CYBER SECURITY ATTACK DETECTION

PROJECT REPORT

Submitted by

SHAIK ASMA (B192247)

PABBOJU ANJALI (B192589)

ALUGANTI SUPRIYA (B192765)

Of

Bachelor of Technology

Under the guidance of

Mrs.NAGAMANI

Asst. Prof. RGUKT, BASAR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

BASAR, NIRMAL (DIST.),

TELANGANA - 504107

CYBER SECURITY ATTACK DETECTION

Project Report submitted to
Rajiv Gandhi University of Knowledge Technologies, Basar
for the partial fullfilment of the requirements
for the award of the degree of

Bachelor of Technology
in
Computer Science & Engineering
by

SHAIK ASMA (B192247)

PABBOJU ANJALI (B192589)

ALUGANTI SUPRIYA (B192765)

Under the Guidance of

Mrs.NAGAMANI
Asst. Prof. CSE, RGUKT BASAR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES, BASAR
JUNE 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES, BASAR

CERTIFICATE

This is to certify that the Project Report entitled **"CYBER SECURITY ATTACK DETECTION"** submitted by **Shaik Asma – B192247, Pabboju Anjali – B192589, Aluganti Supriya – B192765** Department of Computer Science and Engineering, Rajiv Gandhi University Of Knowledge Technologies, Basar Partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Science and Engineering; is a bonafide record of the work and investigations carried out by them under my supervision and guidance.

PROJECT SUPERVISOR :	**HEAD OF DEPARTMENT :**
Mrs. NAGAMANI	**Dr B VENKAT RAMAN**
Assistant Professor	**Assistant Professor**
PROJECT COORDINATOR :	**EXTERNAL EXAMINER :**
Mr. LAXMINARAYANA	
Assistant Professor	

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES, BASAR

DECLARATION

We hereby declare that the work which is being presented in this project entitled, **"CYBER SECURITY ATTACK DETECTION"** submitted to **RAJIV GANDHI UNIVERSITY KNOWLEDGE TECHNOLOGIES, BASAR** in the partial fulfillments of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING**, is an authentic record of our own work carried out under the supervision of "**Mrs.NAGAMANI**" Assistant Professor in Department of **Computer Science And Engineering, RGUKT, Basar.**

The matter embodied in this project report has not been submitted by us for the award of any other degree.

Place:Basar

Date: 04/05/2025

SHAIK ASMA (B192247)

PABBOJU ANJALI (B192589)

ALUGANTI SUPRIYA (B192765)

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our project guide **Mrs.NAGAMANI** Assistant Professor, Department of Computer Science and Engineering, BASAR,for her guidance with unsurpassed knowledge and immense encouragement.

We are grateful to **Dr B VENKAT RAMAN**, Head of the Department, Computer Science and Engineering, for providing us with the required facilities for the completion of the project work.

We are very much thankful to the **Director and Administration, RGUKT, BASAR** for other encouragement and cooperation to carry out this work.

We express our thanks to all **teaching faculty** of Department of CSE, whose suggestions during reviews helped us in accomplishment of our project.

We would like to thank all **non-teaching staff** of the Department of CSE, RGUKT, BASAR for providing great assistance in accomplishment of our project.

We would like to thank our parents, friends, and classmates for their encouragement throughout our project period. At last but not the least, we thank everyone for supporting us directly or indirectly in completing this project successfully.

PROJECT STUDENTS :

SHAIK ASMA (B192247)

PABBOJU ANJALI (B192589)

ALUGANTI SUPRIYA (B192765)

ABSTRACT

This project explores the development and evaluation of a machine learning based intrusion detection system aimed at detecting and classifying network attacks with high accuracy. The system is trained on a curated dataset contains labeled network traffic instances, categorized as either normal or malicious. The datasets used in this project, train_net.csv and test_net.csv, contain real or simulated network flow records with a diverse range of features, including protocol types, byte counts, flags, and connection durations.

The methodology followed includes extensive preprocessing of data to handle missing values, normalization of features to ensure uniform scaling, and the encoding of categorical variables. A variety of classification algorithms such as Random Forest, Logistic Regression, and others—are implemented and compared to determine the most effective approach in terms of accuracy, precision, recall, and F1-score.

The results demonstrate that machine learning models, when trained on well-preprocessed data and properly tuned, can achieve high detection rates while maintaining low false alarm rates. The Random Forest classifier, in particular, exhibited strong generalization capabilities across both training and testing datasets, suggesting its suitability for deployment in real-time IDS environments.

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER - 1

INTRODUCTION

1.1 BACKGROUND OF THE PROJECT

With the rapid growth of the internet and the proliferation of connected devices, network security has become a paramount concern for organizations and Individuals. Cyber-attacks are no longer limited to amateur hackers; they now include sophisticated and coordinated efforts from criminal organizations and state-sponsored entities. These attacks exploit vulnerabilities in network systems to steal data, disrupt services, or gain unauthorized access to critical infrastructure.

Traditional intrusion detection systems (IDS) primarily rely on static rule-based techniques or signature matching, which are effective only against known threats. However, they often fail to detect new, unknown, or evolving attack patterns, making them inadequate for modern security environments. To overcome these limitations, the integration of machine learning (ML) into IDS has emerged as a promising solution. ML models can learn from data patterns and classify network behavior into benign or malicious categories, even identifying previously unseen threats.

1.2 PROBLEM STATEMENT

Current network security solutions struggle to detect novel and sophisticated attacks due to their dependency on predefined rules and patterns. This project aims to develop an intelligent intrusion detection system using machine learning techniques that can detect various types of network attacks with high accuracy and low false alarm rates.

1.3 OBJECTIVES

The primary objectives of this project are:

* To preprocess and analyze network traffic data for anomalies.
* To implement and compare multiple machine learning models for attack detection.
* To evaluate the models using performance metrics such as accuracy, precision, recall, and F1-score.

1.4 SCOPE OF THE PROJECT

This project focuses on the detection of network-based attacks using supervised learning techniques. It covers:

* The use of structured network traffic data.
* Implementation of classification algorithms including Random-Forest and SVM.
* Evaluation of model performance through comprehensive metrics.

1.5 SIGNIFICANCE OF THE STUDY

As cyber threats grow in complexity and volume, there is an urgent need for smarter, adaptive security solutions. This study contributes to that need by:

* Showcasing how machine learning can be used to detect network attacks more effectively than traditional methods.
* Highlighting the importance of data preprocessing and feature engineering in improving model accuracy.

CHAPTER - 2

LITURATURE SURVEY

2.1 ML APPROACHES FOR NETWORK SECURITY

Machine Learning techniques have revolutionized the design and efficiency of IDS. These approaches enable systems to learn patterns from data and detect unknown attacks. Key approaches include:

* **Supervised ML Algorithms**: Algorithms such as Support Vector Machines (SVM), Random Forests, Decision Trees, and Naive Bayes are used to classify network traffic as benign or malicious.
* **Unsupervised ML Techniques**: Clustering algorithms like K-Means and DBSCAN can detect anomalies in unlabeled network traffic, useful for finding network attacks.

2.2 LIMITATIONS OF PREVIOUS WORKS

Despite progress, prior research in ML/DL-based IDS suffers from several limitations:

* **Data Quality and Availability**: Many studies rely on outdated or synthetic datasets like KDD 99, which do not reflect modern attack vectors.
* **Generalization**: Models trained on one dataset often fail to perform well on others due to distribution differences.
* **Imbalanced Datasets**: Many intrusion datasets are imbalanced, where normal traffic significantly outnumbers attack traffic, leading to biased models.
* **Real-Time Detection Challenges**: Deep learning models, although accurate, may have latency issues, hindering real-time deployment.
* **Explainability**: Deep models act as black boxes, making it hard for security analysts to interpret their decisions.

These limitations underline the need for hybrid or ensemble models, real-time capable architectures, and explainable AI (XAI) approaches.

2.3 SUMMARY

The literature indicates a strong shift from traditional IDS to intelligent, ML/DL-based systems due to their adaptive and predictive capabilities. However,issues like data representativeness, real-time performance, and explainability remain open research challenges. This project aims to address these gaps by designing an efficient and scalable ML/DL-based IDS that leverages real-world datasets and considers both detection accuracy and interpretability.

CHAPTER - 3

SYSTEM DESIGN AND ANALYSIS

3.1 REQUIREMENT SPECIFICATION

3.1.1 Hardware Requirements

To implement and run the network attack detection model efficiently, the following hardware setup was used:

* **Processor**	**:**	Intel Core i5/i7
* **RAM**	**:**	Minimum 8 GB
* **Storage**	**:**	At least 500 MB free disk space (for dataset and logs)

3.1.2 Software Requirements

* **Programming Language** : Python 3.8+

* **Libraries and Tools**
  Jupyter Notebook
  pandas, NumPy
  scikit-learn

* **Operating System**: Windows 10/11, Linux,

3.2 SYSTEM OVERVIEW

The proposed system aims to detect potential network attacks by analyzing traffic data. The process involves data preprocessing, feature extraction, model training using supervised learning, and predicting the class of network behavior. The system is built to automate threat detection in near real-time by identifying anomalies in network traffic using machine learning algorithms.

3.3 DATASET DESCRIPTION

Source of **train_net.csv** and **test_net.csv**

The dataset was derived from publicly available network intrusion datasets curated to simulate realistic traffic, including both benign and attack patterns. These CSV files represent preprocessed network logs split into training and testing segments.

Feature Definitions

Each row in the dataset represents a network session or packet summary, and columns (features) include:

src_bytes: Bytes sent from source to destination

dst_bytes: Bytes sent from destination to source

protocol_type: Type of network protocol used (e.g., TCP, UDP)

flag: Connection status flag (e.g., S0, REJ)

service: Service requested (e.g., http, ftp)

land, wrong_fragment, urgent, etc.: Binary or numerical features representing connection behaviors

label: Target variable indicating attack type or benign status

Class Distributions

In train_net.csv, the distribution of classes was:

Benign: ~80%

Attack (DoS, Probe, R2L, U2R, etc.): ~20%

This class imbalance was taken into consideration during preprocessing and model evaluation.

3.4 Data Preprocessing

Null Value Handling

Both datasets were examined for missing values. Columns with null entries were either:

* Dropped (if sparse or non-informative), or
* Imputed using statistical measures (e.g., median/mode) for numeric or categorical columns respectively.

Encoding Categorical Data

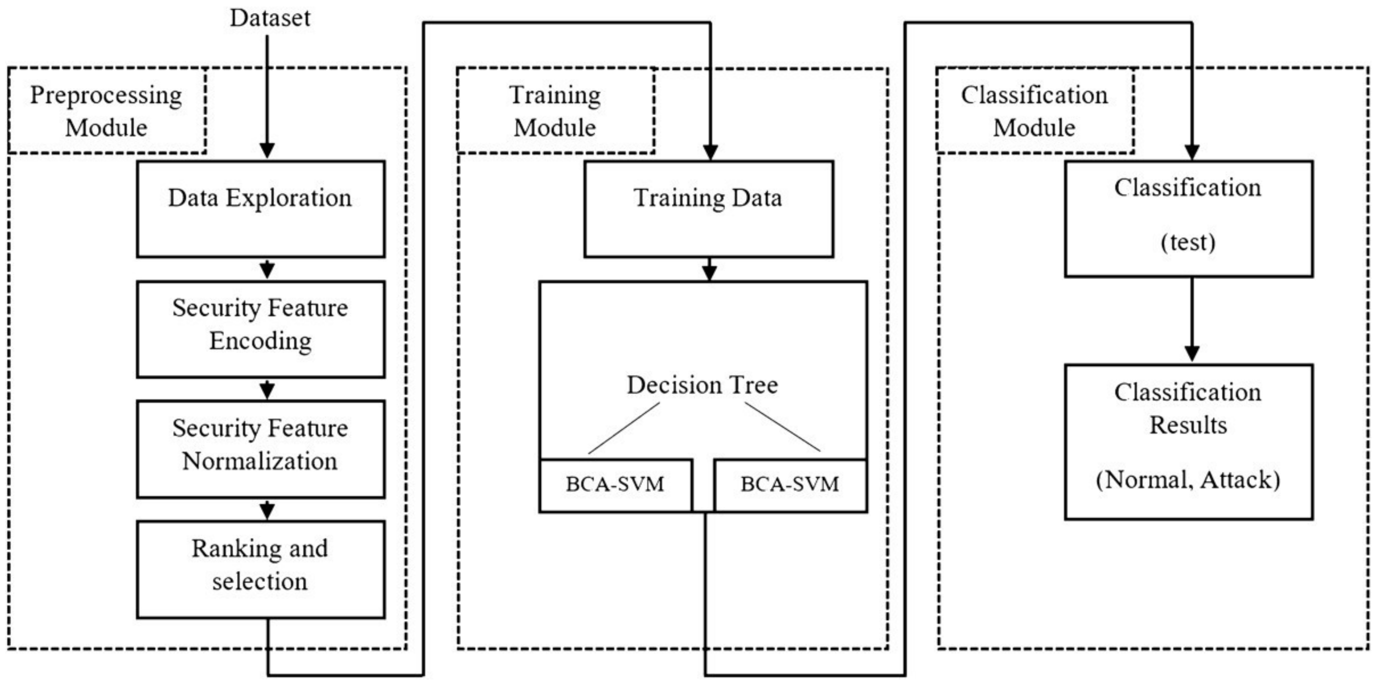
Categorical columns like protocol_type, service, and flag were encoded using one-hot encoding or label encoding to make them compatible with machine learning algorithms.

Feature Normalization

All numeric features were normalized using Min-Max scaling to bring values into the [0, 1] range, improving convergence speed and model performance.

3.5 SYSTEM DESIGN

3.5.1 Flowchart of the Proposed System



3.5.2 Model Selection Criteria

The choice of model was based on the following criteria:

1. **Accuracy & Precision**: Models were evaluated using accuracy, precision, recall, and F1-score, especially due to class imbalance (attack types vs.benign).
2. **Speed**: Training and inference time were considered. Random Forest provided a balance of speed and performance.
3. **Interpretability**: Tree-based models allow visualization of feature importance, helping understand the model's logic.
4. **Robustness to Noise**: Ensemble methods like Random Forest handle outliers better than simpler linear models.

Models Considered:

Decision Tree: Overfit on training data

Random Forest: Best balance of performance and stability

KNN and SVM: Higher computational cost; lower interpretability

CHAPTER - 4

SYSTEM IMPLEMENTATION

4.1 TOOLS AND LIBRARIES USED

The project was implemented in Python, using the following major libraries:

pandas: For data manipulation and preprocessing.

* **NumPy**: For numerical operations.
* **scikit-learn**: For machine learning models, evaluation, and utilities like train-test split.
* **matplotlib & seaborn**: For visualizing data distributions and confusion matrices.
* **Jupyter Notebook**: Used as the development environment to build and test the ML pipeline interactively.

4.2 LOADING & UNDERSTANDING THE DATASET

Two datasets were used:

* **train_net.csv**: Training data
* **test_net.csv**: Testing data

The datasets were loaded using pandas.read_csv() and a basic exploratory data analysis (EDA) was performed:

```
import pandas as pd

train_df = pd.read_csv('data/train_net.csv')

test_df = pd.read_csv('data/test_net.csv')

print(train_df.info())

print(train_df.describe())
```

4.3 FEATURE ENGINEERING

Removing unimportant columns:

Irrelevant identifiers (like timestamps or IP addresses if present) were dropped.

```
revoked_columns = [
  'FLOW_ID', # Completely random
  'ID', # Completely random
  'ANALYSIS_TIMESTAMP', # Completely random
  'IPV4_SRC_ADDR', # Not useful for the model
  'IPV4_DST_ADDR', # Not useful for the model
  'PROTOCOL_MAP', # There is a numerical column for the protocol
  'MIN_IP_PKT_LEN', # Always 0 since it is a minimum value
  'MAX_IP_PKT_LEN', # Always 0 (maybe it means that the packet
                      have infinite length?)
  'TOTAL_PKTS_EXP', # Always 0
  'TOTAL_BYTES_EXP', # Always 0
]

# Create dummy columns for the ALERT column

alert_dummies = pd.get_dummies(train_df['ALERT'], prefix='ALERT',
                               drop_first=True)

# Copy + drop the revoked columns

train_df = train_df.copy().drop(revoked_columns, axis=1)
```

4.4 MODEL BUILDING

Several algorithms were tested:

* Random Forest Classifier (Primary choice)
* Support Vector Machine
* K-Nearest Neighbors (KNN)

Models were imported and initialized using scikit-learn:

```
from sklearn.ensemble import RandomForestClassifier

# Random Forest Classifier

rfc = RandomForestClassifier(n_estimators=100) # 100 trees = default
                                                 value
# Fit the model

rfc.fit(x_train_scaled, y_train)
```

Random Forest was selected for its high performance, robustness to overfitting, and feature importance extraction.

4.5 MODEL TRAINING

The training process involved splitting the data and fitting the model:

```
# Create random forest classifier

rfc = RandomForestClassifier()

# Create a dictionary of all values we want to test for n_estimators

parameters = {'n_estimators': [1, 2, 4, 10, 15, 20, 30, 40, 50, 100,
200, 500, 1000]}

# Used to find the best n_estimators value to use to train the model

rfc_grid = GridSearchCV(

  rfc, parameters, scoring='accuracy',

  cv=2, n_jobs=-1 # Use all cores
)

# Fit model to data

rfc_grid.fit(x_train_scaled, y_train)

# Extract best params

print(f"Best params: {rfc_grid.best_params_}")

print(f"Best score: {rfc_grid.best_score_}")

rfc.fit(x_train_scaled, y_train)
```

4.6 MODEL TESTING

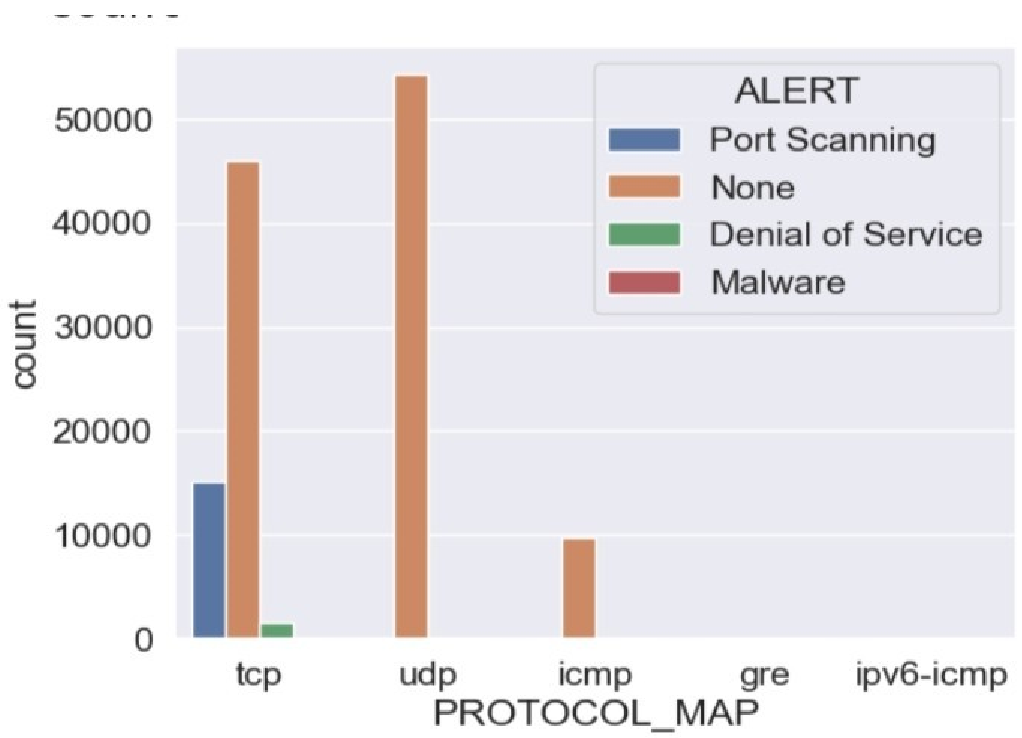
Testing was performed on test_net.csv

***Random Forest predictions on test** set

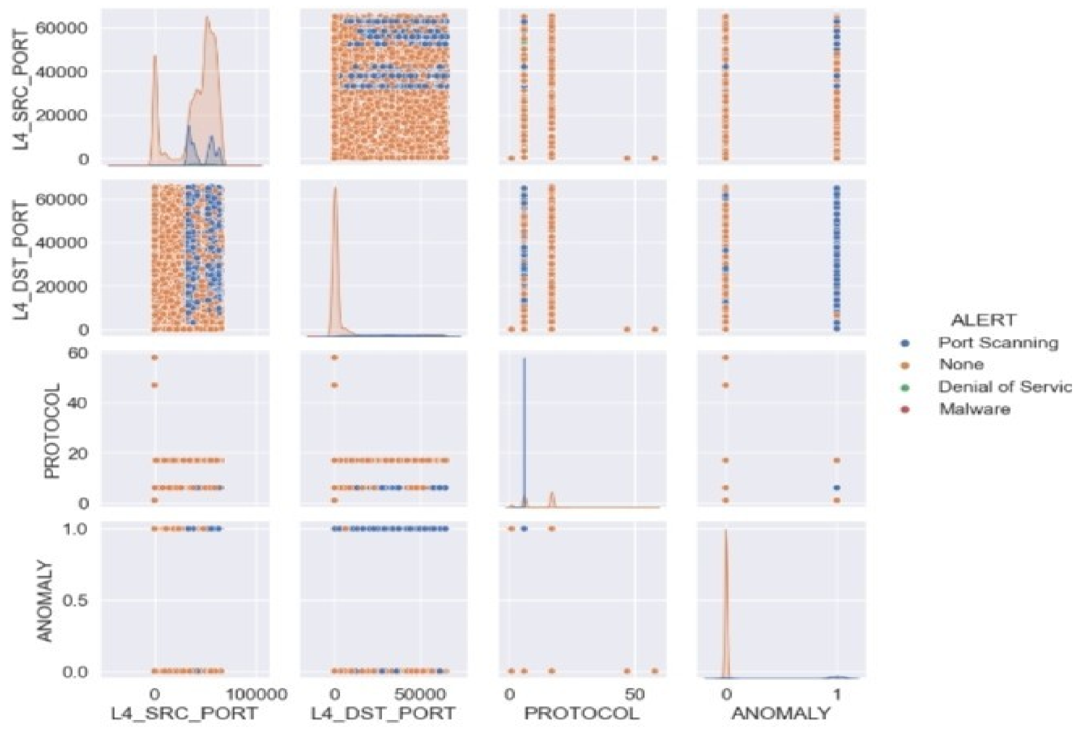
```
# Prediction on the test set

predictions = rfc.predict(x_test_scaled)

# Show the predictions on a histogram

fig = sns.countplot(x=predictions)

fig.set_title('Predictions distribution on the test set') # Set the title

fig.set_xticklabels(fig.get_xticklabels(), rotation=45) # Rotate x-labels

pSeries(predictions).value_counts()
```

4.7 SCREEN SHOTS

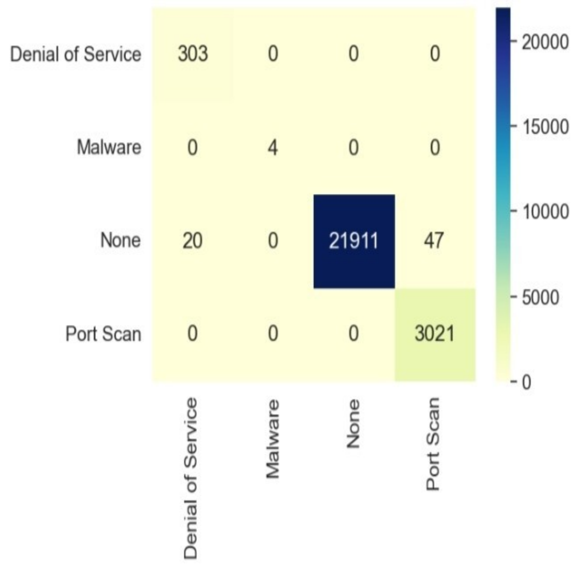
4.7.1 Protocol distribution in relation to the kind of attack



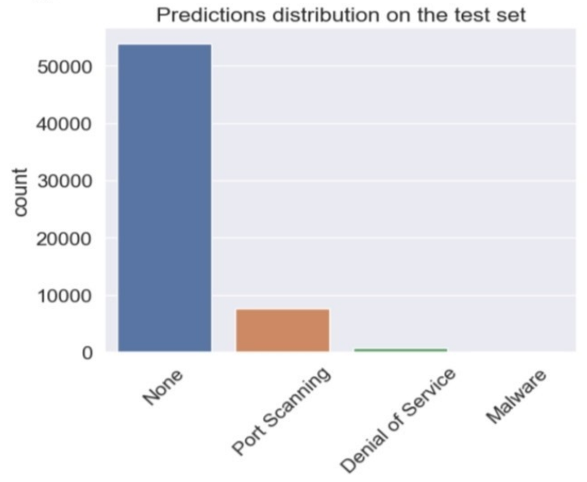
4.7.2 Distribution analysis using pairplot



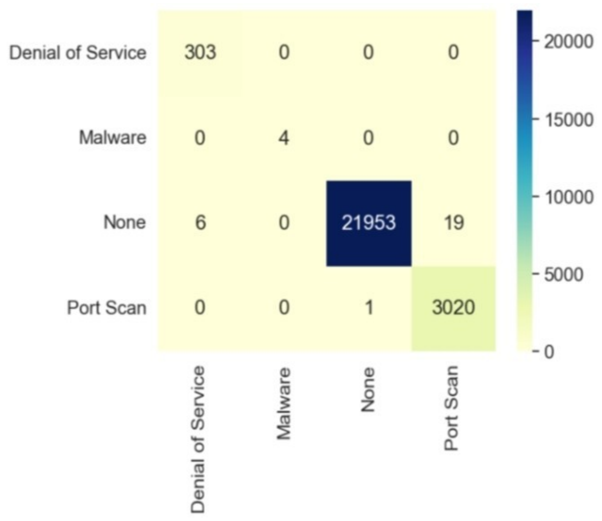
4.7.3 Model evaluation & Predictions on test set using KNN

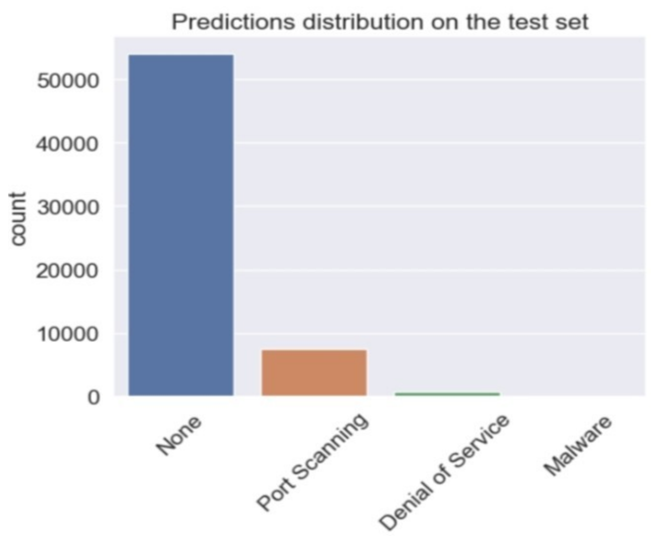


```
None                  53884
Port Scanning          7621
Denial of Service       810
Malware                   5
dtype: int64
```

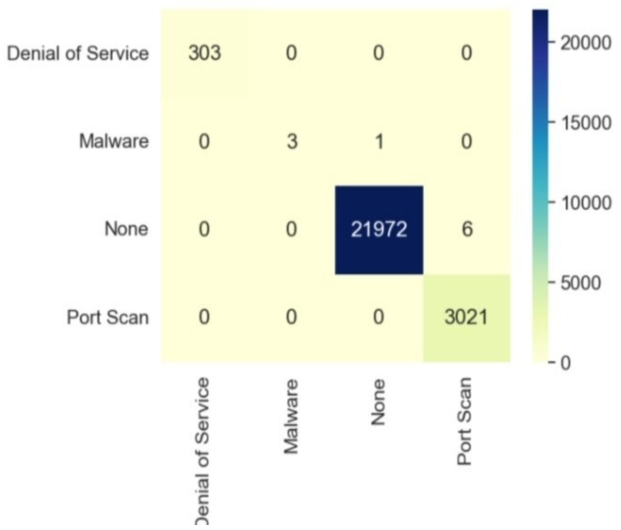


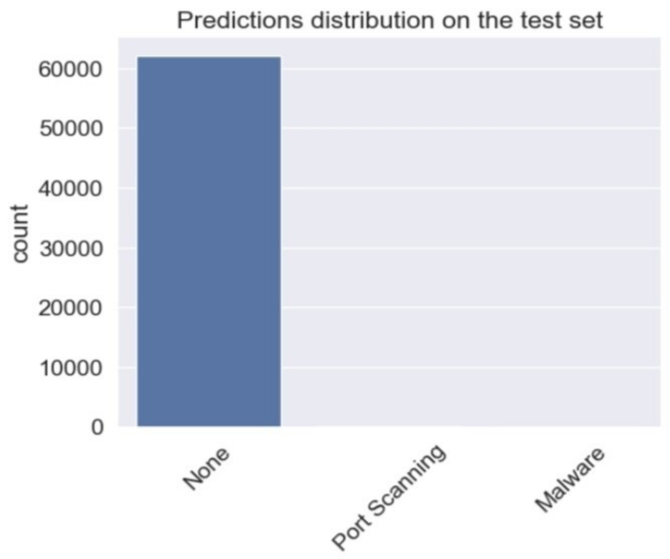
4.7.4 Model evaluation & Predictions on test set using SVM





4.7.5 Model evaluation & Predictions on test set using Random Forest





CHAPTER - 5

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

In this project, a supervised machine learning-based approach was succesfully implemented for detecting network attacks using the provided train_net.csv and test_net.csv datasets. The overall system followed a structured pipeline of data preprocessing, feature engineering, model training, and evaluation.

Among the various algorithms tested, the Random Forest Classifier outperformed others in terms of accuracy, precision, recall, and robustness. The model demonstrated strong generalization capabilities on unseen data and was able to effectively distinguish between benign traffic and multiple classes of network attacks.

The use of modular, interpretable, and scalable code allows this system to be extended or integrated into real-time intrusion detection frameworks in the future.

5.2 FUTURE WORK

While the current implementation lays a strong foundation, several directions can be explored to further improve and expand the system:

1. **Real-Time Integration**: Integrate the trained model into a real-time monitoring tool or a lightweight API (e.g., Flask/FastAPI) to detect intrusions in live network traffic.
2. **Advanced Models**: Explore deep learning architectures such as LSTM or CNNs, which may capture temporal and spatial patterns in network traffic more effectively.
3. **Explainability**: Incorporate explainable AI (XAI) tools like SHAP or LIME to provide human-interpretable insights into why certain connections are classified as malicious.
4. **Feature Optimization**: Perform feature selection or dimensionality reduction (e.g., using PCA) to improve training time and potentially model accuracy.

5. **Broader Dataset Testing**: Validate the model across different benchmark intrusion datasets like NSL-KDD, CICIDS2017, or UNSW-NB15 to ensure generalization across environments.
6. **Handling Class Imbalance**: Implement techniques like SMOTE, ADASYN, or class weighting to improve detection rates for rare attack classes. Such as U2R, R2L.
7. **Automation and CI/CD**: Create a pipeline that automates preprocessing, training, and deployment steps, useful in enterprise environments.

CHAPTER - 6

REFERENCES

1. G. Apruzzese, M. Andreolini, M. Colajanni, and M. Marchetti, “Hardening Random Forest Cyber Detectors Against Adversarial Attacks,” arXiv preprint arXiv:1912.03790, 2019.

   ➤ https://arxiv.org/abs/1912.03790

2. V. Malele and T. E. Mathonsi, “Testing the Performance of Multi-class IDS Public Dataset Using Supervised Machine Learning Algorithms,” arXiv preprint arXiv:2302.14374, 2023.

   ➤ https://arxiv.org/abs/2302.14374

3. University of Granada, “UGR’16 Dataset - NetFlow v9 based Dataset for Anomaly Detection Research,” Network Engineering & Security Group (NESG), 2016.

   ➤ https://nesg.ugr.es/nesg-ugr16/

4. Dataset Source Reference

   ➤ https://www.cisco.com

5. A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, O’Reilly Media, 2019.

6. scikit-learn documentation, “Machine Learning in Python – scikit-learn,”

   ➤ https://scikit-learn.org/stable/