CS 513
Knowledge Discovery and Data Mining

DDoS Attack Detection Classification

Group 16

Group Member

Nancy Gupta
20018167

Aana Khanduri
20021389

Tejas Vijay Adhav
20012193

Ritik Singh
20026035

Contents

Dataset Description

DDoS Attack Detection classification Dataset

Raw Data-

- Number of columns - 23
- Number of rows - 104345
- Numerical columns- 20
- Target Feature - Label (‘Benign or Malicious’)

Data after EDA -

- Number of columns - 23
- Number of rows - 104345
- Numerical columns - 20

Data distribution of test and training -

- 30% for test
- 70% for training

Numerical Columns

- dt
- switch
- pktcount
- bytecount
- dur
- dur_nsec
- tot_dur
- flows
- packetins
- pktperflow
- byteperflow
- pktrate
- Pairflow
- Port_no
- Tx_bytes
- Rx_bytes
- Tx_kbps
- Rx_kbps
- Tot_kbps
- label

Target Column -

- Label (‘Benign or Malicious’)

Problem Statement

The DDoS SDN dataset available on Kaggle contains network traffic flow information collected from a Software-Defined Network (SDN). This data includes features like packet count, byte count, flow duration, and source/destination IP addresses. The challenge lies in accurately classifying this traffic as either benign or malicious (DDoS attack) based on the provided features.

However, it's possible the dataset may suffer from class imbalance, where benign traffic might be the majority. Therefore, our goal is to create a classification model that can accurately detect DDoS attacks even in the presence of potential class imbalance. This will ensure the model prioritizes identifying malicious traffic effectively within the SDN environment.

Objective

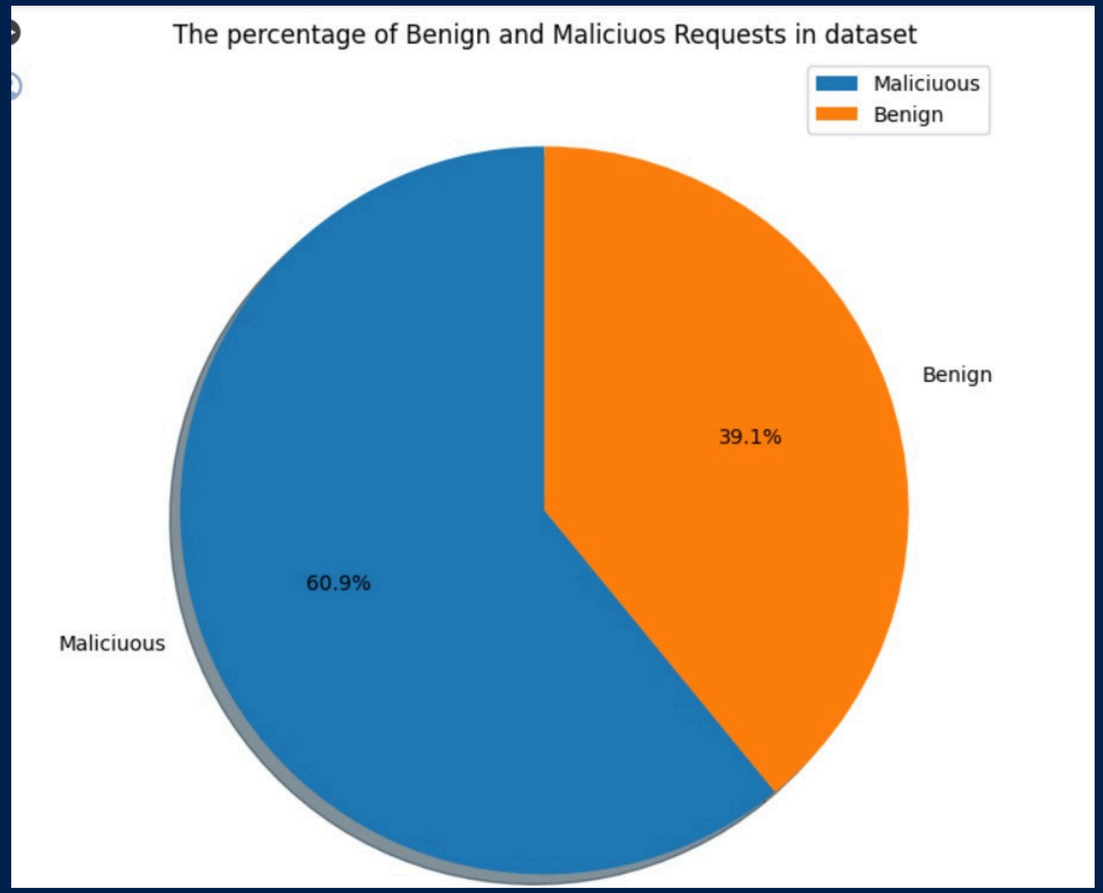
- Exploration and Data Prep
  - Clean and explore the dataset to understand its characteristics and identify anomalies
- Model Development:
  - Build classification models to detect DDoS attacks using various algorithms
- Model Comparison:
  - Compare model performance to choose the most effective one for DDoS detection
- Result Interpretation:
  - Interpret results to understand which features are crucial for detecting DDoS attacks

Exploratory Data Analysis

- The dataset contains 104345 rows and 23 columns
- The dataset contains two types of columns - numerical columns and objective columns.
- Target Variable: Label (‘Benign’ or ‘Malicious’)
- Total number of null values are 1012, which are equally distributed in 2 columns - tot_kbps and rx_kbps
- Numerical columns:
  - dt , switch, pktcount, bytecount, dur, dur_nsec, tot_dur, flows, packetins, pktperflow, byteperflow, pktrate, Pairflow, Port_no, Tx_bytes, Rx_bytes, Tx_kbps, Rx_kbps, Tot_kbps, label
- Objective Columns
  - src, dst and protocol

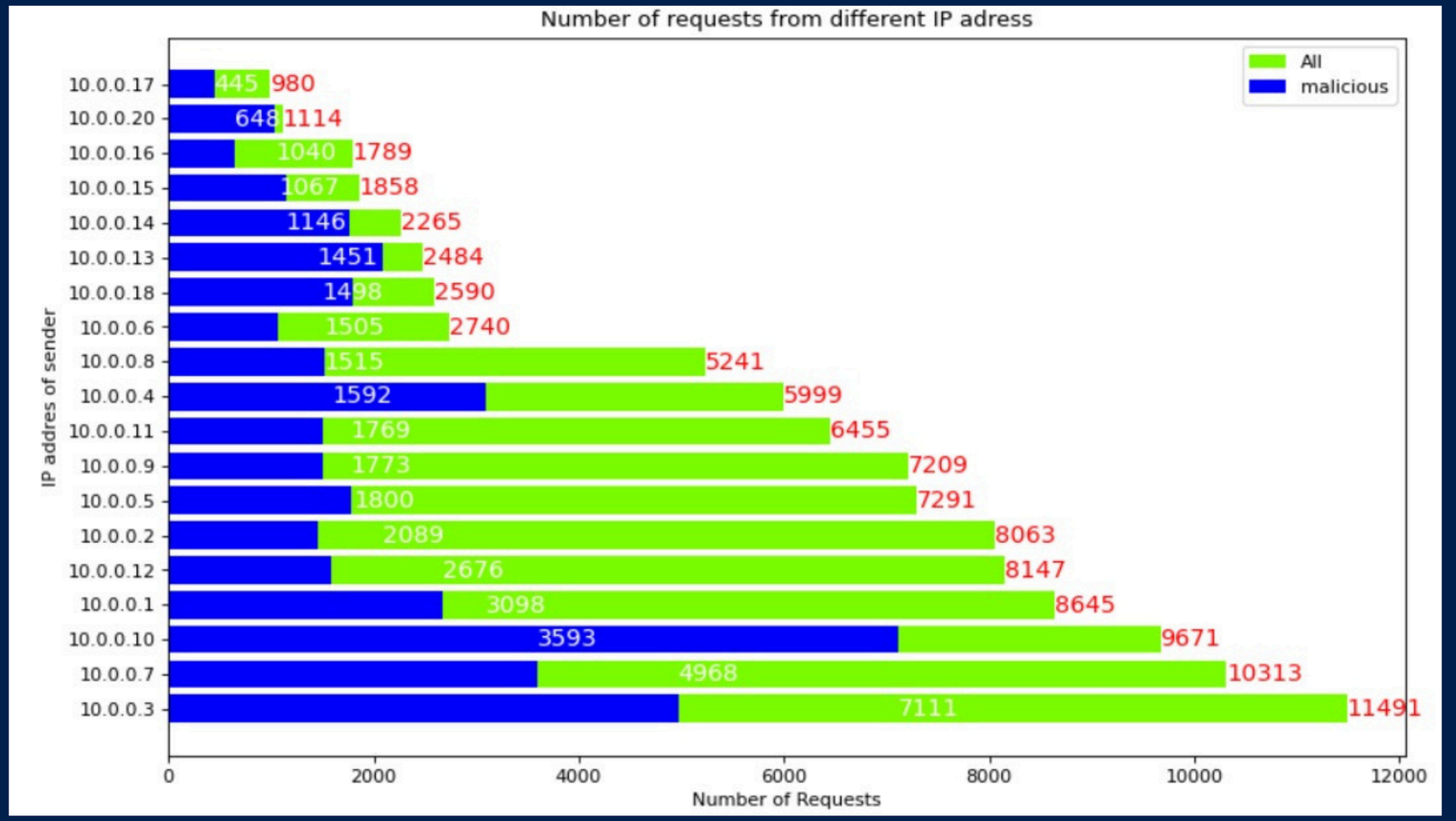
Exploratory Data Analysis

Percentage of Benign and Malicious Requests

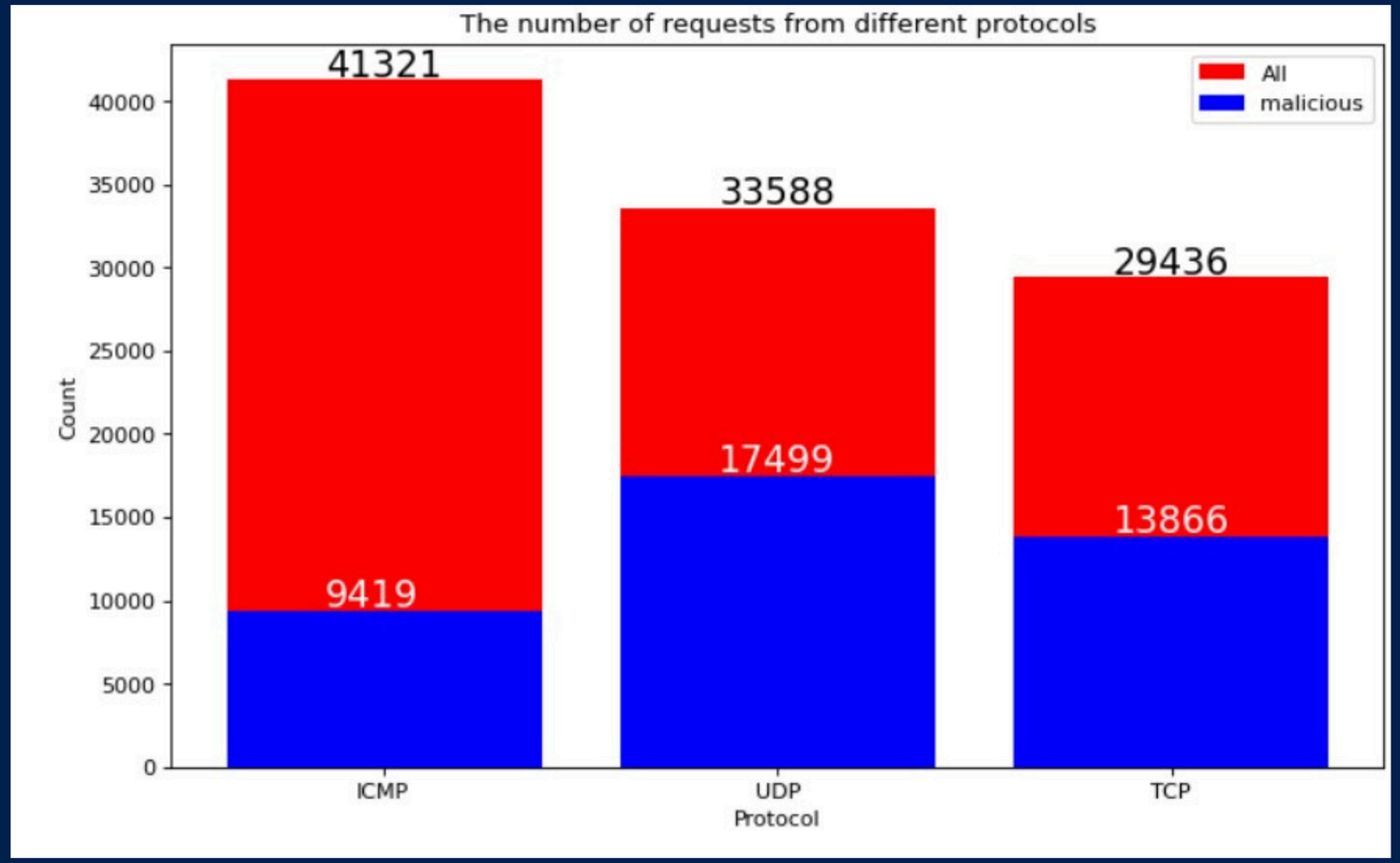


Exploratory Data Analysis

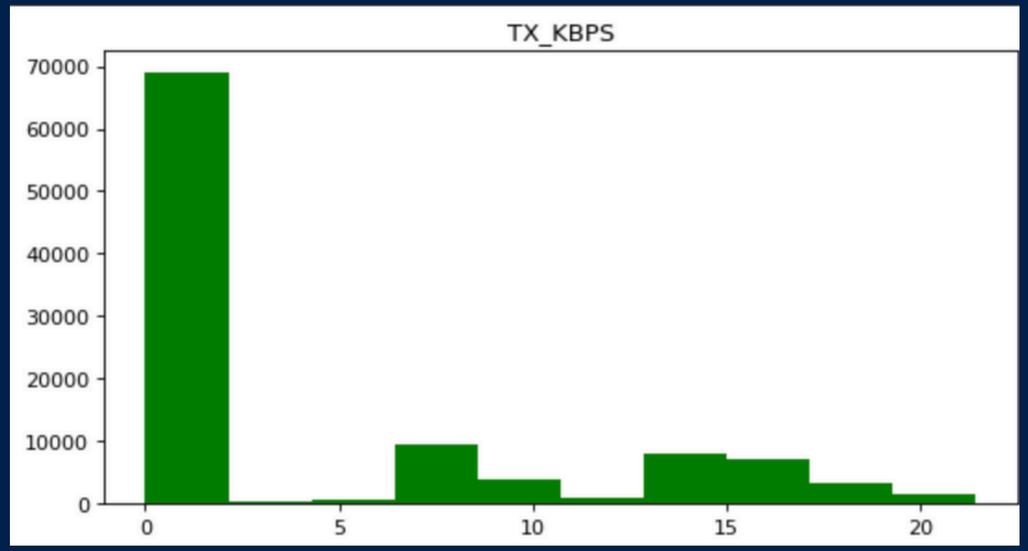
Data Visualization: Exploring the distribution of important numerical features in the DDoS Attack

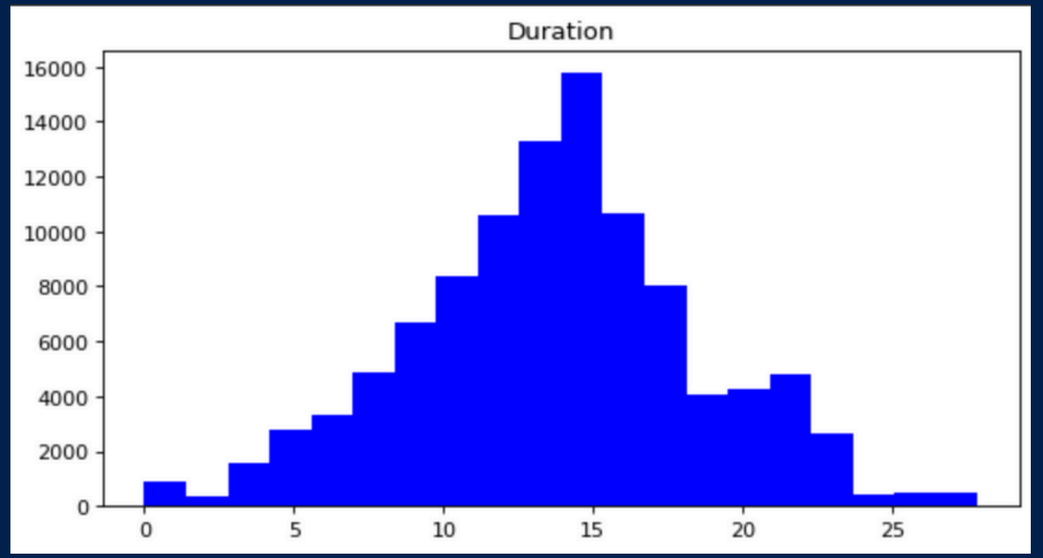


Exploratory Data Analysis

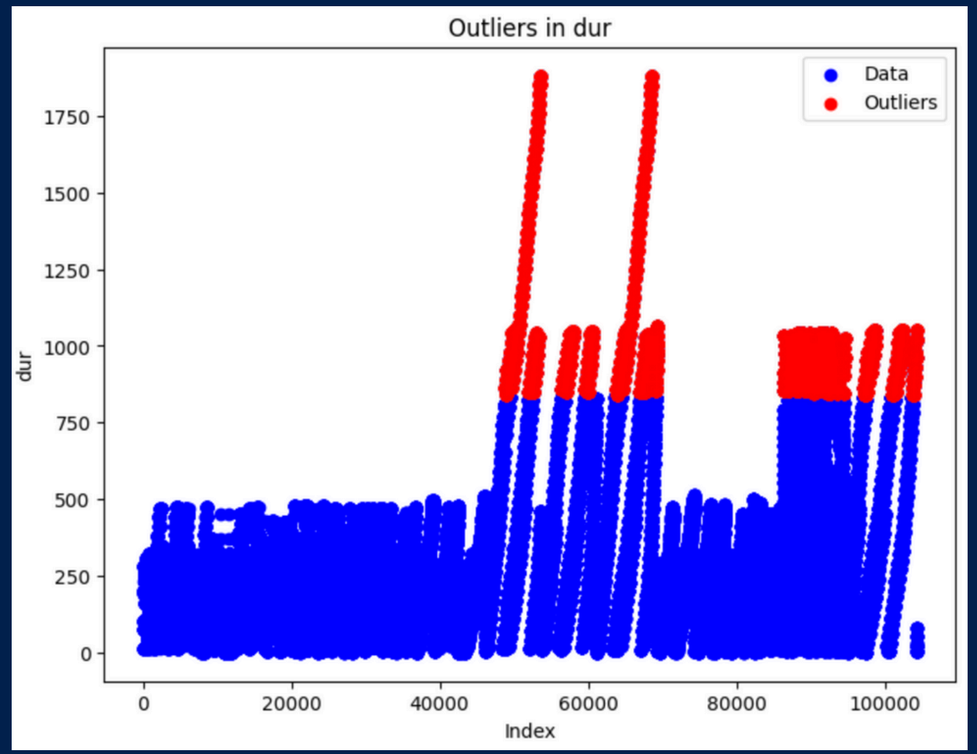


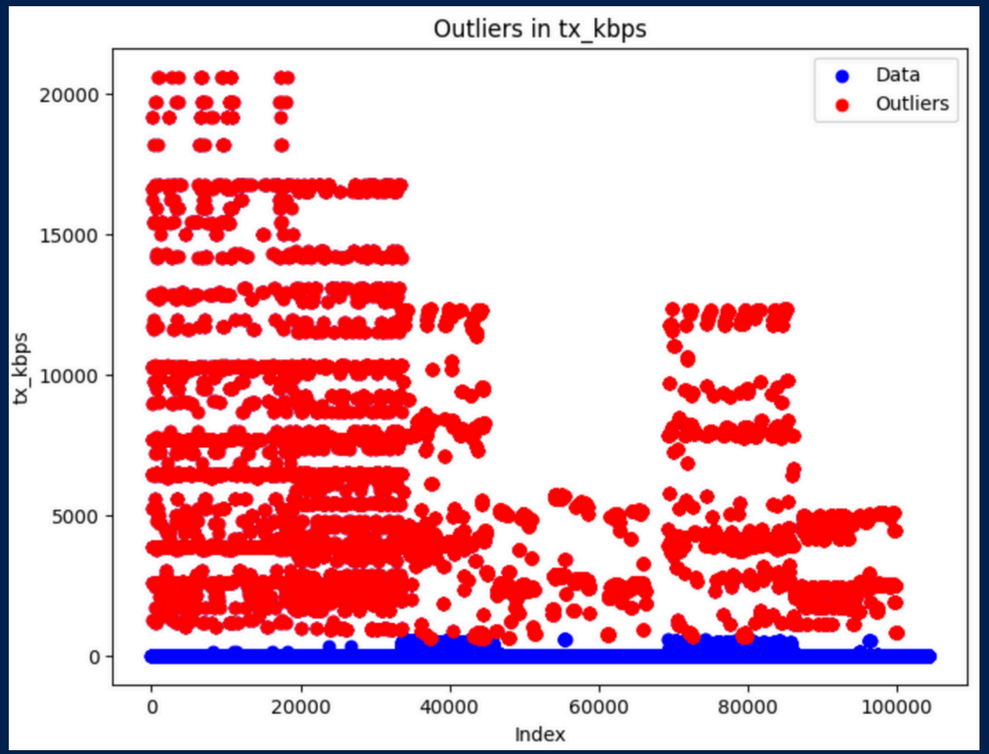
Exploratory Data Analysis



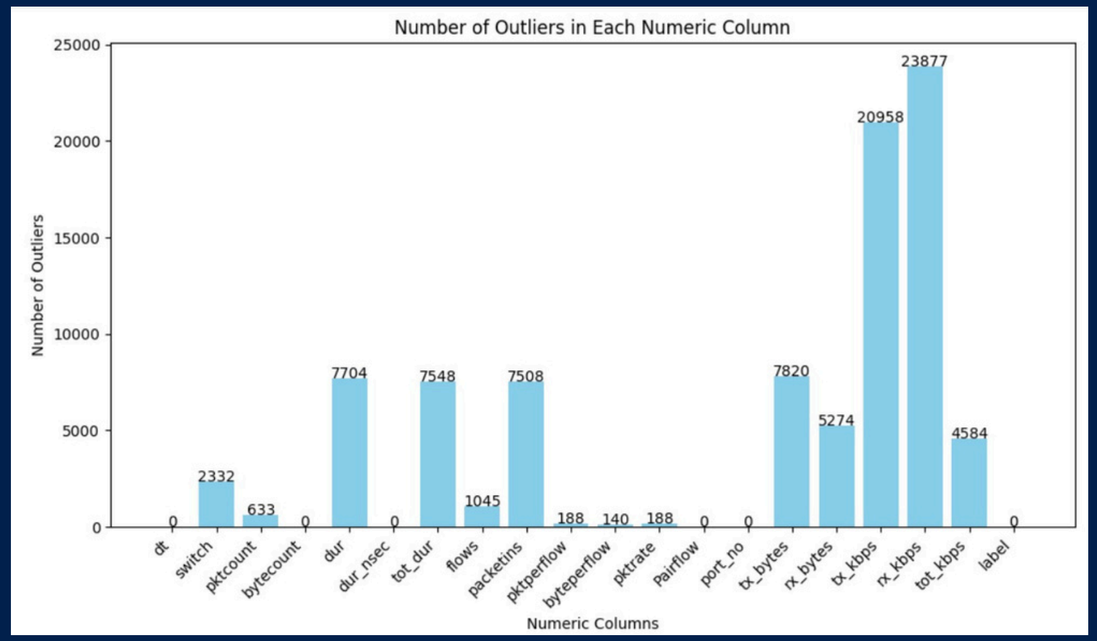


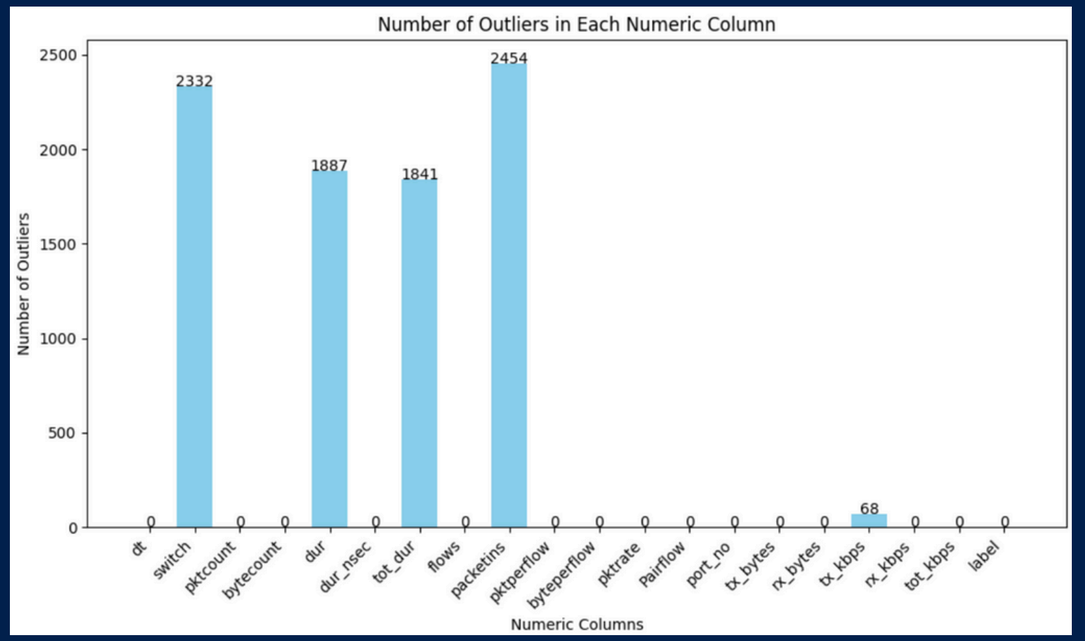
Outliers in the Dataset





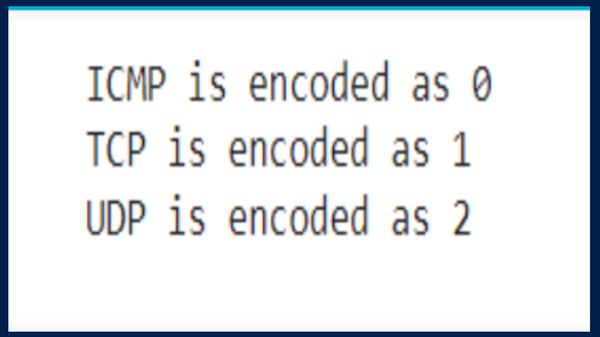
Comparison of Outliers before and after Preprocessing





Preprocessing

- Utilizing Label Encoder for categorical feature encoding in DDoS attack detection.
- Label Encoder transforms categorical features into integer data type, facilitating model training.
- Each unique category is assigned a unique integer, preserving data integrity.
- Advantages of Label Encoder include maintaining dataset dimensionality and simplicity.
- However, Label Encoder may introduce ranking issues, potentially affecting model training if not handled carefully.
- Additionally, handling missing values using Simple Imputer ensures robustness in model training by replacing null values with the mean of respective features.



Change in Null Values

BEFORE UNDERSAMPLING

```
data.isnull().sum()
```

```
dt                0
switch            0
src               0
dst               0
pktcount          0
bytecount         0
dur               0
dur_nsec          0
tot_dur           0
flows             0
packetins         0
pktperflow        0
byteperflow       0
pktrate           0
Pairflow          0
Protocol          0
port_no           0
tx_bytes          0
rx_bytes          0
tx_kbps           0
rx_kbps         506
tot_kbps        506
label             0
dtype: int64
```

AFTER UNDERSAMPLING

```
undersampled_df.isnull().sum()
```

```
dt                0
switch            0
src               0
dst               0
pktcount          0
bytecount         0
dur               0
dur_nsec          0
tot_dur           0
flows             0
packetins         0
pktperflow      171
byteperflow     171
pktrate         171
Pairflow          0
Protocol          0
port_no           0
tx_bytes          0
rx_bytes          0
tx_kbps           0
rx_kbps         426
tot_kbps        426
label             0
dtype: int64
```

Handling Imbalanced Data

From EDA, we saw that there is a great imbalance in data between Benign and Malicious labels. This imbalance in data creates a bias towards a specific output which leads to inappropriate training of the machine learning model.

We can solve this problem of imbalance using different sampling techniques.

```
label
0    63561
1    40784
Name: count, dtype: int64
```

Handling Imbalanced Data

Under Sampling Method

It is a technique employed to address class imbalance in a dataset, where one class is significantly underrepresented compared to the other class. It involves randomly removing instances from the overrepresented class to create a more balanced distribution, preventing the model from being biased toward predicting the majority class.

```
label
1    40784
0    40784
Name: count, dtype: int64
```

What is the result of under-sampling?

Before Under Sampling: Benign: 63561 | Malicious: 40784
After Under Sampling: Benign: 40784 | Malicious: 40784

Principal Component Analysis

- PCA was utilized to reduce the dimensionality of the dataset from 23 input variables to 10 principal components.
- This reduction simplifies analysis and modeling tasks by condensing information while retaining the most important features.
- PCA helps eliminate redundant variables, leading to more efficient computation and visualization.
- It filters out noise and less significant variations in the data, improving signal-to-noise ratio and model robustness.
- The reduced set of principal components maintains meaningful interpretations, enhancing data understanding and actionable insights.

```
Principal Component 1:
pktperflow: 0.3835
byteperflow: 0.3809
pktrate: 0.3766
Protocol: 0.3621
bytecount: 0.3619
pktcount: 0.3390
flows: -0.2213
Pairflow: -0.2172
tx_bytes: -0.1558
tot_dur: -0.1504
dur: -0.1481
rx_bytes: -0.1156
packetins: 0.0535
dur_nsec: 0.0533
tot_kbps: 0.0517
rx_kbps: 0.0321
switch: -0.0320
port_no: -0.0160
tx_kbps: 0.0060
```

```
Principal Component 10:
flows: -0.6995
Pairflow: 0.4431
packetins: -0.3393
rx_bytes: 0.2235
dur_nsec: 0.2132
Protocol: -0.1681
tx_bytes: 0.1655
port_no: -0.1381
pktcount: 0.0929
dur: -0.0721
tot_dur: -0.0702
tot_kbps: -0.0617
switch: 0.0462
pktperflow: 0.0430
byteperflow: -0.0380
pktrate: 0.0299
rx_kbps: 0.0120
bytecount: -0.0090
tx_kbps: 0.0077
```

Predictive Models and Results

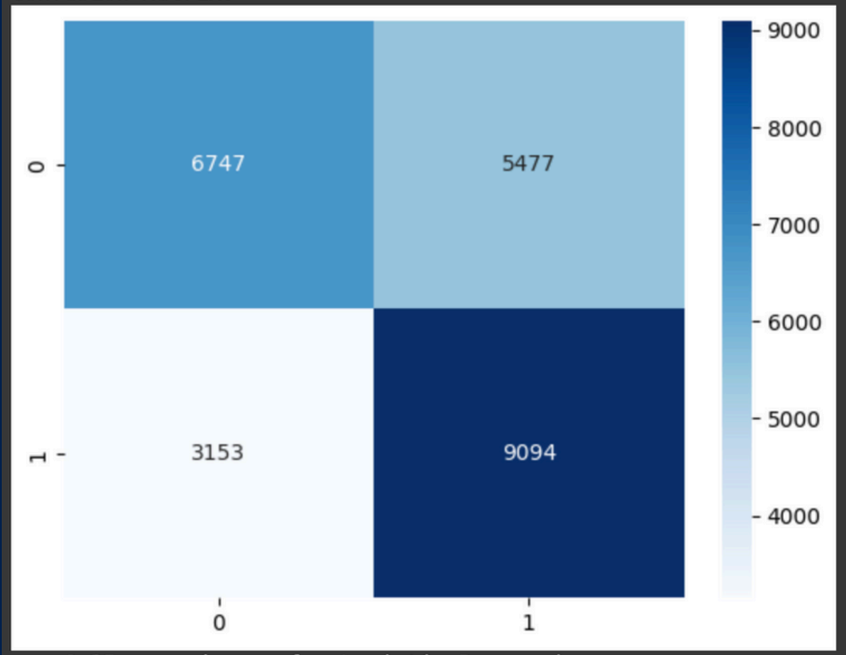
Logistic Regression It predicts the probability of a specific event occurring, making it suitable for classifying DDoS attacks as benign or malicious traffic.	**Random Forest** Builds an ensemble of decision trees, improving accuracy and robustness in DDoS attack detection compared to a single tree.	**Naive Bayes** Classifies traffic based on the assumption of feature independence, offering a simple and efficient approach for DDoS attack detection.	**SVM** SVM finds an optimal hyperplane to separate benign and DDoS traffic data in high-dimensional space, offering robustness to complex patterns.
AdaBoost (Adaptive Boosting) Similar to Gradient Boosting, but assigns weights to training instances, focusing the learning process on more challenging examples for improved DDoS attack detection.	**Decision Tree** Learns a tree-like structure with branching rules based on network traffic features to classify traffic as normal or a DDoS attack.	**KNN (K-Nearest Neighbors)** Classifies traffic based on the majority vote of its k nearest neighbors in the feature space, potentially effective for DDoS attack detection.	**Gradient Boosting** Sequentially builds decision trees, with each tree focusing on correcting the errors of the previous one, potentially leading to highly accurate DDoS attack detection models.

Logistic Regression

Logistic Regression is used for classification tasks, not regression. It models the probability of an instance belonging to a particular class by applying the logistic function to a linear combination of input features.

Accuracy: 64.73%
F1-Score: 0.61

```
Logistic Regression Accuracy: 64.73%
              precision    recall  f1-score   support

           0       0.55      0.68      0.61      9900
           1       0.74      0.62      0.68     14571

    accuracy                           0.65     24471
   macro avg       0.65      0.65      0.64     24471
weighted avg       0.67      0.65      0.65     24471

Confusion Matrix:
```




```
--- 0.35 seconds --- for Logistic Regression
```

Naïve Bayes Classifier

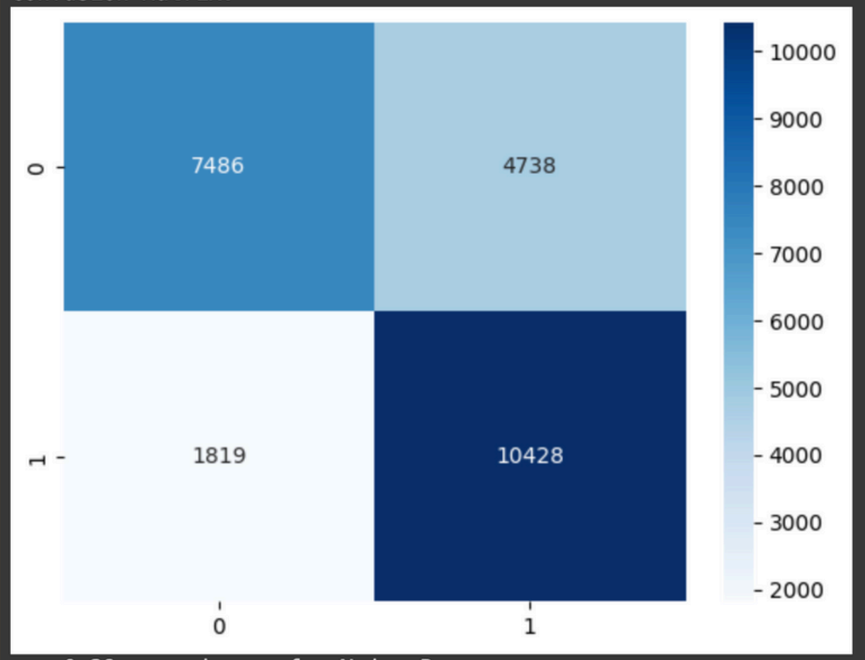
Naïve Bayes assumes independence between features, which is a native assumption, hence the name. It calculates the probability of a given instance belonging to a particular class by considering the joint probability of its features.

Accuracy: 73.21%
F1-Score: 0.70

Naive Bayes Accuracy: 73.21%

	precision	recall	f1-score	support
0	0.61	0.80	0.70	9305
1	0.85	0.69	0.76	15166
accuracy			0.73	24471
macro avg	0.73	0.75	0.73	24471
weighted avg	0.76	0.73	0.74	24471

Confusion Matrix:



--- 0.29 seconds --- for Naive Bayes

K-nearest Neighbors

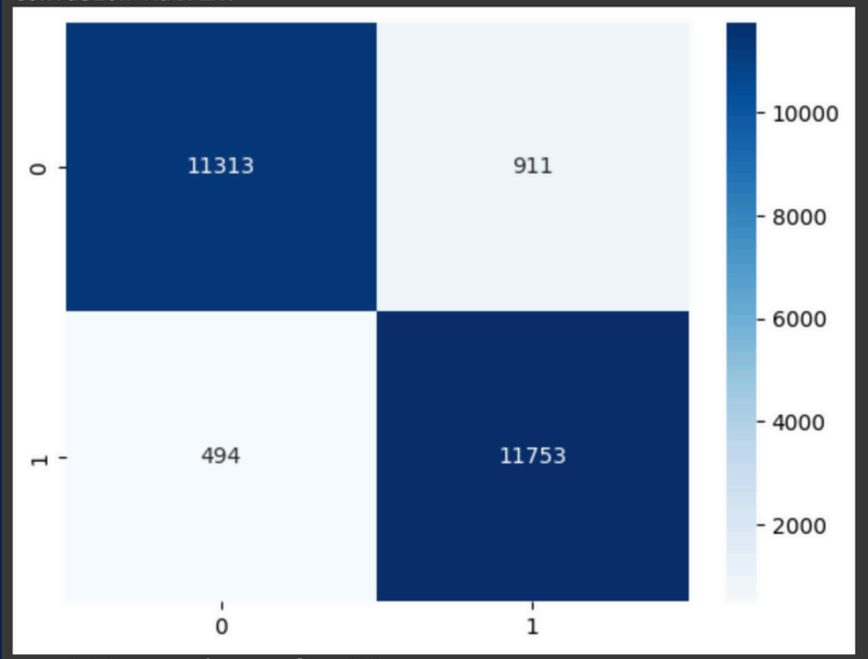
Classifies traffic based on the majority vote of its k-nearest neighbors in the feature space, potentially effective for DDoS attack detection.

Accuracy: 94.26%
F1-Score: 0.94

KNN Accuracy: 94.26%

	precision	recall	f1-score	support
0	0.93	0.96	0.94	11807
1	0.96	0.93	0.94	12664
accuracy			0.94	24471
macro avg	0.94	0.94	0.94	24471
weighted avg	0.94	0.94	0.94	24471

Confusion Matrix:



--- 4.39 seconds --- for KNN

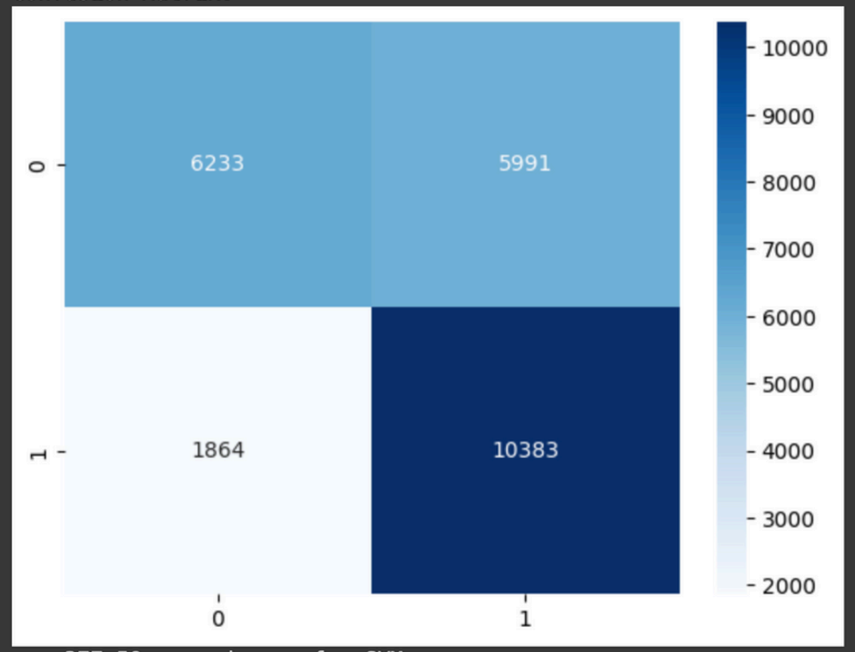
SVM (Linear kernel)

This approach works well if the data is naturally separable in a high-dimensional space using a straight line (hyperplane). It's computationally efficient but might not be suitable for complex, non-linear relationships between features in network traffic data.

Accuracy: 67.9%

F1-Score: 0.61

```
SVM Accuracy with linear kernel: 67.9%
              precision    recall  f1-score   support

           0       0.51      0.77      0.61      8097
           1       0.85      0.63      0.73     16374

    accuracy                           0.68     24471
   macro avg       0.68      0.70      0.67     24471
weighted avg       0.74      0.68      0.69     24471

Confusion Matrix:
```




```
--- 377.59 seconds --- for SVM
```

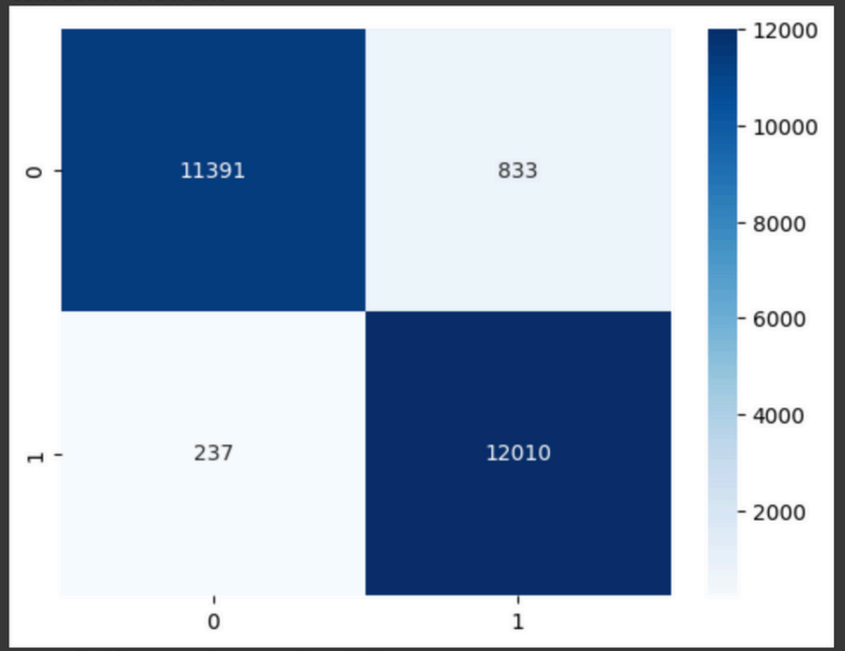
SVM (RBF kernel)

This approach works well if the data exhibits complex, non-linear relationships between features in a high-dimensional space. The Radial Basis Function (RBF) kernel allows the SVM to capture intricate patterns that cannot be separated by a linear hyperplane. While computationally efficient, it may not perform optimally if the data is inherently linear or separable by a straight line.

Accuracy: 95.63%

F1-Score: 0.96

```
SVM Accuracy with rbf kernel: 95.63%
              precision    recall  f1-score   support

           0       0.93      0.98      0.96     11628
           1       0.98      0.94      0.96     12843

    accuracy                           0.96     24471
   macro avg       0.96      0.96      0.96     24471
weighted avg       0.96      0.96      0.96     24471

Confusion Matrix:
```



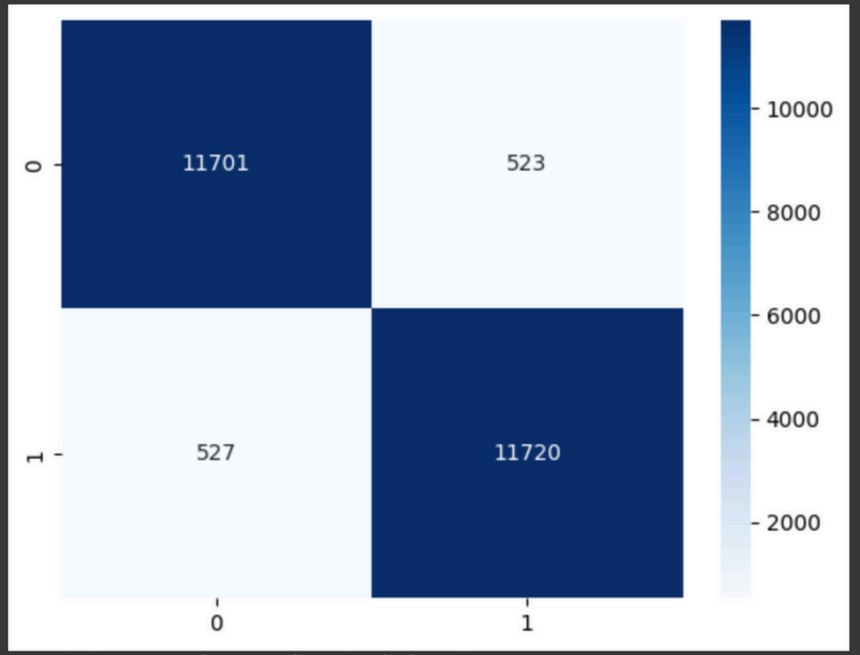

```
--- 42.53 seconds --- for SVM
```

Decision Tree Classifier

A Decision Tree Classifier recursively splits the dataset into subsets based on the most significant features, creating a tree structure to make binary decisions and classify instances.

Accuracy: 95.71%
F1-Score: 0.96

```
Decision Tree Accuracy: 95.71%
              precision    recall  f1-score   support

           0       0.96      0.96      0.96     12228
           1       0.96      0.96      0.96     12243

    accuracy                           0.96     24471
   macro avg       0.96      0.96      0.96     24471
weighted avg       0.96      0.96      0.96     24471

Confusion Matrix:
```




```
--- 1.30 seconds --- for Decision Tree
```

Random Forest Classifier

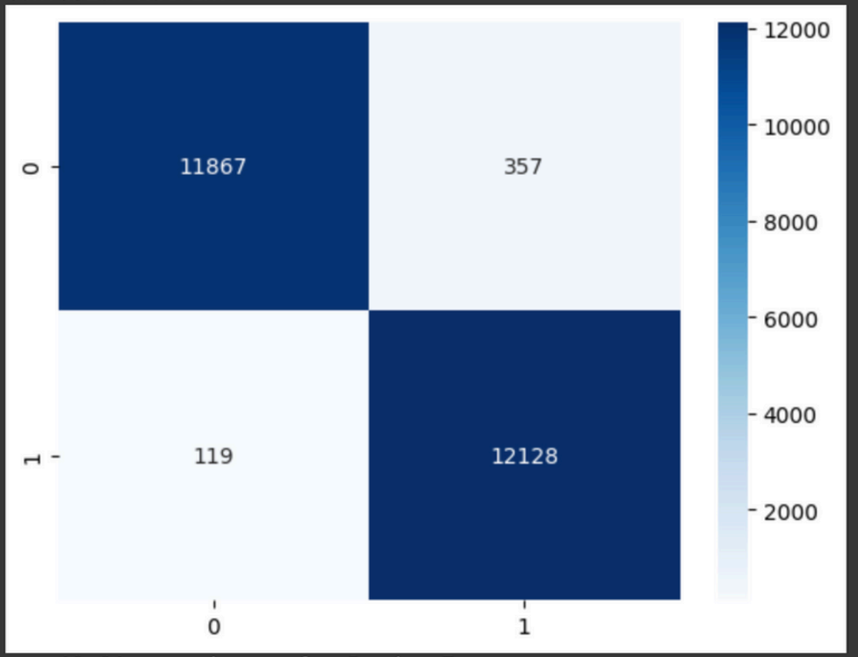
A Random Forest Classifier constructs a multitude of decision trees during training and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

Accuracy: 98.05%
F1- Score: 0.98

Random Forest Accuracy: 98.05%

	precision	recall	f1-score	support
0	0.97	0.99	0.98	11986
1	0.99	0.97	0.98	12485
accuracy			0.98	24471
macro avg	0.98	0.98	0.98	24471
weighted avg	0.98	0.98	0.98	24471

Confusion Matrix:



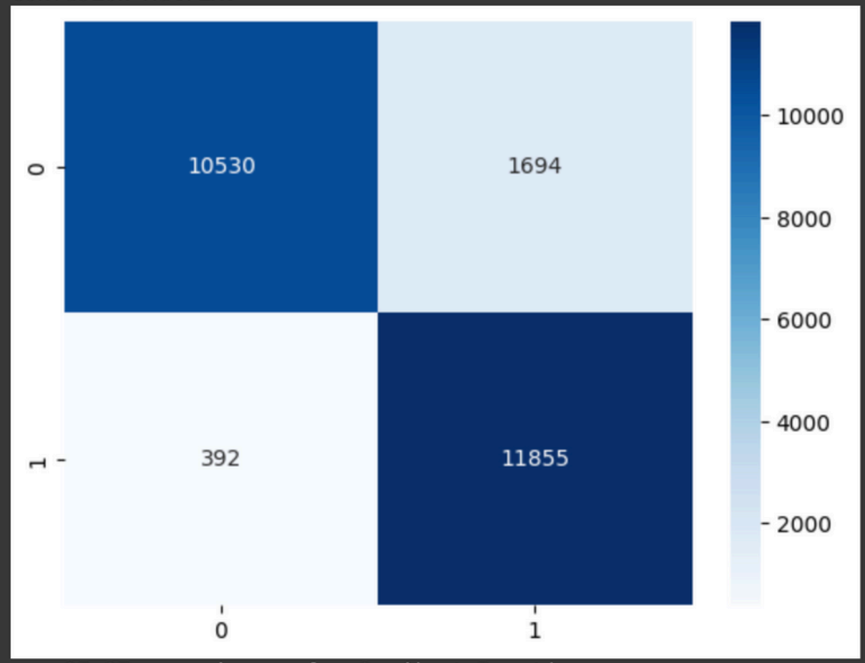
--- 19.64 seconds --- for Random Forest

Gradient Boosting Classifier

Gradient boosting sequentially builds decision trees to progressively improve predictions for tasks like classification and regression.

Accuracy: 91.48%
F1-Score: 0.91

```
Gradient Boosting Accuracy: 91.48%
              precision    recall  f1-score   support

           0       0.86      0.96      0.91     10922
           1       0.97      0.87      0.92     13549

    accuracy                           0.91     24471
   macro avg       0.91      0.92      0.91     24471
weighted avg       0.92      0.91      0.92     24471

Confusion Matrix:
```



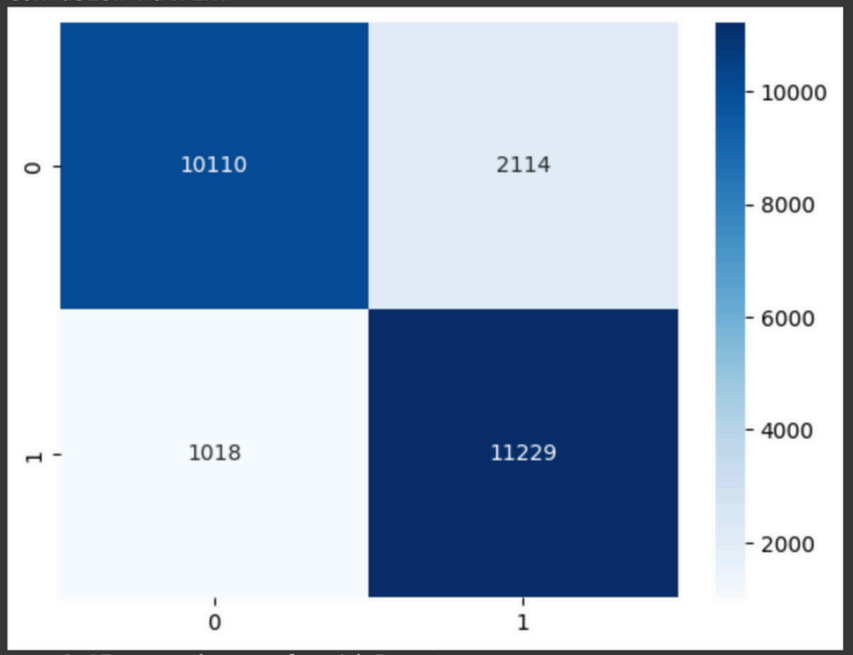

```
--- 25.27 seconds --- for Gradient Boosting
```

AdaBoost Classifier

Focuses on improving classification accuracy by adaptively adjusting the weights of training instances. It emphasizes instances that were misclassified by previous models in the ensemble.

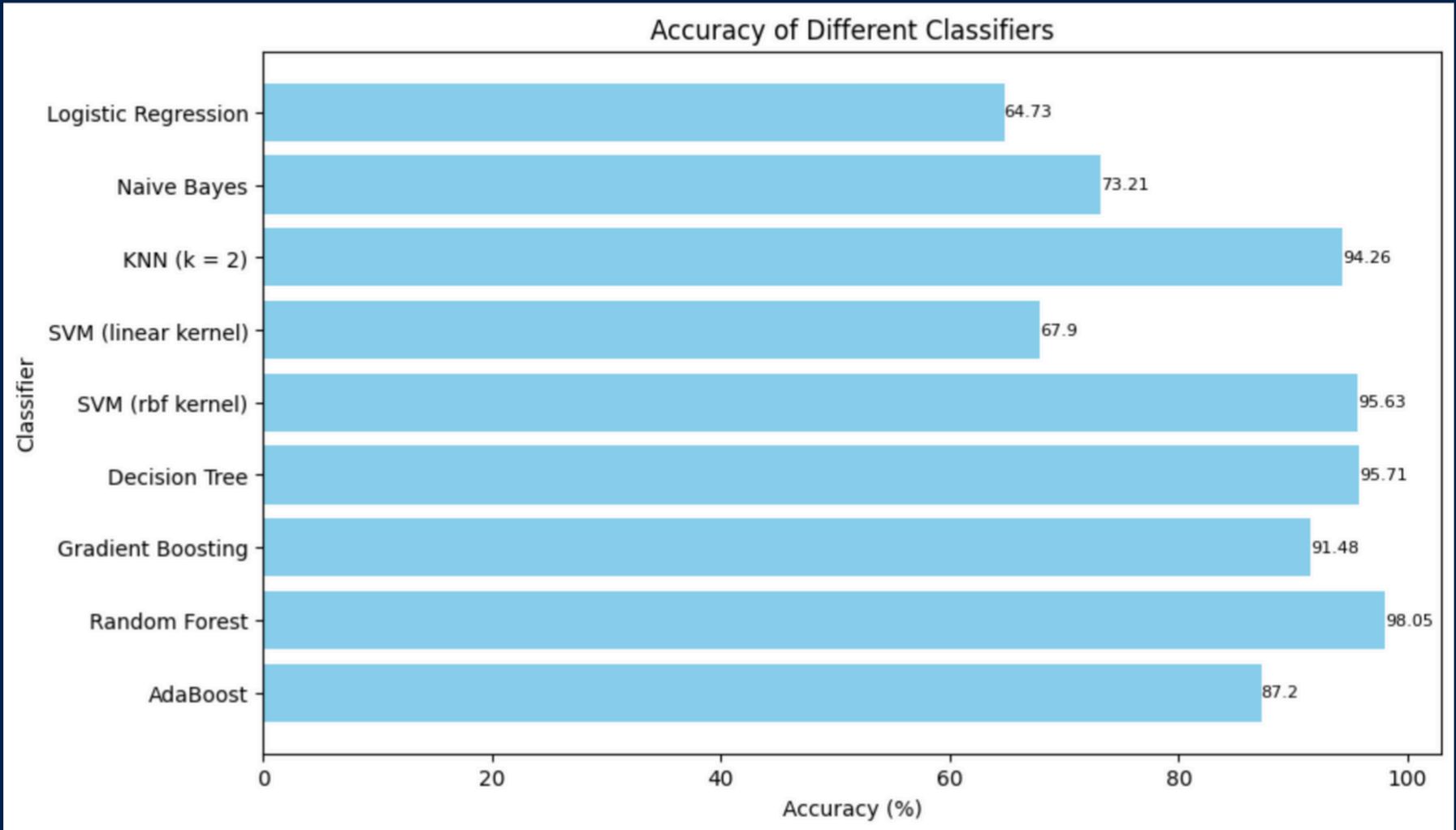
Accuracy: 87.2%
F1-Score: 0.87

```
AdaBoost Accuracy: 87.2%
              precision    recall  f1-score   support

           0       0.83      0.91      0.87     11128
           1       0.92      0.84      0.88     13343

    accuracy                           0.87     24471
   macro avg       0.87      0.88      0.87     24471
weighted avg       0.88      0.87      0.87     24471

Confusion Matrix:
```

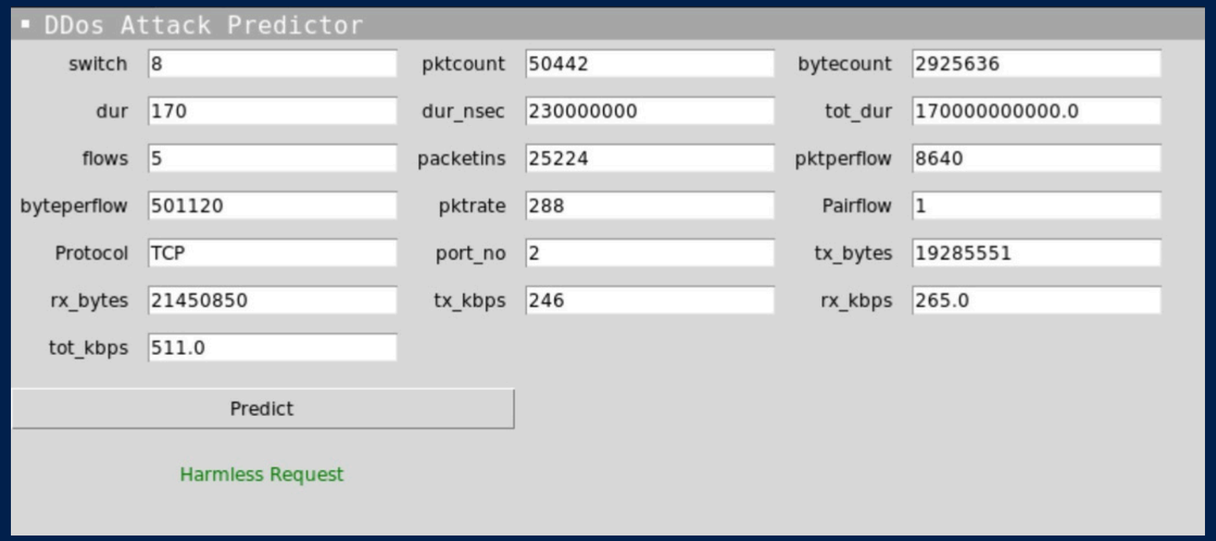


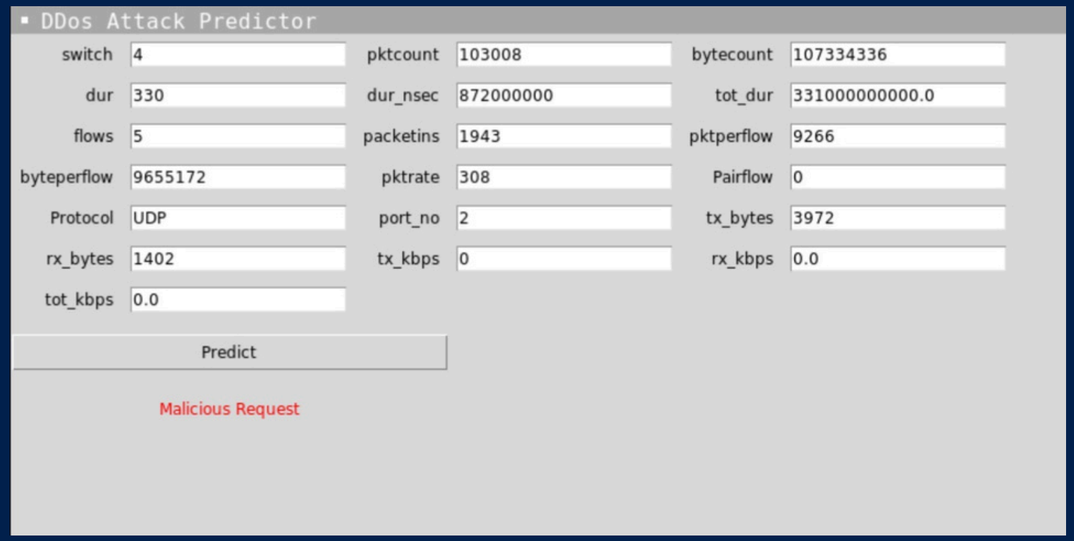

```
--- 6.17 seconds --- for AdaBoost
```

Predictive Models and Results



Visualization of DDoS Attack Detection Results Using Tkinter GUI





Results

Findings from Model Evaluation

- Conducted training and testing on eight distinct machine learning models for classification.
- Evaluated model performance using key metrics such as Accuracy and Confusion Matrix.
- Identified Random Forest as the standout performer among all algorithms.
- Random Forest exhibited the highest Accuracy and showcasing its ability to make correct predictions across the dataset.
- The Second highest was exhibited by Decision Tree and SVM was very close to it.
- The model which performed the worst was Logistic Regression.

Thank You