Multivariate Federated Tree-based Forecasting Combining Resilience and Privacy: Application to Distributed Energy Resources

Lukas Stippel, Simon Camal, Georges Kariniotakis

Multivariate Federated Tree-based Forecasting Combining Resilience and Privacy: Application to Distributed Energy Resources

Lukas Stippel, Simon Camal, Georges Kariniotakis
Mines Paris, PSL University, Centre for processes, renewable energy and energy systems (PERSEE),Sophia Antipolis, France
{Lukas.Stippel, Simon.Camal, Georges.Kariniotakis}@minesparis.psl.eu

***Abstract*—Using spatiotemporal data can increase forecasting accuracy for distributed energy resources such as wind farms, solar power plants, and electric vehicle charging stations. The underlying forecasting models assume that data are shared by the different data owners, which is not always feasible because of privacy and confidentiality constraints. Existing methods for privacy-preserving data-sharing are based mainly on linear or non-linear univariate models. This is quite limiting concerning the potential of non-linear multivariate forecasting approaches, since these models can be computationally more efficient than univariate approaches and can learn non-linear relationships. This paper introduces a new federated forecasting approach implementing a multivariate non-linear learning scheme based on decision trees while enabling fully decentralized computation. Results show higher predictive performance than existing linear or univariate energy forecasting models while preserving privacy.**

***Index Terms*—Distributed energy resources, Energy Forecasting, Federated Learning, Resilience, Privacy constraints**

I. Introduction

The digitalization of power systems and the Internet of Things enable data collection from many sources. This wealth of data improves the forecasting accuracy of, for example, renewable energy sources (RES) production [1], [2] and the electricity demand of electric vehicles (EV) at charging stations, [3] by leveraging spatiotemporal information and consequently improving decisions in power systems. Using existing measurements from neighboring RES plants, wind farms [1], or photovoltaic (PV) plants [2] improves the predictability of their output by up to 20% for next 6 hours ahead. However, data are usually owned by different actors who are unwilling to share information for privacy and business reasons. Additionally, the data stem from heterogeneous sources, and their availability is not always guaranteed. Addressing these issues in a combined approach requires efficient computation strategies. Across all fields, the approach for privacy-preserving data-sharing is typically federated learning (FL) [4]. FL can be separated into "horizontal" federated learning, which solves the problem of data scarcity via data-sharing, and "vertical" federated learning (VFL), which utilizes the information provided by the different data owners to increase the accuracy of the prediction. A common example of VFL in distributed energy resources (DER) is data-sharing between different EV-charging stations in a city to predict the load demand more accurately [3].

In the frame of DER forecasting, a linear VFL model was developed by [1] based on distributed, decentralized privacy-preserving multivariate regression. However, many exogenous variables display non-linear relationships with the target variable (e.g., predicted wind speed vs. active wind power), which can be approximated by applying dedicated transformations like cubic splines. Nevertheless, this requires an exact understanding of the relationship beforehand and is computationally infeasible for numerous heterogeneous exogenous variables.

Therefore, non-linear models, which can capture these relationships by design, are better candidates for VFL-based DER forecasting. The superior leveraging of spatiotemporal information by non-linear models is shown in an unencrypted data-sharing setting for load-forecasting at EV-charging stations in [3], [5]. The two common structures are gradient-boosted tree models (GBT) and neural-network (NN) based models. NN-based models have seen attempts to use homomorphic-encryption approaches, which are infeasible regarding computation time [6], and differential privacy, which lowers the accuracy [7].

Consequently, we focus on GBT-based implementations whose algorithm is a natural fit for VFL [8] because it iterates over each feature separately. Additionally, as discussed in [9], tree-based methods are more resilient than NN structures and require less fine-tuning. Furthermore, they take a native approach to handling missing data by choosing a default direction [10]. Missing data are not usually considered in federated applications. Hence, trees are more attractive for applications where reliability, missing data in the application itself, and ease of usage are essential, like power systems. The most popular implementation for VFL based on the extreme boosted Gradient Trees (XGB) [10] approach is SecureBoost [11]. The authors propose a univariate lossless approach applying homomorphic encryption. The choice of encryption leads to high computation time and privacy issues due to its complexity

Part of this work is carried out in the framework of the AI-NRGY project, funded by France 2030 (Grant No: ANR-22-PETA-0004)

and handling of split instances [12].

A GBT-based univariate VFL model is presented in [13], which masks the instances and utilizes secret-sharing for a lossless,more computationally effective approach. Nevertheless, univariate trees tend to overfit and do not scale efficiently at inference [14]. Thus, a multivariate federated approach seems more suitable to derive DER production or consumption forecasts at multiple sites using local information.

A multivariate extension of Secureboost [11] is proposed in [15], but the extension does not address the security issue of sharing too much information in regard to the instances. The extension is based on the multivariate unencrypted GBT approach of [14]. In this paper, we propose a model that translates the univariate approach of [13] into a multivariate, faster, decentralized model, fitting the requirements of distributed forecasting in DER settings. We present a short summary in Table I, highlighting our contributions.

II. Related Work and Contributions

Our contributions can be summarized as follows.

- We combine and extend the univariate federated approach of [13] and the multivariate unencrypted approach of [14] to achieve a multivariate federated, decentralized tree model. This model is lossless, taking advantage of the properties of secret-sharing [16] to achieve a low computation time.
- We remove the encrypted optimization step of the weight calculation by utilizing the multivariate approach to calculate the respective weights for each party independently.
- We adopt the histogram-based binning method standard to lightgbm [17] in a secret-sharing setting. This allows us to calculate instances based on differences and faster binning.
- We analyze a naive approach to dealing with missing data in a data-sharing setting.

To the best of our knowledge, this is the first work combining advancements in both areas by introducing a multivariate secret-sharing tree method for DER forecasting.

In the next section, we present the methodology, starting with an introduction of our setting, followed by a presentation of the encryption method based on secret-sharing and the developed learning model based on gradient-boosted trees.

III. Methodology

A. Background

It is assumed that there are $D \in \mathbb{N}$ data owners. Let $\mathbf{O}$ describe the set of D owners, with O_m denoting owner $m \in [D]$. Each owner O_m owns a feature matrix $X_m \in \mathbb{R}^{N \times p_m}$ with $N \in \mathbb{N}$ observations and $p_m \in \mathbb{N}$ features. $P = \sum_{m \in [D]} p_m$ describes the total number of features. The respective target is $Y_m \in \mathbb{R}^N$. Following the general setting, we discuss the encryption technique of secret-sharing. Let $x, y \in \mathbb{R}$ belong to the owners O_1 and O_2, respectively, and serve as example variables throughout this article. We denominate an encrypted value as $< x >$. Suppose O_1 wants to encrypt x. O_1 sends a random share $< x >^m \in \mathbb{R}$ to each other data owner $O_m \in \mathbf{O} \setminus O_1$, and constructs its own share as $< x >^1 = x - \sum_{m \in [D] \setminus 1} < x >^m$. Conversely, the decryption is defined as $dec(< x >) = \sum_{m \in [D]} < x >^m = x$.

In secret-sharing, it is possible to perform addition and multiplication with encrypted $< x >$ and $< y >$. We can perform $x + y = dec(< x > + < y >) = dec(< x + y >)$ and $x \odot y = dec(< x > \otimes < y >)$ where $\otimes$ describes the secret-sharing multiplication as defined by [13]. Additionally, secret-sharing possesses commutative, associative, and distributive properties for addition and multiplication and can be vectorized [13]. However, division is impossible and yet necessary in a GBT-based model.

We propose the following modifications to the multiplication process, which allows us to include a dynamic coordinator in the group of data owners.

Suppose that among D owners, owners O_1 and O_2, intend to multiply x and y, respectively, such that $x \cdot y = z$. First, a coordinating owner O_c is selected from $\mathbf{O} \setminus \{O_1, O_2\}$. O_c returns its shares of x and y to O_1 and O_2, respectively. Then, O_c creates a beaver triplet, $a, b, c \in \mathbb{R}$ such that $a * b = c$. Next O_c shares a, b, and c. The remaining owners perform the multiplication as defined in [13]. Afterward, O_1 shares part of its share with O_c. Therefore, the multiplication is completed. Hence, we have reduced the number of participants and decentralized the process.

All the aforementioned properties of secret-sharing lead to Lemma III.1 and Corollary (Cor) III.1.

Lemma III.1. *Let $k, l \in \mathbb{R}^N$ be two vectors, $s_{\mathbf{I}} \in \mathbb{R}^N$ be an indicator vector representing instance space $\mathbf{I}$ and $\lambda > 0$ and $q > 1$. Let $D \in \mathbb{N}$ describe the number of data-owners. Then, the following equation holds and is privacy-preserving.*

$$\frac{\left(\sum_{i \in \mathbf{I}} k_i\right)^2}{\sum_{i \in \mathbf{I}} l_i + \lambda} = \sum_{m \in [D]} (< q >^m \otimes \left(\sum_{i=1}^{N} (< k >^m \otimes < s_{\mathbf{I}} >^m)_i\right) \otimes \left(\sum_{i=1}^{N} (< k >^m \otimes < s_{\mathbf{I}} >^m)_i\right) \otimes < (\frac{1}{q \cdot (\sum_{i \in \mathbf{I}} l_i + \lambda)}) >^m), \quad (1)$$

with $< (\frac{1}{q \cdot \sum_{i \in \mathbf{I}} l_i + \lambda}) >^m$ build via constructing $\sum_{m \in [D]} < q >^m \otimes (\sum_{i=1}^{N} (< s_{\mathbf{I}} >^m \otimes < l >^m)_i + < \lambda >^m)$.

Proof. The proof follows from applying the commutative and distributive properties of secret-sharing. Both multiplications with q result in a multiplication with one and, therefore, do not change the result but enable a privacy-preserving restoration of the divisor. □

Lemma III.1 removes the need to circumvent division, which is usually inaccurate or computationally expensive.

Cor III.1 allows parallel histogram-based binning and calculations on shares of data owners by enabling purely local

TABLE I: Summary of VFL Models in DER-Settings

Model	Decentralized	Multivariate	Computationally feasible	Non-linear learning	Lossless encryption
Lasso ADMM [1]	✓	✓	✓		✓
Secureboost [11]				✓	✓
MPFED-XGB [13]			✓	✓	✓
Proposed Model	✓	✓	✓	✓	✓

share manipulations to construct the right side of instances and the right side of histograms by building the difference between the entire instance/histogram and the left instance/histogram, respectively.

Corollary III.1. *For three values $x,\ y,\ z$ with $x,\ y \in \mathbb{R}$ with $z = x + y$, it holds that if $< x >$ and $< z >$ are encrypted then $< y >=< z > - < x >=< z - x >$ and therefore $< y >^m=< z >^m - < x >^m,\ \forall m \in [D]$.*

B. Federated Multivariate Tree Model

Data-sharing in DER applications is collaborative. Therefore, we assume a semi-honest setting [18]. This means owners will follow the protocol but try to infer as much information from others as possible. Furthermore, joining based on a pre-agreement to collude against one owner violates this assumption [19]. If one assumes that owners will not follow the protocol, measures like zero-knowledge-proofs are necessary [20].

Commonly, for VFL, there are active data owners O_a which own feature matrices X_a and target data Y_a, and passive data owners O_p which only provide additional information via their respective feature matrices X_p. In our setting, each owner is an active owner. Therefore, the output dimension is defined by the number of data owners. Algorithm 1 gives a high-level overview of the real-world application.

Algorithm 1 Systematic steps of the frame work

Inputs: On each data owner $O_m \in \mathbf{O}$ private data in the form of X_m and Y_m.

Output: Trained multivariate treemodel for joint prediction.

```
Draw up a contractual agreement on the targets and period.
Establish communication protocols, hyperparameters, e.g. the number of trees T.
for Tree t ∈ [T] do
    for node in t do
        Perform joint encrypted gain calculation Alg. 3.
        Build tree node.
    end for
    Calculate loss using the current ensemble.
end for
Perform joint predictions Fig. 2.
```

C. Multivariate Encrypted Extreme Gradient Boosted Trees

A multivariate tree (MV-Tree) matches the input vector $x \in \mathbb{R}^P$ to a corresponding leaf $\mathbf{w} \in \mathbb{R}^D$, denoted as $f(x)$. Hence, a prediction can be made as

$$\hat{y} = \sum_{t=1}^{T} f_t(x), \tag{2}$$

which is the sum of the T regression trees.

XGB [10] optimizes the loss via an additive method by adding a $(T+1)th$ tree. Furthermore, the authors add regularization in the form of punishing the number of trees ($\gamma \geq 0$) and the complexity of the trees with $\lambda > 0$. Here, we present the work of [14] to extend it to our privacy-preserving use case. The loss function has to be twice differentiable and convex. Simplifying via the 2nd order Taylor-Approximation gives us the objective for the $(t+1)th$ prediction,

$$\mathcal{L}^{t+1} = \sum_{i=1}^{N} l(\hat{\mathbf{y}}_i, \mathbf{y}_i) + (\mathbf{g})_i^T \mathbf{w} + \frac{1}{2}\mathbf{w}^T (\mathbf{H})_i \mathbf{w} + \frac{\lambda}{2} \mid\mid \mathbf{w} \mid\mid_2^2 + \gamma \cdot T. \tag{3}$$

In this case $(\mathbf{g})_i \in \mathbb{R}^D$ are the gradients and $(\mathbf{H})_i \in \mathbb{R}^{D \times D}$ the Hessian matrix of $\mathcal{L}$ belonging to the ith sample and $\hat{\mathbf{y}}_\mathbf{i}$. According to [10], we transform the loss function to a function of the leaves. Suppose $\mathbf{I}$ is the instance space belonging to the optimal leaf $\mathbf{w}^\star$. Equation (4) gives the solution for $\mathbf{w}^\star$. This is a direct generalization of the univariate case [14]. However, the dimensionality of $(\mathbf{H})_i$ makes an inversion computationally infeasible.

Hence, the choice of the loss function is essential. If the loss function is separable, the gradients become independent. Therefore, the Hessian matrix becomes diagonal and easily invertible. We can calculate the leaf-weight of O_j by solving (5) [14].

$$\mathbf{w}^\star = -\left(\sum_{i \in \mathbf{I}} (\mathbf{H})_i + \lambda \mathbf{I} \mathbf{D}_\mathbf{D}\right)^{-1} \cdot \left(\sum_{i \in \mathbf{I}} (\mathbf{g})_i\right). \tag{4}$$

$$\mathbf{w}_j^\star = -\frac{\sum\limits_{i \in \mathbf{I}} (g_j)_i}{\sum\limits_{i \in \mathbf{I}} (h_j)_i + \lambda}. \tag{5}$$

IV. Extension to a Privacy-Preserving Setting

Considering our privacy-preserving setting, this implies the following. In the DER setting, the common loss functions for regression and classification are the $l2$-loss function and binary cross-entropy, respectively. Both are additively separable by definition. Furthermore, and imperative for the privacy-preserving aspect, calculating gradients and Hessians does not require any interaction between the parties. This property allows us to apply the following procedures for calculating the weights and gains at each split point. Each data owner only requires the information of the instance space $\mathbf{I}$ to calculate

their weight independently. Thus, each jointly constructed tree stops at the leaf's parent. The calculation of encrypted $< \sum_{i \in \mathbf{I}} (\mathbf{g}_j)_i >$ and $< \sum_{i \in \mathbf{I}} (\mathbf{h}_j)_i >$ for owner O_j requires Cor III.1, as we operate on the whole instance space. Algorithm 2 shows the calculation of the weights. Let $< s >$ describe the encrypted indicator vector, identifying the instance, and let therefore $< g_j > \in \mathbb{R}^N$ and $< h_j > \in \mathbb{R}^N$ describe the encrypted vectors of the gradients and the Hessians of owner O_j containing all samples.

Algorithm 2 Calculation of weights

Inputs: On each data owner $O_m \in \mathbf{O}$, $< s >^m$, $< g_j >^m$, $< h_j >^m \in \mathbb{R}^N \; \forall O_j \in \mathbf{O}$, $\lambda > 0$.
Output: w_j on each data owner O_j $j \in [D]$.

1: **for** $O_j \in \mathbf{O}$ in parallel **do**
2: Perform $< s > \otimes < g_j >$
3: $=< g_j s >$ and $< s > \otimes < h_j >=< h_j s >$.
4: Perform on all data owners in parallel $< \sum_{i \in \mathbf{I}} (g_j)_i >^m = \sum_{i=1}^{N} < g_j s >_i^m$ and $< \sum_{i \in \mathbf{I}} (h_j)_i >^m = \sum_{i=1}^{N} < h_j s >_i^m$.
5: Perform $< \sum_{i \in \mathbf{I}} (h_j)_i > + < \lambda >$.
6: Perform $< q > \otimes < \sum_{i \in \mathbf{I}} (g_j)_i >$ and
7: $< q > \otimes < (\sum_{i \in \mathbf{I}} (h_j)_i + \lambda) >$.
8: Restore $q \cdot \sum_{i \in \mathbf{I}} (g_j)_i$ and $q \cdot (\sum_{i \in \mathbf{I}} (h_j)_i + \lambda)$ on the respective owners.
9: **end for**
10: Each owner O_j calculates $w_j = -\frac{\sum_{i \in \mathbf{I}} (\mathbf{g}_j)_i}{\sum_{i \in \mathbf{I}} (\mathbf{h}_j)_i + \lambda}$ in parallel.

In a GBT-based model, the algorithm searches the maximum gain over all possible splits proposed by the partitions of the instance space by each owner, feature, and bin, respectively [10]. Suppose the proposed split partitions the current instance space $\mathbf{I}$ into $\mathbf{I}_R$ and $\mathbf{I}_L$ then the gain would be:

$$gain = \sum_{m \in [D]} \left\{ \frac{\left(\sum_{i \in \mathbf{I}_L} (g_m)_i \right)^2}{\sum_{i \in \mathbf{I}_L} (h_m)_i + \lambda} + \frac{\left(\sum_{i \in \mathbf{I}_R} (g_m)_i \right)^2}{\sum_{i \in \mathbf{I}_R} (h_m)_i + \lambda} - \frac{\left(\sum_{i \in \mathbf{I}} (g_m)_i \right)^2}{\sum_{i \in \mathbf{I}} (h_m)_i + \lambda} \right\} - \gamma. \quad (6)$$

Thus, we illustrate how this process is combined with privacy preservation.

The gain at each node is the sum of the individual owners' gains. However, this information is sensitive. Therefore, denote the overall gain as $gain$ and the individual gain as $gain_{indv}$. We can calculate the gain of an instance $\mathbf{I}$ as $gain = \sum_{m \in [D]} < gain >^m$, with $< gain >^m = \sum_{indv \in [D]} < gain_{indv} >^m$ by applying Cor III.1.

The individual gain can be calculated as in (7).

$$< gain_{indv} >^m =< Leftproposedsplit >^m + < Rightproposedsplit >^m - < Instance >^m . \quad (7)$$

γ is omitted as it can be tuned globally. Here $Leftpropsedsplit$, $Rightproposedsplit$, and $Instance$ represent the respective summands in (6). The $Leftproposedsplit$ and the $Instance$ split can be calculated using Lemma III.1. Utilizing Cor III.1 to compute $Rightproposedsplit$ lowers the computational complexity. Therefore, the gain can be computed losslessly and encrypted. Once calculated in an iteration, the gains are restored to the respective feature owners. Afterwards, the owners broadcast their local maxima to determine the global maximum. The owner with the maximum stores this information securely, encrypting the proposed left instance space. All other owners store the fact that they do not possess the node.

Then, a multiplication between the encrypted instance and the proposed left side is performed. The right instance is constructed via Cor III.1. Algorithm 3 and Fig. 1 explain the algorithm. Let $Bins_t$ describe the left-side binnings proposed by data owner O_t.

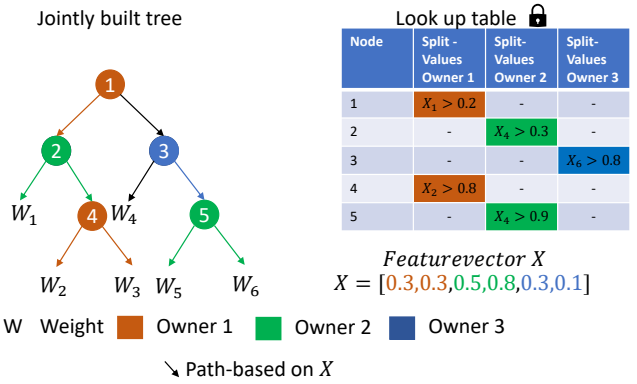

Fig. 1: Tree built through collaboration (left) and the corresponding lookup table (right), where the information is stored locally on each owner.

The prediction process is based on [13]. Each data owner traverses the tree using the knowledge in its possession, eliminating possible paths if they own the node. A possible leaf is denoted by 1, and an impossible leaf is denoted by 0. Multiplying each of these leaf indicator vectors returns the definitive leaf. We perform this unencrypted as the weights are stored and calculated independently. An example can be seen in Fig. 2.

V. Numerical Experiments

A. Data Description

We evaluate the experiments on the public windpower dataset of the Gefcom forecasting challenge 2014 [21]. We chose a cluster of four farms out of the ten available farms (Farm 1-4) to evaluate the performance of the models in

Algorithm 3 Calculation of split

Inputs: On each data owner $O_m \in \mathbf{O}$, $< s >^m$, $< g_i >^m$, $< h_i >^m \ \forall O_i \in \mathbf{O}$, λ, $bin_j \, \forall j \in Bins_t$ $\forall O_t \in \mathbf{O}$.

Output: $< s_L >$ and $< s_R >$ for the node children on the left and right side, respectively.

On each owner O_m set $localgains = \emptyset$.
for $O_t \in \mathbf{O}$ **do**
  Choose one owner to serve as the constructor of q and fulfill the role of coordinator O_c.
  for $bin \in Bins_t$ **do**
    Encrypt $< bin >$
    Calculate $< Leftproposedsplit >$, $< Rightproposedsplit >$ and $< Instance >$ for each $O \in \mathbf{O}$.
    On each data owner $O_m \in \mathbf{O}$ perform $< gain >^m = \sum_{i \in [D]} < Leftproposedsplit >_i^m + < Rightproposedsplit >_i^m - < Instance >_i^m$.
    Restore $gain$ and add it to $localgains$.
  end for
end for
On all data owners $O_m \in \mathbf{O}$. Calculate $localmax = max(localgains)$.
Find $Bestsplit = max(localmax)$.
if $Bestsplit > \gamma$ **then**
  Owner of $Bestsplit$ notes down feature, bin, and split value, and encrypts the respective bin as $< s_{left} >$. The other owners note that they do not possess information about this split.
  Perform $< s > \otimes < s_{left} > = < s_L >$ and
  $< s_R > = < s > - < s_L >$.
  Return $< s_L >$ and $< s_R >$ and perform the splits again.
else
  This node is a leaf, and all owners perform leaf calculations.
end if

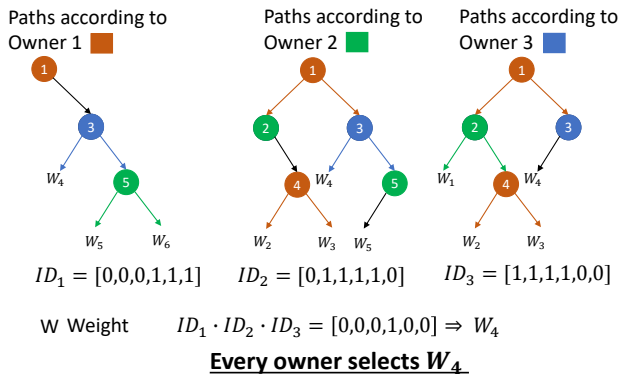

Fig. 2: Identifying the leaf during the joint prediction step.

a sharing setting. The farms are located in Australia, but their precise location is unknown. The dataset includes hourly power production observations normalized by the respective wind farm's nominal capacity. Furthermore, the predicted windspeed, at 10m and 100m, in the u and v directions at the locations of the windfarms are provided. The observed timespan is two years, from January 1, 2012, to November 30, 2013. We selected the first year as a training set and also performed 12-fold cross-validation on it for the hyperparameter tuning. The second year serves as the test set. After feature selection, we choose the six most recent observations. When including the numerical weather predictions (NWP), we provide the linear model with cubed windspeeds, as it allows a fairer comparison. A sliding window approach is applied where the model is trained on 12 months of observations and predicts the next month.

B. Model details and Evaluation metric

The models evaluated are the linear-encrypted Lasso-VAR model presented by [1], an implementation of an unencrypted MV-Tree [22], a data-sharing univariate XGB [10], a data-sharing random-forest [23] and a non-sharing version of the Lasso-AR for benchmarking [1]. The machine utilized has an i7-2300 MHz processor and 32GB of RAM. The MV-Tree was trained using the 8GB NVIDIA RTX A2000 GPU (Graphics Processing Unit). Our evaluation metrics are the Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE).

C. Hyperparameter Tuning

We tune the hyperparameters of our tree-based models using Bayesian-Optimization with the Optuna package [24], as a grid search is infeasible. Optuna's strategy yields better results than a completely random grid search. The linear models are tuned with a hyperparameter grid.

D. Results

First, we evaluate the benefit of data-sharing. The feature importance of lagged power observations at the different farms in our MV-tree is shown in Fig. 3, as a function of the prediction horizon. For the 1-h horizon, the most recent lagged production of each farm '$(t-1)$' is the most important feature, and there is little difference between farms. For further horizons, the feature '$(t-1)$' of Farm 3 dominates over the other farms, and the importance of older lags increases slightly. This shows that the MV-tree selects the relevant spatiotemporal information for each horizon.

Fig. 4 evaluates the improvement of Lasso-VAR and MV-Tree compared to a non-collaborative Lasso-AR only utilizing information of the target site, looking at RMSE and MAE. Table II gives a full comparison of the RMSE without NWP.

The effect of data-sharing is evident by the fact that all models outperform the non-data-sharing linear model, most clearly when comparing both linear approaches. It is observed that non-linearity without specific non-linear input variables is not always superior, but the gradient-based models are superior in performance to the linear and non-linear models by up

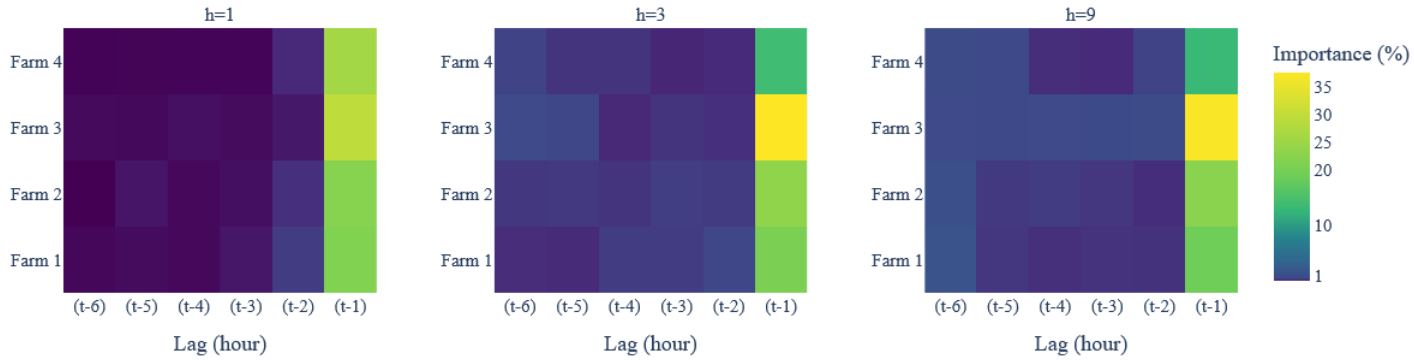

Fig. 3: MV-Tree feature importance at different hourly horizons, only using past observations. Feature importances sum to 100% for each horizon.

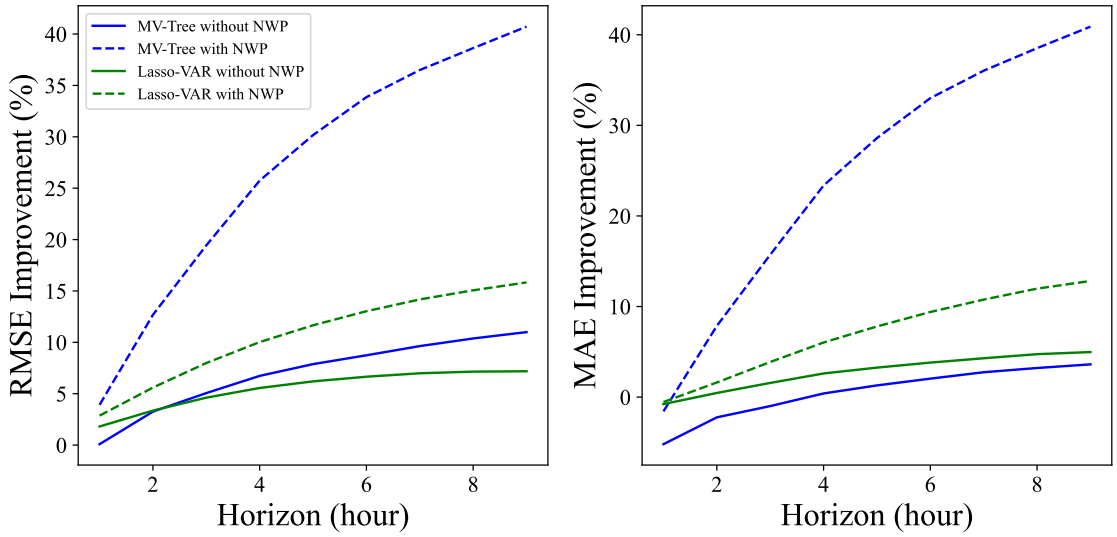

Fig. 4: Improvement of forecasting metrics of the multivariate methods vs. the non-collaborative Lasso-AR. Left: RMSE (%), Right: MAE (%).

to 4%. Even though the performance of MV-Tree and XGB is similar, the multivariate model outperforms the univariate model at farms 3 and 4, where data-sharing is employed the most. When analyzing the impact of NWP, we can see an improvement of up to 40% by the tree model and up to 15% by the linear models. The MV-Tree model outperforms the Lasso-VAR at later horizons in every metric except the MAE, excluding NWP. We assume this is due to the tree's tendency to overfit. They capture outliers better but are slightly wrong more often. Resilience to outliers is crucial in DER real-world applications. Fig. 5 verifies the losslessness of the encrypted model on three farms.

E. Resilience analysis

In real-world spatiotemporal applications, data are commonly complete during training but may have missing features during predictions [26]. A data owner, e.g., a wind farm, might be unavailable due to operational issues. Default directions are used in [10] to address this problem. We evaluate the impact of farms that are missing during the joint prediction if the default direction for missing is always left. The results are shown in Fig. 6. Compared to the impact on the linear model, the proposed model is more resilient, as the default directions handle the missing data more effectively. The proposed algorithm belongs to the regularized boosted trees and is, therefore, by design, resilient to common problems in machine learning. This includes vanishing gradients, which are avoided due to the regularization, and the problem of local minima, which is avoided by convexity and tuning. Finally, there are no mathematical errors, e.g. division by zero. This is due to the choice of $\lambda > 0$ and $q > 1$.

F. Comparison of Sparsification between the Univariate and Multivariate Approaches

For tree-based models, it is difficult to analyze sparsity. Therefore, we compare the number of different features used and their average weight. Both are consistent for univariate and MV-Tree models over all horizons. The multivariate model

TABLE II: Overview of average RMSE errors for hourly horizons 1 to 9, without NWP-data. $\star$ means the improvement is statistically significant according to the Diebold-Mariano test [25].

Model	h=1	h=2	h=3	h=4	h=5	h=6	h=7	h=8	h=9
Lasso-AR	0.0980	0.1487	0.1816	0.2065	0.2263	0.2425	0.2566	0.2679	0.2780
Lasso-VAR	0.0962	0.1436	0.1729	0.1947	0.2119	0.2260	0.2380	0.2484	0.2576
XGB	0.0975	0.1438	0.1725	0.1933	0.2093	0.2221	0.2327	0.2411	0.2489
Random Forest	0.0988	0.1466	0.1763	0.1976	0.2140	0.2266	0.2374	0.2457	0.2527
MV-Tree	0.09783	0.1436$\star$	0.17213$\star$	0.1923$\star$	0.2081$\star$	0.2211$\star$	0.2313$\star$	0.2399$\star$	0.2472$\star$

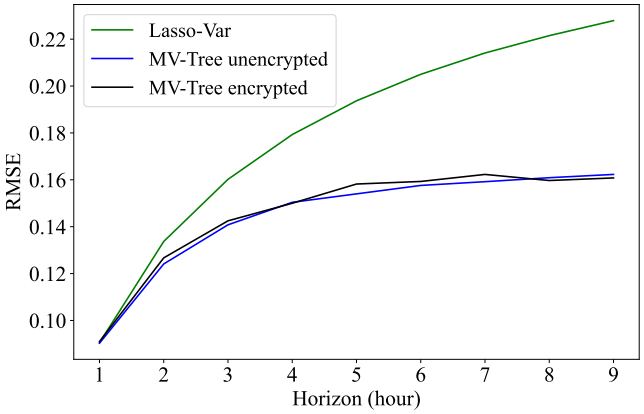

Fig. 5: Verification of losslessness of encrypted model on three farms.

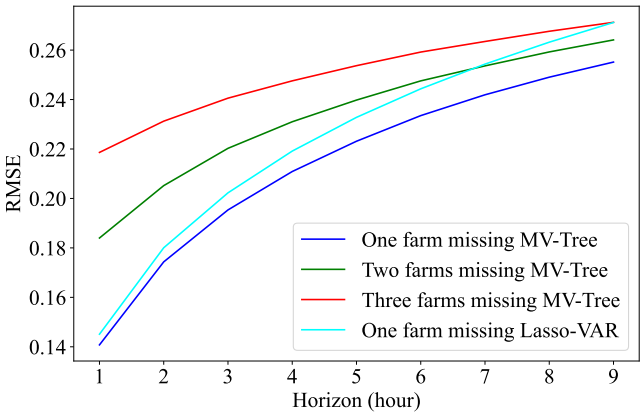

Fig. 6: Impact of farms missing during the inference process.

utilizes each feature on horizons until hour 5 and afterward up to 17 out of 24, while the univariate model utilizes three different features on average. The average weight for the MV-Tree is 0.0047 and 0.3777 for the univariate model. We can conclude the multivariate tree modifies less per iteration, but the univariate models reduce the input more effectively.

G. Security Analysis

For the general discussion of the privacy of secret-sharing, we refer to the analysis of [13]. We assume, contrary to [13], that the coordinator or coordinating owner O_c is willing to collude[1]. First, restoring an encrypted value requires all shares. This can only be achieved if shares are exchanged, hence, during the multiplication step. Suppose, as in Section III-A, that owners O_1 and O_2 want to multiply x and y. x can only be restored if: $< x >^1 = e - \sum_{i \in [D] \setminus \{1,c\}} < x >^i + a$ can be solved, as owner O_1 will not share $< x >^1$ and $< a >^1$. From the multiplication protocol [13], e is publicly known, hence colluding owners require a and all but one share of x. Hence, all owners except O_1 are required: Recall that only O_c can know a. We assign O_c dynamically, and the number of multiplications is significantly greater than the number of owners D as discussed in SectionV-H. Therefore, over all iterations, every owner will be O_c for every owner. This allows two scenarios: First, there is a pre-agreement to collude against one specific owner and restore their shares. This violates the semi-honest setting [19] and, if detected, would lead to said owner leaving the process. Second, as O_c is dynamic, all owners agree to collude against everyone and have to assume that they are themselves colluded against, all privacy is lost, and that the sharing will be stopped if collusion is detected.

In conclusion, the current protocol is threatened if $D-1$ owners collude. However, this violates the settings's assumptions.

A possible problem for federated tree models is if one data owner owns one direct path, as they can restore the instance space [13]. Univariate models force the first split to be performed by the active party to prevent a passive party from owning a direct path. This does not apply to a multivariate setting. Instead, we propose that if an owner owns a direct path, then we choose the second-highest gain at the leaf parent node. We do not currently perform this step.

H. Complexity Analysis

For the discussion of the unencrypted setting, we refer to [22] and [14]. Comparison with homomorphic encryption models is omitted, as [13] demonstrates the superior computational efficiency of secret-sharing models.

Let $depth$ describe the tree depth, N the number of samples, B the number of bins, D the number of data owners, T the total number of trees, and P the total number of input features. We benchmark our model's complexity versus a D-times applied application of the model in [13]. The inference process requires $O(Tdepth + D)$ operations due to the need to multiply indicator vectors. For the univariate model, this

[1]Collusion: Secret or illegal cooperation or conspiracy in order to deceive others.

entails $O(D \cdot (Tdepth + D))$ operations. Furthermore, it requires secret-sharing multiplication, while our data owners have their weights stored unencrypted locally. In the training phase, we compare the number of multiplications as they are the only limiting factor and the most time-consuming [13]. For a full tree, the univariate framework of [13] requires at worst $O((2^{depth} - 1)D(2P + 9P\lceil log_2 B\rceil + 9\lceil log_2 P\rceil + 14))$ multiplications. Our model utilizing Cor III.1 requires at worst $O((2^{depth} - 1)(10D))$ multiplications. This is due to the utilization of randomization, allowing division. Additionally, by calculating the overall gain via Cor III.1, we avoid multiplications. This shows the advantage of the proposed model in terms of complexity. Furthermore, the design suits GPU implementations due to high parallelizability and the avoidance of complex operations.

VI. Conclusion and Future Works

Finding a good trade-off between the potential benefits of data-sharing and computational and privacy-imposed burdens remains a key challenge. Furthermore, encryption faces the problem of being computationally expensive. This work removes the need for complex division approaches in federated tree models by combining two encryption techniques. Additionally, we extend the secret-sharing approach to a more computationally effective setting in a multivariate model while maintaining a high level of privacy. Hence, we address the need for a lossless multivariate model that can capture non-linear relationships. In addition, the resilience to missing data is natively given.

In our future work, we intend to address the security limits by modifying the secret-sharing algorithm by assigning O_c differently and changing the beaver triplet. Also, we aim to address scalability by developing a GPU implementation and adapting online and transfer learning. Further steps include reducing the complexity via targeted projections [22]. Lastly, we aim to include passive parties for relevant public data, e.g., seasonal information.

References

[1] C. Gonçalves, R. J. Bessa, and P. Pinson, "Privacy-preserving distributed learning for renewable energy forecasting," *IEEE Transactions on Sustainable Energy*, vol. 12, no. 3, pp. 1777–1787, 2021.

[2] X. G. Agoua, R. Girard, and G. Kariniotakis, "Short-term spatio-temporal forecasting of photovoltaic power production," *IEEE Transactions on Sustainable Energy*, vol. 9, no. 2, pp. 538–546, 2018.

[3] F. B. Hüttel, I. Peled, F. Rodrigues, and F. C. Pereira, "Deep spatio-temporal forecasting of electrical vehicle charging demand," 2021.

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. PMLR, 20–22 Apr 2017, pp. 1273–1282.

[5] Y. Amara-Ouali, Y. Goude, B. Hamrouche, and M. Bishara, "A benchmark of electric vehicle load and occupancy models for day-ahead forecasting on open charging session data," ser. e-Energy '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 193–207.

[6] J.-W. Lee, H. Kang, Y. Lee, W. Choi, J. Eom, M. Deryabin, E. Lee, J. Lee, D. Yoo, Y.-S. Kim, and J.-S. No, "Privacy-preserving machine learning with fully homomorphic encryption for deep neural network," *IEEE Access*, vol. 10, pp. 30 039–30 054, 2022.

[7] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[8] Y. Liu, Y. Kang, T. Zou, Y. Pu, Y. He, X. Ye, Y. Ouyang, Y.-Q. Zhang, and Q. Yang, "Vertical federated learning: Concepts, advances, and challenges," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–20, 2024.

[9] T. Januschowski, Y. Wang, K. Torkkola, T. Erkkilä, H. Hasson, and J. Gasthaus, "Forecasting with trees," *International Journal of Forecasting*, vol. 38, no. 4, pp. 1473–1481, 2022, special Issue: M5 competition.

[10] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 785–794.

[11] K. Cheng, T. Fan, Y. Jin, Y. Liu, T. Chen, D. Papadopoulos, and Q. Yang, "Secureboost: A lossless federated learning framework," *IEEE Intelligent Systems*, vol. 36, no. 6, pp. 87–98, 2021.

[12] J. G. Chamani and D. Papadopoulos, "Mitigating leakage in federated learning with trusted hardware," 2020.

[13] L. Xie, J. Liu, S. Lu, T.-H. Chang, and Q. Shi, "An efficient learning framework for federated xgboost using secret sharing and distributed optimization," *ACM Trans. Intell. Syst. Technol.*, vol. 13, no. 5, sep 2022.

[14] Z. Zhang and C. Jung, "Gbdt-mo: Gradient-boosted decision trees for multiple outputs," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 7, pp. 3156–3167, 2021.

[15] W. Chen, G. Ma, T. Fan, Y. Kang, Q. Xu, and Q. Yang, "Secureboost+ : A high performance gradient boosting tree framework for large scale vertical federated learning," 2021.

[16] H. Wang and D. S. Wong, "On secret reconstruction in secret sharing schemes," *IEEE Transactions on Information Theory*, vol. 54, no. 1, pp. 473–480, 2008.

[17] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[18] L. Lyu, H. Yu, J. Zhao, and Q. Yang, *Threats to Federated Learning*. Cham: Springer International Publishing, 2020, pp. 3–16.

[19] A. Paverd, A. Martin, and I. Brown, "Modelling and automatically analysing privacy properties for honest-but-curious adversaries," *Tech. Rep*, 2014. [Online]. Available: https://ajpaverd.org/publications/casper-privacy-report.pdf

[20] Y. Wu, S. Cai, X. Xiao, G. Chen, and B. C. Ooi, "Privacy preserving vertical federated learning for tree-based models," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, p. 2090–2103, Aug. 2020.

[21] T. Hong, P. Pinson, S. Fan, H. Zareipour, A. Troccoli, and R. J. Hyndman, "Probabilistic energy forecasting: Global energy forecasting competition 2014 and beyond," *International Journal of Forecasting*, vol. 32, no. 3, pp. 896–913, 2016.

[22] L. Iosipoi and A. Vakhrushev, "Sketchboost: Fast gradient boosted decision tree for multioutput problems," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 25 422–25 435.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[24] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 2623–2631.

[25] F. X. Diebold and R. S. Mariano, "Comparing predictive accuracy," *Journal of Business & Economic Statistics*, vol. 20, no. 1, pp. 134–144, 2002.

[26] N. Polyzotis, S. Roy, S. E. Whang, and M. Zinkevich, "Data management challenges in production machine learning," in *Proceedings of the 2017 ACM International Conference on Management of Data*, ser. SIGMOD '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1723–1726.