APPLYING MACHINE LEARNING FOR INTRUSION DETECTION IN NETWORK SECURITY

Using Random Forest Algorithm

Submitted By
Rincy Mariam Thomas
Directory ID - **rmt21**
UID - **120124044**
Course and Section - **ENPM693 CY01**

Table of Contents

Introduction

In modern cybersecurity, intrusion detection plays a vital role in safeguarding networks from malicious activities. This project aims to utilize machine learning techniques, specifically focusing on the Random Forest algorithm, for intrusion detection in network security. The NSL-KDD dataset, a widely recognized benchmark dataset in this field, is used for training and evaluation.

Objective

The primary objective of this project is to develop an effective intrusion detection system using machine learning. Specifically, the goal is to accurately classify network traffic as either normal or intrusive, thereby enhancing network security.

Why Random Forest?

Random Forest is chosen for its effectiveness in handling classification tasks and reducing overfitting. Its ability to handle high-dimensional data with a large number of features makes it suitable for intrusion detection in network security.

Dataset

The NSL-KDD dataset is widely recognized in the field of network security and intrusion detection. It is chosen due to its popularity, relevance, and suitability as a benchmark for training and evaluating intrusion detection models.

Features:

In the feature selection, the features such as 'src_bytes', 'dst_bytes', 'protocol', 'class', etc., are chosen which provide valuable information about network connections for detecting anomalies or attacks.

Features:

- @attribute 'src_bytes' real - Amount of data sent from the source to the destination.
- @attribute 'dst_bytes' real - Amount of data sent from the destination back to the source.
- @attribute 'protocol' {'tcp','udp', 'icmp'} - Communication protocol used.

Target Variable:

- @attribute 'class' {'normal', 'anomaly'} - Identifies the class of network traffic as either 'normal' or 'anomaly', facilitating intrusion detection and security analysis.

Training and Testing Approach:

The project primarily focuses on training and testing using the provided train dataset. The test dataset will be reserved for a later stage, likely during a presentation to a professor, to validate the model's performance on unseen data.

Utilization of PCA:

PCA (Principal Component Analysis) is employed to reduce the dimensionality of the dataset while retaining most of its variance. This aids in faster computation and potentially better model performance by focusing on the most informative features.

AUC Score (Area Under Curve):

The AUC score is a crucial metric for evaluating the performance of a classification model, especially when dealing with imbalanced datasets. It represents the area under the Receiver Operating Characteristic (ROC) curve. An AUC score close to 1 indicates excellent model performance, while a score close to 0.5 suggests random guessing.

In this case, an AUC score of approximately 0.98 (0.9810941505952206) is outstanding, indicating that this Random Forest model is highly effective in distinguishing between normal and intrusive network traffic.

Receiver Operating Characteristic (ROC) Curve:

The ROC curve is a graphical representation of the trade-off between the true positive rate (TPR) and the false positive rate (FPR) at various classification thresholds.

Here's what the components of the ROC curve mean:

- X-axis (False Positive Rate): This represents the proportion of false positives (incorrectly classified normal instances) to all actual negative instances.

- Y-axis (True Positive Rate): This represents the proportion of true positives (correctly classified intrusive instances) to all actual positive instances.

- Diagonal Dotted Line: This line represents the performance of a random classifier (no discrimination). Any model above this line is better than random guessing.

- Solid Yellow Line: This line represents the ROC curve for this Random Forest model. It rises sharply toward the top left corner, indicating excellent discrimination power.

- The area under the ROC curve (AUC) is the shaded region between the diagonal line and the ROC curve. A high AUC score confirms the model's effectiveness.

Library Choices:

- Pandas: Used for data manipulation and analysis.
- Scikit-learn: Provides efficient tools for machine learning tasks, including preprocessing, model selection, and evaluation.
- Matplotlib: Utilized for data visualization.
- Tabulate: Helps in displaying results in a structured and readable format.

Detailed Code Explanation:

- Each function in the code is thoroughly documented to explain its purpose and functionality.
- Preprocessing steps include converting categorical variables into numerical representations, scaling features, and optionally applying PCA.
- The “train_model” function trains a Random Forest model using GridSearchCV for hyperparameter tuning.
- Evaluation metrics such as accuracy, confusion matrix, classification report, ROC curve, and AUC score are calculated and displayed comprehensively.
- Computational complexity for training and running the model is also provided.

Code 1: Without PCA

Importing Libraries: Import necessary libraries such as Pandas, Scikit-learn modules, Matplotlib, and Tabulate for data manipulation, model building, evaluation, and visualization.

1. Preprocessing Data Function: Define a function preprocess_data to preprocess the dataset by converting the 'class' column to binary, encoding categorical 'protocol' column, and selecting relevant features.
2. Training Model Function: Define a function train_model to split the dataset, standardize features, initialize Random Forest classifier, perform GridSearchCV for hyperparameter tuning, and return the best parameters.
3. Evaluation Model Function: Define a function evaluate_model to evaluate model performance by calculating accuracy, confusion matrix, classification report, ROC curve, AUC score, and computational complexity.
4. Display Data Info Function: Define a function display_data_info to display basic information about the dataset such as number of data points, column names, data types, and first few rows.
5. Main Function: Define the main function to load the dataset, preprocess data, train the model, and evaluate its performance.

6. Path to Dataset: Provide the path to the dataset.
7. Execution of Main Function: Execute the main function.

Code 2: With PCA

1. Importing Libraries: Similar to Code 1, import necessary libraries.
2. Preprocessing Data Function: Similar to Code 1.
3. Training Model Function: Similar to Code 1, but includes PCA transformation of features.
4. Evaluation Model Function: Similar to Code 1, but evaluates model performance using PCA-transformed features.
5. Display Data Info Function: Similar to Code 1.
6. Main Function: Similar to Code 1.
7. Path to Dataset: Similar to Code 1.
8. Execution of Main Function: Similar to Code 1.

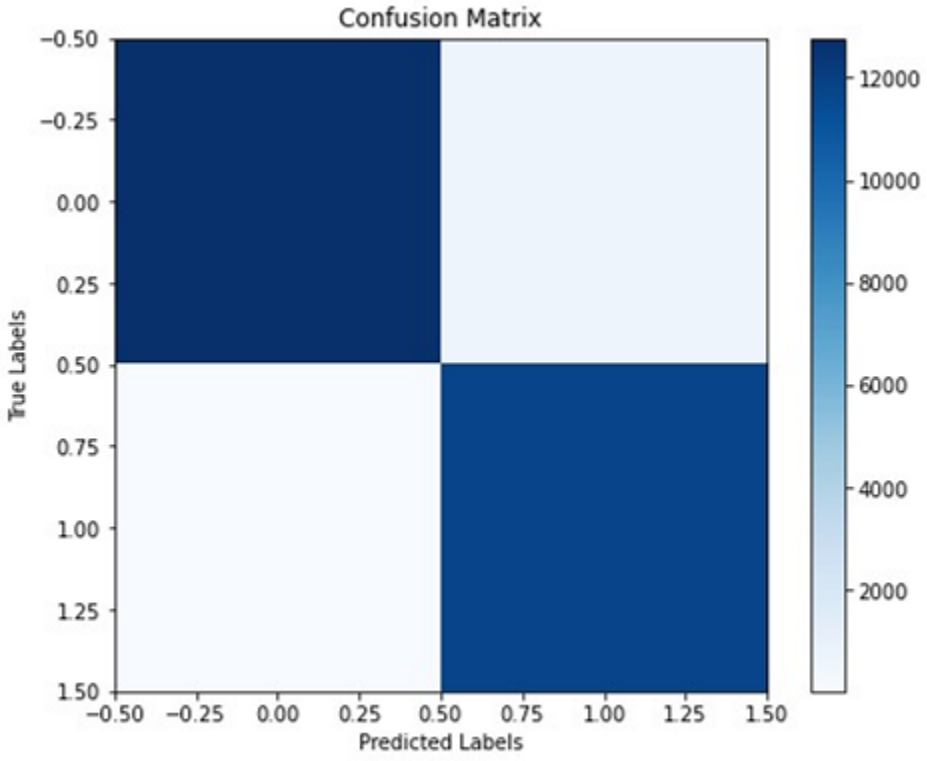
Output Analysis:

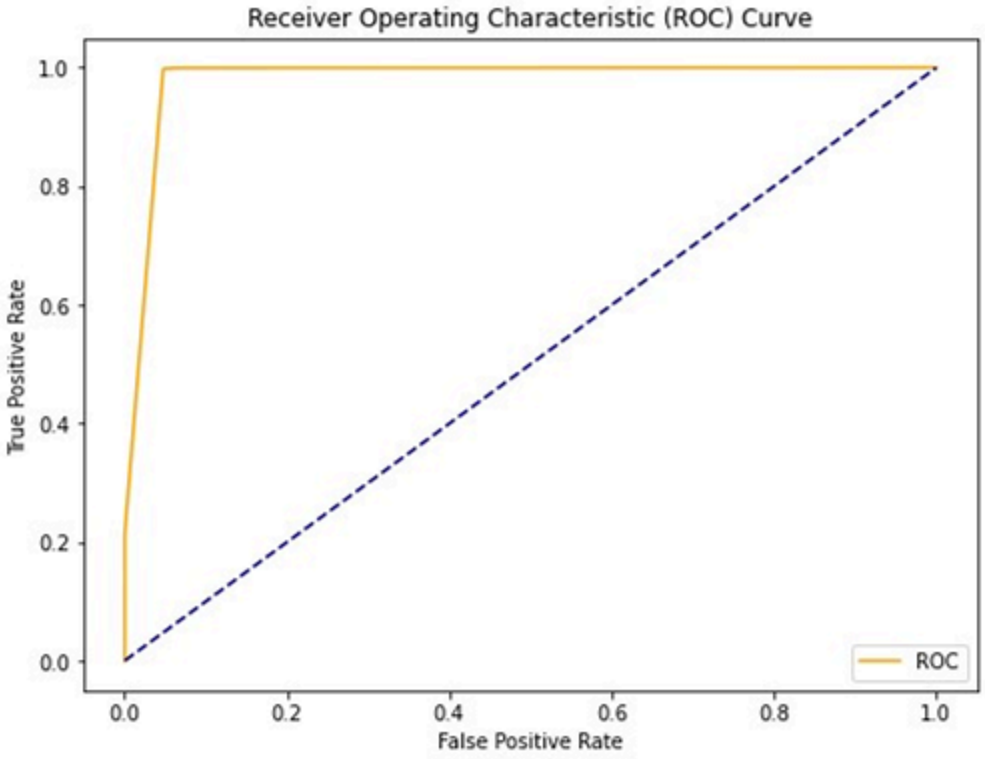
Output 1: Without PCA

```
Confusion Matrix:
+----------+-------------+-------------+
|          | Predicted 0 | Predicted 1 |
+----------+-------------+-------------+
| Actual 0 |    12771    |     651     |
| Actual 1 |     23      |    11750    |
+----------+-------------+-------------+
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.95      0.97     13422
           1       0.95      1.00      0.97     11773

    accuracy                           0.97     25195
   macro avg       0.97      0.97      0.97     25195
weighted avg       0.97      0.97      0.97     25195
```





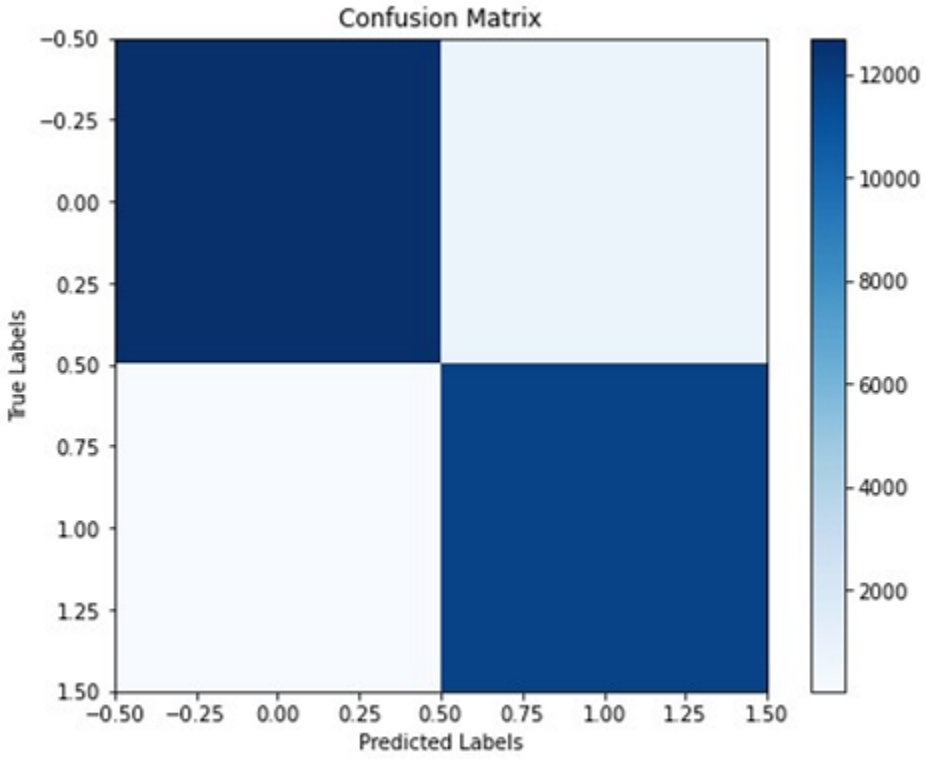

```
AUC Score: 0.9810941505952206
Average Training Complexity (seconds): 4.973517507314682
Average Running Complexity (seconds): 0.22885978718598685
```

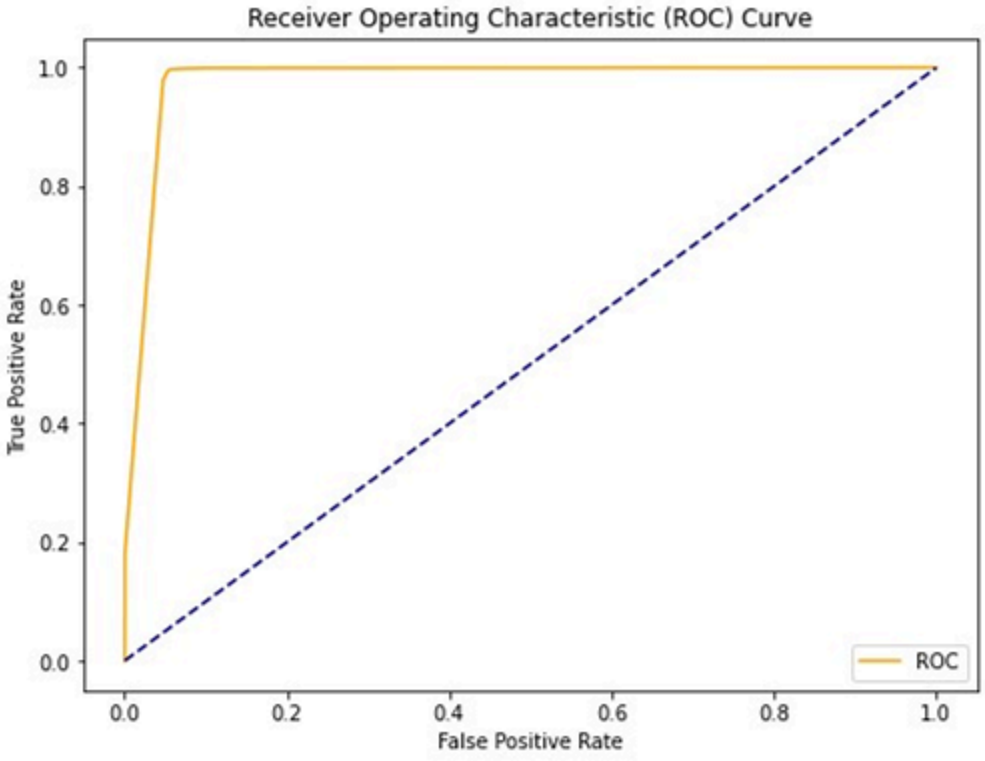
Output 2: With PCA

```
Confusion Matrix:
+----------+-------------+-------------+
|          | Predicted 0 | Predicted 1 |
+----------+-------------+-------------+
| Actual 0 |    12698    |     724     |
| Actual 1 |     45      |    11728    |
+----------+-------------+-------------+
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.95      0.97     13422
           1       0.94      1.00      0.97     11773

    accuracy                           0.97     25195
   macro avg       0.97      0.97      0.97     25195
weighted avg       0.97      0.97      0.97     25195
```






```
AUC Score: 0.979578749164822
Average Training Complexity (seconds): 5.657320469617844
Average Running Complexity (seconds): 0.2597630172967911
```

Conclusions:

- Both models, with and without PCA, achieve high accuracy and AUC scores, indicating their effectiveness in intrusion detection.
- The model with PCA achieves slightly lower accuracy but demonstrates comparable performance overall, while potentially offering computational benefits.
- Fine-tuning hyperparameters such as max depth, min samples split, and number of estimators significantly impact model performance.
- This Random Forest model is performing exceptionally well in distinguishing between normal and intrusive network traffic. It achieves a high true positive rate while maintaining a low false positive rate, which is crucial for network security applications.

Future Directions:

- Feature Engineering: Exploring and creating new features from existing ones could enhance the model's performance by capturing more relevant information from the data.
- Hyperparameter Tuning: Fine-tuning the hyperparameters of the Random Forest classifier could optimize its performance and improve its ability to generalize to unseen data.
- Handling Class Imbalance: Implementing advanced techniques to handle class imbalance, such as using different sampling methods or ensemble methods, could improve the model's ability to detect rare intrusion events.
- Trying Different Algorithms: Experimenting with other classification algorithms beyond Random Forest, such as Gradient Boosting, Support Vector Machines, or Neural Networks, could provide insights into alternative modeling approaches.
- Cross-Validation: Using cross-validation techniques to get a better estimate of the model's performance and reduce overfitting could enhance the robustness of the trained model.

References:

1. NSL-KDD Dataset: NSL-KDD Dataset
2. GitHub Repository: My Project Repository Link
3. Research Papers:
    a. https://ieeexplore.ieee.org/abstract/document/9778286
    b. https://link.springer.com/article/10.1007/s00500-021-05893-0