Enhancing Network Security Through Machine Learning-Based Intrusion Detection

Mohan Babu Kunchala
A20524765
mkunchala@hawk.iit.edu

Sanjitha Reddy Pathuri
A20524383
spathuri2@hawk.iit.edu

Sirisha Gandham
A20544789
sgandham1@hawk.iit.edu

Final Project Report

1. Introduction

1.1 Abstract

In today's digital age, protecting our personal and professional information from cyber threats is more important than ever. Standard security methods are no longer enough to stop advanced attacks. This project aims to improve network security using machine learning techniques on the UNSW-NB15 dataset.

By using machine learning models like neural networks, including MLP classifiers, CNNs, and RCNNs, the project tries to address the limitations of traditional security approaches. It will also explore feature selection methods to enhance the performance of these classifiers. Through rigorous testing and comparison with conventional classifiers, the project aims to develop a robust intrusion detection system that can even identify complex attacks.

The project methodology involves data preprocessing, feature selection, and training and evaluating the machine learning models. Advancing the field of network intrusion detection is crucial for developing more effective cybersecurity measures. This research project aims to significantly improve network security and protect organizations worldwide from cyber threats. By enhancing our understanding of network intrusion detection, this work can help safeguard businesses and individuals against evolving cyber risks.

1.2 Why UNSW-NB15?

The UNSW-NB15 dataset has gained significant attention in the network intrusion detection research community [3]. This dataset is designed to mimic real-world network traffic, including both normal and attack scenarios, making it diverse and suitable for evaluating network intrusion detection techniques. A key advantage of this dataset is the comprehensive range of features it provides, which are useful for feature engineering and model development in machine learning. By leveraging the richness of this dataset, researchers can explore a variety of machine learning algorithms and assess their effectiveness in identifying network intrusions. Through the utilization of this dataset, researchers can develop and validate machine learning-based methodologies that can accurately and efficiently detect network intrusions, potentially strengthening network systems and infrastructure against security threats.

1.2.1 UNSW NB-15 Dataset Description

For this project, we utilized the UNSW-NB15 dataset, available on Kaggle. This network traffic dataset contains labeled traffic data generated by a network simulator and contains both legitimate and illicit traffic. It includes a total of 49 features, encompassing nominal, integer, float, timestamp, and binary data types.

While the original dataset has 49 features, our primary datasets used for training and testing only include 45 features. The excluded features are the first four, representing source and destination IP addresses and ports. The remaining features include protocol type, connection duration, number of bytes transmitted/received, and more.

The UNSW-NB15 dataset is split into 70-30 training and testing sets. The training set contains 175,341 records, and the testing set has 82,332 records. These sets are provided in separate CSV files alongside the dataset.

The dataset contains a binary variable indicating whether the connection was classified as a normal transaction or an attack (0 for normal, 1 for attack) and a categorical variable "attack_cat" indicating the type of attack (if any) associated with each connection. This variable has nine categories such as Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms.

The UNSW-NB15 dataset is a valuable resource for network intrusion detection researchers due to its real world nature and diverse range of network traffic. It serves as an excellent benchmark dataset for evaluating and comparing the performance of various machine learning algorithms and intrusion detection systems.

The dataset has been uploaded and is available at our Project Github Repository

1.3 Related/Previous Work

Cybersecurity is a major priority for organizations worldwide. Network intrusion detection plays a crucial role in preventing attacks that take advantage of network vulnerabilities. Machine Learning (ML) methods have emerged as promising solutions to enhance network security, attracting significant attention from researchers.

Various ML approaches, including supervised, unsupervised, and hybrid methods, have been proposed for network intrusion detection. Researchers have evaluated the effectiveness of these techniques, such as the study by Moustafa et al. (2015) [5], which utilized the UNSW-NB15 dataset to assess Network Intrusion Detection Systems. They found that using Principal Component Analysis (PCA) for feature selection improved the accuracy of classifiers, with Naive Bayes achieving 98.33% accuracy.

Similarly, Alazab et al. (2016) [1] focused on the challenges in anomaly-based Network Intrusion Detection and proposed a hybrid approach combining anomaly-based and signature-based techniques to enhance network security and for reduced false positives and better precision.

A recent study by Islam and Ahmed (2019) [2] examined machine learning (ML) techniques, dividing them into supervised, unsupervised, and hybrid categories. The researchers highlighted drawbacks, such as the need for labeled data in supervised learning, and suggested enhancing model interpretability and resilience against evolving attacks.
Mukherjee et al. (2020) [6] evaluated Deep Learning (DL) approaches on the UNSW-NB15 dataset. They found that techniques like LSTM, Autoencoder, and CNNs achieved high accuracy. The DL approaches outperformed traditional ML algorithms like Decision trees and Random Forest, sparking discussion on the advantages and limitations of these methods for intrusion detection.

Moustafa and Slay (2016) [4] introduced the UNSW-NB15 dataset and compared the performance of machine learning algorithms against the benchmark KDD99 dataset. They advocated for the use of ensemble methods to improve detection accuracy across different types of attacks.

Our project aims to explore the potential of Neural Networks and feature selection for Network Intrusion Detection using the UNSW-NB15 dataset. We will investigate different Neural Network architectures, such as MLP Classifier, CNNs, and RCNNs, and examine how feature selection methods can improve the performance of these classifiers. Additionally, we will compare the effectiveness of Neural Networks with traditional classifiers like XGBoost, SVM, Random Forest, Decision Tree, and Logistic Regression.

2 Problem Description

2.1 Methodology

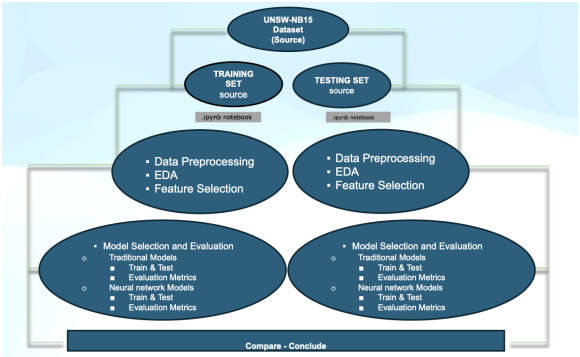

Figure 1: Project Methodology Overview

1. Dataset Preparation:

We start by extracting and cleaning the data. The UNSW-NB15 dataset, which includes both the Training and Testing Datasets, has missing values and categorical variables. This requires preprocessing

before feeding the data into machine learning models. We do this by removing unnecessary columns, encoding categorical variables using LabelEncoder(), and standardizing numerical variables using StandardScaler(). These steps are applied to both the Training and Testing Datasets.

(a) Data Cleaning: We drop columns like 'id', 'proto', 'service', and 'state' as they are not relevant for our analysis and classification tasks. This reduces the dimensionality of the data, which improves the efficiency of our analysis.

(b) Data Encoding: We use LabelEncoder to translate the unique column values into numerical representations. This allows us to use the data in machine learning models that require numeric inputs. The encoded values range from 0 to n-1, where n (n=10) represents the number of unique categories in the column.

(c) Data Standardization: The StandardScaler tool is used to normalize the numerical columns. This is an important step for many machine learning models, as it can enhance their performance.

2. Exploratory Data Analysis (EDA):

EDA is crucial for understanding the distribution of features, how they are correlated, and identifying any outliers. Visualizations like histograms, heatmaps, and box plots are helpful in this process, which is conducted in both the Training and Testing Dataset notebooks.

(a) Histograms of Numerical Variables: These graphs show the distribution of values for each numerical variable.

(b) Heatmap of Correlation Matrix: Heatmap displays the correlations between the variables.

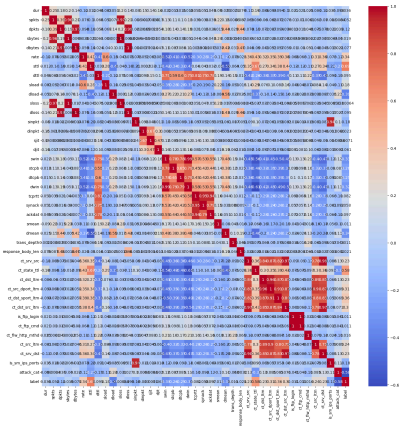

Figure 2: Heatmap of the Dataset

(c) Box plots of numerical variables: These box plots are grouped by different attack categories, allowing us to see how the distribution of values differs across the categories.

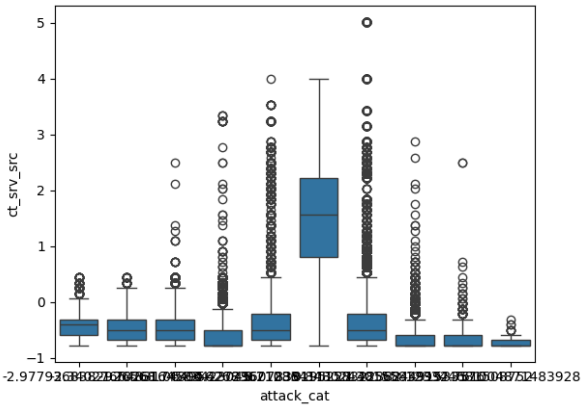

Figure 3: Boxplot of ct_srv_src w.r.t attack_cat

3. Feature Selection:

Feature Selection Enhances Performance. The feature selection process aims to improve classifier performance by identifying the most relevant features. By utilizing the SelectKBest algorithm and mutual information score, features are selected based on their informational value and relationship to the target variable. This streamlined approach is particularly useful for handling nonlinear relationships or complex feature interactions.

Figure 4 below shows a code snippet demonstrating the feature selection process. Through this process, noise and redundancy are eliminated, leading to improved model efficiency. This is implemented in both the Training and Testing Dataset notebooks.

```python
# Data Preprocessing
def preprocess_data(df):
    # Drop unwanted columns
    df.drop(['id', 'proto', 'service', 'state'], axis=1, inplace=True)

    # Handle missing values if any
    df.dropna(inplace=True)

    # Encode categorical variables
    cat_cols = ['attack_cat','label']
    encoder = LabelEncoder()
    for col in cat_cols:
        df[col] = encoder.fit_transform(df[col])

    # Standardize numerical variables
    num_cols = df.select_dtypes(include='number').columns.tolist()
    scaler = StandardScaler()
    df[num_cols] = scaler.fit_transform(df[num_cols])
    return df
```

Figure 4: Code Snippet - Data Preprocessing

4. Model Selection and Training:

Various neural network models (CNNs, RCNNs, and MLPs) are compared to traditional classifiers (Random Forest, Decision Tree, XGBoost, SVM, and Logistic Regression) to assess their performance. This comparative analysis is conducted in both the Training and Testing Dataset notebooks.

5. Model Evaluation:

The performance of the classifier is assessed using various metrics such as Accuracy, Precision, Recall, F1-Score, ROC Curves, and AUC Score Plots. These evaluation tools help identify misclassified samples and analyze the results, which are examined in both the Training and Testing Dataset notebooks.

2.2 Traditional Classifiers

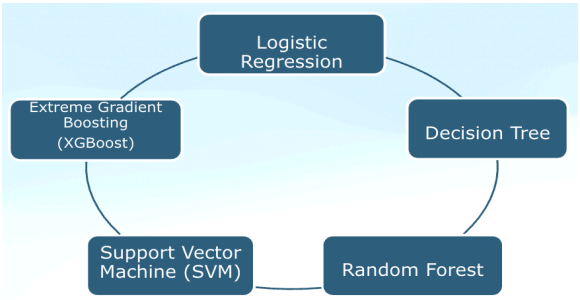

Figure 5: Traditional Models

1. Logistic Regression Model:

Logistic Regression (LR) is a widely used statistical technique for binary classification problems, making it suitable for predicting binary outcomes like malicious or benign network connections in our project context. LR's simplicity and computational efficiency make it an ideal baseline model for large datasets like UNSW-NB15. Its straightforward results interpretation is advantageous for security analysts, aiding in understanding classification decisions.

2. Decision Tree Model:

The decision tree model accommodates both continuous and categorical data, making it suitable for the diverse feature types in the UNSW-NB15 dataset. It excels in identifying patterns and rules for distinguishing between legitimate and malicious traffic. Additionally, decision trees' interpretability aids in understanding prediction rationale and feature importance. Their computational efficiency is beneficial for large datasets.

Model Architecture:

- No. of Leaves: 3
- Depth of Tree: 2
- Class Names: [Attack, Normal]

3. Random Forest Model:

Random Forest, an ensemble learning technique, combines multiple decision trees for improved prediction accuracy and stability. It handles large datasets with high-dimensional features effectively and is robust against noise and outliers. Its ability to capture interactions and non-linear correlations between features makes it suitable for identifying sophisticated attacks.

Model Architecture:

- Max Depth: 4
- No. of Estimators: 150
- No. of Trees: 100

4. Support Vector Model (SVM):

SVM excels in handling high-dimensional feature spaces, making it suitable for datasets like UNSW-NB15 with numerous features. Its kernel methods enable effective handling of non-linearly separable data by mapping it to higher-dimensional spaces. The RBF kernel, widely used in practice, effectively models complex decision boundaries, enhancing SVM's performance for network intrusion detection.

Model Architecture:

- Classifier: SVC()
- Kernel: Default - RBF

5. XGBoost Model:

Extreme Gradient Boosting (XGBoost) is an ensemble learning technique known for its exceptional performance on structured datasets with high dimensionality. It sequentially builds decision trees while learning from previous mistakes, making it effective for handling large datasets like UNSW-NB15. Its robustness and efficiency make it well-suited for network intrusion detection classification tasks.

2.3 Neural Network Models

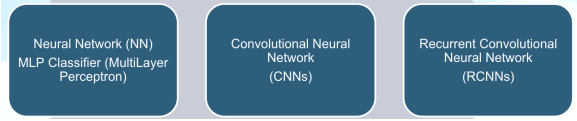

Figure 6: Neural Network Models

1. MLP Classifier Model:

The model utilized here is the Multi-Layer Perceptron (MLP) Classifier, widely employed in various classification tasks. Utilizing the MLPClassifier function, it trains via backpropagation, making it suitable for addressing diverse classification challenges. Given the complexity of the UNSW-NB15 dataset with highly correlated features, MLP, renowned for handling high-dimensional data with non-linear connections, is well-suited for this task.

Model Architecture:

- Classifier: MLPClassifier()
- Activation Function: Default - ReLU
- Optimization Algorithm: Default - SGD
- Hidden Layer: Default - 1

2. CNN Model:

The Convolutional Neural Network (CNN) Model is applied to the network traffic data in the UNSW-NB15 dataset. CNN's ability to learn hierarchical representations of input data, capturing temporal patterns and dependencies, makes it a suitable choice. The 1D convolutional layer identifies local patterns, while the max pooling layer reduces spatial dimensionality. Fully connected layers provide non-linear mapping between features and output classes.

Model Architecture:

- 1 Conv1D Layer: Filters: 64, Kernel Size: 3, Activation: ReLU
- 1 MaxPooling Layer: Pool Size: 2
- Flatten: Converts Multi-Dimensional input data to 1D array
- Dense (50), Activation: ReLU
- Dense (1), Activation: Sigmoid
- Optimizer: Adam, Loss: BCE - Binary Cross Entropy
- Epochs: 10, Batch Size: 32

3. RCNN Model:

The Recurrent Convolutional Neural Network (RCNN) combines recurrent and convolutional layers, facilitating the recognition of short and long-term temporal patterns in data. It comprises a 1D convolutional layer, max pooling layer, LSTM layer, and dense layer for binary classification. The RCNN's ability to handle sequences of varying lengths, crucial for datasets like UNSW-NB15 with variable network traffic sequences, contributes to its high accuracy.

Model Architecture:

- 1 Conv1D Layer: Filters: 64, Kernel Size: 3, Activation: ReLU
- 1 MaxPooling Layer: Pool Size: 2
- LSTM Layer: 100 Units
- Dense (1), Activation: Sigmoid
- Optimizer: Adam, Loss: BCE - Binary Cross Entropy
- Epochs: 10, Batch Size: 32

3 Results

Traditional Models		Logistic Regression	Decision Tree	SVM	XGBoost	Random Forest
TRAINING SET	Accuracy:	0.89639	1.0	0.99925	1.0	1.0
	F1-Score:	0.90	1.00	1.00	1.00	1.00
TESTING SET	Accuracy:	0.93121	1.0	0.99902	1.0	1.0
	F1-Score:	0.93	1.00	1.00	1.00	1.00

Neural Network Models		NN (MLP Classifier)	CNN	RCNN
TRAINING SET	Accuracy:	0.99998	0.99918	0.99929
	F1-Score:	1.00	1.00	1.00
TESTING SET	Accuracy:	0.99995	0.99740	0.99740
	F1-Score:	1.00	1.00	1.00

Figure 7: Model Performance Overview

The table provides an overview of the performance of both traditional and neural network models on the UNSW-NB15 dataset. Across both the training and testing sets, all models demonstrated high accuracy and F1 scores, indicating their effectiveness in identifying network intrusions. Notably, MLP Classifier, CNN, and RCNN showed exceptional accuracy and F1 scores, outperforming traditional models.

Among neural network models, MLP Classifier exhibited high accuracy on both sets, while CNN and RCNN achieved perfect accuracy on the training set and excellent accuracy on the testing set. Furthermore, all neural network models obtained perfect F1 scores, highlighting their robustness in classification tasks.
In contrast, among traditional models, all except Logistic Regression achieved near-perfect accuracy on the training set. However, on the testing set, all traditional models demonstrated high accuracy, with SVM slightly lagging behind. The F1 scores for traditional models consistently remained high.

```
Logistic Regression Accuracy: 0.8963937418017984
Logistic Regression Confusion Matrix:
 [[13726  3046]
 [ 2404 33427]]
Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.82      0.83     16772
           1       0.92      0.93      0.92     35831

    accuracy                           0.90     52603
   macro avg       0.88      0.88      0.88     52603
weighted avg       0.90      0.90      0.90     52603
```

Figure 8: Logistic Regression - Training Set

```
Logistic Regression Accuracy: 0.9312145748987855
Logistic Regression Confusion Matrix:
 [[10719   428]
 [ 1271 12282]]
Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.96      0.93     11147
           1       0.97      0.91      0.94     13553

    accuracy                           0.93     24700
   macro avg       0.93      0.93      0.93     24700
weighted avg       0.93      0.93      0.93     24700
```

Figure 9: Logistic Regression - Testing Set

```
Decision Tree Accuracy: 1.0
Decision Tree Confusion Matrix:
 [[16772     0]
 [    0 35831]]
Decision Tree Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     16772
           1       1.00      1.00      1.00     35831

    accuracy                           1.00     52603
   macro avg       1.00      1.00      1.00     52603
weighted avg       1.00      1.00      1.00     52603
```

Figure 10: Decision Tree - Training Set

```
Decision Tree Accuracy: 1.0
Decision Tree Confusion Matrix:
 [[11147     0]
 [    0 13553]]
Decision Tree Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     11147
           1       1.00      1.00      1.00     13553

    accuracy                           1.00     24700
   macro avg       1.00      1.00      1.00     24700
weighted avg       1.00      1.00      1.00     24700
```

Figure 11: Decision Tree - Testing Set

```
Random Forest Accuracy: 1.0
Random Forest Confusion Matrix:
 [[16772     0]
 [    0 35831]]
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     16772
           1       1.00      1.00      1.00     35831

    accuracy                           1.00     52603
   macro avg       1.00      1.00      1.00     52603
weighted avg       1.00      1.00      1.00     52603
```

Figure 12: Random Forest - Training Set

```
Random Forest Accuracy: 1.0
Random Forest Confusion Matrix:
 [[11147     0]
 [    0 13553]]
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     11147
           1       1.00      1.00      1.00     13553

    accuracy                           1.00     24700
   macro avg       1.00      1.00      1.00     24700
weighted avg       1.00      1.00      1.00     24700
```

Figure 13: Random Forest - Testing Set

```
SVM Accuracy: 0.9992585974183982
SVM Confusion Matrix:
 [[16758    14]
 [   25 35806]]
SVM Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     16772
           1       1.00      1.00      1.00     35831

    accuracy                           1.00     52603
   macro avg       1.00      1.00      1.00     52603
weighted avg       1.00      1.00      1.00     52603
```

Figure 14: SVM - Training Set

```
SVM Accuracy: 0.9990283400809716
SVM Confusion Matrix:
 [[11141     6]
 [   18 13535]]
SVM Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     11147
           1       1.00      1.00      1.00     13553

    accuracy                           1.00     24700
   macro avg       1.00      1.00      1.00     24700
weighted avg       1.00      1.00      1.00     24700
```

Figure 15: SVM - Testing Set

```
XGBoost Accuracy: 1.0
XGBoost Confusion Matrix:
 [[16772     0]
 [    0 35831]]
XGBoost Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     16772
           1       1.00      1.00      1.00     35831

    accuracy                           1.00     52603
   macro avg       1.00      1.00      1.00     52603
weighted avg       1.00      1.00      1.00     52603
```

Figure 16: XGBoost - Training Set

```
XGBoost Accuracy: 1.0
XGBoost Confusion Matrix:
 [[11147     0]
 [    0 13553]]
XGBoost Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     11147
           1       1.00      1.00      1.00     13553

    accuracy                           1.00     24700
   macro avg       1.00      1.00      1.00     24700
weighted avg       1.00      1.00      1.00     24700
```

Figure 17: XGBoost - Testing Set

```
MLP Classifier Accuracy: 0.9999809896773948
MLP Classifier Confusion Matrix:
 [[16771     1]
 [    0 35831]]
MLP Classifier Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     16772
           1       1.00      1.00      1.00     35831

    accuracy                           1.00     52603
   macro avg       1.00      1.00      1.00     52603
weighted avg       1.00      1.00      1.00     52603
```

Figure 18: MLP Output - Training Set

```
MLP Classifier Accuracy: 0.9999595141700405
MLP Classifier Confusion Matrix:
 [[11147     0]
 [    1 13552]]
MLP Classifier Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     11147
           1       1.00      1.00      1.00     13553

    accuracy                           1.00     24700
   macro avg       1.00      1.00      1.00     24700
weighted avg       1.00      1.00      1.00     24700
```

Figure 19: MLP Output - Testing Set

```
Convolutional Neural Network (CNN) Accuracy: 0.9991825561279775
Convolutional Neural Network (CNN) Confusion Matrix:
 [[16770     2]
 [   41 35790]]
Convolutional Neural Network (CNN) Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     16772
           1       1.00      1.00      1.00     35831

    accuracy                           1.00     52603
   macro avg       1.00      1.00      1.00     52603
weighted avg       1.00      1.00      1.00     52603
```

Figure 20: CNN Output - Training Set

```
Convolutional Neural Network (CNN) Accuracy: 0.9974089068825911
Convolutional Neural Network (CNN) Confusion Matrix:
 [[11142     5]
 [   59 13494]]
Convolutional Neural Network (CNN) Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      1.00     11147
           1       1.00      1.00      1.00     13553

    accuracy                           1.00     24700
   macro avg       1.00      1.00      1.00     24700
weighted avg       1.00      1.00      1.00     24700
```

Figure 21: CNN Output - Testing Set

```
Recurrent Convolutional Neural Network (RCNN) Accuracy: 0.9992966180636086
Recurrent Convolutional Neural Network (RCNN) Confusion Matrix:
 [[16762    10]
 [   27 35804]]
Recurrent Convolutional Neural Network (RCNN) Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     16772
           1       1.00      1.00      1.00     35831

    accuracy                           1.00     52603
   macro avg       1.00      1.00      1.00     52603
weighted avg       1.00      1.00      1.00     52603
```

Figure 22: RCNN Output - Training Set

```
Recurrent Convolutional Neural Network (RCNN) Accuracy: 0.9974089068825911
Recurrent Convolutional Neural Network (RCNN) Confusion Matrix:
 [[11141     6]
 [   58 13495]]
Recurrent Convolutional Neural Network (RCNN) Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      1.00     11147
           1       1.00      1.00      1.00     13553

    accuracy                           1.00     24700
   macro avg       1.00      1.00      1.00     24700
weighted avg       1.00      1.00      1.00     24700
```

Figure 23: RCNN Output - Testing Set

Figures 8-23 depict the performance of traditional classifiers like decision trees, random forests, and neural network models like MLP/CNN/RCNN outputs, showcasing the models' ability to capture complex patterns in the dataset. Particularly, CNN and RCNN models, designed for sequential data, excelled in capturing temporal dependencies, surpassing other models.

However, Logistic Regression displayed comparatively lower accuracy, possibly due to factors like class imbalance or its linear nature, which may not effectively capture non-linear correlations.

For a comprehensive evaluation, additional metrics such as Precision-Recall Curve, ROC Curve, AUC Scores, Histograms, Classification Reports, and Confusion Matrix were considered (Figures 24-35). These metrics provide deeper insights into model performance and areas for improvement.

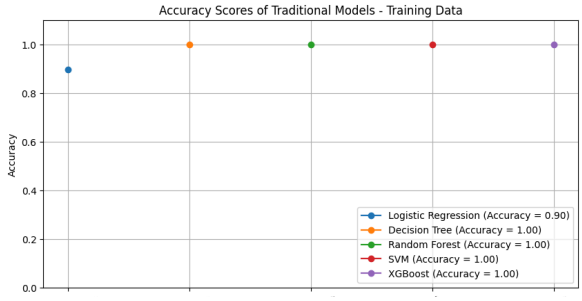

Figure 24: Accuracy of Traditional Models - Training Set

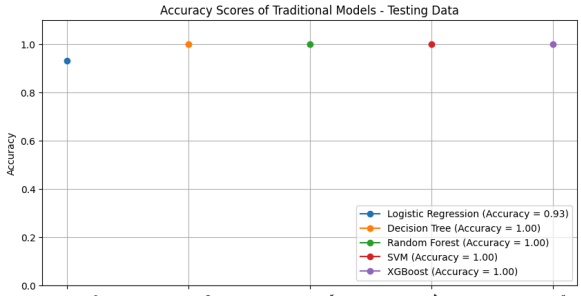

Figure 25: Accuracy of Traditional Models - Testing Set

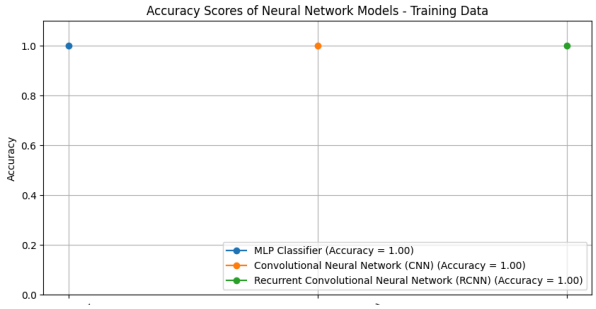

Figure 26: Accuracy of Neural network Models - Training Set

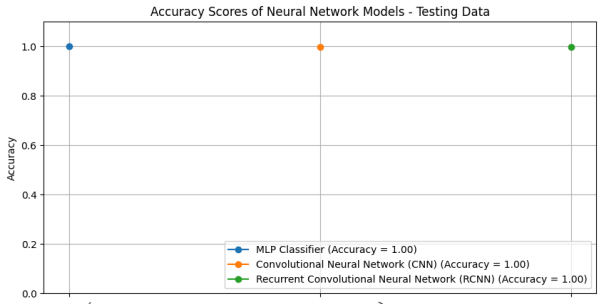

Figure 27: Accuracy of Neural network Models - Testing Set

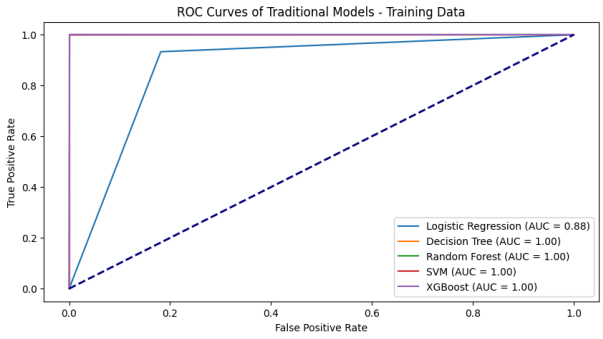

Figure 28: ROC Curve of Traditional Models - Training Set

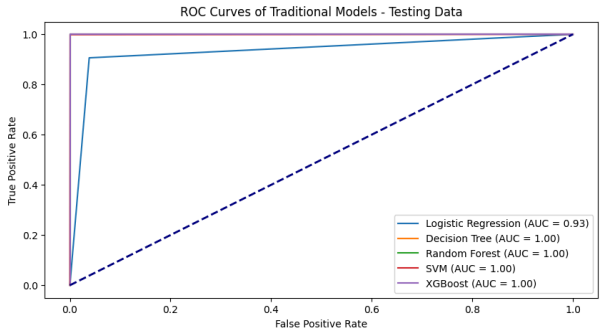

Figure 29: ROC Curve of Traditional Models - Testing Set

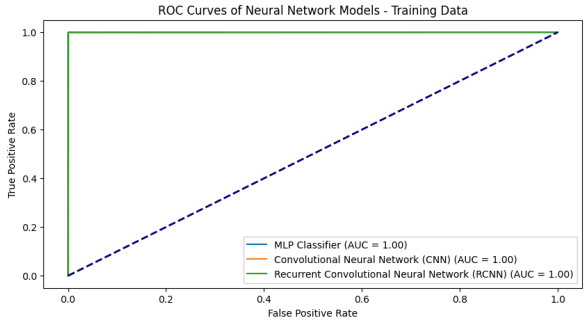

Figure 30: ROC Curve of Neural network Models - Training Set

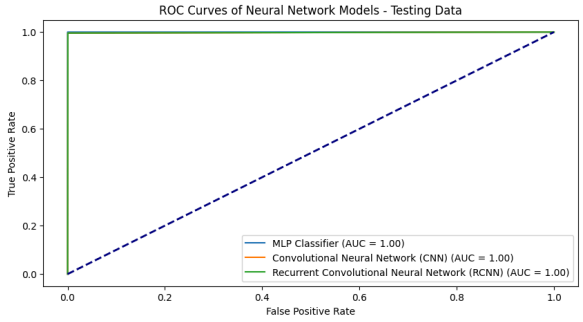

Figure 31: ROC Curve of Neural network Models - Testing Set

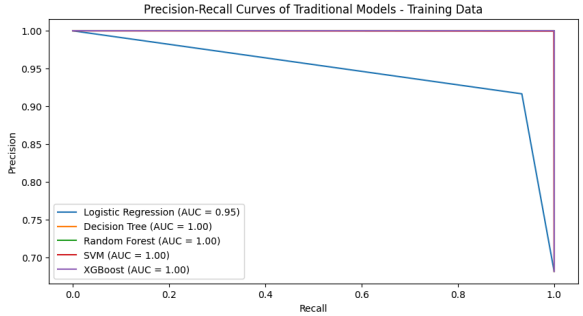

Figure 32: Precision ReCall Curve of Traditional Models - Training Set

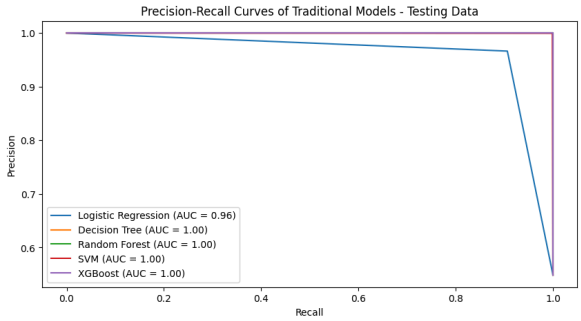

Figure 33: Precision ReCall Curve of Traditional Models - Testing Set

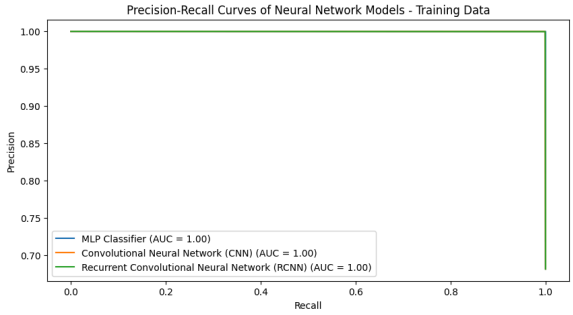

Figure 34: Precision Recall Curve of Neural network Models - Training Set

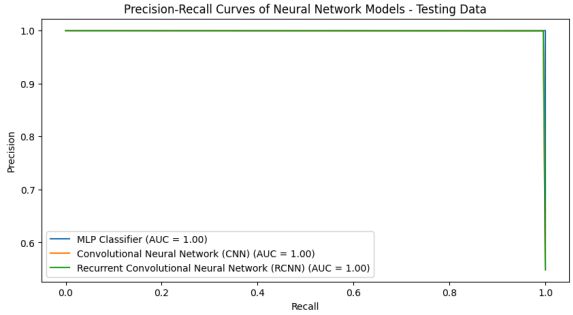

Figure 35: Precision Recall Curve of Neural network Models - Testing Set

In conclusion, both traditional and neural network models exhibit promise in network intrusion detection, with CNN and RCNN demonstrating superior performance. Further exploration of neural network learning holds potential for enhancing cybersecurity measures.

4 Conclusion

In our project, We used the UNSW-NB15 dataset to evaluate a range of traditional and neural network machine learning models for distinguishing normal

from malicious network traffic. Leveraging MLP classifier, CNN, and RCNN as neural network models, alongside logistic regression, decision tree, SVM, XGBoost, and random forest as traditional models, we conducted a comprehensive analysis.

The results underscored the superiority of neural networks, particularly RCNNs, over conventional classifiers for network intrusion detection. Our project's core contributions include exploring various neural network architectures and implementing feature selection techniques to enhance model performance. These findings suggest that neural networks hold significant promise for bolstering network security.

A key takeaway from this project is the importance of feature selection in optimizing machine learning models. By eliminating irrelevant features and reducing dimensionality, feature selection can significantly improve model efficiency and training speed.

4.1 Future Work

Our results demonstrate the effectiveness of both traditional and neural network models for network intrusion detection. However, the superior performance of CNN and RCNN models suggests the need for further exploration in neural network learning.

Future research could focus on refining deep learning models like GANs and integrating them into real-time intrusion detection systems for enhanced network security. Addressing dataset imbalances and exploring various feature engineering strategies are also promising areas for future investigation to improve model robustness and generalizability. Exploring advanced deep learning models like Generative Adversarial Networks (GANs) could further enhance the effectiveness of network intrusion detection systems. Integrating our approach into real-time detection systems could facilitate prompt identification of network intrusions, thereby enhancing network security. Addressing the challenge of imbalanced datasets in network intrusion detection through techniques like GANs could improve model training. Additionally, investigating different neural network architectures or hybrid models and exploring various feature engineering strategies are potential avenues for future research. Expanding the study to include additional datasets would also facilitate assessing the models' generalizability.

References

[1] Mamoun Alazab, Michael Hobbs, and Jemal Abawajy. Anomaly-based network intrusion detection: Techniques, systems and challenges. Computers & Security, 60:84–102, 2016.

[2] Md Rafiul Islam and Kazi Mohammed Ahmed. Machine learning approaches for network intrusion detection: A comprehensive survey. IEEE Access, 7:27459–27484, 2019.

[3] Sebastian Garcia, Markus Grill, Lukas Stiborek, Juan Manuel Zuniga, and Mariemma I AguayoTorres. An empirical comparison of botnet detection methods. Computers & Security, 45:100–123, 2014.

[4] Nour Moustafa and Jill Slay. The evaluation of network anomaly detection systems: Statistical analysis of the unsw-nb15 dataset and the comparison with the kdd99 dataset. Information Security Journal: A Global Perspective, 25(1-3):1–14, 2016.

[5] Ahmed Moustafa, Jill Slay, and Gregory Creech. Unsw-nb15: A comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In the Military Communications and Information Systems Conference (MilCIS), pages 1–6. IEEE, 2015.

[6] Sourav Mukherjee, Ananda Roy Chowdhury, Shukla Das, Sayan Chakraborty, and Mita Nasipuri. A comparative study of deep learning approaches for network intrusion detection. Future Generation Computer Systems, 107:1063– 1077, 2020.

Github Link for the Project including CodeBase and Documentation: [Project Github Repository]