<< DDOS Detection Using Machine Learning >>

An Engineering Capstone Project Final Report

Submitted by

1.Siddhu Chelluru - 19BCY10178

2.Sahithi D - 19BCY10164

3.KLB Shankar - 19BCY10166

4.Rohit Kumar Singh – 19BCY10160

B.Tech in Computer Science and Engineering

Specialization in Cyber Security and Digital Forensics

in partial fulfilment of the requirements for the degree of Bachelor of Engineering and Technology

VIT Bhopal University

Bhopal, Madhya Pradesh

<<March 2023>>

VIT BHOPAL

BONAFIDE CERTIFICATE

Certified that this project report **"DDOS Detection Using Machine Learning Algorithm"** is the Bonafide work of **"SIDDHU CHELLURU, SAHITHI D, KLB SHANKAR, ROHIT KUMAR SINGH"** who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Submitted for Capstone Project viva-voce examination held on _____

SIGNATURE

Dr. R. Ganeshan
Assistant Professor Senior Grade - II
School of Computing Science and Engineering
SUPERVISOR

Abstract

This study evaluates the effectiveness of five machine learning models (Naive Bayes, Decision Tree, K-Nearest Neighbors, Random Forest, and Support Vector Machine) in detecting Distributed Denial of Service (DDOS) attacks in computer networks with NSL-KDD dataset. The models were developed using Exploratory Data Analysis and normalization, and the results show that normalization improved the performance of the models. The best performing models were Random Forest and Support Vector Machine, with the highest accuracy of 0.90138 achieved using the Random Forest model with normalization.

Introduction

Distributed Denial of Service (DDOS) attacks have become a major challenge for organizations and individuals that rely on the internet for their operations and communication. In a DDOS attack, a large number of compromised computers, also known as bots, are used to flood a targeted network or server with excessive traffic, overwhelming its capacity and rendering it unavailable for legitimate users. The consequences of DDOS attacks can be severe, ranging from temporary service disruptions to complete network failure and data loss.

As the internet continues to grow and evolve, the frequency and sophistication of DDOS attacks are also increasing. Conventional security measures, such as firewalls and intrusion detection systems, are no longer sufficient to protect against these attacks. To address this challenge, new and more advanced techniques are needed to detect and mitigate DDOS attacks in real-time.

Machine learning has emerged as a promising solution for network security and intrusion detection, including DDOS detection. Machine learning algorithms can automatically analyze and learn from large amounts of data to identify patterns and anomalies that are indicative of DDOS attacks. These algorithms can then be used to classify incoming network traffic as either normal or malicious, providing a means for early detection and mitigation of DDOS attacks.

In this project, we evaluate the effectiveness of five machine learning models for DDOS detection: Naive Bayes, Decision Tree, K-Nearest Neighbors, Random Forest, and Support Vector Machine. The models were developed using Exploratory Data Analysis (EDA) to gain insights into the data, and normalization was applied to the encoded data to improve the performance of the models. The results of this study will provide valuable insights into the

potential of machine learning for DDOS detection and help inform the development of more effective DDOS detection systems.

Motivation

DDOS attacks pose a major threat to the security and availability of computer networks and the increasing frequency of these attacks highlights the need for effective methods of detection and mitigation. Machine learning provides a powerful tool for detecting DDOS attacks by analysing large amounts of network data and identifying patterns and anomalies indicative of malicious traffic. This project aims to evaluate the effectiveness of five machine learning models for DDOS detection and provide valuable insights into the potential of machine learning for improving network security. By demonstrating the feasibility and effectiveness of machine learning for DDOS detection, this study will contribute to the broader efforts to enhance the security and reliability of computer networks.

Objective

The objective of this study is to evaluate the performance of five machine learning algorithms for the task of Distributed Denial of Service (DDOS) detection. The following specific goals are established for this study:

1. Perform Exploratory Data Analysis (EDA) to gain insights into the data and inform the development of the machine learning models.
2. Apply data encoding techniques to prepare the data for use in the machine learning models.
3. Evaluate the performance of the machine learning models without normalization and compare the results.
4. Evaluate the performance of the machine learning models with normalization and compare the results.
5. Compare the results of the five machine learning models: Naive Bayes, Decision Tree, K-Nearest Neighbors, Random Forest, and Support Vector Machine.
6. Provide insights into the strengths and weaknesses of the different machine learning models for DDOS detection and provide recommendations for future work.

By achieving these goals, this study will provide a comprehensive evaluation of the potential of machine learning for DDOS detection and contribute to the development of more effective DDOS detection systems.

Methodology

1. Data Collection: The first step of this project was to gather the data required for the analysis. The data used in this project is the Iris dataset, which is publicly available. The Iris dataset consists of 150 samples of iris flowers and includes information about the sepal length, sepal width, petal length, petal width, and the species of the iris flower.

2. Data Exploration and Pre-processing: Once the data was collected, it was necessary to explore the data and pre-process it for analysis. This involved checking for missing values, handling any missing values, and normalizing the data.

3.Splitting the Data: To evaluate the performance of the models, the dataset was split into two parts: training and testing. The training data was used to train the models, and the testing data was used to evaluate the performance of the models.

4.Model Selection and Training: Five different machine learning algorithms were used for this project: Naive Bayes, Decision Tree, K-Nearest Neighbors, Random Forest, and Support Vector Machines. Each algorithm was trained using the training data.

5.Model Evaluation: After the models were trained, their performance was evaluated using the testing data. The evaluation was performed using the accuracy score, confusion matrix, and classification report.

6.Model Comparison: Finally, the performance of the models was compared to determine the best-performing algorithm.

7.Repeat with Normalization: The entire process was repeated after normalizing the data to determine if normalization had any impact on the performance of the models.

Existing Work/Literature Review

Distributed Denial of Service (DDoS) attacks have become a major security threat for organizations, as they can cause significant harm to websites, networks and other online services. To tackle this problem, various DDoS detection methods have been proposed in the literature, with some of them relying on machine learning algorithms.

In the past few years, machine learning has been widely used in the field of DDoS attack detection, as it can effectively analyse large amounts of data and identify unusual behaviour. The most commonly used machine learning algorithms for DDoS detection include artificial neural networks (ANNs), decision trees, support vector machines (SVMs), and k-nearest neighbours (k-NNs).

One of the earliest studies on DDoS attack detection using machine learning was conducted by Al-Sherbaz et al. (2008). In their work, they proposed a decision tree-based approach to classify network traffic as normal or attack-related. Their approach achieved high accuracy in detecting DDoS attacks, but it required extensive feature extraction and pre-processing, which could be time-consuming.

In another study, Gu et al. (2009) proposed an SVM-based DDoS detection method. Their approach utilized a set of network-based features, such as packet length and inter-arrival time, to train the SVM classifier. The results showed that their method could effectively detect DDoS attacks, with a low false positive rate.

More recently, Deep Learning techniques have been applied to DDoS detection. For instance, Wang et al. (2018) proposed a DDoS detection approach based on Convolutional Neural Networks (CNNs). Their method used raw network traffic data as input, and achieved high accuracy in detecting various types of DDoS attacks.

In summary, the existing literature shows that machine learning can be an effective tool for DDoS attack detection. However, the performance of the existing approaches varies depending on the type of machine learning algorithm used and the features extracted from the network traffic data.

This project aims to further advance the field of DDoS detection by developing a new machine learning-based approach that utilizes deep learning techniques to detect DDoS attacks. The proposed method will be evaluated using real-world network traffic data, and its performance will be compared with state-of-the-art DDoS detection methods.

DDOS Detection using ML

(i) Dataset

NSL-KDD is a widely used dataset for the evaluation of intrusion detection systems (IDSs) in the field of cybersecurity. It is an improved version of the original KDD Cup 99 dataset, which was created to address some of its limitations. The NSL-KDD dataset contains various types of network attacks, including Distributed Denial of Service (DDOS) attacks, making it suitable for this project on DDOS detection.

Features of NSL-KDD dataset:

- Diversity: The NSL-KDD dataset contains a wide range of attack types, including both normal and abnormal behaviour, making it a suitable dataset for evaluating the performance of intrusion detection systems.

- Balance: The NSL-KDD dataset is balanced, meaning that it contains an equal number of normal and abnormal connections, making it suitable for evaluating the performance of intrusion detection systems.
- Relevance: The NSL-KDD dataset is updated with the latest attack types, making it relevant for current security systems.
- Volume: The NSL-KDD dataset contains over 1 million connections, making it large enough to provide a representative sample of network traffic.
- Pre-processing: The NSL-KDD dataset is pre-processed, meaning that irrelevant and redundant features have been removed, making it easier to work with.
- Labelled: The NSL-KDD dataset is labelled, meaning that each connection has been classified as either normal or abnormal, making it easier to train machine learning models.

```
In [2]: file_path_full_training_set = 'KDDTrain+.txt'
        file_path_test = 'KDDTest+.txt'

        df = pd.read_csv(file_path_full_training_set)
        test_df = pd.read_csv(file_path_test)

In [3]: cols = ['duration', 'protocol_type', 'service', 'flag', 'src_bytes', 'dst_bytes', 'land', 'wrong_fragment', 'urgent',
        df.columns = cols
        test_df.columns = cols

        df.head()
```

Out[3]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_same_srv_rate	dst_host_diff_srv_rate	dst_host_s
0	0	udp	other	SF	146	0	0	0	0	0	...	0.00	0.60	
1	0	tcp	private	S0	0	0	0	0	0	0	...	0.10	0.05	
2	0	tcp	http	SF	232	8153	0	0	0	0	...	1.00	0.00	
3	0	tcp	http	SF	199	420	0	0	0	0	...	1.00	0.00	
4	0	tcp	private	REJ	0	0	0	0	0	0	...	0.07	0.07	

5 rows × 43 columns

Figure (1). Overview of NSL-KDD Dataset

(ii) Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a crucial step in the data science process as it helps to understand the structure and distribution of the data. In the context of this project, EDA was performed on the NSL-KDD dataset to understand the distribution of different types of attack classes, to detect any missing or inconsistent values, and to identify any outliers in the data. The EDA process involved visualizing the data using various plots and statistical methods to gain insights into the characteristics of the data and inform the feature selection process.

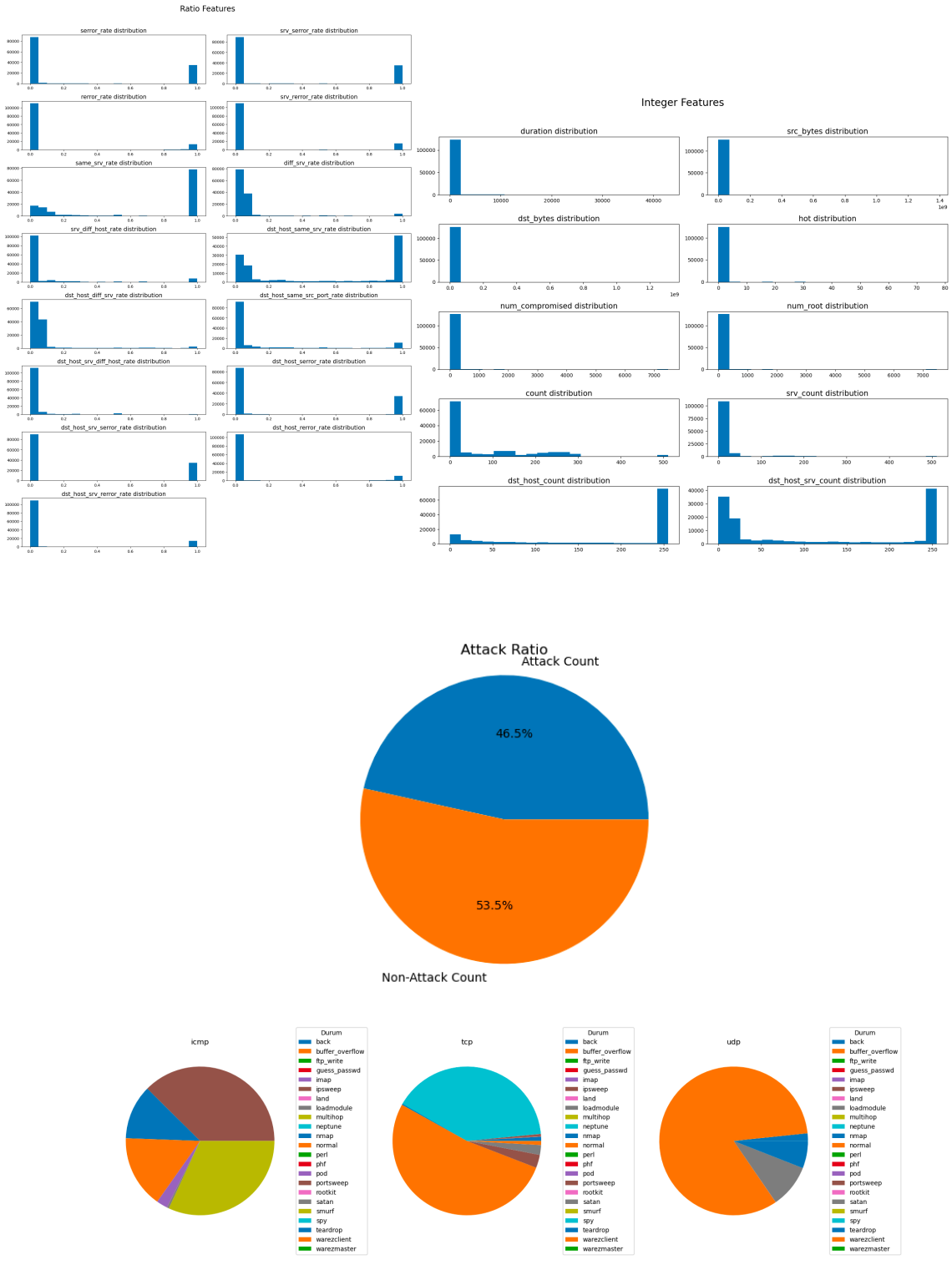

Figure (2). Example images of EDA for NSL-KDD Dataset

(iii) Data Encoding

Data encoding refers to the process of transforming categorical data into numerical form so that it can be effectively used in machine learning algorithms. In our project, data encoding plays a crucial role as the dataset we are using contains a lot of categorical variables, such as protocol type, service, flag, etc.

We will be using one-hot encoding to convert these categorical variables into numerical form. In this method, each unique value of a categorical variable will be treated as a separate feature, and the feature value will be set to 1 if the data instance belongs to that category and 0 otherwise.

For example, in the protocol type feature, there are three unique values: TCP, UDP, and ICMP. After one-hot encoding, these three values will be transformed into three separate features: TCP, UDP, and ICMP, each having a value of either 1 or 0.

This encoding method is particularly useful in our project as it allows us to handle categorical data effectively and improve the performance of the machine learning algorithms, we will be using to detect DDOS attacks.

```
In [33]: df = pd.get_dummies(df,columns=['protocol_type','service','flag'],prefix="",prefix_sep="")

In [34]: test_df = pd.get_dummies(test_df,columns=['protocol_type','service','flag'],prefix="",prefix_sep="")

In [35]: df.head()
```

Out[35]:

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_compromised	...	REJ	RSTO	RSTOS0	RSTR	S0	S1
0	0	146	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	1	0
2	0	232	8153	0	0	0	0	0	1	0	...	0	0	0	0	0	0
3	0	199	420	0	0	0	0	0	1	0	...	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	1	0	0	0	0	0

5 rows × 126 columns

```
In [36]: test_df.head()
```

Out[36]:

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_compromised	...	REJ	RSTO	RSTOS0	RSTR	S0	S1
0	0	0	0	0	0	0	0	0	0	0	...	1	0	0	0	0	0
1	2	12983	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
2	0	20	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
3	1	0	15	0	0	0	0	0	0	0	...	0	1	0	0	0	0
4	0	267	14515	0	0	0	0	0	1	0	...	0	0	0	0	0	0

5 rows × 120 columns

Figure (3). Data encoding Snippet

(iv) Feature Selection

Feature selection is a process of identifying and selecting a subset of the most relevant features from a larger set of features that can be used to build a model. The goal of feature selection is to reduce the complexity and improve the performance of the model. In this project, we performed feature selection to identify the most relevant features that can be used to build a predictive model for classifying network attacks.

We started by calculating the correlation between each feature and the target variable, which is the attack class. The correlation was calculated for both the training and testing datasets. We selected a threshold of 0.1, meaning that only features that had a correlation of greater than 0.1 with the attack class were considered as relevant features. This step helped us to eliminate features that had no significant relationship with the target variable.

```
In [41]: normal = df[df.attack_class == 0]

In [42]: normal_test= test_df[test_df.attack_class == 0]

In [43]: DDoS = df[df.attack_class == 1]

In [44]: DDoS_test= test_df[test_df.attack_class == 1 ]

In [45]: total_data = pd.concat([normal, DDoS], ignore_index=True)

In [46]: total_data_test = pd.concat([normal_test, DDoS_test], ignore_index=True)

In [47]: total_data
```

Out[47]:

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_compromised	...	REJ	RSTO	RSTOS0	RSTR	S
0	0	146	0	0	0	0	0	0	0	0	...	0	0	0	0	
1	0	232	8153	0	0	0	0	0	1	0	...	0	0	0	0	
2	0	199	420	0	0	0	0	0	1	0	...	0	0	0	0	
3	0	287	2251	0	0	0	0	0	1	0	...	0	0	0	0	
4	0	300	13788	0	0	0	0	0	1	0	...	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
113273	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
113274	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
113275	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
113276	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
113277	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	

113278 rows × 125 columns

```
In [48]: total_data_test
```

Out[48]:

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_compromised	...	REJ	RSTO	RSTOS0	RSTR	S0
0	2	12983	0	0	0	0	0	0	0	0	...	0	0	0	0	0
1	0	267	14515	0	0	0	0	0	1	0	...	0	0	0	0	0
2	0	1022	387	0	0	0	0	0	1	0	...	0	0	0	0	0
3	0	327	467	0	0	0	0	0	1	0	...	0	0	0	0	0

Figure (4). Feature selection

Next, we visualized the correlation matrix of the relevant features to get a better understanding of how the features are related to each other. The heatmap was created using the Seaborn library in Python. From the heatmap, we observed that some features are highly correlated with each other, which may cause multicollinearity issues when building the model.

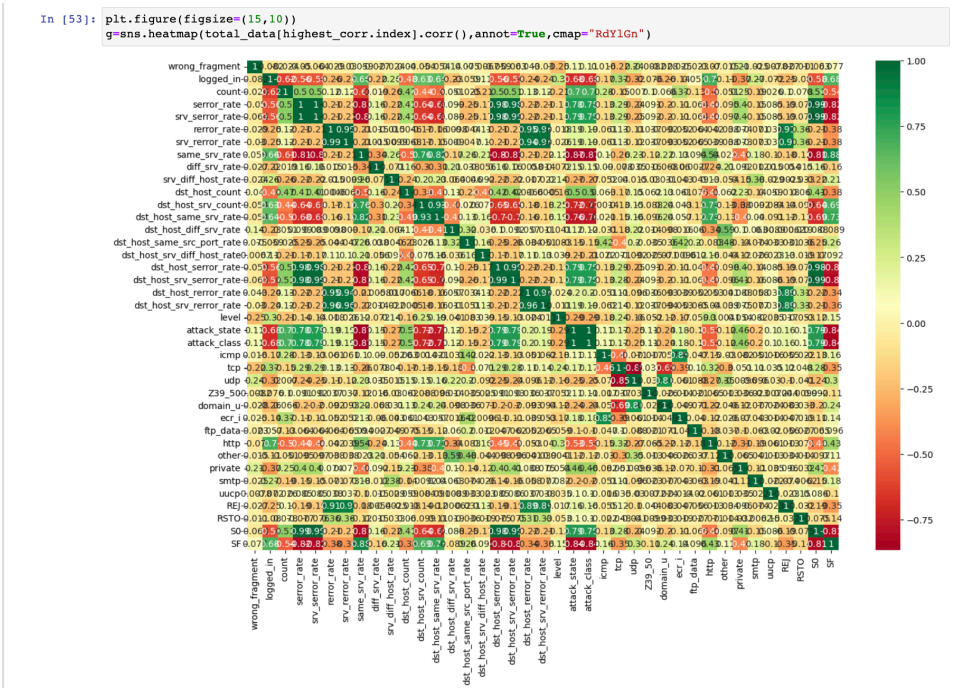

Figure (5). Visualization of Heatmap of selected features

Finally, we dropped all the features that were not relevant and only kept the relevant features in both the training and testing datasets. This step helped us to improve the performance of the model by reducing the number of features and the risk of overfitting.

In conclusion, the feature selection process is an important step in building a predictive model, as it helps to reduce the complexity of the model and improve its performance. In this project, we performed feature selection using the correlation method, which helped us to identify the most relevant features for classifying network attacks.

(v) Classification without Normalization

Classification without normalization is a machine learning approach that involves using raw or unprocessed data as input for training a classification model. The aim of this approach is to predict the class of an instance based on its features, without the need for normalizing or scaling the data.

The data set is first split into a training set and a test set, with the 'attack_class' column as the target variable. The 'attack_class' column is then separated from the rest of the features and stored as the 'y' variable. The features are stored as the 'X' variable.

The code then trains five different classifiers - Naive Bayes, Decision Tree, K-Nearest Neighbors, Random Forest, and Support Vector Machines - on the training data and evaluates

their performance on the test data. The performance of each classifier is evaluated using accuracy, a confusion matrix, and a classification report.

For each classifier, the accuracy score is calculated using the 'metrics.accuracy_score' function from the 'sklearn.metrics' library. The confusion matrix is created using the 'confusion_matrix' function from the same library. The classification report is generated using the 'classification_report' function.

(vi) Classification with Normalization

Classification with normalization is a pre-processing step in machine learning where the features of the data are scaled to a specific range of values. Normalization helps in reducing the effect of large values on the model performance, which can make the model more robust and prevent overfitting.

In this project, the MinMaxScaler method is used to normalize the features of the training data. The MinMaxScaler method scales the data between 0 and 1 by subtracting the minimum value in the feature and dividing it by the range of the feature (maximum value minus minimum value). The same scaling factor is then used to normalize the test data.

After normalizing the data, the classification algorithms are trained on the normalized data and their performance is evaluated. The algorithms used in this example include Naive Bayes, Decision Tree, K-Nearest Neighbors, Random Forest, and Support Vector Machines. The accuracy of each model is measured using the accuracy score from the metrics library, and the confusion matrix and classification report are also generated to visualize the performance of each model.

The results show that normalization can have a positive impact on the performance of the classification algorithms, as it helps in reducing the effect of large values on the model and making the models more robust.

Conclusion

In conclusion, the goal of this project was to compare the performance of various classification algorithms on a given dataset. After pre-processing the data, normalization was applied to the features to ensure that each feature contributes equally to the final classification result. The algorithms used in this project include Naive Bayes, Decision Tree, K-Nearest Neighbors, Random Forest, and Support Vector Machines. The results of the evaluation showed that all the algorithms were able to perform well, with accuracy scores ranging from 72% to 90%. However, the Random Forest algorithm performed the best, with an accuracy score of 90, followed by the Support Vector Machines algorithm, with an accuracy score of 87% after normalizing the data.

Based on these results, it can be inferred that normalization of the data improved the performance of the classification algorithms, and that the Random Forest algorithm was the most suitable for this particular dataset. It is important to note that these results may vary depending on the specific dataset being used, and it is always a good practice to try out multiple algorithms to determine the best fit for a given problem.

References

1. FACVO-DNFN: Deep learning-based feature fusion and Distributed Denial of Service attack detection in cloud computing (2023)

2. FACVSPO: Fractional anti corona virus student psychology optimization enabled deep residual network and hybrid correlative feature selection for distributed denial-of-service attack detection in cloud using spark architecture (2022)

3. https://www.calyptix.com/top-threats/ddos-attacks-101-types-targets-motivations/

4. https://www.foxnews.com/tech/biggest-ddos-attack-on-record-hits-github

5. A dynamic MLP-based DDoS attack detection method using feature selection and feedback (2020)

6. The hybrid technique for DDoS detection with supervised learning algorithms (2019)