ANALYSIS ON ROUTE INFORMATION FAILURE IN IP CORE NETWORKS BY NFV-BASED TEST ENVIRONMENT

Xia Fei[1], Aerman Tuerxun[1], Jiaxing Lu[1], Ping Du[1], Akihiro Nakao[1]
[1]The University of Tokyo, Japan

NOTE: Corresponding author: Ping Du, duping@g.ecc.u-tokyo.ac.jp

Abstract – *Stable and high-quality Internet connectivity is mandatory for 5G mobile networks. However, the pandemic of COVID-19 has forced global and large-scale staying at home and telecommuting in many countries. The increasing traffic has induced more pressure on networks, devices and cloud data centers. It becomes an essential task for network operators to enable their ability to automatically and rapidly detect network and device failures. We propose a highly practical method based on highly practical technology. Our method has a high generalization ability that can efficiently extract features from large-scale unstructured data and ensure high accuracy prediction. First, 997 useful features are extracted from 28GB-per-day network logs. Then, a differential approach is employed to preprocess the extracted features so as to highlight the differences between normal and abnormal states. Third, those features are refined based on the feature importance we calculated. According to our experiment, the proposed feature extraction and refinement method can reduce computation without degrading the performance. Among the five types of failures, we achieve a 100% recall rate in four types and the rest can also reach 71%. Overall, the total average prediction accuracy of the proposed method is 94%.*

Keywords – Core network, failure detection, route information, machine learning

1. INTRODUCTION

The pandemic of COVID-19 has forced global and large-scale staying at home and telecommuting in many countries. The implementation of social restrictions increases Internet traffic, particularly the traffic of remote working, web meetings, and online education. For instance, Netflix has faced a surge in subscriber numbers, with almost 16 million people signing up for accounts in the first three months of 2020 [1]. Zoom's daily active users spiked to 200 million in March 2020, up from 10 million in December 2019 [2]. Such increasing network traffic has induced more network and device failures than before. For example, it is reported that Google has suffered an estimated $1.7M loss in advertisement revenues during their "outage" in December 2020 [3]. Thus, how to automatically and rapidly detect network and device failures has become an essential problem in daily operation.

Network technologies such as 5G, have dramatically changed the telecommunication environment that brings faster speed experience to us. 5G mobile networks require stable, high-quality Internet connectivity, but when a failure happens, the consequences of that failure are extremely serious. In addition, since the Internet is operated mutually among ISPs, even if a failure occurs in a certain ISP domain, the failure spreads rapidly all over the world. However, only experienced ISPs can deal with such a network failure that affects the world. It is desirable that anomaly detection could be performed automatically and promptly. The IP backbone network of one ISP is interconnected with others via Border Gateway Protocol(BGP) routers. A BGP router needs to continuously update the route information from the received internal/external route information and provide appropriate feedback. Thus, these BGP routers play very important roles in 5G services, and in order to maintain a certain level of service, it is desirable to immediately detect hardware and software defects and malfunctions. Moreover, increasing network traffic also brings challenges to data-based optimization. Recent hot AI technologies provide novel approaches that are able to migrate our focus of work from fault handling to fault prediction, which allows operators to take precautions in advance.

Based on the data sets [4] provided by KDDI Corporation, we propose an efficient method to predict network and device failures from large amounts of unstructured log files in real time. Our proposal contains three main steps: *Feature Extraction*, *Feature Refinement*, and *Feature Reduction*. In the Feature Extraction step, we extract 997 features from 28GB per day of unstructured log files, and merge tagged features from the following three kinds of JSON log files: *physical-infrastructure*, *virtual-infrastructure*, and *network-devices*. As for the BGP-related entries, we use the number of next-hops in each array and corresponding prefixes as features.

In order to derive metrics that are changed when there is an the occurrence of a failure, we highlight the difference between normal and abnormal entries and define a new feature named Differential Data to reflect the variations between abnormal data and normal data. After the data processing, one CSV file that could be utilized for training or evaluation is generated.

For the sake of importance analysis, the XGBoost [5] model is trained by us to calculate the importance scores which work as the reference in the *Feature Reduction*

phase via observing the changes in accuracy which are trained by different numbers of features, we have two observations: (1) The highest accuracy is 94% when the number of features is more than 150; (2) Accuracy could achieve 93% if we use only the top 30 most important features, without obvious performance degradation.
According to our evaluation, we achieve a 100% recall rate when detecting the following four network and device failures: *Node-Down*, *Interface-Down*, *Ixnetwork-BGP-Injection*, and *packet loss & delay*. There is a 71% recall rate of Ixnetwork-BGP-Hijacking detection, while the total average accuracy of our proposal is 94%. XGBoost, Random Forest, and LightGBM [6] have been demonstrated in our experiments that they outperform other methods in terms of training and inference time.
In summary, the main contributions in this paper are as follows.

- First, we define a staged method including feature extraction from unstructured network logs, a differential approach to highlight the differences between normal and abnormal states and several ML models to realize failure classification.
- Then, we apply the staged method to six popular machine learning algorithms, including Decision Tree (DT), XGBoost, LightGBM, Multilayer Perceptron (MLP), Random Forest (RF), and Support Vector Machine (SVM). After a comparative evaluation, we reveal that the tree-based models (such as XGBoost) outperform others in detecting network failures.
- Third, we employ a model refinement method to sort the features according to their importance score. We confirm that with the most useful features we gain computational speed without obvious degradation of accuracy.
- Finally, we also find that latency and loss are confused according to the RF confusion matrix so that they are hard to predict inherently.

The rest of this paper is structured as follows. Section 2 describes the relevant research on network fault analysis. In Section 3, we present our extraction method from raw data and comparative analysis of ML-based faults classification. Section 4 shows the experimental results obtained using our method and the evaluation of comparison results. Finally, we provide a brief conclusion in Section 5.

2. RELATED WORK

There has been numerous literature concerning network faults detection. Most approaches rely on predefined rules, thresholds, and expert experiments. Mitchell et al. present a fault detection system for LAN networks [7]. The system is based on a set of rules defined on the data collected from the network monitoring process and the expertise of the network administrators. Although these methods can be realized automatically through scripts, they are usually low inefficiency and high in human labor costs [8, 9]. Therefore, approaches including Finite State Machine (FSM) and probabilistic approaches have also been researched [10–12]. Authors in [10] propose an FSM-based model and realize fault detection of partially observed data sequences. With the aid of FSM, [11] employ a probability approach to choose to synchronize conditions and optimally develop adaptive strategies. However, these traditional methods can hardly handle the frequent and dynamic changes in the network topology. On the other hand, the data volume obtained from managed entities is increasingly large in the era of 5G, and huge benefits can be leveraged from data-driven fault detection methods.
With the spread of the usage of Machine Learning (ML) technology in many fields, more and more studies have been proposed on network fault analysis using ML. Networking itself can also benefit from this promising technology. İF Kilinçer et al. propose a Bayesian method for monitoring and diagnosing faults that may occur in the Internet line [13]. They extract data via edge switching devices in a network campus area and use the Bayesian method to classify. It has been found that the accuracy of the classification results is over 90%. Ruiz et al. propose a probabilistic failure localization algorithm based on Bayesian Networks (BN) to localize and to identify the most probable cause of failures impacting a given service [14]. The authors use time-series monitoring data extracted from several light paths. When a service detects excessive errors, an algorithm uses the trained BN to localize and identify the most probable cause of the errors at the optical layer. Sauvanaud et al. propose anomaly detection and root cause localization for VNF using a supervised machine learning algorithm [15]. This approach detects Service Level Agreements'(SLA) violations based on monitoring data. It can pinpoint the root anomalous VNF VM causing SLA violations and achieve high recall, high precision, and low false alarm rate. Their experiments in [13, 14], and [15] show that the proposed algorithm can achieve high accuracy of fault classification. However, they do not compare their method with multiple ML algorithms or other training conditions.
Srinikethan et al. compare three ML algorithms that include SVM, MLP, and RF performance in terms of their link fault detection [16]. The authors develop a three-stage Machine Learning-based technique for Link Fault Identification and Localization (ML-LFIL) by analyzing the measurements captured from the usual traffic flows, including aggregate flow rate, end-to-end delay, and packet loss. Stadler et al. propose a method to predict service-level metrics from network device statistics using ML [17]. The authors adopt a work-regression tree and RF and investigate their prediction performance. They also compare the performance under several training conditions. References [16] and [17] compare the performance of multiple ML algorithms and seek to ascertain the effect of training conditions. However, their ML model's goal is fault detection and predictive service metrics, and it does not cover enough fault classification.

Qader et al. compare the performance of faults classification using K-Means, Fuzzy C Means (FCM), and Expectation-Maximization (EM) [18]. They use data sets obtained from a network with heavy and light traffic scenarios in the routers and servers and build a prototype to demonstrate the network traffic faults classification under given scenarios. The results show that FCM could achieve higher accuracy than K-Means and EM. However, it requires more time to process data. The authors focus on the data related to the physical interface only. Thus there is insufficient research on faults classification in an Network Function Virtualization(NFV) environment.
Recently, KDDI presented an ML comparison framework for network analysis [4]. It includes four functional blocks: data set generator, preprocessor, ML-based fault classifier, and evaluator. The data set generator can periodically generate failure data, which can be used in the ML-based fault classification task. They use three algorithms [Multilayer Perceptron (MLP), Random Forest (RF), Support Vector Machine (SVM)] to train and evaluate. The result shows that RF provides the highest performance even with a small amount of data, and SVM could improve its performance by increasing training data, feature reduction, or balance adjustment of normal/abnormal samples. However, the feature extraction method and training efficiency are not mentioned in their study. Training efficiency is an important metric for the evaluation of training models. Feature extraction is an essential step in achieving the excellent performance of an ML method. Especially for a large amount of network log data, efficiently extracting useful information from raw data can allow our model to perform better in a much shorter training time.

3. METHODOLOGY

This section introduces the data sets and shows how we extract features. Then we introduce several machine learning models used in this research.

3.1 Data preprocessing

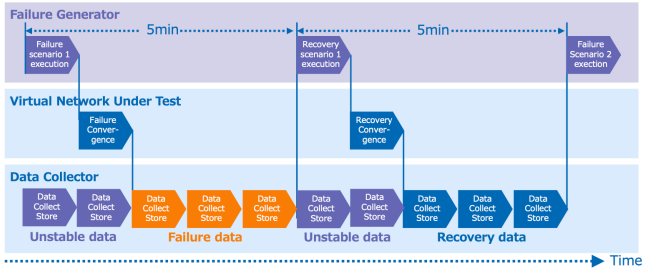


Fig. 1 – Data collection principles [4].

As shown in Fig. 1, The data sets used for this study are created in the NFV-based test environment simulated for a commercial IP core network. In this sense, synthetic data is similar to real data, resulting from the NFV-based test environment. The data sets consist of labels of normal/abnormal traffic, performance monitoring data sets such as traffic volume and CPU/Memory usage ratio, and route information such as Border Gateway Protocols (BGPs) static metrics and BGP route information.
The data collector from KDDI collects and stores data sets every minute from the network. Once a failure is intentionally caused or recovered, the network indicates a failure or normal status after a transition period, corresponding to failure data (orange arrows) and recovery data (blue arrows).
The time interval between a failure and a recovery is 5 minutes (Fig. 1). The data sets for training and evaluation provided by KDDI include four types, as in Table 1, which are *Label-Failure Management*, *Virtual-Infrastructure*, *Physical-Infrastructure* and *Network-Device*.
The training data set consists of 8 days of data, totaling approximately 120G JSON files. The evaluation data set consists of 7 days of data, totaling about 100G of JSON files.

Table 1 – Four types of data sets for learning and evaluation

Category	File Name	Data Format
Label	Label-Failure Management	json
Log Data	Virtual-Infrastructure	json
Log Data	Physical-Infrastructure	json
Log Data	Network-Device	json

3.1.1 *Data collection and merging method*

The JSON file's content is enormous, and most of the information is useless string description information. So we iterate through each object, looking for objects of numeric type. We extract these objects as features from log files (in JSON format) and merge them with labels into a CSV file based on time (Fig. 2).
We utilize paths like "key1/key1-1/key1-1-1..." as keys to extract features from physical, virtual, and network JSON log files for all log files. For BGP-related entries, we use the number of next-hops in each array and their prefixes as features.

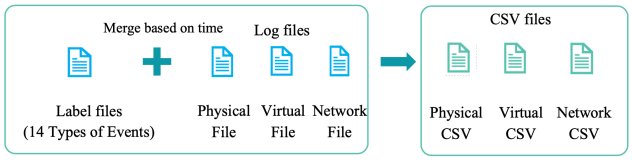


Fig. 2 – Data mergence principles

3.1.2 *Data differential method*

This subsection explains how our comparison framework preprocesses Performance Management (PM) data to put into Machine Learning (ML) models for training.
As shown in Fig. 3, each failure generation cycle is 5min. In the failure generation cycle, the last-minute data in the previous cycle is considered as regular data, and the last-minute data in the current cycle is considered as failure data. To highlight the differences between normal and

abnormal data entries to derive metrics that have changed since a failure, the differential data between the abnormal data and normal data is used as input features. After that, we can get three types of files in the data set, which are physical, virtual, and networks. To train a unified model for diverse network events, we merge all data sets into one CSV file for putting into ML algorithms. The process is shown in Fig. 4. Finally, the data set for training consists of 930 lines with 996 features, and for evaluation consists of 840 lines with 996 features.

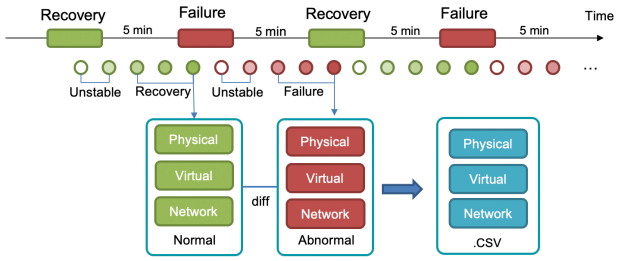


Fig. 3 – Data differential method

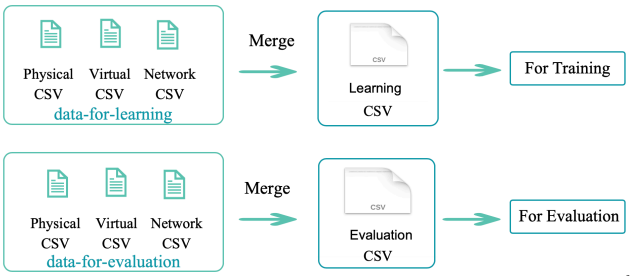


Fig. 4 – Data merging method

3.1.3 Label description

As shown in Table 2, we have five categories of labels for prediction, which are *Type1: node-down*, *Type3: interface-down*, *Type57: tap-loss (delay)*, *Type9: ixnetwork-bgp-injection*, and *Type11: ixnetwork-bgp-hijacking*.

3.2 Machine learning methods

In the related work in [4], Multiplelayer Perceptron (MLP), Support Vector Machine (SVM), and Random Forest (RF) are employed. In this study, as an extension of the related work, three other kinds of tree-based models, Decision Tree (DT), XGBoost (XGB), and LightGBM(LGBM) are also utilized.

3.2.1 Multiplelayer Perceptron (MLP)

MLP is a feed-forward artificial neural network that maps input data to the appropriate output. An MLP is a network of simple neurons called a perceptron which computes a single output from multiple real-valued inputs. A Perceptron forms a linear combination to its input weights and puts the output through a nonlinear activation function. The mathematical representation of MLP output is:

$$y = \varphi(\Sigma_{i=1}^{n} w_i x_i + b) = \varphi(W^T X + b)$$

where W is the vector of weights, X is the vector of inputs, b is the bias, and φ is the activation function.

As a neural network based model, the MLP algorithm is a general function approximation method that can fit complex functions and adequately approximate complex nonlinear relationships. It has a wide range of applications and has features of high accuracy. It is often used to solve classification problems. However, neural networks require manual determination of a large number of parameters, such as network topology, initial values of weights, and thresholds. Learning may be not sufficient when the parameter selection is inappropriate, and it is easy to fall into local extremes. Besides, since MLP is a black-box process, the learning process cannot be observed, and the output is difficult to interpret, which can affect the credibility and acceptability of the results.

3.2.2 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a linear machine working in a high dimensional feature space. SVM employs a nonlinear mapping to map the N-dimensional input vector x into a K-dimensional feature space (K>N). The problem that SVM tries to solve is to find an optimal hyperplane that correctly classifies data points by separating the points of two classes as much as possible. Both classification and regression tasks transform the learning task into a quadratic problem, but the way of creating SVM networks varies depending on the classification and regression tasks [19, 20]. Excellent introductions to SVM can be found in [21].

The main advantages of SVM are (1) able to work with high-dimensional data; (2) high generalization performance without the need to add prior knowledge, even when the dimension of the input space is very high.

Compared to MLP, SVM performs better in classification mode. And in regression mode, MLP has better generalization ability. In most cases, the observed performance difference is negligible [22].

3.2.3 Decision Tree (DT)

A decision tree is a supervised machine learning algorithm that can be applied to both classification and regression problems. Usually, it is top-down tree-like structures that explain the decision-making rules for prediction. The node from where the tree starts is known as a root node. The node where the tree ends is called the leaf node. Each internal node can have two or more branches. A node represents a particular characteristic, while a branch represents a range of values. These ranges of values act as partition points for the set of values of the given characteristic. In Fig. 5, we provide an illustration of a decision tree, which is also used in our experiments:

Table 2 – Five categories of labels for prediction

Scenario	TypeNo.	Type Name	Description
Network Element(NE) Failure	1	Node Down	Unplanned reboot of a NE
Interface Failure	3	Interface Down	Cause an interface down
Interface Failure	57	Packet Loss and Delay	Cause the packet loss and delay on an interface
Route Information Failure	9	BGP Injection	Inject the anomaly route from another SP
Route Information Failure	11	BGP Hijack	Hijack the own origin route by another SP

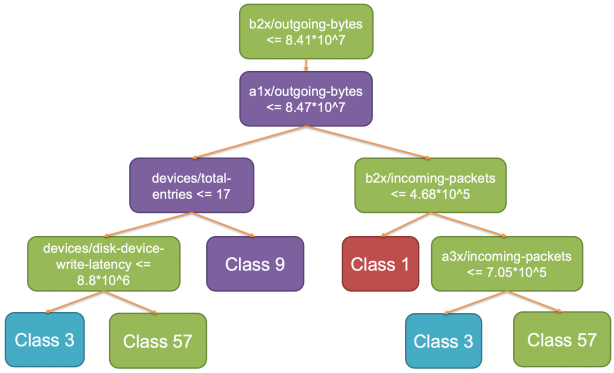


Fig. 5 – Visualization of a decision tree

A decision tree is a very intuitive data structure. Visualizing a decision tree can give valuable insights into the model's learning and how domain-relevant the learning is. In addition, the Decision Tree is fast and performs well on large data sets. It is also unaffected by outliers.
However, there are some shortcomings of a Decision Tree including:

- A decision tree might lead to overfitting when a tree is very deep. As the decision to split the nodes proceeds, each attribute is taken into consideration. It will try its best to fit all the training data accurately, which leads it to learn too much about the features of the training data and lose its ability of generalization.
- A decision tree is greedy and tends to find the local optimal solution instead of considering the global optimal solution. At each step, it uses some technique to find the best node. However, the local best node may not be the global best node.

3.2.4 Random Forest (RF)

To overcome the shortcomings of DT, random forest comes to the rescue. It was first proposed by Tin Kam Ho in 1995 [23]. Random forest consists of a large number of individual decision tree that operate as an ensemble. In the training process of RF, each individual tree in the RF spits out a class prediction and the class with the most votes becomes our model's prediction (Fig. 6). Random forest models are so effective because a large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

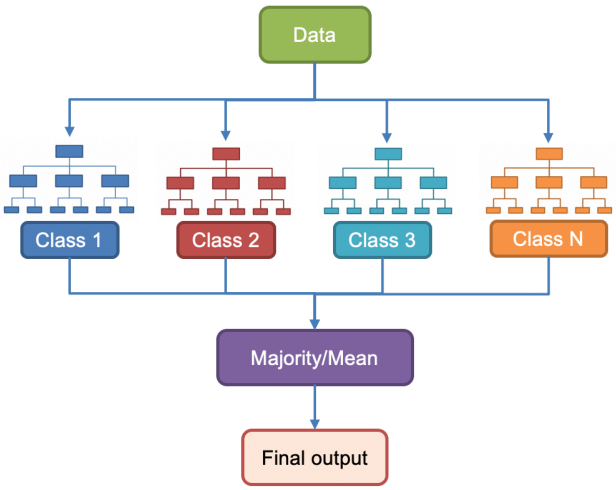


Fig. 6 – Visualization of an RF model making a prediction

Different with the decision tree model, which employs all the given data to determine the classification rules, a random forest classifier can randomly sample from the given data and build more than one decision trees. By considering the classification results of the several decision trees employing different subsets of the original data, the random forest method can effectively avoid overfitting. However, directly proportional to the model complexity, the training cost of random forest model would be higher than the decision tree model.

3.2.5 XGBoost (XGB)

XGBoost (XGB) is a successful machine learning model based on a gradient boosting algorithm proposed by Tianqi Chen [5]. Like random forests, gradient boosting is a set of decision trees. The two main differences are:

- Gradient boosting builds one tree at a time, while random forests builds each tree independently.
- Gradient boosting combines the results along the way, while random forests combines the results at the end of the process.

XGBoost stands for extreme gradient boosting. It is a specific implementation of the gradient boosting method that delivers more accurate approximations using the second-order derivative of the loss function, advanced regularization, and parallel computing.

The objective of XGBoost is to not only prevent overfitting but also optimize the computational resources. This is obtained by simplifying the objective functions that combine predictive and regularization terms and maintain an optimal computational speed.
The additive learning process in XGBoost is to fit the first learner to the whole space of input data and then to fit a second model to these residuals to tackle the drawbacks of a weak learner. This fitting process will be repeated a few times until the stopping criterion is met. The ultimate prediction of the model is obtained by the sum of the prediction of each learner. To learn the set of functions used in the model, XGBoost minimizes the following regularized objective.

$$Obj = \sum_{i=1}^{N} L(y_i, \hat{y}i) + \sum_{m=1}^{M} \Omega(f_m), f_m \in F$$

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda\|\omega\|^2$$

where L is the loss function, n is the number of observations used, Ω is the regularization term, ω is the vector of scores in the leaves, λ is the regularization parameter, and γ is the minimum loss needed to further partition the leaf node. Moreover, XGBoost can be extended to any user-defined loss function by defining a function that outputs the gradient and the Hessian (second-order gradient) and passing it through the "objective" hyper-parameter.
In addition, XGBoost implements several methods to increase the training speed of decision tree that are not directly related to the accuracy of the ensemble. In particular, XGBoost focuses on reducing the computational complexity to find the best split. This is the most time-consuming part of decision tree algorithms. Split-finding algorithms typically list all possible candidate splits and select the one with the highest gain. This requires a linear scan over each sorted attribute to find the best split for each node. To avoid repeatedly sorting the data in each node, XGBoost uses a specific compressed column-based structure in which the data is stored pre-sorted. In this way, each attribute needs to be sorted only once. This column-based storage structure allows finding the best split for each considered attribute in parallel. Instead of scanning all possible candidate splits, XGBoost implements a method based on percentiles of the data, testing only a subset of the candidate splits and calculating their gain using aggregated statistics. More detailed information and computational procedures of the XGBoost algorithm can be found in Tianqi Chen [5].

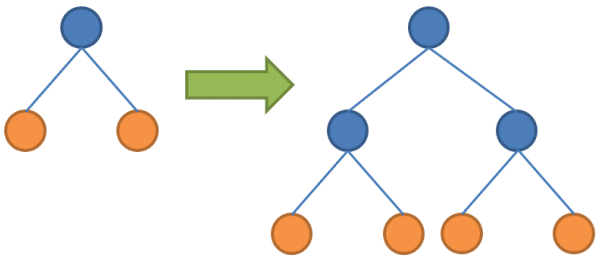


Fig. 7 – XGBoost level-wise tree growth

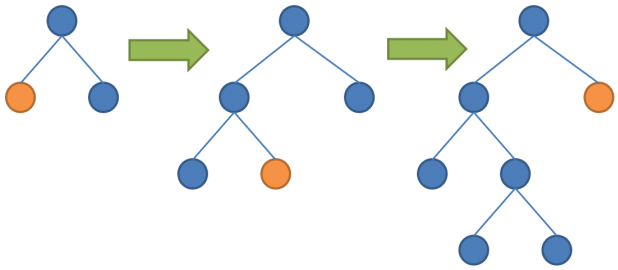


Fig. 8 – LightGBM level-wise tree growth

3.2.6 *LightGBM (LGBM)*

LightGBM is a gradient boosting framework that uses tree-based learning algorithms [6]. It proposed to solve the problems of Gradient Boosting Decision Tree(GBDT) in mass data. The main difference is that the decision tree in LightGBM are grown leaf-wise, as shown in Fig. 7 and Fig. 8, instead of the traditional level-wise that requires checking all of the previous leaves for each new leaf, which improves accuracy and prevents overfitting. Moreover, LightGBM uses a histogram to identify the optimal segmentation point. A histogram replaces the traditional pre-sorted, so in a sense, it sacrifices accuracy for speed. There are three aspects of differences between LightGBM and XGBoost.

- First is the computational complexity. Compared with XGBoost, LightGBM develops two kinds of methods to reduce the dimensions of input features so as to decrease the computational complexity. Based on the graph algorithm, LightGBM employs Exclusive Feature Bundling (EFB) to reduce the total number of input features. At the same time, LightGBM utilizes Gradient-based One-side Sampling (GOSS) to rank the samples according to the gradients. The proportion of features with large gradients increases by selecting the features with large gradients and combining some randomly selected features with smaller gradients.

- Second is the difference in strategy. LightGBM employs a leaf-wise strategy, while XGBoost employs a level-wise one. Resource wasting exists in XGBoost for there are indiscriminate nodes split into all layers even when the gain is minimal. On the other hand, LightGBM only splits leaf nodes with the greatest splitting gains. Such a greedy operation will also lead to overfitting and extremely large tree depth, so the tree depth in LightBGM should be constrained.

- Third is the scale of parallelization operation. Compared with XGBoost which focuses on the parallelism of features, LightGBM can parallelly deal with the features, data processing, and voting operations, which makes it be able to handle a larger data set.

3.2.7 Summary of machine learning methods

In this subsection, we give a brief summary of all the algorithms described above.

The MLP performs well when evaluating a probabilistic performance metric. However, it is not particularly better at handling high-dimensional data sets than other methods due to the large number of parameters that need to be tuned.

The main strengths of SVM are its effects on high-dimensional data and on data sets in which the number of features is greater than the number of observations. It takes less memory consumption due to the use of support vector and the use of various kernel functions, which are used in the decision function. However, SVM would overfit the model if the differences between the number of features and observations is too big.

One of the Decision Tree's major strengths is that it is easy to understand and to analyze. The disadvantage of the Decision Tree is that it is prone to overfitting and has low generalization performance.

Random Forest, XGBoost, and LightGBM all belong to ensemble methods. Ensemble methods combine the predicted results of multiple base estimators. So the results are improved as compared to some individual estimators. There are two main kinds of ensemble methods. The first one, such as Random Forest, includes techniques that consider results from many individual estimators and combine their results using the average. The second one, such as LightGBM and XGBoost, includes techniques that combine many weak estimators to get a decisive result of an ensemble.

4. EXPERIMENTATION AND EVALUATION

In this section, we use the differential data as input features and train the ML model with multiple ML algorithms. After training, we use evaluation data and different evaluation metrics to evaluate the model prediction capacity.

Note that there is no way to know in advance the best values for hyper-parameters, so ideally we need to try all possible values to know the optimal values. Doing this manually could take a considerable amount of time and resources and thus we use GridSearchCV to automate the tuning of hyper-parameters which improves the training efficiency.

4.1 Feature reduction

Using ensembles of decision tree methods like XGBoost may not perform well if the input features have much noise, as this can result in overfitting. In the feature vectors we use, due to the large number of features, the model training is not efficient and the training time becomes very long. Some features not only do not contribute to the modeling, but increase the complexity of the model. From the point of view of machine learning, the reason for feature reduction is to shorten the model computational time and to decrease the number of observations needed for a statistically accurate model. In terms of network management, reducing the feature set means that we can reduce the overhead of network monitoring. So trying to reduce the number of features becomes very important.

Generally, the RF and XGBoost have a built-in function that evaluates the importance of features. Some of the feature importance bar graph plots based on RF and XGBoost modeling are shown in Fig. 9. The features are sorted based on their importance.

In both RF and XGBoost, `activities/prefixes`, `sent/current-prefixes`, `prefixes/total-entries` and `as-path/total-entries` show significant importance compared to the other features. And these feature importance ranking results seem reasonable: 1) when the network goes down or the device goes down, the outgoing bytes definitely change; 2) the decline in the number of the activated links decreases the number of prefixes; 3) decrease of the prefixes changes the total number of the entries; 4) failures of the nodes absolutely affect the network-outgoing-packets.

However, there are large differences between RF and XGBoost feature importance ranking results. For example, `address-family/total-memory` has the highest rank in the result of feature ranking of the RF method, while it is not a key feature in the XGBoost method. Also, feature `network-outgoing-bytes` stands as an important role in XGBoost, while it is ranked much lower than many other features by the RF method. Some studies have reported that the feature importance ranking built-in function of RF is biased and unreliable [24]. Considering that, we choose to use the features that are ranked by XGBoost instead of ranking results by RF method.

Take the result of XGBoost ranking for consideration. If we take the peer router down as an instance, when the peer device is down, according to the default BGP configuration on the experiment routers, after 180 seconds, the link is down. Consequently, the number of next-hops is definitely changed, and also the total number of octets received in input packets from the specified address family includes those received in error. These will not only affect the incoming and outgoing bytes but also cause drops of the packets and reduction of current prefixes. Some differences in features importance ranking could be a result of features dependency. However, in general, the feature rankings obtained in this paper are reasonable and beneficial for future studies.

Dropping features with a low score (no contribution or low contribution to the model) will not influence the accuracy (Fig. 10 and Fig. 11) but benefit the model by reducing training time (Fig. 12). Fig. 10 shows the accuracy and precision using different numbers of features and Fig. 11 shows the accuracy and recall. As the number of input features changes, precision of the failures `Node Down`, `Interface Down`, `BGP Injection`, and `BGP Hijack` have maintained a relatively high accuracy rate, while `Packet Loss & Packet Delay` are relatively low. Recall of the

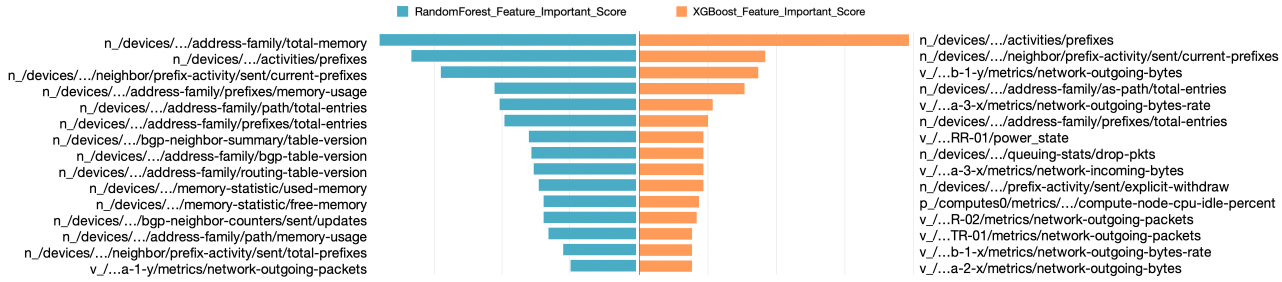


Fig. 9 – Feature score and importance ranking (Left: RF, Right: XGBoost)

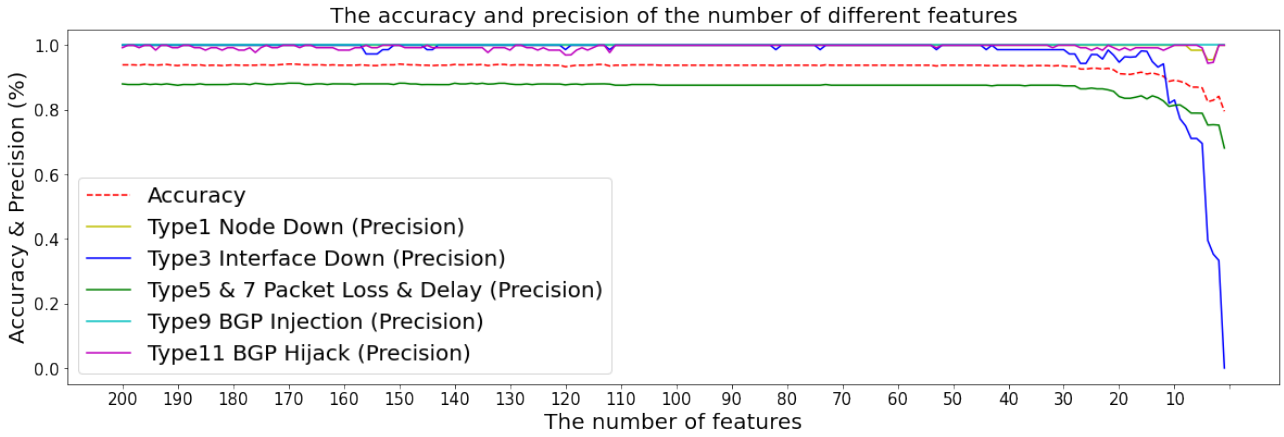


Fig. 10 – Accuracy and precision with different features

failures `Node Down`, `Interface Down`, `BGP Injection`, and `Packet Loss & Packet Delay` have maintained a relatively high accuracy rate, while `BGP Hijack` are relatively low.

The top 150 important features get the best performance which is 94.17% of average accuracy. Also, it can be seen that there are abrupt reductions on `Interface Down` and slight reductions on other failures after the number of features is reduced to 30. So we can use only 30 features to achieve a good performance, almost the same as the best performance, and for the following experimentation, we use these 30 features for training.

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

$$F - measure\ score = \frac{2 * recall * precision}{recall + precision} \quad (3)$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (4)$$

4.2 Evaluation

To measure ML-based classifiers' performance, we use the following evaluation metrics: *Precision*, *Recall*, *F-measure*, and *Accuracy*. These metrics are calculated by

Table 3 – Comparison of detection accuracy of different algorithms

No.	Method	Accuracy	Training Time(s)
1	XGBoost	0.9369	0.33
2	LightGBM	0.9333	2.81
3	Random Forest	0.9274	0.80
4	MLP	0.8131	1.52
5	Decision Tree	0.8095	0.38
6	SVM	0.7905	5.53

Table 4 – Test cases for different labels

TypeNo.	Type Name	Test Cases
1	Node Down	64
3	Interface Down	72
9	BGP Injection	156
11	BGP Hijack	180
57	Packet Loss and Delay	368

Eq (*(1)*), Eq (*(2)*), Eq (*(3)*), and Eq (*(4)*) assigned with the number of True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN).

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. High precision relates to the low false positive rate. The recall is the ratio of correctly predicted positive observations to all observations in the

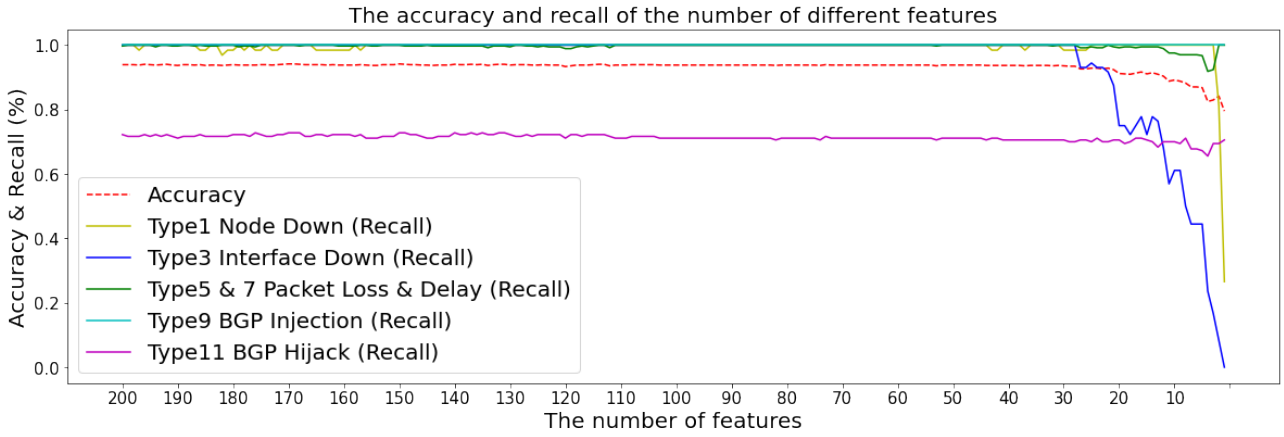


Fig. 11 – Accuracy and recall with different features

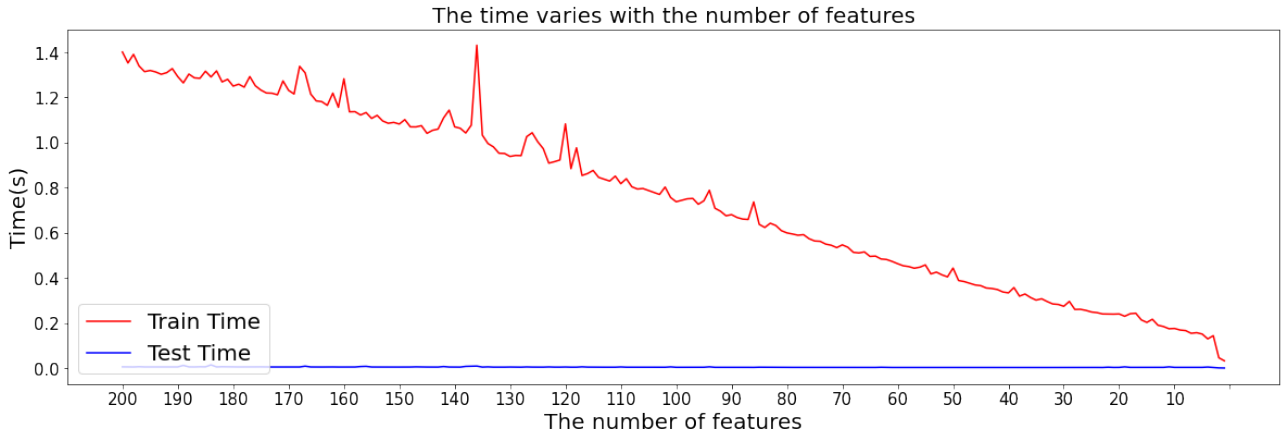


Fig. 12 – Training time with different features

Table 5 – Comparison of experimental results of different machine learning methods

Method	XGBoost			LightGBM		
Evaluation Criteria	Precision	Recall	F-measure	Precision	Recall	F-measure
1: node-down	1.00	**1.00**	1.00	1.00	**1.00**	1.00
3: interface-down	0.99	**1.00**	0.99	0.99	0.97	0.98
5, 7: tap-loss (delay)	0.88	**1.00**	0.93	0.87	**1.00**	0.93
9: ixnetwork-bgp-injection	1.00	**1.00**	1.00	1.00	**1.00**	1.00
11: ixnetwork-bgp-hijacking	1.00	0.71	0.83	1.00	0.70	0.82
Method	Random Forest			Decision Tree		
Evaluation Criteria	Precision	Recall	F-measure	Precision	Recall	F-measure
1: node-down	1.00	**1.00**	1.00	1.00	0.86	0.92
3: interface-down	0.97	**1.00**	0.99	0.82	0.89	0.85
5, 7: tap-loss (delay)	0.88	0.97	0.92	0.85	0.73	0.78
9: ixnetwork-bgp-injection	1.00	**1.00**	1.00	1.00	**1.00**	1.00
11: ixnetwork-bgp-hijacking	0.94	0.72	0.82	0.58	0.76	0.66
Method	SVM			MLP		
Evaluation Criteria	Precision	Recall	F-measure	Precision	Recall	F-measure
1: node-down	0.97	0.97	0.97	1.00	0.83	0.91
3: interface-down	0.92	0.65	0.76	0.92	0.61	0.73
5, 7: tap-loss (delay)	0.68	0.99	0.81	0.71	**1.00**	0.83
9: ixnetwork-bgp-injection	0.99	0.54	0.70	1.00	0.65	0.79
11: ixnetwork-bgp-hijacking	1.00	0.58	0.73	0.97	0.65	0.78

	1	3	9	11	57	SUM
1	64	0	0	0	0	64
3	0	72	0	0	0	72
9	0	0	156	0	0	156
11	0	0	0	128	52	180
57	0	0	0	0	368	368

(a) XGBoost

	1	3	9	11	57	SUM
1	64	0	0	0	0	64
3	0	70	0	0	2	72
9	0	0	156	0	0	156
11	0	1	0	126	53	180
57	0	0	0	0	368	368

(b) LightGBM

	1	3	9	11	57	SUM
1	64	0	0	0	0	64
3	0	72	0	0	0	72
9	0	0	156	0	0	156
11	0	0	0	130	50	180
57	0	2	0	9	357	368

(c) Random forest

	1	3	9	11	57	SUM
1	55	8	0	0	1	64
3	0	64	0	1	7	72
9	0	0	156	0	0	156
11	0	3	0	137	40	180
57	0	3	0	97	268	368

(d) Decision tree

	1	3	9	11	57	SUM
1	62	0	0	0	2	64
3	0	47	0	0	25	72
9	0	2	85	0	69	156
11	1	1	1	104	73	180
57	1	1	0	0	366	368

(e) SVM

	1	3	9	11	57	SUM
1	53	1	0	0	10	64
3	0	44	0	0	28	72
9	0	1	102	2	51	156
11	0	2	0	117	61	180
57	0	0	0	1	367	368

(f) MLP

Fig. 13 – Comparison of confusion matrix of different models

actual class. But the difference from the precision rate is that the recall rate focuses more on the proportion of samples which are True Positive (TP) that are successfully predicted. The F1 score is the weighted average of precision and recall. Therefore, this score takes both false positives and false negatives into account. The core idea of F1 score is to improve precision and recall as much as possible while also hoping that the differences between the two is as small as possible.

Table 3 shows the total accuracy of different models. XGBoost shows the best performance of accuracy with the least training time, and then LightGBM and Random Forest. Decision Tree, MLP, and SVM relatively show lower accuracy. Results prove the stable and outstanding performance of tree-structured models, as well as the lifting performance of the ensemble learning methods. We believe that the reason for the low accuracy of the Decision Tree is the overfitting of the training model. As for MLP, the final classification performance strongly depends on whether the optimal solution can be found. However, the back propagation algorithm in MLP tends to converge to the local optimum, so the classification accuracy cannot be ensured. SVM yields the longest training time but the lowest prediction accuracy. The reason may be concluded into the inherent computational complexity of SVM and the influence of the unrelated features in the data.

Table 5 shows the precision, recall, and F-measure results of each machine learning method on different failures. It can be seen that `Node Down` failure can be well predicted in every model with high precision and recall.

Fig. 13 shows the diagonal of the confusion matrix represents the number of samples that are correctly classified, while others are wrongly classified.

5. CONCLUSION

In this paper, we employ a highly practical and reliable approach to solve the problem about how to automatically and rapidly detect network and device failures. First, we define a staged method including feature extraction from unstructured network logs, a differential approach to highlight the differences between normal and abnormal states and several ML models to realize failure classification. Then, we apply the staged method to six popular machine learning algorithms, including Decision Tree (DT), XGBoost, LightGBM, Multilayer Perceptron (MLP), Random Forest (RF), and Support Vector Machine (SVM). After a comparative evaluation, we reveal that the tree-based models (such as XGBoost) outperform others in detecting network failures. Third, we employ a model refinement method to sort the features according to their importance score. We confirm that with the most useful features can gain computational speed without obvious degradation of accuracy. Finally, we also find that latency and loss are confused according to the RF confusion matrix so that they are hard to predict inherently.

Overall, our results achieve a reliable method for detecting network failures: almost 100% accuracy when detecting network and device failures, 86% accuracy when detecting packet loss and delay, and a total average of 94% accuracy. At the same time, the proposed feature extraction and refinement method can reduce computation without degrading the performance.

From the evaluation results of different types of models, we know that different methods are capable of learning some parts of the problem, but not the whole space of the problem. This may constitute the potential objects for future studies. We can build multiple different learners and we use them to build an intermediate prediction,

one prediction for each learned model. Then we add a new model which learns from the intermediate predictions of the same target.

6. FUTURE WORK

As for future work, we are aiming to compare different machine learning methods, for example, DT, MLP, RF, XGBoost, etc. with different data sets, not only KDDI's, to explore clearer boundaries of the most suitable range between each method so that we could take the optimal solution for diverse failures. In addition, analyzing the topology and applying graphical convolution networks at the stage of determining the importance of features might prove a promising result.

REFERENCES

[1] "Coronavirus: Five firms booming despite the lockdown," IV , {https://www.bbc.com/news/business-52383193}.

[2] "How zoom became so popular during social distancing," IV , {https://www.cnbc.com/2020/04/03/how-zoom-rose-to-the-top-during-the-coronavirus-pandemic.html}.

[3] "Google lost $1.7m in ad revenue during youtube outage, expert says," IV , {https://finance.yahoo.com/news/google-lost-1-7m-ad-144303640.html}.

[4] Junichi Kawasaki, Genichi Mouri, and Yusuke Suzuki, "Comparative analysis of network fault classification using machine learning," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, pp. 1–6.

[5] Tianqi Chen and Carlos Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[6] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, pp. 3146–3154, 2017.

[7] Mitchell Vásquez-Bermúdez, Jorge Hidalgo, María del Pilar Avilés-Vera, José Sánchez-Cercado, and Christian Roberto Antón-Cedeño, "Analysis of a network fault detection system to support decision making," in *International Conference on Technologies and Innovation*. Springer, 2017, pp. 72–83.

[8] Todd E Marques, "A symptom-driven expert system for isolating and correcting network faults," *IEEE Communications Magazine*, vol. 26, no. 3, pp. 6–13, 1988.

[9] Irene Katzela and Mischa Schwartz, "Schemes for fault identification in communication networks," *IEEE/ACM Transactions on networking*, vol. 3, no. 6, pp. 753–764, 1995.

[10] Anastasios T Bouloutas, George W Hart, and Mischa Schwartz, "Fault identification using a finite state machine model with unreliable partially observed data sequences," *IEEE Transactions on Communications*, vol. 41, no. 7, pp. 1074–1083, 1993.

[11] Cynthia S Hood and Chuanyi Ji, "Proactive network-fault detection [telecommunications]," *IEEE Transactions on reliability*, vol. 46, no. 3, pp. 333–341, 1997.

[12] Eleftheria Athanasopoulou and Christoforos N Hadjicostis, "Probabilistic approaches to fault detection in networked discrete event systems," *IEEE transactions on neural networks*, vol. 16, no. 5, pp. 1042–1052, 2005.

[13] İlhan Firat Kilinçer, Fatih Ertam, Orhan Yaman, and Ayhan Akbal, "Automatic fault detection with bayes method in university campus network," in *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*. IEEE, 2017, pp. 1–4.

[14] Marc Ruiz, Francesco Fresi, Alba P Vela, Gianluca Meloni, Nicola Sambo, Filippo Cugini, Luca Poti, Luis Velasco, and Piero Castoldi, "Service-triggered failure identification/localization through monitoring of multiple parameters," in *ECOC 2016; 42nd European Conference on Optical Communication*. VDE, 2016, pp. 1–3.

[15] Carla Sauvanaud, Kahina Lazri, Mohamed Kaâniche, and Karama Kanoun, "Anomaly detection and root cause localization in virtual network functions," in *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2016, pp. 196–206.

[16] Srinikethan Madapuzi Srinivasan, Tram Truong-Huu, and Mohan Gurusamy, "Machine learning-based link fault identification and localization in complex networks," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6556–6566, 2019.

[17] Rolf Stadler, Rafael Pasquini, and Viktoria Fodor, "Learning from network device statistics," *Journal of Network and Systems Management*, vol. 25, no. 4, pp. 672–698, 2017.

[18] Karwan Qader, Mo Adda, and Mouhammd Al-Kasassbeh, "Comparative analysis of clustering techniques in network traffic faults classification," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 5, no. 4, pp. 6551–6563, 2017.

[19] Vladimir N Vapnik, "An overview of statistical learning theory," *IEEE transactions on neural networks*, vol. 10, no. 5, pp. 988–999, 1999.

[20] Alex J Smola and Bernhard Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

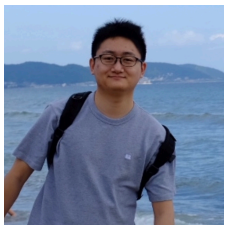
[21] Nello Cristianini, John Shawe-Taylor, et al., *An introduction to support vector machines and other kernel-based learning methods*, Cambridge university press, 2000.

[22] Stanislaw Osowski, Krzysztof Siwek, and Tomasz Markiewicz, "Mlp and svm networks-a comparative study," in *Proceedings of the 6th Nordic Signal Processing Symposium, 2004. NORSIG 2004.* IEEE, 2004, pp. 37–40.

[23] Tin Kam Ho et al., "Proceedings of 3rd international conference on document analysis and recognition," in *IEEE*, 1995, pp. 278–282.

[24] Carolin Strobl, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn, "Bias in random forest variable importance measures: Illustrations, sources and a solution," *BMC bioinformatics*, vol. 8, no. 1, pp. 1–21, 2007.

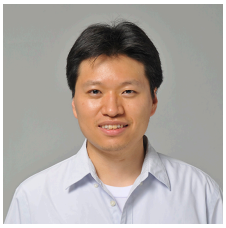
AUTHORS



Xia Fei is currently a master student at the Graduate School of Interdisciplinary Information Studies (GSII), the University of Tokyo. His research work focuses on practical approaches to the AI system. He is interested in doing research in network fault detection, data analysis, and machine learning.

Aerman Tuerxun received his B.Eng of Information Security from the University of Science and Technology of China and is currently pursuing his master degree at the Graduate School of Interdisciplinary Information Studies (GSII), the University of Tokyo. His research interests include local 5G, network slicing, network intelligence etc.

Lu Jiaxing is a Ph.D student at the Graduate School of Interdisciplinary Information Studies (GSII), the University of Tokyo. His research interests include federated learning and edge computing.



Ping Du received a Ph.D. from the Graduate University for Advanced Studies in Japan in 2007. From 2008, he worked for the National Institute of Information and Communication Technologies (NICT) of Japan. Now, he works for the University of Tokyo as a project associate professor. His research interests include optical networks, network security, network virtualization, mobile networks and machine learning etc.

Akihiro Nakao received his BS in physics and ME in information engineering from the University of Tokyo. He worked at IBM Yamato Laboratory, Tokyo Research Laboratory, and IBM Texas Austin. He received his MS and PhD in computer science from Princeton University. Since 2005, he has been an associate professor and then a professor in applied computer science at the Interfaculty Initiative in Information (III) Studies, Graduate School of Interdisciplinary Information Studies, the University of Tokyo. Now, he is a professor in Faculty of Engineering, University of Tokyo. He has been appointed Chairman of the 5G Mobile Network Promotion Forum (5GMF) Network Architecture Committee by the Japanese government.