National College of Ireland

Configuration Manual

MSc Research Project
Cybersecurity

Akash .
Student ID: X20258194

School of Computing
National College of Ireland

Supervisor: Michael Pantridge

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Akash . …….

Student ID: X20258194……

Programme: Cybersecurity…… **Year:** 2021-2022……

Module: MSc Research Project ……..……

Supervisor: Michael Pantridge ……..……

Submission Due Date: 15/08/2022……..……

Project Title: An SDN based Machine Learning and Deep Learning model for DDoS attack detection on IoT Network……..……

Word Count: ……1096……**Page Count**……11……..……..

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: Akash . ……

Date: ……

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

Akash .
X20258194

1 Introduction

This manual contains details regarding the proposed model's setup and requirements, such as technical specifications and software to be installed. Furthermore, this setup handbook explains how to implement the algorithms needed to develop the proposed model.

2 System Configuration

The proposed model uses various machine learning algorithms as well as deep learning. These algorithms use libraries and layers which are power hungry. From dataset processing to calculate accuracy every bit of the model will require processing power. For the same reason we have used a considerable powerful desktop to meet the model's need.

Desktop Specification:

- Performance-oriented CPU: i9 9900k, stable overclock at 5.1ghz liquid metal
- 32 GB DDR4 ram.
- External GPU AMD Radeon 5700xt with 8GB RAM.
- 2TB SSD storage

The model is dependent on tools, software and libraries that would help configure the model. Below are the tools and libraries used in the proposed model.

Software and tools:

Operating System: Windows 10 Pro 64-bit
Python: 3.10.64
Visual Studio Code

Libraries used for the proposed model:

NumPy and Panda were used for dataset extraction and processing
Keras framework was used for analysis of data and its implementation in neural network algorithm.
Sklearn (Scikit-learn) the library for machine learning was used for modelling, regression, classification and other functions.

3 Implementation steps

Post installation of necessary tools, software and libraries, the libraries were imported on python environment. SkLearn (Scikit-Learn), Keras, NumPy and Pandas were some of the important libraries imported.

```
import numpy as np
import pandas as pd
from sklearn import preprocessing
import seaborn as sns
from pylab import plot, show
import matplotlib.pyplot as plt
plt.style.use('ggplot')
from sklearn.feature_selection import VarianceThreshold
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import ExtraTreesClassifier
from prettytable import PrettyTable
from sklearn.metrics import roc_curve, auc
from mlxtend.plotting import plot_confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from keras.models import Model
from keras.layers import LSTM, Activation, Dense, Dropout, Input, Conv1D, Embedding
from keras.models import Sequential
from keras.layers import Conv1D, GlobalMaxPooling1D
from keras.models import Sequential
from keras.layers import Dense,Flatten,Conv1D,BatchNormalization,Dropout
from keras.callbacks import EarlyStopping
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.decomposition import PCA
from sklearn.preprocessing import MinMaxScaler
import warnings
warnings.filterwarnings("ignore")
✓ 9.3s
```

Figure 1 Libraries

Once the libraries are imported, the datasets need to be fetched from the system's directory. As the proposed model needs to maintain two datasets i.e., DDoS traffic dataset and All-traffic dataset. Which later will be used for training the model and testing it.

The model uploads the full features dataset provided by The University of New South Wales specifically for training and testing the dataset. The datasets once fetched are then divided into the DDoS traffic dataset i.e., “data” and All-Traffic dataset i.e., “datatest”.

Figure 2 Loading Datasets and combing them

As the model has fetched the dataset the next step will be to analyze the datasets and prepare it for training and testing. The analysis started with looking for numbers of rows and columns in the dataset and then removing empty spaces in the dataset.

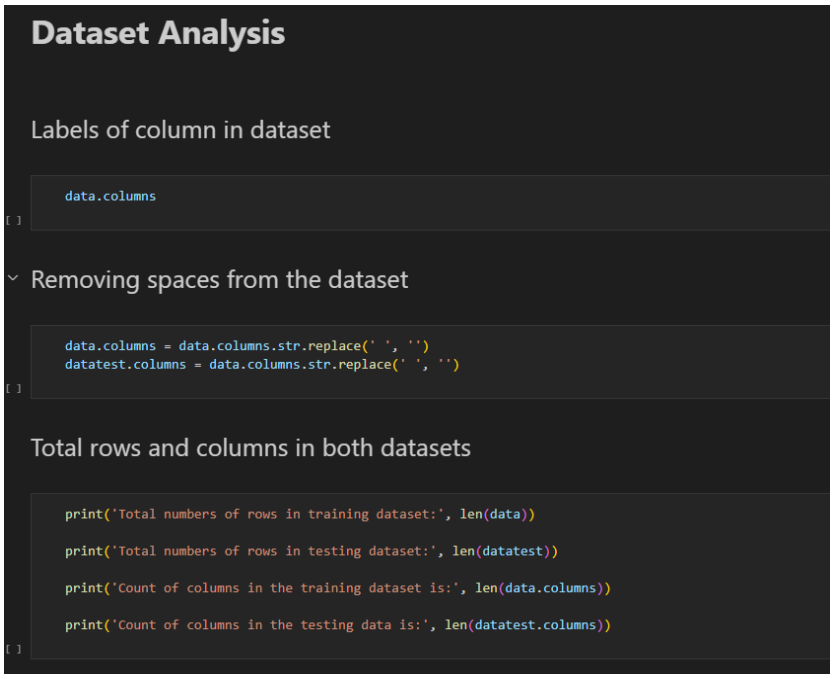

Figure 3 Dataset Analysis

Further the model checks for any Null values in the dataset as most algorithms are unable to predict the accuracy if there is NULL value in the dataset. So, the model drops any NULL value in the both the datasets.

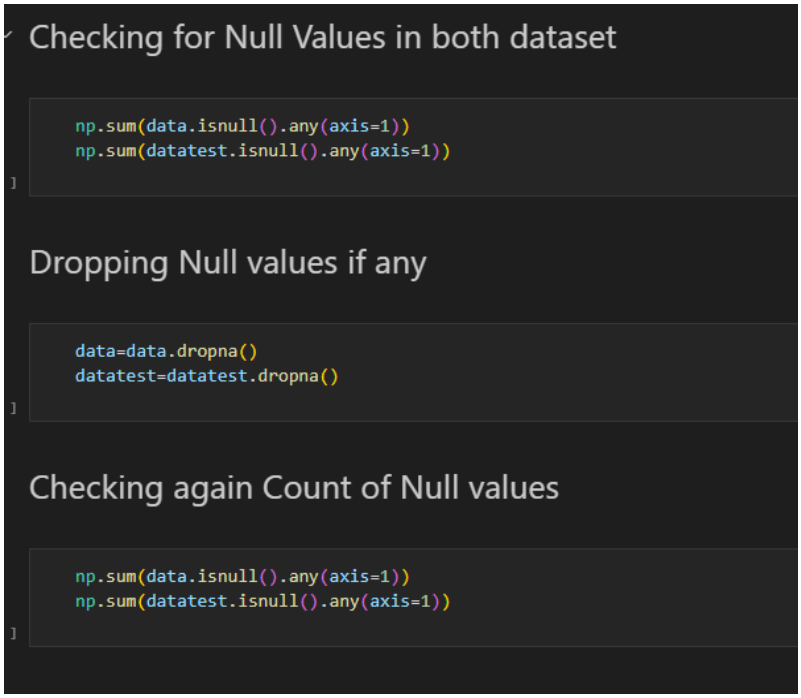

Figure 4 Checking for Null Value

As the dataset was found to have some redundant features as well as infinity values in few of the columns. The model then converts those infinity value to 0 and drop the repetitive columns. Also, any duplicates rows were dropped.

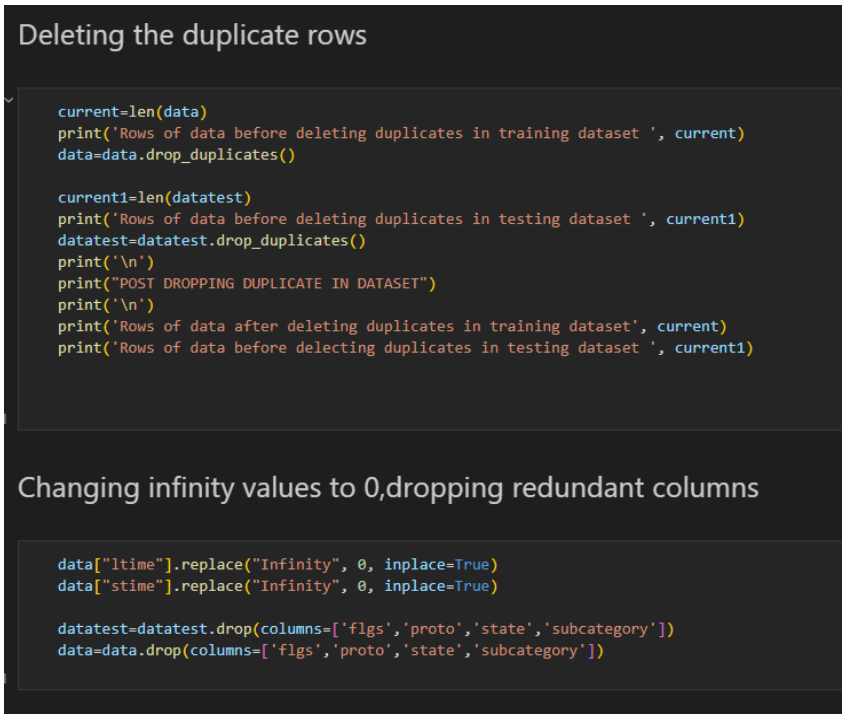

Figure 5 Checking for Duplicates, Infinity and dropping redundant columns.

Since we were using 64-bit data, the memory consumption will be high when processing them. To reduce the memory consumption all the 64-bit data types were converted to 32-bit data types.

Changing the data types from 64-bit to 32-bit to decrease the memory usage

+ Code

```
integer = []
floatt = []
cat=[]
for integerr in data.columns[:-1]:
    if data[integerr].dtype == "int64": integer.append(integerr)
    elif data[integerr].dtype == "float64": floatt.append(integerr)
    elif data[integerr].dtype == "object": cat.append(integerr)


data[integer] = data[integer].astype("int32")
data[floatt] = data[floatt].astype("float32")
```

```
integer = []
floatt = []
cat=[]
for integerr in datatest.columns[:-1]:
    if datatest[integerr].dtype == "int64": integer.append(integerr)
    elif datatest[integerr].dtype == "float64": floatt.append(integerr)
    elif datatest[integerr].dtype == "object": cat.append(integerr)


datatest[integer] = datatest[integer].astype("int32")
datatest[floatt] = datatest[floatt].astype("float32")
```

Figure 6 Converting Datatype to decrease memory usage

The datasets is then seperated into input features and traffic label. It is done as the the model will be proceeding with feature selection and adding label into features doesn't make sense as label will be the decising factor when accuracy is obtained.

- Input_features = X
- Label = y

```
X=data.drop(columns=['Label'])
data['Label']=data['Label'].replace('Normal',0)
data['Label']=data['Label'].replace('DDoS',1)
y=data['Label']
sns.countplot(data= X, x = y)
```

```
X1=datatest.drop(columns=['Label'])
datatest['Label']=datatest['Label'].replace('Normal',0)
datatest['Label']=datatest['Label'].replace('DDoS',1)
y1=datatest['Label']
sns.countplot(data= X1, x = y1)
datatest['Label'].value_counts()
```

```
print('Shape of the input features:  ', X.shape[1])
print('Shape of the input features:  ', X1.shape[1])
```

Figure 7 Input Features and Labels separated.

The model uses Chi-2 and Tree based feature selection for both the datasets to filter the useful features. Post which all the features selected are combined and if there is a common feature selected by both the feature selection algorithm then one of them is dropped.

Figure 8 Feature Selection.

Since the model found there to be still quite several features and there is a possibility of quite a number of features be redundant in nature, so the model then uses principal component analysis to convert these redundant features into a single component.

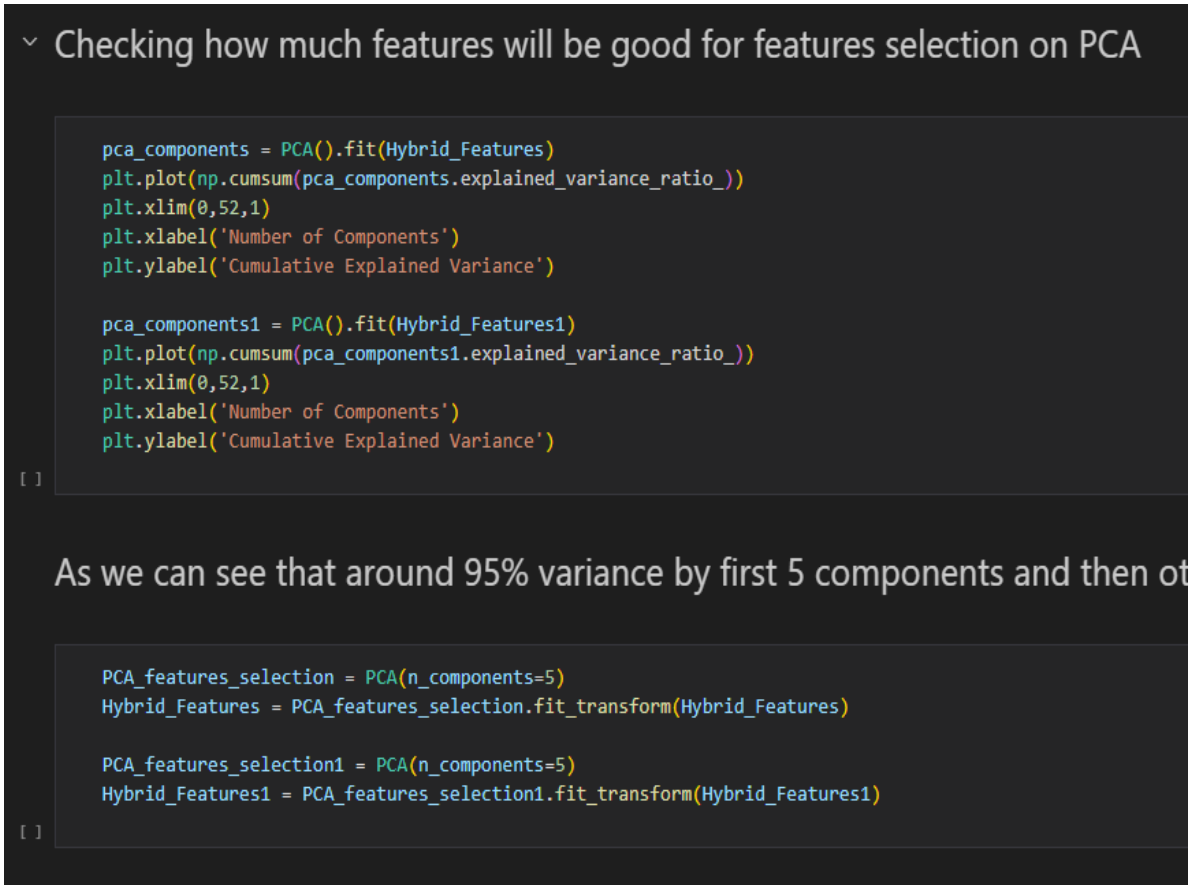

Figure 9 PCA with 5 components.

Post both the dataset goes through PCA then the model proceeds with Training and testing the model with the datasets we have chosen and have applied changes for better and accurate results.

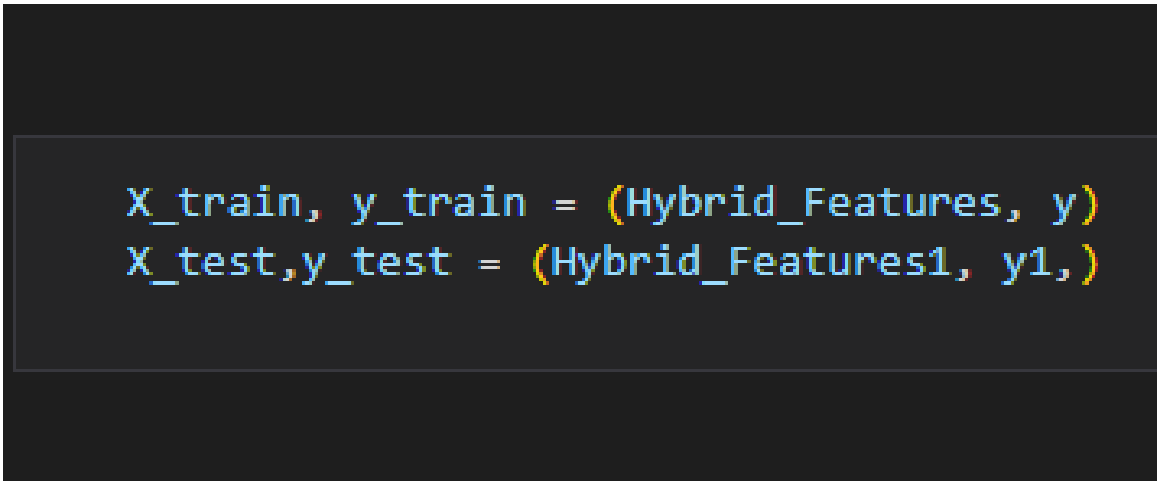

Figure 10 Training and testing Dataset

The Model then uses various machine learning and deep learning algorithms to train the model using the DDoS traffic dataset.

Decision Trees

```
DTC=DecisionTreeClassifier(random_state=10, max_depth=1)
DTC= DTC.fit(X_train , y_train)
DTC
```

Random Forest

```
Ran_For= RandomForestClassifier(n_estimators=1,max_depth=15, random_state=1,max_leaf_nodes=10)
Ran_For= Ran_For.fit(X_train , y_train)
Ran_For
```

GaussianNB Algorithm

```
KNN=GaussianNB()
KNN= KNN.fit(X_train , y_train)
KNN
```

CNN Model

```
X_train=np.array(X_train)
X_test=np.array(X_test)
x_train=X_train.reshape(X_train.shape[0],X_train.shape[1],1)
x_test=X_test.reshape(X_test.shape[0],X_test.shape[1],1)
cnn_model=Sequential()
cnn_model.add(Conv1D(filters=64, kernel_size=4, activation='relu', input_shape=x_train[0].shape))
cnn_model.add(BatchNormalization())
cnn_model.add(Dropout(0.3))
cnn_model.add(Conv1D(filters=128, kernel_size=2, activation='relu'))
cnn_model.add(BatchNormalization())
cnn_model.add(Dropout(0.3))
cnn_model.add(Flatten())
cnn_model.add(Dense(32, activation='relu'))
cnn_model.add(Dropout(0.5))
cnn_model.add(Dense(1, activation='sigmoid'))
cnn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
cnn_model.summary()
```

Training and validating

```
history=cnn_model.fit(x_train,y_train,epochs=100,
        validation_split=0.1,callbacks=[EarlyStopping(monitor='val_loss',min_delta=0.0001, patience=
```

Figure 11 Machine Learning and Deep Learning Models were then trained by the DDoS dataset

Post the mode are trained then the accuracy, metrics, and confusion matrix of all the algorithms are taken into account and then comparted post results are available.

ACCURACY, CONFUSION MATRIX AND METRICS

```
y_pred1 = KNN.predict(X_test)
kn=KNN.score(X_test, y_test)
print('Accuracy score= {:.5f}'.format(KNN.score(X_test, y_test)))
print('\n')
print("confusion matrix")
print('\n')
CR=confusion_matrix(y_test, y_pred1)
TP = CR[0][0]
FP = CR[0][1]
FN = CR[1][0]
TN = CR[1][1]
re=pd.DataFrame()
re['True Positive']=[TP]
re['False Positive']=[FP]
re['False Negative']=[FN]
re['True Negative']=[TN]
print('Count of True Positive = ',TP)
print('Count of False Positive = ',FP)
print('Count of False Negative = ',FN)
print('Count of True Negative = ',TN)
re
print('\n')
print("Precision, Recall, F1")
print('\n')
CR=classification_report(y_test, y_pred1,target_names ={'Normal':0,'DDoS':1})
print(CR)
print('\n')
print('\n')
print("confusion matrix")
print('\n')
CR=confusion_matrix(y_test, y_pred1)
print(CR)
print('\n')

fig, ax = plot_confusion_matrix(conf_mat=CR,figsize=(10, 10),
                                show_absolute=True,
                                show_normed=True,
                                colorbar=True)
plt.show()
```

Figure 12 Code used for Accuracy, metrics and to plot confusion matrix.

Once the algorithms are done predicting the accuracy and other metrics. The results of all these algorithms is presented using pretty table function in a single diagram.

Comparison of all algorithms Results

```
x = PrettyTable()
print('\n')
print("Comparison of all algorithm results")
x.field_names = ["Model", "Accuracy"]

x.add_row(["Decision Trees Algorithm", round(dt,5)])
x.add_row(["Random Forest Algorithm", round(rn,5)])
x.add_row(["Naive Bayes Algorithm", round(kn,5)])
x.add_row(["CNN Model", round(Deep_Learning_acc1,5)])
print(x)
print('\n')
```

Figure 13 Results of all models clubbed into a single table.