Configuration Manual

MSc Research Project
Cybersecurity

Ramesh Jaiswar
Student ID: x20102691

School of Computing
National College of Ireland

Supervisor: Ross Spelman

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Ramesh Jaiswar
Student ID:	X20102691
Programme:	MSC in Cybersecurity **Year:** 2020 – 2021
Module:	MSC Internship
Supervisor:	Ross Spelman
Submission Due Date:	16/08/2021
Project Title:	DDoS Attack prediction and classification at Application Layer for Web protocol using Kmeans – SVM Machine Learning Algorithm
Word Count:	888
Page Count:	7

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……

Date: 16th August 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Ramesh Jaiswar
x20102691

1. Introduction

The research necessitates the use of a dataset containing both DDoS attack and benign traffic characteristics. As a part of data pre-processing, it is necessary to perform feature selection, encoding the class variables, constructing a subset of datasets, and building the model and conduct the evaluation of the model. The goal of the configuration manual is to help users set up this research project code on their system so that they can use it to evaluate the study or modify it to meet their specific needs. The prerequisites and environment set up section offers a complete guidance for creating a project environment as well as a list of requirements for replicating the results achieved through the research. Code Execution section contains the complete developed code as well as the parameters for customizing various portions of the project.

2. Requirement

2.1 System Requirement

The process of machine learning involves overhead of resources on the host machine. Hence, it is critical that the hardware configuration on the employed machine be capable of doing such tasks. The following are the system's minimum requirements:

- CPU: Intel i5 6th Gen or Intel i7 5th Gen Processor with 2.4 GHZ
- RAM: 8gb DDR4
- Storage: 15 GB of free space HDD or SSD

2.2 Windows Machine requirement

- Working internet Connection
- Web Browser – MS Edge/ Chrome/ Firefox
- MS Excel – for analyzing the datasets.

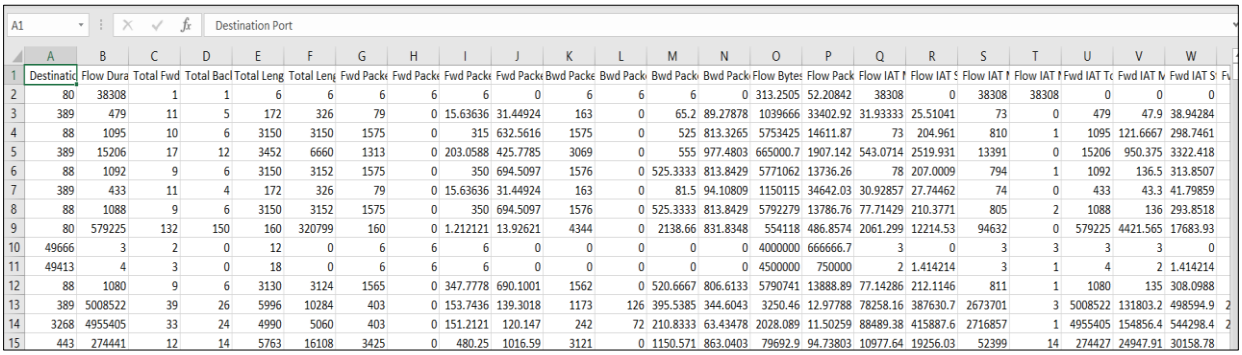
2.3 Software Application requirements

- Anaconda - 64 bits
- Python 3 (Recommended)

3. Dataset Collection

The dataset used to create this model is a CSV file containing web traffic, including both DDoS attack and normal traffic. It is available for download from an internet repository that provides a dataset for cybersecurity research. [1]. The source for downloading the dataset can be found at https://www.unb.ca/cic/datasets/ids-2017.html

The size of the dataset file is around 2.1 GB due to which it requires a strong internet connection. Below snapshot shows the preview of the dataset in excel sheet.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	Destinatio	Flow Dura	Total Fwd	Total Bacl	Total Leng	Total Leng	Fwd Packe	Fwd Packe	Fwd Packe	Fwd Packe	Bwd Packe	Bwd Packe	Bwd Packe	Bwd Packe	Flow Bytes	Flow Pack	Flow IAT I	Flow IAT S	Flow IAT I	Flow IAT I	Fwd IAT To	Fwd IAT M	Fwd IAT St
2	80	38308	1	1	6	6	6	6	6	0	6	6	6	0	313.2505	52.20842	38308	0	38308	38308	0	0	0
3	389	479	11	5	172	326	79	0	15.63636	31.44924	163	0	65.2	89.27878	1039666	33402.92	31.93333	25.51041	73	0	479	47.9	38.94284
4	88	1095	10	6	3150	3150	1575	0	315	632.5616	1575	0	525	813.3265	5753425	14611.87	73	204.961	810	1	1095	121.6667	298.7461
5	389	15206	17	12	3452	6660	1313	0	203.0588	425.7785	3069	0	555	977.4803	665000.7	1907.142	543.0714	2519.931	13391	0	15206	950.375	3322.418
6	88	1092	9	6	3150	3152	1575	0	350	694.5097	1576	0	525.3333	813.8429	5771062	13736.26	78	207.0009	794	1	1092	136.5	313.8507
7	389	433	11	4	172	326	79	0	15.63636	31.44924	163	0	81.5	94.10809	1150115	34642.03	30.92857	27.74462	74	0	433	43.3	41.79859
8	88	1088	9	6	3150	3152	1575	0	350	694.5097	1576	0	525.3333	813.8429	5792279	13786.76	77.71429	210.3771	805	2	1088	136	293.8518
9	80	579225	132	150	160	320799	160	0	1.212121	13.92621	4344	0	2138.66	831.8348	554118	486.8574	2061.299	12214.53	94632	0	579225	4421.565	17683.93
10	49666	3	2	0	12	0	6	6	6	0	0	0	0	0	4000000	666666.7	3	0	3	3	3	3	0
11	49413	4	3	0	18	0	6	6	6	0	0	0	0	0	4500000	750000	2	1.414214	3	1	4	2	1.414214
12	88	1080	9	6	3130	3124	1565	0	347.7778	690.1001	1562	0	520.6667	806.6133	5790741	13888.89	77.14286	212.1146	811	1	1080	135	308.0988
13	389	5008522	39	26	5996	10284	403	0	153.7436	139.3018	1173	126	395.5385	344.6043	3250.46	12.97788	78258.16	387630.7	2673701	3	5008522	131803.2	498594.9
14	3268	4955405	33	24	4990	5060	403	0	151.2121	120.147	242	72	210.8333	63.43478	2028.089	11.50259	88489.38	415887.6	2716857	1	4955405	154856.4	544298.4
15	443	274441	12	14	5763	16108	3425	0	480.25	1016.59	3121	0	1150.571	863.0403	79692.9	94.73803	10977.64	19256.03	52399	14	274427	24947.91	30158.78

4. Packages and Imports for Code

The model was developed on a python code using jupyter notebook, the code involves several packages and imports which are mentioned as follows:

- Matplotlib 3.1.2
- Numpy 1.18.0
- Pyparsing 2.4.6
- scikit-learn 0.22
- scipy 1.4.1
- sklearn
- pandas
- KMeans
- mpl_toolkits
- Seaborn
- train_test_split
- Sklearn.metrics
- SVC

5. Feature Selection and Data Preprocessing

In order to perform the feature selection, the Correlation Coefficient technique is implemented using the below code on the complete dataset.

In [144]:
```
# Correlation matrix
def plotCorrelationMatrix(df, graphWidth):
    #filename = df.dataframeName
    df = df.dropna('columns') # drop columns with NaN
    df = df[[col for col in df if df[col].nunique() > 1]] # keep columns where there are more than 1 unique values
    if df.shape[1] < 2:
        print(f'No correlation plots shown: The number of non-NaN or constant columns ({df.shape[1]}) is less than 2')
        return
    corr = df.corr()
    plt.figure(num=None, figsize=(graphWidth, graphWidth), dpi=80, facecolor='w', edgecolor='k')
    corrMat = plt.matshow(corr, fignum = 1)
    plt.xticks(range(len(corr.columns)), corr.columns, rotation=90)
    plt.yticks(range(len(corr.columns)), corr.columns)
    plt.gca().xaxis.tick_bottom()
    plt.colorbar(corrMat)
    plt.title(f'Correlation Matrix for ', fontsize=15)
    plt.show()
```

Correlation function is created using the above code, and is called by passing the data frame in it, along with the graph width:

]:
```
plotCorrelationMatrix(ddos_test_df1, 19)
```

The output of the function returns is list of features which has strong relationship with the target variable:

In [151]:
```
# with the following function we can select highly correlated features
# it will remove the first feature that is correlated with anything other feature

def correlation(dataset, threshold):
    col_corr = set()  # Set of all the names of correlated columns
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold: # we are interested in absolute coeff value
                colname = corr_matrix.columns[i]  # getting the name of column
                col_corr.add(colname)
    return col_corr
```

In [152]:
```
corr_features = correlation(ddos_test_df1, 0.99)
len(set(corr_features))
```

Out[152]:
```
21
```

In [153]:
```
corr_features
```

Out[153]:
```
{' Average Packet Size',
 ' Avg Bwd Segment Size',
 ' Avg Fwd Segment Size',
 ' Bwd Header Length',
 ' ECE Flag Count',
 ' Fwd Header Length',
 ' Fwd Header Length.1',
 ' Fwd IAT Max',
 ' Idle Max',
 ' Idle Min',
 ' SYN Flag Count',
 ' Subflow Bwd Bytes',
```

6. Coding

The subset of dataset is created using the derived features and the target variable ‘Label’ is deleted from the dataset. Which is then suppled to below set of code, which is used to find the value of K using the Elbow method

In [2]:
```
#Clean Dataset

df = pd.read_csv('Dataset_DDoS1_Benign_DDosHulk_18Features_Unlabelled.csv')
df.dataframeName = 'Dataset_DDoS1_Benign_DDosHulk_18Features_Unlabelled.csv'

nRow, nCol = df.shape
print(f'There are {nRow} rows and {nCol} columns')
```

```
There are 150001 rows and 18 columns
```

In [3]:
```
#finding number of clusters

wcss = []
for i in range(1,11):
   model = KMeans(n_clusters = i, init = "k-means++")
   model.fit(df)
   wcss.append(model.inertia_)
plt.figure(figsize=(10,10))
plt.plot(range(1,11), wcss)
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

The value of K found using elbow method is passed to Kmeans algorithm which creates the cluster of the data based on its characteristics

In [4]:
```
# K means model

centers = np.array(model.cluster_centers_)
model = KMeans(n_clusters = 2, init = "k-means++")
label = model.fit_predict(df)
plt.figure(figsize=(10,10))
uniq = np.unique(label)

df = np.array(df) #for converting df to array
for i in uniq:
    plt.scatter(df[label == i, 0], df[label == i, 1], label = i)
    plt.scatter(centers[:,0], centers[:,1], marker="x", color='k')
#This is done to find the centroid for each clusters.
plt.legend()
plt.show()
```

The output generated from the above code is in the form of array which is then appended to the unlabeled dataset using below piece of code:

In [5]:
```
label # printing label to check if the values is feeded into it
```

Out[5]:
```
array([0, 0, 0, ..., 1, 1, 1])
```

In [6]:
```
labeldf = pd.DataFrame(label) # creating label.csv file to check the 0 and 1 values
labeldf.to_csv('Label.csv')
```

In [7]:
```
ColPosition = 18
Kmeansdata=pd.DataFrame(df)
#Kmeansdata['Decision'] = label

Kmeansdata.insert(ColPosition, "Decision", label)  # adding label field to datafile
```

This gives the dataset, that was labeled using k-means clustering. The newly created dataset is then supplied to SVM model to train and test the model. The snapshot below illustrates the piece of code:

```
[2]: from sklearn.model_selection import train_test_split

     X =svmdf.drop('Decision', axis=1) # droping Decision data, df is now without label for training and testing
     y =svmdf['Decision']

     X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.4,random_state=0, stratify=y)

     X.head()
```

The data is spilt into X – independent variables and y – target Variables which is then used by SVM classifier for training and testing the model

```
In [15]: # Run this block of code if you want to skip the scaler part - before running comment the standard scalar part from above code

         #from sklearn.preprocessing import StandardScaler

         #X_train = StandardScaler().fit_transform(X) #optional - if you need the data to be scaled in standard manner
         #X_test = StandardScaler().fit_transform(X_test)

         from sklearn.svm import SVC

         kername = 'rbf'
         svclassifier = SVC(kernel= kername, C = 0.01)
         svclassifier.fit(X_train, y_train)


         #y_train_pred = svclassifier.predict(X_train)
         y_test_pred = svclassifier.predict(X_test)
         y_test_pred

Out[15]: array([0, 0, 0, ..., 0, 0, 0], dtype=int64)

In [16]: from sklearn.metrics import accuracy_score,confusion_matrix

         confusion_matrix(y_test,y_test_pred)

Out[16]: array([[46197,     0],
                [   50, 13754]], dtype=int64)

In [17]: accuracy_score(y_test,y_test_pred)

Out[17]: 0.9991666805553241
```

The output of the model is measured in terms of accuracy score which is calculated using the confusion matrix.