Report for Study
on

Botnet Attack Detection on IOT Devices

Submitted by

Nabin Kumar Adhikari
Registration No.: 11617707
Section: KEM39
Roll No: B42

School of Computer Science & Engineering
Lovely Professional University,
Phagwara, India

Table of Contents

INTRODUCTION

In the previous years, security has become a serious issue in the online world. Everything that is connected to the internet has a possibility to become prone towards various kinds of attacks. One of the attacks hosted online is Botnet Attack. It is a type of attack where the attacker uses bots to get into the system of a victim. A Bot is a piece of application software that runs automated tasks or scripts over the internet. These applications perform tasks that are simple and repetitive but on much higher rate than any human. A Botnet is a collection of number of Internet-connected devices, with each one running one or more bots. This can be used to perform attacks like DDoS (Distributed Denial-of-service) attack, steal data, access to the device and its connection or send spam. The attacker can have control over the botnet using Command and Control Software. Let alone the risks of losing or stealing any data, the sheer use of botnet attack can cause harm to the victim devices. So, preventing such type of attack is not only important to various internet organizations but also to individuals. Often the organizations or Internet Service Providers that perform large amount of internet-related works have security to prevent these types of attacks. The detection of these types of attacks can be performed by studying the characteristics of the internet traffic and internet-related tasks a device in inspection performs. For this project, we will be using the IOT Botnet Detection dataset provided by UCI Machine Learning Repository and also the dataset provided by Malware Capture Facility of CTU (Czech Technical University) University, Czech Republic, named as CTU-13 dataset.
The Botnet attack dataset usually contains data of traffic of the device in inspection. For the dataset from UCI, it addresses the dataset for botnet attacks gathered from 9 commercial IoT devices authentically infected by Mirai and BASHLITE botnets. As for the dataset from CTU-13, it contains data of botnet traffic that was captured in CTU in 2011 with the traffic mixed of normal traffic, background traffic and real botnet traffic.

The data can be found for study or download on the following links:

UCI Data : https://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT

CTU-13 : https://mcfp.weebly.com/the-ctu-13-dataset-a-labeled-dataset-with-botnet-normal-and-background-traffic.html

This whole project can be found on the following GitHub link:

https://github.com/NabinAdhikari674/Detection-of-IOT-Botnet-Attacks

ABOUT THE DATA

UCI Dataset:

The provided data folder has various folders for traffic data of 9 different IoT commercial devices. Each data folder of the IoT device has:

a. Benign (safe) traffic data in comma-separated values format with '.csv' extension.
b. Mirai (Botnet) attack data in compressed format with '.rar' extension, inside of which, data of different types of attacks are given in csv format.
c. Gafgyt (Botnet), also known as BASHLITE attack, data in compressed format with '.rar' extension, inside of which, data for different types of attacks are given in csv format.

The data set is a Multivariate, Sequential type of data with about 7062606 instances and 115 real attributes. The full data however is not used, but selected instances of selected IoT device is used for simplicity while preparing the model for training and testing. Nevertheless, the model can also be used with all the instances though that may lead to more time consumption due to the huge amount data to be processed. The associated tasks related to this data are Classification or Clustering.
The attributes of the data with their particular information are given below:

Stream Aggregation
H: Stats summarizing the recent traffic from this packet's host (IP)
HH: Stats summarizing the recent traffic going from this packet's host (IP) to the packet's destination host.
HpHp: Stats summarizing the recent traffic going from this packet's host and port (IP) to the packet's destination host and port.
HH_jit: Stats summarizing the jitter of the traffic going from this packet's host (IP) to the packet's destination host.

Time-frame (The decay factor Lambda used in the damped window)
Recent history of the stream is capture in these statistics:
L5, L3, L1, ...

The statistics extracted from the packet stream
weight: The weight of the stream (can be viewed as the number of items observed in recent history)
mean: The mean of the stream.
std: Standard Deviation of the stream.
radius: The root squared sum of the two streams' variances
magnitude: The root squared sum of the two streams' means
cov: an approximated covariance between two streams
pcc: an approximated covariance between two streams

CTU-13 Dataset:

This dataset of botnet traffic was captured with goal to have large capture of real botnet traffic mixed with normal and background traffic. This dataset contains 13 captures of different botnet samples. The provided data folder is highly compressed in '.tar' format and again in '.bz2' format. After extraction, we have 13 different scenarios of data i.e. 13 folders. Each folder has 3 or 4 files defined as follows:

a. Pcap file of the botnet capture in '. pcap' format
b. NetFlow file of the traffic capture in '. binetflow' format
c. A Readme file with information of the files
d. The original executable application (May not be present in some data folders) in '.exe' format

The data set is a Multivariate, Sequential type of data with extremely large number of instances and about 15 attributes. The full data however is not used, but selected instances of selected data folder is used for simplicity while preparing the model for training and testing. Nevertheless, the model can also be used with all the instances though that may lead to more time consumption due to the huge amount data to be processed. The associated tasks related to this data are Classification or Clustering.

The attributes of the data with their particular information are given below:

StartTime: Start time of the capture of traffic data
Dur: Duration of the capture of traffic data or, duration of attack done on devices
Proto: Protocol used as found in the traffic
SrcAddr: Source IP Address from where the traffic
Sport: Source Port Address
Dir: Direction of the dataflow and attack
DstAddr: Destination IP Address
Dport: Destination Port Address
State: State during the capture
sTos: Source Type of Service
dTos: Destination Type of Service
TotPkts: Total Packets transferred or received during the capture
TotBytes: Total size of packets transferred or received during the capture in Bytes
SrcBytes: Size of packers from Source
Label: Label of the attack (if it is successful botnet attack or background or normal)

PREPARING AND CLEANING THE DATASET

UCI Dataset:

This dataset had two files to process, one with benign traffic and another with botnet attack traffic. After importing both files in a DataFrame, a new column ‘Out’ was added. For benign DataFrame, ‘Out’ was filled with value 0 while, for botnet attack DataFrame, ‘Out’ filled with value 1. After this. Both DataFrames were concatenated and were shuffled using the ‘shuffle’ function provided in ‘sklearn.utils’ package. Now, the ‘Out’ column was saved in a new DataFrame named ‘Output’ which was used as output to the input which is the concatenated DataFrame with ‘Out’ column dropped.
The data has undergone some serious cleaning steps since the data comparison between the benign traffic and botnet attack traffic showed pretty obvious outliers and overfitting type of data. In fact, in the provided dataset the columns after number 28 are completely removed due to the above reason. While including the above removed columns to the training dataset and proceeding, we have the case of overfitting of the data.

Following is the code snippet for preparation and cleaning of the data:
(found in ‘pre.py’ in model using UCI Dataset)

```
#Reading data from file
cdata=pd.read_csv("benign_traffic.csv")
scan=pd.read_csv("udpplain.csv")

#Tagging Clean Traffic as 0 and Threat Traffic as 1
scan['Out']=1
cdata['Out']=0

#combining two DataFrames and shuffling using shuffle() from sklearn.utils
comb=pd.concat([cdata,scan],axis=0)
comb=shuffle(comb)

#Defining Output and dropping ‘Out’ from concatenated DataFrame
Output=comb['Out']
comb=comb.drop(['Out'],axis=1)

# Removing all the Outlier Columns
comb=comb.iloc[:,:28]

Output=np.array(Output).flatten()
```

CTU-13 Dataset:

This data had to be handled carefully and had lot of preparations before actually using it. This data had very large set of files to choose from. So, a method to choose the files using File Dialog was implemented. This was done using 'tkinter' package.

The code snippet is given below:
(can be found in 'dataprep.py' in Model using CTU Dataset)

```
try:
    print("Importing Data from File...##",end=" ")
    data=pd.read_csv('flowdata.csv')
    flag=2
    print("Read DONE.\n")
except FileNotFoundError:
    print("\n\n\t\tFile(flowdata.csv) not Found!!!")
    def choser():
        global data;global flag
        choose=str(input("\n\t\tOpen File Picker to Choose file to open?\nEnter [y/n]:   "))
        if choose=='y':
            from tkinter import Tk
            from tkinter.filedialog import askopenfilename
            root=Tk()
            ftypes = [('BinetFlow FIle',"*.binetflow"),('CSV File',"*.csv"),('Excel
           File',"*.xlsx"),('All Types',"*.*")]
            ttl  = "FilePicker"
            filename = askopenfilename(filetypes = ftypes,title = ttl)
            root.withdraw()
            print ("This is Chosen FilePath : ",filename)
            print("\tReading Data from Choosen File...##",end=" ")
            data=pd.read_csv(filename)
            flag=1
            print("Read DONE.\n")
        elif choose=='n':
            print("\n\t\tThere is no File to process")
            print("\n\t\Exiting...")
            exit()
        else:
            print("Wrong Choice...TRY AGAIN\n")
            choser()
    choser()
    print("Chooser DONE")
finally:
    print("Data Ready to Process")
```

After the data was loaded using the pandas library using ‘read_csv()’, it had to be labelled. To do this a function named ‘labeler()” was used in which the ‘Label’ column of the data was checked row-by-row to replace the strings with actual labels of 0 and 1 or even 0,1 and 2.

The code snippet of this function is given below :
(can be found in ‘dataprep.py’ in Model using CTU Dataset)

```
def labeler():
    global bots; j=0
    bots=0
    i=0
    global data
    print("Running Loop for Normal/Background(value=0) or, Botnet Attack(value=1)
    Label Creation...##\n",end=" ")
    for each in data.loc[:,('Label')]:
        if "Background" in each:
            data.loc[i,('Label')]=0
        elif "Normal" in each:
            data.loc[i,('Label')]=1
        elif "Botnet" in each:
            data.loc[i,('Label')]=2
            bots+=1
        i+=1
        if i==row:
            break
       #added a small code to mimic a loading screen
        j+=0.005
        if int(j)==int(np.log(row)):
            print("#",end="")
            j=-1
    print("Labeler LOOP DONE.\nBot Count : ",bots)
```

After the labeling was done, since this data also had some unwanted attributes, those were also removed. Some attributes like IP addresses cannot be used as it is but is needed to extract some information which can be later used in the training process. However, in this project, this may only increase the complexity, time and memory consumed. So, removing these columns is a better choice. Similarly, many other columns were also removed from the data then only the data was ready to be further processed.

After preparing and cleaning the data, and dividing it to Input and Output, it was standardized using either Standard Scaler of SkLearn or, applying the formula for same. The data was also normalized instead of standardization, if required or shown better results.

TRAIN TEST SPLIT

The train test split can be done either using the module called ‘train_test_split’ from ‘sklearn.model_selection’ or, by using the splitter made during this project. The function is also called as ‘train_test_split1’ which takes the original data and the test size as input. This returns test and train split as inputs and target for the model training.

The code snippet of this function from the actual project is given below :

```
def train_test_split(data,percent):
    print("\nRunning Train_Test_Split...\n")
    global row;                                    //int with row number in the input data
    test_size= int((percent/100)*row)
    train_size=row-test_size
    print("   The size of Training Data is : ",train_size)
    print("   The size of Test Data is     : ",test_size,)
    print("                    Total : ",(train_size+test_size))
    length=0
    rows_t=[]
    rows_s=[]
    print("\n\tRandomly Choosing Training Data...\n")
    train=pd.DataFrame(rows_t,columns=names)
    while(length<train_size):
        x1=np.random.choice(data['Sn'])

        if (((x1==train.Sn).any())==False):
            #print("DATA : ",data.iloc[x1-1:x1,:])
            dict1=(data.iloc[x1-1:x1,:]).to_dict(orient='dict')
            rows_t.append(dict1)########
            length=length+1
            train=pd.DataFrame(rows_t,columns=names)
            #print(np.shape(train))
        else:
            print("",end="")
    length=0
    test=pd.DataFrame(rows_s,columns=names)
    print("\tRandomly Choosing Test Data...\n")
    while(length<test_size):
        x1=np.random.choice(data['Sn'])
        if (((x1==train.Sn).any())==False) and (((x1==test.Sn).any())==False):
            dict2=(data.iloc[x1-1:x1,:]).to_dict(orient='dict')
            rows_s.append(dict2)
```

```
        length=length+1
        test=pd.DataFrame(rows_s,columns=names)
    print("\t\tThe Training and Test Data ARE Split")
    return train,test
```

TRAINING AND TESTING

Training of both datasets from different sources was done using same methods. Each model using either dataset (UCI or CTU-13), was trained and tested using at least 4 models or even 5 models. The training includes training with models such as Decision Tree, Logistic Regression, Perceptron, Naïve Bayes and K-Nearest Neighbors.

For both models using either dataset (UCI or CTU-13), following is the code snippet used for training and testing:
(can be found on 'main.py' of both Models)

```
#Training MODEL 1
print("\n\tRunning Logistic Regression . . . \n")
from sklearn.linear_model import LogisticRegression
model1=LogisticRegression(solver='lbfgs',max_iter=10000)
model1.fit(Xtrain,Ytrain)
prediction1=model1.predict(Xtest)
print(model1.score(Xtest,Ytest)*100)
print("Accuracy Score MODEL 1 is : ",accuracy_score(prediction1,Ytest))
```

```
#Training MODEL 2
print("\n\tRunning Preceptron . . . \n")
from sklearn.linear_model import Perceptron
model2=Perceptron(eta0=0.2,max_iter=1000,tol=1e-3,verbose=0,early_stopping = True
,validation_fraction=0.1)
model2.fit(Xtrain,Ytrain)
prediction2=model2.predict(Xtest)
print(model2.score(Xtest,Ytest)*100)
print("Accuracy Score MODEL 2 is : ",accuracy_score(prediction2,Ytest))
```

```
#Training MODEL 3
print("\n\tRunning DecisionTreeRegressor . . . \n")
from sklearn.tree import DecisionTreeRegressor
model3=DecisionTreeRegressor()
model3.fit(Xtrain,Ytrain)
prediction3=model3.predict(Xtest)
print("Number of mislabeled points out of a total %d points : %d"%(Xtrain.shape[0],(prediction3
!= Ytest).sum()))
print("Accuracy Score MODEL 3 is : ",accuracy_score(prediction3,Ytest))


#Training MODEL 4
print("\n\tRunning Gaussian Naive Bayes . . . \n")
from sklearn.naive_bayes import GaussianNB
model4=GaussianNB()
model4.fit(Xtrain,Ytrain)
prediction4=model4.predict(Xtest)
#print(model4.score(Xtest,Ytest)*100)
#print("Accuracy Score MODEL 4 is : ",accuracy_score(prediction4,Ytest))
print("Number of mislabeled points out of a total %d points : %d"%(Xtrain.shape[0],(prediction4
!= Ytest).sum()))


#Training MODEL 5
print("\n\tRunning KNN Classifier . . .\n")
from sklearn.neighbors import KNeighborsClassifier
model5=KNeighborsClassifier()
model5.fit(Xtrain,Ytrain)
prediction5=model5.predict(Xtest)
print(model5.score(Xtest,Ytest)*100)
print("Accuracy Score MODEL 4 is : ",accuracy_score(prediction5,Ytest))
```

Even neural model was designed but was not tested nor completed once completely due to the sheer time and resource consumed by the model.

The whole dataset was huge so the training can take significant amount of time with heavy performance hit on the device due to training process.

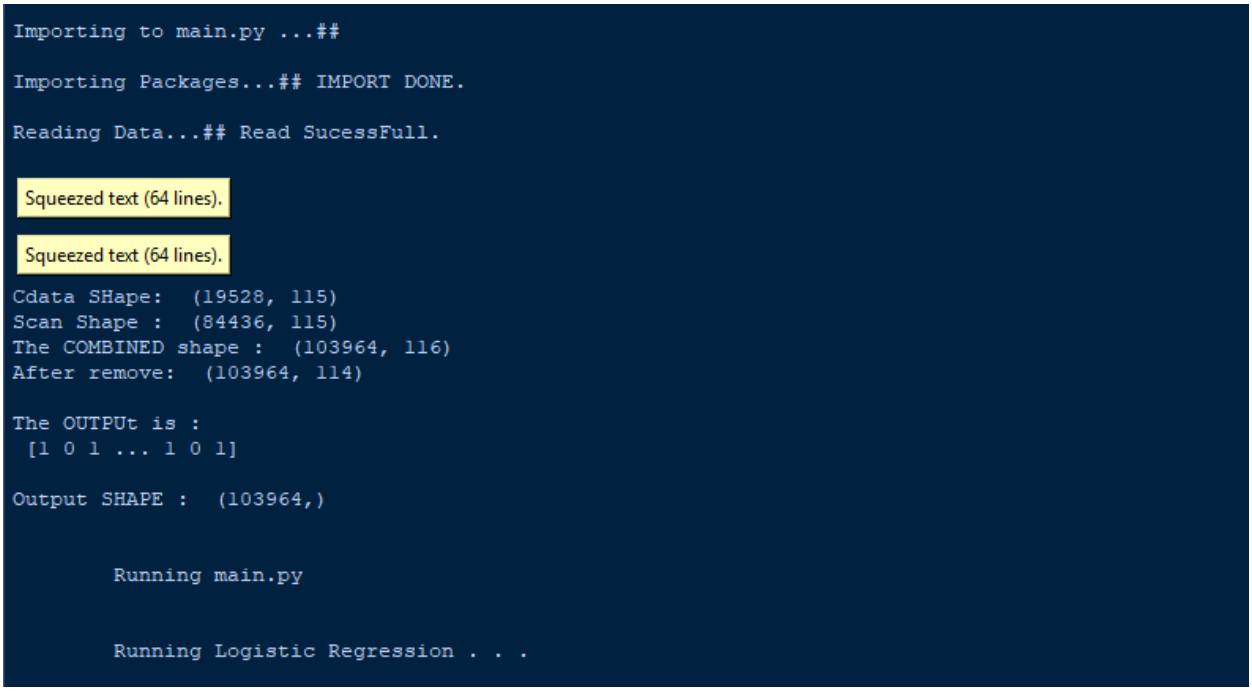
Misclassified samples can be printed as:

```
print("\nThe Misclassified Samples per Descriptions are : \n",np.sum(np.not_equal(y_test,
predictions)))
```

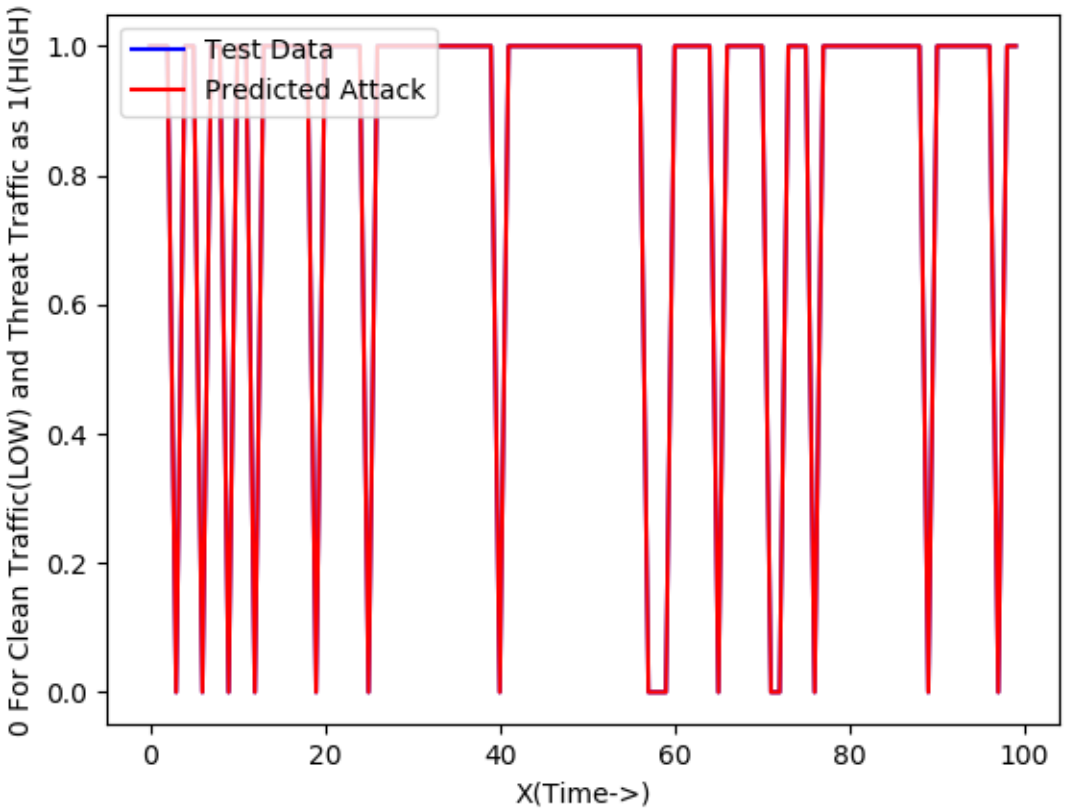
CONCLUSIONS AND RESULTS

Both models using UCI and CTU-13 datasets had pretty ambiguous results. Though the datasets evidently illustrate that the difference in benign traffic and botnet traffic is clearly distinct, the accuracy of the models using both UCI and CTU-13 dataset is almost overwhelming. The accuracy comes in between 98% to 100% for some training models like Decision Tree, Logistic Regression, and even Perceptron. While other models can have varying accuracy between 80% to again 100%. To note, this is not a case of overfitting nor bad data, as different models found online also have same type of accuracy. Furthermore, even the data providers have mentioned study results concluding with 100 % True Positive Rate (TPR).
This concludes that any benign traffic data and botnet traffic data as presented here are evidently distinct.
The **screenshot** of complete running model using either of dataset is given below:

UCI Dataset:

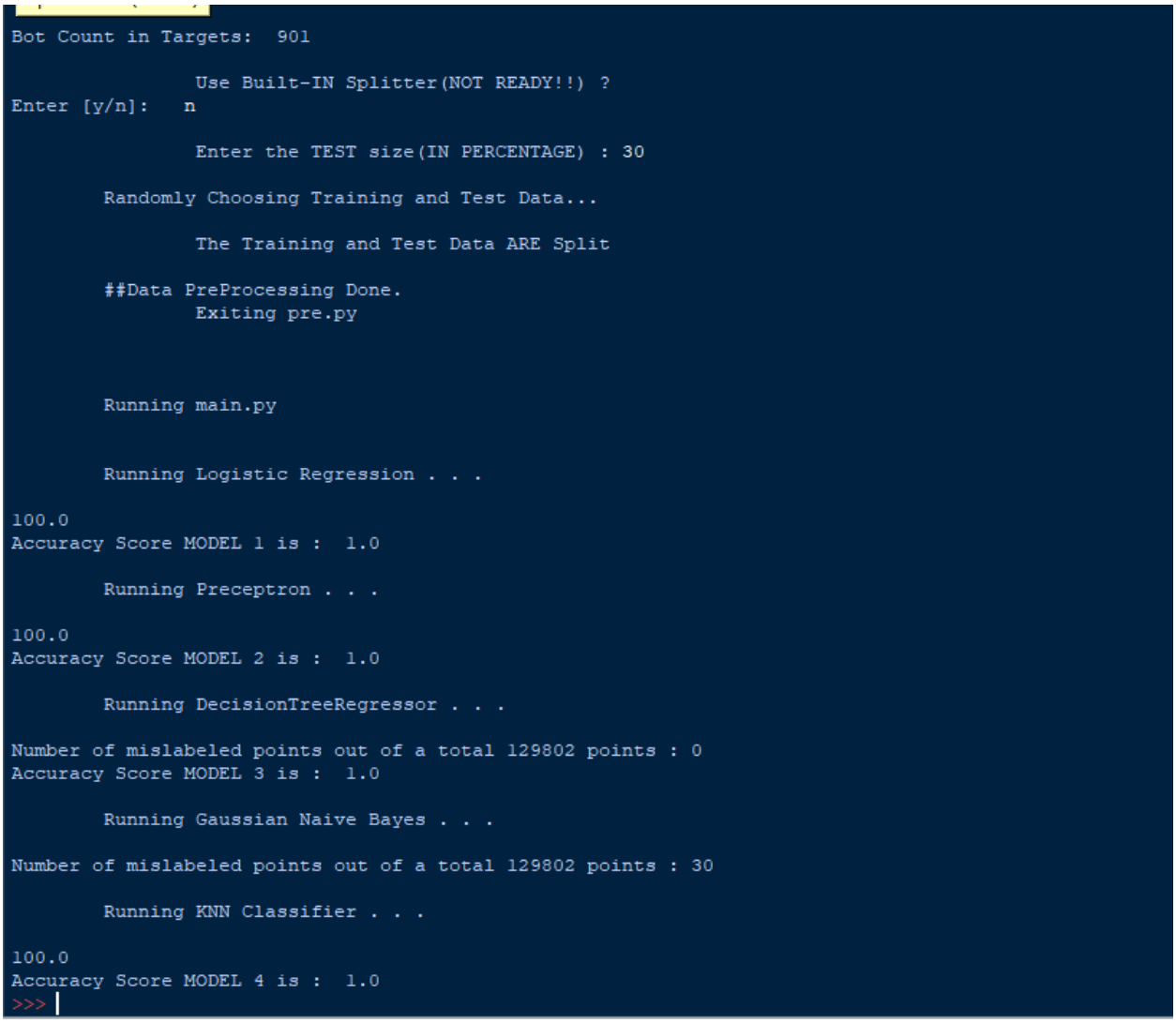


```
99.99358768836166
Accuracy Score MODEL 1 is :  0.9999358768836165

        Running Preceptron . . .

99.86534145559474
Accuracy Score MODEL 2 is :  0.9986534145559475

        Running DecisionTreeRegressor . . .

99.99913968809673

        Running Gaussian Naive Bayes . . .

99.97435075344661
Accuracy Score MODEL 4 is :  0.9997435075344662

        Running KNN Classifier . . .

99.97435075344661
Accuracy Score MODEL 4 is :  0.9997435075344662
>>>
```



CTU-13 Dataset:

```
********Importing to main.py ...**********

                              ######Running pre.py######
Importing Packages...## ######Running dataprep.py######
Importing Packages...## IMPORT DONE.

Importing Data from File...## Read DONE.

Data Ready to Process
Rows and Cols :  129832 16
['Unnamed: 0', 'StartTime', 'Dur', 'Proto', 'SrcAddr', 'Sport', 'Dir', 'DstAddr', 'Dport', 'State',
Bot Count :  901
   Unnamed: 0 StartTime        Dur Proto  ... TotPkts TotBytes SrcBytes Label
0           0   43:28.1   0.000000   tcp  ...       1       60       60     0
1           1   43:32.3  13.431962   tcp  ...       6      388      208     0
2           2   43:32.5  13.350228   tcp  ...       6      388      208     0
3           3   43:32.9  13.010090   tcp  ...       6      388      208     0
4           4   45:09.3  20.990047   tcp  ...       5      308      122     0

[5 rows x 16 columns]

               Dropping Some Columns with Data shape ...## (129832, 16)
DROP DONE.
             Dur  TotPkts  TotBytes  SrcBytes  Label
59836   0.000302        2       271        71      0
47054  19.247005        8       548       342      0
21429   0.000217        2       214        81      0
45249   0.000272        2       327        85      0
91467   0.000259        2       326        86      0


END

IMPORT DONE.

Bot Count in Data:  901
Squeezed text (64 lines).
```



Ran on :

Python 3.6.8 Shell
Python 3.6.8 [MSC v.1916 64 bit (AMD 64 on win32)]

REFERENCES AND CITATIONS

REFERENCES:

https://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT
https://mcfp.weebly.com/the-ctu-13-dataset-a-labeled-dataset-with-botnet-normal-and-background-traffic.html
https://pandas.pydata.org/pandas-docs/stable/reference/api.html
https://stackoverflow.com/questions/16476924/how-to-iterate-over-rows-in-a-dataframe-in-pandas?rq=1
https://stackoverflow.com/questions/36921951/truth-value-of-a-series-is-ambiguous-use-a-empty-a-bool-a-item-a-any-o

CITATIONS:

[1] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, D. Breitenbacher, A. Shabtai, and Y. Elovici 'N-BaIoT: Network-based Detection of IoT Botnet Attacks Using Deep Autoencoders', IEEE Pervasive Computing, Special Issue - Securing the IoT (July/Sep 2018).

[2] “An empirical comparison of botnet detection methods” Sebastian Garcia, Martin Grill, Honza Stiborek and Alejandro Zunino. Computers and Security Journal, Elsevier. 2014. Vol 45, pp 100-123. http://dx.doi.org/10.1016/j.cose.2014.05.011