INTRUSION DETECTION USING ENSEMBLE BASED MACHINE LEARNING

A PROJECT REPORT

Submitted by

BAALA KUMARAN S	**(950620104014)**
CHIDAMBARAM S	**(950620104017)**
DHANUSH BARATHI M	**(950620104018)**
GNANA SUTHARSTAN J	**(950620104022)**

In partial fulfillment for the award of the degree

Of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

EINSTEIN COLLEGE OF ENGINEERING TIRUNELVELI - 12

ANNA UNIVERSITY : CHENNAI – 600 025

MAY & 2024

BONAFIDE CERTIFICATE

Certified that this project report **"INTRUSION DETECTION USING ENSEMBLE BASED MACHINE LEARNING"** is the bonafide work of **"Baala Kumaran S (950620104014), Chidambaram S (950620104014), Dhanush Barathi M (950620104014), Gnana Sutharstan J (950620104014)"** who carried out the project work under my supervision.

SIGNATURE

Dr. M. Suresh Thangakrishan M.E., Ph.D.,

HEAD OF THE DEPARTMENT

Department of CSE

Einstein College of Engineering

Tirunelveli-627012

SIGNATURE

Mr. Eswaramoorthy, M.E, MBA

SUPERVISOR

Assistant Professor

Department of CSE

Einstein College of Engineering

Tirunelveli-627012

Submitted to Project and Viva Examination held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We have successfully completed the project with blessings showered onus by God, the almighty, a project of this nature needs co-operation and support from many for successful completion.

We express our heartfelt thanks to **Mr. A. MATHIVANAN, B.E., M.Sc.(Agri)**, Managing Trustee of Einstein College of Engineering, Tirunelveli, for His moral support and advise.

Our thanks to **Prof. A. AMUTHAVANAN, B.E., M.S (USA), B.L.**, Chairman of our college for making necessary arrangements to do this project.

Our hearty thanks to **Prof. A. EZHILVANAN, MBA.**, Secretary of our college for making necessary arrangements to do this project.

We wish to express our gratitude to **Dr. R. VELAYUTHAM, M.E., Ph.D., FIE**, Principal for the support he provided us to carry out this project successfully.

We are very much thankful to **Dr. M. SURESH THANGAKRISHNAN, M.E., Ph.D.**, Head of the department, Computer Science and Engineering who is always a constant of inspiration for us.

We extend our sincere and heartful thanks to our project coordinator **Mr. E. SUBRAMANIAN, M.E.**, for his valuable support and guidance for making our project successful.

We extend our sincere gratitude and thanks to our guide, **Mr. ESWARAMOORTHY, M.E, MBA**, continuous support and encouragement she extended throughout the project.

We extend our sincere thanks to all teaching and non-teaching staff members and our family members, friends for their help in completing this project.

ABSTRACT

Cybersecurity is an ever-evolving challenge, with the detection and classification of network attacks being of paramount importance in safeguarding critical systems. In this project, we address this challenge by developing a robust intrusion detection system (IDS) using machine learning techniques. We employ a dataset comprising network traffic features, including duration, protocol type, service, and others, to train and evaluate multiple classifiers.

Our methodology involves comprehensive data preprocessing, including one-hot encoding of categorical features and feature selection techniques such as ANOVA, Recursive Feature Elimination (RFE), and SelectKBest. By leveraging these techniques, we identify the most relevant features for classification, reducing dimensionality while preserving critical information.

We implement various classifiers, including Random Forest, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Naive Bayes, and Logistic Regression. Additionally, we explore ensemble learning using a Voting Classifier to combine predictions from multiple base models, enhancing overall performance.

Evaluation of the models is conducted using cross-validation and performance metrics such as accuracy, precision, recall, and F1-score. Furthermore, we analyze the Receiver Operating Characteristic (ROC) curves and Area Under the Curve (AUC) to assess the models' ability to distinguish between classes effectively.

Our results demonstrate the efficacy of the developed IDS in accurately detecting and classifying network attacks. The ensemble approach, in particular, showcases superior performance, highlighting the benefits of combining diverse classifiers for enhanced predictive power.

Our project contributes to advancing the field of cybersecurity by providing a practical framework for building efficient IDS using machine learning techniques. The proposed system offers a proactive defence mechanism against evolving cyber threats, thereby bolstering the resilience of critical infrastructures against malicious activities.

LIST OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION:

An emerging issue of the modern period is the increasing frequency of cyber-attacks. These assaults harm people and businesses alike, and they are able to affect the availability, confidentiality, and integrity of vital data that is transferred across the network. Businesses must defend their networks from hackers, intruders, and other network dangers. Therefore, in order to safeguard critical information and services from potential attacks, a network system needs to include a number of security technologies. An apparatus or program called the Intrusion Detection System (IDS) is used to identify hostile activity or strategy violations within a network. It keeps an eye out for malicious activity or security lapses on the network and alerts the administrator to any possible dangers. Cyber-attacks are growing more harmful every day.

Modern networked corporate environments need a high level of security for reliable and safe information sharing between businesses in order to stay up with the ever-evolving cyber threats. The signature and anomaly-based intrusion detection systems (IDSs) of the past are ill-suited to handle the ever-evolving patterns of network intrusion. Artificial intelligence integration with the intrusion detection system (IDS) holds promise for handling novel attack vectors and maintaining network security.

Over the past few decades, enhanced automation and prediction have been made possible by machine learning (ML), a subset of artificial intelligence, which has been widely applied to enhance intrusion detection. Handling of the extendable, replicable, and customizable datasets is made possible by ML approaches. These techniques teach IDSs to recognize and respond to both known and unknown assaults, making them more resilient. Furthermore, an ensemble of machine learning models (ML models) detects more precisely than a single ML model, which may not always forecast with accuracy. According to Dietterich, ensemble machine learning classifiers perform better than individual classifiers in a range of classification tasks. Furthermore, choosing a subset of pertinent and acceptable characteristics from a vast feature space is known as feature selection (FS).

For network intrusion detection, we have developed a supervised ensemble machine learning framework (SupEnML) in this study by merging several machine learning classifiers from different classification families, including Decision Tree, Logistic Regression, Na¨ıve Bayes, Neural Network,

and Support Vector Machine. Moreover, the classification process of this framework incorporates the ensemble feature selection (EnFS) technique. For feature selection, such as Chi-squared, Random Forests, ANOVA, Recursive Feature Elimination, and LASSO, this EnFS technique combines three major FS method types (Filter based, Wrapper based, and Embedded methods). Majority voting is used to integrate the given features, producing an optimal collection of features.

1.1 BASIC CONCEPTS:

1.1.1 WORKING PRINCIPLES OF IDS:

An Intrusion Detection System (IDS) is a critical component of cybersecurity infrastructure designed to detect and respond to unauthorized access attempts or malicious activities within a computer network or system. Traditional IDSs rely on rule-based approaches or single-machine learning models, which may have limitations in effectively identifying complex and evolving cyber threats.

Ensemble-based machine learning techniques offer a powerful solution to enhance the effectiveness of IDSs. Ensemble learning involves combining multiple machine learning models to improve predictive performance, robustness, and generalization capabilities. In the context of IDS, ensemble methods can integrate diverse detection algorithms, feature sets, or data representations to collectively detect a wide range of intrusions while minimizing false positives.

In an ensemble-based IDS, multiple base learners, such as decision trees, neural networks, support vector machines, or anomaly detection algorithms, are trained on diverse subsets of data or feature representations. These base learners may specialize in different aspects of intrusion detection, such as detecting anomalies in network traffic, identifying patterns in system logs, or recognizing known attack signatures.

The predictions or anomaly scores from individual base learners are then combined using ensemble techniques such as averaging, voting, stacking, or boosting to generate a final ensemble decision. By leveraging the complementary strengths of diverse models and data representations, ensemble-based IDSs can achieve higher detection accuracy, resilience to evasion techniques, and adaptability to emerging threats.

Overall, an Intrusion Detection System using ensemble-based machine learning offers a sophisticated and robust approach to cybersecurity, capable of effectively detecting and responding to a wide range of cyber threats in real-time.

1.2 INTRUSION DETECTION:

Intrusion detection is a cybersecurity mechanism designed to detect unauthorized access or malicious activities within a computer system or network. Its primary goal is to identify and respond to potential security breaches or threats in real-time or near real-time.

There are two main types of intrusion detection systems (IDS):

1.2.1 Host-based Intrusion Detection System (HIDS):

This type of IDS operates on individual computer systems or hosts. It monitors and analyses the logs and activities occurring within the host, looking for signs of malicious behaviour such as unauthorized access attempts, malware activity, or unusual system events.

1.2.2 Network-based Intrusion Detection System (NIDS):

NIDS monitors network traffic in real-time, examining packets flowing through the network. It identifies suspicious patterns or signatures that may indicate unauthorized access attempts, denial of service attacks, or other malicious activities targeting the network.

Intrusion detection systems employ various techniques to identify potential threats, including signature-based detection, anomaly detection, and behaviour analysis. Signature-based detection relies on known patterns or signatures of known attacks to identify and block malicious activities. Anomaly detection involves establishing a baseline of normal behaviour and flagging any deviations from this baseline as potential threats. Behaviour analysis observes the behaviour of users and systems to identify unusual or suspicious activities that may indicate a security breach.

Once suspicious activity is detected, an intrusion detection system can trigger alerts or take automated actions such as blocking traffic, quarantining affected systems, or notifying security personnel for further investigation and response.

1.3 USES OF INTRUSION DETECTION:

Intrusion detection systems (IDS) serve several crucial purposes in cybersecurity. Here are some of the primary uses:

1.3.1 Threat Detection:

The primary purpose of intrusion detection systems is to identify and alert on potential security threats or breaches within a network or system. This includes detecting unauthorized access attempts, malware infections, denial-of-service attacks, and other malicious activities.

1.3.2 Real-Time Monitoring:

IDS continuously monitor network traffic or system activity in real-time or near real-time, allowing for the timely detection of security incidents. This proactive approach helps mitigate potential damage caused by intrusions by enabling rapid response and containment measures.

1.3.3 Incident Response:

IDS alerts provide valuable information to security teams, enabling them to investigate security incidents promptly and take appropriate remedial actions. This may involve isolating compromised systems, blocking malicious traffic, or applying patches to vulnerable software.

1.3.4 Forensic Analysis:

IDS logs and alerts can be invaluable for conducting forensic analysis after a security incident. By reviewing IDS data, security analysts can reconstruct the sequence of events leading up to the intrusion, identify the attack vector, and determine the extent of the compromise.

1.3.5 Compliance Requirements:

Many regulatory standards and industry mandates require organizations to implement intrusion detection systems as part of their cybersecurity framework. Compliance with standards such as PCI DSS, HIPAA, and GDPR often necessitates the deployment of IDS to protect sensitive data and ensure regulatory compliance.

1.3.6 Security Policy Enforcement:

IDS can help enforce security policies by detecting and alerting on policy violations or unauthorized activities. This ensures that organizational security policies are adhered to and helps prevent unauthorized access to critical resources.

1.3.7 Threat Intelligence Integration:

Some IDS solutions integrate with threat intelligence feeds to enhance their detection capabilities. By correlating network activity with known threat indicators, IDS can identify and respond to emerging threats more effectively.

1.3.8 Risk Management:

IDS plays a vital role in risk management by providing visibility into potential security threats and vulnerabilities. By identifying and mitigating security risks proactively, organizations can minimize the likelihood and impact of security breaches.

Overall, intrusion detection systems are essential components of a comprehensive cybersecurity strategy, providing organizations with the visibility and capabilities needed to detect, respond to, and mitigate security threats effectively.

1.4 TERMINOLOGIES:

In the field of intrusion detection and cybersecurity, various terminologies are commonly used to describe concepts, techniques, and technologies. Here are some key terminologies:

1.4.1 Intrusion Detection System (IDS):

A security mechanism designed to detect unauthorized access or malicious activities within a computer system or network.

1.4.2 Intrusion Prevention System (IPS):

A security mechanism that not only detects but also actively prevents unauthorized access or malicious activities by blocking or filtering network traffic.

1.4.3 Signature-based Detection:

A method of intrusion detection that relies on predefined signatures or patterns of known attacks to identify and block malicious activity.

1.4.4 Anomaly-based Detection:

A method of intrusion detection that identifies deviations from normal behaviour or system activity, potentially indicating malicious behaviour.

1.4.5 Machine Learning:

A subset of artificial intelligence (AI) that enables systems to learn from data and make predictions or decisions without being explicitly programmed.

1.4.6 Ensemble Learning:

A machine learning technique that combines multiple diverse models to improve prediction accuracy and robustness.

1.4.7 Feature Engineering:

The process of selecting, transforming, and extracting relevant features from raw data to improve the performance of machine learning models.

1.4.8 False Positive:

An alert or alarm generated by an intrusion detection system incorrectly indicating the presence of a threat or attack when none exists.

1.4.9 False Negative:

Failure of an intrusion detection system to detect a genuine security threat or attack, leading to a lack of alert or alarm.

1.4.10 Packet Sniffing:

The process of capturing and analysing network traffic packets to monitor and inspect data flowing over a network.

CHAPTER 2

LITERATURE SURVEY:

2.1 DDoS intrusion detection through machine learning ensemble.

Author: S. Das, A. M. Mahfouz, D. Venugopal, and S. Shiva.

Journal: IEEE 19th Int. Conf.

The study focus on reduced feature sets to enhance detection accuracy. NIDS detects abnormal activity due to intrusion by an attacker. Anomaly-based intrusion detection and challenges are discussed. Application enhanced domain knowledge with machine learning techniques. NIDS with ensemble models to detect DDoS attacks accurately. Combines classifiers for robust defense against new intrusion patterns. NIDS detects DDoS attacks using ensemble models with reduced features. Achieves 99.1% DDoS attack detection accuracy with NSL-KDD dataset. Compares results with existing approaches, focusing on new attack patterns.

Data collection:

- NSL-KDD dataset with reduced features used for analysis.
- Four classifiers from different families utilized: MLP, SMO, IBK, J48.

System Design:

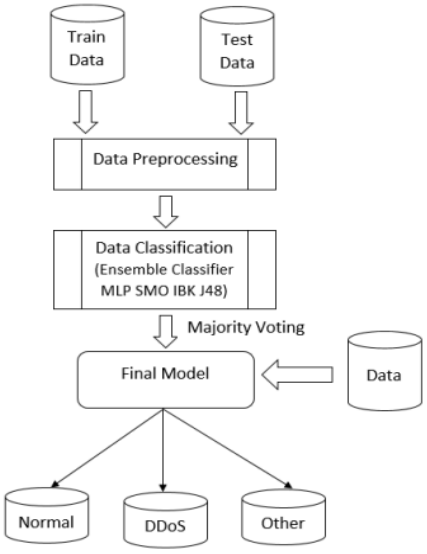

Figure:2.1.1

Method:

- Ensemble models with different classifiers for intrusion detection.
- Reduced feature sets to improve accuracy in detecting DDoS attacks.

Merits:

- Ensemble models combine classifiers for robust defense mechanism against new intrusions.
- Improved accuracy in detecting DDoS attacks with reduced feature set.
- Deep analysis of DDoS attacks to enhance learning capability.

Demerits:

- The existing techniques lack domain knowledge in ensemble methods.
- Models may perform poorly for newer forms of intrusions.

2.2Toward generating a new intrusion detection dataset and intrusion traffic characterization

Author: Iman Sharafaldin, Arash Habibi Lashkari and Ali A.Ghorbani

Journal: ICISSP 2018

The paper addresses the need for reliable intrusion detection datasets. Evaluates network traffic features and machine learning algorithms for attack detection. Highlights the importance of IDS in network security against intrusions. evaluates existing datasets for intrusion detection system performance. Compares outdated datasets like DARPA98, KDD99, ISC2012, and ADFA13. Identifies limitations in existing datasets, such as lack of diversity. Develops a new dataset with benign and common attack network flows. Analyzes network traffic features and machine learning algorithms for detection.

Data collection:

- Dataset creation with benign and common attack network flows.
- Evaluation of network traffic features and machine learning algorithms.

Method:

- The Evaluation of network traffic features and machine learning algorithms.
- Use of CICFlowMeter for extracting 80 features from pcap files.
- Application of Random Forest Regressor for feature selection.
- Creation of attack profiles based on common attack families.
- Development of B-Profile system for generating benign background traffic.

Merits:

- Generating a new IDS dataset with updated attack families.
- Evaluating network traffic features and machine learning algorithms.
- Addressing limitations of existing datasets for intrusion detection systems.

System Design:

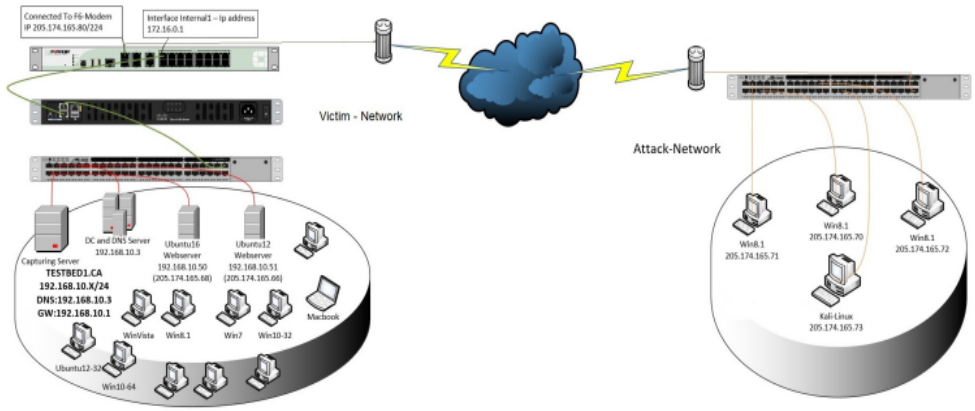

Figure:2.1.2

Demerits:

- Lack of traffic diversity and volumes in existing datasets.
- Outdated datasets with unreliable information for intrusion detection evaluation.
- Some datasets lack feature set, metadata, and do not cover attack variety.
- Existing datasets anonymize packet information and payload, hindering trend reflection.

2.3 An adaptive ensemble machine learning model for intrusion detection

Author: X. Gao, C. Shan, C. Hu, Z. Niu, and Z. Liu.

Journal: IEEE Access 2019.

The paper focuses on intrusion detection technology using ensemble learning models. Proposes Multi-Tree and adaptive voting algorithms for improved detection accuracy. Emphasizes the importance of data quality for detection effectiveness. Proposes adaptive ensemble learning model for intrusion detection technology enhancement. Utilizes NSL-KDD dataset to analyze intrusion detection technology progress. Focuses on improving detection accuracy through ensemble adaptive voting algorithm. Emphasizes the importance of data feature quality for detection effectiveness.

Data Collection:

- NSL-KDD dataset chosen for research due to ideal intrusion detection comparison.
- Preprocessing operations convert character-type fields into usable machine learning inputs

Method:

- Adaptive ensemble learning model with common machine learning algorithms.
- Feature selection, unbalanced sampling, class weight, multi-layer detection for boosting.
- Use of decision tree, SVM, logical regression, k-NN, Ada boost, random forest.

System Design:

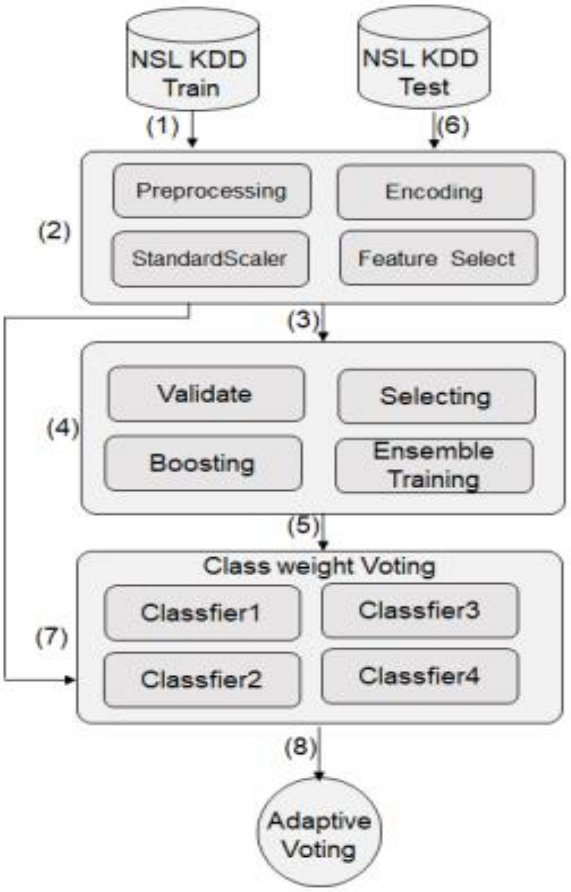

Figure:2.1.3

Merits:

- Adaptive ensemble model improves detection accuracy effectively.
- Ensemble learning gathers advantages of different algorithms for improved detection.

Demerits:

- The traditional network intrusion detection system has drawbacks in timely detection.
- Imbalance in data types affects accuracy, especially for U2R type.

2.4 A visualized botnet detection system based deep learning for the Internet of Things networks of smart cities.

Author: R. Vinayakumar, M. Alazab, S. Srinivasan, Q.-V. Pham, S. K. Padannayil, and K. Simran.

Journal: IEEE Trans. 2020.

The paper research focuses on cyber security, digital forensics, and emerging cyber issues. Proposes a botnet detection system analyzing DNS services in IoT. Researcher focuses on cyber security, digital forensics, and emerging cyber issues.

Data collection:

- Two types: public sources and real-time environment.
- IoT streaming data integration framework proposed.
- General system for generating feature vector from diverse data streams.

System Design:

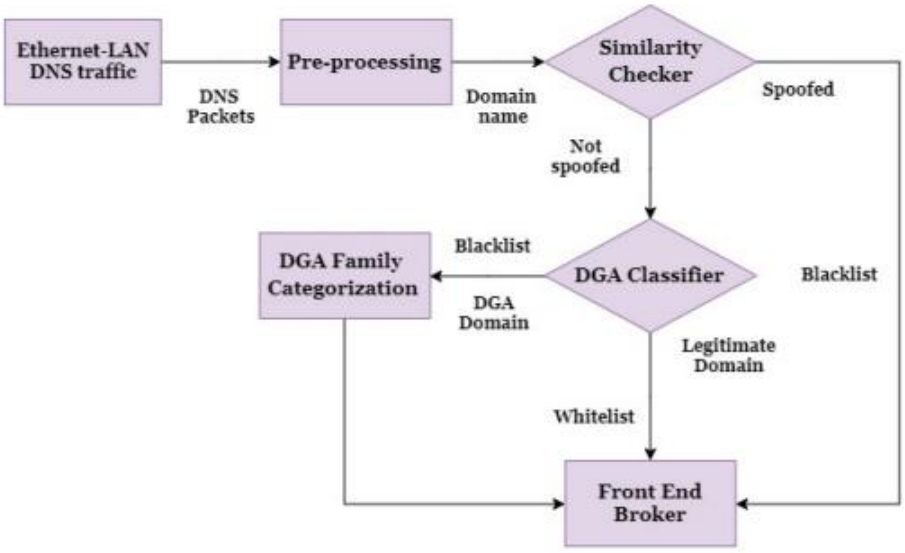

Figure:2.1.4

Method:

- Botnet detection system analyzing DNS services in IoT applications.
- Name spoofing detection techniques based on visual comparisons.

Merits:

- Research focuses on cyber security, digital forensics, and machine learning.
- Siamese network method used for visual similarity in domain names.
- Cost-sensitive deep learning architectures address class imbalance in models.
- Dataset division for training, validation, and testing siamese networks

Demerits:

- The paper face scalability challenges and resource intensiveness when deployed in large-scale IoT networks of smart cities, impacting its feasibility for resource-constrained devices.
- The system's generalizability to diverse IoT devices and botnet behaviors, limited availability of training data, and the interpretability of deep learning models are additional demerits.

2.5 Empirical Evaluation of the Ensemble Framework for Feature Selection in DDoS Attack

Author: S. Das, D. Venugopal, S. Shiva, and F. T. Sheldon.

Journal: IEEE Int. Conf. Edge Comput. 2020.

The Paper focuses on ML-based Intrusion Detection Systems for DDoS attacks. Proposes an ensemble framework for feature selection to enhance classification accuracy. Utilizes seven feature selection methods to optimize feature sets. Emphasizes the importance of feature selection in model classification. DDoS attacks impact global economy, ML-based IDSs aim to reduce incidents. Ensemble framework for feature selection enhances ML classification accuracy.

Data collection:

- NSL-KDD dataset with 41 predictor attributes and 1 target attribute.

Method:

- Embedded methods.
- Wrapper-based methods.
- LASSO Regression.
- Recursive Feature Elimination.

System Design:

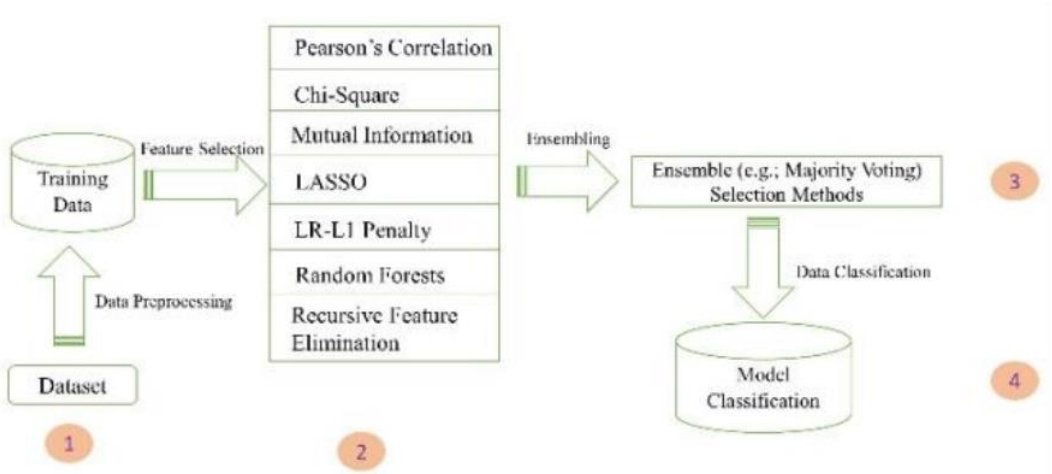

Figure:2.1.5

Merits:

- Ensemble framework enhances feature selection for DDoS attack detection.
- Filter-based methods improve ML model training by selecting relevant features.
- Experimental results show extracted features from seven FS methods.

Demerits:

- The paper only focus on a specific type of DDoS attack or a narrow set of features, which may limit its applicability to broader contexts.
- The use of a small sample size or limited dataset could undermine the robustness and generalizability of the empirical evaluation, raising questions about the reliability of the results in real-world scenarios.

CHAPTER 3

SYSTEM ANALYSIS:

3.1 Existing Project:

The existing system described in the sources is a machine learning-based comprehensive security solution for network intrusion detection using ensemble supervised machine learning framework and ensemble feature selection methods. The system aims to achieve higher accuracy with minimal false positive rates in identifying intrusions in network systems.

The system utilizes three datasets (NSL-KDD, UNSW-NB15, and CICIDS2017) for experimentation and comparative analysis of machine learning models and feature selection methods. The system employs an ensemble feature selection and ensemble classification algorithm to improve overall performance.

The system reduces the feature set for classification models by a significant percentage, resulting in improved efficiency. The system outperforms individual classifiers and demonstrates better performance metrics compared to existing solutions. The system outperforms individual classifiers and demonstrates better performance metrics compared to existing solutions.

Future plans for the system include incorporating the full amount of data, considering unsupervised learning, and exploring adversarial machine learning for enhanced security. The existing system may have limitations in terms of scalability and adaptability to different network environments.

While the ensemble feature selection (EnFS) method yields better performance for most metrics, it may not always generate equal or better results compared to individual feature selection methods for all performance metrics. The system requires a significant amount of computational resources and training time, especially when using certain feature selection methods. The existing system uses binary classification. The system employs machine learning models such as Logistic Regression (LR), Decision Tree (DT), Naive Bayes (NB), Neural Network (NN), and Support Vector Machine (SVM) for classification. These models are trained using training data and tested using testing data, generating a prediction matrix with binary outputs (1 for anomaly and 0 for benign).

The prediction matrix, along with the corresponding labels from the testing data, is used as input for ensemble classifiers. The ensemble classifiers, such as Ensemble Majority Voting (EnsMV), Ensemble Logistic Regression (EnsLR), Ensemble Naive Bayes (EnsNB), Ensemble Neural Network (EnsNN), Ensemble Decision Tree (EnsDT), and Ensemble Support Vector Machine (EnsSVM), are trained and tested to obtain the best performing model for anomaly detection.

3.2 Proposed Project:

This proposed system comprises of mainly four modules and they are,

1. Data Preprocessing
2. Ensemble Feature Selection Algorithms
    - Anova
    - Chi-squared
    - LASSO
    - Recursive Feature Elimination
    - Mutual Information
3. Model Classifications
    - Random Forest
    - Logistic regression (LR)
    - Neural Network (NB)
    - SVM
    - KNN
4. Apply Ensemble method (Majority Voting Classifier)

This project presents a comprehensive methodology for using ensemble-based machine learning. It aims to ensure network security by detecting and preventing unauthorized access and malicious attacks. This system utilizes a comprehensive approach by employing an ensemble supervised machine learning framework and ensemble feature selection methods. This research paper presents a comparative analysis of various machine learning models and feature selection

techniques to achieve heightened accuracy and minimize false positive rates. The Experimental data from the NSL-KDD dataset is used to evaluate the detection model, which successfully identifies 95.7% of intrusions, outperforming existing solutions. The NSL-KDD dataset, which includes class labels for 39 cyberattack types and 41 attributes, was used for detecting network intrusion. In this project Ensemble feature selection techniques were used to select optimal features for the detection model. The performance of different machine learning algorithms, such as Random Forest, Logistic Regression, Neural Network, KNN, and Support Vector Machine, was evaluated. In this project Five feature selection methods are used, namely Anova, Chi-Square, LASSO, mutual information and Recursive Feature Elimination, were chosen based on their performances. The method of choosing the optimal feature subset using the majority voting technique is known as ensemble feature selection. It ranks the features based on the selection of more than half of the feature selection techniques.

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1ARCHITECTURE DIAGRAM:

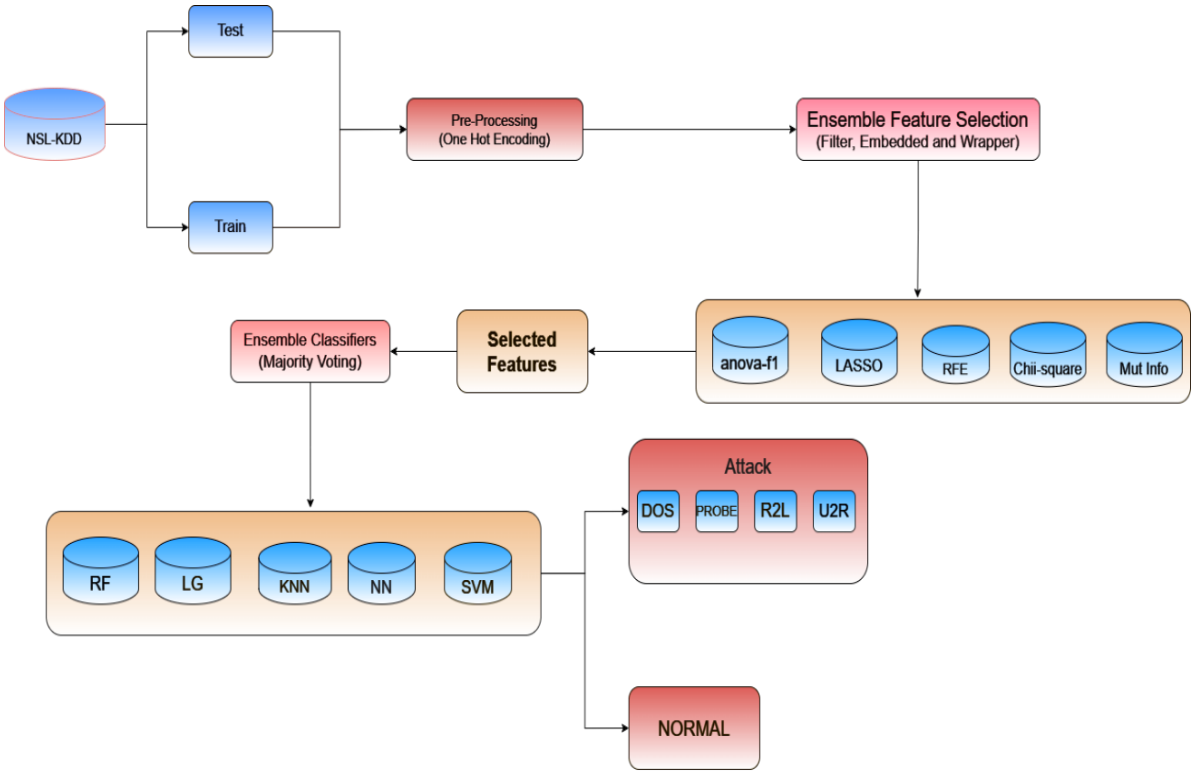

Figure 4.1.1: System Architecture of the Proposed Workflow

The system diagram in **Figure 4.1** describes a comprehensive machine learning process designed specifically for classification tasks, particularly in the context of network intrusion detection using the NSL-KDD dataset. The NSL-KDD dataset, likely a dataset containing network traffic data with different types of attacks, is divided into two subsets: a training set and a testing set. This division is crucial for training and evaluating the performance of the machine learning models. Before feeding the data into the machine learning algorithms, preprocessing is carried out. Label encoding is a technique used to convert categorical variables into a numerical format, which is easier for machine learning algorithms to handle. Feature selection is essential for building effective models, as it helps in selecting the most

relevant features and reducing dimensionality. In this step, ensemble methods for feature selection are employed. These methods could include filter, embedded, and wrapper methods, each serving different purposes in selecting features that contribute most to the predictive power of the models. several ensemble classifiers are utilized in this process, including Random Forest, Logistic Regression, K-Nearest Neighbors, Support Vector Machine, and Neural Network. The ensemble approach combines the predictions of multiple individual classifiers using a majority voting mechanism, potentially improving overall performance and robustness. Finally, the output of the classification process includes different categories representing various network behaviors. These categories could include Denial of Service, Probe, Remote to Local, User to Root, and Normal. Each category likely corresponds to different types of network intrusions or activities, allowing for the identification and classification of network threats.

4.2 MODULE DESCRIPTION

The modules of the system are,

1. Data Preprocessing
2. Ensemble Feature Selection Algorithms
    - Anova
    - Chi-squared
    - LASSO
    - Recursive Feature Elimination
    - Mutual Information
3. Model Classifications
    - Random Forest
    - Logistic regression (LR)
    - Neural Network (NB)
    - SVM
    - KNN
4. Apply Ensemble method (Majority Voting Classifier)

4.2.1 DATA PREPROCESSING

Categorical variables like protocol type, service, and flag are converted into numerical format directly, without creating binary vectors as in one-hot encoding. Label encoding assigns a unique numerical label to each category within a categorical variable. For example, if there are three categories in a variable (e.g., 'TCP', 'UDP', 'ICMP'), they might be encoded as 0, 1, and 2 respectively. After label encoding, numerical features may undergo normalization or standardization to ensure they're on a similar scale, as described earlier.

4.2.2 ENSEMBLE FEATURE SELECTION ALGORITHMS

This module involves selecting the most relevant features from the dataset to improve the performance of the machine learning models and reduce dimensionality. Ensemble feature selection methos are employed, including filter, embedded and wrapper methods. Filter methods evaluate the relevance of features based on statistical measures, embedded methods incorporate feature selection within the model training process, and wrapper methods select features based on the performance of the model.

This project utilizes five feature selection algorithms and they are,

- Anova
- Chi-squared
- LASSO
- Recursive Feature Elimination
- Mutual Information

4.2.2.1 ANALYSIS OF VARIANCE(ANOVA) FEATURE SELECTION TECHNIQUE

ANOVA (Analysis of Variance) feature selection is a statistical method used to identify the most relevant features in a dataset based on their variance and significance in relation to the target variable. In the context of the NSL-KDD dataset for network intrusion detection, ANOVA feature selection aims to determine which features are most discriminative for distinguishing between different types of network traffic, including normal traffic and various types of attacks.

The target variable in the NSL-KDD dataset would likely be the class label indicating the type of network behavior, such as normal traffic or specific types of attacks (Denial of Service, Probe, Remote to Local, User to Root). ANOVA computes the F-value statistic for each feature in the dataset by comparing the variance between the groups (different classes of network behavior) to the variance within the groups. Features with higher F-values indicate larger differences in means between the groups, suggesting they are more relevant for distinguishing between the classes.

4.2.2.2 CHI-SQUARE FEATURE SELECTION TECHNIQUE

Chi-square feature selection is a statistical method used to identify the most significant features in a dataset by assessing the relationship between categorical variables and the target variable. In the context of the NSL-KDD dataset for network intrusion detection, Chi-square feature selection aims to select features that are most informative for distinguishing between different types of network traffic behaviors, including normal traffic and various types of attacks.

The Chi-square test measures the independence between a categorical feature and the class label, assessing whether the observed frequencies of categories within each feature significantly differ from the expected frequencies under the assumption of independence. Features with high Chi-square statistics indicate a strong association with the target variable, suggesting they are more relevant for classification. In the NSL-KDD dataset, Chi-square feature selection would involve computing Chi-square statistics for each categorical feature with respect to the class label, selecting features with the highest Chi-square values as they provide the most discriminatory power for detecting network intrusions.

4.2.2.3 LEAST ABSOLUTE SHRINKAGE AND SELECTION OPERATOR(LASSO) FEATURE SELECTION TECHNIQUE

LASSO (Least Absolute Shrinkage and Selection Operator) feature selection is a method used to identify the most relevant features in a dataset by imposing a penalty on the absolute size of the regression coefficients during model training. In the context of the NSL-KDD dataset for network intrusion detection, LASSO feature selection aims to select features that are most predictive of different types of network behaviors, including normal traffic and various types of attacks. LASSO achieves feature selection by adding a penalty term to the traditional linear regression objective function, which penalizes the absolute values of the regression coefficients. As a result, some coefficients may be shrunk to zero,

effectively eliminating the corresponding features from the model. In the NSL-KDD dataset, LASSO feature selection involves training a linear regression model with L1 regularization, which encourages sparsity in the resulting coefficient vector. Features with non-zero coefficients in the trained LASSO model are considered the most relevant and are selected for further analysis.

4.2.2.4 RECURSIVE FEATURE ELEMINATION FEATURE SELECTION TECHNIQUE

Recursive Feature Elimination (RFE) is a feature selection technique used to identify the most important features in a dataset by iteratively removing the least significant features based on the performance of a predictive model. In the context of the NSL-KDD dataset for network intrusion detection, RFE aims to select features that contribute the most to distinguishing between different types of network behaviors, including normal traffic and various types of attacks. RFE typically starts by training a machine learning model on the entire feature set and ranks the features based on their importance scores, which could be derived from coefficients in linear models or feature importance measures in tree-based models. Then, the least important features are eliminated from the dataset, and the model is retrained on the reduced feature set. This process is repeated until a predetermined number of features is reached or until the performance of the model no longer improves.

4.2.2.5 MUTUAL INFORMATION FEATURE SELECTION TECHNIQUE

Mutual Information (MI) feature selection is a technique used to identify the most informative features in a dataset by measuring the statistical dependency between each feature and the target variable. In the context of the NSL-KDD dataset for network intrusion detection, MI feature selection aims to select features that provide the most relevant information for distinguishing between different types of network behaviors, including normal traffic and various types of attacks. MI quantifies the amount of information that one variable (feature) contains about another variable (target) and is particularly useful when dealing with both categorical and continuous variables. In the NSL-KDD dataset, MI feature selection involves calculating the MI scores between each feature and the target variable, where higher MI scores indicate stronger associations between the feature and the target. Features with high MI scores are considered more informative and are selected for further analysis.

4.3 MODEL CLASSIFICATION

Random Forest is an ensemble learning method known for its robustness and high performance in classification tasks. It operates by constructing multiple decision trees during training, each trained on a random subset of the data and features. The final prediction is made by aggregating the predictions of all individual trees, typically through a majority voting mechanism, resulting in improved accuracy and reduced overfitting. Logistic Regression, on the other hand, is a linear classification algorithm commonly used for binary classification tasks. It models the probability of an instance belonging to a particular class using a logistic function, making it suitable for probabilistic predictions. K-Nearest Neighbors (KNN) is a simple yet effective algorithm that classifies instances based on the majority class of their k nearest neighbors in the feature space. Support Vector Machine (SVM), another powerful algorithm, finds the hyperplane that best separates instances of different classes in the feature space. Neural Network is a versatile and widely used deep learning algorithm that consists of interconnected layers of neurons. It learns complex patterns and relationships from the data through iterative optimization of weights and biases, allowing it to achieve high performance in various classification tasks.

4.4 ENSEMBLE METHOD (MAJORITY VOTING CLASSIFIER) TECHNIQUE

Ensemble majority voting classifier is a powerful technique used in machine learning for improving the accuracy and robustness of classification models. It involves combining the predictions of multiple individual classifiers to make a final decision. In this technique, each base classifier is trained independently on the same dataset but may use different algorithms or subsets of features. During prediction, each classifier assigns a class label to a new instance, and the final prediction is determined by aggregating the individual predictions using a majority voting scheme.

The class label that receives the most votes among all classifiers is selected as the final prediction. Ensemble majority voting helps mitigate the weaknesses of individual classifiers by leveraging the collective knowledge of diverse models, thus reducing the risk of overfitting and improving overall performance. This approach is particularly effective when the base classifiers are diverse and make errors independently, as it tends to produce more accurate and stable predictions compared to any single classifier.

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 OVERVIEW

This chapter contributes a detailed description about the dataset collections, data preprocessing and various algorithms and methodologies involved in implementing the modules in the proposed system.

5.2 DATASET COLLECTION

The caliber of the data used to train a machine learning model has a significant impact on the model's performance. The data must be cleansed in order to facilitate additional analysis, as they include essential information about the issue domain. Here, we train, test, and validate our suggested framework using the well-known datasets: NSL-KDD. The dataset is divided into two parts, which we refer to as the train and test portions.

5.3 DATA PREPROCESSING

Before feeding the ML classifiers, the raw datasets must be cleansed, sanitized, transformed, standardized, and feature reduced. We used the data encoding technique to turn the non-numeric data for the NSL-KDD datasets into numeric since the ML models can handle numeric input. Even though level encoding well-known method for transforming data, we haven't employed them because they add dimension to the data and don't give feature names where MinMax scaling technique have limited range between 0 and 1. This may not be suitable for all types of data or machine learning algorithms. Some algorithms might perform better with different scaling ranges or distributions. So, we used our algorithm to carry out this conversion. We employ the One-Hot encoding technique, which is a powerful and versatile technique for handling categorical data in machine learning and also fits a range of values inside a scale spanning from 0 to 1, to normalize the data. We focused on attacks like Denial of Service (DoS), Root to Local (R2L), User to Root (U2R), Probing (Probe). The first step in creating a balanced training dataset is selecting the data type (attack or benign) that has the fewest data instances in the training dataset. To create a balanced

training set for NSL-KDD dataset, the dataset was split into 75 to 25 ratio where 75% is for training, and 25% is for testing.

5.4 FEATURE SELECTION TECHNIQUES

The method of choosing the optimal feature subset using the majority voting technique is known as ensemble feature selection. We employ five distinct FS techniques. Based on the performances, we have chosen the five Feature Selection methods such as Anova, Mutual Information, Chi-Square, LASSO and Recursive Feature Elimination.

5.4.1 ANOVA

The primary use of ANOVA (Analysis of Variance) in Intrusion Detection Systems (IDS) is its ability to identify and examine differences across various data sets.

Here's the basic formula: The total sum of squares (SST) is decomposed into the SSB and SSW:

$$SST = SSB + SSW$$

5.4.2 LASSO

LASSO improves model interpretability by adding a penalty term on the regression coefficients, which is essential for comprehending and reducing possible security risks.

$$\text{minimize}_{\beta_0,\beta}\left\{\frac{1}{2n}\sum_{i=1}^{n}\left(y_i-\beta_0-x_i^T\beta\right)^2+\lambda\sum_{j=1}^{p}|\beta_j|\right\}$$

Here:

- β_0 is the intercept,
- $\beta = (\beta_1, \beta_2, \ldots, \beta p)$ are the coefficients for the features,
- x_i is the vector of feature values for observation i,
- y_i is the observed output for observation i,
- λ is the regularization parameter.

5.4.3 MUTUAL INFORMATION

This is a statistical method used to determine the relationship between the variables. It assesses the dependency between each feature and the target variable. Mutual Information can be used to identify the features relevant to the target variable. It helps to understand how much information about the target variable is contained in each feature.

5.4.4 RECURSIVE FEATURE ELIMINATION

It is a popular machine learning feature selection method. It works by repeatedly training a model usually a support vector machine or linear regression model and removing the least significant features at each iteration. This doesn't have a specific mathematical formula. Instead, it's an iterative algorithm that involves the following steps:

I. Fit a model.

II. Rank Features.

III. Eliminate Features.

IV. Repeat.

5.4.5 CHI-SQUARE TEST

It is a statistical technique which is used to ascertain if two category variables have a significant relationship or not. The chi-square (χ^2) statistic for a contingency table is calculated using the formula:

$$\chi 2 = \sum \frac{(O_{ij} - E_{ij})}{E_{ij}}$$

Here:

- Oij is the observed frequency,
- Eij is the expected frequency.

The Eij is calculated as:

Eij = (sum of row i) × (sum of column j) / total sample size

5.5 MODEL CLASSIFICATION

5.5.1 SUPPORT VECTOR MACHINE

This model is a supervised ML algorithm used for classification and regression tasks. It finds the optimal hyperplane that best separates data points into different classes by maximizing the margin between them.

Algorithm 1: SVM Feature Ranking Algorithm

1: Preprocess Data: Clean, normalize, and encode features

2: Feature Selection/Extraction: Select or extract relevant features

3: Prepare Training Data: Split data into training and testing sets

4: Design SVM Model: Define the SVM model, kernel function, regularization parameter (C), and

kernel-specific parameters

5: Train Model: Create a hyperplane that best separates the different classes of network traffic based on

the selected features

6: Model Evaluation: Evaluate the performance of the trained SVM model on the testing data using

metrics

7: Deploy Model: Deploy trained model for real-time IDS

5.5.2 NEURAL NETWORK

Neural networks serve as powerful machine learning models aimed to identify and categorize the suspicious activities in computer networks. These networks, inspired by the structure of biological neural systems, comprise interconnected artificial neurons arranged in layers.

Algorithm 2: Neural Network

1: Preprocess Data: Clean, normalize, and encode features

2: Feature Selection/Extraction: Select or extract relevant features

3: Prepare Training Data: Split data into training and testing sets

4: Design Neural Network Architecture: Define layers, neurons, activation functions

5: Initialize Model: Initialize neural network model

6: Train Model:

7: for epoch = 1 to epochs do

8: Update model weights using optimization algorithm

9: end for

10: Evaluate Model: Evaluate model performance on testing data

11: Deploy Model: Deploy trained model for real-time IDS

5.5.3 RANDOM FOREST

This is an ensemble method used for regression tasks and classification in machine learning. It operates by constructing a multitude of decision trees during training and classification or the average prediction (regression) of the individual trees.

Algorithm 3: Random Forest Classifier Model and Classification Results

1: Input: train x: train features, train y: train target, test x: test features, test y: test target

2: Output: Random forest classifier model, Classification results

3: procedure RandomForestClassifier

4: clasif ← RandomForestclassifier(random state = 42, n jobs = 2, n estimators = 10)

5: Fit the train sets in the model

6: for each m ∈ mli do

7: Train m on train X[k feature] and train y

8: Evaluate test X[k feature] and test y

9: end for

10: Analyze the stored performance

11: end procedure

5.5.4 K-NEAREST NEIGHBOR

KNN is a simple yet powerful supervised machine learning algorithm used for classification and regression tasks. It operates by assigning a class label or predicting a value for a query instance based on the majority class or average value of its k nearest neighbors, determined by a chosen distance metric.

Algorithm 4: KNN Classifier and Classification Results

1: Input: train x: train features, train y: train target, test x: test features, test y: test target

2: Output: KNN classifier, Classification results

3: Function: KNNClassifier(train x, train y, test x)

4: for y ∈ D do

5: Measure distance D(y, x) between y and x.

6: end for

7: Select the subset N from the dataset D, where N contains the k training samples which are the k nearest

neighbors of the test sample x.

8: c_x = arg max ∑ I(c = class(y)).

9: return c_x

CHAPTER 6

SYSTEM REQUIREMENTS

Hardware Requirement

Processor	**:**	Ryzen 4000 Series
RAM	**:**	8 GB
HDD	**:**	10 GB

Software Requirement

Operating System	**:**	Windows 11
Language used	**:**	Python
IDE	**:**	Jupyter Notebook

CHAPTER 7

RESULT AND DISCUSSION

7.1 OVERVIEW

This chapter contains results of various implementation steps of the proposed system with following details:

- Dataset Details
- Anova
- Chi-squared
- LASSO
- Recursive Feature Elimination
- Mutual Information
- Ensemble Feature Selection Technique
- Random Forest
- Logistic regression (LR)
- Neural Network (NN)
- SVM
- KNN
- Ensemble Method (Majority Voting Classifier)

7.1.1 DATASET DETAILS

This research delved into an experimental analysis utilizing NSL-KDD dataset:

The NSL-KDD dataset, a revised version of the original KDD Cup 1999 dataset, is a widely used benchmark dataset in the field of network intrusion detection. It comprises network traffic data generated in a simulated environment, encompassing various types of normal activities as well as specific types of network attacks. The dataset is structured with attributes describing different aspects of network connections, including protocol types, service types, flags, and more. It contains both numerical and categorical features, making it suitable for testing and evaluating different machine learning algorithms. The primary goal of using the NSL-KDD dataset is to develop and train predictive models capable of accurately identifying and classifying different types of network intrusions, such as Denial of Service (DoS), Probe, Remote to Local (R2L), and User to Root (U2R). NSL-KDD dataset contains a total of 125,973 records. Each record represents a network connection instance, and each instance is described by a set of attributes that capture various aspects of the network traffic. NSL-KDD dataset consists of 42 features, or attributes, that describe various aspects of network connections. These features capture information such as protocol types, service types, flag values, number of bytes transferred, duration of the connection, and more.

Category	**Train**	**Test**
Normal	67343	9711
DoS	11656	7458
Probe	45927	2421
U2R	52	200
R2l	995	2754
Total	**125973**	**22544**

Table 7.1.1.1 Classification of NSL-KDD Dataset

7.1.2 ANOVA FEATURE SELECTION ALGORITHM

ANOVA FS method effectively identifies a subset of features that demonstrate notable variations across different classes of network behaviour, crucial for distinguishing between normal network traffic and various types of attacks. Upon examination, the selected features exhibit discernible patterns and

characteristics associated with different classes, as indicated by their ANOVA F-values or scores. Features with higher ANOVA scores are deemed more influential in differentiating between network behaviours, suggesting a stronger association with the target variable. The ANOVA FS results underscore the importance of feature selection in enhancing the performance and interpretability of classification models for network intrusion detection tasks. By focusing on the most informative features, the ANOVA FS method enables the creation of more robust and efficient models, thereby improving the overall efficacy of network security measures.

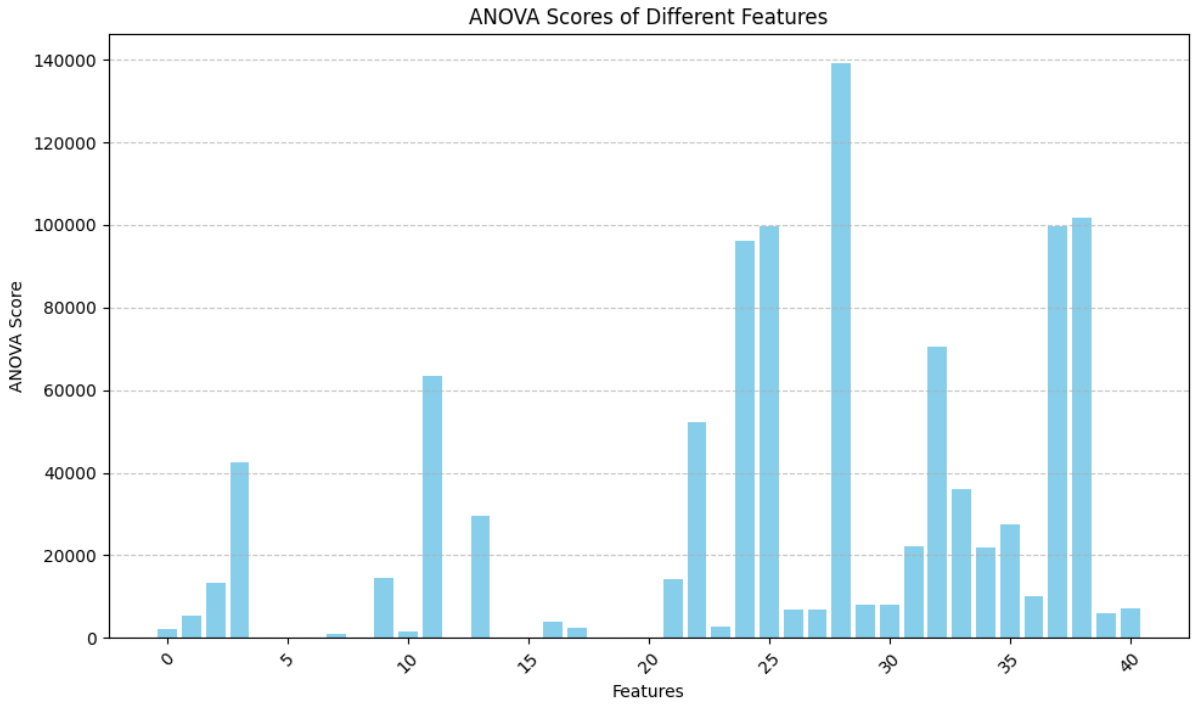

Figure 7.1.2.1 Anova scores for different features in NSL-KDD Dataset

In the **Figure 7.2,** the histogram shows the Anova scores for different features in the NSL-KDD Dataset. In the implementation of the ANOVA feature selection algorithm for the NSL-KDD dataset, the method is configured to select the top 20 features based on their ANOVA scores. This selection process involves identifying the 20 features with the highest ANOVA F-values, indicating their significance in distinguishing between different classes of network behavior. By focusing on the top 20 features, the algorithm prioritizes the most informative attributes for classification, thereby enhancing the efficiency and effectiveness of the subsequent machine learning models.

7.1.3 CHI-SQUARE FEATURE SELECTION ALGORITHM

The implementation of the Chi-square feature selection (FS) method on the NSL-KDD dataset yields significant insights into the relevance of different features for network intrusion detection. Upon analysis, the Chi-square FS method effectively identifies a subset of features that exhibit strong associations with the target variable, which encompasses various classes of network behavior, including normal traffic and different types of attacks. The selected features, chosen based on their Chi-square statistics, demonstrate notable discriminatory power in distinguishing between these classes, thereby facilitating the accurate classification of network traffic. By focusing on features with higher Chi-square statistics, the method prioritizes attributes that capture distinct patterns and characteristics associated with different network behaviours, enhancing the overall effectiveness of intrusion detection systems. The algorithm is configured to select the top 20 features based on their Chi-square statistics. This selection process involves identifying the 20 features with the highest Chi-square statistics, indicating their significance in relation to the target variable, which encompasses various classes of network behaviour, such as normal traffic and different types of attacks. By focusing on the top 20 features, the algorithm prioritizes attributes that exhibit strong associations with the target variable, making them particularly informative for distinguishing between different network behaviours.

7.1.4 LASSO FEATURE SELECTION ALGORITHM

In LASSO feature selection method, a subset of features has been identified as particularly relevant for discerning between various network behaviours, including normal traffic and different types of attacks. Each feature's coefficient obtained from the LASSO model reflects its contribution to the predictive power of the intrusion detection system. The LASSO method effectively shrinks the coefficients of less informative features to zero, effectively removing them from the final feature set. Features with non-zero coefficients are deemed essential for distinguishing between different classes of network behaviour, whereas features with zero coefficients are considered less influential and are effectively pruned from the model.

Here the algorithm has efficiently identified the top 20 most relevant features for network intrusion detection. Through the LASSO method's regularization process, features with the greatest discriminatory power in distinguishing between normal network behavior and various types of attacks have been prioritized. The selection of the top 20 features reflects a strategic

approach to dimensionality reduction, balancing the need for feature parsimony with the imperative to capture critical patterns indicative of network intrusions. By focusing on this subset of features, the LASSO feature selection algorithm streamlines the model's complexity and enhances interpretability while maintaining or even improving its predictive performance.

7.1.5 RECURSIVE FEATURE ELIMINATION ALGORITHM

The Recursive Feature Elimination (RFE) feature selection method applied to the NSL-KDD dataset has yielded insightful results that contribute significantly to enhancing network intrusion detection capabilities. Through the iterative process of selecting and eliminating features based on their importance, RFE systematically identifies the most informative attributes for distinguishing between normal network behaviour and various types of attacks.

RFE feature selection process provides valuable insights into the underlying structure of network traffic and the indicators of potential intrusions. By iteratively evaluating feature importance and relevance, RFE highlights attributes that exhibit consistent predictive power across multiple iterations.

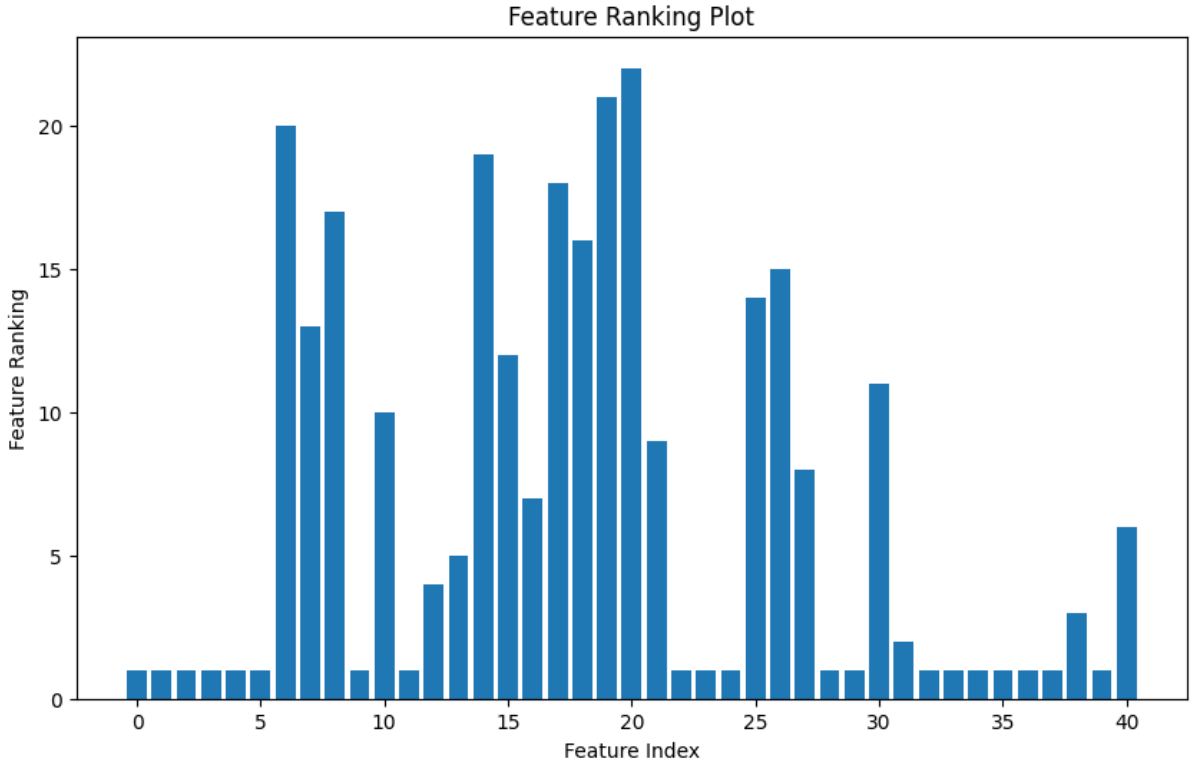

Figure 7.1.5.1 RFE Feature Ranking Plot for all Features

The **Figure 7.1.5.1** shows the RFE Feature ranking for all features in the NSL-KDD Dataset. The algorithm has successfully identified the top 20 most informative features for network intrusion detection. RFE systematically evaluates the importance of each feature by iteratively training the model and selecting or eliminating features based on their impact on model performance. By prioritizing features with the highest discriminatory power, RFE ensures that the selected subset encapsulates the most relevant attributes for distinguishing between normal network behaviour and various types of attacks.

7.1.6 MUTUAL INFORMATION ALGORITHM

Mutual Information quantifies the amount of information shared between features and the target variable, making it effective in identifying features that are highly informative for distinguishing between normal network behavior and various types of intrusions. By assessing the mutual dependence between each feature and the target variable, MI prioritizes attributes with significant discriminatory power, thereby refining the feature subset to include those most relevant for intrusion detection.

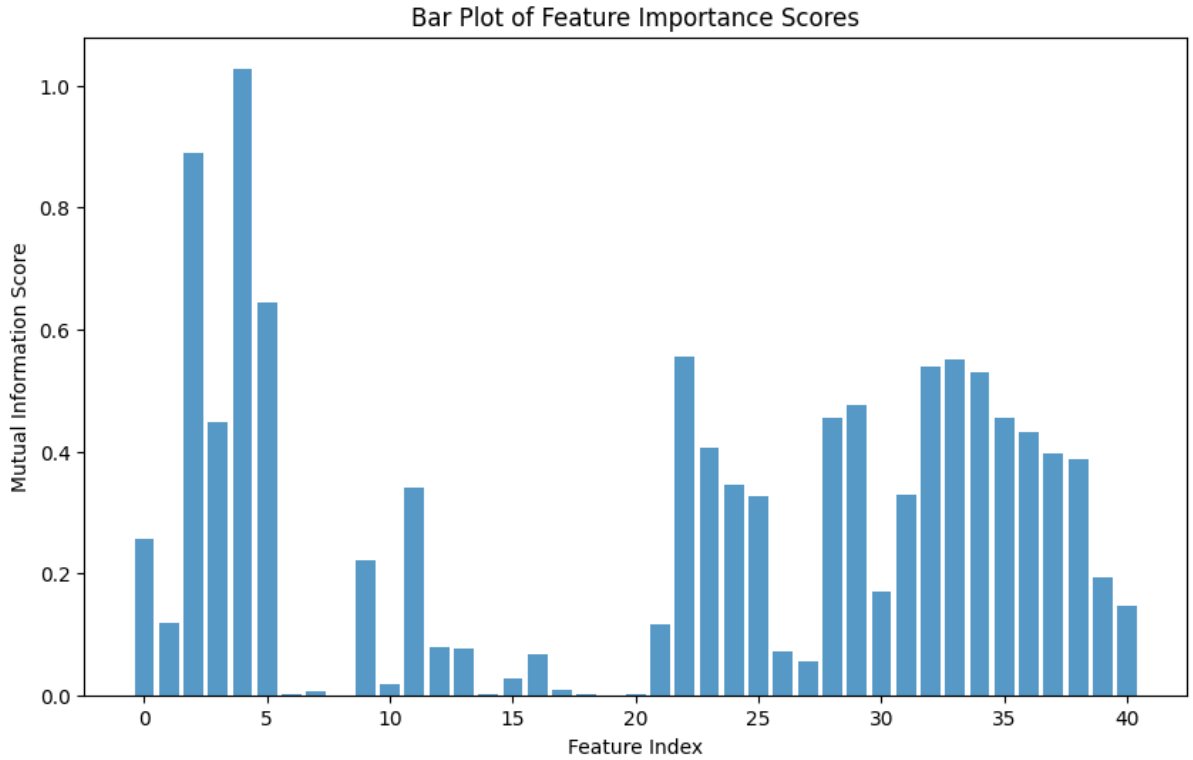

Figure 7.1.6.1 Mutual Information score of all features in NSL-KDD Dataset

Mutual Information feature selection results offer valuable insights into the underlying structure of network traffic and the indicators of potential intrusions. By prioritizing features with high mutual information content, MI highlights attributes that exhibit strong predictive power in detecting network anomalies.

The algorithm has effectively identified the top 20 most informative features for network intrusion detection. By quantifying the amount of information shared between each feature and the target variable, MI prioritizes attributes that are highly relevant for distinguishing between normal network behavior and various types of intrusions.

7.1.7 ENSEMBLE FEATURE SELECTION ALGORITHM

In this research we have adopted a comprehensive approach to feature selection by leveraging ensemble techniques. Specifically, we have devised a strategy where features are selected based on their occurrence across multiple feature selection algorithms. This approach aims to capitalize on the collective wisdom of various algorithms to identify the most informative features for our task of network intrusion detection.

Selected Features
dst_host_same_srv_rate
srv_serror_rate
logged_in
dst_host_srv_serror_rate
diff_srv_rate
dst_host_srv_count
dst_host_diff_srv_rate
dst_host_serror_rate
dst_host_count
same_srv_rate
count
serror_rate
dst_host_same_src_port_rate

Table 7.1.7.2 Features selected by Ensemble Method

By selecting features that consistently appear in more than three different algorithms, we effectively create an ensemble of feature selection methods. This ensemble approach allows us to mitigate the limitations inherent in individual algorithms and enhance the robustness and reliability of our feature selection process.

The ensemble method we employed has identified a total of 13 features shown in **Table 7.5** deemed most critical for our network intrusion detection task. This selection represents the culmination of our ensemble approach, which combines the outputs of multiple feature selection algorithms to distill a concise and informative set of attributes.

7.1.8 RANDOM FOREST MODEL

Receiver Operating Characteristic (ROC) Curve of Random Forest Model is shown in Figure 6.4

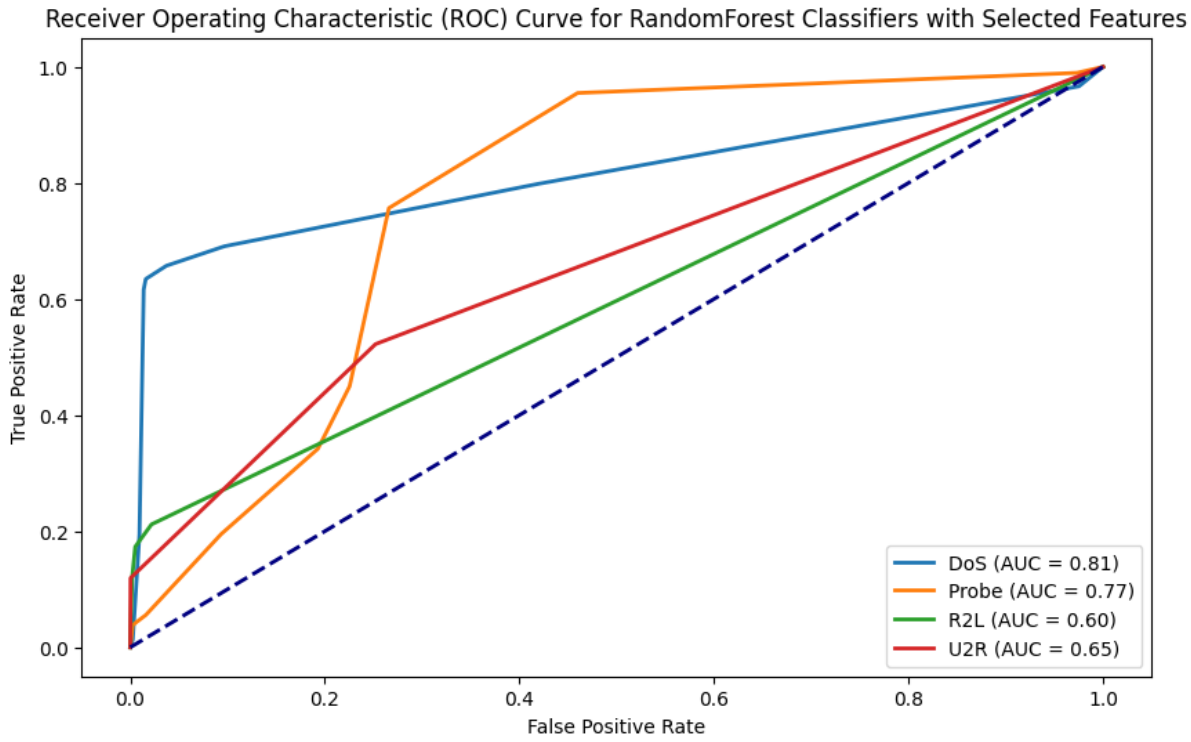

Figure 7.1.8.1 ROC Curve of Random Forest Model

Random Forest model works by creating multiple decision trees during training and outputting the mode of the classes (for classification) or the average prediction (for regression) of the individual trees. This ensemble approach helps to improve the model's accuracy and generalization by reducing overfitting. Random Forest is known for its ability to handle large amounts of data with high dimensionality and to deal with missing values and outliers effectively. It's a powerful tool for detecting and classifying network intrusions in the NSL-KDD dataset.

7.1.9 LOGISTIC REGRESSION MODEL

Receiver Operating Characteristic (ROC) Curve of Logistic Regression Model is shown in Figure 7.7

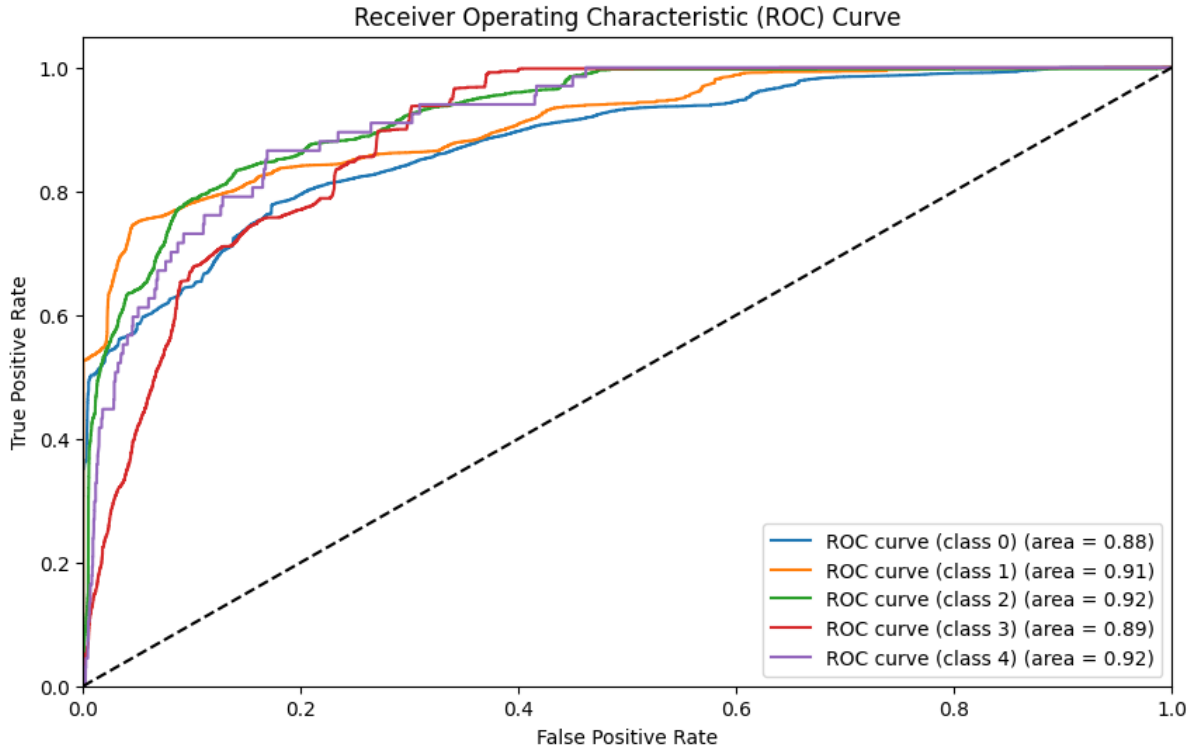

Figure 7.1.9.1 ROC Curve of Logistic Regression Model

Logistic Regression model is a statistical method used for classification tasks, such as identifying network intrusions in the NSL-KDD dataset. Despite its name, it's used for classification rather than regression. Logistic Regression estimates the probability that a given input belongs to a certain class, and then makes a prediction based on that probability. It's a simple yet effective algorithm that works well with linearly separable data and provides interpretable results by giving insights into the influence of the input features on the classification outcome. In the context of intrusion detection, Logistic Regression can be a valuable tool for identifying and categorizing potential network attacks in the NSL-KDD dataset.

7.1.10 NEURAL NETWORK MODEL

Receiver Operating Characteristic (ROC) Curve of Neural Network Model is shown in Figure 7.8

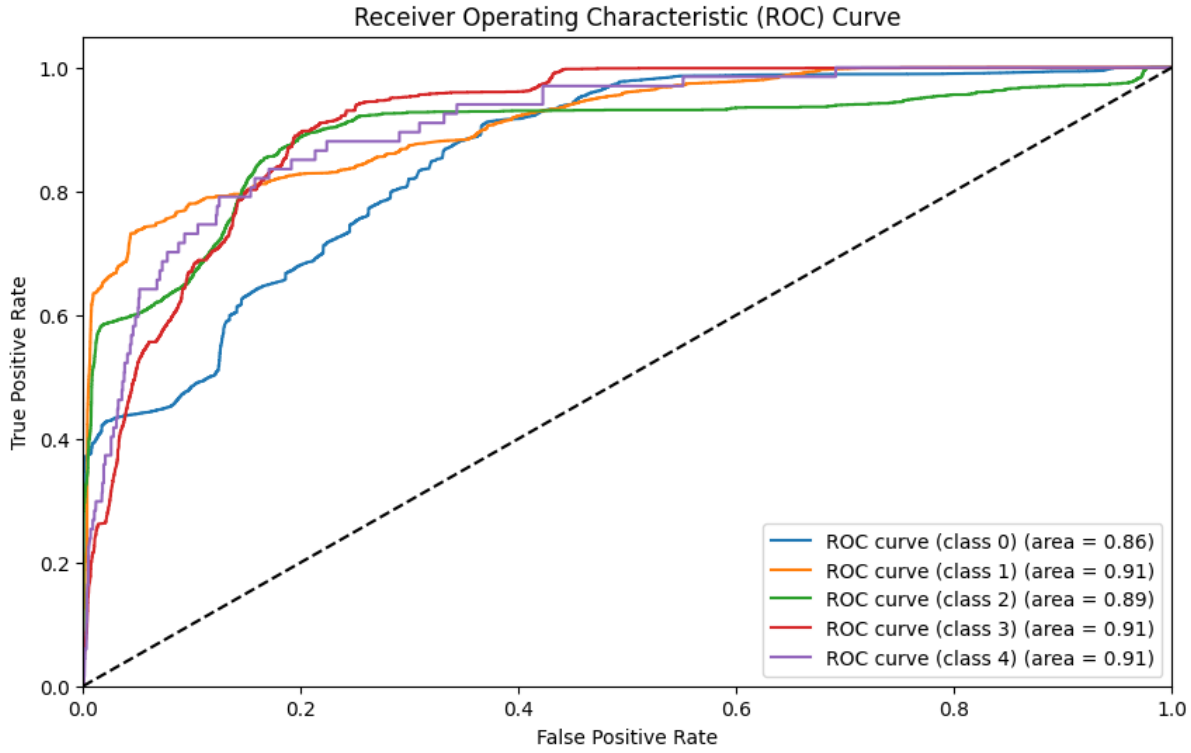

Figure 7.1.10.1 ROC Curve of Neural Network Model

Models of neural networks offer an advanced method for identifying unusual activity in network traffic. By utilizing the concepts of artificial neural networks, these models are capable of identifying intricate patterns and relationships present in the data, which allows them to distinguish between legitimate and malicious network activity. Given their prowess with high-dimensional, non-linear data, neural networks are an excellent fit for the complex, dynamic field of network traffic analysis. Neural Networks increase their predictive performance during training by iteratively modifying the weights and biases of interconnected neurons in an effort to reduce classification errors. Because of its flexibility and capacity for learning, neural networks can adjust to dynamic network conditions and shifting assault tactics, strengthening the resistance of intrusion detection systems.

7.1.11 SUPPORT VECTOR MACHINE MODEL

Receiver Operating Characteristic (ROC) Curve of Neural Network Model is shown in Figure 7.8

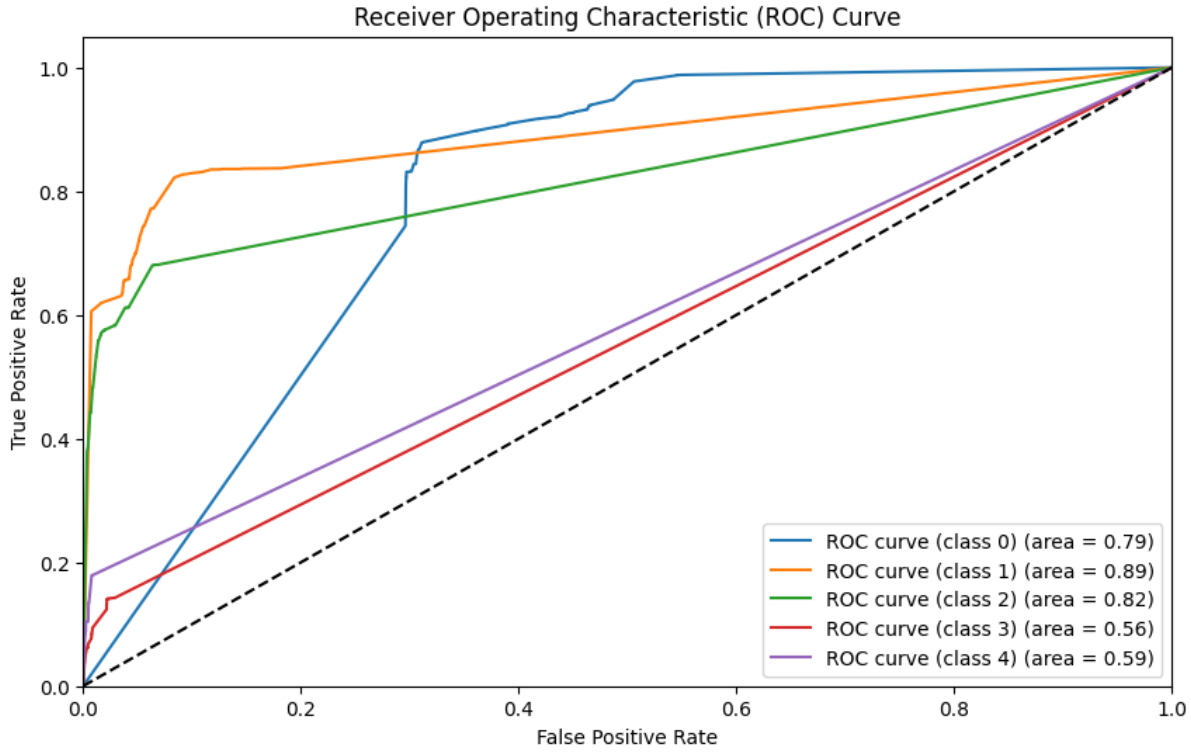

Figure 7.1.11.1 ROC Curve of SVM Model

Network traffic can be classified into two categories: harmful and normal using Support Vector Machine (SVM) models, which provide a reliable and efficient method for doing so. Because SVMs represent excellent at finding intricate patterns and connections in high-dimensional data, they are a good fit for the dynamic and varied world of network traffic analysis. Superior Volumetric Models (SVMs) are designed to create an ideal hyperplane that maximizes the margin between several classes of data points in order to improve classification accuracy and generalization performance. Additionally, by utilizing kernel functions, SVMs can handle both linear and non-linear data transformations and are resistant to overfitting.

7.1.12 K NEAREST NEIGHBOR MODEL

Receiver Operating Characteristic (ROC) Curve of K nearest neighbor Model is shown in Figure 7.9

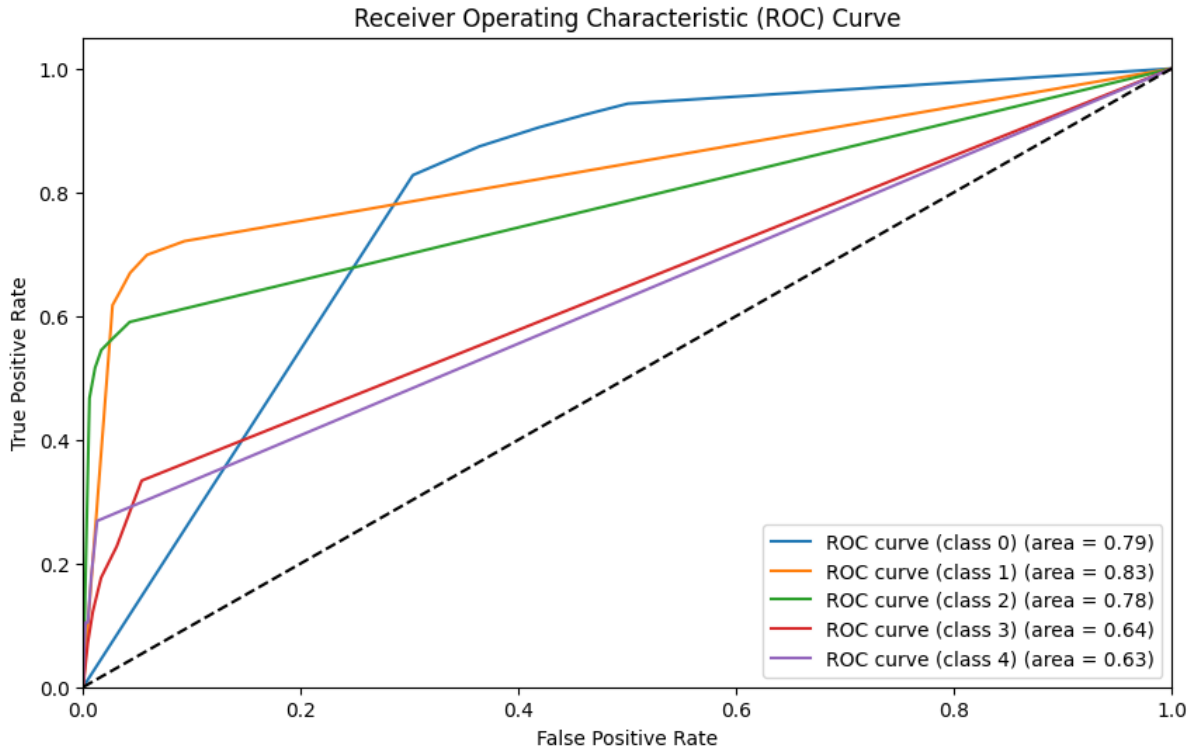

Figure 7.1.12.1 ROC Curve of K Nearest Neighbor Model

K-Nearest Neighbors (KNN) model offers a simple yet effective approach for classifying network traffic based on similarity measures. KNN operates by identifying the k nearest neighbors of a given data point in the feature space and classifying the data point based on the most prevalent class among its neighbors. This proximity-based classification method makes KNN particularly suitable for detecting anomalies in network traffic, as it can capture subtle patterns and similarities between instances. Additionally, KNN is non-parametric and does not make strong assumptions about the underlying distribution of the data, allowing it to adapt well to diverse network environments. After training a KNN model on the NSL-KDD dataset, the model can be used to predict the class labels (e.g., normal, or intrusive) for unseen instances of network traffic. The output would be a set of predicted labels corresponding to each instance in the test dataset.

7.1.13 ENSEMBLE MAJORITY VOTING CLASSIFIER

Receiver Operating Characteristic (ROC) Curve of Ensemble Model is shown in Figure 7.5

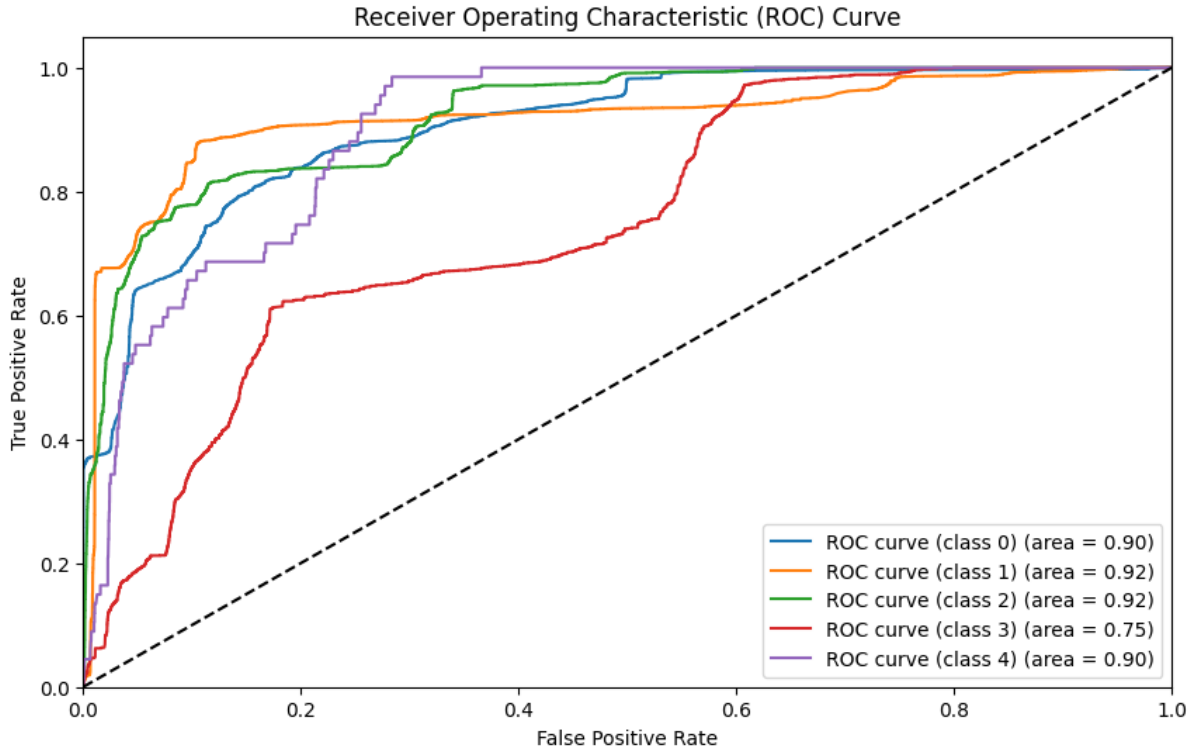

Figure 7.1.13.1 ROC Curve of Ensemble model

In our intrusion detection system (IDS) implementation using the NSL-KDD dataset, we employed an ensemble majority voting classifier to enhance the accuracy and robustness of our model. This ensemble method combines the predictions of multiple base classifiers, such as Random Forest, Logistic Regression, K-Nearest Neighbors, Support Vector Machine, and Neural Network, using a majority voting approach. The result of our ensemble majority voting classifier reflects a consensus decision made by the constituent classifiers. By aggregating the predictions of diverse classifiers, each with its own strengths and weaknesses, our ensemble model benefits from improved generalization performance and resilience to overfitting. The ensemble majority voting classifier demonstrates its efficacy in accurately detecting and classifying instances of network intrusions within the NSL-KDD dataset.

7.2 EVALUATION METRICES FOR ALL MODELS

Several important metrics are used in the classification model assessment for Intrusion Detection System in order to fully evaluate its effectiveness. All Models are evaluated on the following metrics. They are Accuracy, Precision, Recall and F1 Score. Let us see about the brief explanation of the above metrics.

- **Accuracy**

The overall correctness of the categorization model is measured by accuracy. It is computed as the proportion of accurately anticipated cases to all instances.

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{6.1}$$

- **Precision**

The precision of a model lies in its capacity to accurately identify positive cases from those it anticipated to be positive. When the model indicates that an instance is positive, it serves as a gauge of its accuracy.

$$\mathrm{PREC} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{6.2}$$

- **Recall**

Recall quantifies the model's accuracy in identifying every positive instance. It is the proportion of all real positive instances to all correctly projected positive instances.

$$\mathrm{REC} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{6.3}$$

- **F1 Score**

The harmonic mean of recall and precision is the F1 Score. In instances where there is an imbalance in the class distribution, it offers a balance between recall and precision.

$$\text{F1 Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (6.4)$$

Models	**Accuracy**	**Precision**	**Recall**	**F1-Score**
SVM	70.608	63.004	70.608	65.519
RF	94.744	82.208	78.523	78.604
NN	72.422	75.988	72.422	69.494
KNN	93.626	82.736	76.520	77.196
LR	70.941	73.186	70.941	70.165
Ensemble	**95.795**	**83.738**	**82.654**	**81.496**

Table 7.2.1: Performance Evaluation of All Models

CHAPTER 8

CONCLUSION AND FUTURE WORK

8.1 CONCLUSION

In conclusion, our study has demonstrated the effectiveness of integrating ensemble feature selection (EnFS) and ensemble machine learning techniques as the detection mechanism within an Intrusion Detection System (IDS) for identifying network anomalies. Through comprehensive experimentation, we initially evaluated the feature sets produced by four feature selection techniques within the ensemble feature selection framework. By aggregating these feature sets using majority voting, we identified a robust feature subset that consistently outperformed individual feature selection methods. Our comparative analysis revealed that the feature set derived from the EnFS methodology consistently yielded superior outcomes compared to standalone feature sets. Moreover, we identified the best-performing model for integration into any IDS by leveraging this feature set within our ensemble supervised machine learning framework. Our experiments demonstrated that ensemble models trained on the EnFS feature set exhibited significantly better performance compared to single models, achieving an improvement of approximately 80%.

8.2 FUTURE WORK

Future research on Intrusion Detection System could go in a number of directions like,

- **Integration of Advanced Machine Learning Techniques:** Investigating the application of cutting-edge machine learning techniques such as deep learning, reinforcement learning, and adversarial learning to IDS could enhance their ability to detect sophisticated and evolving threats. Exploring novel architectures and algorithms tailored to the unique characteristics of network traffic data could lead to significant improvements in detection accuracy and robustness.

- **Dynamic Adaptation and Self-Learning Mechanisms**: Developing IDS that can dynamically adapt to changing network environments and emerging threats in real-time is crucial. Future research could focus on designing self-learning systems capable of autonomously updating their detection mechanisms based on continuous feedback and environmental changes. This could involve leveraging techniques such as online learning, transfer learning, and ensemble learning to enhance IDS adaptability and resilience.

- **Behavioral Analysis and Anomaly Detection**: Investigating techniques for more nuanced behavioral analysis and anomaly detection could improve the ability of IDS to detect subtle deviations from normal network behavior. Research in this area could involve the development of advanced statistical models, graph-based approaches, and unsupervised learning algorithms capable of identifying anomalous patterns indicative of sophisticated attacks or insider threats.

- **Adversarial Attack and Defense Strategies**: With the rise of adversarial attacks targeting IDS, future research could focus on developing robust defense mechanisms to mitigate the impact of such attacks. This could involve studying adversarial machine learning techniques, game-theoretic approaches, and adversarial training strategies to enhance IDS resilience against evasion and poisoning attacks.

- **Integration of Threat Intelligence and Contextual Information**: Enhancing IDS capabilities through the integration of threat intelligence feeds, contextual information, and domain-specific

knowledge could provide valuable insights into emerging threats and attack trends. Future research could explore methods for incorporating external threat feeds, security advisories, and network context data into IDS decision-making processes to improve detection accuracy and response effectiveness.

- **Scalability and Efficiency:** As network traffic volumes continue to grow exponentially, scalability and efficiency are becoming increasingly important considerations for IDS. Future research could focus on developing lightweight, scalable IDS architectures capable of processing large-scale network data streams in real-time. This could involve leveraging distributed computing, stream processing, and hardware acceleration techniques to optimize IDS performance and resource utilization.

APPENDIX

CODING

Loading the Dataset

```
train_url = '/content/NSL_KDD_Train.csv'
test_url = '/content/NSL_KDD_Test.csv'
col_names = ["duration","protocol_type","service","flag","src_bytes",
"dst_bytes","land","wrong_fragment","urgent","hot","num_failed_logins",
"logged_in","num_compromised","root_shell","su_attempted","num_root",
‘num_file_creations","num_shells","num_access_files","num_outbound_cmds",
"is_host_login","is_guest_login","count","srv_count","serror_rate",
"srv_serror_rate","rerror_rate","srv_rerror_rate","same_srv_rate",
"diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_host_srv_count",
        "dst_host_same_srv_rate","dst_host_diff_srv_rate","dst_host_same_s
        rc_port_rate
        "dst_host_srv_diff_host_rate","dst_host_serror_rate","dst_host_srv_s
        error_rate"
"dst_host_rerror_rate","dst_host_srv_rerror_rate","label"]
df = pd.read_csv(train_url,header=None, names = col_names)
df_test = pd.read_csv(test_url, header=None, names = col_names)
print('Dimensions of the Training set:',df.shape)
print('Dimensions of the Test set:',df_test.shape)
```

Importing all the required libraries

```
import pandas as pd
import numpy as np
import sys
import sklearn
import io
```

```
import random
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
from imblearn.over_sampling import SMOTE
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
import pandas as pd
from sklearn.feature_selection import SelectPercentile, f_classif
from sklearn.feature_selection import RFE
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import MinMaxScaler
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.feature_selection import SelectKBest, mutual_info_regression
from sklearn.linear_model import Lasso
from sklearn.feature_selection import SelectFromModel
from collections import Counter
from sklearn.model_selection import cross_val_score
from sklearn import metrics
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
import pandas as pd
from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import VotingClassifier
```

Label distribution Training and Testing set

```
print('Label distribution Training set:')
print(df['label'].value_counts())
print()
print('Label distribution Test set:')
print(df_test['label'].value_counts())
print('Training set:')
print('Training set:')
for col_name in df.columns:
  if df[col_name].dtypes == 'object' :
    unique_cat = len(df[col_name].unique())
    print("Feature '{col_name}' has {unique_cat}
categories".format(col_name=col_name, unique_cat=unique_cat))
print()
print('Distribution of categories in service:')

print(df['service'].value_counts().sort_values(ascending=False).head())
```

Finding categorical columns in the dataset

```
print('Test set:')
for col_name in df_test.columns:
  if df_test[col_name].dtypes == 'object' :
    unique_cat = len(df_test[col_name].unique())
    print("Feature '{col_name}' has {unique_cat}
categories".format(col_name=col_name, unique_cat=unique_cat))
```

Preprocessing using Label Encoding

```
categorical_columns=['protocol_type', 'service', 'flag']
```

```
df_categorical_values = df[categorical_columns]

testdf_categorical_values = df_test[categorical_columns]

df_categorical_values.head()

df_categorical_values_enc=df_categorical_values.apply(LabelEncoder().fit_transform)

print(df_categorical_values.head())

print('--------------------')

print(df_categorical_values_enc.head())

# test set

testdf_categorical_values_enc=testdf_categorical_values.apply(LabelEncoder().fit_transform)

df['flag']=df_categorical_values_enc['flag']

df['protocol_type']=df_categorical_values_enc['protocol_type']

df['service']=df_categorical_values_enc['service']

df_test['flag']=testdf_categorical_values_enc['flag']

df_test['protocol_type']=testdf_categorical_values_enc['protocol_type']

df_test['service']=testdf_categorical_values_enc['service']

print(df_categorical_values_enc.columns)

print(df.shape)

print(df_test.shape)

print(df.columns)

print(df_test.columns)
```

Changing the Label for classification of Attacks

```
labeldf=df['label']

labeldf_test=df_test['label']

# change the label column
```

```
newlabeldf=labeldf.replace({ 'normal' : 0, 'neptune' : 1 ,'back': 1, 'land': 1, 'pod': 1, 'smurf': 1, 'teardrop': 1,'mailbomb': 1, 'apache2': 1, 'processtable': 1, 'udpstorm': 1, 'worm': 1, 'ipsweep' : 2,'nmap' : 2,'portsweep' : 2,'satan' : 2,'mscan' : 2,'saint' : 2

  ,'ftp_write': 3,'guess_passwd': 3,'imap': 3,'multihop': 3,'phf': 3,'spy': 3,'warezclient': 3,'warezmaster': 3,'sendmail': 3,'named': 3,'snmpgetattack': 3,'snmpguess': 3,'xlock': 3,'xsnoop': 3,'httptunnel': 3,'buffer_overflow': 4,'loadmodule': 4,'perl': 4,'rootkit': 4,'ps': 4,'sqlattack': 4,'xterm': 4})

newlabeldf_test=labeldf_test.replace({ 'normal' : 0, 'neptune' : 1 ,'back': 1, 'land': 1, 'pod': 1, 'smurf': 1, 'teardrop': 1,'mailbomb': 1, 'apache2': 1, 'processtable': 1, 'udpstorm': 1, 'worm': 1,'ipsweep' : 2,'nmap' : 2,'portsweep' : 2,'satan' : 2,'mscan' : 2,'saint' : 2,'ftp_write': 3,'guess_passwd': 3,'imap': 3,'multihop': 3,'phf': 3,'spy': 3,'warezclient': 3,'warezmaster': 3,'sendmail': 3,'named': 3,'snmpgetattack': 3,'snmpguess': 3,'xlock': 3,'xsnoop': 3,'httptunnel': 3,'buffer_overflow': 4,'loadmodule': 4,'perl': 4,'rootkit': 4,'ps': 4,'sqlattack': 4,'xterm': 4})

# put the new label column back

df['label'] = newlabeldf

df_test['label'] = newlabeldf_test

# Define labels to be dropped

to_drop_labels = [0, 1, 2, 3, 4]

# Filter out rows with labels to be dropped

filtered_df =
balanced_train_df[balanced_train_df['label'].isin(to_drop_labels)]

filtered_df_test = df_test[df_test['label'].isin(to_drop_labels)]

print('Train:')

print('Dimensions of filtered data:', filtered_df.shape)

print()

print('Test:')

print('Dimensions of filtered test data:', filtered_df_test.shape)

X = filtered_df.drop('label',axis=1)

Y = filtered_df.label
```

```
X_test = filtered_df_test.drop('label',axis=1)
Y_test = filtered_df_test.label
colNames=list(X)
colNames_test=list(X_test)
```

Feature Selection Algorithms

Anova Algorithm

```
np.seterr(divide='ignore', invalid='ignore');
selector=SelectPercentile(f_classif, percentile=50)
X_new = selector.fit_transform(X,Y)
X_new.shape
true=selector.get_support()
colindex1=[i for i, x in enumerate(true) if x]
colname1=list( colNames[i] for i in colindex1 )
```

Recursive Feature Elimination

```
clf = RandomForestClassifier(n_estimators=10,n_jobs=2)
rfe = RFE(estimator=clf, n_features_to_select=20, step=1)
rfe.fit(X, Y.astype(int))
X_new=rfe.transform(X)
true=rfe.support_
colindex2=[i for i, x in enumerate(true) if x]
colname2=list(colNames[i] for i in colindex2)
colname2
```

Chi-Square Test

```
scaler = MinMaxScaler()

X_new = scaler.fit_transform(X)

selector_chi2 = SelectKBest(chi2, k=20)

X_new_chi2 = selector_chi2.fit_transform(X_new, Y)

colindex3 = selector_chi2.get_support(indices=True)

colname3 = [colNames[i] for i in colindex3]
```

Mutual Information Algorithm

```
# Create a SelectKBest instance with mutual information

selector_mi = SelectKBest(mutual_info_regression, k=20)  # Select the top 20 features based on mutual information

# Fit and transform the data for feature selection based on mutual information

X_new_mi = selector_mi.fit_transform(X, Y)

colindex5 = selector_mi.get_support(indices=True)

colname5 = [colNames[i] for i in colindex5]
```

LASSO Algorithm

```
lasso = Lasso(alpha=0.001)  # You can adjust the alpha parameter for regularization strength

# Use SelectFromModel to select features based on Lasso coefficients

selector_lasso = SelectFromModel(lasso, max_features=20)  # Select only 20 features

# Fit and transform the data for DoS

X_new = selector_lasso.fit_transform(X, Y)

colindex4 = selector_lasso.get_support(indices=True)

colname4 = [colNames[i] for i in colindex4]
```

Ensemble Feature Selection Algorithm

```
def common_in_three_or_more(lists):
# Count occurrences of each number across all lists
count = Counter()
for lst in lists:
count.update(set(lst))
# Select numbers that are common in at least three lists
common_numbers = [num for num, freq in count.items() if freq >= 4]
return common_numbers
# Example usage:
list1 = colname1
list2 = colname2
list3 = colname3
list4 = colname4
list5 = colname5
lists = [list1, list2, list3, list4, list5]
result = common_in_three_or_more(lists)
colindex=l
colname=result
X_rfe = X.iloc[:, colindex]
print(X_rfe)
print(colname)
```

Model Classification

Random Forest Model

```
clf_rf=RandomForestClassifier(n_estimators=10,n_jobs=1)
```

```
clf_rf.fit(X_rfe, Y.astype(int))

X_test2 = X_test.iloc[:, colindex]

print(X_test2)

Y_pred2=clf_rf.predict(X_test2)

# Create confusion matrix

pd.crosstab(Y_test, Y_pred2, rownames=['Actual attacks'],
colnames=['Predicted attacks'])
```

Neural Network Model

```
# Create a Multi-layer Perceptron (MLP) neural network classifier

clf_nn = MLPClassifier(hidden_layer_sizes=(100,), activation='relu',
solver='adam', random_state=42)

# Train the neural network classifier

clf_nn.fit(X_rfe, Y.astype(int))

# Predict using the trained classifier

Y_pred_nn = clf_nn.predict(X_test2)

# Create a DataFrame from the confusion matrix

pd.crosstab(Y_test, Y_pred_nn, rownames=['Actual attacks'],
colnames=['Predicted attacks'])
```

KNN Model

```
clf_KNN=KNeighborsClassifier()

clf_KNN.fit(X_rfe, Y.astype(int))

Y_pred2=clf_KNN.predict(X_test2)

# Create confusion matrix

pd.crosstab(Y_test, Y_pred2, rownames=['Actual attacks'],
colnames=['Predicted attacks'])
```

SVM Model

```
clf_SVM = SVC()
clf_SVM.fit(X_rfe, Y.astype(int))
# Make predictions on the test data
Y_pred2 = clf_SVM.predict(X_test2)
# Create confusion matrix
confusion_mat = confusion_matrix(Y_test, Y_pred2)
# Convert confusion matrix to pandas DataFrame for better visualization
confusion_df = pd.DataFrame(confusion_mat, index=['Actual attacks'],
columns=['Predicted attacks'])
print(confusion_df)
```

Logistic Regression Model

```
# Create Logistic Regression classifiers
clf_LR = LogisticRegression(random_state=0, max_iter=1000)
Y_pred2 = clf_LR.predict(X_test2)
pd.crosstab(Y_test, Y_pred2, rownames=['Actual attacks'],
colnames=['Predicted attacks'])
```

Ensemble Majority Voting Classifier

```
# Initialize classifiers with probability=True
clf_SVM = SVC(probability=True)
clf_LR = LogisticRegression(random_state=0, max_iter=1000)
clf_voting = VotingClassifier(estimators=[
    ('rf', clf_rf),
    ('knn', clf_KNN),
    ('svm', clf_SVM),
    ('NN', clf_nn),
```

```
    ('LR', clf_LR)
], voting='soft')
# Fit the VotingClassifier
clf_voting.fit(X_rfe, Y.astype(int))
# Predict using the VotingClassifier
Y_pred_voting = clf_voting.predict(X_test2)
# Create confusion matrix
confusion_matrix_voting = pd.crosstab(Y_test, Y_pred_voting, rownames=['Actual attacks'], colnames=['Predicted attacks'])
# Print the confusion matrix
print(confusion_matrix_voting)
from sklearn.metrics import precision_score, recall_score, f1_score
# Compute metrics with appropriate averaging
accuracy = cross_val_score(clf_voting, X_test2, Y_test, cv=10, scoring='accuracy')
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
precision = cross_val_score(clf_voting, X_test2, Y_test, cv=10, scoring='precision_macro')
print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() * 2))
recall = cross_val_score(clf_voting, X_test2, Y_test, cv=10, scoring='recall_macro')
print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() * 2))
f1 = cross_val_score(clf_voting, X_test2, Y_test, cv=10, scoring='f1_macro')
print("F-measure: %0.5f (+/- %0.5f)" % (f1.mean(), f1.std() * 2))
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
from sklearn.preprocessing import label_binarize
```

```
# Predict probabilities for each class
Y_probs_voting = clf_voting.predict_proba(X_test2)
# Binarize the labels
y_test_binarized = label_binarize(Y_test, classes=np.unique(Y_test))
# Compute ROC curve and ROC area for each class
fpr = dict()
tpr = dict()
roc_auc = dict()
n_classes = len(np.unique(Y_test))
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test_binarized[:, i], Y_probs_voting[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])
# Plot ROC curve for each class
plt.figure(figsize=(10, 6))
for i in range(n_classes):
    plt.plot(fpr[i], tpr[i], label='ROC curve (class {}) (area =
{:0.2f})'.format(i, roc_auc[i]))
plt.plot([0, 1], [0, 1], 'k--')  # Plot diagonal line
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```

REFERENCES:

1. Das, S., Mahfouz, A.M., Venugopal, D., Shiva, S., 2019. Ddos intrusion detection through machine learning ensemble, in: 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C), IEEE.

URL: http://dx.doi.org/ 10.1109/QRS-C.2019.00090,
doi:10.1109/qrs-c.2019.00090.

2. Das, S., Venugopal, D., Shiva, S., 2020. A Holistic Approach for Detecting DDoS Attacks by Using Ensemble Unsupervised Machine Learning. Springer International Publishing. p. 721–738.

URL: http://dx.doi.org/10.1007/ 978-3-030-39442-4_53,
doi:10.1007/978-3-030-39442-4_53.

3. Gao, X., Shan, C., Hu, C., Niu, Z., Liu, Z., 2019. An adaptive ensemble machine learning model for intrusion detection. IEEE Access 7, 82512–82521.

URL: http://dx.doi.org/10.1109/ACCESS.2019.2923640,
doi:10.1109/access.2019.2923640.

4. Moustafa, N., Slay, J., 2015. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set), in: 2015 Military Communications and Information Systems Conference (MilCIS), IEEE.

URL: http://dx.doi.org/10.1109/ MilCIS.2015.7348942,
doi:10.1109/milcis.2015.7348942.

5. Pham, N.T., Foo, E., Suriadi, S., Jeffrey, H., Lahza, H.F.M., 2018. Improving performance of intrusion detection system using ensemble methods and feature selection, in: Proceedings of the Australasian Computer Science Week Multicon- ference, ACM.

URL: http://dx.doi.org/10.1145/3167918. 3167951,
doi:10.1145/3167918.3167951.

6. Sagi, O., Rokach, L., 2018. Ensemble learning: A survey. WIREs Data Mining and Knowledge Discovery 8.

URL: http://dx.doi. org/10.1002/widm.1249,
doi:10.1002/widm.1249.

7. Sharafaldin, I., Habibi Lashkari, A., Ghorbani, A.A., 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization, in: Proceedings of the 4th International Confer- ence on Information Systems Security and Privacy, SCITEPRESS - Science and Technology Publications.
URL: http://dx.doi. org/10.5220/0006639801080116,
doi:10.5220/0006639801080116.

8. Sudar, K.M., Deepalakshmi, P., 2020. Comparative study on ids using machine learning approaches for software defined networks. International Journal of Intelligent Enterprise 7, 15.
URL: http://dx.doi.org/10.1504/IJIE.2020.104642,
doi:10. 1504/ijie.2020.104642.

9. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A., 2009. A detailed analysis of the kdd cup 99 data set, in: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, IEEE.
URL: http://dx.doi.org/10.1109/CISDA.2009.5356528,
doi:10.1109/cisda.2009.5356528.

10. Vinayakumar, R., Alazab, M., Soman, K.P., Poornachandran, P., Al-Nemrat, A., Venkatraman, S., 2019. Deep learning approach for intelligent intrusion detection system. IEEE Access 7, 41525–41550.
URL: http://dx.doi.org/10.1109/ACCESS.2019.2895334,
doi:10.1109/access.2019.2895334.

11. Vinayakumar, R., Alazab, M., Srinivasan, S., Pham, Q.V., Padan- nayil, S.K., Simran, K., 2020. A visualized botnet detection system based deep learning for the internet of things networks of smart cities. IEEE Transactions on Industry Applications 56, 4436–4456.
URL: http://dx.doi.org/10.1109/TIA.2020.2971952,
doi:10.1109/tia.2020.2971952.

12. Vinayakumar, R., Soman, K.P., Poornachandran, P., 2017. Evaluating effectiveness of shallow and deep networks to intrusion de- tection system, in: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE.