INTERNATIONAL JOURNAL OF Telecommunications
AIR DEFENSE COLLEGE

Traffic Classification in Software Defined Networks based on Machine Learning Algorithms

Sherif Mahgoub [1], Mohamed Ashour [2] ,Mohamed Yakout [3] and Eman AbdElhalim[4]

Citation: Mahgoub, S.; Ashour, M.; Yakout, M.; AbdlElhalim, E. *Inter. Jour. of Telecommunications, IJT'2023, Vol. 04, Issue 01, pp. 01-19, 2024.*

Editor-in-Chief: Youssef Fayed.

Received: 20/11/2023.

Accepted: 08/02/2024.
Published: 08/02/2024.

Publisher's Note: The International Journal of Telecommunications, IJT, stays neutral regard-ing jurisdictional claims in published maps and institutional affiliations.

[1] Electronics and Communications Engineering Department, Faculty of Engineering, Mansoura University; sherifmahgoub2019@gmail.com. [2] Associate Professor at Faculty of Engineering, Mansoura University; mohmoh@mans.edu.eg. [3] Associate Professor at Faculty of Engineering, Mansoura University; myakout@mans.edu.eg. [4] Assistant Professor at Faculty of Engineering, Mansoura University; eman_haleim@mans.edu.eg.

Abstract: A crucial area of research is traffic classification, particularly in light of the advancements in machine learning in software-defined networking. Software-defined networks, which divide the control and data planes, can be automated and controlled by machine learning. Because traditional procedures could not keep up with the expanding use of encryption, the use of techniques for this purpose has increased. In this study, 15 features (the quantity of packets communicated, the average transmission time, and the number of instantly transmitted packets) were used to build traffic flows on the SDN for several protocols, including WWW, DNS, FTP, ICMP, P2P, and VOIP. A real-time dataset was produced by gathering data based on the features that were generated over the SDN controller in the physical network. We use the dataset to test and train a variety of machine learning models, including Random Forest, K Nearest Neighbor, Support Vector Machine, Logistic Regression, Decision Tree, and Naive Bayes. With a 99.8% accuracy rate, Decision Tree emerged as the most successful model for the traffic classification challenge. In order to provide the best classification performance with the lowest cost flow features for traffic classification in SDN, this approach has been identified as machine learning.

Keywords: Software Defined Network SDN ; Traffic Classification TC ; machine learning ML ; Decision Tree DT

1. Introduction

Network traffic classification has turned into a crucial problem for computer science because of the growth and diversification of internet data traffic. Utilizing traffic analysis, classification techniques are used to enhance network service quality, make better use of network resources., identify network assaults and abnormalities[1]. Networks are becoming more complex and dynamic, which has compelled network operators to devise efficient network management methods. Due to the massive amount of generated traffic, traffic classification has become a difficult task in order to distinguish between a variety of applications [2]. The network traffic classification is the foundation of network management, which can manage the corresponding network traffic differently, provide the basis for the subsequent network protocol design, provide the strategies for network attack detection and flow cleaning in network security [3]. In the network, predetermined rules are applied to categorize packets. Information about ports and network packet headers are used as classification criteria. This method works incredibly well for popular apps, but it doesn't work well for apps that use changeable port numbers [4]. Software Defined Networking (SDN) decouples the network control logic from the data plane described in, which most likely breaks the vertical framework of the OSI model [5]. SDN is seen as a potential network paradigm because of its claims to significantly improve resource consumption, simplify network operation and preservation, and simplify network maintenance. But the level of sophistication in the

existing SDN control granularities of flows is insufficient. Unlike traditional network architecture, the SDN isolates the control plane from the data plane, allowing a centralized controller to directly program the network [6]. A high-performance server receives the control plane, and network management is handled by central controller software. Pack transmission is only the responsibility of the data plane on routers or switches that are OpenFlow compliant. Of all the protocols that can be used in SDN architecture, the OpenFlow Protocol (OFP) is the most efficient. Since the contents of the package are not used to determine categorization, they can be classified as encrypted data. The data required for SDN traffic classification is gathered by the ML techniques through the OFP, and the classification can be completed with a low computational cost [7]. Despite the many benefits of SDN, research is still needed to improve routing optimization, traffic classification, quality of service (QoS) maintenance, controller placement, minimize flow table entries, load balancing, identify failure causes, and maintain reliability while scaling [8]. Machine learning-based techniques leverage statistical features extracted from data for traffic classification. Consequently, certain features may impede the classification process in classification domains, and an increase in data dimensionality may lead to a decrease in the prediction capability of an algorithm as well as an increase in computational expenses (such as processing and storage) [9]. Therefore, we go over the most typical and well-liked techniques that have historically been employed in the literature for TC analysis.

However, the existing distributed network architectures are unsuitable for collecting and processing a large-scale data. The advent of the software-defined networking (SDN) enables the more efficient classification of network traffic. With the separation of the control plane and data plane, the control system is centralized and can be employed to significantly improve the collection and processing of large-scale data. In terms of network classification, the statistical flow information can be extracted more easily due to SDN controllers. In this paper, we propose a network traffic classification framework using the SDN architecture. We also apply six ensemble algorithms and analyze their classification performance in terms of the accuracy, precision, recall, F 1-score, training time, and classification time. The data collected in each experiment were considered as training data, and test data were randomly collected to test the trained modules. Moreover, we showed how the selection features technique could affect the final results. The effects of feature selection and model adjustment on the performance of the classifier are examined. This study demonstrates that ML can attain high TC accuracy, making it appropriate for a range of SDN applications. The remainder of the document is structured as follows: The related work is provided in Section 2. The developed framework's content, methodology, and processing steps are presented in Section 3. Furthermore, Section 4 provides the experimental data demonstrating the viability of the suggested pipeline. Section 5 concludes with the work that will be done in the future.

2. Related Work

This section displays related works which utilize machine and deep learning to classify SDN traffic. Various studies have been conducted in recent years by researchers to explain the network TC problem from various perspectives. Su et al. [10] presented a solution to the" fine-grained traffic classification problem" that takes into account how to gather data and understand how ML algorithms work. Their analysis showed that decision tree (DT) and random forest (RF) had scored an accuracy of 99%. Samaan et al. [11] provided a novel video traffic classification method that makes use of an innovative FSVM strategy that allows the weight coefficient C to be adjusted adaptively. Their suggested method had an accuracy rate of 98% on average. Awad et al. [12] provided various ML algorithms for detecting and classifying conflict flows in SDNs. Their algorithms included DT, SVM, the range of flows selected had been between 100000 flows which achieved an accuracy of 98%. Van et al. [13] provided an SDN-IoT traffic classification model. On the given dataset, they ran three machine learning algorithms: RF, K-Nearest Neighbors, and DT. The best performing algorithm was RF classifier which achieved an accuracy of 99% with six features. Aouedi et al. [14] presented data exploration and analysis to pick the most pertinent features for categorizing network traffic. From 87 features in their dataset, they developed a method to determine the top 15 by employing ML classifiers such (DT, RF,

AdaBoost, CatBoost, Light-GBM, and XGBoost). Their method achieved an accuracy of 89.09%. Machoke et al. [15] presented a variety of machine learning (ML) algorithms for locating and classifying conflict flows in the SDN model. Depending on the flow regulations priority, action, protocol, and IP source address, the type of conflict was discovered and classified. Their proposed algorithms achieve accuracy of 90%. Karmous et al. [16] presented TC methods using unique ML models on SDN architecture by applying five distinct ML models (kNN, SVM, DT, and NB). The highest accuracy between them was 99%. Khairi et al. [17] suggested integrating ML technologies with SDN architecture. Three distinct models were applied, NB, SVM, and nearest centroid were used. In all three models, the TC task accuracy is more than 99%. Gnanamonickam et al. [18] demonstrated an SDN-based traffic classification framework. For their investigation, 500 samples were taken from each of the following apps: Dropbox, BitTorrent, Facebook, Linkedin, Skype, Vimeo, YouTube, Vimeo, Web Browsing (HTTP), and Facebook. These traffic samples were classified using machine learning techniques such as random forests (RF), stochastic gradient boosting (SGB), and extreme gradient boosting (EGB). Their results were 95%. Dang et al. [19] proposed an SDN architecture that might use Deep Learning(DL) algorithms to classify traffic. DL methods, such as the Multi-Layer Long Short Term Memory (LSTM) model and a CNN and single-layer LSTM model combination, were employed in their study to classify datasets comprising BitTorrent, HTTP, Secure Shell (SSH), RDP, and Skype traffic. Their results were 96%. An unsupervised profiling approach was utilized in Mondal et al. [20] to investigate flow characteristics and connectivity patterns in a limited amount of time. To solve the TC problem, which they formulated as a graph co-clustering problem with topology and edge attributes, the authors used a hybrid flow clustering (HFC) model. The disadvantage of unsupervised learning is that it requires more time for the user to evaluate the results. After the classification, the user has to spend time interpreting and labeling the classifications. Karn et al. [21] created a deep learning model in the SDN controller using multilayer perceptron (MLP), convolutional neural network (CNN), and stacked auto-encoder (SAE). Seven applications could be classified using the model. The average accuracies for the three models were nearly identical, at 98%. El-serwy et al. [22] used ML to create a model for network traffic classification. The traffic was collected from an SDN/NFV environment that included an ONOS controller and a simple mininet topology with two hosts and one OpenVswitch (OVS). Although the results of different ML algorithms for two different network scenarios were compared, no pre-processing steps were applied to the data, and the class imbalance problem had a significant impact on the results. The TC task is modeled using a multitask learning problem in [23]. Predicted values include bandwidth demand and flow duration in addition to the classification class. They contrasted a sizable amount of datasets with unlabeled traffic classes to a sizable amount with labeled bandwidth and flow length datasets. This strategy aimed to profit from the expense of labeling data for supervised learning. This research establishes a ML-based approach for classifying traffic flows in SDN. The presented system provides network operators with a solution for classifying network traffic in order to improve network efficiency and service quality. In this study, the number of features was chosen in order to prevent complexity and demanding computations in the network application while maintaining compatibility with the implementation (SDN controller). Instead of using a predetermined number of network traffic classes, an optimal number of network traffic classes was once determined using a supervised learning algorithm, making this method a more tailored network TC solution for network operators.

3. Proposed Methodology

There were two portions to this suggested solution. The ML algorithm was constructed in one half, and the network experiment was set up in the other, with the goal of illuminating the concept through the execution of the created ML model on an SDN platform. A comparable dataset was chosen, prepared, and made suitable for machine learning models in the first stage. Once the dataset has been collected and labeled using supervised machine learning, it must be used for training various classification models. In the second phase, an SDN architecture had to be set up, and a network application with a trained machine learning model for real-time classification had to be developed and implemented.

3.1 Description of Datasets

The dataset 'SDN-traffic" used in this paper was from kaggle [24]. It consists of 65 features, 3577296 instances, 78 classes (DNS, FTP, ICMP, P2P, VOIP, WWW and so on) and is stored in a CSV file. This dataset was chosen because it is a real dataset with a sufficient amount of diversity and richness to be effective in identifying different traffic behaviors. However, to make computation easier. As a result, our sub-dataset consists of 6 applications and 404,528 instances **Table 1**. Then, in order to use some traffic as unlabeled data, the target labeled from it was deleted. In this situation, The target label was detected from it. In this situation, 121,342 instances were employed (30%) of unlabeled data and 283,186 instances (70%) of labeled data.

Table 1. Class distribution of applications in dataset.

Application	Number of Flow
WWW	2441
P2P	710
ICMP	409
VOIP	256
FTP	217
DNS	184

3.2 Material and Methods

Figure 1 illustrates the methodology employed in this work, which consists of two primary processing steps:

- data preprocessing.
- traffic classification.

In this case, a variety of feature selection techniques have been applied, as indicated by the category. In the last step, accuracy, training, and classification time are used to evaluate the classification's performance.

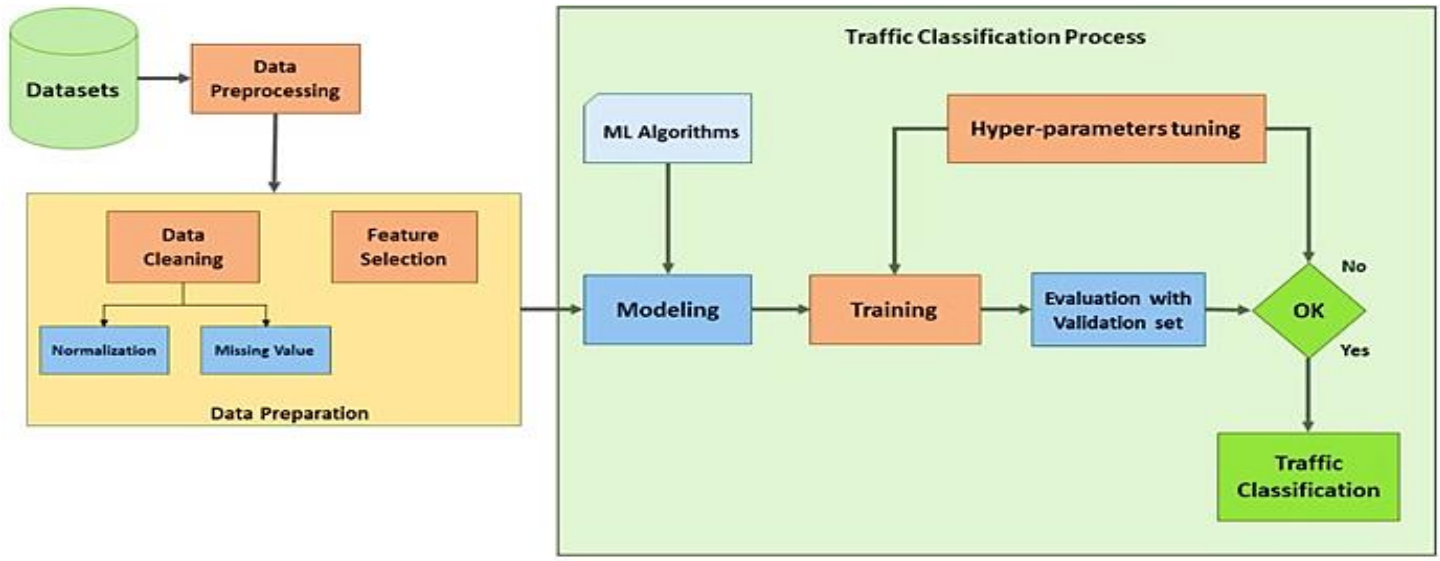

Figure 1: Structure of the experimental methodology

i) Data preprocessing is a data mining approach that modifies the data to make it suitable for classification. It's an initial phase that can be completed with a range of approaches, such as data cleaning and feature selection.

1. Data cleaning: While certain ML models can only handle numerical values, it is imperative to convert or reassign numerical values when the dataset comprises a variety of features of different sorts. In this effort, the IP address have been transformed and time stamp provisional data into numerical numbers.

Additionally, Because the dataset has several features with values on different scales, it must be scaled. You can use Min-Max normalization to control feature scaling. Using this strategy, all features in our dataset are scaled between 0 and 1, with each feature's lowest value set to 0 and its maximum value set to1. Since the packet loss rate feature includes some missing values, as the analysis reveals, NaN values were substituted for the feature's median values in order to produce a justifiable distribution of all the features.

2. Feature Selection: One of the contributions of this work is determining the ideal number of features (F) to produce the best classification performance in an actual dataset. In order to choose the best feature, set for the offline run, features selection approaches have been used. One of the most important steps in machine learning (ML) is feature selection. These algorithms pick a subset of features that are relevant to the goal notion. Feature selection offers numerous advantages, including the ability to

- improve learning algorithms' performance in terms of learning rate and generalization capacity by resolving the dimensionality issue.
- reduce storage requirements. In reality, feature selection is a difficult process because it must increase learning capacity while decreasing the number of features, in addition to the initial dataset's large size.

ii) Traffic Classification: For network management and security, the increasing diversity and complexity of network traffic provide significant challenges. The process of recognizing and classifying network traffic flows is known as traffic classification, and it is essential for detecting abnormal activity, understanding network behavior, and optimizing resource allocation. This paper presents a comprehensive analysis of TC techniques and their applications in enhancing network management in **Figure 2**.

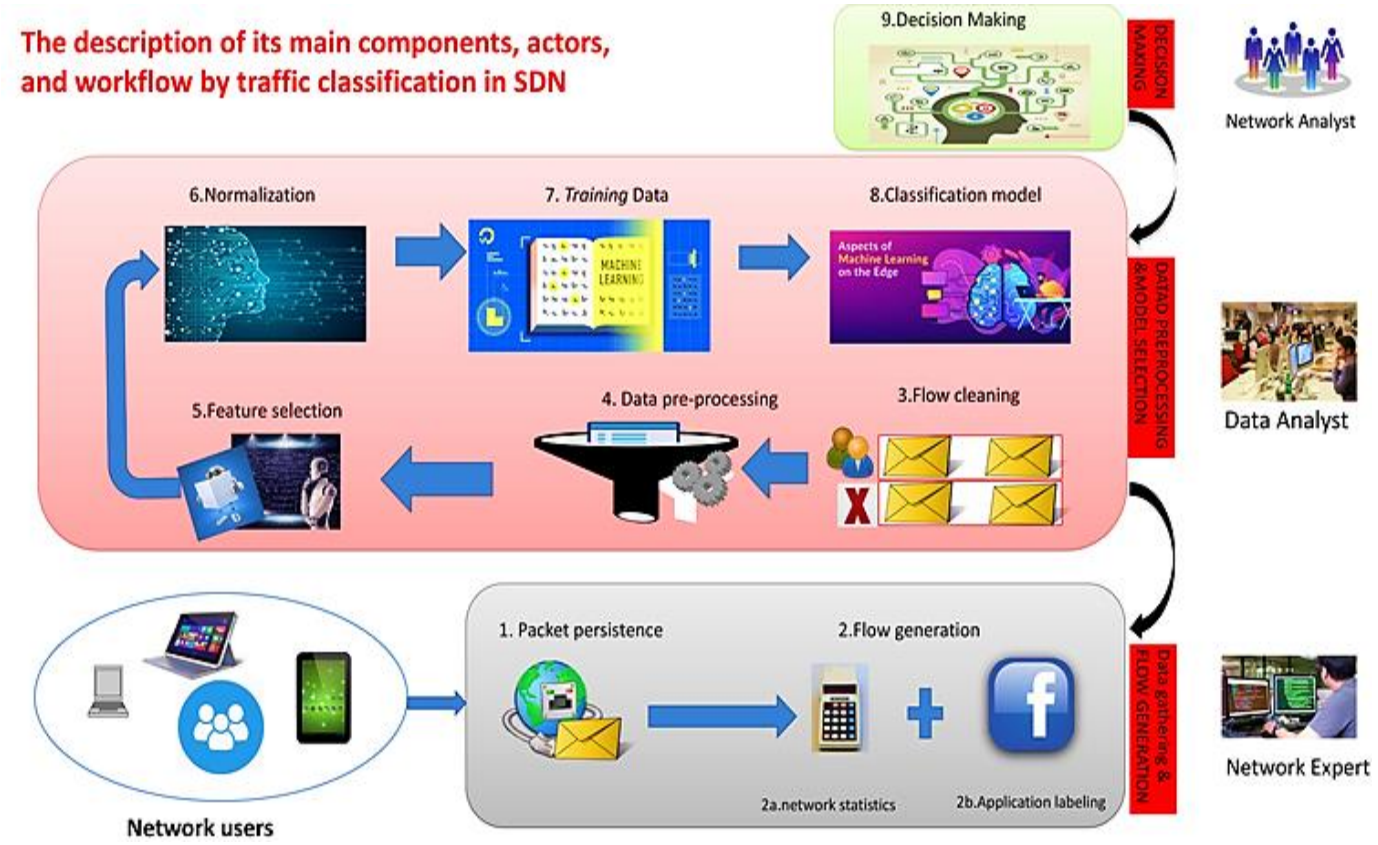

Figure 2: Traffic Classification

The models that were tested are briefly explained here:

- The supervised learning algorithm Support Vector Machine (SVM) uses labeled data to train the model. Often called hyper planes, decision boundaries between labeled data are established by the SVM model. Places that are near these hyper-planes are considered extreme points. In order to maximize these decision limitations, the algorithm will set margins between hyper-planes. Several different kernels are used to optimize these decision borders. The most typically used kernels are sigmoid, polynomial, RBF, and linear. One dimensional or multidimensional real-world data is available. Sometimes, it is also possible to separate these data sets linearly. The linear kernel may work with datasets that are linearly separable. [25,26].
- Decision Tree is Another supervised learning method based on knowledge gleaned from measuring the entropy of the dataset. Every condition and choice in the dataset is displayed graphically. To identify the root node, the dataset's entropy with the greatest information gain will be employed. With the division of branches in this way, the tree will finally be finished. Every internal node serves as an attribute test, with the outcomes displayed in the branches. The leaf represents a class label. The decision tree can include both numerical and categorical data for problems with categorization. Furthermore, nonlinear correlations among features are endorsed [25,26].
- Random Forest is a supervised learning technique that can handle both classification and regression issues. This is a combination of different decision tree methods, and the more trees included, the more accurate the model is. It functions similarly to a decision tree that is based on information gained. Every decision tree in classification will categorize the same problem, and the final conclusion will be determined by taking into account the majority of the out- comes. This model's ability to handle big datasets and handle missing values is by far its most significant advantage [25,26].
- Kth Nearest Neighbor (KNN) is an instance-based supervised learning method. The parameter k of the KNN model indicates how many neighbors need to be considered in order to categorize. After looking at the labels of those neighbors, the model will select the most common label. K should have an odd value in order to prevent inferences. With a large training set, this robust model performs well and can deal with noisy data. But it has trouble with multidimensional datasets, which could lead to a decline in efficacy, accuracy, etc [25,26].
- Na¨ıve Bayes (NB) is a probabilistic ML classifier based on the Bayes theorem. The algorithm makes the assumption that each attribute is impartial and contributes equally to the classification prediction. It is common practice to utilize the Naive Bayes method for classification problems. This approach could result in a passably excellent performance for this work [25,26].
- Logistic Regression (LR) is a simple supervised ML classifier, logistic regression, maps predictions to event probabilities using a cost function that is a sigmoid function. Based on the value chosen for the threshold, the out- put of the function, which has a range of 0 to 1, is used to assign the observations to discrete classes [25,26].

These six supervised learning models were taken for the training and prediction process to determine the model that best generates predictions

4. Experimental Results

The experimental procedure and assessment of our suggested solution were provided in this section.

4.1 Performance and Evaluation Metrics

Accuracy, F1 score, Recall, Precision, and Training time were the used performance metrics [27]. The calculation for accuracy is.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

The weighted average of correctly classified outcomes for the two outcomes is used to determine accuracy for multi-output classification tasks like ours. The F1 score of a test indicates how accurate it is. The results of the test are combined for precision and recall. A real positive or negative result is one for which the model accurately identified the positive or negative category, respectively. On the other hand, when the model predicts the positive or negative class in- accurately, it results in a false positive or negative. To take into consideration label imbalance. the weighted F1 score was employed. Additionally, The training time is monitored. The training time is the length of time needed to run an algorithm on a certain fold of the dataset

$$Precision = \frac{TP}{TP+FP} \tag{4}$$

$$Recall = \frac{TP}{TP+FN} \tag{5}$$

$$F1 - score = \frac{2*Precision*Recall}{Precision+Recall} \tag{6}$$

here; ***TP, TN*** is the number of correctly classes cases, and ***FN, FP*** is the number of incorrectly classes. Generally, accuracy is a frequently used parameter to assess the performance and comparability of the model. However, because it is less sensitive to unbalanced data and enhances the performance of some classes, relying solely on it may be deceptive. Therefore, F1-score, Precision, and Recall measures were applied. Ultimately, all of the networks were trained using the GPU that Kaggle provided. TensorFlow was used to generate the ML models. This article uses scikit-learn library in python as the implementation language to performs model training. Jupyter Notebooks serve as the software environment. All tests are performed on a Dell OptiPlex 7020 with an Intel R Core i5 3.5GHz processor and 8GB of RAM. The OS is Windows 10.

4.2 Results and Discussion

To solve classification problems, various classification algorithms are used. Each classification algorithm employs a distinct mathematical model. As a result, the obtained results vary. Only by experimenting with various classification models can it be determined which model is more successful. This study put the most popular classification algorithms to the test and compared their success rates. Using the Python module Scikit-Learn, the performance of the classification methods was evaluated. This library supports multiple ML models. The KNN, RF, DT, NB, LR, and SVM classification algorithms that are commonly employed in classification problems were the main focus of this study.

In the first experiment, the mentioned ML models were applied on the dataset without applying feature selection technique. Additionally, **Table 2** obtains the results without feature selection The experimental results of the training history include the accuracy function evolution for training. The accuracy for best model for the classification task was DT is about 100%. **Figures 3, 4** represent the ROC curves and the confusion matrices. The DT classification model outperformed other models in all criteria, despite the fact that RF and DT models also produced remarkable results, as seen in the table. The KNN-trained method was the fastest when the model working periods were compared, whereas the NB estimation method was the slowest and performed badly in accuracy tests.

In the second experiment, the mentioned ML models were applied on the dataset after performing feature selection technique. **Table 3** shows the results with feature selection. From the table, although RF and DT models achieved outstanding results, the DT classification model outperformed the other models in general. **Figure 5** displays the confusion matrices for various models. and **Figure 6** demonstrates their respective ROC curves. When the working times of the models were compared, it was discovered that NB was the fastest trained method but achieved a bad accuracy whereas DT was the slowest estimation method but produced a high accuracy of 99.6%. Although the RF performed well, this training time was longer than the other models. Additionally, the proposed model was evaluated with different percentile in the features selection step. from **table 4**, it observed that the proposed model has been achieved an accuracy of 99% at 40% percentile. on the other hand, the proposed model has the least performance with an accuracy 91% at 10% percentile.

Finding pertinent features and selecting the appropriate algorithm is not sufficient; the algorithm's hyper-parameters need to be tuned to obtain the optimal configuration for the dataset. To achieve maximum effectiveness, hyper- parameter modifications were made to the models that were used, including (criterion = "entropy" or "gini", class weights = "none" or "balanced"). The applied algorithms and hyper-parameter settings are displayed in **Table 5,6,7,8**. The effectiveness of decision tree and random forest as machine-learning approaches for both data sets was shown by experimental findings. The accuracy difference between decision tree and random forest was very small, almost insignificant. The effectiveness of decision tree and random forest as machine-learning approaches for both data sets was shown by experimental findings. The accuracy difference between decision trees and random forests was very small, almost in- significant. The decision tree was substantially quicker at making predictions than the random forest. Based on the results of the two aforementioned data sets, the DT may be an appropriate method for solving real-time, fine-grained traffic classification `challenges**. Table 9** represent comparative accuracy of the proposed model with other literature techniques. **Figure 7,8** shows the ROC and CMs for the hyper-parameters ("**entropy**" ,"**balanced**") of DT model, **. Figure 9,10** shows the ROC and CMs for the hyper-parameters ("**entropy**" ,"**none**") of DT model, **. Figure 11,12** shows the ROC and CMs for the hyper-parameters ("**gini**", "**none**") of DT mode **Figure 13,14** shows the ROC and CMs for the hyper-parameters ("**gini**", "**balanced**") of DT model.

The confusion matrices of best classifiers for data subsets are **figs. 7,9,11,13** respectively. The confusion matrices of several best-performance classifier models show the respective statistics in terms of true positives, true negatives, and so on, for different traffic class data subsets. The matrices clearly reveal that the majority of predictions are accurate, and the trained models can be trusted. Three distinct performance indicators are utilized to study and evaluate their behavior and performance to one another. The measurements are F1-score, Recall, and Precision.

The figure 8,10,12,14 representing the ROC. Receiver Operator Characteristic is a probability curve that is utilized for several classes. It is commonly used to compare the performance of ML algorithms on unbalanced datasets. The area under the ROC curve measures the classifier's performance.

Table 2: Results of each algorithm without applying feature selection technique.

Results without feature selection					
Model	**Accuracy(%)**	**F1-score**	**Precision**	**Recall**	**Time(s)**
SVM	98.7	98	100	97	1.7
RF	99.6	100	100	99	1.7
DT	100	100	100	100	1.7
NB	59	61	88	60	1.8
KNN	98.1	98	99	97	1.6
LR	98.82	98	100	97	1.7

Table 3: Classification results of each algorithm after applying feature selection technique.

Results with feature selection					
Model	**Accuracy(%)**	**F1-score**	**Precision**	**Recall**	**Time(s)**
SVM	84.6	78	74	85	9.3
RF	99.53	99	99	99	14.6
DT	99.8	99.6	99.6	99.6	1.1
NB	97.8	97	98	97	1.2
KNN	99.1	99	99	99	1.7
LR	84.59	78	74	85	7.2

Table 4: Classification results of the proposed model at different percentile.

		CLASS					
Percentile	**Metrics**	**DNS**	**FTP**	**ICMP**	**P2P**	**VOIP**	**WWW**
10%	**PER**	0.00	100	90	73	95	97
	REC	0.00	98	93	96	100	94
	F1_Score	0.00	99	92	83	97	96
	Accuracy			**91.23 %**			
30%	**PER**	100	100	100	96	100	100
	REC	100	88	100	100	100	100
	F1_Score	100	94	100	98	100	100
	Accuracy			**99.41%**			
50%	**PER**	100	100	100	96	100	100
	REC	100	88	100	100	100	100
	F1_Score	100	94	100	98	100	100
	Accuracy			**99.41 %**			

Table 5: Detailed results at different hyper-parameters of DT with criterion=entropy, class weights= none.

		CLASS					
Hyper-parameters	Metrics	**DNS**	**FTP**	**ICMP**	**P2P**	**VOIP**	**WWW**
max_depth=1	**PER**	0.00	0.00	0.00	72	0.00	75
max_leaf_nodes=2	**REC**	0.00	0.00	0.00	97	0.00	100
criertion=entropy	**F1_Score**	0.00	0.00	0.00	83	0.00	86
class_weights=none max_features=24	**Accuracy**			**74.29 %**			
max_depth=6	**PER**	100	100	100	100	100	100
max_leaf_nodes=7	**REC**	100	100	100	100	100	100
criertion=entropy	**F1_Score**	100	100	100	100	100	100
class_weights=none max_features=24	**Accuracy**			**100 %**			

Table 6: Detailed results at different hyper-parameters of DT with criterion= gini , class weights= none.

		CLASS					
Hyper-parameters	Metrics	**DNS**	**FTP**	**ICMP**	**P2P**	**VOIP**	**WWW**
max_depth=1	**PER**	0.00	0.00	0.00	72	0.00	75
max_leaf_nodes=2	**REC**	0.00	0.00	0.00	97	0.00	100
criertion=gini	**F1_Score**	0.00	0.00	0.00	83	0.00	86
class_weights=none max_features=24	**Accuracy**			**74.29 %**			
max_depth=5	**PER**	100	100	100	97	100	100
max_leaf_nodes=6	**REC**	100	95	100	99	100	100
criertion=gini	**F1_Score**	100	1000	100	100	100	100
class_weights=none max_features=24	**Accuracy**			**99.53 %**			

Table 7: Detailed results at different hyper-parameters of DT with criterion=entropy, class weights= balanced.

		CLASS					
Hyper-parameters	**Metrics**	**DNS**	**FTP**	**ICMP**	**P2P**	**VOIP**	**WWW**
max_depth=1	**PER**	0.00	0.00	54	0.00	0.08	0.00
max_leaf_nodes=2	**REC**	0.00	0.00	100	0.00	100	0.00
criertion=entropy	**F1_Score**	0.00	0.00	70	0.00	0.14	0.00
class_weights=balanced max_features=24	**Accuracy**			**16.71 %**			
max_depth=6	**PER**	100	100	100	100	100	100
max_leaf_nodes=7	**REC**	100	100	100	100	100	100
criertion=entropy	**F1_Score**	100	100	100	100	100	100
class_weights=balanced max_features=24	**Accuracy**			**100 %**			

Table 8: Detailed results at different hyper-parameters of DT with criterion=gini, class weights= balanced.

		CLASS					
Hyper-parameters	Metrics	**DNS**	**FTP**	**ICMP**	**P2P**	**VOIP**	**WWW**
max_depth=1	**PER**	0.04	0.00	100	0.00	0.00	0.00
max_leaf_nodes=2	**REC**	100	0.00	100	0.00	0.00	0.00
criertion=gini	**F1_Score**	0.08	0.00	100	0.00	0.00	0.00
class_weights=balanced max_features=24	**Accuracy**			**14.22 %**			
max_depth=2	**PER**	0.04	100	100	0.00	0.00	0.00
max_leaf_nodes=3	**REC**	100	88	100	0.00	0.00	0.00
criertion=gini	**F1_Score**	0.08	94	100	0.00	0.00	0.00
class_weights=balanced max_features=24	**Accuracy**			**18.48 %**			
max_depth=4	**PER**	0.06	100	100	100	100	0.00
max_leaf_nodes=5	**REC**	100	88	100	97	100	0.00
criertion=gini	**F1_Score**	0.11	94	100	99	100	0.00
class_weights=balanced max_features=24	**Accuracy**			**40.40 %**			
max_depth=5	**PER**	78	100	100	100	100	100
max_leaf_nodes=6	**REC**	100	88	100	97	100	100
criertion=gini	**F1_Score**	87	94	100	99	100	100
class_weights=balanced max_features=24	**Accuracy**			**98.93 %**			
max_depth=7	**PER**	100	98	100	99	100	100
max_leaf_nodes=8	**REC**	100	98	100	99	100	100
criertion=gini	**F1_Score**	100	98	100	99	100	100
class_weights=balanced max_features=24	**Accuracy**			**99.76%**			

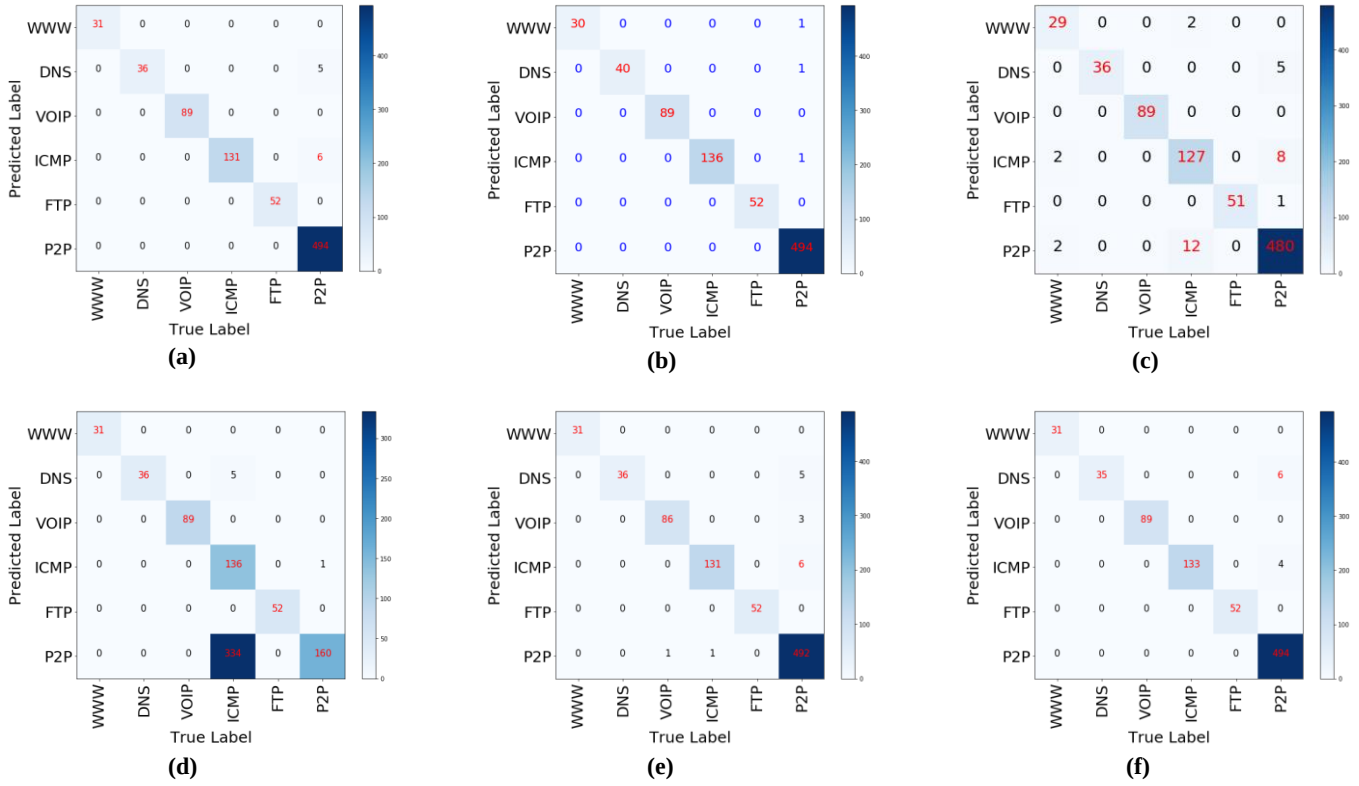

Figure 3: Confusion matrices without of the proposed model with different algorithms. (a) SVM, (b)RF, (c)DT (d) NB, (e) KNN, and (f) LR.

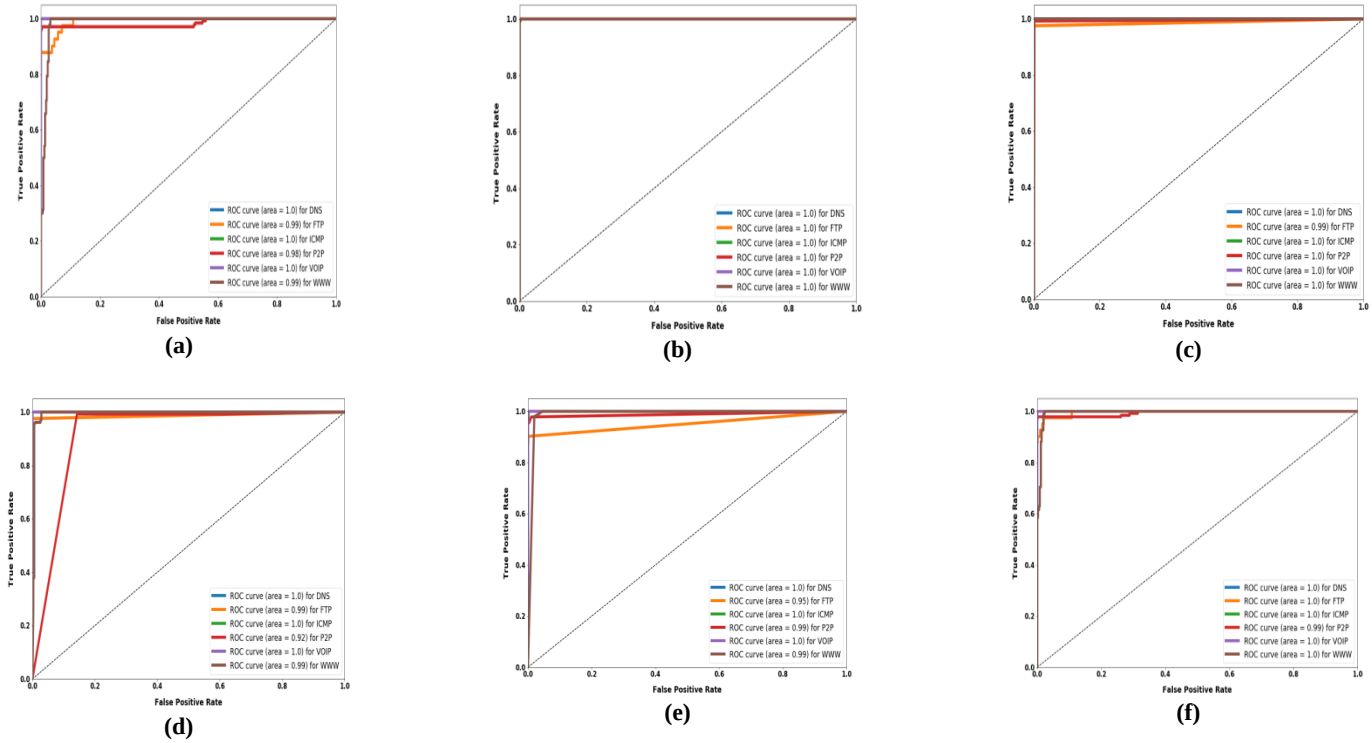

Figure 4: ROCs without of the proposed model with different algorithms (a) SVM, (b) RF, (c) DT (d) NB, (e) KNN, and (f) LR.

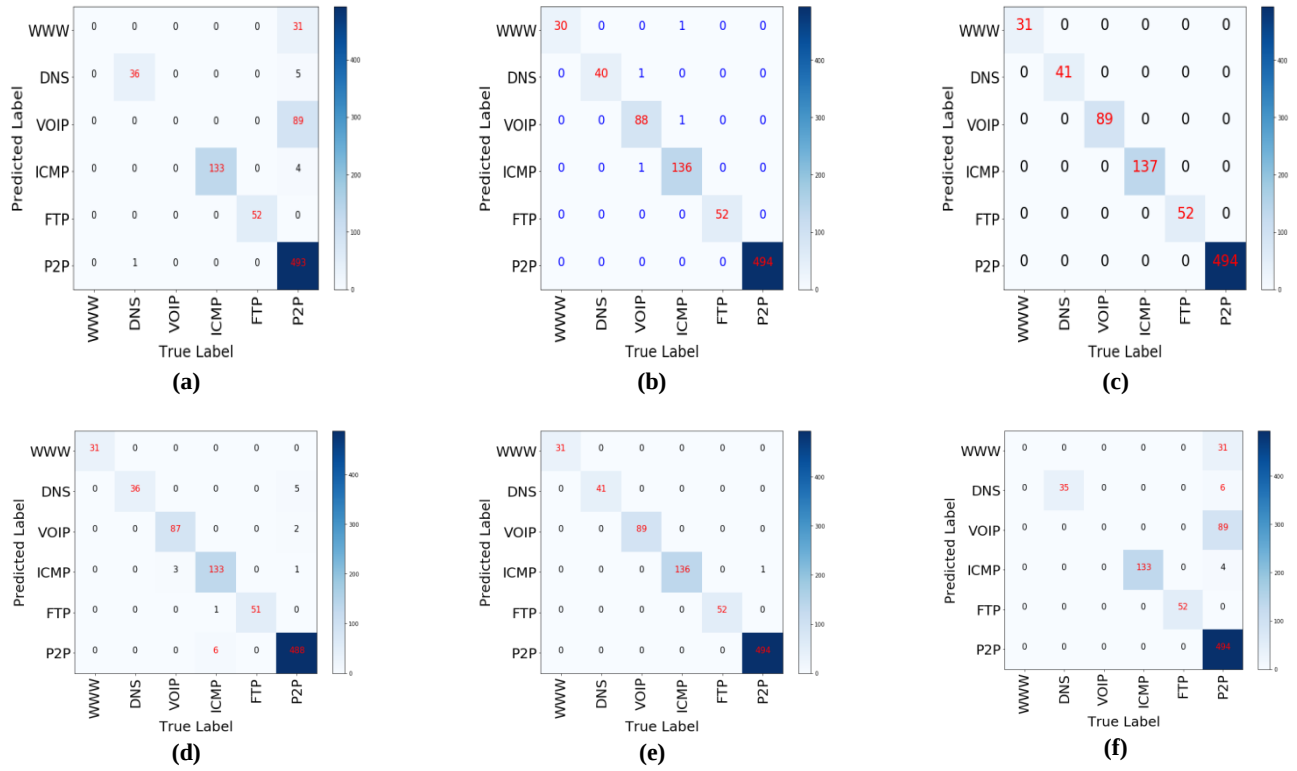

Figure 5: Confusion matrices with of the proposed model with different algorithms (a) SVM,(b) RF, (c) DT, (d) NB, (e) KNN, and (f) LR.

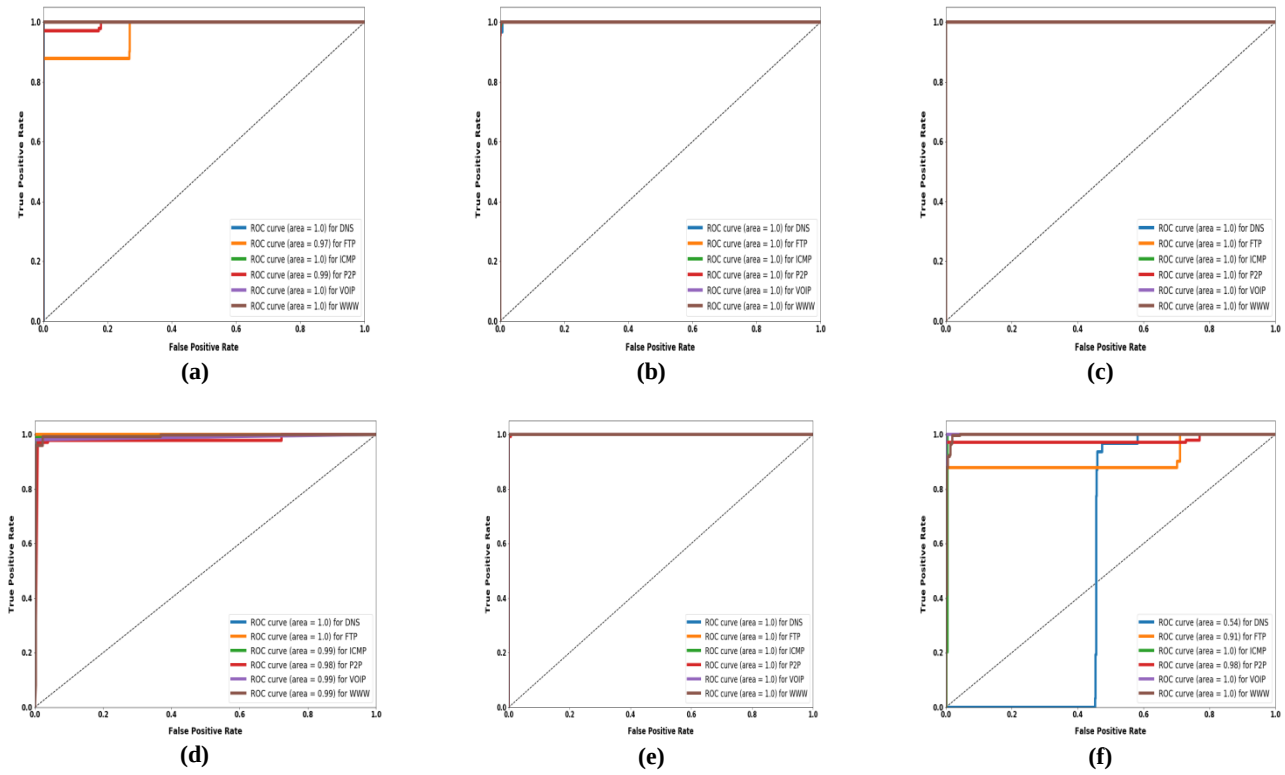

Figure 6: ROCs with of the proposed model with different algorithms (a) SVM,(b) RF, (c) DT, (d) NB, (e) KNN, and (f) LR.

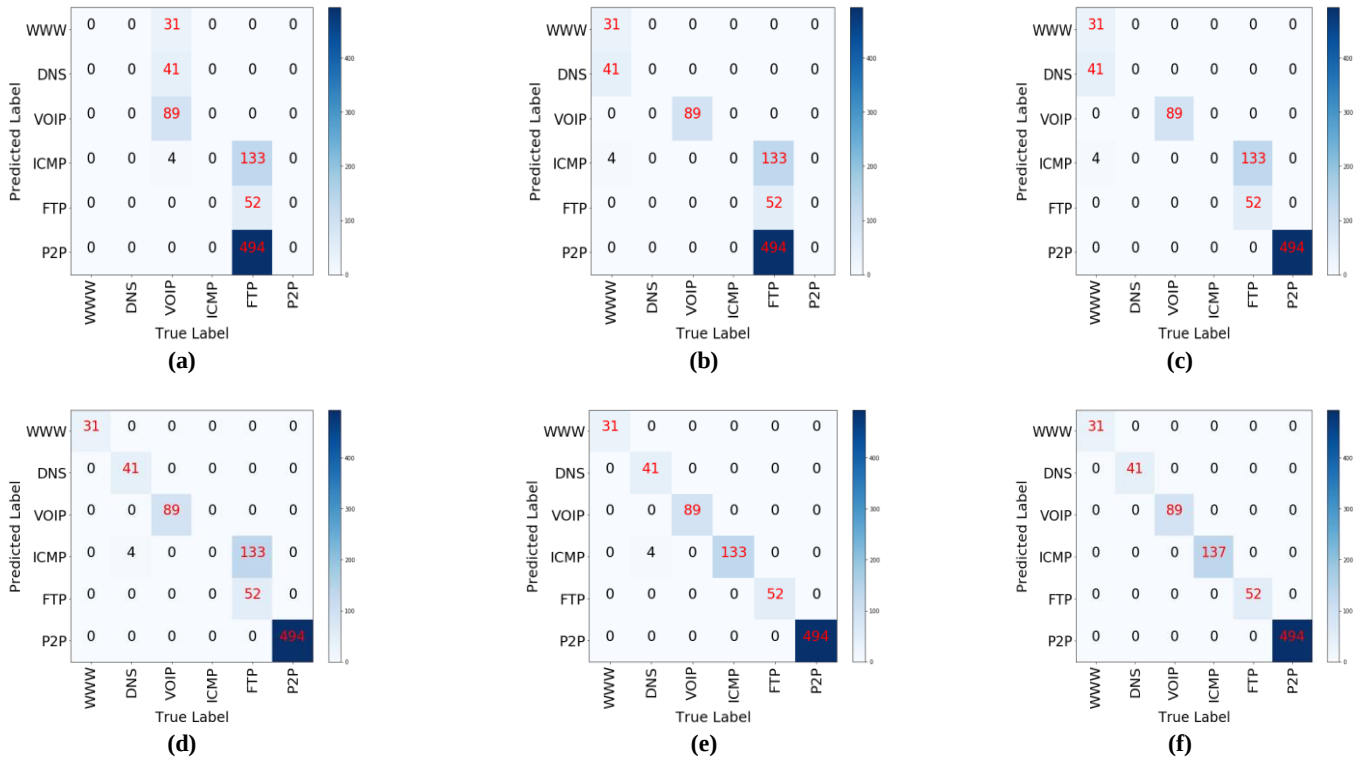

Figure 7: Confusion matrices of the DT**(entropy-balanced)** (a) , (b) , (c) , (d) , (e) and (f) .

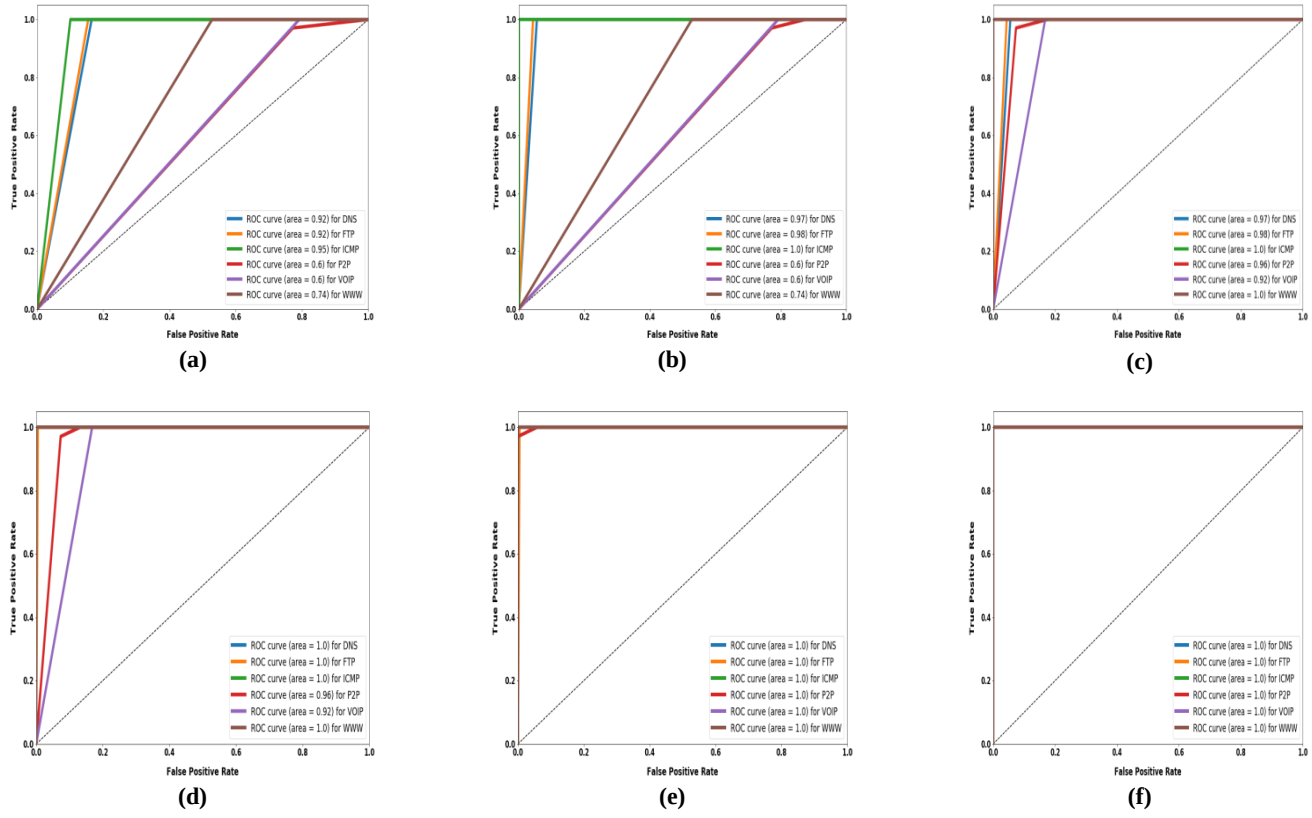

Figure 8: R O Cs of the DT**(entropy-balanced)** (a) , (b) , (c) , (d) , (e) and (f) .

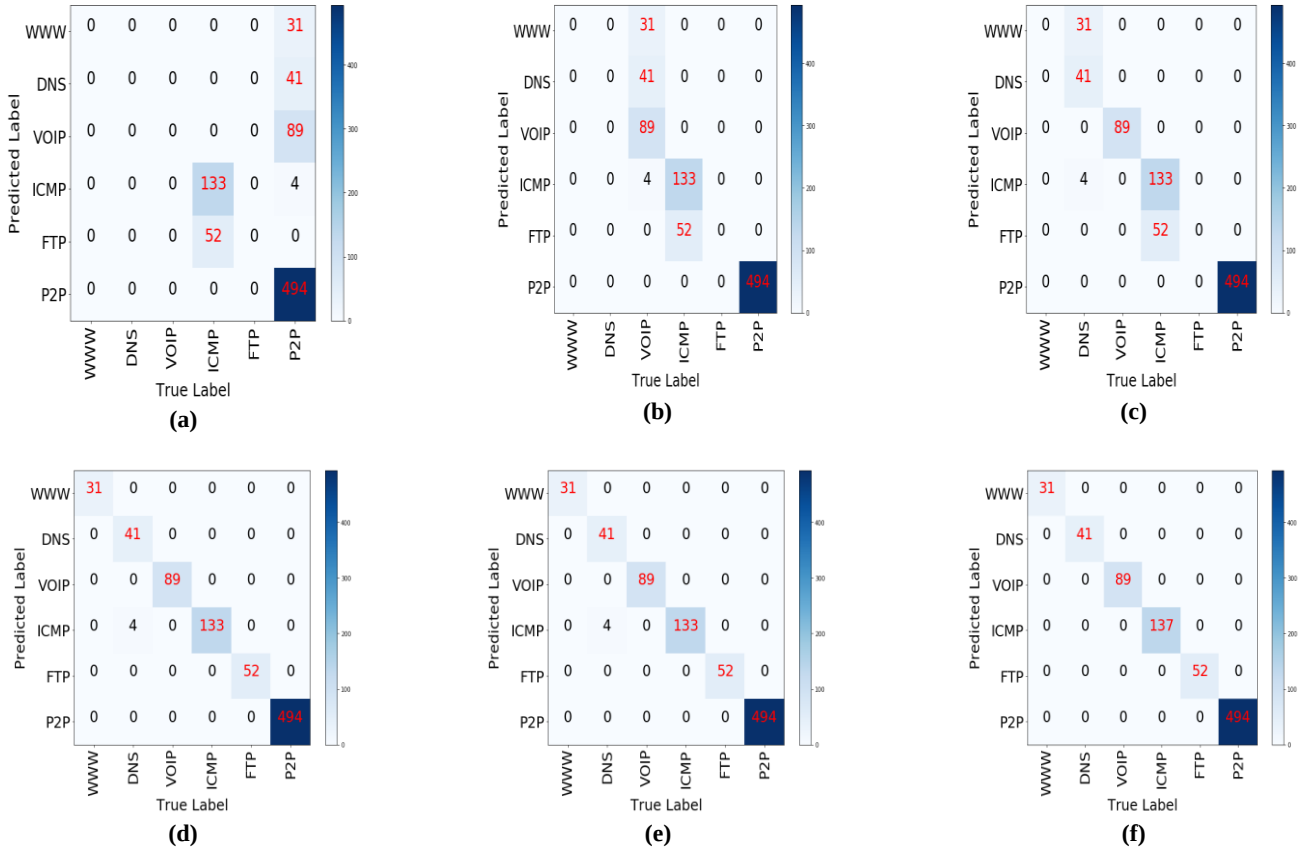

Figure 9: Confusion matrices of the DT**(entropy-None)** (a) , (b) , (c) , (d) , (e) and (f) .

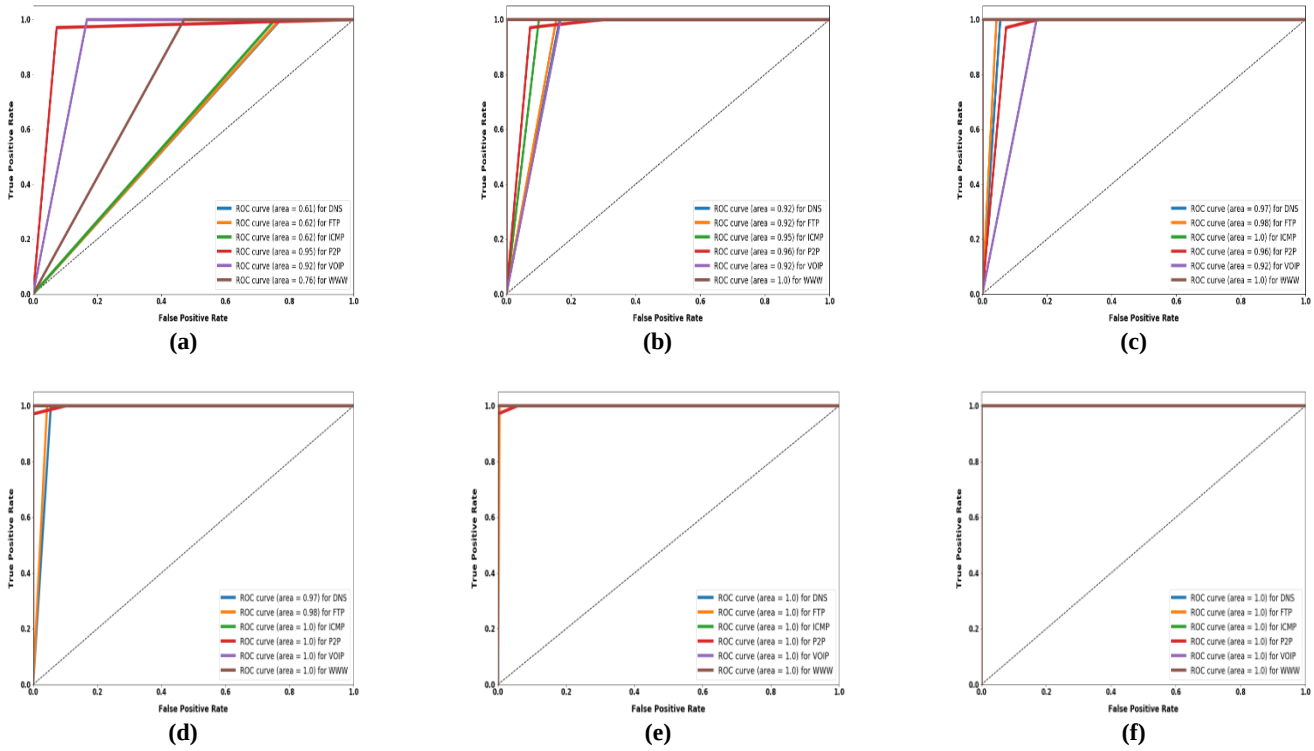

Figure 10: R O Cs of the DT**(entropy-None)** (a) , (b) , (c) , (d) , (e) and (f) .

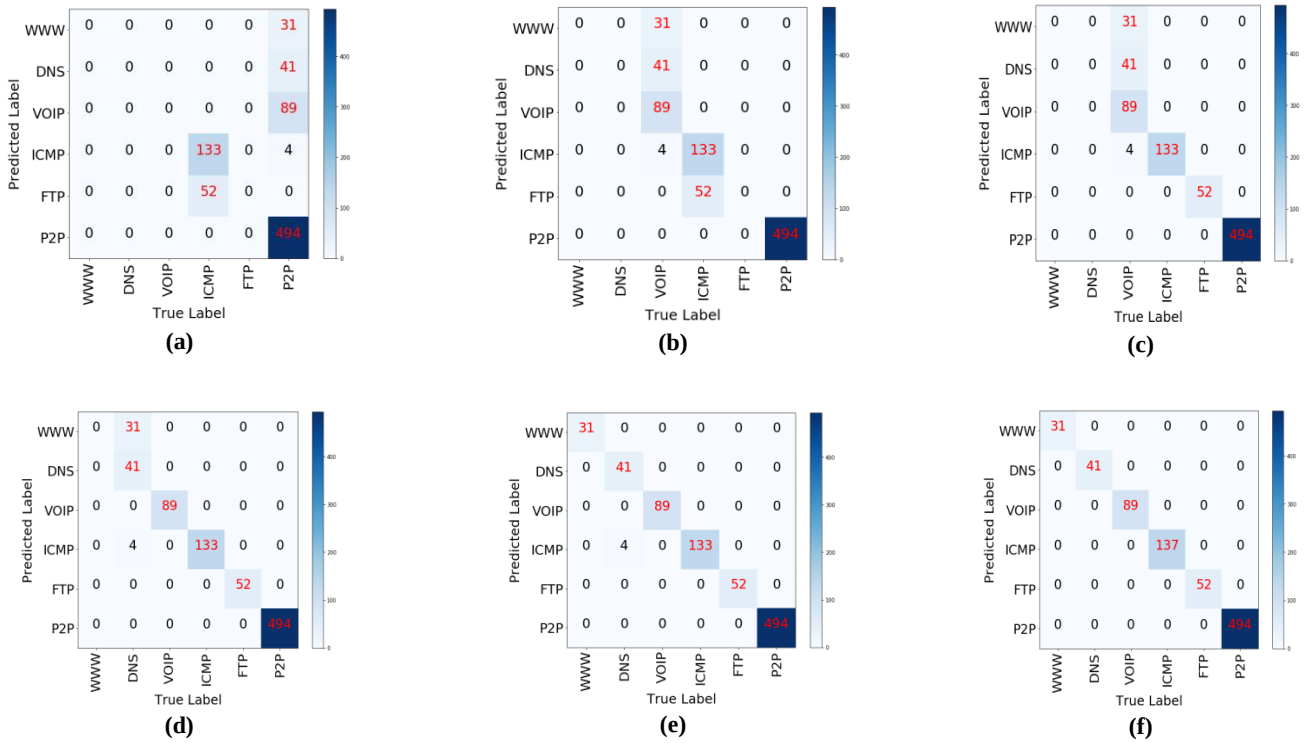

Figure 11: Confusion matrices of the DT**(gini-none)** (a) , (b) , (c) , (d) , (e) and (f) .

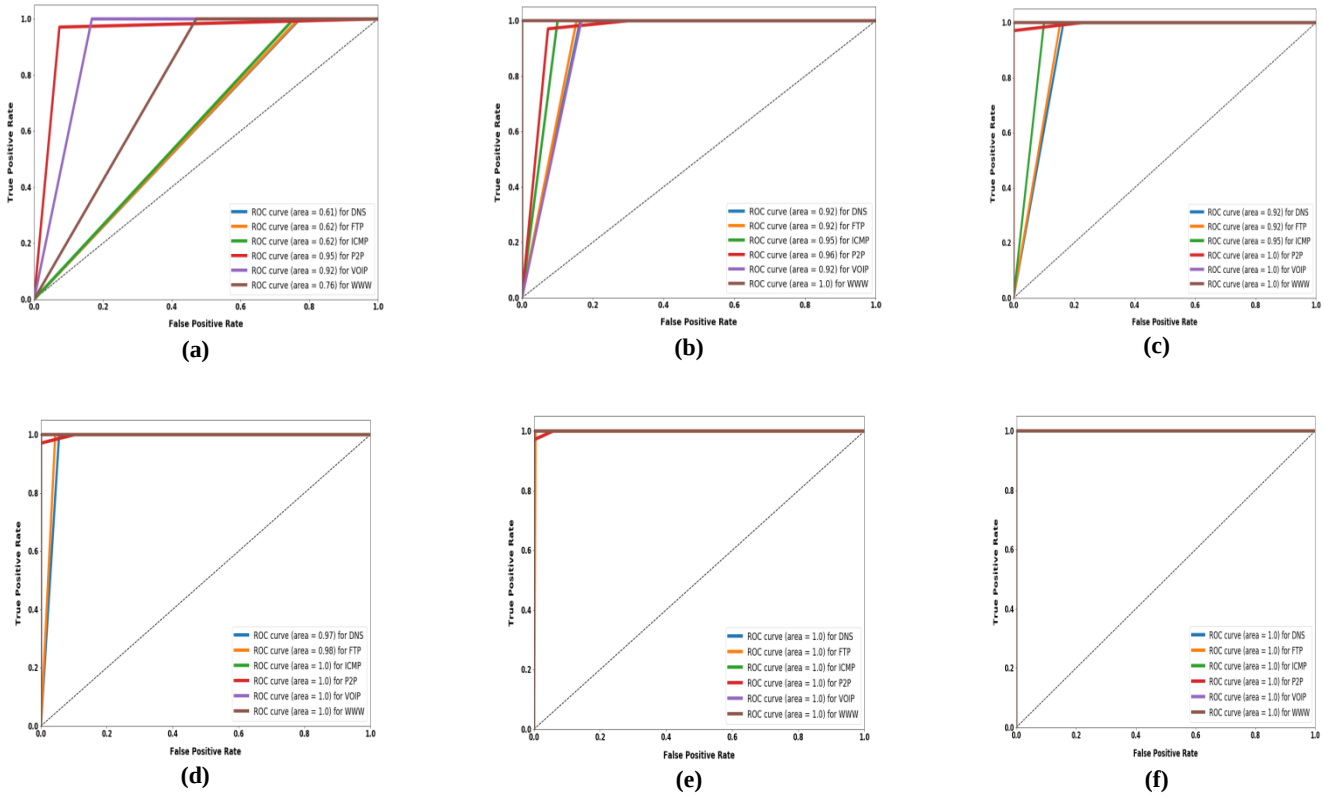

Figure 12: R O Cs of the DT**(gini-none)** (a) , (b) , (c) , (d) , (e) and (f) .

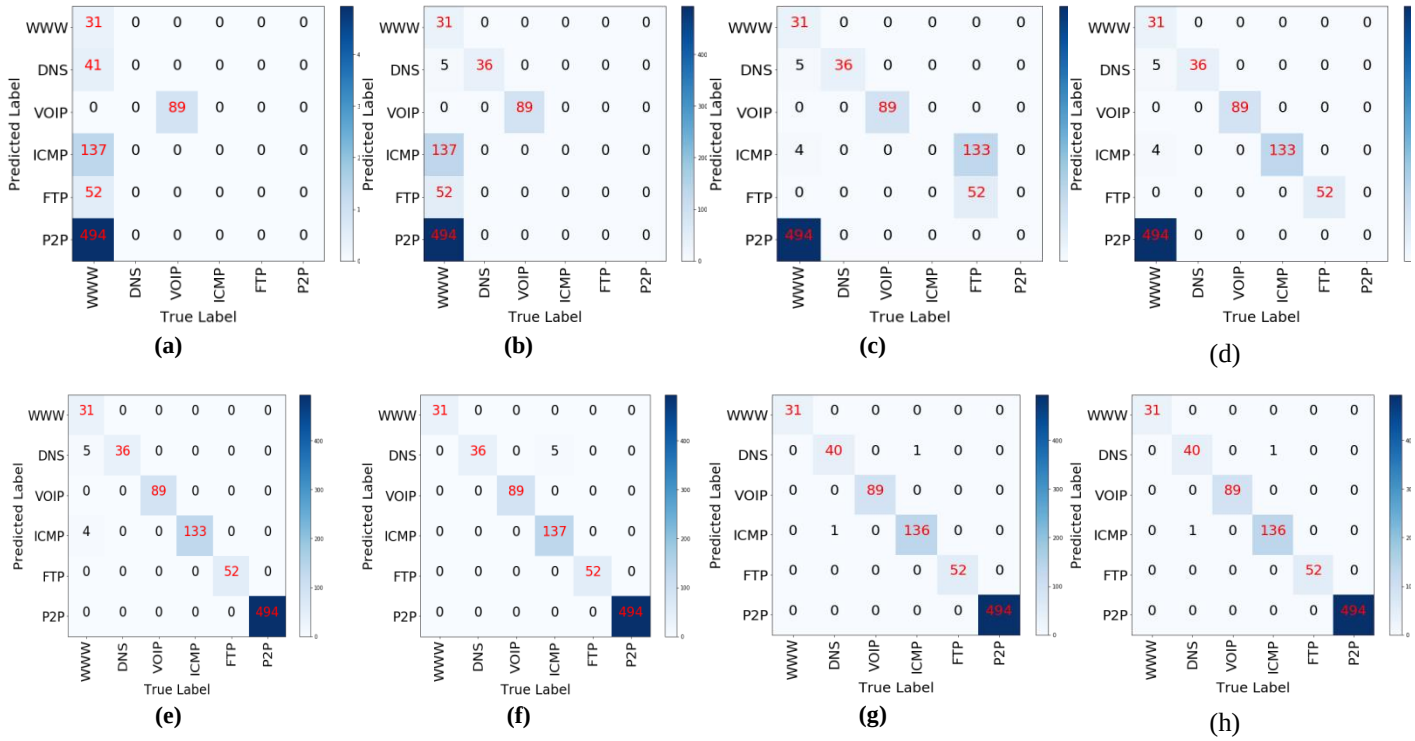

Figure 13: Confusion matrices of the DT**(gini-balanced)** (a),(b),(c), (d), (e), (f), (g)and(h)

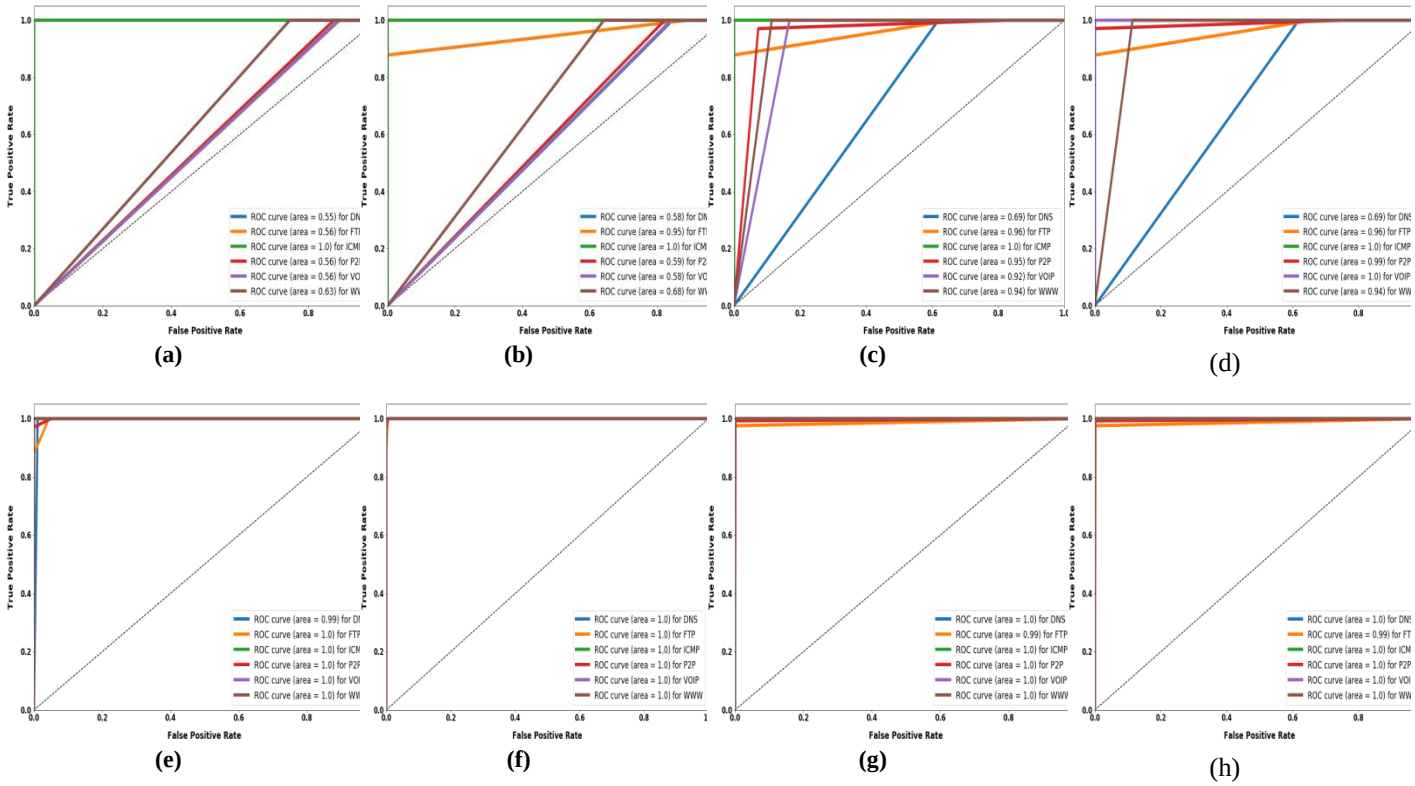

Figure 14: ROCs of the DT**(gini-balanced)** (a),(b), (c),(d) , (e) ,(f) ,(g) and (h)

Table 9. Comparative accuracy of the proposed model with other literature techniques.

Ref	MLTechnique	Features Selection	Model Output	Accuracy
Su et al. [10]	1D ,CNN,CNN-LSTM	11 flow features	6 applications	99%
Samaan et al. [11]	LR,RF,DT	9 flow features	8 applications	98%
Awad et al. [12]	MLMR framework	automatic by algorithm	8 applications	98%
Van et al. [13]	KNN,SVM,RF,LR	10 flow features	6 applications	99%
Aouedi et al. [14]	KNN,SVM,RF,MLP,DT	automatic by algorithm	10 applications	89%
Machoke et al. [15]	eXtreme Gradient Boosting RF,KNN	11 flow features	8 applications	90%
Karmous et al. [16]	KNN	automatic by algorithm	10 applications	99%
Khairi et al. [17]	SVM,DT	7 flow features	8 applications	99.2%
Gnanamonickam et al. [18]	MLT	automatic by algorithm	8 applications	95%
Dang et al. [19]	DT,RF	22 flow features	4 applications	96%
proposed model	k-NN, SVM, DT, RF, NP and LR	15 flow features	6 applications	99.8%

5. Conclusions

In this paper, SDN traffic classification model was described. A dataset was subjected to classification using six ML algorithms: KNN, LR, RF, DT, NB and SVM Classifier. the performance was better with the feature selected DT classifier. It achieved a 99.6% accuracy rate and a 100% F1 score. A number of issues must be resolved in order to move on with the task. First, only a straightforward topology was utilized to test the hypothesis, with the primary emphasis being placed on the correctness of the ML models. Although accuracy is required, the networks in the actual world are much more complicated, therefore accuracy is insufficient. A real-world network's performance is also directly impacted by other variables including scalability, availability, and others. Additionally, When the number of attributes increases, the traffic patterns that the clustering algorithm found must be improved while keeping the level of complexity manageable. Due to the fact that user behavior patterns vary from network tonetwork, this finding is also context-dependent. For instance, the quantity of clusters in a dataset derived from a data center and a dataset derived from a sensor network might differ. Our future work will examine more intricate models based on ensemble learning techniques, which combine multiple learning algorithms to produce better predictive performance than could be obtained from any one learning algorithm. This will broaden the corresponding ML studies and increase the accuracy of detection and prediction while also improving performance evaluation metrics.

References

[1] A. Zarzoor, N. Al-Jamali, and I. Al-Saedi, 'Traffic Classification of IoT Devices by Utilizing Spike Neural Network Learning Approach', *J. Math. Model. Algorithms*, vol. 10, pp. 639–646, May 2023, doi: 10.18280/mmep.100234.

[2] W.-J. Eom, Y.-J. Song, C.-H. Park, J.-K. Kim, G.-H. Kim, and Y.-Z. Cho, 'Network Traffic Classification Using Ensemble Learning in Software-Defined Networks', in *2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, Apr. 2021, pp. 089–092. doi: 10.1109/ICAIIC51459.2021.9415187.

[3] D. Nunez-Agurto, W. Fuertes, L. Marrone, and M. Macas, 'Machine Learning-Based Traffic Classification in Software-Defined Networking: A Systematic Literature Review, Challenges, and Future Research Directions', vol. 49, no. 4, 2022.

[4] A. M. Eldhai, M. Hamdan, S. Khan, M. Hamzah, and M. N. Marsono, 'Traffic Classification based on Incremental Learning Algorithms for the Software-Defined Networks', in *2022 International Conference on Frontiers of Information Technology (FIT)*, Dec. 2022, pp. 338–343. doi: 10.1109/FIT57066.2022.00068.

[5] A. Vulpe, C. Dobrin, A. Stefan, and A. Caranica, 'AI/ML-based real-time classification of Software Defined Networking traffic', in *Proceedings of the 18th International Conference on Availability, Reliability and Security*, in ARES '23. New York, NY, USA: Association for Computing Machinery, Aug. 2023, pp. 1–7. doi: 10.1145/3600160.3605078.

[6] D. Nuñez-Agurto, W. Fuertes, L. Marrone, E. Benavides-Astudillo, and M. Vásquez-Bermúdez, 'Traffic Classification in Software-Defined Networking by Employing Deep Learning Techniques: A Systematic Literature Review', in *Technologies and Innovation*, R. Valencia-García, M. Bucaram-Leverone, J. Del Cioppo-Morstadt, N. Vera-Lucio, and P. H. Centanaro-Quiroz, Eds., in Communications in Computer and Information Science. Cham: Springer Nature Switzerland, 2023, pp. 67–80. doi: 10.1007/978-3-031-45682-4_6.

[7] Ö. Tonkal and H. Polat, 'Traffic Classification and Comparative Analysis with Machine Learning Algorithms in Software Defined Networks', *Gazi Üniversitesi Fen Bilim. Derg. Part C Tasar. Ve Teknol.*, vol. 9, no. 1, pp. 71–83, Mar. 2021, doi: 10.29109/gujsc.869418.

[8] D. Kumar and J. Thakur, 'Handling Security Issues in Software-defined Networks (SDNs) Using Machine Learning', in *Computational Vision and Bio-Inspired Computing*, S. Smys, J. M. R. S. Tavares, and V. E. Balas, Eds., in Advances in Intelligent Systems and Computing. Singapore: Springer, 2022, pp. 263–277. doi: 10.1007/978-981-16-9573-5_20.

[9] V. Elagin, 'Traffic Classification Model in Software-Defined Networks with Artificial Intelligence Elements', *Proc. Telecommun. Univ.*, vol. 9, pp. 66–78, Nov. 2023, doi: 10.31854/1813-324X-2023-9-5-66-78.

[10] C. Su, Y. Liu, and X. Xie, 'Fine-grained Traffic Classification Based on Improved Residual Convolutional Network in Software Defined Networks', *IEEE Lat. Am. Trans.*, vol. 21, no. 4, pp. 565–572, Apr. 2023, doi: 10.1109/TLA.2023.10128928.

[11] S. S. Samaan and H. A. Jeiad, 'Architecting a machine learning pipeline for online traffic classification in software defined networking using spark', *IAES Int. J. Artif. Intell. IJ-AI*, vol. 12, no. 2, p. 861, Jun. 2023, doi: 10.11591/ijai.v12.i2.pp861-873.

[12] M. K. Awad, M. H. H. Ahmed, A. F. Almutairi, and I. Ahmad, 'Machine Learning-Based Multipath Routing for Software Defined Networks', *J. Netw. Syst. Manag.*, vol. 29, no. 2, p. 18, Jan. 2021, doi: 10.1007/s10922-020-09583-4.

[13] J. van Staden and D. Brown, 'An Evaluation of Machine Learning Methods for Classifying Bot Traffic in Software Defined Networks', in *Proceedings of Third International Conference on Sustainable Expert Systems*, S. Shakya, V. E. Balas, and W. Haoxiang, Eds., in Lecture Notes in Networks and Systems. Singapore: Springer Nature, 2023, pp. 979–991. doi: 10.1007/978-981-19-7874-6_72.

[14] 'Performance evaluation of feature selection and tree-based algorithms for traffic classification'. Accessed: Oct. 26, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9473580/

[15] M. Machoke, J. Mbelwa, J. Agbinya, and A. E. Sam, 'Performance Comparison of Ensemble Learning and Supervised Algorithms in Classifying Multi-label Network Traffic Flow', *Eng. Technol. Appl. Sci. Res.*, vol. 12, no. 3, Art. no. 3, Jun. 2022, doi: 10.48084/etasr.4852.

[16] N. Karmous, M. O.-E. Aoueileyine, M. Abdelkader, and N. Youssef, 'Enhanced Machine Learning-Based SDN Controller Framework for Securing IoT Networks', in *Advanced Information Networking and Applications*, L. Barolli, Ed., in Lecture Notes in Networks and Systems. Cham: Springer International Publishing, 2023, pp. 60–69. doi: 10.1007/978-3-031-28694-0_6.

[17] M. H. H. Khairi *et al.*, 'Detection and Classification of Conflict Flows in SDN Using Machine Learning Algorithms', *IEEE Access*, vol. 9, pp. 76024–76037, 2021, doi: 10.1109/ACCESS.2021.3081629.

[18] A. A. S. Gnanamonickam and P. D. Dr. B.Paramasivan M.E., 'Intelligent Traffic Classification Feature Engineering Technique (ITCFFE) For SDN Networks Based On Neural Networks', *resmilitaris*, vol. 13, no. 3, Art. no. 3, Mar. 2023.

[19] T. L. Dang and V. C. Do, 'Fine-Grained Network Traffic Classification Using Machine Learning: Evaluation and Comparison', in *Soft Computing: Biomedical and Related Applications*, vol. 981, N. H. Phuong and V. Kreinovich, Eds., in Studies in Computational Intelligence, vol. 981. , Cham: Springer International Publishing, 2021, pp. 151–162. doi: 10.1007/978-3-030-76620-7_13.

[20] P. K. Mondal, L. P. Aguirre Sanchez, E. Benedetto, Y. Shen, and M. Guo, 'A dynamic network traffic classifier using supervised ML for a Docker-based SDN network', *Connect. Sci.*, vol. 33, no. 3, pp. 693–718, Jul. 2021, doi: 10.1080/09540091.2020.1870437.

[21] G. Karn, B. Sapkota, and B. R. Dawadi, 'Traffic Classification and Load Balancing in SDN Environment', 2023.

[22] A. A. El-serwy, E. AbdElhalim, and M. A. Mohamed, 'Network Slicing Based on Real-Time Traffic Classification in Software Defined Network (SDN) using Machine Learning', *MEJ Mansoura Eng. J.*, vol. 47, no. 3, pp. 1–10, Sep. 2022, doi: 10.21608/bfemu.2022.261455.

[23] O. Belkadi, A. Vulpe, Y. Laaziz, and S. Halunga, 'ML-Based Traffic Classification in an SDN-Enabled Cloud Environment', *Electronics*, vol. 12, no. 2, Art. no. 2, Jan. 2023, doi: 10.3390/electronics12020269.

[24] 'SDN traffic'. Accessed: Oct. 26, 2023. [Online]. Available: https://kaggle.com/code/sherifmahgoub/sdn-traffic

[25] N. Ahmed *et al.*, 'Network Threat Detection Using Machine/Deep Learning in SDN-Based Platforms: A Comprehensive Analysis of State-of-the-Art Solutions, Discussion, Challenges, and Future Research Direction', *Sensors*, vol. 22, no. 20, Art. no. 20, Jan. 2022, doi: 10.3390/s22207896.

[26] H. Qu, J. Jiang, J. Zhao, Y. Zhang, and J. Yang, 'A novel method for network traffic classification based on robust SUPPORT VECTOR MACHINE', *Trans. Emerg. Telecommun. Technol.*, vol. 31, no. 11, p. e4092, Nov. 2020, doi: 10.1002/ett.4092.

[27] J. Zhou, A. H. Gandomi, F. Chen, and A. Holzinger, 'Evaluating the Quality of Machine Learning Explanations: A Survey on Methods and Metrics', *Electronics*, vol. 10, no. 5, p. 593, Mar. 2021, doi: 10.3390/electronics10050593.