

Identifying Malicious Network Traffic Using Machine Learning

Presented by: Amjad Alqahtani

Problem Statement

In the modern digital landscape, network traffic is constantly subjected to potential threats from malicious actors. Detecting such traffic is essential for safeguarding sensitive information and protecting the infrastructure.

My project focuses on using machine learning techniques to detect anomalies in network traffic that may indicate malicious activity, such as cyber attacks like flood attacks.

Importance of Solving the Problem

Cyberattacks can cause significant damage, including financial losses and compromised infrastructure, affecting both organizations and individuals.

Manually checking traffic is not practical given the vast amount of data on the web.

Automated detection systems powered by machine learning are crucial for identifying and responding to cyber threats.

Benefits of Solving the Problem

Organizations with high-security needs, such as financial institutions, hospitals, and government agencies.

Cybersecurity teams responsible for monitoring and defending against network intrusions, as well as individuals using personal devices who are vulnerable to network-based threats.

Literature Review:

Title: Applying Machine Learning Techniques for Detection of Malicious Code in Network Traffic

eDare system operates on ISP infrastructure like switches and routers, to intercept and analyze network traffic for potential malicious content. Malicious code may either be known, based on previous detection history, or unknown. For known threats, eDare detects and blocks them immediately. For unknown threats, it uses machine learning algorithms.

The machine learning algorithms in this module include methods like Decision Trees, Bayesian Networks, and Artificial Neural Networks.

Decision Tree had the highest accuracy at 95.5% and the lowest false positive rate at 3.5%. The overall system's performance was enhanced by using a Risk-Weighting Voting scheme. This resulted in the highest accuracy of 95.8% and the lowest false positive rate of 3.2%.

Literature Review:

Title: A Machine Learning Approach to Classify Network Traffic

Encrypted networks like Tor and VPN, has made it challenging to monitor and classify traffic, especially for detecting malicious activity. Traditional methods, such as protocol analyzers and deep packet inspection, have proven insufficient. This paper leverages machine learning to address these challenges, utilizing the CIC-Darknet 2020 dataset to classify benign and darknet traffic.

Key techniques like SMOTE for balancing and PCA for dimensionality reduction are applied, followed by a comparison of ensemble methods(e.g., AdaBoost, Random Forest, Bagging, Extra Trees, and Gradient Boosting), logistic regression, tree-based classifiers, and Naive Bayes. The models are evaluated using metrics such as Accuracy, Precision, Recall, F1-Score, and MCC.

The ensemble methods, particularly Extra Trees and Decision Trees, delivered the best results, with both train and test accuracies reaching 99% to 100%. These top-performing models also achieved an F1-Score and Matthews Correlation Coefficient (MCC) close to 0.99, demonstrating excellent classification performance. In contrast, models like Logistic Regression and Naive Bayes performed significantly worse, with accuracy and MCC values ranging from 0.46 to 0.54 for Logistic Regression and Gaussian Naive Bayes.

Literature Review:

Title: Detection of Encrypted Malicious Network Traffic using Machine Learning

This paper addresses the challenge of detecting malicious activity within encrypted network traffic, particularly traffic encrypted using Transport Layer Security (TLS).

The paper classifies malicious and benign traffic without needing to decrypt it using machine learning models called Support Vector Machine (SVM) with a Radial Basis Function (RBF) kernel and a Convolutional Neural Network (CNN). Instead of relying on content, these models use TLS record sizes and direction as key features.

Architecture

Two models are proposed: SVM (Support Vector Machine): A non-linear SVM using a Radial Basis Function (RBF) kernel. CNN (Convolutional Neural Network): A one-dimensional CNN that treats the sequence of TLS record sizes and directions as a time series. The model uses several convolutional and pooling layers followed by dense layers, aiming to detect patterns in the traffic data that indicate malicious behavior.

SVM Model:

The SVM model uses the frequency of unique TLS record sizes as input features. These features are stored in a hash vector. The SVM is trained using a sparse hash vector, with only non-zero values representing the frequency of unique TLS record sizes in a session. This method effectively detects malicious encrypted traffic.

Dataset Overview

Name: Network Traffic Data: Malicious Activity Detection

Source: Kaggle Dataset <https://www.kaggle.com/datasets/advaitnmenon/network-traffic-data-malicious-activity-detection>

Size: 8 columns with a still-to-be-verified number of rows.

Input Features:

- Timestamp: Time when the packet was captured.

- Source IP Address: Sender's IP address.

- Destination IP Address: Receiver's IP address.

- Source Port: Communication port used by the sender.

- Destination Port: Communication port used by the receiver.

- Protocol: Network protocol (e.g., TCP, UDP).

- Length: Size of the packet.

Target Variable:

- bad_packet: A binary label (1 = malicious packet, 0 = normal packet).

Dataset Loading and Description



	Time	Source	Destination	Protocol	Length	Source Port	Destination Port
0	0.000000	VMware_8a:60:5f	Broadcast	ARP	60	NaN	NaN
1	0.081233	VMware_8a:12:84	Broadcast	ARP	60	NaN	NaN
2	0.217863	VMware_8a:7e:e9	Broadcast	ARP	60	NaN	NaN
3	0.419426	VMware_8a:b2:34	Broadcast	ARP	60	NaN	NaN
4	0.559886	VMware_8a:4e:1c	Broadcast	ARP	60	NaN	NaN

	Destination Port	bad_packet
0	NaN	0
1	NaN	0
2	NaN	0
3	NaN	0
4	NaN	0

Dataset description

This dataset has 8 columns, with missing values in Source Port and Destination Port.

The bad_packet column is a target variable, indicating whether a packet is malicious (1) or normal (0).

Features like Source, Destination, and Protocol are categorical, while Time, Length, and port columns are numerical.

Preprocessing will be required to handle missing values and encode categorical features for modeling.

Categorical features require encoding for machine learning models, while numerical features are typically scaled or normalized for better model performance.

categorical features often need to be transformed into numerical form using techniques like one-hot encoding, label encoding, or hash encoding.

Dataset Preprocessing: Dataset shape

```



=> ===== Train data =====
Dataframe shape: (3245180, 8)
Dataframe columns: Index(['Time', 'Source', 'Destination', 'Protocol', 'Length', 'Source Port',
                          'Destination Port', 'bad_packet'],
                          dtype='object')

```

Dataset PreProcessing: Dataset structure

=>

	Time	Source	Destination	Protocol	Length	Source Port	Destination Port	bad_packet
0	0.000000	VMware_8a:60:5f	Broadcast	ARP	60	NaN	NaN	0
1	0.081233	VMware_8a:12:84	Broadcast	ARP	60	NaN	NaN	0
2	0.217863	VMware_8a:7e:e9	Broadcast	ARP	60	NaN	NaN	0
3	0.419426	VMware_8a:b2:34	Broadcast	ARP	60	NaN	NaN	0
4	0.559886	VMware_8a:4e:1c	Broadcast	ARP	60	NaN	NaN	0

Dataset Preprocessing: missing values

```

→ ▾ ==== Train data ====
Time                                0
Source                             0
Destination                         0
Protocol                           0
Length                             0
Source Port                        3241718
Destination Port                   3241718
bad_packet                          0
dtype: int64

```

Source Port and Destination Port are likely not useful features for model training due to their high proportion of missing values.

Data Analysis

3.1. Data types

```

1  ## Load the Dataset
2  import pandas as pd
3  df = pd.read_csv('/content/sample_data/NandakumarMenonAdvait_MT_S2.csv')
4  df.dtypes

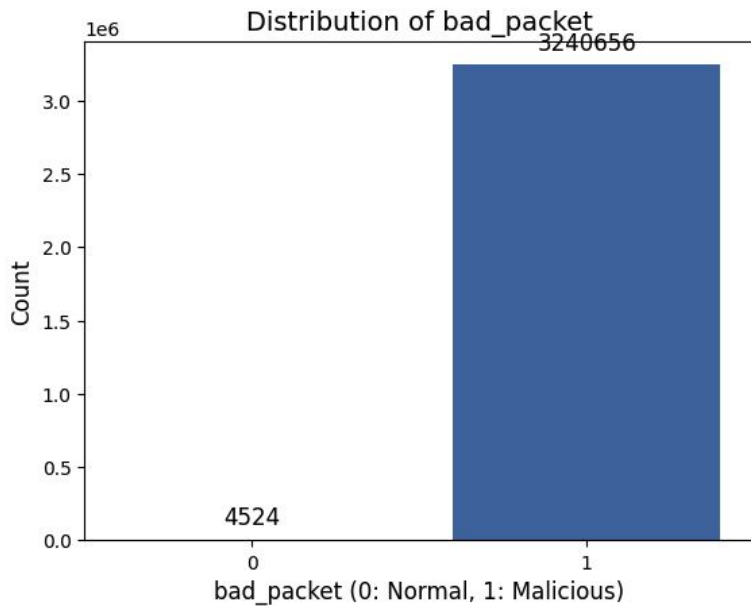
```

	0
Time	float64
Source	object
Destination	object
Protocol	object
Length	int64
Source Port	float64
Destination Port	float64
bad_packet	int64

dtype: object

In this section I analyze the datasets in order to have a better understanding of the data.

Data Analysis: Observing the distribution of the target variable



Given your dataset with 3,240,656 rows labeled 1 (malicious packets) and 4,524 rows labeled 0 (normal packets). The class is imbalanced

```
The size of df_final is: 3245180 rows and 8 columns.
Distribution of 'bad_packet':
bad_packet
1      3240656
0         4524
Name: count, dtype: int64
```

Data Analysis: Check uniqueness

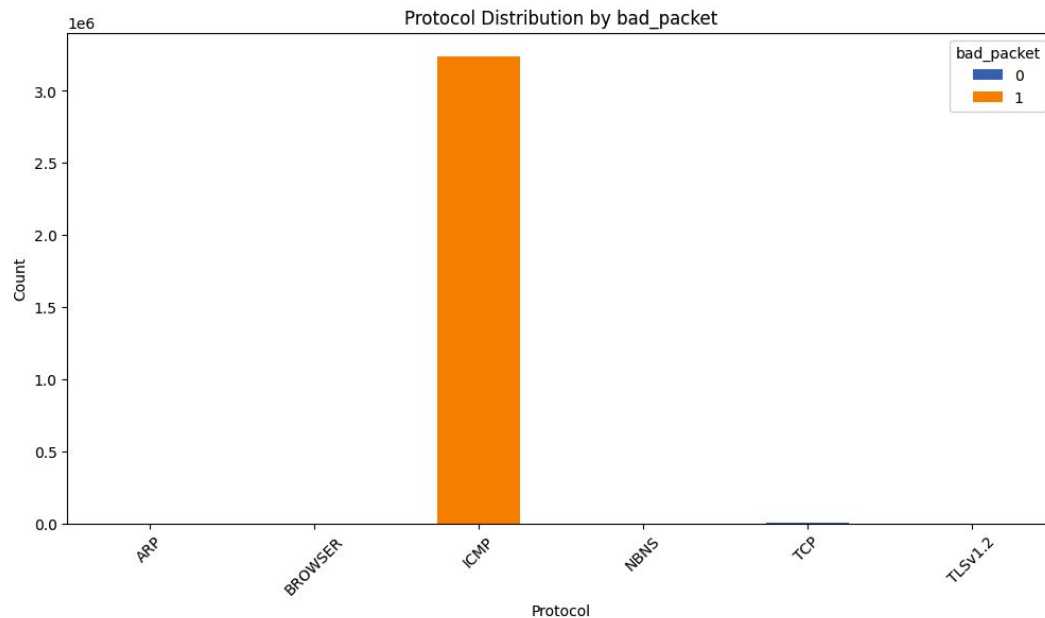
```
Time: 3245180 unique values
Source: 29 unique values
Destination: 9 unique values
Protocol: 6 unique values
Length: 67 unique values
Source Port: 6 unique values
Destination Port: 6 unique values
```

Check the number of unique values in each column

Data Analysis

bad_packet	0	1
Protocol		
ARP	241	0
BROWSER	47	0
ICMP	821	3240656
NBNS	53	0
TCP	2974	0
TLSv1.2	388	0

Protocol Distribution: the most important is ICMP, which is dominated in my dataset



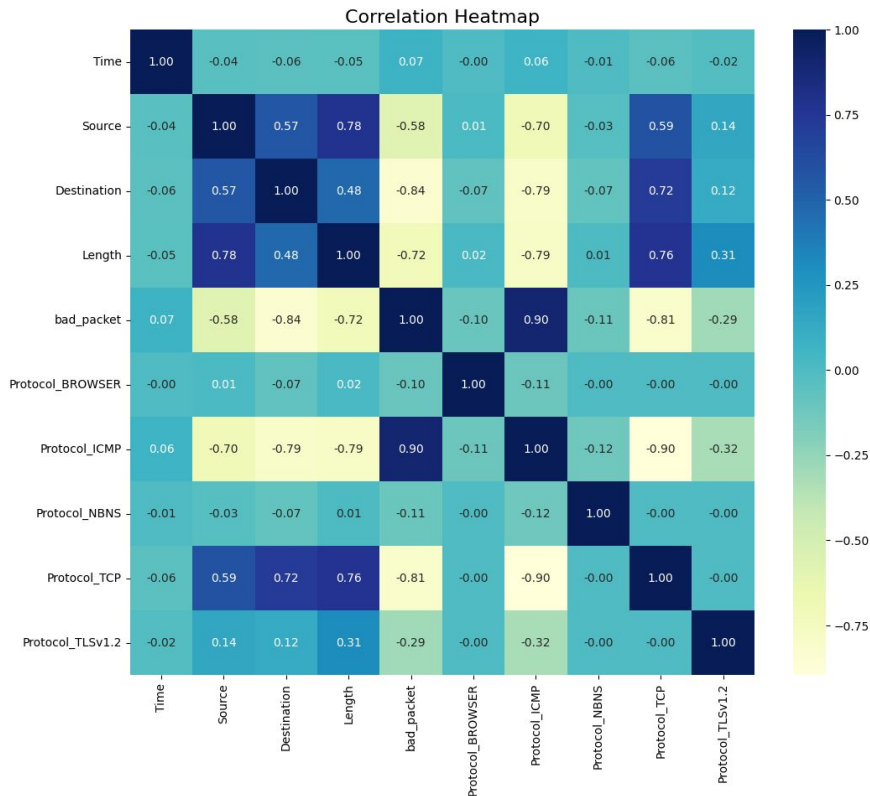
Remove useless columns: source and destination port

Convert categorical columns into numerical format:

- Label Encoding for high-cardinality columns: source and destination address.

- One-Hot Encoding for low-cardinality columns: protocol

Correlation



Correlation with bad_packet:

```

bad_packet      1.000000
Protocol_ICMP   0.904608
Time            0.067949
Protocol_BROWSER -0.101856
Protocol_NBNS   -0.108163
Protocol_TLSv1.2 -0.292670
Source          -0.577653
Length          -0.717578
Protocol_TCP    -0.810598
Destination     -0.837708
Name: bad_packet, dtype: float64
  
```

Dataset preparation

Splitting the dataset for training and testing (70%,30%)

```
Training set size: 2271626 samples  
Test set size: 973554 samples
```

Check if the dataset are balanced

```
Train set distribution:  
bad_packet  
1    0.998601  
0    0.001399  
Name: proportion, dtype: float64
```

```
Validation set distribution:  
bad_packet  
1    0.998617  
0    0.001383  
Name: proportion, dtype: float64
```

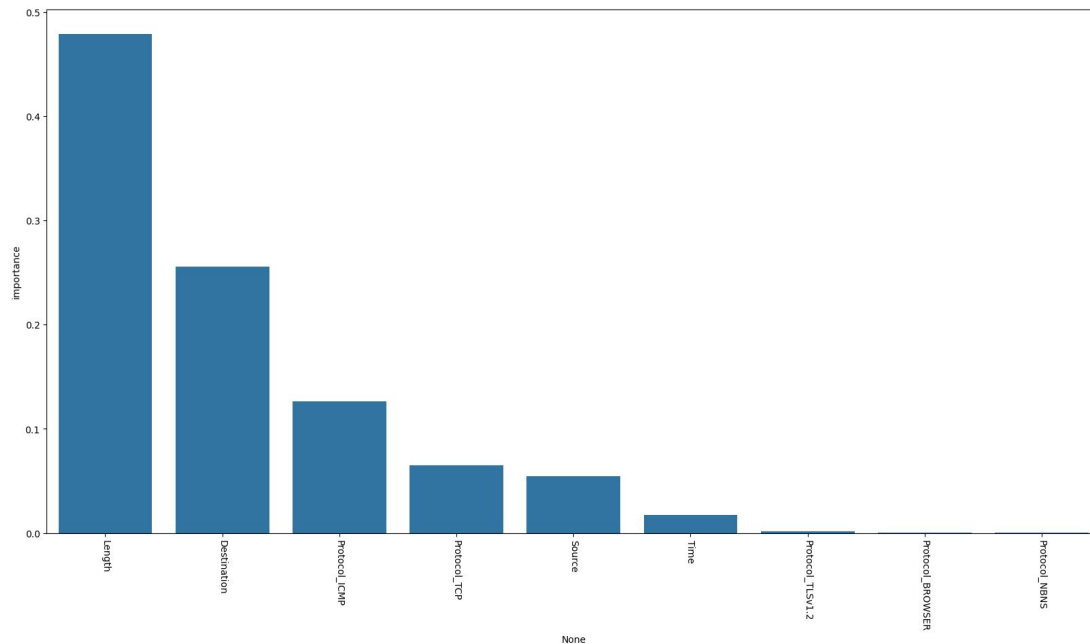
Data Scaling

Scaling the data is important to avoid that some features will have a bigger impact on the model training than others. This is especially important when we are dealing with features that have different units of measure. I use StandardScaler.

Feature Selection:

Apply Random Forest Classifier. Then, I have rank features based on their value of importance.

Drop feature that has value less than 0.02



	importance
Length	0.478866
Destination	0.255469
Protocol_ICMP	0.126139
Protocol_TCP	0.065075
Source	0.054779
Time	0.017196
Protocol_TLSv1.2	0.001998
Protocol_BROWSER	0.000248
Protocol_NBNS	0.000231

Selected features from feature importance

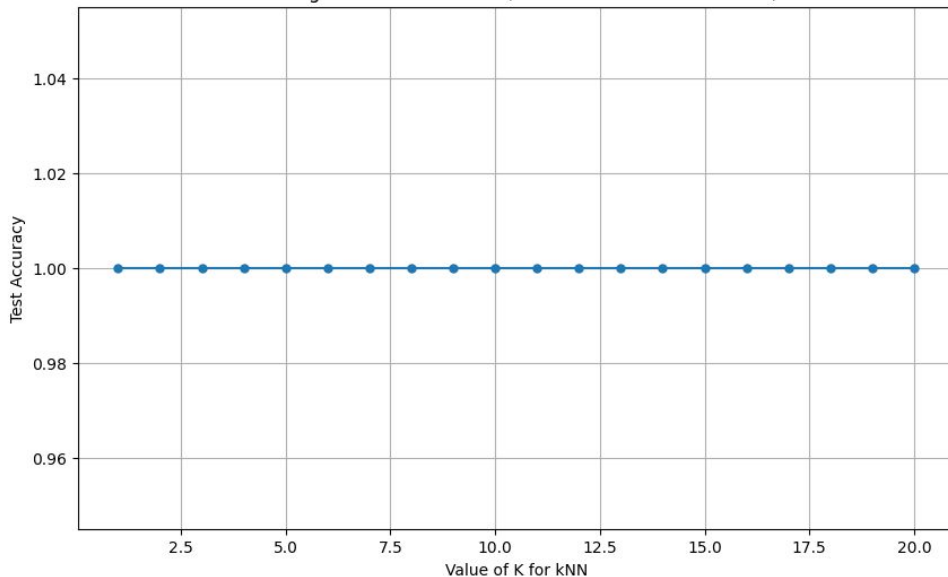
```
selected_features = ['Length', 'Destination', 'Protocol_ICMP', 'Protocol_TCP', 'Source']
```

Model Training and Evaluation

In this section we will train different models and compare their results. We will use the following models:
K-Nearest Neighbors (KNN), SVC, and Random Forest Classifier

Fit model with best K hyperparameter + make predictions

Finding the Best K for kNN (GPU-Accelerated with cuML)



Best k: 1
Best accuracy: 1.0000

Classification Report for Best K:				
	precision	recall	f1-score	support
0	1.00	1.00	1.00	1346
1	1.00	1.00	1.00	972208
accuracy			1.00	973554
macro avg	1.00	1.00	1.00	973554
weighted avg	1.00	1.00	1.00	973554

Model Training and Evaluation

Support Vector Machine Classifier (SVC)

```
Training the SVM model...
Predicting with SVM...
```

SVM Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1346
1	1.00	1.00	1.00	972208
accuracy			1.00	973554
macro avg	1.00	1.00	1.00	973554
weighted avg	1.00	1.00	1.00	973554

Model Training and Evaluation

Random Forest Classifier

```

Training the Random Forest model...
Predicting on the test set...
Random Forest Classification Report:

```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1346
1	1.00	1.00	1.00	972208
accuracy			1.00	973554
macro avg	1.00	1.00	1.00	973554
weighted avg	1.00	1.00	1.00	973554

Further Work

- Handle the imbalance in target variable
- Address any future suggestions for final report

Thank you!