# Dynamic Features Adaptation in Networking: Toward Flexible training and Explainable inference

**Yannis Belkhiter**
IBM Research Europe
Trinity College Dublin
yannis.belkhiter@ibm.com

**Seshu Tirupathi**
IBM Research Europe
Dublin, Ireland
seshutir@ie.ibm.com

**Giulio Zizzo**
IBM Research Europe
Dublin, Ireland
giulio.zizzo2@ibm.com

**Merim Dzaferagic**
Trinity College Dublin
Dublin, Ireland
merim.dzaferagic@tcd.ie

**John D. Kelleher**
ADAPT Research Centre
Trinity College Dublin
john.kelleher@tcd.ie

## Abstract

As AI becomes a native component of 6G network control, AI models must adapt to continuously changing conditions, including the introduction of new features and measurements driven by multi-vendor deployments, hardware upgrades, and evolving service requirements. To address this growing need for flexible learning in non-stationary environments, this vision paper highlights Adaptive Random Forests (ARFs) as a reliable solution for dynamic feature adaptation in communication network scenarios. We show that iterative training of ARFs can effectively lead to stable predictions, with accuracy improving over time as more features are added. In addition, we highlight the importance of explainability in AI-driven networks, proposing Drift-Aware Feature Importance (DAFI) as an efficient XAI feature importance (FI) method. DAFI uses a distributional drift detector to signal when to apply computationally intensive FI methods instead of lighter alternatives. Our tests on 3 different datasets indicate that our approach reduces runtime by up to 2 times, while producing more consistent feature importance values. Together, ARFs and DAFI provide a promising framework to build flexible AI methods adapted to 6G network use-cases.

## 1 Introduction and Related Work

Future communication networks are envisioned as AI-native, where artificial intelligence is integrated into all layers of the network. Two complementary perspectives dominate this vision: AI for 6G, where AI algorithms drive the control and optimization of the network, and 6G for AI, where next-generation networks provide the substrate for AI services [1]. In this paper, we focus on AI for 6G. We study how AI can be embedded into heterogeneous base stations and network nodes to identify mobility patterns based on Radio Access Network (RAN) Key Performance Indicators (KPIs).

The deployment of AI in live networks presents distinct challenges. First, AI control is inherently data-driven, requiring access to KPIs and measurements across different nodes. However, due to vendor-specific implementations, hardware generations, and configuration settings, not all network nodes expose the same set of features [2, 3]. This heterogeneity makes it difficult to train and deploy a single global model across the network. Second, even if all features were consistently available, local conditions vary greatly. A base station at a highway may mostly serve vehicular users, while one in a city center may see a mix of pedestrians, buses, and static users. These environmental differences

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: AI and ML for Next-Generation Wireless Communications and Networking (AI4NextG).

imply that local fine-tuning is critical [4]. Third, data drift is inevitable: traffic patterns, user behavior, and radio conditions evolve over time, rendering static models ineffective [5]. Finally, as AI takes on an increasingly central role in controlling critical infrastructure, explainability becomes essential to ensure trust, operator acceptance, and regulatory compliance.

Feature Importance (FI) offers a promising way of addressing these challenges. In general, FI methods refer to techniques that assign scores to input features based on their contribution to the predictions of a model [6, 7]. These algorithms assign higher scores to features that significantly influence predictions, helping the user to identify which variables most strongly affect the output. For both the effectiveness of the training pipeline and interpretability, FI is a crucial piece of information that enables better comprehension of the prediction and optimization of machine learning models [6, 8], echoing challenges seen in the Network space. In this context, our main contributions include:

- **Effectiveness and adaptation of Adaptive Random Forests (ARFs) on network datasets.** We show that applying ARFs to communication datasets with iterative training provides strong predictive performance. Our results confirm that progressively increasing the number of available features leads to higher accuracy, illustrating the value of applying ARFs network environments.
- **DAFI as a promising explainable AI approach for networks.** We present *Drift-Aware Feature Importance* (DAFI), an algorithm that uses distributional drift to decide when to apply costly but accurate explainability methods versus lightweight alternatives. This drift-aware strategy leads to more efficient iterative monitoring and training of ARFs. In doing so, DAFI provides a practical way to couple explainability with model adaptability in network contexts.

## 2 Limitation of current XAI approaches

The current state-of-the-art of FI methods falls into two categories: model agnostic and model specific [7]. Our approach combines model agnostic (SHAP) and model specific (Mean Decrease in Impurity (MDI)) methods (see Appendices A.1 and A.2, respectively). We selected SHAP and MDI as representative baselines because they are widely adopted state-of-the-art methods [6], including in the Networking field [9]. Although incremental XAI methods have obtained promising results in streaming scenarios [10, 11], their adoption is still limited. In the Networking space, SHAP (or variants) has been applied for KPI reduction in O-RAN controllers [12] and traffic classification tasks [13], while MDI is a common method for tree-based models such as Adaptive Random Forests.

**The need for more efficient feature importance predictors:** Although SHAP is the most widely used method for feature importance for its accurate FI predictions [7], it is computationally expensive [6]. In the context of communication networks, this limitation becomes critical: base stations operate continuously and in real-time. It means that FI values must be recomputed at short intervals of time to capture sudden changes from the available features (i.e. KPIs). Relying solely on SHAP can harm the runtime [14]. Appendices A.3 and A.4 further supports this claim, showing the exponential growth of the SHAP runtime when adding new features or increasing the size of the model, respectively.
In comparison to SHAP, MDI is a common tree-based model specific FI metric, and is a much faster method. However, MDI often disagrees with SHAP analysis, especially when the model is not well adapted to the current data as MDI relies on a model's performance [6, 7, 15].

## 3 DAFI: Drift-aware feature importance

Both model agnostic and model specific XAI FI methods come with limitations. Although SHAP analysis offers the best feature importance score based on the complex and evolving data stream, this method is expensive to run. On the other hand, MDI values are quick to compute but are often not aligned with the SHAP values, especially when there is drift in the data distribution and the tree-based model has not yet been updated accordingly.

The central challenge we address in this work is determining when to use accurate but costly SHAP values versus efficient MDI values, without compromising on the reliability of the results. To achieve this, we propose DAFI (see Figure 1), a *Drift-Aware feature importance* algorithm which dynamically selects the appropriate method based on the observed data distribution shifts. Since the performance of the ARF is subject to drifts in the data distribution, our approach employs SHAP values in the presence of drift and defaults to MDI otherwise. Drift is assessed with the *Kolmogorov-Smirnov (KS)*

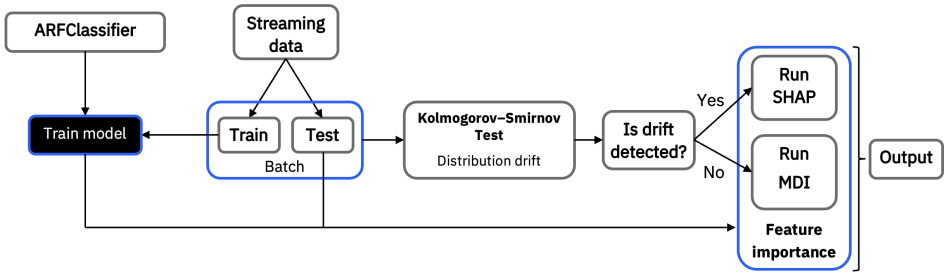


Figure 1: DAFI: Drift-aware Feature Importance algorithm

*Test* [16] (see Appendix A.5). This combination of a drift detection technique with a decision-making framework yields an adaptive, efficient FI approach for dynamic data streams.

**DAFI Algorithm.** We build our DAFI algorithm upon the KS-Test drift detection metric that we presented. For each sample $\mathbf{x}$, DAFI computes the SHAP values if a drift is detected, otherwise it computes the MDI. For each feature, the drift is computed between the previous and current train batches, assuming that we do not have access to the testing data yet. Thus the drift detection can assess if the model has required significant changes in the current training batch. The user needs to set a threshold $\eta$ for the KS-Test. Experimentally, we have set $\eta = 1$ for the Electricity and Weather datasets, and $\eta = 0.125$ for the Network dataset since it has much larger batches, so drifts are less pronounced. Appendix A.6 formally defines our DAFI algorithm.

## 4 Experimental Setup

**Models and datasets.** To test our novel algorithm, we use the *Adaptive Random Forest Classifier* from the river library [17] and 3 binary classification datasets from diverse domains (Networks, Electricity, and Weather). Appendix A.7 describes each dataset. Our main focus is on the Network use-case, so in the main text we only present the results obtained on the *Network* dataset, and include results from other datasets in Appendix A.9.

**Iterative training setup.** For each evaluation, we introduce an iterative training setup. We conducted a prequential split of the datasets in equal batches (80:20 train test - keeping the order). We created 50 batches for the Electricity and Weather datasets, and 40 batches for Network. The reduced number of batches for the Network dataset was chosen for two reasons. First, due to the strong class imbalance in the Network dataset, smaller batches occasionally contained only a single label, limiting the complexity of the stream. Second, using fewer batches allows us to demonstrate that the algorithm remains relatively stable across different batch sizes, as long as both classes are present.

**Dynamic feature adaptation.** To assess the performances of both model and FI methods in a complex and dynamic data stream, we varied the number of features available as training progressed. For each dataset, we started the training with 3 features, added one more feature at epoch 10, and two more features at epoch 20. We ran Feature Importance methods on $n_{\text{samples}}$ samples randomly picked in the testing dataset. For all the experiments, we set our models with $n_{\text{trees}} = 10$, and $n_{\text{samples}} = 50$. All our experiments were carried out on an Apple Macbook Pro M3, 24GB RAM, with a seed of $42$.

**Metrics.** To assess the adaptability of the ARFs on the Network dataset, we report the *training* and *testing accuracies*. To test the efficiency of our DAFI algorithm, we rely on two criteria. First, the *runtime* of feature importance computation is our first way of assessing our algorithm. As SHAP is computationally expensive, one of the objectives of our algorithm is to decrease the runtime. Second, the *feature importance accuracy* of MDI is assessed against SHAP. As SHAP is a reliable and consistent way of computing feature importance, it is used as a ground-truth for our experiment.

**Top-k features** In our complex stream configuration, setting a fixed and reliable $k$ is challenging. Naively, a simple Top-$k$ evaluation selects the $k$ most important features based on their ranked importance scores and compares them between the FI method and SHAP. However, as we are dealing with normalized scores, feature importance scores depend on the number of features present in the dataset. To address this, we introduce a *Dynamic Top-$k$* evaluation, which adapts to variations in feature importance weights. Instead of a fixed Top-$k$ metric, it determines the optimal number of top features to consider by accumulating SHAP feature importance weights in descending order until a predefined cumulative threshold is reached. For our experiment, we fixed this threshold to $0.8$.

Based on the selected Top-$k$ features given by SHAP, we are assessing performance of MDI using *Set* and *Exact matches* analysis. Set matches returns 1 when Top-$k$ of SHAP and MDI sets are same (irrespective of the order or sequence of features), and exact matches returns 1 when Top-$k$ ranks are equal, i.e. feature sequence is also the same. Otherwise, the metrics return 0 (see Appendix A.8).

**Spearman's rank.** To further analyze the effectiveness of our algorithm, we evaluate the correlation between the feature rankings produced by MDI and those generated by SHAP [7]. Specifically, we use *Spearman's rank correlation coefficient*, normalized between 0 and 1. As it is often used in ranking classification contexts, this additional metric helps us to consider more flexibly the differences in the entire ranking of feature importance.

## 5 Evaluation

**Performance of ARFs on Evolving Network Features.** Figure 2 illustrates the performance of ARFs on the Network dataset under evolving feature sets, where new KPIs are progressively introduced at epoch 10 and 20 (1 and 2 features added, respectively). We observe that the addition of new features does not harm, but increases the model's performance. Specifically, the introduction of two new features at epoch 20 improves the accuracy of the ARF from approximately $0.64$ to $0.72$. The results highlight the stability of the ARF model dynamic feature spaces, such as those that may arise due to differences in vendor-specific implementations, or configuration settings in communication networks.

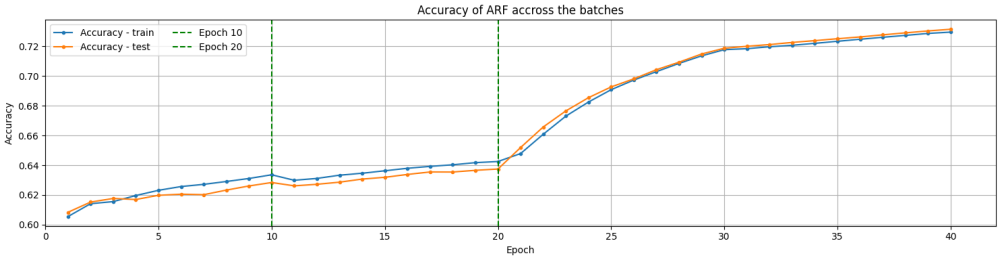


Figure 2: Performance of ARF for benign and attack network classifications with evolving Network KPIs - Network dataset [18] - epoch 0: start with features "mac_dl_mcs", "phy_ul_pusch_sinr", "phy_ul_pucch_sinr" - epoch 10: adding "phy_ul_pusch_rssi" - epoch 20: adding "mac_dl_cqi", "phy_ul_pucch_rssi" - $n_{\text{model}} = 10$

**Drif-Aware Feature Importance (DAFI) for Efficient Explainability.** We present the results of our DAFI algorithm, where accuracy metrics are computed element-wise, and then averaged over the $n = 50$ samples. Table 1 presents for each FI method, on the Network dataset, the runtime along with the discussed FI metrics. Appendix A.9 reports the results on the Electricity and Weather datasets.

| Datasets | FI Method | Runtime (s) | Saved (%) | Top-k set | Top-k exact | Top-k Spearman |
|---|---|---|---|---|---|---|
| Network | SHAP | 1488.32 | 0.00 | 1.00 | 1.00 | 1.00 |
| | MDI | 9.37 | 99.37 | 0.27 | 0.09 | 0.53 |
| | **DAFI** | 674.49 | 54.68 | 0.55 | 0.40 | 0.67 |

Table 1: Runtime and metrics of FI methods applied Element-wise - average over the batches

Regarding the Runtime, SHAP consistently has the highest runtime across each dataset. In contrast, MDI is orders of magnitude faster compared to SHAP, with a consistent saved runtime score of more than 99%. Drift-Aware achieves a good balance between SHAP and MDI, substantially reducing the computational cost. The saved runtime obtained by our algorithm is notably high, reaching 55% for the Network dataset and around 35% for the Electricity and Weather datasets.

However, MDI obtained a relatively low Top-$k$ exact match score on average, confirming our observations from the literature. The DAFI algorithm outperformed MDI for all metrics, demonstrating substantial improvements in Top-$k$ exact match (up to 0.76 in Electricity) and Spearman Top-$k$ match (0.85 in Electricity and 0.83 in Weather) - where MDI reported low scores. This suggests that our algorithm improved the consistency of FI prediction, effectively switching between SHAP and MDI.

## 6 Conclusion

We demonstrated that ARFs, when trained iteratively, can effectively capture the dynamics of communication data streams, with accuracy improving as more features are introduced. To complement this, we proposed DAFI, a drift-aware feature importance method that reduces computational overhead while guiding the selection of the most relevant features. Together, ARFs and DAFI highlight a promising direction for AI-native 6G networks: models that adapt to evolving feature spaces, remain resilient to distributional shifts, and provide transparent insights into their decisions.

**Future work.** While these results highlight the method's potential as a practical alternative to computationally expansive approaches, future research should focus on further validating this method across additional datasets and adaptive scenarios (such as feature removal), as well as exploring its integration with other types of ensemble models, and other FI methods.

## Acknowledgment

This work has been partially supported by the 6G-XCEL project (grant agreement 101139194), funded by the EU Horizon Europe program.

This research was partly supported by the ADAPT Research Centre. The ADAPT Centre for Digital Content Technology is funded under the Research Ireland's Research Centres Programme (Grant 13/RC/2106 II) and is co-funded under the European Regional Development Funds.

## References

[1] Merim Dzaferagic, Marco Ruffini, Nina Slamnik-Krijestorac, Joao F Santos, Johann Marquez-Barja, Christos Tranoris, Spyros Denazis, Thomas Kyriakakis, Panagiotis Karafotis, Luiz DaSilva, et al. Decentralized multi-party multi-network ai for global deployment of 6g wireless systems. *arXiv preprint arXiv:2407.01544*, 2024.

[2] Mohamad Mestoukirdi and Mourad Khanfouci. Reliable vertical federated learning in 5g core network architecture. *arXiv preprint arXiv:2505.15244*, 2025.

[3] K Trichias, M Christopoulou, P Porambage, C Benzaid, V Vassiliou, D Artuñedo Guillén, D Tsolkas, Marios Avgeris, C Papagianni, A Jain, et al. Ai/ml as a key enabler of 6g networks: Methodology, approach and ai-mechanisms in sns ju. 2025.

[4] Samsung Research. Ai-native and sustainable communication: Samsung 6g white paper. Technical report, Samsung Electronics, June 2025. Accessed: 2025-08-27.

[5] Qualcomm Technologies, Inc. How will adaptive intelligence in 6g transform wireless connectivity: 6g foundry white paper. Technical report, Qualcomm, 2024. Accessed: 2025-08-27.

[6] Christoph Molnar. *Interpretable Machine Learning*. 3 edition, 2025.

[7] Gopi Krishnan Rajbahadur, Shaowei Wang, Gustavo A. Oliva, Yasutaka Kamei, and Ahmed E. Hassan. The impact of feature importance methods on the interpretation of defect classifiers. *IEEE Transactions on Software Engineering*, 48(7):2245–2261, July 2022.

[8] Erwan Scornet. Trees, forests, and impurity-based variable importance, 2021.

[9] Bouziane Brik, Hatim Chergui, Lanfranco Zanzi, Francesco Devoti, Adlen Ksentini, Muhammad Shuaib Siddiqui, Xavier Costa-Pérez, and Christos Verikoukis. Explainable ai in 6g o-ran: A tutorial and survey on architecture, use cases, challenges, and future research, 2024.

[10] Fabian Fumagalli, Maximilian Muschalik, Eyke Hüllermeier, and Barbara Hammer. Incremental permutation feature importance (ipfi): towards online explanations on data streams. *Machine Learning*, 112(12):4863–4903, September 2023.

[11] Maximilian Muschalik, Fabian Fumagalli, Barbara Hammer, and Eyke Hüllermeier. Agnostic Explanation of Model Change based on Feature Importance. *KI - Künstliche Intelligenz*, 36(3):211–224, December 2022.

[12] Chinenye Tassie, Brian Kim, Joshua Groen, Mauro Belgiovine, and Kaushik R. Chowdhury. Leveraging explainable ai for reducing queries of performance indicators in open ran. In *ICC 2024 - IEEE International Conference on Communications*, pages 5413–5418, 2024.

[13] Alfredo Nascita, Francesco Cerasuolo, Giuseppe Aceto, Domenico Ciuonzo, Valerio Persico, and Antonio Pescapé. Explainable mobile traffic classification: the case of incremental learning. In *Proceedings of the 2023 on Explainable and Safety Bounded, Fidelitous, Machine Learning for Networking*, SAFE '23, page 25–31, New York, NY, USA, 2023. Association for Computing Machinery.

[14] Osman Tugay Basaran and Falko Dressler. Xainomaly: Explainable and interpretable deep contractive autoencoder for o-ran traffic anomaly detection, 2025.

[15] Mirka Saarela and Susanne Jauhiainen. Comparison of feature importance measures as explanations for classification models. *SN Applied Sciences*, 3(2):272, 2021.

[16] Frank J. Massey. The kolmogorov-smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46(253):68–78, 1951.

[17] Jacob Montiel, Max Halford, Saulo Martiello Mastelini, Geoffrey Bolmier, Raphael Sourty, Robin Vaysse, Adil Zouitine, Heitor Murilo Gomes, Jesse Read, Talel Abdessalem, and Albert Bifet. River: machine learning for streaming data in python, 2020.

[18] Bruno Xavier, Merim Dzaferagic, Marco Ruffini, and Magnos Martinello. RAN Performance Measurements for Security Threats, 2024.

[19] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions, 2017.

[20] Lauri and Wu. Will it rain tomorrow? eda and classification, 2020. Accessed: 2025-08-15.

[21] Heitor M. Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfharinger, Geoff Holmes, and Talel Abdessalem. Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9):1469–1495, October 2017.

[22] M. Harries. Splice-2 comparative evaluation: Electricity pricing. Technical report, The University of South Wales, 1999.

[23] João Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *SBIA Brazilian Symposium on Artificial Intelligence*, pages 286–295, 2004.

[24] S Raksha, Jasmine S Graceline, Jani Anbarasi, M Prasanna, and S Kamaleshkumar. Weather forecasting framework for time series data using intelligent learning models. In *2021 5th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT)*, pages 783–787, 2021.

[25] Bruno Missi Xavier, Merim Dzaferagic, Magnos Martinello, and Marco Ruffini. Performance measurement dataset for open ran with user mobility and security threats. *Computer Networks*, 253:110710, 2024.

# A Appendix

## A.1 SHAP: Definition and Limitations

*SHapley Additive exPlanations* (SHAP) is a model agnostic framework for interpreting model predictions by assigning feature importance values based on cooperative game theory [19]. For a given data point $\mathbf{x} = \{x_1, \ldots, x_j, \ldots, x_d\}$ and a model $f$, SHAP computes the contribution of each feature $x_j \in \mathbf{x}$ to the prediction $f(\mathbf{x})$ by considering all possible feature combinations. The SHAP value $\phi_j$ for feature $j$ is defined as:

$$\phi_j = \sum_{S \subseteq \{1,\ldots,d\} \setminus \{j\}} \frac{|S|!(d - |S| - 1)!}{d!} \left[ f(S \cup \{j\}) - f(S) \right] \tag{1}$$

where $d$ is the number of features, $S$ is a subset of features, $f(S)$ represents the model output with features in $S$ included, and $\phi_j$ is the SHAP value for feature $j$. SHAP not only indicates feature importance score, but also provides directional information - positive values increase the prediction, while negative values decrease it. However, since our focus is on overall feature contribution rather than direction, we take the absolute values of the SHAP scores, and normalize them across all features of the data. This enables score comparison with the Mean Decrease in Impurity method, that consists of positive, normalized values.

## A.2 Mean Decrease in Impurity: A model specific approach

Path-based FI methods are model specific techniques adapted to tree-based architectures, such as Adaptive Random Forests. These algorithms offer a lightweight yet effective approach for analyzing feature importance. Unlike model agnostic methods, which analyze aggregated statistics across all subsets of features, path-based methods compute feature importance dynamically by traversing the decision paths for individual predictions [8]. This approach is particularly useful for adaptive scenarios, where data distribution and model parameters evolve over time. For every feature node encountered along the path, the algorithm assigns an importance weight based on the *Mean Decrease in Impurity* (MDI), a measure derived from the change in node impurity.

Formalized by [8], the algorithm begins by traversing the decision paths for a single instance $\mathbf{x} = \{x_1, \ldots, x_j, \ldots, x_d\}$ within each tree of an ensemble model. Let $\mathcal{T} = \{T_1, T_2, \ldots, T_m\}$ represent the set of $m$ trees in the ensemble model, and $T_i$ denote the $i$-th tree. For each tree $T_i$, we traverse its decision path for $\mathbf{x}$. For every decision node $n_j$ in the path, where the feature $j$ is the splitting criteria, the algorithm evaluates the contribution of the feature $j$ to the final feature importance $\varphi_j$:

$$\varphi_j = \sum_{T_i \in \mathcal{T}} \sum_{n_j \in T_i} \text{MDI}(n_j, T_i) \tag{2}$$

$$\text{where MDI}(n_j, T_i) = \text{G}_{n_j} - \frac{\text{freq}_{\text{left}}}{\text{freq}_{n_j}} \cdot \text{G}_{\text{left}} - \frac{\text{freq}_{\text{right}}}{\text{freq}_{n_j}} \cdot \text{G}_{\text{right}}$$

$$\text{with G}_n = 1 - \sum_{k=1}^{C} p_k^2 \quad \text{and} \quad p_k = \frac{\text{freq}_k}{\text{freq}_n}$$

$\varphi_j$ is the feature importance for feature $j$, $\text{MDI}(n_j, T_i)$ is the Mean Decrease in Impurity for the node $n_j$ in the tree $T_i$, $\text{G}_n$ is the Gini index of the node $n$, $\text{G}_{\text{left}}$ and $\text{G}_{\text{right}}$ are respectively the Gini indexes of the left and right child of $n_j$, $p_k$ is the proportion of samples in class $k$ in node $n$, $\text{freq}_k$ is the number of samples in class $k$, and $\text{freq}_n$ is the total number of samples in node $n$. After computation, the importance for a feature $\varphi_j$ across all nodes and trees is normalized, and features are ranked based on their normalized importance values. The algorithm is presented in appendix (Section **??**).

While the formulation of MDI above applies to binary decision trees, it can be extended to non-binary trees by generalizing the impurity reduction term. Instead of considering only left and right children, the decrease in impurity is computed by subtracting the weighted impurities of all child nodes from the impurity of the parent node. This allows the MDI measure to be applicable to trees with any type of branching structure as defined by [8].

### A.3 Limitation of SHAP

This sub-section offers evidence that relying solely on SHAP can significantly impact the runtime.

| # of features | Training Runtime (s) | SHAP Runtime (s) | Training Accuracy | Testing Accuracy |
|---|---|---|---|---|
| **All (8)** | 1074.55 | 2982 | 89.51 | 82.14 |
| **Top 7 feat. (7)** | 1084.25 | 1414 | 89.70 | 81.40 |
| **Top 4 feat. (4)** | 744.74 | 148.3 | 86.34 | 79.31 |
| **Top 3 feat. (3)** | 682.88 | 61.04 | 86.04 | 77.22 |
| **Top 2 feat. (2)** | 647.20 | 20.9 | 84.03 | 76.28 |

Table 2: Comparison of ARF Performance on the Electricity dataset [20] with different set of features - 80:20 train/test prequential split - SHAP runtime (including setting the explainer and inference time) for $n_{\text{samples}} = 1000$ - $n_{\text{trees}} = 100$

| # of features | Training Runtime (s) | SHAP Runtime (s) | Training Accuracy | Testing Accuracy |
|---|---|---|---|---|
| **All (6)** | 1550.4 | 64.45 | 77.14 | 78.32 |
| **Top 5 feat. (5)** | 3213.2 | 36.34 | 66.89 | 67.90 |
| **Top 4 feat. (4)** | 2720.1 | 17.47 | 65.40 | 66.10 |
| **Top 3 feat. (3)** | 6244.4 | 7.77 | 62.78 | 63.72 |
| **Top 2 feat. (2)** | 4912.5 | 2.41 | 59.86 | 60.70 |

Table 3: Comparison of ARF Performance on the Network dataset [18] with different set of features - 80:20 train/test prequential split - SHAP runtime (including setting the explainer and inference time) for $n_{\text{samples}} = 100$ - $n_{\text{trees}} = 10$

Using a simple online training setting, Table 2 and 3 report a SHAP runtime growing exponentially as the number of features increases - reaching more than 49 minutes for only 1,000 samples with 8 features for the *Electricity* dataset. Moreover, we observe that incorporating more features can improve accuracy for both datasets, at the cost of the SHAP runtime.

These trends are further supported by Table 4 and Figure 3 in Appendix A.4, which illustrates the exponential impact of model size on SHAP runtime. This limitation motivates the need for lightweight, scalable methods for estimating feature importance in ARF models - enabling the identification of the most influential features without compromising runtime or predictive performance.

In comparison, MDI is significantly less computationally expensive than SHAP as it only relies on the prediction paths within the model and so avoids SHAP's exhaustive search of all feature combinations[1]. However, MDI is sensitive to the performance of the model and underperforms during learning phases [6, 7]. Consequently, neither SHAP or MDI in their standard configuration are ideal for streaming data scenarios.

[1]Approximate versions of SHAP have been proposed that avoid the complete search of feature combinations. However, theses methods are not applicable in dynamic streaming scenarios where the feature set may vary through time.

### A.4 Impact of the number of models on SHAP Runtime

Using the SEA synthetic dataset[2], Table 4 compares runtime and the Top-k exact match accuracy of element-wise feature importance of the MDI compared to SHAP. Figure 3 illustrates the results shown in the Table.

| # Trees | Training Runtime | Training Accuracy | Testing Accuracy | SHAP Runtime | MDI Runtime | Top-k exact |
|---|---|---|---|---|---|---|
| 1 | 0.1794 | 0.89 | 0.95 | 4.7189 | 0.0431 | 0.80 |
| 10 | 1.4716 | 0.95 | 0.98 | 23.6637 | 0.9536 | 0.60 |
| 50 | 7.5501 | 0.96 | 0.98 | 108.6549 | 2.0682 | 0.68 |
| 100 | 14.8777 | 0.96 | 0.99 | 216.1023 | 4.3310 | 0.77 |
| 500 | 79.0376 | 0.96 | 0.99 | 1129.4253 | 22.0117 | 0.74 |

Table 4: Influence of number of trees in the forest on Model and Feature importance algorithms Performance - Synthetic dataset with 80:20 train/test split - Metrics of algorithms are given for $n_{\text{samples}} = 100$ samples from the test dataset

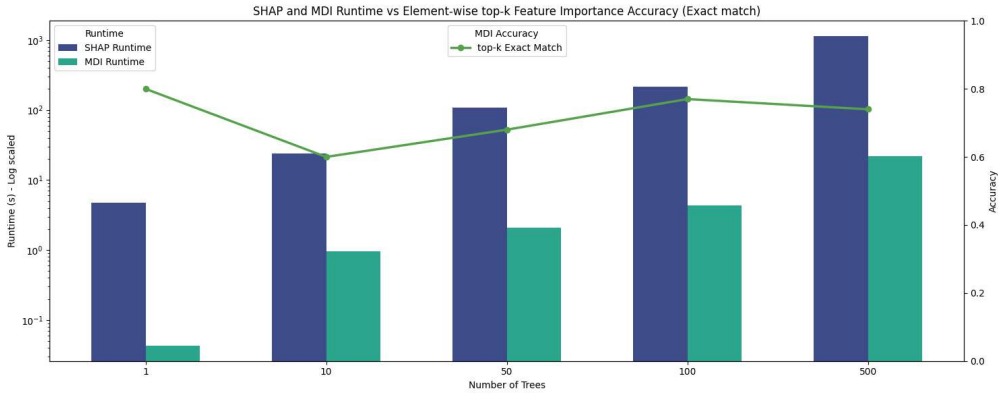


Figure 3: Influence of number of trees in the forest on Model and Feature importance algorithms Performance - Synthetic dataset with 80:20 train/test split - Metrics of algorithms are given for $n_{\text{samples}} = 100$ samples from the test dataset

[2] `https://riverml.xyz/0.14.0/api/datasets/synth/SEA/`

### A.5 KS-TEST

**Kolmogorov-Smirnov Test**. The Kolmogorov-Smirnov test [16] is a metric used to quantify the distance between two data distributions. Given two batches of streaming data, each containing $n$ samples and $d$ features, the test computes the maximum difference between their empirical cumulative distribution functions. Specifically, for two batches of streaming data, $X_{t-1} = \{x_1^{t-1}, x_2^{t-1}, \ldots, x_d^{t-1}\}$ and $X_t = \{x_1^t, x_2^t, \ldots, x_d^t\}$, the KS statistic is defined as:

$$D_i = \sup_{x_i} \left| F_t(x_i^t) - F_{t-1}(x_i^{t-1}) \right| \tag{3}$$

where $F_{t-1}(x_i)$ and $F_t(x_i)$ are respectively the empirical cumulative distribution functions of the previous and current batches for the feature $i$. This statistic quantifies the largest deviation between the distributions of the two batches. A higher value of $D_i$ suggests a more significant shift in the feature's distribution, indicating potential drift. While other methods like ADWIN [**?** ] are tailored for continuous streaming scenarios with adaptive windowing, the KS-Test is better suited for our batch-wise sequential training setup (see Section 4).

### A.6 DAFI algorithm

**Mathematical formalization.** Recalling expression of SHAP, MDI and KS-Test (Equations 1, 2 and 3), we can mathematically formalize the DAFI algorithm. For any sample $\mathbf{x} = \{x_1, \ldots, x_d\}$, we express $\Phi_j$, the feature importance of $x_j$ computed by DAFI:

$$\Phi_j = \mathbf{1}_{\text{drift}}(\mathbf{x}) \cdot \frac{|\phi_j|}{\sum_{i \in \{1,\ldots,d\}} |\phi_i|} + (1 - \mathbf{1}_{\text{drift}}(\mathbf{x})) \cdot \frac{\varphi_j}{\sum_{i \in \{1,\ldots,d\}} \varphi_i} \tag{4}$$

$$\text{with } 1_{\text{drift}}(\mathbf{x}) = \begin{cases} 1 & \text{if } \left(\sum_{i \in \{1,\ldots,d\}} D_i\right) > \eta \\ 0 & \text{otherwise} \end{cases}$$

where $\Phi_j$, $\phi_j$, and $\varphi_j$ are respectively the FI of the feature j in $\mathbf{x}$ computed by DAFI, SHAP and MDI algorithms. $1_{\text{drift}}$ represents the indicator function of the KS-Test drift detector, which returns 1 if the distributional shift between the current and previous batches exceeds a predefined threshold $\eta$, and 0 otherwise.

**Algorithm.** Building on the preceding definitions, we lists the DAFI procedure below (Algorithm 1)

**Algorithm 1** DAFI - Drift-aware Feature Importance algorithm

**Require:** ARF model $\mathcal{T} = \{T_1, T_2, \ldots, T_m\}$, $\mathbf{x} = \{x_1, \ldots, x_n\}$, Drift_threshold $\eta$

```
Detect drift and warning/drift from ARF:
Drift_Detected ← False
Drift_Accum ← 0                    ▷ Set a variable to accumulate the drifts observed for each features
for each feature x_i ∈ x do
    Drift_Accum ← KS_STATS(batch_train[i-1], batch_train[i])[x_i]
    if Drift_Accum > η then
        Drift_Detected ← True
        break                      ▷ Stop if drift is detected for any feature
if Drift_Detected then
    importance ← SHAP(x)
else
    importance ← MDI(x)
return Normalized Importance for all x_i ∈ §
```

### A.7 Dataset description

Table 5 describes each dataset.

| Datasets | Ref. | Domain | # of Samples | # of Features | Target | # of Class |
|---|---|---|---|---|---|---|
| Electricity | [21] | Energy | 45,312 | 8 | Class | 2 |
| Weather | [20] | Climate | 145,460 | 23 | RainTomorrow | 2 |
| Network | [18] | 6G | 3,175,140 | 46 | mob_pattern | 2 |

Table 5: Information about selected datasets

The Electricity dataset contains electricity price, ask and demand of the Australian New South Wales Electricity Market sampled at 5 minute intervals. The dataset was introduced by [22], refined by [23], and later used by [21], to predict whether the moving average of the price of the last 24 hours goes UP or DOWN. Features selected for this dataset are: "period", "transfer", "nswprice", "vicprice", "vicdemand", and "nswdemand". Electricity markets are inherently dynamic, influenced by fluctuation in supply, demand, and external events as weather or policy changes. As a result, models trained on historical electricity data must be adaptable to changing patterns and shifts in market distribution.

The Weather dataset includes daily observations of various Australian meteorological stations. Used by [24], and available from [20], it contains various numerical and categorical data from sensors, such as temperature, humidity, wind speed, pressure, and rainfall. Features selected for this dataset are: "MinTemp", "WindGustSpeed", "Evaporation", "MaxTemp", "Rainfall", and "Sunshine". The objective of this dataset is to predict whether it will rain or not the following day. Weather prediction is useful for a wide range of applications, including agriculture, disaster preparedness, transportation, and outdoor event planning. However, weather data is subject to seasonal variations, sensor malfunction, and evolving patterns, making it prone to data distribution shifts over time. Consequently, drift detection techniques can help maintain predictive accuracy by adapting models to changing environmental conditions.

The Network dataset consists of Radio Access Network (RAN) Key Performance Indicator (KPI) and Performance Monitoring (PM) data collected from the OpenIreland testbed ( [25], [18]). The dataset spans various traffic classes, categorized as either "Benign" (e.g., web browsing, VoIP, IoT, and YouTube video streaming) or "Attack" (e.g., DDoS Ripper, DoS Hulk, PortScan, and Slowloris). In a real-world deployment, mobile networks often consist of a mix of multivendor equipment, where certain measurements may be available in some network nodes but missing in others. Additionally, legacy hardware may only support a subset of parameters, limiting direct comparisons across different network elements. As a result, models trained on such data must be robust to evolving parameters and changing feature availability. This variability underscores the need for drift detection techniques to ensure models remain accurate over time despite shifts in feature distributions. Furthermore, explainability in AI is critical for diagnosing performance fluctuations and understanding why certain classifications are made, particularly when key network metrics are missing or when unexpected behavior arises due to hardware or configuration differences. The data reflects mobility patterns, including static, pedestrian, car, bus, and train scenarios. The dataset features detailed measurements derived from the MAC and PHY layers of the network. At the PHY layer, metrics such as Channel Quality Indicator (CQI), Modulation and Coding Scheme (MCS), and Signal-to-Interference-plus-Noise Ratio (SINR) for both PUSCH and PUCCH channels are included. These metrics quantify channel quality, signal interference, and modulation efficiency. The MAC layer data focuses on traffic behavior and service interactions, including metrics such as data rate, successfully delivered packets, dropped packets, and buffer status. The dataset is useful for analyzing real-world mobile communication environments and classifying RAN mobility types while remaining independent of specific user equipment behavior. Additionally, with both benign and attack traffic classes, it supports the development and validation of intrusion detection systems to enhance network security strategies. Features selected for this dataset are: "mac_dl_mcs", "phy_ul_pusch_sinr", "phy_ul_pucch_sinr", "phy_ul_pusch_rssi", "mac_dl_cqi", and "phy_ul_pucch_rssi".

### A.8 Illustration Top-k set and match metrics

Figure 4 illustrates our Top-$k$ feature importance metrics.

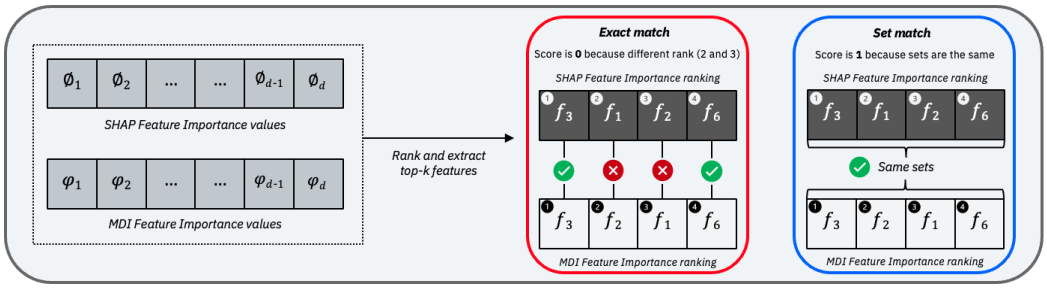


Figure 4: Illustration of the top-k metrics - Exact and Set matches

### A.9 Results on other datasets

Table 6 presents the results of our DAFI algorithm on the Electricity and Weather datasets.

| Datasets | FI Method | Runtime (s) | Saved runtime (%) | Top-k set | Top-k exact | Top-k Spearman |
|---|---|---|---|---|---|---|
| Electricity | SHAP | 1790.79 | 0.00 | 1.00 | 1.00 | 1.00 |
| | MDI | 6.09 | 99.65 | 0.56 | 0.39 | 0.67 |
| | **DAFI** | 1161.60 | 35.13 | 0.85 | 0.76 | 0.85 |
| Weather | SHAP | 1544.23 | 0.00 | 1.00 | 1.00 | 1.00 |
| | MDI | 7.44 | 99.52 | 0.50 | 0.20 | 0.61 |
| | **DAFI** | 1023.09 | 33.75 | 0.82 | 0.66 | 0.83 |

Table 6: Runtime and metrics of FI methods applied Element-wise