Approach to Improving Machine Learning Models for Intrusion Detection System

Bachelor of Science (B.Sc.)
in
Computer Science and Engineering (CSE)

By
Ahmad Ibtisam Labib
ID: C191117

Shamsuddin Ahmmed Chy
ID: C193064

Md. Shahriar Hossain
ID: C191112

TO
FACULTY OF SCIENCE AND ENGINEERING
INTERNATIONAL ISLAMIC UNIVERSITY CHITTAGONG
Spring 22

DECLARATION

We hereby affirm the following statements regarding our thesis:

1. The thesis has been successfully completed as part of our undergraduate degree program at International Islamic University Chittagong.
2. The thesis work does not contain any previously published or third-party content without proper citation.
3. The thesis work has not been previously submitted for any other degree or diploma at any other university or institution.
4. We have appropriately acknowledged all significant sources of contribution in the thesis.

Student's Full Name and Metric ID:

Ahmad Ibtisam Labib	(C191117)
Shamsuddin Ahmmed Chy	(C193064)
Md. Shahriar Hossain	(C191112)

SUPERVISOR'S DECLARATION

I formally state that I have examined this thesis and claim it to be of sufficient quality and scope to be granted for the undergraduate degree of Bachelor of Science in Computer Science and Engineering.

Sazid Zaman Khan

Assistant Professor

Department of Computer Science and Engineering

International Islamic University Chittagong

ACKNOWLEDGMENTS

Alhamdulillah, first and foremost, may all praises be to Allah, the Most Supreme and the Most Merciful, who allowed me to perform and accomplish my research work and leading to the fulfillment of a Bachelor of Computer Science degree.

Secondly, I would express my heartfelt appreciation, to Sazid Zaman Khan sir, for his inspiration, continuous guidance, insightful suggestions, and compassionate encouragement during this educational pursuit. He has always inspired me to be the highest possible version of myself, and I appreciate his expertise and selfless counsel.

I want to extend my gratitude to my parents, who provided constant vibrant support and prayed for me throughout my academic ambitions, and to all the respected faculties, my predecessors and my teammates at International Islamic University Chittagong who constantly supported me throughout my studies. Their blessings have been an inspiration and a source of strength for me.

Finally, I would want to offer my profound appreciation and respect to everyone who helped me in any way throughout the course of our research.

Abstract

In today's digital world, there are several security risks that digital assets must contend with. Systems for detecting intrusions (IDS) are essential security tools that protect digital assets. But their usefulness depends on meeting strict accuracy requirements, and their effectiveness depends on timely alarms. This study offers a novel IDS model that combines deep learning and machine learning methods as a solution to these problems. The study applies several classification techniques, such as Gaussian Naive Bayes (GNB), Random Forest (RF), Decision Tree, K-Nearest Neighbors (KNN), Soft Voting, and Hard Voting, using the well-known KDD Cup-1999 dataset. After a large-scale dataset was processed, the Decision Tree method performed better than the others, with a 99.9% accuracy rate. This study aims to investigate the effects of soft voting and hard voting, a novel application in IDS. Decision Tree proved to be the better performance in spite of these efforts. By offering information about algorithmic efficacy, the research advances the field of intrusion detection and helps decision-makers in the design and deployment of intrusion detection systems. These findings have implications for improving digital asset protection against changing cyber threats.

Keywords: Machine learning, IDS, KDD Cup, Security, DT, GNB, RF, KNN, Ensemble

Table of Contents

List of Figure

List of Tables

ABBREVIATION

The following list provides descriptions of various symbols and abbreviations that will be utilized in the subsequent sections of the document.

IDS: Intrusion Detection System

DT: Decision Tree

KNN: K-Nearest Neighbors

GNB: Gaussian Naive Bayes

RF: Random Forest

CHAPTER 1

Introduction

1.1 Overview

In today's time of digital advancements, security of computer networks and systems is of vast importance. An Intrusion Detection System (IDS) inspects the activities in a system for suspicious behavior or patterns that may indicate system attack or misuse. As the evident need of interconnected software and devices grows, it also brings many unprecedented inconveniences, exposing individuals, organizations to increasing cyber threats. Security of systems is evermore evolving but so is the process of cyber attacks, which urges us to update further and build more solid attack detection and prevention systems. This research aims at addressing the current Intrusion Detection System that uses modern ML Algorithms, to find a comparison of those existing IDS to an extent, and try to improve IDS's by trying to improve some algorithms such as ensemble learning to improve these algorithms potential to enhance detection process by continuously improving their accuracy and adaptability. We will explore the basics of intrusion detection and machine learning, and how they can work together. We will compare and try to improve machine learning models, gather and prepare the necessary data, and thoroughly test how well our Intrusion Detection System (IDS) works in both simulated and if possible to apply in real situations. By the end, we aim to give a better understanding of what existing IDS's can and cannot do and how we might be able to bring improvements. This study aims to provide a complete overview of the capabilities and limits of present IDS systems, as well as to investigate potential enhancements for increased Cyber Security measures using ML algorithm developments.

Our study presents several key contributions :

- Comparative analysis of Existing ML Models
- Approach to an Ensemble Algorithm
- Enhanced Accuracy

1.2 Motivation and Scope of Research

In the fast-changing world of network technology today, keeping networks safe is extremely important. The challenge lies in the myriad methods hackers use to infiltrate systems. Traditional intrusion detection systems (IDSs) have limitations, including poor detection of unknown attacks, high false positives, and resource consumption. Even popular IDS products share these issues. The emergence of Artificial Intelligence and Machine Learning, particularly Ensemble Learning, offers an opportunity to enhance intrusion detection. Our research is motivated by the need to address these limitations and strengthen network security. By integrating ML algorithm based intrusion detection, we aim to develop more effective and adaptable systems that can safeguard critical networks across various sectors totally focused on changing world of Cyber Security. In this thesis, we hope to untangle the complexities of intrusion detection using cutting-edge machine learning methods. The study focuses on analyzing and comparing the performance of various algorithms, offering insight into their strengths and limitations in the context of digital asset protection. Such as Firewalls, Access Control Policies, Incident Response Plans, Network Segmentation.

This Study is more than just an academic undertaking; it is a mission to give valuable insights to the ever-changing world of cyber security. By going into this comparative research, we got to know few features of intrusion detection system using ML and how they work and found some result and conclusion about several things which can up bring such as Accuracy improvement, Data set featuring and leveling betterment using ML algorithm etc . To pave the road for novel advances that will strengthen intrusion detection systems. The study's multidisciplinary character raised its importance in tackling real-world problems, making it an important inquiry at the crossroads of between machine learning and cyber security.

1.3 Research Objective

The research objective of this study are as follows :

1. **Compare among existing IDS :** Study and analysis of existing IDS systems in work and finding out their lacks and places of improvement.

2. **Improve Accuracy :** Enhance the accuracy of proposed model in comparison with the existing models for better performance in detecting attack types.

1.4 Problem Statement

The persistent threat of Distributed Denial of Service (DDoS) attacks on online services highlights the need for robust intrusion detection systems (IDSs). Traditional IDSs struggle with limitations like suboptimal detection of new attack patterns and a propensity for generating false alarms. In the realm of cyber security, the integration of machine learning (ML) and data mining (DM) offers promise for bolstering intrusion detection. However, determining the most suitable ML/DM method remains a complex task. We summarize our problem statement to detect network intrusion detections using the machine learning algorithm over KDD Cup'99 [1] data set. We first describe the data set used in this experiment and then discuss the results obtained. Finally, we evaluate our approach and compare the results with the results obtained by other researchers using different types of algorithms and with the best result of the KDD 99 contest. We tried to find accuracy problems and besides we also found missing of proper Data set featuring and leveling betterment.

1.5 Contribution of the Thesis

Our main focus of contribution on this project are as follows :

1. **Comparative analysis of Existing ML Models:**

   In our study we tried to work with a few more ML algorithms that we have found in several research papers we have attached in the reference section. We have trained 6 type of ML algorithm such as Decision Tree , Random Forest, K-nearest neighbors algorithm , Gaussian Naive Bayes, Ensemble algorithm and a comparative analysis of various models outcomes.

2. **Approach an Ensemble Algorithm:** We tried various algorithms and applied the best ensemble model to show a higher accurate model that existing ones.

3. **Enhanced Accuracy :** We achieved better accuracy than other ensemble model used in the reference we used.

1.6 Organization of the Thesis:

The paper is organized as follows:

- Chapter 1, Discusses our sources of inspiration and motivation, the problems we face, and the objectives we set for ourselves.
- Chapter 2, an examination of all pertinent works in the field that relate to our objectives.
- Chapter 3, study of various ways to improve the methodology and proposal of our methodology process.
- Chapter 4, presents the results and offers a thorough explanation of the experimental study.
- Chapter 5, Describes the conclusion, limitations and future goal based on our study.

Chapter 2

Literature Review

2.1 Introduction

As advancements grow on networking day by day due to exponential growth of use of online systems and networking technologies, landscape of cyber threats are also ever-evolving. Our day to day life is covered by lot of types of networks such as Wi-Fi networks, campus networks, business/company networks, wireless local area networks. Yet as more people utilize wireless networks, the issue of network security is getting more and more critical. Many wide spread networks such as local wi-fi, campus networks, business networks continuously process vast amount of network traffic therefore having the high risk of being vulnerable to unidentified network attacks. Wireless networks are the ones most insecure and vulnerable. So, need of technology for detecting attacks and intrusions is also crucially important for assuring the safety of users data and network assets. That's where the machine learning model approach to such intrusion detection plays a vital role that can be built to monitor real time network traffic and detect such intrusions. Many researches have been done implementing various data mining and machine learning models to use on IDSs to better improve the detection models through several machine learning techniques and there remains many aspect of improvement that can be done on the field.

2.2 Analysis of Literature Review

Almomani et al. [2] presented an intrusion detection system (IDS) that leverages stacking ensemble learning, combining Random Forest, Decision Tree, and k-Nearest-Neighbors algorithms. They evaluated their system on the UNSW-NB15 dataset, comparing its performance to existing IDSs. This research highlights the potential of machine learning and ensemble methods for network security. While the proposed IDS achieved impressive accuracy 97.94% using Random Forest, 76.43% using Naïve Bayes, 96.74% using Decision Tree. Future research could focus on pushing the limits of detection performance, particularly for rare attack types. Also improving interpretability making the system's decision-making process more transparent and understandable.

Anna L et al [3] , The ML/DM methods are described, as well as several applications of each method to cyber intrusion detection problems. There are three main types of cyber analytics in support of IDSs: misuse-based (sometimes also called signature based), anomaly-based, and hybrid. These studies did not actually build an IDS, but examined the performances of ML and DM methods on some cyber security data. In this paper referenced dataset that they

studied was the KDD Cup 1999. Emphasis on study of different types of DM and ML methods for Intrusion Detection System was prioritized on their work whilst defining key factors such as the importance of the data sets for training and testing the systems.

Derya Erhan et al [4], proposed the concept of Network-based Intrusion Detection Systems by detecting attacks from networks end, on the router/the backbone switch. With the two set of database called : TCP SYN Flood and UDP flood datasets, which include attack rates of 1000, 1500, 2000, and 2500 packets/second, respectively on one victim IP Address. They used hping3 software to generate attack packets with randomly generated spoofed source IP addresses. Wireshark software running on a server running with Windows processing system was used to record the traffic. Payloads of packets were deleted, A-class virtual IP addresses replaced source IP addresses.

Jiyeon Kim et al [5], proposed a model based on a Convolutional Neural Network (CNN) and performed binary classification and multiclass classification using the CNN-based model and finally, they evaluated its performance by comparing to a model based on a Recurrent Neural Network (RNN). They tested their models on two different sets of data: one called “KDD-99” and the other called “CIC-IDS-2018”. When they compared the CNN model to the RNN model, they found that the CNN model did better in the task with two choices (binary classification). But in the task with multiple choices (multiclass classification), the RNN model did really well on some types of attacks, getting a perfect score of 100% accuracy, while it showed very poor accuracy on other types of attacks.

Himadri Chauhan et al [6], stated a comparison of classification technique for IDSs, to show which classification technique performs best in respect of accuracy, specificity and computation time. They stated that, IDS technique can be classified as two types: anomaly detection and misuse detection. Focus should not only be the accuracy of the IDS, but also, it’s extensibility and adaptability capacity are also critical in today’s network environments. Using a portion of the original KDD Cup’99 dataset, named NSL-KDD, they applied supervised algorithms such as: DT, RF, NB. Their results showed that DT classifiers are best at classifying intrusions, although RF is the best performant algorithm in respect of showing highest accuracy as near as 99%. Their proposed further improvement was such as, combining different DM algorithms to improve performance which we seek to carry out in our work.

Sydney M. Kasongo and Yanxia Sun et al [7], said about how ML based IDS’s have emerged as the leading systems in the intrusion detection research domain. Machine learning gives systems the ability of learning and improving by using previous data. In their experiment, they applied Supervised Learning methods such as: Decision Tree, Random Forest, KNN, SVM and Logistic Regression on the UNSW NB15 dataset. In their work, they applied XGBoost algorithm for feature selection in conjunction with multiple machine learning techniques. Furthermore, the XGBoost based attribute selection method was applied over the UNSW-NB15 and as a result, 19 optimal features were selected. The experimental results demonstrated that using a reduced (optimal) feature vector has its merits in terms of reducing the models complexity as well as increasing the detection accuracy on test data resulting in better final accuracy.

Mrutyunjaya Panda et al [8], proposed that a solution to ever-growing intrusion attack on any types of networks is through the use of network intrusion detection systems (NIDS), that detect attacks by observing various network activities. Therefore it always faces a cruciality that whether such systems are accurate in identifying attacks, quick to train and generate as few false positives as possible. So they proposed a framework of NIDS based on Naïve Bayes algorithm, as it works through a classification technique making a hypothesis that the whole dataset belongs to a single class and this is amongst the most practical approach for certain types of problems. The technique lets a accessibility of increasing/decreasing training data and also the correct probability of the hypothesis depending on that too. The framework builds the patterns of the network services over data sets labelled by the services. With the built patterns, the framework detects attacks in the datasets using the Naïve Bayes Classifier algorithm. Compared to the Neural Network based approach, their approach achieves higher detection rate, less time consuming and has low cost factor.

Fuat Turkn et al [] states, advancement in technology has led us to use of vastly wide network system in all aspect of our economic, institutional and military work. Thus, the risk of such networks getting attacked rises also. They proposed that, since various networks have various level of dataset, best way to detect intrusion is through feature selection. It can be done using machine learning algos, filtering methods etc. So, their proposed strategy to apply and improve intrusion detection system was to apply ML and DL methods to specifically feature selection wide ranges of data and classifying them into two types as: binary and multiclass. MLP method from DL was used in their work.

Raihan Al Masud et al [10] states, to overcome existing limitation of IDS's ensemble is a well thought approach and can be highly utilized if gradually perfected. They applied hard voting ensemble algorithm alongside other mainstream algorithms such as DT, KNN, SVC. Specifying the attack types in class, their proposed ensemble model was able to show good performance but the struggle was their limitation on showing good performance on all types of attack type classified.

Chapter 3

Methodology

3.1 Introduction

In our research, we formulated the problem by analyzing the topic, conducting a literature review to identify research gaps, and comparing existing works. Utilizing references from previous papers, we sourced a dataset from Kaggle, a renowned platform. Data preprocessing involved filtering and reduction to enhance dataset quality. We explored six ML algorithms, including Decision Tree, Random Forest, KNN, Naive Bayes, and Voting Ensemble. Implementation was carried out using Python and scikit-learn library. Performance evaluation, including accuracy, precision, F1 score, and recall, was conducted through confusion matrices. Comparative analysis with existing IDS systems revealed improved performance in selected ML models on our collected dataset. Steps of our methodology described below

Problem Formulation: To analyze the topic and form the problem on a structure. Then we studied the base papers for our analysis study by completing the literature review. That led us to find the research gaps that we could explore to work on and also compare the existing works.

Data Collection and Pre-processing: We used the previous papers as a reference for sourcing the dataset. Finally we were able to select the specific dataset from the "Kaggle" website, used as most renowned website for collecting different types of datasets.

Collecting the dataset, we performed data pre-processing to convert the dataset into a more meaningful data. We applied filtering to fill out null values, used reduction to eliminate data.

Applying ML Algorithms: We studied 6 algorithmic approaches and applied them on our dataset. Among our set of algorithms of choices we had : Decision Tree, Random Forest, KNN, Naive Bayes, Voting Ensemble applying also the few of said machine learning techniques together as ensemble learning.

Implementation: We implemented some ML approaches with the help of python programming language and sci-kit lbirary in ML. For evaluating the performance, we generated confusion matrices in order to calculate the accuracy, precision, f1 score and recall analysis.

Performance Analysis: As for our performance analysis, we made some analysis of existing IDS systems using various machine learning models comparing key factors such as desired accuracy, F1-score, confusion matrix etc. Then after implementing ML models of our choice on the collected dataset, we found some models to be of better performance than previous works using them.

3.2 Schematic Diagram of The Proposed Model

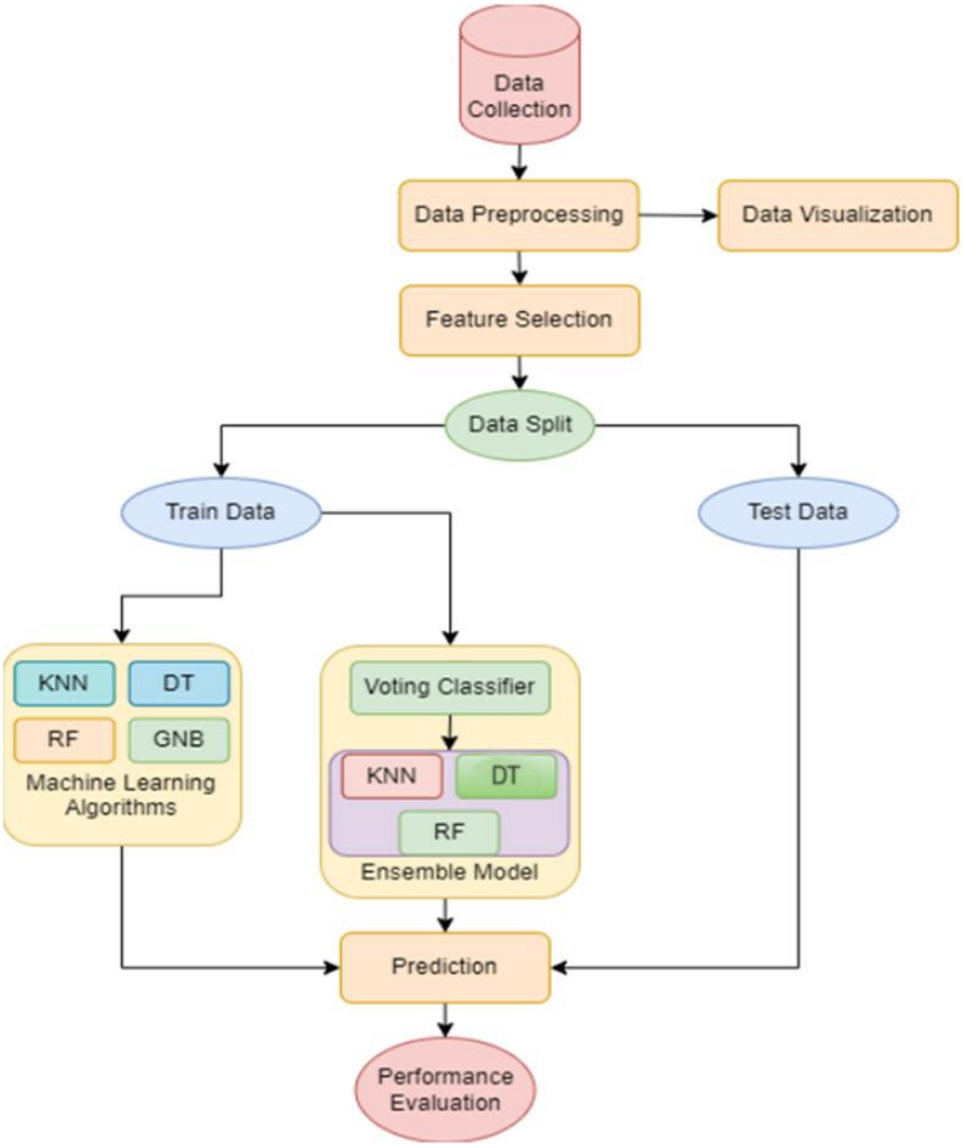
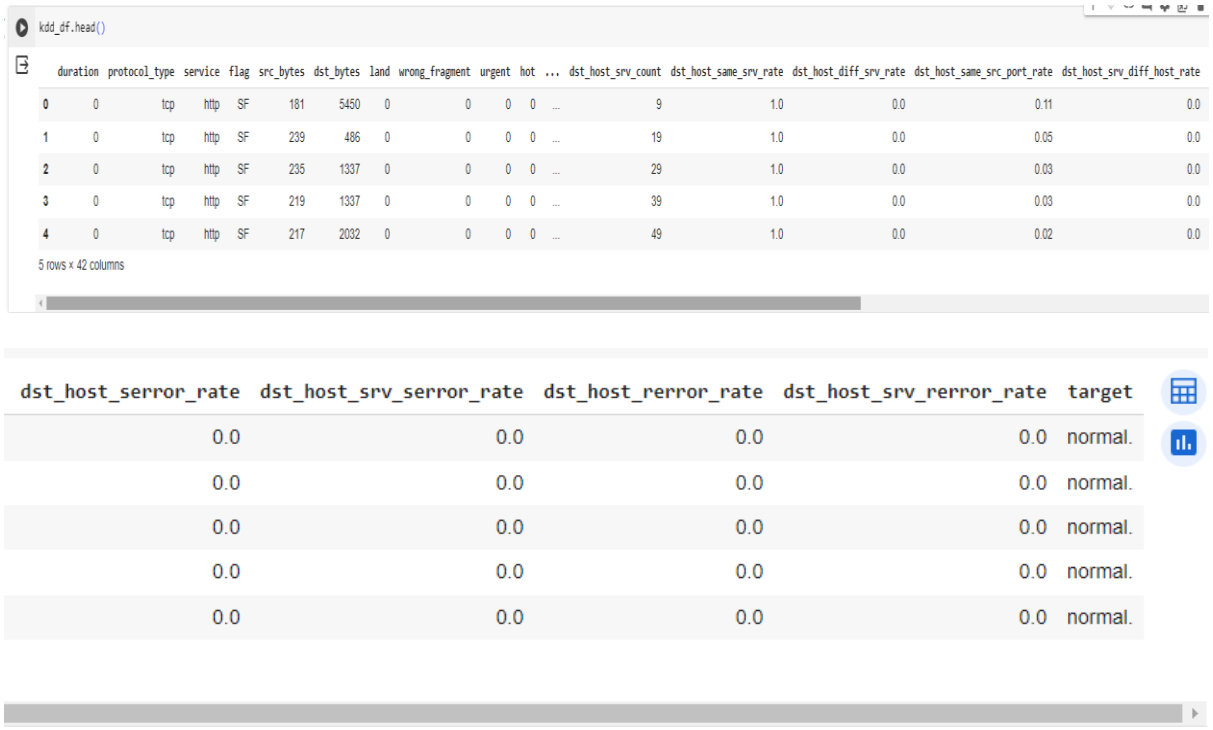


Figure 1 *Diagram of Proposed Model*

In this diagram, we showed precise steps of our study where we first studied the dataset and worked on the dataset. We took 70% data for test, 30% for training, Then choosing various algorithms, we trained those models using our pre-processed dataset and checked each models' performance. Also we applied our proposed novel ensemble model where we combined three best algorithms that showed best singular output.

3.3 Data Collection

For training our models we wanted to use dataset with emriched data containing various types of attack types and so we took the reference papers sample data named “KDD Cup 1999” [10] from the Kaggle website, known as a rich dataset collection platform. “KDD 99” dataset contains 42 feature sets altogether both attack and non-attack types. This dataset was originated from the KDD Cup challenge held by DARPA in 1999.



kdd_df.head()

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst_host_same_srv_rate	dst_host_diff_srv_rate	dst_host_same_src_port_rate	dst_host_srv_diff_host_rate
0	0	tcp	http	SF	181	5450	0	0	0	0	...	9	1.0	0.0	0.11	0.0
1	0	tcp	http	SF	239	486	0	0	0	0	...	19	1.0	0.0	0.05	0.0
2	0	tcp	http	SF	235	1337	0	0	0	0	...	29	1.0	0.0	0.03	0.0
3	0	tcp	http	SF	219	1337	0	0	0	0	...	39	1.0	0.0	0.03	0.0
4	0	tcp	http	SF	217	2032	0	0	0	0	...	49	1.0	0.0	0.02	0.0

5 rows × 42 columns

dst_host_serror_rate	dst_host_srv_serror_rate	dst_host_rerror_rate	dst_host_srv_rerror_rate	target
0.0	0.0	0.0	0.0	normal.
0.0	0.0	0.0	0.0	normal.
0.0	0.0	0.0	0.0	normal.
0.0	0.0	0.0	0.0	normal.
0.0	0.0	0.0	0.0	normal.

Figure 2 *Sample of KDD 00 Dataset*

3.3.1 Data Pre Processing

In the pre processing part, we specified the “Attack_type” column to adjust to visualize with the other featured columns of the dataset.

Scaling and Reducing Data :

As we worked with a dataset containing around 500K+ data, we applied some scaling measures to scale the data into a better and more upgraded state by checking for null values, duplicate rows etc and removing them.

```
[14] kdd_df.duplicated().sum() # Count the number of duplicate rows

     348435

[15] kdd_df.drop_duplicates(inplace=True) # Remove duplicate rows

[16] kdd_df.shape

     (145586, 43)
```

Figure 3 *Scaled Dataset*

Featuring : As for using an enriched data, we applied pre-processing on the dataset to make it more eligible for testing and training by eliminating null values, duplicate rows. Then we applied feature mapping based on prioritized features of the data such as network features - “tcp” “udp”, “icmp” etc. We also featured the data based on “logged_in” and “logged_out” user features. Finally, we sorted the random similar attack types to some classical types and took 5 major attack types as classification for the training data.

PCA & SMOTE : Using PCA (Principal Component Analysis) we converted high dimensional data to low dimensional data by selecting the most important features that capture maximum information about the dataset. The features are selected on the basis of variance that they cause in the output. The feature that causes highest variance is the first principal component.
Using SMOTE technique, we did data balancing after featuring the most important ones so that the training data gives better result all over.

3.4 Algorithm Selection

Our presented set of algorithms that we believe is able to bring out more improvements on our work are such as:

3.4.1 Decision Tree

The decision tree algorithm shines in intrusion detection systems (IDS) by effectively analyzing network traffic data and classifying it as normal or malicious. We can see it as a branching flowchart where each branch is based on a specific feature of the traffic, like packet size or source IP or the type of protocol used. The algorithm traverses this tree, asking questions about the data at each node, until it reaches a leaf node labeled as either "intrusion" or "normal traffic."

The decision tree learns these rules by analyzing a labeled dataset of network traffic, effectively building a knowledge base of malicious patterns. Additionally, their ability to handle high-dimensional data and adapt to new attack patterns makes them a valuable tool for IDS.

3.4.2 Random Forest

Random forest works by implying the method of grouping system. Bagging algorithm is the foundation of this algorithm. Unlike its single-decision-tree rivals, the random forest constructs an ensemble of diverse trees, each trained on a subset of the data and a randomly chosen subset of features. This collective wisdom imbues the forest with several desirable strengths. Its robustness against overfitting stems from the inherent averaging effect of the multitude of trees, while its ability to handle complex, non-linear relationships in network traffic data surpasses the limitations of individual decision trees. Furthermore, the random forest thrives on large datasets, its accuracy scaling gracefully with the volume of information. This efficiency shines in IDS domains, where vast and intricate network logs hold the key to identifying subtle attack patterns.

3.4.3 KNN

Cybersecurity is like a maze, full of changing dangers. IDS are like guards who must constantly adapt to stop new threats. Among the intricate footwork of algorithms, the K-Nearest Neighbors (KNN) steps forward with a unique blend of grace and efficiency.

Unlike a rigid rulebook, KNN's strength lies in its improvisational nature. Imagine a bustling network ballroom, where each data point represents a network event. When a new event waltzes in, KNN doesn't rely on pre-defined steps. Instead, it identifies the k closest

neighbors amongst the known "normal" and "intrusion" data points, drawing insights from their movements.

This nimble approach has several advantages:

- **Adaptability:** New attack patterns, like unexpected dips and twirls, are readily identified by comparing them to the ever-evolving landscape of neighbors. KNN can waltz with the unknown, unlike rule-based systems that stumble on unforeseen steps.
- **Minimal Training:** Unlike complex algorithms requiring extensive training data, KNN only needs a handful of examples to get the rhythm going. This makes it particularly graceful in resource-constrained environments, where every step count.
- **Interpretability:** By observing the neighbors, security analysts gain valuable insights into the attacker's moves. This allows them to adjust their own defensive posture and anticipate future variations in the dance.

However, mastering the KNN requires finesse:

- **Finding the Right Partner (k):** Choosing the optimal number of neighbors (k) is crucial. Too few, and the dance loses nuance; too many, and the steps become muddled. Finding the perfect k is like striking a balance between fluidity and precision.
- **Efficient Search:** As the network data grows, identifying the nearest neighbors can become a computationally expensive endeavor. Choosing the right search algorithm is like selecting the perfect dance partner – someone who can keep up without tripping over the ever-increasing complexity.

Despite these challenges, KNN remains a valuable tool in the IDS arsenal. Its adaptability, efficiency, and interpretability make it a graceful partner in the complex filed of cybersecurity. By combining its strengths with other algorithms and advanced feature selection techniques, we hope KNN can help defenders stay ahead of the ever-evolving threats, ensuring the network's security waltz never misses a beat.

3.4.4 Naive Bayes

The probabilistic classification algorithm known as Gaussian Naive Bayes (GNB) is based on the Bayes theorem. It makes the assumption that features have a Gaussian distribution and are conditionally independent given the class. GNB is computationally efficient despite its simplicity because it works effectively in scenarios when the independence assumption is met. GNB can be used to model the distribution of attributes associated with network activity in the context of intrusion detection, which can help identify possible threats based on statistical patterns.

Although some inconveniences faced by Naive Bayes are such as：

- Naive Bayes classifier beats other classifiers in regard of the true value of independent predictors.

- In case of calculating test results, Naive Bayes requires modest quantity of training data, Due to that, the training of dataset is shorter using this classifier.

3.5 Proposed Algorithm

3.5.1 Voting Ensemble

Voting ensembles can work as a powerful technique for improving the performance of machine learning models for intrusion detection systems. Voting ensembles leverage the predictions of multiple, diverse models. Each individual model within the ensemble is trained on the same dataset. When a new data point arrives, each model makes its own prediction about whether it represents normal or malicious traffic. The final verdict is determined by aggregating the individual predictions through a "voting" process.

Voting Mechanisms:

- **Hard Voting:** The most common prediction among the individual models wins. This is simple and efficient, but it doesn't take into account the confidence levels of each model.
- **Soft Voting:** The predictions of each model are weighted based on their past performance or confidence scores. This can lead to more accurate results, especially when the individual models have varying strengths and weaknesses.

3.5.1.1 Soft Voting

Soft Voting is an ensemble method that combines the predicted probabilities from multiple classifiers to make a final decision. This method allows for more sophisticated decision-making because it takes into account the confidence levels of individual classifiers.

Pros:

i. Soft Voting is useful in intrusion detection for handling confusing or unclear cases.

ii. Soft Voting adds to a strong and flexible intrusion detection system that can handle a different levels of confidence in each classifier's predictions by taking into account the degree of certainty of each prediction.

Cons:

i. Soft voting relies on probability estimates from individual models, and if these estimates are poorly calibrated, it might affect the ensemble performance.

ii. Assigning appropriate weights to each model's prediction in soft voting can be challenging and may require fine-tuning.

3.5.1.2 Hard Voting

Hard voting, a cornerstone of ensemble learning, excels in simplicity and efficiency. Hard Voting aggregates the majority vote from several classifiers to arrive at a final judgment. With this simple method, the class with the highest number of votes from each classifier is chosen.

Pros:

i. Hard voting works best in situations when there is a distinct line separating the classes and when a straightforward democratic procedure of decision-making is adequate. When it comes to intrusion detection, Hard Voting is a practical option in situations where there are clear patterns of malicious and regular network behavior since it offers a dependable way of classifying data based on the most commonly anticipated class.

Cons:

i. Limited Insight into Individual Model Performance:

Hard voting treats all predictions equally, regardless of the confidence or past performance of the individual models. This can mask potential issues with specific models that are consistently making errors. Without understanding which models are struggling, it becomes difficult to diagnose and address the root cause of errors within the ensemble.

ii. Computational Overhead:

Training and running multiple models can be computationally expensive, especially for complex models or large datasets. This can be a limitation for resource-constrained environments where real-time performance is crucial.

3.6 Summary

First we worked on data, First, we worked on out gathered dataset named “KDD Cup 99” which we applied data pre-processing and featured importance the dataset to make it better for our work.
Secondly, we selected various ML models for our test purpose and applied them on our featured dataset to see the results. Lastly, as a unique approach of ML, we proposed ensemble learning [9] also referenced by our base paper and we trained the model with our featured dataset and also analyzed the similar existing works and their differences.

Chapter 4

Result and Discussions

4.1 Introduction

In this section, we present the findings of our study comparing with existing IDS models and our proposed model's performance across our dataset named "KDD Cup 99". The focus is on understanding how well our proposed model detects different types of attack data. We'll walk you through the experiments, detailing the methods we used and the measurements we considered—like accuracy, precision, recall, and F1-score—to assess our model's effectiveness. Our goal is to provide a clear picture of how our proposed model performs. These results are not just for us; they offer insights for anyone considering the practical use of our proposed model. So, let's delve into the outcomes of our analysis and see what the data reveals.

4.2 Experimental Setup

We collected the "KDD Cup 99" dataset from Kaggle. The file contains intrusion attack types in CSV format. We imported the dataset into Google Co-laboratory and pre-processed it according to our need and featured the dataset to make the dataset enriched using data featuring methods. We split the data in test and train portion while also showing a visualization of our dataset. Then we applied various algorithms and trained them with our data to see how well they perform based on data classification we provided.

4.2.1 Integrated Development Environment (IDE)

In this research, we chose Google Co-laboratory and Python for intrusion detection which provides a convenient and accessible approach. Google Co-laboratory offers a cloud-based platform with free GPU support, making it efficient for training machine learning models. Python, a versatile and widely used programming language, along with libraries like SciKit, provides the necessary tools for developing and implementing fire detection algorithms. This combination offers a practical and powerful solution for machine learning tasks related to intrusion detection.

4.2.2 Dataset

A machine learning model needs to be trained on any given dataset before it can give out any prediction model as we desire from it. So, for that purpose, we collect the dataset and then train the model with that and finally test for desired prediction. In our work, we've trained 70% percent of the data and kept 30% for the prediction test purpose.

We used python programming language and its libraries such as Python sci-kit package. Using these, we developed several machine learning techniques such as Decision Tree, Random Forest, K-Nearest Neighbor, Ensemble Learning, Soft Voting Ensemble, Hard Voting Ensemble and others.

Pre-processing

In the pre processing part, we specified the "Attack_type" column to adjust to visualize with the other featured columns of the dataset.

As we worked with a dataset containing around 500K+ data, we applied some scaling measures to scale the data into a better and more upgraded state by checking for null values, duplicate rows etc and removing them.

Featuring

Various methodologies and equations have been followed to determine the appropriate and accurate training dataset. In our main dataset which is named "KDD Cup - 99", we featured five most known attack types as featured columns which are - "normal", "DoS", "Probe", "U2R" and "R2L". The last column we kept as whether it was "normal traffic" or "attack traffic" to show us whether there was attack detected or not.

Data Scaling and Cleaning

We scaled the data containing string value such as "tcp", "udp", "icmp" etc in one classification of numerical value of 0 or 1 indicating "normal" or "attack" respectively. Then we counted them as one class of data altogether. We set the margin of highest 1 for all kinds of attack types as in 100 percent attack type has been detected.

4.3 Evaluation Matrix

The Confusion Matrix is an important tool for evaluating the performance models, offering a thorough assessment beyond accuracy by incorporating essential metrics like precision, recall, and F1-score. Comprising four key values—True Positive, True Negative, False Positive, and False Negative—the matrix provides a detailed breakdown of a model's classification outcomes, making more understanding of its effectiveness in handling different aspects of the data.

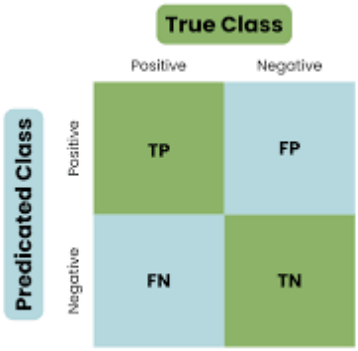


Figure 1 *Confusion Matrix*

True Positive (TP) refers to instances that the model correctly identifies as positive.

True Negative (TN) is when the model correctly identifies instances as negative.

False Positive (FP) occurs when the model incorrectly identifies instances as positive.

False Negative (FN) happens when the model incorrectly identifies instances as negative.

Evaluation Metrics:

Accuracy:

Accuracy represents the percentage of accurate predictions and is calculated as:

Accuracy Score = (TP + TN)/ (TP + FN + TN + FP)

Precision:

Precision is the proportion of positive cases that the model correctly predicts among all the cases it predicted as positive:

Precision Score = True Positives/ (False Positives + True Positives)

Recall (Sensitivity):

Recall indicates the percentage of positive class instances out of all positive cases found by the model:

Recall Score = True Positives / (False Negatives + True Positives)

F1 Score:

The F1 Score is a metric that balances precision and recall, calculated as the harmonic mean of precision and recall:

F1 Score = 2* Precision Score * Recall Score/ (Precision Score + Recall Score)

It is crucial to consider the F1-score alongside Accuracy, especially when False Positive and False Negative have varying costs. Accuracy may perform well when the costs are comparable, but F1-score offers a balanced assessment when these costs differ.

4.4 Experimental Result and Analysis

In previous chapters, we introduced our suggested approach, which required receiving and cleaning data for our system. Then we displayed the final dataset. We proceeded to use several techniques to find out the Training and Testing Accuracy. We will now describe the results of these various algorithms.

In this study, we have used 6 different algorithms to get better training and testing accuracy on the performance of our predictive models. Specifically, we have examined each algorithm's accuracy, precision, recall, and f-measure to determine how well our model performed. Ultimately, we have sought to identify the most accurate algorithm for expected outcomes in this study.

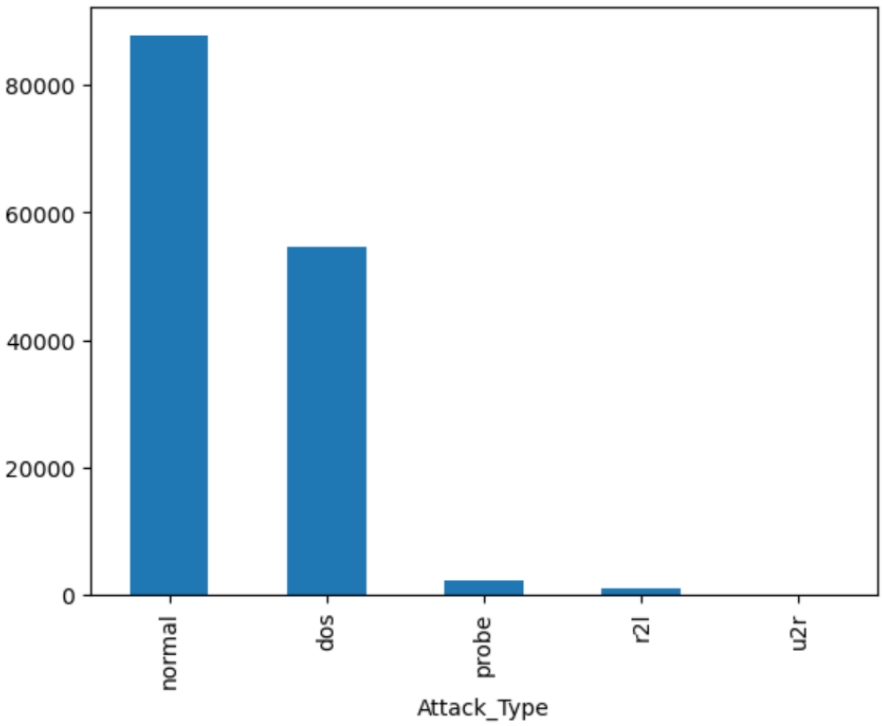


***Figure** 2 Attack Type Classification*

We have taken 5 types of different attach Such as Dos Attack, Normal Attack, Probe Attack, Remote-to-Local (r2l), User-to-root(u2r) to check the accuracy which we have defined as a 5 class, where we have successfully detected DOS Attack and Normal in maximum approach then, we have detected probe attack and r2l and u2r respectively using 6 types of ML algorithm.

4.4.1 Confusion Matrix for Machine Learning Algorithms

The confusion matrix offers a clear and detailed picture of a model's performance. It helps to understand how models predict actual instances right or wrong, like bias towards certain classes. It has four sections: true positive, true negative, false positive, false negative.

4.4.1.1 Decision Tree Classifier

The confusion matrix for the Decision Tree Classifier is shown in Figure 6 the matrix demonstrated that the model accurately predicted 10889 samples to be in Class 0, 17586 samples to be in Class 1, and 427 samples to be in Class 2, 162 samples to be in Class 3, 6 samples to be in Class 4.

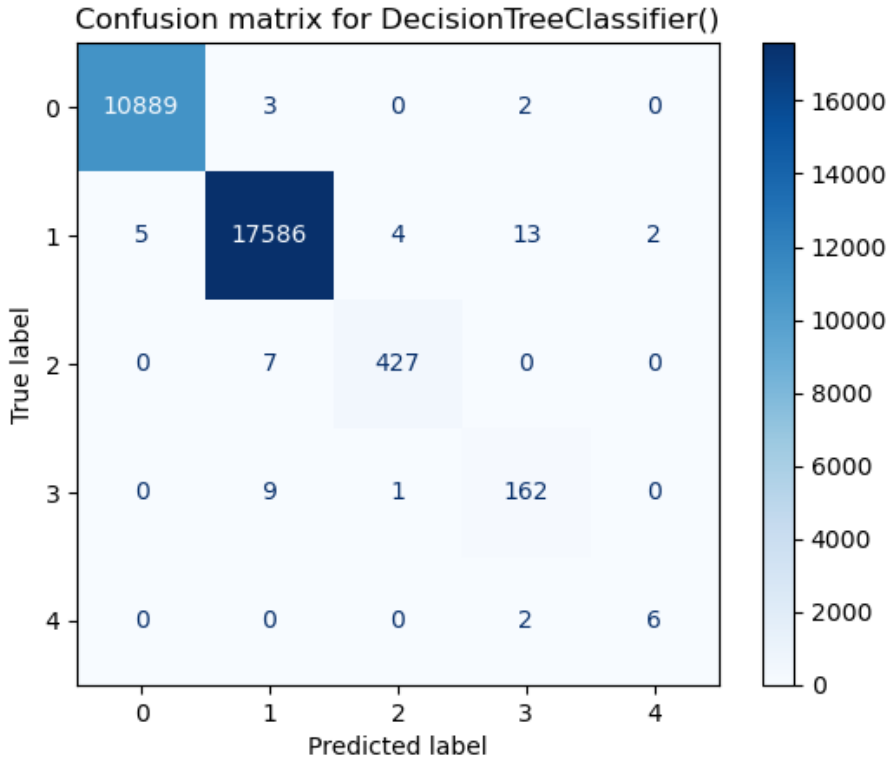


Figure 3 *Decision Tree Confusion Matrix*

However, misclassifications were observed, with certain samples incorrectly predicted as belonging to other categories. 3 samples of Class 0 were misclassified as Class 1 and 0 is as Class 2, 2 samples of Class 0 were misclassified as Class 3 and on the other hand 0 samples of Class 4 were misclassified as Class 4. Same as in class 1, number of sample 5, 4,13,2 were misclassified as respectively Class 0,2,3,4. In class 2, the number of samples 0,7,0,0 were misclassified as respectively Class 0,1,3,4. In class 3, the number of samples 0, 9,1,0 were misclassified as respectively Class 0,1,2,4. In class 4, numbers of samples 0,0,0,2 were misclassified as respectively Class 0,1,2,3

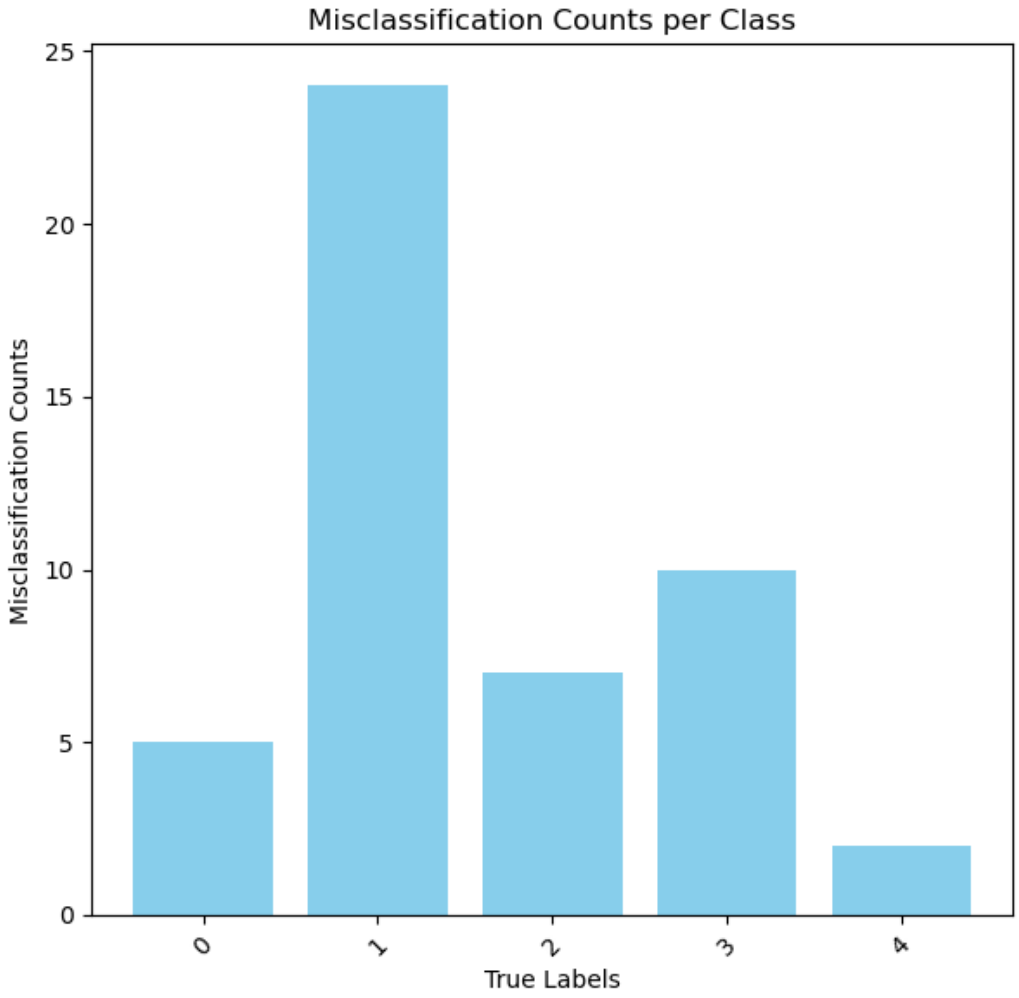


Figure 4 *Decision Tree Misclassification*

With the help of figure 7 we can say in left from Misclassification Counts per class where true labels are mentioned as a 0,1,2,3,4 are respectively Normal, Dos, probe, r2l, u2r ,where we can see class 0 has 5 misclassified from 10894 samples, class 1 has 24 misclassified from 17610 sample, class 2 has 7 misclassified from 434 samples, class 3 has 10 misclassified from 172 samples, class 4 has 2 misclassified from 8 samples .

4.4.1.2 Gaussian NB Classifier

The confusion matrix for the Gaussian NB Classifier is shown in Figure 8 The matrix demonstrated that the model accurately predicted 10674 samples to be in Class 0, 7331 samples to be in Class 1, and 18 samples to be in Class 2, 6 samples to be in Class 3, 6 samples to be in Class 4.

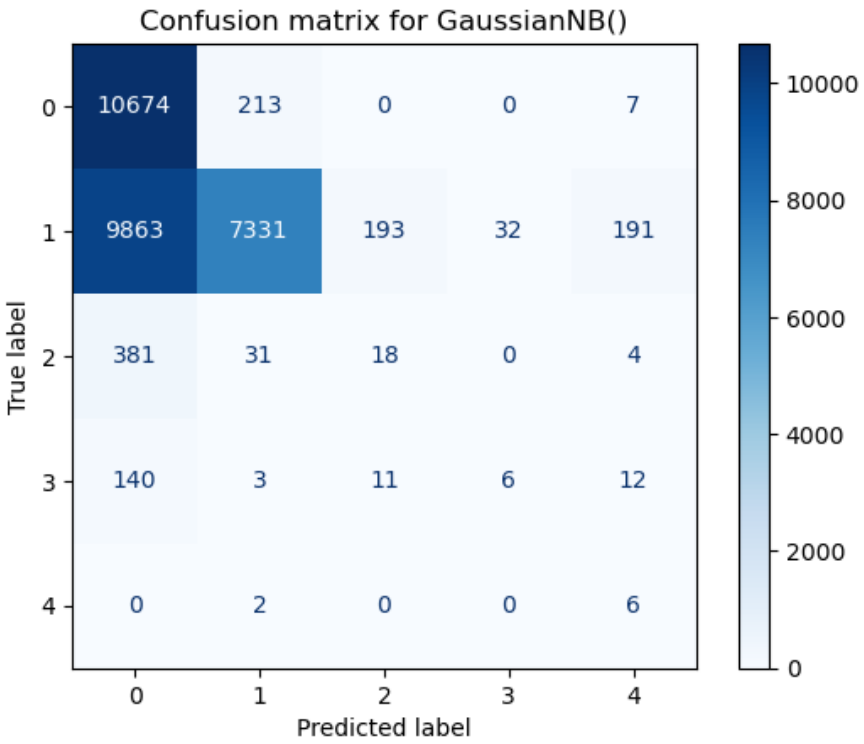


Figure 5 *Naive Bayes Confusion Matrix*

However, misclassifications were observed, with certain samples incorrectly predicted as belonging to other categories. 213 samples of Class 0 were misclassified as Class 1 and 0 is as Class 2 again 0 and 7 samples of Class 0 were misclassified respectively as Class 3,4. Same as in class 1, number of samples 9863, 193, 32, 191 were misclassified as respectively Class 0,2,3,4. In class 2, the number of samples 381, 31, 0,4 were misclassified as respectively Class 0,1,3,4. In class 3, the number of samples 140, 3,11,12 were misclassified as respectively Class 0,1,2,4. In class 4, the number of samples 0,2,0,0 were misclassified as respectively Class 0, 1 , 2 ,3.

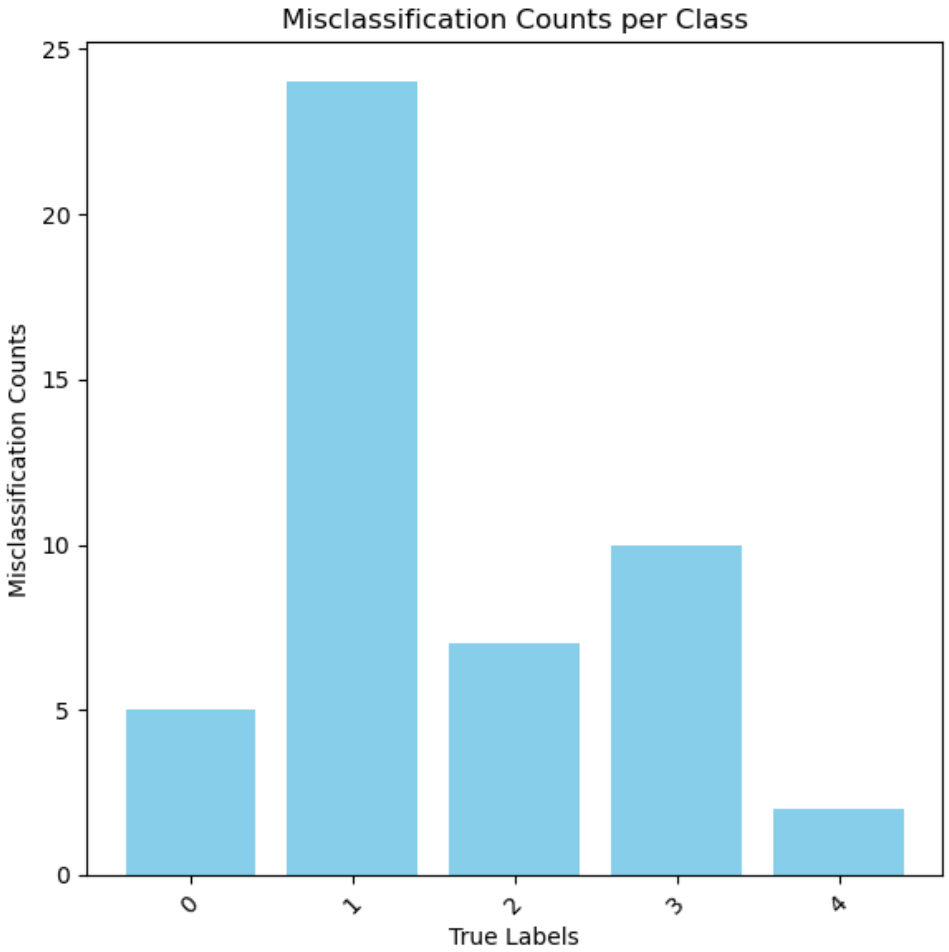


Figure 6 *Naive Bayes Misclassification*

With the help of figure 9 we can say in left from Misclassification Counts per class where true labels are mentioned as a 0,1,2,3,4 are respectively Normal, Dos, probe, r2l, u2r ,where we can see class 0 has 220 misclassified from 10894 samples, class 1 has 10279 misclassified from 17610 sample, class 2 has 353 misclassified from 434 samples, class 3 has 166 misclassified from 172 samples, class 4 has 2 misclassified from 8 samples .

4.4.1.3 Random Forest Classifier

The confusion matrix for the Random Forest Classifier is shown in Figure 10 the matrix demonstrated that the model accurately predicted 10734 samples to be in Class 0, 17605 samples to be in Class 1, and 147 samples to be in Class 2, 0 samples to be in Class 3, 0 samples to be in Class 4.

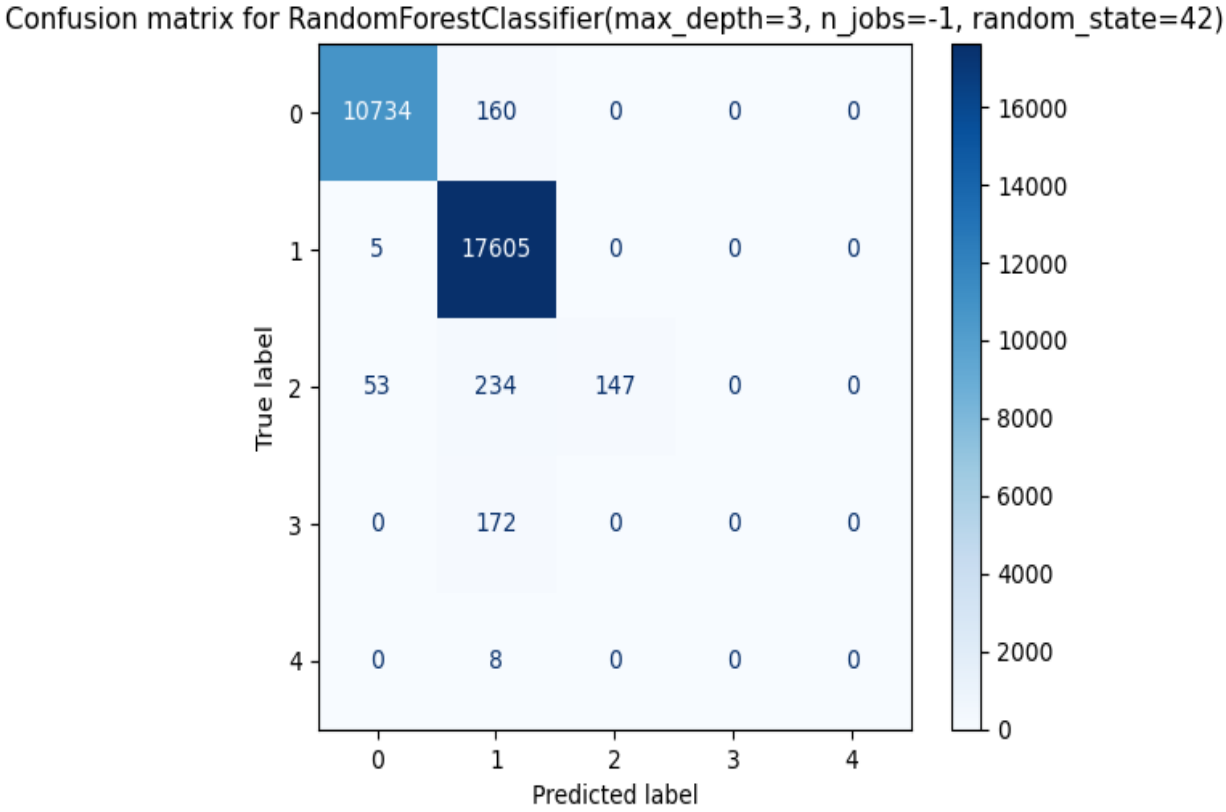


Figure 7 *Random Forest Confusion Matrix*

However, misclassifications were observed, with certain samples incorrectly predicted as belonging to other categories.160 samples of Class 0 were misclassified as Class 1 and 0 samples were as Class 2, 0 samples of Class 0 were misclassified as Class 3 same as for class 4. Same as in class 1, number of samples 5, 0, 0, 0 were misclassified as respectively Class 0,2,3,4. In class 2, number of sample 53, 234, 0, 0 were misclassified as respectively Class 0,1,3 ,4 In class 3, number of sample 0,172,0,0 were misclassified as respectively Class 0,1,2,4. In class 4, the number of samples 0, 8, 0, 0 were misclassified as respectively Class 0,1,2,3.

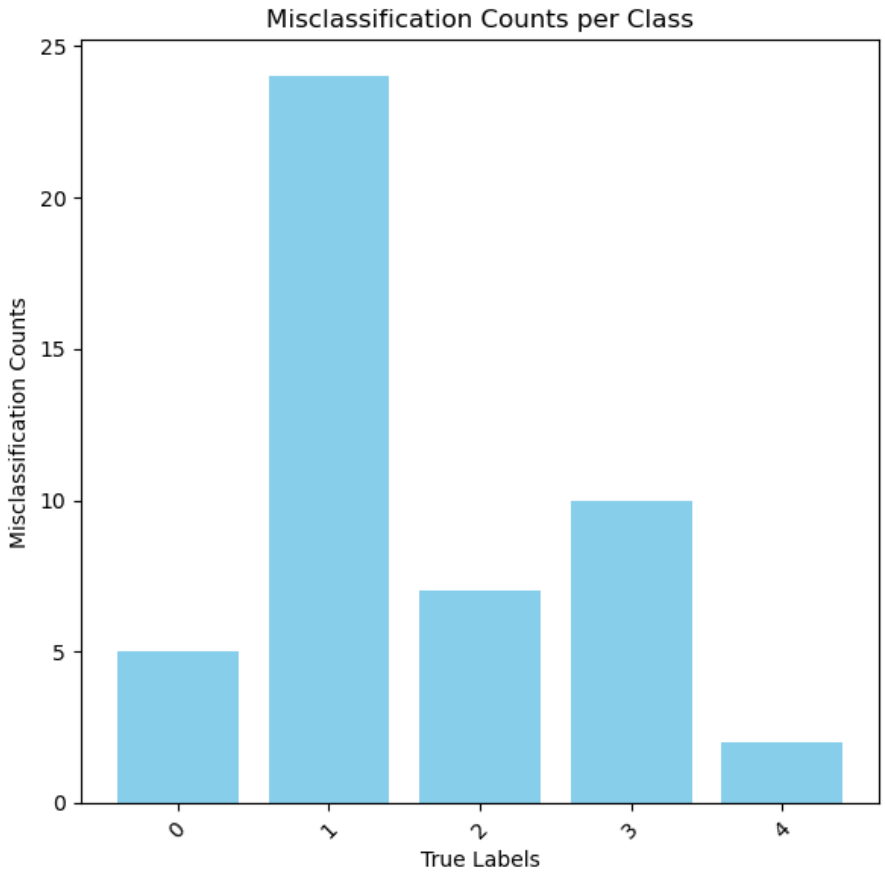


Figure 8 *Random Forest Misclassification*

With the help of figure 11 we can say in left from Misclassification Counts per class where true labels are mentioned as a 0,1,2,3,4 are respectively Normal, Dos, probe, r2l, u2r ,where we can see class 0 has 160 misclassified from 10894 samples, class 1 has 5 misclassified from 17610 sample, class 2 has 287 misclassified from 434 samples, class 3 has 172 misclassified from 172 samples, class 4 has 8 misclassified from 8 samples.

4.4.1.4 Confusion matrix for KN Neighbors Classifier

The confusion matrix for the K-N Neighbors Classifier is shown in Figure 12 The matrix demonstrated that the model accurately predicted 10880 samples to be in Class 0, 17583 samples to be in Class 1, and 379 samples to be in Class 2, 166 samples to be in Class 3, 3 samples to be in Class 4.

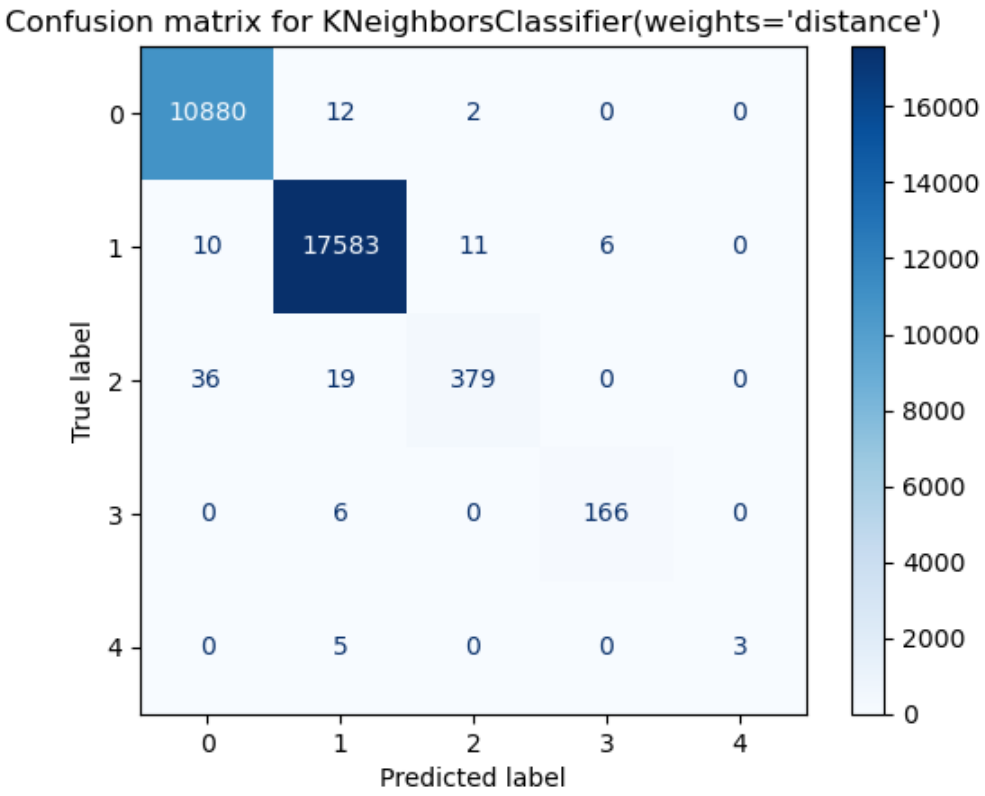


Figure 9 *KNN Confusion Matrix*

However, misclassifications were observed, with certain samples incorrectly predicted as belonging to other categories. 12 samples of Class 0 were misclassified as Class 1 next 2 samples of Class 0 were misclassified as Class 2 and the rest of the samples are 0 which are misclassified as respectively class 3,4. Same as in class 1, the number of samples 10, 11, 6 and 0 were misclassified as respectively Class 0, 2, 3 and 4. In Class 2, the number of samples 36, 19, 0, 0 were misclassified as respectively Class 0, 1, 3, 4. In Class 3, the number of samples 0, 6, 0 and 0 were misclassified as respectively Class 0, 1, 2, 4. In Class 4, the number of samples 0, 5, 0 and 0 was misclassified as respectively Class 0, 1, 2,3.

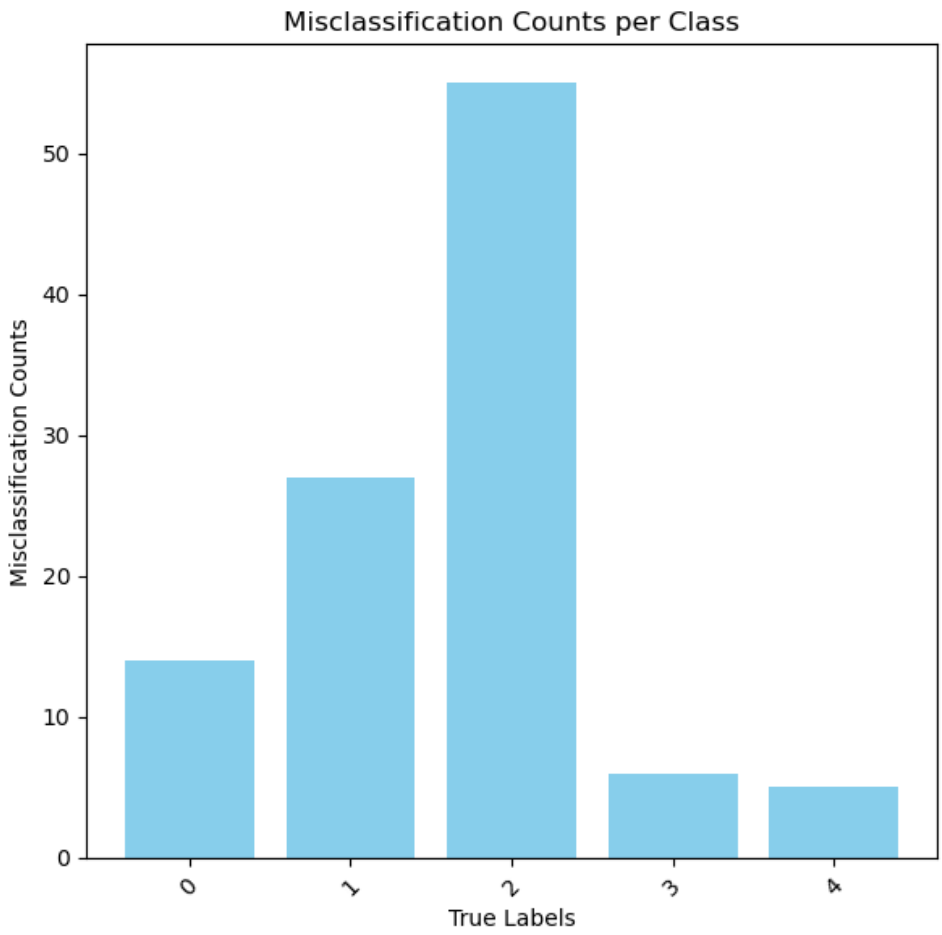


Figure 10 *KNN Misclassification*

With the help of figure 13 we can say in left from Misclassification Counts per class where true labels are mentioned as a 0,1,2,3,4 are respectively Normal, Dos, probe, r2l, u2r ,where we can see class 0 has 14 misclassified from 10894 samples, class 1 has 27 misclassified from 17610 sample, class 2 has 55 misclassified from 434 samples, class 3 has 6 misclassified from 172 samples, class 4 has 5 misclassified from 8 samples.

4.4.1.5 Confusion matrix for Soft Voting Classifier

The confusion matrix for the Soft Voting Classifier is shown in Figure 14 The matrix demonstrated that the model accurately predicted 10886 samples to be in Class 0. 17605 samples to be in Class 1 and 402 samples to be in Class 2, 161 samples to be in Class 3, 3 samples to be in Class 4.

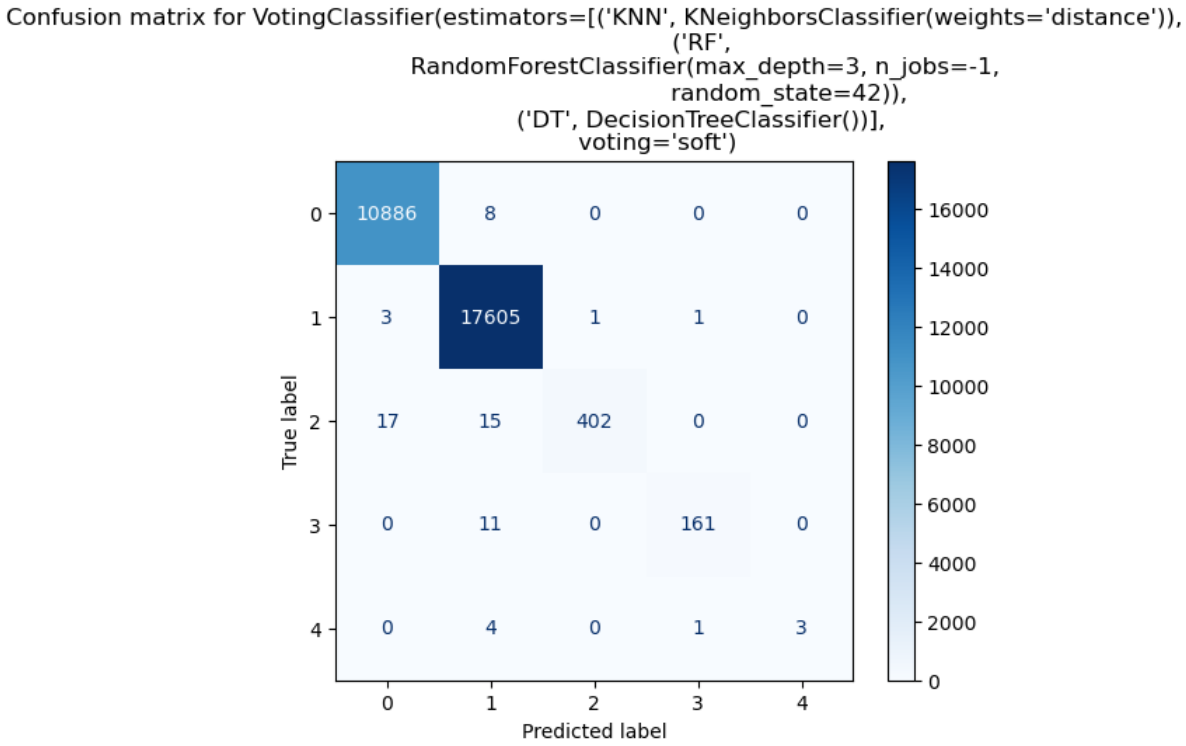


Figure 11 *Soft voting Confusion Matrix*

However, misclassifications were observed, with certain samples incorrectly predicted as belonging to other categories. 8 samples of Class 0 were misclassified as Class 1 and the rest of the 0 samples of class 0 are misclassified as class 1 , 3 samples of Class 1 were misclassified as Class 1 and the next 2 column samples were 1,1 which were misclassified as respectively class 2 and 3 and last 0 sample were misclassified as class 4 . In Class 2, the number of samples 17, 15, 0, 0 were misclassified as respectively Class 0,1,3 ,4. In Class 3, the number of samples 0 and 11,0,0 were misclassified as respectively Class 0,1,2,4. In Class 4, the number of samples 0, 4, 0, 1 were misclassified as respectively Class 0, 1, 2, 3.

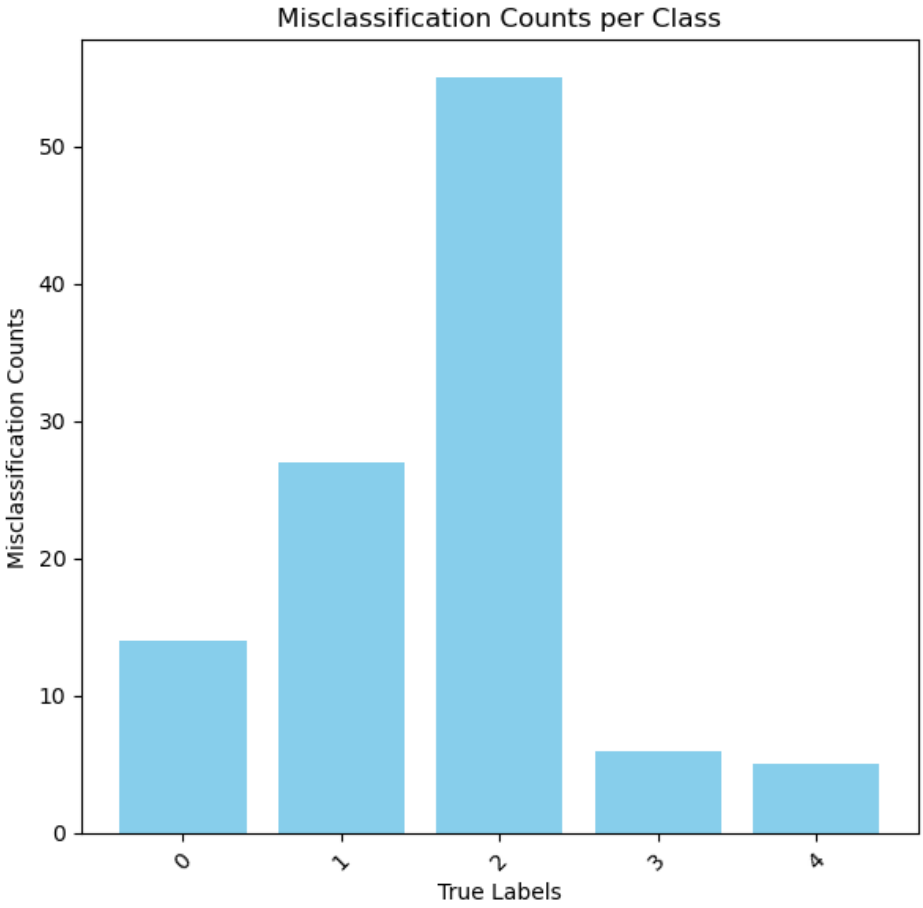


Figure 12 *Soft Voting Misclassification*

With the help of figure 15 we can say in left from Misclassification Counts per class where true labels are mentioned as a 0,1,2,3,4 are respectively Normal, Dos, probe, r2l, u2r ,where we can see class 0 has 8 misclassified from 10894 samples, class 1 has 5 misclassified from 17610 sample, class 2 has 32 misclassified from 434 samples, class 3 has 172 misclassified from 172 samples, class 4 has 5 misclassified from 8 samples.

4.4.1.6 Confusion matrix for Hard Voting Classifier

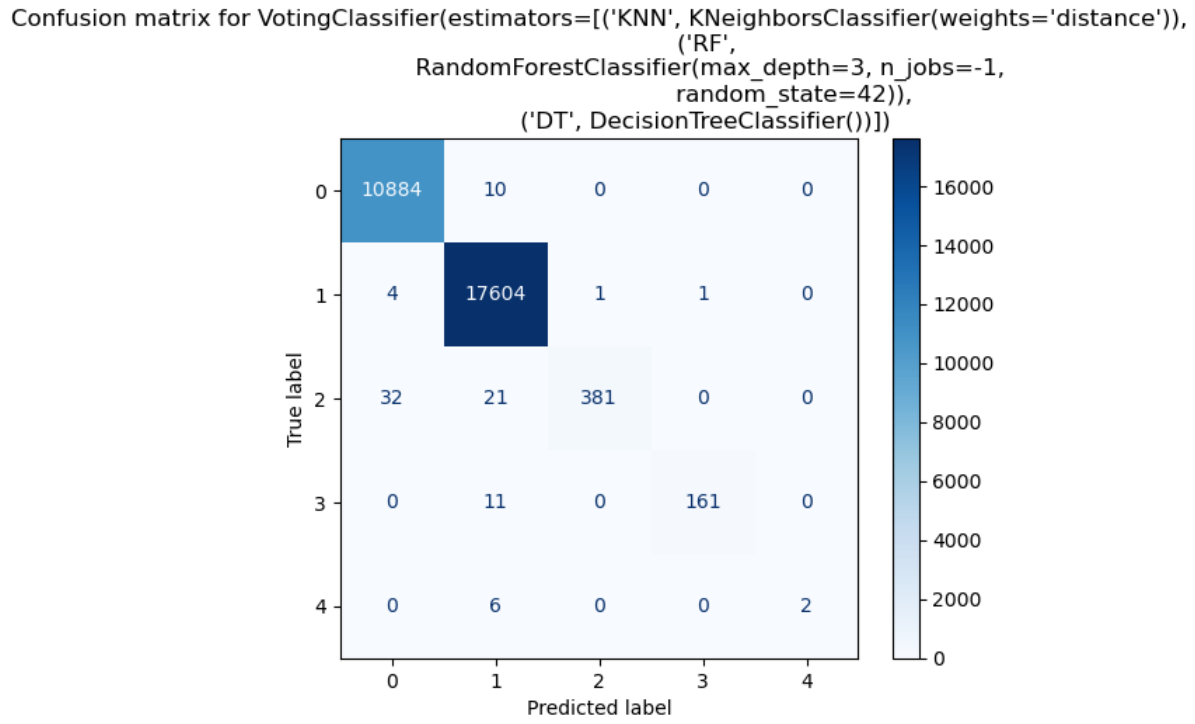


Figure 13 *Hard Voting Confusion Matrix*

The confusion matrix for the Voting Classifier is shown in Figure 16 The matrix demonstrated that the model accurately predicted 10884 samples to be in Class 0, 17604 samples to be in Class 1, and 381 samples to be in Class 2, 161 samples to be in Class 3, 2 samples to be in Class 4. However, misclassifications were observed, with certain samples incorrectly predicted as belonging to other categories. 10 samples of Class 0 were misclassified as Class 1 and the rest of the samples were 0 which were misclassified as respectively class 2, 3, 4. Same as in Class 1, the number of samples 4 and 1, 1, 0 were misclassified as respectively Class 0,2,3,4. In Class 2, the number of samples 32, 21, 0,0 were misclassified as respectively Class 0, 1, 3, 4. In Class 3, the numbers of samples 0 and 11, 0, 0 were misclassified as respectively Class 0, 1, 2, 4. In Class 4, the number of samples 0, 6, 0, 0 were misclassified as respectively Class 0, 1, 2, 3.

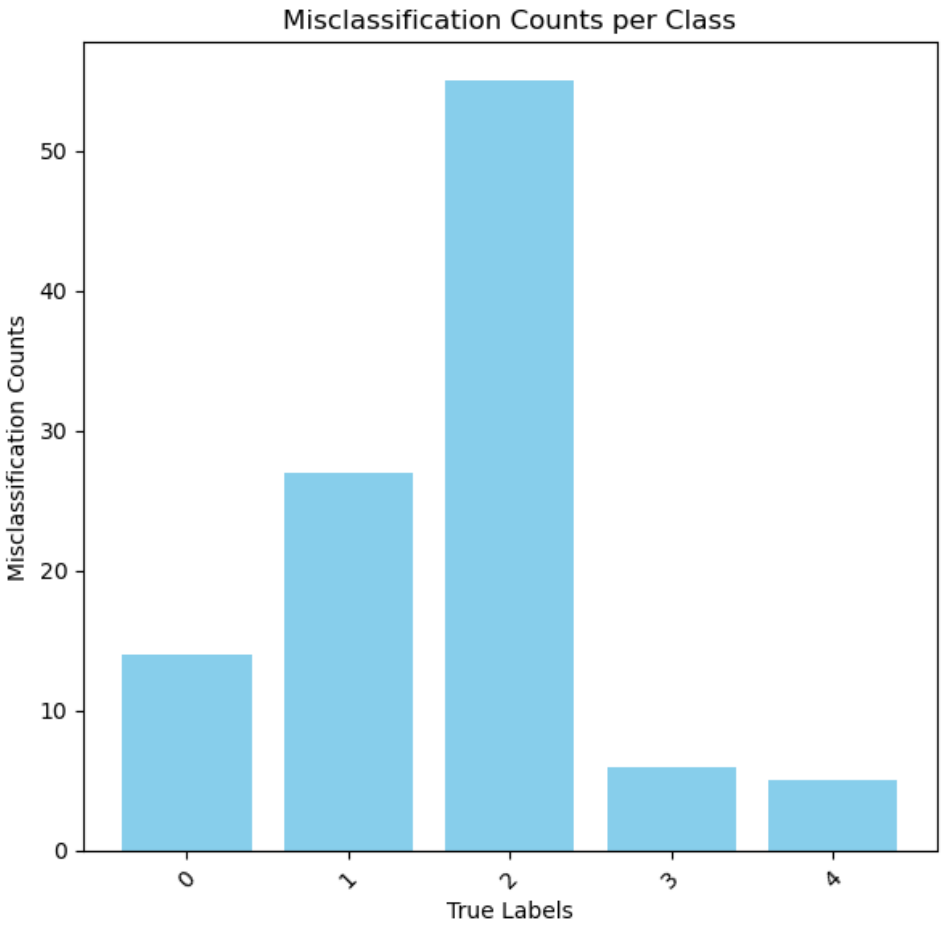


Figure 14 *Hard Voting Misclassification*

With the help of figure 17 we can say in left from Misclassification Counts per class where true labels are mentioned as a 0,1,2,3,4 are respectively Normal, Dos, probe, r2l, u2r ,where we can see class 0 has 10 misclassified from 10894 samples, class 1 has 6 misclassified from 17610 sample, class 2 has 53 misclassified from 434 samples, class 3 has 11 misclassified from 172 samples, class 4 has 8 misclassified from 8 samples.

4.4.2 Comparison between Algorithms

Now we will in down below compare the accuracy, precision, recall, and f-measure of all the algorithms that were applied with parameter tuning.

The results are presented in Figure 4.17, which shows a histogram of the performance of each algorithm. According to our observations the Decision Tree classifier and soft voting achieved the maximum accuracy, 99.8 %, while the Gaussian NB classifier had the lowest accuracy, 40.65 %.

Table 1 *Applied Algorithms Accuracy and Classification Result Analysis*

Algorithm	Accuracy	Precision	Recall	F1-Score
DT	99.8%	99.8%	99.8%	99.8%
RF	86.2%	97.7%	86.2%	90.8%
KNN	99.4%	99.5%	99.4%	99.4%
GNB	40.65%	80.2%	40.6%	39.9%
Soft Voting	**99.8%**	**99.8%**	**99.8%**	**99.8%**
Hard Voting	**99.7%**	**99.7%**	**99.7%**	**99.7%**

From Above table, we can see that, Decision Tree and Soft voting have the highest accuracy rate. There accuracy, precision, recall, F1-Score is 99.8%. Whereas Hard Voting gives 99.7%. Meanwhile KNN performed good with the accuracy of 99.4%. GNB gives lowest accuracy of 40.65% with low precision of 80.2%, low recall of 40.6% and lowest F1-score.

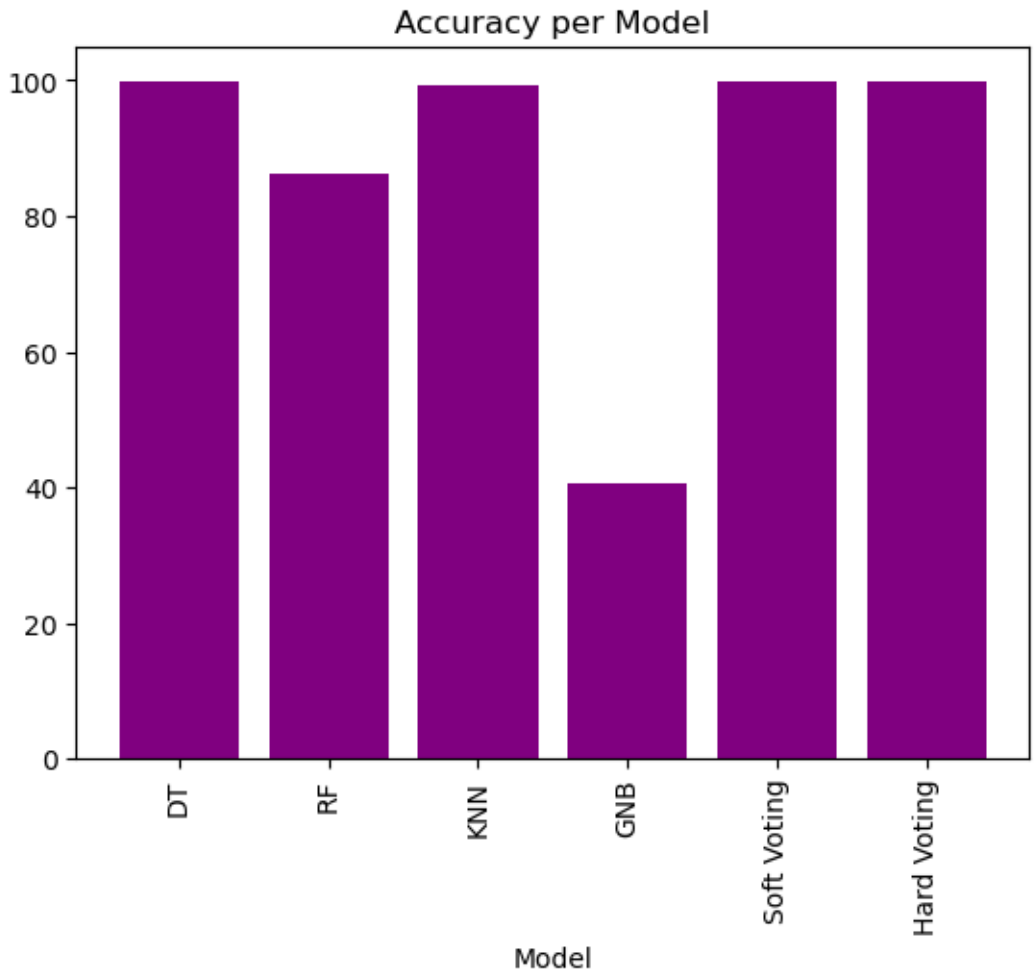


Figure 15 *Comparison between different Algorithm*

From the above graph, we can see that using all 6 algorithms we got pretty good result allover but in some instances some algorithms drop back in performance. We see that DT, KNN and both soft and hard voting performs best in comparison to other remaining algorithms. So we can say that our proposed model shows as well performance and good results as we desired.

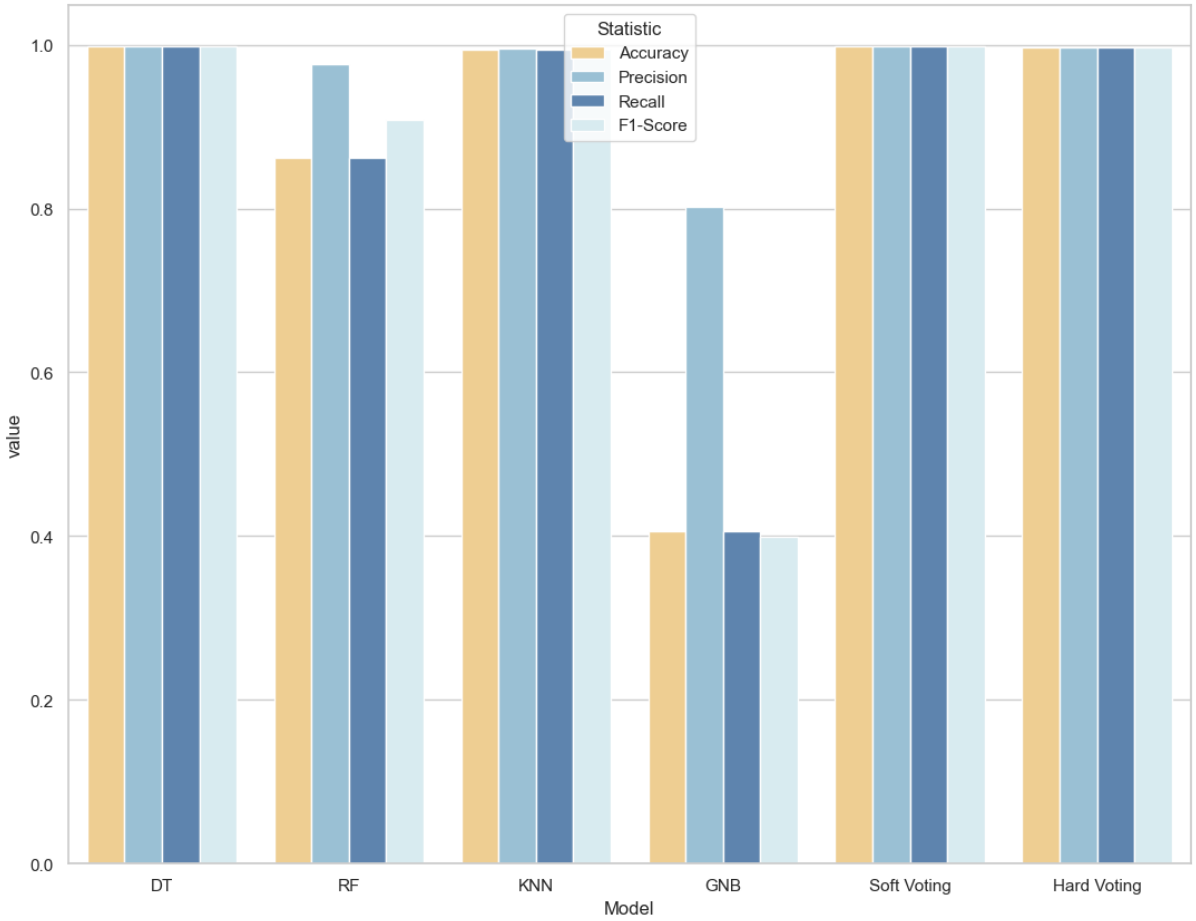


Figure 16 *Comparisons of applied algorithm based on classification report*

Figure 4.9 compares the performance of machine learning algorithms based on their F1 score, recall, and precision. Among the models, the Decision Tree and soft voting showed the highest precision score of 99.9%, recall score of 99.9%, and F1-score of 99.9%. On the other hand, the Gaussian NB model had the lowest precision score of 87.90%, recall score of 87.9%, and Fl-score of 91.3%, indicating comparatively lower performance.

4.4.3 ROC Curve

The ROC curve shows how well a model can distinguish between two classes. It does this by showing the trade-off between correctly identifying positive cases (true positive rate) and incorrectly identifying negative cases (false positive rate) at different thresholds. A smooth curve closer to the top left corner means the model is good at separating the classes, while a diagonal line suggests random guessing. The area under the curve (AUC) summarizes the model's overall performance, with higher values indicating better performance. You can use ROC curves to compare different models and understand how they perform under different conditions. Here class 0 indicates Normal attack, class 1 indicates DOS attack, class 2 indicates probe attacks and class 3 indicates R2L and class 4 indicates U2R.

4.4.3.1 ROC Curve of Decision Tree

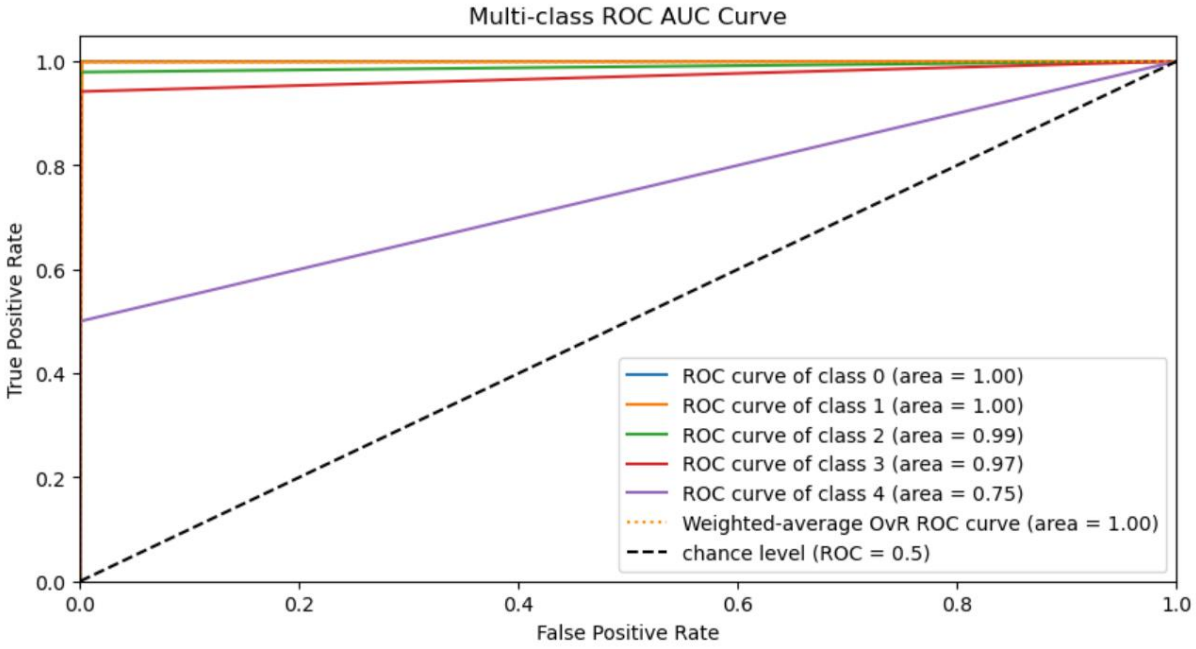


Figure 17 *Decision Tree ROC Curve*

Here the curve closer to top left corner means Decision Tree performs well in separating classes on the dataset. Here AUC is as high as 1 for both class 0 and 1 indicating that model is performing well in distinguishing between major two attack class. The other two classes performed well showing there ROC curves in top left corner with the AUC of 0.99, 0.97. The model has little struggle to distinguish class 4 with AOC of 0.75. overall the model performed well and well-fitted.

4.4.3.2 ROC Curve of Naive Bayes

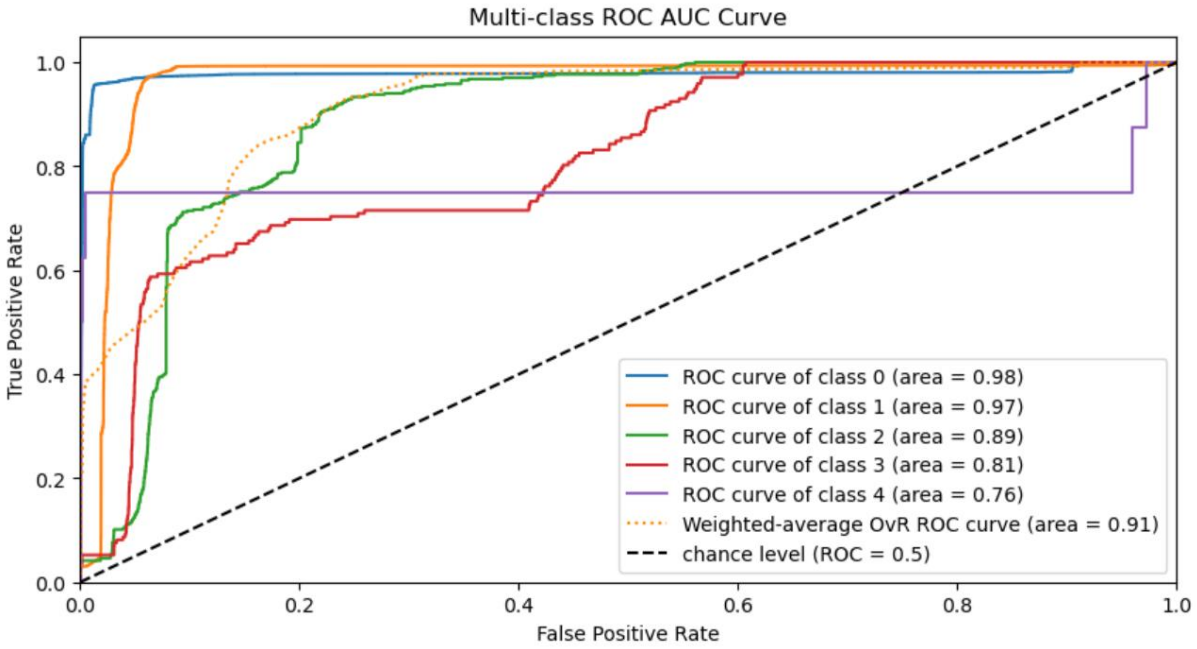


Figure 18 *Naive Bayes ROC Curve*

Here the curve closer to the left corner means Naive Bayes performs comparatively very well in separating class 0 and class 1 with an AUC of 0.98 and 0.97 respectively. Whereas, in case of other class 2, 3 and 4, the algorithm faces struggle with lower AUC of

4.4.3.3 ROC Curve of Random Forest

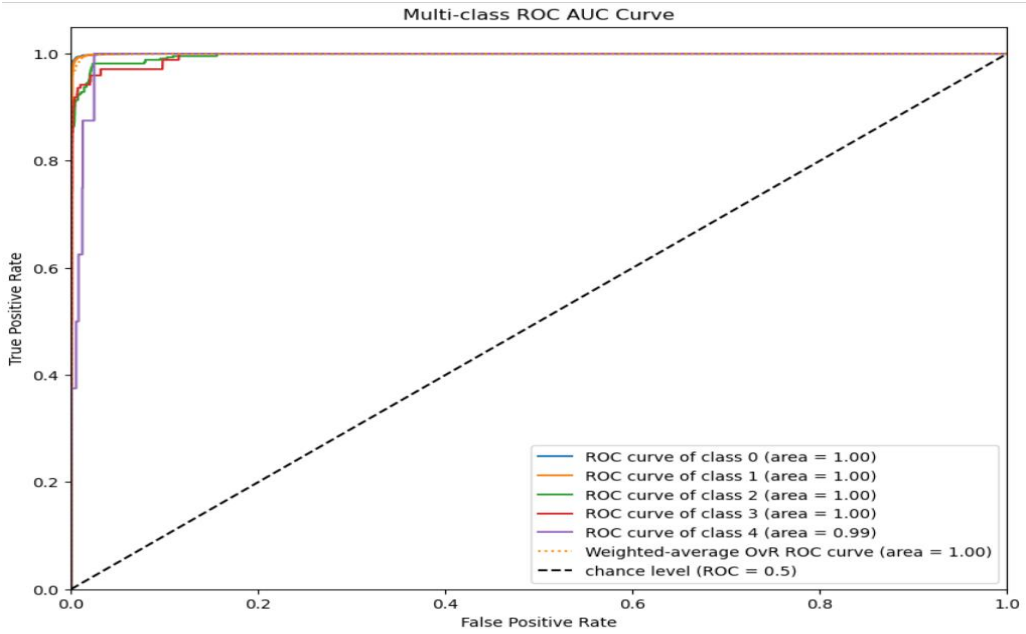


Figure 19 *Random Forest ROC Curve*

Here, The ROC curves shows the model performed well in distinguished five classes. All curves of their class are closer to top left corner with the AUC score of 1.00 and for class 4, AUC score is 0.99. This ROC curves indicates that model performed well in training phase as well as in distinguishing between classes.

4.4.3.4 KNN

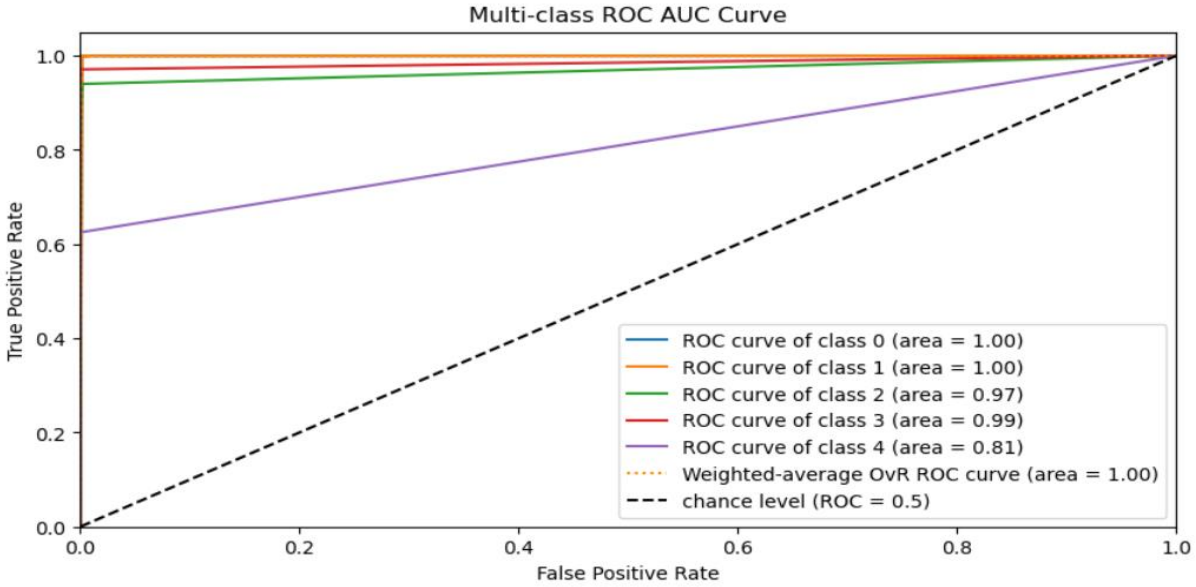


Figure 20 *KNN ROC Curve*

Here, The ROC curves of class 0, 1, 2, 3 are closer to top left corner indicatcitng KNN model performed well in distinguishing between classes with the AUC score of 1.0, 1.0, 0.97, 0.99 respectively. Class 4s has the AUC Score of .81 which is lower than other classes. Allover, KNN here performed well and a well-fitted.

4.4.3.5 Soft Voting

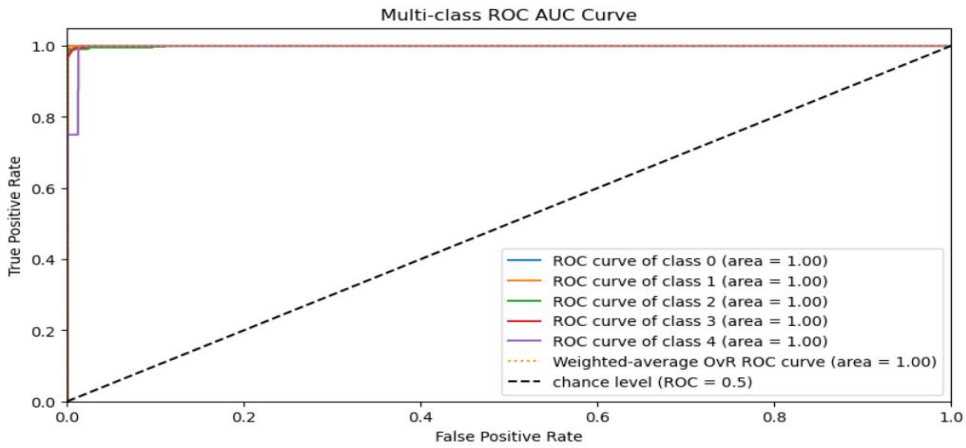


Figure 21 *Soft Voting ROC Curve*

Here the curve for all of the class is seen at most top left corner, therefore indicating a very good performance by the soft voting ensemble algorithm with all of the AUC curve at 1.00. It shows that this algorithm is very well performing in distinguishing all the 5 types of classes or attack dataset.

4.4.3.6 Hard Voting

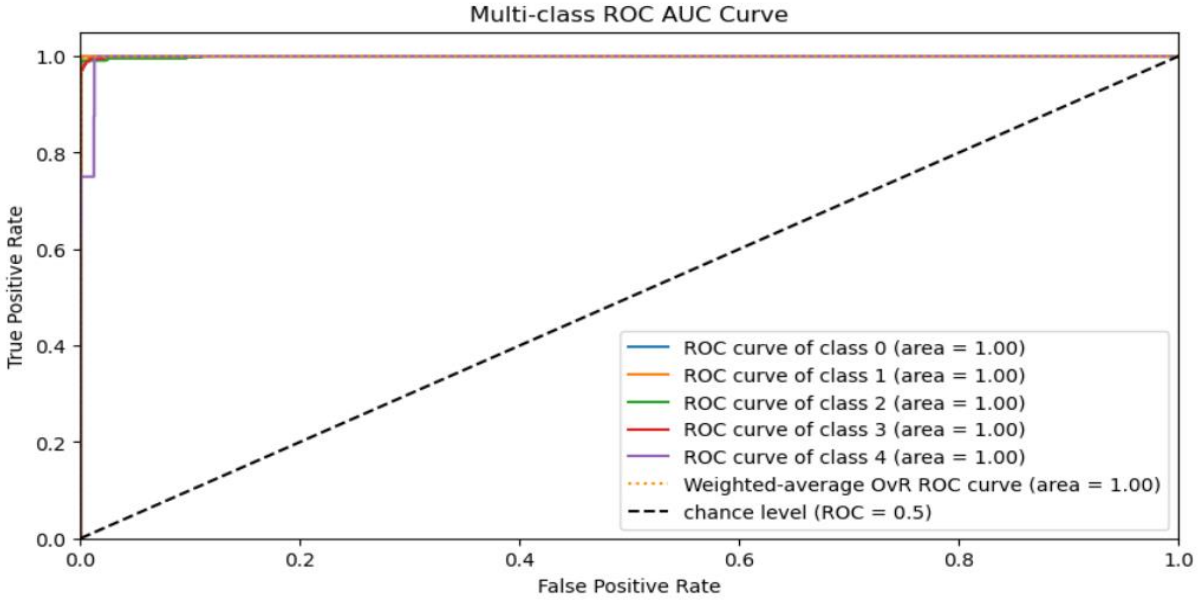


Figure 22 *Hard Voting ROC Curve*

Here, ROC curve shows that, Hard Voting performed well in distinguishing between all 5 classes with the highest AOC score of 1.00 and all the curves are near to the top left corner. This indicates that, Hard voting performed well in distinguished between classes and well-fitted model.

4.4.4 Comparison with paper

Here we studied and showed a compare in reference of a paper that was worked on previously.[9]

Table 2 Comparison of Performance with Previous Work

	Normal Attack	Dos Attack	Probe Attack	R2L	U2R
Reference Paper SVM + KNN + LR (Hard Voting)	97%	77%	69%	01%	16%
Our proposed algorithm DT + RF + KNN (Soft Voting)	**99.93%**	**99.73%**	**99.38%**	**96.51**	**62.5%**
Our proposed algorithm DT + RF + KNN (Hard Voting)	**99.90%**	**99.97%**	**87.78%**	**94.18%**	**25.0%**

4.5 Summary

In this study, we compare between DT, RF, KNN, NB and our proposed algorithm Voting Ensemble on the KDD 99 open dataset. Hard Voting is best at performing well showing all AUC value 1.00. On the secondary, decision tree shows another best performance with an accuracy of 99.8%. Then other algorithms that shows simultaneous higher accuracy are KNN and RF, as we desired. So from the analysis of studied reference and our experimental results, we can visualize that improvement have been made in the ensemble model through our experiment and analysis.

Chapter 5

Conclusion

5.1 Research Summary

As we have completed the task quite successfully, we were able to find some novel ways to improve our research work such as:

The most efficient machine learning approach is found through implementation of various models for intrusion detection system. We were able to show various comparisons among different models of machine learning on IDS dataset named as, KDD Cup 99.

5.2 Contribution of the thesis

Our main focus of contribution on this project is

1. **Comparative analysis of Existing ML Models:**

   IIn our study we tried to work with a few more ML algorithms that we have found in several research papers we have attached in the reference section. We have trained 6 type of ML algorithm such as Decision Tree , Random Forest, K-nearest neighbors algorithm , Gaussian Naive Bayes, Ensemble algorithm and a comparative analysis of various models outcomes.

2. **Approach an Ensemble Algorithm:** We tried various algorithms and applied the best ensemble model to show a higher accurate model than existing ones.

3. **Enhanced Accuracy :** We achieved better accuracy than other ensemble model used in the reference we used.

5.3 Limitations

Although we tried to improve machine learning techniques of various models approach to detecting intrusion on high scale data over a network, we faced various limitations such as computational time, data analysis etc. Also there are options of more combination of ensemble learning model that can and need to be worked on for further advancement on our

work.

5.4 Future work

To enhance prediction accuracy, integration of real-time datasets into our models Intrusion Detection Systems (IDS) can be done, enabling dynamic adaptation to evolving cyber threats. Additionally, explorations of the application of Support Vector Machines (SVM) to assess can improve impact on accuracy rates as we were unable to deploy the model due to various limitations. Also, ensemble model of more advanced ML algorithms and DM methods can enrich the existing ensemble models performance and create options for more improvement on such models. Deployment of our IDS in real-time environments through collaboration with industry partners will validate its practical efficacy in response to genuine cybersecurity threats. Continuous enrichment of used dataset and introducing new datasets, both quantitatively and qualitatively, is slated to strengthen our models. Simultaneously, the exploration of advanced techniques like deep learning architectures and ensemble models aims to unravel intricate patterns within the data, contributing to heightened accuracy. This strategy aims to propel our IDS research into the realms of real-time adaptability, robustness, and heightened accuracy, ensuring its efficacy amidst the ever-evolving landscape of cybersecurity.

Reference

[1[“KDD Cup 1999” dataset collected from www.kaggle.com

[2] I. A. M. M. Ammar Almomani, "Ensemble-Based Approach for Efficient Intrusion Detection in Network Traffic," Intelligent Automation and Soft Computing, vol. 37, no. 2, pp. 2499-2517, 2023.

[3] Anna L, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," IEEE COMMUNICATIONS SURVEYS & TUTORIALS, vol. 18, no. 2, pp. 1153-1176, 2016.

[4] D. Erhan, "Bogaziçi ˘ University distributed denial of service dataset," Data in Brief, vol. 32, 2020.

[5] J. K. ,. H. K. Jiyeon Kim, "CNN-Based Network Intrusion Detection," Multidisciplinary Digital Publishing Institute, vol. 9, 2020.

[6] Sydney M.Kasongo, Yanxia Sun, “Analysis of IDS by Feature Selection on UNSW NB-15 Dataset”, Kasongo and Sun J Big Data (2020) 7:105.

[7] Himadri Chauhan, “A Comparative Study of Classification Technique for Intrusion Detection”.

[8] M. R. P. Mrutyunjaya Panda, "NETWORK INTRUSION DETECTION USING NAIVE BAYES," IJCSNS International Journal of Computer Science and Network Security, vol. 7, no. 12, pp. 258-263.

[9] Fuat Turkn, “Analysis of Intrusion Detection Systems in UNSW-NB15 and NSL-KDD Datasets with Machine Learning Algorithms”.

[10] “Network Intrusion Detection System Using Voting Ensemble Machine Learning”, Md. Raihan-Al-Masud, Hossen Asiful Mustafa, Institute of Information and Communication Technology, Bangladesh University of Engineering and Technology.

[11] Chao Liu. Zhaojun Gu, Jialiang Wang, “A Hybrid Intrusion Detection System Based on Scalable K-means+ Random Forest and Deep Learning”, College of Safety Science and Engineering, Civil Aviation University of China.

[12] Faheem Masoodi, Alwi M Bamhdi, Tawseef A Teli, “Machine Learning for Classification analysis of Intrusion Detection on NSL-KDD Dataset”, Turkish Journal of Computer and Mathematics Education Vol.12 No.10 (2021), 2286-2293.

[13] Moses Garuba, Chunmei Liu, and Duane Fraites, “Intrusion Techniques: Comparative Study of Network Intrusion Detection Systems”, Fifth International Conference on Information Technology: New Generations.