National College of Ireland

Configuration Manual

MSc Internship

Programme Name

Yogesh Parmar

Student ID: X18176402

School of Computing

National College of Ireland

Supervisor: Mr. Vikas Sahni

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	**Yogesh Bharat Parmar**		
Student ID:	**X18176402**		
Programme:	**CyberSecurity**	**Year:**	**2020**
Module:	**MSc Internship**		
Lecturer:	**Mr. Vikas Sahni**		
Submission Due Date:	**17/08/2020**		
Project Title:	**Windows Portable Executor Malware detection using Deep learning approaches**		
Word Count:	**706**	**Page Count:**	**18**

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: Yogesh Bharat Parmar

Date: 17/08/2020

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

Yogesh Bharat Parmar
X18176402

1. Introduction

This manual is developed to clarify the steps for running the research project in detail and also specifies the configuration of the machine which is used to build and run the models. The steps include downloading and installation of the appropriate software and packages, and the minimum necessary configuration for the smooth running of the project.

2. System Configuration

2.1. Hardware

The hardware configuration of the computer conducted for the research is as specified below:

Processor: Intel i5 – 5200U CPU @ 2.20GHz
RAM: 12 GB
Storage: 500 GB HDD
Operating System: 64-bit operating system, Windows 10 Home

Hardware configuration recommended:
Processor: Dual-Core Intel i5 or equivalent
RAM: 6 GB
Storage: 128 GB HDD

2.2. Software

Microsoft Excel: Used for data management, data discovery, and exploratory plots.
Jupyter Notebook: Used for data loading, cleaning and pre-processing of the data. It is also used for model building as well as for its evaluation. Implementation of python was also used.

3. Download and Installation

3.1. Installing latest version of Python

The latest version is recommended for installation of python which must be installed based on the operating system. Figure 1 shows a glimpse where you can download Python from https://www.python.org/downloads/

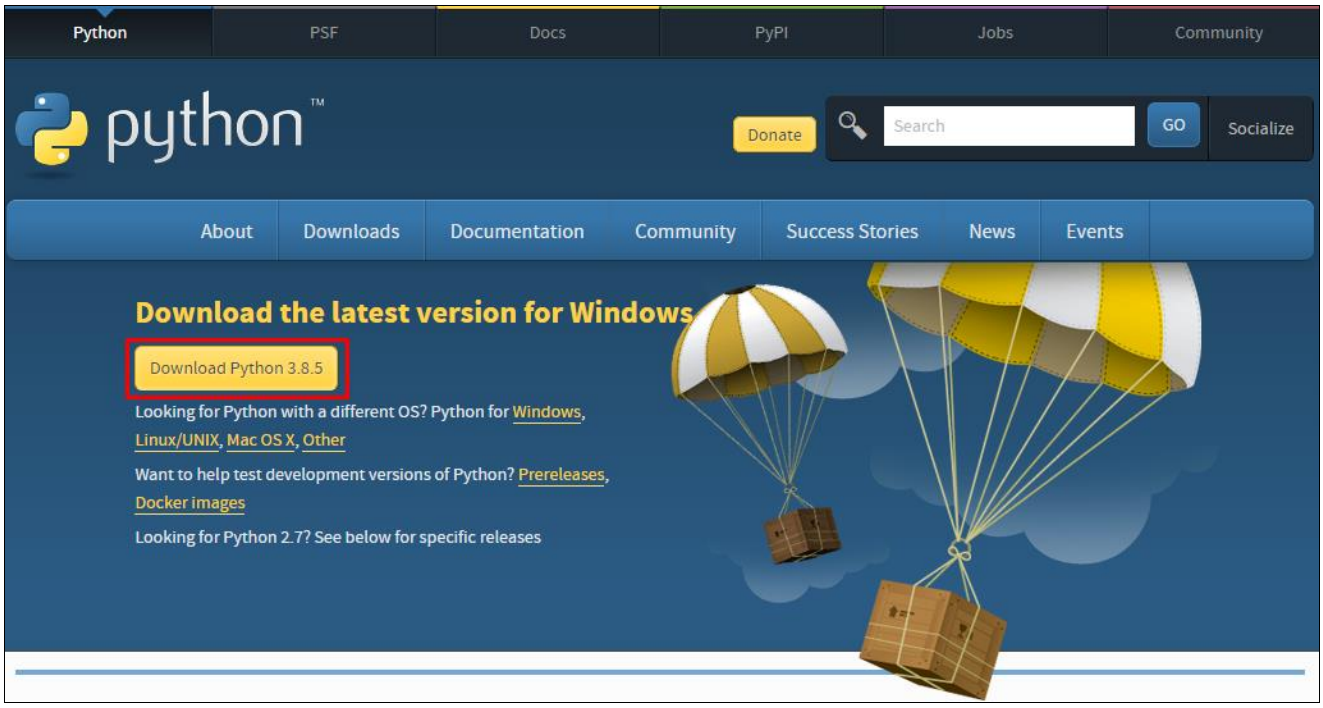


Figure 1: latest version of python to download from official website.

3.2. Installing latest version of Anaconda

Anaconda3 is an open source package manager developed to improve data analysis and the machine learning project. It offers numerous python-based IDEs that are user-friendly that can be used for application creation and performance visualization. Of the IDEs available for download, the most prominent is Jupyter Notebook.

Anaconda3 is an open source which can be downloaded from *https://anaconda.com/distribution*

Figure 2: latest version of Anaconda3 to download from official website.

Upon effective deployment, the Anaconda Navigator will view a window as shown in Figure 3 from which select the appropriate IDE for development. In the current research the Jupyter Notebook has been used.

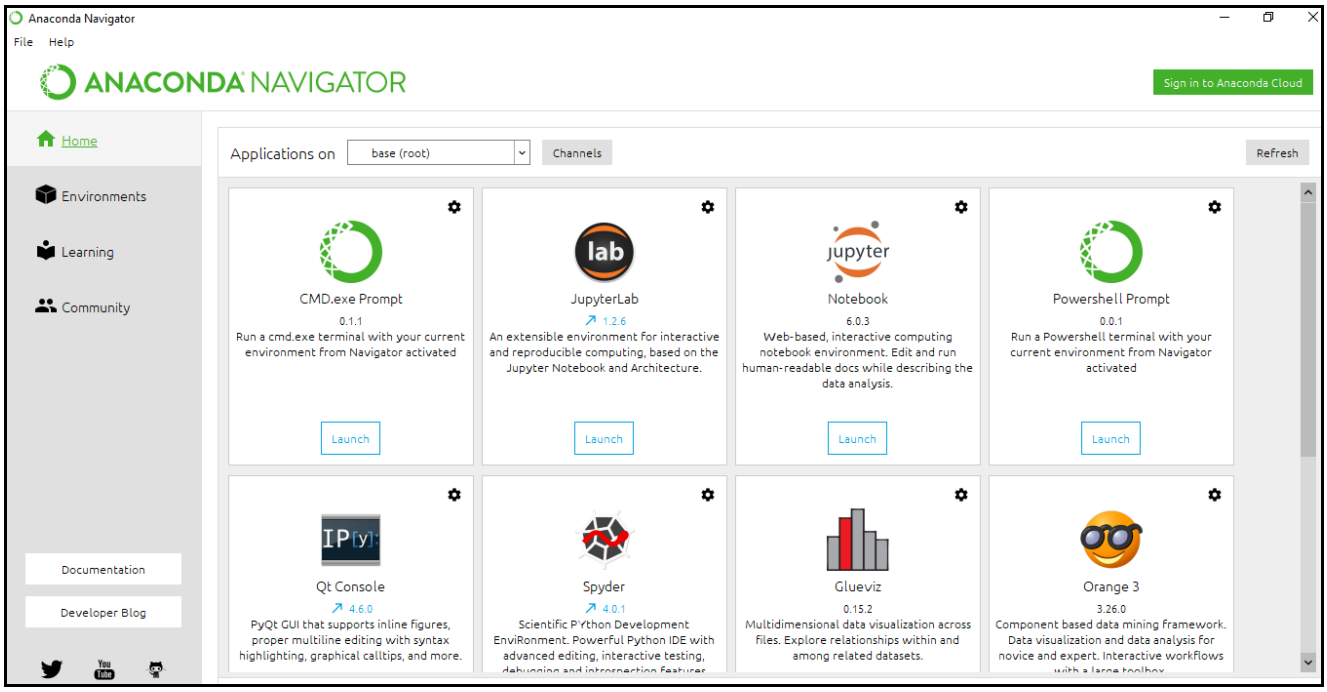


Figure 3: Anaconda Navigator

Clicking onto the Launch button for Jupyter Notebook from the above Anaconda Navigator, we get a page showing us which .ipynb file to open as shown in figure 4. If starting a new developing page just go to new on the right side and click on button script and will start with the new page.

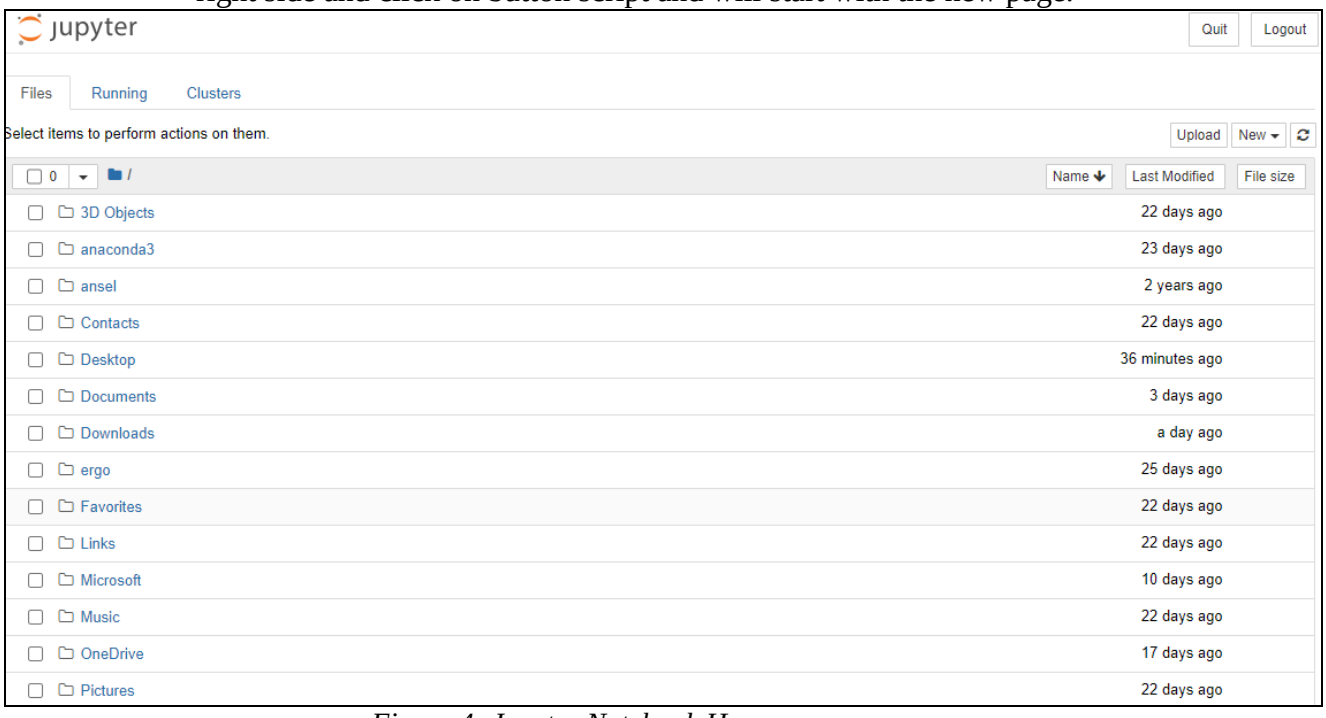


Figure 4: Jupyter Notebook Home page

4. Project Development

Step 1. Importing necessary python libraries for dataset pre-processing

```
#Required Libraries

import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.decomposition import IncrementalPCA
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
import tensorflow as tf
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense
from keras.optimizers import Adam
from keras.layers import Flatten
from keras.layers import TimeDistributed
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D
from keras.layers import Bidirectional
from keras.layers.core import Dropout
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder,MinMaxScaler
from sklearn.metrics import recall_score,precision_score,f1_score
from keras.utils import to_categorical
from sklearn.metrics import confusion_matrix
```

Figure 5: Python Libraries for research

Step 2. Loading the dataset and pre-processing dataset

Loading and preprocessing data

```
#Reading Data

data = pd.read_csv('dataset.csv',header=None)
```

```
#Sneak-peak at data

print('Shape of Data:',data.shape)
data.head()
```

```
Shape of Data: (199970, 487)
```

	0	1	2	3	4	5	6	7	8	9	...	477	478	479	480	481	482	483	484	485	486
0	pe-malicious	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.385031	0.60	0.40	0.565036	0.054403
1	pe-malicious	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.695652	0.20	0.20	0.372974	0.030327
2	pe-malicious	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.163088	1.00	1.00	0.979375	0.203325
3	pe-malicious	0.0	1.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.925532	0.25	0.25	0.648750	0.000407
4	pe-malicious	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.220399	1.00	1.00	0.979375	0.203325

5 rows × 487 columns

Figure 6: loading dataset and pre-processing the data [3]

Step 3. Dropping label column from data, Scaling down features in range, Distribution of label and Converting label to categorical.

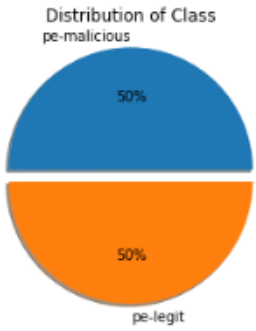
```
#dropping label column from data

label = data[0]
labels=label.unique()
data.drop(0,inplace=True,axis=1)
```

```
#Scaling down features in range [0,1]

scaler = MinMaxScaler(feature_range=[0, 1])
data = scaler.fit_transform(data)
```

```
#Distribution of Label
plt.title('Distribution of Class')
plt.pie(label.value_counts().values,labels = label.value_counts().index,shadow=True,autopct= '%1.00f%%',explode=(0.1,0))
plt.show()
```




```
#Converting label to categorical i.e. [0,1] or [1,0]
lbl_enc = LabelEncoder()
label = lbl_enc.fit_transform(label)
label = to_categorical(label)
```

Figure 7: Pre-processing the data

Step 4. Feature Selection

```
reg=[]
for n_components in range(1,data.shape[1]):
    print('n_components = ', n_components)

    pca = PCA(n_components = n_components)
    pca.fit(data)
    reg.append(np.sum(pca.explained_variance_ratio_))
```

Figure 8: Feature selection

```
plt.figure(figsize=(11,6))
plt.plot(x,reg,'-s',color='r',markersize=3)
plt.plot([78,78],[0.264,1],'--')
plt.plot([85,85],[0.264,1],'--')
plt.plot([92,92],[0.264,1],'--')
plt.xlim((0,486))
```

```
(0, 486)
```

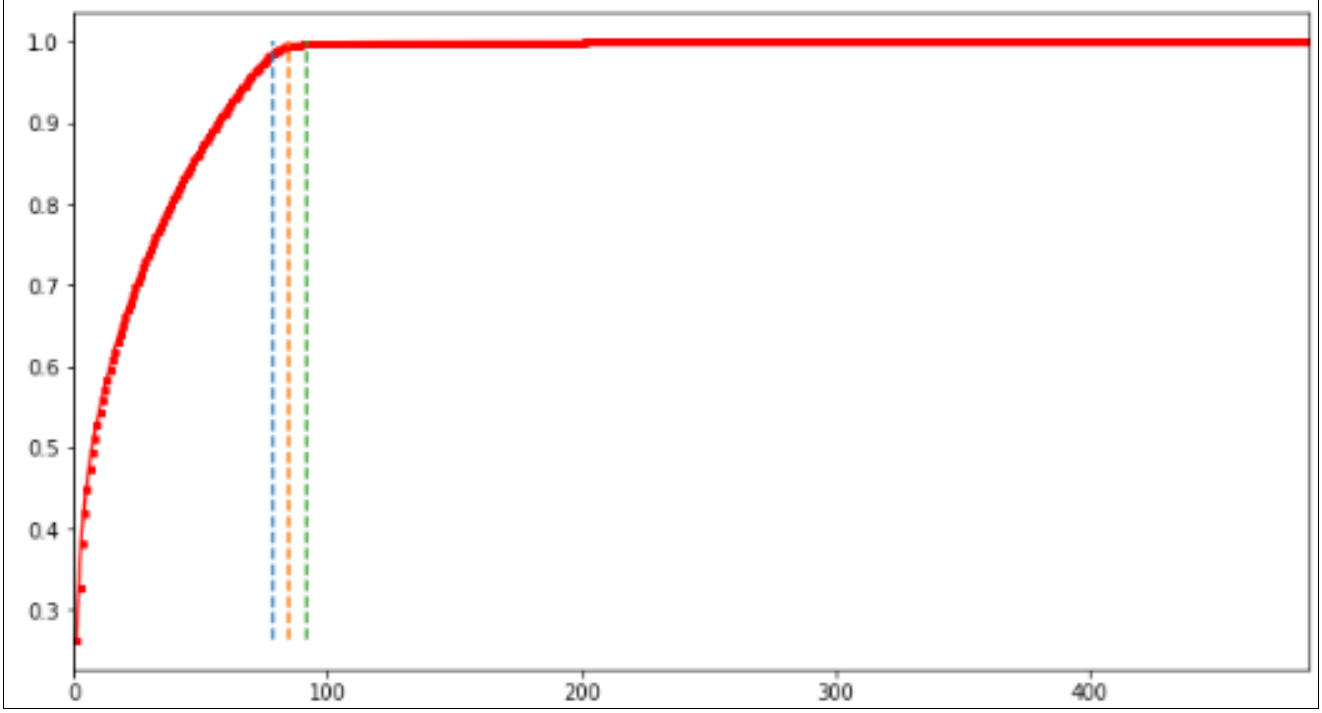


Figure 9: Plotting Feature Selection

Step 5. Feature Extraction

```
pca = PCA(n_components = 85)
data_reduced = pca.fit_transform(data)
np.sum(pca.explained_variance_ratio_)
```

```
0.9875150984650686
```

```
data_reduced.shape
```

```
(199970, 85)
```

```
data_reduced = pd.DataFrame(data_reduced)
corr = data_reduced.corr()
plt.figure(figsize=(15,12))
sns.heatmap(corr)
plt.show()
# We can see that none of the features are not correlated
```

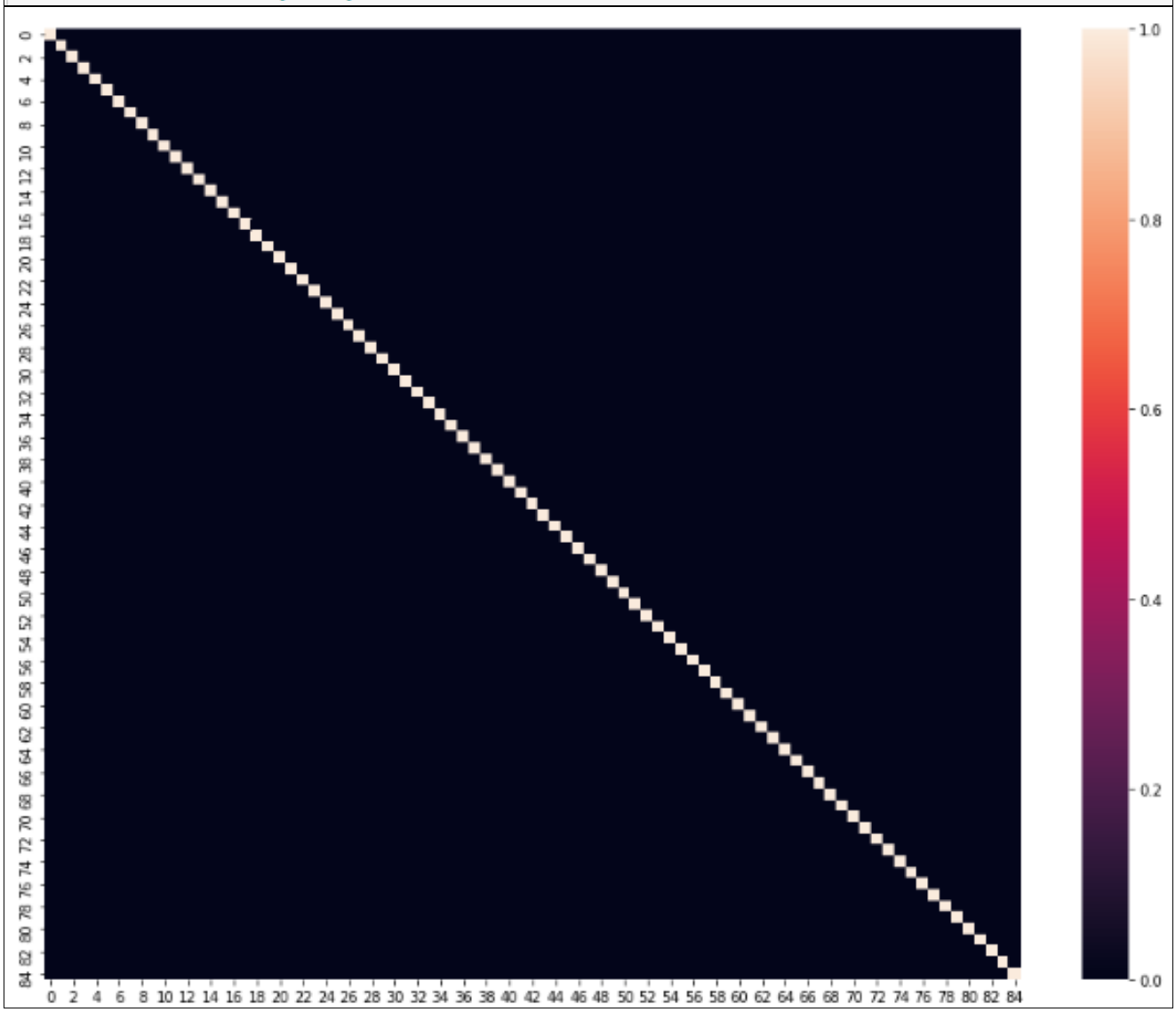


Figure 10: Checking for feature correlation

Step 6. Splitting the dataset into Training set and Validation set in the ratio of 70:30

```
# Splitting dataset into training set and validation set in ratio 70:30

X_train,X_test, Y_train, Y_test = train_test_split(data_reduced,label,stratify=label,test_size=0.30,shuffle=True,random_state=42)
```

Figure 11: Splitting dataset into training and validation set

Step 7. Setting epochs

```
#Number of features
n_features = data_reduced.shape[1]

#number of steps per instance
n_steps= 1

n_outputs = label.shape[1]
epochs = 25
batch_size= 32
verbose= 1
```

Figure 12: Setting Epochs

Step 8. Wrapper for turning Tensorflow metrics into Keras metrics

```
# Wrapper for turning tensorflow metrics into keras metrics

def as_keras_metric(method):
    import functools
    from keras import backend as K
    @functools.wraps(method)
    def wrapper(self, args, **kwargs):
        value, update_op = method(self, args, **kwargs)
        K.get_session().run(tf.local_variables_initializer())
        with tf.control_dependencies([update_op]):
            value = tf.identity(value)
        return value
    return wrapper

# defining precision ,recall abd f_score
precision = as_keras_metric(tf.metrics.precision)
recall = as_keras_metric(tf.metrics.recall)
f1_score=as_keras_metric(tf.contrib.metrics.f1_score)
```

Figure 13: Wrapper for turning tensorflow metrics into keras metrics

Step 9. Re-shaping the data for CNN Model and defining it with epochs accuracy

```
# reshaping data to feed into CNN model

X_train = np.array(X_train).reshape((X_train.shape[0],  n_steps, n_features))
X_test = np.array(X_test).reshape((X_test.shape[0] , n_steps, n_features))
```

```
# define model
model = Sequential()
model.add(Conv1D(filters=64, kernel_size=1, activation='relu', input_shape=(n_steps, n_features)))
model.add(Conv1D(filters=64, kernel_size=1, activation='relu'))
model.add(Dropout(0.5))
model.add(MaxPooling1D(pool_size=1))
model.add(Flatten())
model.add(Dense(256,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(124,activation='relu'))
model.add(Dense(n_outputs,activation='softmax'))
opt = Adam(lr = 0.001)
model.compile(loss="binary_crossentropy", optimizer=opt,metrics=['accuracy',precision,recall,f1_score])
# fit model
history_cnn = model.fit(X_train, Y_train,validation_data=(X_test,Y_test), epochs=epochs, verbose=verbose,batch_size= batch_size)
```

```
Train on 139979 samples, validate on 59991 samples
Epoch 1/25
139979/139979 [==============================] - 49s 347us/step - loss: 0.4436 - acc: 0.7921 - precision: 0.4999 - recall: 0.99
98 - f1_score: 0.7603 - val_loss: 0.3656 - val_acc: 0.8477 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8080
Epoch 2/25
139979/139979 [==============================] - 41s 293us/step - loss: 0.3581 - acc: 0.8483 - precision: 0.5000 - recall: 1.00
00 - f1_score: 0.8222 - val_loss: 0.3415 - val_acc: 0.8550 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8311
Epoch 3/25
139979/139979 [==============================] - 41s 290us/step - loss: 0.3234 - acc: 0.8667 - precision: 0.5000 - recall: 1.00
00 - f1_score: 0.8377 - val_loss: 0.2823 - val_acc: 0.8910 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8445
Epoch 4/25
139979/139979 [==============================] - 41s 293us/step - loss: 0.3030 - acc: 0.8771 - precision: 0.5000 - recall: 1.00
00 - f1_score: 0.8498 - val_loss: 0.2660 - val_acc: 0.8962 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8545
Epoch 5/25
139979/139979 [==============================] - 41s 294us/step - loss: 0.2876 - acc: 0.8850 - precision: 0.5000 - recall: 1.00
00 - f1_score: 0.8583 - val_loss: 0.2781 - val_acc: 0.9015 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8618
Epoch 6/25
139979/139979 [==============================] - 41s 294us/step - loss: 0.2763 - acc: 0.8910 - precision: 0.5000 - recall: 1.00
00 - f1_score: 0.8647 - val_loss: 0.2539 - val_acc: 0.9084 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8675
Epoch 7/25
139979/139979 [==============================] - 41s 296us/step - loss: 0.2699 - acc: 0.8933 - precision: 0.5000 - recall: 1.00
00 - f1_score: 0.8699 - val_loss: 0.2486 - val_acc: 0.9041 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8719
Epoch 8/25
139979/139979 [==============================] - 41s 293us/step - loss: 0.2612 - acc: 0.8969 - precision: 0.5000 - recall: 1.00
00 - f1_score: 0.8737 - val_loss: 0.2617 - val_acc: 0.8994 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8754
Epoch 9/25
139979/139979 [==============================] - 41s 294us/step - loss: 0.2576 - acc: 0.8986 - precision: 0.5000 - recall: 1.00
00 - f1_score: 0.8768 - val_loss: 0.2517 - val_acc: 0.9090 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8783
Epoch 10/25
139979/139979 [==============================] - 41s 294us/step - loss: 0.2530 - acc: 0.9008 - precision: 0.5000 - recall: 1.00
00 - f1_score: 0.8796 - val_loss: 0.2582 - val_acc: 0.9092 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8808
Epoch 11/25
139979/139979 [==============================] - 41s 296us/step - loss: 0.2482 - acc: 0.9029 - precision: 0.5000 - recall: 1.00
00 - f1_score: 0.8820 - val_loss: 0.2394 - val_acc: 0.9160 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8831
Epoch 12/25
139979/139979 [==============================] - 41s 295us/step - loss: 0.2428 - acc: 0.9056 - precision: 0.5000 - recall: 1.00
00 - f1_score: 0.8843 - val_loss: 0.2595 - val_acc: 0.9111 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8853
Epoch 13/25
139979/139979 [==============================] - 41s 294us/step - loss: 0.2396 - acc: 0.9069 - precision: 0.5000 - recall: 1.00
00 - f1_score: 0.8862 - val_loss: 0.2650 - val_acc: 0.9027 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8869
Epoch 14/25
139979/139979 [==============================] - 41s 295us/step - loss: 0.2350 - acc: 0.9088 - precision: 0.5000 - recall: 1.00
00 - f1_score: 0.8877 - val_loss: 0.2587 - val_acc: 0.9067 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8884
Epoch 15/25
139979/139979 [==============================] - 41s 295us/step - loss: 0.2329 - acc: 0.9094 - precision: 0.5000 - recall: 1.00
00 - f1_score: 0.8891 - val_loss: 0.2841 - val_acc: 0.8990 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8896
Epoch 16/25
139979/139979 [==============================] - 41s 293us/step - loss: 0.2287 - acc: 0.9114 - precision: 0.5000 - recall: 1.00
00 - f1_score: 0.8902 - val_loss: 0.2355 - val_acc: 0.9186 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8909
Epoch 17/25
139979/139979 [==============================] - 41s 294us/step - loss: 0.2278 - acc: 0.9115 - precision: 0.5000 - recall: 1.00
00 - f1_score: 0.8916 - val_loss: 0.2343 - val_acc: 0.9199 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8923
Epoch 18/25
139979/139979 [==============================] - 41s 293us/step - loss: 0.2248 - acc: 0.9136 - precision: 0.5000 - recall: 1.00
00 - f1_score: 0.8930 - val_loss: 0.2375 - val_acc: 0.9211 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8936
Epoch 19/25
139979/139979 [==============================] - 41s 295us/step - loss: 0.2237 - acc: 0.9143 - precision: 0.5000 - recall: 1.00
00 - f1_score: 0.8942 - val_loss: 0.2366 - val_acc: 0.9169 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8948
Epoch 20/25
139979/139979 [==============================] - 41s 295us/step - loss: 0.2209 - acc: 0.9155 - precision: 0.5000 - recall: 1.00
00 - f1_score: 0.8953 - val_loss: 0.2377 - val_acc: 0.9200 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8959
Epoch 21/25
139979/139979 [==============================] - 41s 296us/step - loss: 0.2198 - acc: 0.9162 - precision: 0.5000 - recall: 1.00
00 - f1_score: 0.8964 - val_loss: 0.2467 - val_acc: 0.9145 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8969
Epoch 22/25
139979/139979 [==============================] - 41s 295us/step - loss: 0.2175 - acc: 0.9170 - precision: 0.5000 - recall: 1.00
00 - f1_score: 0.8973 - val_loss: 0.2257 - val_acc: 0.9211 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8978
Epoch 23/25
139979/139979 [==============================] - 41s 295us/step - loss: 0.2147 - acc: 0.9183 - precision: 0.5000 - recall: 1.00
00 - f1_score: 0.8983 - val_loss: 0.2321 - val_acc: 0.9243 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8988
Epoch 24/25
139979/139979 [==============================] - 41s 295us/step - loss: 0.2157 - acc: 0.9175 - precision: 0.5000 - recall: 1.00
00 - f1_score: 0.8992 - val_loss: 0.2353 - val_acc: 0.9175 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8996
Epoch 25/25
139979/139979 [==============================] - 41s 295us/step - loss: 0.2124 - acc: 0.9198 - precision: 0.5000 - recall: 1.00
00 - f1_score: 0.9001 - val_loss: 0.2113 - val_acc: 0.9291 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.9005
```

Figure 14: Re-shaping data to feed into CNN model with outcomes

Step 10. Re-shaping the data for RNN Model and defining it with epochs

```
# reshaping data to feed into RNN model

X_train = np.array(X_train).reshape((X_train.shape[0], n_steps, n_features))
X_test = np.array(X_test).reshape((X_test.shape[0] , n_steps, n_features))
```

```
# define model
model = Sequential()
model.add(LSTM(250, activation='relu', return_sequences=True,input_shape=( n_steps, n_features)))
model.add(LSTM(150, activation='relu', return_sequences=True,dropout=0.2))
model.add(LSTM(50, activation='relu',return_sequences=True))
model.add(Bidirectional(LSTM(50, activation='relu')))
model.add(Dense(n_outputs,activation='sigmoid'))
opt = Adam(lr = 0.001)
model.compile(loss="binary_crossentropy", optimizer=opt,metrics=['accuracy',precision,recall,f1_score])
# fit model
history_rnn = model.fit(X_train,Y_train,validation_data=(X_test,Y_test), epochs=epochs, verbose=verbose,batch_size=batch_size)
```

```
Train on 139979 samples, validate on 59991 samples
Epoch 1/25
139979/139979 [==============================] - 140s 1000us/step - loss: 0.4614 - acc: 0.7659 - precision: 0.4999 - recall: 0.
9998 - f1_score: 0.7433 - val_loss: 0.3701 - val_acc: 0.8234 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.791
5
Epoch 2/25
139979/139979 [==============================] - 129s 923us/step - loss: 0.3844 - acc: 0.8157 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8043 - val_loss: 0.3612 - val_acc: 0.8302 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8118
Epoch 3/25
139979/139979 [==============================] - 129s 924us/step - loss: 0.3582 - acc: 0.8346 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8168 - val_loss: 0.3168 - val_acc: 0.8718 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8228
Epoch 4/25
139979/139979 [==============================] - 133s 952us/step - loss: 0.3295 - acc: 0.8576 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8281 - val_loss: 0.3379 - val_acc: 0.8497 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8319
Epoch 5/25
139979/139979 [==============================] - 372s 3ms/step - loss: 0.3139 - acc: 0.8695 - precision: 0.5000 - recall: 1.000
0 - f1_score: 0.8354 - val_loss: 0.2818 - val_acc: 0.8898 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8396
Epoch 6/25
139979/139979 [==============================] - 131s 937us/step - loss: 0.2982 - acc: 0.8792 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8433 - val_loss: 0.2989 - val_acc: 0.8808 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8466
Epoch 7/25
139979/139979 [==============================] - 129s 924us/step - loss: 0.2881 - acc: 0.8851 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8494 - val_loss: 0.2745 - val_acc: 0.8876 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8521
Epoch 8/25
139979/139979 [==============================] - 1401s 10ms/step - loss: 0.2816 - acc: 0.8887 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8545 - val_loss: 0.3185 - val_acc: 0.8687 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8563
Epoch 9/25
139979/139979 [==============================] - 130s 926us/step - loss: 0.2729 - acc: 0.8921 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8580 - val_loss: 0.2534 - val_acc: 0.9030 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8602
Epoch 10/25
139979/139979 [==============================] - 128s 917us/step - loss: 0.2641 - acc: 0.8969 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8622 - val_loss: 0.2529 - val_acc: 0.8947 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8639
Epoch 11/25
139979/139979 [==============================] - 129s 921us/step - loss: 0.2594 - acc: 0.8986 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8656 - val_loss: 0.2426 - val_acc: 0.9089 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8672
Epoch 12/25
139979/139979 [==============================] - 129s 919us/step - loss: 0.2526 - acc: 0.9013 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8688 - val_loss: 0.2516 - val_acc: 0.8998 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8702
Epoch 13/25
139979/139979 [==============================] - 130s 926us/step - loss: 0.2481 - acc: 0.9034 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8714 - val_loss: 0.2587 - val_acc: 0.9023 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8726
Epoch 14/25
139979/139979 [==============================] - 132s 945us/step - loss: 0.2416 - acc: 0.9065 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8738 - val_loss: 0.2466 - val_acc: 0.9035 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8750
Epoch 15/25
139979/139979 [==============================] - 129s 919us/step - loss: 0.2382 - acc: 0.9077 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8761 - val_loss: 0.2215 - val_acc: 0.9158 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8772
Epoch 16/25
139979/139979 [==============================] - 128s 917us/step - loss: 0.2334 - acc: 0.9095 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8783 - val_loss: 0.2329 - val_acc: 0.9060 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8792
Epoch 17/25
139979/139979 [==============================] - 128s 917us/step - loss: 0.2284 - acc: 0.9118 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8802 - val_loss: 0.2188 - val_acc: 0.9135 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8811
Epoch 18/25
139979/139979 [==============================] - 130s 927us/step - loss: 0.2249 - acc: 0.9131 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8820 - val_loss: 0.2075 - val_acc: 0.9211 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8830
Epoch 19/25
139979/139979 [==============================] - 129s 924us/step - loss: 0.2220 - acc: 0.9145 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8839 - val_loss: 0.2079 - val_acc: 0.9201 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8847
Epoch 20/25
139979/139979 [==============================] - 129s 918us/step - loss: 0.2186 - acc: 0.9156 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8855 - val_loss: 0.2181 - val_acc: 0.9159 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8863
```

```
Epoch 21/25
139979/139979 [==============================] - 129s 922us/step - loss: 0.2164 - acc: 0.9164 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8870 - val_loss: 0.2028 - val_acc: 0.9228 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8878
Epoch 22/25
139979/139979 [==============================] - 130s 930us/step - loss: 0.2127 - acc: 0.9180 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8885 - val_loss: 0.1943 - val_acc: 0.9264 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8893
Epoch 23/25
139979/139979 [==============================] - 128s 917us/step - loss: 0.2087 - acc: 0.9200 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8900 - val_loss: 0.2044 - val_acc: 0.9215 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8907
Epoch 24/25
139979/139979 [==============================] - 139s 992us/step - loss: 0.2065 - acc: 0.9209 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8913 - val_loss: 0.1868 - val_acc: 0.9297 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8920
Epoch 25/25
139979/139979 [==============================] - 126s 902us/step - loss: 0.2041 - acc: 0.9208 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8926 - val_loss: 0.1868 - val_acc: 0.9297 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8932
```

Figure 15: Re-shaping data to feed into RNN model with outcomes

Step 11. Re-shaping the data for CONC-LSTM Model and defining it with epochs

```
# reshaping data to feed into CONV-LSTM model

X_train = np.array(X_train).reshape((X_train.shape[0], 1, n_steps, n_features))
X_test = np.array(X_test).reshape((X_test.shape[0] ,1, n_steps, n_features))
```

```
# define model
model = Sequential()
model.add(TimeDistributed(Conv1D(filters=128, kernel_size=1, activation='relu'), input_shape=(None, n_steps, n_features)))
model.add(TimeDistributed(MaxPooling1D(pool_size=1)))
model.add(TimeDistributed(Conv1D(filters=64, kernel_size=1, activation='relu')))
model.add(TimeDistributed(Conv1D(filters=64, kernel_size=1, activation='relu')))
model.add(TimeDistributed(Flatten()))
model.add(LSTM(250, activation='relu', return_sequences=True))
model.add(LSTM(150, activation='relu', return_sequences=True,dropout=0.2))
model.add(LSTM(128, activation='relu',return_sequences=True))
model.add(Bidirectional(LSTM(128, activation='relu')))
model.add(Dense(n_outputs,activation='sigmoid'))
opt = Adam(lr = 0.001)
model.compile(loss="binary_crossentropy", optimizer=opt,metrics=['accuracy',precision,recall,f1_score])
# fit model
history_conv = model.fit(X_train,Y_train,validation_data=(X_test,Y_test), epochs=epochs, verbose=verbose,batch_size=batch_size)
```

```
Train on 139979 samples, validate on 59991 samples
Epoch 1/25
139979/139979 [==============================] - 121s 863us/step - loss: 0.4597 - acc: 0.7671 - precision: 0.4999 - recall: 0.9
998 - f1_score: 0.7322 - val_loss: 0.3398 - val_acc: 0.8556 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.7949
0.4999 - recall: 0.9997 - f1_sco - ETA: 4s - loss: 0.4626 - acc: 0.7649
Epoch 2/25
139979/139979 [==============================] - 103s 736us/step - loss: 0.3296 - acc: 0.8605 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8165 - val_loss: 0.2904 - val_acc: 0.8825 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.831
7.5000 - recall: 1.0000 - f1_score:
Epoch 3/25
139979/139979 [==============================] - 103s 734us/step - loss: 0.2858 - acc: 0.8820 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8417 - val_loss: 0.2580 - val_acc: 0.8991 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8502
Epoch 4/25
139979/139979 [==============================] - 103s 739us/step - loss: 0.2606 - acc: 0.8946 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8567 - val_loss: 0.2475 - val_acc: 0.9030 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8622
- acc: 0.8922 - precision: 0.5000 - recall: 1.0000 - f1_score: 0. - ETA: 27s - loss: 0.2645 - acc: 0.8922 - precision: 0.5000 -
recall: - ETA: 7s - loss: 0.2615 - acc: 0.8940  - ETA: 4s - loss: 0.2611 - acc: 0.8942 - precision:  - ETA: 1s - loss: 0.2607 -
acc: 0.8945 - precision: 0.5000 - recall: 1
Epoch 5/25
139979/139979 [==============================] - 114s 817us/step - loss: 0.2416 - acc: 0.9045 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8669 - val_loss: 0.2613 - val_acc: 0.8862 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8704
Epoch 6/25
139979/139979 [==============================] - 113s 805us/step - loss: 0.2293 - acc: 0.9094 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8733 - val_loss: 0.2321 - val_acc: 0.9108 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8766
c
Epoch 7/25
139979/139979 [==============================] - 114s 814us/step - loss: 0.2166 - acc: 0.9159 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8794 - val_loss: 0.2315 - val_acc: 0.9039 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8820
Epoch 8/25
139979/139979 [==============================] - 116s 827us/step - loss: 0.2066 - acc: 0.9209 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8842 - val_loss: 0.1992 - val_acc: 0.9247 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8867
66 - acc: 0.9208 - precision: 0.5000 - recall: 1.0000 - f1_score:
```

```
Epoch 9/25
139979/139979 [==============================] - 111s 796us/step - loss: 0.1984 - acc: 0.9244 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8889 - val_loss: 0.1889 - val_acc: 0.9316 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8910
Epoch 10/25
139979/139979 [==============================] - 110s 788us/step - loss: 0.1909 - acc: 0.9273 - precision: 0.5000 - recall: 1.0
000 - f1_score: 0.8930 - val_loss: 0.2009 - val_acc: 0.9284 - val_precision: 0.5000 - val_recall: 1.0000 - val_f1_score: 0.8948
Epoch 11/25
139979/139979 [==============================] - 112s 799us/step - loss: 0.1859 - acc: 0.9297 - precision: 0.5001 - recall: 1.0
000 - f1_score: 0.8964 - val_loss: 0.2176 - val_acc: 0.9169 - val_precision: 0.5001 - val_recall: 1.0000 - val_f1_score: 0.8978
Epoch 12/25
139979/139979 [==============================] - 112s 797us/step - loss: 0.1816 - acc: 0.9316 - precision: 0.5001 - recall: 1.0
000 - f1_score: 0.8990 - val_loss: 0.1767 - val_acc: 0.9352 - val_precision: 0.5001 - val_recall: 1.0000 - val_f1_score: 0.9005
Epoch 13/25
139979/139979 [==============================] - 123s 880us/step - loss: 0.1773 - acc: 0.9336 - precision: 0.5002 - recall: 1.0
000 - f1_score: 0.9018 - val_loss: 0.1729 - val_acc: 0.9362 - val_precision: 0.5002 - val_recall: 1.0000 - val_f1_score: 0.9031
Epoch 14/25
139979/139979 [==============================] - 115s 818us/step - loss: 0.1733 - acc: 0.9349 - precision: 0.5002 - recall: 1.0
000 - f1_score: 0.9043 - val_loss: 0.1675 - val_acc: 0.9402 - val_precision: 0.5003 - val_recall: 1.0000 - val_f1_score: 0.9055
Epoch 15/25
139979/139979 [==============================] - 104s 740us/step - loss: 0.1707 - acc: 0.9361 - precision: 0.5005 - recall: 1.0
000 - f1_score: 0.9065 - val_loss: 0.1810 - val_acc: 0.9340 - val_precision: 0.5006 - val_recall: 1.0000 - val_f1_score: 0.9075
Epoch 16/25
139979/139979 [==============================] - 111s 790us/step - loss: 0.1665 - acc: 0.9373 - precision: 0.5007 - recall: 1.0
000 - f1_score: 0.9085 - val_loss: 0.1705 - val_acc: 0.9372 - val_precision: 0.5008 - val_recall: 1.0000 - val_f1_score: 0.9094
Epoch 17/25
139979/139979 [==============================] - 105s 748us/step - loss: 0.1645 - acc: 0.9386 - precision: 0.5008 - recall: 1.0
000 - f1_score: 0.9103 - val_loss: 0.1738 - val_acc: 0.9372 - val_precision: 0.5010 - val_recall: 1.0000 - val_f1_score: 0.9111
oss: 0.1649 - acc: 0.9383 - precision: 0.50 - ETA: 1s - loss: 0.1648 - acc: 0.9385 - precision: 0.5008 - recall: 1.0000 -
Epoch 18/25
139979/139979 [==============================] - 105s 752us/step - loss: 0.1603 - acc: 0.9404 - precision: 0.5012 - recall: 1.0
000 - f1_score: 0.9119 - val_loss: 0.1663 - val_acc: 0.9373 - val_precision: 0.5014 - val_recall: 1.0000 - val_f1_score: 0.9127
Epoch 19/25
139979/139979 [==============================] - 104s 745us/step - loss: 0.1583 - acc: 0.9411 - precision: 0.5015 - recall: 1.0
000 - f1_score: 0.9134 - val_loss: 0.1641 - val_acc: 0.9408 - val_precision: 0.5017 - val_recall: 1.0000 - val_f1_score: 0.9142
ETA: 2s - loss: 0.1584 - acc: 0.9410 - precision: 0.5015 - recall: 1.0000 - f1_score: 0.91 - ETA: 2s - loss: 0.1584 - acc: 0.94
10 - precision: 0.501
Epoch 20/25
139979/139979 [==============================] - 112s 800us/step - loss: 0.1578 - acc: 0.9409 - precision: 0.5020 - recall: 1.0
000 - f1_score: 0.9148 - val_loss: 0.1674 - val_acc: 0.9361 - val_precision: 0.5021 - val_recall: 1.0000 - val_f1_score: 0.9155
Epoch 21/25
139979/139979 [==============================] - 109s 782us/step - loss: 0.1527 - acc: 0.9426 - precision: 0.5023 - recall: 1.0
000 - f1_score: 0.9161 - val_loss: 0.1701 - val_acc: 0.9366 - val_precision: 0.5026 - val_recall: 1.0000 - val_f1_score: 0.9167
recision: 0.5023 - recall: 1.0000 - f1_score: 0.916 - ETA: 3s - loss: 0.1525 - acc: 0.9426 - precis
Epoch 22/25
139979/139979 [==============================] - 110s 788us/step - loss: 0.1514 - acc: 0.9434 - precision: 0.5028 - recall: 1.0
000 - f1_score: 0.9172 - val_loss: 0.1556 - val_acc: 0.9434 - val_precision: 0.5030 - val_recall: 1.0000 - val_f1_score: 0.9178
434 - precision: 0.5028 - recall: 1.0000 - f1_score:  - ETA: 2s - loss: 0.1515 - acc: 0.9434 - precision: 0.5028
Epoch 23/25
139979/139979 [==============================] - 110s 785us/step - loss: 0.1512 - acc: 0.9438 - precision: 0.5034 - recall: 1.0
000 - f1_score: 0.9184 - val_loss: 0.1667 - val_acc: 0.9383 - val_precision: 0.5038 - val_recall: 1.0000 - val_f1_score: 0.9189
Epoch 24/25
139979/139979 [==============================] - 111s 790us/step - loss: 0.1482 - acc: 0.9452 - precision: 0.5041 - recall: 1.0
000 - f1_score: 0.9194 - val_loss: 0.1689 - val_acc: 0.9444 - val_precision: 0.5045 - val_recall: 1.0000 - val_f1_score: 0.9200
Epoch 25/25
139979/139979 [==============================] - 111s 790us/step - loss: 0.1463 - acc: 0.9457 - precision: 0.5048 - recall: 1.0
000 - f1_score: 0.9205 - val_loss: 0.1596 - val_acc: 0.9418 - val_precision: 0.5051 - val_recall: 1.0000 - val_f1_score: 0.9210
```

Figure 16: Re-shaping data to feed into CONV-LSTM model with outcomes

Step 12. Analysing results

Analysing all the outcomes of CNN, RNN and CONV-LSTM for better comparison of all models.

All the models are made comparison with the range of *epochs+1*

```
x = [i for i in range(1,epochs+1)]
```

Figure 17: Mentioned the range of epochs

a. Accuracy Comparison with Training & Validation set

```
plt.figure(figsize=(15,5))
plt.subplot(121)
plt.title("Accuracy comparison")
plt.plot(x,history_cnn.history['acc'],label='CNN')
plt.plot(x,history_rnn.history['acc'],label='RNN')
plt.plot(x,history_conv.history['acc'],label='CONV-LSTM')
plt.legend()
plt.subplot(122)
plt.title("Val_Accuracy comparison")
plt.plot(x,history_cnn.history['val_acc'],label='CNN')
plt.plot(x,history_rnn.history['val_acc'],label='RNN')
plt.plot(x,history_conv.history['val_acc'],label='CONV-LSTM')
plt.legend()
plt.show()
```

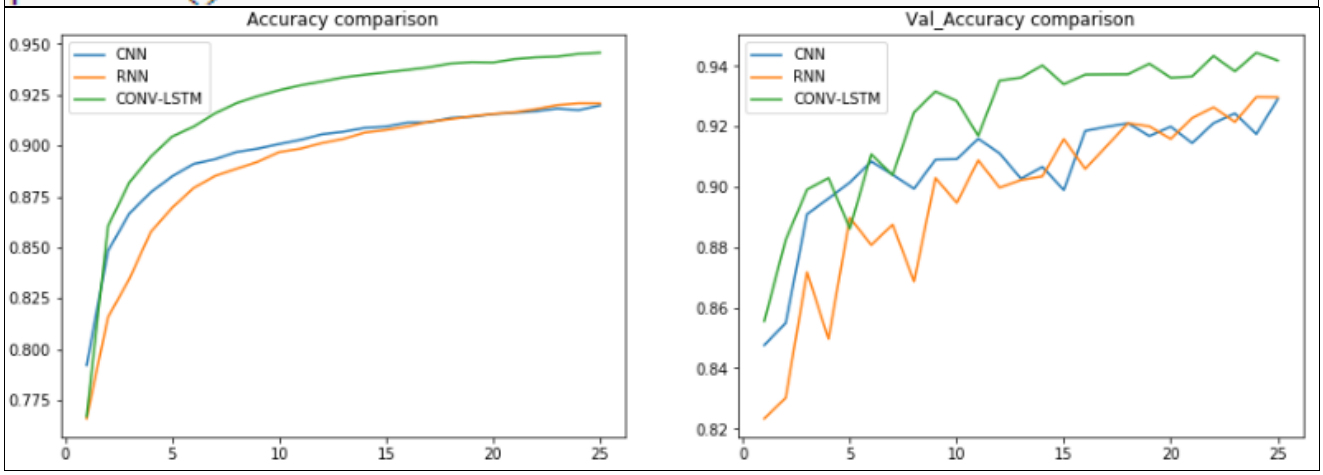


Figure 18: Accuracy Comparison and Value

b. Loss Comparison with Training & Validation set

```
plt.figure(figsize=(15,5))
plt.subplot(121)
plt.title("LOSS comparison")
plt.plot(x,history_cnn.history['loss'],label='CNN')
plt.plot(x,history_rnn.history['loss'],label='RNN')
plt.plot(x,history_conv.history['loss'],label='CONV-LSTM')
plt.legend()
plt.subplot(122)
plt.title("VAL_LOSS comparison")
plt.plot(x,history_cnn.history['val_loss'],label='CNN')
plt.plot(x,history_rnn.history['val_loss'],label='RNN')
plt.plot(x,history_conv.history['val_loss'],label='CONV-LSTM')
plt.legend()
plt.show()
```

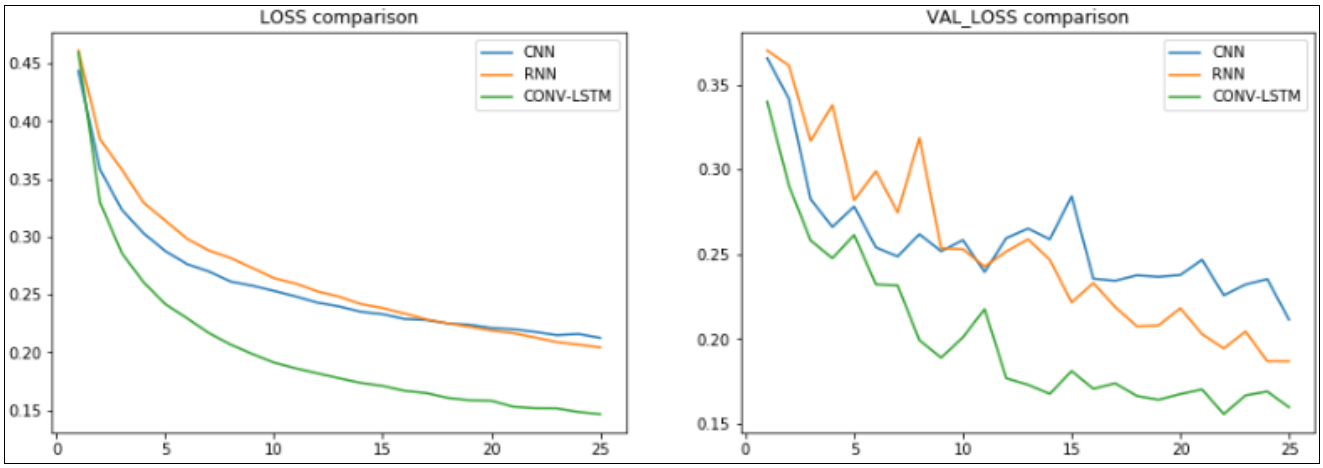


Figure 19: Loss Comparison and Value

c. Precision Comparison with Training & Validation set

```
plt.figure(figsize=(15,5))
plt.subplot(121)
plt.title("PRECISION comparison")
plt.plot(x,history_cnn.history['precision'],label='CNN')
plt.plot(x,history_rnn.history['precision'],label='RNN')
plt.plot(x,history_conv.history['precision'],label='CONV-LSTM')
plt.legend()
plt.subplot(122)
plt.title("VAL_PRECISION comparison")
plt.plot(x,history_cnn.history['val_precision'],label='CNN')
plt.plot(x,history_rnn.history['val_precision'],label='RNN')
plt.plot(x,history_conv.history['val_precision'],label='CONV-LSTM')
plt.legend()
plt.show()
```

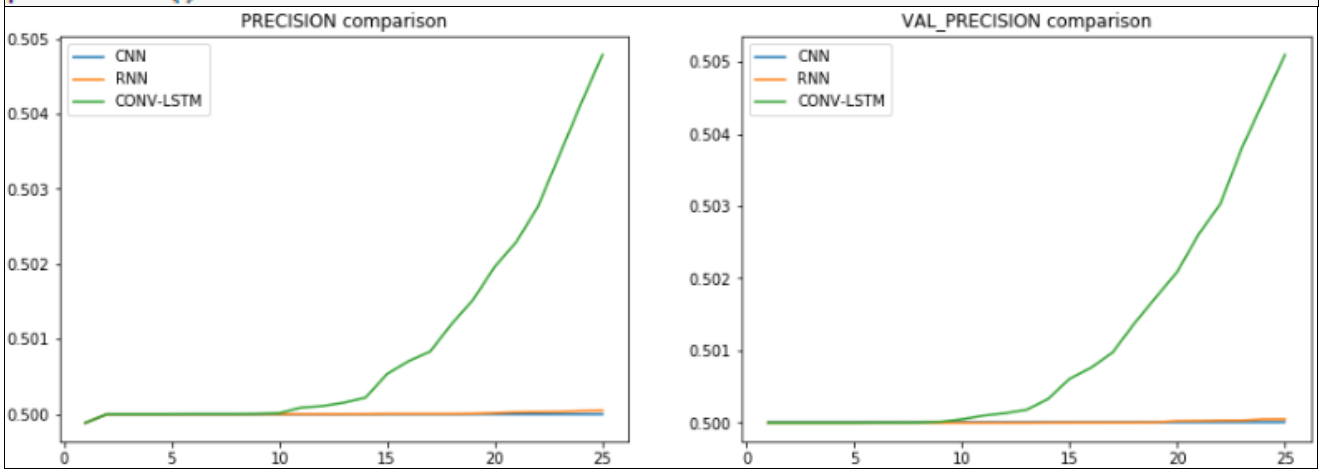


Figure 20: Precision Comparison and Value

d. Recall Comparison with Training & Validation set

```
plt.figure(figsize=(15,5))
plt.subplot(121)
plt.title("RECALL comparison")
plt.plot(x,history_cnn.history['recall'],label='CNN')
plt.plot(x,history_rnn.history['recall'],label='RNN')
plt.plot(x,history_conv.history['recall'],label='CONV-LSTM')
plt.legend()
plt.subplot(122)
plt.title("VAL_RECALL comparison")
plt.plot(x,history_cnn.history['val_recall'],label='CNN')
plt.plot(x,history_rnn.history['val_recall'],label='RNN')
plt.plot(x,history_conv.history['val_recall'],label='CONV-LSTM')
plt.legend()
plt.show()
```

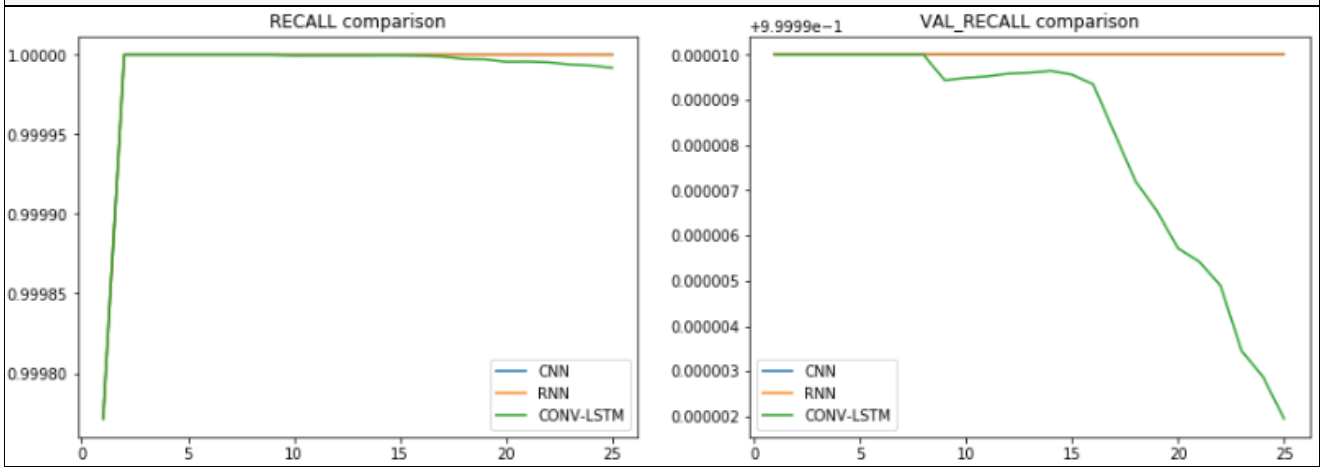


Figure 21: Recall Comparison and Value

e. F1 Score Comparison with Training & Validation set

```
plt.figure(figsize=(15,5))
plt.subplot(121)
plt.title("F1_SCORE comparison")
plt.plot(x,history_cnn.history['f1_score'],label='CNN')
plt.plot(x,history_rnn.history['f1_score'],label='RNN')
plt.plot(x,history_conv.history['f1_score'],label='CONV-LSTM')
plt.legend()
plt.subplot(122)
plt.title("VAL_F1_SCORE comparison")
plt.plot(x,history_cnn.history['val_f1_score'],label='CNN')
plt.plot(x,history_rnn.history['val_f1_score'],label='RNN')
plt.plot(x,history_conv.history['val_f1_score'],label='CONV-LSTM')
plt.legend()
plt.show()
```

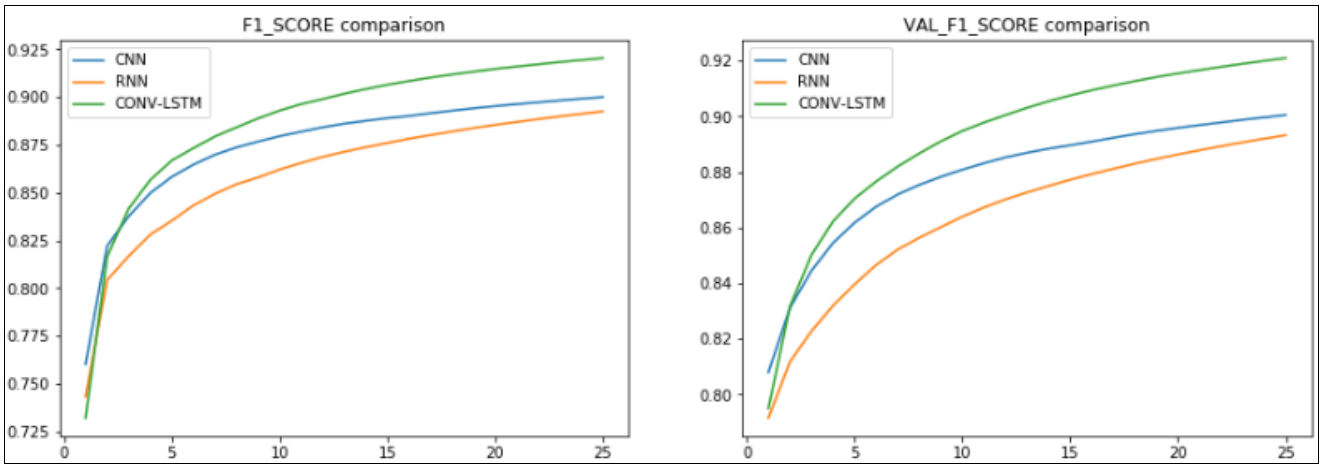


Figure 22: F1 Score Comparison and Value

Step 13. Confusion Matrix

```
y_pred = model.predict(X_test)
Y_pred = np.argmax(y_pred, axis=1)
y_test = np.argmax(Y_test,axis=1)
f_matrix = confusion_matrix(y_test, Y_pred)
plt.title('Confusion Matrix of CONV-LSTM\n')
sns.heatmap(f_matrix/np.sum(f_matrix), annot=True,fmt='.2%',xticklabels=labels,yticklabels=labels,cmap='hot')
plt.show()
```

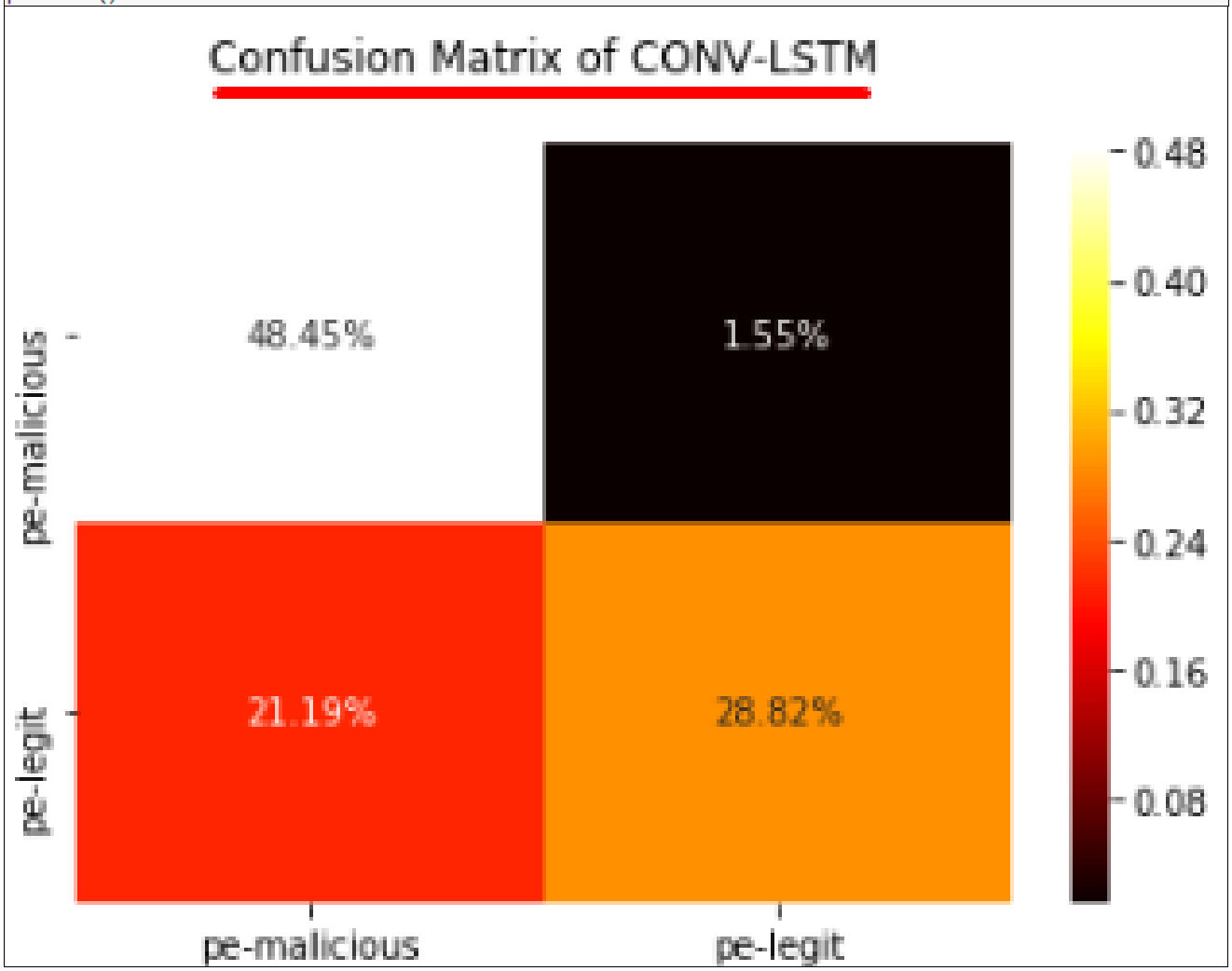


Figure 23: Confusion Matrix

References

[1] ‘Anaconda | Individual Edition’. Anaconda, anaconda.com/products/individual.

[2] ‘Download Python’. Python.Org, python.org/downloads.

[3] Dataset. kaggle.com/johndig90/winpe-ml.