CLASSIFICATION OF SOFTWARE-DEFINED NETWORK TRAFFIC TO PROVIDE QUALITY OF SERVICE

Yadaiah Jella, Dr M Venkat Reddy, Raju Munavath
Assistance Professor, Professor, Assitance Professor
Dept of CSE
Sree Dattha Institute of Engineering and Science

ABSTRACT

Software Defined Networking (SDN) is a revolutionary paradigm that separates the control plane from the data plane, enabling more dynamic and programmable network management. Quality of Service (QoS) is essential for ensuring reliable and predictable network performance. Classifying SDN traffic for QoS management has become crucial in optimizing network resources. Traditional network management systems often rely on fixed configurations and routing protocols. While effective in many scenarios, they may lack the flexibility and adaptability needed to provide granular QoS based on dynamic traffic conditions. The primary challenge is to develop a classification system that accurately identifies different types of traffic in an SDN environment. This involves creating algorithms that can analyze network packets and categorize them based on their characteristics and requirements for quality of service. Therefore, the need of dynamic and complex networks, classifying traffic is vital for ensuring different types of data receive the appropriate level of service. This includes prioritizing real-time communication, ensuring sufficient bandwidth for critical applications, and managing resources efficiently. Traditional networks may struggle to adapt to changing traffic patterns, highlighting the importance of SDN and QoS. The project, "Classification of Software Defined Network Traffic to Provide Quality of Service," aims to revolutionize network management by leveraging the capabilities of SDN for QoS optimization. By developing advanced traffic classification algorithms, this research endeavors to create a system capable of autonomously and accurately categorizing network traffic in real-time. This classification allows for the implementation of specific QoS policies tailored to different types of traffic, ensuring optimal network performance and resource allocation. This advancement holds great promise for enhancing the efficiency and effectiveness of network operations, particularly in environments with diverse and dynamic traffic patterns

INTRODUCTION

1.1 History

In the ever-evolving landscape of networking, the advent of Software Defined Networking (SDN) has marked a paradigm shift, separating the control plane from the data plane. This groundbreaking approach brings forth increased dynamism and programmability in network management, necessitating a keen focus on Quality of Service (QoS) to uphold reliable and predictable network performance. Unlike traditional network management systems, which often hinge on fixed configurations and routing protocols, SDN introduces a level of flexibility and adaptability crucial for granular QoS management amid dynamic traffic conditions. The core challenge lies in crafting a robust classification system adept at accurately discerning various types of traffic within an SDN environment. This undertaking involves the development of sophisticated algorithms capable of scrutinizing network packets, categorizing them based on their distinct characteristics, and aligning them with specific QoS requirements.

In the realm of dynamic and intricate networks, the classification of traffic emerges as a vital linchpin for ensuring that diverse data types receive bespoke levels of service. This encompasses the prioritization of real-time communication, the assurance of ample bandwidth for critical applications, and the efficient management of resources. Traditional networks, with their inherent limitations in adapting to fluctuating traffic patterns, underscore the imperative role of SDN in conjunction with QoS. Enter the ambitious project, "Classification of Software Defined Network Traffic to Provide Quality of Service." Its overarching objective is to revolutionize network management by harnessing the inherent capabilities of SDN to optimize QoS. The crux of this research lies in the development of cutting-edge traffic classification algorithms, paving the way for a system that can autonomously and precisely categorize network traffic in real-time. This classification prowess, in turn, facilitates the implementation of tailored QoS policies for different types of traffic, thereby ensuring an unparalleled level of network performance and resource allocation.

The potential impact of this advancement is substantial, promising to enhance the efficiency and effectiveness of network operations, particularly in environments characterized by diverse and dynamic traffic patterns. As the digital landscape continues to evolve, the fusion of SDN and QoS emerges as a beacon of progress, ushering in a new era of network management where adaptability and precision take center stage.

1.2 Problem statement

In the landscape of modern networking, the integration of Software Defined Networking (SDN) has introduced a revolutionary shift in network management paradigms. However, despite its transformative potential, the effective implementation of Quality of Service (QoS) within SDN environments presents a formidable challenge. Traditional network management systems rely on static configurations and predetermined routing protocols, which may lack the agility required to adapt to the dynamic and diverse nature of contemporary network traffic. The core problem lies in the absence of a comprehensive and adaptable classification system capable of accurately identifying various types of traffic in real-time. Without such a system, the optimization of QoS for different traffic profiles becomes inherently limited, hindering the ability to prioritize critical applications, ensure sufficient bandwidth for real-time communication, and manage resources efficiently. Therefore, the absence of a robust and dynamic traffic classification mechanism stands as a critical impediment to unlocking the full potential of SDN for QoS management, necessitating innovative solutions to address this pressing challenge.

1.3 Motivation

The motivation for undertaking research in the realm of "Classification of Software Defined Network Traffic to Provide Quality of Service" stems from the imperative need to address the limitations of traditional network management systems and propel the capabilities of Software Defined Networking (SDN) towards optimal Quality of Service (QoS) delivery. As the digital landscape continues to evolve, networks encounter increasingly diverse and dynamic traffic patterns, rendering conventional, static approaches inadequate. The motivation lies in unlocking the full potential of SDN, which, by decoupling the control and data planes, offers unparalleled programmability and adaptability. The envisioned research aims to bridge the gap between SDN's inherent flexibility and the pressing demand for real-time, tailored QoS. The absence of a sophisticated traffic classification system impedes the seamless alignment of QoS policies with dynamic traffic conditions. By developing advanced algorithms capable of accurately categorizing network traffic in real-time, the research seeks to empower SDN to autonomously optimize QoS. The ultimate goal is to enhance network efficiency, ensure equitable resource allocation, and meet the diverse demands of contemporary applications, thereby providing a robust foundation for the future of network management in the face of evolving digital landscapes.

1.4 Applications

The application of the research project, "Classification of Software Defined Network Traffic to Provide Quality of Service," holds significant implications for advancing the field of network management and ensuring optimal performance in diverse and dynamic environments.

Firstly, the developed traffic classification algorithms have practical applications in real-time QoS optimization within Software Defined Networking (SDN) architectures. The ability to accurately categorize network traffic enables the implementation of tailored QoS policies, addressing the unique requirements of different data types. This ensures that critical applications receive the necessary bandwidth, real-time communication is prioritized, and network resources are efficiently managed, contributing to enhanced overall network performance.

Moreover, the application of these advanced algorithms extends to the realm of network security. The precise identification of network traffic types allows for more effective monitoring and detection of anomalous patterns or potential security threats. By understanding the characteristics of different traffic, security measures can be dynamically adjusted to respond to emerging risks, thereby fortifying the network against potential vulnerabilities.

Furthermore, the research findings have implications for industries and sectors heavily reliant on network performance, such as telecommunication, finance, healthcare, and transportation. The ability to dynamically adapt QoS parameters based on the nature of the traffic ensures a consistent and reliable network experience for users and applications in these critical domains.

In the context of emerging technologies like the Internet of Things (IoT) and 5G networks, the application of refined traffic classification contributes to efficient resource utilization and responsiveness, which are vital for the seamless integration of these technologies into the existing network infrastructure.

2.LITERATURE SURVEY

In this section, we compare recently published research relevant to our work. Conti et al. [8] developed a framework to infer user actions executed on mobile apps based on packet sizes and their order information. Park and Kim [9] target KakaoTalk, a mobile instant messaging service, and proposed a framework to infer user activities by passive analysis of network traffic. Saltaformaggio et al. [**10**] proposed NetScope, a tool to identify user activities generated by mobile apps, based on the statistics originating from the Internet Protocol (IP) headers. The AppScanner [**11**] framework was implemented for the real-time identification of Android apps from encrypted network traffic. All these methods employ classical ML algorithms such as k-nearest neighbour and random forest. However, their performance heavily depends on human-generated features, which is significantly time-consuming and limited in generalizability.

DL obviates the need to perform feature selections by a domain expert and it has a higher capacity to learn highly complicated patterns compared to traditional ML methods. Recent work has demonstrated the efficacy of DL methods to perform traffic classification. In [12], a framework named ‘Deep Packet’ was presented that achieved both traffic characterization, where the network traffic was categorized into major classes (e.g., FTP and P2P), and application identification (e.g., BitTorrent and Skype). Deep Packet framework embedded Stacked Autoencoder (SAE) and One Dimensional Convolutional Neural

Networks (1D CNN) to classify network traffic. Experiments were conducted using the ISCX public dataset [13] and the model achieved a recall rate of 0.98 in the application identification task and 0.94 in the traffic categorization task.

Recurrent Neural Networks (RNN) and CNN were applied to perform application-level identification in [14]. Their DL model used packet-level features such as ports, payload bytes, TCP window size, inter-arrival times of packets, and packet direction. This combination of the RNN and CNN model attained an accuracy of 96%.

In [5], an end-to-end encrypted traffic classification model with 1D CNN was proposed. Feature extraction, feature selection, and classifiers were integrated into a single framework in the model, which automatically learned the non-linear relationship between the raw input and expected output. This model was validated with the ISCX public dataset and showed a significant improvement over the C4.5 ML method.

In [15], DataNet was introduced as an application-aware framework for smart home networks. It was developed and evaluated using three deep learning-based approaches, namely multilayer perceptron, stacked autoencoder, and CNN. Experiments were conducted using the ISCX public dataset with encrypted data samples from 15 applications. The experimental results showed that recall, precision, and f1 score were all greater than 92%.

3.PROPOSED SYSTEM

3.1 Overview

This project uses the Tkinter library to create a graphical user interface (GUI) for a software application related to the classification of software-defined network (SDN) traffic to provide quality of service (QoS). The application seems to involve the use of machine learning algorithms, specifically an ensemble of classifiers, to classify network traffic into different categories.

- Importing Libraries:
- The code starts by importing necessary libraries for GUI development (tkinter), data visualization (matplotlib, seaborn), data manipulation (numpy, pandas), and machine learning (scikit-learn).
- Creating the Main GUI Window:
- The main GUI window is created using Tkinter. The window title is set, and its dimensions are specified.
- Global Variables:
- Several global variables are declared to store information about the dataset, model, and evaluation metrics.
- Functions:
- Several functions are defined to perform different tasks when specific GUI buttons are clicked. Here's a brief overview of these functions:
- uploadDataset: Allows the user to upload a dataset, displays information about the dataset, and shows a bar graph of network categories.

- DatasetPreprocessing: Preprocesses the dataset by handling missing values, encoding non-numeric data, and normalizing the data.
- splitDataset: Splits the dataset into training and testing sets.
- calculateMetrics: Calculates and displays various classification metrics such as accuracy, precision, recall, and F1 score, and shows a confusion matrix.
- runEnsemble: Trains an ensemble model (Random Forest, Decision Tree, SVM) on the training data and evaluates its performance.
- graph: Displays a bar graph comparing different algorithms based on accuracy, precision, recall, and F1 score.
- predict: Allows the user to upload test data, preprocesses it, and predicts the network traffic classification using the trained ensemble model.
- GUI Components:
- Labels, buttons, and text areas are created and placed in the GUI window to facilitate user interaction.
- Main Event Loop:
- The main event loop (main.mainloop()) keeps the GUI application running and responsive to user actions.

3.3 Proposed Ensemble Algorithm

Ensemble learning is a powerful paradigm in machine learning that leverages the diversity of multiple models to make more accurate and robust predictions. In this project, we employ a Voting Classifier to integrate the capabilities of three distinct algorithms: Random Forest (RF), Decision Tree (DT), and Support Vector Machine (SVM). Each algorithm brings its own unique strengths to the ensemble, creating a comprehensive approach for the classification of software-defined network (SDN) traffic.

Random Forest Algorithm

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

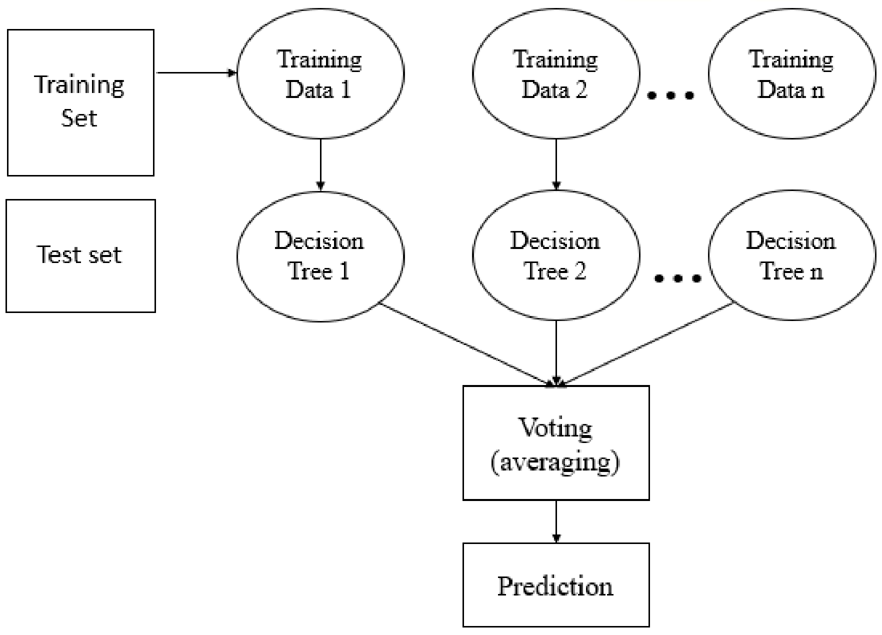


Figure 3.1: Random Forest algorithm

Random Forest algorithm

Step 1: In Random Forest n number of random records are taken from the data set having k number of records.

Step 2: Individual decision trees are constructed for each sample.

Step 3: Each decision tree will generate an output.

Step 4: Final output is considered based on Majority Voting or Averaging for Classification and regression respectively.

Important Features of Random Forest

- **Diversity**- Not all attributes/variables/features are considered while making an individual tree, each tree is different.
- **Immune to the curse of dimensionality**- Since each tree does not consider all the features, the feature space is reduced.
- **Parallelization**-Each tree is created independently out of different data and attributes. This means that we can make full use of the CPU to build random forests.
- **Train-Test split**- In a random forest we don't have to segregate the data for train and test as there will always be 30% of the data which is not seen by the decision tree.
- **Stability**- Stability arises because the result is based on majority voting/ averaging.

3.3.1 Assumptions for Random Forest

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

Below are some points that explain why we should use the Random Forest algorithm

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

3.3.2 Types of Ensembles

Before understanding the working of the random forest, we must look into the ensemble technique. Ensemble simply means combining multiple models. Thus, a collection of models is used to make predictions rather than an individual model. Ensemble uses two types of methods:

Bagging– It creates a different training subset from sample training data with replacement & the final output is based on majority voting. For example, Random Forest. Bagging, also known as Bootstrap Aggregation is the ensemble technique used by random forest. Bagging chooses a random sample from the data set. Hence each model is generated from the samples (Bootstrap Samples) provided by the Original Data with replacement known as row sampling. This step of row sampling with replacement is called bootstrap. Now each model is trained independently which generates results. The final output is based on majority voting after combining the results of all models. This step which involves combining all the results and generating output based on majority voting is known as aggregation.

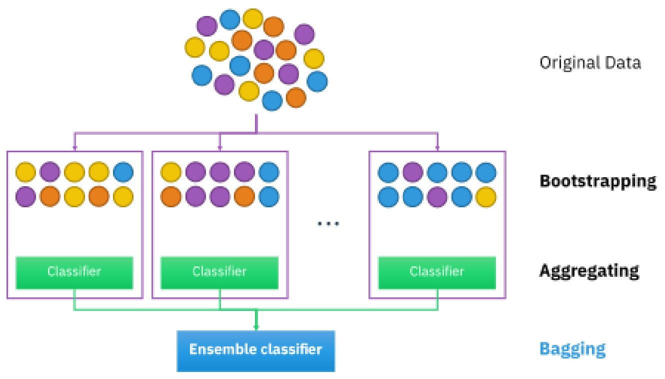


Figure 3.2: RF Classifier analysis.

Boosting– It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy. For example, ADA BOOST, XG BOOST.

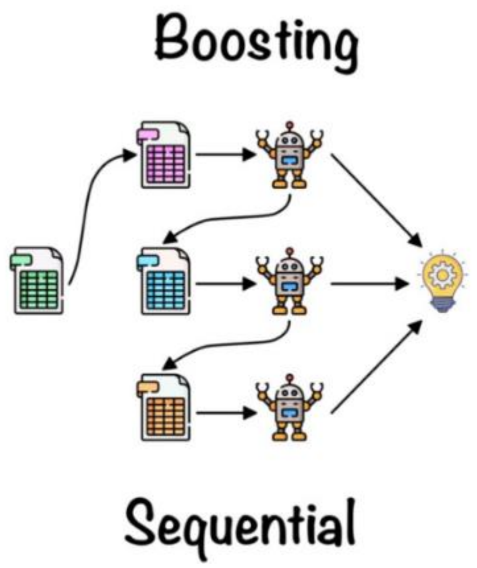


Figure 3.3: Boosting RF Classifier.

Advantages of Random Forest

- It can be used in classification and regression problems.
- It solves the problem of overfitting as output is based on majority voting or averaging.
- It performs well even if the data contains null/missing values.
- Each decision tree created is independent of the other thus it shows the property of parallelization.
- It is highly stable as the average answers given by a large number of trees are taken.
- It maintains diversity as all the attributes are not considered while making each decision tree though it is not true in all cases.
- It is immune to the curse of dimensionality. Since each tree does not consider all the attributes, feature space is reduced.

Roles of voting classifier

Random Forest contributes a level of complexity and adaptability to the ensemble. By constructing multiple decision trees and aggregating their predictions, it captures intricate relationships within the SDN traffic data. This diversity aids in enhancing the overall predictive performance of the Voting Classifier.

Support Vector Machine Algorithm

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

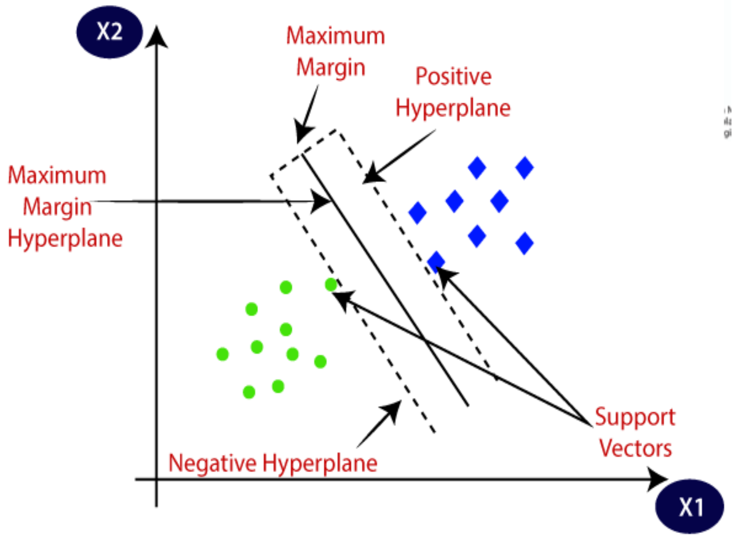


Figure 3.4 Analysis of SVM

Example: SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:

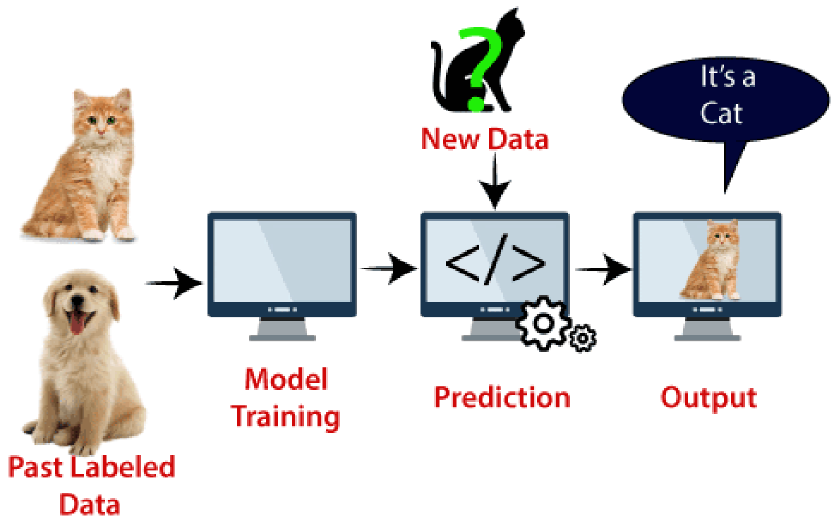


Figure 3.5: Basic classification using SVM

Types of SVM: SVM can be of two types:

Linear SVM: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

Non-linear SVM: Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier

3.3 SVM working

Linear SVM: The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair (x1, x2) of coordinates in either green or blue. Consider the below image:

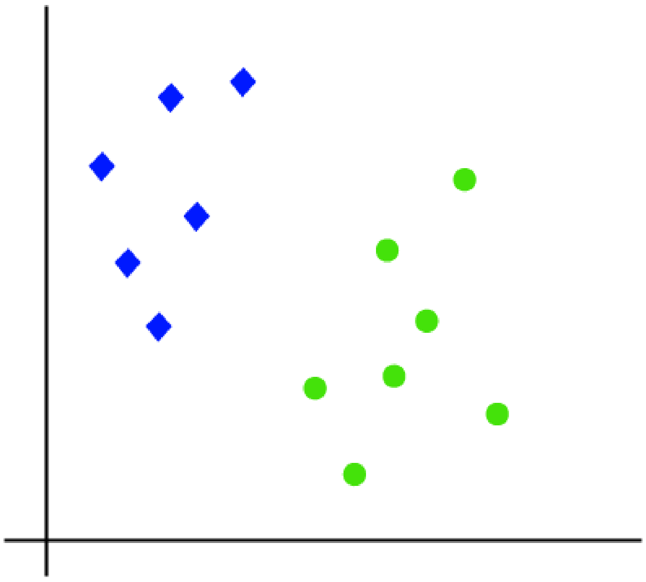


Figure 3.6: Linear SVM

So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:

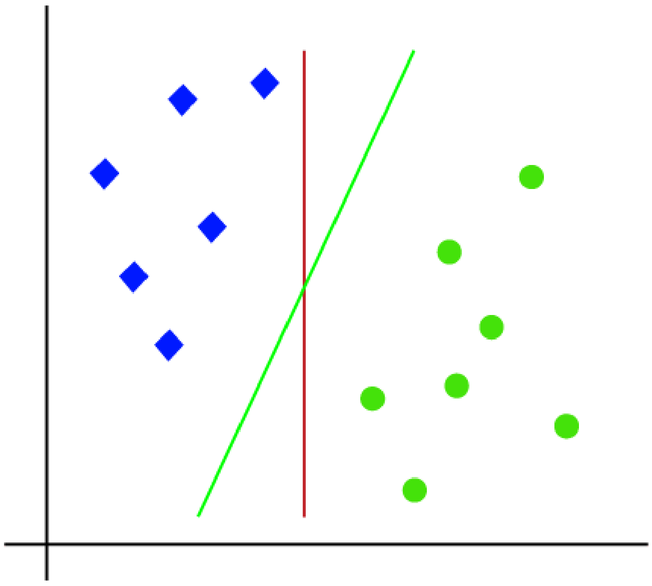


Figure 3.7: Test-Vector in SVM

Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a hyperplane. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as margin. And the goal of SVM is to maximize this margin. The hyperplane with maximum margin is called the optimal hyperplane.

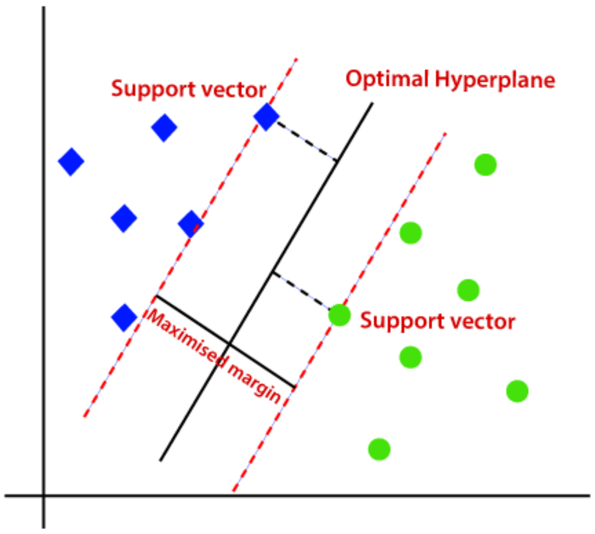


Figure 3.8: Classification in SVM

Non-Linear SVM: If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:

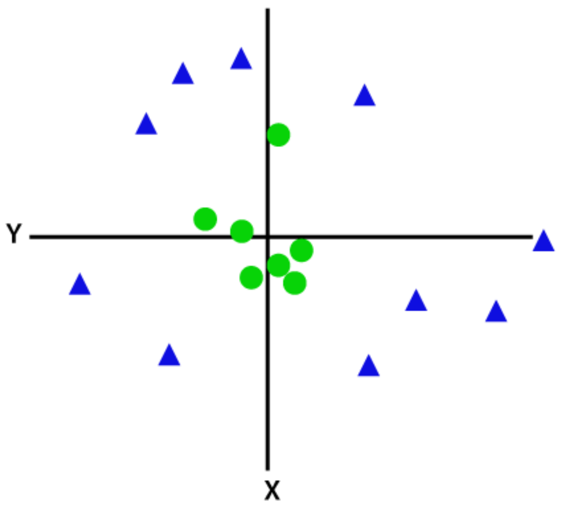


Figure 3.9: Non-Linear SVM

So, to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third-dimension z. It can be calculated as:

$$z=x^2 +y^2$$

By adding the third dimension, the sample space will become as below image:

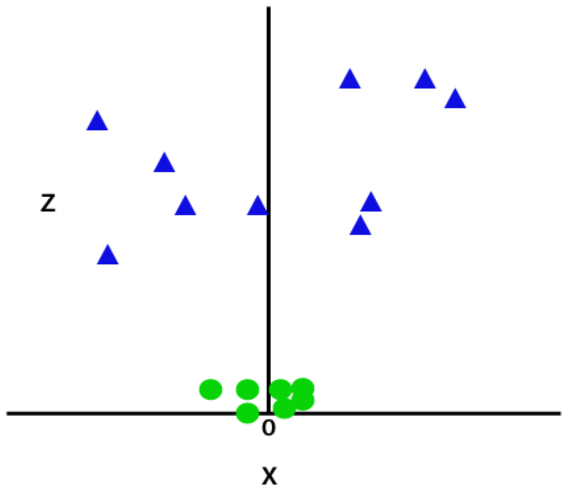


Figure 3.10: Non-Linear SVM data seperation

So now, SVM will divide the datasets into classes in the following way. Consider the below image:

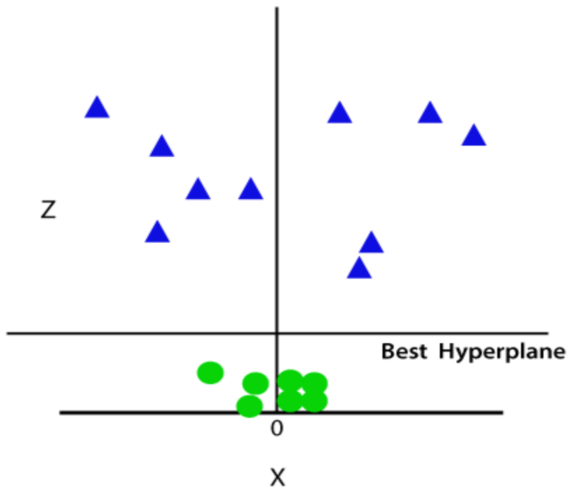


Figure 3.11: Non-Linear SVM best hyperplane

Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:

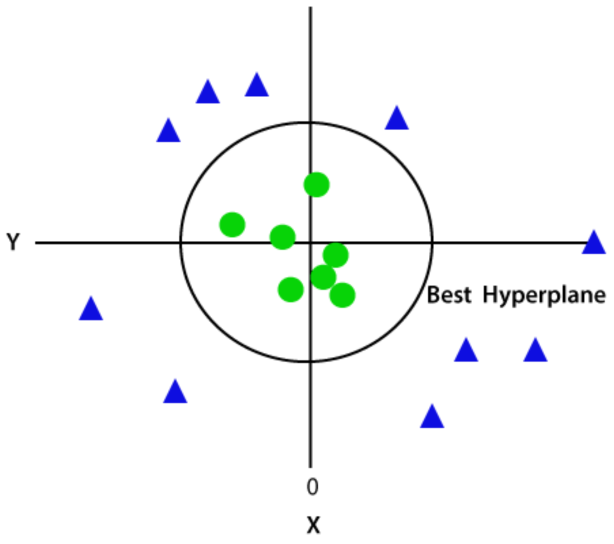


Figure 3.12: Non-Linear SVM with ROC

Hence, we get a circumference of radius 1 in case of non-linear data.

Disadvantages of support vector machine:

- Support vector machine algorithm is not acceptable for large data sets.
- It does not execute very well when the data set has more sound i.e. target classes are overlapping.
- In cases where the number of properties for each data point outstrips the number of training data specimens, the support vector machine will underperform.
- As the support vector classifier works by placing data points, above and below the classifying hyperplane there is no probabilistic clarification for the classification.

Roles of voting classifier

SVM brings its strength in handling complex decision boundaries to the ensemble. By focusing on maximizing the margin between classes, SVM contributes a unique perspective to the classification task. Its inclusion complements the capabilities of Random Forest and Decision Tree, resulting in a more comprehensive ensemble.

Decision tree classifier

A Decision Tree is constructed through a process called recursive partitioning, where the dataset is repeatedly split based on the most significant features. The key components of a Decision Tree include:

- Root Node:
  The initial node that represents the entire dataset.
- Internal Nodes:
  Nodes that represent decisions based on specific features.
- Branches:
  Connections between nodes that depict the outcomes of decisions.
- Leaf Nodes:
  Terminal nodes that represent the final predictions.

Decision Tree Learning Process

The learning process of a Decision Tree involves:

1. Feature Selection: Identifying the most informative feature to split the dataset.

2. Splitting: Dividing the dataset into subsets based on the selected feature.

3. Recursive Partitioning: Repeating the process for each subset until a stopping criterion is met.

4. Pruning (Optional): Trimming the tree to prevent overfitting.

Advantages of Decision Trees

Interpretability: Decision Trees are inherently interpretable, making them suitable for scenarios where understanding the decision-making process is crucial.

Handling non-linearity: They can model complex, non-linear relationships in the data.

Mixed Data Types: Decision Trees can handle both numerical and categorical data.

Feature Importance:They provide insights into feature importance, aiding in feature selection.

Role of Decision Tree in the Ensemble

Role in the Voting Classifier

The inclusion of Decision Trees in the ensemble provides interpretability and transparency. While less complex than Random Forest, Decision Trees can uncover specific patterns in the SDN traffic data that might be missed by other algorithms. This diversity contributes to a more nuanced understanding of the classification task.

Voting Classifier

The Voting Classifier combines the predictions of the RF, DT, and SVM models to make a final prediction. In this project, a hard voting approach is adopted, where each model contributes one vote, and the majority class becomes the ensemble's final prediction. This ensemble approach aims to harness the collective intelligence of the individual models to achieve a more accurate and robust classifier for SDN traffic.

Advantages of the Voting Classifier

Increased Accuracy: Combining predictions from different algorithms often results in a more accurate overall prediction.

Enhanced Robustness: The diversity among the models helps mitigate the impact of individual model weaknesses, leading to a more robust classifier.

Broad Applicability: The ensemble approach makes the classifier suitable for a wide range of SDN traffic scenarios by leveraging the strengths of each contributing algorithm

4.RESULTS

4.1 Results description

Figure 1 shows a representation of the GUI application developed for encrypted network traffic classification. The GUI include buttons, input fields, and visualizations, allowing users to interact with the application and perform tasks related to network traffic classification. Figure 2 depicts the GUI interface after loading the network traffic dataset. The dataset contains 141,530 rows and 85 columns. In the GUI, this dataset is displayed, providing users with an overview of the data they are working with. This view includes a table-like structure showing the sample dataset’s rows and columns.

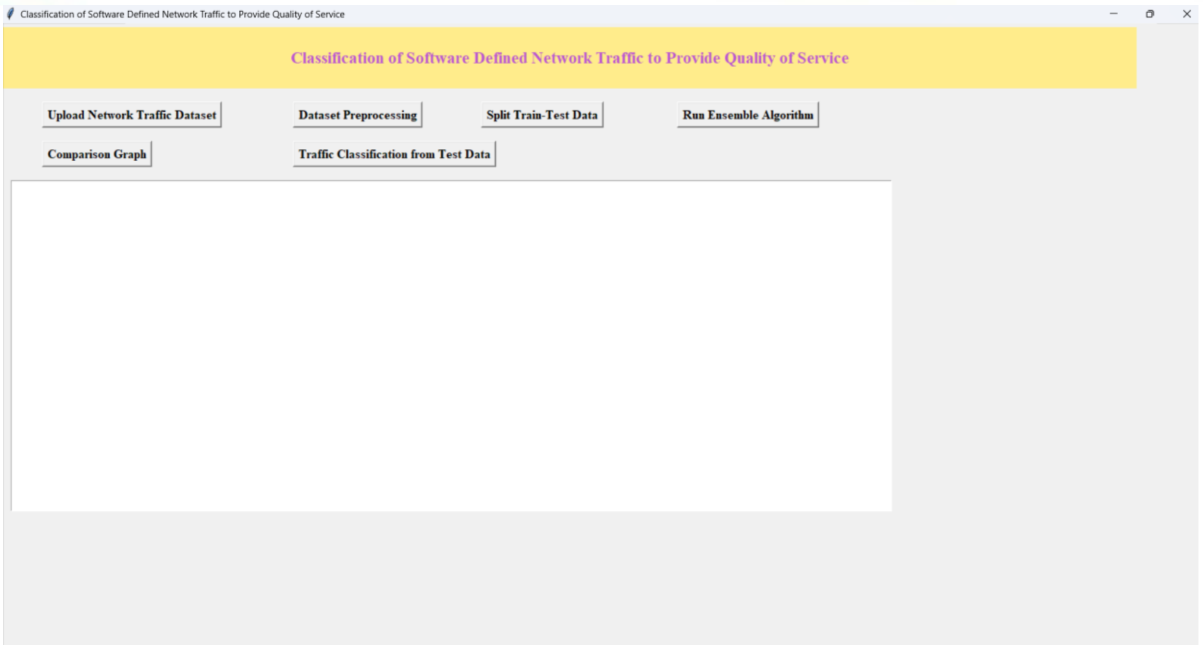
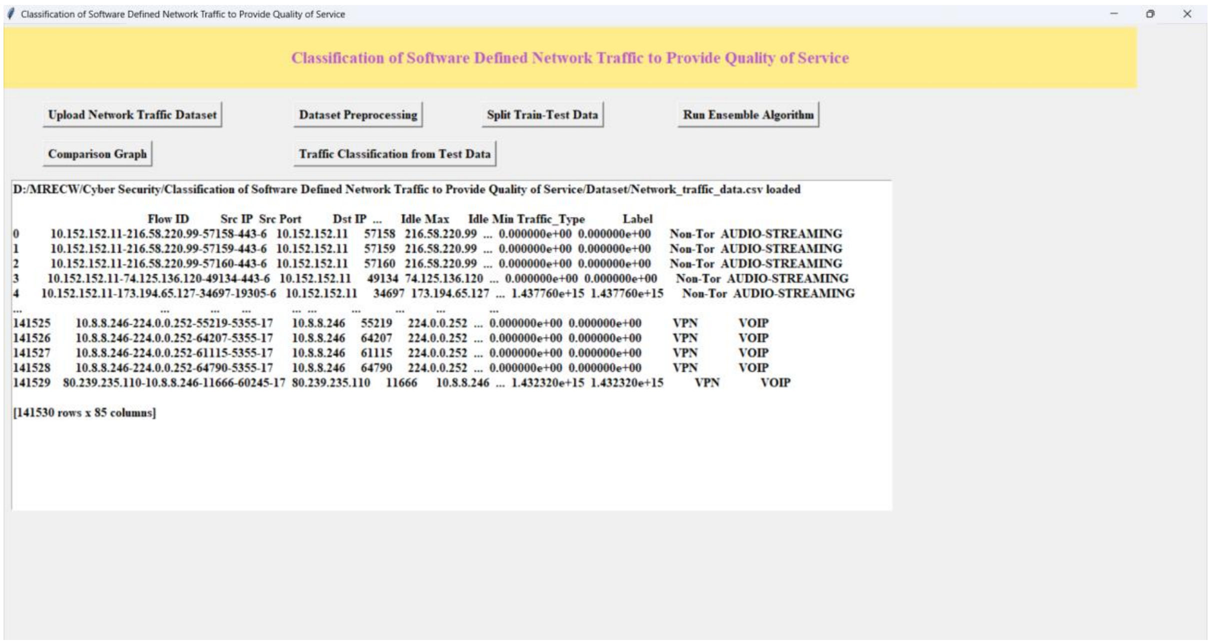

Classification of Software Defined Network Traffic to Provide Quality of Service
Classification of Software Defined Network Traffic to Provide Quality of Service
Upload Network Traffic Dataset
Dataset Preprocessing
Split Train-Test Data
Run Ensemble Algorithm
Comparison Graph
Traffic Classification from Test Data

Figure 1: GUI application of encrypted network traffic classification using Ensemble model.


Classification of Software Defined Network Traffic to Provide Quality of Service
Classification of Software Defined Network Traffic to Provide Quality of Service
Upload Network Traffic Dataset
Dataset Preprocessing
Split Train-Test Data
Run Ensemble Algorithm
Comparison Graph
Traffic Classification from Test Data

```
D:/MRECW/Cyber Security/Classification of Software Defined Network Traffic to Provide Quality of Service/Dataset/Network_traffic_data.csv loaded

                              Flow ID       Src IP  Src Port        Dst IP ...   Idle Max     Idle Min Traffic_Type            Label
0       10.152.152.11-216.58.220.99-57158-443-6  10.152.152.11     57158  216.58.220.99 ... 0.000000e+00 0.000000e+00      Non-Tor  AUDIO-STREAMING
1       10.152.152.11-216.58.220.99-57159-443-6  10.152.152.11     57159  216.58.220.99 ... 0.000000e+00 0.000000e+00      Non-Tor  AUDIO-STREAMING
2       10.152.152.11-216.58.220.99-57160-443-6  10.152.152.11     57160  216.58.220.99 ... 0.000000e+00 0.000000e+00      Non-Tor  AUDIO-STREAMING
3      10.152.152.11-74.125.136.120-49134-443-6  10.152.152.11     49134 74.125.136.120 ... 0.000000e+00 0.000000e+00      Non-Tor  AUDIO-STREAMING
4    10.152.152.11-173.194.65.127-34697-19305-6  10.152.152.11     34697 173.194.65.127 ... 1.437760e+15 1.437760e+15      Non-Tor  AUDIO-STREAMING
...                                         ...            ...       ...            ... ...          ...          ...          ...              ...
141525      10.8.8.246-224.0.0.252-55219-5355-17     10.8.8.246     55219    224.0.0.252 ... 0.000000e+00 0.000000e+00          VPN             VOIP
141526      10.8.8.246-224.0.0.252-64207-5355-17     10.8.8.246     64207    224.0.0.252 ... 0.000000e+00 0.000000e+00          VPN             VOIP
141527      10.8.8.246-224.0.0.252-61115-5355-17     10.8.8.246     61115    224.0.0.252 ... 0.000000e+00 0.000000e+00          VPN             VOIP
141528      10.8.8.246-224.0.0.252-64790-5355-17     10.8.8.246     64790    224.0.0.252 ... 0.000000e+00 0.000000e+00          VPN             VOIP
141529  80.239.235.110-10.8.8.246-11666-60245-17 80.239.235.110     11666     10.8.8.246 ... 1.432320e+15 1.432320e+15          VPN             VOIP

[141530 rows x 85 columns]
```

Figure 2: GUI application after loading the ISCX network traffic dataset of 141530 rows, and 85 columns.

Figure 3 represents a count plot, a type of bar chart, displaying the distribution of network traffic categories present in the loaded dataset. Each category is shown on the X-axis, and the corresponding count or frequency of each category is represented on the Y-axis. This visualization provides insights into the dataset's class distribution, which is crucial for understanding the balance or imbalance among different traffic categories.

Figure 4 shows the GUI interface after performing data preprocessing tasks such as normalization, label encoding, and standard scaling. These preprocessing steps are essential in preparing the dataset for training machine learning models. Normalization ensures that all features have a consistent scale, label encoding converts categorical data into numerical values, and standard scaling standardizes the feature values to have a mean of 0 and a standard deviation of 1.

Figure 5 displays the confusion matrix generated by evaluating the predictions of the Proposed Ensemble model. A confusion matrix is a table used to evaluate the performance of a classification algorithm. It shows the true positive, true negative, false positive, and false negative values, allowing a detailed analysis of the model's performance on each class.

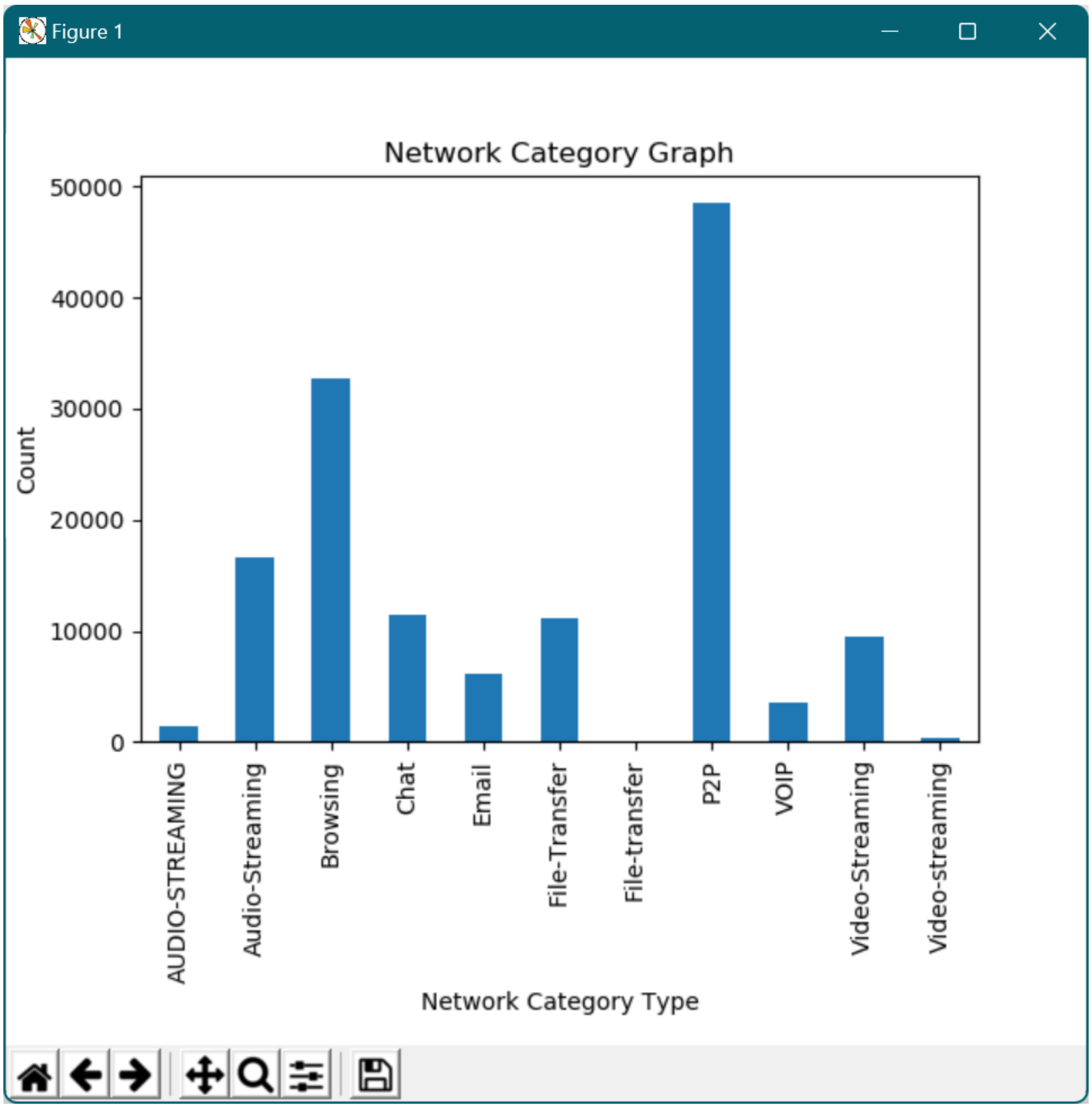
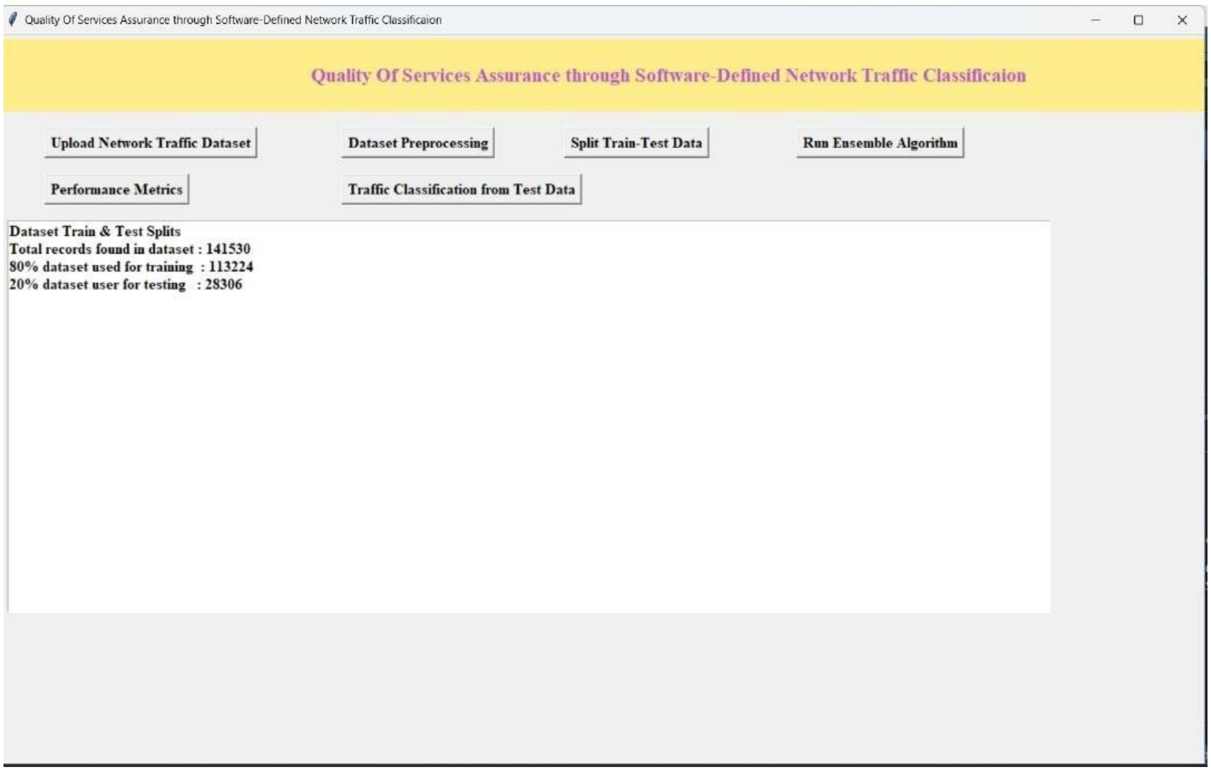
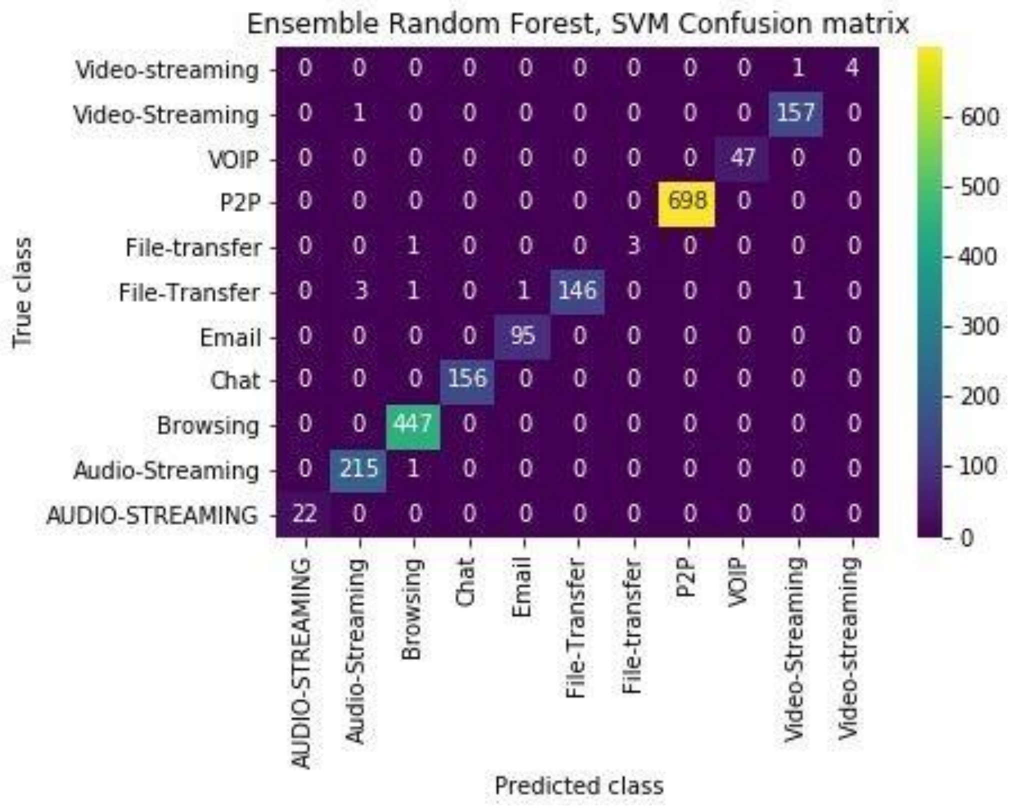


Figure 3: Count plot of network traffic category.


Quality Of Services Assurance through Software-Defined Network Traffic Classificaion
Quality Of Services Assurance through Software-Defined Network Traffic Classificaion
Upload Network Traffic Dataset
Dataset Preprocessing
Split Train-Test Data
Run Ensemble Algorithm
Performance Metrics
Traffic Classification from Test Data

```
Dataset Train & Test Splits
Total records found in dataset : 141530
80% dataset used for training  : 113224
20% dataset user for testing   : 28306
```

Figure 4: GUI application after performing data preprocessing, normalization, label encoding, and standard scaling operations.



True class \ Predicted class	AUDIO-STREAMING	Audio-Streaming	Browsing	Chat	Email	File-Transfer	File-transfer	P2P	VOIP	Video-Streaming	Video-streaming
Video-streaming	0	0	0	0	0	0	0	0	0	1	4
Video-Streaming	0	1	0	0	0	0	0	0	0	157	0
VOIP	0	0	0	0	0	0	0	0	47	0	0
P2P	0	0	0	0	0	0	0	698	0	0	0
File-transfer	0	0	1	0	0	0	3	0	0	0	0
File-Transfer	0	3	1	0	1	146	0	0	0	1	0
Email	0	0	0	0	95	0	0	0	0	0	0
Chat	0	0	0	156	0	0	0	0	0	0	0
Browsing	0	0	447	0	0	0	0	0	0	0	0
Audio-Streaming	0	215	1	0	0	0	0	0	0	0	0
AUDIO-STREAMING	22	0	0	0	0	0	0	0	0	0	0

Figure 5: Confusion matrix of proposed Ensemble Model.

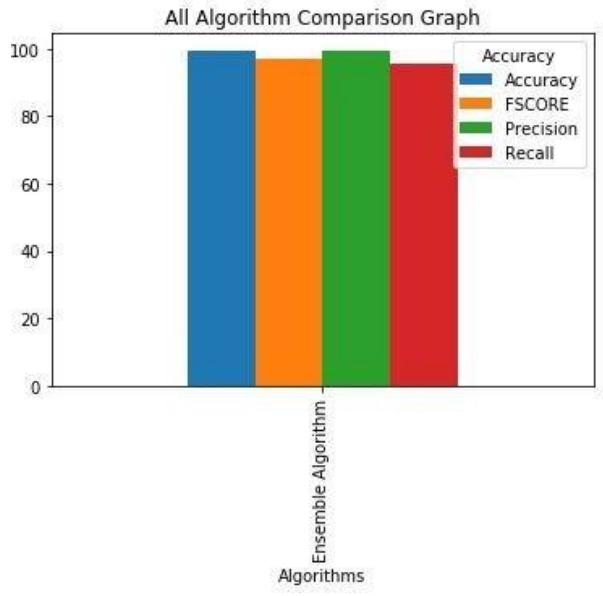
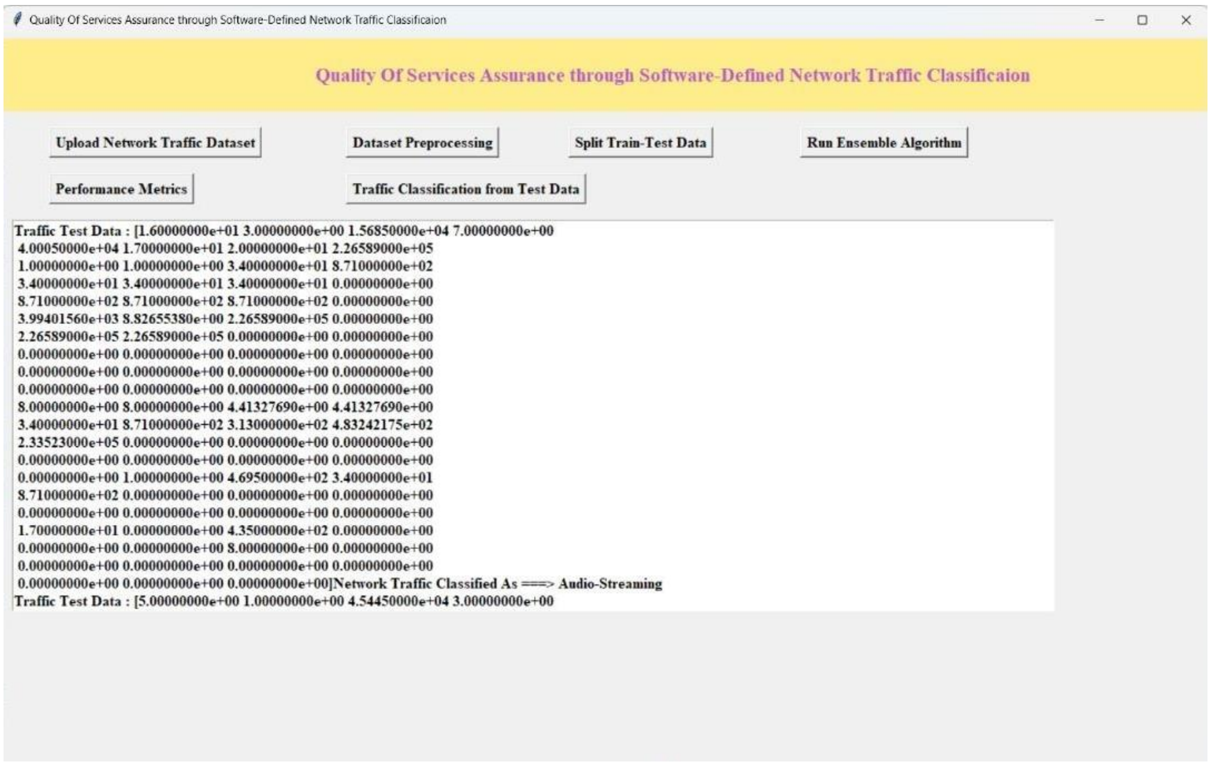


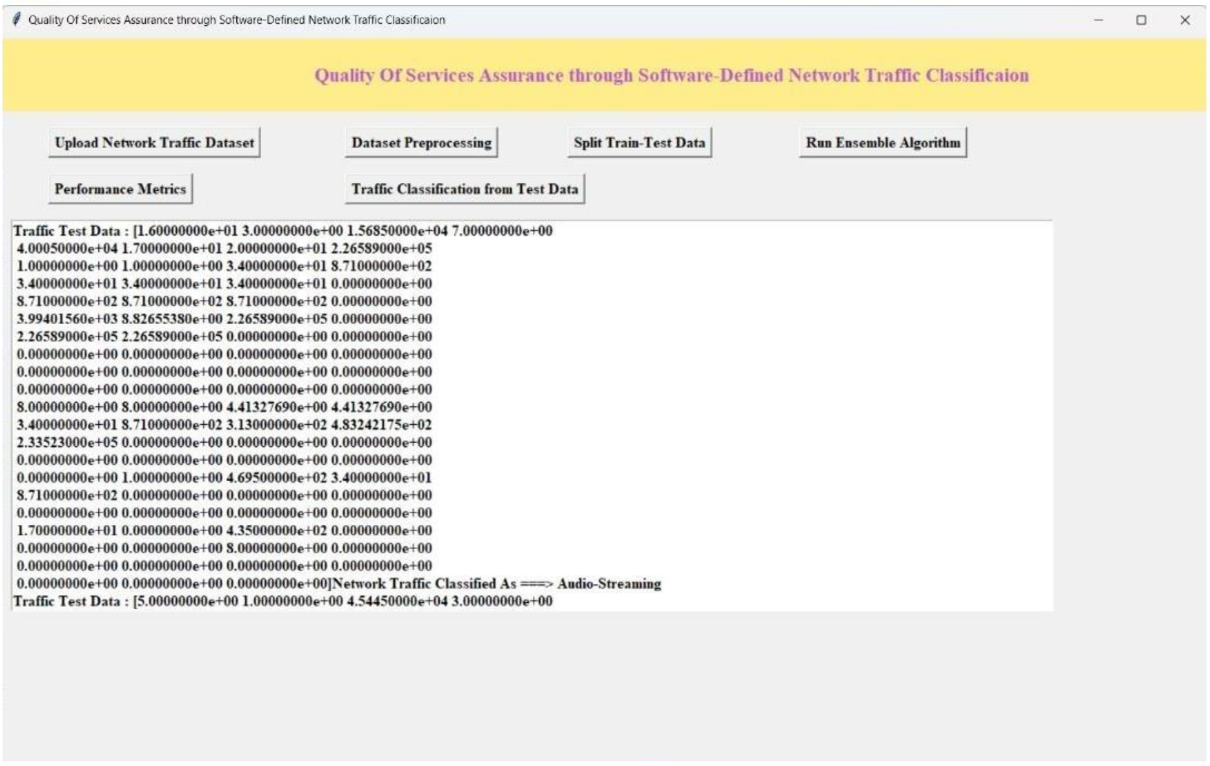
Figure 6: Performance comparison of obtained quality metrics using proposed ensemble models.

Figure 6 presents a visual comparison of the quality metrics obtained from the proposed Ensemble model. These metrics includes accuracy, precision, recall, and F1-score. A bar chart visualization method is used to clearly show the differences in performance between the model performs better in each metric.

Figure 7 shows the GUI interface displaying a sample of predicted traffic classification results on encrypted test data. It includes the input data or features and the corresponding predicted traffic category, allowing users to see how the trained model classifies specific instances of encrypted network traffic.


Quality Of Services Assurance through Software-Defined Network Traffic Classificaion
Quality Of Services Assurance through Software-Defined Network Traffic Classificaion
Upload Network Traffic Dataset
Dataset Preprocessing
Split Train-Test Data
Run Ensemble Algorithm
Performance Metrics
Traffic Classification from Test Data

```
Traffic Test Data : [1.60000000e+01 3.00000000e+00 1.56850000e+04 7.00000000e+00
 4.00050000e+04 1.70000000e+01 2.00000000e+01 2.26589000e+05
 1.00000000e+00 1.00000000e+00 3.40000000e+01 8.71000000e+02
 3.40000000e+01 3.40000000e+01 3.40000000e+01 0.00000000e+00
 8.71000000e+02 8.71000000e+02 8.71000000e+02 0.00000000e+00
 3.99401560e+03 8.82655380e+00 2.26589000e+05 0.00000000e+00
 2.26589000e+05 2.26589000e+05 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 8.00000000e+00 8.00000000e+00 4.41327690e+00 4.41327690e+00
 3.40000000e+01 8.71000000e+02 3.13000000e+02 4.83242175e+02
 2.33523000e+05 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 1.00000000e+00 4.69500000e+02 3.40000000e+01
 8.71000000e+02 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 1.70000000e+01 0.00000000e+00 4.35000000e+02 0.00000000e+00
 0.00000000e+00 0.00000000e+00 8.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00]Network Traffic Classified As ===> Audio-Streaming
Traffic Test Data : [5.00000000e+00 1.00000000e+00 4.54450000e+04 3.00000000e+00
```


Quality Of Services Assurance through Software-Defined Network Traffic Classificaion
Quality Of Services Assurance through Software-Defined Network Traffic Classificaion
Upload Network Traffic Dataset
Dataset Preprocessing
Split Train-Test Data
Run Ensemble Algorithm
Performance Metrics
Traffic Classification from Test Data

```
Traffic Test Data : [1.60000000e+01 3.00000000e+00 1.56850000e+04 7.00000000e+00
 4.00050000e+04 1.70000000e+01 2.00000000e+01 2.26589000e+05
 1.00000000e+00 1.00000000e+00 3.40000000e+01 8.71000000e+02
 3.40000000e+01 3.40000000e+01 3.40000000e+01 0.00000000e+00
 8.71000000e+02 8.71000000e+02 8.71000000e+02 0.00000000e+00
 3.99401560e+03 8.82655380e+00 2.26589000e+05 0.00000000e+00
 2.26589000e+05 2.26589000e+05 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 8.00000000e+00 8.00000000e+00 4.41327690e+00 4.41327690e+00
 3.40000000e+01 8.71000000e+02 3.13000000e+02 4.83242175e+02
 2.33523000e+05 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 1.00000000e+00 4.69500000e+02 3.40000000e+01
 8.71000000e+02 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 1.70000000e+01 0.00000000e+00 4.35000000e+02 0.00000000e+00
 0.00000000e+00 0.00000000e+00 8.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00]Network Traffic Classified As ===> Audio-Streaming
Traffic Test Data : [5.00000000e+00 1.00000000e+00 4.54450000e+04 3.00000000e+00
```

Figure 7: GUI application displaying the sample predicted traffic classification on encrypted test data.

5.CONCLUSION

In conclusion, the project on "Classification of Software Defined Network Traffic to Provide Quality of Service" addresses the critical need for effective Quality of Service (QoS) management in Software Defined Networking (SDN) environments. The abstract outlines the challenges faced by traditional network management systems in adapting to dynamic traffic conditions and emphasizes the significance of SDN in overcoming these challenges.The proposed project aims to revolutionize network management by introducing advanced traffic classification algorithms tailored for SDN. By separating the control plane from the data plane, SDN provides a more dynamic and programmable approach to network management. The project recognizes the essential role of QoS in ensuring reliable and predictable network performance. It underscores the limitations of traditional networks that often rely on fixed configurations and routing protocols, making them less adaptable to changing traffic patterns.The core contribution of the project lies in the development of algorithms capable of analyzing network packets and categorizing them based on their characteristics and QoS requirements. This dynamic classification system enables the implementation of specific QoS policies in real-time, allowing for the prioritization of critical applications, efficient resource management, and optimal network performance.

REFERENCES

[1] Flurry Analytics, ComScore, Pandora, Facebook, NetMarketShare. 2015. (accessed on 15 December 2021).

[2] Al-Mejibli, I.S.; Alharbe, N.R. Analyzing and evaluating the security standards in wireless network: A review study. *Iraqi J. Comput. Inform.* **2020**, *46*, 32–39.

[3] Taylor, V.F.; Spolaor, R.; Conti, M.; Martinovic, I. Robust smartphone app identification via encrypted network traffic analysis. *IEEE Trans. Inf. Forensics Secur.* **2017**, *13*, 63–78.

[4] Aceto, G.; Ciuonzo, D.; Montieri, A.; Pescape, A. Mobile Encrypted Traffic Classification Using Deep Learning. In Proceedings of the Network Traffic Measurement and Analysis Conference (TMA), Vienna, Austria, 26–29 June 2018; pp. 1–8.

[5] Wang, W.; Zhu, M.; Wang, J.; Zeng, X.; Yang, Z. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In Proceedings of the IEEE International Conference on Intelligence and Security Informatics (ISI), Beijing, China, 22–24 July 2017; pp. 43–48.

[6] Pathmaperuma, M.H.; Rahulamathavan, Y.; Dogan, S.; Kondoz, A.M. In-App Activity Recognition from Wi-Fi Encrypted Traffic. In *Intelligent Computing. SAI 2020*; Arai, K., Kapoor, S., Bhatia, R., Eds.; Advances in Intelligent Systems and Computing; Springer: Cham, Switzerland, 2020; Volume 1228.

[7] Zhang, F.; He, W.; Liu, X.; Bridges, P.G. Inferring users' online activities through traffic analysis. In Proceedings of the ACM Conference on Wireless Network Security, Hamburg, Germany, 14–17 June 2011.

[8] Conti, M.; Mancini, L.V.; Spolaor, R.; Verde, N.V. Analyzing Android encrypted network traffic to identify user actions. *IEEE Inf. Forensics Secur.* **2016**, *11*, 114–125.

[9] Park, K.; Kim, H. Encryption is not enough: Inferring user activities on KakaoTalk with traffic analysis. In *16th International Workshop on Information Security Applications*; Kim, H.-W., Choi, D., Eds.; ser. WISA '15; Springer: Berlin/Heidelberg, Germany, 2015; pp. 254–265.

[10] Saltaformaggio, B.; Choi, H.; Johnson, K.; Kwon, Y.; Zhang, Q.; Zhang, X.; Xu, D.; Qian, J. Eavesdropping on fine-grained user activities within smartphone apps over encrypted network traffic. In *WOOT'16 Proceedings of the 10th USENIX Conference on Offensive Technologies*; USENIX Association: Berkeley, CA, USA, 2016; pp. 69–78.

[11] Taylor, V.F.; Spolaor, R.; Conti, M.; Martinovic, I. AppScanner: Automatic Fingerprinting of Smartphone Apps from Encrypted Network Traffic. In Proceedings of the IEEE European Symposium on Security and Privacy (Euro S&P), Saarbruecken, Germany, 21–24 March 2016; pp. 439–454.

[12] Lotfollahi, M.; Zade, R.S.H.; Siavoshani, M.J.; Saberian, M. Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Comput.* **2020**, *24*, 1999–2012.

[13] Draper-Gil, G.; Lashkari, A.H.; Mamun, M.S.I.; Ghorbani, A. Characterization of encrypted and VPN traffic using time-related features. In Proceedings of the ICISSP, Rome, Italy, 19–21 February 2016; pp. 407–414.

[14] Lopez-Martin, M.; Carro, B.; Sanchez-Esguevillas, A.; Lloret, J. Net-work traffic classifier with convolutional and recurrent neural networks for internet of things. *IEEE Access* **2017**, *5*, 18042–18050.

[15] Wang, P.; Ye, F.; Chen, X.; Qian, Y. DataNet: Deep learning based en-crypted network traffic classification in SDN home gateway. *IEEE Access* **2018**, *6*, 55380–55391.