature [229]. This could also explain why an increase in the standard deviation of the night-time respiratory rate and the night-time average heart rate were correlated with a higher risk of UTI. Considering the clinical manifestations of UTIs in older adults, our feature importance results agree with the current understanding of UTIs in PLWD.

The proof-of-concept study presented in this chapter was conducted in real-world in-home settings on data collected in (near) real-time using off-the-shelf and low-cost sensory technologies. We designed clinically meaningful engineered features to predict UTIs determined by clinicians, urine sample analysis, and a clinical monitoring team. We provide preliminary evidence for the use of such an operation and model, which, with further testing, could prove to reduce delays in detecting UTIs in PLWD and potentially reduce the number of avoidable hospital admissions when used to support clinicians with care. The proposed approach can be scaled rapidly and enable human-in-the-loop decision support by taking advantage of

technological advancements, cloud computing, and machine learning. Moreover, risk stratification allows for model calibration to improve patient outcomes and care delivery whilst balancing the cost associated with testing for UTIs. Within an ongoing study or in production, group thresholds can be modified over time to account for the resources of the care team, as explored in Section 4.4.2. SHAP analysis allow for the presentation of explainable results (such as in Section 4.4.1), allowing clinicians to explore why the UTI algorithm has made a given prediction.

Our model will be continually monitored on newly collected data to ensure that performance consistently meets a minimum standard. We will routinely evaluate the model on a test set to track its performance and monitor feature importance to confirm that the algorithm produces clinically-founded results. This will enable rapid error debugging and maintain a high level of quality in the predictions.

4.6 Summary

When building a predictive model for the detection of UTI, it became evident that data collected from different homes can sometimes contain noise or vary in distribution. When remote healthcare monitoring systems are further scaled, this could lead to adverse effects in trained models. To begin resolving this issue, in the next chapter, we propose a method to learn from data generated by multiple sources, where some of those sources produce noisy features or labels at an unknown rate. However, in some instances, the inhomogeneity in labelling across sources might not be considered noise. For these cases, we extend our analysis in a follow-up work [72], and later [73], which studies the use of techniques for training models that can learn labellings conditioned by the data source.

Furthermore, in remote healthcare monitoring, observations of features are collected continuously; however, for some tasks, labelling data is expensive. To tackle this, in Chapter 6, we will study how language models can be used to produce prior distributions over the parameters of predictive models, improving their accuracy when labelled data is sparse. Additionally, Chapter 7 will discuss methods for using the continuously collected unlabelled data for the planning of care delivery.

Chapter 5

Mitigating the Effects of Data Sources with Unknown Reliability

This chapter is based on Capstick et al. (2025) [74] and is licensed under a Creative Commons Attribution 4.0 International License¹.

In Chapter 4, we studied the use of passive sensors and explainable machine learning methods for the prediction of UTI, an adverse health condition. As part of this data analysis, it became evident that data from different homes can contain varied levels of noise. This inspired the current chapter, where we provide a technical contribution based on learning from noisy features and labels originating from multiple data sources.

Focussing on supervised learning, our method aims to train neural networks on each data source for a number of steps proportional to the source’s estimated reliability. This is achieved through a dynamic re-weighting strategy motivated by likelihood tempering. In this way, we allow training on all sources during the warm-up stage and reduce the influence of the less reliable sources on learning during the later training stages, with the aim to prioritise those sources with higher perceived reliability. Through diverse experiments, we show that this can significantly improve model performance when trained on mixtures of reliable and unreliable data sources and maintain performance when models are trained on reliable sources only. These methods are evaluated on public datasets, commonly tested within the noisy data literature, allowing for fair comparisons.

¹<http://creativecommons.org/licenses/by/4.0/>

5.1 Introduction to the Problem

Data sources can have differing levels of noise, but in many applications, they are merged to form a single dataset. For example, in healthcare, data sources can refer to different devices, sites, or human labellers, which are often combined [60, 61]. These sources may not provide the same level of data quality and could contain noisy features, incorrect labelling, or missing values. These problems, if not addressed, can have detrimental effects on the performance and robustness of machine learning models trained on the combined data [62–64].

In this work, we consider the supervised learning case, where data is produced by multiple sources that should be producing data from the same distribution, but where some sources are actually producing noisy data at an unknown rate. Our goal is to train a neural network on data in this setting that minimises error on a non-noisy test set. Since we focus on a new area of research, in this study we mainly focus on the instance-independent noise setting, in which observations from a given source have an equal probability of being noisy.

Two possible solutions for this context naturally arise: Preprocessing can be used to remove out-of-distribution observations from training [65–67] but this requires the user to define “out-of-distribution” for the features and labels, and disregarding data assumes that nothing can be learnt from these noisy examples [68]. Secondly, techniques can be applied to train neural networks on noisy data, of which many exist [121, 156]. However, these methods consider the noise in the data as a whole and do not utilise the information that can be gained from knowing the data source of an observation.

Within federated learning, work has explored the use of noisy data techniques to address differing data quality issues across data sources [69]. However, federated learning approaches are constrained by the assumption that data cannot be transferred between sources, resulting in limited performance when methods are translated to settings where data sharing is permitted.

For this, we propose Loss Adapted Plasticity (LAP). Inspired by the current noisy data literature and likelihood tempering, LAP is a general method for training neural networks on multi-source data with mixed reliability. Since we do not know the true noise level of the sources a priori, we maintain a history of the empirical risk on data from each source that we use to temper the likelihood during training. This temperature is scheduled such that the number of steps in which a model is trained on a source is proportional to its estimated reliability. An overview of this process is given in Figure 5.1. Hence, a

model trained with LAP will benefit from seeing examples from all sources during early training, and reduce learning on less reliable data later in training, when the model is prone to memorising noisy data points [62, 63, 70, 71]. Training a model in this way considerably reduces error on a non-noisy test set when trained on data containing noisy sources.

Our method is related to weighted least squares [105], which associates a weight with each data point in a dataset when computing the likelihood. This is often used when the assumption of constant variance of residuals in regression is violated, as is the case in heteroskedastic regression. In our work, we weight samples on a per-source rather than a per-datum basis, which allows our method to share weights between data points and reduce the learning complexity.

Since we focus on mitigating the effects of training with noisy data, when viewed through the lens of domain generalisation, our method schedules the training on domains relative to how much information they provide about the test domain. This is conceptually different from the current literature in domain generalisation but could inspire new research.

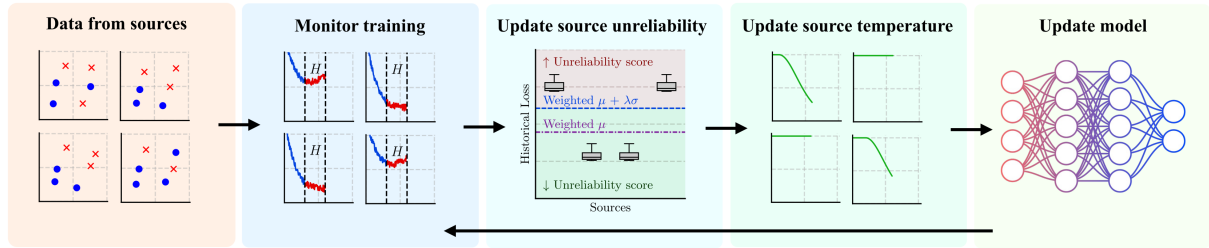


Figure 5.1: Loss Adapted Plasticity. Our method dynamically re-weights source contributions during training by comparing the historical loss values.

To illustrate the effectiveness of our method, we present results in various settings and datasets and compare our method to baselines from the literature on noisy learning [71, 128, 157–159] and federated learning [69].

Our work makes the following contributions:

- The proposed method, LAP, improves the performance of a neural network trained on data generated from sources with mixed reliability, across a diverse set of experiments in classification and for multiple data modalities.
- When there is no noise within a dataset, LAP maintains performance compared to the standard training method. This means that using our method does not result in less performant models when the data does not contain noise.

- We provide insights into the mechanics of LAP and how it enables improved performance for models trained with noisy data. As part of this, we present explanations of the hyperparameters introduced and offer intuitions for selecting values.
- We provide implementation details and code for further development, use in new settings, and ease of reproducibility.

5.2 Noisy data

Data noise can take a variety of forms and impact both the quality of labels and features. In this chapter, we study noise originating from various uncertainties in the data. In particular, we study measurement imprecision due to sensor accuracy, missing data, or incorrect labelling.

Noisy labels can be introduced at any point during collection and, when human experts are involved, are practically inevitable [121–123]: An expert may have insufficient information [124, 125]; expert labels are incorrect [124]; the labelling is subjective [126]; or there are communication problems [127].

Further, noisy features can be introduced by poor data processing or measurement errors [128].

Noisy data and features can negatively impact the training of a neural network and its performance on a clean test set. When training a neural network, we use gradient updates to iterate towards an optimal set of model weights. This is usually done by minimising the empirical risk of model parameters θ on a dataset of samples $x, y \in \mathcal{D}$ of size N :

$$R(\mathcal{D}; \theta) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(f_{\theta}(x_i), y_i)$$

Where $R(\mathcal{D}; \theta)$ is the empirical risk and f_{θ} corresponds to the neural network with parameters θ . Updates to the parameters θ are guided by $\mathcal{L}(f_{\theta}(x_i), y_i)$, which defines the error between a prediction $f_{\theta}(x_i)$ and the true target y_i . When x_i or y_i are noisy, these parameter updates can be misleading, forcing the model to learn relationships that may not exist in the clean data. Cross-entropy loss naturally downweights the influence of imprecision in labels when the model is uncertain about its prediction. However, if the model is more certain about a correct prediction that is incorrectly labelled in the dataset, it will be heavily penalised. Therefore, a model trained with highly noisy data will likely not converge to the same parameters as a model trained with non-noisy data. This can result in a reduction of accuracy on non-noisy test examples and affect the performance of a model in production.

5.2.1 Noisy Data Sources

As the mechanisms causing noisy data can affect data sources differently, we may observe varied data quality across sources.

In this setting, we have access to a dataset $\mathcal{D}$ that contains observations from multiple mutually exclusive data sources $s \in \mathcal{S}$. The dataset $\mathcal{D}$ is the union of all data from the sources $s \in \mathcal{S}$:

$$\mathcal{D} = \bigcup_{s \in \mathcal{S}} \mathcal{D}_s$$

Since each data source could be separately affected by a noisy data mechanism, this leads to a dataset $\mathcal{D}$ with a total noise rate ϵ , defined as the fraction of all data points that are unreliable:

$$\epsilon = \frac{1}{N} \sum_{s \in \mathcal{S}} |\mathcal{D}_s| \times \epsilon_s$$

Here, each source has its own noise rate $\epsilon_s \in [0, 1]$ defining the proportion of the data in that source $\mathcal{D}_s$ that is affected by noise.

5.3 Why Do We Need New Methods?

Currently, methods designed to mitigate the effects of noise train a neural network on noisy data to achieve minimal error on a non-noisy test dataset. We extend this by aiming to achieve minimal error on a non-noisy test dataset, after being trained on *multi-source* data with unknown noise levels.

To see why this could be helpful, consider the 10 data sources $\{s_i\}_{i=1}^{10}$ where one source, s_c , is producing noisy data with a probability of 0.5 and all other sources are producing clean data. Given a new observation from the noisy source, $x^c \in s_c$, consider the probability that x^c is *noisy*: $p(x^c \notin \mathcal{R})$, where $\mathcal{R}$ is the set of reliable data and assume that we know the noise rate of s_c : $p(x \notin \mathcal{R} \mid x \in s_c) = 0.5$.

Without knowing the data point's source: The probability that a data point x is unreliable (since $p(x \notin \mathcal{R} \mid x \notin s_c) = 0$) is: $p(x \notin \mathcal{R}) = p(x \in s_c)p(x \notin \mathcal{R} \mid x \in s_c) = 0.05$.

When knowing the source of the data point: The probability that a data point x^c is unreliable ($x^c \in s_c$) is $p(x^c \notin \mathcal{R} \mid x^c \in s_c) = 0.5$. Similarly, $p(x^{s \neq c} \notin \mathcal{R} \mid x^{s \neq c} \notin s_c) = 0$.

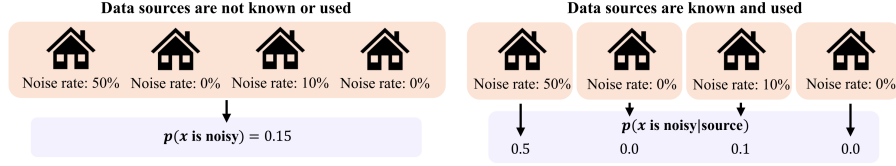


Figure 5.2: Comparing source noise estimation with and without source knowledge. This illustrates that knowing which source generated a data point provides important information about its probability of being noisy.

A similar example is given in Figure 5.2, where we consider multiple noisy sources. Here, we demonstrate that we can gain more information on the probability of a given sample being noisy by conditioning it on the source it originates from. Therefore, we hypothesise that this additional information is useful for model training when some data sources are noisy.

5.4 Our Proposed Methods

Inspired by Co-teaching, ARFL, and loss tempering, we propose Loss Adapted Plasticity (LAP), illustrated in Figure 5.1. We continually calculate a reliability score for each source as a function of its historical empirical risk on the training data. This is used to re-weight its negative log-likelihood during training such that the more reliable a source is, the more influence it has during training. We take advantage of the fact that during training, neural networks learn general data patterns before noisy data and are observed to fit to noisy examples only after substantial progress is made in fitting to clean examples. This is discussed in many works, such as Zhang et al., Arpit et al., Li et al., Han et al., Wang et al., Shen and Sanghavi, Arazo et al., Yu et al. (2021, 2017, 2021, 2018, 2021, 2019, 2019, 2019) [62, 63, 128, 157, 158, 230–232], which present both theoretical and empirical justifications.

Throughout this chapter, we will use the following shorthand notation: $p_{\theta}(\cdot) = p(\cdot|\theta)$.

5.4.1 Tempered Likelihood

Within the literature, tempered likelihood is often used to represent our belief about the aleatoric uncertainty of a dataset [233]. A temperature parameter, applied to the likelihood over samples from a dataset, down-weights their contributions in the calculation of the tempered posterior compared to the prior. As discussed in Kapoor et al. (2022) [233], this can lead to improved estimates of the uncertainty over the posterior parameters of a model and to improved accuracy. In our work, we use these ideas to temper each source

independently. Therefore, in our approach, these temperatures correspond to the perceived unreliability of each source. When training a neural network, this tempered likelihood is used to re-weight the contributions of each source to the model training.

With a dataset $\mathcal{D}$, made up of data $\mathcal{D}_s$ collected from sources $s \in S$, and a temperature parameter for each source T_s , the tempered log-likelihood for a model parametrised by θ is:

$$\log p_{\text{temp},\theta}(\mathcal{D}) = \sum_{s \in S} T_s \log p_{\theta}(\mathcal{D}_s) \quad (5.1)$$

However, since a priori T_s is unknown and we do not have access to the noise levels of each source, our prior belief is that $T_s = 1$. For this particular choice of temperatures, the tempered log-likelihood equals the non-tempered log-likelihood $\log p_{\theta}(\mathcal{D})$. Therefore, when training starts, we begin with an unmodified optimisation method.

An optimal model in our setting would achieve a maximum log-likelihood on the *clean* data. Since our sources contain a mix of noisy and non-noisy data points, we aim to approximate this. Defining $\mathcal{D}_R$ as all clean data in $\mathcal{D}$, we could set the temperature of all noisy sources to $T_s \rightarrow 0$, eliminating their contribution to the overall tempered likelihood. By calculating the weighted mean of the log-likelihood by source, we arrive at the following approximation:

$$\max_{\theta} \frac{1}{|S|} \log p_{\theta}(\mathcal{D}_R) \approx \max_{\theta} \frac{1}{\sum_{s \in S} T_s} \sum_{s \in S} T_s \log p_{\theta}(\mathcal{D}_s \cap \mathcal{D}_R) \quad (5.2)$$

This is an approximation since some sources will contain both noisy and clean samples, and so the approximation would improve as the noise levels of the sources containing noisy examples increases and the number of non-noisy sources increases.

In practice, most sources will contain both clean and noisy examples to varying degrees. Our approach, therefore, detailed in the next section, iteratively updates T_s , allowing automatic scheduling of the temperature value.

5.4.2 Source Reliability Estimation

To schedule the temperature value during training, we use the historical empirical risk of the model on the batches. The more frequently a data source produces empirical risk values greater than the mean of all other sources, the smaller T_s becomes, and the less influence that source has during training. Under the

dynamics that neural networks learn clean data before noisy data, this allows us to take advantage of the approximation given in the previous section during late training, and learn from clean data within noisy sources earlier in model training.

To provide greater control over this process, we set $T_s = f(C_s)$, where $f : [0, +\infty) \rightarrow (0, 1]$ is a positive and monotonically decreasing function. This allows us to generalise the method to other scheduling functions and interpret the value of C_s as a source’s perceived unreliability.

We update C_s by considering the number of times that a given data source provides data that achieves a significantly greater Negative Log-Likelihood (NLL) than the average of the other data sources. To do this, we maintain a history (of length H) of NLLs for each of the batches. When updating the value C_s for a single source, we consider the weighted mean $\mu_{s'}$ and standard deviation $\sigma_{s'}$ of the NLL values of all other sources from their stored history. If the mean NLL value over the history for the source in question μ_s is greater than $\mu_{s'} + \lambda\sigma_{s'}$ for a chosen λ (hyperparameter $\lambda > 0$) then we increase C_s , otherwise, we reduce it:

$$C_s = \begin{cases} C_s + 1, & \mu_s > \mu_{s'} + \lambda\sigma_{s'} \\ C_s - 1, & \text{otherwise} \end{cases} \quad (5.3)$$

To ensure that C_s remains positive and within the domain of f , we clip its value from below at 0. Alternatively, you could consider using another update rule for C_s or transforming it with a function to maintain positivity; however, this would remove the intuition behind C_s , in that it records a sense of how many update steps the source s has been perceived as unreliable. This means that C_s corresponds to how much a source’s data historically deviates from those patterns learnt by the model. This translates to only decreasing the source’s temperature C_s (and therefore its perceived unreliability) if its log-likelihood is at most λ standard deviations less than the expected tempered log-likelihood on all other sources.

This hyperparameter λ , which we refer to as the *leniency*, relates to the probability of incorrectly reducing the temperature of a non-noisy source when all sources are non-noisy. When a dataset contains noisy sources, we expect the temperature of noisy sources to be small (i.e.: $f(C_s) \ll 1$) after a number of training steps corresponding to its relative noise level – naturally enabling learning from noisy sources for a length of time that reflects their “usefulness”.

Calculating C_s

In the following, we present a detailed description of the process for calculating C_s based on a history of losses. Given a dataset $\mathcal{D}$ of features and labels that is generated by sources S , $\mathcal{S} = \{s_1, \dots, s_S\}$. We denote a subset of $\mathcal{D}$ generated by the source s as $\mathcal{D}_s \subset \mathcal{D}$, with each data point corresponding to a single source.

In a single update step of a model: we have the perceived *unreliability* for each source $C = \{C_s\}_1^S$ (initially, all sources are considered reliable and so $C_s = 0 \forall s$); a batch of features, labels, and source values from $\mathcal{D}$; and a history of training losses, L with length H for all sources. Here, $L \in \mathbb{R}^{S \times H}$, and $l_H^s \in L$ denote the mean loss of data from source s on the most recent batch containing data in $\mathcal{D}_s$. A larger C_s denotes a larger estimated noise level for the source s . In each step, we update the value C_s (and hence the temperature) for a source s using the empirical risk of all other sources s' as in Algorithm 1. Further, we define $f(C_s)$ as:

$$f(C_s) = (1 - d_s) = 1 - \tanh^2(0.005 \cdot \delta \cdot C_s) \quad (5.4)$$

Many forms of f would be suitable, however, we found experimentally that Equation 5.4 provides good learning and satisfies some nice properties discussed in Section 5.4.3. We refer to d_s as the depression value; and δ is the depression strength, controlling the rate of depression.

Algorithm 1 Calculating C_s at a given step

Require: $\lambda > 0$: Leniency

Require: $L \in \mathbb{R}^{S \times H}$: Source loss history of length H

Require: $C \in \mathbb{R}^S$: The current unreliability

$L_s = L[s]$ {Loss history on source s }

$C_s = C[s]$ {Unreliability for source s }

$\mu_s = \text{mean}(L_s)$

$W[s'] = f(C[s'])$

$\mu_{s'} = \text{weighted_mean}(L[s'], \text{weights} = W[s'])$

$\sigma_{s'}^2 = \text{weighted_var}(L[s'], \text{weights} = W[s'])$

if $\mu_s > \mu_{s'} + \lambda \sigma_{s'}$ **then**

$C_s = C_s + 1$

else if $\mu_s \leq \mu_{s'} + \lambda \sigma_{s'}$ **then**

$C_s = \max\{0, C_s - 1\}$

end if

Maximum Tempered Likelihood Estimation on Clean Data

When the dataset contains no noise and batches are large, we can show theoretically that maximising the tempered likelihood provides the same update as the unmodified maximum likelihood training at each parameter step, with a high probability.

We start with the unmodified likelihood, which is iteratively maximised. At every step, because our temperatures are calculated separately from the parameter optimisation method, we can assume that they are constant for a single-parameter update. We will denote all data in the data set as $\mathcal{D}$, while $\mathcal{D}_s$ refers to data from the source $s \in S$. Similarly, we will use the shorthand $p_\theta(\cdot) = p(\cdot|\theta)$.

$$\begin{aligned}
 \arg \max_{\theta} \log p_\theta(\mathcal{D}) & \quad \text{[Unmodified likelihood]} \\
 &= \arg \max_{\theta} \sum_{s \in S} \log p_\theta(\mathcal{D}_s) \quad \text{[Observations are independent]} \\
 &= \arg \max_{\theta} \sum_{s \in S} T_s \log p_\theta(\mathcal{D}_s) \quad \text{[When } T_s = 1 \forall s \in S\text{]}
 \end{aligned}$$

This provides the parameter that maximises the tempered likelihood. Therefore, if $T_s = 1$ for all steps, we maximise the same objective as in the unmodified maximum likelihood estimation at each step of the optimisation process.

Since all of the data is non-noisy, the expectation of the log-likelihood of a single observation will remain the same, including across sources. Therefore, given a sample B of observations from $\mathcal{D}$, those observations in B that are contained within each source will have the same mean log-likelihood on average. This is because the expectation of a sample mean is equal to the population mean.

$$\mathbb{E}\left[\frac{1}{|B \cap \mathcal{D}_i|} \log p_\theta(B \cap \mathcal{D}_i)\right] = \mathbb{E}\left[\frac{1}{|B \cap \mathcal{D}_j|} \log p_\theta(B \cap \mathcal{D}_j)\right] \quad \forall i, j \in S$$

When the batch size is large, we assume that the observations in B are split by source according to the proportional source sizes. Let $s_i = |\mathcal{D}_s|/|\mathcal{D}|$ be the proportional size of a source. If the central limit theorem is met, then in a single step, the difference of a given source's mean log-likelihood ($\mu_i = \frac{1}{|B \cap \mathcal{D}_i|} \log p_\theta(B \cap \mathcal{D}_i)$) and the mean of all of the other sources' log-likelihoods ($\mu_{i'} = \frac{1}{|B \cap \mathcal{D}'_i|} \log p_\theta(B \cap \mathcal{D}'_i)$) is distributed as:

$$\mu_{i'} - \mu_i \sim N\left(0, \frac{\text{Var}[\log p_\theta(\mathcal{D})]}{s_i |B|} + \frac{\text{Var}[\log p_\theta(\mathcal{D})]}{(1 - s_i) |B|}\right)$$

By setting $\text{Var}[\log p_\theta(\mathcal{D})] = \sigma_{\text{LL}}^2$ and simplifying the fraction, this becomes:

$$\mu_{i'} - \mu_i \sim N\left(0, \frac{\sigma_{\text{LL}}^2}{s_i(1-s_i)|B|}\right)$$

The probability that we increase the perceived unreliability of the source i at a given step with a history length of 1, using Equation 5.3 and $\lambda > 0$ is therefore:

$$p(\text{increase } C_s) = p\left(\mu_{i'} - \mu_i > \lambda \sqrt{\frac{\sigma_{\text{LL}}^2}{s_i(1-s_i)|B|}} > 0\right) > 0.5 \quad (5.5)$$

This means that under these assumptions, we expect to reduce C_s at each step in the temperature calculations, and because we clip C_s below at 0, we expect $T_s = 1$ when all data is clean. When the history length is more than 1, this can be extended by also including the expected log-likelihood over the different parameters θ at each model update step.

Equation 5.5 also theoretically describes how the value of λ influences training. This equation can be used to precisely set λ to define the probability of incorrectly increasing C_s for a given non-noisy source.

Therefore, if the batch size is large enough such that the central limit theorem applies, we expect that maximum tempered likelihood estimation provides the same update as maximum likelihood estimation at a given step. This is more likely the larger λ is.

5.4.3 Implementation Details

We require that our temperature, $f(C_s)$, monotonically decreases as C_s increases so that we reduce the impact of a source on the overall tempered likelihood as its perceived unreliability increases. We also would like $f(C_s)$ to be robust to small changes in C_s around 0 to ensure that fluctuations in the log-likelihood between batches do not incorrectly lead to reducing the temperature of a source. Because of its use as an activation function and so its wide accessibility, we choose to use:

$$f(C_s) = 1 - \tanh^2(0.005 \cdot \delta \cdot C_s) = 1 - d_s$$

Where δ , which we refer to as the *depression strength*, rescales the value C_s to control the rate of reduction in temperature. In our analysis, we additionally scale δ by 0.005 so that its value is more human-readable. We refer to $\tanh^2(0.005 \cdot \delta \cdot C_s)$ as the depression value d_s for simplicity in later sections.

During training, the gradient contributions from source s are therefore scaled as follows:

$$\hat{g}_s = f(C_s)g_s = (1 - d_s)g_s \quad (5.6)$$

Where $\hat{g}_s$ is the gradient update contribution from a source s . As $(1 - d_s)$ is a scalar, this method can be interpreted as scaling gradient contributions (presented here), loss re-weighting, or likelihood tempering.

Note that in this construction, we make no assumptions about the type of observations contained in $\mathcal{D}$. Therefore, our method applies to images, natural language, time series, and tabular data for both regression and classification tasks.

5.4.4 Design Decisions

To further understand the implementation of our method, we briefly expand upon some of the decisions made when designing our algorithm.

Weighting the Mean and Standard Deviation

We use the weighted mean and standard deviation for comparing sources with each other. Intuitively, this is done to allow for the identification of noisy sources by our method when different data sources might have significantly different noise levels. As we will see, this is the case in Figure 5.5, where we have one source producing 100% noise and other sources with noise levels 5% and 2.5%. Here, because the noise in the sources is so varied, sources with lower noise levels would likely never have an average loss more than the unweighted mean of the loss trajectory plus the threshold created from the leniency multiplied by the unweighted standard deviation (Algorithm 1). By weighting the mean and standard deviation with the calculated temperature, we can ensure that all data sources with noise will be discovered in a number of model update steps proportional to their noise level. This is shown in Figure 5.4, where the weighted mean and standard deviation of the loss moves further to the left, as the weight reduces the influence of the unreliable source (on the right).

Why the Mean + Standard Deviation for Thresholding?

When calculating whether a source should be considered more or less unreliable, we need to be able to calculate the reliability compared to the other sources. We therefore use a mean and standard deviation

over previous loss values, so that we can calculate a *relative* reliability. It also provides an intuitive idea about the probability that a non-noisy source is incorrectly identified as noisy on a single step.

Why Do We Clip Reliability at 0?

In Algorithm 1, we clip C_s at 0 to ensure that when we have sources with noise levels of different magnitudes, once the sources with larger noise levels have been heavily weighted and therefore have insignificant effects on the weighted mean and standard deviation, we are able to start reducing the reliability of other noisy sources immediately, rather than waiting for C_s to increase from some negative value to 0. Also, since we use the $\tanh^2$ function to calculate the loss (or gradient) scaling, clipping C_s at 0 prevents us from introducing an error in which loss from reliable sources is scaled with the same factor as unreliable sources.

Explanation for Using $\tanh^2$

We find that this choice of f provides some nice properties: $y = \tanh^2 x$ has a gradient of 0 at $x = 0$ and is asymptotic to $y = 1$, which are important qualities for calculating the model update scaling. The stationary point at $x = 0$ also means that small perturbations in the unreliability of each source around $C_s = 0$ have little effect on the scaling of model updates, which reduces the consequences of randomness in loss values from a given source and ensures that source contributions are only significantly scaled once it is clear that their loss values are consistently higher than those of the other sources. Secondly, since $y = \tanh^2 x$ is asymptotic to $y = 1$, the scale factor, $1 - d_s$ for sources that are calculated to be unreliable eventually reduces to 0, allowing the model to “ignore” these data points as if they were masked in late training. Further, the use of $\tanh$ as an activation function in other areas of machine learning means that it is widely accessible.

Explanation for Scaling δ

The value of 0.005 used within the $\tanh^2$ function allows the depression strength parameter δ to be on the scale of 1, as shown in Figure 5.5, providing more meaning to the shape of the function. This value is only used to aid the interpretation of the hyperparameters, and its value could easily be absorbed by δ . Using a different value has no effect on the results as long as δ is scaled proportionally.

5.4.5 Computational Complexity

The computation required to apply LAP to a single source is $O(S + B)$, and the additional memory cost is $O(S \times H)$ (where S is the number of unique sources in the dataset, B is the batch size and H is the length of the loss history), as we calculate the source means and standard deviations online. For multiple sources in a batch, the additional compute becomes $O(S \times S_b + B)$ (where S_b is the number of unique sources in a batch). The extra memory cost, in this case, is $O(S \times H + S_b \times S)$. Although vectorised in the current implementation, $O(S \times S_b)$ could be performed with S_b parallel jobs of $O(S)$, since each source calculation is independent, significantly improving its speed for larger numbers of sources. However, this is already significantly faster than the baselines tested, in which Co-teaching trains two models, IDPA trains a model for twice as long, and RRL applies k -nearest neighbours at each epoch.

In practice, using our implementation with an MLP with hidden sizes 20 – 2000 – 2000 – 5, and batches of 512 samples (with 20 features, 5 classes, and data generated from 10 sources) the forward-backward pass through the model (without source re-weighting) takes $15300 \mu s \pm 164 \mu s$ and the re-weighting of losses takes $469 \mu s \pm 10.8 \mu s$, increasing the time by 3%. With 50 and 100 sources, the re-weighting of losses takes $1000 \mu s \pm 11.3 \mu s$ and $1990 \mu s \pm 49.2 \mu s$ respectively.

Using the same set-up, with batches of 2048 in size and 10 sources, the forward-backward pass (without re-weighting) takes $41000 \mu s \pm 689 \mu s$ whilst the loss re-weighting takes $467 \mu s \pm 8.51 \mu s$ (no significant change as the batch size increases), increasing the time to compute the loss and gradients on a batch by 1%.

5.4.6 Availability of Code

We make our code publicly available:

- For generating all figures and results:
 - <https://github.com/alexcapstick/unreliable-sources>
 - <https://github.com/tmi-lab/unreliable-sources>
- For applying our methods to new tasks:
 - https://github.com/alexcapstick/loss_adapted_plasticity

– https://github.com/tmi-lab/loss_adapted_plasticity

Our code includes a pip installable library for ease of use (Figure 5.3).



Figure 5.3: The python package. `pip install loss_adapted_plasticity`.

This code is implemented in python (v3.11.5), mainly using the following packages:

- datasets (v2.19.0) [234]: For loading text datasets.
- numpy (v1.26.2) [211]: For manipulation of arrays and computing mathematical equations.
- pandas (v2.1.3) [212]: For manipulating the datasets and results.
- scikit-learn (v1.3.2) [213]: For loading and preprocessing the data.
- torch and torchvision (v2.1.0 and v0.16.1) [214]: For building and manipulating neural networks.
- transformers (v4.46.0) [235]: For loading and using language models.
- wfdb (v4.1.2) [234]: For processing ECG data.

Other packages required for specific experiments are given in the publicly available code.

All tested datasets are publicly available and easily accessible. Additionally, within the code, we use the default Pytorch Dataset objects for CIFAR-10, CIFAR-100, and F-MNIST and provide Pytorch Dataset objects that will automatically download, unzip, and load the data for all other experiments. We also provide code to load Tiny-ImageNet and ImageNet from a local directory since an agreement is required before accessing them. This makes the reproduction of the work presented simple to perform.

The baselines were made available by the authors on GitHub:

- ARFL [162]: MIT License: <https://github.com/lishenghui/arfl>.
- IDP [158]: License not provided: <https://github.com/QizhouWang/instance-dependent-label-noise>.

- Co-teaching [157]: License not provided: <https://github.com/bhanML/Co-teaching>.
- RRL [128]: BSD 3-Clause License: <https://github.com/salesforce/RRL>.

All experiments were carried out on a machine with a single Nvidia A100 (80GB VRAM) and 32GB of RAM. The full research project required more compute cost than the experiments reported in the paper to design and implement LAP, as well as to choose training parameters for the standard training model that were then used for LAP and the baselines. In total, using this hardware, experiments took approximately 15 days to complete.

5.5 Evaluation of LAP

5.5.1 Baseline Methods

To contextualise our approach, we evaluate varied baseline methods: (1) ARFL, designed to tackle label and input noise in a federated learning setting [69]; (2) IDPA, a probabilistic method for instance-dependent label noise [158], which modifies the training loss; (3) “Co-teaching”, which trains two models simultaneously to perform sample selection based on loss values during training [157]; (4) CDR, which regularises model weights such that important parameters, learnt during the early training phase, do not considerably change when the model fits to the noisy data [71]; (5) Label Smoothing, which uses a combination of the true label and a uniformly weighted soft label during training [159]; (6) RRL, which uses contrastive learning and a k nearest neighbours search to enforce a smoothness constraint on learnt representations [128], modifying model architecture and training loss; and (7) an identical model but without any specific modifications for tackling noise.

Baseline methods were selected based on the availability of code, their use as baselines in the literature, and for variety in the methods used to evaluate the performance of LAP. Although our setting assumes noise is independent of features, we felt it was still beneficial to include IDPA, which is designed for instance-dependent label noise. Note that because ARFL is a federated learning approach, data cannot be shuffled in the same way as the other models, since each client trains on a single source.

It is important to note that both IDPA and Co-teaching require the training of a model twice, making LAP significantly less computationally expensive.

All baselines are implemented using the available code and trained using the recommended parameters

with the model and data we test.

5.5.2 Benchmark Datasets

In our evaluation, we use publicly available datasets that are commonly tested within the noisy data literature, allowing for a fair comparison with baselines.

As a reminder, the content of each of the datasets is as follows:

- **CIFAR-10** and **CIFAR-100** [92] are datasets made up of 60,000 RGB images of size 32×32 divided into 10 and 100 classes, with 6000 and 600 images per class, respectively. The task in this dataset is to predict the correct class of a given image.
- **F-MNIST** [95] is made up of 70,000 greyscale images of clothing of size 28×28 divided into 10 classes with 7000 images per class, which are flattened into observations with 784 features. The task of this dataset is to predict the correct class of a given image.
- **PTB-XL** [104] consists of 21,837 ECG recordings of 10 second length, sampled at 100Hz, from 18,885 patients, labelled by 12 nurses. The task of this dataset is to predict whether a patient has a normal or abnormal cardiac rhythm.
- **CIFAR-10N** [93] is a dataset made up of all CIFAR-10 examples, but with human labelling categorised into different levels of quality. For our case, to allow for the most flexibility in experiment design, we utilised the worst of the human labels.
- **GoEmotions** [96] is a natural language dataset in which the task is to correctly classify the emotion of Reddit comments from a possible 28 emotions. This dataset contains 171,820 text examples, annotated by 82 raters, with each rater contributing somewhere between 1 and 9320 labels (mean: 1676, standard deviation: 1477). To produce the training and test set, we randomly split the examples in the 80:20 ratio.
- **IMDB** [102] is a natural language dataset with a sentiment analysis task. The goal is to correctly classify a movie review as positive or negative based on its text. It contains 25,000 reviews in the training set and 25,000 reviews in the testing set, divided equally between positive and negative sentiment.

- **MNIST** [103] contains 60,000 training and 10,000 testing images of handwritten digits in black and white with a resolution of 28×28 . The goal of this dataset is to correctly identify the digit drawn in the image (from 0 to 9).
- **Tiny-ImageNet** [101] contains 110,000 RGB images of size 64×64 of 200 classes, with the goal of correctly classifying an image into the given class. There are 100,000 training images and 10,000 images in the test set.
- **ImageNet** [101] contains 1,281,167 RGB images scaled to size 64×64 from 1000 classes, with the goal of correctly classifying an image into the given class. There are 1,231,167 training images and 50,000 images in the test set.

CIFAR-10, CIFAR-100, and F-MNIST are managed under the MIT License. PTB-XL is made available with the Creative Commons Attribution 4.0 International Public License², CIFAR-10N is available with the Creative Commons Attribution-Non-Commercial 4.0 International Public License³, GoEmotions is made available under the Apache-2.0 license⁴, and MNIST is available under the the Creative Commons Attribution-Share-Alike 3.0 license⁵. IMDB⁶ is made publicly available, and Tiny-ImageNet and ImageNet are available after agreeing to the access terms⁷.

All datasets tested are publicly available and easily accessible. Additionally, within the supplementary code we use the default Pytorch Dataset objects for CIFAR-10, CIFAR-100, F-MNIST, and MNIST, and provide Pytorch Dataset objects that will automatically download, unzip, and load the data for PTB-XL, CIFAR-10N, GoEmotions, and IMDB. We additionally provide code to load Tiny-ImageNet and ImageNet from a local directory since it requires an agreement before accessing. This makes the reproduction of our work presented simple to perform.

5.5.3 Model Architectures

Within our experiments, we tested many model architectures, described below. Due to limitations in computation and to allow for our work to be easily reproducible, we use different levels of model capacity,

²<https://creativecommons.org/licenses/by/4.0/>.

³<https://creativecommons.org/licenses/by-nc/4.0/>.

⁴<https://github.com/google-research/google-research>.

⁵<https://creativecommons.org/licenses/by-sa/3.0/>

⁶<https://ai.stanford.edu/~amaas/data/sentiment/>

⁷<https://www.image-net.org/download>

which additionally illustrates that our method is applicable in varied settings. All models are also available in the supplementary code, implemented in Pytorch.

- **Low capacity Multilayer Perceptron (MLP):** The MLP used in Section 5.5 consists of 3 linear layers that map the input to dimension sizes of 16, 16, and the number of classes. In between these linear layers, we apply dropout with a probability of 0.2 and a ReLU activation function.
- **Low capacity CNN:** This model consisted of 3 convolutional blocks followed by 2 fully connected layers. Each convolutional block contained a convolutional layer with a kernel size of 3 with no padding, a stride and dilation of 1, a ReLU activation function, and a max pooling operation with a kernel size of 2. The linear layers following these convolutional blocks maps the output to a feature size of 64 and then the number of classes, with a ReLU activation function in between.
- **High capacity CNN with contrastive learning:** This CNN is inspired by the model used for the CIFAR-10 experiments in Li et al. (2021) [128] and requires considerably more compute cost. It is based on the ResNet architecture [236], except that it uses a pre-activation version of the original ResNet block. This block consists of a batch normalisation operation, a convolutional layer (of kernel size 3, padding of 1, and varied stride, without a bias term), another batch normalisation operation, and another convolutional layer (with the same attributes). Every alternate block contains a skip connection (after a convolutional layer, with a kernel size of 1 and a stride of 2, had been applied to the input). The input passes through a convolutional layer before the blocks. This is followed by a linear layer that maps the output of the convolutions to the number of classes. This model also contains a data reconstruction component that allows for an unsupervised training component.
- **ResNet 1D:** This model is designed for time-series classification and is based on the 2D ResNet model [236], except with 1D convolutions and pooling. The model is made up of 4 blocks containing two convolutional layers split by a batch normalisation operation, a ReLU activation function, and a dropout layer. These convolutional blocks reduce the resolution of their input by a quarter and increase the number of channels linearly by the number of input channels in the first layer of the model. Each block contains a skip connection that is added to the output of the block before passing through a batch normalisation operation, ReLU activation function and dropout layer. This is followed by a linear layer that transforms the output from the convolutional blocks to the number of output classes.

- **ResNet 2D:** This model is designed for image classification and is exactly the ResNet 20 architecture presented in He et al. (2016) [236] or the ResNet 18 or Resnet 50 implementation in available with Pytorch⁸.
- **Transformer Encoder [129]:** This model is designed for the natural language-based emotion prediction task given by the GoEmotions dataset. This model uses an embedding layer of size 256, positional encoding, and 2 transformer encoder layers (based on the implementation in Pytorch⁹) with 4 heads, and an embedding size of 256. This is followed by a linear layer that maps the output from the transformer encoder to the 28 emotion classes.
- **LSTM [237]:** The natural language model contains an embedding layer, which maps tokens to vectors of size 256, a predefined LSTM module from Pytorch¹⁰ with 2 layers and a hidden size of 512, a dropout layer with a probability of 0.25, and a fully connected layer mapping the output from the LSTM module to 2 classes.

5.5.4 Experimental Design

To evaluate LAP, we employ various techniques to produce noisy data, extending the methods in Li et al., Li et al., Han et al., Wang et al. (2022, 2021, 2018, 2021) [69, 128, 157, 158] and test on ten datasets from computer vision, healthcare time-series, and natural language processing for diverse experiments.

Following Li et al., Li et al., Han et al., Wang et al. (2022, 2021, 2018, 2021) [69, 128, 157, 158], we test our proposed method and baselines on CIFAR-10, CIFAR-100 [92], and F-MNIST [95] forming many comparisons with the literature. Along with Tiny-ImageNet and ImageNet [101], these five datasets form well-studied computer vision tasks, easing reproducibility. Additionally, we use a human-labelled version of CIFAR-10, titled CIFAR-10N [93], for which we use the “worst labels”, allowing us flexibility in our experiments. We then study an electrocardiograph (ECG) dataset, PTB-XL [104]; a time-series classification task with the goal of classifying normal and abnormal cardiac rhythms and for which noise can be simulated following Wong et al. (2012) [238] and random labelling to understand LAP applied to time-series data with multiple types of noise. Additionally, a natural language sentiment analysis task allows us to compare LAP with the literature, for which we use the IMDB data set [102] that contains movie reviews and their sentiment. Next, we use the GoEmotions dataset [96], a natural language emotion

⁸Pytorch:ResNet

⁹Pytorch:TransformerEncoderLayer

¹⁰Pytorch:LSTM

prediction task, which contains unbalanced source sizes and class distributions in the real world, allowing us to test the robustness of LAP to varied source constructions.

To simulate data sources for CIFAR-10, CIFAR-100, F-MNIST, ImageNet, and IMDB, data is uniformly split into 10 distinct groups, and for Tiny-ImageNet, we use 100 groups to study larger numbers of sources. For CIFAR-10 and CIFAR-100, 4 and 2 of these sources are chosen to be noisy respectively; for F-MNIST, ImageNet, and Tiny-ImageNet 6, 5 and 40 are chosen; and for IMDB 4 are chosen. These are in line with the noise rates used in the literature (often 20%, 40%, 50%). For CIFAR-10, in Section 5.5.13, we also significantly increase the number of sources. To generate noise for the vision datasets, we extended the methods in Li et al. (2022) [69]: (1) Original Data: No noise is applied to the data; (2) Chunk Shuffle: Split features into distinct chunks and shuffles. This is only done on the first and/or second axes of a given input; (3) Random Label: Randomly assign a new label from the same code; (4) Batch Label Shuffle: For a given batch of features, randomly shuffle the labels; (5) Batch Label Flip: For a given batch of data, assign all features in this batch a label randomly chosen from the same batch; (6) Added Noise: Add Gaussian noise to the features, with mean = 0 and standard deviation = 1; (7) Replace with Noise: Replace features with Gaussian noise, with mean = 0 and standard deviation = 1.

For IMDB and GoEmotions, we use random labelling.

PTB-XL was labelled by 12 nurses, naturally forming data sources. However, since this data is high quality, synthetic noise is required. We add Gaussian noise to sources' ECG recordings (simulating electromagnetic interference as in Wong et al. (2012) [238]) and label flipping to simulate human error in labelling. This also allows us to test the setting with multiple noise types. Here, data from sources are upsampled so that each source contains the same number of observations. For experiments with PTB-XL, we linearly increase the number of noisy sources from 1 to 8 (out of 12 in total), and for each number of noisy sources, we set the noise level for each source linearly from 0.25 to 1.0. For example, when training with 4 noisy sources, sources have noise levels of 0.25, 0.50, 0.75, and 1.0.

Since GoEmotions is annotated by 82 raters, their identification number is used to form the data sources. Some raters contributed a handful of data points, whilst others labelled thousands; with each rater providing a different distribution of class labels. This allowed us to explore the real-world case of data with sources of imbalanced size and label distribution.

Although CIFAR-10N contains real-world noise, we still have to split the data into sources. For each experiment, we assign sources to evaluate the varied levels of noise and the number of noisy sources. As

in PTB-XL, we linearly increase the number of noisy sources from 1 to 7 (out of 10 in total), and for each set of noisy sources, we linearly increase the noise level from 0.25 to 1.0.

For this, seven base model architectures (of multiple sizes) are evaluated: A Multilayer Perceptron, Convolutional Neural Networks, a 1D and 2D ResNet [236], an LSTM [237], and a transformer encoder [129].

In all experiments, data points contain an observation, source, and target, which are assigned to mini-batches in the ordinary way. Features and labels are passed to the model for training, whilst sources are used by LAP. The test sets contain only clean observations.

5.5.5 Intuition for the Hyperparameters

Figure 5.4 shows how the training process evolves over time when using LAP. At first, all sources are considered equally reliable (that is, $T_s = 1$, as previously discussed), and therefore the weighted mean of the loss values is the mean of all sources. As the source temperature changes, the weighed mean plus λ standard deviations of the losses moves toward the sources with lower expected losses, allowing more learning from these over time.

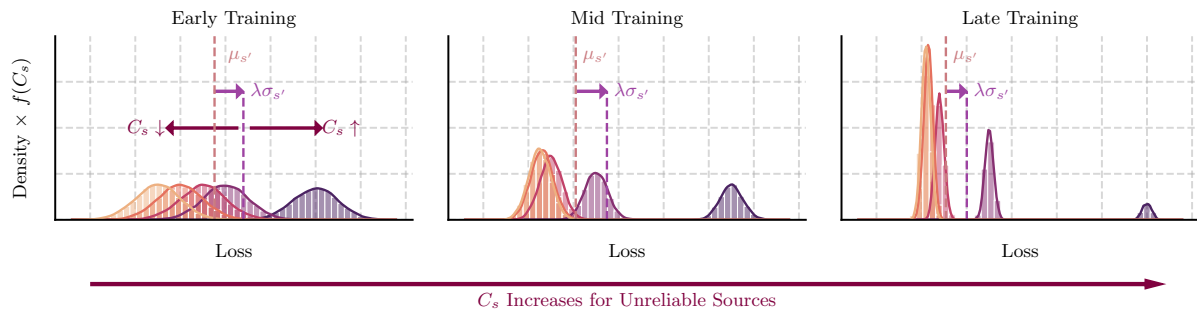


Figure 5.4: Visualisation of Equation 5.3. Each colour represents the loss values from a single source over a small number of steps, with its density weighted by its temperature, $w(s) = f(C_s)$. This shows how sources contribute to $\mu_{s'}$ and $\sigma_{s'}^2$ as their C_s changes during training and given the leniency λ . These values are synthetic and for demonstration.

Figure 5.5 illustrates how $1 - d_s$ (the temperature) in Equation 5.6 develops during training for each source and different values of H , δ , and λ .

The synthetic data is created using Scikit-Learn’s `make_moons`¹¹ function, which produced 10,000 synthetic observations with 2 features. Each observation is assigned a label based on which “moon” the observation corresponds to. Then, to produce synthetic sources and noise levels for each source, we

¹¹Scikit-Learn:make_moons

randomly assign each data point a source number from 0 to 4, so that we have 5 sources in total. The data points for each source were then made noisy by randomly flipping labels such that each source had a noise level of 0.0, 0.025, 0.05, 0.25, or 1.0.

A simple MLP with hidden sizes of (100, 100) and ReLU activation functions is trained on this data using the Adam optimiser with a learning rate of 0.01, a weight decay of 0.0001, and with $(\beta_1, \beta_2) = (0.9, 0.999)$ for 50 epochs with a batch size of 128. Data is shuffled at each epoch before being assigned to mini-batches. When conducting the experiments, the values of δ , H , and λ are varied as described in the graphs shown in Figure 5.5 to experiment with different values of these newly introduced parameters. Unless otherwise specified, the values chosen are $H = 25$, $\delta = 1.0$, $\lambda = 1.0$. These experiments take approximately 5 minutes to complete when using the compute described in Section 5.4.6.

The first row in Figure 5.5 shows how varying the depression strength, δ affects the rate at which the learning on a source is reduced. The second row shows how the history length, H , influences $1 - d_s$, where we can see that a larger value allows for a “smoother” transition. With the exception of small or large values, this parameter has the least effect on training. The third row illustrates how the leniency λ in Equation 5.3 and Algorithm 1 affects training. Small values of λ reduce the influence of noisy sources earlier, whilst larger values miss some noisy sources altogether. Here, the sources are coloured by their noise level, demonstrating that LAP reduced the influence of a source on training in a number of steps corresponding to the noise level of the source.

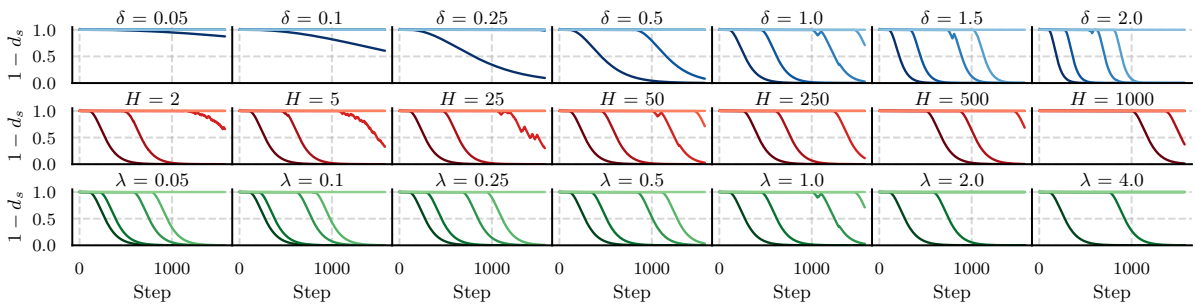


Figure 5.5: Effect of the introduced parameters on training. Section 5.4, introduces three parameters that control the effects of LAP. $1 - d_s$ is multiplied by the gradient (equivalently, loss) contribution from a given source before the model is updated. Here, we show these values for each source (the different coloured lines) during model training on synthetic data. Unless stated in the title of a given plot, the parameters of LAP were set to $H = 25$, $\delta = 1.0$, $\lambda = 1.0$. We had 5 sources with noise levels of 0.0, 0.025, 0.05, 0.25, and 1.0 (a darker colour indicates a higher noise rate).

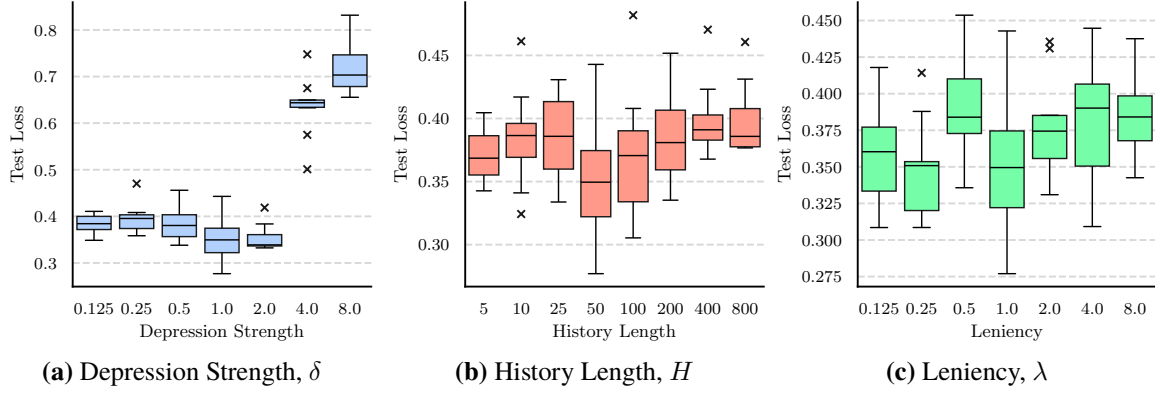


Figure 5.6: Sensitivity to Hyperparameters. Here, we demonstrate the sensitivity of the model performance based on the LAP hyperparameters introduced in Section 5.4 by varying their values whilst keeping the others fixed at $(\delta, H, \lambda) = (1.0, 50, 1.0)$.

5.5.6 Assessing the Sensitivity of LAP to these Hyperparameters

In Figure 5.6, we present the change in cross-entropy loss on the test set of the synthetic data used in Figure 5.5 and described in Section 5.5.5. In this experiment the depression strength δ , history length H , and leniency λ were kept at $(\delta, H, \lambda) = (1.0, 50, 1.0)$ unless otherwise specified.

Depression strength: We observe that significantly increasing the depression strength δ can lead to a noticeable increase in test loss, particularly when $\delta \geq 2.0$ (Figure 5.6a). This is because large values of δ force training contributions from noisy data sources to be reduced early in training (as shown in Figure 5.5), whilst they might still be useful for learning a robust model. Suppressing them too quickly might prevent the model from learning important patterns in the noisy data which are useful for predicting on the test set, leading to worse performance. This highlights the need to choose a depression strength δ that strikes the right balance between filtering noise and learning from all available data. We find that a value of $\delta = 1.0$ generally performs well.

History length: When studying the history length H , we see that the test loss remains relatively stable across the range of values (Figure 5.6b). There is some improvement for the history length values of $H = 50$ and $H = 100$, but there is little difference in performance between the use of large and small values of H . This indicates that whilst increasing H allows for LAP to consider more contextual information when calculating the source weighting, it also makes our method slower to react to rapid changes in the training loss due to the significantly larger history being considered. We find that a H value of 25 to 50 generally performs well.

Leniency: In Figure 5.6c, we present the test loss as we vary the value of leniency λ . The loss values for this parameter remain relatively stable across its different values, suggesting that the model is fairly robust to variations in leniency. It may not be a critical parameter for tuning in this particular set-up since for all of the leniency values tested in Figure 5.5, the two noisiest sources were heavily weighted during training. In scenarios where noise levels are very low, it might be necessary to reduce the leniency to capture them. In contrast, when it is believed that patterns can be learnt from noisy data, it might be beneficial to use a larger leniency value, which intuitively lengthens the amount of time before noisy sources are weighted (Figure 5.5). In general, we find that a value of $\lambda = 1.0$ performs well.

These experiments take approximately 5 minutes to complete when using the compute described in Section 5.4.6.

In our experiments, the parameters of LAP were set at $H = 25$, $\delta = 1.0$, $\lambda = 0.8$ unless otherwise stated. These were chosen based on the discussion here and the synthetic experiments presented in Figure 5.5. In practice, optimising these parameters for each experiment separately should allow improved performance over those presented in this work.

5.5.7 LAP and the Baselines

In this experiment, we used two different models applied to three different datasets, with 7 noise settings.

- **CIFAR-10 and CIFAR-100:** All training data is randomly split into 10 sources, with 4 and 2 sources chosen to be noisy for CIFAR-10 and CIFAR-100, respectively. In this experiment, these noisy sources are chosen to be 100% noisy so that we can understand how our method performs on data containing highly noisy sources. Noise is introduced based on the description given in Section 5.4. The ResNet 20 model is trained on this data and tasked with predicting the image class using cross-entropy loss. The model is trained for 40 epochs in both cases, with the SGD optimiser and a learning rate of 0.1, a momentum of 0.9, and a weight decay of 0.0001, in batches of size 128. When training with LAP, we use $H = 50$, $\delta = 1.0$, and $\lambda = 1.0$ for CIFAR-10 and $H = 25$, $\delta = 0.5$, and $\lambda = 1.0$ for CIFAR-100 chosen using the validation data (which is also noisy) after training on data with random label noise. When training on CIFAR-100, we found that using a warm-up of 100 steps improved performance.
- **F-MNIST:** All training data is randomly split into 10 sources, with 6 chosen to be 100% noisy. The

low capacity MLP model is trained for 40 epochs using the Adam optimiser, and a learning rate of 0.001 on batches of size 200. When training with LAP, we use $H = 25$, $\delta = 0.5$, and $\lambda = 0.5$ chosen using the validation data (which is also noisy) after training on data with random label noise.

To allow fairness in comparison with ARFL, global updates on this model were performed the same number of times as the number of epochs other models were trained for, with all clients being trained on data for a single epoch before each global update. All other parameters for the ARFL model are kept the same. Since ARFL is a federated learning method, each client is trained on sources separately. When IDPA was tested, the parameters were chosen as in Wang et al. (2021) [158] using the default parameters in the implementation. Finally, when testing “Co-teaching”, the parameters are chosen as in Han et al. (2018) [157] where possible, and scaled proportionally by the change in the number of epochs between their setting and our setting where they depended on the total number of epochs. The forgetting rate was set as the default given in the implementation code, 0.2, since we do not assume access to the true noise rate. For CDR, the parameters for CIFAR-10 and 100 were chosen as given in Xia et al. (2021) [71], and for F-MNIST, we used the same parameters as CIFAR-10. When testing Label Smoothing [159], we used a smoothing rate of 0.1 for CIFAR-10, CIFAR-100, and F-MNIST as we found that using negative smoothing had adverse effects on accuracy. These experiments take approximately 2 days to complete when using the compute described in Section 5.4.6.

Table 5.1 shows the results of the baselines and LAP on CIFAR-10, CIFAR-100 and F-MNIST with varied noise. This illustrates that in a high noise setting, LAP often allows for improved test accuracy over the baselines: four different approaches to learning from noisy labels (IDPA, Co-teaching, CDR, and Label Smoothing), an approach from federated learning (ARFL), and training a model in the standard way. Interestingly, IDPA and Co-teaching achieve a higher accuracy on the original data in CIFAR-10 and 100, likely because IDPA and Co-teaching require the training of a model twice, compared to LAP, ARFL and standard training.

On CIFAR-10 and 100, the performance improvement with LAP is most apparent with the random labelling noise. For example, on CIFAR-100, LAP achieves a +18.4% increase in top-5 accuracy over standard training, whilst IDPA shows poorer performance. Although the performance improvements of LAP are less dramatic on F-MNIST due to the simpler nature of the dataset, the model still demonstrates better accuracy in certain noise conditions, particularly with chunk shuffling and random label noise.

Table 5.2 shows the values presented in Table 5.1 as a percentage difference from the standard training

Table 5.1: Comparison of LAP with the baselines. Mean $\pm$ standard deviation of the maximum test accuracy (%) of 5 repeats of the baselines and LAP on synthetic data with different noisy types. For CIFAR-100 these numbers represent the top 5 accuracy. For CIFAR-10, CIFAR-100, and F-MNIST, the number of noisy sources are 4, 2, and 6 out of 10 respectively. Unreliable sources are each 100% noisy. All values in bold are within 1 standard deviation of the maximum score. Note that IDPA and Co-teaching require almost twice the training time in comparison to Standard, ARFL, CDR, Label Smoothing (LS), and LAP.

		Model Types						
	Noise Type	Standard	ARFL (2022)	IDPA (2021)	Co-teaching (2018)	CDR (2021)	LS (2022)	LAP (Ours)
CIFAR-10	Original Data	77.82 $\pm$ 0.37	74.89 $\pm$ 1.67	79.89 $\pm$ 1.01	80.04 $\pm$ 0.49	77.2 $\pm$ 1.59	78.47 $\pm$ 0.72	77.52 $\pm$ 0.9
	Chunk Shuffle	72.12 $\pm$ 0.95	68.42 $\pm$ 0.71	72.68 $\pm$ 0.96	74.77 $\pm$ 0.3	70.02 $\pm$ 1.35	70.1 $\pm$ 0.97	74.29 $\pm$ 1.43
	Random Label	67.84 $\pm$ 1.84	69.93 $\pm$ 1.88	55.66 $\pm$ 1.54	68.61 $\pm$ 1.78	65.61 $\pm$ 0.97	67.92 $\pm$ 2.66	73.44 $\pm$ 0.9
	Batch Label Shuffle	68.75 $\pm$ 1.36	67.12 $\pm$ 1.93	68.19 $\pm$ 1.48	71.68 $\pm$ 0.92	68.21 $\pm$ 1.06	67.35 $\pm$ 1.23	73.94 $\pm$ 0.33
	Batch Label Flip	67.12 $\pm$ 0.7	66.16 $\pm$ 1.96	69.65 $\pm$ 1.54	71.18 $\pm$ 0.76	66.59 $\pm$ 1.79	68.39 $\pm$ 2.14	73.85 $\pm$ 0.92
	Added Noise	70.23 $\pm$ 0.76	68.59 $\pm$ 1.38	70.94 $\pm$ 1.61	72.29 $\pm$ 0.61	68.3 $\pm$ 1.32	69.48 $\pm$ 1.33	72.63 $\pm$ 1.24
	Replace With Noise	72.79 $\pm$ 0.67	67.01 $\pm$ 1.02	71.96 $\pm$ 1.53	73.87 $\pm$ 0.57	71.02 $\pm$ 1.04	72.97 $\pm$ 0.83	73.84 $\pm$ 0.55
CIFAR-100	Original Data	75.95 $\pm$ 1.01	60.42 $\pm$ 1.91	77.78 $\pm$ 0.95	76.25 $\pm$ 0.54	75.91 $\pm$ 1.11	75.42 $\pm$ 0.49	76.01 $\pm$ 0.54
	Chunk Shuffle	68.86 $\pm$ 0.74	56.76 $\pm$ 2.69	66.82 $\pm$ 0.85	69.58 $\pm$ 0.51	68.65 $\pm$ 0.91	68.15 $\pm$ 0.95	70.41 $\pm$ 0.96
	Random Label	58.34 $\pm$ 0.91	48.85 $\pm$ 3.2	49.5 $\pm$ 1.03	61.2 $\pm$ 0.54	57.12 $\pm$ 0.76	58.3 $\pm$ 0.93	69.05 $\pm$ 0.38
	Batch Label Shuffle	64.8 $\pm$ 0.97	58.2 $\pm$ 4.45	64.34 $\pm$ 2.09	69.34 $\pm$ 0.69	65.69 $\pm$ 1.15	64.73 $\pm$ 1.12	69.31 $\pm$ 1.4
	Batch Label Flip	61.89 $\pm$ 0.97	56.51 $\pm$ 3.36	64.61 $\pm$ 2.1	69.04 $\pm$ 1.19	60.01 $\pm$ 1.16	60.64 $\pm$ 2.05	69.82 $\pm$ 0.78
	Added Noise	65.28 $\pm$ 0.77	58.35 $\pm$ 4.16	64.44 $\pm$ 1.11	66.02 $\pm$ 0.36	65.24 $\pm$ 1.32	65.38 $\pm$ 1.28	67.69 $\pm$ 1.17
	Replace With Noise	67.76 $\pm$ 1.33	58.88 $\pm$ 4.33	66.06 $\pm$ 0.42	68.52 $\pm$ 1.31	68.48 $\pm$ 1.31	67.58 $\pm$ 0.92	68.71 $\pm$ 1.22
F-MNIST	Original Data	83.64 $\pm$ 0.33	82.0 $\pm$ 0.4	83.69 $\pm$ 0.65	79.09 $\pm$ 1.14	82.42 $\pm$ 2.27	83.74 $\pm$ 0.27	83.52 $\pm$ 0.21
	Chunk Shuffle	77.12 $\pm$ 1.79	77.69 $\pm$ 0.51	77.45 $\pm$ 2.21	74.74 $\pm$ 1.21	76.45 $\pm$ 1.7	78.17 $\pm$ 0.75	82.18 $\pm$ 1.26
	Random Label	76.22 $\pm$ 8.06	77.31 $\pm$ 3.82	76.57 $\pm$ 7.16	77.41 $\pm$ 4.47	75.33 $\pm$ 7.27	74.79 $\pm$ 4.84	81.8 $\pm$ 1.37
	Batch Label Shuffle	82.65 $\pm$ 0.13	79.11 $\pm$ 1.71	82.82 $\pm$ 0.43	82.3 $\pm$ 0.32	82.88 $\pm$ 0.44	82.63 $\pm$ 0.45	82.45 $\pm$ 0.38
	Batch Label Flip	80.25 $\pm$ 1.87	78.85 $\pm$ 1.78	82.31 $\pm$ 0.27	80.43 $\pm$ 0.81	81.29 $\pm$ 0.58	80.56 $\pm$ 0.67	79.87 $\pm$ 0.7
	Added Noise	77.88 $\pm$ 1.86	73.28 $\pm$ 1.31	78.42 $\pm$ 2.1	75.72 $\pm$ 1.55	76.21 $\pm$ 2.37	79.02 $\pm$ 0.75	78.72 $\pm$ 2.02
	Replace With Noise	79.77 $\pm$ 0.8	74.12 $\pm$ 1.36	80.64 $\pm$ 0.42	78.76 $\pm$ 1.06	78.24 $\pm$ 0.93	79.94 $\pm$ 0.64	82.94 $\pm$ 0.31

method. This more clearly demonstrates the size of the accuracy improvement from using LAP on CIFAR-10, CIFAR-100, and F-MNIST.

In these experiments, LAP achieves the best accuracy or is within a few percentage points of the best accuracy. In the latter case, the best accuracy is then often achieved by a method that requires longer training but is less robust to other noise types. Therefore, LAP overall achieves consistently higher accuracy while requiring less computation. This illustrates LAP’s robustness across varied datasets and noise types, showing that it consistently maintains or improves accuracy in high-noise conditions where other models struggle.

5.5.8 LAP in Conjunction with RRL

Since RRL requires significant changes to the model architecture and data augmentation, we separately test the large architecture and the experimental setup used in Li et al. (2021) [128] for a better comparison.

All training data is randomly divided into 10 sources, with 4 chosen to be 100% noisy. The high capacity

Table 5.2: Comparison of LAP with the baselines. Mean $\pm$ standard deviation of the percentage difference of the maximum test accuracy (%) of 5 repeats of the baselines and LAP on synthetic data with different noisy types. For CIFAR-100 these numbers represent the top 5 accuracy. For CIFAR-10, CIFAR-100, and F-MNIST, the number of noisy sources are 4, 2, and 6 out of 10 respectively. Unreliable sources are each 100% noisy. All values in bold are within 1 standard deviation of the maximum score.

		Model Types						
	Noise Type	Standard	ARFL (2022)	IDPA (2021)	Co-teaching (2018)	CDR (2021)	LS (2022)	LAP (Ours)
CIFAR-10	Original Data	-	-3.76% $\pm$ 2.40	2.66% $\pm$ 1.53	2.86% $\pm$ 0.45	-0.79% $\pm$ 2.03	0.84% $\pm$ 1.23	-0.38% $\pm$ 1.53
	Chunk Shuffle	-	-5.12% $\pm$ 1.59	0.77% $\pm$ 0.49	3.68% $\pm$ 1.50	-2.89% $\pm$ 3.05	-2.79% $\pm$ 1.84	3.02% $\pm$ 2.22
	Random Label	-	3.10% $\pm$ 2.45	-17.90% $\pm$ 3.40	1.15% $\pm$ 1.43	-3.22% $\pm$ 3.44	0.15% $\pm$ 4.01	8.34% $\pm$ 3.93
	Batch Label Shuffle	-	-2.31% $\pm$ 3.95	-0.80% $\pm$ 1.68	4.29% $\pm$ 2.08	-0.76% $\pm$ 2.18	-2.00% $\pm$ 2.74	7.59% $\pm$ 2.57
	Batch Label Flip	-	-1.40% $\pm$ 3.67	3.77% $\pm$ 2.10	6.06% $\pm$ 1.05	-0.77% $\pm$ 3.30	1.90% $\pm$ 3.59	10.04% $\pm$ 0.85
	Added Noise	-	-2.33% $\pm$ 1.51	1.02% $\pm$ 2.23	2.94% $\pm$ 0.91	-2.72% $\pm$ 2.79	-1.05% $\pm$ 2.28	3.43% $\pm$ 2.41
	Replace With Noise	-	-7.92% $\pm$ 2.26	-1.12% $\pm$ 2.70	1.50% $\pm$ 0.42	-2.42% $\pm$ 1.50	0.26% $\pm$ 1.52	1.46% $\pm$ 1.63
CIFAR-100	Original Data	-	-20.42% $\pm$ 3.05	2.42% $\pm$ 0.77	0.41% $\pm$ 1.84	-0.02% $\pm$ 2.36	-0.67% $\pm$ 1.91	0.10% $\pm$ 1.91
	Chunk Shuffle	-	-17.59% $\pm$ 3.31	-2.95% $\pm$ 1.83	1.06% $\pm$ 1.15	-0.29% $\pm$ 1.84	-1.03% $\pm$ 1.33	2.26% $\pm$ 1.72
	Random Label	-	-16.20% $\pm$ 6.51	-15.13% $\pm$ 1.72	4.93% $\pm$ 1.96	-2.06% $\pm$ 1.89	-0.07% $\pm$ 0.90	18.40% $\pm$ 2.38
	Batch Label Shuffle	-	-10.15% $\pm$ 7.25	-0.68% $\pm$ 3.85	7.03% $\pm$ 1.49	1.38% $\pm$ 1.39	-0.08% $\pm$ 2.65	6.96% $\pm$ 0.93
	Batch Label Flip	-	-8.64% $\pm$ 6.29	4.41% $\pm$ 3.94	11.57% $\pm$ 2.15	-3.03% $\pm$ 2.58	-2.00% $\pm$ 3.63	12.84% $\pm$ 2.60
	Added Noise	-	-10.60% $\pm$ 6.71	-1.27% $\pm$ 2.35	1.14% $\pm$ 1.34	-0.06% $\pm$ 2.33	0.15% $\pm$ 1.18	3.70% $\pm$ 2.24
	Replace With Noise	-	-13.02% $\pm$ 7.46	-2.48% $\pm$ 2.26	1.12% $\pm$ 0.75	1.11% $\pm$ 3.49	-0.23% $\pm$ 2.76	1.46% $\pm$ 3.72
F-MNIST	Original Data	-	-1.95% $\pm$ 0.83	0.07% $\pm$ 0.71	-5.44% $\pm$ 1.34	-1.44% $\pm$ 2.87	0.13% $\pm$ 0.67	-0.13% $\pm$ 0.31
	Chunk Shuffle	-	0.78% $\pm$ 2.35	0.47% $\pm$ 3.93	-3.06% $\pm$ 1.88	-0.81% $\pm$ 4.05	1.39% $\pm$ 2.17	6.61% $\pm$ 3.46
	Random Label	-	2.28% $\pm$ 11.45	0.75% $\pm$ 6.82	2.79% $\pm$ 15.20	0.30% $\pm$ 18.72	-0.85% $\pm$ 13.32	8.21% $\pm$ 10.42
	Batch Label Shuffle	-	-4.29% $\pm$ 2.05	0.20% $\pm$ 0.57	-0.42% $\pm$ 0.29	0.28% $\pm$ 0.53	-0.02% $\pm$ 0.47	-0.24% $\pm$ 0.43
	Batch Label Flip	-	-1.67% $\pm$ 4.39	2.61% $\pm$ 2.31	0.26% $\pm$ 2.64	1.33% $\pm$ 2.29	0.43% $\pm$ 2.28	-0.43% $\pm$ 2.86
	Added Noise	-	-5.86% $\pm$ 3.02	0.77% $\pm$ 4.43	-2.71% $\pm$ 3.78	-2.09% $\pm$ 4.25	1.53% $\pm$ 3.13	1.14% $\pm$ 3.75
	Replace With Noise	-	-7.07% $\pm$ 1.66	1.10% $\pm$ 1.08	-1.26% $\pm$ 1.20	-1.91% $\pm$ 1.20	0.22% $\pm$ 0.57	3.99% $\pm$ 1.05

CNN with contrastive learning presented in Li et al. (2021) [128] is trained for 25 epochs, with all other parameters kept as in the original work. In addition to this model, a version (with the same parameters) is trained using LAP with $H = 25$, $\delta = 1.0$, and $\lambda = 0.8$. These models were trained on batch sizes of 128 using stochastic gradient descent with a learning rate of 0.02, momentum of 0.9, and weight decay 0.0005. Data is randomly assigned to mini-batches, with each batch containing multiple sources. These experiments take approximately 3 days to complete when using the compute described in Section 5.4.6.

We test RRL with and without LAP to illustrate its ability to be used in conjunction with other methods. Table 5.3 presents these results on CIFAR-10, and further highlights the use of LAP to improve model performance on data generated from multiple sources. Here, LAP supplements the accuracy obtained by RRL, which uses a contrastive learning approach to tackle noisy features and labels that is improved upon by further applying our method to make use of the source values. We observe that LAP increases all metric scores except for “Replace With Noise”, where the difference is relatively small, and “Original Data”, where both methods perform equally. Again, the most substantial increase in accuracy comes from random labelling noise, where LAP improves RRL by approximately +5%.

In addition, Table 5.4 shows the percentage difference in accuracy when using LAP over not using LAP for

Table 5.3: RRL and RRL + LAP results. Mean $\pm$ standard deviation of the maximum test accuracy (%) of 10 repeats of RRL and LAP on CIFAR-10 data with different types of noise. Here, 4 out of 10 sources are 100% noisy.

Noise Type	Model Types	
	RRL (2021)	RRL + LAP (Ours)
Original Data	87.67 $\pm$ 0.37	87.54 $\pm$ 0.22
Chunk Shuffle	82.93 $\pm$ 0.29	84.27 $\pm$ 0.31
Random Label	76.04 $\pm$ 1.43	80.31 $\pm$ 0.58
Batch Label Shuffle	77.66 $\pm$ 0.71	80.84 $\pm$ 0.51
Batch Label Flip	78.81 $\pm$ 0.66	82.02 $\pm$ 0.45
Added Noise	78.51 $\pm$ 0.74	81.70 $\pm$ 0.48
Replace With Noise	80.05 $\pm$ 0.65	79.00 $\pm$ 0.75

several different noise levels and numbers of noisy sources. In these experiments, the data in CIFAR-10 is randomly divided into 10 sources, the row of Table 5.4 defining the number of noisy sources, of which all have a noise rate as given by the column of Table 5.4. As before, the high capacity CNN with contrastive learning presented in Li et al. (2021) [128] is trained for 25 epochs, with all other parameters kept as in the original work. In addition, a version of this model (with the same parameters) is trained using LAP with $H = 25$, $\delta = 1.0$, and $\lambda = 0.8$. These models are trained on batch sizes of 128 using stochastic gradient descent with a learning rate of 0.02, momentum of 0.9, and weight decay 0.0005. Data is randomly assigned to mini-batches, with each batch containing multiple sources. These experiments take approximately 5 days to complete when using the compute described in Section 5.4.6.

Table 5.4: Different noise level and number of sources. Mean $\pm$ standard deviation (%) of the *percentage difference* in maximum test accuracy between RRL + LAP and RRL (2021) over 5 repeats when training a model on CIFAR-10 for different noise levels and numbers of sources. Here a positive value represents an improvement in accuracy when using LAP. Here, U corresponds to the number of unreliable sources out of 10 sources in total. Batch label flipping was used to introduce noise.

U	Noise Level			
	25%	50%	75%	100%
2	0.88% $\pm$ 0.19	1.61% $\pm$ 0.60	2.95% $\pm$ 0.64	3.16% $\pm$ 0.51
4	0.09% $\pm$ 0.71	1.25% $\pm$ 0.58	2.01% $\pm$ 0.96	5.24% $\pm$ 1.24
6	0.02% $\pm$ 0.43	0.46% $\pm$ 0.50	3.74% $\pm$ 0.80	11.61% $\pm$ 1.66

When the whole dataset noise level is small (2 sources with 25% noise), we see small performance increases when using LAP; likely because noisy sources have small impacts on the performance of models trained without LAP. Furthermore, for lower noise levels, increasing the number of noisy sources reduced the performance improvement. This is likely because LAP is reducing noisy source training contributions early, when there is still information to learn. This can be remedied by increasing the leniency (λ); however, since we were limited by compute cost, the parameters of LAP were fixed across the experiments.

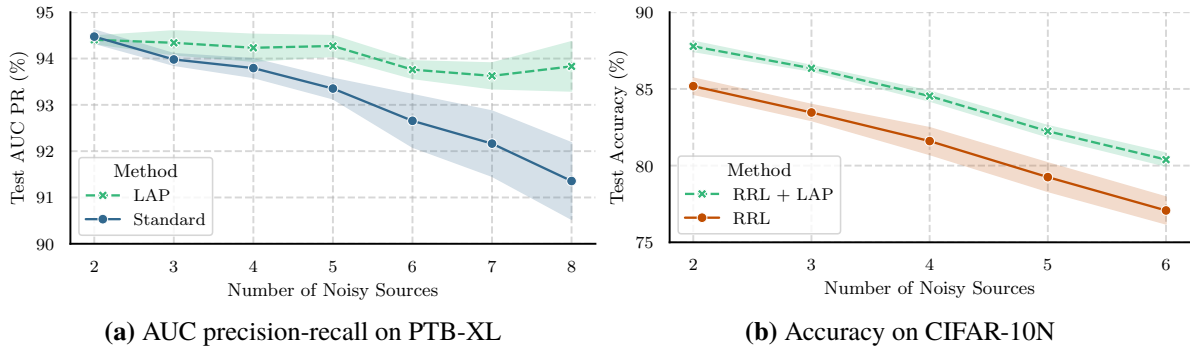


Figure 5.7: LAP results with a varied number of sources and noise levels. In (a) we show the area under the precision-recall curve for standard training and using LAP on PTB-XL with label noise and simulated ECG interference noise for 12 total sources. In (b), we show the accuracy on CIFAR-10N with real human labelling noise when using RRL and RRL + LAP, with 10 sources. In both, the noise of the sources varies linearly from 25% to 100% for each number of noisy sources. The lines and error bands represent the mean and standard deviation of the maximum value for each of the 5 repeats. These figures illustrate that LAP maintains higher performance as noise rates increase.

5.5.9 Varied Noise Level and Source Sizes with Real-World Time-Series Data

Figure 5.7a compares the results of not using and using LAP for a time-series classification task on normal and abnormal cardiac rhythms with both label and input noise.

This experiment allows us to test the performance of using LAP on different numbers of sources with varied noise levels. Firstly, data is divided into 12 sources based on the clinician performing the labelling of the ECG recording. Then, for a given number of sources (increasing along the x axis), the noisy sources have noise levels that are set at linear intervals between 25% and 100%. These sources are made noisy following the suggestions by Wong et al. (2012) [238] to simulate electromagnetic interference and using label flipping to simulate human error in labelling. Based on this data, we train the ResNet 1D model for 40 epochs using the Adam optimiser with a learning rate of 0.001 and a batch size of 64. This model is trained with and without LAP with $H = 25$, $\delta = 1.0$, and $\lambda = 0.8$. Data is randomly assigned to mini-batches, with each batch containing multiple sources. These experiments take approximately 16 hours to complete when using the compute described in Section 5.4.6.

Figure 5.7a shows the Area Under the Precision-Recall Curve (AUC PR) on the test set for each number of noisy sources. These results show that LAP allows for improved model training on data with sources of varied noise and with multiple noise types. This figure also illustrates that in this setting, LAP is robust to increases in noise within the dataset, as AUC PR does not degrade significantly, especially when compared to not using LAP. Furthermore, the standard deviation of the results is smaller, suggesting that

LAP is more consistent in its AUC PR. When only two sources contain noise, the total noise rate in the dataset is only 10%, which has little negative effect on standard training.

5.5.10 Real-World Noisy Labels

To further demonstrate the effectiveness of LAP in conjunction with other noisy data methods and on real-world noise, we now present the results on a dataset with human labelling of varied noise rates.

Here, images from CIFAR-10 are used as the inputs and the labelling collected by Wei et al. (2022) [93] is used as the noisy labels to produce CIFAR-10N. The data is first randomly split into 10 sources. Then, for a given number of sources (increasing along the x axis), the noisy sources have noise levels that are set at linear intervals between 25% and 100%, replacing the true CIFAR-10 targets with the real noisy human labels from Wei et al. (2022) [93]. The high capacity CNN with contrastive learning presented in Li et al. (2021) [128] is trained for 25 epochs, with all other parameters kept as in the original work. As before, a version of this model (with the same parameters) is trained using LAP with $H = 25$, $\delta = 1.0$, and $\lambda = 0.8$. These models are trained on batch sizes of 128 using stochastic gradient descent with a learning rate of 0.02, momentum of 0.9, and weight decay 0.0005. Data is randomly assigned to mini-batches, with each batch containing multiple sources. These experiments take approximately 30 hours to complete when using the compute described in Section 5.4.6.

Figure 5.7b illustrates the results on CIFAR-10N, a noisy human-labelled version of CIFAR-10, where we observe a pattern similar to that depicted in Figure 5.7a. In both cases, LAP improved the test accuracy after training on noisy data at all noise levels tested, and the standard deviation of this accuracy was reduced. However, we also note that in this case, the accuracy of LAP degraded faster than the AUC PR in Figure 5.7a as the number of noisy sources increased. This could be for various reasons, such as differing noise types, model baselines, and data types.

Additionally, we also show the results of this experiment when using the same model architecture as in Table 5.1, and with standard training as a baseline, instead of RRL. Figure 5.8 illustrates that the results of this experiment follow the same trends as before, which assures us of the applicability of LAP in a variety of settings, across model capacities. These experiments take approximately 4 hours to complete when using the compute described in Section 5.4.6.

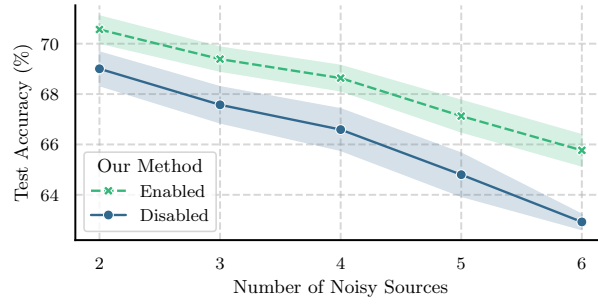


Figure 5.8: Accuracy values on CIFAR-10N for an increasing number of noisy sources with a lower capacity model. The lines and error bands represent the mean and standard deviation of the maximum test accuracy for each of the 5 repeats with random allocation of noisy sources. The noise of the sources increases linearly from 25% to 100% for each number of noisy sources. In total, there are 10 sources. Here, we test the low capacity CNN (Section 5.5.4).

5.5.11 LAP on a Noisy Natural Language Task

We also tested LAP on a natural language task, in which the goal is to predict the sentiment from the review of the movie.

In this set of experiments, we wish to test the use of LAP in a natural language setting. Firstly, we load the training and testing sets and randomly split the training set into 10 sources uniformly. We then choose 4 of the sources to be 100% unreliable and introduce noise through random labelling and random permuting of the order of the text. We truncate or extend each review such that it contains exactly 256 tokens. We use an LSTM to predict the sentiment of movie reviews by training the model for 40 epochs with a batch size of 128, a learning rate of 0.001, and the Adam optimiser [239]. A version of this model (with the same parameters) is also trained using LAP with $H = 25$, $\delta = 1.0$, and $\lambda = 0.8$. As before, data is randomly assigned to mini-batches, with each batch containing multiple sources. Since IDPA, Co-teaching, CDR, and Label Smoothing (LS) are not benchmarked on this dataset, for the baseline-specific parameters we used the same values as given for CIFAR-10. These experiments take approximately 12 hours to complete when using the compute described in Section 5.4.6.

The results with two types of noise are presented in Table 5.5. Here, LAP clearly outperforms the baselines by a significant margin, with IDPA and CDR not improving standard training. Additionally, we expect that when the data is not noisy, using LAP should not limit performance, as we see here. However, it is interesting that Co-teaching performs better in the “Original Data”. We hypothesise that the IMDB dataset contains some noisy labels (through human error), which were split uniformly across data sources; a limitation of our method that we discuss in Chapter 8.

Table 5.5: Baselines and LAP results on a natural language task. Mean $\pm$ standard deviation of maximum test accuracy (%) of 5 repeats of the methods on the IMDB dataset with different types of noise. Here, 4 out of 10 sources are 100% noisy.

Noise Type	Model Types					
	Standard	IDPA (2021)	Co-teaching (2018)	CDR (2021)	LS (2022)	LAP (Ours)
Original Data	82.81 $\pm$ 0.79	83.24 $\pm$ 0.56	85.12 $\pm$ 0.8	82.49 $\pm$ 1.49	83.89 $\pm$ 0.91	83.2 $\pm$ 0.83
Random Label	65.01 $\pm$ 0.65	64.61 $\pm$ 0.71	67.18 $\pm$ 0.69	64.8 $\pm$ 0.91	66.34 $\pm$ 1.13	71.95 $\pm$ 2.94

5.5.12 Real-world and Imbalanced Data Sources

To supplement the results of PTB-XL, we present LAP on an additional dataset containing real-world data sources, GoEmotions. This dataset allows us to explore the scenario in which source sizes and the classes they contain vary significantly, which could be a common scenario in real-world use cases. As LAP weights sources based on their log-likelihood, it is important to consider the robustness of our method to variations in the distribution of classes in sources, as well as their sizes. In the GoEmotions dataset, our training set contains source sizes in the range of 1 to 9320 with a mean size of 1676 and a standard deviation of 1477.

In Figure 5.9, we show the distribution of source sizes and the number of classes within each source. Figure 5.9a demonstrates the imbalance in the size of each source, with some sources containing thousands of data points, whilst others contain hundreds or tens of data points. A dataset with this construction allows us to test the robustness of LAP to settings with uneven source sizes, a realistic situation in real-world data collection. Figure 5.9b shows the distribution of the number of unique classes contained within each source.

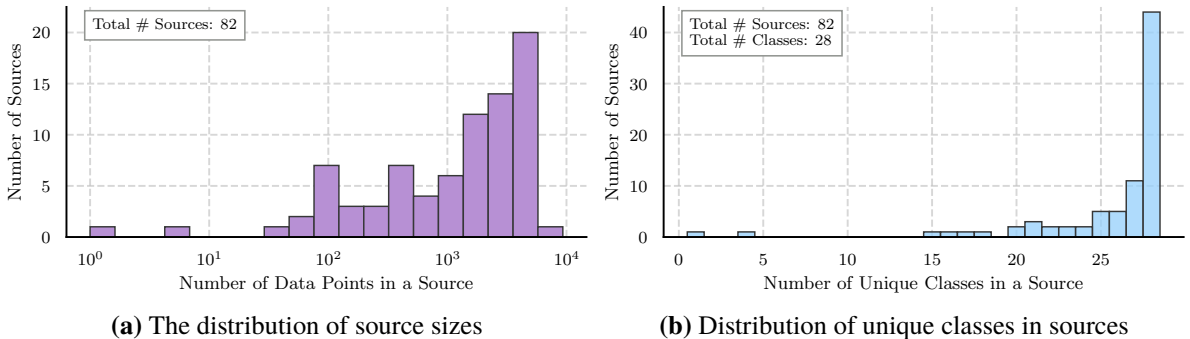


Figure 5.9: Source and class distributions in GoEmotions. In (a), we present the number of sources with a given size, plotted on a log scale. This demonstrates the imbalance in the size of the sources in the dataset, with some containing thousands of data points, whilst others contain just a few hundred. In (b), we additionally explore the number of classes within each data source, showing that around half of all sources do not contain all classes, with two sources containing less than 5 classes.

In particular, we observe that around half of all sources do not contain all the classes, with two sources containing fewer than 5 classes. The imbalance in the class distribution across sources is more apparent when we list the largest three classes for some of the sources:

- Source 1 with size 37: class 28 = 40.54%, class 19 = 8.11%, class 26 = 5.41%
- Source 2 with size 1137: class 28 = 38.43%, class 19 = 10.82%, class 2 = 5.89%
- Source 4 with size 2702: class 5 = 17.84%, class 28 = 11.84%, class 11 = 8.62%
- Source 8 with size 1253: class 28 = 14.45%, class 23 = 12.05%, class 21 = 10.45%
- Source 16 with size 3224: class 5 = 13.43%, class 23 = 12.38%, class 1 = 10.92%
- Source 32 with size 175: class 28 = 50.86%, class 8 = 18.86%, class 2 = 6.86%
- Source 64 with size 623: class 28 = 18.46%, class 4 = 11.88%, class 5 = 6.58%

This demonstrates that sources contain large variations in the number of data points and the distributions of classes they contain. By evaluating our method on this dataset, we can verify LAP’s improved performance in settings in which the class distribution is uneven across sources, which may affect the source log-likelihood during training and strain the assumption we made that non-noisy sources contain similar data distributions.

For our experiment, we chose 30 of the total 82 raters to produce noisy labels. The Transformer Encoder was trained for 25 epochs with a batch size of 256 and a learning rate of 0.001 using the Adam optimiser [239]. A version of this model (with the same parameters) is also trained using LAP with $H = 25$, $\delta = 1.0$, and $\lambda = 0.8$. As before, data is randomly assigned to mini-batches, with each batch containing multiple sources. Since IDPA, Co-teaching, CDR, and Label Smoothing are not benchmarked on this dataset, for the baseline-specific parameters, we used the same values as given for CIFAR-10. These experiments take approximately 12 hours to complete when using the compute described in Section 5.4.6.

Table 5.6 demonstrates the top-5 accuracy of the methods for this task, and suggests that LAP is robust to varied source data distributions in this real-world dataset. Whilst LAP, CDR, Label Smoothing (LS) and standard training perform similarly when trained on the original data, using LAP leads to improved top-5 accuracy on the test set when random labels are introduced in training. On GoEmotions, random labelling had small effects on the performance of the standard training method, reducing the top-5 accuracy of the

test set from 80.4 to 77.0. However, LAP was still able to significantly reduce that performance loss by achieving a top-5 accuracy of the test set of 78.7. Here, LAP produced a higher top-5 accuracy than the baselines, with less variability in the results.

Table 5.6: Results on GoEmotions, an imbalanced sources dataset. Mean $\pm$ standard deviation of maximum test top-5 accuracy (%) over 5 repeats of the methods on the GoEmotions data with noisy labels and random sentence permutation. Here, 30 out of 82 sources are 100% noisy.

Noise Type	Model Types					
	Standard	IDPA (2021)	Co-teaching (2018)	CDR (2021)	LS (2022)	LAP (Ours)
Original Data	80.43 $\pm$ 0.05	79.66 $\pm$ 0.21	79.7 $\pm$ 0.31	80.37 $\pm$ 0.1	80.35 $\pm$ 0.07	80.44 $\pm$ 0.15
Random Label	76.96 $\pm$ 0.58	76.05 $\pm$ 0.84	77.05 $\pm$ 0.7	76.05 $\pm$ 0.31	76.02 $\pm$ 0.57	78.74 $\pm$ 0.41

5.5.13 CIFAR-10 with a Large Numbers of Sources

To understand the effectiveness of our method as the number of sources grows, we extended the results in Table 5.1 by significantly increasing the number of unique sources. Here, we use the low capacity CNN, with all other settings kept the same as in Table 5.1, except with the number of epochs reduced from 40 to 25 as the model architecture is smaller. These experiments take approximately 12 hours to complete when using the compute described in Section 5.4.6.

Table 5.7: Comparison of LAP with the baselines with varied numbers of sources. Mean $\pm$ standard deviation of maximum test accuracy (%) of 5 repeats of the baselines and LAP on synthetic data with different noisy types and numbers of sources. The number of noisy sources ranges from 10 to 50,000, corresponding to “LAP- n ”, where n denotes the number of sources in the training data. For all experiments, 40% of the sources were chosen as unreliable, with 100% noise rate. Since there are 50,000 training data points, LAP-50,000 corresponds to LAP acting over sources that have size 1: i.e the standard noisy data setting.

Noise Type	Model Types							
	Standard	IDPA (2021)	Co-teaching (2018)	CDR (2021)	LS (2022)	LAP-10	LAP-1250	LAP-50,000
Original Data	69.46 $\pm$ 1.12	70.86 $\pm$ 0.32	67.73 $\pm$ 0.83	70.88 $\pm$ 0.51	70.2 $\pm$ 0.66	69.6 $\pm$ 1.08	69.57 $\pm$ 1.07	70.28 $\pm$ 0.5
Chunk Shuffle	65.92 $\pm$ 1.32	66.34 $\pm$ 0.99	63.55 $\pm$ 0.41	66.18 $\pm$ 0.81	66.22 $\pm$ 0.84	66.92 $\pm$ 0.56	66.08 $\pm$ 1.02	65.47 $\pm$ 0.98
Random Label	60.85 $\pm$ 1.67	57.93 $\pm$ 1.59	61.37 $\pm$ 1.01	61.23 $\pm$ 1.16	61.1 $\pm$ 1.25	66.5 $\pm$ 0.43	66.15 $\pm$ 1.16	66.49 $\pm$ 0.65
Batch Label Shuffle	62.23 $\pm$ 0.62	60.08 $\pm$ 0.13	63.9 $\pm$ 0.44	64.63 $\pm$ 0.88	64.36 $\pm$ 1.02	66.09 $\pm$ 0.65	66.71 $\pm$ 0.48	65.73 $\pm$ 1.13
Batch Label Flip	59.94 $\pm$ 2.48	61.02 $\pm$ 1.9	63.05 $\pm$ 2.43	61.57 $\pm$ 1.06	61.84 $\pm$ 0.92	64.48 $\pm$ 1.11	66.42 $\pm$ 1.22	65.41 $\pm$ 1.65
Added Noise	59.86 $\pm$ 1.38	61.19 $\pm$ 0.33	59.59 $\pm$ 1.37	60.38 $\pm$ 0.84	60.05 $\pm$ 1.05	63.58 $\pm$ 1.03	59.52 $\pm$ 1.06	60.55 $\pm$ 1.08
Replace With Noise	65.07 $\pm$ 0.63	65.85 $\pm$ 0.64	64.1 $\pm$ 1.13	65.91 $\pm$ 1.03	65.7 $\pm$ 0.78	64.89 $\pm$ 1.28	66.39 $\pm$ 1.66	65.09 $\pm$ 0.43

The results of this is shown in Table 5.7. In this experiment, we see that LAP performs well across the different source sizes, demonstrating its robustness as the number of sources increases. In fact, LAP-50,000 corresponds to an experiment where the size of each source is equal to 1 – the standard noisy data setting. It is interesting to see that in this case, LAP often performs as well or better than the baselines,

suggesting its usefulness in a setting for which it was not originally designed.

5.5.14 Testing a Possible Failure Case for our Method

To demonstrate the robustness of LAP, we now construct an experiment designed to strain the assumptions we made when introducing our method.

To work as intended, our proposed method assumes that all non-noisy sources contain similar data distributions. This allows us to say that the weighted likelihood ratio between a source under inspection and the other sources indicates whether to increase or decrease our reliability score (Equation 5.3). However, in real-world use cases, some sources might only produce a single class or a more challenging subset of classes. In such a case, we would like to understand whether our method can correctly identify the noisy sources without mistakenly labelling the more challenging data source as noisy.

To construct this setting, we combined data from both MNIST [103] and CIFAR-10 [92]. First, we evenly split the MNIST data into 99 sources and applied random label noise to 95 of them. These numbers of sources were chosen to create an extreme setting with many noisy sources, which we hypothesised would be a challenging setting to learn in. We then randomly chose 2 classes from CIFAR-10 and assigned them to a single source, giving us 99 MNIST sources (with 95 of them noisy) and a single CIFAR-10 source (which remained non-noisy). We then kept the same non-noisy class labelling for MNIST from the original dataset (i.e.: in the non-noisy data, a handwritten digit 0 was assigned class 0, etc.) and mapped the two classes from CIFAR-10 to new classes: 10 and 11 – giving 12 classes in total. Since CIFAR-10 is more challenging to classify than MNIST, this allowed us to construct a dataset in which almost all sources are easier to classify than the final source, which contains 2 classes of more difficult data to separate. We also ran the same experiment with no random label noise applied to the MNIST sources.

To ensure that the CIFAR-10 and MNIST images are the same size and have the same number of filters, we applied a greyscale and resizing transformation to the CIFAR-10 data. We then use a simple CNN consisting of two convolutional layers with 32 and 64 filters, respectively, ReLU activation functions, max-pooling layers (with kernel size 2), and two linear layers mapping representations to sizes 9216, 128, and finally 13. We use dropout, cross-entropy loss, and the Adam optimiser with learning rate 0.001 and batches of size 128, trained for 25 epochs. The LAP parameters used were $(\delta, H, \lambda) = (1.0, 50, 4.0)$ for the original data experiment and $(0.1, 50, 1.0)$ for the random label experiment. These were chosen using 5 runs on a validation set.

Table 5.8: Difficult data results. Mean $\pm$ standard deviation of maximum test accuracy (%) over 5 repeats of standard training and LAP on a combination of MNIST and CIFAR-10 data with noisy labels. Here, 95 out of 100 sources are 100% noisy.

Noise Type	Model Types	
	Standard	LAP (Ours)
Original Data	98.38 $\pm$ 0.4	98.36 $\pm$ 0.39
Random Label	67.43 $\pm$ 2.83	96.29 $\pm$ 0.83

Table 5.8 shows the surprising results of this experiment. Since this data is made up of significant noise, it causes a large degradation in the accuracy of a standard training model when random labelling is applied to the data. However, LAP shows significantly higher performance on the test set than the standard training method, while allowing for comparative accuracy in the absence of noise. Here, we observe that even in a contrived experiment designed as a failure case for our proposed method, LAP still produces increased accuracy over this baseline.

Table 5.9: Difficult data results on the CIFAR-10 classes. Mean $\pm$ standard deviation of maximum test accuracy (%) on the CIFAR-10 classes over 5 repeats of standard training and LAP trained on a combination of MNIST and CIFAR-10 data with noisy labels. This is the accuracy on just the CIFAR-10 test set, which is a subset of the test set corresponding to Table 5.8.

Noise Type	Model Types	
	Standard	LAP (Ours)
Original Data	0.97 $\pm$ 0.01	0.95 $\pm$ 0.06
Random Label	0.98 $\pm$ 0.01	0.97 $\pm$ 0.02

Furthermore, Table 5.9 shows that we do see a small reduction in performance on the more challenging source (containing the CIFAR-10 data). On the original data, in all five runs, LAP labelled the source containing the CIFAR-10 data as non-noisy as desired. However, when random label noise is applied to MNIST sources, the source containing CIFAR-10 data was incorrectly considered noisy in two of the five runs. In these two cases, the validation accuracy was lower than in the other three runs:

- CIFAR-10 was incorrectly considered noisy: 23.50% and 24.39%.
- CIFAR-10 was correctly considered non-noisy: 25.46%, 25.19%, 25.53%.

Note that the validation accuracy is low because it contains the noisy labels as well as the clean labels. Therefore, with more runs and an ensemble of the higher validation accuracy runs, higher accuracy would likely be achieved on the test set and, in particular, on the CIFAR-10 source.

In the extreme case presented here, which violates the assumptions we made during Section 5.4, we find

that our method is robust to challenging source data distributions and achieves markedly greater accuracy on the test set under noisy learning, whilst achieving comparative accuracy on the single challenging (but non-noisy) CIFAR-10 source.

5.5.15 Tiny-ImageNet Results

In an effort to demonstrate the potential of LAP further, we evaluated our method on Tiny-ImageNet [101], a subset of ImageNet that contains 100,000 images from 200 classes. In this experiment, we wanted to test the use of LAP on larger images with larger numbers of sources. Firstly, we load the training and testing sets and randomly split the training set into 100 sources uniformly. We then choose 40 of the sources to be 100% unreliable and introduce noise through random labelling. The dataset is then split into the ratio 9 : 1 to produce a validation set. For this experiment, we train a ResNet 50 architecture using the baseline training method provided for ImageNet in Pytorch¹². This trains the model for 90 epochs with a batch size of 256 using stochastic gradient descent with an initial learning rate of 0.1, momentum of 0.9, and weight decay of 0.0001, as well as a learning rate scheduler that multiplies the learning rate by 0.1 every 30 epochs. A version of this model (with the same parameters) is also trained using LAP with $H = 25$, $\delta = 1.0$, and $\lambda = 0.8$. As before, data is randomly assigned to mini-batches, with each batch containing multiple sources.

The results of this are given in Table 5.10 and again demonstrate the expected performance improvement when using LAP for datasets with sources of unknown noise, and the maintenance of performance when data is non-noisy. These experiments take approximately 14 hours to complete when using the compute described in Section 5.4.6.

Table 5.10: Tiny Imagenet results. Mean $\pm$ standard deviation of maximum test top-5 accuracy (%) over 5 repeats of standard training and LAP on Tiny Imagenet data with noisy labels. Here, 40 out of 100 sources are 100% noisy.

Noise Type	Model Types	
	Standard	LAP (Ours)
Original Data	61.32 $\pm$ 0.61	60.62 $\pm$ 0.71
Random Label	46.27 $\pm$ 1.04	54.48 $\pm$ 0.57

¹²<https://pytorch.org/blog/how-to-train-state-of-the-art-models-using-torchvision-latest-primitives/>

5.5.16 ImageNet Results with Multiple Noise Types

We additionally present our results on ImageNet, which allows us to test our method on a large-scale dataset. Within this experiment, as in that presented in Figure 5.7a, we allow both input and label noise. In Figure 5.7a, all noisy sources contain two types of noise, however in the experiment presented in Table 5.11, we allow different sources to contain different types of noise.

For this experiment, we train a ResNet 50 architecture using the baseline training method provided for ImageNet in Pytorch¹². This trains the model for 90 epochs with a batch size of 256 using stochastic gradient descent with an initial learning rate of 0.1, momentum of 0.9, and weight decay of 0.0001, as well as a learning rate scheduler that multiplies the learning rate by 0.1 every 30 epochs. We divide the ImageNet data into 10 sources, of which 5 are chosen to be noisy (2 containing input noise and 3 containing label noise). The images are loaded as 8-bit integer arrays and interpolated to (64, 64) before the input noise is synthesised by randomly adding uniform integers from $[-64, 64]$. They are then transformed to 32-bit floats and normalised using the mean and standard deviation available on the same Pytorch training script. Label noise is added by randomly replacing labels in noisy sources. This dataset is then split into the ratio 9 : 1 to produce a validation set for the analysis in Section 5.5.16. A version of the ResNet 50 model (with the same parameters) is also trained using LAP with $H = 25$, $\delta = 1.0$ and $\lambda = 0.8$. As before, data is randomly assigned to mini-batches, with each batch containing multiple sources. These experiments are repeated 5 times for each setting. These experiments take approximately 32 hours to complete when using the compute described in Section 5.4.6.

These results further illustrate that the use of LAP improves model performance when training data is noisy since it achieves a greater maximum top-5 test accuracy than the standard training method.

Table 5.11: Imagenet results. Mean $\pm$ standard deviation of maximum test top-5 accuracy (%) over 5 repeats of standard training and LAP on Imagenet data with noisy labels and noisy inputs. Here, 5 out of 10 sources are 100% noisy, with three of them containing label noise and 2 containing input noise.

Noise Type	Model Types	
	Standard	LAP (Ours)
Input and Label Noise	68.05 $\pm$ 0.26	70.61 $\pm$ 0.26

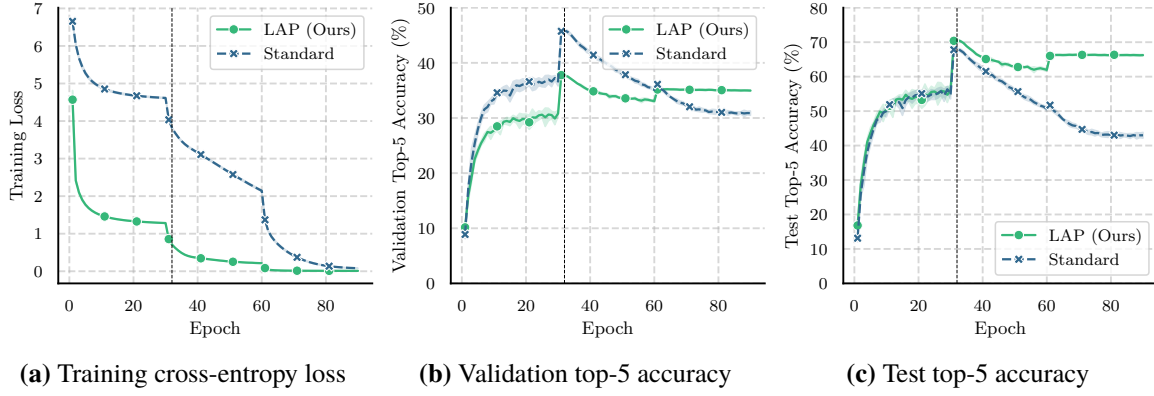


Figure 5.10: Performance on ImageNet. In (a) we show the cross-entropy loss on the training data at each epoch where the loss using LAP represents the tempered cross-entropy loss. Note that in this figure, the training loss for LAP appears lower due to the weighting of sources we use. This figure shows that both training loss curves do not look unusual. In (b) we present the validation top-5 accuracy at each epoch. In (c) we present the test top-5 accuracy at each epoch to demonstrate the fitting to the noise that occurs when not using LAP during late training. Here, 5 out of a total of 10 sources are 100% noisy, with 2 containing input noise, and 3 containing label noise. The two steps in performance occur at the points at which we scale the learning rate using a scheduler (epoch 30 and 60). The lines and error bands represent the mean and standard deviation over the 5 repeats. The vertical black dashed line represents the epoch at which LAP achieved maximum top-5 accuracy on the test set.

Late Training Test Accuracy

The noisy data literature often presents model performance results on the last n epochs since, in reality (as we do not have access to clean test labels), there is no way of knowing when to stop training and reduce overfitting to noisy data. In our work, in an effort to be most fair to the standard training baseline, we present the maximum performance over all of the training to ensure that we are not presenting results in which the standard training baseline has significantly overfit to the noisy data. However, in Table 5.12, we also present the average test accuracy over the last 10 epochs, as often reported in the literature. These results show the significant improvement that the use of LAP can have on model training when it is unknown if the standard training method is overfitting to noisy data.

Table 5.12: Imagenet late training results. Mean $\pm$ standard deviation of test top-5 accuracy (%) over the last 10 epochs of standard training and LAP on Imagenet data with noisy labels and noisy inputs, repeated 5 times. Here, 5 out of 10 sources are 100% noisy, with three of them containing label noise and 2 containing input noise.

Noise Type	Model Types	
	Standard	LAP (Ours)
Input and Label Noise	42.96 ± 0.64	66.26 ± 0.21

Similarly, Figure 5.10 presents the training, validation, and testing performance at each epoch during the

training of a neural network using the standard method and LAP. We can see that the training curves appear as usual (Figure 5.10a, with LAP's training curve representing the tempered loss) and the validation accuracy is greater for standard training (since the validation set contains noisy labels), but when tested on the clean labels (Figure 5.10c), it is clear that using LAP enables greater robustness to noisy data. Early stopping could be used here to achieve the maximally achieving models (since the epoch of maximum accuracy on the test set is the same as the validation set), which is why we find it more informative to report the maximum test accuracy as presented in all other experiments.

5.5.17 Using Empirical Risk to Improve UTI Prediction Across Sexes

As an extension to the work presented in Chapter 5, we studied the case in which data sources produce conflicting labels that are non-noisy in Capstick and Barnaghi (2023) [72]. In this case, we wish to learn labellings that are source-dependent, similar to the multi-task learning setting. In Fan et al. (2024) [73], we combined this with the work in this chapter to study how empirical risk can be used to identify groups of participants in the Minder study that the model is learning less from when predicting UTI. Using this information, we were able to find distinct groups of participants in our study that are more challenging to learn from, allowing us to build fairer models. By labelling these groups of participants as tasks, we evaluated simple methods from the multi-task learning literature [240] and found that we can improve the accuracy of UTI prediction compared to standard neural networks, whilst reducing their bias towards some participants.

In Fan et al. (2024) [73], we trained an MLP model with two hidden layers of sizes 30 and 10 and a ReLU activation layer, using the Adam optimiser [107]. We performed 5-fold cross-validation on the dataset, where participants only appear in either the training or testing sets on each fold. An MLP model was chosen for this analysis, instead of the logistic regression model given in Chapter 4, to provide more variation in the multi-task learning strategies evaluated.

Table 5.13: Cross-validation performance on the UTI prediction task, split by sex. An MLP model is trained on the Minder UTI dataset using a method for multi-task learning. This shows the mean $\pm$ standard deviation (%) of the accuracy of 5-fold cross-validation, where participants only appear in either the training or testing set.

	Precision		Sensitivity		Accuracy	
	Female	Male	Female	Male	Female	Male
Standard	17.45 $\pm$ 28.07	50.03 $\pm$ 44.88	4.95 $\pm$ 9.99	30.67 $\pm$ 39.28	52.27 $\pm$ 27.97	83.61 $\pm$ 13.92
Multi-task Method	48.73 $\pm$ 41.86	64.91 $\pm$ 38.84	39.65 $\pm$ 41.48	60.00 $\pm$ 41.62	75.10 $\pm$ 17.53	86.68 $\pm$ 12.00

We find that the use of separate final predictive layers for each patient group is an effective method for improving their predictions. The results of this method are given in Table 5.13, edited from Fan et al. (2024) [73]. This method resulted in fairer UTI predictions across sexes when tested on new participants using cross-validation and reduced the gap in accuracy between male and female participants by 63.05%; from an average accuracy on the female and male cohort of 52.27% and 83.61%, respectively, to 75.10% and 86.68%, respectively.

5.6 Discussion

As is the case with all methods designed for training models on noisy data, consideration must be given to the underlying cause of noise. For example, in some scenarios, noise within a dataset could be attributed to observations on minority groups during data collection, rather than as a result of errors in measurements, labelling, or data transfer. This is why we began with the assumption that all data (if non-noisy) is expected to be generated from the same underlying probability distribution. If data from minority groups are not carefully considered, *any* technique to learn from noisy data, or standard training could lead to an unexpected predictive bias [221].

In this work, we presented LAP, a method designed for training neural networks on data generated by many data sources with unknown noise levels. In Section 5.5, we observed that using LAP during training improves model performance when trained on data generated by a mixture of noisy and non-noisy sources, and does not hinder performance when all data sources are free of noise. Our results also illustrate that applying the proposed method is beneficial at varied noise levels and when training on multiple sources with differing levels of noise, whilst being robust to different types of noise. Moreover, we show that: (1) Our method is robust to an extreme source class distribution that tests the limits of the assumptions we made when proposing LAP (Section 5.5.14); (2) The improvements in results translate to large-scale datasets (ImageNet), with multiple noise types (Section 5.5.16); and (3) LAP is robust to overfitting of noisy data, and therefore achieves significantly greater performance during late training (Section 5.5.16). Furthermore, our implementation (Section 5.4.6) allows this method to be easily applied to any neural network training where data is generated by multiple sources, and our analysis in Section 5.4 and Section 5.5.6 provides a detailed description of the parameters introduced and their intuitions.

5.7 Summary

This work shows that LAP provides improved model performance over the baselines in a variety of noise settings and equal performance on non-noisy data. This is achieved whilst being robust to a multitude of tasks and being cheaper to compute than the baselines designed to tackle noisy data. We imagine many scenarios where LAP is relevant.

When designing the predictive model for UTI (Chapter 4), it became clear that in remote healthcare monitoring, collecting continuous measurements over features is scalable, but the labelling of data can be costly or challenging. In addition, the work in the current chapter studied how to mitigate the effects of noisy labels or features in large datasets. However, when datasets are small, other techniques must be employed.

Placing prior distributions over parameters of predictive models offers an opportunity to begin resolving these issues since well-specified priors improve the accuracy of models when labels are sparse and allow for reliable estimates of predictive uncertainty. However, in some settings, expert prior elicitation can be costly or infeasible. In the next chapter, we will study whether large language models can elicit expert priors directly or be used to supplement the elicitation process, reducing the cost required to specify prior distributions over the parameters of predictive models.

Chapter 6

Using Large Language Models for Expert Prior Elicitation in Predictive Modelling

This chapter is based on Capstick et al. (2025) [82] and is licensed under a Creative Commons Attribution 4.0 International License¹.

As we have discovered in Chapter 4, labelling data can be time-consuming and costly, particularly when the outcome is related to an adverse health condition. In addition, Chapter 5 discussed the variable quality of features and labels in healthcare data when datasets are large. However, in some cases, we have access to only a few labelled examples, some of which might be noisy. These challenges can be addressed together, by placing prior distributions over the parameters of our predictive models. However, performing prior elicitation with experts can be expensive, difficult, or infeasible in many situations.

In this chapter, we introduce AutoElicit to extract knowledge from LLMs and construct priors for predictive models. In contrast to the literature, our framework uses LLMs to elicit priors over the parameters of linear models directly, which we evaluate on public and private predictive tasks. We find these priors are informative and can be refined using natural language. As well as improving performance when data is scarce, their use allows us to provide estimates of predictive uncertainty and mitigate the effects of noisy data. Additionally, we perform a careful study contrasting AutoElicit with in-context learning and demonstrate how to perform model selection between the two methods. We find that AutoElicit yields priors that can substantially reduce error over uninformative priors, using fewer labels, and consistently outperform in-context learning. We show that AutoElicit saves over six months of labelling effort when

¹<http://creativecommons.org/licenses/by/4.0/>

building a new predictive model for Urinary Tract Infections (UTIs) from sensor recordings of people living with dementia.

6.1 Introduction to the Problem

Labelling data in medicine can be a costly and time-consuming undertaking [39, 75]. We are particularly interested in the development of automated tools to assist with healthcare monitoring for people living with dementia. In our study, participant data is recorded through passive monitoring devices. Although input features are continuously collected at scale, the process of labelling adverse health conditions, such as UTIs, can be challenging. Expert knowledge of a task, as privileged information [76], can provide prior distributions of model parameters, which often reduces the number of samples needed to learn performant models while also enabling more accurate estimates of predictive uncertainty and mitigating the effects of noisy data. In Chapter 4, we consult experts when engineering features for the prediction of UTI, incorporating prior knowledge into our data pre-processing. However, expert knowledge about model parameters can be difficult to obtain due to the challenges of performing expert prior elicitation, particularly in low-resource settings [77].

Linear models are widely used in real-world applications, especially in healthcare and as part of our wider study on dementia care, due to their interpretability, robustness, and clinical familiarity [1, 78, 79]. However, many studies do not use prior distributions on their models, possibly due to the challenges associated with elicitation, but would likely benefit greatly from them.

Due to their diverse training data and rate of development [80], Large Language Models (LLMs) offer a new opportunity to fully or partially automate the prior elicitation process, especially given recent advances in reasoning methods [81]. By prompting an LLM to approximate a specific form of expertise in a given field, it is possible to extract insights about a predictive model's prior based on the LLM's training data. This motivates the construction of a probabilistic framework to extract prior task knowledge from language models in the form of distributions over the parameters of a linear predictive model. In this way, AutoElicit combines developments in LLMs with trusted and robust linear predictive models, that are already widely understood and utilised in real-world applications.

In-context learning has been studied as a way to use LLMs for predictive tasks [132]. Here, LLMs are given a task description and demonstration examples and asked to make predictions on unseen data

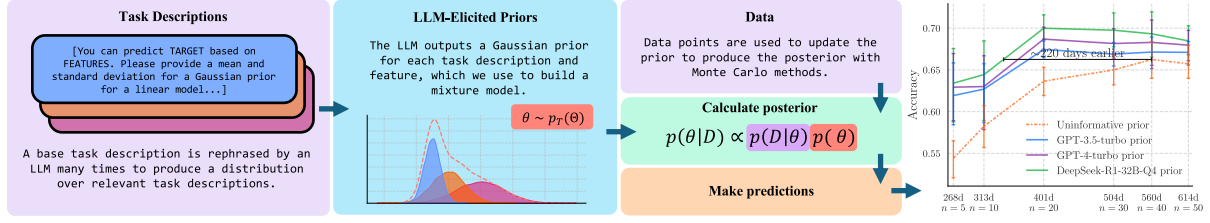


Figure 6.1: AutoElicit: Prior elicitation using a language model. The left-hand figure demonstrates the process, whilst the right-hand figure illustrates the benefits of using AutoElicit, achieving the same peak accuracy after $n \approx 15$ labels; 220 days earlier in the study.

directly [131]. There are several reasons why this prompted prediction may not suffice. First, LLMs can incur a large computational cost to run, rendering their use in high-frequency or low-resource settings unlikely. Second, applications may require arithmetic operations within models, an area where LLMs continue to struggle [241]. Third, LLMs, unless explicitly trained on a specific predictive problem, lack transparency and may not perform Bayesian inference [175], which can lead to unreliable posterior updates and predictions. To closely examine in-context learning for numerical predictive tasks, we propose a strategy to approximate the in-context prior and posterior distributions when imitating a linear model, which we can compare to LLM-elicited priors. Using these, we also empirically evaluate whether in-context learning approximates Bayesian inference before using model selection.

Our work makes the following contributions:

- We introduce AutoElicit, an algorithm to generate LLM-elicited priors over linear models. We find using LLMs for expert prior elicitation (Figure 6.1) provides higher posterior accuracy and lower mean square error than an uninformative prior on various numerical predictive tasks, improving models with fewer examples.
- To compare AutoElicit with In-Context Learning (ICL), we develop an approach to extract the implicit priors used in ICL, which we find to be less informative. To understand the differences, we study whether an LLM performs Bayesian inference during ICL. We extract the ICL posterior distribution and measure its divergence from posterior Monte Carlo samples of the extracted prior. We find that the LLM inconsistently approximates Bayesian inference on real-world datasets.
- We study when ICL is better than AutoElicit and show how to use the Bayes factor to guide model selection, finding that LLM-elicited priors are the preferred choice for all tasks tested. When predicting UTIs in our Minder study on dementia care, we find that AutoElicit priors result in performant accuracy over six months earlier than with uninformative priors (Figure 6.1).

Our results provide insight about when and how to use LLMs to build better linear models, with fewer samples.

6.2 Prior Elicitation with Experts

Bayesian inference incorporates knowledge about a predictive task before data is observed. A prior $p(\theta)$ is placed on a predictive model, which holds intuitions about the parameters θ . Well-specified priors provide calibrated estimates of predictive uncertainty through the posterior predictive distribution and reduce the number of data points needed to construct comparably accurate predictive models.

Prior elicitation constructs prior distributions on model parameters, typically through expert judgement [142, 242]. This can be resource intensive and requires access to one or more domain experts who have been trained in prior elicitation [77].

6.3 Linear Models

Because of their simplicity, linear models have interpretable parameter weights, making them trusted in clinical contexts. The effect of model weights on predictions are explicitly understood as they multiply directly with features, making them ideal to use with prior elicitation in healthcare.

6.4 In-Context Learning

In-context learning uses the knowledge that LLMs acquire during training to make predictions on unseen data. Here, LLMs attempt to learn a function for predicting unseen data based on a task description and demonstration examples.

Let the space of all predictive tasks be $\mathcal{T}$, a single classification or regression task be $T \in \mathcal{T}$, and the space of possible LLM prompts be $\mathcal{I}$. Given an LLM M and K task descriptions $I_k \sim p(I|T) = p_T(I)$, a prediction of $\tilde{y}$ on features $\tilde{x}$ using in-context learning with training data $\mathcal{D}$ is:

$$p_{M,T}(y|\tilde{x}, \mathcal{D}) = \sum_{k=1}^K p_{M,T}(\tilde{y}|\tilde{x}, \mathcal{D}, I_k) p_{M,T}(I_k)$$

Although the mechanisms of in-context learning are under active research, we denote its implicit parameters

as φ , implied by a task description and dataset:

$$p_{M,T}(y|\tilde{x}, \mathcal{D}) = \int_{\Phi} \sum_{k=1}^K p_{M,T}(y|\tilde{x}, \varphi) p_{M,T}(\varphi|\mathcal{D}, I_k) p_{M,T}(I_k) d\varphi$$

Here φ is itself described by a prior distribution $p_{M,T}(\varphi|I_k)$ and a posterior distribution $p_{M,T}(\varphi|\mathcal{D}, I_k)$.

φ is latent, since in-context learning only returns $\tilde{y} \sim p_{M,T}(y|\tilde{x}, \mathcal{D}, I)$.

6.5 AutoElicit: Using Language Models to Elicit Priors

We aim to make expert prior elicitation more accessible by using the knowledge of LLMs in robust and clinically trusted predictive models. This motivates us to develop a framework, AutoElicit, for eliciting general prior distributions for linear models from LLMs (Figure 6.1), which often have both technical knowledge of probability distributions and domain knowledge of a task. Our goal is to improve model performance with less data compared to an uninformative prior, whilst allowing model parameters to be interpretable.

Given an LLM M and task T , we obtain a single Gaussian prior for each feature, for each task description I_k by asking the LLM to provide its best guess of the mean and standard deviation: $(\mu_k, \sigma_k) \sim p_{M,T}(\mu, \sigma|I_k)$. Here, we use $K = 100$ task descriptions, produced by asking an LLM to rephrase one human-written “system” and “user” role 10 times and taking their product.

Taking a mixture over the task descriptions, we construct a prior $p_{M,T}(\theta)$ over linear model parameters θ :

$$p_{M,T}(\theta) = \sum_{k=1}^K \pi_k \mathcal{N}(\theta|\mu_k, \sigma_k^2) \text{ where } (\mu_k, \sigma_k) \sim p_{M,T}(\mu, \sigma|I_k) \text{ and } \boldsymbol{\pi} \sim \text{Dir}(\mathbf{1}) \quad (6.1)$$

With $\boldsymbol{\pi} = (\pi_1, \dots, \pi_K)$ and $\mathbf{1} = (1, \dots, 1) \in \mathbb{R}^K$. Since mixtures of Gaussians approximate any distribution with enough components, we can fully capture the prior belief of the LLM. We also expect that LLMs have learnt how to parametrise Gaussian distributions. However, our framework also allows mixtures of other probability distributions to be elicited. Corresponding to the number of task descriptions, this mixture has $K = 100$ components, but other numbers are studied in Section 6.10.4.

The posterior predictive distribution on unseen data $(\tilde{x}, \tilde{y})$, given an observed dataset $\mathcal{D}$, becomes:

$$p_{M,T}(y|\tilde{x}, \mathcal{D}) = \int_{\Theta} \sum_{k=1}^K p_{M,T}(y|\tilde{x}, \theta) p_{M,T}(\theta|\mathcal{D}, I_k) p_{M,T}(I_k) d\theta$$

Where $p_{M,T}(I)$ describes the distribution over task descriptions and $p_{M,T}(\theta|\mathcal{D}, I)$ describes the posterior distribution over the parametrisation of a predictive model, which is calculated using the available data $\mathcal{D}$ and the AutoElicit prior over θ .

This $p_{M,T}(\theta)$ is the prior that we use for our linear models and is an alternative to one elicited from experts. However, additionally including expert information E in the prompts can provide yet more informative priors: $p_{M,T}(\theta|E)$. This way, AutoElicit also supplements a typical prior elicitation process but only requires experts to provide knowledge as natural language, likely needing less training and cost. This approach enables the use of an LLM’s knowledge in new environments, with more controls and transparency.

The natural comparison for AutoElicit priors is an uninformative prior. However, since we use an LLM’s prior knowledge and training data to make new predictions, this is also an alternative to in-context learning. The difference is that prior elicitation uses a marginal distribution and Bayesian inference, which address some of the shortfalls of in-context learning on numerical predictive tasks.

6.6 Energy Distance and Statistic

To quantitatively compare different priors, we will make use of the energy statistic [243], which is based on the energy distance metric [244] and is a method to measure the similarity of two probability distributions. Importantly, it allows us to measure the distance between two N -dimensional probability distributions, whilst other common methods, such as the Kolmogorov–Smirnov test and Hellinger distance, are restricted to single-dimensional probability distributions.

With X and Y as independent random variables, and $\{x_1, x_2, \dots, x_p\}$ and $\{y_1, y_2, \dots, y_q\}$ as samples from X and Y respectively, with $x_i, y_j \in \mathbb{R}^N$, the energy distance $D(X, Y)$ is defined as:

$$D(X, Y)^2 = 2\mathbb{E}[M_{XY}] - \mathbb{E}[M_{XX}] - \mathbb{E}[M_{YY}]$$

Where $M_{XY_{i,j}} = \|x_i - y_j\|_2$ is the pairwise distance matrix. From this, the energy statistic is defined as:

$$H(X, Y) = \frac{D(X, Y)^2}{2\mathbb{E}[M_{XY}]} \quad (6.2)$$

This statistic takes values between 0 and 1 and measures how similar two statistical distributions are. We implement the energy distance and statistic within our code, which is optimised to run on large numbers of samples.

6.7 In-Context Learning

To compare in-context learning and AutoElicit, we must consider: (1) Does the LLM use the same elicited prior in-context? and (2) Does in-context learning approximate Bayesian inference? To answer these, we extract the prior and posterior distributions of the in-context model.

6.7.1 Extracting the In-Context Prior and Posterior

We can write the implicit in-context model as: $f_\varphi : \mathcal{X} \rightarrow \mathcal{Y}$ where f represents the class of the predictive model, which we specify in the prompt, φ is the random variable associated with its parameters, and $\mathcal{X}$ and $\mathcal{Y}$ correspond to its domain and range. Based on the model class f corresponds to, $y \in \mathcal{Y}$ will either be a predicted target or its probability.

In-context learning returns a y given an x . Therefore, since we do not have access to φ directly, we propose to use $p_{M,T}(y|x, \varphi)$ and maximum likelihood estimation (MLE) to generate a sampling distribution for $p_{M,T}(\varphi)$ and $p_{M,T}(\varphi|\mathcal{D})$.

To do this, given a language model M and task T , we construct the sample distribution [105]:

$$\varphi_{\text{MLE}} \sim \text{MLE}_{M,T}(f_\varphi(X)|X, I) \quad (6.3)$$

Where $\text{MLE}_{M,T}(f_\varphi(X)|X, I)$ refers to the MLE of the parameters φ given input X , in-context predictions $f_\varphi(X)$, and task description I .

For this, we say that a dataset (X, Y) is generated by a given φ sampled from $\varphi \sim p_{M,T}(\varphi)$ and X . By sampling $x \sim \mathcal{U}[\mathcal{X}]$ from a uniform distribution over $\mathcal{X}$, we can create S different datasets of the form:

$$(X, Y)^{(s)} = \{y_n \sim f_\varphi(x_n) : n = 1 : N, x \sim \mathcal{U}[X]\} \quad (6.4)$$

With these, we can obtain a sample of the estimated prior on φ for the domain of $\mathcal{X}$.

$$\varphi_{\text{MLE}}^{(s)} = \text{MLE}_{M,T}(Y^{(s)}|X^{(s)}, I)$$

Further, additionally providing training data $\mathcal{D}$, returns posterior samples:

$$\varphi_{\text{MLE}}|\mathcal{D} \sim \text{MLE}_{M,T}(f_{\varphi}(X)|X, \mathcal{D}, I) \quad (6.5)$$

By repeatedly sampling, we can form distributions that estimate $p_{M,T}(\varphi)$ and $p_{M,T}(\varphi|\mathcal{D})$. In our experiments, for each φ_{MLE} , we sample 25 data points from $X \sim \mathcal{U}[-5, 5]$, five times for each of the 100 task descriptions. As the real data is normalised, these samples of X will cover the domain of f_{φ} with a high probability.

When specifying linear f_{φ} , we calculate the MLE in closed form as we have $y = \varphi_0 + \varphi \cdot x$, which is either a regression label or the logit of a class. Therefore, large approximation errors will be due to the LLM not using a linear f_{φ} .

$p_{M,T}(\varphi)$ and $p_{M,T}(\varphi|\mathcal{D})$ are distributions over the hidden parameters of the in-context model, giving us new visibility into the LLM's predictive process; assuming the model class can be specified in a prompt. Using these, we can test whether an LLM is using the elicited prior in-context and if it is approximating Bayesian inference.

6.7.2 Is the LLM Using the Elicited Prior In-Context?

To quantify the difference between the in-context prior ($p_{M,T}(\varphi)$) and the one we elicit ($p_{M,T}(\theta)$), we will use the energy statistic [243]. This takes values $\in [0, 1]$ with 0 indicating that the two distributions are equal and evaluates how consistently the LLM uses its background knowledge.

6.7.3 The In-Context Posterior

We also test whether the LLM is approximating Bayesian inference during in-context learning. We use the approximated in-context prior $p_{M,T}(\varphi)$ and Monte Carlo methods [245, 246] to sample a posterior, and compare it to the extracted in-context posterior $p_{M,T}(\varphi|\mathcal{D})$. The energy statistic quantifies the difference in these distributions. If they differ, it suggests the LLM is not performing Bayesian inference in-context

and is not acting predictably after observing data from $\mathcal{D}$.

6.8 Bayes Factor for Model Selection

If the LLM is approximating Bayesian inference in-context, we use the Bayes factor for model selection between in-context learning and AutoElicit priors. This model selection strategy calculates the marginal likelihood ratio of two statistical models to determine under which a given dataset was likely generated.

Given two models α_0 and α_1 with parametrisations θ_0 and θ_1 , the Bayes factor over a dataset $\mathcal{D}$ is:

$$\text{BF}(\alpha_0, \alpha_1; \mathcal{D}) = \frac{p(\mathcal{D}|\alpha_0)}{p(\mathcal{D}|\alpha_1)} = \frac{\int_{\Theta} p(\theta_0|\alpha_0)p(\mathcal{D}|\theta_0, \alpha_0)d\theta_0}{\int_{\Theta} p(\theta_1|\alpha_1)p(\mathcal{D}|\theta_1, \alpha_1)d\theta_1} \quad (6.6)$$

Both marginal likelihoods will be calculated differently since in-context learning and AutoElicit provide prior predictions in different ways.

6.9 Availability of Code

We make our code publicly available for ease of reproducibility:

- For generating all figures and results:
 - <https://github.com/alexcapstick/llm-elicited-priors>
 - <https://github.com/tmi-lab/llm-elicited-priors>
- For applying our methods to new tasks:
 - <https://github.com/alexcapstick/autoelicit>
 - <https://github.com/tmi-lab/autoelicit>

Our code includes a pip installable library for ease of use (Figure 6.2).



Figure 6.2: The AutoElicit python package: `pip install autoelicit`.

This open-source python package makes it simple to obtain LLM-elicited priors:

```

# import the elicitation function
from autoelicit.gpt import get_llm_elicitation_for_dataset

# elicit priors for a dataset
get_llm_elicitation_for_dataset(
    # the language model client requires a .get_result(messages)
    client=llm_client,
    # the task descriptions to iterate over
    system_roles=system_roles,
    user_roles=user_roles,
    # the dataset feature names
    feature_names=data.feature_names,
)

```

Within the code, we provide classes that wrap common LLMs, whilst making it easy to adapt the methods to future language models.

This code is implemented in python (v3.11.9), using pip (v24.0). For our experiments, we use the packages:

- blackjax (v1.2.3) [247]: For Monte Carlo sampling on the in-context prior samples.
- jax (v0.4.31) [248]: For calculating some metrics and use with blackjax.
- numpy (v1.26.4) [211]: For manipulation of arrays and computing mathematical equations.
- ollama (v0.4.7): For loading the DeepSeek model.
- openai (v1.51.2): For communicating with the OpenAI API.
- pandas (v2.2.2) [212]: For manipulating the datasets and results.
- pymc (v5.17.0) [249]: For Monte Carlo sampling on the elicited priors.
- scikit-learn (v1.5.1) [213]: For loading and preprocessing the data.
- transformers (v4.46.0) [235]: For loading and using Llama models.

6.10 Evaluation of AutoElicit

In all experiments, we use OpenAI’s GPT-3.5-turbo (specifically GPT-3.5-turbo-0125), GPT-4-turbo (GPT-4-turbo-2024-04-09)², and DeepSeek-R1-32B-Q4³ with a temperature of 0.1 and all other settings as default.

6.10.1 The LLMs Tested

Within the main experiments, we evaluate three language models:

- GPT-3.5-turbo⁴ is one of OpenAI’s least expensive models and is described as a model for simple tasks.
- GPT-4-turbo⁵ is one of OpenAI’s more sophisticated models.
- DeepSeek-R1-32B-Q4⁶, a 32 billion parameter model from DeepSeek based on Qwen 32B and distilled from DeepSeek-R1.

These represent three distinct types of language models. GPT-3.5-turbo is a cost-effective option for those who are able to use OpenAI’s API and removes the need to run the language model locally. Similarly, GPT-4-turbo is hosted by OpenAI, however, it is a considerably more expensive option. In contrast, DeepSeek-R1-32B-Q4 is an open-source model that can be deployed locally; allowing priors to be elicited in a secure environment and without API costs.

6.10.2 The Datasets Tested

In this work, we evaluate our method on 6 datasets covering both classification and regression, 1 of which is synthetic, 1 is privately collected, and 4 are publicly accessible. This allows us to provide both reproducible experiments and test the language model’s ability to suggest a good prior on datasets not seen during training.

The contents of each dataset are as follows:

²<https://platform.openai.com/docs/api-reference/chat>

³<https://ollama.com/library/deepseek-r1:32b>

⁴<https://platform.openai.com/docs/models>

⁵<https://platform.openai.com/docs/models>

⁶<https://ollama.com/library/deepseek-r1:32b>

- $y = 2x_1 - x_2 + x_3$: To evaluate language models as tools for prior elicitation, we construct a synthetic dataset where we have control over the parameters. Because of this, we can provide the language model with a complete description of the task and test its ability to convert unambiguous text into prior distributions. This dataset contains 3 features sampled from a normal distribution (mean = 0 and standard deviation = 1) and targets calculated using the equation $y = 2x_1 - x_2 + x_3 + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 0.05)$.
- The Minder dataset collected as part of our study on dementia care [1] and discussed in Section 2.1: This dataset contains 10 features engineered (using a subset of those in Chapter 4) from measurements of in-home activity and physiology. The raw features were measured using Passive Infrared (PIR) sensors located around the homes of people living with dementia, and a sleep mat positioned in the participants' beds. The targets correspond to days with a clinical labelling of positive or negative UTI which were calculated through the analysis of a provided urine sample in consultation with a healthcare professional. For this analysis, we used a subset of the data collected between 2021-07-01 and 2024-01-30, which contains 78 people and 371 labelled days of UTI, with 270 negatives and 101 positives. When training and testing on this dataset, we z-normalised the training data and used the same statistics to normalise the test data. Importantly, because this dataset was privately collected, it has not appeared in the language model's training.
- The Heart Disease dataset [97, 98]: This dataset contains 10 physiological features and targets with a positive or negative diagnosis of heart disease for 299 people. Within this dataset, we have 161 examples of people without heart disease and 138 examples of people with heart disease, totalling 299 data points. When training and testing on this dataset, we z-normalised the training data and used the same statistics to normalise the test data, except where features are categorical. For the categorical features, we perform no scaling.
- The Hypothyroid dataset [99, 100]: This dataset, processed from the original, consists of 150 examples of patients' thyroid-related blood levels and whether they have a hypothyroidism diagnosis or not. This dataset contains 4 features, with 75 examples of a patient with hypothyroidism and 75 examples of a patient without hypothyroidism. When training and testing on this dataset, we z-normalised the training data and used the same statistics to normalise the test data.
- The Diabetes dataset [94]: After preprocessing, this dataset contains physiological measurements and demographic information from 442 people with diabetes. Here, the task is to predict a quantitative measure of their diabetes disease progression one year after this baseline. This dataset consisted

of 10 features and the targets were regression values with a maximum value of 10.0, minimum value of 0.0, and a mean value of 3.96. When training and testing on this dataset, we z-normalised the training data and used the same statistics to normalise the test data.

- The Breast Cancer dataset [91]: This dataset is made up of 569 data points with 30 features computed from an image of a breast mass that are labelled as either malignant or benign. In total, this dataset contains 357 examples of benign tumours and 212 examples of malignant tumours. When training and testing on this dataset, we z-normalised the training data and used the same statistics to normalise the test data.

Where datasets are publicly available, we perform minimal processing for ease of reproducibility. In some cases, we replace the feature names with ones that are more descriptive (taken from the dataset descriptions also available publicly) to aid the language model’s comprehension of the data. All of these changes to the raw data are available within the supplementary code, which will automatically download and process the data.

6.10.3 Dataset Memorisation

Since some of the tested datasets are publicly available, some of their contents may have appeared within the training data for language models. Before we run our experiments, we will use methods proposed in Bordt et al. (2024) [171] to quantify the extent to which GPT-3.5-turbo already knows information about the datasets.

We start with the header completion test, which checks whether the language model has memorised the first few rows of the dataset. Here, the column names, along with a given number of the first rows, are given to the language model as a string, with the final of these rows truncated. The language model is then asked to generate the next 500 tokens.

We then use the Levenshtein distance [250, 251], a string matching metric, to calculate the agreement between the true header completion and the one provided by the language model. This counts the number of single-character changes required to transform one string into another, only allowing for insertions, deletions, and substitutions. The value can take a minimum value of 0, suggesting that the strings are equal, and a maximum value of the longest length of the two strings.

To make the Levenshtein distance comparable across datasets, we calculate its normalised value by

dividing the Levenshtein distance by the length of either the true string or the language model’s completion, whichever is longer. Because of this, the normalised Levenshtein distance reported is not formally a metric as it does not follow the triangle inequality. However, it allows us to compare the header completions across datasets more easily:

$$\hat{L}(s_1, s_2) = \frac{\text{Levenshtein}(s_1, s_2)}{\min\{|s_1|, |s_2|\}}$$

Here, $\text{Levenshtein}(s_1, s_2)$ calculates the Levenshtein distance between two strings, and $\min\{|s_1|, |s_2|\}$ returns the minimum length of the strings. This new value ranges between 0 and 1, with 0 indicating a perfect match between the strings and 1 implying that they were different at all positions.

Table 6.1: GPT-3.5-turbo header completion test results. Normalised Levenshtein distance between the correct header completion and the language model’s completion for each of the datasets tested. A value of 0 indicates that the true completion and language model’s completion were equal and a value of 1 means that they were different at all positions.

Heart Disease	Diabetes	Hypothyroid	Breast Cancer
0.00	0.21	0.22	0.00

The values in Table 6.1 show the Levenshtein distance between the true dataset header completion and the one made by the LLM. This demonstrates that the LLM has memorised the first few rows of the Breast Cancer and Heart Disease datasets, shown by a normalised Levenshtein distance of 0. This could be because the first few rows of these datasets commonly appear within the language model’s training data. For the Diabetes and Hypothyroid datasets, we see some level of memorisation, with the language model being able to reproduce a large amount of the true header. As we might expect, when we perform the header test on the synthetic dataset generated using the relationship $y = 2x_1 - x_2 + x_3$, we get a normalised Levenshtein distance of 0.7, which shows a large difference between the true header string and the completion from the language model. The UTI task was not tested, as it is a private dataset.

Table 6.2: GPT-3.5-turbo row completion test results. Normalised Levenshtein distance between the correct row completion and the language model’s completion for each of the datasets tested. A value of 0 indicates that the true completion and language model’s completion were equal and a value of 1 means that they were different at all positions.

Heart Disease	Diabetes	Hypothyroid	Breast Cancer
0.33 ± 0.05	0.38 ± 0.05	0.20 ± 0.05	0.43 ± 0.13

We supplement these results with those of Table 6.2, which shows the results of the row completion test. Here we calculate the normalised Levenshtein distance of the true completions and the language model’s completion of a random row from the dataset. In these tests, the language model is provided with 10

rows (in order from the original dataset) and is asked to complete the next row. The values in Table 6.2 represent the mean and standard deviation of the test using 25 random rows. Here, in contrast to the header test, we see that the language model has not memorised all rows from the dataset. We likely see these results because when the datasets appear within the language model’s training data, their column names and initial rows are usually given, but not the entire dataset. This often occurs when datasets are described in repositories or blog posts, or when used in machine learning tutorials.

Within our experiments, we normalise the training and testing data and sample random observations from the dataset. Because of this, observations predicted during in-context learning will likely not have been memorised, since they are normalised based on a training data subset and are selected from random rows of the dataset. Also, although the language model can complete the first few rows of the datasets with some success, it is unable to complete random rows. Therefore, we hypothesise that any memorisation of the tested datasets has little impact on the in-context predictions.

Further, since for prior elicitation with AutoElicit, we only prompt the language model for a prior distribution of a predictive model given the feature names and a description of the dataset, the memorisation of rows from the data is unlikely to have had an impact on these results. In addition, we will see that elicited priors provide improved predictions over the uninformative prior for the private and synthetic datasets (with no dataset memorisation), as well as the public datasets.

6.10.4 How Informative are AutoElicit Priors?

To evaluate the usefulness of the AutoElicit priors, we start by comparing their posterior performance to an uninformative prior when used for the tasks given in Section 6.10.2. For each dataset, we select 50% of the data points to form the test set and use the remaining data points to form a training set. This is repeated 10 times to produce 10 folds of random training and testing splits.

To sample the posterior distribution over the parameters of the linear model, we use the No-U-Turn sampler [246]. We start by performing 1000 adaptation steps to tune the step size with a target acceptance rate of 0.8, then generate 5000 linear model parameter values for each of the 5 chains. With the posterior samples of the linear model parameters, we make predictions on all of the data points in the dataset’s test set. These are used to calculate the predictive performance metrics for each task.

For the UTI task, we split the dataset so that each participant’s data is only in the testing or training set,

since some participants produced multiple examples. For the other classification tasks, the splits will be stratified using the targets. This ensures that the training and testing folds are balanced and there is no data leakage. As part of our analysis, we evaluate the posterior performance after the model has observed a given number of data points. For this, we randomly select the corresponding number of examples from the training set to use as training data.

We have 100 task descriptions for each task, generated by asking an LLM to rephrase a single system role and user role 10 times and taking their product. These task descriptions are used to elicit prior distributions from the LLMs. As an example, for the synthetic task, one of the task descriptions is as follows:

- **System role:** You're a linear regression predictor, estimating the target based on some input features. The known relationship is: 'target' = 2 * 'feature 0' - 1 * 'feature 1' + 1 * 'feature 2'. Use this to predict the target value.
- **User role:** I am a data scientist working with a dataset for a prediction task using feature values to predict a target. I would like to apply your model to predict the target for my samples. My dataset consists of these features: ['feature 0', 'feature 1', 'feature 2']. All the values have been standardised using z-scores. The known relationship is: 'target' = 2 * 'feature 0' - 1 * 'feature 1' + 1 * 'feature 2'. Based on the correlation between each feature and target, whether positive or negative, please guess the mean and standard deviation for a normal distribution prior for each feature. I need this for creating a linear regression model to predict the target. Provide your response as a JSON object with the feature names as keys, each containing a nested dictionary for the mean and standard deviation. A positive mean suggests positive correlation with the outcome, negative for negative correlation, and a smaller standard deviation indicates higher confidence. Only respond with JSON.

From these, the LLMs will produce 100 Gaussian priors for each feature in the dataset. These can be used to build a mixture of priors using Equation 6.1, where we weight the Gaussian distribution in each dimension and each task description separately. With these, we will construct a logistic or linear regression model, where we set the prior distribution over the bias term as a mixture of 100 distributions of $\theta_0 \sim \mathcal{N}(\theta_0|0, 1)$. In the case of linear regression, we also use a noise term $\epsilon \sim \text{Half-Cauchy}(\epsilon|\beta = 1)$, with a Half-Cauchy prior.

Figure 6.3 shows the histogram of parameter values from the GPT-3.5-turbo-elicited prior for the synthetic

task, $y = 2x_1 - x_2 + x_3$. Here, the language model correctly provides prior distributions centred around the correct parameter values and offers a small standard deviation. This demonstrates that LLMs can combine their knowledge of probability with task descriptions to form accurate priors when the prompt is fully descriptive.

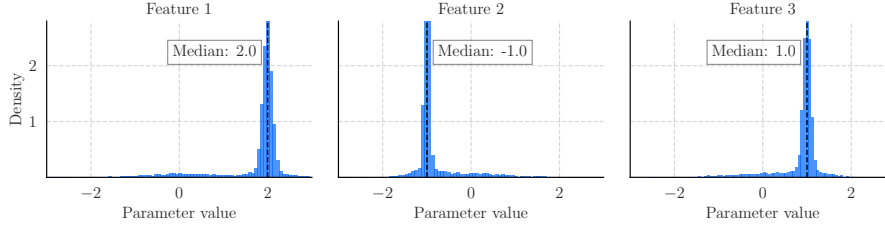


Figure 6.3: GPT-3.5-turbo-elicited priors for the synthetic task, $y = 2x_1 - x_2 + x_3$. We sample 10,000 parameter values from the mixture generated from the Gaussian priors elicited from GPT-3.5-turbo when given complete task descriptions. This figure shows the histograms of those values in each feature dimension.

We compare the posterior performance of the LLM-elicited priors with an uninformative prior, which we set as $\theta \sim \mathcal{N}(\theta|0, 1)$ for each feature. We use the No-U-Turn sampler [246] to sample the posterior distribution over the linear model parameters, using 5 chains of 5000 samples. In total, for each dataset, for each training and testing split, we generate 25,000 posterior samples of the linear model parameters. Using these samples, we make predictions on the test set for each task, which we use to calculate the posterior metrics of accuracy or Mean Squared Error (MSE).

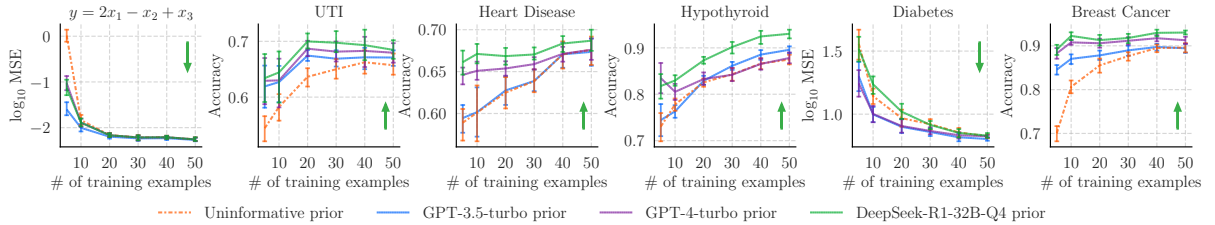


Figure 6.4: Test performance of the posterior distribution for varied training data sizes. The average and 95% confidence interval of the mean posterior accuracy or MSE of 10 splits of the dataset using AutoElicit priors and an uninformative prior ($\theta \sim \mathcal{N}(\theta|0, 1)$). These are calculated on a test set of each dataset, with the green arrow pointing in the direction of metric improvement.

Figure 6.4 compares the posterior performance of a predictive model using an uninformative prior $\theta \sim \mathcal{N}(\theta|0, 1)$, with those elicited from three LLMs. This illustrates that AutoElicit priors provide substantially better accuracy and MSE than an uninformative prior, and that at least one of the LLMs provides the greatest performance for all training sizes and tasks.

When eliciting priors for $y = 2x_1 - x_2 + x_3$, we specify the equation in the prompt to test each LLM’s ability to provide priors given complete descriptions. In Figure 6.4, the LLM-elicited priors provide an

order of magnitude lower test MSE compared to an uninformative prior after 5 observed examples and an equivalent MSE after 20, as the task was simple enough to be learnt with only a few examples. This is expected, as we saw that GPT-3.5-turbo returned sharp distributions centred on the true parameter values (Figure 6.3), providing the prior with the lowest posterior MSE.

For Heart Disease, the uninformative and GPT-3.5-turbo-elicited priors produce the same mean posterior accuracy, as this LLM-elicited prior contains many standard Normal distributions. This is reassuring, as GPT-3.5-turbo responded with an uninformative prior when unsure of its knowledge. However, GPT-4-turbo and DeepSeek-R1-32B-Q4 returned considerably better priors. In each case, the appropriate prior can be chosen using the Bayes factor and available data.

For Hypothyroid prediction, DeepSeek-R1-32B-Q4 provides the best accuracy by a large margin. Of the OpenAI models, smaller training sizes favour GPT-4-turbo, whilst larger training sizes favour, GPT-3.5-turbo. This is because the GPT-4-turbo-elicited prior was too strict to allow for updated posterior parameters, but was more informative than an uninformative prior for smaller training sizes.

For the Diabetes task, both OpenAI LLMs provide considerably improved MSE over the uninformative prior for training sizes up to 50 observed data points, whilst DeepSeek-R1-32B-Q4 provides worse MSE.

In the Breast Cancer task, the accuracy improvement from the elicited DeepSeek-R1-32B-Q4 and GPT-4-turbo priors is maintained across all training sizes, whilst GPT-3.5-turbo provides an improvement up to 40 observed data points. This can be seen in these distributions, which show a long tail in the accuracy distribution for the uninformative prior for most training sizes.

Of particular note are the results on the UTI task, a private dataset where all AutoElicit priors provide greater accuracy over the uninformative prior for all training sizes. Since we have access to the underlying study that collected this dataset, we can examine in more detail how the accuracy of the predictive model varies in time, as well as with the number of data points collected.

In Figure 6.5 we plot the time it took to collect a given number of labels against the corresponding accuracy when we model the task with an uninformative prior and those elicited from LLMs. This illustrates the usefulness of AutoElicit, as we were able to achieve greater accuracy earlier in the study, accelerating model development and validation.

Since UTI label collection is costly and slow, our predictive model could have achieved greater accuracy significantly earlier in the study. This is paramount in a clinical environment, where an improved UTI

detection model could have resulted in fewer unplanned hospital admissions.

It is important to note that the accuracy in Figure 6.5 is not achieved as a result of training on the earliest collected data points. This is because, in our analysis, we perform cross-validation and use a subset of the data from a later time period. Changes to the deployed in-home devices were made during the study, such as the addition of a device to collect sleep information, which we used within the dataset evaluated in the main results. However, if all data collection devices had been used from the beginning of the study, we could expect the accuracy in time to resemble the figure presented here.

Also of importance, in Figure 6.4, are the results on the classification tasks of the priors elicited from DeepSeek-R1-32B-Q4, a 4-bit quantised version of the distilled DeepSeek-R1 model. These suggest the reasoning qualities of more recently released LLMs are of significant use for eliciting priors, even when using a strong quantisation.

By further studying the accuracy distributions, as in Figure 6.6, we can additionally examine the variation in posterior results. In most cases, the LLM-elicited priors produce less variation in their posterior accuracy distributions, with less pronounced tails in their distributions.

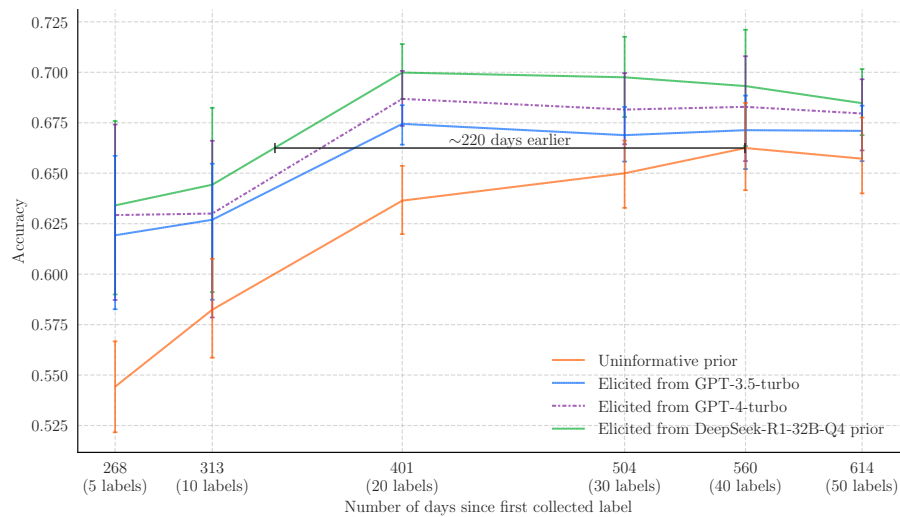


Figure 6.5: Posterior accuracy of the elicited priors on the UTI dataset considering label collection time. The average mean posterior accuracy and 95% confidence interval on the UTI test set after observing the given number of data points against the time it took for the data points to be collected in the study. The horizontal axis shows the time in days to achieve the given performance and the number of labels collected in that time.

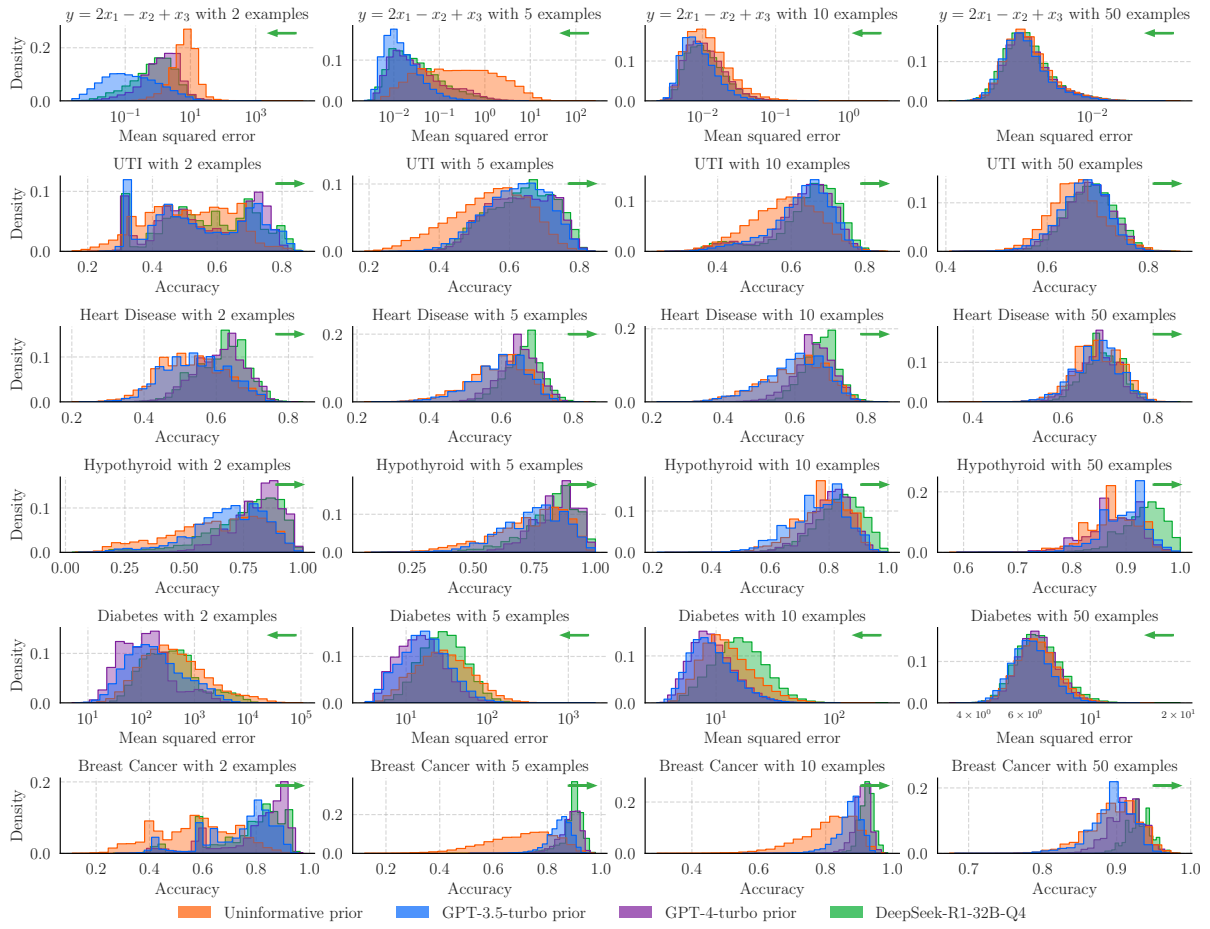


Figure 6.6: Test accuracy of the posterior distribution over the model weights of a logistic and linear regression model for varied numbers of training data points. We evaluate the accuracy and MSE of different prior distributions over the weights of a logistic and linear regression model. Each of the Gaussians in the mixture is extracted from a language model and is tested against an uninformative prior $\mathcal{N}(\theta|0, 1)$. These results are calculated on a test set of the datasets after each model is trained on the given number of data points. These values are the result of 5000 sampled parameters and 10 random splits of the training and testing data. The green arrows point in the direction of improvement in the metric.

Evaluating a More Uninformative Prior

In Figure 6.7 we compare the results presented in Figure 6.4 with an uninformative prior with a greater standard deviation, $\theta \sim \mathcal{N}(\theta|0, 100)$. This allows us to test a flat prior, which will provide an average posterior accuracy similar to maximum likelihood estimation.

This flatter prior distribution ($\theta \sim \mathcal{N}(\theta|0, 100)$) provides less performant posterior distributions for the regression tasks, likely because it samples model parameters that lead to large output values far from the true values. However, for the classification tasks, the change in performance is less clear. On the Breast Cancer task, the larger standard deviation in the prior leads to significantly worse performing posterior distributions, whilst on the Hypothyroid dataset, we see a significant improvement over $\theta \sim \mathcal{N}(\theta|0, 1)$. In

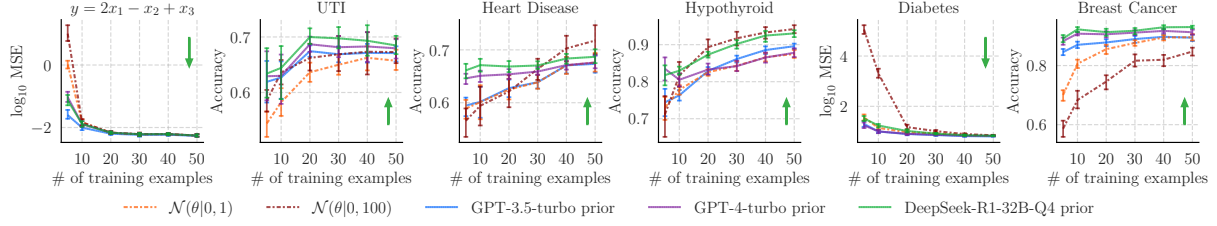


Figure 6.7: Further comparison with a “wider” uninformative prior The average and 95% confidence interval of the mean posterior accuracy or MSE of 10 splits of the dataset using AutoElicit priors and two uninformative priors, $\theta \sim \mathcal{N}(\theta|0, 1)$ and $\theta \sim \mathcal{N}(\theta|0, 100)$. These are calculated on a test set of each dataset, with the green arrow pointing in the direction of metric improvement.

fact, the wider standard deviation led to an improvement over the elicited priors for the Hypothyroid dataset, suggesting that this task benefits from models with larger model weights. When considering the Heart Disease prediction task, we find that the larger standard deviation performs better than $\theta \sim \mathcal{N}(\theta|0, 1)$ on the whole, but performs worse than the AutoElicit priors for smaller training sizes. For the UTI task, the flatter prior led to an improved posterior distribution over the standard normal prior, but still performed worse than the elicited prior distributions.

These results illustrate the remarkable effectiveness of using AutoElicit priors, which considerably improve posterior predictions, particularly for smaller training sizes, where label collection might have been challenging. By using LLMs, our framework makes expert prior elicitation more accessible.

How Many Ways Should We Describe a Task?

In this section, we examine how many task descriptions are required to be able to get a similar prior distribution to that elicited within our experiments, which used 100 task descriptions.

Figure 6.8 shows the energy statistic of samples from the prior distribution elicited using fewer task descriptions compared to samples from the prior distribution elicited using 100 task descriptions (from the same language model). In particular, we compare a subset of task descriptions with the full 100 task descriptions.

We see that for most tasks, a large number of task descriptions is required, since reducing the number of descriptions from 100 to 49 produced prior distributions that are significantly different for all datasets tested. Using 81 task descriptions produces prior distributions that are more similar but still show differences for some datasets (for example, $y = 2x_1 - x_2 + x_3$ and Breast Cancer).

The Breast Cancer dataset required more task descriptions to reduce the variation in the prior relative

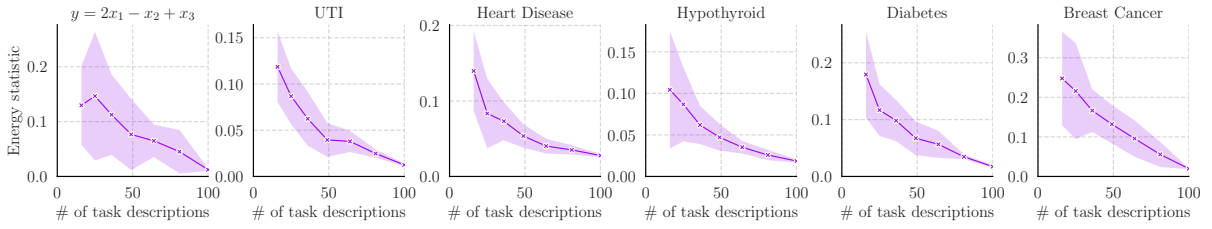


Figure 6.8: Measuring how similar the GPT-3.5-turbo-elicited prior distributions are with fewer task descriptions. We reduce the number of task descriptions and calculate the energy statistic between 10,000 samples of the prior elicited from the language model using the reduced set of descriptions and all of the 100 previous task descriptions. We repeat each experiment 10 times and show the mean and standard deviation.

to the other tasks. In this case, it is likely that the individual components of the mixture-based prior distribution varied significantly. This meant that subsets of the 100 components were relatively more different from the mixture of 100 components than for the other datasets.

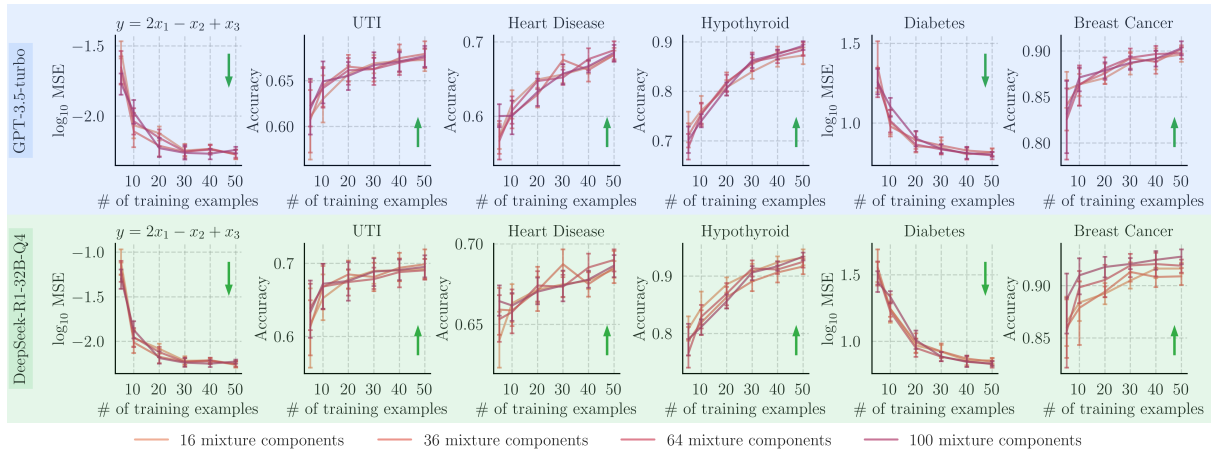


Figure 6.9: AutoElicit with different numbers of mixture components. This figure presents the posterior accuracy or mean squared error of a linear model after observing the given number of training data points, over 10 repeats. The rows of the figure correspond to the different language models tested, and the colour of the lines corresponds to the number of mixture components in the prior.

To supplement this, we calculate the posterior performance with varied numbers of mixture components in our prior, shown in Figure 6.9. We see little variation (except for Breast Cancer with DeepSeek-R1-32B-Q4) in the accuracy or mean squared error as we increase the number of components in the prior, suggesting our framework is robust to the number of task descriptions. We hypothesise that a greater number of mixture components makes the framework more resilient to LLMs that frequently hallucinate, especially on tasks with large numbers of features (such as Breast Cancer).

Figure 6.8 showed that the density of the priors changed with the number of mixture components. However, this has a small effect on the posterior performance. This is likely because the majority of elicited components cover the sample space that provides good posterior performance, whilst the large

changes in Figure 6.8 are due to more unique components.

Cost of AutoElicit Priors

The priors elicited from GPT-3.5-turbo and GPT-4-turbo incur a cost, as we use OpenAI’s API. To elicit all of the priors used in this section, we were charged \$0.41 (on average \$0.068 per dataset) for GPT-3.5-turbo and \$7.56 (on average \$1.26 per dataset) for GPT-4-turbo. As a rule, the more features a dataset contains or the longer the task description, the more expensive priors are to elicit since the number of input and output tokens is greater. However, we did not make use of the batch API and instead made each API call sequentially. All experiments in this work could be implemented using the batch API since no new API call depends on a previous API result, which may significantly reduce costs⁷.

Eliciting the DeepSeek-R1-32B-Q4-based prior distributions incurred no API cost, because the model is open-source and we were able to run it locally. However, to do this, we used an Nvidia A100 (80GB VRAM), which will likely not be available in a low-resource setting. The API costs for the non-quantised model will apply in these cases⁸.

Since there are numerous LLMs hosted by different vendors, it is likely that our framework can be used for a range of budgets. However, as we will see, each LLM provides priors with differing posterior performance. The Bayes factor should therefore be used to confirm an elicited prior is better than an uninformative prior given observed data.

6.10.5 AutoElicit with Other Language Models

In this section, we will compare the results presented in Figure 6.4 with other commonly used language models. We start by exploring the performance of some other OpenAI models, before considering some of the open-source Llama and Qwen models.

Other OpenAI Models

We compare the results of priors elicited from GPT-3.5-turbo, DeepSeek-R1-32B-Q4 (open-source), and GPT-4-turbo (which incurs a significant cost) with GPT-4o-mini (which is less expensive). To elicit all

⁷<https://openai.com/api/pricing/>

⁸https://api-docs.deepseek.com/quick_start/pricing

priors presented here, GPT-4o-mini cost \$0.32 (on average \$0.053 per dataset), whilst GPT-4-turbo cost \$7.56 (on average \$1.26 per dataset) and GPT-3.5-turbo cost \$0.41 (on average \$0.068 per dataset).

Figure 6.10 shows that GPT-4o-mini often produces prior distributions that are equal to or worse than GPT-3.5-turbo in terms of their posterior performance in all tasks except one. Further, for the synthetic, Diabetes, and Heart Disease prediction tasks, GPT-4o-mini provides prior distributions that achieve similar or worse performance to the *uninformative* prior. In all tasks except Diabetes progression prediction, DeepSeek-R1-32B-Q4 provides better accuracy than GPT-4o-mini.

However, for the UTI and Breast Cancer predictive tasks, GPT-4o-mini achieves posterior accuracy similar to that of GPT-3.5-turbo at a lower price point. Surprisingly, on the Hypothyroid dataset GPT-4o-mini provides prior distributions that achieve greater accuracy than those from GPT-3.5-turbo and GPT-4-turbo for all training sizes, whilst requiring significantly less cost.

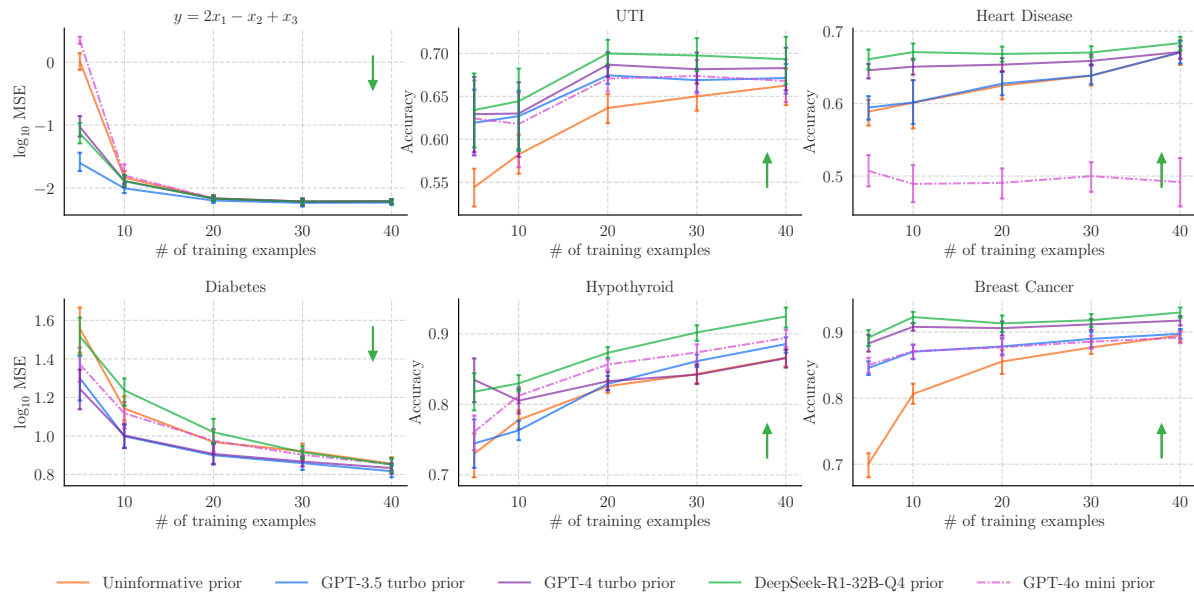


Figure 6.10: Test accuracy of the posterior distribution for some OpenAI language models. Here we show the posterior results when using priors elicited from two other OpenAI language models and an uninformative prior $\mathcal{N}(\theta|0, 1)$. These results are calculated on a test set of the datasets after each model is trained on the given number of data points. The values are the result of 5000 sampled parameters for each of 5 chains, and 10 random splits of the training and testing data. For the regression tasks, we report the MSE on log-scale. This figure shows the line plot of the average mean posterior accuracy or MSE over the 10 splits, with the error bars representing the 95% confidence interval. The green arrows point in the direction of improvement in the metric.

Open-Source Language Models

It is also important to consider the posterior performance of priors elicited from some open-source models, which can be used without an internet connection. Here, we explore the priors elicited from three Llama [135, 252, 253] models, one Qwen model [254, 255], and one DeepSeek model [256] to those elicited from the OpenAI models presented previously:

- DeepSeek-R1-32B-Q4⁹, a 32 billion parameter model from DeepSeek based on Qwen 32B and distilled from DeepSeek-R1.
- Llama-3.2-3B-Instruct¹⁰, a 3 billion parameter model from Meta’s latest collection of v3.2 Llama models.
- Llama-3.1-8B-Instruct¹¹, an 8 billion parameter model from Meta’s v3.1 Llama collection.
- Llama-3.1-70B-Instruct-BF16¹², a bfloat16 quantisation of Llama-3.1-70B-Instruct, a 70 billion parameter model from Meta’s v3.1 Llama collection.
- Qwen-2.5-14B-Instruct¹³, a 14 billion parameter model from Alibaba Cloud’s v2.5 Qwen collection.

In contrast to the OpenAI models, these LLMs are publicly available and open-source. This means that they can be used without the need for connecting to an online API, and when locally hosted, all data and inference can be performed on-device, enabling the use of our framework within a closed and secure environment.

Figure 6.11 highlights the inconsistent posterior accuracy of the priors elicited from the Llama family of models. The 3B and 8B variants are significantly smaller in capacity to those in Section 6.10.4; justifying their inconsistent elicited priors. However, in some of the datasets, they still produce good posterior performance.

Specifically for Breast Cancer, the 3B parameter Llama model provides an intermediate accuracy between an uninformative prior and those elicited from GPT-3.5-turbo, and for the synthetic and Hypothyroid tasks, the 8B parameter Llama provides priors of equal quality to GPT-3.5-turbo. However, for the UTI

⁹<https://ollama.com/library/deepseek-r1:32b>

¹⁰<https://huggingface.co/meta-llama/llama-3.2-3b-instruct>

¹¹<https://huggingface.co/meta-llama/llama-3.1-8b-instruct>

¹²<https://huggingface.co/meta-llama/llama-3.1-70b-instruct>

¹³<https://huggingface.co/qwen/qwen2.5-14b-instruct>

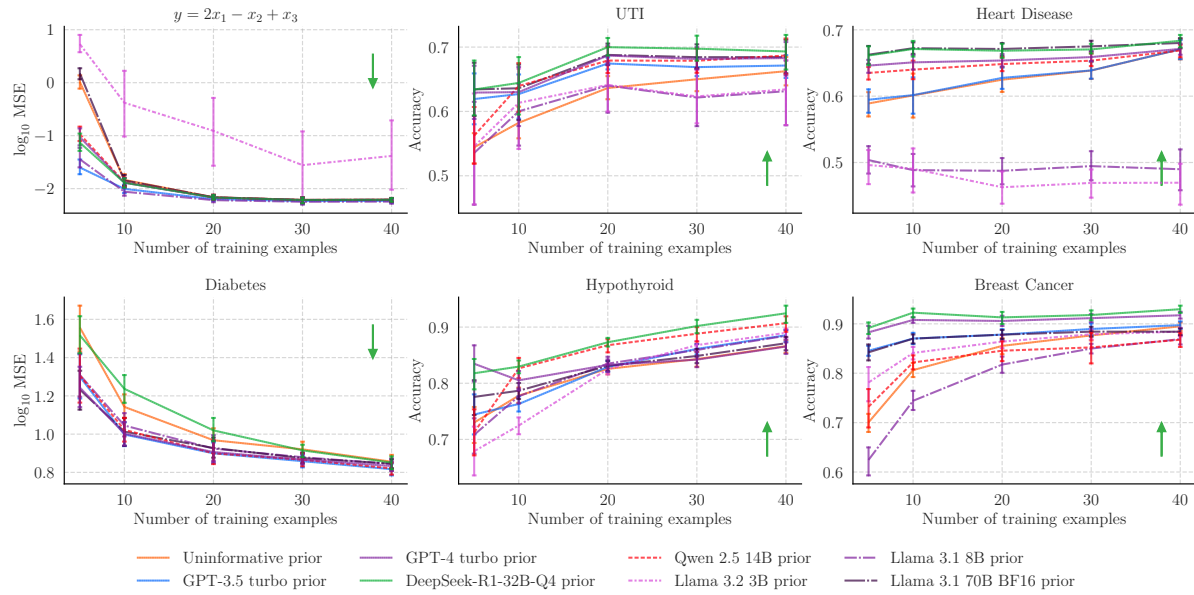


Figure 6.11: Test accuracy of the posterior distribution for open-source language models. Here, we show the results when using priors elicited from two of the Llama language models and an uninformative prior $\mathcal{N}(\theta|0, 1)$ on a linear model. These results are calculated on a test set of the datasets after each model is trained on the given number of data points. The values are the result of 5000 sampled parameters for each of 5 chains, and 10 random splits of the training and testing data. For the regression tasks, we report the MSE on log-scale. This figure shows the line plot of the average mean posterior accuracy or MSE over the 10 splits, with the error bars representing the 95% confidence interval. The green arrows point in the direction of improvement in the metric.

and Heart Disease datasets, both the 3B and 8B Llama models provide prior distributions that are of worse quality than even the uninformative prior. It is possibly not surprising that Llama-3.2-3B-Instruct and Llama-3.1-8B-Instruct provided priors that were equal or worse than Llama-3.1-70B-Instruct-BF16 in all cases except for the synthetic task, since these models are significantly smaller in size.

Llama-3.1-70B-Instruct-BF16 is more competitive, sometimes providing priors that allow for better posterior distributions than GPT-3.5-turbo, but on other occasions, providing priors with posterior performance similar to the uninformative prior. On the UTI and Heart Disease datasets, the prior distributions elicited from Llama-3.1-70B-Instruct-BF16 provide better posterior accuracy than those from GPT-3.5-turbo and comparable values to GPT-4 turbo or DeepSeek-R1-32B-Q4. Surprisingly, on Heart Disease prediction in particular, priors from the 70B Llama model provided a posterior accuracy greater than GPT-4-turbo and equal to DeepSeek-R1-32B-Q4. Additionally, when predicting on the Breast Cancer task, priors from Llama-3.1-70B-Instruct-BF16 achieved the same posterior accuracy as GPT-3.5-turbo for all of the training sizes tested. These three datasets suggest that Llama-3.1-70B-Instruct-BF16 can be a useful model for eliciting prior distributions for predictive tasks.

Qwen-2.5-14B-Instruct produces priors with similarly inconsistent posterior performance. For Heart Disease and Hypothyroid prediction, it achieves a competitive posterior accuracy, which is surprising given its size. For the UTI task, after seeing 10 demonstrations, it performs comparably to Llama-3.1-70B-Instruct-BF16 and GPT-4-turbo, and better than GPT-3.5-turbo. However, on the Breast Cancer and synthetic tasks, it performs worse than GPT-3.5-turbo, but is likely smaller in capacity. Overall, Qwen-2.5-14B-Instruct provides inconsistent performance across tasks, but has the benefit of performing similarly to Llama-3.1-70B-Instruct-BF16 on many tasks with a significantly smaller compute requirement.

The fact that DeepSeek-R1-32B-Q4 is an intermediate size between Qwen-2.5-14B-Instruct and Llama-3.1-70B-Instruct-BF16, and provides considerably better performance with a higher quantisation is a testament to the power of recent advances in reasoning [81] and test-time scaling [257].

Therefore, it is important, given some elicited prior distributions, to calculate the prior likelihood with some already collected data to understand which prior is most appropriate for a task.

6.10.6 Choosing Between Priors

As highlighted, the prior distributions elicited from language models can vary. We consistently find that DeepSeek-R1-32B-Q4, GPT-3.5-turbo and GPT-4-turbo provide better prior distributions than an uninformative prior; however, this may not be the case for all datasets or language models. Therefore, given prior distributions elicited from different language models, we would like to choose the best one for our predictive task.

For this, given some collected data, we can use the Bayes factor as a model selection technique to decide between the prior distributions elicited from different language models (Section 6.8).

Furthermore, by using a temperature parameter, it is also possible to control the strength of the elicited prior, which can indicate our trust in the distribution.

6.10.7 Can Experts Supplement the Process?

An expert interacting with the language model might be able to provide further information about a task that improves the elicited priors. Therefore, it is important to study the extent to which task descriptions affect the elicited priors.

To first understand how the detail of the task description affects the performance of the priors elicited from LLMs, we study the synthetic task $y = 2x_1 - x_2 + x_3$ and describe the relationship between features and targets in increasing detail. The posterior performance of these priors is given in Figure 6.12.

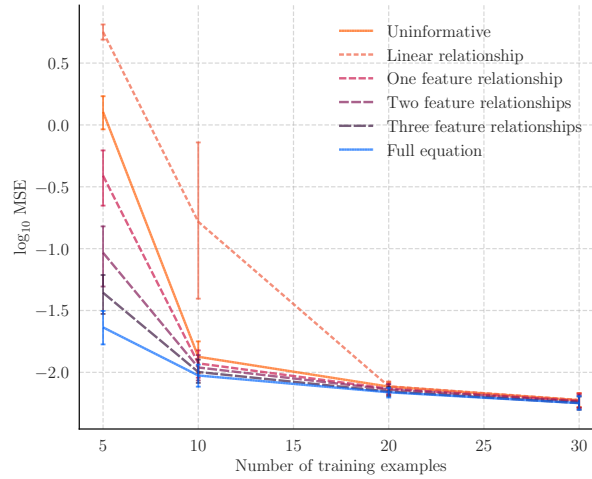


Figure 6.12: MSE using the GPT-3.5-turbo-elicited priors for the task: $y = 2x_1 - x_2 + x_3$ where we have used increasingly more detailed task descriptions. Here, feature values are generated using $X \sim \mathcal{U}(-5, 5)$ and the targets are calculated using $y = 2x_1 - x_2 + x_3$. Within each task description, we provide the language model with increasingly more information about the task to test how the descriptions affect the priors elicited. The lines depict the average mean posterior MSE over the 5 dataset splits, whilst the error bars represent the 95% confidence interval.

For each of the levels of detail shown here, we included the following text in the task description:

- **Linear relationship:**

“The target is linear in features”

- **One feature relationship:**

“The target is a linear combination of the features and that when ‘feature 0’ increases by 1, the target increases by 2”

- **Two feature relationships:**

“The target is a linear combination of the features and that when ‘feature 0’ increases by 1, the target increases by 2, and when ‘feature 1’ increases by 1, the target decreases by 1”

- **Three feature relationships:**

“The target is a linear combination of the features and that when ‘feature 0’ increases by 1, the target increases by 2, and when ‘feature 1’ increases by 1, the target decreases by 1, and when ‘feature 2’ increases by 1, the target increases by 1”

- **Full equation:**

“The ‘target’ = 2 * ‘feature 0’ - 1 * ‘feature 1’ + 1 * ‘feature 2’”

For each prior elicited using these levels of detail, in Figure 6.12, the corresponding $\log_{10}$ MSE on the test set can be seen for a linear regressor after observing the given number of data points. This figure illustrates the effects of varied descriptions and shows that the improvements made to the MSE diminish as the task description contains more detail. The biggest improvement comes from providing the language model with a single feature relationship – enabling an MSE that is two orders of magnitude better. This suggests that it is most helpful to provide more detail when the language model initially knows very little about a task.

Providing the language model with improved levels of detail allows it to produce prior distributions that enable our linear regression model to achieve a given error rate with fewer data points. As we might expect, the more detail provided in the task description, the better our prior distributions are.

However, it is unexpected that using an uninformative prior ($\theta \sim \mathcal{N}(\theta|0, 1)$) provides better posterior performance than specifying to the language model that the relationship between the target and features is linear (Figure 6.12). This is explained by inspecting the prior distribution elicited when given this minimal information, which shows that the language model responded with a similar mean and standard deviation for a Gaussian for each of the 100 variations of the prompts:

- Feature 0: mean: 0.5 ± 0.11 , std: 0.12 ± 0.05
- Feature 1: mean: -0.32 ± 0.08 , std: 0.08 ± 0.06
- Feature 2: mean: 0.56 ± 0.26 , std: 0.14 ± 0.09

This demonstrates the sometimes unexpected behaviour of language models. Here, the LLM produced Gaussians for the prior distribution with small standard deviations (reflecting a high confidence in the distribution) and means that imply the language model knew the underlying equation defining the relationship between the target and features. We might have expected that after only being told the relationship was linear, the language model would produce Gaussians with 0 mean and standard deviation 1, reflecting the uncertainty in the exact relationship between each feature and the target.

Interestingly, although “Three feature relationships” and “Full equation” contain the same information written differently, the language model provided better prior distributions when given the full equation.

Fully specifying the equation is not realistic in a real-world setting where we are unsure about the exact relationship between features and targets, but this provides an example of the benefit of making the task description as information-dense as possible.

In Figure 6.13, we show the parameter distributions elicited from the language model for the different descriptions we provided. As we supplied additional information on each feature, the language model responds by generating a prior distribution that reflects its updated knowledge of the task. This demonstrates the potential of using an expert to construct more detailed task descriptions, which the language model can translate into parameter distributions. This experiment demonstrates that in a controlled setting a language model can translate natural language into increasingly more informative priors, and also motivates the use of our framework within traditional prior elicitation sessions.

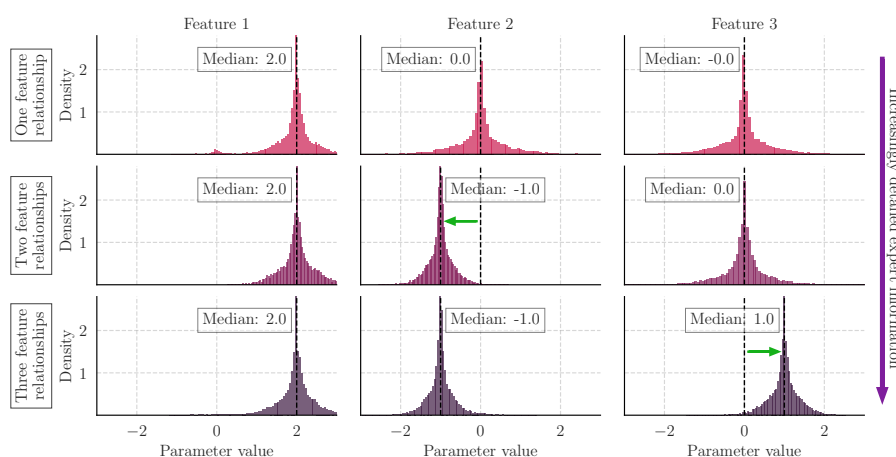


Figure 6.13: GPT-3.5-turbo-elicited parameter distribution as increasingly more information is included. These histograms show the distribution of parameter values for the features as more expert information is provided. In each row, we provide information in the task description of an additional feature, with the green arrow indicating the effect this has on the parameter distribution. In particular, in the first row we provide information about the relationship between the target and the first feature, in the second row we provide information about the second feature, and the in the third row we provide information about the third feature.

We also studied this on the UTI predictive task, where we list three features that are positively correlated with the label [1]: “It is known that the more previous UTIs a patient had, the more likely they are to have a UTI in the future. Someone with a UTI will also wake up more during the night. Someone with a UTI will have a greater bathroom usage.”

Figure 6.14 shows that this information is incorporated when LLMs provide priors, as these three features become more positively predictive of a UTI. GPT-3.5-turbo had already identified this relationship, shown by the positive median of the priors, which explains the small improvement in posterior accuracy this additional information provides. This demonstrates that including expert information in the task

description allows AutoElicit priors to combine learnt insights with provided domain knowledge.

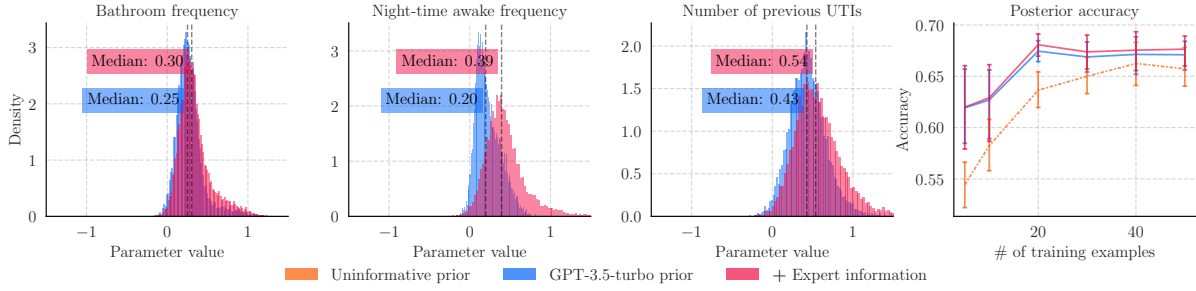


Figure 6.14: UTI prior parameters with expert information. The histograms in the three left hand plots show the distribution of parameter values for the features that we provide expert information for in the task description. In all three cases, we state that the features are positively correlated with UTI risk. In the right-hand plot, we show the posterior accuracy of these distributions for different numbers of observed samples. The average and 95% confidence interval of the mean posterior accuracy over the 10 test splits are shown.

Therefore, AutoElicit can also be used to supplement typical prior elicitation methods, with the benefit that experts can convey their knowledge as natural language.

6.10.8 The In-Context Priors and Posteriors

To compare in-context learning to AutoElicit in detail, we extracted the prior and posterior parameter distributions used by the LLM to make predictions. This is done by sampling a maximum likelihood estimator as described in Section 6.7.1.

To approximate the in-context prior, for each of the tasks studied, we randomly generate 5 sets of 25 feature values from $X \sim \mathcal{U}[-5, 5]$ and ask the language model to predict their regression label or positive classification probability based on the task description given and with the specified model class (in our case, a linear or logistic regressor). This is done for each of the 100 task descriptions that we have for the dataset. We can then fit a linear regressor to the predicted regression labels or the classification logits of the predicted probabilities¹⁴, using MLE. The parameters of this linear model are then taken as a single sample of the approximated in-context predictive model. When approximating the in-context posterior parameters, we provide the LLM with 25 demonstration examples alongside the task description.

For each of these linear model samples, we can calculate the regression labels or the classification logits using the same data given to the language model for in-context predictions. This allows us to calculate an error in our estimation of the in-context model, which depends on how well the in-context learner is

¹⁴Calculated using the inverse logistic function: $\text{logits} = \ln p / (1 - p)$

approximating a linear model.

In total, for each task description, we collect 5 samples of the in-context model parameters. Once this is repeated for all of the 100 task descriptions, we have 500 samples of the in-context model.

Figures 6.15 and 6.16 show GPT-3.5-turbo’s in-context prior and posterior distributions approximated with MLE. To quantify the error, we calculate the MSE between the LLM’s in-context predictions and the approximated model’s predictions, shown in the bottom row of Figures 6.15 and 6.16. This allows us to study the implicit parameters used by the LLM to make predictions in-context.

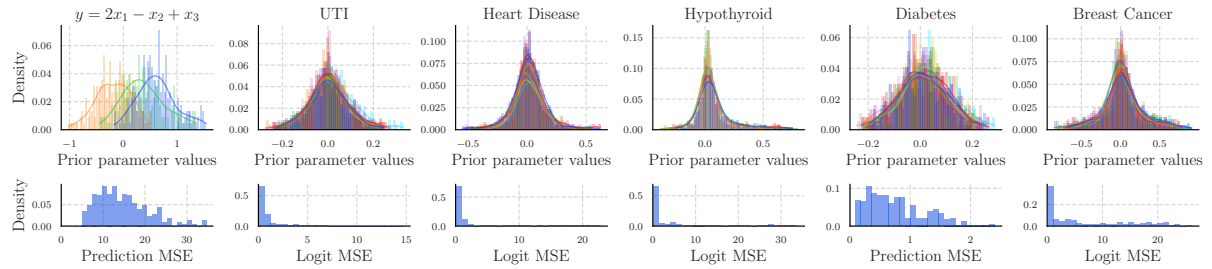


Figure 6.15: Extraction of GPT-3.5-turbo’s in-context prior. The bottom row shows the MSE between the LLM’s in-context predictions and the MLE model’s predictions, whilst the top row shows the parameter distribution of these MLE models, with each colour representing a single feature. For ease of visualisation, we exclude values outside of the (2.5%, 97.5%) percentiles and the bias term.

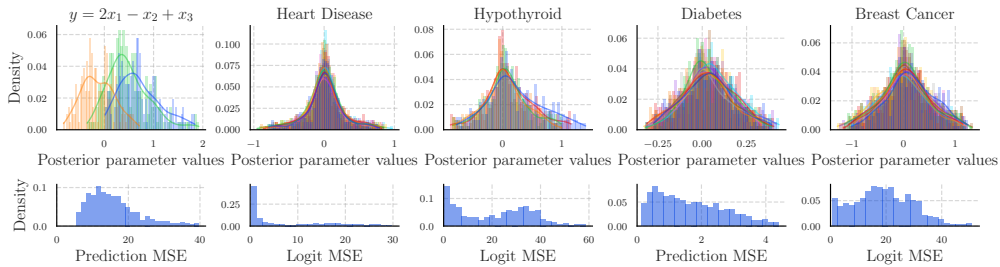


Figure 6.16: Extraction of GPT-3.5-turbo’s in-context posterior. The bottom row shows the MSE between the LLM’s in-context predictions and the MLE model’s predictions, whilst the top row shows the parameter distribution of these MLE models, with each colour representing a single feature. For ease of visualisation, we exclude values outside of the (2.5%, 97.5%) percentiles and the bias term.

The estimated priors in Figure 6.15 show variations in their values across the datasets, suggesting the LLM uses its own knowledge in-context before observing demonstrations. The low MSE on the real-world tasks indicates that the LLM was reliably imitating a linear model, allowing us to achieve a low approximation error. However, the priors are not particularly informative, as they are often centred close to 0.

The estimated in-context posteriors in Figure 6.16 differ from the priors, suggesting the LLM has updated its knowledge based on demonstrations. For the Diabetes and Heart Disease tasks, the LLM mostly continued to produce linear predictions, and so our error increases, but remains low. However, the Breast

Cancer and Hypothyroid dataset produced a large approximation error, as the LLM struggled to imitate a linear model. We do not estimate the in-context posterior for the UTI dataset as it is private, and so we are unable to provide demonstrations.

This approximation of the LLM’s in-context parameter distribution gives us new visibility into its predictive process. In general, we better approximate the LLM’s in-context prior than posterior. This is likely due to limitations in the LLM’s capacity to interpret demonstrations for training alongside instructions to provide linear predictions in the same prompt. As the abilities of LLMs improve, we expect these approximation errors to reduce, as they are due to the language model’s inconsistent imitation of a linear model.

Does the LLM Use the Model Class We Specify?

When extracting the internal parameters of the in-context model, we assume that the language model is applying the model class that we specify in the task description.

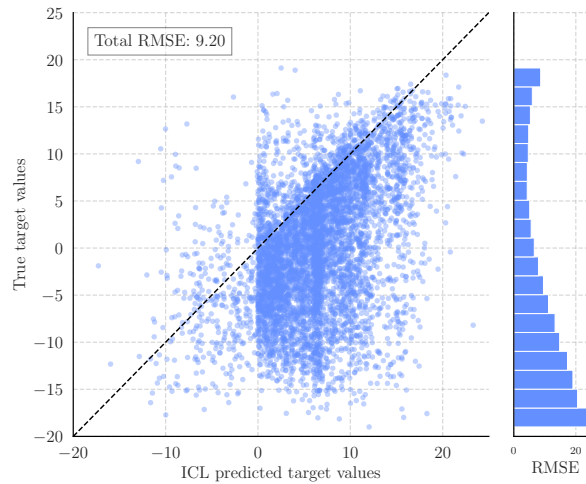


Figure 6.17: GPT-3.5-turbo in-context predictions for the task: $y = 2x_1 - x_2 + x_3$ compared to the true values. Here, feature values are generated using $X \sim \mathcal{U}(-5, 5)$ and the targets are calculated using $y = 2x_1 - x_2 + x_3$, which is specified fully in the in-context prompt. The language model is then asked to make predictions on X . This shows the true targets y against those predicted by the language model, as well as the binned root MSE.

However, we see that for the synthetic task, the language model does not use the same model in context as the one we specify. This is shown by the high approximation error for the synthetic task in Figure 6.15. We can further understand this by calculating the MSE between the true y values and those returned by the language model’s in-context predictions when we ask it to mimic $y = 2x_1 - x_2 + x_3$.

In Figure 6.17, we show the true targets against the in-context predictions. Interestingly, even given the simple linear equation that we ask the language model to mimic, we find that it struggles to provide good

predictions in-context. This suggests that the language model is not always using the model class for in-context predictions that we specify in the task description, indicating that in its current state, in-context learning might be inadequate for predictive tasks with primarily numerical features. This figure also shows that the language model was hesitant to predict negative values and preferred positive predictions – a strange behaviour that clearly restricts its use as a simulator of a linear model.

However, for all other prior extraction tasks presented in Section 6.10.8, we find that the language model uses a linear model in-context, demonstrated by the low MSE loss values between the predictions of our MLE samples and the true in-context predictions, shown in Figure 6.15. This is often not the case when the language model is presented with demonstration samples, as in the posterior extraction presented in Figure 6.16, in which the language model uses a linear combination of features to predict the targets for the Heart Disease and Diabetes tasks only.

6.10.9 Is the Language Model Consistent?

Here, we compare these in-context priors with the AutoElicit priors. This is done by sampling 10,000 parameter values from the elicited distributions and calculating the energy statistic between this sample and the extracted in-context sample. We can also compare the histograms of the parameter values using both methods. By doing this, we test whether the language model is using the same distribution as we elicit using AutoElicit when it makes predictions in-context.

Table 6.3 suggests the LLM uses different priors during in-context learning compared to those we elicit, since they differ according to their energy statistic.

Table 6.3: Approximated GPT-3.5-turbo in-context prior compared to the elicited prior. The energy statistic between the LLM’s approximate in-context prior, and the LLM-elicited prior. Since we extract a single prior distribution for both methods, no confidence intervals are given here. Lower values indicate that the two distributions are more similar.

UTI	Heart Disease	Hypothyroid	Diabetes	Breast Cancer
0.39	0.18	0.15	0.38	0.31

This is supported by Figure 6.18, which presents histograms of the parameter values. Notable are the large differences between the elicited prior distributions and the ones used when performing in-context learning, which supports the findings in Section 6.10.9. In the in-context case, the prior distributions often have modes close to 0, suggesting that the internal model is not sure about the relationship between

features and targets. In contrast, the elicited priors often represent a stronger belief in the correlation between features and targets, with most histograms achieving modes greater than or less than 0.

Interestingly, on the Breast Cancer dataset, prior elicitation resulted in two distinct modes in the prior distribution for Feature 2 (“mean texture”), which could represent uncertainty in the direction of correlation.

Furthermore, the prior distributions on the parameters for the Heart Disease and Hypothyroid datasets have more variation in their values in comparison to that extracted from the language model’s internal model. This tells us that the elicited prior parameters represent more uncertainty in their value and are therefore more flexible when calculating the posterior on observed data.

Either the distribution we elicit is an unreliable representation of the LLM’s knowledge, or it is unable to use this knowledge when making predictions in-context. As in-context priors are often centred around 0, we suspect it is the latter, and that prior elicitation more effectively makes use of the LLM’s background knowledge.

6.10.10 Does In-Context Learning Use Bayesian Inference?

It is important that in-context learning performs Bayesian inference so that new data updates the LLM’s knowledge reliably. We empirically study this by comparing the approximated in-context posterior with the Monte Carlo (MC) posterior samples of the approximated in-context prior.

When extracting the posterior distribution over the in-context learner, we repeat the process 5 times, and in each repeat, provide the language model with 25 demonstration points from the corresponding dataset. We then fit a Gaussian kernel density estimator (with a bandwidth factor of 0.25) to our in-context model prior parameter samples. We then use the No-U-Turn sampler [246], with 1000 adaptation steps and a target acceptance rate of 0.8, and the same demonstration examples given during in-context learning to sample posterior parameters. Here, we use 100 chains of 10,000 samples to build an approximate posterior distribution.

The energy statistic is then used to measure the difference between the extracted in-context posterior distribution and the Monte Carlo approximation of the posterior based on the extracted in-context prior distribution. Table 6.4 shows the mean and standard deviation of the energy distance between these distributions, for all of the datasets tested, including those that have a high approximation error when we estimate the in-context posterior.

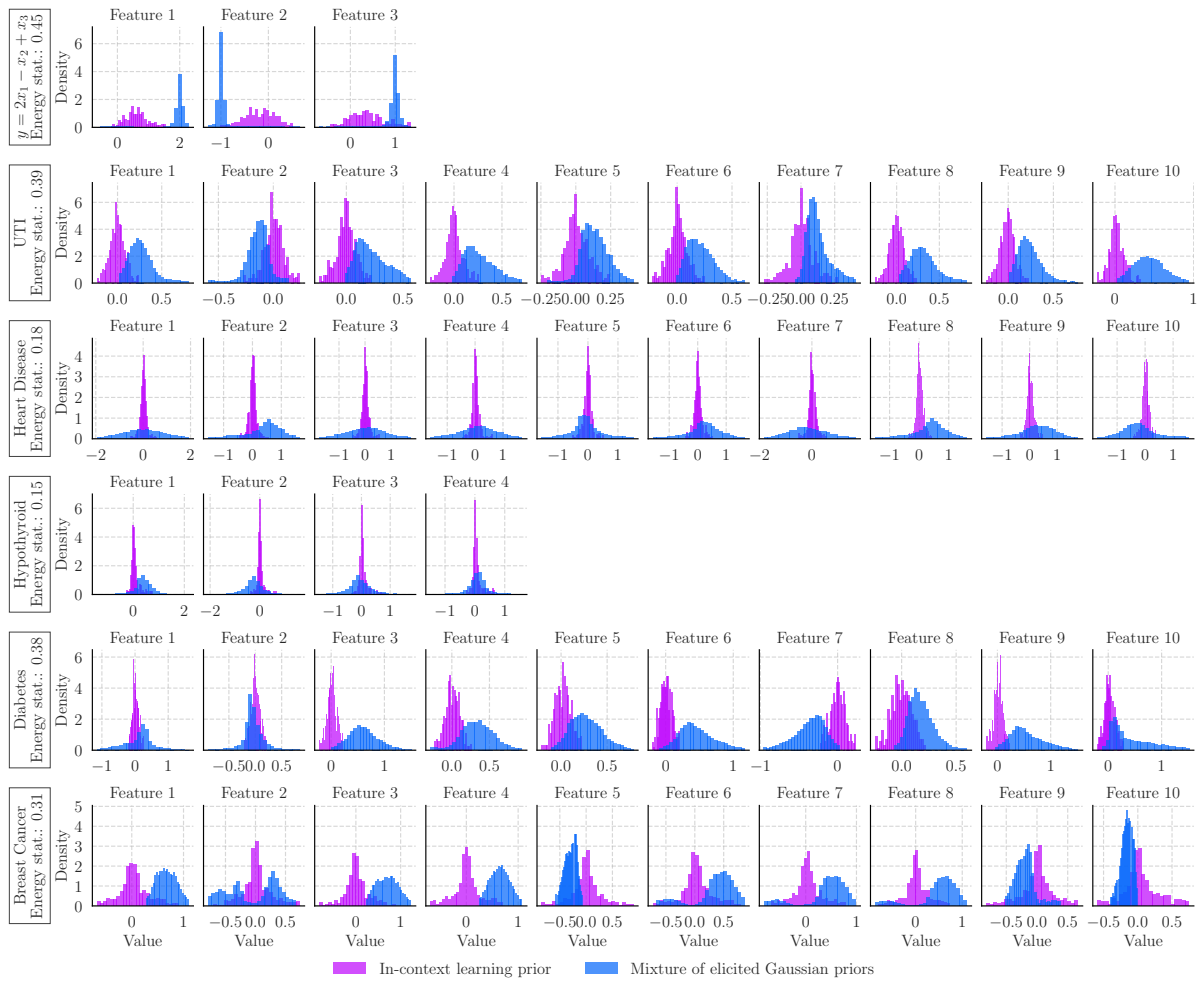


Figure 6.18: Distribution of the prior parameters under elicitation and those approximated from the internal model for the different predictive tasks for GPT-3.5-turbo. Here, we compare the prior distributions of the different predictive tasks, as defined by the internal model used in-context, and that which we elicit from the language model. Here, we draw the histogram separately in each dimension, but calculate the energy distance across all features combined.

In Figure 6.19, we show the posterior samples for those datasets where in-context learning produced linear predictions, providing a low posterior approximation error. For the Diabetes task, samples from the approximated in-context posterior and MC sampling on the approximated in-context prior visually agree and produce a low energy statistic, suggesting the LLM is performing Bayesian inference in-context. However, for the Heart Disease task, the two posterior samples differ, showing that the LLM inconsistently applies the Bayes rule in-context. This suggests LLM-elicited priors are more reliable, particularly when access to the posterior distribution is important, such as for uncertainty estimation.

In addition, Figure 6.20 shows the distribution of these posterior parameters for the datasets where our approximation of the in-context posterior distribution is unreliable; since here the LLM failed to return linear predictions.

Table 6.4: Difference between approximated posterior and MC on the approximated prior for GPT-3.5-turbo. The energy statistic between the language model’s internal posterior extracted through MLE sampling and the posterior samples from applying Monte Carlo techniques to the language model’s internal prior. Values are $\in [0, 1]$ with values closer to zero, indicating that the two distributions are more similar. Here, we show the mean and standard deviation of the energy statistic for the 5 different training sets tested as observed data.

Heart Disease	Hypothyroid	Diabetes	Breast Cancer
0.04 ± 0.01	0.06 ± 0.01	0.02 ± 0.00	0.01 ± 0.00

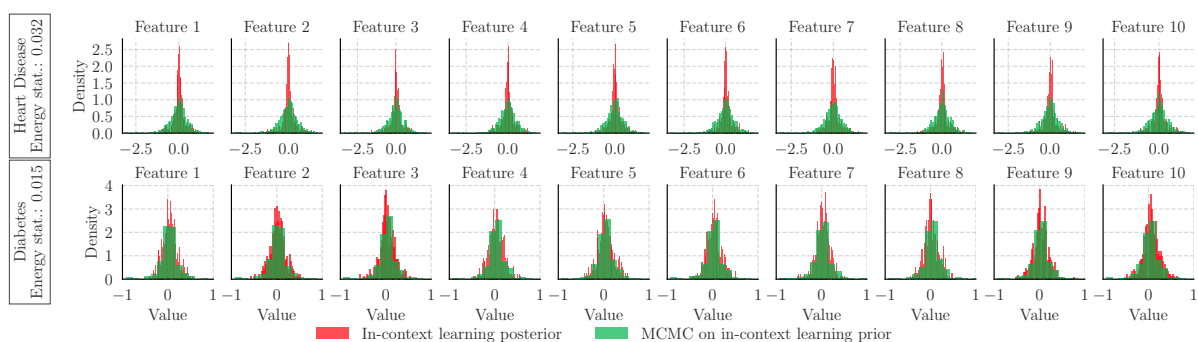


Figure 6.19: The in-context posterior distribution for the tasks with low errors. We compare the posterior extracted from GPT-3.5-turbo’s in-context predictions against the distribution that we sample using Monte Carlo methods on the extracted prior distribution. Both posterior distributions are based on the same 25 data point sample from the corresponding dataset. We draw the histogram separately in each dimension, but calculate the energy distance across all features combined. For the diabetes dataset we have removed the long tails for improved visualisation.

For the Breast Cancer dataset, these two distributions match closely and show a strong agreement in their parameter values, suggesting that for this dataset, in-context learning could be performing Bayesian inference. On the other hand, for the Hypothyroid predictive task, the parameter distributions differed significantly, with the in-context prior having a less varied distribution. This is reflected in the energy distance, which is the largest of all of the datasets tested. These comparisons must be made with the approximation error in mind, presented in Section 6.10.8, which suggests that the posterior in-context distributions for the Breast Cancer and Hypothyroid tasks are not reliable.

However, it is feasible that future LLMs can reliably follow instructions to mimic a linear model and be able to perform Bayesian inference in-context, requiring a model selection strategy to decide between in-context learning and LLM-elicited priors.

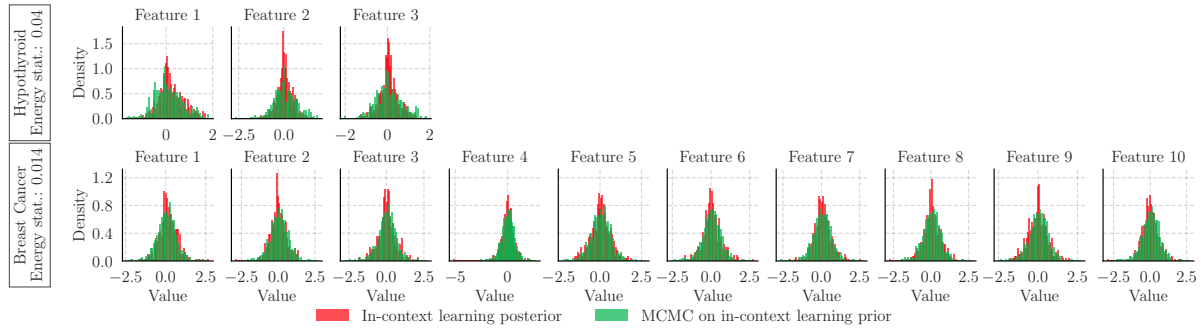


Figure 6.20: The different posterior samples for GPT-3.5-turbo for the tasks with high errors. Here, we compare the distribution of the posterior extracted from the language model’s in-context predictions against the distribution that we sample using Monte Carlo methods on the extracted prior distribution. Both posterior distributions are based on the same 25 data point sample from the corresponding dataset. We draw the histogram separately in each dimension, but calculate the energy distance across all features combined.

6.10.11 Using Bayes Factor for Model Selection

Assuming that in-context learning approximates Bayesian inference, we can calculate the Bayes factor between it and the AutoElicit priors.

For a single dataset, we start by randomly selecting 25 data points 5 times, and consider the prior predictive distribution. For AutoElicit (method α_0 in Equation 6.6), we approximate the marginal likelihood by sampling the prior predictive log-likelihood for these data points. In particular, we sample 500 parameters for our linear model and make predictions on the 25 data points to get samples of $p_{M,T}(\mathcal{D}|\alpha_0)$. To sample the posterior distribution over the parameters of the linear model, we use the No-U-Turn sampler [246]. We start by performing 1000 adaptation steps to tune the step size with a target acceptance rate of 0.8, then generate 5000 linear model parameter values for each of the 5 chains. The posterior linear model parameters are then used to make predictions on each of the 25 data points, over 5 splits of the datasets, which are used to calculate the likelihood and subsequently, the mean and standard deviation in the Bayes factor.

For in-context learning (method α_1 in Equation 6.6), we can approximate $p_{M,T}(\mathcal{D}|\alpha_1)$, since the LLM provides predictions on features directly. We prompt the LLM with 100 task descriptions and ask it to predict 25 data points from the dataset. This is repeated over 5 splits of the datasets to produce the mean and standard deviation in the Bayes factor.

For both sets of prior predictive samples, we calculate their log-likelihood using either cross-entropy loss or MSE, depending on whether the task is classification or regression. For each dataset, test split, and

method, we calculate the mean of this log-likelihood over the samples. With these values, we measure the difference between the value for the elicited prior predictions and the in-context predictions to obtain the log Bayes factor.

We then report the mean and standard deviation of the Bayes factor over the splits in Figure 6.21, which helps us to decide, given a dataset, which method is the most appropriate, assuming that in-context learning approximates Bayesian inference.

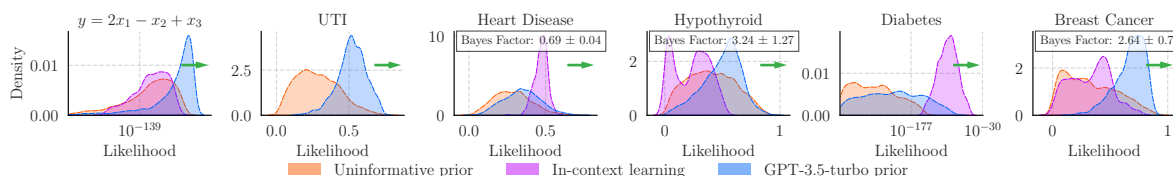


Figure 6.21: The distribution of the prior marginal likelihood and the Bayes factor for GPT-3.5-turbo. This shows the distribution of prior likelihoods, calculated using 25 data points from each dataset, over 5 test splits and 100 samples of prompts. The Bayes factor is calculated between prior elicitation and in-context learning, with a number greater than 1 favouring prior elicitation. For the regression tasks, we exclude values outside of the (2.5%, 97.5%) percentiles in the histograms.

Figure 6.21 shows the marginal likelihood distribution for in-context learning, AutoElicit, and an uninformative prior along with the Bayes factor (where reliably defined). Unlike in-context learning, AutoElicit priors are the better choice over an uninformative prior for all datasets tested. Whether AutoElicit or in-context learning provides better priors depends on the task. However, in only two of the five datasets tested, in-context learning reliably produced linear predictions, and in only one of those, it approximated Bayesian inference. Considering this, the cost, and the ease of interpretability, AutoElicit is likely the better option.

This model selection strategy is designed to be more relevant as LLMs become more reliable and their cost reduces.

6.11 Using AutoElicit on a New Dataset

To apply AutoElicit techniques to a new dataset and to measure the reliability of in-context learning, we recommend the following steps:

1. First, create a description of the dataset along with a description of the language model's role. As part of this, provide descriptive features and target names. It will be helpful to split this task description into what the language model is expected to do (system role) and the task that the user

is giving it (user role). Please see our collection of task descriptions in the supplementary code or the example given in Section 6.10.4.

2. Ask a language model to rephrase the user role and system role many times. In our experiments, we asked the language model to rephrase both descriptions 10 times for each. We provide code to do this automatically.
3. Take the product of the system roles and user roles and provide each to the language model. For each of the unique system and user role combinations, record the mean and standard deviation of the Gaussian prior elicited from the language model.
4. Use these individual Gaussian priors to build a mixture of Gaussians as a prior distribution for a linear model.
5. Using the data points you have available from the predictive task, sample from the posterior predictive distribution to predict on unseen data.

To measure whether prior elicitation or in-context learning is better for a given task, assuming the language model is approximating Bayesian inference:

1. Firstly, sample the prior predictive distribution, with the elicited priors found above, on the data points already collected.
2. Then, to perform in-context learning, use a similar task description to those above but with a different system and user role that specifies the language model should make predictions instead of providing prior distributions. Again, ask a language model to rephrase this new set of descriptions.
3. To sample the prior predictive distribution, using the product of these descriptions, ask a language model to make predictions on the data directly.
4. Measure the Bayes factor between the likelihood of the prior predictive distribution using prior elicitation and the in-context model on the already collected data.
5. Using the Bayes factor and by factoring in resource cost, decide whether in-context learning or prior elicitation is more appropriate.

To empirically check whether the language model is approximating Bayesian inference to produce its in-context predictions or to test whether it is being faithful to the priors we elicit, we can follow the below procedure:

1. Using the same task descriptions as above, provide the language model with randomly generated feature values (we used 25 samples from $X \sim \mathcal{U}[-5, 5]$) for each task description and ask it to make predictions in-context.
2. By additionally providing the seen data as demonstrations to the language model, again ask the language model to make predictions on randomly generated feature values for each task description.
3. With randomly generated feature values and in-context predictions, use MLE to estimate the underlying linear model for each task description. For the classification case, this is done more easily if the language model is asked to provide the probability of a positive prediction rather than the prediction itself.
4. The resulting linear models approximated from the in-context predictions can now be used to approximate the prior and posterior distribution of the in-context model.

Using the above, we can now (1) empirically test whether the language model is faithful to the prior distributions we elicit and (2) empirically test whether the language model is approximating Bayesian inference in-context.

For the former, we can use a measure of the difference in samples (in our case, the energy distance, Section 6.6) to calculate the difference between the in-context model's prior distribution and the elicited prior distribution. This will inform us of whether the language model is being faithful when it provides us with prior knowledge of the parameters of a linear model or whether the language model uses that prior knowledge when making predictions in context. If the prior distribution elicited differs from the one that the language model is using in-context, then calculating the Bayes factor could be important to ensure that we select the most appropriate predictive model for the given task. If it is not clear whether the language model is approximating Bayesian inference in-context, this should be studied to ensure the Bayes factor is a suitable model selection tool.

For this, we can follow the below procedure:

1. Use a Kernel Density Estimator (KDE) to approximate a smoother version of the prior distribution

over the in-context model. For our experiments, we used a Gaussian kernel density estimator with a bandwidth factor of 0.25.

2. Using the KDE and Monte Carlo methods, given the collected data, we sample a posterior distribution from the in-context learner's prior.
3. Use a method to calculate the difference between two samples of a distribution to understand the differences between the posterior samples calculated through Monte Carlo sampling of the in-context prior and the samples of the in-context posterior.

This last procedure might indicate that our in-context learner is not approximating Bayesian inference and is therefore not suitable for a given task, as it could be important that our predictive model is updating its prior knowledge in a reliable way.

6.12 Discussion

We show that, by asking a language model to play the role of a domain expert, we are able to elicit prior distributions for predictive tasks, potentially reducing the cost of data collection and improving the accuracy of our models when we have access to few labelled examples.

We find the priors elicited from GPT-3.5-turbo, GPT-4-turbo, and DeepSeek-R1-32B-Q4 achieve better accuracy or MSE with fewer data points than an uninformative prior for a variety of tasks. This improvement is most visible when few training examples are available, suggesting its use in environments where label collection is challenging.

Incorporating expert guidance into the task descriptions enables further improvements to predictive accuracy, as the LLM updates elicited distributions as we would expect. Therefore, AutoElicit can also be used to simplify the typical expert prior elicitation process, by only requiring experts to provide insights as natural language. We find that prior elicitation with AutoElicit is an improved alternative to in-context learning in our setting, as we can inspect priors, confidently perform Bayesian inference, and often achieve reduced error.

On our private dataset, we find a large and quantifiable performance improvement when predicting cases of UTI. Within our study on dementia care, the resulting model created with LLM-elicited priors can provide better predictions of adverse health conditions and reliable uncertainty estimates, all whilst remaining

interpretable and trusted by clinicians.

6.13 Summary

This work studied the use of language models to overcome issues that arise from the cost of labelling data. Another challenge facing the deployment of remote healthcare monitoring systems is the difficulty of extracting clinical insights from continuously collected unlabelled data. In the following chapter, we will study how language models can also be used to encode time-series data into features that allow for vector searching, clustering, and measuring similarity.

Chapter 7

Time-series Representation Learning Using Fine-Tuned Language Models

This chapter is based on Capstick et al. (2024) [2] and is licensed under a Creative Commons Attribution 4.0 International License¹.

In Chapter 6, we proposed the use of language models to generate prior distributions for predictive models, improving model performance on tasks with few labelled examples. However, our dataset also contains a considerable number of unlabelled examples, which requires the use of unsupervised machine learning methods.

In this chapter, we focus on using representation learning to extract clinical insights from the unlabelled in-home activity data recorded within the Minder study. Our method converts sequences of in-home activity to text strings which are encoded using a language model. This language model is fine-tuned to transform data from the same participants within a 30-day window into similar embeddings in the vector space, allowing for clustering and vector searching over participants and days, and the identification of activity deviations to aid with personalised delivery of care. We demonstrate the utility of the learnt representations through examples of the clustering of participants, participant similarity analysis, and vector searching.

¹<http://creativecommons.org/licenses/by/4.0/>

7.1 Introduction to the Problem

In remote healthcare monitoring applications, time-series data is continuously collected using wearables or Internet of Things devices. This scale of data collection necessitates machine learning techniques for effective analysis and efficient delivery of care. Additionally, large time-series data can be challenging or infeasible to label [87], suggesting the use of representation learning through self-supervised, semi-supervised, or unsupervised methods to extract insights from unlabelled data. These techniques transform data into useful encodings for downstream tasks and can be performed on datasets with few or no labels [83–86].

Viewing a neural network as a feature extractor followed by a predictor [105], supervised learning can be thought of as a combination of representation learning and predictive modelling. This notion inspires most of the self-supervised or semi-supervised representation learning literature to date, since predictive tasks can be defined using the time-series itself. This leads to representation learning objectives using data masking, shuffling, contrastive learning [258], or a combination of these.

Prior works [189, 190, 194, 195] have studied the use of structured state space models or large language models for numerical-based time-series representation learning. However, these works do not consider that time-series values can have semantic meaning, such as locations in a home.

In this chapter, we focus on building representation learning methods for collections of time-series data containing irregularly appearing discrete values with semantic meaning. For example, this could be a sequence of electronic healthcare records or a sequence of in-home activity, where each new measurement in time is a categorical value. In particular, we designed and evaluated our method for encoding unlabelled in-home activity data, which was collected passively using low-cost sensors from the homes of people with dementia as part of the Minder study.

We present initial results of learnt representations of activity data that benefits from advances in large language modelling, text encoding, and vector search by fine-tuning a language model’s pre-trained embeddings of different in-home locations. Applying these techniques to our dataset, we allow for activity clustering, searching for similar participants and patterns, and identifying changes in activity. This technology can help clinical teams deliver personalised care or assess the risk of morbidities by referencing similar participants with known health trajectories and care requirements.

7.2 The Unlabelled Minder Dataset

The Minder dataset, discussed in Section 2.1, includes in-home activity and sleep data recorded from people living with dementia in their homes. In this chapter, we focus on unlabelled activity data recorded between 2021-07-01 and 2024-01-30, containing 65,962 days in total collected from 134 people. The average number of days recorded per participant is 492 (with a minimum of 5, a median of 513 and a maximum of 943). Figure 7.1 shows the distribution of the number of days recorded from each person living with dementia, and 7.2 shows the sensor recordings by the time of day, illustrating the richness of the dataset.

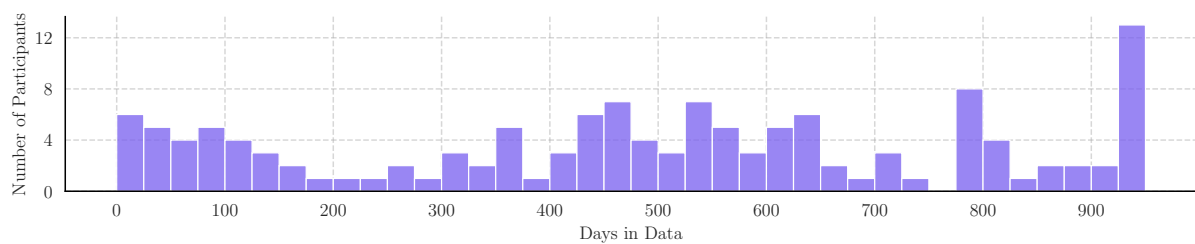


Figure 7.1: Number of days recorded. Histogram showing the number of participants with a given number of days of data recorded.

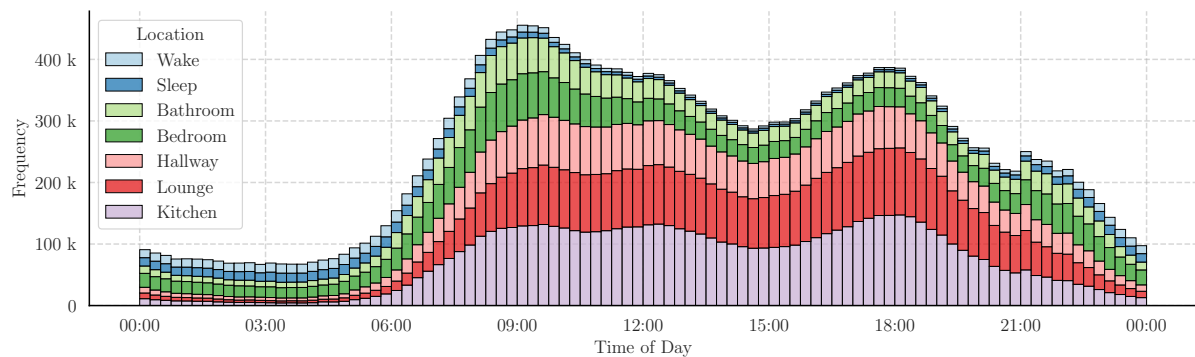


Figure 7.2: Time of location recordings. Histogram showing the number of sensor recordings by time of day, with locations that are ordered by their total frequency.

Each data point contains the participant ID, the time and date of a record (to the nearest second), and the location at which activity was detected or whether the sleep mat recorded someone entering or exiting the bed. This amounted to 24,467,307 data points, shown in Figure 7.2.

As part of the Minder study, a subset of participants opted to regularly provide urine samples, which were analysed for suspicion of a Urinary Tract Infection (UTI). These are the same labels used from our analysis in Chapter 4. When using these labels, we require participants to have at least 1 negative label and 2 positive labels, resulting in 154 days labelled as positive (77 days) or negative (77 days) of a UTI.

7.3 Language Modelling

Language models are typically auto-regressive, trained on a collection of sequences $\mathcal{U} = \{U_1, \dots, U_N\}$, with each sequence $U_i = (u_1, \dots, u_{n_i})$ containing tokens u_i representing a single piece of text from a vocabulary $\mathcal{V}$ [192]. Since each predicted token is dependent on all previous tokens in the sequence, for a model parametrised by θ , we have $\log p(U_i|\theta) = \sum_{j=1}^{n_i} \log p(u_j|u_{0:j-1}, \theta)$. The model parameters θ are learnt by maximising this likelihood over the entire dataset, i.e, over all $\mathcal{U}$. The tokeniser component of a language model encodes an input text sequence into a sequence of tokens from $\mathcal{V}$, which is vital for training, as it converts text into numerical vectors.

The transformer architecture [129], including an encoder and decoder, is a popular choice for language modelling. BERT [113], a language modelling architecture based solely on the transformer encoder, is trained using the log-likelihood $\log p(U_i|\theta) = \sum_{j=1}^{n_i} \log p(u_j|u_{-j}, \theta)$ (with u_{-j} as all tokens except u_j) and designed to be fine-tuned [105]. This training objective allows the language model to look at all words in a sequence, before and after the given word, when calculating the embeddings. As in the transformer encoder architecture, this model outputs an embedding for each input token, contextualising them within a sequence.

To calculate sentence embeddings from word embeddings, Reimers and Gurevych (2019) [259] used a mean-pooling strategy to produce a single vector (of length given by the length of the token embedding) representing an entire sequence (with a maximum length defined by the BERT architecture). In parallel, Wang et al. (2020) [260] presented a method for distilling a BERT-like model into smaller models, which are less resource-intensive to fine-tune. The authors made this distilled model, MiniLM (and subsequently MiniLM V2), publicly available. These works led to a sentence embedding model based on a fine-tuned MiniLM V2 model [235, 259, 260] referred to in this chapter as “SE-MiniLM”.

7.4 Fine-Tuning Language Models

Fine-tuning is a process in which a pre-trained model is trained further on a subset of the original training dataset or on a new dataset and task.

Large models, such as language models, require a considerable amount of computational cost to train, since their size implies a vast parameter search space. In general, a significant number of tasks involving

language require similar subsets of this parameter search space, suggesting that efforts are repeated when models are trained for different language applications. Pre-training and fine-tuning aims to address this problem. Foundation models, which have been pre-trained on broad text datasets, contain parameters that are applicable to natural language and can be fine-tuned on specific tasks with less compute cost than would have been required for randomly initialised models. Fine-tuning therefore can be thought of as using the prior knowledge acquired by models during pre-training, to provide an initial guess of the optimal parameters for a new task. This viewpoint assumes that the optimisation problem is non-convex, where the optimal model parameters found will be in a local minimum, conditioned on the pre-trained parameters. Mathematically, given pre-trained parameters θ_{pt} for a model f and a fine-tuning dataset of N samples $(\mathbf{x}_n, y_n) \in \mathcal{D}_{\text{ft}}$, the parameters over M updates of gradient descent $\{\theta_1, \dots, \theta_{M+1}\}$, with learning rate α and loss function $\mathcal{L}$, are related by:

$$\theta_{M+1} = \theta_{\text{pt}} - \alpha \nabla_{\theta} \left[\sum_{n=1}^N \mathcal{L}(y_n, f(\mathbf{x}_n, \theta)) \right] \Big|_{\theta=\theta_{\text{pt}}} - \dots - \alpha \nabla_{\theta} \left[\sum_{n=1}^N \mathcal{L}(y_n, f(\mathbf{x}_n, \theta)) \right] \Big|_{\theta=\theta_M}$$

First parameter update during fine-tuning

Pre-trained parameters

Final parameter update during fine-tuning

This illustrates how the fine-tuned parameters θ_{ft} , acquired through M gradient update steps, depend on the initial pre-trained parameters θ_{pt} , demonstrating that prior knowledge restricts the parameter fine-tuning search space. In reality, other gradient-based optimisation methods, such as stochastic gradient descent or Adam, are commonly used to calculate the fine-tuned parameters, but the same general insight holds true.

7.5 Large Language Models for Time-Series Representation Learning

This work presents initial results from a system capable of encoding days of recorded activity data into vectors that are useful for care delivery. In summary, we transform each participant's daily time-series of events into strings of text that are encoded using a sentence embedding model as illustrated in Figure 7.3.

For each participant and day, we have access to the sensor recordings to the nearest second. As the tested language model is based on a predefined transformer architecture, it requires an input sequence of less than 256 tokens in length. Therefore, due to the irregular frequency of our time-series, we must aggregate recordings so that they can be processed by the language model. We do this by calculating the modal sensor reading for each 20 minute period (and uniformly sampling from these modes when more than one exists), which transforms each day into a sequence of locations, 72 tokens in length. Where there

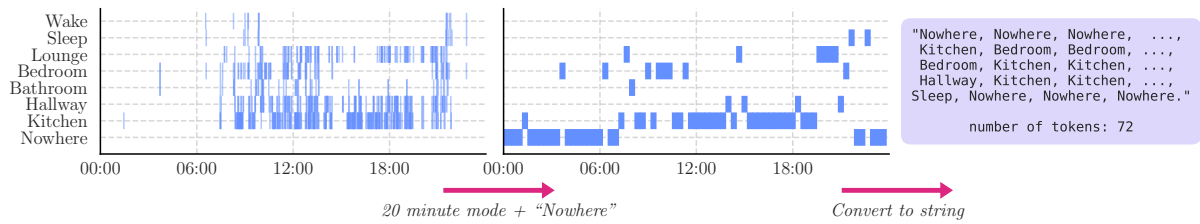


Figure 7.3: Preprocessing of data. For a single participant, we present their data collected over a single day. The left graph shows the raw measurements; the central graph shows the measurements after the mode of each 20 minute window is taken and a token for no activity is assigned (“Nowhere”); and the right graph shows the day as a single text string that can be interpreted by a language model.

is no sensor reading for one of the 20 minute periods, then we assign it the location “Nowhere”. This results in a discrete-valued and regular-frequency time-series, which is transformed into a text string to be interpreted by a tokeniser and language model.

Sequence (or sentence) embedding models can be fine-tuned using a triplet loss [261], measuring the cosine similarity between the embeddings of an anchor sequence and a similar and dissimilar sequence, such that similar embeddings are learnt for similar sequences.

The SE-MiniLM model described in Section 7.3 is fine-tuned on these day-strings using a triplet loss and by assigning a similar and dissimilar example for every string. Inspired by Franceschi et al. (2019) [83], for a given day-string, we uniformly sample another string generated from the same participant within a 30-day window and label it as similar, and uniformly sample a string from a different participant to label as dissimilar. In this way, the model learns personalised representations of days that allow the detection of activity changes. This is because the model learns differences between long-term changes in behaviour and will group participants with similar daily routines into similar regions in the vector space. In particular, this method makes use of the pre-trained model’s learnt semantic relationship of locations which we hypothesise allows for more meaningful encoding of activity.

For fine-tuning SE-MiniLM, we randomly sample triplets of text strings from the dataset: One query sentence, one similar sentence, and one dissimilar sentence. For each epoch, we randomly sample 100,000 triplets from the dataset and assign them to batches of size 256. We evaluate sentence embeddings using a triplet loss applied over each triplet, which is optimised using the AdamW algorithm [262] with a learning rate of 2×10^{-5} and a weight decay of 0.01. A linear warm-up learning rate scheduler is used with a warm-up of 10,000 steps.

With these learnt embeddings, days can be clustered using unsupervised techniques or searched over using vector search methods. Further, similarity can be measured over time to estimate changes in behaviour.

To find similar sequences using a trained model, semantic vector search applies a similarity measure to objects' learnt embeddings which reveal those that occupy similar areas in a latent space. This can be done using various similarity measures, but in this work we use cosine similarity, the most common method.

7.5.1 Availability of Code

The code to fine-tune the model and process the data presented in this work is made publicly available². However, the contents of the dataset and the Jupyter³ notebook used to generate the figures will not be made publicly available due to their sensitive nature.

This code is implemented in python (v3.11.5), using the following packages:

- numpy (v1.26.4) [211]: For manipulation of arrays and computing mathematical equations.
- pandas (v2.2.2) [212]: For manipulating the datasets and results.
- scikit-learn (v1.5.1) [213]: For loading and preprocessing the data.
- sentence-transformers (v2.4.0) [259]: For using methods to embed sequences based on language models.
- torch (v2.1.0) [214]: For building and manipulating neural networks.
- transformers (v4.46.0) [235]: For loading and using language models.

7.6 The Embedding Space

To visualise the embedding space in two dimensions, we used t-distributed Stochastic Neighbour Embedding (t-SNE) [263], a common dimension reduction method for visualisation. We set the perplexity to 30, the early exaggeration factor to 12, and the learning rate to 1377 (number of samples divided by 48) [213]. The embeddings are z-normalised using the mean and standard deviation of each feature before we apply t-SNE.

In Figure 7.4, we present some participant journeys in the embedding space. In particular, we plot the 25 participants with the most recorded days of data. We notice that some participants' activity is significantly

²Here: <https://github.com/alexcapstick/Text-Encoders-For-Daily-Movement-Data> or here: <https://github.com/tmi-lab/Text-Encoders-For-Daily-Movement-Data>

³<https://jupyter.org/>

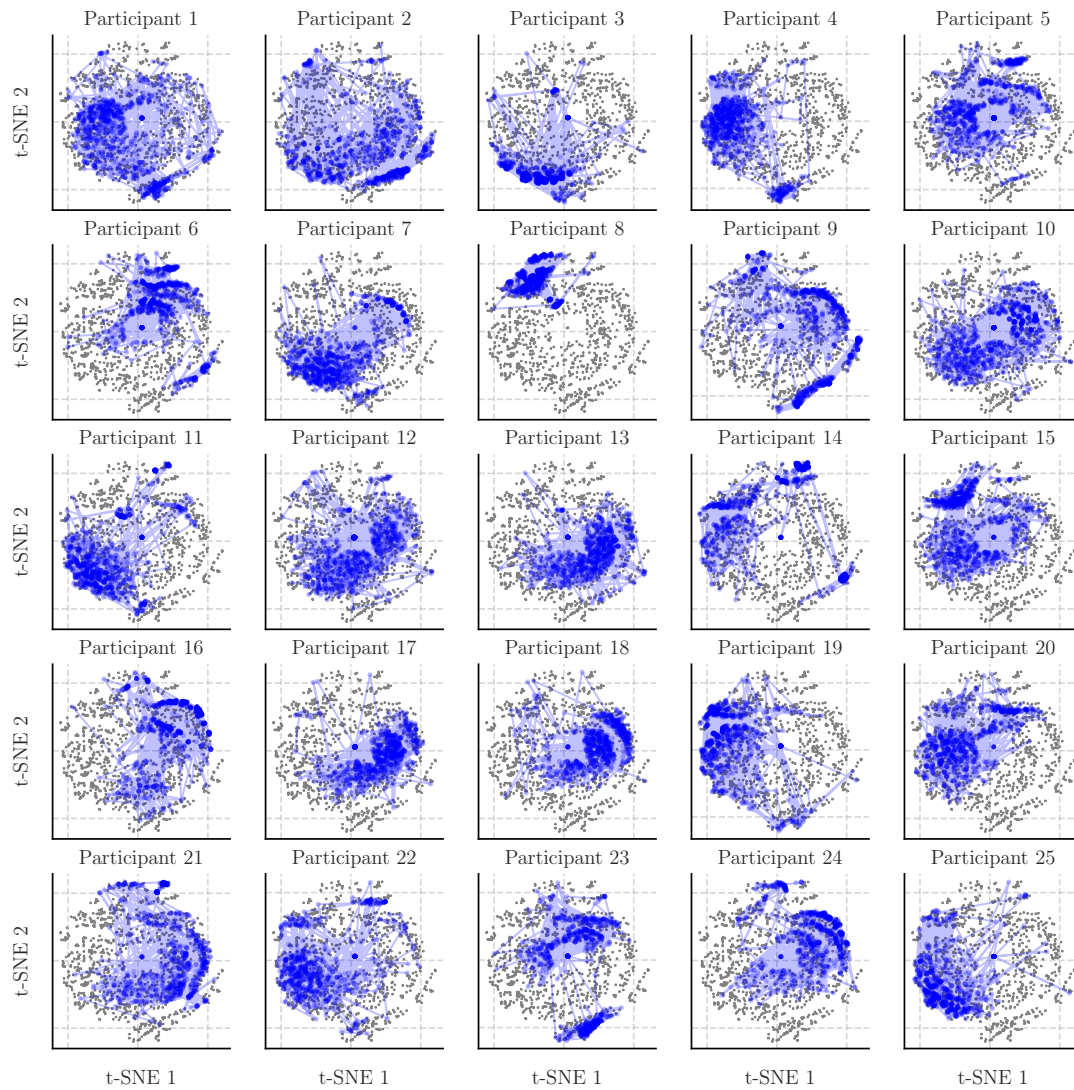


Figure 7.4: t-SNE of embeddings and journeys. The t-SNE embeddings of 1000 randomly sampled days of data (in grey) with the embeddings of a given participant's data overlaid (in blue) with consecutive days connected by a line. Here, we show the 25 participants with the most recorded days of data.

more localised than others, suggesting less variability in their day-to-day activity. On the other hand, some participants, for example Participant 1, cover a significant amount of the embedding space over time.

Figure 7.4 demonstrates that participants have unique activity maps. This variation between participants of activity in the latent space further motivates the use of a Date-ID split for the evaluation of our urinary tract infection risk prediction model (presented in Chapter 4).

This visualisation also allows us to identify participants who have similar journeys in the transformed embedding space. Participants 15 and 20, for example, occupy similar areas in the transformed vector space, under t-SNE.

7.7 Clustering

To further interpret the learnt representations, we use k -means to cluster the embedded days of data. As described in Section 2.3.3, k -means makes some strict assumptions about the structure of the data, including: Clusters are isotropic, there is a fixed number of non-hierarchical clusters, features are independent, and the distance metric used is meaningful in this context. Although these assumptions are restrictive and likely not satisfied by our embedded vectors, k -means is fast to run, making it effective for the purposes of demonstration in this section. We apply k -means to the embeddings before t-SNE is used for visualisation.

We use the k -means ++ algorithm [264], a Euclidean distance metric, and numbers of clusters varying from 2 to 10. Before applying the clustering model, the embeddings are normalised using the mean and standard deviation of each feature. Table 7.1 shows the silhouette scores of this, in which 5 clusters are chosen as optimal, revealing 5 categories of days.

Table 7.1: Silhouette score for k -means clustering on the base and fine-tuned models.

Fine-tuning	Number of clusters								
	2	3	4	5	6	7	8	9	10
True	0.39	0.47	0.51	0.57	0.54	0.53	0.52	0.52	0.51
False	0.42	0.21	0.18	0.16	0.16	0.14	0.14	0.17	0.16

Looking at some of the days represented by these clusters, we can intuitively explain what they might represent. In Figure 7.5, we show days of data in cluster 5, which correspond to days with low levels of recorded activity, which are likely due to sensor recording failures, participants being away from home, or days with little in-home activity (possibly due to adverse health conditions). These days were likely clustered together, as they represent a type of day that appears in a similar way among all participants.

Similarly, Figure 7.6 shows 8 random days from cluster 1, which could represent days with high amounts of kitchen activity. Although this cluster could be used to semantically classify a day, it does not appear to have meaningful clinical utility, suggesting that more sophisticated methods for learning clusters are required. In practice, clinicians could provide labels for some of the types of days participants have, which we can use to label other similar days.

Figure 7.7 shows the t-SNE embeddings and their cluster values. By viewing these embeddings for different participants (on the upper right-hand side), we can visualise the journeys in the embedding space for 4 participants. The lower right-hand plot shows the cluster value for each day of data for 4 participants, which tells us how frequently different participants move between clusters or when activity

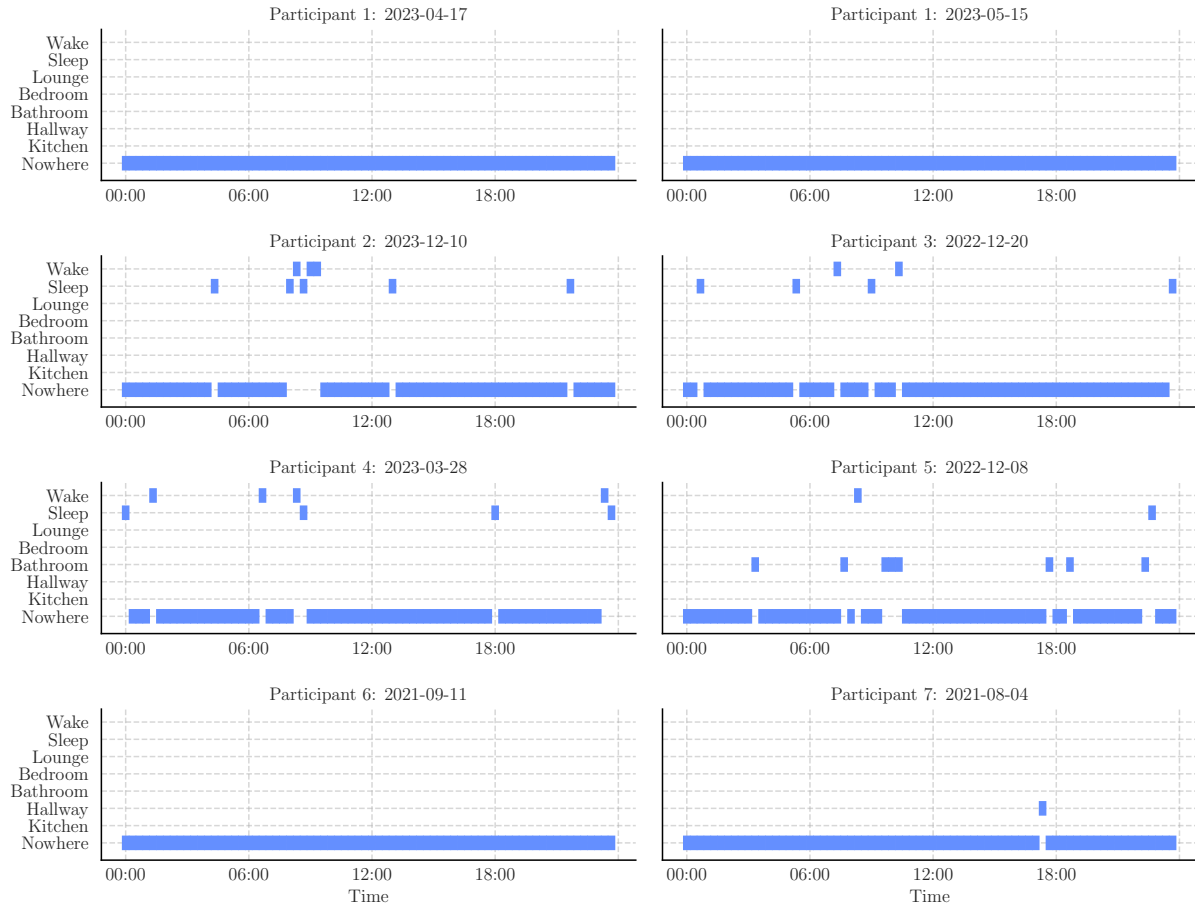


Figure 7.5: Days of data in cluster 5. We present 8 randomly chosen days of data from cluster 5, which is characteristic of sparse recorded activity.

changes. By exploring with clinicians what these clusters represent semantically, we can gain insight into a participant's behaviour and care requirements over time. For example, we found that cluster 5 is characteristic of small amounts of recorded activity and potentially adverse health conditions.

We can also visualise the proportion of participants in each cluster over time, visualising the behaviour of the cohort as a whole. In Figure 7.8, we present the proportion of participants with a given cluster label for each day in the dataset. Given a semantic understanding of the different days that each cluster represents, we can visualise the behaviour of the participant cohort as a whole. We observe that cluster 2 generally increases in size over the length of the dataset, whilst cluster 3 seems to rise and fall in size between September 2021 and September 2022.

Interestingly, there is a significant increase in the size of cluster 5 in July 2022. After consulting the software team within the Minder study, it was found that this was caused by an issue with the data collection system, in which no data was collected on 9th and 10th July 2022. Although not directly related to patient care, our system highlighted a significant data loss through the interpretation of cluster 5.



Figure 7.6: Days of data in cluster 1. We present 8 randomly chosen days of data from cluster 1, which is characteristic of frequent bedroom and kitchen activity.

7.8 Vector Searching for Similar Days

Using the cosine similarity between embeddings, for each day of data we can identify the other most similar days.

To illustrate this, in Figure 7.9, we show a day of data (chosen at random) and the 9 most similar days as measured by cosine similarity. These days show similar aspects, and all have large levels of Kitchen and Hallway activity during the day, with small amounts of Lounge, Bedroom, and Bathroom activity.

We can also use this to find the days that are most similar to those labelled with adverse health conditions. For example, since we know the days that participants were tested for a UTI, we can calculate the similarity between days labelled as positive and negative for each participant.

To do this, we identify the participants and dates associated with a UTI label. We then keep only the participants with at least two positive UTI labels and one negative UTI label. This is done to ensure

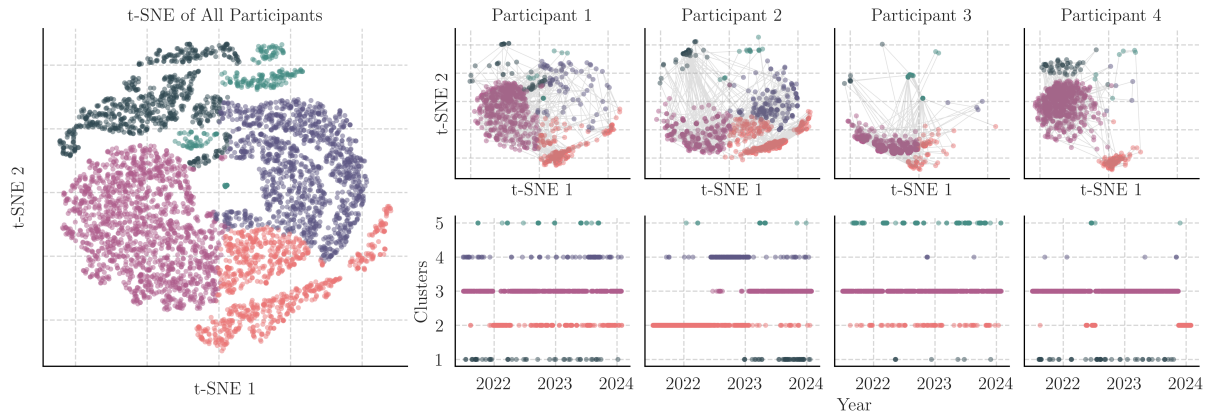


Figure 7.7: t-SNE of embeddings. The left plot shows the t-SNE transformation of 5000 day-string embeddings, coloured by their cluster value. On the upper row of the right-hand figures, we show the t-SNE embeddings for the 4 participants with the most data, where a line indicates consecutive days. On the lower row, we show the daily cluster values in time for the same participants.

that for a given participant, we can calculate the similarity between the positive labels (excluding the cosine similarity between the same days) and the similarity between the positively labelled days and the negatively labelled days. After this, we are left with 77 positively and 77 negatively labelled days produced by 14 participants. For each of these participants, we calculate the average cosine similarity between positive days of UTI and the average cosine similarity between the positive and negative days of UTI. We can then find the mean and standard deviation of these values to assess the utility of similarity searching for days of adverse health conditions.

We find that the mean ($\pm$ standard deviation) of the average intra-participant cosine similarity between positive and negative days of UTI is $0.53 (\pm 0.36)$ and $0.39 (\pm 0.50)$, respectively. This suggests that for each participant, the positive days of UTI are more similar to each other than to the negative days of UTI.

Although it provides an interesting example, the utility of this method for the classification of adverse health conditions is not clear and is likely not as performant as learning a classifier from engineered features as presented in Chapter 4. However, given a classifier that detects a general spectrum of adverse health conditions, using similarity searching might be helpful in explaining a given prediction.

7.9 Inferring Clinical and Care Needs

This method provides us with the ability to visualise changes in participant behaviour over time. To do this, we can measure the cosine similarity between each participant for each day to visualise how their activity varies. This is shown in Figure 7.10, which reveals when participants have frequent changes in

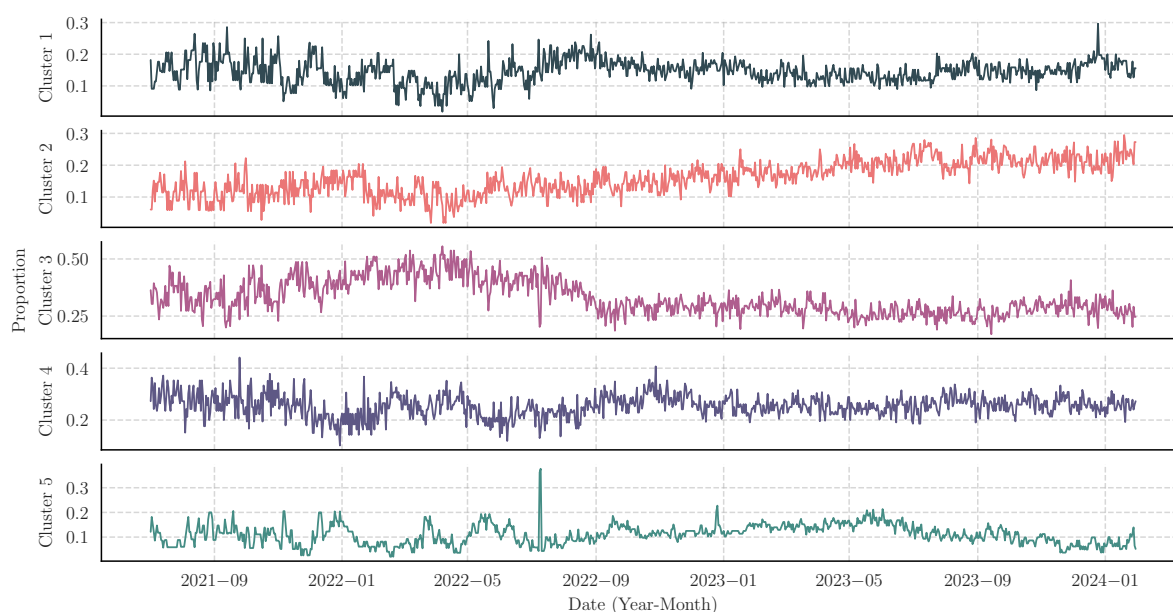


Figure 7.8: Cluster proportions. For each day in the dataset, we show the proportion of days clustered with each label.

home activity and when participants have more consistent activity. For example, participants 1 and 6 have frequent changes in daily activity with small regions of consistency in their activity. However, participant 2 had very consistent activity that suddenly switched to a different state of consistent activity, suggesting a noticeable change in their daily routines. Visualising intra-participant similarity, as we have here, could notify healthcare monitoring teams of changes in behaviour that might require changes in care delivery or further investigation. It also allows them to see, at a glance, the consistency of a participant’s in-home activity over an extended period of time.

7.10 Discussion

In remote healthcare monitoring, the vast majority of collected data will be unlabelled. Therefore, how to extract clinical insights from this data is an important open area of research. Representation learning presents an option to encode unlabelled activity data into embeddings that are helpful for downstream predictive tasks, similarity searching, behavioural change detection, or clustering.

This chapter proposed a method for using pre-trained language models for representation learning. In this way, our method uses the prior knowledge that a model captures during training on large text datasets to encode time-series data with states that hold semantic meaning.

We show by example that these representations allow for the measurement of behavioural changes,

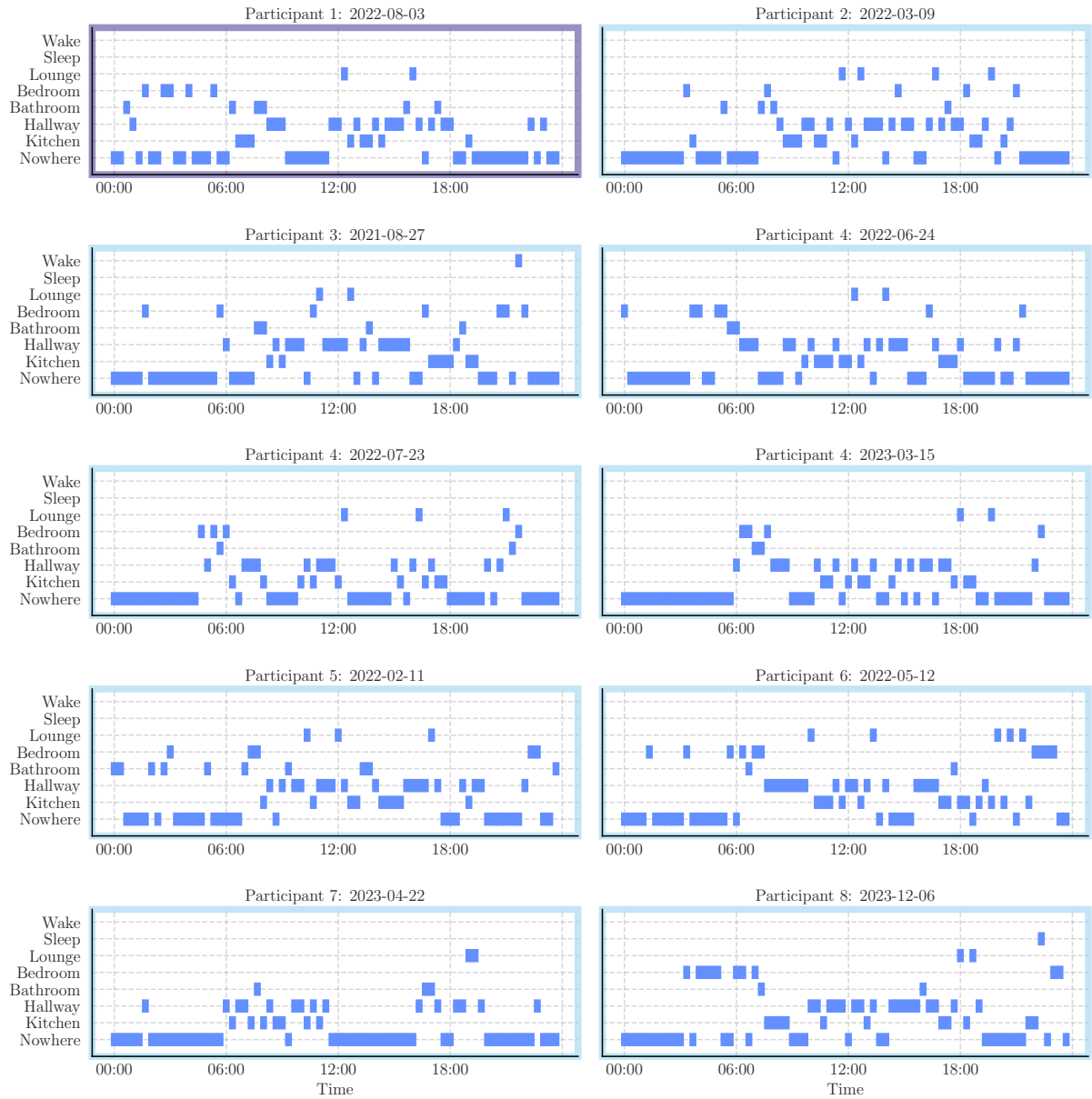


Figure 7.9: Vector search over days. Given a day of data (shown in purple in the top left), we present the 9 most similar other days as measured by cosine similarity (shown in blue).

providing caregivers and monitoring teams with visibility over longitudinal changes in activity and sleep data. By comparing changes between participants, monitoring teams can identify homes that produce more or less consistent data and view periods of instability in routines. Additionally, these representations can be used as part of a clustering method to identify different types of daily activity. The longitudinal changes in these clustering states can be examined at a participant or cohort level to provide insights on the trajectory of daily activity across a study or identify seasonal patterns.

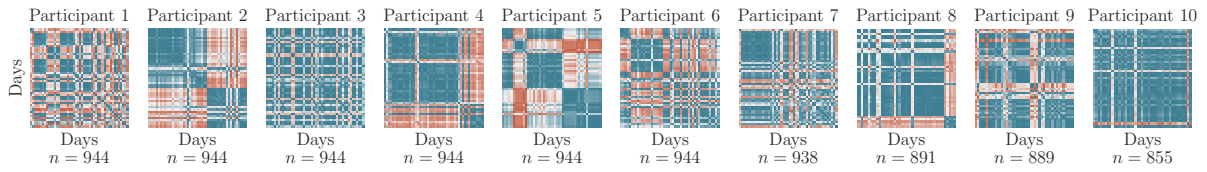


Figure 7.10: Day similarity. Each image shows the cosine similarity between every 20th recorded day for 10 participants. Blue, white, and red correspond to a cosine similarity of 1, 0, and -1 respectively. The range of the days is represented by each axis of the plots and is given by n .

7.11 Summary

This chapter presents the initial results of a discrete-valued time-series representation learning method, focussed on embedding days of in-home activity data, and with wide applications to other domains. These learnt embeddings are useful for clustering, vector search, and measuring behavioural changes which provide clinicians with additional tools to understand their study participants at an individual or group level, informing the planning of care delivery.

Chapter 8

Conclusions and Future Work

In this thesis, we have studied and proposed methods to tackle some of the current challenges facing remote healthcare monitoring systems. We started by focussing on a real-world case study for predicting adverse health conditions using in-home sensors, specifically Urinary Tract Infections (UTIs). This project helped us to contextualise some of the research areas that required development to build robust systems for remote healthcare monitoring. We identified three main areas to contribute to current research: (1) Learning from noisy data; (2) training with sparsely labelled data; and (3) extracting insights from unlabelled data. These pieces of research constituted the main body of this thesis. As part of this research, we have made the following contributions:

- We proposed a UTI risk prediction model, designed in collaboration with clinicians to rely on symptom-led engineered features and produce interpretable and calibrated predictions.
- We proposed a method to learn from data generated by multiple sources with unknown noise levels and evaluated our approach on public datasets commonly used in the noisy data literature.
- We designed a framework to transfer the knowledge of Large Language Models (LLMs) obtained during training to linear predictive models. This allowed us to combine the explosive progress of research on LLMs with robust predictive models in use within healthcare and trusted by clinicians. Using LLMs to supplement the prior elicitation process has the additional benefit of improving the performance of linear models over standard priors, whilst requiring fewer labels to train.
- Using LLMs, we proposed a representation learning method for activity-based time-series data for remote healthcare monitoring. The embeddings enabled clustering, vector searching, and the

calculation of daily similarity scores. The proposed method allows for the effective exploration of unlabelled data and the visualisation of changes in in-home behaviour that can improve the planning of care delivery.

These research directions have advanced the knowledge of remote monitoring methods focused on dementia care but are also more broadly applicable to healthcare systems.

8.1 Extended Applications of This Research

Several aspects of the research presented in this thesis have inspired methods used in other works that focus on remote healthcare monitoring. In addition to this, we have completed further research aimed at assessing the validity of specific systems for effective care delivery.

8.1.1 Using Entropy-Based Features for Time-Series Pattern Recognition

To predict the risk of UTI in the Minder study, we used an engineered feature representing the entropy of daily PIR sensor activations in homes. However, designing features for specific tasks can be time-consuming and sometimes requires expert knowledge.

Within Huang et al. (2024) [265], we further studied the use of entropy-based features for time-series predictive tasks involving neurological conditions as an alternative to raw data, simplifying the modelling pipeline. We found that in many cases, using entropy-based features provided improved performance over raw inputs, whilst being simple to implement. This could allow for faster development of predictive models with reduced error when access to expert information is limited.

In this particular work, my contributions were to the evaluation methodology, data curation, validation of results, and manuscript preparation. Specifically, I provided suggestions for benchmarks and code to process the data, I verified the results by reviewing the code, and I commented on and edited the manuscript.

8.1.2 Predicting Events of Agitation

Agitation is a neuropsychological symptom characterised by persistent behaviours that indicate emotional distress and affects more than 30% of people living with dementia [266]. Episodes of agitation can have

an impact on both those with dementia and their caregivers [267] and are frequently associated with other adverse health conditions, including fall-related injuries [268].

The prediction of agitation events, described in Bafaloukou et al. (2025) [58], uses some of the evaluation and stratification methods presented in Chapter 4. This work used passively collected in-home activity, physiology, and environmental data to detect episodes of agitation in people living with dementia within the Minder study.

Additionally, using methods for explainability, this work highlighted features with possible interventional actions that could lead to reduced risk of agitation. To facilitate the design of these personalised interventions, this study presented an interactive interface for clinicians to study how changes in the participant's environment could affect the predicted risk of agitation. Future work aims to study these interventions in an ongoing study.

In this particular work, my contributions were to the methodology, code, validation of results, and manuscript preparation. Specifically, I provided suggestions and code for evaluating methods and the presentation of the results, and I commented on and edited the manuscript.

8.1.3 Early Dementia Risk Stratification Using Speech Biomarkers

Speech monitoring provides an opportunity to assess disease progression as it can be collected continuously and passively, and significant research efforts have been carried out to apply machine learning to voice recordings [269].

Lima et al. (2025) [59] presented the use of machine learning models to predict a person's risk of dementia based on spoken descriptions of an image. This work applied some methods that were inspired by the presentation of results given in Chapter 4. This included using methods for explainable predictions and risk stratification, allowing us to provide interpretable predicted risk groups instead of a diagnosis. This enables the calibration of predicted risk and the use of our method as a decision-support tool.

We found evidence that spoken language could be used as a biomarker to detect and track cognitive impairment and decline, potentially enabling the use of in-home systems that can monitor the risk of dementia or its progression through conversations. This work showed that we can accurately distinguish between patients with and without dementia based on their speech and that the predicted risk of dementia was correlated with scores in cognitive assessments without explicitly being trained for their prediction.

This was shown to be effective in English speech data from DementiaBank [269] as well as an additional dataset containing English and Spanish speakers, demonstrating the merit of the approach across languages.

In this particular work, my contributions were to the evaluation methodology, code, validation of results, and manuscript preparation. Specifically, I provided suggestions and code for the evaluation of methods and visualisation of the results, verified the results by reviewing the code, and commented on and edited the manuscript.

8.1.4 Identifying Dementia Specific Sleep Patterns

Sleep disturbances are one of the most common symptoms of dementia, with a growing body of research suggesting that sleep disorders can act as predictors of mild cognitive impairment [270–272].

In Rigny et al. (2024) [272], we compared the data collected from the sleep mat in our Minder study to data collected from a general population; with more than 100 people generating 9500 person-nights of data in total. Utilising unlabelled data, this work found sleep patterns that were specific to participants with dementia, suggesting the use of passive sleep monitoring devices for disease tracking and, more generally, remote healthcare systems.

In particular, we found that people with dementia were more likely to cluster in a group characterised by higher heart rates and respiratory rates at night, higher Wakefulness After Sleep Onset (WASO) and light sleep, and lower mean durations of Rapid Eye Movement (REM) and Non-rapid Eye Movement (NREM) sleep. These findings align with research in the literature, providing evidence for the effectiveness of extracting clinical insights through sleep monitoring with passive devices.

In this particular work, my contributions were to the evaluation methodology, and manuscript preparation. Specifically, I provided suggestions and code for the evaluation of methods, and I commented on and edited the manuscript.

8.1.5 In-home Assessment-Based Measurement of Longitudinal Changes in Behaviour

Cognitive assessments such as the Neuropsychiatric Inventory (NPI) and the Bristol Activities of Daily Living Scale (BADL) are used to measure behavioural and psychological symptoms of people with dementia, including their ability to perform activities of daily living. The use of these in-home cognitive assessments can provide a cost-effective method to improve and personalise care delivery.

In Walsh et al. (2025) [273], we studied how cognitive decline impacts the ability of people with dementia to perform daily tasks and how it affects the mental health of their caregivers. Using cognitive assessments that were completed at 3 month intervals within the Minder cohort, we clustered participants into three groups based on their severity. This revealed that mild cognitive decline was associated with greater anxiety for the carer, whilst the group with the most severe cognitive decline interacted more with community services.

This work uses clinically validated assessments, performed within participants' homes, to help identify the care needs of patients. Since these approaches are scalable, the use of cognitive assessments to aid in care planning allows for a more efficient deployment of resources and personalised care delivery.

In this particular work, my contributions were to the methodology, data processing, code, and manuscript preparation. Specifically, I provided suggestions and code for the processing of data and the evaluation of methods, and I commented on and edited the manuscript.

8.2 Future Research Directions

There are some limitations and future research directions that allow for continued development in remote healthcare monitoring for people living with dementia.

8.2.1 Risk Prediction of Urinary Tract Infections

Although the work presented in Chapter 4 was conducted using readily available and low-cost sensors (with preliminary analysis of sensor importance presented in Section 4.4.1), future research could aim to understand the relationship between the cost of devices and the performance of UTI risk prediction. For example, do less or more expensive in-home sensor deployments provide proportional changes to model performance? In particular, the deployed PIR sensors allow us to collect data at a low cost but do not allow for the distinction between data generated by the person of study and other members of the house. However, we found that including patient-specific data from the sleep mat significantly improves predictive performance (Section 4.4.1). Therefore, future work could explore other methods of passively collecting day-time personalised data, possibly incurring a higher cost.

In Section 4.4.1, we discuss model reliability and calibration and find that our model overestimates the risk of UTI, likely due to data imbalance in the training set. This motivates the stratification of risk, which

allows the monitoring team to balance the sensitivity and specificity with the number of generated alerts. However, when deploying this system, work should be done to understand the trade-off between false positives and false negatives to ensure that risk groups are well-calibrated. In our retrospective silent testing, we further refined these thresholds to meet the specific requirements of the clinical monitoring team, but a process for the continuous assessment of risk stratification thresholds would be informative. Finally, whilst this work focused on an important section of the population (people living with dementia), it would be helpful to apply these methods to a larger cohort study or one containing older adults in community living environments such as care homes, assisted living, or skilled nursing facilities.

8.2.2 Mitigating the Effects of Noisy Data Sources

The research in Chapter 5 reveals some future directions. In our experiments, we study models of varied capacity (for example, Table 5.1 and Section 5.5.10, and Table 5.3 and Section 5.5.16), however, it would be valuable to study the effects of ill-specified models on noisy data techniques, as most methods assume that models attain smaller losses on the non-noisy data points (for example: in our work and Co-teaching [157]), or that logits are reliable (for example: in RRL [128] and IDPA [158]).

Additionally, we focus on the setting in which knowing the source of a data point provides additional information when learning from noisy data. However, when the noise level is uniform across *all* sources, the source value does not provide additional information about the likelihood of a data point being noisy. In this case, LAP performs as well as standard training. Here, it might be beneficial to apply a traditional noisy data technique in addition to our method, such as with RRL + LAP, studied in the experiments (Table 5.3, Figure 5.7b, and Section 5.5.8).

8.2.3 Eliciting Prior Distributions from Language Models

Section 6.10.7 explores how experts can guide AutoElicit priors with natural language. However, studying further methods for incorporating experts into our framework would be beneficial. This could be achieved, for example, by fine-tuning LLMs on relevant literature.

Additionally, since AutoElicit priors are a product of their training, they might contain unwanted bias. An approach to avoid this would be to assign all features that are sensitive to unwanted bias, directly or indirectly, with an uninformative prior. Given an elicited prior and observed examples, the Bayes factor

can then be used to choose between the elicited prior and an uninformative prior. In addition, experts can steer LLMs away from unwanted bias through task descriptions. Valuable future research could explore the link between different alignment methods for LLMs and elicited priors, possibly as a way to assess the success of the alignment process.

Finally, it could be helpful to extend our results to neural networks. An immediate option is to build on the literature by asking the LLM to suggest data augmentation techniques for a task. This would effectively reduce the cost of data collection by extending a given dataset. Extending our results to non-linear models such as random forests or Gaussian processes would also be valuable.

8.2.4 Using Language Models for Time-Series Representation Learning

There are a few directions for future research, based on the work presented in Chapter 7, that are worth highlighting.

Firstly, it would be worthwhile to evaluate how synonyms of location tokens could change the learnt representations (for example, if “lounge” is replaced by “living room”). By experimenting with synonyms of the locations used during training, we can gain insights into how a fine-tuned model could be used in new environments (for instance, we could explore whether a model fine-tuned on in-home data generates meaningful representations of data collected from an assisted living environment).

Furthermore, it would be valuable to consider whether a state-space model [194], which avoids requiring a maximum sequence length, might be a useful replacement for the BERT-based model presented in this work. This might also allow for the encoding of longer and shorter windows of time, possibly allowing comparisons of behaviour between different time durations.

Additionally, alternative methods for clustering that involve domain knowledge from clinicians could be used to improve the utility of activity clustering and further demonstrate the use of identifying days similar to those labelled with adverse health conditions.

8.3 A New Remote Healthcare Monitoring System

We envision a future remote healthcare monitoring system that promotes preventative care and reduces the need for reactive care. This system will utilise low-cost passive sensors to continuously generate in-home

individual activity, physiology, and environmental data to be processed by a collection of machine learning pipelines. These pipelines will consist of decision-support tools to predict adverse health conditions and representation learning methods to transform data into meaningful embeddings to aid care planning. The effectiveness of these pipelines in generating clinical insight will be monitored and critically examined at regular intervals to ensure that the remote healthcare system operates efficiently. This system would also incorporate information from electronic healthcare records to generate the most accurate alerts and intervention recommendations.

The ideal remote healthcare monitoring system should also be integrated with healthcare services to ensure that outputs, such as predictions from machine learning pipelines or visualisations of raw data, are available to general practitioners and other medical professionals. This would make remote healthcare monitoring data and clinical insights available to monitoring teams and caregivers, who can manage high-frequency requests, as well as primary, secondary, and tertiary care providers when required.

8.4 Lessons Learnt

This thesis consists of research projects that provided meaningful insights and contributed to the use of machine learning and remote monitoring for healthcare applications. However, as part of this doctoral research, we faced many setbacks that are not discussed within the main body of this work. Lessons were learnt about the navigation of the research landscape, the design of investigation methodologies, and time management. This section provides a reflection on some of those challenges.

Simpler models often generalise better and are easier to interpret: Access to labelled remote healthcare monitoring data is limited. This is due to a multitude of reasons, including cost, natural rarity in adverse health conditions, or system integration issues. When initially building the UTI risk prediction model, we utilised supervised or semi-supervised deep learning methods involving large model architectures. Following further review and evaluation, we found that these models failed to generalise to new participants and were difficult for clinicians to interpret. As part of this investigation, we learnt that when simple models such as logistic regression are combined with carefully engineered features, designed in collaboration with domain experts, we often achieve superior performance, reliability, and interpretability compared to more complex architectures. This lesson led to the development of the UTI predictive model presented in Chapter 4 and inspired the evaluation of language models for prior elicitation in Chapter 6.

Limited labels can be overcome: Limited access to labelled examples is a common problem in healthcare, particularly in remote monitoring settings. During this doctoral study, it has become evident that clinical insights can still be extracted from datasets using other methods. Using unsupervised techniques for clustering, applying representation learning to unlabelled data, or specifying prior distributions on the parameters of predictive models all allow for actionable insights to be generated from data with few labels.

Solving real problems motivates impactful research: During this doctoral study, it has become clearer to me that I am personally motivated to solve real-world problems. When I started my studies, I initially focused on making technical contributions based on what I perceived to be the challenges facing remote healthcare monitoring. However, after focussing on the UTI risk prediction model, in collaboration with clinicians, I found that I tackled research problems in a more solution-orientated approach. I observed that new innovations in healthcare research have the potential to improve people's quality of life. Focussing on advancing current methods for remote healthcare monitoring, with the guiding principle of improving care for people with dementia, provided considerable motivation.

8.5 Concluding Remarks

Dementia is considered a significant health challenge in the UK, affecting around 982,000 people. This is projected to rise to 1.4 million by 2040 [4], requiring new methods for the efficient and effective delivery of care. Since the presentation of dementia symptoms can vary from patient to patient, providing personalised care will represent one of the greatest challenges of the present century.

It is important for those affected by dementia to maintain their wellbeing, independence, and quality of life as long as possible. We believe this is best achieved by providing care in their familiar home environments. By using continuously collected data, in combination with automated systems, remote healthcare monitoring can be deployed to provide actionable insights that empower caregivers and allow clinicians to plan effective care delivery, reducing pressure on healthcare services.

This thesis provides a step toward this goal by proposing methods aimed at reducing unplanned hospital admissions and improving the quality of life of those affected by dementia and other long-term conditions. We hope this work inspires further research into remote healthcare monitoring systems. We envision that future developments will reduce the workload on healthcare systems by improving preventative care and reducing the need for reactive care.

We are mindful that every data point in the Minder dataset is generated by someone affected by dementia. For this research to be possible, we are indebted to our study participants for their engagement in our study and for sharing their experiences. We hope that the results of this thesis will contribute to discovery science focussed on improving the outcomes of all those affected by dementia.

This doctoral study has equipped me with the tools to pursue my scientific interests. It has provided an opportunity to develop my strengths as an independent researcher and collaborator. Working alongside academics and healthcare professionals within the UK and internationally has taught me that solving real-world problems is a key motivator for my research.

Bibliography

- [1] Alexander Capstick, Francesca Palermo, Kimberley Zakka, Nan Fletcher-Lloyd, et al. Digital remote monitoring for screening and early detection of urinary tract infections. *npj Digital Medicine*, 7(1), Jan 2024. ISSN 2398-6352. doi: 10.1038/s41746-023-00995-5. URL <https://www.nature.com/articles/s41746-023-00995-5>.
- [2] Alexander Capstick, Tianyu Cui, Yu Chen, and Payam Barnaghi. Representation learning of daily movement data using text encoders. *International Conference on Learning Representations (ICLR) Workshop on Learning from Time Series For Health*, 2024. URL <https://arxiv.org/abs/2405.04494>.
- [3] Alzheimer’s Research UK, Sep 2015. URL <https://www.alzheimersresearchuk.org/news/dementia-in-the-future/>.
- [4] Alzheimer’s Society UK, May 2024. URL <https://www.alzheimers.org.uk/blog/how-many-people-have-dementia-uk>.
- [5] Martin Prince, Anders Wimo, Maëlen Guerchet, Gemma-Claire Ali, et al. World Alzheimer report 2015. The global impact of dementia: An analysis of prevalence, incidence, cost and trends. Research report, Alzheimer’s Disease International, Sep 2015. URL <https://unilim.hal.science/hal-03495438>.
- [6] World Health Organization. *Global action plan on the public health response to dementia 2017–2025*. World Health Organization, Geneva, 2017. ISBN 978-92-4-151348-7. URL <https://iris.who.int/handle/10665/259615>.
- [7] Malaz Boustani, Cathy Schubert, and Youcef Sennour. The challenge of supporting care for dementia in primary care. *Clinical Interventions in Aging*, 2(4):631–636, 2007.

- [8] Gill Livingston, Jonathan Huntley, Andrew Sommerlad, David Ames, et al. Dementia prevention, intervention, and care: 2020 report of the Lancet Commission. *The Lancet*, 396(10248):413–446, 2020.
- [9] NHS England, 2024. URL <https://qof.digital.nhs.uk/search/index.asp>.
- [10] Chloë Hood, Aimée Morris, and Samantha Ofili. National audit of dementia care in general hospitals 2018–19: Round four audit report, 2019.
- [11] Elizabeth L Sampson, Martin R Blanchard, Louise Jones, Adrian Tookman, and Michael King. Dementia in the acute hospital: Prospective cohort study of prevalence and mortality. *The British Journal of Psychiatry*, 195(1):61–66, 2009.
- [12] Henry Brodaty and Marika Donkin. Family caregivers of people with dementia. *Dialogues in Clinical Neuroscience*, 11(2):217–228, 2009.
- [13] Richard Schulz and Lynn M Martire. Family caregiving of persons with dementia: Prevalence, health effects, and support strategies. *The American Journal of Geriatric Psychiatry*, 12(3):240–249, 2004.
- [14] Asuka Koyama, Masateru Matsushita, Mamoru Hashimoto, Noboru Fujise, et al. Mental health among younger and older caregivers of dementia patients. *Psychogeriatrics*, 17(2):108–114, 2017.
- [15] Si Ying Tan, Jennifer Sumner, Yuchen Wang, and Alexander Wenjun Yip. A systematic review of the impacts of remote patient monitoring (RPM) interventions on safety, adherence, quality-of-life and cost-related outcomes. *npj Digital Medicine*, 7(1), Jul 2024. ISSN 2398-6352. doi: 10.1038/s41746-024-01182-w. URL <http://dx.doi.org/10.1038/s41746-024-01182-w>.
- [16] Office for National Statistics (ONS). Profile of the older population living in england and wales in 2021 and changes since 2011, Apr 2023. URL <https://www.ons.gov.uk/peoplepopulationandcommunity/birthsdeathsandmarriages/ageing/articles/profileoftheolderpopulationlivinginenglandandwalesin2021andchangessince2011/2023-04-03>.
- [17] Michael W Condry and Xiaohong Iris Quan. Remote patient monitoring technologies and markets. *IEEE Engineering Management Review*, 51(3):59–64, 2023.
- [18] NHS. Supporting care with remote monitoring, 2023. URL <https://transform.england.nhs.uk/covid-19-response/technology-nhs/supporting-the-innovation-collaboratives-to-expand-their-remote-monitoring-plans/>.

- [19] NHS. Evaluation of the regional scale programme and the national innovation collaborative - executive summary, Oct 2021. URL <https://transform.england.nhs.uk/key-tools-and-info/evaluation-of-the-regional-scale-programme-and-the-national-innovation-collaborative-executive-summary/>.
- [20] Farzan Sasangohar, Elise Davis, Bit A Kash, Sohail R Shah, et al. Remote patient monitoring and telemedicine in neonatal and pediatric settings: Scoping literature review. *Journal of Medical Internet Research*, 20(12):e9403, 2018.
- [21] Ha Uk Chung, Alina Y. Rwei, Aurélie Hourlier-Fargette, Shuai Xu, et al. Skin-interfaced biosensors for advanced wireless physiological monitoring in neonatal and pediatric intensive-care units. *Nature Medicine*, 26(3):418–429, Mar 2020. ISSN 1546-170X. doi: 10.1038/s41591-020-0792-9. URL <http://dx.doi.org/10.1038/s41591-020-0792-9>.
- [22] L Nelson Sanchez-Pinto, María del Pilar Arias López, Halden Scott, Kristen Gibbons, et al. Digital solutions in paediatric sepsis: Current state, challenges, and opportunities to improve care around the world. *The Lancet Digital Health*, 6(9):e651–e661, Sep 2024. ISSN 2589-7500. doi: 10.1016/S2589-7500(24)00141-9. URL [http://dx.doi.org/10.1016/S2589-7500\(24\)00141-9](http://dx.doi.org/10.1016/S2589-7500(24)00141-9).
- [23] José Oscar Olmedo-Aguirre, Josimar Reyes-Campos, Giner Alor-Hernández, Isaac Machorro-Cano, et al. Remote healthcare for elderly people using wearables: A review. *Biosensors*, 12(2):73, Jan 2022. ISSN 2079-6374. doi: 10.3390/bios12020073. URL <http://dx.doi.org/10.3390/bios12020073>.
- [24] Mariana Peyroteo, Inês Augusto Ferreira, Luís Brito Elvas, João Carlos Ferreira, and Luís Velez Lapão. Remote monitoring systems for patients with chronic diseases in primary health care: Systematic review. *JMIR mHealth and uHealth*, 9(12):e28285, 2021.
- [25] Raquel Leirós-Rodríguez, Jose L. García-Soidán, and Vicente Romo-Pérez. Analyzing the use of accelerometers as a method of early diagnosis of alterations in balance in elderly people: A systematic review. *Sensors*, 19(18):3883, Sep 2019. ISSN 1424-8220. doi: 10.3390/s19183883. URL <http://dx.doi.org/10.3390/s19183883>.
- [26] Rosaria Rucco, Antonietta Sorriso, Marianna Liparoti, Giampaolo Ferraioli, et al. Type and location of wearable sensors for monitoring falls during static and dynamic tasks in healthy elderly: A review. *Sensors*, 18(5):1613, 2018.

- [27] Mirza Mansoor Baig, Hamid GholamHosseini, Aasia A Moqem, Farhaan Mirza, and Maria Lindén. A systematic review of wearable patient monitoring systems – current challenges and opportunities for clinical adoption. *Journal of Medical Systems*, 41:1–9, 2017.
- [28] Yao Guo, Xiangyu Liu, Shun Peng, Xinyu Jiang, et al. A review of wearable and unobtrusive sensing technologies for chronic disease management. *Computers in Biology and Medicine*, 129: 104163, 2021.
- [29] Ann-Kathrin Schalkamp, Kathryn J Peall, Neil A Harrison, and Cynthia Sandor. Wearable movement-tracking data identify Parkinson’s disease years before clinical diagnosis. *Nature Medicine*, 29(8):2048–2056, 2023.
- [30] Wei Yue, Yunjian Guo, Jong-Chul Lee, Enkhzaya Ganbold, et al. Advancements in passive wireless sensing systems in monitoring harsh environment and healthcare applications. *Nano-Micro Letters*, 17(1):1–48, 2025.
- [31] Hande Alemdar and Cem Ersoy. Wireless sensor networks for healthcare: A survey. *Computer networks*, 54(15):2688–2710, 2010.
- [32] Shirin Enshaeifar, Ahmed Zoha, Andreas Markides, Severin Skillman, et al. Health management and pattern analysis of daily living activities of people with dementia using in-home sensors and machine learning techniques. *PloS one*, 13(5):e0195605, 2018.
- [33] Shirin Enshaeifar, Ahmed Zoha, Severin Skillman, Andreas Markides, et al. Machine learning methods for detecting urinary tract infection and analysing daily living activities in people with dementia. *PloS one*, 14(1):e0209909, 2019.
- [34] Tom M Mitchell and Tom M Mitchell. *Machine learning*, volume 1. McGraw-hill New York, 1997.
- [35] Charles E Leiserson, Neil C Thompson, Joel S Emer, Bradley C Kuszmaul, et al. There’s plenty of room at the top: What will drive computer performance after moore’s law? *Science*, 368(6495): eaam9744, 2020.
- [36] Ons Aouedi, Thai-Hoc Vu, Alessio Sacco, Dinh C Nguyen, et al. A survey on intelligent Internet of Things: Applications, security, privacy, and future directions. *IEEE Communications Surveys and Tutorials*, 2024.

- [37] Petroc Taylor. Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2023, with forecasts from 2024 to 2028, 2024. URL <https://www.statista.com/statistics/871513/worldwide-data-created/>.
- [38] Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, et al. A guide to deep learning in healthcare. *Nature medicine*, 25(1):24–29, 2019.
- [39] Marzyeh Ghassemi, Tristan Naumann, Peter Schulam, Andrew L Beam, et al. A review of challenges and opportunities in machine learning for health. *AMIA Summits on Translational Science Proceedings*, 2020:191, 2020.
- [40] Stan Benjamens, Pranavsingh Dhunoo, and Bertalan Meskó. The state of artificial intelligence-based FDA-approved medical devices and algorithms: An online database. *NPJ Digital Medicine*, 3(1):118, 2020.
- [41] Betsy Foxman and Patricia Brown. Epidemiology of urinary tract infections. *Infectious Disease Clinics of North America*, 17(2):227–241, Jun 2003. ISSN 08915520. doi: 10.1016/S0891-5520(03)00005-9.
- [42] Inês Linhares, Teresa Raposo, António Rodrigues, and Adelaide Almeida. Frequency and antimicrobial resistance patterns of bacteria implicated in community urinary tract infections: A ten-year surveillance study (2000–2009). *BMC Infectious Diseases*, 13(1):19, Dec 2013. ISSN 1471-2334. doi: 10.1186/1471-2334-13-19.
- [43] Lona Mody and Manisha Juthani-Mehta. Urinary tract infections in older women: A clinical review. *JAMA*, 311(8):844, Feb 2014. ISSN 0098-7484. doi: 10.1001/jama.2014.303.
- [44] Myriam Gharbi, Joseph H Drysdale, Hannah Lishman, Rosalind Goudie, et al. Antibiotic management of urinary tract infection in elderly patients in primary care and its association with bloodstream infections and all cause mortality: Population based cohort study. *BMJ*, 364, 2019.
- [45] Gauri P. Godbole, Nicole Cerruto, and Ruchir Chavada. Principles of assessment and management of urinary tract infections in older adults. *Journal of Pharmacy Practice and Research*, 50(3): 276–283, Jun 2020. doi: 10.1002/jppr.1650. URL <https://doi.org/10.1002/jppr.1650>.
- [46] Florian Me Wagenlehner, Christoph Lichtenstern, Caroline Rolfes, Konstantin Mayer, et al. Diagnosis and management for urosepsis: Items in urosepsis. *International Journal of Urology*, Jun 2013. ISSN 09198172. doi: 10.1111/iju.12200.

- [47] Sari Tal, Vladimir Guller, Shmuel Levi, Rita Bardenstein, et al. Profile and prognosis of febrile elderly patients with bacteremic urinary tract infection. *Journal of Infection*, 50(4):296–305, 2005.
- [48] Greg S Martin, David M Mannino, and Marc Moss. The effect of age on the development and outcome of adult sepsis. *Critical Care Medicine*, 34(1):15–21, 2006.
- [49] Mitchell M Levy, Antonio Artigas, Gary S Phillips, Andrew Rhodes, et al. Outcomes of the surviving sepsis campaign in intensive care units in the USA and Europe: A prospective cohort study. *The Lancet Infectious Diseases*, 12(12):919–924, Dec 2012. ISSN 14733099. doi: 10.1016/S1473-3099(12)70239-6.
- [50] Zoe Arvanitakis, Raj C. Shah, and David A. Bennett. Diagnosis and management of dementia: Review. *JAMA*, 322(16):1589, Oct 2019. ISSN 0098-7484. doi: 10.1001/jama.2019.4782.
- [51] Claudia Cooper, Rebecca Lodwick, Kate Walters, Rosalind Raine, et al. Inequalities in receipt of mental and physical healthcare in people with dementia in the uk. *Age and Ageing*, page ageing;afw208v1, Dec 2016. ISSN 0002-0729, 1468-2834. doi: 10.1093/ageing/afw208.
- [52] Christopher M. Callahan, Wanzhu Tu, Timothy E. Stump, Daniel O. Clark, et al. Errors in self-reports of health services use. *Alzheimer Disease and Associated Disorders*, 29(1):75–81, Jan 2015.
- [53] Sandeep Toot, Mike Devine, Ajiri Akporobaro, and Martin Orrell. Causes of hospital admission for people with dementia: A systematic review and meta-analysis. *Journal of the American Medical Directors Association*, 14(7):463–470, 2013.
- [54] Ahsan Rao, Amna Suliman, Sabine Vuik, Paul Aylin, and Ara Darzi. Outcomes of dementia: Systematic review and meta-analysis of hospital administrative database studies. *Archives of Gerontology and Geriatrics*, 66:198–204, 2016.
- [55] Naaheed Mukadam and Elizabeth L. Sampson. A systematic review of the prevalence, associations and outcomes of dementia in older general hospital inpatients. *International Psychogeriatrics*, 23(3):344–355, Apr 2011. ISSN 1041-6102, 1741-203X. doi: 10.1017/S1041610210001717.
- [56] Sumit Majumder, Emad Aghayi, Moein Noferesti, Hamidreza Memarzadeh-Tehran, et al. Smart homes for elderly healthcare—recent advances and research challenges. *Sensors*, 17(11):2496, 2017.

- [57] Riitta Turjamaa, Aki Pehkonen, and Mari Kangasniemi. How smart homes are used to support older people: An integrative review. *International Journal of Older People Nursing*, 14(4):e12260, 2019.
- [58] Marirena Bafaloukou, Ann-Kathrin Schalkamp, Nan Fletcher-Lloyd, Alexander Capstick, et al. An interpretable machine learning tool for in-home monitoring of agitation episodes in people living with dementia: A proof-of-concept study. *Lancet eClinicalMedicine*, 80, 2025. URL <https://doi.org/10.1016/j.eclinm.2024.103032>.
- [59] Maria R Lima, Alexander Capstick, Fatemeh Geranmayeh, Ramin Nilforooshan, et al. Evaluating spoken language as a biomarker for automated screening of cognitive impairment. *arXiv preprint arXiv:2501.18731*, 2025. URL <https://arxiv.org/abs/2501.18731>.
- [60] Divya Tomar and Sonali Agarwal. A survey on data mining approaches for healthcare. *International Journal of Bio-Science and Bio-Technology*, 5(5):241–266, 2013.
- [61] Emilie Baro, Samuel Degoul, Régis Beuscart, and Emmanuel Chazard. Toward a literature-driven definition of big data in healthcare. *BioMed Research International*, 2015, 2015.
- [62] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3): 107–115, Mar 2021. ISSN 0001-0782, 1557-7317. doi: 10.1145/3446776.
- [63] Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, et al. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, page 233–242. PMLR, Jul 2017. URL <https://proceedings.mlr.press/v70/arpit17a.html>.
- [64] Behnam Neyshabur, Srinadh Bhojanapalli, David Mcallester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [65] Dragan Gamberger, Nada Lavrac, and Saso Dzeroski. Noise detection and elimination in data preprocessing: Experiments in medical domains. *Applied Artificial Intelligence*, 14(2):205–223, Feb 2000. ISSN 0883-9514, 1087-6545. doi: 10.1080/088395100117124.
- [66] Jaree Thongkam, Guandong Xu, Yanchun Zhang, and Fuchun Huang. Support vector machine for outlier detection in breast cancer survivability prediction. In Yoshiharu Ishikawa, Jing He,

- Guandong Xu, Yong Shi, et al., editors, *Advanced Web and Network Technologies, and Applications*, page 99–109, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-89376-9.
- [67] Sarah Jane Delany, Nicola Segata, and Brian Mac Namee. Profiling instances in noise reduction. *Knowledge-Based Systems*, 31:28–40, 2012. ISSN 0950-7051. doi: <https://doi.org/10.1016/j.knsys.2012.01.015>.
- [68] Yisen Wang, Weiyang Liu, Xingjun Ma, James Bailey, et al. Iterative learning with open-set noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8688–8696, 2018.
- [69] Shenghui Li, Edith Ngai, Fanghua Ye, and Thiemo Voigt. Auto-weighted robust federated learning with corrupted data sources. *ACM Transactions on Intelligent Systems and Technology*, 13(5):1–20, Oct 2022. ISSN 2157-6904, 2157-6912. doi: 10.1145/3517821.
- [70] Sheng Liu, Jonathan Niles-Weed, Narges Razavian, and Carlos Fernandez-Granda. Early-learning regularization prevents memorization of noisy labels. *Advances in Neural Information Processing Systems*, 33, 2020.
- [71] Xiaobo Xia, Tongliang Liu, Bo Han, Chen Gong, et al. Robust early-learning: Hindering the memorization of noisy labels. In *International Conference on Learning Representations*, 2021.
- [72] Alexander Capstick and Payam Barnaghi. Predicting targets with data from non-conforming sources. *Tiny Papers Track at International Conference on Learning Representations (ICLR)*, 2023. URL <https://dblp.org/rec/conf/iclr/CapstickB23.html>.
- [73] Kexin Fan, Alexander Capstick, Ramin Nilforooshan, and Payam Barnaghi. Urinary tract infection detection in digital remote monitoring: Strategies for managing participant-specific prediction complexity. In *2024 IEEE International Conference on e-health Networking, Application & Services (HealthCom)*, pages 1–6, 2024. URL <https://ieeexplore.ieee.org/abstract/document/10880784>.
- [74] Alexander Capstick, Francesca Palermo, Tianyu Cui, and Payam Barnaghi. Training neural networks on data sources with unknown reliability. *Information Fusion*, 2025. URL <https://doi.org/10.1016/j.inffus.2025.103239>.
- [75] Michael Cooper, Zongliang Ji, and Rahul G Krishnan. Machine learning in computational

- histopathology: Challenges and opportunities. *Genes, Chromosomes and Cancer*, 62(9):540–556, 2023.
- [76] Vladimir Vapnik, Rauf Izmailov, et al. Learning using privileged information: Similarity control and knowledge transfer. *Journal of Machine Learning Research*, 16(1):2023–2049, 2015.
- [77] Petrus Mikkola, Osvaldo A Martin, Suyog Chandramouli, Marcelo Hartmann, et al. Prior knowledge elicitation: The past, present, and future. *arXiv preprint arXiv:2112.01380*, 2112, 2021.
- [78] Ryan Tennant, Jennifer Graham, Juliet Kern, Kate Mercer, et al. A scoping review on pediatric sepsis prediction technologies in healthcare. *npj Digital Medicine*, 7(1), Dec 2024. ISSN 2398-6352. doi: 10.1038/s41746-024-01361-9. URL <http://dx.doi.org/10.1038/s41746-024-01361-9>.
- [79] Carlos R. Ramírez Medina, Jose Benitez-Aurioles, David A. Jenkins, and Meghna Jani. A systematic review of machine learning applications in predicting opioid associated adverse events. *npj Digital Medicine*, 8(1), Jan 2025. ISSN 2398-6352. doi: 10.1038/s41746-024-01312-4. URL <http://dx.doi.org/10.1038/s41746-024-01312-4>.
- [80] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, et al. A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435*, 2023.
- [81] Jie Huang and Kevin Chen-Chuan Chang. Towards reasoning in large language models: A survey. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1049–1065, Toronto, Canada, Jul 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.67. URL <https://aclanthology.org/2023.findings-acl.67/>.
- [82] Alexander Capstick, Rahul G Krishnan, and Payam Barnaghi. AutoElicit: Using large language models for expert prior elicitation in predictive modelling. *International Conference on Machine Learning (ICML)*, 2025. URL <https://arxiv.org/abs/2411.17284>.
- [83] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. Unsupervised scalable representation learning for multivariate time series. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [84] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, et al. Time-series representation learning via temporal and contextual contrasting. In *Proceedings of the Thirtieth International Joint*

- Conference on Artificial Intelligence*, page 2352–2359, Montreal, Canada, Aug 2021. International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-9-6. doi: 10.24963/ijcai.2021/324. URL <https://www.ijcai.org/proceedings/2021/324>.
- [85] George Zerveas, Srideepika Jayaraman, Dhaval Patel, et al. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 2114–2124, New York, NY, USA, Aug 2021. Association for Computing Machinery. ISBN 978-1-4503-8332-5. doi: 10.1145/3447548.3467401. URL <https://dl.acm.org/doi/10.1145/3447548.3467401>.
- [86] Ling Yang and Shenda Hong. Unsupervised time-series representation learning with iterative bilinear temporal-spectral fusion. In *Proceedings of the 39th International Conference on Machine Learning*, page 25038–25054. PMLR, Jun 2022. URL <https://proceedings.mlr.press/v162/yang22e.html>.
- [87] Travers Ching, Daniel S. Himmelstein, Brett K. Beaulieu-Jones, et al. Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface*, 15(141): 20170387, Apr 2018. doi: 10.1098/rsif.2017.0387.
- [88] Jin Cui, Alexander Capstick, Payam Barnaghi, and Gregory Scott. Two-stage representation learning for analyzing movement behavior dynamics in people living with dementia. *AAAI Workshop on Large Language Models and Generative AI for Health*, 2025. URL <https://arxiv.org/abs/2502.09173>.
- [89] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. URL <http://ilpubs.stanford.edu:8090/422/>. Previous number = SIDL-WP-1999-0120.
- [90] Francesca Palermo, Yu Chen, Alexander Capstick, Nan Fletcher-Loyd, et al. TIHM: An open dataset for remote healthcare monitoring in dementia. *Nature Scientific Data*, 10(11):606, Sep 2023. ISSN 2052-4463. doi: 10.1038/s41597-023-02519-y. URL <https://www.nature.com/articles/s41597-023-02519-y>.
- [91] William Wolberg, Olvi Mangasarian, Nick Street, and W. Street. Breast cancer Wisconsin (diagnostic). UCI Machine Learning Repository, 1995. DOI: <https://doi.org/10.24432/C5DW2B>.

- [92] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 2009.
- [93] Jiaheng Wei, Zhaowei Zhu, Hao Cheng, Tongliang Liu, et al. Learning with noisy labels revisited: A study using real-world human annotations. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=TBWA6PLJZQm>.
- [94] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407 – 499, 2004. doi: 10.1214/0090536040000000067. URL <https://doi.org/10.1214/0090536040000000067>.
- [95] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.
- [96] Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, et al. GoEmotions: A Dataset of Fine-Grained Emotions. In *58th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.
- [97] Robert C. Detrano, András Jánosi, Walter Steinbrunn, Matthias Emil Pfisterer, et al. International application of a new probability algorithm for the diagnosis of coronary artery disease. *The American Journal of Cardiology*, 64 5:304–10, 1989. URL <https://api.semanticscholar.org/CorpusID:23545303>.
- [98] William Steinbrunn Andras Janosi. Heart disease, 1989. URL <https://archive.ics.uci.edu/dataset/45>.
- [99] Ross Quinlan. Thyroid disease. UCI Machine Learning Repository, 1986.
- [100] Peter D Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, 2:369–409, 1994.
- [101] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, et al. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009.
- [102] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, et al. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011.

- Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/P11-1015>.
- [103] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [104] Patrick Wagner, Nils Strodthoff, Ralf-Dieter Bousseljot, Dieter Kreiseler, et al. PTB-XL, a large publicly available electrocardiography dataset. *Scientific Data*, 7(1), May 2020. doi: 10.1038/s41597-020-0495-6. URL <https://doi.org/10.1038/s41597-020-0495-6>.
- [105] Kevin P. Murphy. *Probabilistic machine learning: An introduction*. The MIT Press, 2022. ISBN 978-0-262-04682-4.
- [106] Dong C Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989.
- [107] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL <http://arxiv.org/abs/1412.6980>.
- [108] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. doi: 10.1023/a:1010933404324. URL <https://doi.org/10.1023/a:1010933404324>.
- [109] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16*, pages 785–794, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939785. URL <http://doi.acm.org/10.1145/2939672.2939785>.
- [110] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013. URL <http://arxiv.org/abs/1301.3781>.
- [111] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546. IEEE, 2005.

- [112] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.
- [113] Jacob Devlin, Ming-Wei Chang, Kenton Lee, et al. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, Jun 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL <https://aclanthology.org/N19-1423>.
- [114] Francois Chollet and François Chollet. *Deep learning with Python*. Simon and Schuster, 2021.
- [115] Tlameo Emmanuel, Thabiso Maupong, Dimane Mpoeleng, Thabo Semong, et al. A survey on missing data in machine learning. *Journal of Big Data*, 8:1–37, 2021.
- [116] Ardalan Mirzaei, Stephen R Carter, Asad E Patanwala, and Carl R Schneider. Missing data in surveys: Key concepts, approaches, and applications. *Research in Social and Administrative Pharmacy*, 18(2):2308–2316, 2022.
- [117] Kevin P Murphy. *Probabilistic machine learning: Advanced topics*. MIT press, 2023.
- [118] Derek Doran, Sarah Schulz, and Tarek R Besold. What does explainable ai really mean? A new conceptualization of perspectives. *Workshop in Comprehensibility and Explanation in AI and ML 2017*, 2017.
- [119] Dang Minh, H Xiang Wang, Y Fen Li, and Tan N Nguyen. Explainable artificial intelligence: A comprehensive review. *Artificial Intelligence Review*, pages 1–66, 2022.
- [120] Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41:647–665, 2014.
- [121] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.
- [122] Benoit Frenay and Michel Verleysen. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):845–869, May 2014. ISSN 2162-2388. doi: 10.1109/TNNLS.2013.2292894.

- [123] Don McNicol. *A Primer of Signal Detection Theory*. Psychology Press, New York, Nov 2004. ISBN 978-1-4106-1194-9. doi: 10.4324/9781410611949.
- [124] Ray J. Hickey. Noise modelling and evaluating learning from examples. *Artificial Intelligence*, 82 (1–2):157–179, Apr 1996. ISSN 00043702. doi: 10.1016/0004-3702(94)00094-8.
- [125] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):20–28, 1979. ISSN 0035-9254. doi: 10.2307/2346806.
- [126] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.
- [127] Xingquan Zhu and Xindong Wu. Class noise vs. attribute noise: A quantitative study. *Artificial Intelligence Review*, 22(3):177–210, Nov 2004. ISSN 1573-7462. doi: 10.1007/s10462-004-0751-8.
- [128] Junnan Li, Caiming Xiong, and Steven C.H. Hoi. Learning from noisy data with robust representation learning. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, page 9465–9474, Montreal, QC, Canada, Oct 2021. IEEE. ISBN 978-1-66542-812-5. doi: 10.1109/ICCV48922.2021.00935. URL <https://ieeexplore.ieee.org/document/9710292/>.
- [129] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, et al. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS’17*, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.
- [130] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, et al. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. Survey Certification.
- [131] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, page 1877–1901. Curran Associates, Inc., 2020.
- [132] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, et al. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

- [133] Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, et al. TabLLM: Few-shot classification of tabular data with large language models. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, page 5549–5581. PMLR, Apr 2023.
- [134] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, et al. PaLM: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023. ISSN 1533-7928.
- [135] Hugo Touvron, Louis Martin, Kevin Stone, et al. Llama 2: Open foundation and fine-tuned chat models, 2023.
- [136] Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, et al. Large language models in medicine. *Nature Medicine*, 29(8):1930–1940, Jul 2023. ISSN 1546-170X. doi: 10.1038/s41591-023-02448-8. URL <http://dx.doi.org/10.1038/s41591-023-02448-8>.
- [137] Harsha Nori, Nicholas King, Scott Mayer McKinney, Dean Carignan, and Eric Horvitz. Capabilities of GPT-4 on medical challenge problems. *arXiv preprint arXiv:2303.13375*, 2023.
- [138] Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, et al. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180, Jul 2023. ISSN 1476-4687. doi: 10.1038/s41586-023-06291-2. URL <http://dx.doi.org/10.1038/s41586-023-06291-2>.
- [139] Daniel Kerrigan, Jessica Hullman, and Enrico Bertini. A survey of domain knowledge elicitation in applied machine learning. *Multimodal Technologies and Interaction*, 5(12):73, Nov 2021. ISSN 2414-4088. doi: 10.3390/mti5120073. URL <http://dx.doi.org/10.3390/mti5120073>.
- [140] Paul H Garthwaite, Joseph B Kadane, and Anthony O’Hagan. Statistical methods for eliciting probability distributions. *Journal of the American Statistical Association*, 100(470):680–701, Jun 2005. ISSN 1537-274X.
- [141] Anthony O’Hagan, Caitlin E. Buck, Alireza Daneshkhah, J. Richard Eiser, et al. *Uncertain Judgements: Eliciting Experts’ Probabilities*. Wiley, Jul 2006. ISBN 9780470033319. doi: 10.1002/0470033312. URL <http://dx.doi.org/10.1002/0470033312>.
- [142] Jeremy Oakley and Tony O’Hagan, 2019. URL <https://shelf.sites.sheffield.ac.uk/>.
- [143] Alan Godfrey, Matthew Brodie, KS van Schooten, Mina Nouredanesh, et al. Inertial wearables as pragmatic tools in dementia. *Maturitas*, 127:12–17, 2019.

- [144] Vyara Valkanova and Klaus P Ebmeier. What can gait tell us about dementia? Review of epidemiological and neuropsychological evidence. *Gait & Posture*, 53:215–223, 2017.
- [145] Lindsay Bahureksa, Bijan Najafi, Ahlam Saleh, Marwan Sabbagh, et al. The impact of mild cognitive impairment on gait and balance: A systematic review and meta-analysis of studies using instrumented assessment. *Gerontology*, 63(1):67–83, 2016.
- [146] Mina Nouredanesh, Alan Godfrey, Jennifer Howcroft, Edward D Lemaire, and James Tung. Fall risk assessment in the wild: A critical examination of wearable sensor use in free-living conditions. *Gait & Posture*, 85:178–190, 2021.
- [147] Majd Alwan. Passive in-home health and wellness monitoring: Overview, value and examples. In *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4307–4310. IEEE, 2009.
- [148] Shirin Enshaeifar, Payam Barnaghi, Severin Skillman, David Sharp, et al. A digital platform for remote healthcare monitoring. In *Companion Proceedings of the Web Conference 2020*, WWW '20, page 203–206, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370240.
- [149] Bo Tan, Alison Burrows, Robert Piechocki, Ian Craddock, et al. Wi-Fi based passive human motion sensing for in-home healthcare applications. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 609–614. IEEE, 2015.
- [150] Qingchao Chen, Yang Liu, Bo Tan, Karl Woodbridge, and Kevin Chetty. Respiration and activity detection based on passive radio sensing in home environments. *IEEE Access*, 8:12426–12437, 2020.
- [151] Charalambos Hadjipanayi, Maowen Yin, Alan Bannan, Adrien Rapeaux, et al. Remote gait analysis using ultra-wideband radar technology based on joint range-doppler-time representation. *IEEE Transactions on Biomedical Engineering*, 2024.
- [152] Paul Little, Sheila Turner, Kate Rumsby, Greg Warner, et al. Developing clinical rules to predict urinary tract infection in primary care settings: Sensitivity and specificity of near patient tests (dipsticks) and clinical scores. *British Journal of General Practice*, 56(529):606–612, 2006.
- [153] Warren J McIsaac, Rahim Moineddin, and Sue Ross. Validation of a decision aid to assist physicians

- in reducing unnecessary antibiotic drug use for acute cystitis. *Archives of Internal Medicine*, 167(20):2201–2206, 2007.
- [154] Lindsey C. Yourman, Tyler J. Kent, Juhi S. Israni, Kelly J. Ko, and Adriane Lesser. Association of dementia diagnosis with urinary tract infection in the emergency department. *Journal of the American College of Emergency Physicians Open*, 1(6):1291–1296, Oct 2020. doi: 10.1002/emp2.12268. URL <https://doi.org/10.1002/emp2.12268>.
- [155] Marilyn J Rantz, Marjorie Skubic, Richelle J Koopman, Lorraine Phillips, et al. Using sensor networks to detect urinary tract infections in older adults. *Proceedings of IEEE International Conference on e-Health Networking Applications and Services (Healthcom)*, pages 142–149, 2011.
- [156] Bo Han, Quanming Yao, Tongliang Liu, Gang Niu, et al. A survey of label-noise representation learning: Past, present and future. *CoRR*, abs/2011.04406, 2020. URL <https://arxiv.org/abs/2011.04406>.
- [157] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, et al. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *NeurIPS*, pages 8535–8545, 2018.
- [158] Qizhou Wang, Bo Han, Tongliang Liu, Gang Niu, et al. Tackling instance-dependent label noise via a universal probabilistic model. In *AAAI*, pages 10183–10191, 2021.
- [159] Jiaheng Wei, Hangyu Liu, Tongliang Liu, Gang Niu, et al. To smooth or not? When label smoothing meets noisy labels. In *International Conference on Machine Learning*, pages 23589–23614. PMLR, 2022.
- [160] WA Wright. Bayesian approach to neural-network modeling with input uncertainty. *IEEE Transactions on Neural Networks*, 10(6):1261–1270, 1999.
- [161] Jakub Konečný, H Brendan McMahan, Felix X Yu, Peter Richtárik, et al. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.
- [162] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.
- [163] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, et al. Generalizing to unseen domains: A survey on domain generalization. *IEEE Transactions on Knowledge and Data Engineering*, 35(8):8052–8072, 2022.

- [164] Shanghua Gao, Ada Fang, Yepeng Huang, Valentina Giunchiglia, et al. Empowering biomedical discovery with AI agents. *Cell*, 187(22):6125–6151, 2024.
- [165] Kristy Choi, Chris Cundy, Sanjari Srivastava, and Stefano Ermon. LMPriors: Pre-trained language models as task-specific priors. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022.
- [166] Joris M Mooij, Jonas Peters, Dominik Janzing, Jakob Zscheischler, and Bernhard Schölkopf. Distinguishing cause from effect using observational data: Methods and benchmarks. *Journal of Machine Learning Research*, 17(32):1–102, 2016.
- [167] Noah Hollmann, Samuel Müller, and Frank Hutter. Large language models for automated data science: Introducing caafe for context-aware automated feature engineering. *Advances in Neural Information Processing Systems*, 36, 2024.
- [168] Shujian Zhang, Chengyue Gong, Lemeng Wu, Xingchao Liu, and Mingyuan Zhou. AutoML-GPT: Automatic machine learning with GPT, 2023. URL <https://arxiv.org/abs/2305.02499>.
- [169] Lei Zhang, Yuge Zhang, Kan Ren, Dongsheng Li, and Yuqing Yang. MLCopilot: Unleashing the power of large language models in solving machine learning tasks. In Yvette Graham and Matthew Purver, editors, *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2931–2959, St. Julian’s, Malta, Mar 2024. Association for Computational Linguistics.
- [170] Henry Gouk and Boyan Gao. Automated prior elicitation from large language models for Bayesian logistic regression. In *The 3rd International Conference on Automated Machine Learning*, 2024.
- [171] Sebastian Bordt, Harsha Nori, Vanessa Cristiny Rodrigues Vasconcelos, Besmira Nushi, and Rich Caruana. Elephants never forget: Memorization and learning of tabular data in large language models. In *First Conference on Language Modeling*, 2024.
- [172] David Selby, Kai Spriestersbach, Yuichiro Iwashita, Dennis Bappert, et al. Quantitative knowledge retrieval from large language models. *arXiv preprint arXiv:2402.07770*, 2024.
- [173] Jian-Qiao Zhu and Thomas L Griffiths. Eliciting the priors of large language models using iterated in-context learning. *arXiv preprint arXiv:2406.01860*, 2024.

- [174] Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit Bayesian inference. In *International Conference on Learning Representations*, 2021.
- [175] Sewon Min, Xinxu Lyu, Ari Holtzman, Mikel Artetxe, et al. Rethinking the role of demonstrations: What makes in-context learning work? In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, page 11048–11064. Association for Computational Linguistics, Dec 2022. doi: 10.18653/v1/2022.emnlp-main.759.
- [176] Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? Investigations with linear models. In *The Eleventh International Conference on Learning Representations*, 2022.
- [177] Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, et al. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR, 2023.
- [178] Siddhartha Dalal and Vishal Misra. The matrix: A Bayesian learning model for LLMs. *arXiv preprint arXiv:2402.03175*, 2024.
- [179] James Requeima, John F Bronskill, Dami Choi, Richard E. Turner, and David Duvenaud. LLM processes: Numerical predictive distributions conditioned on natural language. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL <https://openreview.net/forum?id=HShs7q1Njh>.
- [180] Noah Hollmann, Samuel Müller, Katharina Eggenberger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations*, 2022.
- [181] Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, et al. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, Jan 2025. ISSN 1476-4687.
- [182] Yaqing Wang, Quanming Yao, James T. Kwok, et al. Generalizing from a few examples: A survey on few-shot learning. *ACM Comput. Surv.*, 53(3), Jun 2020. ISSN 0360-0300. doi: 10.1145/3386252. URL <https://doi.org/10.1145/3386252>.

- [183] Raia Hadsell, Dushyant Rao, Andrei A. Rusu, et al. Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 24(12):1028–1040, Dec 2020. ISSN 13646613. doi: 10.1016/j.tics.2020.09.004.
- [184] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2): 423–443, Feb 2019. ISSN 1939-3539. doi: 10.1109/TPAMI.2018.2798607.
- [185] Peng Xu, Xiatian Zhu, and David A. Clifton. Multimodal learning with transformers: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1–20, 2023. ISSN 0162-8828, 2160-9292, 1939-3539. doi: 10.1109/TPAMI.2023.3275156.
- [186] Abdullah Alanazi. Using machine learning for healthcare challenges and opportunities. *Informatics in Medicine Unlocked*, 30:100924, Jan 2022. ISSN 2352-9148. doi: 10.1016/j.imu.2022.100924.
- [187] Alec Radford, Jong Wook Kim, Chris Hallacy, et al. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, page 8748–8763. PMLR, Jul 2021. URL <https://proceedings.mlr.press/v139/radford21a.html>.
- [188] Tomas Mikolov, Ilya Sutskever, Kai Chen, et al. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.
- [189] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.
- [190] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, et al. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL <https://openreview.net/forum?id=gerNCVqqtR>.
- [191] Tianyang Lin, Yuxin Wang, Xiangyang Liu, et al. A survey of transformers. *AI Open*, 3:111–132, 2022. ISSN 26666510. doi: 10.1016/j.aiopen.2022.10.001.
- [192] Nate Gruver, Marc Anton Finzi, Shikai Qiu, and Andrew Gordon Wilson. Large language models are zero-shot time series forecasters. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

- [193] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, et al. Time-llm: Time series forecasting by reprogramming large language models. In *International Conference on Learning Representations*, 2024.
- [194] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.
- [195] Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *International Conference on Machine Learning (ICML)*, 2024.
- [196] Jason Wei, Maarten Bosma, Vincent Zhao, et al. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2021.
- [197] Monica Agrawal, Stefan Hegselmann, Hunter Lang, et al. Large language models are few-shot clinical information extractors. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1998–2022, Abu Dhabi, United Arab Emirates, Dec 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.130. URL <https://aclanthology.org/2022.emnlp-main.130>.
- [198] Ning Ding, Yujia Qin, Guang Yang, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(33):220–235, Mar 2023. ISSN 2522-5839. doi: 10.1038/s42256-023-00626-4.
- [199] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, et al. Language models as knowledge bases? In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, page 2463–2473, Hong Kong, China, Nov 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1250. URL <https://aclanthology.org/D19-1250>.
- [200] Damir Krinitski, Rafal Kasina, Stefan Klöppel, and Eric Lenouvel. Associations of delirium with urinary tract infections and asymptomatic bacteriuria in adults aged 65 and older: A systematic review and meta-analysis. *Journal of the American Geriatrics Society*, 69(11):3312–3323, Aug 2021. doi: 10.1111/jgs.17418. URL <https://doi.org/10.1111/jgs.17418>.

- [201] Henry J Woodford and James George. Diagnosis and management of urinary infections in older people. *Clinical Medicine*, 11(1):80–83, Feb 2011. doi: 10.7861/clinmedicine.11-1-80. URL <https://doi.org/10.7861/clinmedicine.11-1-80>.
- [202] Walter LJM Devillé, Joris C Yzermans, Nico P van Duijn, and Dick Bezemer. The urine dipstick test useful to rule out infections. A meta-analysis of the accuracy. *BMC Urology*, 2004.
- [203] Honglin Li, Magdalena Anita Kolanko, Shirin Enshaeifar, Severin Skillman, et al. Deep representation for connected health: Semi-supervised learning for analysing the risk of urinary tract infections in people with dementia. *arXiv preprint arXiv:2011.13916*, 2020. doi: 10.48550/arxiv.2011.13916. URL <https://arxiv.org/abs/2011.13916>.
- [204] Marco Aste, Massimo Boninsegna, Antonino Freno, and Edmondo Trentin. Techniques for dealing with incomplete data: A tutorial and survey. *Pattern Analysis and Applications*, 18(1):1–29, Sep 2014. doi: 10.1007/s10044-014-0411-9. URL <https://doi.org/10.1007/s10044-014-0411-9>.
- [205] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer New York, 2009. doi: 10.1007/978-0-387-84858-7. URL <https://doi.org/10.1007/978-0-387-84858-7>.
- [206] Jesse Davis and Mark Goadrich. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine Learning, ICML '06*, page 233–240, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933832. doi: 10.1145/1143844.1143874. URL <https://doi.org/10.1145/1143844.1143874>.
- [207] Lorenzo Beretta and Alessandro Santaniello. Nearest neighbor imputation algorithms: A critical evaluation. *BMC Medical Informatics and Decision Making*, 16(S3), Jul 2016. doi: 10.1186/s12911-016-0318-z. URL <https://doi.org/10.1186/s12911-016-0318-z>.
- [208] Shangzhi Hong and Henry S. Lynn. Accuracy of random-forest-based imputation of missing data in the presence of non-normality, non-linearity, and interaction. *BMC Medical Research Methodology*, 20(1), Jul 2020. doi: 10.1186/s12874-020-01080-1. URL <https://doi.org/10.1186/s12874-020-01080-1>.
- [209] W. J. Youden. Index for rating diagnostic tests. *Cancer*, 3(1):32–35, 1950. ISSN 1097-0142. doi:

- 10.1002/1097-0142(1950)3:1<32::aid-cnrcr2820030106>3.0.co;2-3. URL [http://dx.doi.org/10.1002/1097-0142\(1950\)3:1<32::AID-CNCR2820030106>3.0.CO;2-3](http://dx.doi.org/10.1002/1097-0142(1950)3:1<32::AID-CNCR2820030106>3.0.CO;2-3).
- [210] Theresa A Rowe and Manisha Juthani-Mehta. Urinary tract infection in older adults. *Aging Health*, 9(5):519–528, 2013.
- [211] Charles R. Harris, K. Jarrod Millman, Stefan J. van der Walt, Ralf Gommers, et al. Array programming with NumPy. *Nature*, 585(7825):357–362, Sep 2020. doi: 10.1038/s41586-020-2649-2. URL <https://doi.org/10.1038/s41586-020-2649-2>.
- [212] Wes McKinney. Data structures for statistical computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56–61, 2010. doi: 10.25080/Majora-92bf1922-00a.
- [213] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [214] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems* 32, pages 8024–8035. Curran Associates, Inc., 2019.
- [215] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, et al., editors, *Advances in Neural Information Processing Systems* 30, pages 4765–4774. Curran Associates, Inc., 2017.
- [216] Morris H. DeGroot and Stephen E. Fienberg. The comparison and evaluation of forecasters. *The Statistician*, 32(1/2):12, Mar 1983. ISSN 00390526. doi: 10.2307/2987588.
- [217] Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine learning - ICML '05*, page 625–632, Bonn, Germany, 2005. ACM Press. ISBN 978-1-59593-180-1. doi: 10.1145/1102351.1102430.
- [218] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR, 06–11 Aug 2017. URL <https://proceedings.mlr.press/v70/guo17a.html>.

- [219] Scott M Lundberg, Bala Nair, Monica S Vavilala, Mayumi Horibe, et al. Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. *Nature Biomedical Engineering*, 2(10):749, 2018.
- [220] Léa Deltourbe, Livia Lacerda Mariano, Teri N. Hreha, David A. Hunstad, and Molly A. Ingersoll. The impact of biological sex on diseases of the urinary tract. *Mucosal Immunology*, 15(5):857–866, Aug 2022. ISSN 19330219. doi: 10.1038/s41385-022-00549-0.
- [221] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Computing Surveys*, 54(6):1–35, Jul 2022. ISSN 0360-0300, 1557-7341. doi: 10.1145/3457607.
- [222] James Pickett, Cathy Bird, Clive Ballard, Sube Banerjee, et al. A roadmap to advance dementia research in prevention, diagnosis, intervention, and care by 2025. *International Journal of Geriatric Psychiatry*, 33(7):900–906, 2018.
- [223] Abdullatif Aydin, Kamran Ahmed, Iftikhar Zaman, Muhammad Shamim Khan, and Prokar Dasgupta. Recurrent urinary tract infections in women. *International Urogynecology Journal*, 26(6):795–804, Nov 2014. doi: 10.1007/s00192-014-2569-5. URL <https://doi.org/10.1007/s00192-014-2569-5>.
- [224] Nicholas Harrington, David Torres Barba, Quan M. Bui, Andrew Wassell, et al. Nocturnal respiratory rate dynamics enable early recognition of impending hospitalizations. *medRxiv preprint*, 2022. doi: 10.1101/2022.03.10.22272238.
- [225] Christopher W. Seymour, Vincent X. Liu, Theodore J. Iwashyna, Frank M. Brunkhorst, et al. Assessment of clinical criteria for sepsis: For the third international consensus definitions for sepsis and septic shock (sepsis-3). *JAMA*, 315(8):762, Feb 2016. ISSN 0098-7484. doi: 10.1001/jama.2016.0288.
- [226] Gary B. Smith, David R. Prytherch, Paul Meredith, Paul E. Schmidt, and Peter I. Featherstone. The ability of the national early warning score (NEWS) to discriminate patients at risk of early cardiac arrest, unanticipated intensive care unit admission, and death. *Resuscitation*, 84(4):465–470, Apr 2013. ISSN 03009572. doi: 10.1016/j.resuscitation.2012.12.016.
- [227] C.P. Subbe. Validation of a modified early warning score in medical admissions. *QJM*, 94(10):521–526, Oct 2001. ISSN 14602393. doi: 10.1093/qjmed/94.10.521.

- [228] Martin J. Tobin, Tejvir S. Chadha, Gilbert Jenouri, Stephen J. Birch, et al. Breathing patterns. *Chest*, 84(2):202–205, Aug 1983. ISSN 00123692. doi: 10.1016/S0012-3692(15)33498-X.
- [229] Emmanuel Chartier-Kastler and Kate Davidson. Evaluation of quality of life and quality of sleep in clinical practice. *European Urology Supplements*, 6(9):576–584, 2007. ISSN 1569-9056.
- [230] Yanyao Shen and Sujay Sanghavi. Learning with bad training data via iterative trimmed loss minimization. In *International Conference on Machine Learning*, pages 5739–5748. PMLR, 2019.
- [231] Eric Arazo, Diego Ortego, Paul Albert, Noel O’Connor, and Kevin McGuinness. Unsupervised label noise modeling and loss correction. In *International Conference on Machine Learning*, pages 312–321. PMLR, 2019.
- [232] Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, et al. How does disagreement help generalization against label corruption? In *International Conference on Machine Learning*, pages 7164–7173. PMLR, 2019.
- [233] Sanyam Kapoor, Wesley J Maddox, Pavel Izmailov, and Andrew G Wilson. On uncertainty, tempering, and data augmentation in Bayesian classification. *Advances in Neural Information Processing Systems*, 35:18211–18225, 2022.
- [234] Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, et al. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic, Nov 2021. Association for Computational Linguistics. URL <https://aclanthology.org/2021.emnlp-demo.21>.
- [235] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, Oct 2020. Association for Computational Linguistics. URL <https://www.aclweb.org/anthology/2020.emnlp-demos.6>.
- [236] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

- [237] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.
- [238] W.Y. Wong, R. Sudirman, N.H. Mahmood, S.Z. Tumari, and N. Samad. Study of environment based condition of electromagnetic interference during ecg acquisition. In *2012 International Conference on Biomedical Engineering (ICoBE)*, pages 579–584, 2012. doi: 10.1109/ICoBE.2012.6178983.
- [239] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. URL <https://arxiv.org/abs/1412.6980>.
- [240] Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609, Dec 2022. ISSN 1041-4347, 1558-2191, 2326-3865. doi: 10.1109/TKDE.2021.3070203.
- [241] Georgios Spithourakis and Sebastian Riedel. Numeracy for language models: Evaluating and improving their ability to predict numbers. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2104–2115, 2018.
- [242] John Paul Gosling. SHELF: The sheffield elicitation framework. *Elicitation: The science and art of structuring judgement*, pages 61–93, 2018.
- [243] Gábor J Székely and Maria L Rizzo. Energy statistics: A class of statistics based on distances. *Journal of Statistical Planning and Inference*, 143(8):1249–1272, 2013.
- [244] Gábor J Székely and Maria L Rizzo. On the uniqueness of distance covariance. *Statistics & Probability Letters*, 82(12):2278–2282, 2012.
- [245] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, Apr 1970. ISSN 0006-3444. doi: 10.1093/biomet/57.1.97. URL <http://dx.doi.org/10.1093/biomet/57.1.97>.
- [246] Matthew D Hoffman, Andrew Gelman, et al. The No-U-Turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.
- [247] Alberto Cabezas, Adrien Corenflos, Junpeng Lao, Rémi Louf, et al. BlackJAX: Composable Bayesian inference in JAX. *arXiv preprint arXiv:2402.10797*, 2024.

- [248] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, et al. JAX: Composable transformations of Python + NumPy programs, 2018. URL <http://github.com/jax-ml/jax>.
- [249] Oriol Abril-Pla, Virgile Andreani, Colin Carroll, Larry Dong, et al. PyMC: A modern, and comprehensive probabilistic programming framework in Python. *PeerJ Computer Science*, 9: e1516, Sep 2023. ISSN 2376-5992. doi: 10.7717/peerj-cs.1516. URL <http://dx.doi.org/10.7717/peerj-cs.1516>.
- [250] Vladimir I Levenshtein. Binary codes capable of correcting spurious insertions and deletions of ones. *Problems of Information Transmissions*, 1(1):8–17, 1965.
- [251] V Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Proceedings of the Soviet Physics Doklady*, 1966.
- [252] MetaAI. Introducing Meta Llama 3: The most capable openly available LLM to date, 2024. URL <https://ai.meta.com/blog/meta-llama-3/>.
- [253] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. The Llama 3 herd of models, 2024. URL <https://arxiv.org/abs/2407.21783>.
- [254] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.
- [255] Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL <https://qwenlm.github.io/blog/qwen2.5/>.
- [256] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning, 2025. URL <https://arxiv.org/abs/2501.12948>.
- [257] Andy L Jones. Scaling scaling laws with board games. *arXiv preprint arXiv:2104.03113*, 2021.
- [258] Randall Balestriero, Mark Ibrahim, Vlad Sobal, et al. A cookbook of self-supervised learning. *arXiv preprint arXiv:2304.12210*, 2023.
- [259] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL <https://arxiv.org/abs/1908.10084>.

- [260] Wenhui Wang, Furu Wei, Li Dong, et al. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems*, volume 33, page 5776–5788. Curran Associates, Inc., 2020.
- [261] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *Similarity-Based Pattern Recognition: Third International Workshop, SIMBAD 2015, Copenhagen, Denmark, October 12-14, 2015. Proceedings 3*, pages 84–92. Springer, 2015.
- [262] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=Bkg6RiCqY7>.
- [263] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL <http://jmlr.org/papers/v9/vandermaten08a.html>.
- [264] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics. ISBN 9780898716245.
- [265] Yushan Huang, Yuchen Zhao, Alexander Capstick, Francesca Palermo, et al. Analyzing entropy features in time-series data for pattern recognition in neurological conditions. *Artificial Intelligence in Medicine*, page 102821, 2024. ISSN 0933-3657. doi: <https://doi.org/10.1016/j.artmed.2024.102821>. URL <https://www.sciencedirect.com/science/article/pii/S0933365724000630>.
- [266] Mary Sano, Jeffrey Cummings, Stefanie Auer, Sverre Bergh, et al. Agitation in cognitive disorders: Progress in the international psychogeriatric association consensus clinical and research definition. *International Psychogeriatrics*, 36(4):238–250, 2024.
- [267] Kazuko Hiyoshi-Taniguchi, Carl B. Becker, and Ayae Kinoshita. What behavioral and psychological symptoms of dementia affect caregiver burnout? *Clinical Gerontologist*, 41(3):249–254, Dec 2017. ISSN 1545-2301. doi: 10.1080/07317115.2017.1398797.
- [268] Howard Fillit, Myrlene S Aigbogun, Patrick Gagnon-Sanschagrin, Martin Cloutier, et al. Impact of agitation in long-term care residents with dementia in the united states. *International Journal of Geriatric Psychiatry*, 36(12):1959–1969, 2021.

- [269] Saturnino Luz, Fasih Haider, Sofia De la Fuente, Davida Fromm, and Brian MacWhinney. Detecting cognitive decline using speech only: The ADReSSo Challenge. *arXiv preprint arXiv:2104.09356*, 2021.
- [270] Shu-Yi Huang, Yu-Zhu Li, Ya-Ru Zhang, Yu-Yuan Huang, et al. Sleep, physical activity, sedentary behavior, and risk of incident dementia: A prospective cohort study of 431,924 UK Biobank participants. *Molecular Psychiatry*, 27(10):4343–4354, 2022.
- [271] Michael R Irwin and Michael V Vitiello. Implications of sleep disturbance and inflammation for Alzheimer’s disease dementia. *The Lancet Neurology*, 18(3):296–306, 2019.
- [272] Louise Rigny, Nan Fletcher-Lloyd, Alexander Capstick, Ramin Nilforooshan, et al. Assessment of sleep patterns in dementia and general population cohorts using passive in-home monitoring technologies. *Nature Communications Medicine*, 4(1):222, 2024. URL <https://www.nature.com/articles/s43856-024-00646-0>.
- [273] Chloe Walsh, Alexander Capstick, Nan Fletcher-Lloyd, Jessica True, et al. Longitudinal study of care needs and behavioural changes in people living with dementia using in-home assessment data. *Nature Communications Medicine*, 5(1):14, 2025. URL <https://www.nature.com/articles/s43856-024-00724-3>.
- [274] Alexander Capstick, Francesca Palermo, Honglin Li, Payam Barnaghi, et al. Clinically relevant and explainable feature extraction from remote monitoring data for the prediction of urinary tract infections. *Alzheimer’s & Dementia, Wiley*, 19:e064683, 2023. URL <https://alz-journals.onlinelibrary.wiley.com/doi/full/10.1002/alz.064683>.