Learning Data-Driven Uncertainty Set Partitions for Robust and Adaptive Energy Forecasting with Missing Data

Akylas Stratigakos, *Member, IEEE,* and Panagiotis Andrianesis, *Member, IEEE*

***Abstract*—Short-term forecasting models typically assume the availability of input data (features) when they are deployed and in use. However, equipment failures, disruptions, cyberattacks, may lead to missing features when such models are used operationally, which could negatively affect forecast accuracy, and result in suboptimal operational decisions. In this paper, we use adaptive robust optimization and adversarial machine learning to develop forecasting models that seamlessly handle missing data operationally. We propose linear- and neural network-based forecasting models with parameters that adapt to available features, combining linear adaptation with a novel algorithm for learning data-driven uncertainty set partitions. The proposed adaptive models do not rely on identifying historical missing data patterns and are suitable for real-time operations under stringent time constraints. Extensive numerical experiments on short-term wind power forecasting considering horizons from 15 minutes to 4 hours ahead illustrate that our proposed adaptive models are on par with imputation when data are missing for very short periods (e.g., when only the latest measurement is missing) whereas they significantly outperform imputation when data are missing for longer periods. We further provide insights by showcasing how linear adaptation and data-driven partitions (even with a few subsets) approach the performance of the optimal, yet impractical, method of retraining for every possible realization of missing data.**

***Index Terms*—Short-term forecasting, wind power forecasting, missing data, adaptive robust optimization, data-driven uncertainty set partitioning, adversarial learning.**

I. Introduction

Variable renewable energy sources, such as wind and solar, dominate low-carbon power systems. To deal with their inherent uncertainty and variability, system operators manage operational risk based on a forward-looking grid status estimation [1]. For instance, they run short-term scheduling applications to evaluate the reliability of market-based dispatch, which are based on short-term energy forecasts with a horizon ranging from a few minutes to several hours ahead [2].

A. Background and Motivation

A critical assumption underpinning the forecasting models is that input data, *a.k.a.* features, such as weather forecasts and production measurements, will always be available when needed, i.e., when the models are deployed to produce "operational" forecasts. However, equipment failures, disruptions, cyberattacks, may lead to missing data when the forecasting model is deployed [3], thus compromising forecast accuracy and leading to suboptimal operational decisions [4]. For example, weather data are sometimes published with a delay [5]; similarly, some production measurements may not be available in real-time but retrieved ex-post. As a result, forecasters are often provided with complete data sets for model training and offline testing, but in real time, when these models need the (same) features to produce forecasts operationally, some of them may be unavailable. Hence, robustness against missing data in real time is critical. Perhaps most importantly, because such models support near real-time processes, dealing with missing data should comply with stringent time constraints.

B. Related Work

So far, missing data are handled with imputation [6], i.e., a missing value is replaced with a single or multiple "plausible" values [7]. The imputation method can be simple (e.g., single mean imputation) or based on an iterative process (e.g., multiple imputation). When the training data set contains observations with missing data, imputation is applied both before model training and when the model is used operationally. In such a case, single imputation methods (e.g., mean imputation) may suffice for consistent forecasts [8]; however, they may rely on ad hoc, potentially hard-to-verify, assumptions about the missingness mechanism. Moreover, historical observations with missing data may not be available. In the latter, more challenging, case, which is our main interest, imputation methods can only rely on iterative approaches, as single imputation introduces forecast bias [9]. Alternatively, a new model can be retrained, either on the fly or offline, considering only the available features; retraining is shown to be asymptotically optimal [9]. However, both can soon become impractical. Iterative imputation requires additional training after the missing data are realized, which becomes computationally prohibitive for real-time processes, whereas retraining requires an additional model for each of the missing feature combinations, whose number grows exponentially with the number of features. To address these shortcomings, the pertinent idea is to avoid imputation altogether by embedding missing data within the forecasting model. An insightful recent work [10] develops a hierarchy of regression models, placed

This work was supported in part by the Leverhulme Trust International Professorship (LIP-2020-002) and in part by the European Research Council (ERC) Starting Grant CRISP under Agreement 101165616.

A. Stratigakos is with the Department of Electrical and Electronic Engineering, Imperial College London, London, SW7 2AZ, U.K.: a.stratigakos@imperial.ac.uk. P. Andrianesis is with the Center for Processes, Renewable Energy and Energy Systems (PERSEE), Mines Paris, PSL University, 06904 Sophia Antipolis, France: panagiotis.andrianesis@minesparis.psl.eu.

between mean imputation and retraining, with parameters that adapt to the available features.

The application domain for this paper pertains to energy forecasting. In particular, we focus on short-term wind power forecasting with a horizon of up to several hours ahead, which is a critical process for operational reliability. In this process, the industry standard is imputation with persistence forecasts, i.e., forward-fill the last measured value over the next periods. In [11], several imputation methods are compared with retraining without missing features; retraining performs best. Considering historical observations with missing data, [12] imputes missing features using spatial information across adjacent wind plants, whereas [13] develops neural network-based models with layer biases that adapt to available features. In [14], [15], which take a probabilistic approach that jointly considers imputation and forecasting, a sampling procedure is implemented to estimate the missing features and derive probabilistic forecasts when the models are used operationally.

To deal with missing data in energy forecasting applications with complete training data sets, [16] proposes a robust regression that minimizes the worst-case loss due to missing features, for linear models and piecewise linear loss functions. Adversarial learning [17] is employed in [18] for neural network-based load forecasting using a robust formulation; however, this requires expressing neural network training as a mixed-integer problem that is repeatedly solved, which may render it computationally prohibitive.

A critical design component of robust formulations [19] is the *uncertainty set* that captures possible uncertainty realizations — see, e.g., [20]–[22] for applications of constructing uncertainty sets for wind power forecasting. Large uncertainty sets may lead to overly conservative model parameters. Finite adaptability [23], introduced in the context of adaptive optimization problems with discrete decisions, reduces conservativeness by partitioning the uncertainty set into smaller, less conservative, subsets. Finding a good partition is hard; [24] proposes several partitioning methods for adaptive problems with discrete decisions, whereas [25] develops a tree-based learning approach for data-driven partitions.

C. Aim and Contribution

In this paper, we develop short-term wind power forecasting models that seamlessly adapt to available features when the models are deployed and used operationally. Inspired by [10], we consider linear- and neural network-based forecasting models with adaptive parameters. We implement adversarial learning by formulating model training as an adaptive robust optimization problem; namely, we combine linear adaptation with uncertainty set partitioning and minimize the worst-case loss due to missing features. We examine the impact of linear adaptation, compare data-driven to ad-hoc partitioning (implemented in [16], [18]), and assess our methods against standard industry imputation approaches. Notably, our methods do not rely on identifying historical missing data patterns, and are suitable for deployment in real-time operations under stringent time constraints.

Our main contribution is three-fold. Firstly, we propose a tailored gradient-based strategy to train robust and adaptive forecasting models by rapidly finding adversarial missing feature examples. Secondly, we develop a novel tree-based algorithm to learn data-driven uncertainty set partitions. We posit that a small number of missing feature combinations is critical to forecast accuracy when data are missing; our partitioning algorithm finds such combinations and, thus, approaches the optimal, yet impractical, retraining method. Thirdly, we provide insights through extensive numerical experimentation using real-world data on short-term wind power forecasting. The results indicate that when data are missing for very short periods (e.g., the latest wind power measurement is missing), our methods are on par with imputation; when data are missing for longer periods, our methods perform significantly better, hedging effectively against worst-case scenarios.

D. Paper Structure

The rest of the paper is organized as follows. First, we formulate adaptive forecasting models (Section II) and develop a partition learning methodology (Section III). Next, we detail our experimental setup (Section IV) and present the results of the numerical experiments (Section V). Finally, we conclude and provide directions for future work (Section VI).

II. Training with Missing Data

In this section, we present preliminaries on forecasting models with missing data using uncertainty sets (Subsection II-A), we formulate robust and adaptive robust forecasting models (Subsection II-B), and we describe a training process with adversarial examples (Subsection II-C).

A. Model Preliminaries

We use bold font lowercase (uppercase) for vectors (matrices), calligraphic for sets, and $|\cdot|$ to denote set cardinality. Consider a forecasting task with $y \in \mathcal{Y}$ being the target variable and $\mathbf{x} \in \mathcal{X}$ a p-size feature vector. For example, in short-term wind power forecasting, y is the production of the s-th plant at period $t+h$ (t is the time index and h the forecast horizon), whereas $\mathbf{x}$ typically concatenates the historical production across adjacent farms for periods $[t, t-\tau]$ (τ is the maximum historical lag) and other relevant information such as the weather. We assume access to an n-size training data set without any missing data, denoted by $\mathcal{D} = \{(y_i, \mathbf{x}_i)\}_{i\in[n]}$, where $[n] = \{1, \dots, n\}$.

In a standard regression setting, we train a forecasting model $f : \mathcal{X} \to \mathcal{Y}$, parameterized by $\boldsymbol{\theta}$, by minimizing the empirical loss over $\mathcal{D}$, given by

$$L_{\mathcal{D}}\big(y, f(\mathbf{x};\boldsymbol{\theta})\big) = \frac{1}{n}\sum_{i\in[n]} l\big(y_i, f(\mathbf{x}_i;\boldsymbol{\theta})\big), \tag{1}$$

where $l : \mathbb{R}\times\mathbb{R} \to \mathbb{R}_+$ is a convex loss function, e.g., the mean squared error or the quantile loss, whereas y and $\mathbf{x}$ refer (with some abuse of notation) to the items of data set $\mathcal{D}$.

1) Base Forecasting Models: We focus on gradient-based forecasting models, which comprise two widely used classes of models, linear regression and neural networks, hereinafter referred to as base forecasting models.

a) Linear Regression (LR): We consider an LR base forecasting model given by

$$f(\mathbf{x};\boldsymbol{\theta}) = \mathbf{w}^\top \mathbf{x} + b, \tag{2}$$

where $\mathbf{w}$ are linear weights, b is the bias, and $\boldsymbol{\theta} := (\mathbf{w}, b)$.

b) Neural Network (NN): We consider an M-layer feed-forward NN base forecasting model with a rectified linear unit activation function, given by

$$\mathbf{g}^1 = \mathbf{W}^0\mathbf{x} + \mathbf{b}^0, \tag{3a}$$

$$\mathbf{g}^{m+1} = \max(\mathbf{W}^m\mathbf{g}^m + \mathbf{b}^m, \mathbf{0}), \qquad 1 \le m < M, \tag{3b}$$

$$f(\mathbf{x};\boldsymbol{\theta}) = \mathbf{w}^\top \mathbf{g}^M + b, \tag{3c}$$

where $\mathbf{W}^m$ and $\mathbf{b}^m$ are the appropriately-sized weights and bias, respectively, of the m-th hidden layer, $\mathbf{w}$ and b are the weights and bias, respectively, of the output linear layer, and $\boldsymbol{\theta} := \{(\mathbf{W}^{0:M-1}, \mathbf{b}^{0:M-1}), (\mathbf{w}, b)\}$.

2) Modeling Missing Data: In our setting, missing data pertain to feature availability. Although we assume that all features are available when we train the forecasting model, at the time the model is deployed (and used to produce operational forecasts), some features may not be available. Thus, we model the availability of the j-th feature by introducing binary variable α_j, which equals 1 when the feature is missing, and 0 otherwise. Let $\mathcal{P}$ denote the set of features that may not be available (indeed, some features are always available, e.g., calendar variables). Feature availability is thus reflected in the feature vector as follows:

$$\mathbf{x}(\boldsymbol{\alpha}) = \begin{cases} x_j(1-\alpha_j), & \forall j \in \mathcal{P}, \\ x_j, & \text{otherwise.} \end{cases} \tag{4}$$

We note that (4) replaces a missing value of x_j with zero as a placeholder. For a given realization of missing features, say $\hat{\boldsymbol{\alpha}}$, we derive a respective set of parameters $\hat{\boldsymbol{\theta}}$, given by

$$\hat{\boldsymbol{\theta}} \in \arg\min_{\boldsymbol{\theta}} \; L_{\mathcal{D}}\big(y, f(\mathbf{x}(\hat{\boldsymbol{\alpha}});\boldsymbol{\theta})\big), \tag{5}$$

where $\mathbf{x}_i(\hat{\boldsymbol{\alpha}})$ is defined by (4) by applying (the same) $\boldsymbol{\alpha}$ to all observations.

Algorithm 1 presents a batch gradient descent approach to solve (5), which requires controlling for a set of hyperparameters (learning rate η, maximum number of iterations K, and patience threshold Φ). It starts by creating training/validation subsets, without shuffling, such that $\mathcal{D} = \mathcal{D}^{\text{train}} \cup \mathcal{D}^{\text{val}}$ and randomly initializing base forecasting model parameters $\hat{\boldsymbol{\theta}}$ (lines 1 and 2). At iteration k, a training-loss-based gradient update is performed (line 4), and the performance is evaluated by monitoring the validation loss (line 5). The algorithm terminates (line 3) when the maximum number of iterations, K, is reached or the validation loss does not improve for Φ consecutive iterations (monitored by counter ϕ in lines 6-10). It returns parameters $\hat{\boldsymbol{\theta}}$, and the resulting validation loss, $\hat{L}$. Implementing a mini-batch version of Algorithm 1 with an adaptive learning rate is left to the interested reader.

Effectively, solving (5) is equivalent to first observing the missing features and then retraining a model without them, which enables full recourse against the realization of $\boldsymbol{\alpha}$ and exhibits strong performance [11]. However, regardless if implemented on the fly or offline, retraining would soon become impractical, to say the least, as the number of possible combinations grows exponentially with the number of features.

Algorithm 1 `Training Process`

Input: Data set $\mathcal{D}$, missing feature realization $\hat{\boldsymbol{\alpha}}$, learning rate η, number of iterations K, patience threshold Φ.
Output: Parameters $\hat{\boldsymbol{\theta}}$, loss $\hat{L}$.

1: Set $k = 0$, $\phi = 0$. Split $\mathcal{D} = \mathcal{D}^{\text{train}} \cup \mathcal{D}^{\text{val}}$.
2: Initialize $\boldsymbol{\theta}^0$ randomly. Set $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}^0, \hat{L} = +\infty$.
3: **while** $k < K$ and $\phi < \Phi$ **do**
4: $\quad \boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \eta \nabla_{\boldsymbol{\theta}^k} L_{\mathcal{D}^{\text{train}}}\big(y; f(\mathbf{x}(\hat{\boldsymbol{\alpha}}); \boldsymbol{\theta}^k)\big)$.
5: $\quad$ Compute $L_{\text{val}} = L_{\mathcal{D}^{\text{val}}}\big(y; f(\mathbf{x}(\hat{\boldsymbol{\alpha}}); \boldsymbol{\theta}^{k+1})\big)$.
6: $\quad$ **if** $L_{\text{val}} < \hat{L}$ **then**
7: $\quad\quad \hat{\boldsymbol{\theta}} = \boldsymbol{\theta}^{k+1}$, $\hat{L} = L_{\text{val}}$, $\phi = 0$;
8: $\quad$ **else**
9: $\quad\quad \phi \leftarrow \phi + 1$.
10: $\quad$ **end if**
11: $\quad k \leftarrow k + 1$
12: **end while**

B. Robust and Adaptive Formulations of Base Forecasting Models with Missing Data

In this work, we model the uncertainty in feature availability using an uncertainty set defined as follows:

$$\mathcal{U} = \{\boldsymbol{\alpha} \in \{\mathbf{0}, \mathbf{1}\}^{|\mathcal{P}|} \mid \mathbf{1}^\top \boldsymbol{\alpha} \le \gamma\}, \tag{6}$$

where γ is the maximum number of potentially missing features, *a.k.a.* the uncertainty budget, with $\gamma \le |\mathcal{P}|$. Next, we present robust and adaptive robust formulations for base forecasting models that employ the uncertainty set (6).

1) Robust Forecasting (RF): A first approach to deal with feature availability uncertainty is to employ an RF formulation that minimizes the worst-case loss when up to γ features are missing, given by

$$\texttt{RF}: \quad \min_{\boldsymbol{\theta}} \max_{\boldsymbol{\alpha} \in \mathcal{U}} L_{\mathcal{D}}\big(y, f(\mathbf{x}(\boldsymbol{\alpha});\boldsymbol{\theta})\big), \tag{7}$$

where f is the base forecasting model, LR or NN, defined by (2) or (3), respectively. Not surprisingly, depending on γ and the respective uncertainty set $\mathcal{U}$, (7) may lead to an overly conservative model parametrization.

2) Adaptive Robust Forecasting (ARF): To reduce conservativeness, inspired by [10], we employ an ARF formulation given by

$$\texttt{ARF}: \quad \min_{\boldsymbol{\theta}(\boldsymbol{\alpha})} \max_{\boldsymbol{\alpha} \in \mathcal{U}} L_{\mathcal{D}}\Big(y, f\big(\mathbf{x}(\boldsymbol{\alpha}); \boldsymbol{\theta}(\boldsymbol{\alpha})\big)\Big), \tag{8}$$

where the base forecasting model parameters are a function of the uncertain parameters, i.e., $\boldsymbol{\theta}(\boldsymbol{\alpha})$, and hence adapt to the available features. Considering linear adaptation, i.e., parameters $\boldsymbol{\theta}(\boldsymbol{\alpha})$ that are linear in $\boldsymbol{\alpha}$, we obtain the following formulations for the base forecasting models.

a) LR: For the LR base model (2), assuming the bias is modeled with a constant feature that equals 1 and is always available, we set $\boldsymbol{\theta}(\boldsymbol{\alpha}) = \mathbf{w} + \mathbf{D}\boldsymbol{\alpha}$ to obtain

$$f(\mathbf{x}(\boldsymbol{\alpha}); \boldsymbol{\theta}(\boldsymbol{\alpha})) = (\mathbf{w} + \mathbf{D}\boldsymbol{\alpha})^\top \mathbf{x}(\boldsymbol{\alpha}).$$

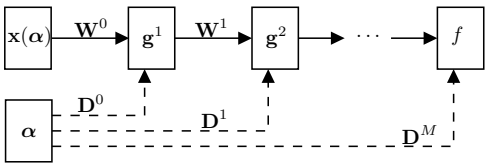


Fig. 1. A feed-forward neural network with linear correction.

Notably, $\mathbf{w}$ represents the model with all features available and $D_{z,j}$ is the linear correction applied to w_z when the j-th feature is missing. In the special case of an RF formulation with piecewise linear loss l, (7) can be solved using reformulation-based approaches [16]. However, in energy forecasting tasks, the number of observations n can be very large (e.g., in the order of 10^4 for hourly resolution series), which may challenge the scalability of such approaches.

b) NN: For the NN base model (3), we pass $\boldsymbol{\alpha}$ onto each layer and apply a linear correction to its weight matrix — see Fig. 1 for a visualization. Assuming the respective weight matrices absorb layer biases, a linearly adaptive M-layer feed-forward NN model is given by

$$\begin{aligned}
&\mathbf{g}^1 = (\mathbf{W}^0 + \mathbf{D}^0\boldsymbol{\alpha})\mathbf{x}(\boldsymbol{\alpha}),\\
&\mathbf{g}^{m+1} = \max\big((\mathbf{W}^m + \mathbf{D}^m\boldsymbol{\alpha})\mathbf{g}^m, \mathbf{0}\big),\ 1 \le m < M,\\
&f\big(\mathbf{x}(\boldsymbol{\alpha}); \boldsymbol{\theta}(\boldsymbol{\alpha})\big) = (\mathbf{w} + \mathbf{D}\boldsymbol{\alpha})^\top \mathbf{g}^M,
\end{aligned}$$

where $\mathbf{D}^m$ comprises the linear correction terms of the m-th hidden layer, and $\boldsymbol{\theta}(\boldsymbol{\alpha}) = \{(\mathbf{W}^{0:m-1} + \mathbf{D}^{0:m-1}\boldsymbol{\alpha}), (\mathbf{w} + \mathbf{D}\boldsymbol{\alpha})\}$ denotes the adaptive model parameters. The number of rows of $\mathbf{D}^m$ equals the number of columns of $\mathbf{W}^m$ and, with a slight abuse of notation, the matrix-vector addition $\mathbf{W}^m + \mathbf{D}^m\boldsymbol{\alpha}$ denotes a broadcasting operation, where vector $\mathbf{D}^m\boldsymbol{\alpha}$ is copied and added to each row of $\mathbf{W}^m$. Similarly to the LR base model, $\mathbf{W}^m$ denotes the layer weights when all features are available and $D^m_{z,j}$ is the linear correction applied to $W^m_{z,j}$ when the j-th feature is missing.

C. Adversarial Training with Missing Data

The RF (7) and ARF (8) formulations typically cannot be approximated by a tractable reformulation. In what follows, based on the idea of training with adversarial examples [17], we develop a tractable gradient-based algorithm to approximate the solution of (7) and (8). Since the same methodology applies to both RF and ARF formulations, we focus our exposition on the former. In brief, adversarial training involves: *(i)* initializing model parameters $\boldsymbol{\theta}$, *(ii)* finding an approximately optimal $\boldsymbol{\alpha}$ for the inner maximization problem (adversarial example), *(iii)* implementing a gradient update of $\boldsymbol{\theta}$ at the optimum of the inner problem using Danskin's theorem [26], and *(iv)* repeating steps *(ii)-(iii)* until convergence. We note that Danskin's theorem holds for the unique optimum of the inner maximization problem, which we may not be able to find. However, in practice, this method is used in adversarial training to compute useful gradients regardless [17].

Given a fixed set of parameters $\hat{\boldsymbol{\theta}}$, an adversarial example $\boldsymbol{\alpha}^{\text{adv}}$ is given by

$$\boldsymbol{\alpha}^{\text{adv}} \in \arg\max_{\boldsymbol{\alpha}\in\mathcal{U}} L_{\mathcal{D}}\big(y; f(\mathbf{x}(\boldsymbol{\alpha}); \hat{\boldsymbol{\theta}})\big), \tag{9}$$

Algorithm 2 `Adversarial Example`

Input: Data set $\mathcal{D}'$, feature set $\mathcal{P}'$, model parameters $\hat{\boldsymbol{\theta}}$, $\boldsymbol{\alpha}^{\text{opt}}$.
Output: Adversarial $\boldsymbol{\alpha}^{\text{adv}}$.

1: Set $\boldsymbol{\alpha}^{\text{adv}} = \boldsymbol{\alpha} = \boldsymbol{\alpha}^{\text{opt}}$, $L^{\text{adv}} = L_{\mathcal{D}'}\big(y; f(\mathbf{x}(\boldsymbol{\alpha}^{\text{adv}}); \hat{\boldsymbol{\theta}})\big)$.
2: Create $\mathcal{C} = \mathcal{P}'$.
3: **while** $\mathbf{1}^\top\boldsymbol{\alpha}^{\text{adv}} < \gamma$ **do**
4: **for** $j \in \mathcal{C}$ **do**
5: Set $\alpha_j = 1$ and $\alpha_{j'} = 0$, $\forall j' \in \mathcal{C}$, $j' \neq j$.
6: Compute $L(j) = L_{\mathcal{D}'}\big(y; f(\mathbf{x}(\boldsymbol{\alpha}); \hat{\boldsymbol{\theta}})\big)$.
7: **end for**
8: Select $j^* \in \arg\max_{j\in\mathcal{C}} L(j)$.
9: **if** $L(j^*) \ge L^{\text{adv}}$ **then**
10: set $\alpha^{\text{adv}}_{j^*} = 1$, $L^{\text{adv}} = L(j^*)$, remove j^* from $\mathcal{C}$
11: **else**
12: break.
13: **end if**
14: **end while**

which is typically some norm maximization problem, non-convex, and thus hard to solve. Previous energy forecasting applications [18], [27] addressed this problem, for $l = (\cdot)^2$ (mean squared error loss), by finding adversarial examples that strictly minimize or maximize the resultant forecast. Inspired by well-known feature importance metrics, we develop a greedy search algorithm to solve (9) and rapidly find adversarial examples, detailed in Algorithm 2.

Algorithm 2 uses as input a data set $\mathcal{D}'$, a feature set $\mathcal{P}'$, a specific set of parameters $\hat{\boldsymbol{\theta}}$, and an "optimistic" scenario, $\boldsymbol{\alpha}^{\text{opt}} = \mathbf{0}$, in which all features in $\mathcal{P}'$ are available. It returns an adversarial example $\boldsymbol{\alpha}^{\text{adv}}$ as follows. Initially, $\boldsymbol{\alpha}^{\text{adv}}$ is set equal to $\boldsymbol{\alpha}^{\text{opt}}$, and a copy of $\mathcal{P}'$, denoted by $\mathcal{C}$, is created (lines 1 and 2). The empirical loss (1) over $\mathcal{D}'$ is evaluated for each potentially missing feature, setting (temporarily, one by one) the respective entry of $\boldsymbol{\alpha}^{\text{adv}}$ equal to 1 (lines 4-7). Then, the feature with the largest loss, j^*, is selected, the respective entry of $\alpha^{\text{adv}}_{j^*}$ is fixed to 1, and j^* is removed from $\mathcal{C}$ (lines 8-10). The evaluation is repeated until reaching γ missing features (line 3) or the loss stops increasing (lines 9, 12).

Algorithm 3 details the full adversarial training process to solve (7) and learn parameters $\boldsymbol{\theta}^{\text{adv}}$. In a nutshell, Algorithm 3 modifies Algorithm 1 by introducing adversarial examples found with Algorithm 2. Specifically, it initially uses Algorithm 1 for $\hat{\boldsymbol{\alpha}} = \boldsymbol{\alpha}^{\text{opt}}$ to derive $\boldsymbol{\theta}^{\text{opt}}$, an "optimistic" set of parameters that optimizes performance for non-adversarial inputs, and sets $\boldsymbol{\theta}^{\text{adv}}$ equal to $\boldsymbol{\theta}^{\text{opt}}$ (line 2). At iteration k, it finds an adversarial example $\boldsymbol{\alpha}^{\text{adv}}$ using Algorithm 2 in the training data set (line 4) and performs a gradient update based on the training loss (line 5). It then finds a new adversarial example using Algorithm 2 in the validation data set (line 6), and computes the validation loss (line 7). Similarly to Algorithm 1, Algorithm 3 terminates when the maximum number of iterations, K, is reached or the validation loss does not improve for Φ consecutive iterations. It returns parameters $\boldsymbol{\theta}^{\text{adv}}$, and the resulting adversarial loss, L^{adv}.

Algorithm 3 `Adversarial Training Process`

Input: Data set $\mathcal{D}$, feature set $\mathcal{P}'$, learning rate η, number of iterations K, patience threshold Φ, $\boldsymbol{\alpha}^{\text{opt}}$.
Output: Parameters $\boldsymbol{\theta}^{\text{adv}}$, loss L^{adv}.

Set $k = 0$, $\phi = 0$. Split $\mathcal{D} = \mathcal{D}^{\text{train}} \cup \mathcal{D}^{\text{val}}$.
Run Algorithm 1 for $\hat{\boldsymbol{\alpha}} = \boldsymbol{\alpha}^{\text{opt}}$, and set $\boldsymbol{\theta}^{\text{adv}} = \boldsymbol{\theta}^0 = \boldsymbol{\theta}^{\text{opt}}$, $L^{\text{adv}} = +\infty$.
while $k < K$ and $\phi < \Phi$ **do**
  Run Algorithm 2 for $\mathcal{D}^{\text{train}}$, $\mathcal{P}'$, $\boldsymbol{\theta}^k$, $\boldsymbol{\alpha}^{\text{opt}}$, to get $\boldsymbol{\alpha}^{\text{adv}}$.
  $\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \eta \nabla_{\boldsymbol{\theta}^k} L_{\mathcal{D}^{\text{train}}}\big(y; f(\mathbf{x}(\boldsymbol{\alpha}^{\text{adv}}); \boldsymbol{\theta}^k)\big)$.
  Run Algorithm 2 for $\mathcal{D}^{\text{val}}$, $\mathcal{P}'$, $\boldsymbol{\theta}^{k+1}$, $\boldsymbol{\alpha}^{\text{opt}}$, to get $\boldsymbol{\alpha}^{\text{adv}}$.
  Compute $L_{\text{val}} = L_{\mathcal{D}^{\text{val}}}\big(y; f(\mathbf{x}(\boldsymbol{\alpha}^{\text{adv}}); \boldsymbol{\theta}^{k+1})\big)$.
  if $L_{\text{val}} < L^{\text{adv}}$ **then**
    $\boldsymbol{\theta}^{\text{adv}} = \boldsymbol{\theta}^{k+1}$, $L^{\text{adv}} = L_{\text{val}}$, $\phi = 0$;
  else
    $\phi \leftarrow \phi + 1$.
  end if
  $k \leftarrow k + 1$.
end while

III. Uncertainty Set Partitioning

In this section, we discuss the adaptation to missing data via uncertainty set partitioning (Subsection III-A) and propose an algorithm to learn partitions from data (Subsection III-B).

A. Missing Data Adaptation via Uncertainty Set Partitioning

Leveraging finite adaptability [23]–[25], we enhance the model capacity to adapt to missing data by partitioning the uncertainty set $\mathcal{U}$ into smaller, hence less conservative, sets. Let $\mathcal{T} = \{\mathcal{U}_q\}_{q\in[Q]}$ denote a partition of the uncertainty set $\mathcal{U}$ into Q disjoint subsets, such that $\mathcal{U} = \bigcup_{q\in[Q]} \mathcal{U}_q$. For each subset, $\mathcal{U}_q$, we solve (7) and (8), to learn a set of model parameters for the RF and ARF formulation, respectively. Indeed, any realization of $\boldsymbol{\alpha}$ would fall into exactly one subset. Hence, when the forecasting model is deployed, i.e., $\boldsymbol{\alpha}$ is realized, we can identify the partition for the specific realization and use the respective model parameters. In the extreme case of partitioning $\mathcal{U}$ at each possible missing feature combination, i.e., $Q = |\mathcal{U}|$, the outcome would be equivalent to retraining without the missing features [11].

Since the number of subsets, Q, controls the model complexity, the critical question is how to partition $\mathcal{U}$. Previous works [16], [18] split $\mathcal{U}$ using a collection of equality-constrained uncertainty sets with increasing budget. Namely, (6) is partitioned at $\gamma + 1$ subsets given by

$$\mathcal{U}_\ell^{\text{fixed}} = \{\boldsymbol{\alpha} \in \{\mathbf{0}, \mathbf{1}\}^{|\mathcal{P}|} \mid \mathbf{1}^\top \boldsymbol{\alpha} = \ell\},\ \ell \in \{0, 1, \ldots, \gamma\}, \quad (10)$$

i.e., considering each integer value in the range $[0, \gamma]$. We refer to (10) as *fixed* partitioning, since it depends solely on the uncertainty budget and does not consider the available data or the forecasting problem. However, fixed partitioning is ad-hoc and may be overly conservative. We posit that data-driven partitioning can jointly improve the loss for adversarial and non-adversarial examples while requiring fewer subsets.

B. Data-driven Uncertainty Set Partitions

In this subsection, we propose a novel tree-based algorithm to learn a data-driven partition $\mathcal{T}$, which, in contrast to fixed partitioning, leverages available data and tailors the learned partition to the forecasting model. For ease of exposition, we present the approach for the RF formulation; the application to the ARF formulation is straightforward.

Broadly, given an uncertainty set, say $\mathcal{U}_\ell$, we estimate a lower bound on the empirical loss, $\texttt{LB}_\ell$, running Algorithm 1 for the "optimistic" scenario $\boldsymbol{\alpha}_\ell^{\text{opt}}$, and an upper bound, $\texttt{UB}_\ell$, running Algorithm 3 to derive the adversarial loss. We then search over all the potentially missing features in $\mathcal{P}_\ell$, select the one that minimizes the relative gap

$$\texttt{RelGap}(\mathcal{U}_\ell) = \frac{\texttt{UB}_\ell - \texttt{LB}_\ell}{\texttt{LB}_\ell},$$

and split the uncertainty set into two subsets based on the availability of the selected feature.

Algorithm 4 details the partition process, which requires controlling for the number of subsets, Q, and an upper bound on the maximum relative gap, ε, across all subsets. The partition is initialized at the original uncertainty set (6), denoted by $\mathcal{U}_0$ (line 1), for which $\texttt{RelGap}(\mathcal{U}_0)$, $\boldsymbol{\theta}_0^{\text{opt}}$, and $\boldsymbol{\theta}_0^{\text{adv}}$ are computed (line 2). For the current partition $\mathcal{T}$, the subset with the largest relative gap, $\mathcal{U}_\ell$, is first selected (line 4), followed by the feature that leads to the largest loss, j^*, evaluated using Algorithm 2, iterating over the potentially missing features in the set $\mathcal{P}_\ell$, with $\boldsymbol{\theta} = \boldsymbol{\theta}_\ell^{\text{opt}}$ (line 5). Set $\mathcal{U}_\ell$ is then split into $\mathcal{U}_{|\mathcal{T}|+1}$ and $\mathcal{U}_{|\mathcal{T}|+2}$, by appending equality constraints $a_{j^*} = 0$ and $a_{j^*} = 1$, respectively, and the respective optimistic scenarios $\boldsymbol{\alpha}_{|\mathcal{T}|+1}^{\text{opt}}, \boldsymbol{\alpha}_{|\mathcal{T}|+2}^{\text{opt}}$ are updated (line 6). For the new subsets, the relative gaps and optimistic/adversarial model parameters are obtained using Algorithms 1 and 3 (lines 7 and 8). Since $\mathcal{U}_{|\mathcal{T}|+1}$ fixes a feature as always available, $\texttt{LB}_{|\mathcal{T}|+1} = \texttt{LB}_\ell$, whereas $\texttt{UB}_{|\mathcal{T}|+1} < \texttt{UB}_\ell$. Similarly, since $\mathcal{U}_{|\mathcal{T}|+2}$ fixes the same feature as always missing, $\texttt{LB}_{|\mathcal{T}|+2} > \texttt{LB}_\ell$, whereas $\texttt{UB}_{|\mathcal{T}|+2} = \texttt{UB}_\ell$. Once the partition is updated (line 9), the process (lines 4-9) is repeated until Q subsets are created or until the maximum relative gap, $\max_{\mathcal{U}_q \in \mathcal{T}} \texttt{RelGap}(\mathcal{U}_q)$, becomes smaller than ε (line 3).

Algorithm 4 learns the data-driven partition, for which the forecasting model is trained. At the time the forecasting model is deployed, we observe the realization of the missing features, say $\hat{\boldsymbol{\alpha}}$, identify the (unique) subset, say $\mathcal{U}_q$, which contains this realization, and derive forecasts as follows: *(i)* in case $\hat{\boldsymbol{\alpha}} = \boldsymbol{\alpha}_q^{\text{opt}}$, then we use model parameters $\boldsymbol{\theta}_q^{\text{opt}}$; *(ii)* otherwise, we use model parameters $\boldsymbol{\theta}_q^{\text{adv}}$.

Figure 2 shows an illustrative example, considering a model with up to 3 missing features, i.e., 8 potential combinations of missing features. The initial uncertainty set $\mathcal{U}_0$ is split on feature α_1 to derive subsets $\mathcal{U}_1 = \mathcal{U}_0 \cap \{\alpha_1 = 0\}$ and $\mathcal{U}_2 = \mathcal{U}_0 \cap \{\alpha_1 = 1\}$, with the respective index sets of potentially missing features being $\mathcal{P}_1, \mathcal{P}_2 = \{2, 3\}$ and the "optimistic" scenarios being $\boldsymbol{\alpha}_1^{\text{opt}} = (0, 0, 0)$ and $\boldsymbol{\alpha}_2^{\text{opt}} = (1, 0, 0)$. For $\mathcal{U}_1$, the lower bound remains the same ($\texttt{LB}_1 = \texttt{LB}_0$), whereas the upper bound improves ($\texttt{UB}_1 < \texttt{UB}_0$). For $\mathcal{U}_2$, the upper bound remains the same ($\texttt{UB}_2 = \texttt{UB}_0$), whereas the lower bound improves ($\texttt{LB}_2 > \texttt{LB}_0$). For this example, we assume

Algorithm 4 `Learn Partition`

Input: Data set $\mathcal{D}$, uncertainty set $\mathcal{U}$, number of subsets Q, maximum relative gap threshold ε.
Output: Partition $\mathcal{T}$.

Set $\mathcal{U}_0 = \mathcal{U}$, $\mathcal{P}_0 = \mathcal{P}$, $\mathcal{T} = \{\mathcal{U}_0\}$.
Compute $\boldsymbol{\theta}_0^{\text{opt}}$, $\boldsymbol{\theta}_0^{\text{adv}}$, $\texttt{RelGap}(\mathcal{U}_0)$:
(a) Run Algorithm 1 for $\hat{\boldsymbol{\alpha}} = \boldsymbol{\alpha}_0^{\text{opt}} = \mathbf{0}$, to get $\boldsymbol{\theta}_0^{\text{opt}} = \hat{\boldsymbol{\theta}}$, and set $\texttt{LB}_0 = \hat{L}$.
(b) Run Algorithm 3 for $\mathcal{P}_0$, $\boldsymbol{\alpha}_0^{\text{opt}}$, to get $\boldsymbol{\theta}_0^{\text{adv}} = \boldsymbol{\theta}^{\text{adv}}$, and set $\texttt{UB}_0 = L^{\text{adv}}$.
while $|\mathcal{T}| < Q$ and $\max_{\mathcal{U}_q \in \mathcal{T}} \texttt{RelGap}(\mathcal{U}_q) > \varepsilon$ **do**
  Select $\mathcal{U}_\ell \in \arg\max_{\mathcal{U}_q \in \mathcal{T}} \texttt{RelGap}(\mathcal{U}_q)$.
  Run lines 1-8 of Algorithm 2 for $\mathcal{P}_\ell$, $\boldsymbol{\theta}_\ell^{\text{opt}}$, $\boldsymbol{\alpha}_\ell^{\text{opt}}$, to get feature j^* (to split).
  Split $\mathcal{U}_\ell$ into $\mathcal{U}_{|\mathcal{T}|+1}$ and $\mathcal{U}_{|\mathcal{T}|+2}$, as follows:
  (a) $\mathcal{U}_{|\mathcal{T}|+1} = \mathcal{U}_\ell \cap \{\alpha_{j^*} = 0\}$;
  $\mathcal{U}_{|\mathcal{T}|+2} = \mathcal{U}_\ell \cap \{\alpha_{j^*} = 1\}$.
  (b) Set $\boldsymbol{\alpha}_{|\mathcal{T}|+1}^{\text{opt}} = \boldsymbol{\alpha}_{|\mathcal{T}|+2}^{\text{opt}} = \boldsymbol{\alpha}_\ell^{\text{opt}}$.
  Update $\alpha_{|\mathcal{T}|+1,j^*}^{\text{opt}} = 0$, and $\alpha_{|\mathcal{T}|+2,j^*}^{\text{opt}} = 1$.
  (c) Set $\mathcal{P}_{|\mathcal{T}|+1} = \mathcal{P}_{|\mathcal{T}|+2} = \mathcal{P}_\ell \setminus \{j^*\}$.
  Compute $\boldsymbol{\theta}_{|\mathcal{T}|+1}^{\text{opt}}, \boldsymbol{\theta}_{|\mathcal{T}|+1}^{\text{adv}}, \texttt{RelGap}(\mathcal{U}_{|\mathcal{T}|+1})$:
  (a) Set $\boldsymbol{\theta}_{|\mathcal{T}|+1}^{\text{opt}} = \boldsymbol{\theta}_\ell^{\text{opt}}$ and $\texttt{LB}_{|\mathcal{T}|+1} = \texttt{LB}_\ell$.
  (b) Run Algorithm 3 for $\mathcal{P}_{|\mathcal{T}|+1}$, $\boldsymbol{\alpha}_{|\mathcal{T}|+1}^{\text{opt}}$, to get $\boldsymbol{\theta}_{|\mathcal{T}|+1}^{\text{adv}} = \boldsymbol{\theta}^{\text{adv}}$, and set $\texttt{UB}_{|\mathcal{T}|+1} = L^{\text{adv}}$.
  Compute $\boldsymbol{\theta}_{|\mathcal{T}|+2}^{\text{opt}}, \boldsymbol{\theta}_{|\mathcal{T}|+2}^{\text{adv}}, \texttt{RelGap}(\mathcal{U}_{|\mathcal{T}|+2})$:
  (a) Run Algorithm 1 for $\hat{\boldsymbol{\alpha}} = \boldsymbol{\alpha}_{|\mathcal{T}|+2}^{\text{opt}}$, to get $\boldsymbol{\theta}_{|\mathcal{T}|+2}^{\text{opt}} = \hat{\boldsymbol{\theta}}$ and set $\texttt{LB}_{|\mathcal{T}|+2} = \hat{L}$.
  (b) Set $\boldsymbol{\theta}_{|\mathcal{T}|+2}^{\text{adv}} = \boldsymbol{\theta}_\ell^{\text{adv}}$ and $\texttt{UB}_{|\mathcal{T}|+2} = \texttt{UB}_\ell$.
  Update partition $\mathcal{T} \leftarrow \{\mathcal{T} \setminus \mathcal{U}_\ell\} \cup \{\mathcal{U}_{|\mathcal{T}|+1}, \mathcal{U}_{|\mathcal{T}|+2}\}$.
end while

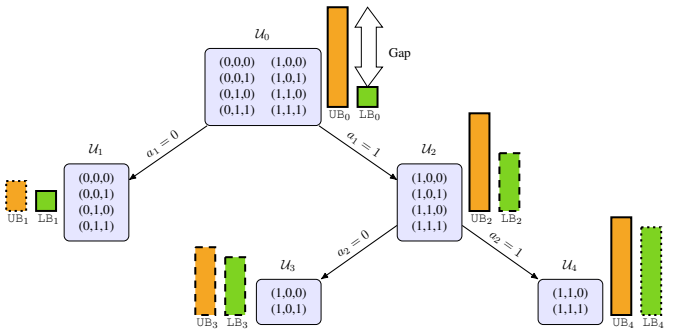


Fig. 2. Partitioning of set $\mathcal{U}_0$. Orange and green bars indicate the subset's lower (LB) and upper (UB) bound. The bar outline indicates the same height.

$\texttt{RelGap}(\mathcal{U}_1) < \varepsilon$ and $\texttt{RelGap}(\mathcal{U}_2) > \varepsilon$, hence $\mathcal{U}_2$ is split further. The final partition is $\mathcal{T} = \{\mathcal{U}_1, \mathcal{U}_3, \mathcal{U}_4\}$.

IV. Experimental Setup

In this section, we detail the data and the forecast setting of our numerical experiments on short-term wind power forecasting (Subsection IV-A), benchmark performance without missing data (Subsection IV-B), and present the forecasting models with missing data (Subsection IV-C).

A. Data and Forecast Setting

We use wind power measurements with a 15-minute granularity from 8 adjacent wind power plants with individual nominal capacities ranging from 80 to 300 MW and a total nominal capacity of 1127.45 MW [28]. The wind power plants are located in the North load zone of the NYISO balancing area, spanning approximately a 55×80 km area of semi-complex terrain. The data set spans 2018 and we apply a 50/50 training/test split, with the last 15% of the training data set reserved for validation and hyperparameter tuning.

We examine forecast horizons for $h = \{1, 4, 8, 16\}$ steps ahead, i.e., from 15 minutes to 4 hours ahead, which are used in short-term scheduling applications of system operators for reliability purposes. A separate forecasting model is trained for each forecast horizon. We considered measurements for periods t, $t-1$, and $t-2$, for the 8 plants as features for $8 \times 3 = 24$ features. To account for the weather impact, which becomes particularly relevant as the horizon increases [29], we considered an additional feature of an hourly production forecast, issued from [30] 6 hours earlier (at period $t-24$) using the latest available weather forecasts. We considered that the feature is always available operationally, as these forecasts are issued in batches. Henceforth, missing data refer to the wind plant production measurements, i.e., $|\mathcal{P}| = 24$.

B. Benchmarks without Missing Data

We consider the following base forecasting models:

- `Pers`, a persistence forecasting model, industry's standard for very short-term wind power forecasting, where the last measured value is used as forecast;
- `LR` model (2) with loss function $l = (\cdot)^2$ (mean squared error), and
- `NN` model (3), with the same loss function and 4 hidden layers of 50 nodes each.

We implemented a mini-batch version of Algorithm 1 with batch size 512, learning rate $\eta = 10^{-3}$ adapted dynamically with the Adam optimizer, maximum number of iterations $K = 1000$, and patience threshold $\Phi = 20$. For `NN`, we additionally included weight decay parameter 10^{-5}, which induces a small regularization effect.

Table I presents the root mean squared error (RMSE) (without missing data), normalized by actual observations, for a medium-sized wind plant of 100 MW nominal capacity. The results indicate that `Pers` performs the worst with the highest RMSE across all horizons. For very short-term forecast horizons up to one hour ahead (see $h = 1, 4$), `LR` performs best. For horizons longer than 2 hours ahead (see $h = 8, 16$), `NN` is the best-performing model and its relative improvement over `LR` increases with the horizon. Overall, the performance is consistent with state-of-the-art short-term wind power forecasting in semi-complex terrain [29].

C. Forecasting Models with Missing Data

Given a base forecasting model (LR or NN), we consider the following methods for handling missing data:

TABLE I
RMSE (%) WITHOUT MISSING DATA FOR DIFFERENT HORIZONS h (15 MINUTES). BOLD FONT INDICATES THE BEST MODEL PER h.

Horizon	$h=1$	$h=4$	$h=8$	$h=16$
Pers	3.51	9.91	15.51	23.83
LR	**2.58**	**7.97**	12.12	16.73
NN	2.76	8.29	**11.69**	**14.74**

- Imputation (Imp) where missing data are imputed using persistence forecasts, industry's standard practice;
- RF (7) where model parameters are optimized for the worst-case realization of missing data, and
- ARF (8) with linear adaptation to the worst-case realization of missing data.

We further consider two partitioning approaches (applicable to RF and ARF):

- fixed — see (10), where we split $\mathcal{U}$ at each integer value in the range $[0, \gamma]$, with $\gamma = 24$, and
- learn^Q, where we learn Q partitions using Algorithm 4, setting $Q = 10$ and $\varepsilon = 0.1\%$.

To ease the exposition, we refer to an forecasting model with missing data by first using the base forecasting model, followed by the method to handle missing data, and the partitioning approach (if applicable) in parenthesis. For example:

- LR−Imp denotes an LR base forecasting model with imputation;
- NN−RF(fixed) denotes an NN base forecasting model, with an RF formulation (7) and fixed partitioning;
- NN−ARF(learn^{10}) denotes an NN base forecasting model, with an ARF formulation (8) and $Q = 10$ learned partitions, and so forth.

We implemented a mini-batch version of Algorithm 3 with the same hyperparameters as Algorithm 1. For fixed partitions, we adapted Algorithm 2 and constructed adversarial examples for equality-constrained uncertainty sets of the form (10) by sampling missing features with a uniform probability distribution.

V. RESULTS

In this section, we present forecasting results with missing data (Subsection V-A), showcase and interpret the proposed methods (Subsection V-B), and explore the sensitivity to the number of subsets learned (Subsection V-C). All results pertain to the same 100 MW wind plant of Table I.[1]

A. Performance with Randomly Missing Data

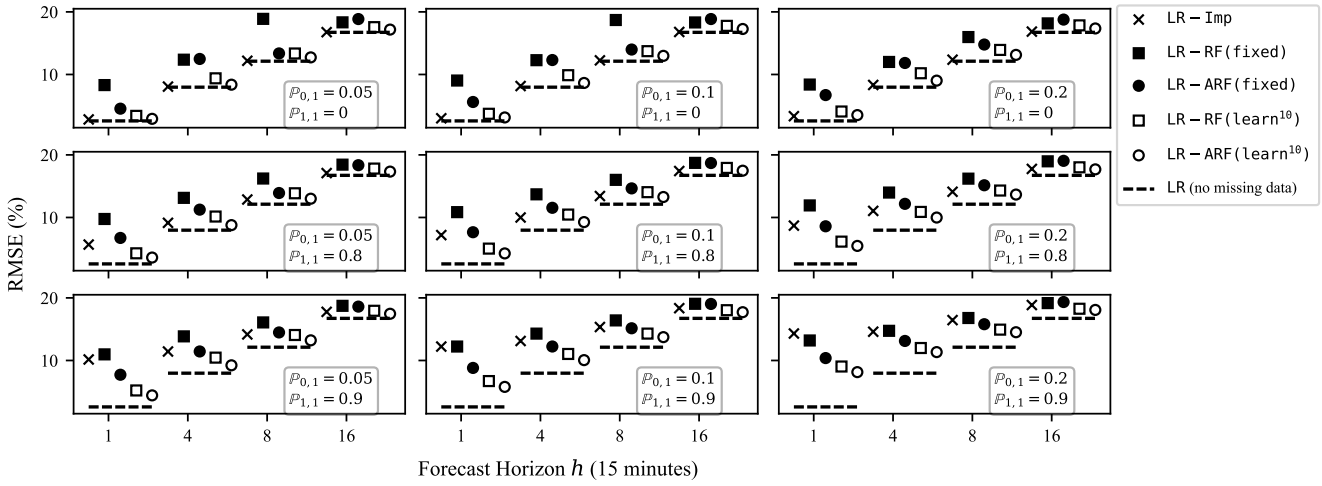
We consider randomly missing data, i.e., the probability of missing a measurement is independent of its value, described by a transition matrix. Let $\mathbb{P}_{0\to1}$ be the probability that a measurement is missing in a certain period, whereas it was available in the previous period. Further, let $\mathbb{P}_{1\to1}$ be the probability that a measurement is missing in a certain period, whereas it was missing in the previous period.

[1]The code for recreating the experiments is available on GitHub: https://github.com/akylasstrat/wind-forecast-missing-data.

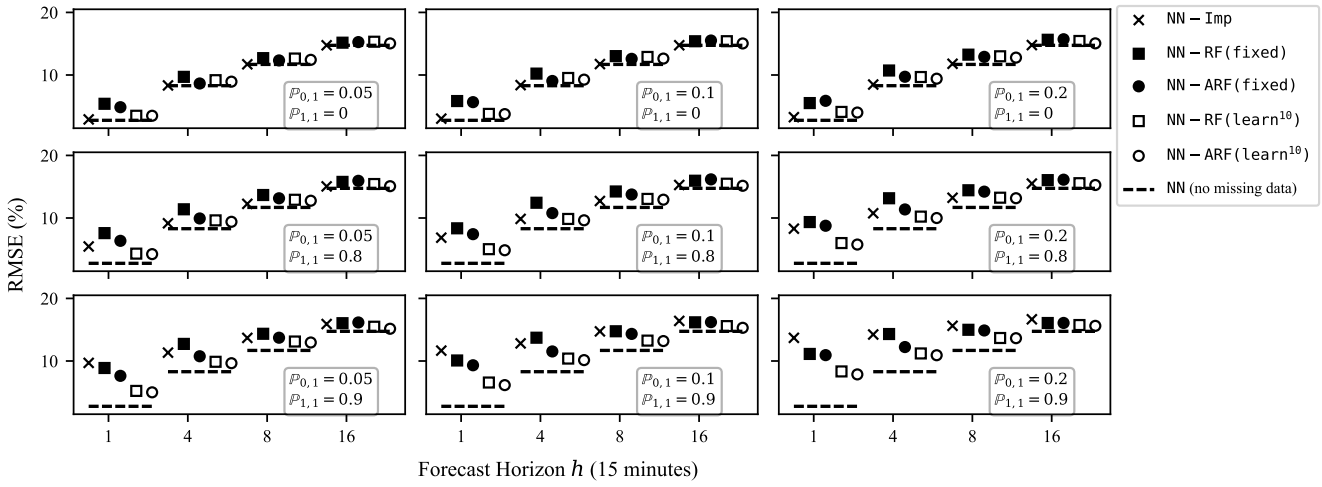
Figure 3 shows the average RMSE over 10 iterations for LR and NN base forecasting models (top and bottom, respectively), and for different probabilities $\mathbb{P}_{0\to1}$ and $\mathbb{P}_{1\to1}$, where the probabilities are applied to the 8 available wind plants. For each base forecasting model, the 9 subplots are arranged so that each row represents $\mathbb{P}_{1\to1}$ equal to $0.0, 0.8, 0.9$, and each column represents $\mathbb{P}_{0\to1}$ equal to $0.05, 0.1, 0.2$. As we move to the right and bottom, data are missing more frequently ($\mathbb{P}_{0\to1}$ increases) and remain unavailable for longer periods ($\mathbb{P}_{1\to1}$ increases), respectively. Indeed, the RMSE for each forecasting model increases as we move to the right and bottom as more data are missing.

1) LR base forecasting models: Figure 3a illustrates the average RMSE for the LR base forecasting models. Consider first the imputation-based model LR−Imp ($\times$ marker). In the top row, where $\mathbb{P}_{1\to1} = 0$, which implies that the data are missing for a single period, LR−Imp performs best, with an RMSE close to the benchmark value without missing data (dashed line). As we move to the middle and bottom rows and the probability $\mathbb{P}_{1\to1}$ increases, i.e., data are missing for longer periods, LR−Imp degrades considerably. The LR−ARF(learn^{10}) model (empty circle marker) performs better in the middle and the bottom row followed closely by LR−RF(learn^{10}) (empty square marker); both stay very close to LR−Imp in the top row. The LR−ARF(fixed) and LR−RF(fixed) models perform worse in the top row, whereas they become comparable with LR−Imp in the middle row, and occasionally better in the bottom row, where LR−ARF(fixed) generally performs better than LR−RF(fixed). Considering the performance for different forecast horizons, we observe that the RMSE difference compared to the benchmark (without missing data) decreases for longer horizons, as evident by the span of the RMSE between the different methods and the benchmark. For instance, consider the bottom right subplot which is the case with the most missing data; the RMSE difference of LR−ARF(learn^{10}) with the benchmark (without missing data) is about 5.56% for $h = 1$, 3.37% for $h = 4$, 2.39% for $h = 8$, and 1.71% for $h = 16$. In general, learned partitions outperform fixed partitions, but the improvement decreases with the horizon. For instance, the best-performing LR−ARF(learn^{10}) outperforms LR−ARF(fixed) by $39\%, 22\%, 8\%$ and 7%, for horizons h equal to $1, 4, 8$ and 16, respectively, averaged across all subplots. Comparing the ARF with the RF method, a general observation is that the difference is larger for fixed partitions and shorter horizons, whereas it decreases for learned partitions and longer horizons.

2) NN base forecasting models: Figure 3b illustrates the average RMSE for the NN base forecasting models. Overall, the forecasting models perform similarly to the LR base forecasting models. The imputation-based model NN−Imp performs the best in the top row, closely followed by NN−RF(learn^{10}) and NN−ARF(learn^{10}). The latter perform best (and remain close to each other) in the middle and bottom rows. In general, it is evident that the span between the different models in Fig. 3b is quite smaller compared to Fig. 3a, and still decreases for longer horizons. However, the RMSE of the best-performing NN−ARF(learn^{10}) model



(a) LR base forecasting models.



(b) NN base forecasting models.

Fig. 3. RMSE (%) for different forecast horizons with randomly missing data for $\mathbb{P}_{0\to1} = \{0.05, 0.1, 0.2\}$ and $\mathbb{P}_{1\to1} = \{0.0, 0.8, 0.9\}$, averaged over 10 runs. The dashed line shows performance without missing data.

is, in most cases, higher compared to the respective RMSE of LR−ARF(learn^{10}) for $h = 1, 4$ (where LR prevails in Table I) whereas it is smaller for $h = 8, 16$ (where NN prevails in Table I). Similarly to the LR base forecasting models, learned partitions outperform fixed partitions and the improvement decreases with the horizon. For instance, the best-performing NN−ARF(learn^{10}) outperforms NN−ARF(fixed) by $32\%, 6\%, 4\%$,and 4%, for horizons h equal to $1, 4, 8,$ and 16, respectively, averaged across all subplots. However, the difference between the ARF and RF methods is smaller compared to the LR base models, indicating that linear adaptation benefits the latter more.

B. Illustration of the Partitioning and Adaptation Methods

We illustrate how the proposed methods deal with missing data by examining the learned data-driven partitions and respective model parameterization.

1) Splits: Consider the LR−ARF(learn^{Q}) model for $h = 1$. Table II shows the first 2 splits learned by Algorithm 4. At $\mathcal{U}_0$, the measurement of plant 2 (the target plant) at period t is identified as the most important feature ($j^* = 3$) and $\mathcal{U}_0$ is split into $\mathcal{U}_1$ ($\alpha_3 = 0$) and $\mathcal{U}_2$ ($\alpha_3 = 1$). RelGap($\mathcal{U}_1$) is small (0.28%), which is expected as $\mathcal{U}_1$ is conditioned on the most important feature being available; $\mathcal{U}_1$ is not split further. Conversely, RelGap($\mathcal{U}_2$) remains large (32.42%). Algorithm 4 next identifies the measurement of plant 5 at period t (feature 12) as the most important feature and $\mathcal{U}_2$ is split into $\mathcal{U}_3$ ($\alpha_{12} = 0$) and $\mathcal{U}_4$ ($\alpha_{12} = 1$). Plant 5 is adjacent to the target plant 2 and their output is highly correlated, which justifies the importance of feature 12. The two splits shown in Table II induce the data-driven partition $\mathcal{T} = \{\mathcal{U}_1, \mathcal{U}_3, \mathcal{U}_4\}$. Algorithm 4 proceeds to split on measurements from either plant 2 or adjacent plants until $Q = 10$ subsets are found. All subsequent splits occur on subsets induced by previously fixing a feature as missing. Similar results are observed for the NN base forecasting models, with initial splits being the same as the ones presented in Table II, whereas later splits occasionally differ from the LR base forecasting models.

TABLE II
FIRST 2 SPLITS OF ALGORITHM 4 FOR LR−ARF (LEARN) ($h = 1$).

Subset	Next Split	$100 \cdot$ UB	$100 \cdot$ LB	RelGap (%)
$\mathcal{U}_0$	Feat. 3 (plant 2, t)	5.05	0.08	61.21
$\mathcal{U}_1 = \mathcal{U}_0 \cap \{\alpha_3 = 0\}$	-	0.28	0.08	2.40
$\mathcal{U}_2 = \mathcal{U}_0 \cap \{\alpha_3 = 1\}$	Feat. 12 (plant 5, t)	4.71	0.14	32.42
$\mathcal{U}_3 = \mathcal{U}_2 \cap \{\alpha_{12} = 0\}$	-	0.52	0.14	2.71
$\mathcal{U}_4 = \mathcal{U}_2 \cap \{\alpha_{12} = 1\}$	Feat. 4 (plant 5, $t-1$)	4.71	0.19	23.26

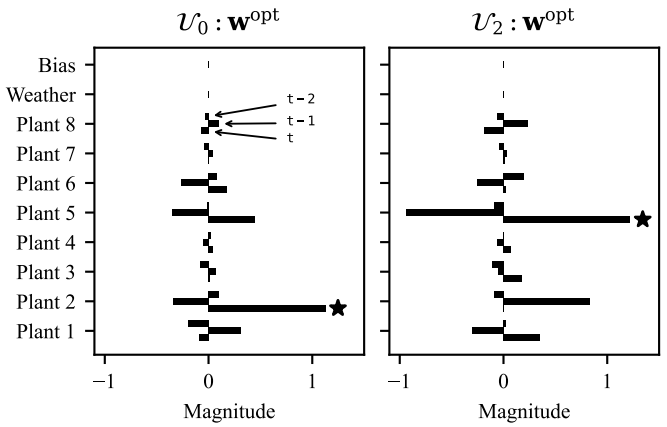


Fig. 4. Optimistic weights for subsets $\mathcal{U}_0, \mathcal{U}_2$ for LR−ARF (learn) ($h = 1$). The star marker indicates the selected feature selected for a split. Measurement data are grouped per wind plant, in the order of $t, t-1, t-2$ moving upwards.

2) Optimistic Weights: Figure 4 plots the optimistic weights, $\mathbf{w}^{\text{opt}}$, for the LR−ARF (learn^Q) model for subsets $\mathcal{U}_0$, $\mathcal{U}_2$. All features, except the bias, represent normalized wind production. The left subplot shows that the measurement of plant 2 at period t has the highest absolute weight among all features, which explains the first split, followed by the measurement of plant 5 at period t. The latter becomes the most important feature for $\mathcal{U}_2$, as shown in the right subplot, which explains the second split. Notably, a few critical features may impact forecast accuracy considerably, as happens for plant 2 and 5 measurements. Algorithm 4 essentially approximates the optimal but impractical retraining; it finds a small number of critical missing feature combinations and learns dedicated (or "optimistic") parameters for each, whereas the remaining, less important, combinations are dealt with by adversarial training. Conditioning parameters on feature availability also enables higher flexibility compared to fixed partitions, leading to higher accuracy for fewer subsets.

3) Longer Horizons: For longer horizons, we observe that both the impact of missing data and the gain from data-driven partitions decreases. As h increases, the weather-based feature becomes increasingly important, leading to different data-driven partitions compared to $h = 1$. For instance, considering LR−ARF (learn^{10}), for $h = 4, 8$, the first split occurs at the measurement of plant 5 at period t, whereas for $h = 16$ it occurs at the measurement of plant 6 at period t. Moreover, splits now occur on subsets induced by previously fixing features as available, which did not occur for $h = 1$.

4) RF vs. ARF: We now examine the effect of the adaptation methods, i.e., compare RF and ARF formulations. By design, for a given subset $\mathcal{U}_q$, both formulations learn the same optimistic parameters (as $\mathbf{D}^{\text{opt}} = \mathbf{0}$) and, hence, the same data-driven partitions. Figure 5 plots the adversarial weights of LR−RF and LR−ARF for $\mathcal{U}_0$. The left subplot shows that RF leads to conservative parameters as it sets all measurement weights close to 0 (which could be missing) and assigns higher weights to the bias and the weather forecast (that are always available). The right subplot shows the adversarial weights of ARF, conditioned on the missing measurement of plant 2 at period t. The linear corrections $\mathbf{D}^{\text{adv}}_{[3,:]}$ counterbalances the nullified w_3^{adv}, which is large and positive, by increasing the rest of the weights; higher weights are assigned to the weather forecast and measurements of plants 2 and 5 at period $t-1$, which are highly correlated with the measurement of plant 2 at period t. Note that these adversarial weights would change (adapt) if a different feature was missing. Overall, the ARF formulation leads to a less conservative (adaptive) parametrization compared to the RF formulation.

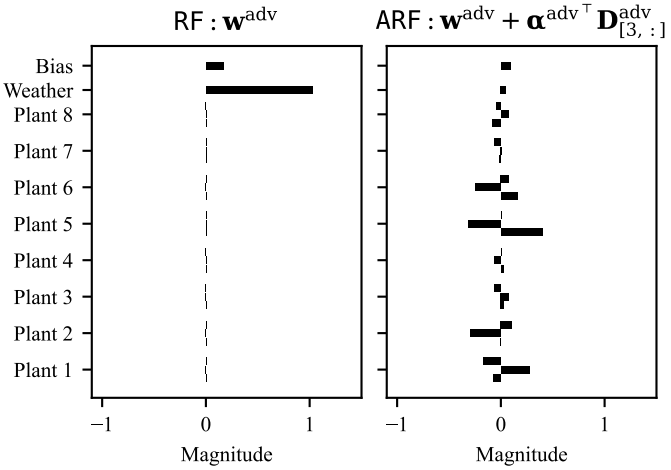


Fig. 5. Adversarial weights for LR−RF and LR−ARF ($h = 1$) for $\mathcal{U}_0$.

TABLE III
LR−ARF (LEARN^Q) PERFORMANCE AS A FUNCTION OF Q ($h = 1, \mathbb{P}_{0\to1} = 0.2, \mathbb{P}_{1\to1} = 0.9$).

Q	1	2	5	10	20	50	100
Test RMSE (%)	16.06	13.03	10.54	8.14	7.51	7.45	7.43
Max. RelGap (%)	61.21	32.42	17.94	6.16	2.58	1.89	1.64

C. Impact of Number of Subsets Q

We now examine the impact of the number of subsets Q of the data-driven partitioning algorithm. Consider the LR−ARF (learn^Q) model trained for $h = 1$ and evaluated for $\mathbb{P}_{0\to1} = 0.2, \mathbb{P}_{1\to1} = 0.9$ — see bottom right subplot of Fig. 3a. Table III presents the RMSE as a function of Q alongside the maximum relative gap, $\max_{\mathcal{U}_q \in \mathcal{T}} \texttt{RelGap}(\mathcal{U}_q)$. As expected, increasing Q lowers all metrics in Table III, leading to a lower RMSE than LR−Imp for as few as $Q = 2$ subsets and LR−ARF (fixed) for $Q = 5$. This is intuitive as a small number of features, namely measurements from plants 2 and adjacent plants, impact accuracy considerably. The RMSE plateaus for $Q > 20$; setting $Q = 100$ leads to an RMSE difference of 0.08% compared to $Q = 20$.

The maximum RelGap is highly correlated with the RMSE, decreasing rapidly for small Q and slowly for larger Q.

For longer forecast horizons, we observe that all metrics plateau earlier, which is expected as the impact of missing data and, subsequently, the gains of data-driven partitions are smaller. For $Q = 1$, the maximum RelGap is approximately 5.6% for $h = 4$, 2.2% for $h = 8$, and 0.83% for $h = 16$. For RelGap values smaller than 3% the effect on the RSME is in the second decimal. Overall, the maximum RelGap and its decrease rate correctly identify the potential benefits of further splits. Therefore, one could assume that if the RMSE, evaluated on a hold-out set, is satisfactory and the maximum RelGap decreases slowly, Algorithm 4 could arguably terminate early.

VI. Conclusions

Missing data during real-time operations compromises forecast accuracy and leads to suboptimal downstream decisions. Using ideas from adaptive robust optimization and adversarial machine learning, we developed linear- and neural network-based short-term wind power forecasting models whose parameters seamlessly adapt to the available features. Our robust and adaptive forecasting models do not rely on historical missing data patterns and are suitable for real-time operations under stringent time constraints. We conducted extensive numerical experiments on short-term wind power forecasting with horizons ranging from 15 minutes to 4 hours ahead, comparing against the industry standard imputation approach. Our proposed models were on par with imputation when data were missing for very short periods (i.e., when the latest wind power measurement was missing) and significantly better when data were missing for longer periods, hence hedging effectively against worst-case scenarios.

References

[1] Global Power System Transformation Consortium, "Vision for the control room of the future report," 2023. [Online]. Available: https://globalpst.org/vision-for-the-control-room-of-the-future-report/

[2] U. Helman, B. F. Hobbs, and R. P. O'Neill, "The design of US wholesale energy and ancillary service auction markets: Theory and practice," in *Competitive electricity markets*. Elsevier, 2008, pp. 179–243.

[3] N. Polyzotis, S. Roy, S. E. Whang, and M. Zinkevich, "Data management challenges in production machine learning," *Proc. ACM SIGMOD Int. Conf. Management Data*, vol. Part F1277, pp. 1723–1726, 2017.

[4] A. Coville, A. Siddiqui, and K.-O. Vogstad, "The effect of missing data on wind resource estimation," *Energy*, vol. 36, no. 7, pp. 4505–4517, 2011.

[5] European Centre for Medium-Range Weather Forecasts, "2016 Survey: MARS," 2016. [Online]. Available: https://www.ecmwf.int/en/newsletter/149/news/survey-shows-mars-users-broadly-satisfied

[6] D. B. Rubin, "Inference and missing data," *Biometrika*, vol. 63, no. 3, pp. 581–592, 1976.

[7] I. R. White, P. Royston, and A. M. Wood, "Multiple imputation using chained equations: Issues and guidance for practice," *Statistics in medicine*, vol. 30, no. 4, pp. 377–399, 2011.

[8] D. Bertsimas, A. Delarue, and J. Pauphilet, "Simple imputation rules for prediction with missing data: Contrasting theoretical guarantees with empirical performance," *arXiv:2104.03158*, 2024.

[9] J. Josse, J. M. Chen, N. Prost, G. Varoquaux, and E. Scornet, "On the consistency of supervised learning with missing values," *Stat. Papers*, vol. 65, no. 9, pp. 5447–5479, 2024.

[10] D. Bertsimas, A. Delarue, and J. Pauphilet, "Adaptive optimization for prediction with missing data," *arXiv:2402.01543*, 2024.

[11] R. Tawn, J. Browell, and I. Dinwoodie, "Missing data in wind farm time series: Properties and effect on forecasts," *Electr. Power Syst. Res.*, vol. 189, p. 106640, 2020.

[12] A. Pierrot and P. Pinson, "Data are missing again—reconstruction of power generation data using k k-nearest neighbors and spectral graph theory," *Wind Energy*, vol. 28, no. 1, p. e2962, 2025.

[13] H. Wen, "Probabilistic wind power forecasting resilient to missing values: an adaptive quantile regression approach," *Energy*, vol. 300, p. 131544, 2024.

[14] H. Wen, P. Pinson, J. Gu, and Z. Jin, "Wind energy forecasting with missing values within a fully conditional specification framework," *Int. J. Forecasting*, 2023.

[15] ——, "Tackling missing values in probabilistic wind power forecasting: A generative approach," *arXiv:2403.03631*, 2024.

[16] A. Stratigakos, P. Andrianesis, A. Michiorri, and G. Kariniotakis, "Towards resilient energy forecasting: A robust optimization approach," *IEEE Trans. Smart Grid*, pp. 1–1, 2023.

[17] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Int. Conf. on Learning Representations*, 2018.

[18] W. Xu and F. Teng, "Availability adversarial attack and countermeasures for deep learning-based load forecasting," in *2023 IEEE Belgrade PowerTech*, 2023, pp. 01–06.

[19] D. Bertsimas and D. den Hertog, *Robust and adaptive optimization*. Dynamic Ideas LLC, 2020, vol. 958.

[20] A. Wasilkoff, P. Andrianesis, and M. Caramanis, "Day-ahead estimation of renewable generation uncertainty set for more efficient market clearing," in *2023 IEEE Power & Energy Society General Meeting (PESGM)*, 2023, pp. 1–5.

[21] P. Andrianesis, D. Bertsimas, T. Koukouvinos, and A. G. Koulouras, "Ensembling wind forecasting models to construct data-driven uncertainty sets in robust optimization," in *2024 IEEE Power & Energy Society General Meeting (PESGM)*, 2024, pp. 1–5.

[22] D. Bertsimas, T. Koukouvinos, and A. G. Koulouras, "Constructing uncertainty sets from covariates in power systems," *IEEE Trans. Power Syst.*, pp. 1–12, 2025.

[23] D. Bertsimas and C. Caramanis, "Finite adaptability in multistage linear optimization," *IEEE Trans. Autom. Control*, vol. 55, no. 12, pp. 2751–2766, 2010.

[24] K. Postek and D. d. Hertog, "Multistage adjustable robust mixed-integer optimization via iterative splitting of the uncertainty set," *INFORMS J. on Computing*, vol. 28, no. 3, pp. 553–574, 2016.

[25] A. Stratigakos, S. Pineda, J. M. Morales, and G. Kariniotakis, "Interpretable machine learning for DC optimal power flow with feasibility guarantees," *IEEE Trans. Power Syst.*, 2023.

[26] J. M. Danskin, *The theory of max-min and its application to weapons allocation problems*. Springer Science & Business Media, 2012, vol. 5.

[27] Y. Chen, Y. Tan, and B. Zhang, "Exploiting vulnerabilities of load forecasting through adversarial attacks," in *Proc. of the Tenth ACM Int. Conf. on Future Energy Syst.*, 2019, pp. 1–11.

[28] R. Bryce, C. Feng, B. Sergi, R. Ring-Jarvi, W. Zhang, and B.-M. Hodge, "Solar, wind, and load forecasting dataset for MISO, NYISO, and SPP balancing areas," 2023. [Online]. Available: https://www.nrel.gov/docs/fy24osti/83828.pdf

[29] S. Alessandrini and S. Sperati, "Characterization of forecast errors and benchmarking of renewable energy forecasts," in *Renewable Energy Forecasting*, ser. Woodhead Publishing Series in Energy, G. Kariniotakis, Ed. Woodhead Publishing, 2017, pp. 235–256.

[30] M. Galen, N. Grue, A. Lopez, D. Heimiller, M. Rossol, G. Buster, and T. Williams, "The renewable energy potential (rev) model: A geospatial platform for technical potential and supply curve modeling," 2021. [Online]. Available: https://www.nrel.gov/docs/fy19osti/73067.pdf