Explainable Machine Learning for Performance Anomaly Detection and Classification in Mobile Networks [⋆]

Juan M. Ramírez[a,*], Fernando Díez[b], Pablo Rojo[c], Vincenzo Mancuso[a], Antonio Fernández-Anta[a]

[a]*IMDEA Networks Institute, 28918, Madrid,*
[b]*Universidad Politecnica de Madrid,*
[c]*Nokia CNS,*

Abstract

Mobile communication providers continuously collect many parameters, statistics, and key performance indicators (KPIs) with the goal of identifying operation scenarios that can affect the quality of Internet-based services. In this regard, anomaly detection and classification in mobile networks have become challenging tasks due to both the huge number of involved variables and the unknown distributions exhibited by input features. This paper introduces an unsupervised methodology based on both a data-cleaning strategy and explainable machine learning models to detect and classify performance anomalies in mobile networks. Specifically, this methodology dubbed explainable machine learning for anomaly detection and classification (XMLAD) aims at identifying features and operation scenarios characterizing performance anomalies without resorting to parameter tuning. To this end, this approach includes a data cleaning stage that extracts and removes outliers from experiments and features to train the anomaly detection engine with the cleanest possible dataset. Moreover, the methodology considers the differences between discretized values of the target KPI and labels predicted by the anomaly detection engine to build the anomaly classification engine which identifies features and thresholds that could cause performance anomalies. The proposed methodology incorporates two decision tree classifiers to build explainable models of anomaly detection and classification engines whose decision structures recognize features and thresholds describing both normal behaviors and performance anomalies. We evaluate the XMLAD methodology on real datasets captured by operational tests in commercial networks. In addition, we present a testbed that generates synthetic data using a known TCP throughput model to assess the accuracy of the proposed approach.

Keywords: anomaly detection and classification, data cleaning, decision tree classifiers, explainable machine learning, mobile networks.

1. Introduction

Mobile networks have shown remarkable growth over the last two decades. More precisely, fifth-generation (5G) mobile technologies have played an important role in wireless network access providing high-speed connectivity for an increasing number of heterogeneous devices [2, 3]. With the consolidation of the 5G and the rising of the 6G, mobile networks exhibit increasingly complex structures, with distributed servers operating in a coordinated way to ensure reliable network services. With mobile networks becoming increasingly complex, it has become necessary to develop different approaches based on learning architectures to solve operating problems [4], such as network traffic [5], dynamic spectrum access [6] and energy-efficient computation [7]. In this context, mobile network providers continuously collect information about the communication system with the aim of detecting operation instances that could unveil the misfunctioning of different network components. However, network health diagnosis and anomaly identification have become challenging tasks because of the increasing structural complexity of mobile networks requiring a large number of variables to monitor the communication system performance.

In the context of computer networks, anomalies can be categorized into two groups: (i) *performance anomalies* and (ii) *security anomalies* [8, 9]. On one hand, performance anomalies comprise operation scenarios related to network efficiency loss or misfunctioning. These scenarios induce undesired events such as large-time file downloads or low-quality connections. On the other hand, security anomalies are associated with malicious intrusions aiming at affecting partially or totally the communication network. In this work, we focus on detecting and classifying performance anomalies in mobile networks. Existing approaches leverage the machine learning framework with supervised (cf., [10, 11]) and unsupervised mechanisms (cf., [12, 13, 14]) to detect performance anomalies, outages, or faults. For example, Falk et al. introduced

[⋆]A preliminar version of this manuscript is included in the proceedingns of the 20th Mediterranean Communication and Computer Networking Conference (MedComNet 2022) [1].

[*]Corresponding author.

Email address: `juan.ramirez@imdea.org` (Juan M. Ramírez) in [10] an architecture based on supervised learning models for detecting network outages from 4G-LTE performance data. On the same topic, Moulay et al. reported in [11] a methodology combining supervised and unsupervised learning models for fault identification in cellular networks. However, labels are frequently unavailable for performance anomaly detection in mobile networks, and labeling is a costly data engineering process [15]. Moreover, due to the rapid growth and evolution of mobile networks, the labeling process typically provides insufficient information to describe the wide range of anomalous scenarios that arise as a result of incorporating new technologies and devices into the network architecture. Various works [12, 13] developed unsupervised and non-parametric machine learning architectures for predicting network outages using a small set of basic features extracted manually from available data. Also, an unsupervised anomaly detection algorithm was reported in [14] to detect abrupt changes affecting multiple KPIs in traffic time series. These methods typically operate ad-hoc, and, more importantly, they do not attempt to determine what aspects of the network are causing the anomalies. Moreover, these methods require fine hyperparameter tuning.

This paper presents a machine learning methodology for detecting and classifying mobile network performance anomalies that is unsupervised and non-parametric. The proposed methodology referred to as explainable machine learning for anomaly detection and classification (XMLAD) identifies in an unsupervised fashion both network performance indicators (features) and operation scenarios (experiments) that could cause performance anomalies. To this end, the proposed approach includes a data-cleaning stage that discards features and experiments that allow us to characterize the normality model so that deviations from the model can be detected. After that, XMLAD considers both the classification error yielded by the anomaly detection engine and the statistical modeling of features in order to optimize the anomaly classification engine that identifies features that cause abnormal behaviors. The proposed methodology incorporates two decision tree classifiers that generate explainable structures that allow the identification of both normal operation scenarios and performance anomalies. Additional contributions of this paper are summarized as follows:

1. This work includes a data preprocessing stage divided into three steps: missing value identification and removal, low variability detection and removal, and outlier detection in features. These steps extract a set of relevant features based on data statistics. Moreover, we incorporate a training set preparation stage divided into two modules: outlier detection in experiments and a feature selection method based on a 2D clustering technique. Specifically, the aim of the training set preparation is to obtain samples and labels to train the anomaly detection engine. Notice that the anomaly detection engine should describe the network's normal behavior such that misclassified samples characterize anomalous scenarios.
2. The XMLAD methodology uses an unsupervised and non-parametric approach to identify problematic features and experiments causing anomalies. In order to accomplish this, XMLAD applies a 1D clustering technique to each feature vector with the aim of identifying different behaviors implicitly embedded in feature distributions. Afterward, the procedure considers the labels predicted by the anomaly detection engine to determine the misclassification density for each cluster. Moreover, the methodology automatically selects the columns with the highest misclassification densities as problematic features. Subsequently, XMLAD optimizes an anomaly classification engine that recognizes the features inducing performance anomalies.
3. We evaluate the performance of the proposed methodology on data collected by drive tests over real networks. Specifically, we test the XMLAD approach on three datasets by considering different test types such as HTTP file download, HTTP live page download, and QUIC file download.
4. Finally, we build a testbed that generates synthetic data to evaluate the performance of the proposed methodology. We use the testbed to evaluate the accuracy of the XMLAD under different conditions. We also describe the statistical models characterizing features that form synthetic datasets.

We implement the XMLAD methodology in Python and test its performance using real and synthetic datasets. More precisely, we use datasets with data obtained from measurements in operational networks: we have two datasets with data gathered by Nokia and one with our measurement operated through the MONROE platform [16]. Nokia datasets comprise hundreds of features but few experiments and they were collected by Nokia for auditing purposes in European countries. On the other hand, MONROE data is more abundant as it was acquired by means of an open-access platform for testing networks across Europe, although it contains much fewer features than Nokia's data. XMLAD is also tested using a synthetic dataset that we generated and which contains a known rate of anomalies. It is worth noting that the synthetic dataset contains thousands of experiments but only a very restricted number of key features.

Compared to our preliminary conference paper [1], this extended version introduces the procedure to automatically select those features with high classification errors as problematic features. Furthermore, the proposed methodology uses the proportional approach to select the number of output classes of the anomaly classification engine without resorting to a threshold search. In this sense, XMLAD is a significant extension of the KLNX methodology proposed in our conference paper, which justifies why we here we use a different name (i.e., XMLAD instead of KLNX).

Furthermore, here we include a comprehensive evaluation of the XMLAD methodology by means of the execution of multiple experiments on datasets profiling the performance of real mobile networks. Finally, this extended version extensively evaluates the impact of the various components of data preprocessing and training set preparation on the performance of the anomaly detection engine.

Table 1: Notation summary

Notation	Description
$\boldsymbol{D}$	Original dataset
$\boldsymbol{z}$	Target KPI vector
$\boldsymbol{y}$	Discretized KPI vector
$\mathcal{N}_1$	Features indices retained by the missing value detection
$\mathcal{N}_2$	Features indices retained by the low variability detection
$\mathcal{N}_3$	Features indices retained by the outlier detection in features
$\mathcal{M}_4$	Row indices retained by the outlier detection in experiements
$\mathcal{N}_5$	Features indices retained by feature selection stage
$\boldsymbol{D}_1$	Output dataset of the missing value detection
$\boldsymbol{D}_2$	Output dataset of the low variability detection
$\boldsymbol{D}_3$	Output dataset of the outlier detection in features
$\boldsymbol{D}_4$	Input dataset of the outlier detection in experiments $[\boldsymbol{z}, \boldsymbol{D}_3]$
$\boldsymbol{T}$	Output dataset of the outlier detection in experiments
$\boldsymbol{C}$	Pearson correlation matrix for the feature selection stage
$\boldsymbol{C}'$	Projection of $\boldsymbol{C}$ into the 2D Euclidean space
$\boldsymbol{X}_{train}$	Input dataset to train the anomaly detection engine
$\boldsymbol{y}_{train}$	Output label vector to train the anomaly detection engine
$\boldsymbol{\Gamma}$	Training set of the anomaly detection engine $\{\boldsymbol{X}_{train}, \boldsymbol{y}_{train}\}$
$\boldsymbol{X}_{test}$	Input dataset to evaluate the anomaly detection engine
$\boldsymbol{y}_{pred}$	Predicted labels by the anomaly detection engine
$\boldsymbol{y}_{diff}$	Differences between predicted labels and discretized KPI values
$\boldsymbol{y}_{anm}$	Vector of anomalous scenarios
$\mathcal{N}_r$	Features indices retained by the relevant feature identification
$\boldsymbol{\chi}$	Input dataset to train the anomaly classification engine
$\boldsymbol{\zeta}$	Output label vector to train the anomaly classification engine
$\boldsymbol{\Pi}$	Training set of the anomaly classification engine $\{\boldsymbol{\chi}, \boldsymbol{\zeta}\}$
$f_\theta(\cdot)$	Function performed by the discretization model
$g_\phi(\cdot)$	Function performed by the anomaly detection engine
$h_\varphi(\cdot)$	Function performed by the anomaly classification engine

1.1. Related Work

Several anomaly detection methods have been designed in the context of communication networks. Methods based on signal processing tools typically address the anomaly detection problem in network traffic time series [17, 18, 19, 20]. Specifically, these methods consider a single feature (univariate) to detect anomalous spikes in traffic data and the designed processing structures are tailored to the data at hand. Supervised machine learning techniques have been also proposed to detect anomalies in traffic time series [21, 22] requiring ground truth labels that are not always available. Moreover, unsupervised machine learning techniques have been introduced [23, 24, 14] to identify spikes in traffic time series whose performance is severely affected by the hyperparameter tuning. In the context of performance anomalies in communication networks, machine learning architectures based on supervised models have been introduced [10, 11]. These methods use multiple features (multivariate) to identify performance anomalies. However, these approaches discard many available features by selecting manually a small set of basic features and they also require the knowledge of ground truth labels to train the supervised models [12, 13]. In addition, it can be observed that these supervised techniques usually do not provide information about features that can cause abnormal behaviors. Recently, an unsupervised method based on principal component analysis (PCA) and root cause analysis was presented to identify features causing anomalies [25]. This method manually selects the problematic features to build the finite state machine that searches anomaly root causes and the selected principal components hide the thresholds related to anomalies in a specific network aspect.

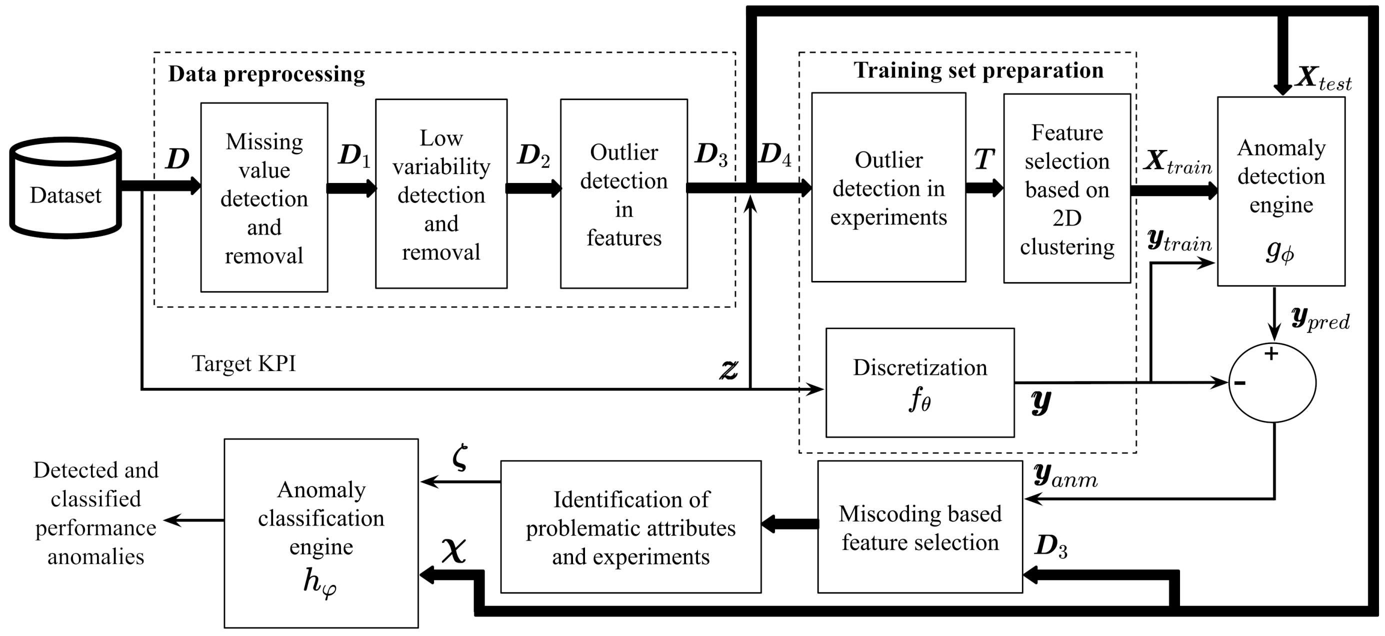
In contrast to the anomaly detection methods in traffic time series, the XMLAD methodology is a multivariate approach that detects and classifies anomalies in mobile networks considering the entire set of available features. Compared to previous machine learning methods [10, 12, 13, 11], the proposed approach includes a data cleaning stage that automatically extracts from the set of available variables those features related to a particular network task, e.g., HTTP File Download. Furthermore, the data cleaning stage is a nonparametric and unsupervised procedure with a self-tune of hyperparameters based on data statistics that identify features and experiments describing the normal network behavior. Instead of using a manual selection procedure of the problematic features as reported in [25], our framework embeds an unsupervised and nonparametric technique to automatically recognize features and experiments causing performance anomalies based on both the statistical feature modeling and the misclassification generated by the anomaly detection engine. Finally, this framework includes an anomaly classification stage that identifies in an unsupervised way the anomalous scenarios and the set of features causing these anomalies, which is a fundamentally novel contribution of our work.

1.2. Notations

In this work, we consider bold fonts for vectors and matrices, in lowercase and uppercase, respectively, e.g. $\boldsymbol{a}$ and $\boldsymbol{A}$. The i-th component of the vector $\boldsymbol{a}$ is denoted as $\boldsymbol{a}(i)$ and $\boldsymbol{A}(i, j)$ represents the entry of $\boldsymbol{A}$ at the location (i, j). In addition, the i-th row and the j-th column of the matrix $\boldsymbol{A}$ are denoted as $\boldsymbol{A}(i, :)$ and $\boldsymbol{A}(:, j)$, respectively. Consider $\boldsymbol{A}$ a matrix with dimensions $m \times n$, and subset $\mathcal{M} \subseteq \{1, \ldots, m\}$ of row indices and subset $\mathcal{N} \subseteq \{1, \ldots, n\}$ of column indices, then, $\boldsymbol{A}(\mathcal{M}, :)$ and $\boldsymbol{A}(:, \mathcal{N})$ are submatrices of $\boldsymbol{A}$ containing the rows included in $\mathcal{M}$ and the columns included in $\mathcal{N}$, respectively. F_V indicates the cumulative distribution function of the random variable V. The lower quartile, the upper quartile, and the interquartile range of the observation entries included in vector $\boldsymbol{a}$ are denoted by $Q_1^{(\boldsymbol{a})}$, $Q_3^{(\boldsymbol{a})}$, $\mathrm{IQR}(\boldsymbol{a})$, respectively. Table 1 shows a summary of notations used in this paper.

1.3. Paper organization

The paper is organized as follows. Section 2 describes the proposed anomaly detection and classification methodology. Section 3 describes the real datasets and introduces the modeling to generate synthetic data. The results of the proposed methodology are displayed in Section 4 for both synthetic data and real measurements. Finally, we summarize some concluding remarks in Section 5.



Data preprocessing
Dataset
D
Missing value detection and removal
D_1
Low variability detection and removal
D_2
Outlier detection in features
D_3
D_4
Training set preparation
Outlier detection in experiments
T
Feature selection based on 2D clustering
X_{train}
y_{train}
X_{test}
Anomaly detection engine g_ϕ
y_{pred}
Target KPI
z
Discretization f_θ
y
+
-
y_{anm}
Miscoding based feature selection
D_3
Identification of problematic attributes and experiments
ζ
$\mathcal{X}$
Anomaly classification engine h_φ
Detected and classified performance anomalies

Figure 1: Flowchart of the XMLAD methodology.

2. The XMLAD methodology

The XMLAD methodology is a three-phase, unsupervised, and nonparametric approach for detecting and classifying performance anomalies in mobile networks. This section presents the preparation of the dataset to model the anomaly detection engine, which is the first phase of the proposed methodology. The second and third phases focus on optimizing detection and classification engines. The rationale behind the design of the three phases is that a clean model of the dataset's regularities will need to be constructed in order to identify and classify anomalies.

The first step of the XMLAD approach is the loading of the database containing performance indicators (features) in operation scenarios (experiments). This database also includes the target KPI vector. Then, XMLAD applies a preprocessing stage to the loaded data in order to obtain the cleanest dataset for training an anomaly detection engine, also known as the *knowledge tree.* Fig. 1 illustrates the flowchart of the XMLAD methodology. Notice that arrows can have two line widths, thick lines for matrices or datasets with multiple features and thin lines for vectors. As can be seen, the modeling of the anomaly detection engine is divided into three subphases: (i) data preprocessing, (ii) training set preparation, and (iii) knowledge tree training.

2.1. Data preprocessing

As shown in Fig. 1, this subphase consists of three steps: null cell detection and removal, identification and removal of features with low variability, and detection and removal of features with a large number of outliers. More precisely, the first step of this subphase is to remove features that contain a large number of missing values. The second step involves discarding features with low variability in their samples. As a final step, this subphase removes features that have a large number of outliers or gross errors.

2.1.1. Missing value detection and removal

Drive tests typically collect data related to different test types. However, when drive tests run a specific test, several operation indicators are not stored. Consequently, it is necessary the development of practical tools to extract the relevant features for a particular test type. Specifically, this module recovers the features recorded for a specific task by identifying and removing columns with many null cells. For this purpose, let $\boldsymbol{D}$ be a matrix with dimensions $m_0 \times n_0$, which contains the data extracted from the loaded dataset, where m_0 represents the number of experiments and n_0 represents the number of features. Observe that the target KPI has not been included in $\boldsymbol{D}$. In this process, rows are removed from $\boldsymbol{D}$ in order to generate an auxiliary matrix $\boldsymbol{D}'$, which will later be used to remove features. To do so, the methodology determines the number of missing values or null cells (for example, `NaN` in numeric arrays) at each row of $\boldsymbol{D}$ as follows:

$$\boldsymbol{d}(i) = \sum_{j=1}^{n_0} \text{isnull}(\boldsymbol{D}(i,j)) \quad \text{for } i = 1, \ldots, m_0, \tag{1}$$

where $\boldsymbol{d}$ is an m_0-dimensional vector whose i-th element contains the number of null cells in the i-th row of $\boldsymbol{D}$, and isnull$(\cdot)$ is a nonlinear function that detects null cells.

Afterward, this step extracts from $\boldsymbol{d}$ the row indices $\mathcal{M}_1 \subseteq \{1, \ldots, m_0\}$ whose number of null cells is less than a threshold defined by median$(\boldsymbol{d})$ + 3MAD$(\boldsymbol{d})$ [26], where median$(\boldsymbol{d})$ denotes the sample median of the components included in $\boldsymbol{d}$, and MAD$(\boldsymbol{d})$ stands for the median of the absolute deviations defined as

$$\mathrm{MAD}(\boldsymbol{d}) = \rho^{(\boldsymbol{d})} \left(\mathrm{median} \left(|\boldsymbol{d} - \mathrm{median}(\boldsymbol{d})| \right) \right), \tag{2}$$

with $\rho^{(\boldsymbol{d})}$ as a scalar parameter, whose value is determined directly from the data as follows [27]:

$$\rho^{(\boldsymbol{d})} = \frac{1}{\left(\boldsymbol{Q}_3^{(\bar{\boldsymbol{d}})} - \mu_{\bar{\boldsymbol{d}}} \right)}, \tag{3}$$

where $\bar{\boldsymbol{d}}$ is a scaled version of $\boldsymbol{d}$, i.e., $\bar{\boldsymbol{d}} = \boldsymbol{d}/\sigma_{\boldsymbol{d}}$, $\sigma_{\boldsymbol{d}}$ denotes the sample standard deviation of the observations included in $\boldsymbol{d}$, and $\mu_{\bar{\boldsymbol{d}}}$ represents the sample mean of the scaled version of $\boldsymbol{d}$. Hence, the subset $\mathcal{M}_1$ is given by

$$\mathcal{M}_1 = \{k \in \{1, \ldots, m_0\} | \boldsymbol{d}(k) < \mathrm{median}(\boldsymbol{d}) + 3\mathrm{MAD}(\boldsymbol{d})\}. \tag{4}$$

Next, the submatrix $\boldsymbol{D}'$ is extracted from the original dataset, which contains samples belonging to the subset of row indices $\mathcal{M}_1$, i.e., $\boldsymbol{D}' = \boldsymbol{D}(\mathcal{M}_1, :)$.

Then, the procedure identifies and retains only those columns in $\boldsymbol{D}'$ that do not contain missing values. This step obtains the subset of column indices as follows:

$$\mathcal{N}_1 = \left\{ k \in \{1, \ldots, n_0\} \middle| \sum_{i \in \mathcal{M}_1} \mathrm{isnull}(\boldsymbol{D}'(i,k)) = 0 \right\}. \tag{5}$$

Finally, a submatrix $\boldsymbol{D}_1 = \boldsymbol{D}(:, \mathcal{N}_1)$ is obtained from the original dataset $\boldsymbol{D}$, containing the features in $\mathcal{N}_1$.

2.1.2. Low variability detection

After discarding features with a large number of missing values, the approach aims at detecting features with low variability in their samples. These features do not contribute to the performance improvement of the anomaly detection model. Notice that features with zero variability are easily detected by computing the standard deviation of every measurement set. However, the criterion for detecting features with low variability can be set in a more flexible manner. In particular, we use the interquartile range (IQR) as an indicator of low variability. The IQR is defined as the difference between the upper quartile and the lower quartile, i.e., $\mathrm{IQR}(\boldsymbol{a}) = Q_3^{(\boldsymbol{a})} - Q_1^{(\boldsymbol{a})}$. Therefore, a feature vector with a zero-valued IQR contains at least half of the samples equal to the median value. Since the anomaly detection model should be optimized around the central tendency of the features that, in turn, describe the network's normal behavior, we assume that a feature with zero-valued IQR makes a negligible contribution to the performance improvement of the detection model.

Consider $\boldsymbol{D}_1$, a matrix with dimensions $m_0 \times n_1$, where m_0 is the number of experiments and $n_1 = |\mathcal{N}_1|$ is the number of features extracted by the procedure described in Section 2.1.1. The purpose of this step is to extract the features from $\boldsymbol{D}_1$ whose IQR is greater than zero. Hence, the subset of selected indices consists of

$$\mathcal{N}_2 = \{k \in \mathcal{N}_1 | \mathrm{IQR}(\boldsymbol{D}_1(:,k)) \neq 0\}. \tag{6}$$

A submatrix is extracted from $\boldsymbol{D}_1$ containing the feature indices included in $\mathcal{N}_2$, i.e., $\boldsymbol{D}_2 = \boldsymbol{D}_1(:, \mathcal{N}_2)$.

2.1.3. Outlier detection in fetures

An outlier is a measurement that deviates significantly from the mass of samples. Outliers can occur in a dataset as a result of anomalous scenarios, sampling errors, and storing errors. Furthermore, the persistence of sampling errors leads to features with many outliers, which may introduce bias into the modeling of the anomaly detection engine. During this step of the data preprocessing process, features with a large number of outliers are discarded. In this regard, let $\boldsymbol{D}_2$ be the input matrix of size $m_0 \times |\mathcal{N}_2|$. In addition, consider $\boldsymbol{\theta} = \mathrm{outlier}(\boldsymbol{a})$ a vector with the same length as $\boldsymbol{a}$ whose i-th element $\boldsymbol{\theta}(i)$ indicates the presence of an outlier in the i-th element of $\boldsymbol{a}$, i.e.,

$$\boldsymbol{\theta}(i) = \begin{cases} 1, & \text{if } |a(i) - \mathrm{median}(\boldsymbol{a})| > 3\mathrm{MAD}(\boldsymbol{a}), \text{ and} \\ 0, & \text{otherwise,} \end{cases} \tag{7}$$

XMLAD builds a matrix whose j-th column contains the outlier locations at the respective j-th feature, i.e.,

$$\boldsymbol{B}'(:,j) = \mathrm{outlier}(\boldsymbol{D}_2(:,j)), \quad \forall j \in \mathcal{N}_2. \tag{8}$$

This step obtains the number of outliers in each feature in the following manner:

$$\boldsymbol{o}_c(j) = \sum_{i=1}^{m_0} \boldsymbol{B}'(i,j), \quad \forall j \in \mathcal{N}_2. \tag{9}$$

Afterward, the methodology extracts column indices with fewer outliers than threshold $\mathrm{median}(\boldsymbol{o}_c) + 3\mathrm{MAD}(\boldsymbol{o}_c)$, i.e.,

$$\mathcal{N}_3 = \{k \in \mathcal{N}_2 | \boldsymbol{o}_c(k) < \mathrm{median}(\boldsymbol{o}_c) + 3\mathrm{MAD}(\boldsymbol{o}_c)\}. \tag{10}$$

In this case, the features with a low number of outliers are obtained as $\boldsymbol{D}_3 = \boldsymbol{D}_2(:, \mathcal{N}_3)$. It is pertinent to note that the outlier detection stage does not rely on parameter tuning since it uses parameters extracted directly from feature statistics.

2.2. Training set preparation subphase

During the second subphase of the knowledge tree modeling, training data and training labels are extracted in order to build the knowledge tree. At this stage, the methodology concatenates the target KPI vector $\boldsymbol{z}$ to the matrix obtained in the previous stage, i.e. $\boldsymbol{D}_4 = [\boldsymbol{z}, \boldsymbol{D}_3]$, where $\boldsymbol{D}_4$ has dimensions $m_0 \times n_4$, with $\mathcal{N}_4 = \mathcal{N}_3 \cup \{z\}$ and $n_4 = |\mathcal{N}_4|$. First, this subphase detects outliers in the experiments (i.e., the rows of $\boldsymbol{D}_4$). After that, a clustering-based feature selection method is implemented. Lastly, training labels are generated by discretizing the target KPI.

2.2.1. Outlier detection in experiments

Throughout this study, we assume that samples around the central tendency of each feature are associated with the network's normal behavior. Thus, training samples with outliers can introduce bias into the knowledge model and should be avoided. For this purpose, XMLAD builds a matrix from $\boldsymbol{D}_4$ whose columns are given by

$$\boldsymbol{B}''(:,j) = \mathrm{outlier}(\boldsymbol{D}_4(:,j)), \quad \forall j \in \mathcal{N}_4. \tag{11}$$

Furthermore, the vector including the number of outliers in each row is described as

$$\boldsymbol{o}_r(i) = \sum_{j=1}^{n_4} \boldsymbol{B}''(i,j), \quad \text{for } i = 1, \ldots, m_0. \tag{12}$$

This step detects the subset of row indices with zero outliers, i.e., $\mathcal{M}_4 = \{k \in \{1, \ldots, m_0\} \mid \boldsymbol{o}_r(k) = 0\}$. Finally, the method obtains a submatrix from $\boldsymbol{D}_4$ that extracts the experiments belonging to the set of row indices $\mathcal{M}_4$, in other words, $\boldsymbol{T} = \boldsymbol{D}_4(\mathcal{M}_4, :)$. Observe that this stage removes outlier experiments from the target KPI and features.

2.2.2. Feature selection

Redundant information in features typically affects the modeling of machine learning engines. With an increase in the number of features, this problem becomes more pronounced. To overcome this limitation, we use a feature selection technique to extract a limited set of features from the data available. This technique belongs to the field of dimensionality reduction (DR), which extracts a subset of features following a selection rule [28].

To select the training features, consider $\boldsymbol{T}$ the input matrix with dimensions $m_4 \times n_4$, where $m_4 = |\mathcal{M}_4|$ stands for the number of experiments and n_4 represents the number of features. A correlation indicator matrix $\boldsymbol{C}'$ is built as follows:

$$\boldsymbol{C}' = \mathbf{1}_{n_4 \times n_4} - \text{abs}(\boldsymbol{C}), \tag{13}$$

where $\mathbf{1}_{n \times n}$ is a matrix of size $n \times n$ with one-valued entries, abs($\mathbf{A}$) is a matrix with the absolute value of the elements of $\boldsymbol{A}$, and $\boldsymbol{C}$ denotes the correlation matrix of size $n_4 \times n_4$. Specifically, each element of $\boldsymbol{C}$ at the location $(i,\ j)$ corresponds to the Pearson correlation coefficient between $\boldsymbol{T}(:, i)$ and $\boldsymbol{T}(:, j)$, in other words:

$$\boldsymbol{C}(i,j) = \frac{\text{cov}\left(\boldsymbol{T}(:, i), \boldsymbol{T}(:, j)\right)}{\sigma_{\boldsymbol{T}(:,i)}\sigma_{\boldsymbol{T}(:,j)}}, \tag{14}$$

for $i = 1, \ldots, n_4$ and $j = 1, \ldots, n_4$, where cov($\boldsymbol{a}, \boldsymbol{b}$) denotes the sample covariance of two n-dimensional sample vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, which is defined as $\text{cov}(\boldsymbol{a}, \boldsymbol{b}) = \frac{1}{n}\sum_{i=1}^{n}(a(i) - \mu_{\boldsymbol{a}})(b(i) - \mu_{\boldsymbol{b}})$.

In this stage, we focus on clustering features that are correlated. To achieve this goal, we leverage multidimensional scaling (MDS), which projects multidimensional points into a smaller space while maintaining a distance (dissimilarity) measure among them [29]. In essence, we use MDS to map features into a 2D Euclidean space from information embedded in $\boldsymbol{C}'$, which contains the reference distances. Therefore, the distance between points in Euclidean space reflects the correlation between them.

To select the training features, we include a model-based clustering method to the projected correlation points. Basically, a model-based clustering technique assumes that the points belonging to each cluster obey to a 2D Gaussian distribution, therefore, the projected set is modeled as a mixture of normal distributions. The clustering method includes an expectation-maximization (EM) algorithm to compute the maximum likelihood estimates describing each

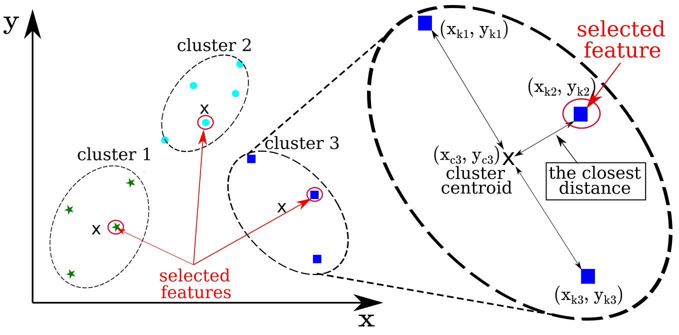


Figure 2: Representation of the feature selection procedure based on the 2D clustering technique.

Gaussian distribution. Moreover, the model-based clustering recognizes the optimal model according to the Bayes information criterion (BIC) [30]. Specifically, this stage evaluates various clustering models, each of which has a certain number of output clusters, and then, the proposed approach selects the model with the best BIC value. Notice that this study evaluates the number of output clusters within the range from 1 to $\lceil n_4/2 \rceil$ [31] where $\lceil a \rceil$ is the ceil function that returns the minimum integer greater than a.

During the training stage, features exhibiting a high correlation with the target KPI may introduce bias into the knowledge tree. Therefore, the decision structure may rely on factors that do not provide relevant information about the underlying processes affecting network performance. In this work, we first discard the cluster containing the target KPI. Subsequently, the methodology extracts the feature with the closest distance to the cluster centroid for the remaining clusters. Specifically, consider $(x_k,\ y_k)$ the location of the k-th feature projection and $(x_{c_i},\ y_{c_i})$ the coordinate of the cluster centroid, for $c_i = 1, \ldots, \mathcal{C}$, where $\mathcal{C}$ is the number of clusters. The set of selected features is determined as

$$\mathcal{N}_5 = \Big\{k \in \mathcal{N}_4 | \exists c_i \in [1, \mathcal{C}] : \\ k = \arg\min_{k' \in \mathcal{N}_4} \sqrt{(x_{k'} - x_{c_i})^2 + (y_{k'} - y_{c_i})^2}\Big\}. \tag{15}$$

Finally, the training set is obtained as $\boldsymbol{X}_{train} = \boldsymbol{T}(:, \mathcal{N}_5)$. Fig. 2 illustrates the feature selection procedure based on the 2D clustering technique.

2.2.3. Discretization

Target KPI samples typically exhibit floating point representations with values lying on an infinite set. Since our goal is to detect and classify anomalies, the proposed methodology discretizes the target KPI samples into a finite number of class labels. Each category should describe a particular operation scenario. As a result, the XMLAD builds a discretization model $f_\theta(\cdot)$ that estimates the binning intervals by considering a vector of training samples with continuous values. In this regard, consider the target KPI vector $\boldsymbol{z}$. Therefore, the training set of the discretization model can be defined as $\boldsymbol{z}' = \boldsymbol{z}(\mathcal{M}_4)$, i.e., $\boldsymbol{z}'$

contains the KPI samples belonging to the subset of indices $\mathcal{M}_4$, which is obtained in Section 2.2.1. Furthermore, the XMLAD resorts to proportional discretization to automatically determine the number of classes [32]. Under the proportional discretization approach, the number of classes depends on the size of the input vector, i.e., $w = \lfloor (\log_2 n)/2 \rfloor$, where w is the number of target categories, n is the input vector size, and $\lfloor a \rfloor$ is the floor operator that returns the integer part of a. In addition, discretization uses the k-means binning strategy to map the continuous KPI values into the discretized classes. In summary, the discretization strategy assigns the class labels based on the distance between the continuous entry and centroids yielded by the k-means clustering. Finally, the training labels are obtained from the KPI measurements as $\boldsymbol{y}_{train} = f_\theta(\boldsymbol{z}')$, where $f_\theta(\cdot)$ represents the nonlinear function implementing the discretization.

2.3. Knowledge model training subphase

This subphase aims at building a knowledge model from the extracted training sets to detect performance anomalies in mobile networks. We select the decision tree classifier to model the anomaly detection engine. A decision tree classifier is a supervised pattern recognition system that predicts the output classes based on simple decision rules considering the values of the input features. We select the decision tree classifier because the tree structure is explainable and easy to understand. Decision tree classifiers typically exhibit lower accuracies than those yielded by other black-box classification methods. Nevertheless, the tree structure allows the user to identify the features and thresholds that describe the output classes [33].

In order to optimize the decision tree classifier, we use the training set $\boldsymbol{\Gamma} = \{\boldsymbol{X}_{train},\, \boldsymbol{y}_{train}\}$, where $\boldsymbol{X}_{train}$ is the input training matrix of size $m_\ell \times n_\ell$, with m_ℓ as the number of training experiments and n_ℓ is the number of features extracted by the feature selection method based on 2D clustering. Furthermore, $\boldsymbol{y}_{train}$ consists of an m_ℓ-dimensional vector that contains the ground truth labels generated during the discretization process. The decision tree is trained using the classification and regression tree (CART) algorithm with the Gini impurity index as the loss function to be minimized [33]. Specifically, the Gini impurity index assesses the misclassification rate in a tree leaf, and it is computed as $G = 1 - \sum_{i=1}^{k} p_i^2$. In this expression, k represents the number of output labels, and p_i stands for the rate between the number of training labels assigned with the i-th class and the total number of samples evaluated by the tree leaf. When a tree level is aggregated, the number of training samples required to train a decision tree classifier doubles. Hence, we constrain the depth of the decision tree to $\lfloor (\log_2 m_\ell)/2 \rfloor$ to avoid overfitting. Additionally, the minimum number of training samples per leaf is limited to five [34].

Lastly, cost-complexity pruning is applied to the decision tree in order to reduce the likelihood of overfitting. Using a cost-complexity pruning algorithm, unnecessary tree nodes are eliminated by balancing the trade-off between classification error and model size. This algorithm minimizes a linear cost-complexity function with a parameter $\alpha \geq 0$, referred to as the complexity parameter, which controls the relative influence of the model complexity on the tree size. A cross-validation procedure is implemented to automatically select the appropriate α value from a set of effective values. A cross-validation procedure evaluates the cost-complexity function over an interval of α. For each value of α, the training set is divided into smaller subsets. Using this method, multiple training and testing operations are implemented, where one subset is used to train the model and the remaining ones to test it. Following this, the average classification accuracy is calculated. It is significant to note that a different tree structure is evaluated for each value of α. After cross-validation, the tree structure with the highest classification accuracy is selected. The decision tree classifier is fitted with the training set $\boldsymbol{\Gamma}$, the maximum number of levels (depth), the minimum number of training samples per leaf, and the optimal α. The function performed by the knowledge tree over the input data is denoted as $g_{\boldsymbol{\phi}}(\cdot)$.

2.4. Network performance anomalies

The XMLAD methodology tests the knowledge model after the training stage in order to detect anomalies in performance. Accordingly, the proposed approach utilizes all experiments to evaluate the decision tree classifier. More precisely, the set of test samples is obtained as $\boldsymbol{X}_{test} = \boldsymbol{D}(:, \mathcal{N}_5)$, where $\boldsymbol{X}_{test}$ is a submatrix of size $m_0 \times |\mathcal{N}_5|$ that extracts the variables identified by the feature selection method directly from the loaded data. Then, the anomaly detection engine is tested to predict the output classes, i.e., $\boldsymbol{y}_{pred} = g_{\boldsymbol{\phi}}(\boldsymbol{X}_{test})$. Our approach implements the discretization process $f_\theta(\cdot)$ to obtain the reference labels, i.e., this method determines $\boldsymbol{y} = f_\theta(\boldsymbol{z})$. Note that the anomaly detection model has been optimized to identify "normal" operation scenarios. Thus, misclassified experiments may be considered anomalous. In addition, the differences between the predicted labels and the discretized KPIs can provide information about the deviation from the normal operation of the samples under test. Next, we introduce the steps of the anomaly detection procedure.

2.4.1. Identification of anomalous scenarios

This stage estimates the element-wise difference between the set of class labels predicted by the anomaly detection model and the set of discretized KPIs, i.e., $\boldsymbol{y}_{diff} = \boldsymbol{y}_{pred} - \boldsymbol{y}$. In this context, three scenarios are identified: (i) the label predicted by the knowledge tree matches with respect to the corresponding discretized KPI, i.e., $\boldsymbol{y}_{diff}(i) = 0$; (ii) the class label predicted indicates a worse performance compared to the discretized KPI, i.e., $\boldsymbol{y}_{diff}(i) > 0$; or (iii) the performance predicted is better than the discretized KPI, i.e., $\boldsymbol{y}_{diff}(i) < 0$. Since this work focuses on detecting and classifying scenarios in which the performance observed is worse than predicted, our attention is oriented to scenario (iii). Therefore, the XMLAD methodology builds a vector of *errors* for anomalous scenarios as $\boldsymbol{y}_{anm}(i) = \boldsymbol{y}_{diff}(i)$ if $\boldsymbol{y}_{diff}(i) < 0$ (and $\boldsymbol{y}_{anm}(i) = 0$ otherwise), for $i = 1, \ldots, m_0$.

2.4.2. Identification of relevant features

Observe that the training stage considers a reduced set of features to optimize the anomaly detection model $g_{\boldsymbol{\phi}}$. Nevertheless, the methodology requires the evaluation of a large number of features to build the anomaly classification model. Hence, this phase recovers the features discarded by the preprocessing phase. In essence, this phase retrieves the matrix $\boldsymbol{D}_3$ outputted by the outlier detection in features (Section 2.1.3) with dimensions $m_0 \times |\mathcal{N}_3|$.

To quantify the relevance of each feature with respect to the vector of anomalous scenarios $\boldsymbol{y}_{anm}$, we compute the miscoding (mscd) index. This metric essentially estimates the amount of information sharing a feature vector (in our case is the i-th column of $\boldsymbol{D}_3$, i.e., $\boldsymbol{D}_3(:, i)$) and a target vector $\boldsymbol{y}_{anm}$. In other words, the miscoding assesses how much a feature vector can describe the target vector pattern. The miscoding has been recently proposed in [35] to select the relevant features that better describe a misclassification behavior. To compute this metric, consider the matrix $\boldsymbol{D}_3$ whose $|\mathcal{N}_3|$ columns correspond to the input features. Therefore, the miscoding index between the i-th feature $\boldsymbol{D}_3(:, i)$ and the vector of anomalous scenarios $\boldsymbol{y}_{anm}$ is estimated by

$$\text{mscd}(\boldsymbol{D}_3(:, i), \boldsymbol{y}_{anm}) = \frac{1 - \text{NCD}(\boldsymbol{D}_3(:, i), \boldsymbol{y}_{anm})}{\sum_{i=1}^{m_0} (1 - \text{NCD}(\boldsymbol{D}_3(:, i), \boldsymbol{y}_{anm}))}, \quad (16)$$

for $i = 1, \ldots, |\mathcal{N}_3|$, with $\text{NCD}(\boldsymbol{a}, \boldsymbol{b})$ as the normalized compression distance between vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. The miscoding estimates are sorted in decreasing order such that features that contribute more to the anomaly detection are firstly selected. In this sense, the number of selected features to evaluate the anomaly classification engine is given by $n_r = \min(|\mathcal{N}_3|, \lfloor \log_2(m_0)/2 \rfloor^2)$. Specifically, let $\boldsymbol{e}$ be the vector of sorted miscoding estimates such that $\boldsymbol{e}(0) = \text{mscd}(\boldsymbol{D}_3(:, i_0), \boldsymbol{y}_{anm})$, $\boldsymbol{e}(1) = \text{mscd}(\boldsymbol{D}_3(:, i_1), \boldsymbol{y}_{anm})$, $\ldots$, $\boldsymbol{e}(n_r) = \text{mscd}(\boldsymbol{D}_3(:, i_{n_r}), \boldsymbol{y}_{anm})$ and $\boldsymbol{e}(0) \geq \boldsymbol{e}(1) \geq \ldots \geq \boldsymbol{e}(n_r)$. Hence, the subset of relevant feature indices can be written as

$$\mathcal{N}_r = \{k \in \{1, \ldots, n_1\} | k = i_u, \text{ for } u = 0, 1, \ldots, n_r\}. \quad (17)$$

2.5. Classification tree modeling

This phase detects the features that exhibit a strong association with anomalies from relevant features. For this purpose, we apply a model-based 1D clustering to every relevant feature in order to identify the feature intervals that can contain anomalous scenarios. Then, a procedure that detects problematic features and experiments is developed to generate the training labels of the anomaly classification engine.

2.5.1. Detection of problematic features and experiments

Since anomalous scenarios are related to unusual values generated by some network aspects, we focus on detecting feature intervals related to abnormal behaviors. To this end, the XMLAD methodology applies a 1D model-based clustering to every relevant feature. More precisely, we assume that the statistical model of each relevant feature obeys a mixture of Gaussian distributions with different parameters. In this case, model parameters are estimated using the EM algorithm and the number of clusters is selected in an unsupervised manner based on the BIC [30]. Before applying the 1D clustering, each relevant feature is scaled using standard normalization, i.e., $\hat{\boldsymbol{a}} = (\boldsymbol{a} - \mu_{\boldsymbol{a}})/\sigma_{\boldsymbol{a}}$, where $\hat{\boldsymbol{a}}$ is the normalized set, $\boldsymbol{a}$ is the original vector, $\mu_{\boldsymbol{a}}$ and $\sigma_{\boldsymbol{a}}$ are, respectively, the sample mean and the standard deviation of $\boldsymbol{a}$. The idea behind this procedure is to identify feature intervals with a high rate of misclassification with the goal of recognizing the problematic experiments.

Then, the XMLAD approach estimates the anomaly density of each cluster, i.e., the mean error in the cluster normalized to the overall mean error (cf. Section 2.4.1 for the definition of error). Assume that the clustering generates a vector $\boldsymbol{y}_{\text{clust}}$ with cluster labels $c_j = 1, \ldots, \mathcal{C}_J$, where $\mathcal{C}_J$ is the number of detected clusters. Thus, the anomaly density of the vector anomalous scenarios for each cluster ρ_{c_j} is obtained as the normalized average of the subvector $\boldsymbol{y}_{anm}(\mathcal{M}_{c_j})$, where $\mathcal{M}_{c_j}$ is a subset of row indices such that

$$\mathcal{M}_{c_j} = \{k \in \{1, \ldots, m_0\} | \boldsymbol{y}_{\text{clust}}(k) = c_j\}, \quad (18)$$

for $c_j = 1, \ldots, \mathcal{C}_J$. For each feature this step obtains a set of densities $\{\rho_{c_j}\}_{c_j=1}^{\mathcal{C}_J}$. Then, we implement a procedure to select those features with the highest anomaly densities. It is worth noting that these features are considered problematic features, i.e., they are variables associated with network aspects that could induce the anomaly. As a result, each anomalous scenario can be described by listing problematic features. Taking into account that a problematic feature can contain two types of experiments (problematic and nonproblematic), the number of categories that should detect the anomaly classification engine reduces to $\eta_{classes} = 2^{n_{prob}}$, where n_{prob} is the number of problematic features. To generate a classification model with high accuracy, we use the proportional approach to determine the number of problematic features that the classifier should consider, i.e., $n_{prob} = \lceil \log_2(\log_2(m_0)) \rceil$, where $\lceil a \rceil$ returns the least integer greater than or equal to a. Furthermore, at each problematic feature, we consider the samples in clusters with high anomaly densities as problematic experiments with a label `T$P`, where `$` is the ID assigned to the problematic feature. Otherwise, if experiments are in clusters with low anomaly densities, we consider them as nonproblematic samples, with a label `T$N`.

2.5.2. *Classification model training*

In this stage, we train the anomaly classification engine using the features used to train the knowledge tree and the relevant features selected to identify problematic features and experiments. In other words, we obtain a feature set with column indices $\mathcal{N}_c = \mathcal{N}_3 \cup \mathcal{N}_r$. Then, the input training set is obtained as a submatrix extracted from the original dataset $\boldsymbol{D}$ containing the column indices $\mathcal{N}_c$, i.e., $\boldsymbol{\chi} = \boldsymbol{D}(:, \mathcal{N}_c)$. The input training set attempts to consider the information that builds the first decision tree classifier g_ϕ and the knowledge embedded in the miscoding estimates.

In addition, the method builds the output training set $\boldsymbol{\zeta}$ from the labeling of problematic and non-problematic experiments, described in Section 2.5.1. This labeling generates two types of classes: (i) the `compliant` class that identifies the samples that do not contain any problematic feature, i.e., this class describes the network's normal behaviors, or (ii) the class of anomalous scenarios, which are labeled as a concatenated semicolon list of problematic and non-problematic features. For example, consider that the method detects four problematic features with number assignation:

- `packet.loss` $\rightarrow 1$
- `maximum.segment.size` $\rightarrow 2$
- `round.trip.time` $\rightarrow 3$
- `start.snr` $\rightarrow 4$

An experiment with class `T1P;T2N;T3N;T4P` identifies an anomalous scenario whose variables `packet.loss` and `start.snr` cause the abnormal functioning in the experiment of interest. Afterward, this stage trains a decision tree engine to model the anomaly classification engine. We use a decision tree classifier because the decision tree structures provide an explainable visualization tool that identifies features and thresholds related to normal and abnormal operative scenarios. In this case, the proposed approach optimizes the decision tree classifier by implementing the CART algorithm with the training set $\boldsymbol{\Pi} = \{\boldsymbol{\chi}, \boldsymbol{\zeta}\}$. The training stage also uses cost-complexity pruning and cross-validation to remove unnecessary complexities in the decision tree classifier. Finally, the non-linear function performed by the classification engine is represented with $h_\varphi(\cdot)$.

3. Datasets

In this paper, we evaluate the performance of the XMLAD methodology on both measurements collected by drive tests on real mobile networks and synthetic datasets. In order to recreate behaviors similar to those observed in real experiments, we first describe the datasets derived from real drive tests. Afterward, we introduce statistical models characterizing the synthetic measurements.

Table 2: Characteristics of the measurement subsets used to evaluate the performance of the XMLAD methodology.

Dataset	Test type	Rows	Cols	Task name
NokiaFL	HTTP FILE DL	1,730	442	HTTP 5MB Download
NokiaLP	HTTP LIVEPAGE DL	1,185	393	HTTPS Static Kepler 30s
MonroeQL	Data Exchange	3,951	188	Server-Client Exchange

3.1. *Nokia datasets*

Table 2 displays a summary of the characteristics of the real measurement sets used to evaluate the performance of the XMLAD methodology. As seen in this table, we consider three real datasets: the NokiaFL dataset, the NokiaLP dataset, and the MonroeQL dataset. In 2019, Nokia acquired the first two datasets with the aim of auditing the performance of 4G mobile networks in various European countries [36]. These datasets contain measurements of tens of features profiling different aspects of the mobile network such as TCP traffic, routing, and radio links, among others. Furthermore, NokiaFL and NokiaLP datasets provide the statistics of different network aspects including count, average, maximum, minimum, and percentiles. Hence, every row extracted from these datasets corresponds to a particular experiment collecting information on thousands of features such as date, location, test type, performance measurements, statistics, and KPI outcomes. It is worth noting that these datasets were collected by using a mobile testing device to generate the stimuli that induce the communication network response, hence, the dataset does not contain sensitive information about customers. Additionally, sensitive information about the mobile network operator was removed.

Firstly, the NokiaFL dataset contains information on the network performance when the testing device downloads a 5 MB file. For this dataset, we select the session duration as the target KPI. More precisely, the NokiaFL dataset includes those qualified experiments performed under the LTE technology and extracted the numerical features only. It can be observed in Table 2 that the NokiaFL measurement set contains 1,730 experiments and 442 features. On the other hand, the NokiaLP dataset contains the measurements when the mobile testing device fetches a live web page. Specifically, this dataset captures the mobile network profile while the test device fetches a static website. For this measurement set, we also extracted the numerical features and selected the session duration as the target KPI. As seen in Table 2, the NokiaLP dataset has dimensions of 1,185 rows and 393 columns.

3.2. *MonroeQL dataset*

In the MonroeQL dataset, we derive the data collection from measurement campaigns conducted with MONROE, an open-access platform designed to evaluate mobile network performances across Europe [16]. Specifically, this dataset was collected to assess the performance of mobile networks that use protocols other than TCP. To be more

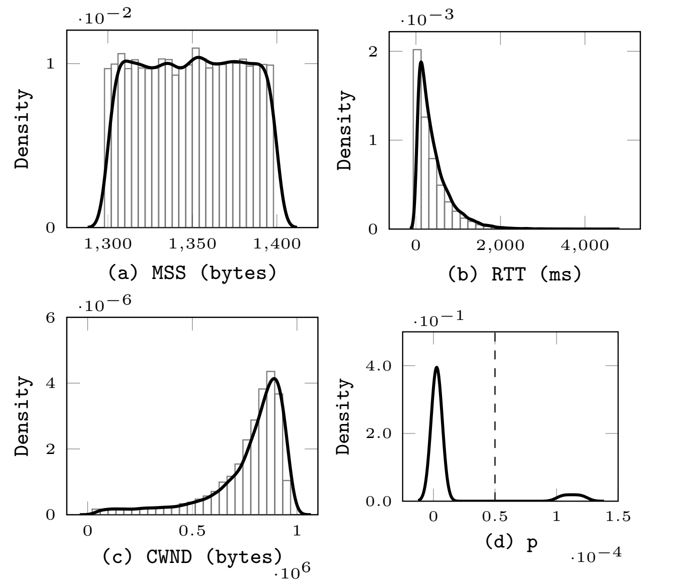


Figure 3: Histogram and kernel density estimation (KDE) of synthetic samples for the parameters (a) *MSS*, (b) *RTT*, (c) *CWND*, and (d) p (KDE only).

precise, the packets are sent over the Quick UDP Connections (QUIC) protocol, which provides the reliability of TCP and the speed of UDP [37].

In this work, we test the proposed methodology on the MonroeQL dataset, which contains the measurements during data exchange between a control server and a client in the MONROE network. In order to collect information for this dataset, mobile clients were located in Norway, Sweden, and Spain. Afterward, the drive test captured the data exchange information for different qualities of the network connectivity. Due to the difficulty in extracting information from QUIC environments, the raw dataset was stored in *qlog* format [38]. After that, the MonroeQL dataset was constructed from *qlog* files to obtain a measurement set formatted similar to that used for Nokia datasets [39]. This dataset includes measurements and statistics (mean, median, percentiles, minimum, maximum, etc.) of traffic variables such as round trip time, congestion window, session time, and throughput. As can be seen in Table 2, the MonroeQL dataset contains 188 features and 3,951 experiments.

3.3. Synthetic data modeling

To evaluate the behavior of the methodology in a more controllable way, we also build a testbed that randomly generates each feature vector with entries following a particular statistical model. Table 3 depicts the notation summary related to the synthetic data modeling. The TCP throughput B_{tcp} (in bytes/s) is then computed analytically using the simple Mathis model [40],

$$B_{tcp} = \min\left(\frac{MSS}{RTT\sqrt{p}}, \frac{CWND}{RTT}\right), \tag{19}$$

where *MSS* is the maximum segment size (in bytes), *RTT* is the round trip time, *CWND* is the congestion window,

Table 3: Notation summary for the synthetic data modeling

Notation	Description
B_{tcp}	TCP throughput
L_{tcp}	TCP session duration
MSS	Maximum segment size
RTT	Round-trip time
$CWND$	Congestion Window
p	Packet loss probability
f_{mss}	Probability density function of *MSS*
F_{RTT}	Cumulative density function of *RTT*
F_{cwnd}	Cumulative density function of *CWND*
F_p	Cumulative density function of p
lb_{mss}, ub_{mss}	Parameters of the uniform distribution f_{mss}
β	Rate of negative exponential distribution of F_{RTT}
RTT_{min}	Minimum RTT value
$\bar{\bar{\mu}}_{cwnd}, \sigma_{cwnd}$	Parameters of lognormal component of F_{cwnd}
lb_{cwnd}, ub_{cwnd}	Parameters of the uniform component of F_{cwnd}
ϵ_{cwnd}	Rate of lognormal samples of F_{cwnd}
μ_p, σ_p	Parameters of lognormal component of F_p
lb_p, ub_p	Parameters of uniform component of F_p
ϵ_p	Rate of packet loss probability outliers

and p is the packet loss probability. In addition, we set download size *FLSZ* to 5 MB in order to estimate the session duration as

$$L_{tcp} = \frac{FLSZ}{B_{tcp}}. \tag{20}$$

Notice that we generate just four TCP features, i.e., *MSS*, *RTT*, *CWND*, p, which greatly simplifies the possibility to illustrate and evaluate the operation of the XMLAD methodology.

Synthetic parameters are generated so as to resemble the behavior observed in the NokiaFL dataset. The *MSS* is drawn from a uniform probability density function (pdf), i.e., $f_{mss} \sim u(lb_{mss}, ub_{mss})$, with bounds $lb_{mss} = 1300$ bytes and $ub_{mss} = 1400$ bytes. Fig. 3(a) displays the histogram and the kernel density estimation (KDE) obtained from an *MSS* synthetic vector with 20,000 samples. The KDE curve is estimated using a Gaussian kernel [41]. *RTT* samples are obtained from a shifted exponential distribution, i.e.:

$$F_{RTT} = \text{Exponential}(\beta) * RTT_{min}, \tag{21}$$

where the convolution operator '$*$' simply tells that we sum a constant minimum value $RTT_{min} = 30$ms to the one generated with a negative exponential distribution with rate $\beta = 400(\text{ms})^{-1}$ (cf. Fig. 3(b)). To generate the synthetic samples of *CWND* and p, we use a ϵ-contaminated mixture model [27] with a uniform and a (shifted and mirrored) log-normal distribution, i.e., with CDF given by

$$\begin{aligned} F_{cwnd}(x) = \epsilon_{cwnd}&\text{Lognormal}\left(\mu_{cwnd}, \sigma_{cwnd}\right)\left(1.0 \cdot 10^6 - x\right) \\ &* \left(1 - \epsilon_{cwnd}\right) u\left(lb_{cwnd}, ub_{cwnd}\right)(x), \end{aligned} \tag{22}$$

where the log-normal distribution has mean μ_{cwnd} and standard deviation σ_{cwnd}, and is mirrored around the value of 1 MB (cf. Fig. 3(c)), while $u\left(lb_{cwnd}, ub_{cwnd}\right)$ denotes the uniform distribution with bounds $lb_{cwnd} = 40$ kB and $ub_{cwnd} = 700$ kB. The contamination parameter ϵ_{cwnd} is set to 0.90, $\sigma_{cwnd} = 0.65$, and $\mu_{cwdn} = \log(\bar{\bar{\mu}}_{cwnd}) + \sigma^2_{cwnd}$, where $\bar{\bar{\mu}}_{cwdn} = 0.1$ MB is the desired mode of the lognormal.

So far, we have described synthetic features reproducing what is observed in the NokiaFL dataset. Additionally,

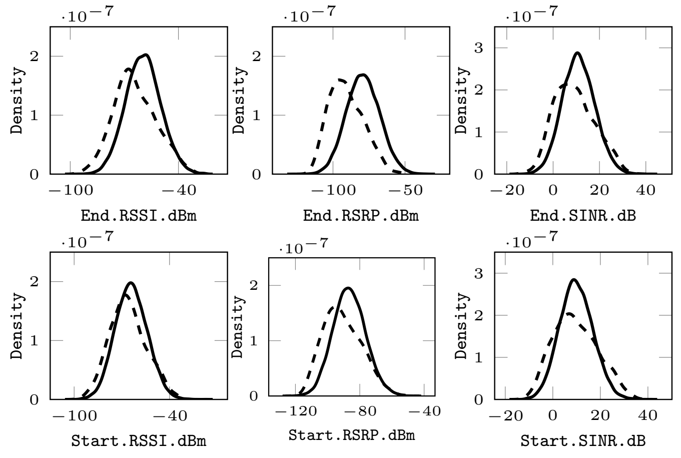


Figure 4: KDE yielded by different radio parameters for both the Nokia dataset (dashed lines) and the synthetic data (continuous lines).

we select the packet loss probability (p) to induce anomalous scenarios in a controllable way with a mixture model, i.e.:

$$F_p(x) = (1-\epsilon_p)\text{Lognormal}\left(\mu_p, \sigma_p\right)(x) * \epsilon_p u\left(lb_p, ub_p\right)(x), \quad (23)$$

where the mode of the log-normal pdf is set to $\bar{\bar{\mu}}_p = 2.50 \times 10^{-6}$, $\sigma_p = 0.20$, $lb_p = 1.00 \times 10^{-4}$, and $lb_p = 1.25 \times 10^{-4}$. Upon a closer look at the mixed model (23), it can be observed that a specific rate of samples ϵ_p exhibits much higher packet loss probabilities, that should be identified as anomalous. Fig. 3(d) shows the KDE of the synthetic packet loss probabilities obtained for $\epsilon_p = 0.10$. The synthetic dataset also includes six radio parameters. These features are generated so as to be correlated with the TCP throughput B_{tcp}, and the corresponding models rely on a Poisson distribution. The KDE curves obtained for the various radio parameters for the NokiaFL dataset and a synthetic dataset are displayed in Fig. 4.

In the synthetic dataset, anomalies are forced by means of abnormal values of the loss probability, which are generated with probability ϵ_p. We have also tested other synthetic datasets in which (part of the) anomalies are introduced by altering the value of a parameter *after* having generated the KPI. This corresponds to scenarios in which a parameter is measured incorrectly or, in general, it does not fully correspond to the target KPI for some random reason. However, due to space limitations, in what follows we only comment on anomalies introduced in p with the ϵ-contaminated model described above, whose results are also simpler to interpret.

4. Results and analysis

In this study, the XMLAD methodology was implemented in Python using the Scikit-learn [42] and the Nescience libraries[1]. Specifically, Nescience is an open-source library that contains routines computing the miscoding metric. In this section, we evaluate the XMLAD methodology in detail using the NokiaFL dataset. In addition, we examine the impact of the data cleaning included in the data preprocessing and the training set preparation on the detection performance of the proposed methodology. Then, we analyze the decision structures generated by anomaly detection and classification engines using the NokiaLP and MonroeQL datasets. Finally, we analyze the performance of the XMLAD methodology using synthetic datasets.

4.1. NokiaFL dataset

The NokiaFL dataset contains the network profiling when a mobile testing device downloads a 5MB file. For this dataset, we select the *session duration* as the target KPI. First, the XMLAD methodology applies the data preprocessing stage to remove columns with many null cells, low column variabilities, and outliers. Table 4 shows the sizes of data matrices $\boldsymbol{D}_1$, $\boldsymbol{D}_2$, and $\boldsymbol{D}_3$ yielded by the missing value detection and removal, low variability detection, and the outlier detection in features, respectively. We also include the size of the original NokiaFL dataset for comparison purposes. Notice that the missing value detection discards most of the features 384 (86.88% of the original number of columns). In drive tests conducted on mobile networks, if an experiment runs a particular test type, e.g, HTTP file download, several parameters related to other test types are neither measured nor aggregated. Consequently, the data matrix for this test type would contain a large number of columns with many null cells. Thus, the missing value detection allows the extraction of the information associated with a particular test type. It can be also observed that the low variability detection and the outlier detection in features remove 28 (6.33%) and 14 (3.17%) columns, respectively. Table 5 displays the list of columns extracted by the data preprocessing stage and a brief description of each feature. Observe that a refers to the testing mobile device and b refers to the HTTP server hosting the download file. Furthermore, selective acknowledgment, reference signal received power, received signal strength indicator, and signal-to-interference plus noise ratio are denoted using SACK, RSRP, RSSI, and SINR, respectively.

In the training set preparation, the outlier detection in experiments yields the data matrix $\boldsymbol{T}$ with dimensions 1,385 rows and 26 columns (see the last row of Table 4). This set of rows contains the scenarios describing the normal network behavior. Afterward, the proposed methodology implements the feature selection technique that projects the correlation matrix $\boldsymbol{C}$ into the Euclidean

[1]The source code for the proposed methodology is available at the following URL: `https://github.com/GCGImdea/NetPredict-Public/CommNetworks2022`.

Table 4: Dimensions of the data matrices yielded by different subphases of the XMLAD methodology for the NokiaFL dataset.

XMLAD Subphase	Data matrix	Rows	Cols
Original input	$\boldsymbol{D}$	1,730	442
Missing value detection	$\boldsymbol{D}_1$	1,730	58
Low variability detection	$\boldsymbol{D}_2$	1,730	30
Outlier detection in features	$\boldsymbol{D}_3$	1,730	26
Outlier detection in experiments	$\boldsymbol{T}$	1,385	26
Feature selection	$\boldsymbol{X}_{train}$	1,385	12

Table 5: Description of the features outputted by the data preprocessing stage for the NokiaFL dataset. In this dataset, `a` denotes the testing mobile device and `b` indicates the HTTP server.

#	**Feature**	**Description**
1	`abs.packetlost.sum`	Sum of packets lost during the test
2	`ack.pkts.sent.a2b`	ACK packets sent from a to b
3	`actual.data.bytes.b2a`	Data transmitted from b to a
4	`actual.data.pkts.b2a`	Packets transmitted from b to a
5	`avg.segm.size.b2a`	Average segment size from b to a
6	`avg.win.adv.a2b`	Average receiving window from b to a
7	`dsack.pkts.sent.a2b`	Duplicated SACK packets sent from b to a
8	`duplicate.acks.a2b`	Duplicated ACK packets from a to b
9	`end.rsrp.dbm`	RSRP in dBm at the end of the test
10	`end.rssi.dbm`	RSSI in dBm at the end of the test
11	`end.sinr.db`	SINR in dB at the end of the test
12	`initial.window.bytes.b2a`	Initial window size in bytes from b to a
13	`max.win.adv.a2b`	Maximum receiving window from a to b
14	`outoforder.pkts.b2a`	Out of order packets from b to a
15	`pushed.data.pkts.b2a`	Pushed data packets from b to a
16	`rexmt.data.bytes.b2a`	Retransmitted data packets from b to a
17	`rexmt.data.pkts.b2a`	Retransmitted data bytes from b to a
18	`rtt.from.3whs.a2b`	3-way handshaking round trip time
19	`start.rsrp.dbm`	RSRP in dBm at the start of the test
20	`start.rssi.dbm`	RSSI in dBm at the start of the test
21	`start.sinr.dbm`	SINR in dB at the start of the test
22	`tcp.first.sec.volume.dl`	Volume downlink during the first second
23	`tcp.first.sec.volume.ul`	Volume uplink during the first second
24	`time.to.first.byte.s`	Time to the receiving of the first data byte
25	`truncated.data.b2a`	Truncated data from b to a
26	`unique.bytes.sent.b2a`	Unique bytes sent from b to a

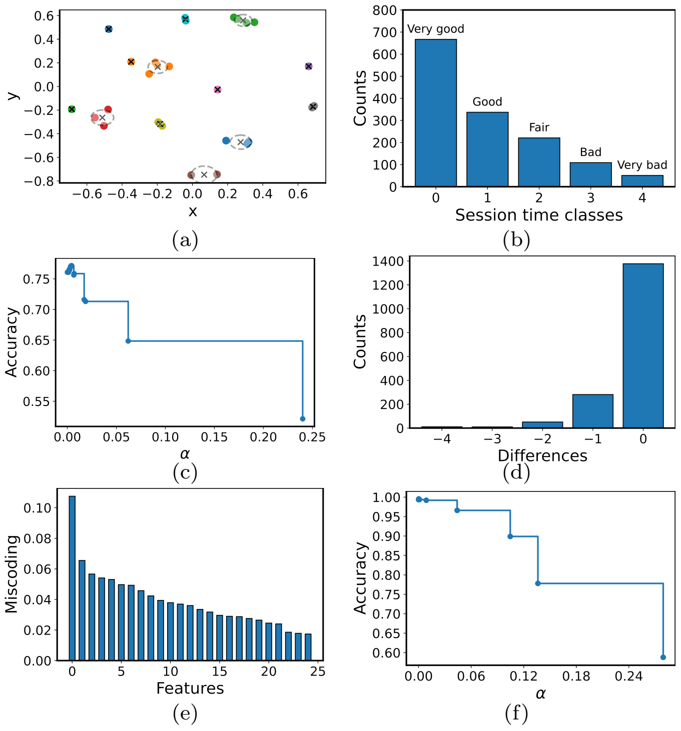


Figure 5: (a) Clustering on the projections of the correlation matrix $\boldsymbol{C}$ into 2D Euclidean space obtained by the model-based technique for the NokiaFL measurement set. (b) Histogram of the training labels obtained by the discretization model for the NokiaFL dataset. (c) Accuracy versus α exhibited by the anomaly detection model during the pruning process. (d) Histogram of the vector of anomalous scenarios $\boldsymbol{y}_{anm}$. (e) Sorted barplot of the miscoding between every column of $\boldsymbol{D}_3$ and $\boldsymbol{y}_{anm}$. (f) Accuracy versus α exhibited by the anomaly classification model during the pruning process.

space. The feature selection method also runs the model-based 2D clustering algorithm to reduce the number of variables and remove the features highly correlated with the target KPI. Fig. 5(a) displays the clusters detected by the model-based technique from the projection of the correlation matrix $\boldsymbol{C}$ into the 2D space (cf. Section 2.2.2). This figure also shows centroids and the ellipses of the corresponding covariance matrices. Recall that the feature selection stage removes the cluster containing the target KPI. In addition, the methodology selects the feature that is closest to the centroid in each of the remaining clusters. Specifically, the features selected are listed below:

1. `rexmt.data.bytes.b2a`,
2. `tcp.first.sec.volume.dl`,
3. `end.rssi.dbm`,
4. `unique.bytes.sent.b2a`,
5. `rtt.from.3whs.a2b`,
6. `pushed.data.pkts.b2a`,
7. `duplicate.acks.a2b`,
8. `abs.packetlost.sum`,
9. `avg_win_adv_a2b`,
10. `end.sinr.db`,
11. `initial.window.bytes.b2a`,
12. `tcp.first.sec.volume.ul`.

Notice that the dimensions of the training sample set are $m_\ell = 1,385$ experiments and $n_\ell = 12$ features. Furthermore, the methodology applies the discretization process to the target KPI. In this case, the number of class labels is determined by $w = \lfloor (\log_2 m_\ell)/2 \rfloor = 5$. Fig. 5(b) displays the histogram of the training labels outputted by the discretization process. Then, the XMLAD approach optimizes the knowledge tree $g_{\boldsymbol{\phi}}(\cdot)$ with the training set $\boldsymbol{\Gamma} = \{\boldsymbol{X}_{train}, \boldsymbol{y}_{train}\}$. To this end, the decision tree pruning process first generates a set of effective values of the regularization parameter α such that the tree complexity is reduced as the value of α increases. In this work, we select the value of α that generates the knowledge tree with the best classification accuracy. The curve of the accuracy versus α, exhibited by the decision tree during the pruning process, is illustrated in Fig. 3(c). As can be observed in this figure, the pruning process selects the decision tree with a classification accuracy of over 0.75.

Once built the knowledge model with the best α and the training set $\boldsymbol{\Gamma} = \{\boldsymbol{X}_{train}, \boldsymbol{y}_{train}\}$, we obtain the decision tree structure whose flowchart is depicted in Fig. 7. As can be observed in this figure, the knowledge tree learns parameters and thresholds that describe scenarios within the network's normal behavior. For every experiment, the decision tree first compares the `tcp.first.sec.volume.dl` with respect to the threshold $3,071,795$ at the root node. If the inequality rule is true, the decision tree moves to the left side in the next tree level, otherwise, the structure moves to the right side. For example, the knowledge tree assigns the "Good" label to scenarios with `tcp.`

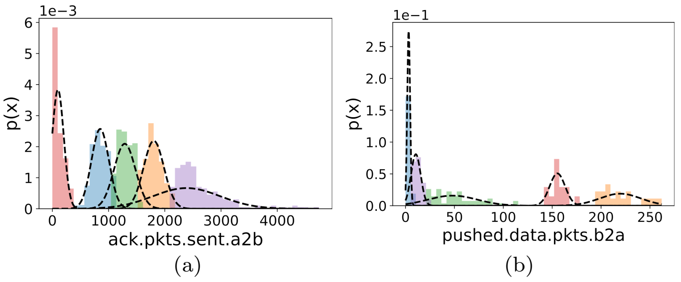


Figure 6: Histograms of feature clusters obtained by the model-based technique for two variables: (a) `ack.pkts.sent.a2b`, and (b) `pushed.data.pkts.b2a`. The dashed lines show the probability density functions fitting for clusters using the parameters estimated by the EM algorithm.

`first.sec.volume.dl` greater than $1,665,557$ or lower or equal to $3,071,795$ bytes. If we analyze in detail the decision structure of the knowledge tree in Fig. 7, we observe that four features describe the network behavior: (a) `tcp.first.sec.volume.dl`, (b) `tcp.first.sec.volume.ul`, (c) `duplicate.acks.a2b`, and (d) `avg.win.adv.a2b`. These features characterize different key behaviors in an HTTP Transfer Download. More precisely, `tcp.first.sec.volume.dl` describes the initial delays during the first second of the downlink. In TCP, the first second is usually limited by latency rather than network capacity. Higher values indicate better performances. Uplink direction volume should not be a relevant factor, it should contain the HTTP request length. In this case, it is detecting different URLs used for this test so performance differences are associated with the test server. In addition, `duplicate.acks.a2b` reflects packet loss and higher values should be associated with worse performance. Finally, `avg.segm.size.b2a` is the receiving window, normally very low values could limit performance but it is usually a consequence of the performance: higher speeds trigger higher RWIN. In this case, it also reflects differences in the terminal used for testing.

Afterward, the methodology tests the knowledge tree using the entire set of experiments. In this phase, we compute the vector of differences by subtracting the labels predicted by the knowledge tree and the labels estimated by the discretization of the target KPI. The histogram of the vector of anomalous scenarios $\boldsymbol{y}_{anm}$ is shown in Fig. 5(d). As can be observed in this figure, the number of detected anomalies decreases as the error between the predicted labels and the discretized KPIs increases. Note that each element of the vector of differences assesses the misclassification degree of the knowledge tree with respect to the corresponding discretized KPI. For instance, a difference value -4 indicates that the knowledge tree predicted a "Very good" performance when the respective discrete KPI exhibits a "Very bad" behavior. For this dataset, the knowledge tree detects 353 (20.40% of the original number of experiments) anomalies.

Then, the methodology considers both the matrix obtained at the outlier detection in features $\boldsymbol{D}_3$ and the vector of anomalies $\boldsymbol{y}_{anm}$ to identify the features that better characterize anomalous behaviors. To this end, we compute the miscoding (16) for each feature with respect to the vector of anomalies $\boldsymbol{y}_{anm}$. Fig. 5(e) shows the barplot of the miscoding values for the set of features in decreasing order. Specifically, the sorted set of features according to the miscoding metric is listed as follows:

1. `actual.data.pkts.b2a`
2. `time.to.first.byte.s`
3. `avg.segm.size.b2a`
4. `abs.packetlost.sum`
5. `ack.pkts.sent.a2b`
6. `duplicate.acks.a2b`
7. `outoforder.pkts.b2a`
8. `rexmt.data.pkts.b2a`
9. `pushed.data.pkts.b2a`
10. `rexmt.data.bytes.b2a`
11. `initial.window.bytes.b2a`
12. `truncated.data.b2a`
13. `actual.data.bytes.b2a`
14. `unique.bytes.sent.b2a`
15. `rtt.from.3whs.a2b`
16. `dsack.pkts.sent.a2b`
17. `end.sinr.db`
18. `avg.win.adv.a2b`
19. `start.sinr.dbm`
20. `tcp.first.sec.volume.dl`
21. `tcp.first.sec.volume.ul`
22. `max.win.adv.a2b`
23. `end.rssi.dbm`
24. `start.rssi.dbm`
25. `end.rsrp.dbm`

Subsequently, a model-based 1D clustering is applied to each feature are evaluated according to the order exhibited by the miscoding coefficients. Fig. 6 illustrates the normalized histograms of the clusters derived from the model-based clustering technique for two features: (a) `ack.pkts.sent.a2b`, and (b) `pushed.data.pkts.b2a`. Fig. 6 also shows the density curve fitting for each cluster using the parameters estimated by the EM algorithm. As observed in this figure, a feature can be represented as a mixture of Gaussian distributions with different means, where each distribution can be associated with a particular operative status of the corresponding network aspect. Moreover, the methodology applies the procedure based on the 1D clustering of every feature and the vector of anomalous scenarios to detect both problematic features and problematic experiments. In this case, the number of relevant features is determined by $w_p = \lfloor \log_2((\log_2 m)/2) \rfloor = 3$. For this dataset, XMLAD identifies the problematic features that are listed below:

1. `ack_pkts_sent_a2b`
2. `actual_data_bytes_b2a`
3. `tcp.first.sec.volume.dl`

Fig. 8 displays the scatter plots of $\boldsymbol{y}_{anm}$ versus the feature value for the three detected problematic columns. This figure also shows the clustering of each problematic feature into two classes: (T\$P) problematic experiments, and (T\$N) non-problematic experiments, where \$ indicates the number of the feature. As can be seen in this figure, the proposed approach detects as problematic scenarios those experiments with `ack.pkts.sent.a2b` greater than $2,000$ units, `ack.pkts.sent.a2b` a greater than $2,000$ and smaller than $4,000$ units, and `tcp.first.sec.volume.dl` smaller than 300 kB. Then, the methodology implements the CART algorithm to optimize the anomaly classification tree from the input training data $\boldsymbol{\chi}$ and the output training labels $\boldsymbol{\zeta}$. Specifically, the decision tree classifier learns the rules to detect anomalous scenarios and identify the features that can induce performance anomalies. In this stage, the methodology runs a pruning pro-

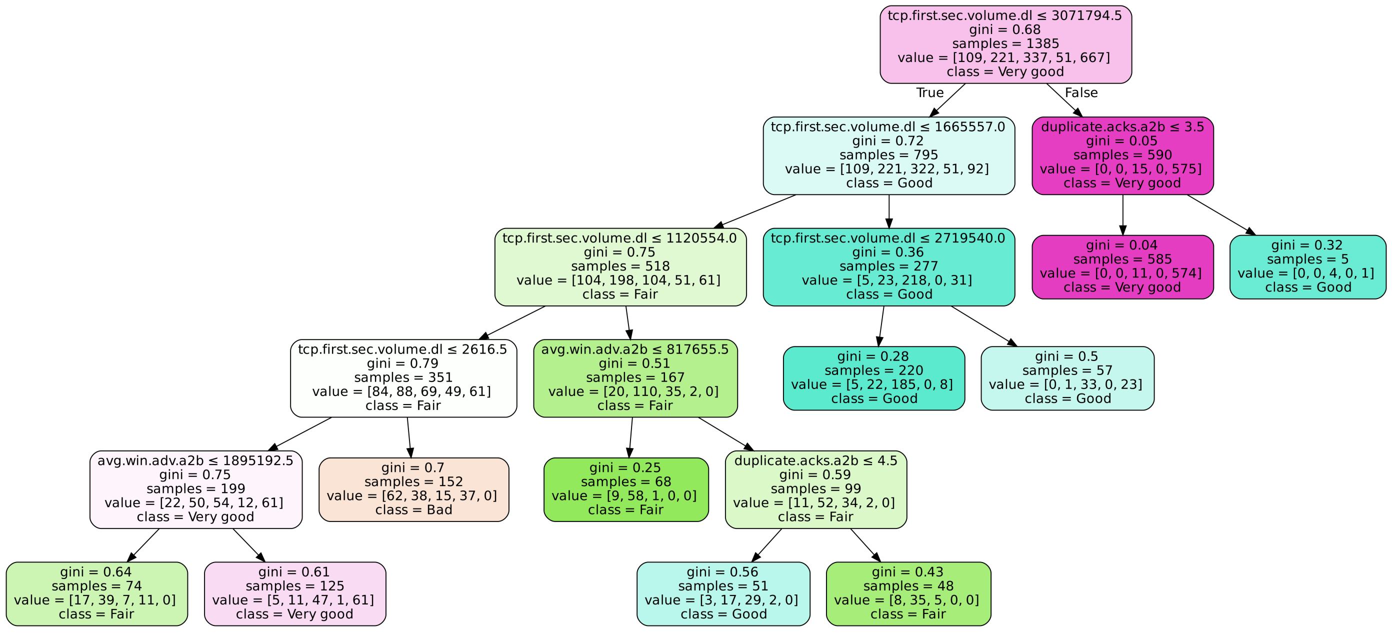


Figure 7: Schematic representation of the tree structure of the anomaly detection engine optimized for the NokiaFL dataset. Each decision tree node (leaf) contains the Gini impurity measure, the number of evaluated training samples, and the output class label.

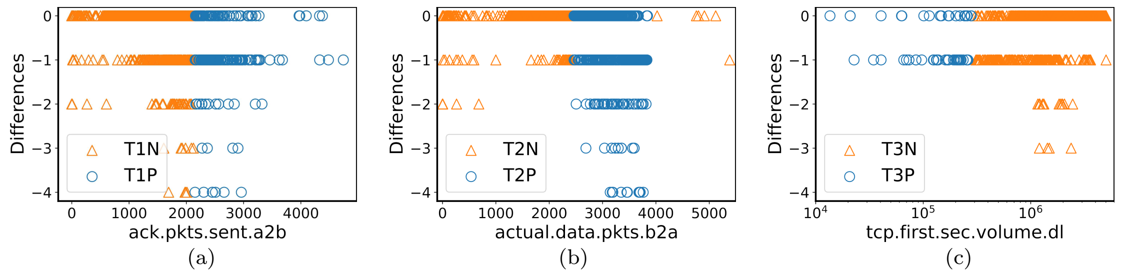


Figure 8: NokiaFL dataset. Scatter plots of the vector of anomalous scenarios $\boldsymbol{y}_{anm}$ versus the feature value for problematic features. Non-problematic experiments are labeled as T$N and problematic samples are categorized as T$N, where $ is the feature number.

cess to build the best classification tree. Fig. 5(f) displays the accuracy versus α exhibited by the anomaly classification model during the pruning process. Notice that the selected classification tree reaches a classification accuracy near 1.0. The flowchart of the decision tree structure for the classification model is depicted in Fig. 9. In this case, the features closely related to session duration anomalies (or problematic features) are (a) `ack.pkts.sent.a2b`, (b) `actual.data.pkts.b2a`, and (c) `tcp.first.sec.volume.dl`. Note that `tcp.first.sec.volume.dl` is present in both the knowledge model and the classification tree. The ACK packets and actual data bytes are related to packet loss and retransmissions and the last one is with delay and latency.

4.1.1. Cleaning matters!

In this section, we evaluate the impact of both the data preprocessing and the training set preparation on the performance of the anomaly detection engine. To this end, we assess the anomaly detection performance of the proposed methodology for different data-cleaning scenarios. Specifically, Table 6 indicates the implemented subphases of the XMLAD methodology for each data cleaning scenario. As seen in Table 6, we evaluate five different data cleaning scenarios. It should note that the missing value detection subphase has been implemented in all scenarios in order to avoid errors in the machine learning models caused by missing values (Nans). More precisely, Scenario 1 implements all data-cleaning subphases, and Scenario 2 removes the low variability detection only. Scenario 3 discards the low variability detection and the outlier detection in features. Moreover, Scenario 4 eliminates low variability detection, outlier detection in features, and outlier detection in experiments. Finally, Scenario 5 disregards low variability detection, outlier detection in features, anomaly

Table 6: Implemented XMLAD modules in the various scenarios of data cleaning.

XMLAD phase	XMLAD subphase	Scenario				
		1	2	3	4	5
Data preprocessing	Missing value detection	✓	✓	✓	✓	✓
	Low variability detection	✓	✗	✗	✗	✗
	Outliers in features	✓	✓	✗	✗	✗
Training set preparation	Outliers in experiments	✓	✓	✓	✗	✗
	Feature selection	✓	✓	✓	✓	✗

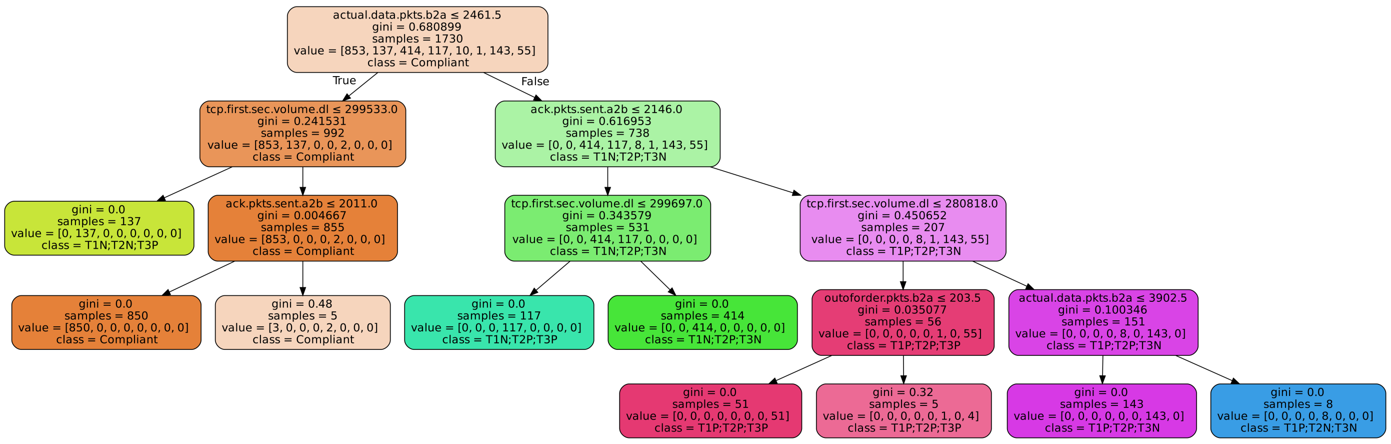

actual.data.pkts.b2a ≤ 2461.5
gini = 0.680899
samples = 1730
value = [853, 137, 414, 117, 10, 1, 143, 55]
class = Compliant
True
False
tcp.first.sec.volume.dl ≤ 299533.0
gini = 0.241531
samples = 992
value = [853, 137, 0, 0, 2, 0, 0, 0]
class = Compliant
ack.pkts.sent.a2b ≤ 2146.0
gini = 0.616953
samples = 738
value = [0, 0, 414, 117, 8, 1, 143, 55]
class = T1N;T2P;T3N
gini = 0.0
samples = 137
value = [0, 137, 0, 0, 0, 0, 0, 0]
class = T1N;T2N;T3P
ack.pkts.sent.a2b ≤ 2011.0
gini = 0.004667
samples = 855
value = [853, 0, 0, 0, 2, 0, 0, 0]
class = Compliant
tcp.first.sec.volume.dl ≤ 299697.0
gini = 0.343579
samples = 531
value = [0, 0, 414, 117, 0, 0, 0, 0]
class = T1N;T2P;T3N
tcp.first.sec.volume.dl ≤ 280818.0
gini = 0.450652
samples = 207
value = [0, 0, 0, 0, 8, 1, 143, 55]
class = T1P;T2P;T3N
gini = 0.0
samples = 850
value = [850, 0, 0, 0, 0, 0, 0, 0]
class = Compliant
gini = 0.48
samples = 5
value = [3, 0, 0, 0, 2, 0, 0, 0]
class = Compliant
gini = 0.0
samples = 117
value = [0, 0, 0, 117, 0, 0, 0, 0]
class = T1N;T2P;T3P
gini = 0.0
samples = 414
value = [0, 0, 414, 0, 0, 0, 0, 0]
class = T1N;T2P;T3N
outoforder.pkts.b2a ≤ 203.5
gini = 0.035077
samples = 56
value = [0, 0, 0, 0, 0, 1, 0, 55]
class = T1P;T2P;T3P
actual.data.pkts.b2a ≤ 3902.5
gini = 0.100346
samples = 151
value = [0, 0, 0, 0, 8, 0, 143, 0]
class = T1P;T2P;T3N
gini = 0.0
samples = 51
value = [0, 0, 0, 0, 0, 0, 0, 51]
class = T1P;T2P;T3P
gini = 0.32
samples = 5
value = [0, 0, 0, 0, 0, 1, 0, 4]
class = T1P;T2P;T3P
gini = 0.0
samples = 143
value = [0, 0, 0, 0, 0, 0, 143, 0]
class = T1P;T2P;T3N
gini = 0.0
samples = 8
value = [0, 0, 0, 0, 8, 0, 0, 0]
class = T1P;T2N;T3N

Figure 9: Schematic representation of the tree structure of the anomaly classification engine built from the NokiaFL dataset.

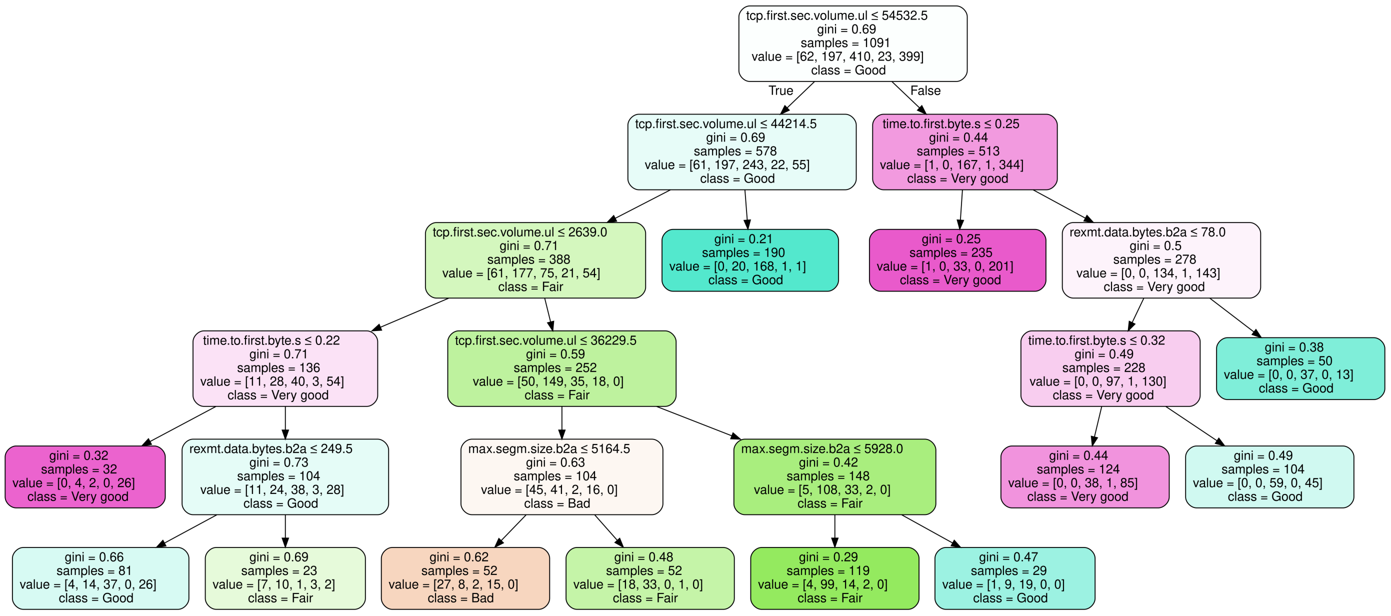


Figure 10: Schematic representation of the anomaly detection engine built for the NokiaLP dataset.

Table 7: Dimensions of the input training samples and recall of the anomaly detection engine for different data cleaning scenarios.

Scenario	Training features	Training samples	Detection recall
1	12	$1,385$	100.00%
2	12	$1,637$	77.42%
3	20	682	49.46%
4	27	$1,730$	39.78%
5	58	$1,730$	36.56%

detection in experiments, and feature selection based on model-based clustering.

Table 7 depicts the dimensions of the input sample set that trains the anomaly detection engine for different data-cleaning scenarios. Table 7 also displays the recall values of the anomaly detection engine for the various data cleaning scenarios. As part of an anomaly detection process, the recall indicates how many anomalous experiments are correctly detected, which is expressed as

$$recall = \frac{TP}{TP + FN}, \tag{24}$$

where TP is the number of correctly detected anomalies (true positives) and FN is the number of incorrectly labeled normal experiments (false negatives). This metric decreases when the number of normal behaviors incorrectly labeled as anomalies increases, describing a worse detection performance. To compute the recall metric, we select as the reference set the vector of outliers of the target KPI yielded by the MAD method, i.e.,

$$\boldsymbol{r}_v = \text{outlier}(\boldsymbol{z}), \tag{25}$$

where $\boldsymbol{r}_v$ denotes the reference vector, outlier$(\cdot)$ is defined in (7), and $\boldsymbol{z}$ stands for the target KPI vector. Table 7 shows that the recall metric decreases as the discarded cleaning stages increases. It is important to mention that the vector of anomalous scenarios should contain information about the detected anomalies in order to calculate the miscoding metric correctly and, consequently, select the relevant features that describe anomalies. When more data cleaning stages are discarded, an important percentage of information is lost.

4.2. The NokiaLP dataset

In this case, we attempt to determine the features and experiments inducing anomalous session times when a test device (terminal a) fetches a live static web page (terminal b). Fig. 10 depicts the decision tree structure of the anomaly detection engine for the NokiaLP dataset. In particular, the features modeling the network's normal behavior are (a) `tcp.first.sec.volume.ul`, (b) `time.to.first.byte.s`, (c) `rexmt.data.bytes.b2a`, and (d) `max.segm.size.b2a`. The decision tree first compares `tcp.first.sec.volume.ul` with respect to the 54,533 bytes. If the feature value is less than the threshold, the decision tree moves to the left side and assigns the "Good" label to the experiment, otherwise, the decision structure moves to the right side and uses the "Very good" label for the experiment. It is important to note that `tcp.first.sec.volume.ul` reflects the volume of data delivered to the user during the first second of the connection, so higher values indicate better performance. As can be seen in Fig 10, the anomaly detection engine recognizes and explains this pattern. Furthermore, it can be seen that the comparison with different levels of `tcp.first.sec.volume.ul` can categorize up to four different classes: Bad, Fair, Good, and Very Good. This suggests that `tcp.first.sec.volume.ul` is a relevant feature for describing the network's normal behavior. As mentioned before, this feature reflects the size of the HTTP request detecting different URLs used for this test so performance differences are associated with the test server.

Fig. 11 shows the scatter plots of the vector of anomalous scenarios versus the problematic feature values. These scatter plots also show the clusters of problematic and non-problematic experiments. As can be observed in Fig. 11, the XMLAD methodology identifies as problematic scenarios those experiments with (a) `rtt.from.3whs.a2b` between 50 and 135 ms, (b) `time.to.first.byte.s` greater than 0.5 s, and `tcp.first.sec.volume.ul` with zero bytes. Fig. 12 illustrates the flowchart of the tree structure of the anomaly classification engine for the NokiaLP dataset. For each testing scenario, the anomaly classification model first compares the `tcp.first.sec.volume.ul` with respect to 583 bytes. If this variable is less than the threshold, the classification engine detects an anomaly and indicates that the anomaly is caused by the `tcp.first.sec.volume.ul` (T2P). Notice in Fig. 11(b) that problematic experiments T2P exhibit zero-value, hence, anomalies labeled with T2P are related to those experiments that could not upload data during the first second. It can be noticed that T1P and T3P anomalies are related to `rtt.from.3whs.a2b` and `time.to.first.byte.s`, respectively. For instance, the classification engine identifies performance anomalies caused by the three problematic features (T1P; T2P; T3P) when `tcp.first.sec.volume.ul` is less than 583 bytes, `time.to.first.byte.s` is less than 0.47 s, and `rtt.from.3whs.a2b` is greater than 51.5 ms.

Table 8: Dimensions of the data matrices yielded by different subphases of the XMLAD methodology for the MonroeQL dataset.

XMLAD Subphase	Data matrix	Rows	Cols
Original input	$\boldsymbol{D}$	3,951	188
Missing value detection	$\boldsymbol{D}_1$	3,951	181
Low variability detection	$\boldsymbol{D}_2$	3,951	120
Outlier detection in features	$\boldsymbol{D}_3$	3,951	111
Outlier detection in experiments	$\boldsymbol{T}$	222	111
Feature selection	$\boldsymbol{X}_{train}$	222	4

4.3. The MonroeQL dataset

For this dataset, we select the *average session time* as the target KPI. Table 8 displays the data sizes obtained by the data preprocessing and training set preparation. In contrast to the results yielded from the NokiaFL dataset, for this dataset, the low variability identification step discards the largest number of features (61, 32.45% of the original number of input features). Furthermore, null cell identification and outlier detection in features remove 11 (5.85%) and 9 (4.79%) columns, respectively. In this case, several variables with all empty cells were set to zero during the data extraction process, and therefore, the low variability identification removes features with a large number of zero-valued cells that do not contribute to the modeling of the anomaly detection engine.

The outlier detection in experiments extracts a small number of operation scenarios (222, 5.62% of the original number of experiments) to build training samples and labels. On the other hand, the number of class labels determined by the discretization process is given by $w = \lfloor(\log_2 m_\ell)/2\rfloor = 3$. We can see in Fig. 13(a) the histogram of the classes generated by the discretization process. Then, the methodology applies the feature selection method based on the 2D clustering over projections of the correlation matrix. Fig. 13(b) depicts the clusters detected by the feature selection technique. The features are listed next:

1. `abs.cwin.volstep.30kb`,
2. `abs.cwin.50`,
3. `abs.rtt.timestep.320ms`,
4. `abs.cwin.timestep.160.ms`.

Fig. 14 shows the decision tree structure of the fitted anomaly detection classifier for the MonroeQL dataset. For this dataset, the decision tree classifier compares the `abs.cwin.50` with respect to 40,619 bytes at the root node. If the inequality rule is true at the root node, the decision tree evaluates the `abs.cwin.50` with respect to 20,219 bytes. The decision tree classifier assigns the "Bad" label to the sample if the `abs.cwin.50` is lower or equal to 20,219 bytes, i.e., the decision tree identifies a bad network performance when the `abs.cwin.50` is lower than 20,219

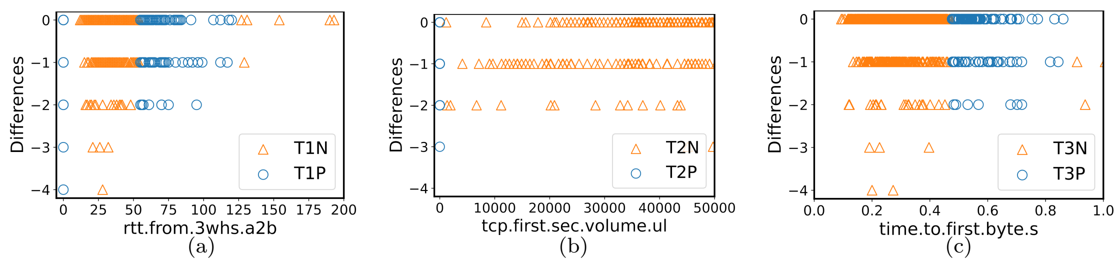


Figure 11: Scatter plots of the vector of anomalous scenarios $\boldsymbol{y}_{anm}$ versus the feature value for the detected problematic features and for the NokiaLP dataset.

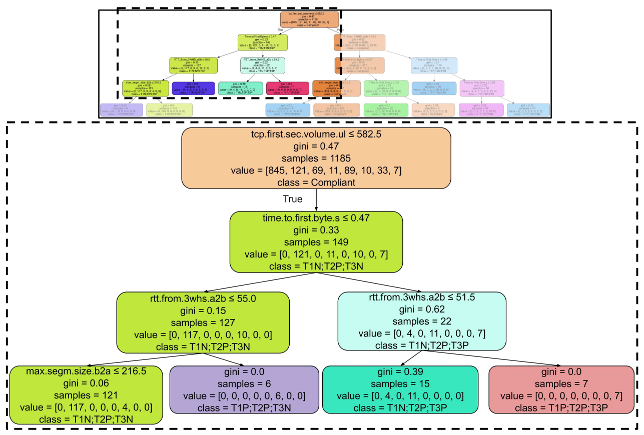


Figure 12: Schematic representation of the anomaly classification tree built for the NokiaLP dataset.

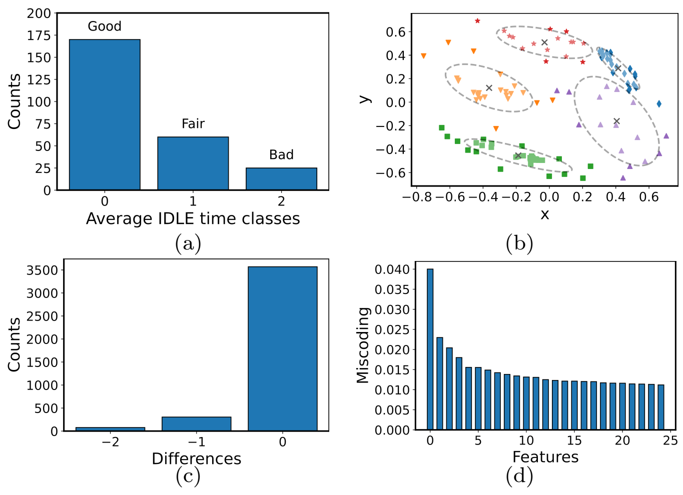


Figure 13: MonroeQL dataset. (a) Histogram of the training labels obtained by the discretization model. (b) Clustering on the projections of $\boldsymbol{C}$ into the 2D Euclidean space obtained by the model-based technique. (c) Histogram of the vector of anomalous scenarios $\boldsymbol{y}_{anm}$. (d) Sorted barplot of the miscoding between every column of $\boldsymbol{D}_3$ and $\boldsymbol{y}_{anm}$.

bytes. In summary, the decision rule detects a bad network performance for $20,219 <$ `abs.cwin.50` $\leq 40,619$, and `abs.rtt.timestep.320ms` > 0.15 s. For this dataset, the network's normal behavior is described with two features: (a) `abs.cwin.50`, (b) `abs.rtt.timestep.320ms`. Fig. 13(c) and (d) illustrate the histogram of the vector of anomalous scenarios and the barplot of the miscoding estimates for the set of relevant features, respectively. More precisely, the sorted features according to the miscoding are shown below:

1. `abs.idletime.max`
2. `abs.idletime.timestep.10240ms`
3. `abs.segmentsizes.timestep.640ms`
4. `abs.idletime.timestep.320ms`
5. `abs.idletime.timestep.1280ms`
6. `abs.segmentsizes.timestep.320ms`
7. `abs.rtt.volstep.1000kb`
8. `abs.idletime.timestep.2560ms`
9. `abs.rtt.75`
10. `abs.segmentsizes.firstsec`
11. `delay`
12. `abs.idletime.timestep.20480ms`
13. `abs.rtt.timestep.5120ms`
14. `abs.rtt.timestep.10240ms`
15. `abs.rtt.50`
16. `abs.rtt.volstep.1650kb`
17. `abs.idletime.timestep.5120ms`
18. `abs.rtt.avg`
19. `abs.rtt.25`
20. `abs.idletime.volstep.1650kb`
21. `abs.rtt.timestep.1280ms`
22. `abs.rtt.timestep.2560ms`
23. `abs.segmentsizes.volstep.240kb`
24. `abs.rtt.volstep.240kb`
25. `abs.idletime.volstep.1000kb`

Fig. 15 shows the scatter obtained for the problematic features and for each problematic feature, we can observe the clusters of problematic and non-problematic experiments. The proposed methodology detects network problems for `abs.cwin.50`, `abs.rtt.timestep.2560.ms`, and `abs.rtt.volstep.240kb` greater than 300 ms approximately. In addition, the decision tree structure of the

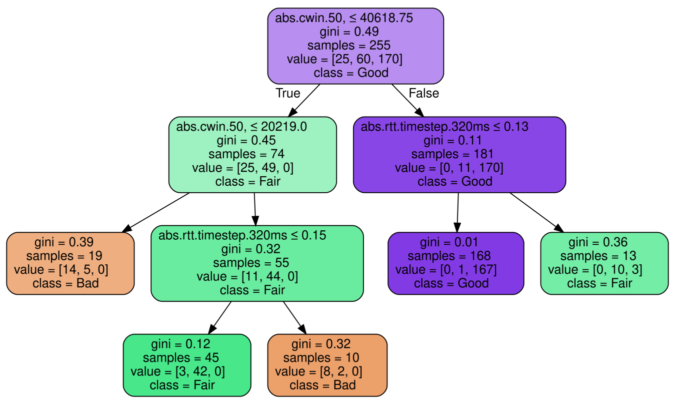


Figure 14: Schematic representation of the anomaly detection engine built for the MonroeQL dataset.

anomaly classification engine is displayed in Fig. 16. If the `abs.rtt.timestep.2560.ms` is greater than 0.32 s, the anomaly classification model detects an anomaly that could be induced by `abs.rtt.timestep.2560.ms` (T2P). Once verified that `abs.rtt.timestep.2560.ms` < 0.32 s, the decision tree compares `abs.rtt.volstep.240kb` with respect to 0.03 s. If the inequality is true, the XMLAD identifies an anomaly that could be induced by `abs.cwin.50` (T1P), otherwise, the methodology detects anomalies

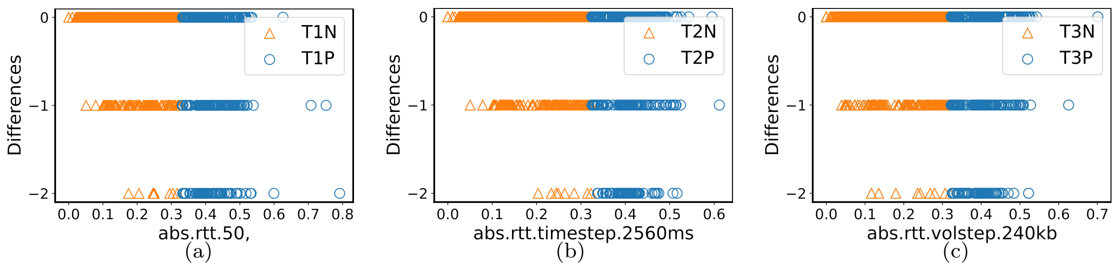


Figure 15: Scatter plots of the vector of anomalous scenarios $\boldsymbol{y}_{anm}$ versus the feature value for problematic features and for the MonroeQL dataset.

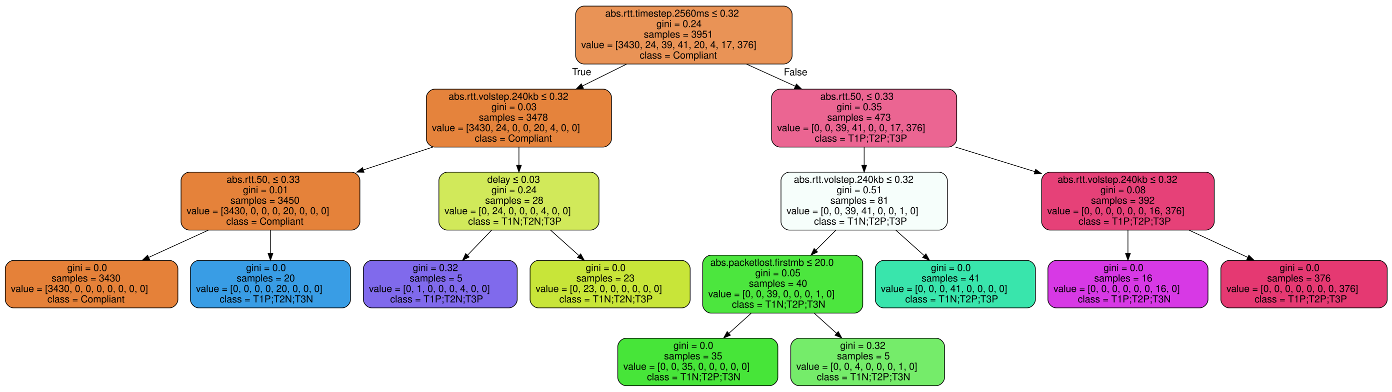


Figure 16: Schematic representation of the anomaly classification engine built for the MonroeQL dataset.

Table 9: Simulation parameters for the synthetic dataset

Parameter	value	Parameter	value
m_0	20,000 samples	n_1	10 features
lb_{mss}	1,300 bytes	ub_{mss}	1,400 bytes
β	$400(\text{ms})^{-1}$	RTT_{min}	30ms
$\bar{\bar{\mu}}_{cwnd}$	100kB	σ_{cwnd}	0.65
lb_{cwnd}	40kB	lb_{cwnd}	700kB
ϵ_{cwnd}	0.90	$\bar{\bar{\mu}}_p$	2.50×10^{-6}
σ_p	0.20	lb_p	1.00×10^{-4}
lb_p	1.25×10^{-4}	ϵ_p	0.10

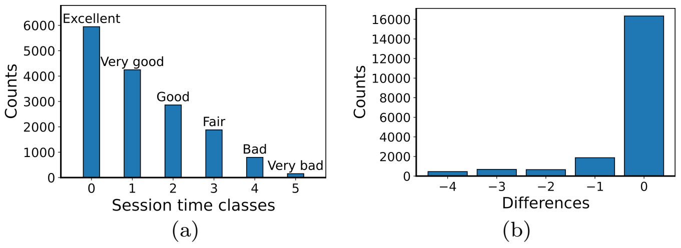


Figure 17: Synthetic dataset. (a) Histogram of the training labels obtained by the discretization model. (b) Histogram of the vector of anomalous scenarios $\boldsymbol{y}_{anm}$.

caused by both `abs.cwin.50` (T1P) and `abs.rtt.volstep.240kb` (T3P). RTT refers to buffering. Higher values normally indicates radio buffering. Time and Volume steps refers to interval for statistic calculation. Small values (Volume < 500KB or Time < 500ms) normally describe TCP aggressiveness and Big Values are more related to TCP ability to fill and maintain channel utilization. All the cases describe Radio limitations in different phases of the test (Median, End or Beginning).

Remarkably, the results of the analysis carried out by means of the XMLAD methodology are meaningful, as they were validated by a Nokia expert on anomaly/alarm detection and remediation procedures. What has been shown so far hints at the fact that XMLAD can identify anomalies, but does not tell if the results are accurate or not. For a more objective evaluation of the results achievable with XMLAD, we will consider synthetic data in Section 4.4, in which we purposely introduce anomalies, and hence can use them as ground truth.

4.4. Synthetic datasets

This section evaluates the performance of the proposed methodology from a synthetic dataset generated by the statistical models described in Section 3.3. Specifically, the testbed generates a dataset synthetic dataset with $m_0 = 20,000$ experiments and $n_1 = 10$ features. It should note the generated dataset contains a packet loss probability (p) feature whose entries are drawn as independent samples following the statistical model (23) with a predefined rate of outliers $\epsilon_p = 0.10$. A summary of the simulation parameters used to generate synthetic data can be found in Table 9. As observed in Fig. 3(d), the statistical model generates a bimodal distribution with two separated modes. In the context of the TCP throughput model described by (19) and (20), outliers in packet loss probability could lead to large session times. This behavior favors the outlier detection in experiments at the training set preparation stage and the 1D clustering at the detection of problematic features and experiments.

For a specific realization of the synthetic dataset, the data cleaning stage extracts a matrix with $m_\ell = 15,785$ experiments and $n_\ell = 4$ features. In essence, the feature selection method based on the 2D clustering tech-

Table 10: Class labels of the anomaly classification engine for the synthetic dataset

Class label	Type of problem
`Compliant`	No problem
`T1N:T2N;T3P`	p problem
`T1N:T2P;T3N`	RTT problem
`T1P:T2N;T3N`	$CWIN$ problem
`T1N:T2P;T3P`	RTT problem and p problem
`T1P:T2N;T3P`	$CWIN$ problem and p problem
`T1P:T2P;T3N`	$CWIN$ problem and RTT problem
`T1P:T2P;T3P`	$CWIN$ problem and RTT problem and p problem

nique identifies the variables `RTT`, `MSS`, `CWND`, and `End.RSSP.dBm`. The data cleaning stage discards the entire set of experiments (100%) with large packet loss probability values drawn from the uniform distribution of the model (23). On the other hand, the discretization process yields six class labels, i.e., $w = \lfloor (\log_2 m_\ell)/2 \rfloor = 6$. Fig. 17(a) displays the histogram of the training labels outputted by the discretization process for the synthetic dataset. Then, the methodology optimizes the anomaly detection engine $g_\phi(\cdot)$ using the CART algorithm from the training set $\mathbf{\Gamma} = \{\boldsymbol{X}_{train}, \boldsymbol{y}_{train}\}$. The decision tree structure of the anomaly detection is omitted due to space limitations.

Then, we obtain the vector of anomalous scenarios from the entire set of experiments of the original data. Fig. 17(b) displays the histogram of the vector of anomalous scenarios $\boldsymbol{y}_{anm}$. Afterward, XMLAD implements the detection of problematic features and experiments. For this realization of the synthetic dataset, the methodology detects `p`, `CWIN`, and `RTT` as features causing anomalous session times. Fig. 18 shows the scatter plots of the vector of anomalous scenarios versus the magnitude of the problematic features. Furthermore, these scatter plots display the problematic and non-problematic samples. From problematic features and experiments, the methodology builds the output labels for the anomaly classification engine. To be more precise, this procedure generates eight different classes of anomalies that are described in Table 10.

The anomaly classification engine is fitted using the entire set of experiments and features. Fig. 19 illustrates the decision tree structure of the anomaly classification engine. In this figure, the classifier first compares the `CWIN` with respect to the threshold 564506 bytes. If an experiment has a `CWIN` feature value lower than the threshold, the classifier identifies an anomaly caused by network aspects related to the `CWIN` parameter, otherwise, it describes a normal operation scenario. In the next tree level, the anomaly classifier sets a threshold 5×10^{-5} to separate packet loss probabilities into two classes: normal and problematic. For example, for $\mathtt{p} \leq 5 \times 10^{-5}$, the classifier identifies operation scenarios that are not affected by the packet loss probability. On the contrary, the classifier detects performance anomalies for $\mathtt{p} > 5 \times 10^{-5}$. Dashed lines in Fig. 3(d) and Fig. 18(c) correspond to the boundaries determined by the classifier, i.e., $\mathtt{p} = 5 \times 10^{-5}$. For this experiment, the classification engine is able to identify the entire set of anomalies (100%) introduced in the distribution of p with the uniform term of (23). Additionally, XMLAD identifies anomalous scenarios related to `RTT`, due to the heavy tails exhibited by the corresponding statistical model. Although the distributions of `CWIN` and `RTT` are not multimodal, the 1D clustering at these problematic experiments detects problematic samples. Fig. 19 shows how the introduced anomalies (p-anomalies) are successfully classified but also how other performance anomalies are detected (due to high `RTT`, which is also a meaningful result) and how both types of anomalies are combined in the data set. As a result, we have successfully classified the samples and identified which ones have none, one, or both anomalies.

Finally, we compare the anomaly detection performance of the XMLAD methodology with respect to those obtained by the TTrees [35] and Kim [25] techniques. To this end, Fig. 20 shows the recall curves (24) with the corresponding 95% confidence interval yielded by the various anomaly detection methods for different rates of packet loss outliers. More precisely, we observe the recall curves for the TTrees method, the Kim method with $\alpha = 0.01$ and $\alpha = 0.001$, the anomaly detection engine of the XMLAD methodology (XMLAD model 1), and the anomaly classification engine of the XMLAD methodology (XMLAD model 2). Each point of the curves is obtained by averaging 100 realizations of the corresponding experiment and at each trial, a new synthetic dataset is generated with the number of operation scenarios fixed to $m_0 = 2,000$. The remaining simulation parameters are the same as those shown in Table 9 except the rate of packet loss outliers ϵ_p that varies from 0.01 to 0.15. As can be seen in this figure, both the anomaly detection engine and the anomaly classification engine of the XMLAD approach exhibit superior performance with respect to the other state-of-the-art methods.

5. Conclusions

In this work, we have developed a methodology based on data-cleaning procedures and explainable learning models to detect and classify anomalous operation scenarios in mobile networks. More precisely, a preprocessing stage was included in the proposed methodology, called explainable machine learning for anomaly detection and classification (XMLAD) anomaly detection, to train the anomaly detection engine with the cleanest dataset such that the training data properly describes the network's normal behavior. Further, two interpretable learning models were incorporated to detect and classify anomalies. These explainable learning models output tree structures whose decision rules enabled the identification of features and thresholds that describe the network's normal behavior and the recognition of the samples that could induce performance anomalies. We built a testbed to generate synthetic datasets whose feature samples obey well-defined statistical models and the KPI sets follow a known TCP model. The performance of the proposed methodology was evaluated on

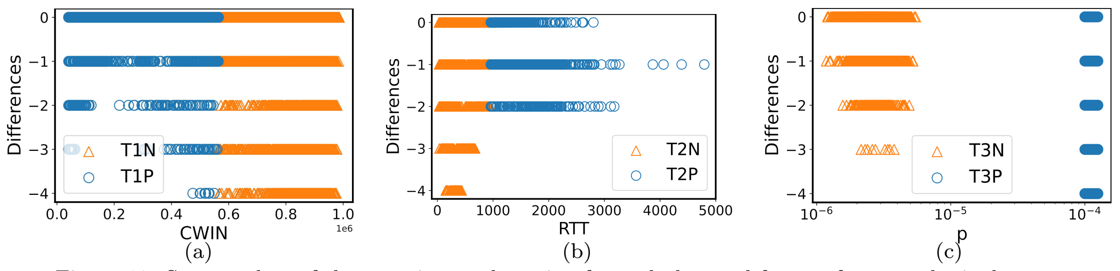


Figure 18: Scatter plots of the experiment clustering for each detected feature from synthetic dataset.

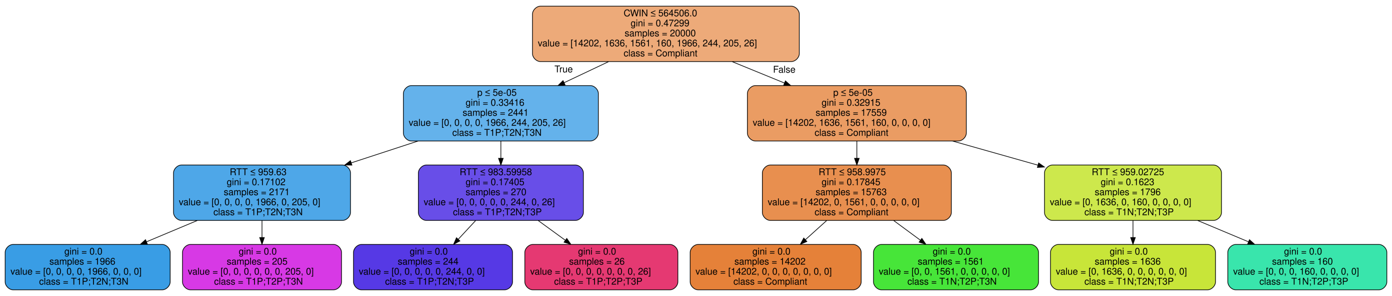


Figure 19: Structure of the anomaly classification decision tree built for the synthetic dataset.

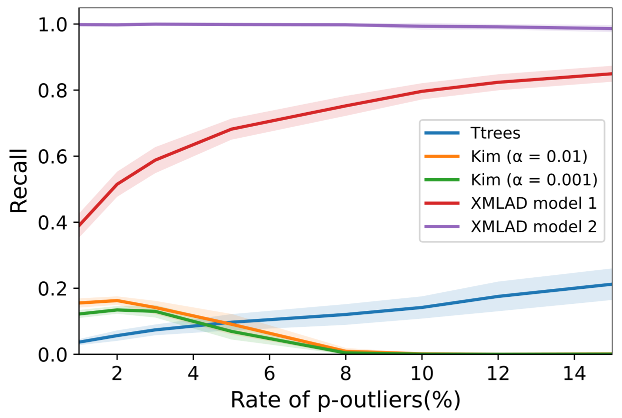


Figure 20: Recall curves with the corresponding 95% confidence interval versus the rate of packet loss probability outliers (p-outliers) using synthetic data for different anomaly detection techniques.

synthetic and real datasets. In this sense, the proposed approach efficiently detected the performance anomalies and provided information about the set of features that can cause network misfunctioning. In addition, the decision trees generated interpretable structures enabling the understanding of both the network's normal behavior and anomaly classification. In future work, we are interested in evaluating other types of anomalies as well as different learning models such as deep neural networks.

6. Acknowledgements

This work has been partially supported by the Project AEON-CPS (TSI-063000-2021-38), funded by the Ministry of Economic Affairs and Digital Transformation and the European Union NextGeneration-EU in the framework of the Spanish Recovery, Transformation and Resilience Plan. This work has been also supported by the Project EDGEDATA-CM (S2018/TCS-4499), funded by the Department of Education and Research of the Regional Government of Madrid, through the 2018 RD technology program for research groups, co-financed by the Operational Programs of the European Social Fund (ESF) and the European Regional Development Fund (ERDF). The work of Juan M. Ramirez has been supported by the Project ECID: Edge Computing for Intelligent Driving (PID2019-109805RB-I00) funded by the Spanish State Research Agency, Spanish Ministry of Science and Innovation. This paper has been supported by the projects Netpredict4 and Netpredict5 funded by Nokia Spain.

References

[1] J. M. Ramírez, P. Rojo, F. Díez, V. Mancuso, A. Fernández Anta, Cleaning matters! preprocessing-enhanced anomaly detection and classification in mobile networks, in: 2022 20th Mediterranean Communication and Computer Networking Conference (MedComNet), 2022, pp. 103–112. doi:10.1109/MedComNet55087.2022.9810378.

[2] N. Al-Falahy, O. Y. Alani, Technologies for 5g networks: Challenges and opportunities, IT Professional 19 (2017) 12–20. doi:10.1109/MITP.2017.9.

[3] M. Shafi, A. F. Molisch, P. J. Smith, T. Haustein, P. Zhu, P. De Silva, F. Tufvesson, A. Benjebbour, G. Wunder, 5G: A tutorial overview of standards, trials, challenges, deployment, and practice, IEEE Journal on Selected Areas in Communications 35 (2017) 1201–1221. doi:10.1109/JSAC.2017.2692307.

[4] C. Zhang, P. Patras, H. Haddadi, Deep learning in mobile and wireless networking: A survey, IEEE Communications Surveys & Tutorials 21 (2019) 2224–2287. doi:10.1109/COMST.2019.2904897.

[5] M. Abbasi, A. Shahraki, A. Taherkordi, Deep learning for network traffic monitoring and analysis (ntma): A survey, Computer Communications 170 (2021) 19–41.

[6] M. Chen, A. Liu, W. Liu, K. Ota, M. Dong, N. N. Xiong, Rdrl: A recurrent deep reinforcement learning scheme for dynamic spectrum access in reconfigurable wireless networks, IEEE Transactions on Network Science and Engineering 9 (2022) 364–376. doi:10.1109/TNSE.2021.3117565.

[7] M. Chen, W. Liu, T. Wang, S. Zhang, A. Liu, A game-based deep reinforcement learning approach for energy-efficient com-

putation in mec systems, Knowledge-Based Systems 235 (2022) 107660.

[8] M. Thottan, C. Ji, Anomaly detection in IP networks, IEEE Transactions on Signal Processing 51 (2003) 2191–2204. doi:10.1109/TSP.2003.814797.

[9] M. H. Bhuyan, D. K. Bhattacharyya, J. Kalita, Network anomaly detection: Methods, systems and tools, IEEE Communications Surveys Tutorials 16 (2014) 303–336. doi:10.1109/SURV.2013.052213.00046.

[10] E. Falk, R. Camino, R. State, V. K. Gurbani, On non-parametric models for detecting outages in the mobile network, in: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2017, pp. 1139–1142. doi:10.23919/INM.2017.7987448.

[11] M. Moulay, R. A. G. Leiva, P. J. R. Maroni, J. Lazaro, V. Mancuso, A. F. Anta, A novel methodology for the automated detection and classification of networking anomalies, in: 39th IEEE Conference on Computer Communications, INFOCOM Workshops 2020, Toronto, ON, Canada, July 6-9, 2020, IEEE, 2020, pp. 780–786. URL: https://doi.org/10.1109/INFOCOMWKSHPS50562.2020.9162710. doi:10.1109/INFOCOMWKSHPS50562.2020.9162710.

[12] V. K. Gurbani, D. Kushnir, V. Mendiratta, C. Phadke, E. Falk, R. State, Detecting and predicting outages in mobile networks with log data, in: 2017 IEEE International Conference on Communications (ICC), 2017, pp. 1–7. doi:10.1109/ICC.2017.7996706.

[13] V. B. Mendiratta, M. Thottan, Rich network anomaly detection using multivariate data, in: 2017 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 2017, pp. 48–51. doi:10.1109/ISSREW.2017.36.

[14] G. Yu, Z. Cai, S. Wang, H. Chen, F. Liu, A. Liu, Unsupervised online anomaly detection with parameter adaptation for KPI abrupt changes, IEEE Transactions on Network and Service Management 17 (2020) 1294–1308. doi:10.1109/TNSM.2019.2962701.

[15] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, K.-R. Müller, A unifying review of deep and shallow anomaly detection, Proceedings of the IEEE 109 (2021) 756–795. doi:10.1109/JPROC.2021.3052449.

[16] O. Alay, A. Lutu, M. Peón-Quirós, V. Mancuso, T. Hirsch, K. Evensen, A. Hansen, S. Alfredsson, J. Karlsson, A. Brunstrom, A. Safari Khatouni, M. Mellia, M. A. Marsan, Experience: An open platform for experimentation with commercial mobile broadband networks, in: Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking, MobiCom '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 70–78. URL: https://doi.org/10.1145/3117811.3117812. doi:10.1145/3117811.3117812.

[17] W. Lu, A. A. Ghorbani, Network anomaly detection based on wavelet analysis, EURASIP Journal on Advances in Signal Processing 2009 (2008) 1–16.

[18] A. H. Yaacob, I. Tan, S. F. Chien, H. K. Tan, Arima based network anomaly detection, in: 2010 Second International Conference on Communication Software and Networks, 2010, pp. 205–209. doi:10.1109/ICCSN.2010.55.

[19] H. Kasai, W. Kellerer, M. Kleinsteuber, Network volume anomaly detection and identification in large-scale networks based on online time-structured traffic tensor tracking, IEEE Transactions on Network and Service Management 13 (2016) 636–650. doi:10.1109/TNSM.2016.2598788.

[20] M. Thill, W. Konen, T. Bäck, Online anomaly detection on the webscope s5 dataset: A comparative study, in: 2017 Evolving and Adaptive Intelligent Systems (EAIS), 2017, pp. 1–8. doi:10.1109/EAIS.2017.7954844.

[21] N. Laptev, S. Amizadeh, I. Flint, Generic and scalable framework for automated time-series anomaly detection, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '15, Association for Computing Machinery, New York, NY, USA, 2015, p. 1939–1947. URL: https://doi.org/10.1145/2783258.2788611. doi:10.1145/2783258.2788611.

[22] D. Liu, Y. Zhao, H. Xu, Y. Sun, D. Pei, J. Luo, X. Jing, M. Feng, Opprentice: Towards practical and automatic anomaly detection through machine learning, in: Proceedings of the 2015 Internet Measurement Conference, IMC '15, Association for Computing Machinery, New York, NY, USA, 2015, p. 211–224. URL: https://doi.org/10.1145/2815675.2815679. doi:10.1145/2815675.2815679.

[23] J. Dromard, G. Roudière, P. Owezarski, Online and scalable unsupervised network anomaly detection method, IEEE Transactions on Network and Service Management 14 (2017) 34–47. doi:10.1109/TNSM.2016.2627340.

[24] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, J. Chen, Z. Wang, H. Qiao, Unsupervised anomaly detection via variational auto-encoder for seasonal KPIs in web applications, in: Proceedings of the 2018 World Wide Web Conference, WWW '18, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 2018, p. 187–196. URL: https://doi.org/10.1145/3178876.3185996. doi:10.1145/3178876.3185996.

[25] C. Kim, V. B. Mendiratta, M. Thottan, Unsupervised anomaly detection and root cause analysis in mobile networks, in: 2020 International Conference on Communication Systems and networks (COMSNETS), 2020, pp. 176–183. doi:10.1109/COMSNETS48256.2020.9027328.

[26] C. Leys, C. Ley, O. Klein, P. Bernard, L. Licata, Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median, Journal of Experimental Social Psychology 49 (2013) 764–766. doi:https://doi.org/10.1016/j.jesp.2013.03.013.

[27] P. Huber, Robust statistics, volume 523, John Wiley & Sons, 2004.

[28] P. Ghamisi, N. Yokoya, J. Li, W. Liao, S. Liu, J. Plaza, B. Rasti, A. Plaza, Advances in hyperspectral image and signal processing: A comprehensive overview of the state of the art, IEEE Geoscience and Remote Sensing Magazine 5 (2017) 37–78.

[29] I. Borg, P. Groenen, Modern multidimensional scaling: Theory and applications, Springer Science & Business Media, 2005.

[30] L. Scrucca, M. Fop, T. B. Murphy, A. E. Raftery, mclust 5: clustering, classification and density estimation using gaussian finite mixture models, The R journal 8 (2016) 289.

[31] G. J. McLachlan, S. Rathnayake, On the number of components in a gaussian mixture model, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 4 (2014) 341–355.

[32] S. García, J. Luengo, J. A. Sáez, V. López, F. Herrera, A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning, IEEE Transactions on Knowledge and Data Engineering 25 (2013) 734–750. doi:10.1109/TKDE.2012.35.

[33] L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone, Classification and regression trees, Routledge, 2017.

[34] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Springer Series in Statistics, Springer New York Inc., New York, NY, USA, 2001.

[35] M. Moulay, R. G. Leiva, V. Mancuso, P. J. Rojo Maroni, A. F. Anta, Ttrees: Automated classification of causes of network anomalies with little data, in: 2021 IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2021, pp. 199–208. doi:10.1109/WoWMoM51794.2021.00033.

[36] V. Mancuso, M. P. Quirós, C. Midoglu, M. Moulay, V. Comite, A. Lutu, Ö. Alay, S. Alfredsson, M. Rajiullah, A. Brunström, et al., Results from running an experiment as a service platform for mobile broadband networks in Europe, Computer Communications 133 (2019) 89–101.

[37] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W.-T. Chang, Z. Shi, The quic transport protocol: Design and internet-scale deployment, in: Proceedings of the Conference of the ACM Special Interest Group on Data Communication, SIGCOMM '17, Associ-

ation for Computing Machinery, New York, NY, USA, 2017, p. 183–196. URL: https://doi.org/10.1145/3098822.3098842. doi:10.1145/3098822.3098842.

[38] R. Marx, W. Lamotte, J. Reynders, K. Pittevils, P. Quax, Towards quic debuggability, in: Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC, EPIQ'18, Association for Computing Machinery, New York, NY, USA, 2018, p. 1–7. URL: https://doi.org/10.1145/3284850.3284851. doi:10.1145/3284850.3284851.

[39] M. Moulay, R. G. Leiva, P. J. R. Maroni, F. Diez, V. Mancuso, A. F. Anta, Automated identification of network anomalies and their causes with interpretable machine learning: The cian methodology and ttrees implementation, Computer Communications (2022).

[40] M. Mathis, J. Semke, J. Mahdavi, T. Ott, The macroscopic behavior of the TCP congestion avoidance algorithm, SIGCOMM Comput. Commun. Rev. 27 (1997) 67–82. URL: https://doi.org/10.1145/263932.264023. doi:10.1145/263932.264023.

[41] G. James, D. Witten, T. Hastie, R. Tibshirani, An introduction to statistical learning, volume 112, Springer, 2013.

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, the Journal of machine Learning research 12 (2011) 2825–2830.