DEEP LEARNING Project#2 REPORT

1- 201805020 - Begüm İŞ
2- 201805070 - Emrah FİDAN

1. Introduction

Denial of Service (DoS) attacks remain one of the most significant threats to web services and network infrastructure. These attacks attempt to overwhelm systems by flooding them with excessive traffic, rendering services unavailable to legitimate users. With the increasing sophistication of attack methods, particularly in HTTP-based DoS attacks, traditional detection methods often prove insufficient. This project explores various machine learning and deep learning approaches to detect and classify DoS attacks, leveraging both traditional classification algorithms and advanced neural network architectures. By comparing different methodologies, we aim to identify the most effective approach for real-time DoS attack detection in HTTP traffic.

2. Data Preprocessing and Feature Engineering

The dataset is loaded from 'DoS_Attack_HTTP_Dataset.csv'. Initial exploration includes examining the dataset's structure, checking for missing values, duplicates, and basic statistical summaries.

2.1 Exploratory Data Analysis (EDA)

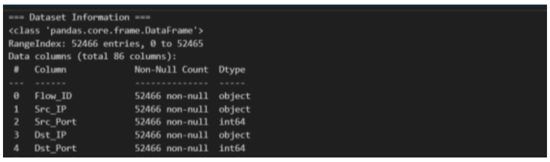
The analysis includes examination of:

- Dataset shape and dimensions
- General information about data types
- Statistical summaries of numerical features
- Memory usage analysis
- Missing value detection
- Duplicate row identification
- Unique value counts for each column

Dataset

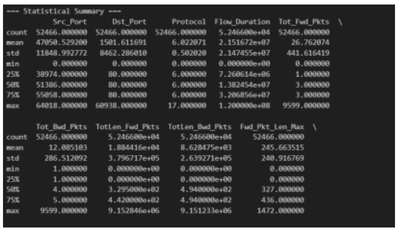
```
print("Dataset Shape:", df.shape)

Dataset Shape: (52466, 86)
```

General information about data types



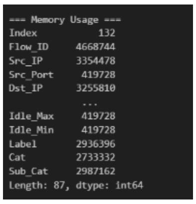
```
=== Dataset Information ===
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52466 entries, 0 to 52465
Data columns (total 86 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Flow_ID       52466 non-null  object
 1   Src_IP        52466 non-null  object
 2   Src_Port      52466 non-null  int64
 3   Dst_IP        52466 non-null  object
 4   Dst_Port      52466 non-null  int64
```

Statistical summaries



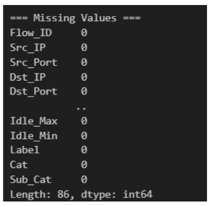
```
=== Statistical Summary ===
           Src_Port      Dst_Port      Protocol  Flow_Duration  Tot_Fwd_Pkts  \
count  52466.000000  52466.000000  52466.000000   5.246600e+04  52466.000000
mean   47050.529200   1501.611691      6.022071   2.151672e+07     26.762074
std    11848.992772   8462.286010      0.502020   2.147455e+07    441.616419
min        0.000000      0.000000      0.000000   0.000000e+00      0.000000
25%    38974.000000     80.000000      6.000000   7.260614e+06      1.000000
50%    51386.000000     80.000000      6.000000   1.382454e+07      3.000000
75%    55058.000000     80.000000      6.000000   3.206856e+07      3.000000
max    64018.000000  60938.000000     17.000000   1.200000e+08   9599.000000

       Tot_Bwd_Pkts  TotLen_Fwd_Pkts  TotLen_Bwd_Pkts  Fwd_Pkt_Len_Max  \
count  52466.000000     5.246600e+04     5.246600e+04     52466.000000
mean      12.085103     1.884416e+04     8.628475e+03       245.663515
std      286.512092     3.796717e+05     2.639271e+05       240.916769
min        1.000000     0.000000e+00     0.000000e+00         0.000000
25%        1.000000     0.000000e+00     0.000000e+00         0.000000
50%        4.000000     3.295000e+02     4.940000e+02       327.000000
75%        5.000000     4.420000e+02     4.940000e+02       436.000000
max     9599.000000     9.152846e+06     9.151233e+06      1472.000000
```

Memory usage analysis



```
=== Memory Usage ===
Index           132
Flow_ID     4668744
Src_IP      3354478
Src_Port     419728
Dst_IP      3255810
             ...
Idle_Max     419728
Idle_Min     419728
Label       2936396
Cat         2733332
Sub_Cat     2987162
Length: 87, dtype: int64
```

Missing Values



```
=== Missing Values ===
Flow_ID     0
Src_IP      0
Src_Port    0
Dst_IP      0
Dst_Port    0
           ..
Idle_Max    0
Idle_Min    0
Label       0
Cat         0
Sub_Cat     0
Length: 86, dtype: int64
```

Duplicated row identification

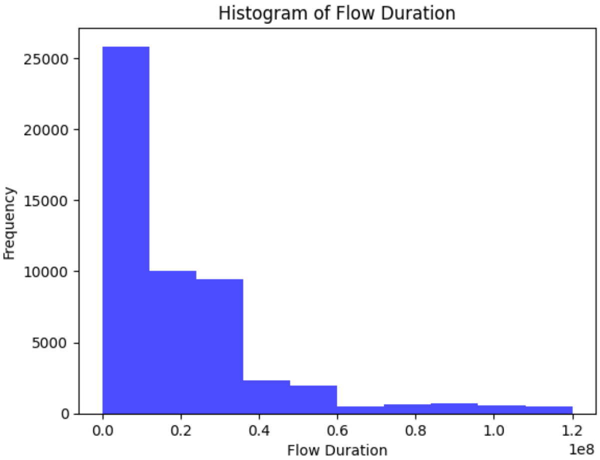
```
=== Duplicate Rows ===
0
```

2.2 Visualization Analysis

Various visualization methods were used to understand how the data is distributed.

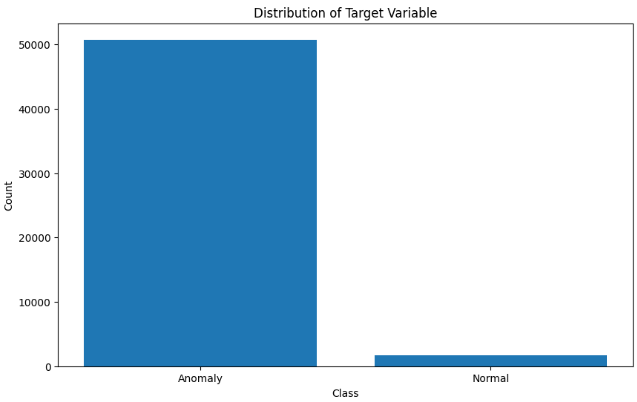
2.2.1 Flow Duration Analysis

Histogram visualization of flow duration to understand the distribution of network traffic time spans



2.2.2 Target Variable Distribution

Bar plot showing the distribution of attack vs. normal traffic

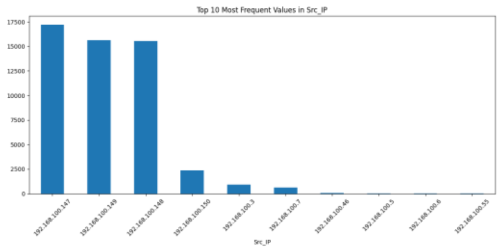


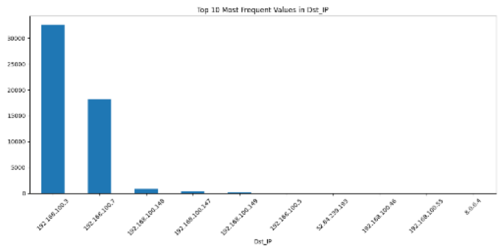
2.2.3 Categorical Features Analysis:

Bar charts for categorical columns.

Top 10 most frequent values visualization for each categorical feature.

Here are two examples:

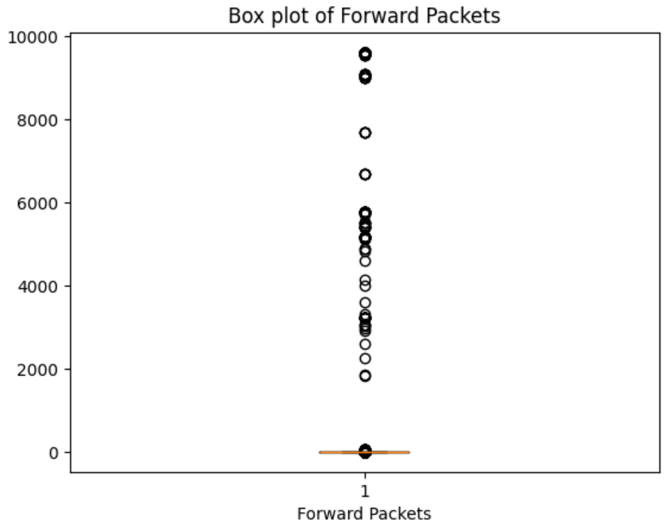




2.2.4 Network Traffic Patterns:

Box plot analysis of forward packets to identify potential outliers

Distribution patterns of network traffic characteristics



2.3 Feature Engineering

The analysis implements several sophisticated feature engineering techniques to create new meaningful features:

1. Packet Rate Calculation:
   - Derived feature calculating packets per second (Tot_Fwd_Pkts / Flow_Duration)
2. Packet Size Distribution:
   - Ratio of forward packets to total packets
   - Helps identify unusual traffic patterns
3. Protocol Frequency Analysis:
   - Calculation of protocol frequency distribution
   - Normalized protocol occurrence rates
4. Data Quality Management:
   - Handling of infinite values
   - Replacement of NaN values

```python
#  Feature Engineering
def engineer_features(df):
    # Create copy to avoid modifying original dataframe
    df_engineered = df.copy()

    # Create new feature: Packet rate (packets per second)
    df_engineered['Packet_Rate'] = df_engineered['Tot_Fwd_Pkts'] / df_engineered['Flow_Duration']

    # Create new feature: Packet size distribution
    df_engineered['Packet_Size_Distribution'] = df_engineered['Tot_Fwd_Pkts'] / \
        (df_engineered['Tot_Fwd_Pkts'] + df_engineered['Tot_Bwd_Pkts'])

    # Create new feature: Protocol frequency
    protocol_counts = df_engineered['Protocol'].value_counts(normalize=True)
    df_engineered['Protocol_Frequency'] = df_engineered['Protocol'].map(protocol_counts)

    # Handle infinite values
    df_engineered = df_engineered.replace([np.inf, -np.inf], np.nan)
    df_engineered = df_engineered.fillna(0)

    return df_engineered

# Apply feature engineering
df_engineered = engineer_features(df)
print("Shape after feature engineering:", df_engineered.shape)
print("\nNew features preview:")
print(df_engineered[['Packet_Rate', 'Packet_Size_Distribution', 'Protocol_Frequency']].head())
```

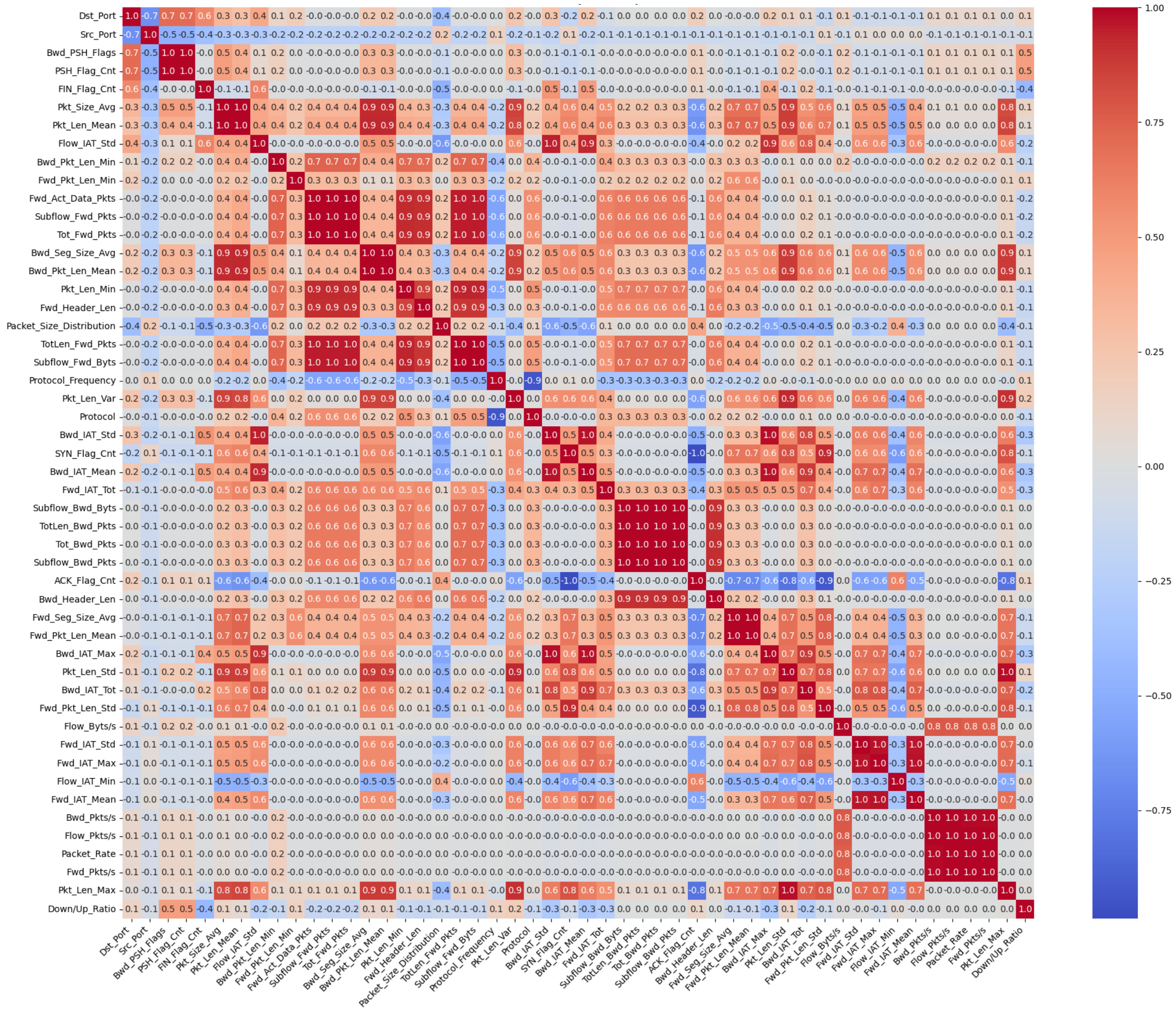
```
Shape after feature engineering: (52466, 89)

New features preview:
    Packet_Rate  Packet_Size_Distribution  Protocol_Frequency
0  5.995506e-07                     0.375            0.997846
1  5.994947e-07                     0.375            0.997846
2  5.995809e-07                     0.375            0.997846
3  5.996144e-07                     0.375            0.997846
4  5.996432e-07                     0.375            0.997846
```

2.4 Future Selection

Two primary feature selection methods were implemented:

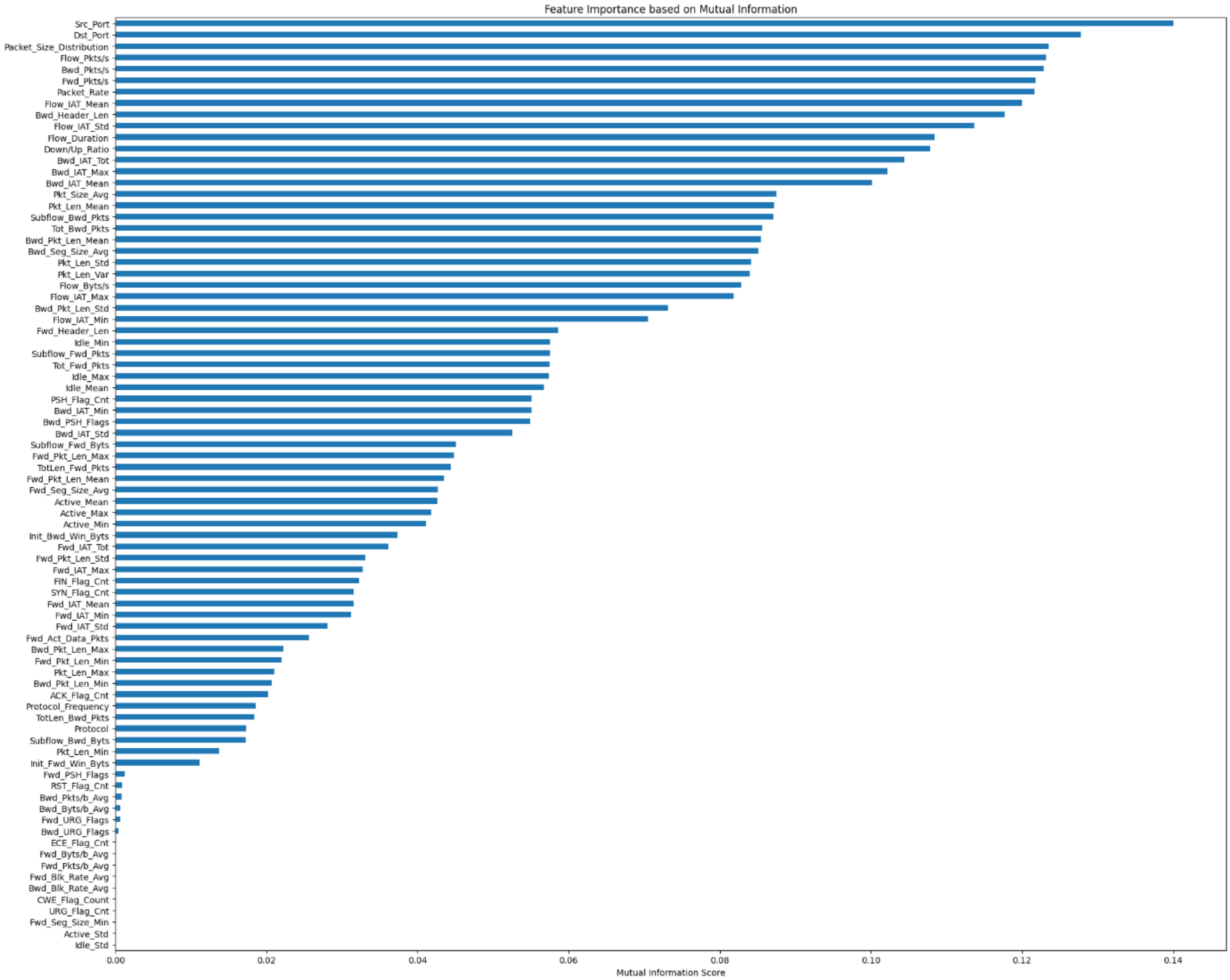
2.4.1. Correlation-Based Selection

- Generation of correlation matrix for numerical features
- Selection of top 50 features based on correlation with target variable
- Visualization using a heatmap for feature relationships
- Identification of highly correlated features



2.4.2 Mutual Information Selection

- Implementation of mutual_info_classif for feature importance scoring
- Visualization of feature importance scores
- Selection of top 50 features based on mutual information



2.5 Final Preprocess

The final preprocessing steps include:

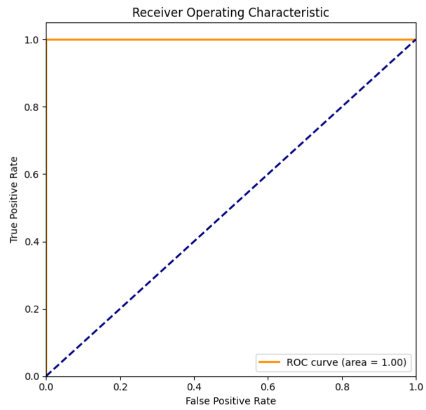
1. Feature Standardization:
   - Implementation of StandardScaler for numerical features
   - Normalization of feature scales
2. Label Encoding:
   - Conversion of categorical target variable to numerical format
3. Data Export:
   - Saving of preprocessed features to CSV

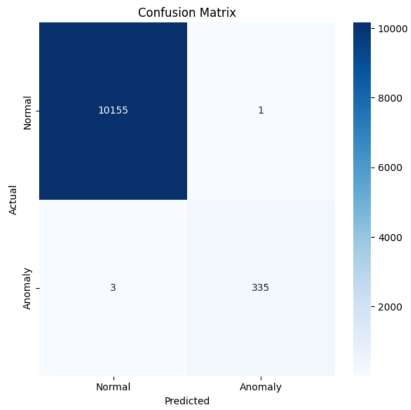
3. Traditional Machine Learning Approaches

This project focuses on evaluating various machine learning models for detecting Denial of Service (DoS) attacks. The implementation includes training and comparing four different models: Logistic Regression, Random Forest, XGBoost, and LightGBM.

3.1 Logistic Regression

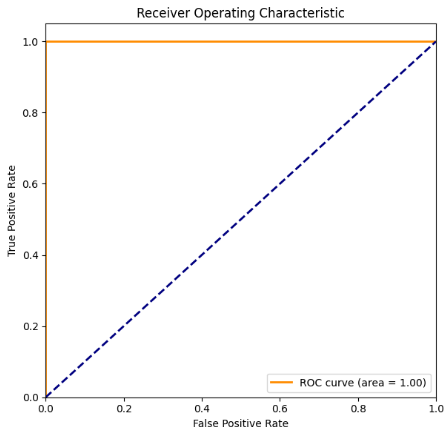
```
Logistic Regression:
Accuracy: 0.9996
Precision: 0.9970
Recall: 0.9911
F1 Score: 0.9941
AUC: 1.0000
```

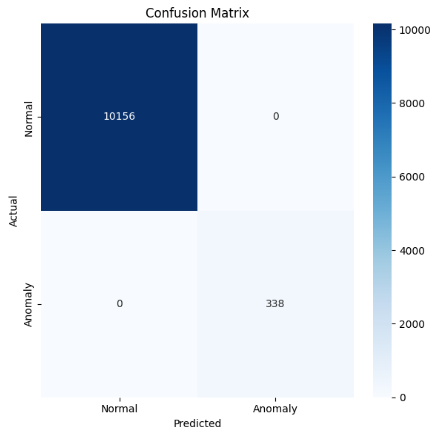




3.2 Random Forest

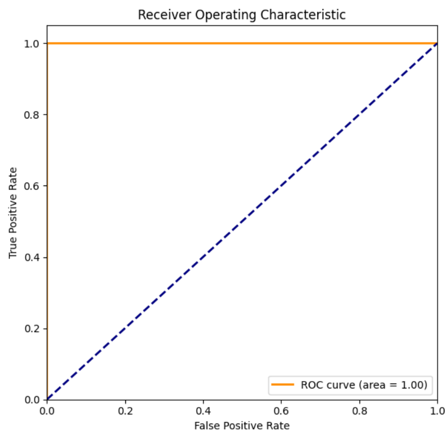
```
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1 Score: 1.0000
AUC: 1.0000
```

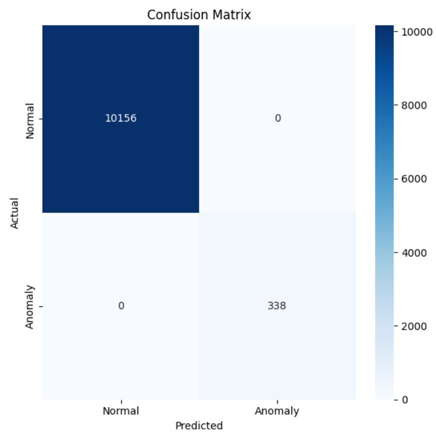




3.3 XGBoost

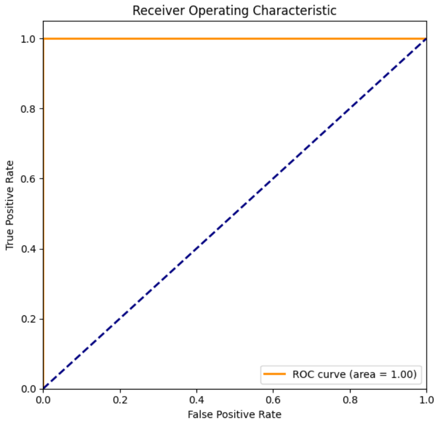
```
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1 Score: 1.0000
AUC: 1.0000
```

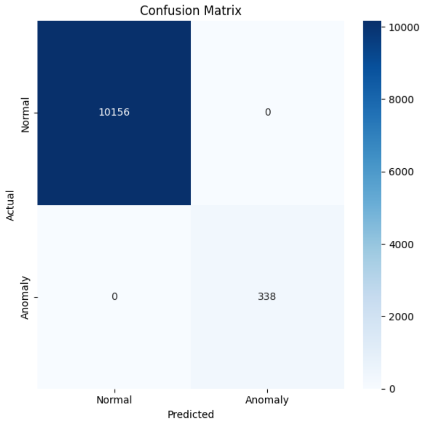




3.4 LightGBM

```
LightGBM:
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1 Score: 1.0000
AUC: 1.0000
```



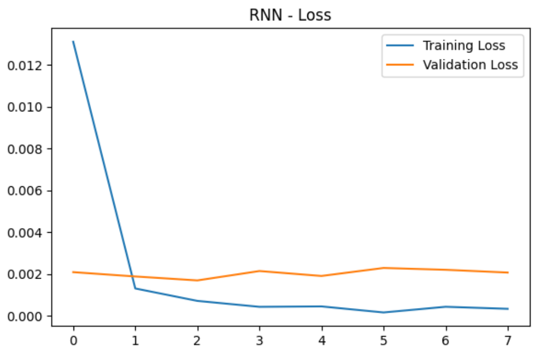


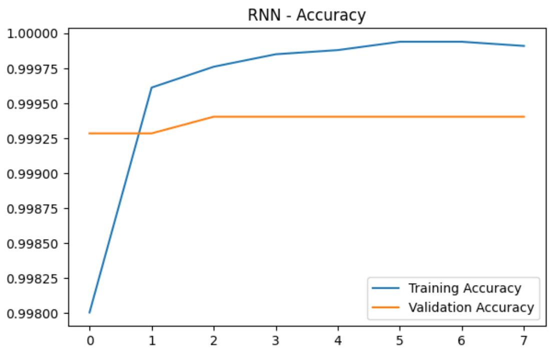
4. Deep Learning Approaches

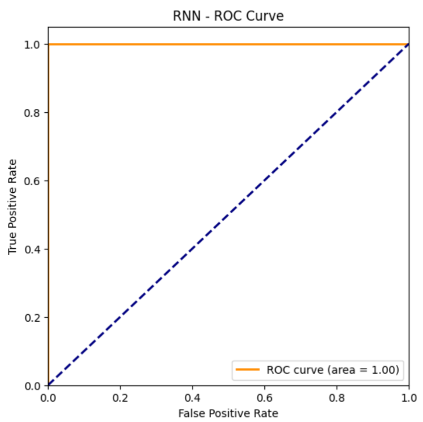
This project implements and compares three deep learning architectures (RNN, LSTM, and GRU) for detecting DDoS attacks using network traffic data.

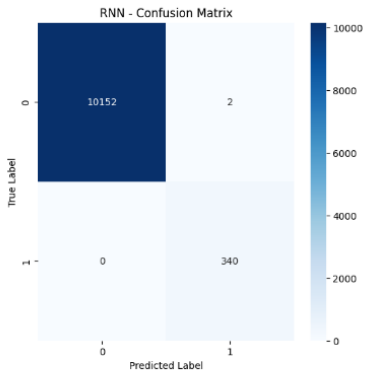
4.1 Recurrent Neural Networks (RNN)

```
RNN - Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     10154
           1       0.99      1.00      1.00       340

    accuracy                           1.00     10494
   macro avg       1.00      1.00      1.00     10494
weighted avg       1.00      1.00      1.00     10494
```



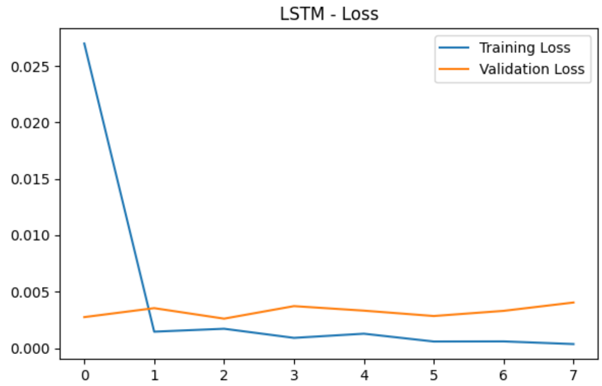


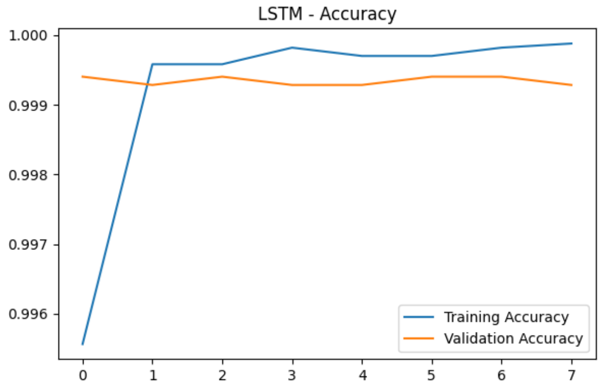


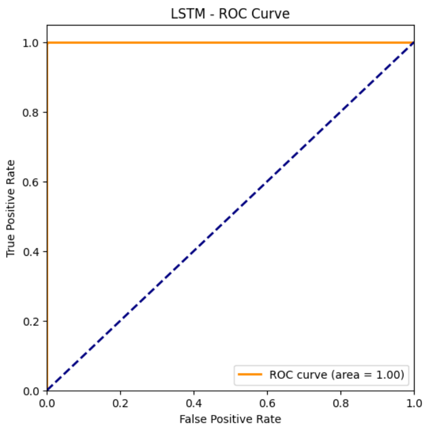


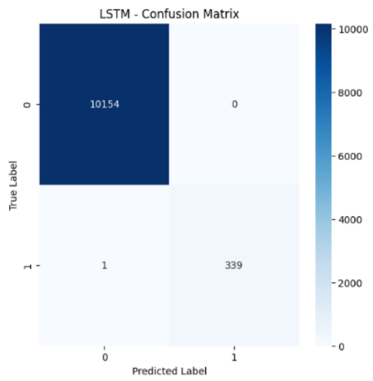
4.2 Long Short-Term Memory (LSTM)

```
LSTM - Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     10154
           1       1.00      1.00      1.00       340

    accuracy                           1.00     10494
   macro avg       1.00      1.00      1.00     10494
weighted avg       1.00      1.00      1.00     10494
```



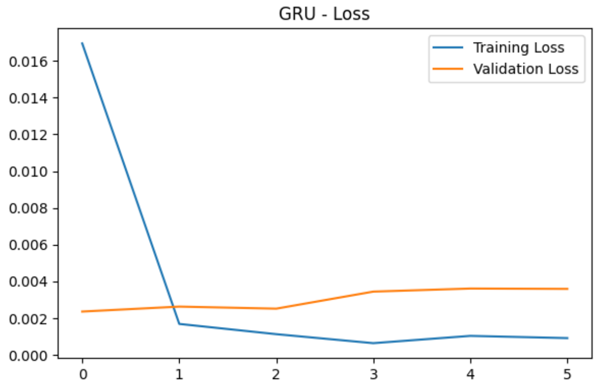


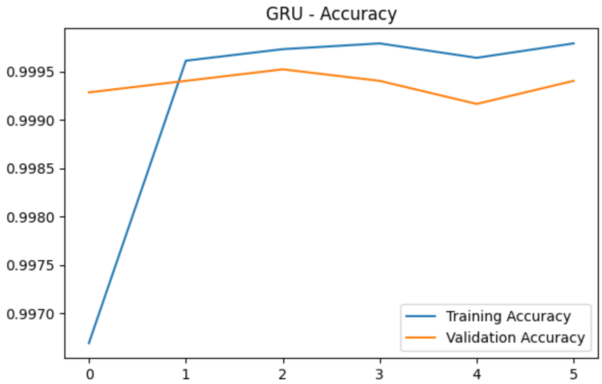


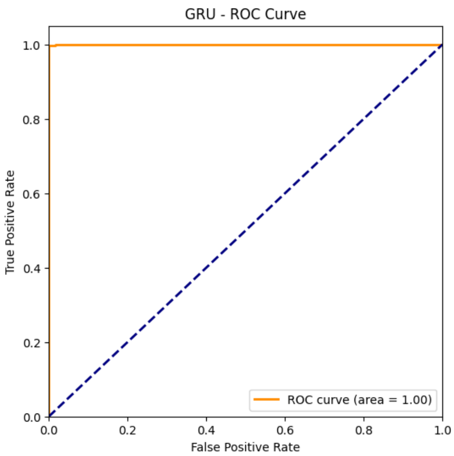


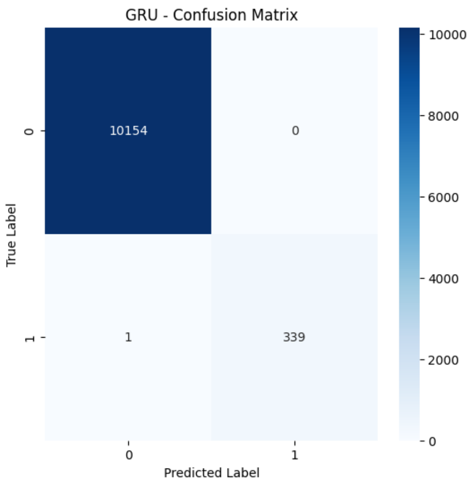
4.3 Gated Recurrent Unit (GRU)

```
GRU - Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     10154
           1       1.00      1.00      1.00       340

    accuracy                           1.00     10494
   macro avg       1.00      1.00      1.00     10494
weighted avg       1.00      1.00      1.00     10494
```





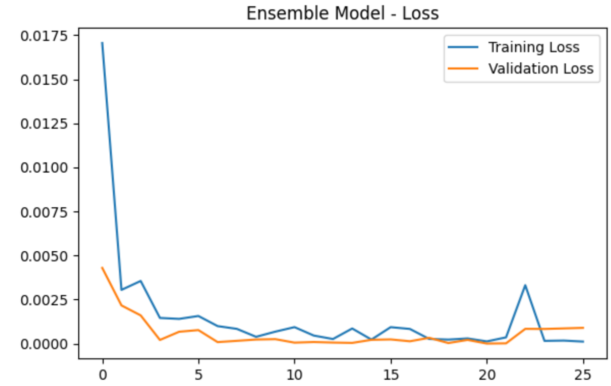


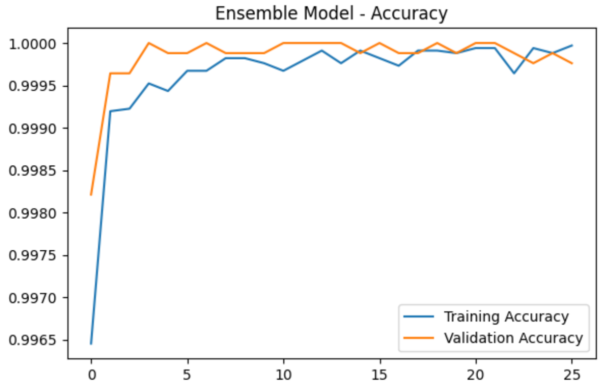


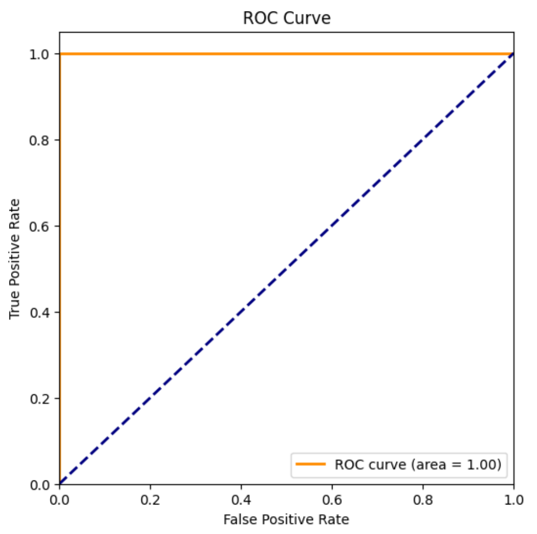
5. Complex Neural Network Architecture

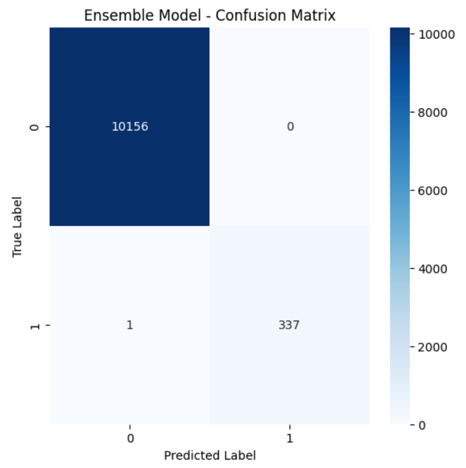
This project focuses on building and evaluating an ensemble deep learning model to detect anomalies in network traffic data. The ensemble model combines Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM), Bidirectional LSTM (BiLSTM), and Gated Recurrent Units (GRU) to leverage the strengths of each architecture.

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     10156
           1       1.00      1.00      1.00       338

    accuracy                           1.00     10494
   macro avg       1.00      1.00      1.00     10494
weighted avg       1.00      1.00      1.00     10494
```









6. Results and Comparisons

The results show excellent performance across all models in detecting HTTP-based DoS attacks. Traditional machine learning models (Random Forest, XGBoost, and LightGBM) achieved perfect scores (1.0000) in all metrics. Deep learning models (RNN, LSTM, GRU) and the ensemble model also performed remarkably well, with accuracy above 0.9998.

Even the basic Logistic Regression model showed strong performance with an accuracy of 0.9996. The perfect AUC scores (1.0000) across all models indicate excellent ability to distinguish between normal traffic and attacks.

For practical use, both traditional and deep learning approaches prove highly effective. Random Forest, XGBoost, or LightGBM might be preferred for their perfect performance and lower computational needs. However, deep learning models like LSTM or GRU would be equally good choices for real-time detection systems.

Overall, any of these models would be suitable for implementing a reliable DoS attack detection system.

Models	Accuracy	Precision	Recall	F1-Score	AUC
Logistic Regression	0.9996	0.9970	0.9911	0.9941	1.0000
Random Forest	1.0000	1.0000	1.0000	1.0000	1.0000
XGBoost	1.0000	1.0000	1.0000	1.0000	1.0000
LightGBM	1.0000	1.0000	1.0000	1.0000	1.0000
RNN	0.9998	0.9942	1.0000	0.9971	1.0000
LSTM	0.9999	1.0000	0.9971	0.9985	1.0000
GRU	0.9999	1.0000	0.9971	0.9985	1.0000
Ensemble	0.9999	1.0000	0.9970	0.9985	1.0000