Project Title:

Explainable Network Intrusion Detection System Based on Ensemble Learning

Submitted By:

Sonia Yadav

Under the Supervision of

Prof. Sugata Gangopadhyay

Department of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY ROORKEE

Certificate

It is certified that the work contained in the project report titled **"Explainable Network Intrusion Detection System Based on Ensemble Learning "**, by **"Sonia Yadav"**, has been carried out under my/our supervision and that this work has not been submitted elsewhere for a degree.

Signature of Supervisor(s)

Prof. Sugata Gangopadhyay

Computer Science Department

I.I.T. Roorkee

Feb, 2025

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)

_____

(Name of the student)

_____

(Roll No.)

_____

Date: _____

Abstract

As network infrastructures grow in complexity with time, cyber attacks and intrusions pose real threats. In a bid to counteract such threats, powerful Network Intrusion Detection Systems (NIDS) that harness the power of Machine Learning (ML) have been deeply studied. Despite that, their usefulness mostly hinges on the quality of feature choice, scalability, and predictability explanation. [2]This piece is aimed at improving intrusion detection through preprocessing and investigating the UNR-IDD (University of Nevada - Reno Intrusion Detection Dataset) and NSL - KDD dataset. The NSL-KDD data set is an enhanced version of its predecessor KDD"99 data set. We have analyzed the NSL-KDD data set and utilized to investigate the performance of different classification algorithms in identifying the anomalies in the network traffic patterns. We employ multiple feature selection methods, and testing different ML models, such as Logistic Regression, Random Forest,SVM, KNN, and Decision Trees on both datasets. The research also investigates the scalability and computational complexity of these models, suggesting an optimal method based on performance measures like accuracy, precision, recall, and F1-score. The interpretability of the suggested model is also improved using Local Interpretable Model-agnostic Explanations (LIME) and ensemble learning, offering insights into model decisions. The results contribute to the development of a more stable and interpretable intrusion detection system.

Keywords:UNR-IDD, NSL-KDD, NIDS, LIME, scalability, ML models, performance metrics, ensemble learning.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

Today's computer networks and their associated applications have witnessed unprecedented growth with deployments like the Internet of Things (IoT), smart homes, and software-defined networks (SDNs). Yet, this speedy expansion has also grown the likelihood of network intrusions, which are a constant threat to network infrastructures. These attacks try to breach fundamental security concepts: **availability, authority, confidentiality, and integrity.** As a result their capacity to emulate regular network traffic, such threats are difficult to identify using conventional security measures [2]

To offer better protection against intrusions, the use of **Machine Learning (ML) for Network Intrusion Detection Systems (NIDS)** has gained momentum in the past decade. Various open-source datasets have been introduced to aid ML-based NIDS research, but most of them are plagued by severe limitations. One common problem is their excessive dependence on **flow-level statistics**, which may restrict adaptability across various network environments. Furthermore, some datasets have missing or incomplete records, which adversely affect model performance[2]

To solve these issues, we examine the **University of Nevada - Reno Intrusion Detection Dataset (UNR-IDD) [2]and NSL - KDD(NSL-KDD is short for Network Intrusion Detection System.)**, using **feature selection techniques** to enhance model efficiency. Different machine learning models, including **Logistic Regression, Random Forest, SVM, KNN, and Decision Trees**, are trained and tested for **both binary and multi-class classification** tasks. To improve model performance, **ensemble learning techniques** are investigated. In addition, in order to enhance model interpretability, **Local Interpretable Model-agnostic Explanations (LIME)** is implemented, providing an understanding of the models' decision-making process..

Through this study, we aim to develop an **efficient, scalable, and interpretable intrusion detection framework**, contributing to the advancement of cybersecurity solutions.

1.2 Problem Statement

Contemporary network infrastructures are subjected to escalating cybersecurity threats, which necessitates efficient intrusion detection. Classical Network Intrusion Detection Systems (NIDS) tend to be plagued by high false positives, bad scalability, and low interpretability. Machine Learning (ML)-based NIDS provide enhancements, but their success relies on good feature selection, model efficiency, and explainability.

This research improves intrusion detection by using feature selection methods, comparing ML models (Logistic Regression, Random Forest, SVM, KNN, Decision Trees), and enhancing interpretability through Local Interpretable Model-agnostic Explanations (LIME) and ensemble learning. The aim is to create a scalable, accurate, and interpretable NIDS framework to improve cybersecurity defenses against new threats.

Chapter 2

Literature Review

2.1 Introduction to Intrusion Detection System

An Intrusion Detection System is software or a system that monitors network traffic and detects an intrusion or unwanted activities in the network. IDS scan the networks to find out if someone is trying to penetrate the network illegally. In other words, it keeps an eye on the network's traffic to identify intrusion in the network.
Intrusion Detection System if properly configured will help you to:

- Monitor inbound and outbound network traffic.
- Analyze the patterns in the network continuously.
- Send an alarm immediately after detecting unwanted intrusion and activities in the network.

2.2 Different types of Intrusion Detection Systems

2.2.1 Network Intrusion Detection System (NIDS)

Network Intrusion Detection System sets up across the network at a specific planned point. NIDS monitors the traffic on the network from all devices. Similarly, it examines the traffic passing on the entire subnet and verifies it with the packet metadata and content. If NIDS detects any intrusion in the network, a warning alert is sent to the admin of that network. The best advantage of NIDS is that if it is installed in the same location where the firewall is located, then it will detect if someone is trying to attack the firewall. In other words, with the help of NIDS, the firewall will also be protected from any policy breaching.

2.2.2 Host Intrusion Detection System (HIDS)

Organizations install a Host Intrusion Detection System (HIDS) on independent networked devices. However, HIDS examines the incoming and outgoing traffic of the device only. It detects suspicious activities

on the device and alerts the administrator. HIDS also checks whether system files are misplaced or not, for that it takes the screen capture of the current file system and verifies it with the screen capture of the previous file system. This file system stores the analytical information of network traffic. For instance, if the files are misplaced or changed it sends an alert to the administrator.

2.2.3 Protocol-based IDS (PIDS)

Organizations set up a Protocol-based Intrusion Detection System at the front end of the server. It interprets the protocols between the server and the user. PIDS monitors the HTTPS server regularly to secure the web. Similarly, it allows the HTTP server which is related to the protocol.

2.2.4 Application Protocol-based IDS (APIDS)

As we have seen that PIDS is set up at the front end of the server. Similarly, APIDS is set up within a group of servers. It interprets communication with the applications within the server to detect the intrusion.

2.2.5 Hybrid Intrusion Detection System

As the name says Hybrid Intrusion Detection system is a mixture of two different IDS. Hybrid System develops a network system by combining host agents with network information. In conclusion, Hybrid System is more responsive and effective as compared to other IDS. [7]

2.3 Types of Intrusion Detection Systems Methods

2.3.1 Signature-based Intrusion Detection Method

The IDS developed the Signature-based intrusion detection method to examine the network traffic and to detect attack patterns. For instance, it verifies the network traffic with the log data to identify the intrusion. If this method detects any intrusion then the IDS solution creates a signature of it and adds it to the list. The patterns which are detected are known as sequences and these sequences are a specific number of bytes or a set of 0's and 1's in the network. However, it is easy to detect the attacks whose patterns are existed in the system in the form of signatures. But to detect new attacks whose signature is not yet created is difficult.

2.3.2 Anomaly-based Intrusion Detection Method

As we have seen that it is difficult to detect unknown or new malware attacks with the help of the SignatureBased Detection method. Therefore, organizations use the anomaly-based intrusion detection method to identify those new and unknown suspicious attacks and policy breaching which the Signature-based detection method cannot identify easily.
However, new intrusion techniques and malware are increasing rapidly. This method uses Machine learning to create an activity model. If this method detects any receiving patterns which are not found in the model, then the method declares these patterns as malicious patterns. In conclusion, the anomaly-based detection system is better in comparison to the Signature-based method.

2.3.3 Hybrid Detection Method:

A Hybrid method uses both Signature and Anomaly-based intrusion detection methods together. However, the main reason behind the development of a hybrid detection system is to identify more potential attacks with fewer errors. [7]

Chapter 3

Description of Datasets

3.1 UNR-IDD Dataset

This dataset has shape (37411, 34) and consists of various network traffic features categorized into three groups.

Feature Category	**Description**
Port Statistic	Captures basic port-level traffic details, including Received Packets, Received Bytes, Sent Packets, Sent Bytes, Port Alive Duration, Packets Rx Dropped, Packets Tx Dropped, Packets Rx Errors, Packets Tx Errors.
Delta Port Statistic	Represents the changes in port statistics over time, including Delta Received Packets, Delta Received Bytes, Delta Sent Packets, Delta Sent Bytes, Delta Port Alive Duration, Delta Packets Rx Dropped, Delta Packets Tx Dropped, Delta Packets Rx Errors, Delta Packets Tx Errors.
General Network Statistic	Provides higher-level network insights, including Connection Point, Total Load/Rate, Total Load/Latest, Unknown Load/Rate, Unknown Load/Latest, Latest Bytes Counter, is_valid, TableID, ActiveFlowEntries, PacketsLookedUp, PacketsMatched, MaxSize, Switch ID.
Binary Labels (Binary Classification)	Normal (Legitimate Traffic), Attack (Intrusion).
Multi-Class Labels	Normal, TCP-SYN, PortScan, Overflow, Blackhole, Diversion.

Table 3.1: Feature Description of UNR-IDD Dataset

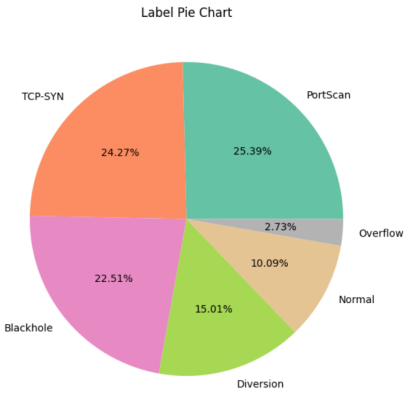

Figure 3.1: Pie chart for multiclass classification for UNR-IDD Dataset

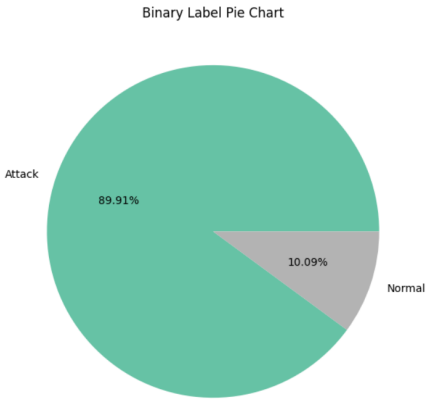

Figure 3.2: Pie chart for Binary class classification for UNR-IDD Dataset

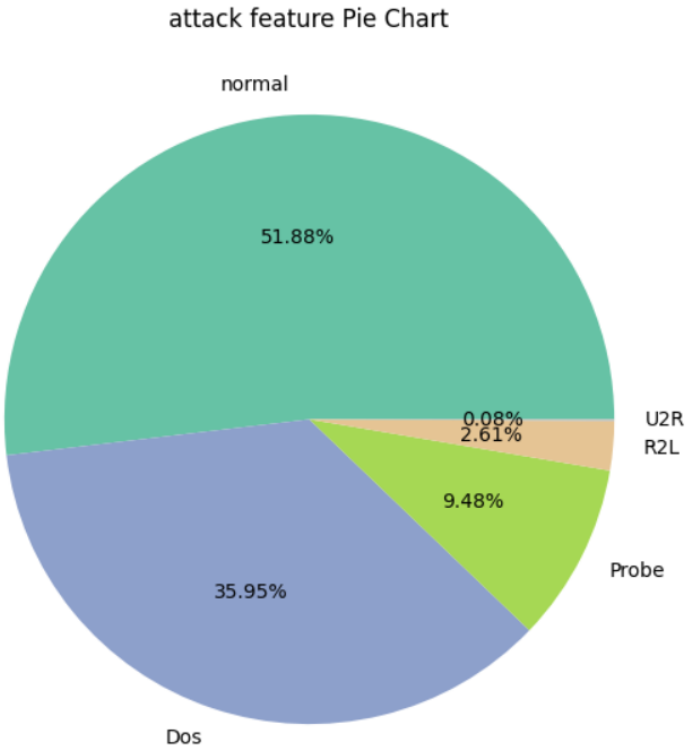

Figure 3.3: Pie chart for multi-class classification for NSL-KDD Dataset

3.2 NSL-KDD Dataset

This dataset has shape (125973, 43), it also consists of various network traffic features.

Feature Name	**Description**
duration	Length of the connection in seconds
protocol_type	Type of protocol (TCP, UDP, ICMP)
service	Network service on the destination (e.g., HTTP, FTP)
flag	Connection status flag (e.g., SF, REJ)
src_bytes	Bytes sent from source to destination
dst_bytes	Bytes sent from destination to source
land	1 if source and destination IP addresses/ports are the same, else 0
wrong_fragment	Number of wrong fragments in packets
urgent	Number of urgent packets
hot	Number of "hot" indicators (e.g., system calls)
num_failed_logins	Count of failed login attempts
logged_in	1 if login was successful, else 0
num_compromised	Number of compromised conditions
root_shell	1 if root shell was obtained, else 0
su_attempted	1 if "su root" command attempted, else 0
num_root	Number of root accesses
num_file_creations	Number of file creation operations
num_shells	Number of shell prompts invoked
num_access_files	Number of operations on control/access files
srv_count	Number of connections to the same service in past two seconds
serror_rate	Percentage of connections with SYN errors
srv_serror_rate	Percentage of connections with SYN errors in the same service
rerror_rate	Percentage of connections with REJ errors
srv_rerror_rate	Percentage of connections with REJ errors in the same service
same_srv_rate	Percentage of connections to the same service
diff_srv_rate	Percentage of connections to different services
srv_diff_host_rate	Percentage of different hosts accessed for the same service
dst_host_count	Number of connections to the same destination host
dst_host_srv_count	Number of connections to the same service at destination host
dst_host_same_srv_rate	Percentage of same-service connections to the same host
attack	Type of attack (e.g., normal, DoS, probe, R2L, U2R)
level	Attack severity level

Table 3.2: Description of NSL-KDD Dataset Features [3]

Attack Features	Changed Label
neptune, smurf, pod, back, teardrop, land, apache2, processtable, udpstorm, mailbomb, worm	Dos
warezclient, guess$_p asswd$, ftp_write, $multihop$, $imap$, $warezmaster$, phf, spy, snmpgetattack, snmpguess, httptunnel, sendmail, xlock, xsnoop, named	R2L
ipsweep, portsweep, nmap, satan, saint, mscan	Probe
rootkit, buffer_overflow, loadmodule, perl, ps, xterm, sqlattack	U2R
normal	Normal

Table 3.3: Mapping of Features to Changed Labels on NSL-KDD Dataset

3.3 Data Preprocessing

3.3.1 Introduction to Data Preprocessing

> "Data preprocessing in Machine Learning is a crucial step that helps enhance the quality of data to promote the extraction of meaningful insights from the data. Data preprocessing in Machine Learning refers to the technique of preparing (cleaning and organizing) the raw data to make it suitable for a building and training Machine Learning models." [5]

3.3.2 Data preprocessing in our datasets

UNR-IDD Dataset and NSL-KDD do not have missing values, avoiding any missing data handling. This makes preprocessing easier and guarantees data completeness for training and testing machine learning models.

UNR-IDD Dataset has a **single** duplicate value that is dropped, while NSL-KDD Dataset has **2015** duplicate values. For integrity of data, these duplicates should be dealt with using the code provided for efficient preprocessing and analysis.

We perform Label encoding to convert categorial features to numerical features.
In both of our datasets, we used Standardization to normalize our data.
We perform feature selection methods such as ANOVA, Mutual Information, and Recursive Feature Elimination (RFE), as well as Principal Component Analysis (PCA) for dimensionality reduction.

Chapter 4

Machine learning Models

4.1 Overview

Machine learning algorithms learn to identify relationships and patterns within data. Through input of past data, such algorithms can make predictions, categorize information, group data points, lower dimensions and even create new content. [10]

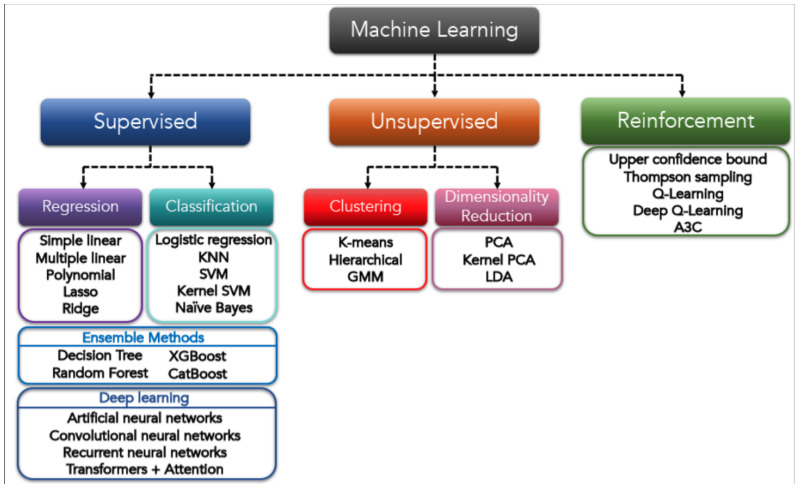

Figure 4.1: Machine Learning Models[8]

4.2 Performance Metrics

In essence, in classification problems the key evaluation metrics that are taken into account are accuracy, precision, recall and F1-Score. These will be calculated using values which indicate the rate of classifications. Such rates are as True positive, True Negative, False Positive and False Negative. These core measures are useful to ascertain the aforementioned performance measures

Accuracy measures the percentage of correct classification in total num-

ber of instances. And can be calculated as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision measures the number of correct detections with the number of incorrect instances. Calculated as:

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

Recall measures the number of correct predictions with the number of missed instances, given as:

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

F1-Score measures the average of precision and recall. The harmonic mean of mentioned two metrics can be found as:

$$F1\text{-}Score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

With the evaluation of these measures the model is prepared for real-time applications. When the evaluation criteria provide low range of values, thereby the parameters of the algorithms should be modified and again evaluated. [9]

4.3 Machine Learning model used

4.3.1 Logistic Regression (LR)

The LR is a type of logarithm linear model. The LR algorithm computes the probabilities of different classes through parametric logistic distribution, calculated as shown in Formula :

$$P(Y = k \mid x) = \frac{e^{w_k \cdot x}}{1 + \sum_{k}^{K-1} e^{w_k \cdot x}}$$

where k = 1,2...K-1. The sample x is classified into the maximum probability class. An LR model is easy to construct, and model training is efficient. However, LR cannot deal well with nonlinear data, which limits its application. [6]

For Multi-class on NSL-KDD Dataset: We used max_iter=1500, random_state=42, multi_class='multinomial', and solver='lbfgs' to ensure convergence and efficient optimization.

4.3.2 Support Vector Machine (SVM)

The strategy in SVMs is to find a max-margin separation hyperplane in then-dimension feature space. SVMs can achieve gratifying results even with small-scale training sets because the separation hyperplane is determined only by a small number of support vectors. However, SVMs are sensitive to noise near the hyperplane. SVMs are able to solve linear problems well. For nonlinear data, kernel functions are usually used. [6]

For Multi-class and binary-class on UNR-IDD: We used a polynomial kernel of degree 3 and cost parameter as C=10.
For Multi-class on NSL-KDD Dataset: We used a polynomial kernel with random_state=42

4.3.3 K-Nearest Neighbor (KNN)

The core idea of KNN is based on the manifold hypothesis. If most of a sample's neighbors belong to the same class, the sample has a high probability of belonging to the class. Thus, the classification result is only related to the top-k nearest neighbors. The smaller k is, the more complex the model is and the higher the risk of overfitting. Conversely, the larger k is, the simpler the model is and the weaker the fitting ability. [6]

For Multi-class on UNR-IDD: We used hyperparameter of 3 nearest neighbours. **For Binary-class on UNR-IDD:** We used hyperparameter of 5 nearest neighbours. **For Multi-class on NSL-KDD:** We used hyperparameter of 5 nearest neighbours.

4.3.4 Decision tree

The decision tree algorithm classifies data using a series of rules. The model is tree like, which makes it interpretable. The decision tree algorithm can automatically exclude irrelevant and redundant features. The learning process includes feature selection, tree generation, and tree pruning. [6]

For multi-class on NSL-KDD, we used hyperparameters like random_state=42, max_depth=15, min_samples_split=5, and criterion='entropy' to control tree growth and optimize decision-making based on information gain.

4.3.5 Random Forest:

Random forests, also known as random decision forests, are an ensemble learning technique that combines multiple decision trees to solve classification and regression problems. In classification tasks, the final prediction is determined by majority voting among the trees, whereas in regression, the output is the average prediction of all trees.

The prediction of a random forest model for an input $\mathbf{x}$ is given by:

$$f_{\mathrm{RF}}(\mathbf{x}) = \frac{1}{T} \sum_{t=1}^{T} f_t(\mathbf{x}),$$

where T represents the total number of trees, and $f_t(\mathbf{x})$ denotes the output of the t-th decision tree [1]. **For multi-class on NSL-KDD**we used hyperparameters like random_state=42,n_estimators=200,max_depth=20,min_samples_leaf=2,max_features='sqrt'

4.4 Ensemble Learning Techniques used

4.4.1 Voting Ensemble Learning

In this the predictions by each model are considered as a **"vote"** .The predictions which we get from the majority of the models are used as the final predictions.

4.4.2 Stacking Ensemble Learning

Stacking ensemble learning is a method where predictions of various (base -levels) models are combined to create a new model(meta-model).The meta-model is utilized for prediction on the test set. Base level algorithms are trained on a full training dataset.Meta model is trained on the combination of all base level model's predictions as feature. Stacking is beneficial when the outcome of the individual algorithms are significantly different.

Chapter 5

Explainable artificial intelligence

5.1 Overview

Explainable artificial intelligence (XAI) describes a set of methods and methodologies that allow machine learning algorithms to generate output and results that can be comprehended and trusted by human users.

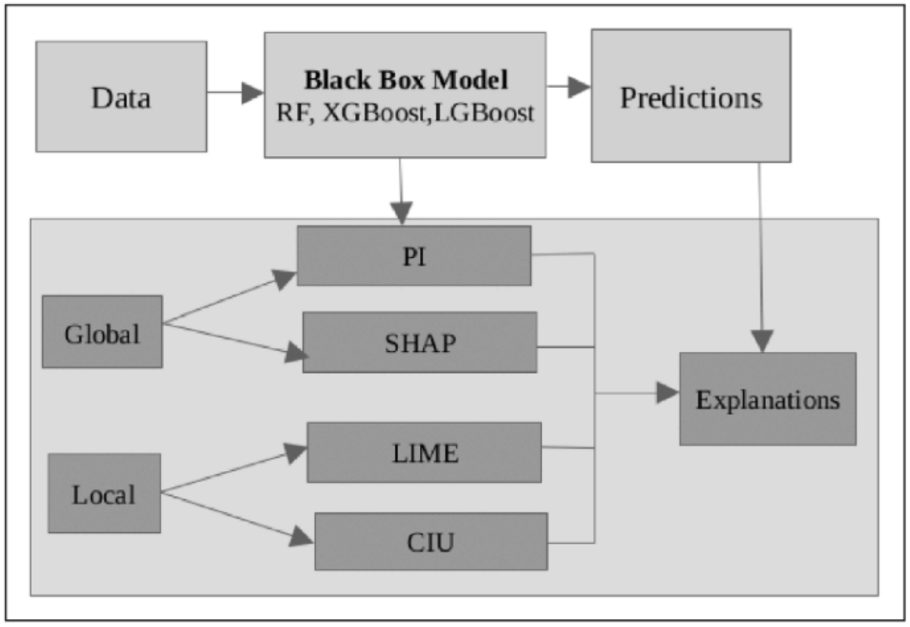

Figure 5.1: Explainability Methods [4].

5.2 Local interpretable model-agnostic explanations (LIME)

LIME is one of the most used XAI methods that makes use of a local model approximation to give understandable and explainable information regarding the most important and influential factors in the model's predictions. LIME can be used in python using the lime package, which offers various tools and functions to create

and interpret LIME explanations.

5.3 Performance of LIME on UNR-IDD Dataset

Figure 5.2: LIME for multiclass selecting single sample

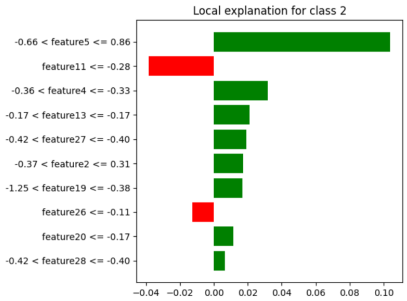

Figure 5.3: LIME for multiclass selecting single sample

Figure 5.4: LIME for multiclass selecting 100 samples

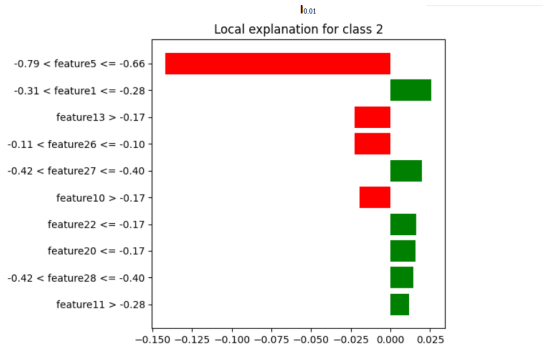

Figure 5.5: LIME for multiclass selecting 100 samples

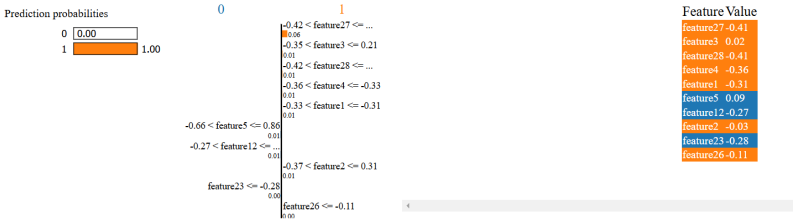

Figure 5.6: LIME for binary-class selecting 1 sample

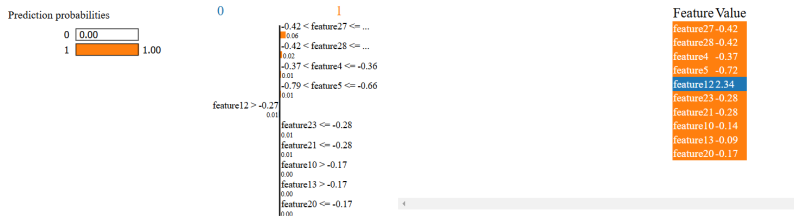

Figure 5.7: LIME for binary-class selecting 100 sample

5.4 Scalability and Throughput Testing for UNR-IDD

In the given figure 5.8 we can clearly see that upon increasing the size of training data our accuracy keeps on increasing.

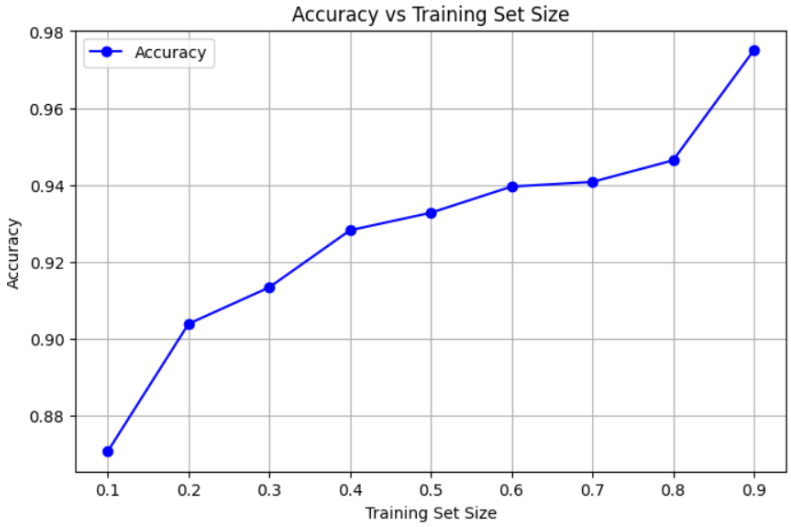

Figure 5.8: Scalability and Throughput Testing Accuracy

Chapter 6

Results and Discussion:

Features	Binary Label
Attack	1
Normal	0

Table 6.1: Feature representation with binary labels for binary-class on UNR-IDD Dataset

Features	Multi-class Label
TCP-SYN	1
Blackhole	2
Diversion	3
Overflow	4
Normal	5
PortScan	6

Table 6.2: Feature representation with Multi-class Label for multi-class on UNR-IDD Dataset

Attack Category	Encoded Value
Dos	0
Probe	1
R2L	2
U2R	3
Normal	4

Table 6.3: Label Encoding for Attack Categories for multi-class on NSL-KDD Dataset

6.1 Performance of Logistic Regression

6.1.1 For UNR-IDD

For UNR-IDD when using logistic regression we got an overall accuracy of 73% on multi-class classification and 100% overall accuracy for binary classification and other features as shown in table 6.5 6.4

Multi-class labels	Precision	Recall	F1 Score
1	0.61	0.94	0.74
2	0.73	0.75	0.74
3	0.68	0.55	0.61
4	1.00	0.22	0.36
5	1.00	1.00	1.00
6	0.88	0.57	0.69

Table 6.4: Performance metrics for multi-class classification

Binary-class Labels	**Precision**	**Recall**	**F1 Score**
0	1.00	1.00	1.00
1	1.00	1.00	1.00

Table 6.5: Precision, Recall, and F1 Score for Binary-class Labels

6.1.2 For NSL-KDD

For NSL-KDD when using Logistic Regression we got an overall accuracy of 89% on multi-class classification and other features as shown in table 6.6

Multi-Class Label	**Precision**	**Recall**	**F1-Score**
0	0.97	0.86	0.91
1	0.73	0.92	0.82
2	0.79	0.73	0.76
3	0.16	0.52	0.24
4	0.93	0.95	0.94

Table 6.6: Logistic Regression Classification Report for Multiclass on NSL-KDD Dataset

6.2 Performance of SVM

6.2.1 For UNR-IDD

For UNR-IDD when using svm we got an overall accuracy of 76% on multi-class classification and 100% overall accuracy for binary classification and other features as shown in table 6.7 6.8

SVM Classification Labels	Precision	Recall	F1 Score
1	0.59	0.97	0.73
2	0.82	0.82	0.82
3	0.82	0.69	0.75
4	0.92	0.23	0.37
5	1.00	1.00	1.00
6	0.94	0.53	0.68

Table 6.7: Performance metrics for SVM classification

SVM Binary Labels	Precision	Recall	F1 Score
0	1.00	1.00	1.00
1	1.00	1.00	1.00

Table 6.8: Performance metrics for SVM binary classification

6.2.2 For NSL-KDD

For NSL-KDD when using SVM we got an overall accuracy of 95% on multi-class classification and other features as shown in table 6.9

Multi-Class Label	**Precision**	**Recall**	**F1-Score**
0	0.97	0.95	0.96
1	0.86	0.95	0.90
2	0.88	0.94	0.91
3	0.44	0.49	0.47
4	0.98	0.96	0.97

Table 6.9: SVM Classification Report for Multiclass on NSL-KDD Dataset

6.3 Performance of Random Forest

6.3.1 For UNR-IDD

For UNR-IDD when using Random Forest we got an overall accuracy of 94% on multi-class classification and 100% overall accuracy for binary classification and other features as shown in table 6.11 6.10

RF Multi-class Labels	Precision	Recall	F1 Score
1	0.89	0.92	0.91
2	0.98	0.97	0.98
3	0.99	0.96	0.97
4	0.97	0.70	0.81
5	1.00	1.00	1.00
6	0.90	0.92	0.91

Table 6.10: Performance metrics for RF multi-class classification

RF Binary Labels	Precision	Recall	F1 Score
0	1.00	1.00	1.00
1	1.00	1.00	1.00

Table 6.11: Performance metrics for SVM binary classification

6.3.2 For NSL-KDD

For NSL-KDD when using Random Forest we got an overall accuracy of 97% on multi-class classification and other features as shown in table 6.12

Multi-Class Label	**Precision**	**Recall**	**F1-Score**
0	0.99	0.98	0.98
1	0.90	0.98	0.94
2	0.97	0.91	0.94
3	0.47	0.14	0.22
4	0.97	0.97	0.97

Table 6.12: Random Forest Classification Report for Multiclass

6.4 Performance of Decision Tree

6.4.1 For UNR-IDD

For UNR-IDD when using Decision Tree we got an overall accuracy of 91% on multi-class classification and 100% overall accuracy for binary classification and other features as shown in table 6.13 6.14

Multi-Class Label	**Precision**	**Recall**	**F1-Score**
1	0.87	0.86	0.87
2	0.95	0.96	0.95
3	0.95	0.94	0.95
4	0.72	0.66	0.69
5	1.00	1.00	1.00
6	0.88	0.88	0.88

Table 6.13: Decision Tree Classification Report for Multiclass on UNR-IDD Dataset

Binary Class	**Precision**	**Recall**	**F1-Score**
0	1.00	1.00	1.00
1	1.00	1.00	1.00

Table 6.14: Decision Tree Classification Report for Binary Class on UNR-IDD Dataset

6.4.2 For NSL-KDD

For NSL-KDD when using Decision Tree we got an overall accuracy of 90% on multi-class classification and other features as shown in table 6.15

Multi-Class Label	Precision	Recall	F1-Score
0	0.98	0.95	0.96
1	0.89	0.78	0.83
2	0.90	0.68	0.78
3	0.35	0.13	0.19
4	0.86	0.98	0.91

Table 6.15: Classification Report for Decision Tree (Multiclass) on NSL-KDD Dataset

6.5 Performance of KNN

6.5.1 For UNR-IDD

For UNR-IDD when using KNN we got an overall accuracy of 85% on multi-class classification and 100% overall accuracy for binary classification and other features as shown in table 6.17 6.16

Multi-Class Label	Precision	Recall	F1-Score
1	0.76	0.82	0.79
2	0.93	0.88	0.91
3	0.89	0.89	0.89
4	0.59	0.35	0.43
5	1.00	1.00	1.00
6	0.80	0.80	0.80

Table 6.16: KNN Classification Report for Multiclass on UNR-IDD Dataset

Binary Class	Precision	Recall	F1-Score
0	1.00	1.00	1.00
1	1.00	1.00	1.00

Table 6.17: KNN Classification Report for Binary class on UNR-IDD Dataset

6.5.2 For NSL-KDD

For NSL-KDD when using KNN we got an overall accuracy of 96% on multi-class classification and other features as shown in table 6.18

Multi-Class Label	Precision	Recall	F1-Score
0	0.98	0.96	0.97
1	0.88	0.96	0.92
2	0.93	0.93	0.93
3	0.47	0.54	0.50
4	0.98	0.97	0.97

Table 6.18: KNN Classification Report for Multiclass on NSL-KDD Dataset

6.6 Performance of Ensemble methods

6.6.1 Voting Ensemble Learning

For UNR-IDD

Classifier	Train Accuracy (%)	Test Accuracy (%)
Hard Voting Classifier	95.67	91.41
Soft Voting Classifier	97.59	90.99

Table 6.19: Comparison of Hard and Soft Voting Classifiers: Overfitting Analysis for multi class on UNR-IDD Dataset

By analyzing the table ,we prefer hard voting due to less overfitting for multi class on UNR-IDD Dataset

Classifier	Train Accuracy (%)	Test Accuracy (%)
Voting Classifier (Binary)	100.0	100.0

Table 6.20: Accuracy of Voting Classifier on Binary Classification on UNR-IDD Dataset

For NSL KDD

Metric	Score (%)
Train Accuracy	99.27
Test Accuracy	92.76
Cross-validation Mean Accuracy	99.17

Table 6.21: Accuracy Scores of the Voting Classifier for multi class on NSL-KDD dataset

6.6.2 Stacking Ensemble Learning

Classifier	Accuracy (%)
Stacking Classifier (Multi-Class)	94.65

Table 6.22: Stacking Classifier Accuracy for multi class on UNR-IDD Dataset

In comparing Stacking Ensemble and Voting Classifier, Stacking Ensemble performs better. Thus, Stacking would be better to use compared to Voting to gain the optimal performance and accuracy in ensemble learning.

Chapter 7

Summary and Conclusions

7.1 Summary

In this research, we tried out different machine learning algorithms like Random Forest (RF), Decision Tree, K-Nearest Neighbors (KNN), Logistic Regression, and Support Vector Machine (SVM). We also tried ensemble learning techniques like Voting Classifier and Stacking Classifier. Explainable AI methods, i.e., Local Interpretable Model-agnostic Explanations (LIME), were also used to improve interpretability. For the first dataset, UNR-IDD, the Random Forest model gave the highest performance for multi-class classification at 94% overall accuracy. Likewise, for the second dataset, NSL-KDD, Random Forest performed better than other models in multi-class classification at 97% accuracy. On the side of ensemble learning, Stacking Ensemble performed better than the Voting Classifier at 94.65% accuracy.

7.2 Conclusions

According to the experimental findings, the Random Forest model performed better consistently on both datasets. The Stacking Ensemble approach performed better than the Voting Classifier and was thus the best option for maximum performance in ensemble learning. Explainable AI techniques like LIME were useful; however, their performance was dataset-dependent, with limitations being seen in the NSL-KDD dataset.

7.3 Future Work

For future enhancement, the size of the dataset can be augmented to counteract overfitting in machine learning as well as ensemble models. Explainable AI methods were also not effective on the NSL-KDD dataset, so more optimization and tuning are required in the future.

References

[1] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[2] Tapadhir Das. Unr-idd dataset, 2025. Accessed: March 2, 2025.

[3] GeeksforGeeks. Intrusion detection system using machine learning algorithms, 2024. Accessed: March 3, 2025.

[4] Swetha Hariharan et al. Xai for intrusion detection system: comparing explanations based on global and local scope. *Journal of Computer Virology and Hacking Techniques*, 19(2):217–239, 2023.

[5] Alireza Hasannejad. Data preprocessing in machine learning, 2024. Accessed: 2025-03-01.

[6] Hongyu Liu and Bo Lang. Machine learning and deep learning methods for intrusion detection systems: A survey. *Applied Sciences*, 9(20):4396, 2019.

[7] Malla Reddy College of Engineering and Technology (MRCET). *Intrusion Detection Systems*. [Accessed: 01-03-2025].

[8] Thaddeus Segura. Introduction to machine learning. `https://thaddeus-segura.com/intro_to_ml/`, n.d. Accessed: 2025-03-02.

[9] Shaikh Shakeela et al. Optimal ensemble learning based on distinctive feature selection by univariate anova-f statistics for ids. *International Journal of Electronics and Telecommunications*, pages 267–275, 2021.

[10] TechTarget. Machine learning (ml). `https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML`, n.d. Accessed: 2025-03-02.