MACHINE LEARNING

DDOS Attack Detection

SUBMITTED TO:
Dr Sangeetha Viswanathan

SUBMITTED BY:

RAMYA AJAY(CB.EN.P2CYS22004)

Abstract: Cyberattacks can trigger power outages, military equipment problems, and breaches of confidential information, i.e., medical records could be stolen if they get into the wrong hands. Due to the great monetary worth of the data it holds, the banking industry is particularly at risk. As the number of digital footprints of banks grows, so does the attack surface that hackers can exploit. In this project, I will be detecting distributed denial-of-service (DDOS) attacks using a classification model for the prediction of the attack.

I. Introduction

DoS attacks cause denial of service to legitimate requests by exhausting the resources of network and services. To maximize the impact, the attack will be launched from distributed sources, called distributed denial of service attack. In the majority of the cases, these attacks are launched by botnets. The largest DDoS attack on the latest records happened on Feb 2018 as revealed by git hub.

In this project we process the data, exploring the variables relationship between the attributes and we model the data using the classification model: Logistic Regression .Then data loading,data preparation, train-test split and data visualization are covered.

II. Dataset

For this task, I have collected a dataset from Kaggle, which contains historical information about the extracted DDOS flows and and it is combined with "Benign " flows which are extracted separately from the same base dataset and made into a single largest dataset.

Balanced Dataset(dataset_sdn.csv):
--> Has total datapoints of 104345(ddos + benign)
--> important features are

```
'src',
'pktcount',
'dst',
'byteperflow',
'pktperflow',
'pktrate',
'tot_kbps',
```

```
    'rx_kbps',
    'flows',
    'bytecount',
    'dt',
    'Protocol',
    'dur',
    'tot_dur'
```

III. Methodology

1) Data Collection:

The DDOS Dataset is selected from Kaggle repository. We developed a machine learning algorithm that classify transactions into 2 classes (Benings or Malicious) using the attributes and achieve maximum accuracy as possible through learning.

2) Data Pre-processing:

The dataset has been cleaned and it is standard quality before the module analysis is proceeded. Data set often contains missing values and extreme values called outliers and these values can affect out test and even sometimes it even can cause module to fail. It is better to remove all outliers and fill the missing values with near values. In our dataset, we do not have any missing values or any sort of outliers.

```
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
label           0
dtype: int64
```

Splitting of dataset

Training and Testing

X - Data frame containing input data

Y - Output data/result which must be predicated

In this, we have divided the dataset into training set 70% and testing set 30%

```
def __init__(self, data):
        self.data = data
        X = preprocessing.StandardScaler().fit(self.data).transform(self.data)
        self.X_train, self.X_test, self.y_train, self.y_test = train_test_split(X, y,
random_state=42, test_size=0.3)
```

3) Data Modelling (Classification)

The project is carried on using the classifier model, Logistic regression. This is to make prediction module obtained during the training process.

Logistic Regression:

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.

Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**.

4) Model Evaluation

- **Recall:** Out of all the positive classes, how much we predicted correctly. It should be high as possible.

$$\text{TPR /Recall / Sensitivity} = \frac{TP}{TP + FN}$$

• **Precision:** Out of all the positive classes we have predicted correctly, how many are actually positive.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive} = \frac{True\ Positive}{Total\ Predicted\ Positive}$$

• **Accuracy:** Out of all the classes, how much we predicted correctly. It should be high as possible.

• **F1-Score:** It is difficult to compare two models with low precision and high recall or vice versa. So, to make them comparable, we use F-Score. F-score helps to measure Recall and Precision at the same time. It uses Harmonic Mean in place of Arithmetic Mean by punishing the extreme values more.

$$F1 = 2 \times \frac{Precision \cdot Recall}{Precision + Recall}$$

IV. Python Code and Results:

- Importing all the required python libraries.

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
from matplotlib.pyplot import figure

from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn import preprocessing

from sklearn.model_selection import GridSearchCV
import time
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
```

- Data Analysis

```
data.info()
```

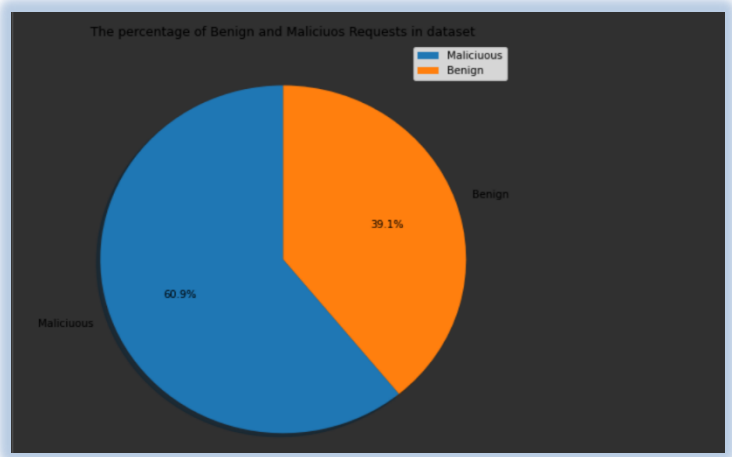
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 104345 entries, 0 to 104344
Data columns (total 23 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   dt           104345 non-null  int64
 1   switch       104345 non-null  int64
 2   src          104345 non-null  object
 3   dst          104345 non-null  object
 4   pktcount     104345 non-null  int64
 5   bytecount    104345 non-null  int64
 6   dur          104345 non-null  int64
 7   dur_nsec     104345 non-null  int64
 8   tot_dur      104345 non-null  float64
 9   flows        104345 non-null  int64
```

```
 10  packetins     104345 non-null  int64
 11  pktperflow    104345 non-null  int64
 12  byteperflow   104345 non-null  int64
 13  pktrate       104345 non-null  int64
 14  Pairflow      104345 non-null  int64
 15  Protocol      104345 non-null  object
 16  port_no       104345 non-null  int64
 17  tx_bytes      104345 non-null  int64
 18  rx_bytes      104345 non-null  int64
 19  tx_kbps       104345 non-null  int64
 20  rx_kbps       103839 non-null  float64
 21  tot_kbps      103839 non-null  float64
 22  label         104345 non-null  int64
dtypes: float64(3), int64(17), object(3)
```

Here we see that the label contains boolean values: 0 - Benign, 1-Maliciuous

```
data.label.unique()
array([0, 1])
```

```
labels = ["Maliciuous",'Benign']
sizes = [dict(data.label.value_counts())[0], dict(data.label.value_counts())[1]]
plt.figure(figsize = (13,8))
plt.pie(sizes, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
plt.legend(["Maliciuous", "Benign"])
plt.title('The percentage of Benign and Maliciuos Requests in dataset')
plt.show()
```



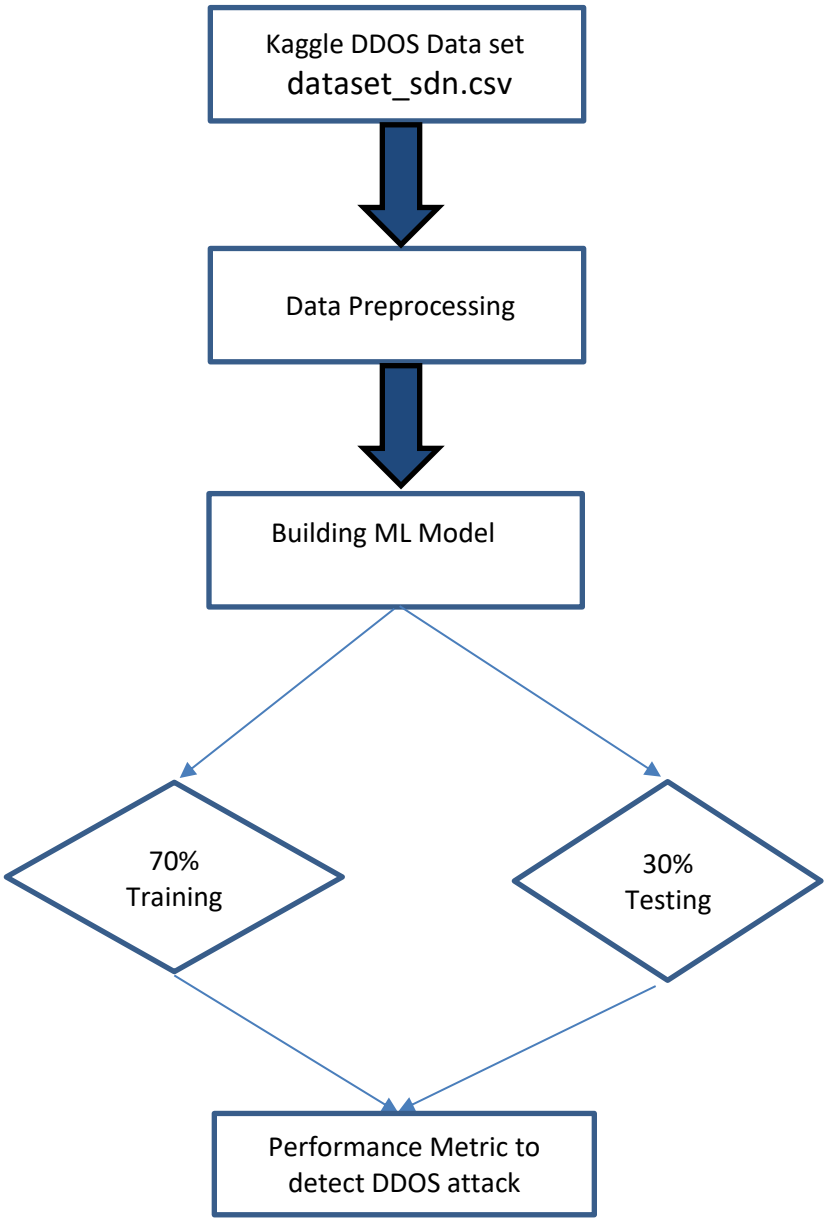
- Training a machine learning model using logistic regression and defining Evaluation metrics function for each classifier

```
class Model:
    global y
    def __init__(self, data):
        self.data = data
        X = preprocessing.StandardScaler().fit(self.data).transform(self.data)
        self.X_train, self.X_test, self.y_train, self.y_test = train_test_split(X, y,
random_state=42, test_size=0.3)

    def LogisticRegression(self):
        solvers = ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']

        start_time = time.time()
        results_lr = []
        accuracy_list = []
        for solver in solvers:
            LR = LogisticRegression(C=0.03, solver=solver).fit(self.X_train, self.y_tr
ain)
            predicted_lr = LR.predict(self.X_test)
            accuracy_lr = accuracy_score(self.y_test, predicted_lr)
            #print("Accuracy: %.2f%%" % (accuracy_lr * 100.0))
            #print('######################################################################')
            results_lr.append({'solver' : solver, 'accuracy': str(round(accuracy_lr *
100, 2)) + "%",
                               'Coefficients': {'W' : LR.coef_, 'b': LR.intercept_}
})

            accuracy_list.append(accuracy_lr)

        solver_name = solvers[accuracy_list.index(max(accuracy_list))]
        LR = LogisticRegression(C=0.03, solver=solver_name).fit(self.X_train,self.y_tr
ain)
        predicted_lr = LR.predict(self.X_test)
        accuracy_lr = accuracy_score(self.y_test, predicted_lr)
        print("Accuracy: %.2f%%" % (accuracy_lr * 100.0), '\n')
        print("#########################################################################
#")
        print('Best solver is : ', solver_name)
        print("#########################################################################
#")
        print(classification_report(predicted_lr, self.y_test), '\n')
        print("#########################################################################
#")
        print("--- %s seconds --- time for LogisticRegression" % (time.time() -
 start_time))
```

- Printing function run time and accuracy,precision,recall and F1 score for logistic regression.

```
M.LogisticRegression()

Accuracy: 76.64%

##########################################################################
Best solver is :  liblinear
##########################################################################
              precision    recall  f1-score   support

           0       0.84      0.79      0.81     20024
           1       0.66      0.72      0.69     11128

    accuracy                           0.77     31152
   macro avg       0.75      0.76      0.75     31152
weighted avg       0.77      0.77      0.77     31152


##########################################################################
--- 5.511197566986084 seconds --- time for LogisticRegression
```

V.Flow Diagram



VI.Conclusion

The logistic regression has exactly accuracy of 76.64%.

In future, research can be use more refine technique to give more accuracy and compare with different classifier.