Deep Science Publishing
https://doi.org/10.70593/978-93-49307-78-0_3

Chapter 3

Feature selection and machine learning model optimization for DDoS detection

3.0 Project Implementation

3.1 Introduction

In this chapter, the Implementation design is divided into two sections, the system setup design, and the machine learning flow process design. Both are explained in detail in the section.

The dataset used for the experiment was obtained from the open-source database CSE-CIC-IDS2018 ('IDS 2018' 2022) and the generated data set.

The open-source dataset was used to train a large data set that contained 300967 instances of benign and DDoS datasets while the generated data set contained 28, 972 instances of benign and DDoS datasets. The dataset contained many fields in which "32 out of 80" features were used for the open data set and "31 out of 84" was used for the newly generated dataset.

A comparative study of supervised machine learning algorithms which will be used to predict the accuracy and clarity of how DDoS attacks are detected in the cloud will be presented and evaluated in terms of performance and accuracy.

3.2 System Setup

Figure 8 depicts the system set-up of this experiment comprising of host pc, kali Linux VM, Owncloud Platform and Oracle Virtual Box. Table 4 shows the components needed to set up the system for the experiment.

Table 4: Requirements for system setup design

S/N	Components	Description
1	Hardware	Intel Core i7 8th Gen PC
2	Operating System	Windows 10
3	Memory	34 GB RAM
4	Oracle VirtualBox	Virtual Machine v 6.0
5	Owncloud VirtualBox	UCS 4.4-with-owncloud 1
6	Kali Linux VirtualBox	Kali-Linux-2021.4a-VirtualBox-amd64
7	Slowloris Tool	Slowloris DDoS attack script

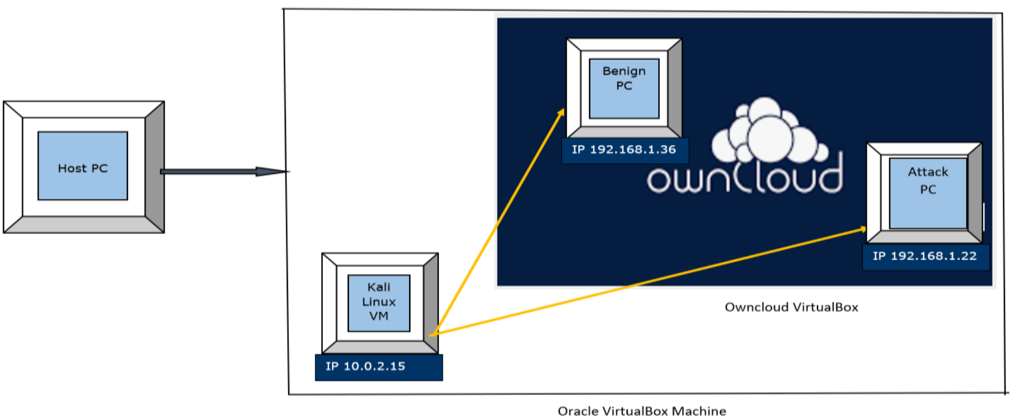

Figure 8: System Setup Design.

Brief description of the System setup elements as shown in Figure 8.

3.2.1 Network Type

A cloud network was deployed, specifically, Owncloud VirtualBox. This application was downloaded as an ova file from the internet and imported as a VirtualBox machine in the oracle VM installed in the host machine (researchers Computer). Other resources were configured such as setting up a graphical user interface, setting the application package, setting up an email server, and setting up two systems shown in Figure 8 for the collection of benign and DDoS traffics.

3.2.2 Kali Linux VM

This is a free and open-source operating system, its source code is freely accessible online, allowing you to examine it and modify it to suit your needs.

As shown in Figure 8 the system was used as an attacker PC which was used to conduct DDoS attacks on the targeted PC (attack pc) and for sending normal traffic to the own cloud benign PC.

3.2.3 Oracle VirtualBox

This is a system where the Owncloud VM and kali Linux VM were hosted which helps to experiment with a tight connection.

3.2.4 Host PC

This is the researcher's PC where all the VMs were hosted to perform the experiment.

3.2.5 Performing of DDoS attack in Owncloud VM

The DDoS attacks were conducted on the Owncloud VirtualBox environment which is a free and open-source platform for the generation of attacks. The attack was created in a safe environment using the slowloris tool for eight hours daily for three days. The attacking machine comprises Kali-Linux-2021.4a-VirtualBox-amd64, Debian 64 operating system with an IP 10.0.2.15 hosted on Oracle VirtualBox. Several terminals were opened representing several systems as shown in Figure 9 on each of the terminals, the slowloris script known as the bot was used with the IP address (192.168.1.36) of the attack PC (Target system) to send attack traffic to the attack PC seen in Figure 9.

The slowloris tool created traffic at the sink nodes during the execution of the DDoS attack, and the Wireshark packet analyser captured strange traffic during this time. The system then received the recorded traffic.

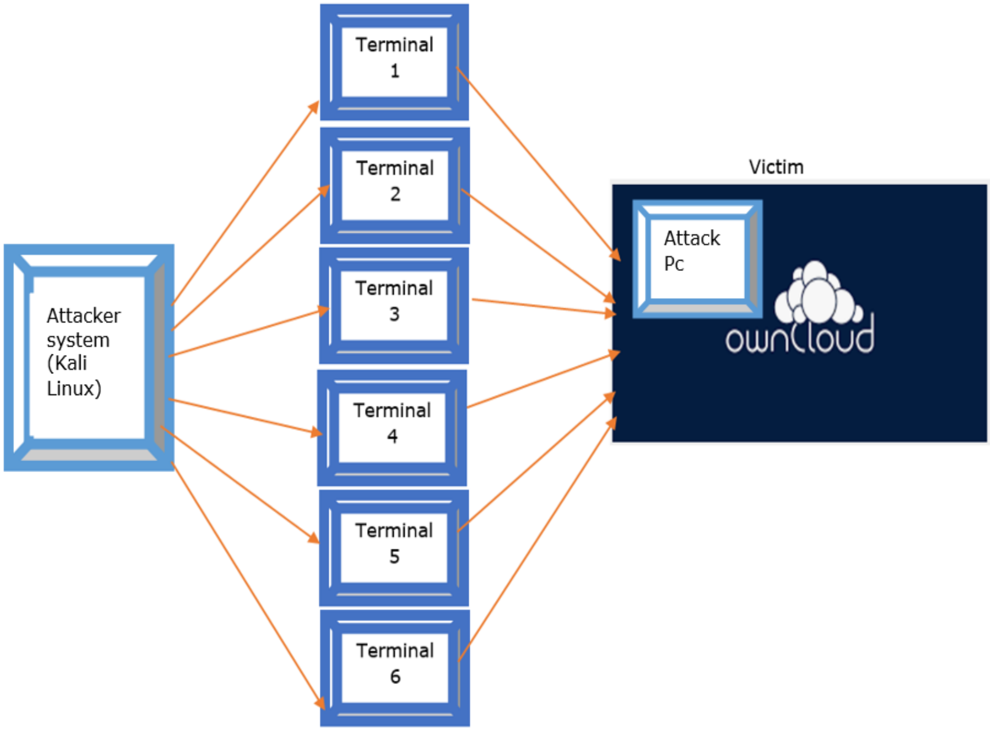

Figure 9: Performing DDoS attacks on Owncloud.

Table 5: Specification of tools and duration of DDoS attack traffic

Attack tool	Duration	Attacker	IP	Victim	IP
Slowloris	8 hours in a day. Total of 3 days	Kali Linux VM	10.0.2.15	Owncloud	192.168.1.36

3.2.6 Performing of Benign Traffic

Another system was set up in the Owncloud VM named benign PC with IP address 192.168.1.36 as shown in Figure 9 which was used to capture normal traffic. To build normal traffic, an email service was set up in the Owncloud benign PC such that there will be mail communication in and out of the system to create an SMTP protocol. Other tools like Nmap and ping were used to send traffic to the benign system to create varieties of protocols for normal traffic captured by Wireshark.

Table 6: Methods for collection and duration of benign traffic

Method	Duration	System	IP	Platform	IP
Nmap, Email, Ping, web browsing	6 hours a day, a total of 3 days	Kali Linux VM	10.0.2.15	Owncloud	192.168.1.22

3.2.7 Collection of Pcap Files

The DDoS and benign traffic captured by Wireshark was saved on the desktop of the Owncloud and collected as a pcap file. The pcap file was transferred to the host system through email configured in the Owncloud, the ones with large sizes were saved to one drive and then transported to the host PC. Furthermore, a filtering method was used to select real DDoS traffic from the pcap generated to ascertain that only DDoS traffic was collected. On the other hand, Benign traffic saved as a pcap file was transferred through email to the host PC and the large sizes were saved to one drive in the Owncloud and finally transported to the host PC. These two pcap files (DDoS and benign) were stored in the host PC for further processing. Machine learning process design as the second phase will be used to explain other processes involved in the experiment setup. Figure 10 and 11 shows the sample pcap files collected for the experiment.

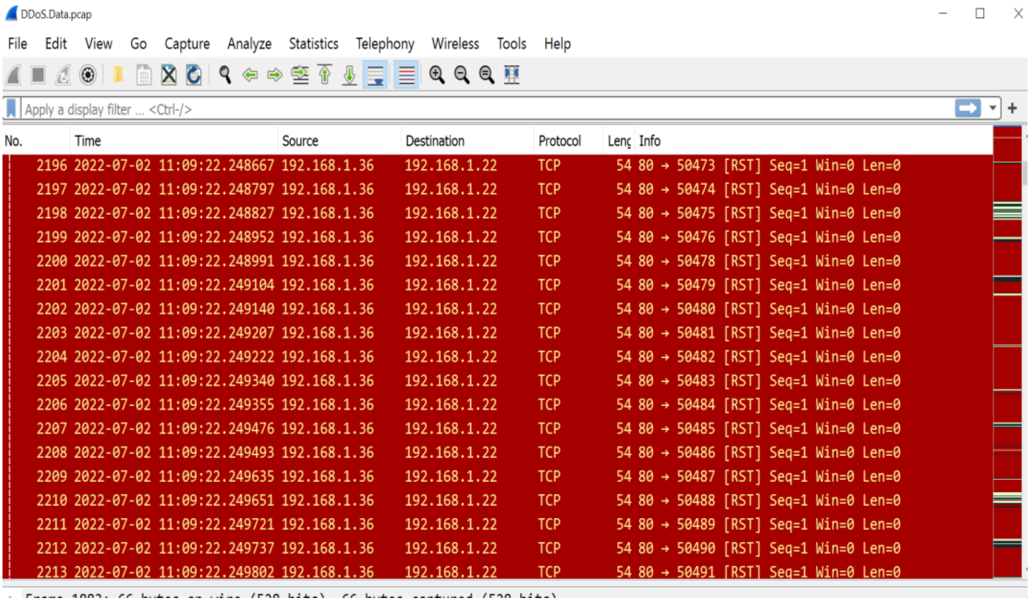

Figure 10: Sample DDoS pcap file.

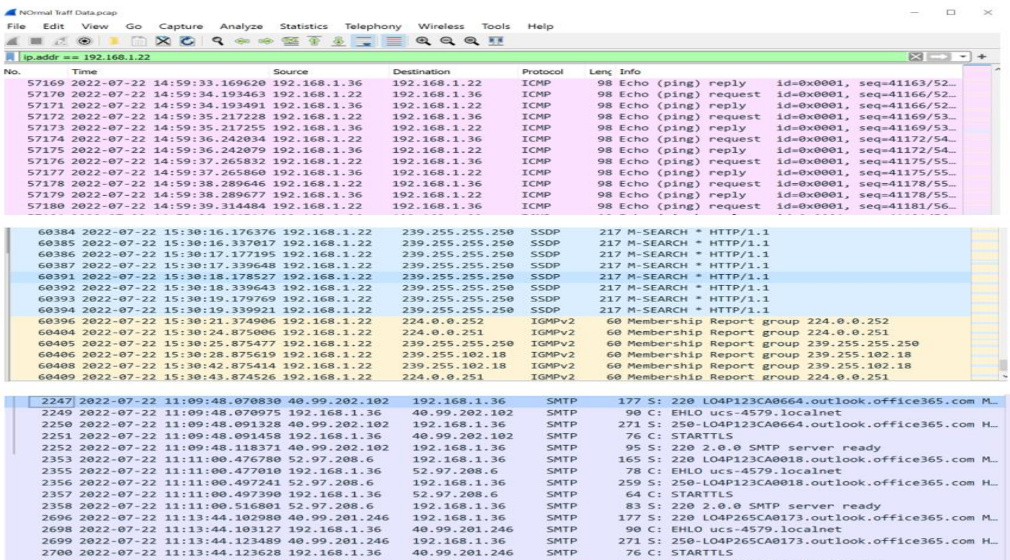

Figure 11: Sample Benign pcap file.

The next section continues with the Machine learning flow process.

3.3 Machine Learning Flow Process Design

This process was used in the second phase of the implantation process. Each flow content has been briefly explained in how they were used in this section. The components used to perform the ML process are seen in Table 7.

Table 7: Components Utilised

S/N	Components	Description
1	Hardware	Intel Core i7 8th Gen PC
2	Operating System	Windows 10
3	Memory	34 GB RAM
4	Libraries	Pandas, Sklearn, Matplotlib
5	Storage	MS Excel
	Feature extraction tool	CICFlowMeter 4.0
6	Programming language	Python 3.9
7	IDE	(Anaconda Jupyter Notebook)

Figure 12 illustrates the ML flow process design using supervised learning models.

3.3.1 Raw Data Collection

The raw data collected as pcap files stored on the host PC is required to be transformed into a format that is compatible with model building in machine learning. A tool named CICFlowMeter was used to make this transformation of the raw data into a possible real dataset CSV format as shown in Figure 13, this was stored in the host PC as an excel file. CICFlowMeter can be accessed using this link (https://github.com/ahlashkari/CICFlowMeter).

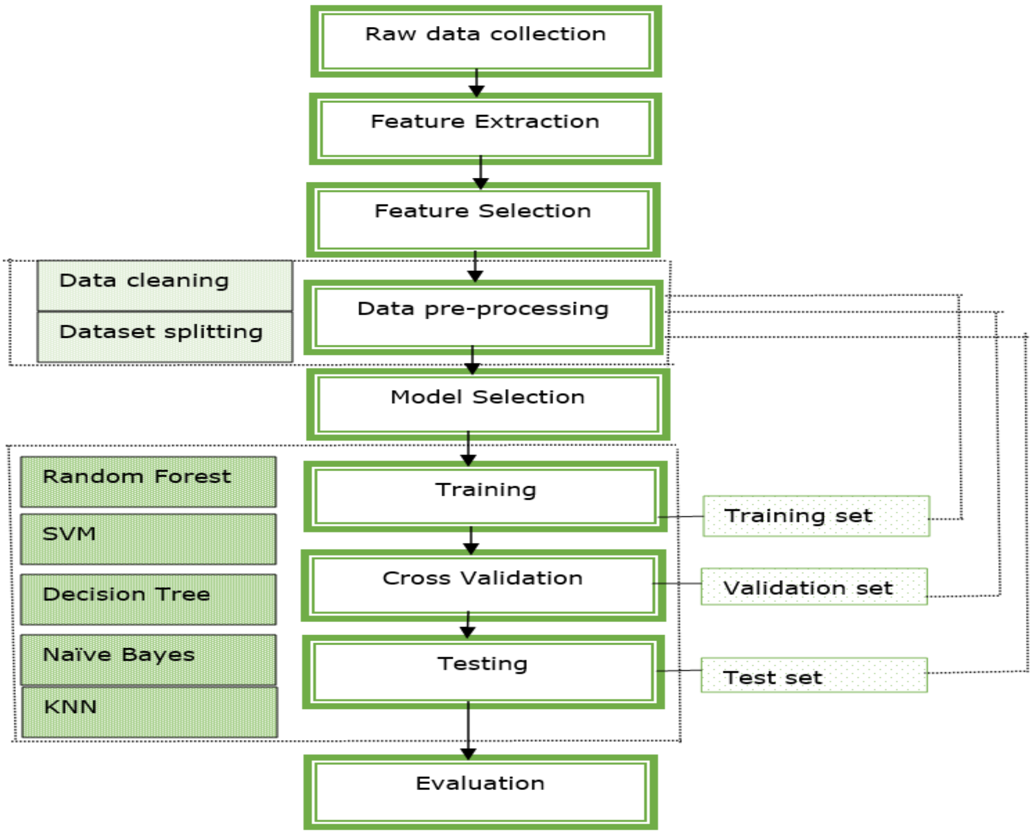

Figure 12: Machine Learning Flow process design.

3.3.2 Feature Extraction

CICFlowMeter 4.0 is a network traffic flow generator and analyser. over 84 statistical network traffic features, such as Duration, Number of packets, Number of bytes, Length of packets, etc., can be extracted separately in the forward and backward directions when using it to generate bidirectional flows, where the first packet determines the forward (source to destination) and backward (destination to source) directions. Picking features from a list of already-existing features, adding new features, and adjusting the flow timeout period are extra functionalities in this tool (Lashkari et al. 2017; Patil et al. 2022). Figures 14 and 15 (with CICFlowMeter 3.0) show the type of features extracted for this experiment.

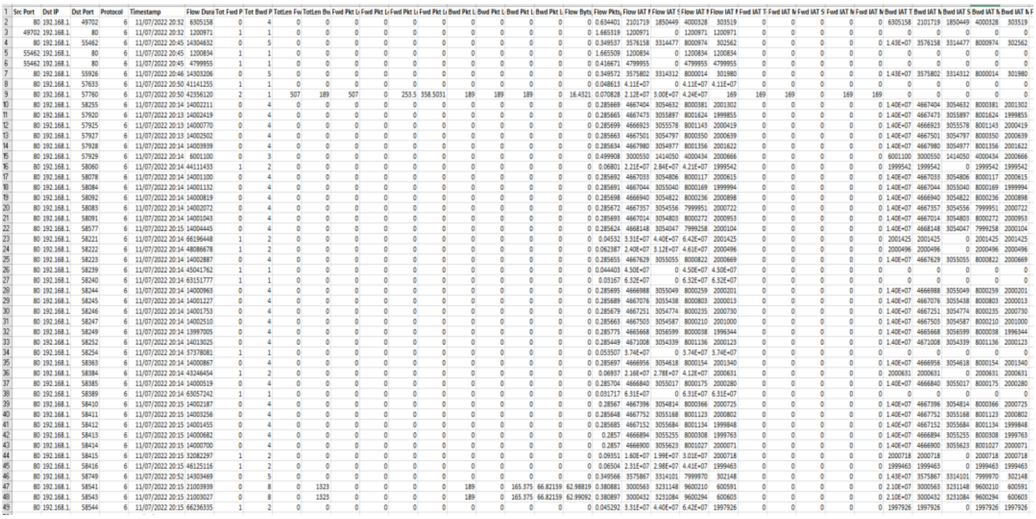

Figure 13: Sample of raw data transformed to CSV format.

1	Flow ID	22	Flow Pkts/s	43	Fwd Pkts/s	64	Fwd Blk Rate Avg
2	Src IP	23	Flow IAT Mean	44	Bwd Pkts/s	65	Bwd Byts/b Avg
3	Src Port	24	Flow IAT Std	45	Pkt Len Min	66	Bwd Pkts/b Avg
4	Dst IP	25	Flow IAT Max	46	Pkt Len Max	67	Bwd Blk Rate Avg
5	Dst Port	26	Flow IAT Min	47	Pkt Len Mean	68	Subflow Fwd Pkts
6	Protocol	27	Fwd IAT Tot	48	Pkt Len Std	69	Subflow Fwd Byts
7	Timestamp	28	Fwd IAT Mean	49	Pkt Len Var	70	Subflow Bwd Pkts
8	Flow Duration	29	Fwd IAT Std	50	FIN Flag Cnt	71	Subflow Bwd Byts
9	Tot Fwd Pkts	30	Fwd IAT Max	51	SYN Flag Cnt	72	Init Fwd Win Byts
10	Tot Bwd Pkts	31	Fwd IAT Min	52	RST Flag Cnt	73	Init Bwd Win Byts
11	TotLen Fwd Pkts	32	Bwd IAT Tot	53	PSH Flag Cnt	74	Fwd Act Data Pkts
12	TotLen Bwd Pkts	33	Bwd IAT Mean	54	ACK Flag Cnt	75	Fwd Seg Size Min
13	Fwd Pkt Len Max	34	Bwd IAT Std	55	URG Flag Cnt	76	Active Mean
14	Fwd Pkt Len Min	35	Bwd IAT Max	56	CWE Flag Count	77	Active Std
15	Fwd Pkt Len Mean	36	Bwd IAT Min	57	ECE Flag Cnt	78	Active Max
16	Fwd Pkt Len Std	37	Fwd PSH Flags	58	Down/Up Ratio	79	Active Min
17	Bwd Pkt Len Max	38	Bwd PSH Flags	59	Pkt Size Avg	80	Idle Mean
18	Bwd Pkt Len Min	39	Fwd URG Flags	60	Fwd Seg Size Avg	81	Idle Std
19	Bwd Pkt Len Mean	40	Bwd URG Flags	61	Bwd Seg Size Avg	82	Idle Max
20	Bwd Pkt Len Std	41	Fwd Header Len	62	Fwd Byts/b Avg	83	Idle Min
21	Flow Byts/s	42	Bwd Header Len	63	Fwd Pkts/b Avg	84	Label

Figure 14: 84 features extracted for a new dataset using CICFlowMeter 4.0.

1	Dst Port	21	Flow IAT Max	41	Pkt Len Min	61	Bwd Byts/b Avg
2	Protocol	22	Flow IAT Min	42	Pkt Len Max	62	Bwd Pkts/b Avg
3	Timestamp	23	Fwd IAT Tot	43	Pkt Len Mean	63	Bwd Blk Rate Avg
4	Flow Duration	24	Fwd IAT Mean	44	Pkt Len Std	64	Subflow Fwd Pkts
5	Tot Fwd Pkts	25	Fwd IAT Std	45	Pkt Len Var	65	Subflow Fwd Byts
6	Tot Bwd Pkts	26	Fwd IAT Max	46	FIN Flag Cnt	66	Subflow Bwd Pkts
7	TotLen Fwd Pkts	27	Fwd IAT Min	47	SYN Flag Cnt	67	Subflow Bwd Byts
8	TotLen Bwd Pkts	28	Bwd IAT Tot	48	RST Flag Cnt	68	Init Fwd Win Byts
9	Fwd Pkt Len Max	29	Bwd IAT Mean	49	PSH Flag Cnt	69	Init Bwd Win Byts
10	Fwd Pkt Len Min	30	Bwd IAT Std	50	ACK Flag Cnt	70	Fwd Act Data Pkts
11	Fwd Pkt Len Mean	31	Bwd IAT Max	51	URG Flag Cnt	71	Fwd Seg Size Min
12	Fwd Pkt Len Std	32	Bwd IAT Min	52	CWE Flag Count	72	Active Mean
13	Bwd Pkt Len Max	33	Fwd PSH Flags	53	ECE Flag Cnt	73	Active Std
14	Bwd Pkt Len Min	34	Bwd PSH Flags	54	Down/Up Ratio	74	Active Max
15	Bwd Pkt Len Mean	35	Fwd URG Flags	55	Pkt Size Avg	75	Active Min
16	Bwd Pkt Len Std	36	Bwd URG Flags	56	Fwd Seg Size Avg	76	Idle Mean
17	Flow Byts/s	37	Fwd Header Len	57	Bwd Seg Size Avg	77	Idle Std
18	Flow Pkts/s	38	Bwd Header Len	58	Fwd Byts/b Avg	78	Idle Max
19	Flow IAT Mean	39	Fwd Pkts/s	59	Fwd Pkts/b Avg	79	Idle Min
20	Flow IAT Std	40	Bwd Pkts/s	60	Fwd Blk Rate Avg	80	Label

Figure 15: Open-source dataset 80 extracted features with CICFlowMeter 3.0.

CICFlowMeter tool was used to extract the raw data (Pcap files) to a real dataset into a CSV format with the type of features shown in Figures 14 and 15 which is suitable for building algorithms in an ML system. The next section describes the extracted dataset used for this study.

3.3.2.1 Extracted Dataset Description

This research consists of a newly generated dataset and an open-source dataset CSE-CIC-IDS2018 ('IDS 2018' 2022) consisting of the benign and DDoS classes that make up the binary class label which forms up the class attribute. DDoS is an attack traffic label, and benign is a normal traffic class. The benchmark dataset was deployed in this study for the training of large data sizes since limited time is required to complete this research. Therefore, generating a larger sample data size could not be achieved. However, choosing an open-source dataset for this study was difficult due to the lack of DDoS-specific datasets, despite being one of the most severe security attacks in cloud computing.

1. Newly Generated Dataset

This data consists of 28972 label sample sizes of benign and DDoS traffic flow and 84 features. The major tool used to perform DDoS attacks is slowloris on different terminals seen as systems against the system in the Owncloud.

2. CSE-CIC-IDS2018 Dataset

HTTP denial of service: In this case, the major tools used are Slowloris and LOIC, which have been shown to render Web servers fully unreachable with only one attacking machine. The dataset includes numerous attack types, such as DoS, Infiltration, DDoS, and brute force. Only DDoS attacks are taken into consideration for this study.

The dataset provided both normal and attack traffic consisting of 155047 benign and DDoS labels and 80 features. The open source dataset can be accessed using this link (https://www.kaggle.com/datasets/solarmainframe/ids-intrusion-csv and https://www.unb.ca/cic/datasets/ids-2018.html).

Two major attributes of the dataset are Class labels and class distribution

Class labels: Class labels is the target features in the features extracted from the dataset. Each dataset element represents a momentary sample of the network activity. These occurrences are classified based on the type of traffic, such as whether it is malicious or benign. The new and benchmark datasets are encoded to maintain homogeneity in the class labelling scheme. Binary classification designates Normal traffic as Benign and malicious traffic as a DDOS attack. Table 8 depicts the class labels of the datasets.

Table 8: Class labels for the datasets

Label	Scenario
Benign	Normal Traffic
DDoS attack	Malicious Traffic

Class distribution records for the new dataset: Each dataset used in this study is presented in the table with the number of records of each class label.

The main data generated for this research Figure 16 shows the number of records of each class label. The percentage records of the class labels were represented using a pie chart as shown in Figure 17.

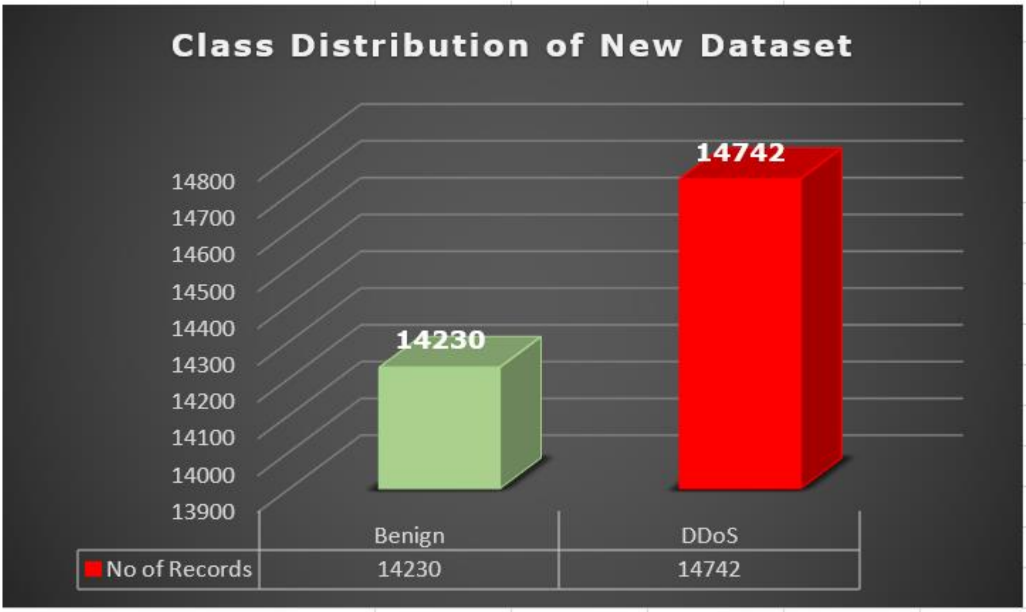

Figure 16: Class distribution of the new dataset.

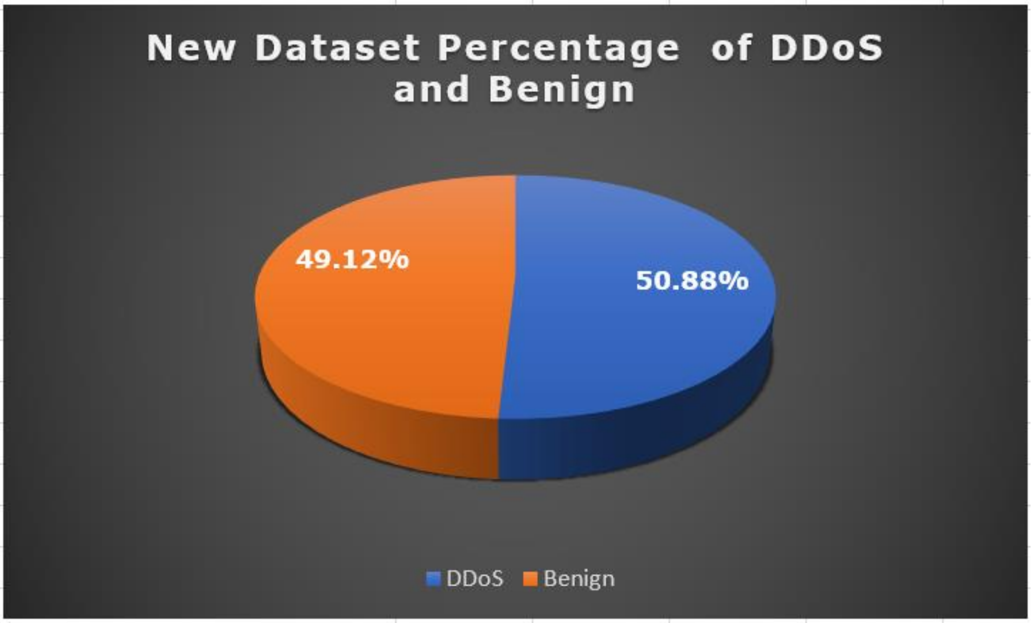

Figure 17: New dataset class labels percentage records.

Class Distribution for CSE-CIC-IDS2018 Dataset

For the open-source dataset that is used for larger sample size training, the records of each class label are shown in Figure 18. The percentage of class label records is displayed in Figure 19.

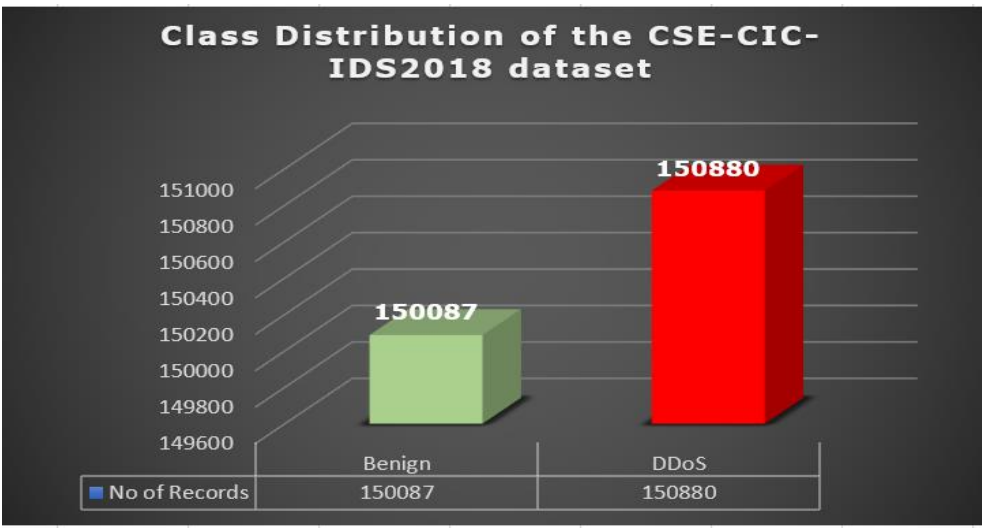

Figure 18: Class distribution for Open-source dataset.

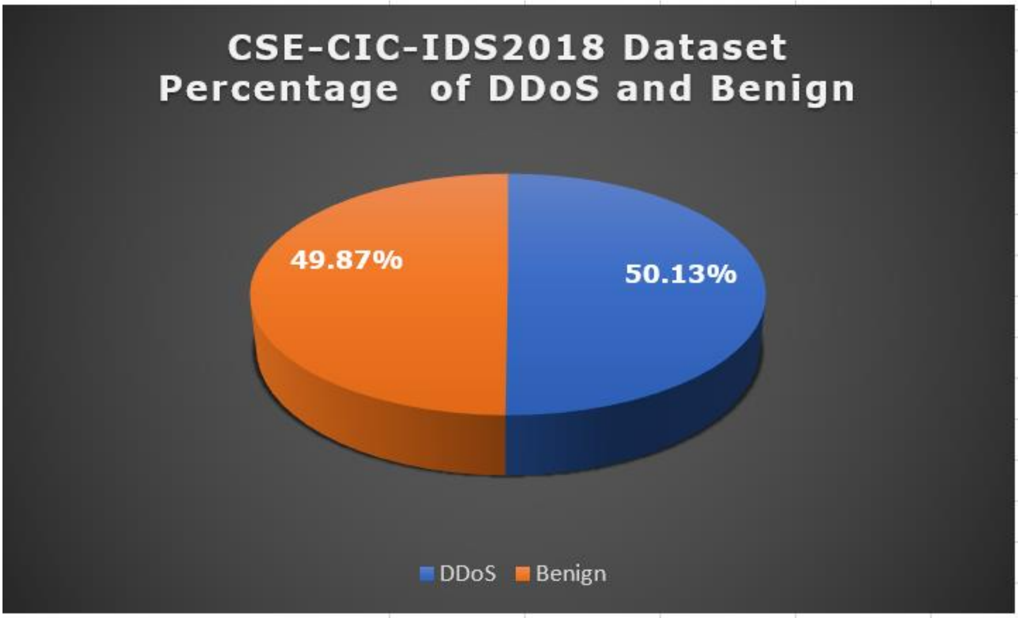

Figure 19: Open-source dataset class labels percentage records.

3.3.3 Feature Selection

This involves choosing necessary features that have a greater impact on the output variable. It implies that we should only choose characteristics (independent variables) that have a strong relationship to the output variable. It is the step that matters the most when building a machine learning model (Swapnilbobe 2021). Since the dataset generated contains lots of features, training the model with those features will impact the accuracy of the model to go down. Therefore, to tackle this issue, only the features that have a greater impact on the dependent variable should be used (the output variable).

3.3.3.1 Correlation Coefficient

The linear relationship between two or more variables is measured using correlation. We can forecast one variable based on another using correlation. Because the desirable variables have a strong correlation with the target, correlation can be used to select features. Variables should also be uncorrelated among themselves while being correlated with the objective. One can estimate one variable from another if the two are correlated. As a result, if two features are correlated, the model only actually requires one of them as the other does not provide any new information (Gupta 2020). The correlation coefficient method was used to detect the features with the highest positive correlation and highest negative correlation. These features were removed to achieve high accuracy.

3.3.3.2 Feature Selection for The New Dataset

Out of 84 features extracted from the new dataset using CICFlowMeter 4.0, 31 features were selected for Model evaluation using correlation coefficient and its heatmap. These features greatly impacted the results achieved in this experiment. Figure 20 illustrates the heatmap of features selected. Also, to remove the duplicate heat map, a masking technique was used as shown in Figure 21. Figure 22 shows the 31 features selected for ML model training, testing and validation.

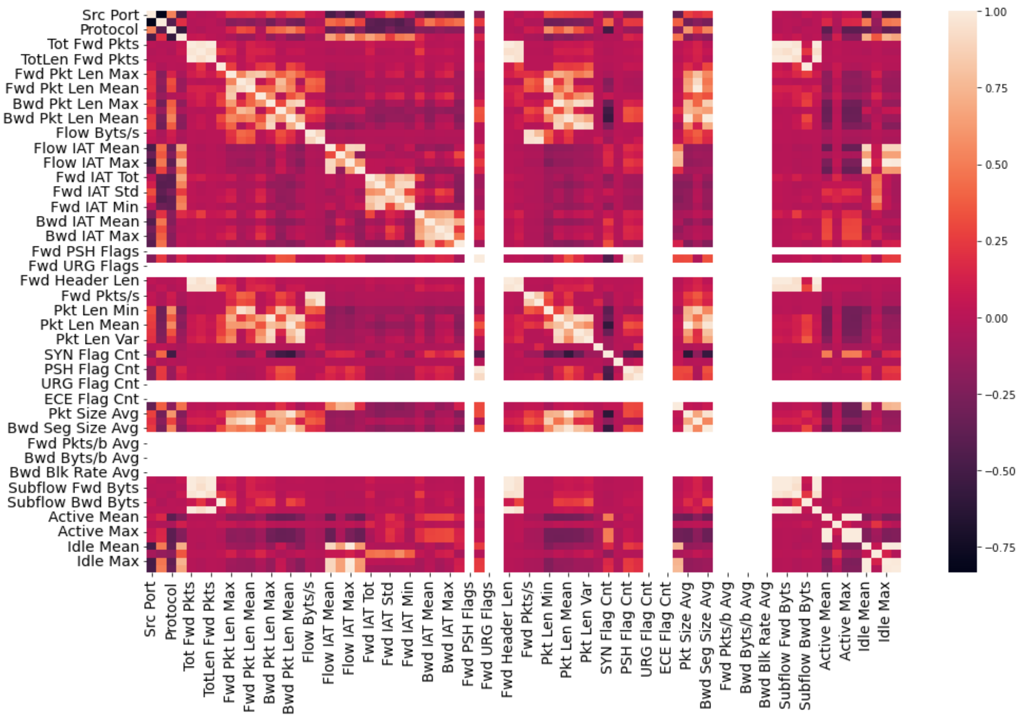

Figure 20: New dataset feature selection heat map.

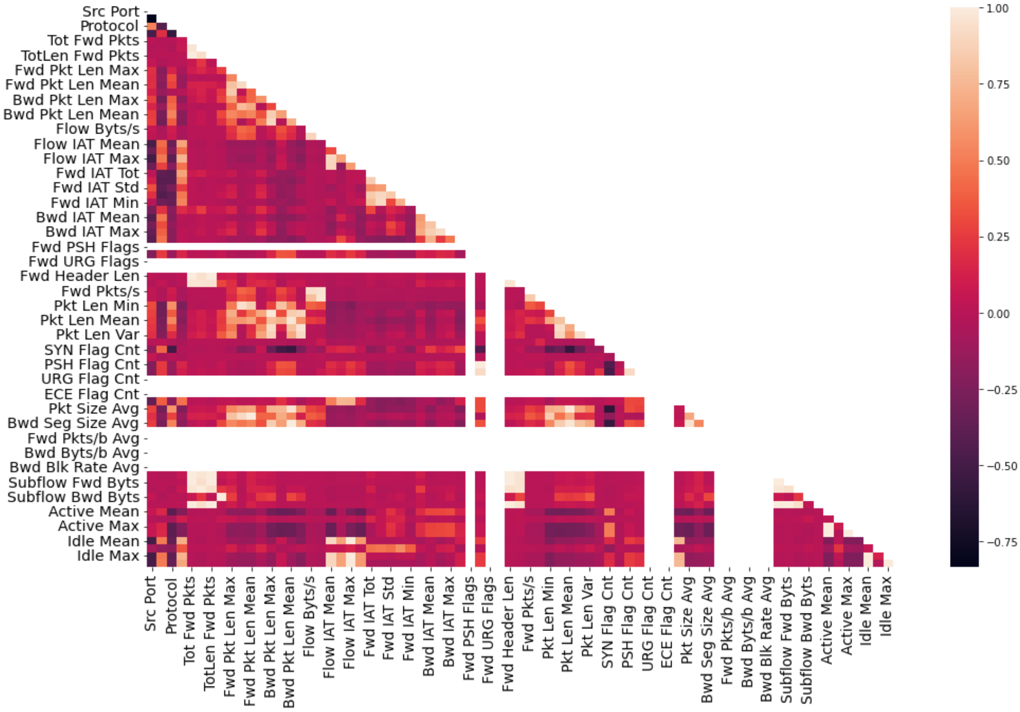

Figure 21: Application of Mask on the heat map.

```
Features = ['Src Port', 'Protocol', 'Tot Fwd Pkts', 'TotLen Fwd Pkts', 'Fwd Pkt Len Max',
            'Fwd Pkt Len Mean', 'Bwd Pkt Len Max', 'Bwd Pkt Len Mean', 'Flow Byts/s', 'Flow IAT Mean',
            'Flow IAT Max', 'Fwd IAT Tot', 'Fwd IAT Std', 'Fwd IAT Min', 'Bwd IAT Mean',
            'Bwd IAT Max', 'Fwd Header Len', 'Fwd Pkts/s', 'Pkt Len Min', 'Pkt Len Mean',
            'Pkt Len Var', 'SYN Flag Cnt', 'PSH Flag Cnt', 'Pkt Size Avg', 'Bwd Seg Size Avg',
            'Subflow Fwd Byts', 'Subflow Bwd Byts', 'Active Mean', 'Active Max', 'Idle Mean', 'Idle Max']
```

Figure 22: 31 Features selected for new dataset ML building.

3.3.3.3 Feature Selection for the CSE-CICIDS2018 Dataset

32 features were selected for model training, testing and validation. Figure 23 and 24 (application of mask) shows the heatmap feature selection. Figure 25 depicts the main feature of the evaluation.

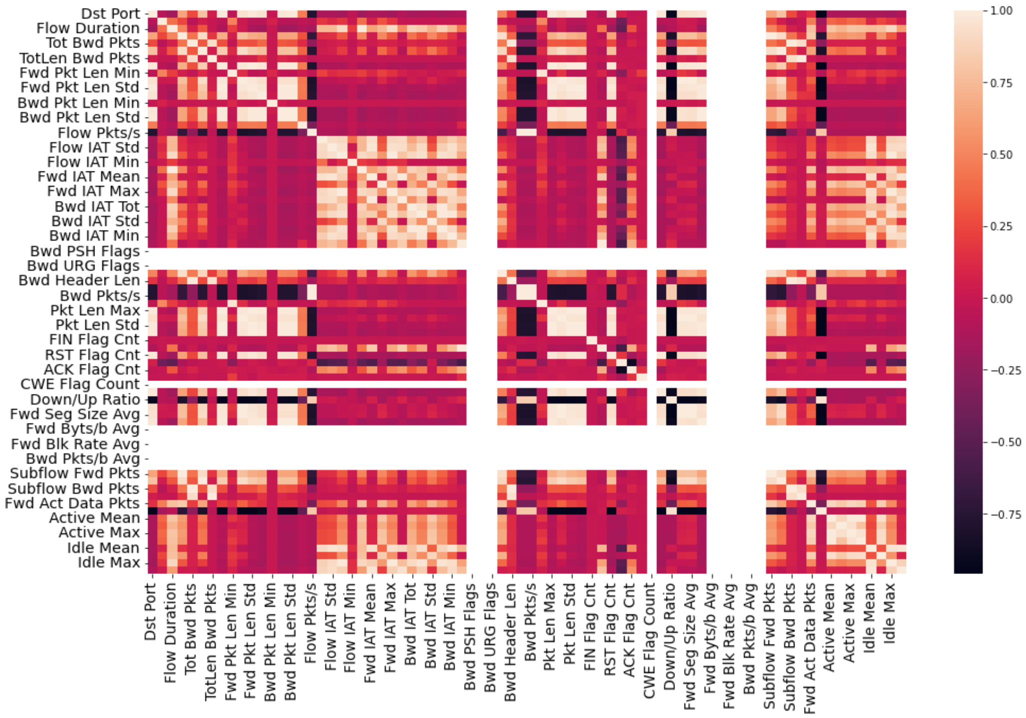

Figure 23: Open-source dataset feature selection heat map.

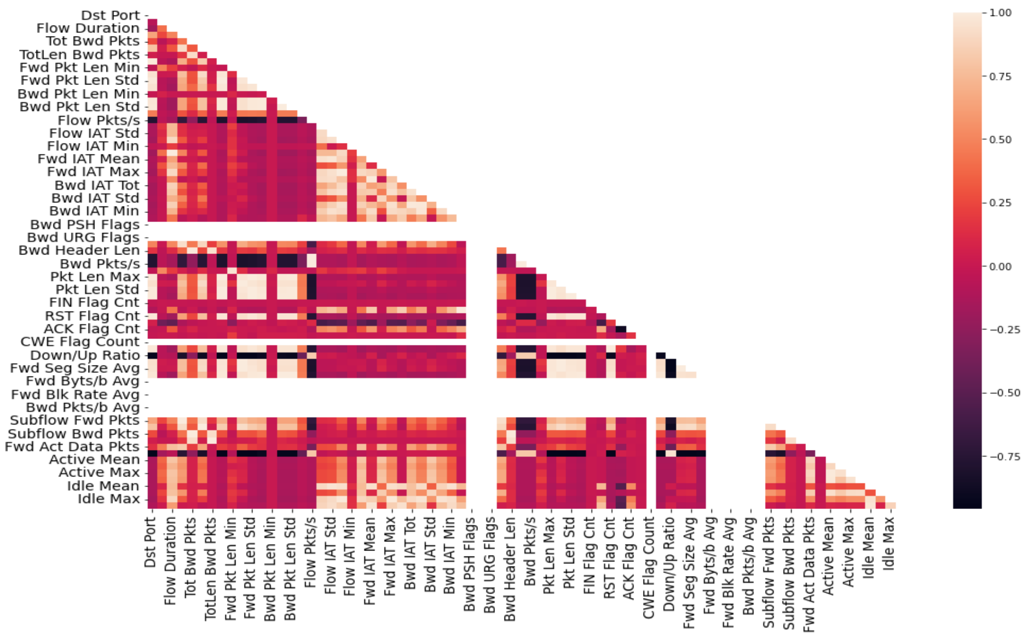

Figure 24: Application of mask on the heat map.

```
Features = ['Dst Port', 'Flow Duration', 'Tot Bwd Pkts', 'TotLen Bwd Pkts', 'Fwd Pkt Len Min',
            'Fwd Pkt Len Std', 'Bwd Pkt Len Min','Bwd Pkt Len Std', 'Flow Pkts/s', 'Flow IAT Std',
            'Flow IAT Min', 'Fwd IAT Mean', 'Fwd IAT Max', 'Bwd IAT Tot', 'Bwd IAT Std',
            'Bwd IAT Min', 'Bwd Header Len', 'Bwd Pkts/s', 'Pkt Len Max', 'Pkt Len Std',
            'FIN Flag Cnt', 'RST Flag Cnt', 'ACK Flag Cnt', 'Down/Up Ratio','Fwd Seg Size Avg',
            'Subflow Fwd Pkts', 'Subflow Bwd Pkts', 'Fwd Act Data Pkts', 'Active Mean', 'Active Max', 'Idle Mean', 'Idle Max']
```

Figure 25: 32 Main features used for ML model building.

3.3.4 Data Pre-processing

This involves transforming unusable raw data into something that can be used. Apply label encoding to the categorical class label to transform it into discrete form (0,1), where 0 represents a class that is benign, and 1 represents a DDoS attack.

3.3.4.1 Data Cleaning and Transformation

Before using the dataset to build ML models, cleaning the data was essential. For instance, checking strings and negative values can disrupt the smooth running of the process. In this section, cleaning, like checking missing data and removing undefined data, was performed to ensure that it was transformed into a smooth processing dataset.

Missing Data: In machine learning, handling missing data is essential since it can cause any model to make inaccurate predictions. Considering this, null values are removed by moving the most recent valid observation up the column axis. The fillna method from the pandas' library is used to do this (Pandas 2022), as seen in the example below.

```
data.fillna(method ='ffill', inplace = True)
```

Undefined data: Data that has null values removed may be undefinable. After propagation, a null field with no cells on its left becomes NaN since there are no cells to supply a value. These values are therefore decoded as 0. The Fillna approach is used for all of this.

```
data=data.fillna(0)
```

However, no missing values and strings were found in the chosen datasets.

Transformation: Modeling may not be possible given the format of the data that was collected. In these situations, data and data types must be changed before being fed into the models according to the CRISP-DM technique. Since models do not perform well with strings or do not work at all, several data features were converted into numeric or float.

3.3.5 Dataset Splitting

The datasets were divided into two subsets: training and testing also used for validation. For ML model building, this split was done in an 80:20 ratio, respectively. Where 80% is for training and 20% for testing and validation. The split is performed using the train_test_split helper method from the scikit-learn library. This method divides training into two stages: the training set and the validation set. First, the model is trained using the training set. The model's performance on data that it has not been trained on is then estimated using the validation set. A stratified 5-fold technique was used for validation in the context of this study.

3.3.6 Modelling

The building of the learning model and the creation of the forecasted labels are the two components of the classification phase. Scikit-learn, a Python toolkit for data mining, data analysis, and machine learning, is used to carry out these tasks.

3.3.6.1 Selection of Models

Five models were selected namely Random Forest, Support Vector Machine, Naïve Bayes, Decision Tree and K-Nearest Neighbours for training and validation of the datasets (New and open datasets) used in this experiment. These models were selected based on their performance in intrusion detection.

3.3.7 Training

The chosen algorithms were supplied with training data during the training phase so they can be used to develop machine learning models. The training set is utilised for training.

The target attribute was present in the input data source at this stage of the procedure (class label). To map the input characteristics to the target attribute, patterns must be found in the training set. A model is created based on the patterns that were observed. New DDoS datasets and an open-source dataset were used in this study as the input data source, and the target attribute is the nature of network traffic—either an attack or normal traffic—DDoS attack or Benign. The open dataset (CSE-CICIDS2018 DDoS dataset) was deployed and trained since a large sample data size ranging from 100000 to 300000 could not be generated due to the limited time assigned in this research. The data size was extracted from the open-source dataset for training and the results were also compared using the new data generated.

On each of the datasets provided, five algorithms are trained. A variety of techniques from the scikit-learn library are used during training. Table 9 shows the method used for each model. Figure 26 shows the Sample code used for training the dataset.

Table 9: Scikit-learn Python library classifiers used for building models.

Model	**Scikit-learn Methods & Classifiers**
Random Forest	sklearn.ensemble.RandomForestClassifier
SVM	sklearn.svm.LinearSVC
Naive Bayes	sklearn.naive_bayes.GaussianNB
K-NN	sklearn.neighbor.KNeighborsClassifier
Decision Tree	sklearn.tree.DecisionTreeClassifier

```
In [155]: #Percentage split 80% train, testing 20%
          from sklearn.model_selection import train_test_split
          x_train, x_test, y_train, y_test =train_test_split(x,y,train_size=0.8,test_size=0.2,random_state=42)

In [156]: RF = RandomForestClassifier()
          classifiers1 = ["RF"]
          scores = []

In [157]: import time
          s=time.time()
          res1 = time.time()
          RF.fit(x_train,y_train)
          y_pred = RF.predict(x_test)
          score = accuracy_score(y_test, y_pred)*100
          scores.append(score)
          res2 = time.time()
          print('Random Forest took ',res2-res1,'seconds')
          print("Accuracy of RF ", score )
          conf_matrix = confusion_matrix(y_test,y_pred)
          report = classification_report(y_test,y_pred)
          print("Confusion Matrix:\n",conf_matrix)
          print("Report:\n","class",report)
```

Figure 26: Sample python code for ML model training (Wan n.d. 2019).

3.3.8 Validation

5-fold cross-validation was used to validate the model after training. To evaluate if the model performs were fit for the dataset, cross-validation was used. This approach lessened the overfitting mistakes that happen when a model is too closely fitted to a variety of data samples. Cross-validation is carried out in iterations, with each iteration requiring the division of the dataset into k subsets, or folds. As shown in Figure 27, the model is trained on k-1 folds while the other fold is saved for testing. Up until every fold has functioned as a test fold, this process was repeated. The process is finished by calculating the average value, which was then used to summarise the evaluation metric.

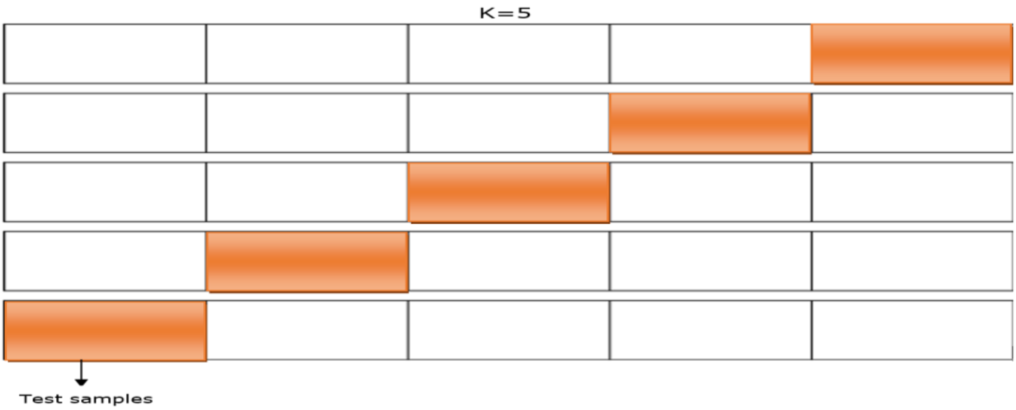

Figure 27: 5-Fold cross-validation

A stratified k-fold technique is employed in this study with the validation dataset (20% of the whole set). A variant of k-fold cross-validation known as stratified k-fold makes sure that the distribution of classes is uniform across all folds. Utilizing the StratifiedKFold function with k=5 from the scikit-learn library was implemented. Figure 28 shows the sample code for ML model validation.

```
#Cross Validation
from sklearn.model_selection import cross_val_score,KFold
from sklearn.model_selection import cross_val_score,StratifiedKFold
```

```
import time
s=time.time()
res1 = time.time()
RF=RandomForestClassifier()
stratifiedkf=StratifiedKFold(n_splits=5)
score=cross_val_score(RF,x,y,cv=stratifiedkf)
res2 = time.time()
print('RF took ',res2-res1,'seconds')
print("Cross Validation Scores are {}".format(score))
print("Average Cross Validation score :{}".format(score.mean()*100))
```

Figure 28: Sample python code for ML model validation (Wan n.d. 2019).

3.3.9 Testing

The models were tested with imaginary data at the end of the modelling phase. The test set those results from the data split is the unseen data used at this point (20%). Testing is done to evaluate a model's ability to represent data and how well it will function in the future. This study made sure that any model modifications were made before testing, ensuring that testing data would only be utilised once. Accuracy, precision, recall, and

F-measure are the main performance metrics that were created to analyse the performance of the DDoS datasets. Although computation time was included.

3.3.10 Evaluation

Creating performance metrics is an essential step in analysing the effectiveness of the algorithm. Several metrics are generated for this study including:

Accuracy

The number of cases that a classification model correctly and wrongly classifies is one approach to quantifying its performance. A confusion matrix is a popular representation of these values. A tabulated visual representation of the effectiveness of supervised learning systems is called a confusion matrix. Instance counts in actual classes are represented by rows, while instance counts in predictive classes are represented by columns where 0 is Benign traffic and 1 is attack traffic. The confusion matrix for a binary classification task is shown in Figure 29.

		Predicted Class	
		0	1
Actual Class	0	1000	100
	1	200	50

Figure 29: Illustration of confusion matric for a binary classifier.

The efficiency of a single classifier can be determined using a confusion matrix. However, combining the matrix's components into a single value is more useful and understandable.

In this work, the accuracy measure is used to summarise the matrix and is calculated as follows:

Accuracy = properly *labelled instances/Total instances x100%*

Equation 1: Computation of accuracy

Precision

Accuracy alone usually is insufficient to judge the efficiency of a learning algorithm. Even while accuracy shows whether the model is being trained properly, it does not provide precise information about the application. As a result, other performance criteria, like precision, are used. The proportion of accurately categorised positives, or true positives, is referred to as precision. There are numerous circumstances in which false positives could have an impact. In the context of this study, a high false positive rate suggests that legitimate traffic may be mistakenly classified as harmful. Beyond academics, this may lead to a waste of time and resources.

Precision is calculated as follows:

Precision = *True positives/True positives x False positives*

Equation 2: Computation of precision

Recall

Recall is an additional performance measure. How many of the real positives were discovered or identified is measured by recall. It is also a very crucial statistic because undiscovered positives or false negatives may have negative impacts in specific circumstances. For example, an algorithm that does not account for all DDoS attack instances means that malicious network traffic will go undetected, increasing the possibility of disruption to the system and the users. Recall is calculated as:

Recall = *True positives/True positives x False negatives*

Equation 3: Computation of recall

F1- Score/Measure

The F-measure is a metric that combines precision and recall giving a total accuracy score of an algorithm. An algorithm that successfully detects threats while producing a few false alarms has a high F-measure score because it has low false positives and false negatives. F1-score is calculated as follows:

F1-score = *2 x (Precision x Recall/ Precision + Recall)*

Equation 4: Computation of F1-score

Computation Time

Processing time is the last performance metric used in this evaluation. This indicates the length of time a model needs to train rather than being directly tied to classification. This measure provides information about the efficiency of the algorithm being computed within a time frame. The reported processing time was based on a windows 10 operating system with 34GB RAM and an i7 processor.