Towards Explainable Weather Forecasting Through FastLAS

Talissa Dreossi[1], Agostino Dovier[1], Andrea Formisano[1], Mark Law[2], Agostino Manzato[3], Alessandra Russo[4], and Matthew Tait[2]

[1] University of Udine, Udine (Italy) name.surname@uniud.it
[2] ILASP Limited, London (UK) mark@ilasp.com, m.tait@ilasp.com
[3] ARPA FVG - OSMER, Palmanova (Udine, Italy), agostino.manzato@arpa.fvg.it
[4] Imperial College London (UK) a.russo@imperial.ac.uk

Abstract. Weather forecasting is important for saving lives, protecting property, and supporting economic activities. It provides timely warnings for severe weather, improves agricultural planning, and aids in disaster management. Neural networks and deep learning methods can achieve impressive accuracy in weather prediction, but their black-box nature lacks in explainability. To address this limitation, we investigated the potential of FastLAS, an Inductive Logic Programming (ILP) framework, to produce reliable and, more important, explainable weather predictions. FastLAS learns ASP programs whose syntax and structural semantics resemble natural human language, making them easily understandable and interpretable by humans. The supportedness of stable models allows a clear explanation of the predictions. Our empirical evaluation on data from an Italian weather forecasting center shows that our approach is capable of learning predictive models from small dataset (a few samples instead of the thousands needed by neural networks) achieving an accuracy higher than statistical machine learning base lines.

Keywords: ILP · Logic Programming · AI Explainability

1 Introduction

Weather forecasting is essential for anticipating and reducing the impact of severe weather events, protecting lives and property. It also plays a key role in agriculture, transportation, and disaster management, helping ensure timely and well-informed decisions. Traditional methods of weather prediction have greatly improved with neural networks and deep learning techniques, which offer impressive accuracy. However, these methods often function like black boxes, making their predictions hard to understand. This lack of explainability and interpretability can be disadvantageous. Explainability in weather prediction is critical for several reason: it enhances user trust, facilitates the validation of model outputs by domain experts, and aids in the refinement of forecasting models. An overview of several topic related to machine learning explainability in the field of weather prediction is well described by Flora et al. [9]. For these reasons, this work aims

to create a reliable system capable of predicting rain levels while also providing explanations. This is accomplished using the Inductive Logic Programming (ILP) system FastLAS [14]. FastLAS is designed for speed and scalability, capable of handling large datasets and complex background knowledge efficiently. It incorporates a scoring function to find optimal solutions customized to specific needs. By focusing on a smaller, OPT-sufficient subset of possible solutions, it can quickly identify the best-fitting hypotheses, making it useful for our purpose.

The paper is organized as follows. After briefly recalling the basic notions of Answer Set Programming, Sect. 2 reviews weather prediction methods and introduces FastLAS highlighting its features and how it works. Sect. 3 briefly describe some related work. Sect. 4 describes our approach to modeling weather prediction using FastLAS. Finally, Sect. 5 presents and analyzes the experimental results, evaluating forecast accuracy, comparing with existing methods, and discussing our outcome.

2 Background

Answer Set Programming (ASP) is a declarative programming paradigm born for non-monotonic reasoning and, thanks to the efficiency of ASP solvers, widely used for modeling and solving difficult combinatorial problems. A program is a set of rules of the form $H :- A_1, \ldots, A_n, \texttt{not}\, B_1, \ldots, \texttt{not}\, B_m$, where H, A_i, and B_j are atoms of the form $p(t_1, \ldots, t_n)$ where p is a predicate symbol, t_i are constant or variable symbols, and $n \geq 0$. If a rule r has not variable symbols, it is said to be *ground*. H is the *head* of r and $A_1, \ldots, A_n, \texttt{not}\, B_1, \ldots, \texttt{not}\, B_m$ is the *body* of r. A literal is either an atom A or its default negation $\texttt{not}\, A$, i.e., a *naf-literal*. Rules with an empty body (m=n=0) are called facts. Rules without the head (corresponding to the case $H = \texttt{false}$) are called constraints or denials. The body of a constraint is a conjunction of literals that cannot be satisfied simultaneously. Note that a constraint can be replaced by a rule having as head a fresh atom q and a body made of $\texttt{not}\, q$ and the literals of the constraint.

Semantics of ASP programs is based on the notion of *stable models* that applies to the ground version of the programs (in short, the ground version of a program P is the program containing all the instances of the rules of P obtained by replacing, in all possible ways, all the variables with constant symbols occurring in P). A set of ground atoms S is a stable model of a program P if it is the unique minimum model of the reduct P^S of P. The reduct is obtained from P by removing all rules whose body is not satisfied by the atoms in S, and removing all naf-literals from the remaining rules [10].

An ASP solver is capable of determining whether a program P admits or not any stable model, and, in the latter case, of computing the set $AS(P)$ of all such models. An important property of stable models is that if an atom A belongs to a stable model S, then there is (at least) a rule r in (the ground version of) P whose body is satisfied by S and such that A is the head r. Such r allows to generate an explanation of the truth of A, that is: we not only know that A is true but we also know the reason for this (i.e., the rule r and the true literals in

its body) — note that since the same atom can occur as head of different rules, in general there can be multiple alternative explanation.

The ASP language admits various forms of syntactic extensions. One of these concerns constructs that introduce controlled forms of non-determinism in programs, such as *choice rules*, *cardinality constraints*, etc. Also, *built-in atoms* of the form $E_1\ op\ E_2$, which we will refer to as *conditions*, can be used in rules body to model arithmetic comparisons between numerical expressions . Here E_1 and E_2 are expressions involving numerical constants, variables, and arithmetic operators, whereas $op \in \{<, \leq, =, !=, >, \geq\}$. During grounding the variables in these expressions are replaced by constants, then the expressions are evaluated and replaced by `true/false` depending on the values obtained.

Inductive Logic Programming (ILP) and FastLAS. ILP [21] is a subfield of machine learning that focuses on learning logical rules from examples and background knowledge. The goal is to find a hypothesis (a set of logical rules) that explains the given examples within the context of the background knowledge (namely, a set of rules). Different ILP frameworks have been proposed in the literature [13]. In the context of this paper, we are interested in the *Learning from Answer Sets (LAS)*, a state-of-the-art ILP framework for learning from noisy examples [16].

LAS [17] is a paradigm for learning answer set programs. A learning problem associated with LAS is called a LAS *task*; precisely, a LAS task is a tuple $T = \langle B, S_M, E\rangle$ where B is an ASP program called *background knowledge*, S_M is a set of rules allowed to be in a hypothesis, known as *hypothesis space*, and E is the set of *examples*. The hypothesis space is specified through a *mode bias*: a pair of sets of mode declarations $\langle M_h, M_b\rangle$, where M_h (resp. M_b) are called *head* (resp. *body*) *mode declarations*. They describe which predicates can appear in the head or body of a rule, respectively. A mode declaration is a literal whose arguments are either `var(t)` or `const(t)`, for some constant `t` (called a *type*). Informally, a literal is *compatible* with a mode declaration m if it can be constructed by replacing every instance of `var(t)` in m with a variable of type `t`, and every `const(t)` in m with a constant of type `t`.

In a LAS framework, examples are based on the notion of a *partial interpretation*. A partial interpretation e_{pi} is a pair of sets of ground atoms $\langle e^{inc}, e^{exc}\rangle$, where e^{inc} and e^{exc} are referred to as the *inclusions* and *exclusions* sets respectively. An interpretation I is said to *extend* a partial interpretation e_{pi} if and only if $e^{inc} \subseteq I$ and $e^{exc} \cap I = \emptyset$. Differently from conventional ILP where the background knowledge is the same for all examples, in a LAS framework examples may be context-dependent. This means that some examples should be explained in the context of some information, whereas others should be explained in different contexts. A *context-dependent partial interpretation* e_{cdpi} is a tuple $e = \langle e_{pi}, e_{ctx}\rangle$, where e_{pi} is a partial interpretation and e_{ctx} is an ASP program, called the *context* of the example. A program P is said to *accept* e_{cdpi} if there is at least one stable model A of $P \cup e_{ctx}$ that extends e_{pi}.

In real world setting, data are *noisy*. Noise can be captured by allowing examples to be weighted with a notion of *penalty*. A *weighted context-dependent*

partial interpretation (WCDPI) e is a tuple $\langle e_{id}, e_{pen}, e_{cdpi} \rangle$, where e_{id} is an identifier for e, e_{pen} is a positive integer, called a *penalty*, e_{cdpi} is a context-dependent partial interpretation. A task is called *noisy LAS* task when examples are WCDPIs. We can now formally define the notion of a noisy LAS task.

Definition 1. *A* Noisy LAS *task is a tuple* $T^{noise} = \langle B, S_M, E \rangle$ *where* B *is an ASP program,* S_M *is the search space, and* E *is a finite set of WCDPIs. A hypothesis* $H \subseteq S_M$ *is an inductive solution of* T^{noise} *iff* $\forall e \in E$, $B \cup H$ *accepts* e. *We denote with* $\mathcal{T}^{noise}$ *the set of all Noisy LAS tasks.*

If a hypothesis does not accept an example, it *pays the penalty* of that example. Informally, penalties are used to calculate the *cost* associated with an hypothesis for not explaining examples. The cost function of a hypothesis is the sum over the penalties of all of the examples that are not accepted by the hypothesis, augmented with the length of the hypothesis. A *scoring function* is a function $\mathcal{S} : \text{ASP} \times \mathcal{T}^{noise} \to \mathbb{R}_{>0}$, where ASP is the set of all possible ASP programs. The goal of a *noisy* LAS tasks is to find a hypothesis that minimises the cost function over a given hypothesis space with respect to a given set of WCDPI examples. We referred to such an hypothesis as an *optimal* hypothesis. This is formally defined as follows.

Definition 2. *Let* $\mathcal{S}$ *be a scoring function. A hypothesis* H *is said to be an* optimal solution *of a noisy LAS task* T^{noise} *w.r.t.* $\mathcal{S}$ *if* H *is a solution of* T^{noise} *and there is no other solution* H' *of* T^{noise} *such that* $\mathcal{S}(H', T^{noise}) < \mathcal{S}(H, T^{noise})$.

FastLAS [14] is a noisy LAS system. It supports user-defined scoring functions, allowing domain-specific optimisation criteria to be used to bias the search; for example, scoring functions can be used to bias towards the cheapest, least risky, safest or most secure set of rules. Furthermore, continuous data types (such as real numbers) are common in machine learning; for example, when data is being collected from sensors. Many ILP approaches are unable to deal with such data types (without some form of discretisation). FastLAS supports numeric data types, together with binary comparisons over these types.

Example 1. Consider a task with an empty background knowledge, $M_h = \{\texttt{p}\}$, $M_b = \{\texttt{sensor\_value(num\_var(sv))}\}$ and two examples:
$e_1 = \langle \langle \{\texttt{p}\}, \emptyset \rangle, \{\texttt{sensor\_value(0.5).}\} \rangle$ and $e_2 = \langle \langle \emptyset, \{\texttt{p}\} \rangle, \{\texttt{sensor\_value(0.4).}\} \rangle$.
One solution to this task is the rule `p:-sensor_value(V),V >= 0.5`. Note that there are many other values that could be used instead of `0.5`; however, it is sufficient to restrict the search space to only consider those constants that occur at least once in the examples.

3 Related works

Weather prediction has been faced in different ways. Traditional methods include numerical weather prediction techniques [11], which eventually can be followed by statistical and machine learning models [22,20] or directly using AI

methods [24,25,18]. These models can achieve high accuracy, but, as previously mentioned, interpreting complex weather models, such as neural networks, remains a significant challenge, and so, XAI techniques are being developed to explain how these models arrive at their predictions. A recent study by Labe et al. [12] employed explainable AI with Artificial Neural Networks (ANNs). This research successfully analyzed historical data to identify factors contributing to rising summer temperatures in the United States. There exist many others with similar purpose, such as [23], which explore various XAI approaches for weather prediction. However, none of those are using inductive logic programming, even though there are studies that exploit ILASP [15,7] and FastLAS [3,5], showing their capability in learning rule-based models. ILP, particularly through ILASP and FastLAS, offers significant potential in creating interpretable and accurate rule-based models, which can be highly beneficial for weather prediction tasks. Moreover, FastLAS has proved to perform great when specific rules are preferred over general once: Drozdov et al. [8] used it to build a system for security policy where more restrictive rules are preferable. It has been also exploited in combination with neural networks [6]. In that work, Cunnington et al. present a novel approach that integrates neural networks with symbolic reasoning, aiming to combine the learning capabilities of neural networks with the interpretability and logical structure of symbolic methods. This hybrid model enhances the explainability and robustness of machine learning systems while maintaining high performance. Numerous methods for enhancing the explainability of ASP have been proposed in the literature. Among them, we mention the proposal by Alviano et al. [1]. In their work, they demonstrate how it is possible to link the presence/absence of an atom in an answer set to the logic rules involved in its inference. Another example of tool for explainability in ASP is the one by Cabalar et al. [4], where *causal graphs* are exploited to represent rules, the nodes of the graph, while edges represent the order of rule applications.

4 Methods

Dataset. The dataset used to train the ILP system consists of data collected from multiple meteorological stations, even though the training is focused on a single station. This approach was chosen to ensure it works effectively on a small region before extending it to a larger area. These data are available from the ARPA FVG - OSMER web site (`https://www.osmer.fvg.it/archivio.php?ln=&p=dati`). The stations are located in the Friuli Venezia Giulia region (Italy). Specifically, we start focusing on the station situated in Udine. In our future work we will apply the method to multiple meteorological stations. The data collected span two sets of three months each: Set_{jfm} for January, February, March 2024, and Set_{amj} for April, May, June 2024, based on hourly data. Atmospheric variables are essential components of Earth's atmosphere that play fundamental roles in shaping weather patterns and influencing various natural processes. Understanding the interplay of the relationship between these atmospheric parameters is crucial for comprehending the dynamics of weather systems and climate patterns. With

Table 1. Dataset summary (first six months of 2024)

	Samples	Missing samples	Max value	Min value	Mean value	Standard deviation	Coefficient of Variation
Jan–Mar	S_{jfm}						
Rain mm	2185	1	17.1	0.0	0.23	0.96	4.23
Temperature °C	2174	12	20.6	-5.0	7.81	4.62	0.59
Humidity %	2185	1	97.0	23.0	79.49	15.03	0.19
Wind speed km/h	2185	1	26.0	1.0	6.03	3.73	0.62
Pressure hPa	2185	1	1026.0	978.0	1004.05	10.17	0.01
Apr–Jun	S_{amj}						
Rain mm	2184	1	45.7	0.0	0.23	1.61	6.89
Temperature °C	2161	24	32.1	1.0	17.55	5.67	0.32
Humidity %	2181	1	99.0	27.0	73.75	17.47	0.23
Wind speed km/h	2173	12	34.0	1.0	7.37	4.23	0.57
Pressure hPa	2182	3	1018.0	987.0	1002.59	0.57	0.01

our work, we aim to develop principally a diagnostic system capable of providing explanations for the levels of rain that have occurred. However, the system is able to predict what it will probably happen in the following hour, based on what is happening in the current (and previous) hours. The variables used in the model include temperature, wind speed, wind direction, percentage of humidity, and pressure. As shown in Table 1, the dataset comprehends 2186 samples for each period, with few missing values. Table 1 shows some statistical description for each variable, including extremes, mean, σ and a measure of dispersion (Coefficient of Variation, CV=σ/μ). From January to March, the data reflects colder and more stable weather, characterized by low mean rainfall (0.23mm) and high mean humidity (79.49%). In contrast, April to June saw a transition to warmer and more variable weather. While average rainfall remained low, there were greater extremes, with a maximum of 45.7mm, and also the dispersion is higher (6.9 vs. 4.2). Humidity decreased slightly, wind speeds increased, and pressure remained relatively stable, with a slight decrease in mean value.

The variables in question can exhibit a wide spectrum of values. However, managing a large number of possibilities is impractical with the current versions of the ILP tools. Therefore, we decided to reshape, such as normalizing between 0 and 100, the domain of some of these variables to ensure the grounding process remains memory-efficient. Our considerations are based on the report "The Climate of Friuli Venezia Giulia" [2]. This document offers essential insights into region's climate by analyzing key meteorological and climatic variables, including precipitation (rain, snow, and hail), temperature, solar radiation, sky conditions, and wind. It covers the period from 1991 to 2020. This timeframe served as the reference period for calculating domain ranges.

Background knowledge. As background knowledge for our learning task, we defined few *auxiliary* predicates with simple rules. We defined the predicate `previous(T1,T2)` to indicate that timestamp `T1` is one hour earlier than `T2`. Similarly, we introduced predicates for each parameter to describe its increase or change between two timestamps. For example: `increased_temperature(T1,T2)` means that the temperature at timestamp `T2` is higher than in `T1`. For defining these predicates the parameter's value change is associated with a one-hour difference. With this basic background knowledge, it was not possible to learn rules containing conditions (see Sect. 2) due to scalability issues. In order to achieve that, it was necessary to reduce the number of variable by removing

arguments timestamps to the predicates mentioned above. In fact, as we will see in the next subsection, examples' context contains just two consecutive timestamps so the predicates in learned rules have to refer to two consecutive timestamp too without exploiting any variable argument. When we have to use them across multiple timestamps, rather than just between two, we must remap the predicates to include the relevant timestamps again. The background rules are shown below (we report just `current_temperature`, `previous_temperature` and `increased_temperature` as for the other weather variables the definition of the rules are the same):

```
current_temperature(X) :- temperature(T1,X), temperature(T2,_), T1 > T2.
previous_temperature(X) :- temperature(T1,X), temperature(T2,_),T1 < T2.
increased_temperature :- current_temperature(X), previous_temperature(Y), X<Y.
```

Examples. As mentioned in Sect. 4, the dataset consists of 3 months data. Training was conducted using a cross-validation approach, comprising 10 folds, which are the subsets into which a dataset is divided for this approach. Each fold consisted of 4 days for training and the remaining days for validation. However, due to the low amount of data with high precipitation levels, it cannot be ensured that the training sets of different folds are entirely distinct. Hence, there may be instances where the training sets of different folds overlap by at most one day.

Each example represent an hour of a day. Since our goal is to learn rules capable of predicting rain levels, each inclusion contains solely the predicate `rain`, while the context contains all other variables (if present) at the same time and one hour prior. Additionally, we incorporate the `rain` value from the previous timestamp into the context to get higher levels of consistency during the learning process. Rainfall data, measured in mm,[5] has been categorized to simplify the analysis. The categories are as follows: *none* (0 mm), *level_1* ($0 <$ rain ≤ 2 mm), *level_2* ($2 <$ rain ≤ 6 mm) and *level_3* (rain > 6 mm). Similarly, wind speed values have been categorized in 16 clusters (N, NW, NNW,...). We give an encoding example of one datapoint of our dataset:

```
#pos(id169@2, { rain(9162000, "none") },
     { rain(9162000, "level_1"), rain(9162000, "level_2"),
       rain(9162000, "level_3") },
     {temperature(9162000, 61).     wind_speed(9162000, 9).
      wind_direction(9162000, "E"). humidity(9162000, 4).
      pressure(9162000, 57).
      temperature(9158400, 62).     wind_speed(9158400, 6).
      wind_direction(9158400, "E"). humidity(9158400, 4).
      pressure(9158400, 58).        rain(9158400, "none").  }).
```

FastLAS aims to minimize the uncovered examples, actually minimizing the weights assigned to each of them by the programmer. In particular, we have assigned weights to the examples in such a way that the *sum of the weights* for each level is equal. This ensures that even if there are fewer examples in a particular category, its overall weight remains the same. According to ILP terminology, every example used is a "positive" one. We are requiring all the examples are covered by at least one answer set (brave induction/reasoning).

[5] 1mm stands for 1mm of water –1 liter– in a basin of base $1m \times 1m$, in an hour.

Hypothesis Space and Bias. The hypothesis space we aim to generate must include rules that reference the present to explain the future. This implies that we prefer predicate like `increased_temperature` in the body of the rule, rather than simply `temperature` (which refers to a single timestamp). Additionally, to maintain low memory usage, we limit the rules to a maximum of two variables. In contrast, the head of the rule will exclusively contain the predicate `rain`, with a constant (`level_0,..., level_3`) indicating the rain level and a variable representing the timestamp. Since we are using non-recursive Observational Predicate Learning (OPL) tasks, which means that it can only formulate hypotheses for predicates that are directly observed in the examples, the predicate `rain` is not permitted in the body (while it is in rules of background knowledge). This might be a limitation as it is conceivable that the rain level of a timestamp depends on the level of rain at previous timestamp. As mentioned, we prefer results that include predicates referencing prior times relative to the prediction instead of current ones. Therefore, the scoring function is defined as follows: every predicate that does refer to just a single timestamp will have a penalty of 10. The only exceptions are wind direction and humidity predicates, which have a penalty of 5. This adjustment was made after numerous observations indicated that these factors are quite important, as they were frequently used in the learned rules. A higher penalty for these predicates resulted in fewer examples being covered. Lastly, a negative penalty was given to predicate `previous` to encourage its use.

```
#bias("penalty(5, in_head)  :- head(_).").
#bias("penalty(10, body(X)) :- in_body(X), X = wind_speed(_,_).").
#bias("penalty(10, body(X)) :- in_body(X), X = temperature(_,_).").
#bias("penalty(5, body(X))  :- in_body(X), X = wind_direction(_,_).").
#bias("penalty(10, body(X)) :- in_body(X), X = pressure(_,_).").
#bias("penalty(5, body(X))  :- in_body(X), X = humidity(_,_).").
#bias("penalty(-1, body(X)) :- in_body(X), X = previous(_,_).").
```

In summary, we penalize rules that contain predicates in the body referring to a specific timestamp, while reducing penality if the predicate `previous` is present. On the other hand, in the encoding with fewer variables, the scoring function was simpler due to the nature of the encoding: we applied a single penalty indiscriminately to every predicate.

```
#bias("penalty(1, head(X)) :- in_head(X).").
#bias("penalty(1, body(X)) :- in_body(X).").
```

5 Results

In this section, we present the results obtained from our experiments using the basic FastLAS system [14] and a recent extension that allows for numerical variables in the search space, referred in this section as FastLAS*. We compare our results against SVM, Random Forest and decision tree used as base lines. By using FastLAS on our weather prediction model, we found a set of helpful and precise rules that improve how we understand and predict the weather. These rules not only show that our approach is feasible but also give us valuable insights into how weather patterns work. We will highlight some key rules it generated.

These rules reveal important correlations and dependencies, showing some of the interactions between different weather factors. Here are some examples that show the importance of the rules FastLAS discovered:

```
rain(V0,"level_2") :- time(V0), time(V1), not increased_temperature(V1,V0),
                      not changed_wind_direction(V1,V0), previous(V1,V0),
                      increased_wind(V1,V0), wind_direction(V0,"N").
rain(V0,"level_2") :- time(V0), previous_pressure(V0, V_0_hPa),
                      previous_temperature(V0, V_0_deg_temperature),
                      V_0_hPa <= 58, V_0_deg_temperature >= 46.
```

The first rule states that if there is no increase in temperature, the wind direction remains "N", and the wind speed increases, then the expected rain level is 2. The second rule, which incorporates conditions, states that if the temperature an hour ago was greater than 46 and the pressure was below 58, then we again expect a rain level of 2. As shown, we were able to learn rules with negations but it is not possible with FastLAS to learn choice rules.

For each fold the system was able to learn, more or less, between 10 and 25 rules. Table 2 presents the means of performance metrics of our models across 10 folds of cross-validation. The metrics are divided into two main categories: macro-averaging and micro-averaging. Under both categories we report the precision and recall values:

$$Precision_{Macro} = \frac{1}{N}\sum_{i=1}^{N}\frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FP}_i} \qquad Precision_{Micro} = \frac{\sum_{i=1}^{N}\mathrm{TP}_i}{\sum_{i=1}^{N}(\mathrm{TP}_i + \mathrm{FP}_i)}$$

$$Recall_{Macro} = \frac{1}{N}\sum_{i=1}^{N}\frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FN}_i} \qquad Recall_{Micro} = \frac{\sum_{i=1}^{N}\mathrm{TP}_i}{\sum_{i=1}^{N}(\mathrm{TP}_i + \mathrm{FN}_i)}$$

where N is the number of classes, TP_i is the number of true positives for class i, FP_i is the number of false positives for class i, and FN_i is the number of false negatives for class i. We also evaluated accuracy and Peirce Skill Score (PSS) [19] as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \qquad PSS = \frac{TP}{TP + FN} - \frac{FP}{FP + TN}$$

The PSS ranges from -1 to 1. A PSS of 1 indicates a perfect forecast, while a PSS of 0 indicates no skill (the forecast is no better than random chance). Negative values indicate that the prediction we are getting is the opposite of what we are expecting. The PSS we got is rather low but still not 0. For the set S_{jfm} it is significantly lower probably due to the lack of rain's events or to the generality of the learned rules. Moreover, FastLAS model demonstrates a slightly better PSS compared to FastLAS*, which can be attributed to a more effectively designed scoring function. In the future, we plan to enhance the scoring function for FastLAS* as well.

Additionally, the table includes the mean accuracy and weighted F1. The results given by FastLAS, i.e., the length, the penalty associated with them, the sum of length and penalty (L+P), and the number of examples not covered

Table 2. Summary of means of performance metrics

	Macro-averaging		Micro-averaging				
	Precision	Recall	Precision	Recall	Accuracy	PSS	Weighted F1
Jan–Mar	Set_{jfm}						
FastLAS	0.25	0.45	0.54	0.91	0.91	0.14	0.80
FastLAS*	0.30	0.35	0.66	0.70	0.70	0.11	0.71
SVM	0.38	0.34	0.82	0.82	0.82	-0.16	0.79
RandomForest	0.55	0.44	0.81	0.81	0.80	0.44	0.79
Decision Tree	0.45	0.44	0.74	0.74	0.74	0.29	0.76
Apr–Jun	Set_{amj}						
FastLAS	0.27	0.36	0.67	0.76	0.75	0.76	0.10
FastLAS*	0.25	0.35	0.61	0.69	0.69	0.68	0.03
SVM	0.33	0.32	0.87	0.87	0.87	0.82	0.07
RandomForest	0.38	0.32	0.85	0.85	0.83	0.82	0.14
Decision Tree	0.35	0.34	0.77	0.77	0.77	0.78	0.15

during training are shown in the last columns. The results indicate consistent performance across folds, with accuracy and weighted F1-scores remaining high, although the precision and recall values exhibit some variability.

Obviously, since we do not have many data of high rain levels, FastLAS performs better when it has to predict no rain even if, as mentioned in previous section, we gave a weight that uniformed the total weight of each level. That said, the overall accuracy is, for S_{jfm}, always above 0.85.

On the other hand, the results of the FastLAS runs that were able to learn rules with conditions are less accurate. In particular, as you can see in Table 2, the accuracy is lower but the precision is quite often higher.

As previously mentioned, we compared our results with those obtained using other techniques, summarized in Table 2. Regarding macro-averaging metrics, we observe that they are generally low across all models. Notably, the Random Forest model shows a slightly higher performance, achieving a precision of 0.55. In contrast, the FastLAS model excels in micro-averaging recall, and also achieves the highest levels of accuracy and weighted F1 score. As PSS, in the first period, the Random Forest shows a much higher performance than FastLAS, while with the second this difference is reduced, but Random Forest still performs better.

The graphs provided in Fig.1 are line charts comparing the accuracy of the different models across the 10 folds of the cross-validation. As we can see, for Set_{jfm} FastLAS generally maintains high accuracy across most folds, with accuracy mostly above 80%. It shows some variability but is relatively stable compared to other models. The other variant of the model, FastLAS*, shows significant fluctuations in accuracy. It performs very poorly in folds 3 and 7, where accuracy drops to around 50%, while in other folds, it achieves accuracy comparable to other models. The SVM model, similar to Random Forest and Decision Tree models, has a relatively stable performance with accuracies mostly clustered around 70-80%, never really reaching the accuracy of FastLAS.

In the graph on the right in Fig.1, we can see that the accuracy of the two FastLAS models is worse compared to the other dataset (S_{jfm}). However, in some folds, they still manage to, if not surpass, at least maintain the same

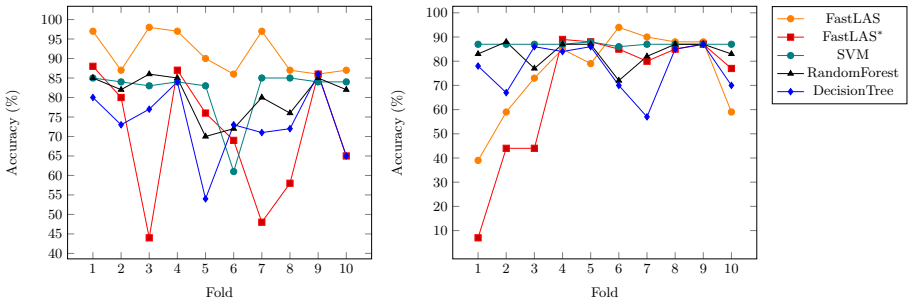


Fig. 1. Accuracy for each fold for each model. Set_{jfm} (left) and Set_{amj} (right)

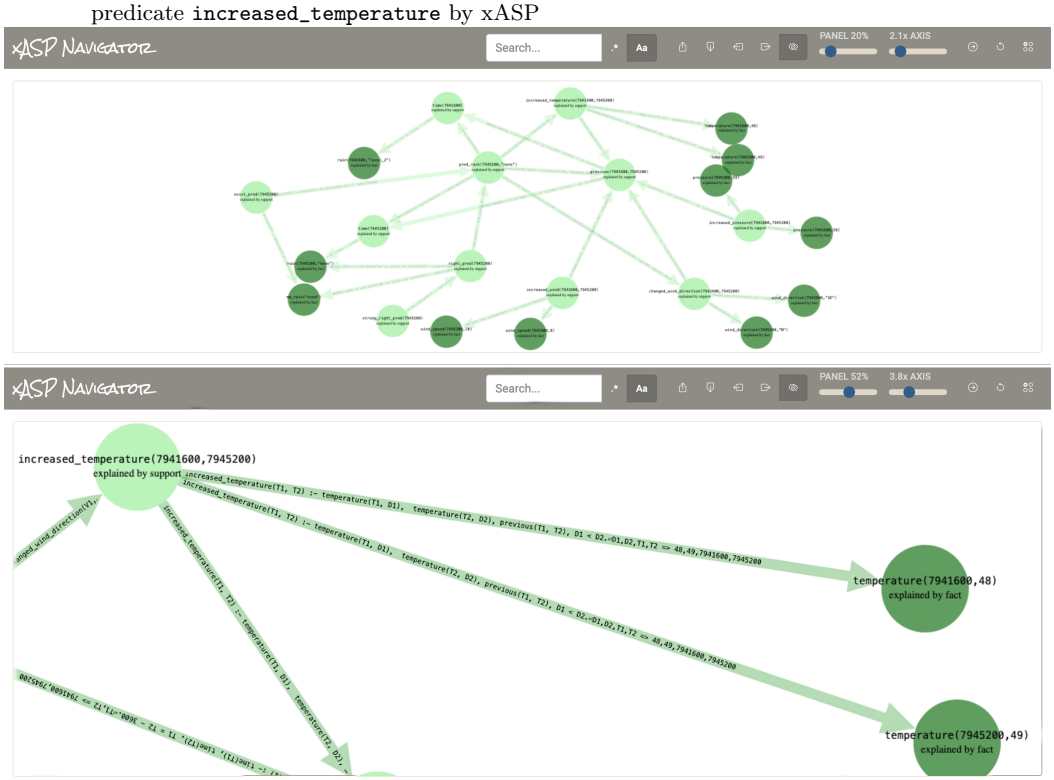
accuracy as other machine learning systems. In particular, FastLAS has a peak performance of 94% in the sixth fold. Concerning FastLAS*, its accuracy reaches up to 89% in the fourth fold. These results are not highly satisfactory, but they are promising. Therefore, with improvements to the model, we are likely to achieve better outcomes. As final step, we used xASP [1] to explain the results we obtained through a graph. In Figure 2 we show the rendering for the complete answer set and a zoom on a part of it.

6 Conclusions and Future work

We explored the application of FastLAS, a system for learning answer set programs, to predict rain levels. Our primary objective was to assess the viability of FastLAS as a predictive model in the domain of weather forecasting and to compare its performance with existing models. The results of our study indicate that FastLAS holds significant promise in this context. The key advantages of FastLAS lie in its ability to generate explainable and interpretable models, which is crucial for enhancing user trust and understanding in weather predictions. Our findings suggest that the performance of FastLAS can be further improved with additional testing and the integration of more diverse datasets, by expanding the geographic scope of our data collection and incorporating information from a wider range of weather stations and seasons, considering also other discretization levels for the rain, this will be one of the primary directions for future work. Since in absence of all data, many stable models could be plausible for our tool, we plan to use some of the reliable weather prediction systems to select which one is more feasible, and, thanks to the capability of explainability of ASP models, to prepare an automatic report on the lines of those usually explained by experts in communication media, by a post-processing of the results of xASP.

Acknowledgments. This research is partially supported by Interdepartment Project on AI (Strategic Plan UniUD–2022-25), by NextGenerationEU-PNRR project *MaPSART*- "Future Artificial Intelligence Research", and by INdAM-GNCS 2024 project LCXAI: Logica Computazionale per eXplainable Artificial Intelligence.

Fig. 2. Full explanation of the answer set and Zoomed-in view of the explanation of predicate `increased_temperature` by xASP



References

1. Alviano, M., Trieu, L.L.T., Son, T.C., Balduccini, M., et al.: Advancements in xASP, an XAI system for answer set programming. In: CILC (2023)
2. ARPA FVG: Riepilogo anno 2022, chap. 1. SOC OSMER e GRN - Osservatorio Meteorologico Regionale (2023), `https://www.meteo.fvg.it/pubblicazioni/meteo-fvg/2022/meteo.fvg_2022-riepilogo_it.pdf`
3. Baugh, K.G., Cingillioglu, N., Russo, A.: Neuro-symbolic rule learning in real-world classification tasks. In: Martin, A., Fill, H., Gerber, A., Hinkelmann, K., Lenat, D., Stolle, R., van Harmelen, F. (eds.) Proc. of AAAI-MAKE 2023. CEUR Workshop Proceedings, vol. 3433. CEUR-WS.org (2023)
4. Cabalar, P., Fandinno, J., Fink, M.: Causal graph justifications of logic programs. Theory and Practice of Logic Programming **14**(4-5), 603–618 (2014)
5. Cunnington, D., Cipcigan, F., Ferreira, R.N.B., Booth, J.: Symbolic learning for material discovery. arXiv preprint arXiv:2312.11487 (2023)
6. Cunnington, D., Law, M., Lobo, J., Russo, A.: Ffnsl: Feed-forward neural-symbolic learner. Machine Learning **112**(2), 515–569 (2023)

7. Dreossi, T.: Exploring ILASP through logic puzzles modelling. In: CEUR Workshop Proceedings. vol. 3428. CEUR-WS (2023)
8. Drozdov, A., Law, M., Lobo, J., Russo, A., Don, M.W.: Online symbolic learning of policies for explainable security. In: 2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA). pp. 269–278 (2021). https://doi.org/10.1109/TPSISA52974.2021.00030
9. Flora, M.L., Potvin, C.K., McGovern, A., Handler, S.: A machine learning explainability tutorial for atmospheric sciences. Artificial Intelligence for the Earth Systems **3**(1), e230018 (2024)
10. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: ICLP/SLP. vol. 88, pp. 1070–1080. Cambridge, MA (1988)
11. Kalnay, E.: Atmospheric Modeling, Data Assimilation and Predictability. Cambridge University Press (2002)
12. Labe, Z.M., Johnson, N., Delworth, T.L.: Changes in United States summer temperatures revealed by explainable neural networks. Authorea Preprints (2023)
13. Law, M.: Inductive learning of answer set programs. Ph.D. thesis, Imperial College London, UK (2018)
14. Law, M., Russo, A., Bertino, E., Broda, K., Lobo, J.: Fastlas: Scalable inductive logic programming incorporating domain-specific optimisation criteria. In: Proc. of the AAAI conference on artificial intelligence. vol. 34, pp. 2877–2885 (2020)
15. Law, M., Russo, A., Broda, K.: Inductive learning of answer set programs. In: Logics in Artificial Intelligence, JELIA 2014. pp. 311–325. Springer (2014)
16. Law, M., Russo, A., Broda, K.: Inductive learning of answer set programs from noisy examples. arXiv preprint arXiv:1808.08441 (2018)
17. Law, M., Russo, A., Broda, K.: The ilasp system for inductive learning of answer set programs. arXiv preprint arXiv:2005.00904 (2020)
18. Li, L., Carver, R., Lopez-Gomez, I., Sha, F., Anderson, J.: Generative emulation of weather forecast ensembles with diffusion models. Science Advances **10**(13) (2024)
19. Manzato, A.: A note on the maximum Peirce skill score. Weather and Forecasting **22**(5), 1148–1154 (2007)
20. McGovern, A., et al.: Using artificial intelligence to improve real-time decision-making for high-impact weather. Bulletin of the American Meteorological Society **98**(10), 2073–2090 (2017)
21. Muggleton, S.H.: Inductive logic programming. New Gener. Comput. **8**(4), 295–318 (1991). https://doi.org/10.1007/BF03037089
22. Rasp, S., Pritchard, M.S., Gentine, P.: Deep learning to represent subgrid processes in climate models. Proceedings of the national academy of sciences **115**(39), 9684–9689 (2018)
23. Toms, B.A., Barnes, E.A., Hurrell, J.W.: Assessing decadal predictability in an earth-system model using explainable neural networks. Geophysical Research Letters **48**(12), e2021GL093842 (2021)
24. Weyn, J.A., Durran, D.R., Caruana, R.: Can machines learn to predict weather? Using deep learning to predict gridded 500-hPa geopotential height from historical weather data. J. of Advances in Modeling Earth Systems **11**(8), 2680–2693 (2019)
25. Weyn, J.A., Durran, D.R., Caruana, R.: Improving data-driven global weather prediction using deep convolutional neural networks on a cubed sphere. J. of Advances in Modeling Earth Systems **12**(9), e2020MS002109 (2020)