Configuration Manual

MSc Research Project
MSC. Cyber Security

Sweety Kaushik
Student ID: X19205783

School of Computing
National College of Ireland

Supervisor: Ross Spelman

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sweety Kaushik		
Student ID:	X19205783		
Programme:	MSc. in Cyber Security	**Year:**	2020-21.
Module:	Internship		
Lecturer:	Ross Spelman		
Submission Due Date:	16-08-2021		
Project Title:	**Enhanced the intrusion detection accuracy rate and performance using deep CNN-LSTM.**		
Word Count:1600		**Page Count: 10**	

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: ……………………………………………Sweety Kaushik………………………………

Date: ………………………………….....16-08-2021……………………………………………

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

Sweety Kaushik
Student ID: X19205783

1. System Requirements

Software OR Hardware Requirements: - In this configuration manual i explained the steps in details about the requirements of basic software's and hardware to run the implemented project smoothly. I have mentioned all the description of hardware specification and software version to execute the code.

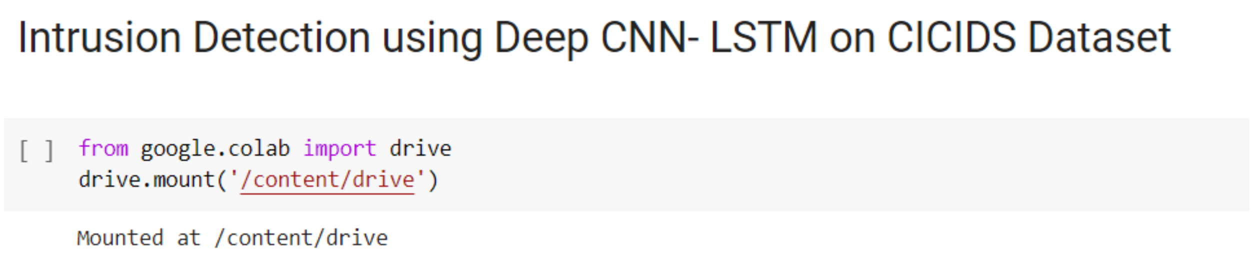
Software: -

Python Ver.3.8.2
Google Colaboratory (Google platform)
Jupyter Notebook - ver. 6.03
TensorFlow - Ver. 2.4
Anaconda - Ver. 1.9

Hardware: -
The hardware configuration of machine where i executed my implemented project are as follows.

RAM: - 12 GB
Storage (SSD): - 512GB
Processor: - AMD Ryzen 5 4500U with Radeon Graphics 2.38 GHz
Operating System: - Windows 10 Home Single Language, 64 Bit Architecture.

- Here, my mount point directory is /content/drive/mydrive/cicids where my dataset and model exist in fig1.

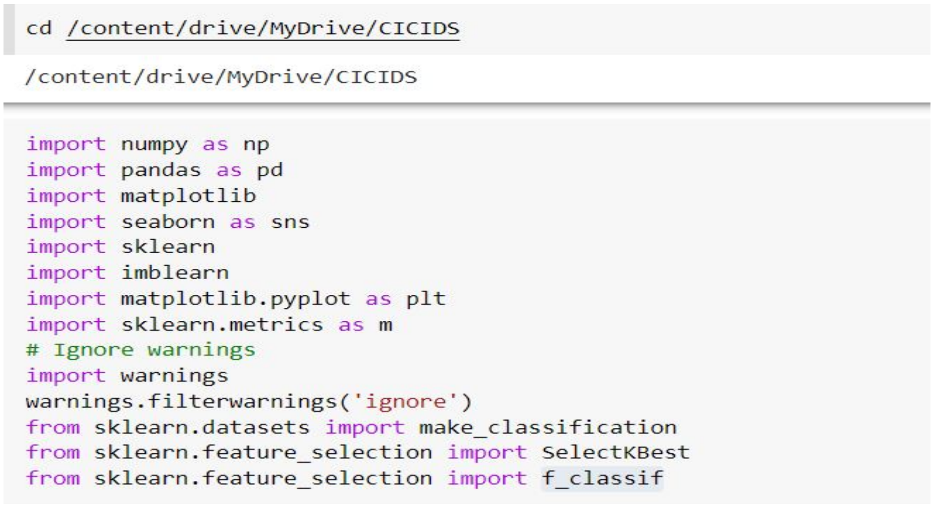

Intrusion Detection using Deep CNN- LSTM on CICIDS Dataset

```
[ ] from google.colab import drive
    drive.mount('/content/drive')

    Mounted at /content/drive
```

1. Importing The Libraries: -

Pandas and numpy library were imported to use for data pre-processing as shown in Figure 2. We imported 'matplotlib' and 'seaborn' library for generating visualization and animation, then we imported sklearn library that is useful package for machine learning in python.

make_classification from sklearn.dataset is used to create a random n class classification problem visualization, selectKBest is imported from sklearn.feature_selection and it is preferred for best predictors for target variables,f_classif it was also imported from sklearn.feature_selection that is mainly used for categorical target which is based on ANOVA and we used Analysis of variance (ANOVA) in feature selection to select the best features that we will discuss later.

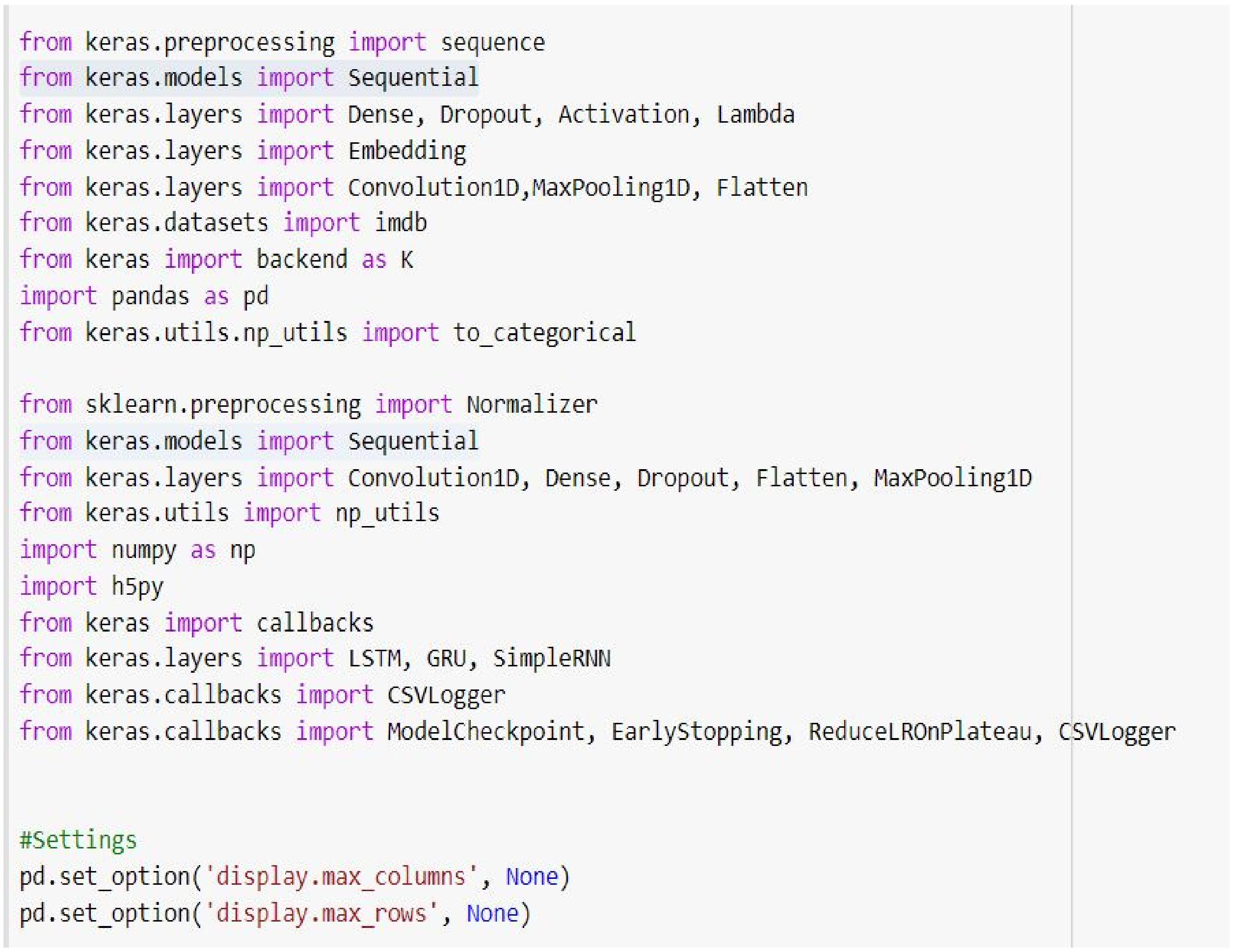


```
cd /content/drive/MyDrive/CICIDS

/content/drive/MyDrive/CICIDS
```

```
import numpy as np
import pandas as pd
import matplotlib
import seaborn as sns
import sklearn
import imblearn
import matplotlib.pyplot as plt
import sklearn.metrics as m
# Ignore warnings
import warnings
warnings.filterwarnings('ignore')
from sklearn.datasets import make_classification
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
```

Fig. 1. Scikit Libraries

1.2). All these libraries were imported for the use of deep learning architecture. Here, we used various of library function to create model, In the first line, I imported sequence function from keras.preprocessing library for listing of sequences, 'Sequential' function was imported from keras.models library which i used for the purpose of initiating the CNN model. Dense, Dropout, Activation, Lambda functions were imported from keras.layers library as all these four functions will be used in CNN layer.

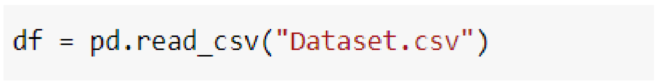


```
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Lambda
from keras.layers import Embedding
from keras.layers import Convolution1D,MaxPooling1D, Flatten
from keras.datasets import imdb
from keras import backend as K
import pandas as pd
from keras.utils.np_utils import to_categorical

from sklearn.preprocessing import Normalizer
from keras.models import Sequential
from keras.layers import Convolution1D, Dense, Dropout, Flatten, MaxPooling1D
from keras.utils import np_utils
import numpy as np
import h5py
from keras import callbacks
from keras.layers import LSTM, GRU, SimpleRNN
from keras.callbacks import CSVLogger
from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau, CSVLogger


#Settings
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

Fig. 2 Keras Libraries

2. Load The Data: -

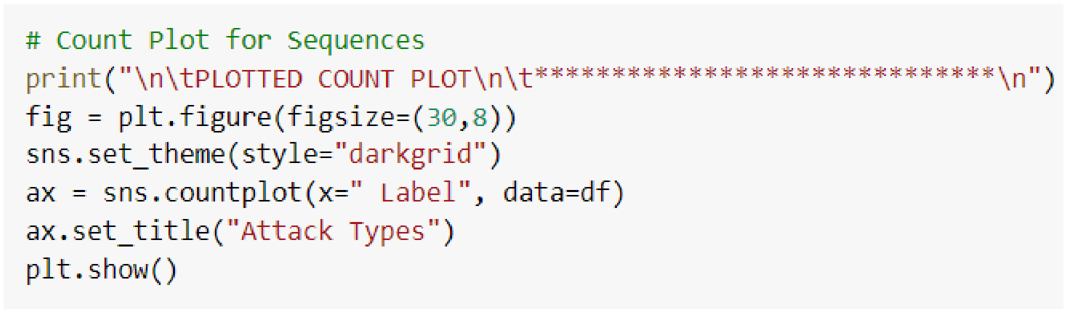
here, we imported our cicids 2017 dataset and here we used pandas (pd) function 'pd.read_csv and then dataset filename to load. So this is the overall dataset here. as you can see so this is the label so label means what type of attacks and before labels this is the features.



```
df = pd.read_csv("Dataset.csv")
```

2.1. Count plot for sequences (Attack types)-

I just plotted all the things here. So 'seaborn'(sns) themes and seaborn countplot to plots count for sequences. In 'fig' i gave a figure count to plot and for seaborn theme, i chose 'whitegrid' for visualization of count plots for sequences and then plotted all attack types and 'set_title' for main heading which is "Attack Types".

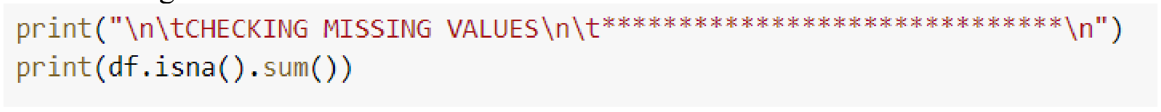


```
# Count Plot for Sequences
print("\n\tPLOTTED COUNT PLOT\n\t******************************\n")
fig = plt.figure(figsize=(30,8))
sns.set_theme(style="darkgrid")
ax = sns.countplot(x=" Label", data=df)
ax.set_title("Attack Types")
plt.show()
```

Fig.3 Attack Types

3. Data Pre-Processing: -

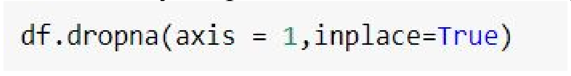
Now in pre-processing of data stage I used 'df.isna' to count for any missing values from data frames(df), and here we are getting null missing values but in Flow Bytes/s features we found 1372 missing values .



```
print("\n\tCHECKING MISSING VALUES\n\t******************************\n")
print(df.isna().sum())
```

Fig.4 Data Pre-Processing

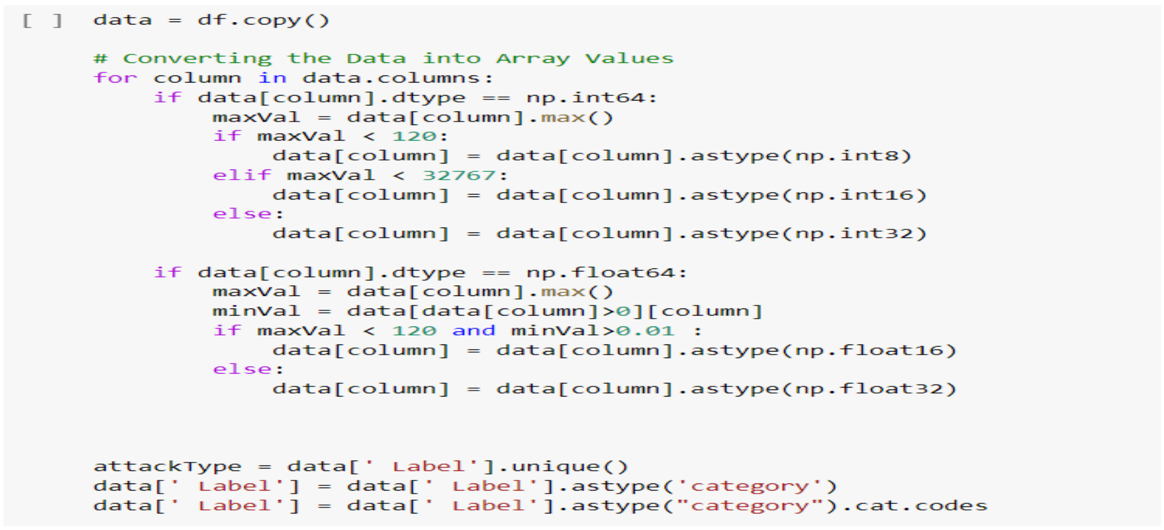
3.1. Now we used 'df.dropna' function to remove the missing values that we found 1372, it will basically drop all rows that contains any missing values.



```
df.dropna(axis = 1,inplace=True)
```

3.2 Data conversion into Arrays -

Here, we wrote all codes to convert the data into array values, it is like a scaling process that do the minmax scaling, it basically checks each and every iteration for row and column. because each and every data consists of minimum and maximum values, we also applied here if and else condition , like 'if maxval<120' it will check the iteration, if maximum value is less than the given value (120) then it will convert data into integer8 type , if maximam value 'maxval<32767' is less than 32767 then it will convert data into integer16 (int16) or else if more than 32767 then it will convert to integer32 (int32) and type32 means it will convert into more float so we can reduce the size of it and store the maximum value of 32767 in integer32.
integer8 = 8bit size
integer16 = 16bit size
integer32 = 32bit size.
if the maximum value (maxval<120) is between 120 to 0.01 then it means we have to convert the value into float16 values or else we need to convert the values into float32.
here we defined our attacktypes data label form like benign, portscan.... and convert into numerical values 0,1,2.

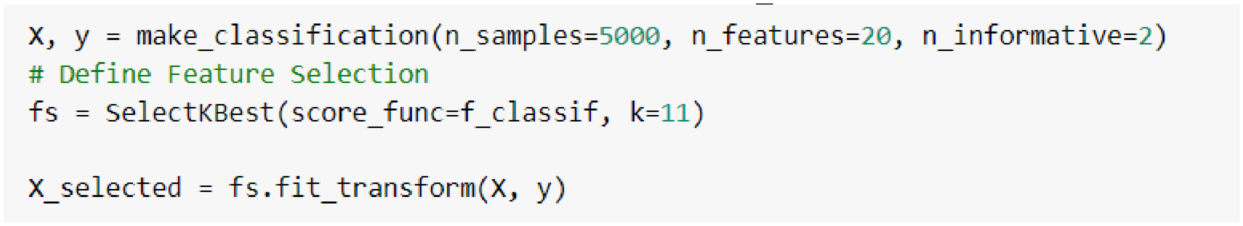


```
[ ]  data = df.copy()

     # Converting the Data into Array Values
     for column in data.columns:
         if data[column].dtype == np.int64:
             maxVal = data[column].max()
             if maxVal < 120:
                 data[column] = data[column].astype(np.int8)
             elif maxVal < 32767:
                 data[column] = data[column].astype(np.int16)
             else:
                 data[column] = data[column].astype(np.int32)

         if data[column].dtype == np.float64:
             maxVal = data[column].max()
             minVal = data[data[column]>0][column]
             if maxVal < 120 and minVal>0.01 :
                 data[column] = data[column].astype(np.float16)
             else:
                 data[column] = data[column].astype(np.float32)



     attackType = data[' Label'].unique()
     data[' Label'] = data[' Label'].astype('category')
     data[' Label'] = data[' Label'].astype("category").cat.codes
```

Fig. 5. Data conversion into arrays

4. Feature Selection: -

Here, we used Anova feature selection as it is the very effective as compared to other features,
It will generate the maximum number of samples which is 5000 and features values is 20 and we reduce the minus values and we will accept only positive values and we take 20 features apart from total number of features, 'n_informative=2' means it's a threshold value so normally the threshold default value is 1 but our data is having more features like 20 so we set threshold value 2.
In Anova we used two features selection selectKBest and f_classif.

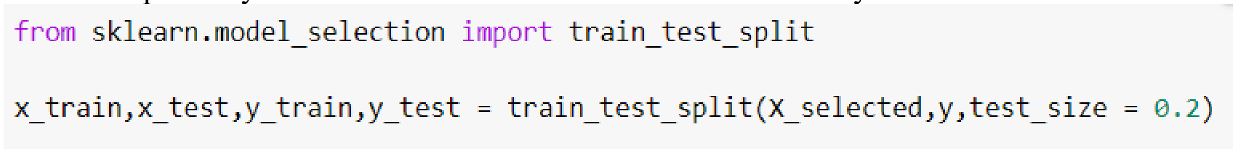


```
X, y = make_classification(n_samples=5000, n_features=20, n_informative=2)
# Define Feature Selection
fs = SelectKBest(score_func=f_classif, k=11)

X_selected = fs.fit_transform(X, y)
```

Fig. 6 Anova Feature Selection

5. Splitting into Dependent and Independent Variables: -

In the stage of dataset splitting into train and test, I split dataset by using 'train_test_split' function' from library sklearn.model_selection. here we divided our dataset into 80% for training and 20% for testing, and the way & reason for splitting the dataset in ratio of 20:80 is that it will probably minimize the losses and enhanced the accuracy.

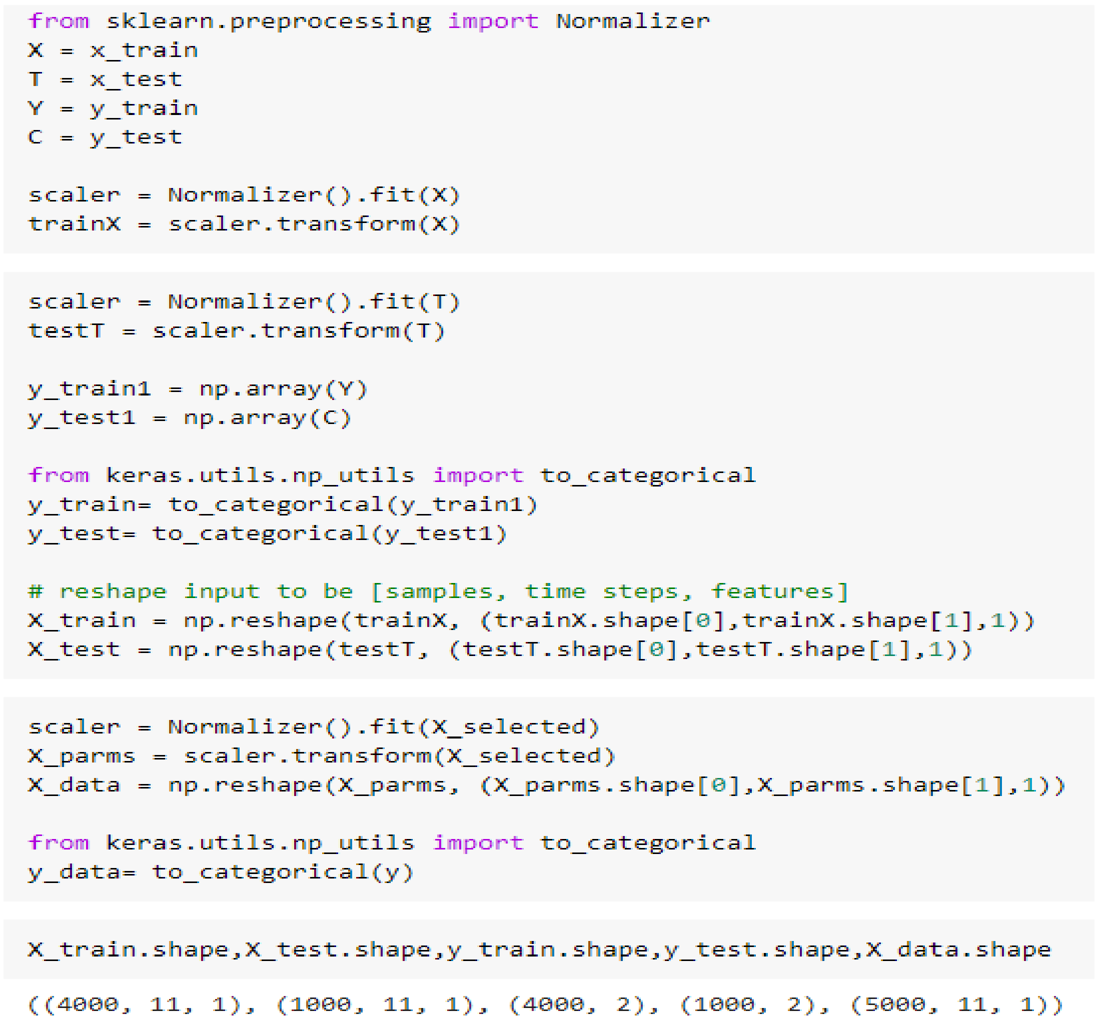


```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(X_selected,y,test_size = 0.2)
```

Fig. 7 Split into Train and Test

6. Normalizing

In normalisation we just make a copy of x_train, y_train. We are using normalising because We must normalise all the things into scaled formed. Scaled form means (0 to 1). So, all the things will be normalised into it. We have to normalise only x, because x is dependent variable which we are passing the thing.
We must convert the categorical of data using this y so this keras function automatically convert the y into y_data = categorical value. So, after that scaling, we have to convert the y data into categorical form. Then in the x_train.shape, I just print all the shape here.



```
from sklearn.preprocessing import Normalizer
X = x_train
T = x_test
Y = y_train
C = y_test

scaler = Normalizer().fit(X)
trainX = scaler.transform(X)
```

```
scaler = Normalizer().fit(T)
testT = scaler.transform(T)

y_train1 = np.array(Y)
y_test1 = np.array(C)

from keras.utils.np_utils import to_categorical
y_train= to_categorical(y_train1)
y_test= to_categorical(y_test1)

# reshape input to be [samples, time steps, features]
X_train = np.reshape(trainX, (trainX.shape[0],trainX.shape[1],1))
X_test = np.reshape(testT, (testT.shape[0],testT.shape[1],1))
```

```
scaler = Normalizer().fit(X_selected)
X_parms = scaler.transform(X_selected)
X_data = np.reshape(X_parms, (X_parms.shape[0],X_parms.shape[1],1))

from keras.utils.np_utils import to_categorical
y_data= to_categorical(y)
```

```
X_train.shape,X_test.shape,y_train.shape,y_test.shape,X_data.shape
```

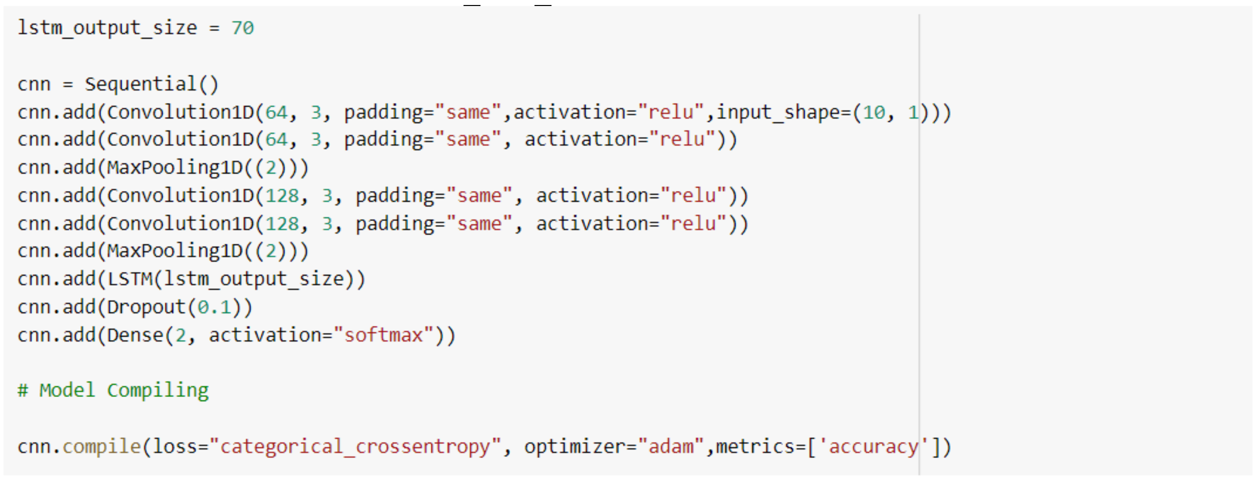
```
((4000, 11, 1), (1000, 11, 1), (4000, 2), (1000, 2), (5000, 11, 1))
```

Fig.8 Normalizing

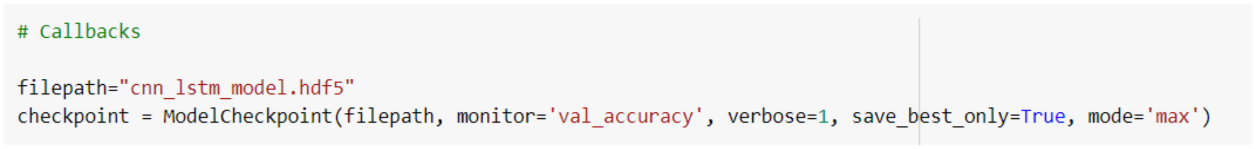
7. DEEP CNN – LSTM MODEL

So this is the deep CNN-LSTM model . Lstm_output_size=70 (So based upon that Each 70 Output will be passed to as a input layer. So first iteration will take 70 and next iteration will take 70 into another input for the Neural network. So here we are Hybriding that LSTM into cnn for our better Result and accuracy. So here you can see that 64 units and 3 hidden layers and padding = same. Activation function we are using 'relu'. Here we are using convolution neural network and in the next line 2 maxpooling layer and LSTM layer and Dropout. Based upon that we have to predict whether it's yes or no so for the reason we are using dense = 2 So ultimately we are using the activation= softwax. So this all are the model declaration and

we have to compile it So in the last line this is the model compiling here and optimizer= ‘adam’. Adam will be the best for our process.
so i have saved the model here "cnn_lstm_model.hdf5”.



```
lstm_output_size = 70

cnn = Sequential()
cnn.add(Convolution1D(64, 3, padding="same",activation="relu",input_shape=(10, 1)))
cnn.add(Convolution1D(64, 3, padding="same", activation="relu"))
cnn.add(MaxPooling1D((2)))
cnn.add(Convolution1D(128, 3, padding="same", activation="relu"))
cnn.add(Convolution1D(128, 3, padding="same", activation="relu"))
cnn.add(MaxPooling1D((2)))
cnn.add(LSTM(lstm_output_size))
cnn.add(Dropout(0.1))
cnn.add(Dense(2, activation="softmax"))

# Model Compiling

cnn.compile(loss="categorical_crossentropy", optimizer="adam",metrics=['accuracy'])
```

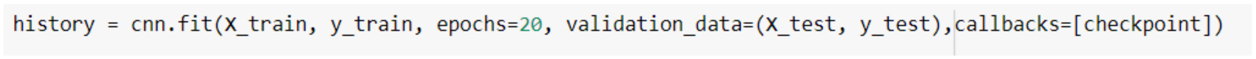


```
# Callbacks

filepath="cnn_lstm_model.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')
```

Fig.9 Deep CNN LSTM model

8. Model Fitting

In the model fitting "cnn.fit is loaded here. so you can see in each and every iteration the exact values of loss , accuracy and validation accuracy. If the accuracy does not improve means it will stop the iteration. So in the last, i put value 20 in epoch so it will run till 20 iteration. So here I saved the model and loaded the model here.

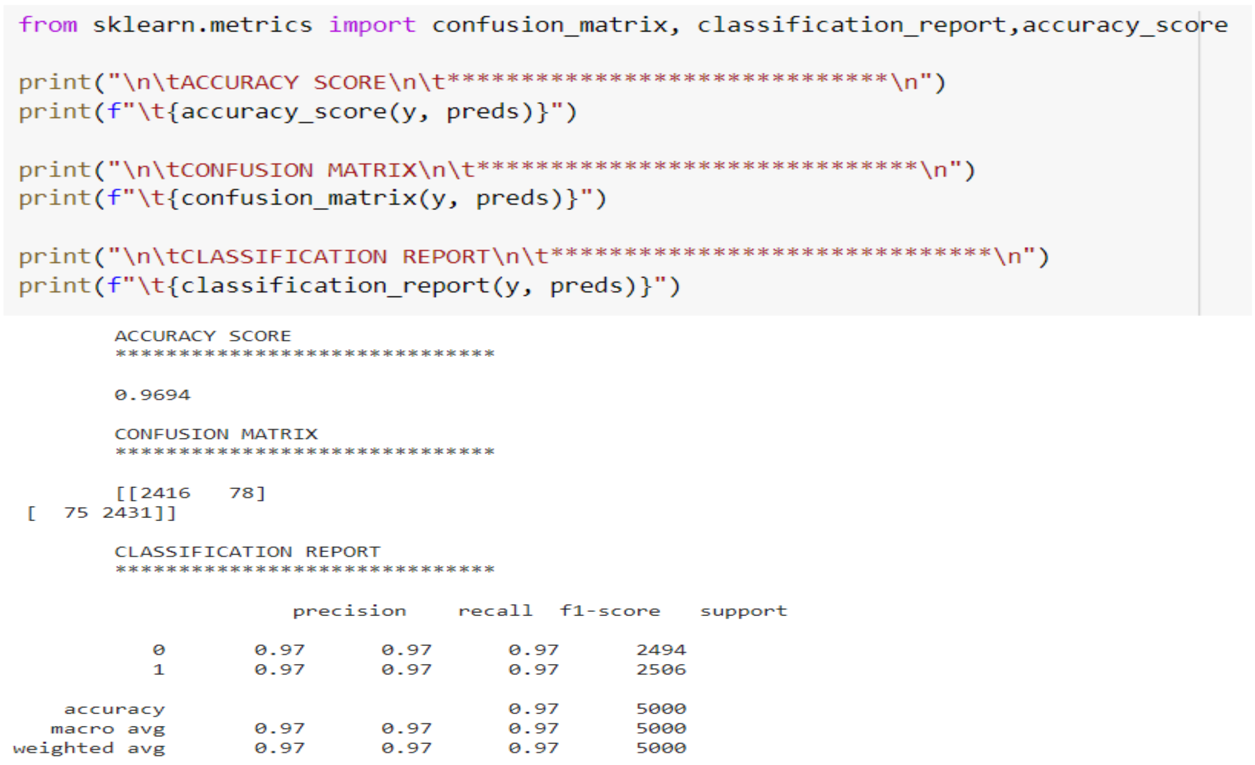


```
history = cnn.fit(X_train, y_train, epochs=20, validation_data=(X_test, y_test),callbacks=[checkpoint])
```

Fig. 10 Model Fitting

9. Performance Metrics -

The accuracy result was acquired with the following function 'accuracy_score' and I imported this library from 'sklearn.metrics' but here i imported many other functions like classification_report & confusion_matrix and we printed the all three results one by one. again, we imported confusion_matrix function from sklearn.metrics library, we acquired the confusion matrix by using the same scikit learn library.



```python
from sklearn.metrics import confusion_matrix, classification_report,accuracy_score

print("\n\tACCURACY SCORE\n\t******************************\n")
print(f"\t{accuracy_score(y, preds)}")

print("\n\tCONFUSION MATRIX\n\t******************************\n")
print(f"\t{confusion_matrix(y, preds)}")

print("\n\tCLASSIFICATION REPORT\n\t******************************\n")
print(f"\t{classification_report(y, preds)}")
```

```
        ACCURACY SCORE
        ******************************

        0.9694

        CONFUSION MATRIX
        ******************************

        [[2416   78]
 [  75 2431]]

        CLASSIFICATION REPORT
        ******************************

              precision    recall  f1-score   support

           0       0.97      0.97      0.97      2494
           1       0.97      0.97      0.97      2506

    accuracy                           0.97      5000
   macro avg       0.97      0.97      0.97      5000
weighted avg       0.97      0.97      0.97      5000
```

Fig.11. Performance Metrics