

FHT-R: Federated Online Regression with Coordinated Hoeffding Trees for Air Quality Prediction

Daniela Menezes, Douglas Vidal, André Riker and Glauco Gonçalves

Abstract—This paper presents a case study on federated online regression for $PM_{2.5}$ prediction in distributed air quality monitoring. $PM_{2.5}$ is a widely used air quality indicator that refers to fine particulate matter smaller than $2.5 \mu m$. Originally, the Fed-VFDT framework was designed for classification. In this work, we adapt Fed-VFDT to regression using Hoeffding Tree regressors. Each monitoring station is modeled as a federated client that learns from its own local data stream, while a coordinating server selects global splits using quorum, majority vote, or best merit. Experiments on the Beijing Multi-Site Air Quality dataset, partitioned into 12 federated clients, showed that majority vote achieved the best predictive performance. These results indicate that coordinated federated tree growth is a feasible approach for distributed online air quality regression.

Keywords—Machine Learning, Federated Learning, Hoeffding Trees, Regression, Data Streams.

I. INTRODUCTION

Air pollution is a major challenge for urban environments, with effects on public health, quality of life, and environmental sustainability. According to the World Health Organization (WHO), almost the entire global population breathes air that exceeds recommended air quality limits, and ambient air pollution is associated with millions of premature deaths every year [1]. Among the pollutants monitored in urban areas, fine particulate matter ($PM_{2.5}$) is one of the most relevant indicators for air quality assessment, since it is explicitly addressed in international air quality guidelines [2].

Urban air quality monitoring has also become a distributed data problem. Measurements are commonly collected by networks of monitoring stations that produce time-ordered observations of pollutant concentrations and meteorological conditions. Because local factors such as traffic, weather, and nearby emission sources vary across stations, the resulting data streams may differ even within the same urban area [3]. This spatial heterogeneity complicates centralized modeling, while continuously transferring all raw observations to a central server may also be costly or undesirable, particularly in infrastructure-constrained deployments [4].

All authors are with the Federal University of Pará (UFPA), Belém-PA, Brazil. E-mails: daniela.pereira@itec.ufpa.br, douglas.vidal@itec.ufpa.br, ariker@ufpa.br, glaucogoncalves@ufpa.br. The authors would like to thank the National Council for Scientific and Technological Development (CNPq) (Grant nos. 408326/2023-9 and 444978/2024-0), the National Institute of Science and Technology (INCT) STREAM (CNPq Grant no. 409179/2024-8), and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

These characteristics motivate learning approaches that operate directly over distributed streams. Federated Learning (FL) provides one such alternative by keeping data on the client side and sharing only model-related information with a coordinating server [5], thus reducing continuous raw data transfer while preserving station-level learning.

Another challenge regarding this scenario is that air quality data streams are not static. Seasonal effects, meteorological variation, traffic flow, and other urban factors can change the data distribution over time, causing a problem commonly known as concept drift [6]. This non-stationary behavior can degrade models built under the assumption of a fixed data distribution. Whereas conventional offline learning models usually require data accumulation and periodic retraining in these scenarios, online learning methods address this limitation through incremental updates, which allows the model to process observations sequentially and adapt as new data arrive.

In this work, we study a federated online regression approach for $PM_{2.5}$ prediction. Our proposed method is based on Federated Very Fast Decision Trees (Fed-VFDT), originally designed for coordinated split decisions over data streams [7]. We name the proposed model as Federated Hoeffding Tree Regressor (FHT-R). While Fed-VFDT focuses on classification, FHT-R adapts the same coordinated splitting idea to a regression task through Hoeffding Tree regressors. Each monitoring station is represented as a client that learns from its own local stream, while split decisions are coordinated by the server according to an aggregation strategy.

The experiments use the air quality dataset with $PM_{2.5}$ as the prediction target. The objective is to evaluate whether the coordinated growth of a federated regression tree can provide accurate regression performance while preserving the communication-efficient behavior expected from online decision trees. A favorable outcome would indicate that this type of model can support distributed air quality monitoring scenarios by enabling prediction across multiple stations with lower dependence on centralized raw data collection. This is particularly relevant in large-scale monitoring settings and environments with limited infrastructure.

The remainder of this paper is organized as follows. Section II reviews related works on federated learning, data stream mining, and air quality monitoring. Section III presents the proposed federated regression method, including the client-side workflow and the server-side split coordination procedure. Section IV describes the experimental setup, including the dataset, partitioning strategy, implementation details, and

evaluation protocol. Section V reports and discusses the experimental results. Finally, Section VI concludes the paper and outlines directions for future work.

II. RELATED WORKS

The foundation of streaming decision trees was established by Domingos and Hulten with the Very Fast Decision Tree (VFDT) algorithm [8]. The VFDT introduced the use of the Hoeffding Bound to make split decisions with statistical guarantees using only a fraction of the data, allowing for single-pass learning. To address the challenge of non-stationary environments, Hulten et al. proposed the Concept-adapting Very Fast Decision Tree (CVFDT) [6], which maintains a sliding window of data to detect and adapt to concept drift by pruning obsolete branches. To extend this family of algorithms to regression tasks, Ikononovska et al. proposed the Fast Incremental Model Trees with Drift Detection (FIMT-DD) [9]. This work adapts the VFDT structure for continuous targets by replacing classification criteria such as Gini index or entropy with Standard Deviation Reduction (SDR). FIMT-DD demonstrated that online trees could successfully model complex regression surfaces in evolving streams. However, like the original VFDT, FIMT-DD assumes centralized data availability. This assumption is often infeasible in large-scale Internet of Things (IoT) networks due to the communication overhead and centralized aggregation requirements involved in transmitting raw high-frequency data streams.

Real-world air quality sensing systems also face practical constraints related to distributed data collection and limited infrastructure. Lima et al. present SELVA [4], an Internet of Nature platform for wildfire and air quality monitoring in the Amazon region that integrates low-cost sensors, satellite data, and online/mobile interfaces. This work emphasizes the difficulty of expanding monitoring coverage due to limited infrastructure and the high cost of reference-grade air quality networks. Although SELVA is not a federated learning system, it illustrates a realistic deployment scenario in which federated methods may be especially valuable.

In the federated domain, the Fed-VFDT framework proposed by Silva et al. [7] serves as the architectural baseline for this work. Fed-VFDT addresses the challenge of structural inconsistency in distributed decision trees through a coordinated split protocol. In this framework, clients train local VFDT models over their own data streams until a split condition is reached. The client then freezes local training and sends split statistics to a central server. The server aggregates client proposals, selects a global split decision, and broadcasts it back to the federation. This protocol keeps the tree structure synchronized across clients without sharing raw data.

The Fed-VFDT formulation is designed for classification tasks and evaluates split candidates using classification criteria. In contrast, the present work adapts the same coordinated splitting protocol to a regression task, where the target variable is continuous and split quality is evaluated through variance reduction.

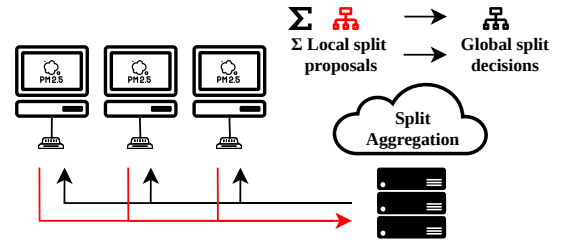


Fig. 1: Conceptual overview of the proposed federated regression workflow.

III. PROPOSED METHOD

The proposed federated model simulates a distributed air quality monitoring scenario in which each client processes a local data stream and collaborates with a central server to maintain a coordinated regression tree.

As illustrated in Fig. 1, each monitoring station is modeled as a federated client and trains a local Hoeffding Tree regressor on its own data stream and sends split proposals to the server. Local training remains decentralized, while split coordination is performed by the central server. The server aggregates these proposals and broadcasts a coordinated global split decision, keeping the tree structure synchronized across clients.

A. Federated Learning Architecture

The proposed FHT-R contains K clients and one coordinating server. Each client corresponds to one air quality measurement station and receives a local stream S_k composed of station-level observations. Each observation is represented as a pair $(x_i^{(k)}, y_i^{(k)})$, where $x_i^{(k)}$ is the feature vector and $y_i^{(k)}$ is the $PM_{2.5}$ concentration to be predicted. Each client maintains its own instance of a federated Hoeffding Tree regressor and updates the model incrementally as data arrive.

Algorithm 1 summarizes the client-side workflow adopted by FHT-R. For each new observation, the client first predicts the target value and then updates the local model with the true value. It also updates the regression metrics used in the evaluation, namely Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and coefficient of determination (R^2).

When a leaf reaches the local split condition, the client stores a pending split proposal and pauses local training until the server returns a decision. The split proposal sent to the server contains the client identifier, the tree path of the candidate leaf, the selected feature, the split threshold, the split merit, the Hoeffding bound value, leaf statistics, and child statistics when available.

The *split merit* value is inherited from the Fed-VFDT coordination, where split proposals are compared using a local merit score. In Fed-VFDT, this score is associated with classification criteria such as the Gini index [7]. In this work, the local model is a `HoeffdingTreeRegressor`; therefore, the model uses the variance reduction merit computed for a regression split.

At each round, the server receives the split proposals submitted by the clients and applies an aggregation strategy to select a global split. If a valid split is selected, the server

Algorithm 1 Client-side workflow

Require: Local data stream S_k and local regressor T_k

- 1: **if** a valid global split decision is received **then**
- 2: Update the local tree and resume training
- 3: **end if**
- 4: **if** the client is frozen with a pending split proposal **then**
- 5: Send the pending proposal and metrics to the server
- 6: **return**
- 7: **end if**
- 8: **while** the local stream is not exhausted **do**
- 9: Read the next observation $(x_i^{(k)}, y_i^{(k)})$
- 10: Predict $\hat{y}_i^{(k)}$ and update the local regressor
- 11: Update MAE, RMSE, and R^2
- 12: **if** a split proposal is generated **then**
- 13: Freeze local training
- 14: Send the split proposal and metrics to the server
- 15: **return**
- 16: **end if**
- 17: **end while**
- 18: **return** Current metrics

broadcasts the selected feature, threshold, tree path, and associated statistics to the clients. Each client then locates the corresponding leaf by its path and replaces it with a branch node using the received split information. This procedure keeps the tree structure coordinated across clients while avoiding raw data transmission.

Training continues while at least one client still has local data to process. When all clients report that their streams are exhausted, the server sends a final stop signal and terminates the experiment.

B. Server-Side Split Aggregation

Let P_t be the set of viable split proposals received by the server in round t . Each proposal $p \in P_t$ contains a feature f_p , a tree path r_p , a threshold θ_p , and a split merit m_p . Before aggregation, proposals with merit lower than the minimum split merit are discarded.

Algorithm 2 summarizes the server-side procedure used to aggregate split proposals and coordinate global tree growth. In this paper, three aggregation strategies have been considered, described as follows.

1) *Quorum-Based Method*: The **Quorum** method starts making groups $P_{r,f}$ of proposals by tree path and feature according to $P_{r,f} = \{p \in P_t : r_p = r \wedge f_p = f\}$. A group is valid only if it receives support from at least $\lceil \alpha K \rceil$ clients, where α is the quorum threshold. Among the valid groups, the server selects the group with the highest sum of split merits:

$$P_{r,f}^* = \arg \max_{P_{r,f}} \sum_{p \in P_{r,f}} m_p. \quad (1)$$

The final selected proposal is the highest-merit proposal within the selected group:

$$p^* = \arg \max_{p \in P_{r^*,f^*}} m_p. \quad (2)$$

2) *Majority-Based Method*: The **Majority vote** also makes groups of proposals $P_{r,f}$ and it selects the group with the largest number of proposals:

$$P_{r,f}^* = \arg \max_{P_{r,f}} |P_{r,f}|. \quad (3)$$

Algorithm 2 Server-side split aggregation

Require: Client results received at round t

- 1: Collect client metrics and split proposals
- 2: Aggregate MAE, RMSE, and R^2 across clients
- 3: Discard proposals below the minimum split merit
- 4: Select a proposal according to the configured aggregation strategy
- 5: **if** a valid proposal is selected **then**
- 6: Aggregate compatible child statistics
- 7: Build the global split decision using the selected path, feature, threshold, and statistics
- 8: **else**
- 9: Build an empty decision for the current round
- 10: **end if**
- 11: **if** at least one client is still active **then**
- 12: Broadcast the decision and continue training
- 13: **else**
- 14: Broadcast the stop signal and terminate
- 15: **end if**

Ties are solved using the split merit, and the final selected proposal is the highest-merit proposal in the winning group, as in (2).

3) *Best Merit Method*: Finally, the **Best merit** method selects the proposal with the highest split merit:

$$p^* = \arg \max_{p \in P_t} m_p. \quad (4)$$

It is worth noting that this strategy requires all clients to send proposals before approving a global split. Moreover, if no proposal is selected in a given round, no global split is broadcast and clients resume local training.

IV. EXPERIMENTAL SETUP

This section describes the experimental setup used for a case study of FHT-R in distributed air quality monitoring.

A. Dataset and Preprocessing

The experiments use the Beijing Multi-Site Air Quality dataset, which contains hourly air quality and meteorological measurements collected from monitoring stations in Beijing between March 1, 2013 and February 28, 2017 [10]. The original public dataset contains measurements from 12 monitoring stations and 420,768 records. In this case study, the full dataset was used with PM_{2.5} as target, after removing rows without target values.

After preprocessing, 412,029 records and 43 features remained; PM_{2.5} ranged from 2 to 999 $\mu\text{g}/\text{m}^3$, revealing strong variability and extreme values. Missing numerical feature values were imputed with the corresponding medians, while wind direction and station identifiers were one-hot encoded.

To simulate a distributed monitoring scenario, the data were partitioned by stations, so the 12 stations define the 12 federated clients, with each client receiving only the observations from its corresponding station. This partitioning preserves the natural spatial separation of the monitoring network, with approximately 34 thousand records per client.

B. Implementation and Evaluation

The experiments were implemented in Python 3.11.9. The local online regression models were implemented with River

0.22.0 [11], and the federated orchestration was implemented with Flower 1.15.0 [12]. The experiment scripts execute the server and clients as separate processes. All experiments were conducted on a 64-bit Windows 11 Home Single Language system (version 25H2) equipped with an Intel Core i5-8265U processor at 1.60 GHz and 8 GB of RAM (2400 MHz).

The case study evaluates the three server-side aggregation strategies presented in Section III-B. Each strategy was executed 30 times. The local Hoeffding Tree configuration includes the grace period, which defines how many new observations a leaf must accumulate before a new split attempt is considered, the confidence parameter δ used in the Hoeffding bound, and the minimum split merit threshold used to filter weak split proposals. The remaining model and aggregation parameters are summarized in Table I.

TABLE I: Model and aggregation parameters

Parameter	Value
Grace period	50
δ	10^{-5}
Minimum split merit	10^{-5}
Leaf prediction	Mean
α (quorum threshold)	0.6 (60%)

Regression performance was evaluated using MAE and RMSE, which are standard accuracy measures for continuous predictions [13], and the coefficient of determination (R^2), a widely used measure of explained variance in regression [14]. The number of federated rounds and the average execution time per round were also used to compare the coordination cost of each aggregation strategy. To assess the stability of the results across the 30 executions, 95% confidence intervals were computed for the sample means [15].

V. RESULTS AND DISCUSSION

Table II reports the average results over 30 executions for each aggregation strategy. The time column shows the average execution time per federated round, obtained by dividing the total time by the number of rounds. Because these values summarize repeated executions, their stability is discussed later in the section through 95% confidence intervals.

As shown in Table II, majority vote achieved the best performance. Given the $\text{PM}_{2.5}$ mean ($79.79 \mu\text{g}/\text{m}^3$) and standard deviation ($80.82 \mu\text{g}/\text{m}^3$), its MAE and RMSE values of 24.53 and $39.75 \mu\text{g}/\text{m}^3$, respectively, were both below the target standard deviation, while the larger RMSE suggests occasional larger errors. Its R^2 of 0.755 indicates that 75.5% of the target variance was explained. It also gave the best trade-off among accuracy, coordination rounds, and total time.

The Quorum method produced the lowest average time per round, but it also required substantially more rounds and yielded the worst predictive performance. This suggests that its 60% support requirement was too restrictive for station-partitioned $\text{PM}_{2.5}$ streams, where local pollutant behavior may differ across monitoring sites. The Best merit aggregation approach achieved results close to majority vote, but required more rounds and had slightly higher error, indicating that the best individual split proposal is not always the most representative choice for the federation as a whole.

TABLE II: Average results for $\text{PM}_{2.5}$ over 30 runs per aggregation strategy

Strategy	MAE	RMSE	R^2	Rounds	Time/round (s)	Total time (s)
Quorum	39.42	62.56	0.396	348.00	0.71	247.88
Majority vote	24.53	39.75	0.755	126.13	1.23	155.99
Best merit	24.72	39.94	0.752	204.00	1.47	301.47

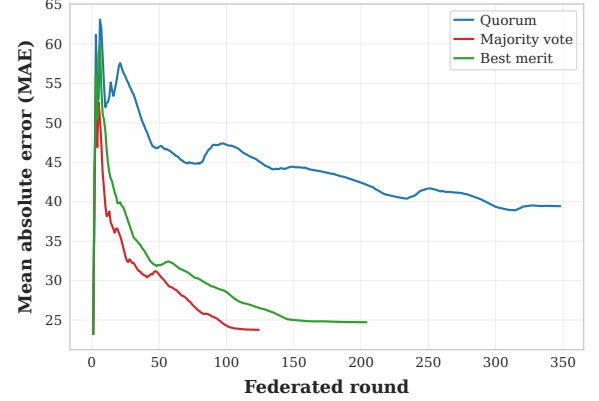


Fig. 2: MAE across federated rounds for the best run of each aggregation strategy.

These results suggest that FHT-R can remain effective even when each client learns only from its own station-level stream. Even under this station-level separation, the coordinated model achieved accurate $\text{PM}_{2.5}$ predictions, indicating that the proposed regression adaptation can capture useful patterns across different stations without requiring all raw observations to be centralized. This overall behavior is most clearly illustrated by the majority-vote strategy, which achieved the highest R^2 among the evaluated approaches.

From the perspective of environmental monitoring, this behavior is relevant because air quality data is continuously generated and affected by local conditions. A federated online regressor can therefore support continuous $\text{PM}_{2.5}$ assessment while preserving the distributed structure of the monitoring network. This does not mean that the proposed method is a complete monitoring system, but it indicates that FHT-R can serve as a promising modeling component for distributed air quality analysis, especially in scenarios where communication cost and monitoring infrastructure are important constraints.

Figure 2 shows the MAE evolution across federated rounds for the best run of each aggregation strategy, selected according to the lowest final MAE among the 30 executions. The Majority vote method reaches lower MAE values in fewer rounds, while quorum evolves more slowly and stabilizes at a substantially higher error level. Best merit exhibits an intermediate behavior, converging more slowly than majority vote and to a slightly higher final MAE. This behavior reinforces the interpretation that majority vote enables more efficient coordinated growth of the regression tree. In practical terms, the model not only reaches better predictive accuracy, but also does so with fewer coordination rounds and the lowest total execution time among the evaluated strategies.

A closer inspection of the Quorum executions helps explain this result. Across the 30 runs, the Quorum strategy required 348 rounds on average, but only 10 produced an aggregated split. Of the other 338 rounds, 336 failed because no feature reached the 60% support threshold, while the last two

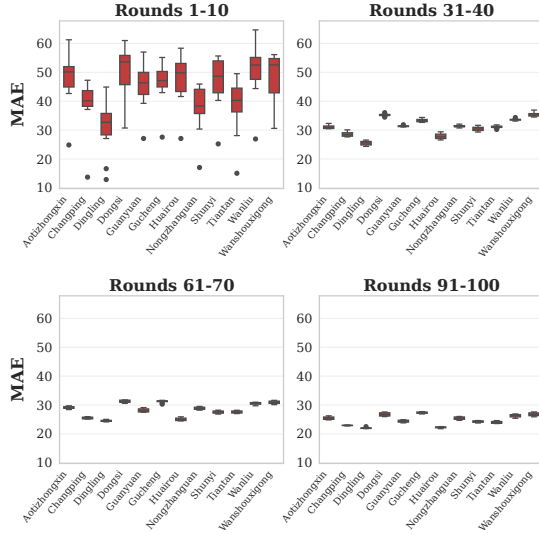


Fig. 3: Client-level MAE distributions for majority vote across selected round intervals.

corresponded to stopping and shutdown after data exhaustion. This indicates that the quorum rule was too restrictive for the $PM_{2.5}$ streams, slowing coordinated tree growth, increasing total execution time, and degrading predictive performance.

For majority vote, the confidence intervals were narrow: [24.3806, 24.6884] for MAE, [39.4780, 40.0340] for RMSE, and [0.7520, 0.7588] for R^2 . This indicates low variability across runs. For quorum and best merit, MAE, RMSE, R^2 , and the number of rounds showed little variability across their respective 30 executions, resulting in confidence intervals with nearly identical bounds for these metrics. However, best merit showed high variability in time per round due to a runtime outlier in one execution.

To further examine client-level behavior over time, Fig. 3 shows the MAE distributions across the 12 stations for the best majority-vote execution. Each panel summarizes a 10-round interval. In the earlier intervals, the client-level MAE values are higher and more dispersed across stations. In the later intervals, the distributions become lower and tighter, indicating that the error decreases over time and becomes more homogeneous across clients. This is an important result because a distributed model should not only perform well on average, but should also avoid leaving individual stations with persistently poor predictions. The reduction and narrowing of the client-level MAE distributions suggest that FHT-R becomes progressively more stable across the monitoring network.

Overall, the results indicate that the proposed regression adaptation is suitable for the evaluated $PM_{2.5}$ prediction task. Among the evaluated aggregation strategies, majority vote was the most effective option, combining the lowest prediction errors, the highest R^2 , fewer coordination rounds, and the lowest total execution time. These findings support the use of coordinated Hoeffding Tree regressors as a viable approach for online regression in distributed air quality prediction scenarios.

VI. CONCLUSIONS

This paper presented a case study on federated online regression for air quality prediction using FHT-R, an adaptation

of the Fed-VFDT coordination protocol to Hoeffding Tree regressors. The evaluation used the full Beijing Multi-Site Air Quality dataset, partitioned by monitoring station to simulate 12 federated clients. Among the evaluated aggregation strategies, majority vote achieved the best predictive performance, with the lowest MAE and RMSE and the highest R^2 , while also requiring fewer rounds than quorum and best merit. These results support the viability of FHT-R for online regression on the evaluated station-partitioned dataset.

As future work, the proposed approach can be extended to the prediction of other pollutants relevant to air quality assessment, such as nitrogen dioxide (NO_2), which is also explicitly addressed in international air quality guidelines and remains important for public health. Another natural direction is to evaluate the method in national monitoring scenarios, including Brazilian environmental contexts such as the Amazon region, where distributed air quality sensing and infrastructure constraints reinforce the relevance of federated approaches.

REFERENCES

- [1] World Health Organization, “Ambient (outdoor) air pollution,” 2022, WHO fact sheet.
- [2] —, “WHO global air quality guidelines: particulate matter (PM_{2.5} and PM₁₀), ozone, nitrogen dioxide, sulfur dioxide and carbon monoxide,” 2021, 22 September 2021. Available from the WHO publication portal.
- [3] S. Zhang, B. Guo, A. Dong, J. He, Z. Xu, and S. X. Chen, “Cautionary tales on air-quality improvement in Beijing,” *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 473, no. 2205, p. 20170457, 2017.
- [4] G. d. A. Lima, I. O. Ribeiro, M. C. O. C. Filho, R. Trevisan, R. V. Andreoli, A. Lacerda, E. G. d. Costa, and R. A. F. d. Souza, “Uma aplicação de internet da natureza para monitoramento de queimadas e da qualidade do ar na amazônia,” in *Anais do XVI Workshop de Computação Aplicada à Gestão do Meio Ambiente e Recursos Naturais (WCAMA 2025)*, 2025, pp. 177–185.
- [5] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, “Advances and open problems in federated learning,” *Foundations and Trends in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.
- [6] G. Hulten, L. Spencer, and P. Domingos, “Mining time-changing data streams,” in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2001, pp. 97–106.
- [7] P. R. Silva, J. Vinagre, and J. Gama, “Fed-VFDT: Federated very fast decision trees with coordinated splitting over data streams,” in *2025 IEEE 37th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2025, pp. 653–660.
- [8] P. Domingos and G. Hulten, “Mining high-speed data streams,” in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2000, pp. 71–80.
- [9] E. Ikononovska, J. Gama, and S. Dzeroski, “Learning model trees from evolving data streams,” *Data Mining and Knowledge Discovery*, vol. 23, no. 1, pp. 128–168, 2011.
- [10] S. Chen, “Beijing multi-site air quality,” 2017.
- [11] J. Montiel, M. Halford, S. M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse, A. Zouitine, H. M. Gomes, J. Read, T. Abdesslem *et al.*, “River: machine learning for streaming data in Python,” *Journal of Machine Learning Research*, vol. 22, no. 110, pp. 1–8, 2021.
- [12] D. J. Beutel, T. Topal, A. Mathur, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, P. P. B. de Gusmão, and N. D. Lane, “Flower: A friendly federated learning research framework,” *arXiv preprint arXiv:2007.14390*, 2020.
- [13] R. J. Hyndman and A. B. Koehler, “Another look at measures of forecast accuracy,” *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, 2006.
- [14] T. O. Kvålseth, “Cautionary note about r^2 ,” *The American Statistician*, vol. 39, no. 4, pp. 279–285, 1985.
- [15] D. C. Montgomery and G. C. Runger, *Applied Statistics and Probability for Engineers*, 7th ed. Wiley, 2020.