Explainable Artificial Intelligence in Communication Networks: A Use Case for Failure Identification in Microwave Networks

Omran Ayoub[a], Nicola Di Cicco[b], Fatima Ezzeddine[a], Federica Bruschetta[c], Roberto Rubino[c], Massimo Nardecchia[c], Michele Milano[c], Francesco Musumeci[b], Claudio Passera[c], Massimo Tornatore[b]

[a]*Department of Innovative Technologies, University of Applied Sciences and Arts of Southern Switzerland, Lugano, Switzerland*
[b]*Department of Electronics, Information and Bioengineering, Politecnico di Milano, Milan, Italy*
[c]*SIAE Microeletronica, Milan, Italy*

Abstract

Artificial Intelligence (AI) has demonstrated superhuman capabilities in solving a significant number of tasks, leading to widespread industrial adoption. For in-field network-management application, AI-based solutions, however, have often risen skepticism among practitioners as their internal reasoning is not exposed and their decisions cannot be easily explained, preventing humans from trusting and even understanding them. To address this shortcoming, a new area in AI, called Explainable AI (XAI), is attracting the attention of both academic and industrial researchers. XAI is concerned with explaining and interpreting the internal reasoning and the outcome of AI-based models to achieve more trustable and practical deployment. In this work, we investigate the application of XAI for network management, focusing on the problem of automated failure-cause identification in microwave networks. We first introduce the concept of XAI, highlighting its advantages in the context of network management, and we discuss in detail the concept behind Shapley Additive Explanations (SHAP), the XAI framework considered in our analysis. Then, we propose a framework for a XAI-assisted ML-based automated failure-cause identification in microwave networks, spanning model's development and deployment phases. For the development phase, we show how to exploit SHAP for feature selection and how to leverage SHAP to inspect misclassified instances during model's development process, and how to describe model's global behavior based on SHAP's global explanations. For the deployment phase, we propose a framework based on predictions uncertainty to detect possibly wrong predictions that will be inspected through XAI.

Keywords: Explainable Artificial Intelligence, Machine Learning, Automated Network Management
PACS: 0000, 1111
2000 MSC: 0000, 1111

1. Introduction

With the increasing availability of monitoring data and the recent advances in computing platforms, Artificial Intelligence (AI) and Machine Learning (ML) are becoming key tools for network operators to automate network management. AI-based solutions have already demonstrated superhuman capabilities in solving a wide range of real-world networking problems, and addressing, among others, challenging failure management problems as failure detection, failure-cause identification, failure prediction and localization, leading to widespread industrial adoption. However, AI-based solutions have often risen skepticism among practitioners as they are mostly used as "black boxes". As a matter of fact, their internal reasoning is not exposed, thus preventing humans from trusting and even understanding them.

To address this shortcoming, efforts are being made in the field of eXplainable Artificial Intelligence (XAI) to explain decisions of a ML model with the aim of transforming the black box into a "transparent" box [1], [2], [3], [4], [5]. XAI is concerned with explaining the motivations behind AI-based models in automated decision making. By explaining AI-based models, practitioners can understand how the AI arrived to the decision, i.e., what are the reasons behind AI decisions, and hence, know whether decisions are based on correct or wrong reasoning, which can provide crucial insights into how a model may be improved. For instance, XAI can shed light on how model features are used as driving factors towards decisions, thus relating input features to model's output, and allowing humans to have an improved understanding of a model's behavior.

Hence, XAI is key to increase trust in automated network management. It can be exploited either prior to the deployment of the ML model, i.e., while designing the ML model (*development phase*) or after the deployment of the ML model (*deployment phase*).

- In *development phase*, XAI can be used to explain a model's global behavior (how features, based on their values and interaction with other features, impact model's decisions) or at group level, e.g., focusing on

model behavior with respect to a particular label in a classification problem. Leveraging these explanations, domain experts can debug model's reasoning and either verify or negate model's behavior, or, in other words, can answer the question, *are model's decisions in the classification problem based on correct reasoning?*

- In *deployment phase*, XAI can be used to explain, in real-time, specific decisions, relating to a specific instance, taken by the AI model.

In this work, we focus on the application of XAI for automated failure-cause identification in microwave networks. We model the problem as a supervised multi-class classification [6] and we encompasses the application of XAI in both *development* phase and *deployment* phase. As a specific XAI framework, we selected Shapley Additive Explanations [7] (SHAP will be discussed in detail in Sec. 3). For the *development* phase, we first show that SHAP can be leveraged to perform feature selection, consequently reducing number of features (and hence limiting computational time). Then, we apply SHAP to extract explanations and explain model's behavior. We show how these explanations help in debugging model's behavior and in extracting insights about the problem at hand. For the *deployment* phase, we use SHAP to explain individual decisions (these explanations are referred to as local explanations), which can be exploited by a network operator to inspect decisions in real time, before initiating actions based on such decisions. Let us consider, for instance, the case of automated failure-cause identification. Based on ML model decision, network operators can timely take the most appropriate countermeasure to repair a failure, thus reducing service unavailability compared to time-consuming analysis of failure logs performed by human domain experts. However, initiating a repair action based on a wrong failure-cause identification can lead to significant and unnecessary costs for the operator. In such a scenario, XAI can help reveal to the network operator the driving forces behind a decision, an information that can be used to asses whether to trust a model's decision or not, and hence, whether to initiate recovery procedures based on that decision.

Considering such capabilities provided by XAI, one could be tempted to extensively use XAI to explain several (if not all) decisions of the deployed ML models. However, this would contradict the objective of automated decision-making, as explaining decisions requires a domain expert that interprets and validates the explanations. Motivated by this, we propose a novel framework based on model's uncertainty calculations to filter out possible misclassifications and propose a numerical and systematic approach to highlight decisions where a deeper inspection based on XAI framework might be beneficial.

We summarize the contribution of this paper as follows:

- We introduce the concept of XAI, highlighting its benefits in the context of network management.
- We discuss the application of SHAP, a XAI framework, in a ML model's development process and during model's deployment.
- We show how to perform a SHAP-assisted feature selection. Results show that feature selection based on SHAP eliminated more than 60% of the features while improving model's performance.
- We show how to leverage SHAP to inspect misclassified instances during model's development process, and how to describe model's global behavior based on SHAP's global explanations.
- For the deployment phase, we propose a framework based on predictions uncertainty to detect possibly wrong predictions that will be inspected through XAI. Results show that using uncertainty thresholds allows to selectively investigate with XAI only a small fraction of predictions, providing precious insight on how to continuously improve the model's performance.

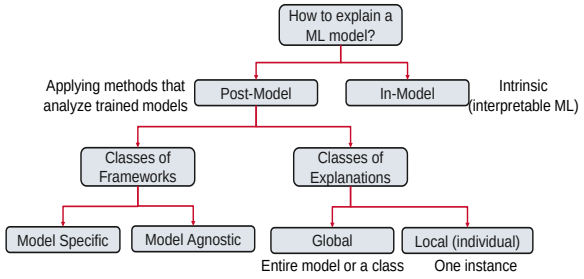


Figure 1: Overview of classes of frameworks and explanations.

The rest of the paper is organized as follows. Section 2 provides an overview on XAI, highlighting the benefits of XAI for communication networks, and discusses related work. Section 3 provides an overview of SHAP, the XAI framework adopted in this work. Section 4 describes the problem of failure-cause identification in microwave networks. Section 5 discusses numerical results of automated failure-cause identification, while Sec. 6 discusses our application of XAI and presents numerical results. Finally, Sec. 7 concludes the paper.

2. Background on Explainable Artificial Intelligence

This section provides an overview on XAI and then discusses related work on the application of XAI for network management.

2.1. Overview on XAI

Explainability vs. Interpretability. *Explainability* and *interpretability* are two popular terms in the context of XAI. Interpretability of an AI model has been defined

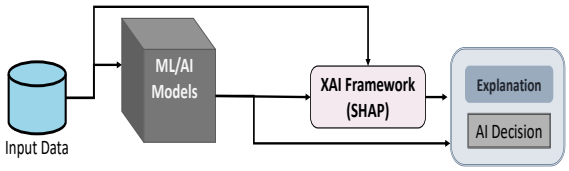


Figure 2: Post-hoc explanation.

qualitatively as the degree to which a human can understand the cause of a decision of that model [8], [9], [10]. Interpretability is inherently based on the model, as some models, such as rule-based models, are interpretable by-design and can be easily understood, while other models, such as artificial neural networks, cannot be interpreted.

In contrast, explainability is associated with revealing the internal logic of the model with the aim of justifying a particular decision. The need for explainability in AI originates from the fact that ML models, in their majority, are complex, and their internal reasoning cannot be explained [11] [4]. XAI aims to demonstrate how ML models reason, showing how a model arrived to its decision, or, in other words, which features are the driving forces towards a decision. Note that some XAI frameworks use interpretable ML models that provide information of their decision process *by design*, which can then be transformed into explanations.

Why Explain?: This question is strictly related to the stakeholders and the use case in which the ML model and XAI framework are applied. For instance, in engineering, XAI is mostly exploited to improve performance and enhance understanding and trust. For policy-makers and legal auditors, instead, explanations are required to ensure a model meets compliance and regulations. For automated network management, XAI helps understanding which features are contributing to decisions and by how much a feature is influencing a particular decision, which allows practitioners to verify if the model's reasoning is correct, and to debug the model if explanations reveal that model's reasoning is not correct. Moreover, in some tasks, ML models learn relationships between input and output that domain experts are unaware about. In such cases, explanations uncover insights that help reserve-teach domain experts about the problem at hand.

How to explain a model? Models' decisions can be explained *in-model* or *post-model* (also referred to as *post-hoc*). In-model explainability refers to using a model that is intrinsically interpretable, in contrast to using complex ML models (black boxes) that lack interpretability. Post-hoc explanainability refers to explaining a complex ML models after the model has taken its decision (i.e., in a post-hoc manner). In our work, we are interested in explaining pre-trained models with the aim of understanding their reasoning, and this process is performed by using XAI frameworks in a *post-hoc* manner (see Figure 2).

Classes of XAI Frameworks. XAI frameworks can be either model-specific or model-agnostic. The application of model-specific frameworks is restricted to specific ML models, and relies on interpreting parameters of a specific model. Model-agnostic, on the contrary, can be applied to any ML model as they do not consider any components of the original ML model, which is an advantage with respect to model-specific methods.

Classes of Explanations. Explanations are divided into two classes, global and local. A global explanation explains the whole model's behavior, while a local explanation provides explanation to a specific observation (to a specific decision taken by a model for one particular data point). Depending on the aim, both types of explanations can be necessary to explain the behavior of a model.

Evaluating Explanations. The evaluation of explanations of XAI frameworks is a topic that is attracting a lot of attention. Currently, the correctness of an explanation is judged by human intervention, i.e., a human (domain expert) has to examine an explanation and to either validate or nullify its correctness, as in [12, 13]. Some recent works have started to appear where metrics and frameworks to evaluate the quality of explanations are proposed [14, 15, 16, 17, 18], however there is yet no standard approach for quantifying correctness of explanations. In our application of XAI, we rely on domain experts to judge explanations.

2.2. Related Work

Recent works highlight the challenges and opportunities of XAI either in a broad sense, such as Refs. [2], [3], [4] and [5], or with focus on communication networks as Refs. [19], [20] and [21]. In particular, Ref. [19] discusses in detail the need for explainability to improve trust between humans and AI for 6G networks, while Ref. [20] focuses on the need and requirements of explainable and traceable AI for achieving zero-touch networks.

Other works focused on the application of XAI in network management [22]-[23]. Ref. [22] proposed an XAI-based approach for resource reservation in sliced networks, to explain real-time reservation decisions and to diagnose potential faults during model's development. Ref. [24] applies XAI in the unsurpervised learning domain to traffic prediction to explain decisions taken by a YouTube video-quality classifier working under encrypted traffic. Ref. [25] proposes a XAI-based framework for the problem of network function virtualization. The application of XAI is also attracting researchers in optical networks [26, 27, 23]. In Ref. [26], authors exploit XAI to explain ML models for estimating quality-of-transmission of lightpaths in optical networks to gain insights about the problem, while Ref. [27] explains ML models for the problem of fault localization uncovering insights about the reasoning of the ML model. In Ref. [23], authors apply XAI to explain global behavior of a ML model for root-cause identification.

3. SHAP: SHapley Additive Explanations

3.1. Approximation Models

Simple ML models, such as linear regression, have an easy to understand interpretation. Complex models, such as ensemble or deep learning models, on the contrary, are not easy to understand and, hence, we cannot use these models as explanations. Instead, simpler and interpretable-by-design models can be used to approximate the behavior of the original complex model, providing an interpretable approximation of the original complex model. These models are referred to as interpretable approximation models, i.e., models that imitate the behavior of other uninterpretable ML models such as, e.g., ensemble and deep learning models, while providing a description of its own behavior, consequently explaining the behavior of the complex model. For instance, to interpret an ensemble-based model, a logistic regression model can be used as an approximation model to explain decisions boundaries and provide a description of model's behavior. Note that the decision boundary of the approximation model will only coincide with that of the original complex model in a local space, specifically in the proximity of the instance whose prediction is explained, and not globally. A well-known XAI framework that is based on approximation models is Local Interpretable Model-Agnostic Explanations (LIME) [28].

A LIME explanation is generated as follows (see Fig. 3). Select an instance x (black point in the figure) of the dataset X with its label to be explained. Perturb dataset X (i.e., change features' values of data points in X) to generate a new data set Z of a larger size with respect to original dataset X. Perturbation of original data is performed to generate new observations similar to original ones to be additionally considered when generating explanations, with the aim of better describing the behavior of the original black box model. Then, using the original black box model, predict target values for all instances in Z and weight elements in Z with respect to the proximity (also referred to as *neighborhood*) to instance x (heat-map like red circle in the figure). Note that the neighborhood is determined by giving data points weights according to their proximity to the instance to be explained. Train an interpretable approximation model g on Z and respective predictions. Finally, interpret a decision of the explainable approximation model g, and return an explanation (contributions of all features towards the prediction) for instance x.

3.2. SHAP

SHAP is a game-theory-based model-agnostic framework to explain the output of ML models by estimating each feature's contribution and influence on the model [7]. SHAP builds on the concept of *Shapley value* (designed by *Lloyd Shapley*) to assign fair payouts to players based

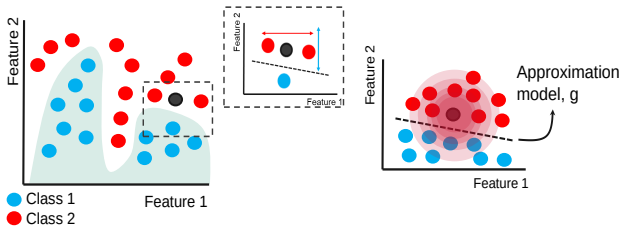


Figure 3: Graphical representation of LIME algorithm.

on their contribution to the total payout in a game. Assuming players cooperate in a coalition and receive a certain profit from this cooperation, the Shapley (or SHAP) value is computed as the average marginal contribution of a player across all coalitions of the players. In the context of explainability, the SHAP framework calculates the SHAP value (i.e., the marginal contribution or importance value) of each input feature of a prediction instance by iterating through all permutations of the input features, where a feature can be seen as a player in a game and the prediction is the payout to be distributed. More specifically, SHAP approximates the individual contribution of each feature for each data point by estimating the model output without using it versus model outputs that do include it.

$$\phi_i(f,x) = \sum_{z' \subseteq x'} \frac{|z'|!(M-|z'|-1)!}{M!}[f_x(z') - f_x(z'/i)] \quad (1)$$

Eqn. 1 shows how to calculate the SHAP value $\phi_i(f,x)$ for a feature i and for a given data point x and a ML model f. The summation considers two terms over all possible coalitions of the set of features of the data point x. The first terms, $\frac{|z'|!(M-|z'|-1)!}{M!}$, represents a weight given to each coalition, where $|z'|$ is the cardinality of the set of features in z' and M is the total number of features. This weight is high for small and large coalitions while it is low for medium-sized coalitions. The second term, $[f_x(z') - f_x(z'/i)]$, is the contribution (or importance) of feature i, which is the difference between $f_x(z')$, the output of the model for data point x of the coalition with feature i, and $f_x(z'/i)$, the output of the model without feature i. Note that same values of a feature in different data points can contribute differently to the output, depending on values of other features for each of the data points.

By approximating the SHAP value of each feature and for each data point, we can observe model's behavior for individual predictions (local explanations) and by aggregating all local explanations, we can observe global model's behavior (global predictions).

4. Use Case: Failure Identification in Microwave Networks

This section describes, as use case for the application of XAI, the problem of how to automate failure identification in microwave networks using machine learning. We

Table 1: Distribution of data points over failure classes.

Failure Cause	**# of 45-minutes windows**
C_0 - *Deep Fading*	284
C_1 - *Extra Attenuation*	581
C_2 - *Interference*	49
C_3 - *Low Margin*	190
C_4 - *Self-Interference*	187
C_5 - *Hardware Failure*	1222

first describe a microwave link and then detail the various failure root-causes in microwave networks, discussing the typical countermeasures adopted to contrast them. Finally, we present the data used in our analysis.

4.1. Microwave Link

A microwave link can function (transmit and receive) in a bidirectional manner, from site A to site B and from site B to site A, given that transmitting and receiving equipment are present at both sites. The link consists of three main elements, i.e., 1) the microwave radio, 2) the transmission line and 3) the antenna. The microwave radio can be placed at different locations, i.e., either inside a building (full-indoor), in proximity of the antenna (full-outdoor), or by adopting a hybrid solution, where the electronic devices are distributed between an outdoor unit (ODU) and indoor unit (IDU). At the transmitter side, the microwave radio is responsible of generating the analogue signal, while, at the receiver side, it demodulates the signal. The transmission line (typically a coaxial cable) connects the microwave radio to the antenna. The antenna is usually parabolic-shaped and is characterized by its gain, size and directivity function, i.e., the capability of concentrating the transmitted/received power to/from specific directions.

The performance of a microwave link is monitored by evaluating the number of severely errored seconds (SES), where a SES is a one-second period with at least 30% errored blocks, where an errored block is a group of consecutive bits in which at least one is errored. A microwave link is then considered unavailable when the number of consecutive SES exceeds 10 in at least one of the microwave link. The unavailability is then measured in terms of *Un-Availability Seconds* (UAS), which represent the amount of time (expressed in seconds) when the number of errors exceeds a certain threshold. Note that a microwave link can experience UAS for a period of time and then go back to normal functioning. This is because the microwave link can be frequently affected by external factors, such as the atmosphere, which may affect the functioning of the link temporarily. We refer the reader to [6] for more details.

4.2. Failure Classes

We consider six macro categories of failure causes for a microwave link (labeled C_0 to C_5). Five failure causes are propagation-related, i.e., they are caused by atmospheric variables or the existence of temporary barriers, while one corresponds to hardware failure, which is caused by equipment damaged due to, for example, aging or high temperature. Each of these failures require a distinct countermeasure to eliminate the fault. For instance, one failure measure may require a reconfiguration that can be done remotely while another may require on-site intervention, which in some cases may be, as previously discussed, very costly and time consuming. In the following, we briefly describe each failure type, and highlight the typical countermeasures adopted in each case (we refer the reader to [6] for a detailed description of the failures causes).

1) *Deep Fading* consists of a strong increase of channel attenuation causing a severe drop in signal-to-noise ratio. Among possible causes we can mention heavy rain, snow or fog, all atmospheric events leading to multipath and shadowing effects. Counter-measure: temporary reduction of link's modulation format (no on-site intervention).

2) *Extra Attenuation* occurs when received power is well below a minimum power threshold and can be caused by, e.g., path obstruction (due to the presence of permanent obstacles), antenna misalignment, mounting issues. Counter-measure: on-site intervention required to fix problem.

3) *Interference* occurs when a receiving antenna receives multiple bit streams due to overlap of other transmissions at its frequency, due to unexpected reflections from other links, causing it to fail to distinguish the bit stream destined to it. Counter-measure: remote intervention is enough to reconfigure link's frequency.

4) *Low Margin* occurs when the link has been misconfigured due to human error, which causes UAS events to occur. Counter-measure: remote human intervention is required to correctly configure link's parameter.

5) *Self-Interference* occurs when the link creates local signal reflections and spurious signals which are propagated to the receiver radio component, due to degradation of the hardware used to eliminate signal reflections, causing random UAS on the link. Counter-measure: on-site human intervention is required to substitute hardware components and re-configure link parameters.

6) *Hardware Failure* consists of either temporary or permanent equipment failure. Counter-measure: on-site human intervention is required to replace hardware equipment causing the failure.

4.3. Data

Data used in our study consists of 2513 data points collected from more than 225 thousand point-to-point microwave links. The 2513 data points were all hand-labeled by domain experts. Tab. 1 shows the distribution of the data points among the six failure classes. Each data point consists of a 45-minutes window observation, obtained as concatenation of three 15-minutes windows where the last window suffers from at least one UAS event. As features,

Table 2: Performance metrics of each of the three models considered in our study.

Model	Accuracy	Precision	Recall	F1-Score	F1-score per class					
					C_0	C_1	C_2	C_3	C_4	C_5
RF	0.93	0.94	0.93	0.93	0.85	0.91	0.88	0.76	0.97	0.97
ANN	0.88	0.84	0.84	0.82	0.69	0.90	0.89	0.73	0.97	0.94
XGB	0.93	0.93	0.93	0.93	0.85	0.92	0.88	0.76	0.98	0.98

we consider 5 features pertaining to link's design parameters and performance metrics, and 30 corresponding to measurements collected from the microwave link (10 from each 15-minute window). The set of features considered corresponds to that used by domain expert to identify failure causes on a microwave link. A detailed description of the feature set is reported in Tab. 3, and a detailed description of the data preprocessing can be found in Ref. [6].

5. Performance Evaluation

This section discusses numerical evaluations of the supervised ML models for failure-cause identification in microwave networks to show the reader the performance of the ML models without considering the application of XAI.

The problem of failure-cause identification in microwave networks is modeled as a supervised multi-class classification problem. As input, the supervised ML model takes a 45-minutes window observation on a microwave link, consisting of three 15-minutes windows in which the last window suffers from at least one UAS event. For a given link in a given 45-minutes window, a total of 35 features, describing link's design parameters and performance metrics, are used to model data points input. As output, the model provides a label corresponding to one of the 6 failure causes discussed previously.

We consider three different ML algorithms, namely, Artificial Neural Network (ANN), Random Forest (RF) and Extreme Gradient Boosting (XGB) for failure-cause identification. In particular, ANN and RF were adopted in our previous work [6] where details on hyperparameter selection can be found, and XGB was adopted in [29], where hyperparameters have been also optimized. In particular, ANN and RF were adopted in our previous work [6], where details on hyperparameter selection can be found. For XGB algorithm we tested different combinations of hyperparameters and used the classifier with highest classification accuracy. We vary *eta* parameter (learning rate) and *subsample* between 0.1 and 1 with a step of 0.1, and vary *max depth* between 1 and 10 with a step of 1. For *XGB*, the hyperparameters selected are *eta* $= 0.3$, *max depth* $= 7$ and *subsample* $= 0.9$.

The numerical evaluations of the supervised failure-cause identification can be found in table 2, where we show, for the three adopted ML models, Accuracy, the weighted averages of Precision, Recall and F1-score, and the F1-score for each of the classes. Results show that, in general, the three algorithms have a comparable performance with a slight advantage for XGB. Specifically, XGB has the best accuracy (93.6%) outperforming RF and ANN that have 93.04% and 88.66%, respectively. In terms of Precision, Recall and F1-Score, the XGB and RF algorithms show a similar performance, outperforming the ANN which shows a performance 10% lower for all metrics. Tab. 2 also compares the performance of the models in terms of F1-Score for the various failure classes. Results also show that XGB and RF show similar F1-score values ranging between 76% (C_3 for both models) and up to 98% (for C_4 and C_5 for XGB). Overall, numerical results show that performance of the models, RF and XGB in particular, is fairly acceptable, with a slight advantage for XGB.

6. XAI-Assisted Automated Failure Identification in Microwave Networks

In this section, we present our application of SHAP for both the *development phase* (subsection 6.1) and the *deployment pahse* (subsection 6.2). For the development phase, we first exploit SHAP for i) feature selection, ii) local explanations and iii) global explanations. In particular, we have used TreeSHAP, a model-specific SHAP framework that finds an exact computation of SHAP values with a lower complexity with respect to other versions of SHAP, by exploiting Decision Trees (DTs) structures to disaggregate the contribution of each input in a DT or DT-ensemble model [30]. For the *deployment* phase, we leverage uncertainty estimation and decomposition to filter potentially untrustworthy predictions to be explained via XAI. As ML model, we use XGB, which showed highest accuracy among the ML models in previous section.

Our proposed SHAP framework is described in Fig. 4. First, feature selection based on SHAP is performed (Steps 1-4). In Step (1), data is passed to the ML model and the model is trained and model's hyperparameters are optimized. In Step (2), SHAP calculates Shapley values of features and passes features' rank to the feature selection method in Step (3), which performs feature selection (discussed in detail later) and eliminates features in Step (4). Upon the termination of the feature selection process, the process of debugging model's misclassifications using SHAP starts with the data with reduced feature set (Step 5). In Step (6), the trained ML model classifies data points, and misclassified instances are passed to SHAP. In Step (7), misclassified instances are explained using SHAP and are then inspected by an expert who might detect

Table 3: Features describing a 45-minute window of the radio link. Feature names with '*' are measurement features with three different values, one for each 15-minutes slot.

Type	Feature	Name	Description
Link Characteristics	f_1	LowThr	Minimum received power tolerated on the link with any modulation format used (dBm)
	f_2	Ptx	Nominal transmitted power when the minimum modulation format is used (dBm)
	f_3	Thr_min	Minimum received power threshold tolerated by the link with its current modulation format (dBm)
	f_4	RxNominal	Nominal received power at the maximum modulation format (dBm)
	f_5	acmEngine	A flag which indicates if the Adaptive Code Modulation (ACM) is enabled on a given microwave link
G.828 metrics	f_6, f_7, f_8	ES*	Number of one-second periods with at least one ES in the 15-minutes slot
	f_9, f_{10}, f_{11}	SES*	Number of one-second periods with at least one SES in the 15-minutes slot
Power values	f_{12}, f_{13}, f_{14}	txMaxA*	Maximum power transmitted from site A in in the 15-minutes slot (dBm)
	f_{15}, f_{16}, f_{17}	txminA*	Minimum power transmitted from site A in the 15-minutes slot (dBm)
	f_{18}, f_{19}, f_{20}	rxmaxA*	Maximum power received at site A in the 15-minutes slot (dBm)
	f_{21}, f_{22}, f_{23}	rxminA*	Minimum power received from site A in the 15-minutes slot (dBm)
	f_{24}, f_{25}, f_{26}	txMaxB*	Maximum power transmitted from site B in the 15-minutes slot (dBm)
	f_{27}, f_{28}, f_{29}	txminB*	Minimum power transmitted from site B in the 15-minutes slot (dBm)
	f_{30}, f_{31}, f_{32}	rxmaxB*	Maximum power received at site B in the 15-minutes slot (dBm)
	f_{33}, f_{34}, f_{35}	rxminB*	Minimum power received from site B in the 15-minutes slot (dBm)

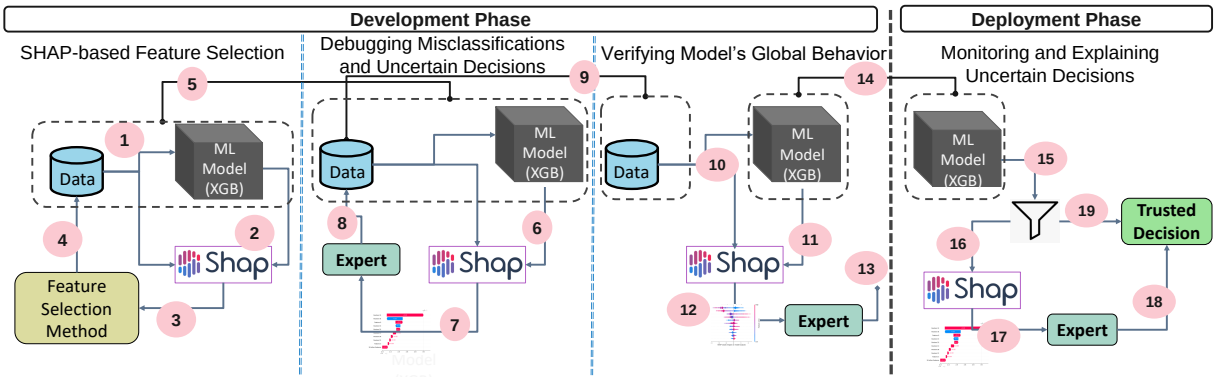


Figure 4: Overview of our proposed SHAP framework for the *development* and *deployment* phases.

mislabeled data points and modify input data accordingly (Step 8). Modified input data is then passed to model for retraining (Steps 9 and 10). In Step (11), SHAP is passed the input data and the model to perform global explanations which are passed to an expert (Step 12) who can verify the model's global behavior (Step 13). The model is then moved for deployment (Step 14). In deployment, the process of explaining uncertain decisions takes place. Decisions are passed to a filter based on uncertainty value in Step (15). Decisions with uncertainty values exceeding a predefined threshold (explained in detail later) are moved to SHAP (Step 16) and local explanations are extracted (Step 17) and passed to expert who can contest model's decision based on explanations (Step 18) to have a trusted decision. Decisions whose uncertainty value does not exceed the threshold are considered trusted (Step 19).

6.1. Development

Feature Selection with SHAP. Feature selection is the process of optimizing the number of input variables by eliminating features that are either redundant or unnecessary without experiencing harmful information loss. For any machine learning task, feature selection is a key method to reduce complexity. Additionally, for examining explanations, a reduced set of features may make models easier to be understood by practitioners and domain experts.

In this section, we use ranking-based feature selection methods. We exploit SHAP to rank features (i.e., we rank features based on their SHAP value), and then give the ranking as input to the feature selection method. We compare the results of the SHAP-based feature selection process against more traditional feature importance frameworks based on decision trees for the XGB model we adopt in this paper.

With respect to the tree-based feature importance process, SHAP is expected to yield smaller size of final set of features and higher classification accuracy. This is based on the fact that classical features importance (features' impact on model's performance) in gradient boosting models might be, in some cases, misleading and unreliable [31]. In boosting models, importance of a feature is calculated for a single decision tree by the amount that each attribute split point improves the performance measure (the accuracy), weighted by the number of observations the node is responsible for. Specifically, each feature has a *Gain*, which is a modification of information gain that reduces its bias (but don't solve the issue totally), and refers to the increase in accuracy carried by the feature to the tree branch it is on, i.e., the difference in accuracy before and

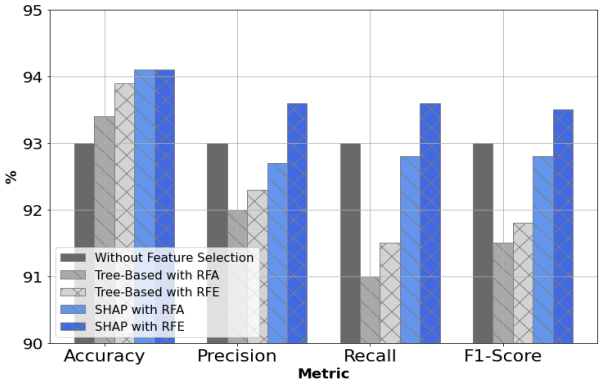


Figure 5: Feature Selection Classification Results.

after adding a new split on a feature X to the branch. This *Gain* denotes the related feature's relative contribution to the model, which is derived by considering each feature's contribution for each tree in the model.

As feature selection methods, we consider two methods: Recursive Feature Elimination (RFE) and Recursive Feature Addition (RFA). In RFE, at each stage of the search, the least important features are iteratively eliminated. Conversely, in RFA, the algorithm eliminates the highest-ranked feature and at the end of the process, it returns the reverse ranking [32]. In our simulations, we considered the addition/elimination of 1 feature at each iteration. In both approaches, the algorithm stops when the feature set is empty.

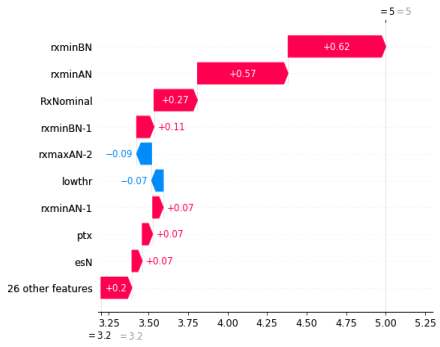
The feature selection process with SHAP is as follows (depicted in left part of Fig. 4). Starting with the initial set of features, a model is trained and the SHAP values are computed resulting in a feature rank. Then, a feature is eliminated or added (based on RFE or RFA) to produce a new feature set. A new model is then trained (with tuning of hyperparameters) with the new set of features. The process repeats until reaching the stopping condition. Finally, the resulting set of features is that whose model yielded best performance in terms of accuracy.

In terms of resulting set of features, RFE based on SHAP reduced the set of features to 11 while that based on tree-based feature importance resulted in a set of 17 features. RFA based on SHAP resulted in a set of 25 features while that based on tree-based feature importance resulted in a set of 31 features.

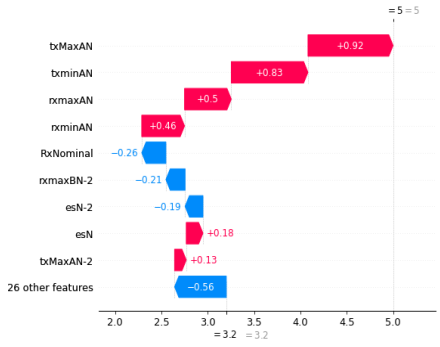
Figure 5 reports results comparing the performance of the models considering the resulting set of features in each of the cases. Results show that feature selection based on SHAP, either with RFA or RFE, has highest accuracy of around 94.1% (1% improvement with respect to that without feature selection and slightly better than that of feature selection based tree-based feature importance). In terms of precision, recall and F1-score, RFE based on SHAP shows highest performance with an improvement of around 0.5% from the case without feature selection. As for RFA with SHAP, it outperforms tree-based feature importance and a comparable performance to the case without feature selection. These findings demonstrate that SHAP can be exploited, and is in fact highly efficient, when combined with feature selection approaches, as it reduced the set of features to 11 (starting with 35) while showing an increase in performance measures.

Explaining Individual Decisions (Local Explanations)

A local explanation provides explanations to a specific decision taken by the model (referring to one particular data point). In the development phase, local explanations can be used to debug model reasoning, whether for correct or wrong decisions. In other words, such explanations allow to debug the way the model arrived to the decision. For instance, when the model misclassifies a data point, model's decision can be explained with the aim of inferring its reasoning and hence, the reasons behind the misclassification. A correct reasoning hints to a possible mis-labelling of the data point whose decision being explained. On the contrary, a wrong reasoning indicates that the model has learned a wrong relationship between features and a particular class. Note that this is an iterative process that involves the domain experts, which validate the behavior of the model. Such analysis can be leveraged by domain experts to gain insights on the problem at hand, allowing to know when the model might misclassify one class of failure to another, and therefore derive additional guidelines that would allow to avoid taking costly wrong countermeasures in future occurrences.



(a)



(b)

Figure 6: Two examples of SHAP local explanations.

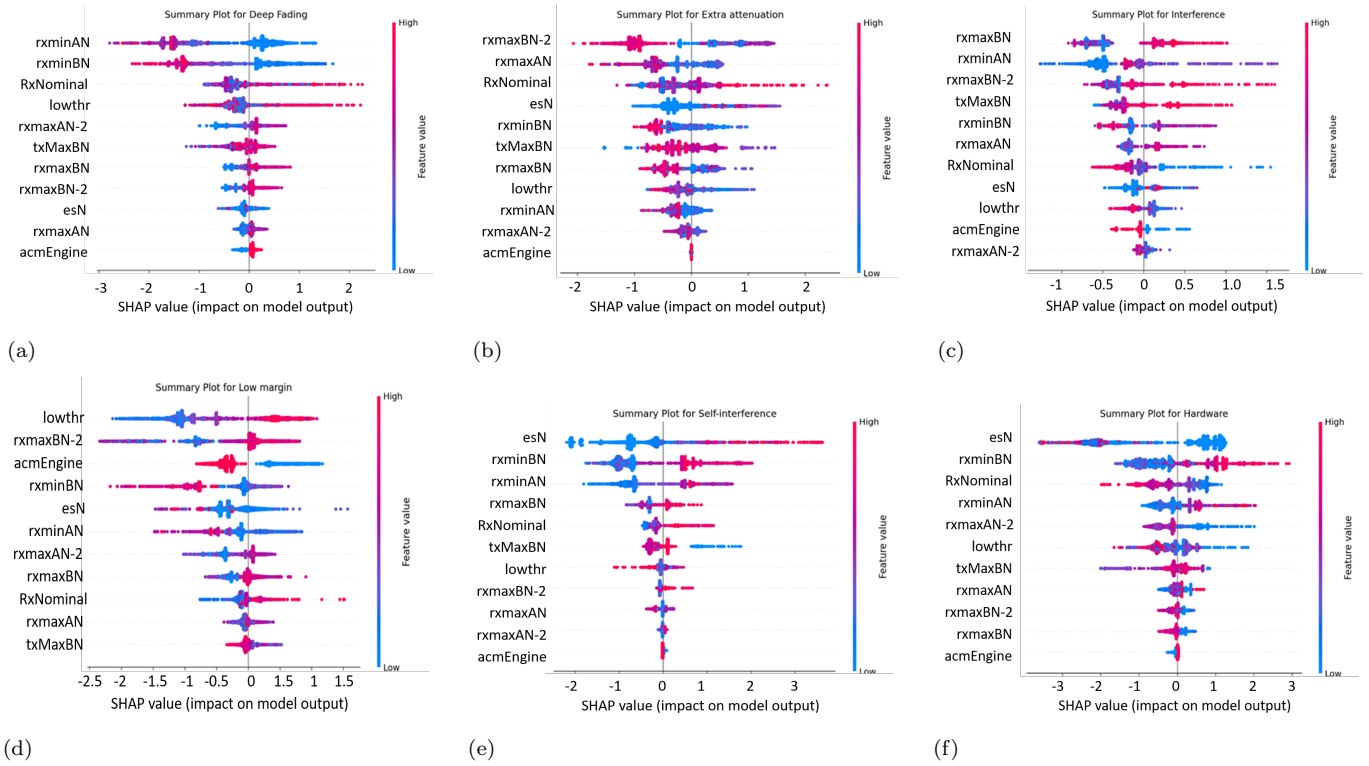


Figure 7: Summary plot of SHAP values for (*a*) Deep Fading, (*b*) Extra Attenuation, (*c*) Interference, (*d*) Low Margin, (*e*) Self-Interference and (*f*) Hardware Failure.

We show an example of two local explanations with the aim of clarifying how feature importance in local explanations can be used to debug model's behavior. We show in Fig. 6 a SHAP *waterfall* plots for two data point classified as class 5. The explanation can be read as follows. The y-axis is the list of features in descending order of importance according to SHAP, while the x-axis represents the SHAP value. Each of the explanations shows how the model arrived to its decision (i.e., to the classified class) starting from model's *base value*, which is the expected value of model's output (=3.2 in our case), by showing bars for each of the features either pointing towards the decision taken by the model or against it. Fig. 6(a) refers to a correctly classified instance. The explanation shows that most features in the instance pushed the model's decision towards the correct label (i.e., all features have a positive impact towards classifying the instance as class 5). In Fig. 6(b) we show an explanation of a wrongly classified instance (classified as class 3 instead of class 5) where some features have negative impact towards classifying the instance as class 5. Analyzing which features pushed model's decision wrongly, domain experts can identify reasons why model mis-classified these instances. In the development and testing phase, such misclassification can be due to the fact that the instance is mislabeled, and in that case, the label of an instance can be corrected by domain experts, or simply due to that fact that it is an instance that is hard for the model to classify. In the latter case, the explanation can help domain experts why the model fails to correctly classify such instances. During deployment phase, domain experts can leverage on these explanations to analyze model's decision before taking any actions (discussed in detail later in Sec. 6.2).

Explaining Model's Global Behavior

To explain model's global behavior, we calculate SHAP values of all points in our data set and plot SHAP's *summary plot* for each class of failure. A summary plot (shown in Fig. 7) combines feature importance with feature values to explain model's behavior. The y-axis indicates feature names listed in descending order of their importance. On the x-axis is the SHAP value. Each point of the summary plot on a given row (i.e., for a given feature) is a SHAP value for the feature for a given data point in our data set (a 45-minutes window for a given link, in our case), positioned based on its SHAP value. If a point has a positive SHAP value, it means that it contributes positively towards the prediction of the class being explained, whereas if the plot has a negative SHAP value, it means that it contributes negatively against the prediction of the class being explained. Each point is given a color which qualitatively represents the original feature value in a low-to-high scale. The overlapping points in vertical direction reflect the distribution of SHAP values for each feature. By examining summary plots of each class of failure, we understand the relationship between value of a feature (color of a point) and the impact on the prediction (SHAP value

of a point) towards a class of failure (decisions being explained). This knowledge can be leveraged to have a global understanding of model's behavior, and, when analyzed per failure-cause class, allows to extract decisive insights. While domain experts know, at a global level, that a set of performance metrics (features) is necessary for failure-cause identification, discovering which features are linked to specific failures is important and can result decisive for implementing AI-driven solutions for predictive maintenance, for instance.

We now show how to use these explanations to understand model's behavior. Consider Fig. 7(a), which shows the summary plot of *Deep Fading*. The explanation figure shows that *rxminAN* and *rxminBN* are the two most important features for identifying *Deep Fading*, while *acmEngine* is the least important one. This piece of information extracted from the explanation figure can be either negated or verified by domain experts. In other words, domain experts can answer the question: *Is it true that rxminAN and rxminBN are of out-most importance and that acmEngine is of least importance when identifying Deep Fading?*. If the answer is always yes, domain experts can verify that the model bases its reasoning on correct features while, if this is not what domain experts expect, it means that either the model based its reasoning on wrong features or that it learned new relationships among the features that domain experts were unaware about. In either cases, the *summary plot* provides additional information to debug model's behavior in more details. Consider, for instance, the row of *rxminAN* for *Deep Fading*. When *rxminAN* has low values (blue-colored points), it has positive SHAP values, suggesting that it drives model's decision towards *Deep Fading*. Instead, when feature value is high (red- and purple-colored points), it has negative SHAP values, meaning it drives model's decision against *Deep Fading*. We can see a similar behavior for *rxminBN* while for *rxmaxAN-2*, for instance, high (or low) feature values drive decision towards (or against) *Deep Fading*. This analysis can be repeated for all features and for all failure-cause classes with the aim of having a global understanding of model's behavior.

Limitations We now discuss the main limitations of our application of SHAP. As previously stated, the calculation of Shapley values is complex. In our application of SHAP, we have considered the use of TreeSHAP [30], which is a model-specific SHAP framework applicable only to tree-based models and is capable of estimating the Shapley values efficiently. With other ML models such as ANNs, the use of SHAP yields a high degree of complexity (high computational amount of time), particularly when data sets are relatively large.

6.2. Deployment Phase

During deployment phase the trained model acts as a black-box that, given the input data, outputs a prediction. Even if the model achieved high testing and validation accuracy, these metrics alone can be a bad proxy for trustworthiness. This is because a model may output arbitrarily high class probabilities for misclassified and out-of-distribution samples [28, 33]. To continuously monitor the model's trustworthiness at deployment time, we need to 1) quantitatively assess the model's uncertainty about its own predictions, and 2) to provide an explanation for the samples whose predictions we deem to be untrustworthy. We can adress the two issue above 1) via uncertainty quantification and decomposition [34], and 2) via local explanations through XAI.

Uncertainty in model predictions can be decomposed in aleatoric (or data) uncertainty and epistemic (or knowledge) uncertainty [35]. Aleatoric uncertainty is caused by the inherent randomness in the input data, whereas epistemic uncertainty is caused by the lack in the model's knowledge. The latter may be due to samples either out of distribution or sparsely covered by the training set.

For gradient boosted classification models, aleatoric and epistemic uncertainty can be captured by considering an ensemble of models. The output of each model in the ensemble is a categorical distribution over the output classes. Let $\mathbf{x}$ be an input sample, y its output, θ be the model parameters, and $\mathcal{D}$ be the training dataset. Then, the model uncertainty on the output y is defined as [34, 36]:

$$\underbrace{\mathcal{I}(y,\theta|\mathbf{x},\mathcal{D})}_{\text{Epistemic}} = \underbrace{\mathcal{H}\left[p(y|\mathbf{x},\mathcal{D})\right]}_{\text{Total}} - \underbrace{\mathbb{E}_{p(\theta|\mathcal{D})}\mathcal{H}\left[p(y|\mathbf{x},\theta)\right]}_{\text{Aleatoric}} \tag{2}$$

$$\approx \mathcal{H}\left[\frac{1}{M}\sum_{m=1}^{M} p(y|\mathbf{x},\theta^{(m)})\right] - \frac{1}{M}\sum_{m=1}^{M}\mathcal{H}\left[p(y,|\mathbf{x},\theta^{(m)})\right] \tag{3}$$

where $\mathcal{I}[\cdot]$ is the mutual information, $H[\cdot]$ is the entropy, and $\theta^{(m)}$ are the parameters of the m-th model in the ensemble. The total uncertainty is computed as the entropy of the predictive posterior $p(y|\mathbf{x},\mathcal{D})$, which is approximated by taking the average of the prediction probabilities for each model in the ensemble. The aleatoric uncertainty is computed as the expected entropy of each model $p(y|\mathbf{x},\theta^{(m)})$ in the ensemble. The epistemic uncertainty is computed as the difference between total uncertainty and aleatoric uncertainty, and is equal to the mutual information between the model output y and the model parameters θ. Note that an estimate of total uncertainty can be derived from a single model by computing the entropy of its predictive distribution. However, an ensemble of models is needed for estimating knowledge uncertainty.

Ensembles can be easily generated by training Stochastic Gradient Boosting (SGB) models using different random seeds. XGB SGB models can be trained by specifying the subsample hyperparameter present in the XGB API. The subsample hyperparameter is a value in $(0, 1)$ that determines the fraction of the training set subsampled at each gradient boosting update. Even though SGB does not guarantee sampling models from the true posterior $p(\theta|\mathcal{D})$, the resulting uncertainty estimates are good enough in practice [36].

As outlined in [37], XAI and uncertainty analysis operate in orthogonal domains. XAI evaluates what drives the model to a certain decision, whereas uncertainty analysis evaluates the model's confidence for a certain prediction. Linking XAI and uncertainty would then allow to explain *what causes a model to be certain/uncertain about a prediction*. In particular, since inspecting the explanations for each sample at deployment time is not practically feasible, we can leverage uncertainty to *selectively apply XAI to potentially untrustworthy predictions*.

Evaluating the distribution of uncertainties in the predictions can be used to identify potentially-misclassified samples. Potentially misclassified samples can be then further analyzed using XAI to understand which features are causing confusion in the deployed model. However, a misclassified sample may or may not belong to a class inside the training set. The latter scenario may present itself in the case of new failure classes appearing due to, e.g., network aging. Therefore, we can distinguish between the following use-cases for uncertainty analysis:

1. Flag potentially misclassified samples.
2. Flag potentially out-of-distribution samples.

These two use-cases are not mutually exclusive. For instance, an input sample might have been misclassified due to being out-of-distribution. Depending on which use-case we want to address, we need to leverage the appropriate measure of uncertainty.

Misclassifications in general are caused both by noise in the input data and by the model's lack of knowledge. Therefore, we expect that misclassified samples can be discriminated by higher values of total uncertainty. Input samples flagged as misclassified can be then further inspected using XAI.

Since out-of-distribution samples are not covered by the training data, we can expect that the model's lack of knowledge will play a greater role in the total prediction uncertainty. Therefore, we expect that out-of-distribution samples can be discriminated by higher values of knowledge uncertainty. Again, input samples flagged as out-of-distribution can be further inspected using XAI.

For both use-cases, the uncertainty estimates will be used as scores for a binary classifier. In particular, we will consider the following two binary classification problems:

1. **Misclassification detection**. Predictions are labelled as 1 if correctly classified, 0 otherwise.
2. **Out-of-distribution detection**. Predictions are labelled as 1 if the corresponding input is out-of-distribution data, and 0 otherwise. To simulate out-of-distribution data appearing at deployment time, we remove entirely one class from the training set, but we keep the test set unaltered. This procedure is performed for every class in the training set and results are averaged.

We evaluate the performance of the proposed binary classifiers with AUC-ROC. However, for misclassification detection the number of misclassified samples is very low compared to correct classification, therefore AUC-ROC might result artificially high. Because of this, for misclassification detection only we also measure Prediction-Rejection-Ratio (PRR) [36]. We define predictions flagged as either misclassified or from o.o.d. data as "rejected". In particular, assume that rejected misclassified test samples can be corrected after a domain expert inspection. The prediction-rejection plot shows the improvement of the overall classification error in the test set as a function of the percentage of rejected test samples, i.e., the percentage of test samples that will need to be inspected. The PRR is computed based on a prediction-rejection plot as the one shown in Fig. 8. Let A_{oracle} the area delimited by the random and the oracle curves, and A_{XGB} the area delimited by the random and XGBoost Ensemble curves. The PRR is therefore computed as $PRR = \frac{A_{\text{XGB}}}{A_{\text{oracle}}}$. As such, a PRR = 1 indicates performance equal to an oracle rejector, whereas a PRR = 0 indicates performance equal to a random rejector. For example, for the train-test split of Fig. 8, we have PRR = 0.84, and we can reduce our classification error to less than 2% by rejecting 10% (50 samples) of the test set.

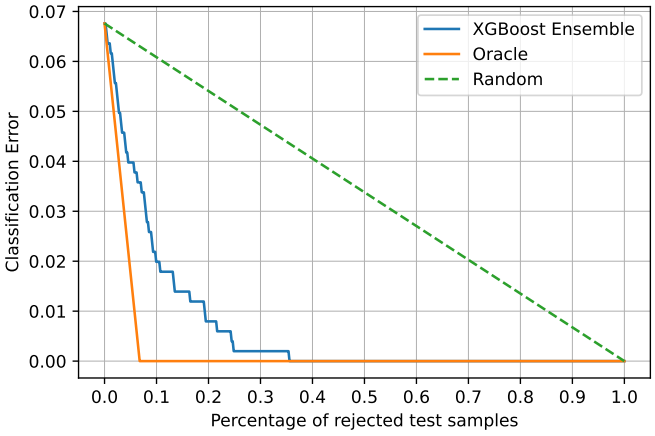


Figure 8: Prediction-rejection plot of an XGBoost Ensemble for misclassification detection in a single 80/20 train-test split, using total uncertainty as rejection score.

In Table 4 we report PRR and AUC-ROC for misclassification and out-of-distribution detection. We compare the performance between a single XGB model against an ensemble of 10 XGB SGB models. Moreover, we compare the PRR and AUC-ROC obtained by thresholding both total uncertainty (TU) and knowledge uncertainty (KU). Note that from a single XGB model we can get an estimate of the total uncertainty by computing the entropy of the prediction probabilities. Values are computed by averaging over ten random 80/20 train-test splits for the considered dataset.

For misclassification detection, the performance metrics are fairly good for all the considered cases, with the XGB ensemble slightly outperforming the single XGB model.

For the XGB ensemble, in contrast to what we initially conjectured, we observe that knowledge uncertainty discriminates misclassifications better than total uncertainty. This gives us a precious insight: for this dataset and this trained model, misclassifications can be reliably imputed to the model's lack of knowledge, i.e., to the fact that there are underrepresented classes in the training set. Therefore, the performance of our model can indeed be improved by collecting more data for the least represented classes. Note that we could not have drawn this conclusion without uncertainty decomposition.

For out-of-distribution detection the performance metrics are again fairly good for all the considered cases, with knowledge uncertainty from the XGB ensemble having the best discriminative power, as expected. Therefore, samples from classes unseen during training can be filtered, labeled properly by domain experts and inserted in the training set, continuously improving the multi-class classification model over time.

Finally, Tab. 5 shows the classification error and rejected test samples for misclassification detection while varying KU uncertainty threshold. Results show that system classification error can be reduced by around 50% by rejecting 6.3% of the test samples.

7. Conclusion

In this work, we investigate the use of eXplainable Artificial Intelligence (XAI) for automated failure-cause identification in microwave networks. We first model the problem of failure-cause identification in operational microwave networks as a supervised multi-class classification problem and then propose a framework for the application of XAI for the *development* phase and the *deployment* phase. As a XAI framework, we rely on Shapley Additive Explanations (SHAP). For the *development* phase, we first leverage SHAP in the process of feature selection. Results show that SHAP-assisted feature selection allows reducing the number of features more than tree-based feature selection approach, while achieving an improvement in terms of model's accuracy. We then show how to exploit SHAP to describe model's local and global to enhance trust in the model prior to its deployment. For the *deployment* phase, and since explaining all decisions of models require efforts from domain experts and might be unnecessary such as in the case of correct predictions, we propose a framework based on model's uncertainty to explain and refer to domain expert predictions that are most likely to be wrong. Results show that with uncertainty thresholding it is possible to filter potentially misclassified predictions, reducing the number of samples that a domain expert needs to visually inspect. On average, we can reduce the system classification error by more than 50% by rejecting 6.3% of the test samples. The trade-off between classification error and samples to be inspected can be freely tuned by setting the appropriate uncertainty threshold. As a future work, we plan to leverage the proposed framework for uncertainty quantification and local explanations to detect model drift. Additionally, as a possible future research direction, we emphasize on exploring the application of explainable artificial intelligence techniques in other wireless communication domains.

References

[1] U. Bhatt, A. Xiang, S. Sharma, A. Weller, A. Taly, Y. Jia, J. Ghosh, R. Puri, J. M. Moura, P. Eckersley, Explainable machine learning in deployment, in: Proceedings of the 2020 conference on fairness, accountability, and transparency, 2020, pp. 648–657.

[2] A. Das, P. Rad, Opportunities and challenges in explainable artificial intelligence (xai): A survey, arXiv preprint arXiv:2006.11371 (2020).

[3] A. Adadi, M. Berrada, Peeking inside the black-box: a survey on explainable artificial intelligence (xai), IEEE access 6 (2018) 52138–52160.

[4] W. Samek, T. Wiegand, K.-R. Müller, Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models, arXiv preprint arXiv:1708.08296 (2017).

[5] D. Gunning, M. Stefik, J. Choi, T. Miller, S. Stumpf, G.-Z. Yang, Xai—explainable artificial intelligence, Science Robotics 4 (37) (2019) eaay7120.

[6] F. Musumeci, L. Magni, O. Ayoub, R. Rubino, M. Capacchione, G. Rigamonti, M. Milano, C. Passera, M. Tornatore, Supervised and semi-supervised learning for failure identification in microwave networks, IEEE Transactions on Network and Service Management 18 (2) (2020) 1934–1945.

[7] S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, Advances in neural information processing systems 30 (2017).

[8] T. Miller, Explanation in artificial intelligence: Insights from the social sciences, Artificial intelligence 267 (2019) 1–38.

[9] M. Du, N. Liu, X. Hu, Techniques for interpretable machine learning, Communications of the ACM 63 (1) (2019) 68–77.

[10] S. Chakraborty, R. Tomsett, R. Raghavendra, D. Harborne, M. Alzantot, F. Cerutti, M. Srivastava, A. Preece, S. Julier, R. M. Rao, et al., Interpretability of deep learning models: A survey of results, in: 2017 IEEE smartworld, ubiquitous intelligence & computing, advanced & trusted computed, scalable computing & communications, cloud & big data computing, Internet of people and smart city innovation (smartworld/SCALCOM/UIC/ATC/CBDcom/IOP/SCI), IEEE, 2017, pp. 1–6.

[11] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, K.-R. Müller, How to explain individual classification decisions, The Journal of Machine Learning Research 11 (2010) 1803–1831.

[12] J. van der Waa, E. Nieuwburg, A. Cremers, M. Neerincx, Evaluating xai: A comparison of rule-based and example-based explanations, Artificial Intelligence 291 (2021) 103404.

[13] S. Mohseni, J. E. Block, E. D. Ragan, A human-grounded evaluation benchmark for local explanations of machine learning, arXiv preprint arXiv:1801.05075 (2018).

[14] S. Mohseni, N. Zarei, E. D. Ragan, A multidisciplinary survey and framework for design and evaluation of explainable ai systems, ACM Transactions on Interactive Intelligent Systems (TiiS) 11 (3-4) (2021) 1–45.

[15] J. Zhou, A. H. Gandomi, F. Chen, A. Holzinger, Evaluating the quality of machine learning explanations: A survey on methods and metrics, Electronics 10 (5) (2021) 593.

[16] G. Vilone, L. Longo, Notions of explainability and evaluation approaches for explainable artificial intelligence, Information Fusion 76 (2021) 89–106.

[17] S. R. Islam, W. Eberle, S. K. Ghafoor, Towards quantification

Table 4: Performance metrics of the uncertainty-based binary classifiers.

Model		Misclassification		Out-of-distribution
		AUC-ROC	PRR	AUC-ROC
XGB Single	TU	0.92	0.84	0.88
	KU	-	-	-
XGB Ensemble	TU	0.93	0.85	0.88
	KU	0.93	0.87	0.89

Table 5: Classification error vs. rejected test samples for misclassification detection w.r.t. fixed knowledge uncertainty thresholds.

Classification Error%	Rejected Test Samples%	#Rejected Test Samples	KU Threshold
6.56	0	0	-
5.05	1.81	10	0.0713
4.00	4.20	22	0.0479
3.06	6.30	32	0.0344
2.25	9.05	46	0.0206
1.01	16.1	82	0.00433
0	46.7	236	0.0000710

of explainability in explainable artificial intelligence methods, in: The thirty-third international flairs conference, 2020.

[18] A. Rosenfeld, Better metrics for evaluating explainable artificial intelligence, in: Proceedings of the 20th international conference on autonomous agents and multiagent systems, 2021, pp. 45–50.

[19] W. Guo, Explainable artificial intelligence for 6g: Improving trust between human and machine, IEEE Communications Magazine 58 (6) (2020) 39–45.

[20] B. Dutta, A. Krichel, M.-P. Odini, The challenge of zero touch and explainable ai, Journal of ICT Standardization (2021) 147–158.

[21] D. Thakker, B. K. Mishra, A. Abdullatif, S. Mazumdar, S. Simpson, Explainable artificial intelligence for developing smart cities solutions, Smart Cities 3 (4) (2020) 1353–1382.

[22] P. Barnard, I. Macaluso, N. Marchetti, L. A. DaSilva, Resource reservation in sliced networks: An explainable artificial intelligence (xai) approach (2021).

[23] C. Zhang, D. Wang, L. Wang, L. Guan, H. Yang, Z. Zhang, X. Chen, M. Zhang, Cause-aware failure detection using an interpretable xgboost for optical networks, Optics Express 29 (20) (2021) 31974–31992.

[24] A. Morichetta, P. Casas, M. Mellia, Explain-it: Towards explainable ai for unsupervised network traffic analysis, in: Proceedings of the 3rd ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks, 2019, pp. 22–28.

[25] S. Sharma, A. Nag, L. Cordeiro, O. Ayoub, M. Tornatore, M. Nekovee, Towards explainable artificial intelligence for network function virtualization, in: Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies, 2020, pp. 558–559.

[26] O. Ayoub, A. Bianco, D. Andreoletti, S. Troia, S. Giordano, C. Rottondi, On the application of explainable artificial intelligence to lightpath qot estimation, in: Proceedings of the Optical Fiber Communication Conference (OFC), 2021.

[27] O. Karandin, O. Ayoub, F. Musumeci, Y. Hirota, Y. Awaji, M. Tornatore, If not here, there. explaining machine learning models for fault localization in optical networks, in: Proceedings of the Optical Fiber Communication Conference (OFC), 2021.

[28] M. T. Ribeiro, S. Singh, C. Guestrin, "Why should I trust you?" Explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp. 1135–1144.

[29] O. Ayoub, F. Musumeci, F. Ezzedine, C. Passera, M. Tornatore, On using explainable artificial intelligence for failure identification in microwave networks, in: 25th Conference on Innovation in Clouds, Internet and Networks (ICIN), 2022.

[30] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, S.-I. Lee, From local explanations to global understanding with explainable ai for trees, Nature machine intelligence 2 (1) (2020) 56–67.

[31] J. Tang, S. Alelyani, H. Liu, Feature selection for classification: A review, Data classification: Algorithms and applications (2014) 37.

[32] M. Kuhn, K. Johnson, et al., Applied predictive modeling, Vol. 26, Springer, 2013.

[33] A. Bendale, T. E. Boult, Towards open set deep networks, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Los Alamitos, CA, USA, 2016, pp. 1563–1572. doi:10.1109/CVPR.2016.173.

[34] S. Depeweg, J.-M. Hernandez-Lobato, F. Doshi-Velez, S. Udluft, Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning, in: International Conference on Machine Learning, PMLR, 2018, pp. 1184–1193.

[35] A. D. Kiureghian, O. Ditlevsen, Aleatory or epistemic? does it matter?, Structural Safety 31 (2) (2009) 105–112, risk Acceptance and Risk Communication.

[36] A. Malinin, L. Prokhorenkova, A. Ustimenko, Uncertainty in gradient boosting via ensembles, in: International Conference on Learning Representations, 2021.
URL https://openreview.net/forum?id=1Jv6b0Zq3qi

[37] D. Seuß, Bridging the gap between explainable AI and uncertainty quantification to enhance trustability, CoRR abs/2105.11828 (2021). arXiv:2105.11828.
URL https://arxiv.org/abs/2105.11828