Cleaning Matters! Preprocessing-enhanced Anomaly Detection and Classification in Mobile Networks

Juan Marcos Ramírez*, Pablo Rojo†, Fernando Díez‡, Vincenzo Mancuso* and Antonio Fernández Anta*
*IMDEA Networks Institute, Spain †Nokia CNS Spain ‡Universidad Politécnica de Madrid, Spain

***Abstract*—Mobile communications providers often monitor key performance indicators (KPIs) with the goal of identifying anomalous operation scenarios that can affect the quality of Internet-based services. In this regard, anomaly detection and classification in mobile networks has become a challenging task due to the unknown distributions exhibited by the collected data and the lack of interpretability of the embedded (machine learning) models. This paper proposes an unsupervised end-to-end methodology based on both a data cleaning strategy and explainable machine learning models to detect and classify performance anomalies in mobile networks. The proposed approach, dubbed clean and explainable anomaly detection and classification (KLNX), aims at identifying attributes and operation scenarios that could induce anomalous KPI values without resorting to parameter tuning. Unlike previous methodologies, the proposed method includes a data cleaning stage that extracts and removes experiments and attributes considered outliers in order to train the anomaly detection engine with the cleanest possible dataset. Additionally, machine learning models provide interpretable information about features and boundaries describing both the normal network behavior and the anomalous scenarios. To evaluate the performance of the proposed method, a testbed generating synthetic data is developed using a known TCP throughput model. Finally, the methodology is assessed on a real data set captured by operational tests in commercial networks.**

***Index Terms*—anomaly detection and classification, data preprocessing, explainable machine learning, mobile networks.**

I. Introduction

Mobile communication systems have shown a remarkable evolution over the last two decades. In this sense, the fifth-generation (5G) technologies have played a significant role in wireless network access ensuring high-speed connectivity for an increasing number of heterogeneous devices [1]. In order to provide reliable network services, mobile communication operators continuously collect information about the system's performance. More precisely, the collected data is evaluated to detect operation instances that could unveil the misfunctioning of different network components. Therefore, a technique that identifies the problematic operation instances and detects the network aspects that could induce performance anomalies is required.

In the context of communications networks, anomalies are categorized into two groups: (i) *performance anomalies* and (ii) *security anomalies* [2], [3]. In this regard, performance anomalies typically induce network efficiency loss, e.g., large-time file downloads or low-quality connections. On the other hand, security anomalies are related to malicious intrusions aiming at affecting partially or totally the network system. In this work, we focus on detecting and classifying performance anomalies. Existing approaches leverage machine learning with supervised (cf., [4]) and unsupervised mechanisms (cf., [5]), and they are often designed ad-hoc for specific problems and require fine-tuning of hyper-parameters. More importantly, they do not hint at what caused the detected anomaly.

In this paper, we propose an automated methodology to detect and classify performance anomalies in mobile networks. The proposed methodology—referred to as "clean and explainable" anomaly detection and classification (KLNX)—is an unsupervised technique that identifies both network performance indicators (attributes) and operation scenarios (experiments) that could induce anomalies. Specifically, KLNX incorporates a data cleaning stage that discards outliers in the attributes and the target KPI to optimize the anomaly detection engine with the knowledge describing the network's normal behavior. Furthermore, this framework includes two interpretable learning models that identify attributes and thresholds explaining both the network's normal behavior and the performance anomaly classification. Hence, KLNX focuses on detecting and classifying operative anomalies. Additional contributions of this paper are summarized as follows:

1) We develop a data preprocessing module that extracts attributes and samples to fit the anomaly detection engine. This approach also introduces a feature selection method based on a model-based clustering technique.
2) The methodology includes a 1D model-based clustering that automatically identifies attributes and experiments that could cause performance anomalies.
3) We build a testbed that generates synthetic datasets to evaluate the performance of the proposed methodology.

The methodology is evaluated using both real measurement and synthetic data. In particular, the real network data were collected by Nokia for auditing purposes in multiple European countries. This dataset comprises hundreds of features and a limited number of samples, challenging the modeling of the machine learning models. On the other hand, KLNX is tested using a synthetic dataset containing an attribute with a known rate of outliers. It is worth noting that the synthetic dataset contains thousands of experiments but a restricted number of attributes.

A. Notation used in the paper

We use 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). Furthermore, 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. Additionally, 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

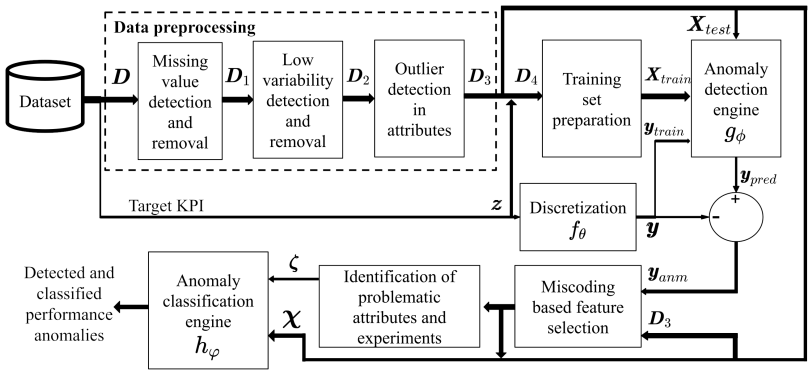


Figure 1: Flowchart of the KLNX methodology.

Table I: Notation summary

Notation	Description
$\boldsymbol{D}$	Original dataset
$\boldsymbol{z}$	Target KPI vector
$\boldsymbol{y}$	Discretized KPI vector
$\boldsymbol{D}_1$	Output of the missing value detection and removal
$\boldsymbol{D}_2$	Output of the low variability detection and removal
$\boldsymbol{D}_3$	Output of the outlier detection in attributes
$\boldsymbol{D}_4$	Input of the outlier detection in experiments
$\boldsymbol{X}_{train}$	Input data to train the anomaly detector
$\boldsymbol{y}_{train}$	Output labels to train the anomaly detector
$\boldsymbol{X}_{test}$	Input data to evaluate the anomaly detector
$\boldsymbol{y}_{pred}$	Output labels of the anomaly detector
$\boldsymbol{y}_{anm}$	Vector of anomalous scenarios
$\boldsymbol{\chi}$	Input data to train the anomaly classifier
$\boldsymbol{\zeta}$	Output labels to train the anomaly classifier
$f_\theta(\cdot)$	Function performed by the discretization model
$g_\phi(\cdot)$	Function performed by the anomaly detector
$h_\varphi(\cdot)$	Function performed by the anomaly classifier

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. We use F_V to indicate the cumulative distribution function of the random variable V, and refer to the lower quartile, the upper quartile, and the interquartile range of the observation entries included in vector $\boldsymbol{a}$ by means of $Q_1^{(\boldsymbol{a})}$, $Q_3^{(\boldsymbol{a})}$, $\mathrm{IQR}(\boldsymbol{a})$, respectively. Table I shows a summary of notations used in this paper.

B. Paper organization

The paper is organized as follows. Section II describes the data preprocessing stage and Section III illustrates the anomaly detection process. Then, Section IV details the procedure to classify network performance anomalies. Section V introduces the modeling to generate the synthetic datasets and Section VI displays the results obtained by the proposed methodology on synthetic data and real measurements. We discuss the related work in Section VII and give concluding remarks in Section VIII.

II. KLNX and Knowledge Modeling

KLNX is a three-phase unsupervised ML methodology for network performance anomaly detection and classification. In this section, we present the modeling of the anomaly detection engine of KLNX, which is the first phase of the proposed methodology. The next sections will present the second phase, i.e., the anomaly identification, and the third phase, i.e., anomaly classification. The rationale behind the design of three phases is that, before being able to identify and classify anomalies, we need to generate a clean model for the regularities of the dataset under analysis.

KLNX firstly loads a database of performance indicators (attributes or features) collected in operation scenarios (experiments). The target KPI samples are also included in the loaded data. Then, a preprocessing procedure is applied to the loaded data with the goal of obtaining the cleanest dataset to train an anomaly detection engine, also referred to as the *knowledge tree,* such that the correctly classified samples describe the normal network behavior. Fig. 1 illustrates the flowchart of the KLNX methodology. As can be seen, this phase is divided into three subphases: (i) data preprocessing, (ii) training set preparation, and (iii) knowledge model training.

A. Data preprocessing subphase

As can be observed in Fig. 1, this subphase is subdivided into three steps: detection and removal of attributes with null cells, identification and removal of attributes with low variability, and removal of attributes with a large number of outliers. Hence, the subphase applies firstly a procedure to remove features containing a large number of missing values. Secondly, it discards attributes whose samples exhibit low variability. We assume that low-variability attributes do not provide the information required to improve the performance of the knowledge tree. Finally, this subphase removes features with a large number of outliers or gross errors.

1) Missing value detection and removing: This module discards those attributes with a large number of null cells. In this regard, let $\boldsymbol{D}$ be a matrix with dimensions $m_0 \times n_0$ that contains the information extracted from the loaded dataset, where m_0 is the number of experiments and n_0 is the number of attributes. Notice that the target KPI is not included in $\boldsymbol{D}$. This step firstly removes rows of $\boldsymbol{D}$ to obtain an auxiliary matrix $\boldsymbol{D}'$, which will later be used to remove attributes. To do so, it determines the number of missing values or null cells (for example, `NaN` in numeric arrays) at each row of $\boldsymbol{D}$ as follows

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

where $\boldsymbol{d}$ is an m_0-dimensional vector whose i-th element contains the number of null cells at the i-th row of $\boldsymbol{D}$, and $\mathrm{isnull}(\cdot)$ is a nonlinear function that detects null cells. Subsequently, this step extracts from $\boldsymbol{d}$ the row indices $\mathcal{M}_1 \subseteq \{1,\ldots,m_0\}$ whose number of missing values are lower than a threshold defined by $\mathrm{median}(\boldsymbol{d}) + 3\mathrm{MAD}(\boldsymbol{d})$ [6], with $\mathrm{median}(\boldsymbol{d})$ as the sample median of the components included in $\boldsymbol{d}$, and $\mathrm{MAD}(\boldsymbol{d})$ representing 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 depends on the distribution assumed for the non-outlier samples [7]. KLNX obtains $\rho^{(\boldsymbol{d})}$ directly from the data as

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

where $\bar{\boldsymbol{d}}$ is a scaled version of $\boldsymbol{d}$, i.e., $\bar{\boldsymbol{d}} = \boldsymbol{d}/\sigma_{\boldsymbol{d}}$. It should be noted that $\sigma_{\boldsymbol{d}}$ denotes the sample standard deviation of the observations included in $\boldsymbol{d}$ and $\mu_{\boldsymbol{d}}$ is the sample mean. Hence, the subset $\mathcal{M}_1$ is defined as

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

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

Next, $\boldsymbol{D}'$ is used to filter attributes by detecting and preserving the column indices of $\boldsymbol{D}'$ that do not contain any missing value. In other words, the method obtains the subset

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

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

2) Low variability detection: After discarding attributes with a large number of missing values, our approach detects features whose samples exhibit low variability. We focus on removing attributes that do not contribute to the performance improvement of the pattern recognition model. Notice that attributes with zero variability are easily identified by computing the standard deviation of every measurement set. Nevertheless, the criterion to detect features with low variability can be set with more flexibility. In particular, we use the interquartile range (IQR), which is defined as the difference between the upper quartile and the lower quartile, i.e., $\text{IQR}(\boldsymbol{a}) = Q_3^{(\boldsymbol{a})} - Q_1^{(\boldsymbol{a})}$, as an indicator of low variability. Therefore, an attribute vector with zero-valued IQR contains at least half of the samples equal to the median value. Since the knowledge model should be optimized with samples around the central tendency of the sample set that, in turn, describes the network's normal behavior, we assume that an attribute with zero-valued IQR makes a negligible contribution to the performance improvement of the knowledge 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 attributes extracted by the procedure described in Section II-A1. This step extracts the features from $\boldsymbol{D}_1$ whose IQR is nonzero. Therefore, the subset of selected indices is

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

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

3) Outlier detection in attributes: An outlier is a sample whose value significantly deviates from the mass of samples. In practical problems, anomalous scenarios, sampling errors, and storing faults can introduce outliers in the dataset. The persistence of the measurement errors generates attributes with a large number of outliers that can introduce bias in the knowledge tree modeling. This stage removes attributes with a large number of outliers. To this end, let $\boldsymbol{D}_2$ be the input matrix with size $m_0 \times |\mathcal{N}_2|$. In addition, consider a function that indicates whether the i-th element of $\boldsymbol{a}$ is an outlier:

$$\text{outlier}(\boldsymbol{a})(i) = \begin{cases} 1, & \text{if } \left(\boldsymbol{a}(i) > \zeta^+\right) \text{ or } \left(\boldsymbol{a}(i) < \zeta^-\right), \text{ and} \\ 0, & \text{otherwise,} \end{cases} \quad (7)$$

with thresholds defined as $\zeta^{\pm} = \text{median}(\boldsymbol{a}) \pm 3\text{MAD}(\boldsymbol{a})$. KLNX builds a matrix whose j-th column contains the outlier locations at the respective j-th attribute, i.e.:

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

This step obtains the number of outliers in each attribute as

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

Next, KLNX extracts the column indices whose number of outliers is lower than $\text{median}(\boldsymbol{o}_c) + 3\text{MAD}(\boldsymbol{o}_c)$, i.e.:

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

Then, the attributes with a low number of outliers are obtained as $\boldsymbol{D}_3 = \boldsymbol{D}_2(:, \mathcal{N}_3)$. It can be noted that the outlier detection stage is based on parameters extracted directly from attribute statistics, thus, this stage does not resort to parameter tuning.

B. Training set preparation subphase

The second subphase of the knowledge tree modeling extracts both training data and training labels required 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 $n_4 = |\mathcal{N}_4|$. This subphase firstly detects outliers in the experiments (i.e., the rows of $\boldsymbol{D}_4$). Then, a feature selection method based on a clustering technique is implemented. Finally, a discretization is applied to the target KPI to generate the training labels.

1) Outlier detection in experiments: In this work, we assume that samples around the central tendency of each attribute are associated with the normal behavior. Hence, training samples with outliers can introduce bias in the knowledge model, and so it should be avoided. To this end, KLNX builds a matrix from $\boldsymbol{D}_4$ whose columns are given by

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

Additionally, the vector containing the number of outliers in each row is obtained as

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

The subset of row indices with zero outliers are identified, 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, $\boldsymbol{T} = \boldsymbol{D}_4(\mathcal{M}_4, :)$. This stage removes experiments considered outliers from the target KPI and attributes.

2) Feature selection: Redundant information in attributes typically affects classifier learning. This problem becomes more accentuated as the number of features increases. To overcome this drawback, we include a feature selection technique that extracts a reduced set of attributes from the available data. Notice that feature selection belongs to the field of dimensionality reduction (DR) techniques that extract a subset of available attributes following a selection rule [8].

To select the training attributes, 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 denotes the number of attributes. Then, a correlation indicator matrix $\boldsymbol{C}'$ is built as

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

where $\mathbf{1}_{n \times n}$ is a matrix with dimensions $n \times n$ with one-valued components, $\text{abs}(\mathbf{A})$ is a matrix with the absolute value of the elements of $\boldsymbol{A}$, and $\boldsymbol{C}$ denotes the correlation matrix with dimensions $n_4 \times n_4$. More precisely, each entry of $\boldsymbol{C}$ at the location (i, j) corresponds to the Pearson correlation coefficient between $\boldsymbol{T}(:,i)$ and $\boldsymbol{T}(:,j)$, this is,

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

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

At this stage, we focus on clustering attributes that are correlated. To do that, we leverage multidimensional scaling (MDS), which projects multidimensional points into a space with smaller dimensions attempting to preserve a distance (dissimilarity) measure among points provided as in a matrix [9]. Then, in this step, MDS is used to map the n_4 attributes into a 2D Euclidean space using $\boldsymbol{C}'$ as the distance matrix of reference. As a consequence, the distance between points in the space will reflect the correlation among them.

Then, to select the training attributes, a model-based clustering method is applied to the projected attributes. In essence, the model-based clustering assumes that the points belonging to each cluster follow a 2D Gaussian distribution, therefore, the entire dataset is modeled as a mixture of normal distributions. Thus, the clustering implements an expectation-maximization (EM) algorithm to compute the maximum likelihood estimates describing each distribution. In addition, the model-based clustering selects the optimal model according to the Bayes information criterion (BIC) [10].

It is worth noting that attributes exhibiting high correlations with respect to the target KPI can introduce bias in the knowledge tree during the training stage. More precisely, the decision structure may depend on features that do not provide relevant information about the underlying processes affecting the network performance. Hence, the next step discards the cluster containing the target KPI. For each of the remaining clusters, KLNX extracts the attribute with the closest distance with respect to its cluster centroid. More 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 selected clusters. The set of selected attributes is therefore

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

Finally, the training set is obtained as $\boldsymbol{X}_{train} = \boldsymbol{T}(:, \mathcal{N}_5)$.

3) Discretization: Target KPI samples are typically represented using floating-point numbers whose values may lay on an infinite set. Since our goal is to detect and classify performance anomalies, the proposed methodology discretizes the target KPI samples into a finite number of class labels, where each category should describe a particular operation scenario. To this end, a discretization model $f_\theta(\cdot)$ is built whose binning intervals are estimated 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 II-B1. In addition, KLNX resorts to proportional discretization to determine the number of target classes automatically [11]. Under the proportional discretization, the number of categories depends on the input vector size, in other words, $w = \lfloor (\log_2 n)/2 \rfloor$, where w is the number of target classes, n is the size of the input vector, and $\lfloor a \rfloor$ is the floor operator that extracts the integer part of a. Furthermore, the discretization uses the k-means binning strategy to map the continuous KPI values into a finite number of categories. The discretization strategy assigns the class labels based on the distances with respect to the 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.

C. Knowledge model training subphase

This subphase aims at building a knowledge model from the extracted training data to detect network performance anomalies. This approach builds a decision tree classifier to characterize the network behavior. In general, decision tree classifiers exhibit lower accuracy than those yielded by other black-box classification methods. However, we select the decision tree model because the tree structure is interpretable and easy to understand.

In this work, the decision tree is built using the training set $\boldsymbol{\Gamma} = \{\boldsymbol{X}_{train},\, \boldsymbol{y}_{train}\}$, where $\boldsymbol{X}_{train}$ is the input data matrix with dimensions $m_\ell \times n_\ell$, where m_ℓ is the number of training samples and n_ℓ is the number of attributes selected by the feature selection procedure. Furthermore, $\boldsymbol{y}_{train}$ is an m_ℓ-dimensional vector containing the ground truth labels obtained from the discretization. The decision tree is trained by implementing the classification and regression tree (CART) algorithm with the Gini impurity as the loss function to be optimized [12]. Specifically, Gini impurity assesses the misclassification rate of the output label at a given tree node, which is computed as $G = 1 - \sum_{i=1}^{k} p_i^2$, where k is the number of output labels, and p_i is the rate between the number of training labels assigned with the i-th class and the total number of samples evaluated by the tree node.

Notice that the number of training samples required for decision trees doubles as a tree level is aggregated, hence, the depth of the decision tree is constrained to $\lfloor (\log_2 m_\ell)/2 \rfloor$ to avoid overfitting. In addition, the minimum number of training samples per leaf is limited to five [13].

Finally, a cost-complexity pruning is applied to the decision tree to reduce the probability of overfitting. Cost-complexity pruning is an algorithm that removes unnecessary tree nodes by balancing the trade-off between the classification error and the model size. This algorithm minimizes a linear cost-complexity function with a parameter $\alpha \geq 0$, referred to as the complexity parameter, that controls the relative influence of the model complexity with respect to the tree size. In this regard, a cross-validation procedure is implemented to automatically select the proper α among a set of effective values. More precisely, cross-validation evaluates the cost-complexity function across an α interval. For each value of α, the procedure split the training set into smaller subsets. This method implements multiple training and testing operations, where one subset is considered to train the model and the remaining ones are used for testing. Afterward, the average classification accuracy is computed. Notice that a different tree structure is evaluated for each value of α. Then, cross-validation selects the tree structure with the best classification accuracy average. Finally, 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 $g_{\boldsymbol{\phi}}(\cdot)$.

III. Network performance anomalies

After the training stage, KLNX tests the knowledge model with the goal of detecting network performance anomalies. To this end, the proposed approach evaluates the decision tree classifier using the entire set of experiments. Specifically,

the set of test samples is obtained as $\boldsymbol{X}_{test} = \boldsymbol{D}(:, \mathcal{N}_5)$, where $\boldsymbol{X}_{test}$ is a submatrix with dimensions $m_0 \times |\mathcal{N}_5|$ that extracts the indicators 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 uses the discretization model $f_\theta(\cdot)$ to obtain the reference labels, i.e. this method obtains $\boldsymbol{y} = f_\theta(\boldsymbol{z})$. Note that the classification engine has been optimized to identify normal operation scenarios. Thus, misclassified experiments may be considered anomalous. Moreover, the difference between the predicted label and the discretized KPI can offer information about the deviation from the normal operation of the sample under test. Next, we introduce the steps of the anomaly detection procedure.

A. Identification of anomalous scenarios

This stage estimates the element-wise difference between the set of class labels predicted by the knowledge model and the set of discretized KPI samples, i.e. $\boldsymbol{y}_{diff} = \boldsymbol{y}_{pred} - \boldsymbol{y}$. Notice that, for target KPIs similar to the session duration, a lower value of the class label indicates a better network performance. In this context, three scenarios are identified: (i) the label predicted by the knowledge tree matches with respect to the discretized KPI, i.e. $\boldsymbol{y}_{diff}(i) = 0$; (ii) the class label predicted indicates a worse network performance compared to the discretized KPI, i.e. $\boldsymbol{y}_{diff}(i) > 0$; or (iii) the network performance predicted is better than the 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). Hence, KLNX 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 $i = 1, \ldots, m_0$. This approach can be also applied when a larger KPI value indicates a better network performance as is the case of the TCP throughput. In this case, every entry of the vector of anomalous scenarios reduce to (i) $\boldsymbol{y}_{anm}(i) = 0$ when the predicted label is less than or equal to the respective discretized KPI, and (ii) $\boldsymbol{y}_{anm}(i) > 0$ otherwise.

B. Identification of relevant features

Notice that the training stage considers a reduced set of attributes to optimize the knowledge model $g_{\boldsymbol{\phi}}$. However, the methodology requires a large number of attributes to classify anomalies. Hence, this phase recovers some of the attributes discarded by the preprocessing phase. Specifically, it retrieves the matrix $\boldsymbol{D}_3$ obtained by the outlier detection in attributes (Section II-A3) with dimensions $m_0 \times |\mathcal{N}_3|$.

To quantify the relevance of each attribute with respect to the vector of anomalous scenarios $\boldsymbol{y}_{anm}$, this work computes the miscoding (mscd) index. This metric has been recently proposed in [14] to assess the relationship level between a sequence of attributes and the output vector. To compute the metric, consider the matrix $\boldsymbol{D}_3$ whose $|\mathcal{N}_3|$ columns correspond to the input attributes. Therefore, the miscoding index between the i-th attribute $\boldsymbol{D}_3(:, i)$ and the vector of anomalous scenarios $\boldsymbol{y}_{anm}$ is estimated as

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

for $i = 1, \ldots, |\mathcal{N}_3|$, with $\mathrm{NCD}(\boldsymbol{a}, \boldsymbol{b})$ 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 attributes 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 $\boldsymbol{e}(0) = \mathrm{mscd}(\boldsymbol{D}_3(:, i_0), \boldsymbol{y}_{anm}), \boldsymbol{e}(1) = \mathrm{mscd}(\boldsymbol{D}_3(:, i_1), \boldsymbol{y}_{anm}), \ldots, \boldsymbol{e}(n_r) = \mathrm{mscd}(\boldsymbol{D}_3(:, i_{n_r}), \boldsymbol{y}_{anm})$ and $\boldsymbol{e}(0) \geq \boldsymbol{e}(1) \geq \ldots \geq \boldsymbol{e}(n_r)$. Therefore, the subset of relevant attribute indices can be written as

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

IV. Classification tree modeling

KLNX uses the set $\mathcal{N}_r$ of attributes extracted from the identification of relevant features to classify performance anomalies. In addition, KLNX considers the entire set of experiments included in the loaded dataset to build a classification tree. In particular, KLNX uses class labels generated by a stage that detects problematic experiments and features.

A. Detection of problematic attributes and experiments

Once selected the set $\mathcal{N}_r$ of relevant attributes, KLNX implements a procedure to identify attributes and operation scenarios that could induce anomalies. To this end, this stage applies a one-dimensional (1D) clustering to the normalized samples of every relevant attribute. More precisely, we resort to the model-based clustering method that determines the optimal model according to the BIC criterion [10]. Each relevant feature is scaled using the 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 attribute vector, $\mu_{\boldsymbol{a}}$ and $\sigma_{\boldsymbol{a}}$ are, respectively, the sample median and the sample standard deviation of $\boldsymbol{a}$.

KLNX estimates the anomaly density of each cluster, i.e., the mean error in the cluster normalized to the overall mean error (cf. Section III-A for the definition of error). Assume that the clustering generates a vector $\boldsymbol{y}_{\mathrm{clust}}$ 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

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

for $c_j = 1, \ldots, \mathcal{C}_J$. For each attribute the methodology obtains a set of densities $\{\rho_{c_j}\}_{c_j=1}^{\mathcal{C}_J}$. If at least one density index is greater than a threshold ε, the proposed approach identifies the attribute as a problematic feature. Furthermore, experiments included in clusters with $\rho_{c_j} > \varepsilon$ are labeled as problematic samples with a label `T$P`, where `$` is the number assigned to the problematic feature under consideration. Otherwise, experiments in clusters with $\rho_{c_j} \leq \varepsilon$ are labeled as non-problematic samples, with a label `T$N`. An anomaly class can therefore be described by listing problematic features.

B. Classification model training

This stage trains the anomaly classification engine using the attributes used to train the knowledge tree and the relevant attributes based on the miscoding metric. In essence, we obtain an attribute 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 $\boldsymbol{D}$ that contains the column indices $\mathcal{N}_c$, i.e., $\boldsymbol{\chi} = \boldsymbol{D}(:, \mathcal{N}_c)$. This training set attempts to

consider the information that builds the first decision tree 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 IV-A. This labeling generates two types of classes: (i) the `Compliant` class that identifies the samples that do not contain any problematic attribute (ii) or a class labelled with a concatenated semicolon list of problematic and non-problematic attributes. For example, consider that the method detects four problematic attributes with number assignation: (1) `packet_loss`, (2) `maximum_segment_size`, (3) `round_trip_time`, (4) `start_SNR`. An experiment with class `T1P;T2N;T3N;T4P` indicates that variables `packet_loss` and `start_SNR` are attributes that could induce an anomaly in the experiment of interest.

Subsequently, this stage trains a decision tree to model the anomaly classification engine. We use a decision tree classifier due to its interpretability. In other words, the decision tree structure describes the attribute thresholds related to a specific type of anomaly. In this case, the proposed approach optimizes the decision tree by implementing the CART algorithm with the training set $\mathbf{\Pi} = \{\boldsymbol{\chi}, \boldsymbol{\zeta}\}$. The training stage also uses cost-complexity pruning to remove unnecessary tree leaves. Finally, the nonlinear function performed by the classification engine is represented with $h_\varphi(\cdot)$.

V. Datasets

We evaluate KLNX on both real and synthetic data. Since the synthetic data attempt to reproduce behaviors similar to those yielded by the real data, we first describe the real dataset. Then, we outline the modeling of the synthetic data.

A. Nokia dataset

This dataset was collected by Nokia in 2019 to test the performance of 4G mobile networks in various European countries [15]. The Nokia dataset includes TCP traffic profiles and measurements of radio variables when a user device downloads a 3 MB file (`HTTP_FILE_DL`). Each dataset row contains information about a single test. Overall, there are 1730 rows and 1326 attributes (including, for most parameters measured, the minimum, maximum, average, etc., observed in the experiment).

B. Synthetic data modeling

To evaluate the behavior of the methodology in a more controllable way, we also build a testbed that randomly generates each attribute vector with entries following a particular statistical model. The TCP throughput B_{tcp} (in bytes/s) is then computed analytically using the simple Mathis model [16],

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

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

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

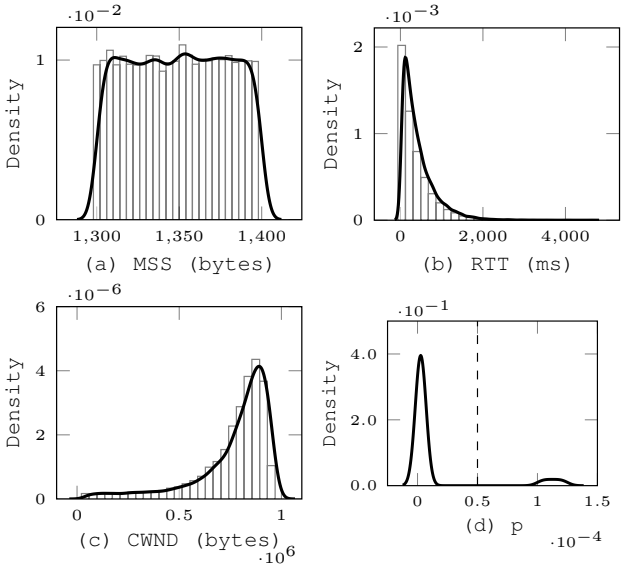


Figure 2: Histogram and kernel density estimation (KDE) of synthetic samples for the parameters (a) MSS, (b) RTT, (c) $CWND$, and (d) p (KDE only).

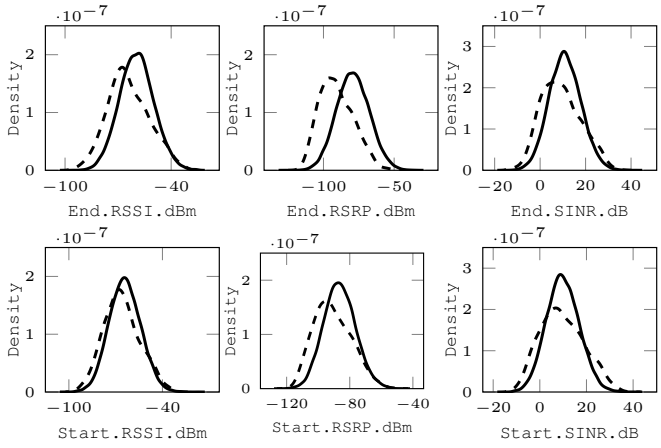


Figure 3: KDE of different radio parameters for both the Nokia dataset (dashed lines) and the synthetic data (continuous lines).

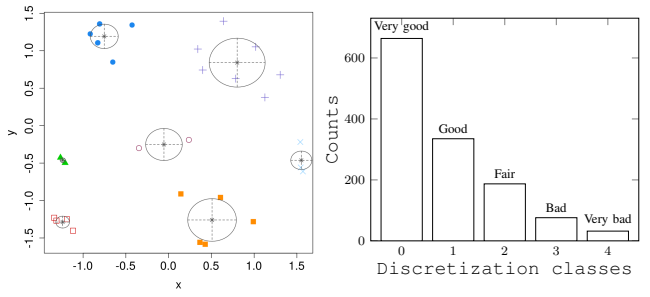
Notice that we generate just four TCP attributes, i.e., MSS, RTT, $CWND$, p, which greatly simplifies the possibility to illustrate and evaluate the operation of KLNX.

Synthetic parameters are generated so as to resemble the behavior observed in the Nokia dataset. The MSS is drawn from a uniform probability density function (pdf), i.e., $MSS \sim U(lb_{mss}, ub_{mss})$, with bounds $lb_{mss} = 1300$ bytes and $ub_{mss} = 1400$ bytes. Fig. 2(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 [17]. 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. 2(b)).

To generate the synthetic samples of $CWND$ and p, we use a ϵ-contaminated mixture model [7] with a uniform and a



(a) (b)

Figure 4: (a) Clustering obtained by the model-based technique for the Nokia dataset. (b) Histogram of the training labels obtained by the discretization model for the Nokia dataset.

(shifted and mirrored) log-normal distribution, i.e., with CDF given by

$$F_{_{cwnd}}(x) = (1-\epsilon_{_{cwnd}})\,U\left(lb_{_{cwnd}}, ub_{_{cwnd}}\right)(x) * \epsilon_{_{cwnd}}\text{Lognormal}\left(\mu_{_{cwnd}}, \sigma_{_{cwnd}}\right)(1.0\cdot 10^6 - x), \quad (22)$$

where the log-normal distribution has mean $\mu_{_{cwnd}}$ and standard deviation $\sigma_{_{cwnd}}$, and is mirrored around the value of 1 MB (cf. Fig. 2(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 $\epsilon_{_{cwnd}}$ is set to 0.90, $\sigma_{_{cwnd}} = 0.65$, and $\mu_{_{cwdn}} = \log(\bar{\mu}_{_{cwnd}}) + \sigma^2_{_{cwnd}}$, where $\bar{\mu}_{_{cwdn}} = 0.1$ MB is the desired mode of the log-normal.

So far, we have described synthetic attributes reproducing what is observed in the Nokia dataset. Additionally, we select the packet loss probability (p) to induce anomalous scenarios in a controllable way with an ϵ-contaminated 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 $\bar{\mu}_{_p} = 2.50 \times 10^{-6}$, $\sigma_{_p} = 0.20$, $lb_{_p} = 1.00 \times 10^{-4}$, 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 $\epsilon_{_p}$ exhibits much higher packet loss probabilities, that should be identified as anomalous. Fig. 2(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 attributes 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 Nokia dataset and a synthetic dataset are displayed in Fig. 3.

In the synthetic dataset, anomalies are forced by means of abnormal values of the loss probability, which are generated with probability $\epsilon_{_p}$. 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.

VI. Results and Analysis

We implemented KLNX using Python code and the Scikit-learn library [18]. Our KLNX source code for anomaly detection and classification will be released as open-source.

A. Nokia dataset

For this dataset, we select the session duration as the target KPI. Fig. 4(a) displays the seven clusters detected by the model-based technique from the projected correlation matrix in the feature selection step (cf., Section II-B2). This figure also shows every cluster centroid and the circles of the corresponding standard deviations. Recall that the attribute selection stage removes the cluster containing the target KPI. In addition, the methodology selects the attribute that is closest to the centroid in each of the other six clusters. More precisely, the features selected are listed below:

1) `initial_window_bytes_b2a`,
2) `End.SINR.dB`,
3) `Time.to.First.Byte.s`,
4) `pushed_data_pkts_b2a`,
5) `End.RSSI.dBm`,
6) `rexmt_data_pkts_b2a`.

After applying the preprocessing stage, KLNX obtains a subset with $m_\ell = 1,294$ experiments and $n_\ell = 6$ attributes to train the knowledge model $g_\phi(\cdot)$. Fig. 4(b) displays the histogram of the training labels output by the discretization process. In this case, the number of class labels is $w = \lfloor(\log_2 m_\ell)/2\rfloor = 5$. Once trained the knowledge model, we obtain a decision tree whose flowchart is depicted in Fig. 5. For every experiment, the decision tree firstly compares the `initial_window_bytes_b2a` parameter with respect to the threshold (2,588 bytes) at the root node. If the inequality rule is true, the decision tree structure moves to the left side in the next tree level, otherwise, the flowchart moves to the right side. For example, the knowledge tree assigns the "Very bad" label to scenarios with `initial_window_bytes_b2a` greater than 1,294 bytes or lower or equal than 2,588 bytes, and `Time.to.First.Byte.s` greater than 0.58 s.

Afterward, the methodology evaluates the knowledge tree using the entire set of experiments and obtains the vector of anomalies. The histogram of the vector of anomalous scenarios $\boldsymbol{y}_{anm}$ is shown in Fig. 6(a). It can be observed that the number of predicted anomalies decreases as the difference between the predicted labels and the discretized KPIs increases. For this dataset, the knowledge model detects 611 anomalies. Then, the methodology considers both the data matrix obtained at the outlier detection in attributes $\boldsymbol{D}_3$ and the vector $\boldsymbol{y}_{anm}$ to identify relevant features. For this dataset, the number of relevant attributes is $n_r = 25$. Fig. 6(b) shows the sorted miscoding estimates that quantify the relationship level between features and $\boldsymbol{y}_{anm}$. These attributes are evaluated according to the order exhibited by the miscoding coefficients.

Then, the methodology applies the procedure to detect both the problematic attributes and problematic experiments. For this dataset, KLNX identifies seven problematic attributes that are listed below:

1) `Time.to.First.Byte.s`,
2) `ack_pkts_sent_a2b`,
3) `avg_win_adv_a2b`,
4) `duplicate_acks_a2b`,
5) `max_win_adv_a2b`,
6) `pure_ack_sent_a2b`,
7) `truncated_data_b2a`,

Fig. 7 displays the scatter plots of the attribute value versus $\boldsymbol{y}_{anm}$ for three problematic attributes. This figure also shows the cluster of problematic experiments. As can be seen,

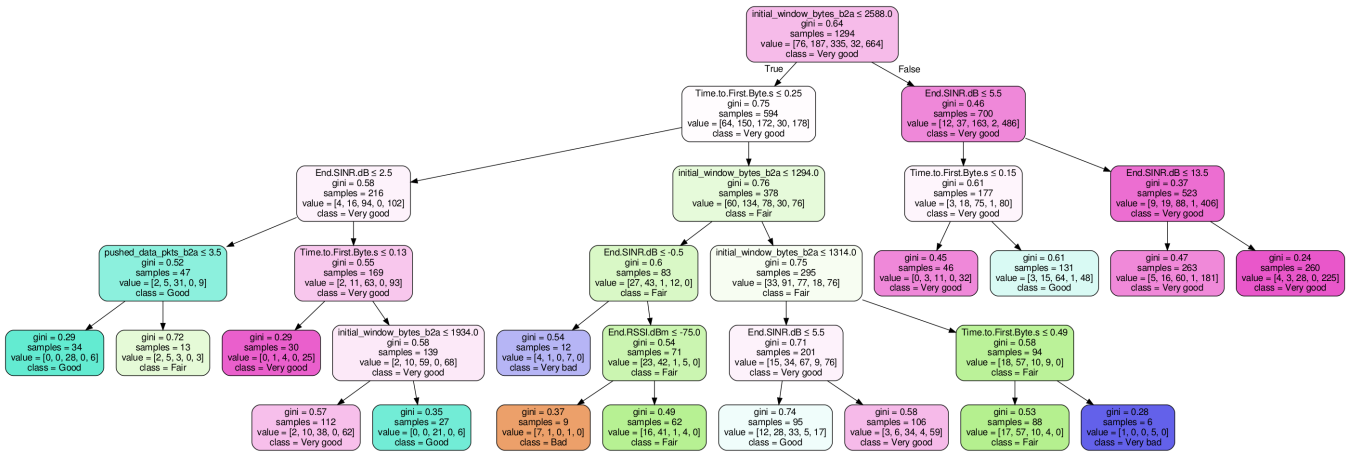


Figure 5: Schematic representation of the knowledge tree built for the Nokia dataset. Each decision tree node (leaf) contains the Gini impurity measure, the number of evaluated training samples, and the selected output class label.

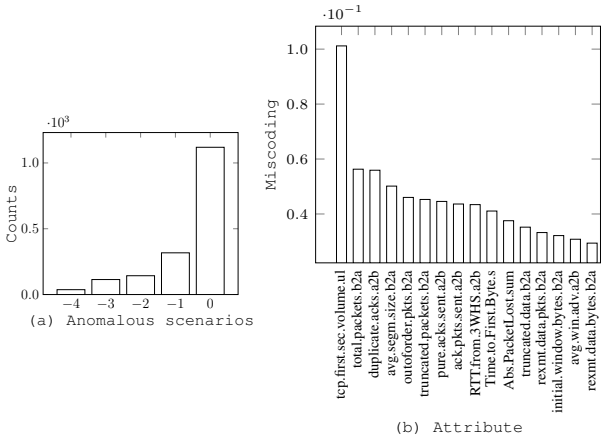


Figure 6: (a) Histogram of the vector of anomalous scenarios $\boldsymbol{y}_{anm}$, and (b) Miscoding scores of the 16 most relevant features.

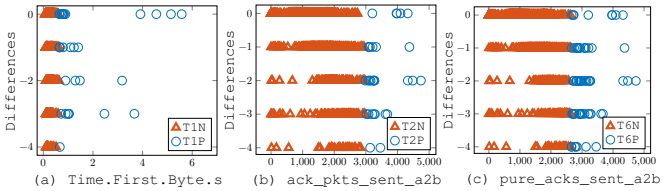


Figure 7: Scatter plot of the attribute value versus the vector of anomalous scenarios $\boldsymbol{y}_{anm}$ for three problematic attributes. Non problematic experiments are labeled as `T$N` and problematic samples are categorized as `T$N`, where `$` is the attribute number.

KLNX detects as problematic samples those scenarios with `Time.to.First.Byte.s` greater than 1s, `ack_pkts_sent_a2b` greater than $3,000$, and `pure_ack_sent_a2b` greater than $3,000$. Then, the methodology implements the CART algorithm to fit the classification model from the input training data $\boldsymbol{\chi}$ and the output training labels $\boldsymbol{\zeta}$. Specifically, the classification tree learns the rules to detect anomalous operation scenarios and identify the attributes that can induce performance anomalies. The decision tree structure of the classification model is not displayed due to its large size. If we analyze in detail the decision structure of the knowledge tree shown in Fig. 5, we observe that radio features such as SINR and RSSI have been selected to classify the experiment performance. Other features such as TCP initial Window and Time to First Byte characterize the server used for each

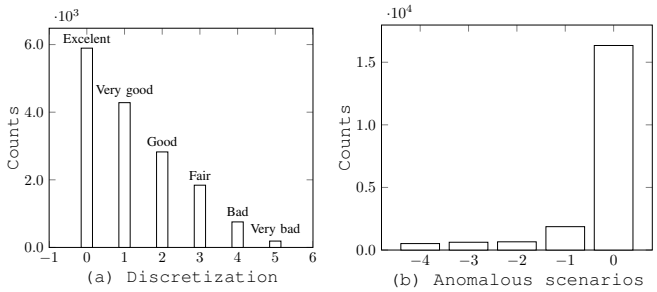


Figure 8: (a) Histogram of the training class labels yielded by the discretization process, (b) Histogram of the vector of anomalous scenarios $\boldsymbol{y}_{anm}$.

test. These features are selected in the classification stage to identify performance anomalies causing deviations from the knowledge tree. The only exception in this example could be `Time.to.First.Byte.s` that is present in both models. We interpret that this particular feature contains information related to the different servers used as well as delay-related anomalies.

Remarkably, the results of the analysis carried out by means of KLNX 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 KLNX can identify anomalies, but does not tell if the results are accurate or not. For a more objective evaluation of the results achievable with KLNX, we next consider synthetic data in which we purposely introduce anomalies, and hence can use them as ground truth.

B. *Synthetic datasets*

For the synthetic dataset, we also select the session duration as the target KPI. A first experiment evaluates the performance of the proposed methodology against outliers in packet loss probabilities. To this end, we generate the p measurement set using the mixture model (23) with $\epsilon_p = 0.10$. As can be observed in Fig. 2(d), the mixture model generates a bimodal distribution with two separated modes. In addition, outliers in packet loss probability could lead to large session times. This behavior favors the outlier detection in experiments at the preprocessing stage and the 1D clustering of attribute samples at problematic experiment identification.

In particular, the testbed generates a synthetic dataset with $m_0 = 20,000$ experiments and $n_1 = 10$ attributes to

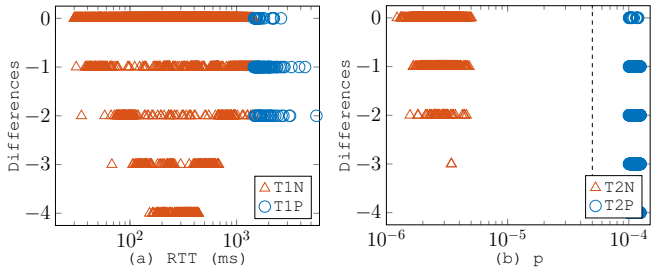


Figure 9: Scatter plots of the experiment clustering for each detected attribute from the synthetic dataset.

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

Class label	Type of problem
Compliant	No problem
T1PT2N	RTT problem
T1NT2P	p problem
T1PT2P	RTT problem and p problem

evaluate the proposed approach. For this dataset, the data preprocessing stage extracts $m_\ell = 15,785$ experiments and $n_\ell = 4$ attributes (`RTT`, `MSS`, `CWND`, and `End.RSSP.dBm`) to train the knowledge model. It should be noted that the preprocessing stage discards the entire set of experiments (100%) with large packet loss probability values drawn from the uniform distribution of the model (23). Observe that the number of class labels outputted by the discretization process is set to 6. Fig. 8(a) displays the histogram of the training labels provided by the discretization process.

Then, the knowledge tree $g_\phi(\cdot)$ is optimized using the CART algorithm from the training data $\mathbf{\Gamma} = \{\boldsymbol{X}_{train}, \boldsymbol{y}_{train}\}$ extracted by the preprocessing stage. The decision tree structure of the knowledge model is omitted due to space limitations. After testing the knowledge model using the entire set of $m_0 = 20,000$ experiments, the proposed method obtains the vector of anomalous scenarios whose histogram is depicted in Fig. 8(b).

Due to the limited availability of synthetic attributes, KLNX selects the entire set of original attributes to detect problematic attributes, i.e. $n_r = n_\ell = 10$. Afterward, the methodology identifies the attributes that could induce anomalous operation scenarios. To this end, the proposed approach detects the attribute clusters that exhibit dense concentrations of misclassified samples. For this dataset, p and RTT are detected as attributes that can cause anomalous session times. Subsequently, this module detects both normal and problematic experiments for each identified attribute. Fig. 9(a) and 9(b) show the experiment clustering for each identified attribute and the corresponding labeling (`T$N` or `T$P`). From this identification, the procedure builds the output labels for the anomaly classification engine. More precisely, this procedure generates four different classes that are shown in Table II with the respective type of problem.

Then, the anomaly classification engine is fitted using the entire set of experiments and attributes. Fig. 10 depicts the decision structure of the anomaly classification engine. As can be seen in this figure, the classifier sets a threshold 5×10^{-5} to separate packet loss probabilities into two classes: normal and problematic. For example, for $p \leq 5 \times 10^{-5}$, the classifier identifies operation scenarios that are not affected by the packet loss probability. Dashed lines in Fig. 2(d) and Fig. 9(b) correspond to the boundaries determined by

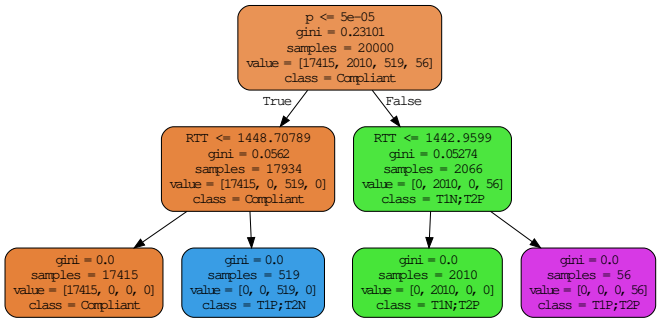


Figure 10: Structure of the anomaly classification decision tree built for the synthetic dataset.

the classifier, i.e., $p = 5 \times 10^{-5}$. 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, KLNX identifies anomalous scenarios related to the `RTT` attribute, due to the queue of the exponential distribution used. Additionally, although the exponential distribution is not multimodal, the 1D clustering at the problematic experiment identification detects two clusters with problematic samples. Fig. 10 shows how the introduced anomalies (p-anomalies) are successfully classified by the root node but also how another performance anomaly is 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.

VII. Related Work

Various anomaly detection methods have been reported in the context of communication networks. Techniques based on signal processing tools have been introduced to detect spikes in network traffic time series [19]–[22]. In particular, the recognition models implemented by these methods are tailored to the data at hand and they also require fine parameter tuning. Anomaly detection based on supervised learning methods [4], [23], [24] requires ground-truth class labels that are not always available. Furthermore, anomaly detection techniques based on unsupervised learning methods have been recently reported [5], [25], [26]. These methods provide outstanding solutions and do not need ground truth labels. However, the performance of unsupervised techniques is severely affected by parameter tuning and the detection process usually does not provide information about features that can be related to the network's abnormal behaviors. It should note that KLNX is an unsupervised methodology with a self-tune of parameters based on data statistics. Moreover, this framework includes an anomaly classification stage that provides information about the operation scenarios considered as anomalies and the set of attributes causing these anomalies, which is a fundamentally novel contribution of our work.

Anomaly detection methods typically analyze a particular problem such as server failures, transient congestions, and network overloads, among others [3]. Recently, abrupt changes were characterized in [5] as anomalous events affecting various KPIs. However, these methods are not focused on covering scenarios where network performance is affected by multiple network aspects (e.g., TCP performance, packet loss, and radio behavior). On this matter, we have identified three categories of anomalies: (i) *operative anomalies*, (ii)

sampling anomalies, and (iii) *modeling anomalies*. Specifically, *operative anomalies* consist of events whose performance indicators and KPIs significantly deviate from the values for which the network was designed. This class of anomaly includes operation scenarios with low throughput, large session times, and bad performance indicators. We have tested KLNX with respect to *operative anomalies* with real and synthetic datasets. Secondly, *sampling anomalies* are abnormal scenarios caused by measurement errors in one or multiple performance indicators. This kind of anomaly can be present in the real dataset tested in this paper, and we have tested it also via synthetic data, although the results are not shown here due to lack of space. Finally, *modeling anomalies* are operation instances whose KPI cannot be explained by the performance indicators, e.g., operation scenarios with large session times but with good performance indicators [14]. This class of anomalies will be considered in future works.

VIII. 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 clean and explainable (KLNX) 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 attributes 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 attribute samples obey well-defined statistical models and the KPI sets follow a known TCP model. The performance of the proposed methodology was evaluated on synthetic and real datasets. In this sense, the proposed approach efficiently detected the performance anomalies and provided information about the set of attributes that can cause network misfunctioning. In addition, the decision trees generated interpretable structures enabling the understanding of both the network's normal behavior and the 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.

IX. Acknowledgements*

The work was supported by the DiscoLedger project (PDC2021-121836-I00) funded by MCIN/AEI /10.13039/501100011033 and the European Union through the Next Generation EU/ PRTR program.

References

[1] Mansoor Shafi, Andreas F. Molisch, Peter J. Smith, Thomas Haustein, Peiying Zhu, Prasan De Silva, Fredrik Tufvesson, Anass Benjebbour, and Gerhard Wunder, "5G: A tutorial overview of standards, trials, challenges, deployment, and practice," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 6, pp. 1201–1221, 2017.

[2] Marina Thottan and Chuanyi Ji, "Anomaly detection in IP networks," *IEEE Transactions on Signal Processing*, vol. 51, no. 8, pp. 2191–2204, 2003.

[3] Monowar H. Bhuyan, Dhruba Kumar Bhattacharyya, and Jugal Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 303–336, 2014.

[4] Mohamed Moulay, Rafael A. García Leiva, Pablo J. Rojo Maroni, Javier Lazaro, Vincenzo Mancuso, and Antonio Fernández 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*. 2020, pp. 780–786, IEEE.

[5] Guang Yu, Zhiping Cai, Siqi Wang, Haiwen Chen, Fang Liu, and Anfeng Liu, "Unsupervised online anomaly detection with parameter adaptation for KPI abrupt changes," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1294–1308, 2020.

[6] Christophe Leys, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata, "Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median," *Journal of Experimental Social Psychology*, vol. 49, no. 4, pp. 764–766, 2013.

[7] Peter Huber, *Robust statistics*, vol. 523, John Wiley & Sons, 2004.

[8] Pedram Ghamisi, Naoto Yokoya, Jun Li, Wenzhi Liao, Sicong Liu, Javier Plaza, Behnood Rasti, and Antonio Plaza, "Advances in hyperspectral image and signal processing: A comprehensive overview of the state of the art," *IEEE Geoscience and Remote Sensing Magazine*, vol. 5, no. 4, pp. 37–78, 2017.

[9] Ingwer Borg and Patrick Groenen, *Modern multidimensional scaling: Theory and applications*, Springer Science & Business Media, 2005.

[10] Luca Scrucca, Michael Fop, T Brendan Murphy, and Adrian E Raftery, "mclust 5: clustering, classification and density estimation using gaussian finite mixture models," *The R journal*, vol. 8, no. 1, pp. 289, 2016.

[11] Salvador García, Julián Luengo, José Antonio Sáez, Victoria López, and Francisco Herrera, "A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 4, pp. 734–750, 2013.

[12] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone, *Classification and regression trees*, Routledge, 2017.

[13] Trevor Hastie, Robert Tibshirani, and Jerome Friedman, *The Elements of Statistical Learning*, Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[14] Mohamed Moulay, Rafael Garcia Leiva, Vincenzo Mancuso, Pablo J. Rojo Maroni, and Antonio Fernández 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.

[15] Vincenzo Mancuso, Miguel Peón Quirós, Cise Midoglu, Mohamed Moulay, Vincenzo Comite, Andra Lutu, Özgü Alay, Stefan Alfredsson, Mohammad Rajiullah, Anna Brunström, et al., "Results from running an experiment as a service platform for mobile broadband networks in Europe," *Computer Communications*, vol. 133, pp. 89–101, 2019.

[16] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm," *SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 3, pp. 67–82, jul 1997.

[17] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, *An introduction to statistical learning*, vol. 112, Springer, 2013.

[18] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al., "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[19] Wei Lu and Ali A Ghorbani, "Network anomaly detection based on wavelet analysis," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, pp. 1–16, 2008.

[20] Asrul H. Yaacob, Ian Tan, Su Fong Chien, and Hon Khi Tan, "Arima based network anomaly detection," in *2010 Second International Conference on Communication Software and Networks*, 2010, pp. 205–209.

[21] Hiroyuki Kasai, Wolfgang Kellerer, and Martin 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*, vol. 13, no. 3, pp. 636–650, 2016.

[22] Markus Thill, Wolfgang Konen, and Thomas Bäck, "Online anomaly detection on the webscope s5 dataset: A comparative study," in *2017 Evolving and Adaptive Intelligent Systems (EAIS)*, 2017, pp. 1–8.

[23] Nikolay Laptev, Saeed Amizadeh, and Ian 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*, New York, NY, USA, 2015, KDD '15, p. 1939–1947, Association for Computing Machinery.

[24] Dapeng Liu, Youjian Zhao, Haowen Xu, Yongqian Sun, Dan Pei, Jiao Luo, Xiaowei Jing, and Mei Feng, "Opprentice: Towards practical and automatic anomaly detection through machine learning," in *Proceedings of the 2015 Internet Measurement Conference*, New York, NY, USA, 2015, IMC '15, p. 211–224, Association for Computing Machinery.

[25] Juliette Dromard, Gilles Roudière, and Philippe Owezarski, "Online and scalable unsupervised network anomaly detection method," *IEEE Transactions on Network and Service Management*, vol. 14, no. 1, pp. 34–47, 2017.

[26] Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, Jie Chen, Zhaogang Wang, and Honglin Qiao, "Unsupervised anomaly detection via variational auto-encoder for seasonal KPIs in web applications," in *Proceedings of the 2018 World Wide Web Conference*, Republic and Canton of Geneva, CHE, 2018, WWW '18, p. 187–196, International World Wide Web Conferences Steering Committee.