Configuration Manual

MSc Research Project
Cybersecurity

Ayobami Jaiyeola
Student ID: X19220511

School of Computing
National College of Ireland

Supervisor: Michael Pantridge

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Jaiyeola Ayobami…………………………………………………………………………………
Student ID:	X19220511……………………………………………………………………………………..……
Programme:	Cybersecurity………………………………………… **Year:** …2021………………..
Module:	……MSc Academic Internship…………………………………………………………….………
Supervisor:	Michael Pantridge…………………………………………………………………….………
Submission Due Date:	…16/8/2021……………………………………………………………………………….………
Project Title:	Robust Intrusion Detection System Model for Internet of Things.………
Word Count:	………………529……………… **Page Count**…………………12……………………….……..

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: ………Jaiyeola Ayobami……………………………………………………………

Date: ……12/8/2021………………………………………………………………………………

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

Ayobami Jaiyeola
X19220511

1 Introduction

The steps and processes taken in the development of this project for an Intrusion Detection System for IoT is presented in this Configuration Manual. It describes all necessary settings and software tools needed to replicate the experimental setup for the project.

2 System Specification

The system configuration used in the project are:

- Operating System: Windows 10
- Processor: Intel Core i5 10th Gen
- Hard Drive: 1TB
- RAM: 8GB

3 Software Tools

Some of the software tools used to implement this project are:

- Python
- JavaScript
- Jupyter Notebook

3.1 Software Installation

This presents the processes taken in installing the tools used.

- Download and Installation of Python 3.9.6. The download link is https://www.python.org/downloads

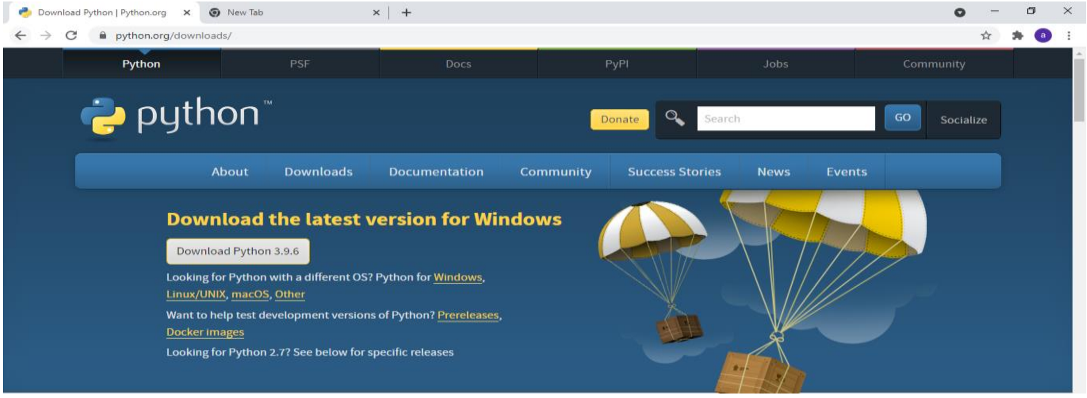

Fig 1: Python Download

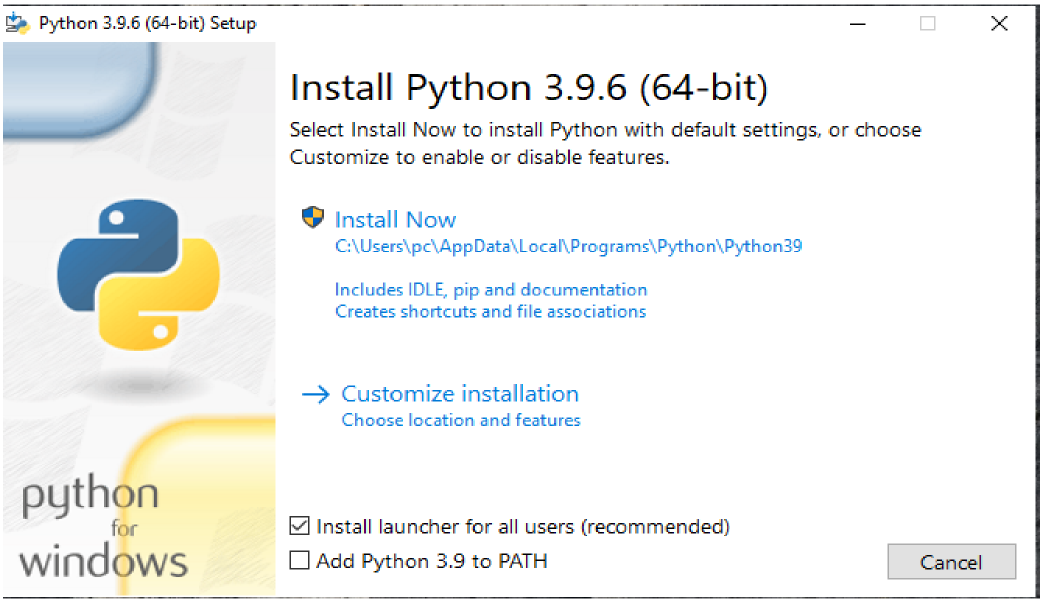

Fig 2: Python Installation

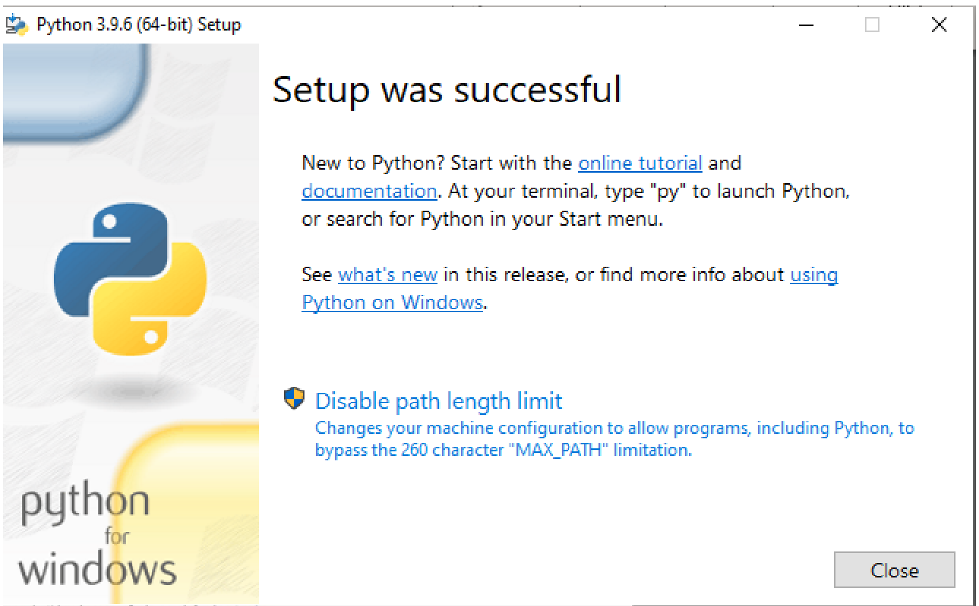

Fig 3: Completion of Installation

```
Command Prompt - py
Microsoft Windows [Version 10.0.19043.1165]
(c) Microsoft Corporation. All rights reserved.

C:\Users\pc>py#
'py#' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\pc>py
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Fig 4: Confirmation of Python Installation

4 Implementation

The libraries from python used in implementing this project:

- Scikit-Learn
- Keras
- Pandas
- Pickle
- Numpy

```
In [1]: from google.colab import drive
        drive.mount('/content/drive')

Mounted at /content/drive
```

Data is contained in 8 different CSV files, each containing different attack data at different times. So first thing we must do is merge all the data from files into one pandas DataFrame.

```
In [2]: import pandas as pd
        import glob

In [3]: # Saving all .csv files in folder to list.
        path = "/content/drive/MyDrive/MachineLearning/iot_ids/data/"
        files = [file for file in glob.glob(path + "**/*.csv", recursive=True)]

In [4]: [print(f) for f in files]
```

Fig 5: Mounting google drive in google colab

2 Preliminary data analysis

Some general info about the dataset. It contains roughly 2.5 million records across 79 columns. Data consists of mostly int64 and float64 types, except 3 attributes of 'object' type.

Dataset contains of network traffic data during different attacks, represented with values like: port numbers, IP adresses, packet lenghts, SYN/ACK/FIN/.. flag counts, packet size and other..

```
In [31]: dataset.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2433719 entries, 0 to 2433718
Data columns (total 79 columns):
 #   Column                       Dtype
---  ------                       -----
 0    Destination Port            int64
 1    Flow Duration               int64
 2    Total Fwd Packets           int64
 3    Total Backward Packets      int64
 4   Total Length of Fwd Packets  int64
 5    Total Length of Bwd Packets int64
 6    Fwd Packet Length Max       int64
 7    Fwd Packet Length Min       int64
 8    Fwd Packet Length Mean      float64
 9    Fwd Packet Length Std       float64
 10  Bwd Packet Length Max        int64
```

Fig 6: Data Analysis

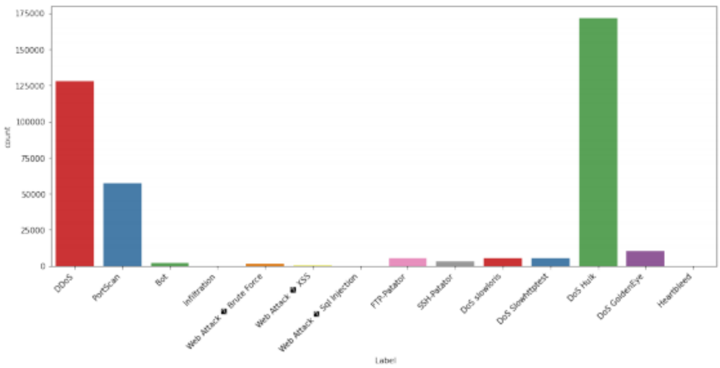

Fig 7: Attack distribution

1 Data preparation

In this chapter, final data preparation steps are taken before we use the data for model traning and testing.

These steps include:

```
Data scaling
Label encoding
Data splitting
```

```
[3]: import pandas as pd
     import numpy as np # linear algebra
     import pickle
```

```
[4]: # Load cleaned dataset.

     dataset= pd.read_csv("/content/drive/MyDrive/MachineLearning/iot_ids/data/clean/
     ↪dataset_clean_dropna.csv")
     data_cols= dataset.columns
```

```
[5]: data_cols
```

```
[5]: Index(['DestinationPort', 'FlowDuration', 'TotalFwdPackets',
            'TotalBackwardPackets', 'TotalLengthofFwdPackets',
            'TotalLengthofBwdPackets', 'FwdPacketLengthMax', 'FwdPacketLengthMin',
            'FwdPacketLengthMean', 'FwdPacketLengthStd', 'BwdPacketLengthMax',
```

Fig 8: Data Preparation

2.1 Data Cleaning

This chapter contains data cleaning code. We go through the process of renaming columns, removing NaN and non-finite values (-inf, inf) to get the data ready for visualization and model training.

2.2 Renaming columns

```
In [38]: # Removing whitespaces in column names.

         col_names = [col.replace(' ', '') for col in dataset.columns]
         dataset.columns = col_names
         dataset.head()
```

Out[38]:

	DestinationPort	FlowDuration	TotalFwdPackets	...	IdleMax	IdleMin	Label
0	54865	3	2	...	0	0	BENIGN
1	55054	109	1	...	0	0	BENIGN
2	55055	52	1	...	0	0	BENIGN
3	46236	34	1	...	0	0	BENIGN
4	54863	3	2	...	0	0	BENIGN

[5 rows x 79 columns]

Fig 9: Data Cleaning

1.1 Scaling the data

The next few cells contain the code for scaling the data into the size adequate for the ML algorithm

[7]:
```
# Splitting dataset into features and labels.

y = dataset['Label']
X = dataset.loc[:, dataset.columns != 'Label'].astype('float64')
cols= X.columns
```

[8]:
```
X.head()
```

[8]:
```
   DestinationPort  FlowDuration  TotalFwdPackets  …  IdleStd  IdleMax
IdleMin
0             53.0       30913.0              2.0  …      0.0      0.0
0.0
1             53.0       50724.0              2.0  …      0.0      0.0
0.0
2             53.0       54624.0              2.0  …      0.0      0.0
0.0
3            443.0     2002601.0              2.0  …      0.0      0.0
0.0
4            443.0       17582.0              2.0  …      0.0      0.0
0.0
```

Fig 10: Data Scaling

```
[5 rows x 78 columns]
```

[9]:
```
X.shape
```

[9]:
```
(486447, 78)
```

[10]:
```
# For scaling the data, we use RobustScaler class from sklearn.

from sklearn.preprocessing import MinMaxScaler, StandardScaler, RobustScaler
```

For scaling the data we used RobustScaler class from sklearn. RobustScaler is used to perserve outliers in the data.

[11]:
```
scaler = RobustScaler()
scaler.fit(X)

X = scaler.transform(X)
```

[12]:
```
# Checking if scaling has been succesful.
X[0]
```

[12]:
```
array([-6.92307692e-02, -5.31456041e-03,  0.00000000e+00,  0.00000000e+00,
        2.60061920e-01,  9.80891720e-02,  1.58371041e-01,  2.00000000e+00,
        8.29787234e-01,  0.00000000e+00,  6.34037820e-02,  1.90361446e+00,
        2.88209607e-01,  0.00000000e+00,  1.72889669e-01,  4.15975526e-03,
```

Fig 11: Data Scaling

2 Label encoding

Label encoding is done when dataset contains categorical values (ex. 0-5, A/B/C, 55+). It is used to turn categorical values into numerical values by replacing data categories with integers starting with 0.

3

[13]:
```
from sklearn.preprocessing import LabelEncoder
```

'Lables' column contains categorical values - 15 of them (14 types of attacks in our dataset + 1 normal state).

To convert this into numerical values we will use 'LabelEncoder' class from sklearn.

[14]:
```
LE = LabelEncoder()

LE.fit(y)
y = LE.transform(y)
pickle.dump(LE,open("/content/drive/MyDrive/MachineLearning/iot_ids/models/
↪label_encoder.sav","wb"))
```

[15]:
```
LE=pickle.load(open("/content/drive/MyDrive/MachineLearning/iot_ids/models/
↪label_encoder.sav","rb"))
```

Fig 12: Label Encoding

3 Splitting the data

Final step to data preparation is splitting the data into traning and testing sets. For this there already exists sklearn function that does all the splitting for us. This step is important so we can have representative data for evaluating our model. Both train and test samples should contain similar data variance.

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

```
[19]: # The next step is to split training and testing data. For this we will use
      ↪sklearn function train_test_split().

      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2)
```

4

```
      X_train.shape, X_test.shape, y_train.shape, y_test.shape
[19]: ((389157, 78), (97290, 78), (389157,), (97290,))
[20]: X_train
```

Fig 13: Data Splitting

1.1 Data visualization

So, by now we know our dataset has 78 features and is split into 15 categories (14 attacks and 1 "normal" state). Next step is to try and visualize what the dataset looks like in feature space. For this we will use principal component analysis (PCA) to reduce dimensionality and then pass the reduced dataset to t-SNE (t - Distributed Stohastic Neighbor Entities) for visual representation in 2D space.

```
In [30]: from sklearn.decomposition import PCA
         from sklearn.manifold import TSNE

         %matplotlib inline

In [31]: # We are going to pick 10.000 random rows from the dataset for visualization purposes
         # Setting the random seed for reproducability of results.

         np.random.seed(42)

         rand_perm = np.random.permutation(dataset.shape[0])

In [32]: feature_cols = dataset.columns[:-1]

         dataset_subset = dataset.reindex(rand_perm[:10000])

In [33]: dataset_subset = dataset_subset.replace([np.inf, -np.inf], np.nan)
         dataset_subset.dropna(inplace=True)
```

Fig 14: Feature Extraction

```
In [34]: data_subset = dataset_subset[feature_cols].values

In [35]: # Performing the principal component analysis. With just 19 components the variance r

         pca = PCA(n_components=19)
         pca_res = pca.fit_transform(data_subset)

         # data_subset = None
         np.sum(pca.explained_variance_ratio_)

Out[35]: 0.9999419771403467

In [36]: # Computing t-SNE.

         tsne = TSNE(n_components=2, verbose=0, perplexity=40, n_iter=1000)
         tsne_res = tsne.fit_transform(data_subset)
         print("Complete")

Complete

In [37]: dataset_subset['tsne_firstD'] = tsne_res[:,0]
         dataset_subset['tsne_secondD'] = tsne_res[:,1]

In [38]: import seaborn as sns

         plt.figure(figsize=(16,16))

         sns.scatterplot(
             x="tsne_firstD", y="tsne_secondD",
             palette=sns.color_palette("hls", 13),
             data=dataset_subset,
             hue="Label",
             legend="full",
```

Fig 15: Feature Extraction

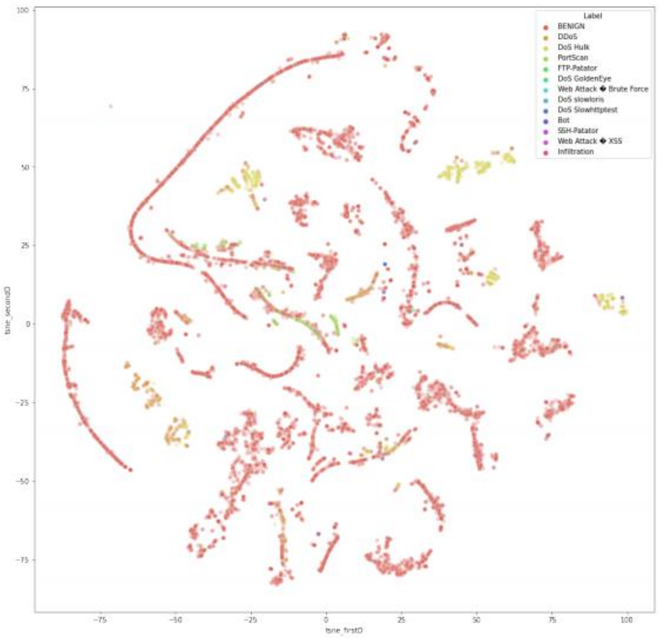

Fig 16: Visual representation in 2D of dimensionality of the data

4 Model training

In this chapter we try to evaluate different algorithms to get the most efficient. Meanwhile, the goal is to prove that CNN-LSTM is more efficient.

```
[22]: #Evaluate Models
      from sklearn import metrics
      from matplotlib import pyplot as plt


      models=[]
```

Fig 17: Model Training

5 K-Nearest Neighbor Classifier

```
[23]: from sklearn.neighbors import KNeighborsClassifier
      knn =  KNeighborsClassifier()
      knn.fit(X_train, y_train)
```

5

```
[23]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                           metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                           weights='uniform')

[24]: pickle.dump(knn,open("/content/drive/MyDrive/MachineLearning/iot_ids/models/
       ↪knn_model.sav","wb"))

[25]: knn_model= pickle.load(open("/content/drive/MyDrive/MachineLearning/iot_ids/
       ↪models/knn_model.sav", "rb"))

[26]: models.append(("K-Nearest Neighbors Classifier",knn_model))
```

Fig 18: K-NN Classifier Model Training

6 Logistic Regression

```
[27]: from sklearn.linear_model import LogisticRegression

      lgr = LogisticRegression(n_jobs=-1, random_state=0)
      lgr.fit(X_train, y_train)
```

```
[27]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                         intercept_scaling=1, l1_ratio=None, max_iter=100,
                         multi_class='auto', n_jobs=-1, penalty='l2', random_state=0,
                         solver='lbfgs', tol=0.0001, verbose=0, warm_start=False)
```

```
[28]: pickle.dump(lgr,open("/content/drive/MyDrive/MachineLearning/iot_ids/models/
       ↪lgr_model.sav","wb"))
```

```
[29]: lgr_model= pickle.load(open("/content/drive/MyDrive/MachineLearning/iot_ids/
       ↪models/lgr_model.sav", "rb"))
```

```
[30]: models.append(("Logistic Regression",lgr_model))
```

Fig 19: Logistic Regression Classifier Model Training

6.1 Naive Baye Classifier

```
[31]: from sklearn.naive_bayes import BernoulliNB
      nb=BernoulliNB()
```

```
[32]: nb.fit(X_train, y_train)
```

```
[32]: BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)
```

```
[33]: pickle.dump(nb,open("/content/drive/MyDrive/MachineLearning/iot_ids/models/
       ↪nb_model.sav","wb"))
```

```
[34]: nb_model= pickle.load(open("/content/drive/MyDrive/MachineLearning/iot_ids/
       ↪models/nb_model.sav", "rb"))
```

6

```
[35]: models.append(("Naive Baye Classifier",nb_model))
```

```
[36]: import seaborn as sns
      import pandas as pd
      import matplotlib.pyplot as plt
```

Fig 20: Naïve Bayes Classifier Model Training

7 CNN-LSTM

```
[40]: import json
      import os
      import tensorflow as tf
      import time
      import numpy as np
      from tensorflow.keras.models import Sequential
      from tensorflow.keras.layers import Convolution1D, MaxPooling1D, Dense,␣
       ↪Flatten, Conv2D, Dropout,  TimeDistributed, LSTM
      # from keras.callbacks import TensorBoard, EarlyStopping
```

19

```
[41]: print(tf.__version__)
      import distutils
      if distutils.version.LooseVersion(tf.__version__) < '1.14':
          raise Exception('This notebook is compatible with TensorFlow 1.14 or␣
       ↪higher')

      2.5.0
```

Fig 21: Importing CNN-LSTM model

10 DEFINE MODEL

```
[44]: def get_model(X_train):
    cnn_lstm = Sequential()
    cnn_lstm.add(Convolution1D(32, 3,␣
↪padding="same",activation="relu",input_shape = (X_train.shape[1], 1)))
    print("Layer 1 : ", cnn_lstm.output_shape)
```

20

```
    cnn_lstm.add(Convolution1D(64, 3,␣
↪padding="same",activation="relu",input_shape = (78, 1)))
    print("Layer 2 : ", cnn_lstm.output_shape)
    cnn_lstm.add(MaxPooling1D(pool_size=(5)))
    print("Layer 3 : ", cnn_lstm.output_shape)
    cnn_lstm.add(Dropout(0.5))
    print("Layer 4 : ", cnn_lstm.output_shape)
    cnn_lstm.add(LSTM(128, return_sequences=True))
    print("Layer 5 : ", cnn_lstm.output_shape)
    cnn_lstm.add(LSTM(256, return_sequences=False))
    print("Layer 6 : ", cnn_lstm.output_shape)
    cnn_lstm.add(Dense(256, activation="relu"))
    print("Layer 7 : ", cnn_lstm.output_shape)
    print("Layer 7 : ", cnn_lstm.output_shape)
    cnn_lstm.add(Dropout(0.5))
    print("Layer 8 : ", cnn_lstm.output_shape)
    cnn_lstm.add(Dense(14, activation="softmax"))
    print("Layer 9 : ", cnn_lstm.output_shape)
    return cnn_lstm
```

Fig 22: CNN-LSTM Classifier Model Training

```
with strategy.scope():
  cnn_lstm = get_model(X_train)
  opt= opt = tf.keras.optimizers.Adam()
  cnn_lstm.compile(optimizer=opt,
                 loss=tf.keras.losses.sparse_categorical_crossentropy,
                 metrics=['accuracy'])

cnn_lstm.summary()
```

Fig 23: CNN-LSTM Optimizer

```
from sklearn.model_selection import cross_val_score


for name, model in models:
    labels=d.unique()
    scores = cross_val_score(model, X_train, y_train, cv=10)
    y_predict=model.predict(X_train)
    accuracy = metrics.accuracy_score(y_train, y_predict)
    cm = metrics.confusion_matrix(y_train, y_predict)
    df_cm = pd.DataFrame(cm, index = [i for i in labels],
                  columns = [i for i in labels])
    classification = metrics.classification_report(y_train, y_predict)
    print()
    print('============================== {} Model Evaluation␣
↪==============================='.format(name))
    print()
    print ("Cross Validation Mean Score:" "\n", scores.mean())
    print()
    print ("Model Accuracy:" "\n", accuracy)
    print()
    print()
    print("Classification report:" "\n", classification)
    print()
    plt.figure(figsize = (15,15))
    sns.set(font_scale=1.4)
    sns.heatmap(df_cm, annot=True)
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

Fig 24: Performnce metrics calculation for models

5 Results and Evaluation

Algorithms	Accuracy	Precision	Recall	F1 Score	ROC Curve
Naïve Bayes	74.25	92.0	74.0	80.0	80.0
K-Nearest Neighbor	98.72	99.0	99.0	99.0	99.0
Logistic Regression	81.38	81.0	81.0	80.0	80.0
CNN-LSTM	99.82	100	100	100	100

Fig 11: Results from algorithms