Configuration Manual

MSc Research Project
Data Analytics

Dimple
Student ID: x20176759

School of Computing
National College of Ireland

Supervisor: Prof. Vladimir Milosavljevic

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Dimple
Student ID:	x20176759
Programme:	MSc in Data Analytics
Year:	2021-22
Module:	Research Project
Supervisor:	Prof. Vladimir Milosavljevic
Submission Due Date:	16th December 2021
Project Title:	Configuration Manual
Word Count:	814
Page Count:	14

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:	Dimple
Date:	31st January 2022

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

Dimple
x20176759

1 Introduction

Specifically, this setup manual outlines the steps that must be taken in order to imitate the planned study and get the intended findings. Procedures and code snippets for implementing the models, as well as the steps and code snippets for testing and assessing the model are included in the manual. In the end, the accuracy of the model will classify the sentiments of public perception towards the Covid-19 vaccine on Twitter Yang and Sornlertlamvanich (2021).

2 System Configuration

This research may be carried out on any computer system that meets the following minimum requirements for software and tools:

Hardware Configuration		Software Configuration
Operating System	macOS Monterey 12.0.1	Microsoft Office suite
RAM	8 GB	IDE - Google collab
Hard Disk	500 GB	Python 3.6.9
Processor	Apple M1	

Figure 1: Hardware and Software Requirements

Among the MS office suite's offerings are Microsoft Excel and Microsoft Word. Reporting in Microsoft Word and a few combining methods in Microsoft Excel are used to analyze the data. All data preparation, EDA, model construction, and assessment processes are performed in Python using Google Collab. Google Collab is a browser-based application of Google Research mounted to Google drive that allows doing python code. It provides 12 GB RAM and fast processing speed. It has Python version 3.6.9.

3 Project Development

- The first step is the collection of data set from the Kaggle website.
- Open Google drive and upload the data set .csv file.
- Open Google Collab, create a new notebook in figure 2.

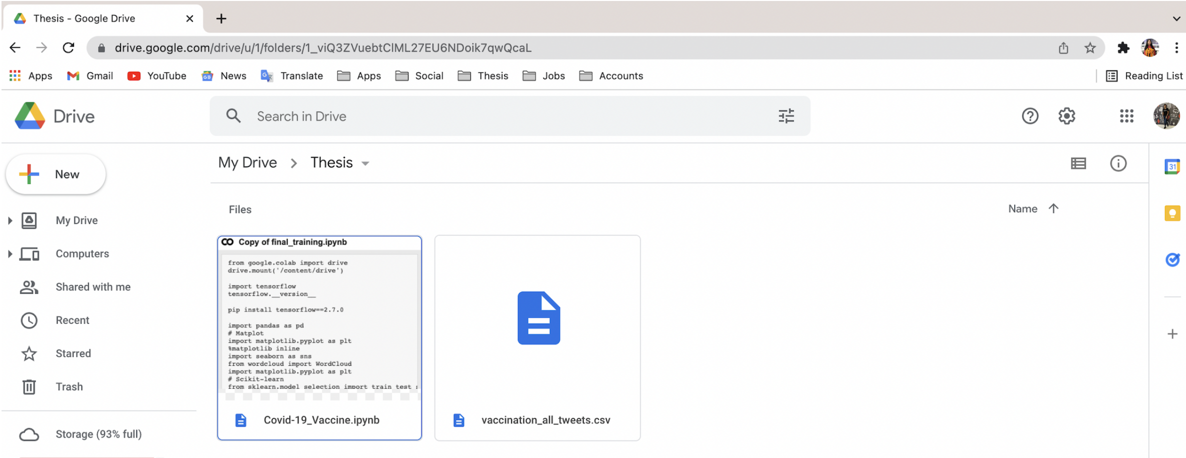

Figure 2: Upload file Google Drive

- Mount Google Collab with open Google Drive as shown in figure 3.

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

    Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Figure 3: Mount Google Drive in Google collab Python Notebook

- Import all necessary python libraries that are used to implement all tools and techniques as shown in figure 4.

Main Libraries are: TensorFlow
Matplotlib
Scikit-learn
Pandas
NLTK
Keras
RE

```
# Tensorflow
import tensorflow
tensorflow.__version__

import pandas as pd

# Matplot
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Scikit-learn
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from sklearn.manifold import TSNE
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_selection import VarianceThreshold
from sklearn.utils import shuffle

# Keras
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Reshape, Activation, Dense, Dropout, Embedding, Flatten, Conv1D, MaxPool1D, LSTM, Bidirectional
from keras import utils
from keras.callbacks import ReduceLROnPlateau, EarlyStopping
from tensorflow.keras.regularizers import l2
import tensorflow

# nltk
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

# Word2vec
import gensim

# Utility
import re
import numpy as np
import os
from collections import Counter
import logging
import time
import pickle
import itertools
import re
```

Figure 4: Import Libraries

- Read the CSV file and find the information and size of the data in figure 5.

```
data = pd.read_csv('/content/drive/My Drive/Thesis/vaccination_all_tweets.csv')
data.head()
```

	id	user_name	user_location	user_description	user_created	user_followers	user_friends	user_favourites	user_verified	date	text	hashtags	source	retweets	favorites	is_retweet
0	1340539111971516416	Rachel Roh	La Crescenta-Montrose, CA	Aggregator of Asian American news; scanning di...	2009-04-08 17:52:46	405	1692	3247	False	2020-12-20 06:06:44	Same folks said daikon paste could treat a cyt...	['PfizerBioNTech']	Twitter for Android	0	0	False
1	1338158543359250433	Albert Fong	San Francisco, CA	Marketing dude, tech geek, heavy metal & '80s ...	2009-09-21 15:27:30	834	666	178	False	2020-12-13 16:27:13	While the world has been on the wrong side of ...	NaN	Twitter Web App	1	1	False
2	1337858199140118533	eli🇱🇹🇪🇺👌	Your Bed	heil, hydra 🖐☺	2020-06-25 23:30:28	10	88	155	False	2020-12-12 20:33:45	#coronavirus #SputnikV #AstraZeneca #PfizerBio...	['coronavirus', 'SputnikV', 'AstraZeneca', 'Pf...	Twitter for Android	0	0	False
3	1337855739918835717	Charles Adler	Vancouver, BC - Canada	Hosting "CharlesAdlerTonight" Global News Radi...	2008-09-10 11:28:53	49165	3933	21853	True	2020-12-12 20:23:59	Facts are immutable, Senator, even when you're...	NaN	Twitter Web App	446	2129	False
4	1337854064604966912	Citizen News Channel	NaN	Citizen News Channel bringing you an alternati...	2020-04-23 17:58:42	152	580	1473	False	2020-12-12 20:17:19	Explain to me again why we need a vaccine @Bor...	['whereareallthesickpeople', 'PfizerBioNTech']	Twitter for iPhone	0	0	False

```
[6] display(data.shape, str(data.shape[0])+" tweets in dataset")

(224249, 16)
'224249 tweets in dataset'
```

```
data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 224249 entries, 0 to 224248
Data columns (total 16 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   id                224249 non-null  int64
 1   user_name         224247 non-null  object
 2   user_location     158652 non-null  object
 3   user_description  207884 non-null  object
 4   user_created      224249 non-null  object
 5   user_followers    224249 non-null  int64
 6   user_friends      224249 non-null  int64
 7   user_favourites   224249 non-null  int64
 8   user_verified     224249 non-null  bool
 9   date              224249 non-null  object
 10  text              224249 non-null  object
 11  hashtags          175290 non-null  object
 12  source            224130 non-null  object
 13  retweets          224249 non-null  int64
 14  favorites         224249 non-null  int64
```

Figure 5: Import dataset

• Visualize the count of tweets against the location of the user and the count of tweets against the platform-wise tweets in figure 6 and figure 7 respectively.

```
# Visulizing Tweet Count vs Location
plt.figure(figsize=(15,10))
data['user_location'].value_counts().nlargest(20).plot(kind='bar')
plt.xticks(rotation=60)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19]), <a list of 20 Text major ticklabel objects>)
```

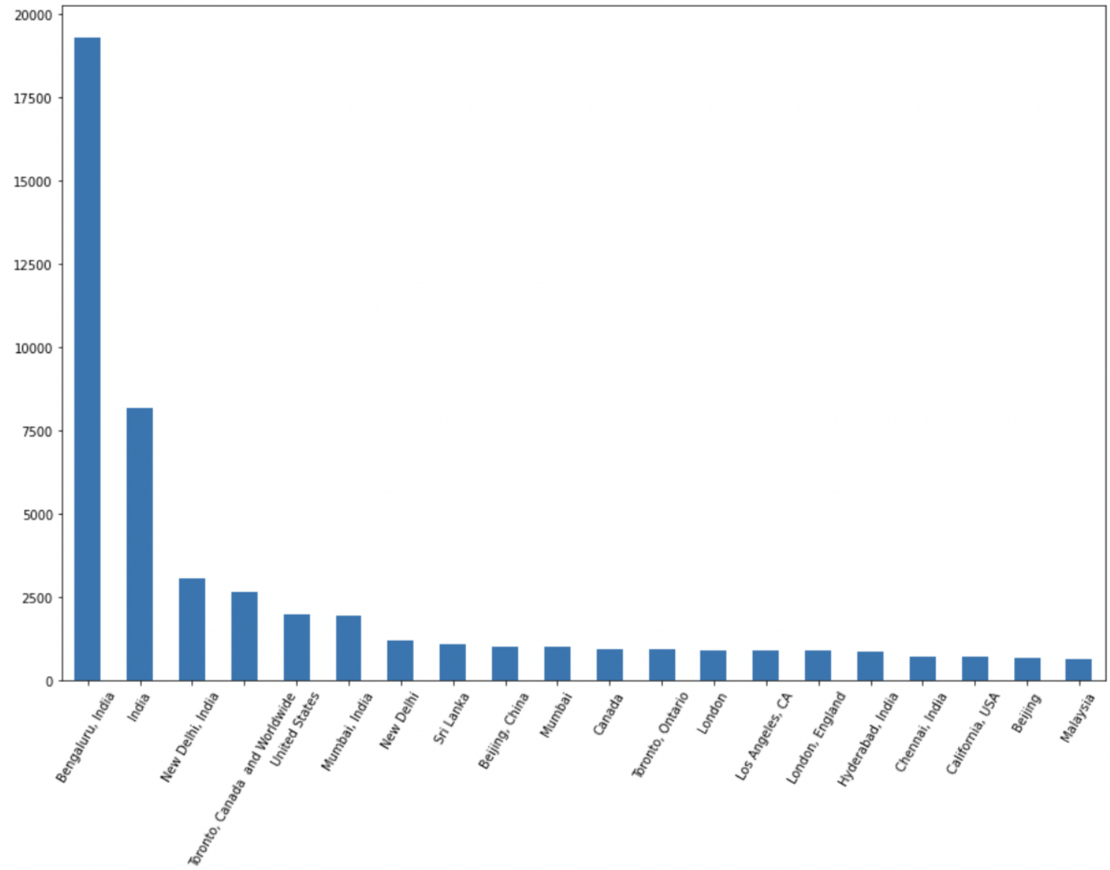

Figure 6: Tweets count vs User Location

```python
#Visualizing Tweet Platform-wise Distribution
plt.figure(figsize=(15,10))
data['source'].value_counts().nlargest(6).plot(kind='bar')
plt.xticks(rotation=80)
```

```
(array([0, 1, 2, 3, 4, 5]), <a list of 6 Text major ticklabel objects>)
```

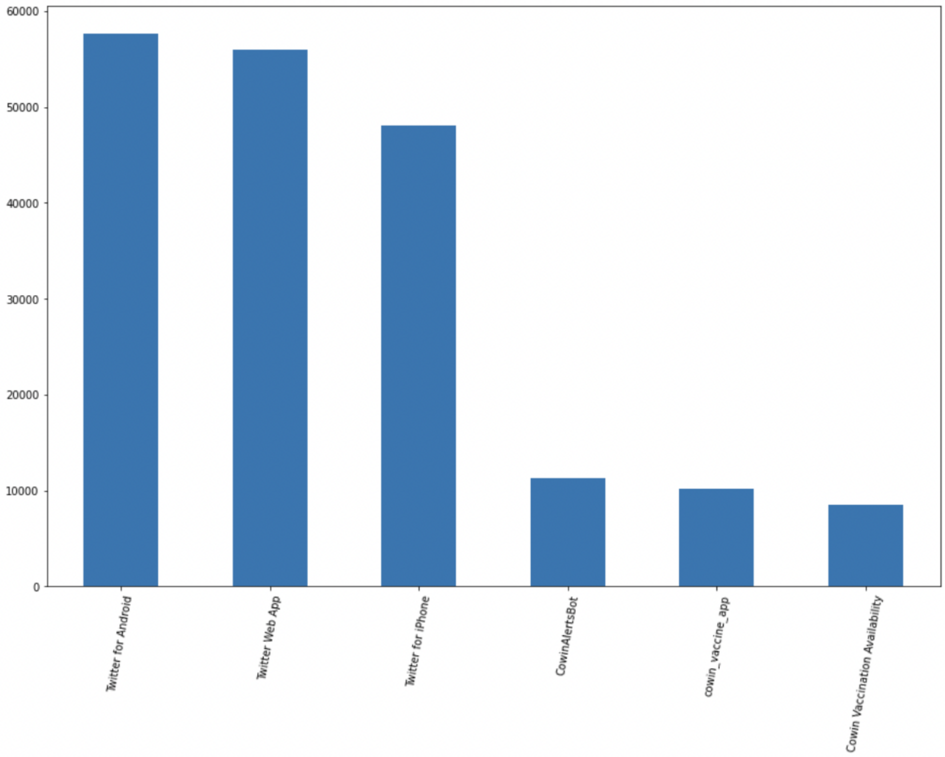

Figure 7: Tweets count vs Platform-wise tweets

- Removing duplicate tweets and unnecessary columns, the dataset has 222424 rows and 3 columns in figure 8.

```python
[14] # dropping unnecessary
     data.drop(columns={"id","user_name","user_description","user_created","user_followers",\
                        "user_friends","user_favourites","user_verified","hashtags","source","retweets","favorites","is_retweet"},inplace=True)
```

```python
[19] data.head(10)
```

	user_location	date	text
0	La Crescenta-Montrose, CA	2020-12-20	Same folks said daikon paste could treat a cyt...
1	San Francisco, CA	2020-12-13	While the world has been on the wrong side of ...
2	Your Bed	2020-12-12	#coronavirus #SputnikV #AstraZeneca #PfizerBio...
3	Vancouver, BC - Canada	2020-12-12	Facts are immutable, Senator, even when you're...
4	NaN	2020-12-12	Explain to me again why we need a vaccine @Bor...
5	Birmingham, England	2020-12-12	Does anyone have any useful advice/guidance fo...
6	Austria, Ukraine and Kosovo	2020-12-12	it is a bit sad to claim the fame for success ...
7	NaN	2020-12-12	There have not been many bright days in 2020 b...
8	NaN	2020-12-12	Covid vaccine; You getting it?\n\n #CovidVacci...
9	Islamabad	2020-12-12	#CovidVaccine \n\nStates will start getting #C...

```python
data.shape
```

```
(222424, 3)
```

Figure 8: Data set after cleaning

• Pre-processing of data using natural language processing techniques in figure 9.

```
[16] #Preprocessing
def preprocess(data):

    #Removing URLs with a regular expression
    url_pattern = re.compile(r'https?://\S+|www\.\S+')
    data = url_pattern.sub(r'', data)

    # Remove Emails
    data = re.sub('\S*@\S*\s?', '', data)

    # Remove new line characters
    data = re.sub('\s+', ' ', data)

    # Remove distracting single quotes
    data = re.sub("\'", "", data)

    return data

temp = []
#Splitting pd.Series to list
data_to_list = data['text'].values.tolist()
for i in range(len(data_to_list)):
    temp.append(preprocess(data_to_list[i]))
```

```
#removes punctuations
def sent_to_words(sentences):
    for sentence in sentences:
        yield(gensim.utils.simple_preprocess(str(sentence), deacc=True)) # deacc=True removes punctuations


data_words = list(sent_to_words(temp))
```

Figure 9: Data Pre-processing

• Sentiment analysis of data using categorization and the Text Blob approach finds polarity and subjectivity of tweets in figure 10.

```
from textblob import TextBlob
# Function to assign polarity and subjectivity to the tweets
def blob_fun(text):
  senti = TextBlob(text)
  senti_polarity = senti.sentiment.polarity
  senti_subjectivity = senti.sentiment.subjectivity

  if senti_polarity > 0:
    res = 'Positive'

  elif senti_polarity < 0:
    res = 'Negative'

  elif senti_polarity == 0:
    res ="Neutral"

  result = {'polarity':senti_polarity,'subjectivity':senti_subjectivity,'sentiment':res}

  return result
```

Figure 10: Labelling data

- Split pre-processed and labelled data into train and test in figure 11.

```
[46] X_train, X_test, y_train, y_test = train_test_split(tweets,y, test_size=0.2)
```

Figure 11: Split data into Train and Test

- Implementation of Stacking Model of the Decision tree, Random Forest Classifier, and XGBClassifier in figure 12.

```
#Stacking Model

from xgboost import XGBClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier
from sklearn.feature_selection import SelectFromModel
from sklearn.feature_selection import f_classif
from sklearn.feature_selection import mutual_info_classif
from sklearn.feature_selection import SelectKBest,SelectPercentile
from sklearn.tree import DecisionTreeClassifier
from mlxtend.classifier import StackingClassifier

def plot_roc_curve(fpr, tpr):
    plt.plot(fpr, tpr, color='orange', label='ROC')
    plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend()
    plt.show()

xgb=XGBClassifier()
rf2=RandomForestClassifier()
dt=DecisionTreeClassifier()
def stacking(X_train,y_train,X_test,y_test):
  classifiers=[xgb,dt]
  sc = StackingClassifier(classifiers,meta_classifier=rf2)
  sc.fit(X_train,y_train)
  print("Stacking Classifier :Test set")
  y_pred = sc.predict(X_test)
  print ("Stacking Classifier :Accuracy : ", accuracy_score(y_test,y_pred)*100)
  #confusion Matrix
  matrix =confusion_matrix(y_test, y_pred)
  class_names=[0,1]
  fig, ax = plt.subplots()
  tick_marks = np.arange(len(class_names))
  plt.xticks(tick_marks, class_names)
  plt.yticks(tick_marks, class_names)
  sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="YlGnBu" ,fmt='g')
  ax.xaxis.set_label_position("top")
  plt.tight_layout()
  plt.title('Confusion matrix', y=1.1)
  plt.ylabel('Actual label')
  plt.xlabel('Predicted label')
  plt.show()
```

Figure 12: Stacking Model

- Implementation of Recurrent Neural Networks (RNN) in figure 13.

```
#RNN
model2 = Sequential()
model2.add(layers.Embedding(max_words, 128))
model2.add(layers.LSTM(64,dropout=0.5))
model2.add(layers.Dense(16, activation='relu'))
model2.add(layers.Dense(8, activation='relu'))
model2.add(layers.Dense(1,activation='sigmoid'))

model2.compile(optimizer='adam',loss='binary_crossentropy', metrics=['accuracy'])
checkpoint2 = ModelCheckpoint("rnn_model.hdf5", monitor='val_accuracy', verbose=1,save_best_only=True, mode='auto', period=1,save_weights_only=False)
history = model2.fit(X_train, y_train, epochs=5,validation_data=(X_test, y_test),callbacks=[checkpoint2])

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)
#Train and validation accuracy
plt.plot(epochs, acc, 'b', label='Training accurarcy')
plt.plot(epochs, val_acc, 'r', label='Validation accurarcy')
plt.title('Training and Validation accurarcy')
plt.legend()
plt.figure()
#Train and validation loss
plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and Validation loss')
plt.legend()
plt.show()
#Testing and Graphs
y_pred_keras = model2.predict(X_test).ravel()
fpr_keras, tpr_keras, thresholds_keras = roc_curve(y_test, model2.predict(X_test))
from sklearn.metrics import roc_auc_score
from matplotlib import pyplot
pyplot.plot([0, 1], [0, 1], linestyle='--')
pyplot.plot(fpr_keras, tpr_keras, marker='.')
pyplot.show()
#Confusion Matrix
prediction= model2.predict(X_test)
YClass= np.zeros((len(prediction)))
acc,scor= model2.evaluate(X_test,y_test)
acc, scor
for i in range(len(prediction)):
    if prediction[i][0]>=0.5:
            YClass[i]=1
    else:
            YClass[i]=0

matrix1 = confusion_matrix(y_test, YClass)
plot_confusion_matrix(cm=matrix1,target_names=['Possitive', 'Negative'])
```

Figure 13: RNN Model

- Implementation of Convolutional Neural Networks (CNN) in figure 14.

```
from keras.layers import Dense, Embedding, GlobalMaxPooling1D, Flatten, Conv1D, Dropout, Activation
NUM_FILTERS = 250
KERNEL_SIZE = 3
HIDDEN_DIMS = 250
# CNN Model
print('Build model...')
model = Sequential()

# we start off with an efficient embedding layer which maps
# our vocab indices into EMBEDDING_DIM dimensions
model.add(Embedding(max_words,max_len, input_length=max_len))
model.add(Dropout(0.2))

# we add a Convolution1D, which will learn NUM_FILTERS filters
model.add(Conv1D(NUM_FILTERS,
                 KERNEL_SIZE,
                 padding='valid',
                 activation='relu',
                 strides=1))

# we use max pooling:
model.add(GlobalMaxPooling1D())

# We add a vanilla hidden layer:
model.add(Dense(HIDDEN_DIMS))
model.add(Dropout(0.2))
model.add(Activation('relu'))

# We project onto a single unit output layer, and squash it with a sigmoid:
model.add(Dense(1))
model.add(Activation('sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()

# fit a model
model.fit(X_train, y_train,
          batch_size=128,
          epochs=2,
          validation_split=0.1,
          verbose=2)
```

Figure 14: CNN Model

- Implementation of long short-term memory (LSTM) in figure 15.

```
#LSTM Model
model_lstm= Sequential()
model_lstm.add(Embedding(max_words,max_len,input_length=X_train.shape[1]))
model_lstm.add(LSTM(100,return_sequences=True))
model_lstm.add(Dense(50,activation='relu'))
model_lstm.add(Flatten())
model_lstm.add(Dense(460,activation='relu'))
model_lstm.add(Dense(180,activation='relu'))
model_lstm.add(Dropout(0.5))
model_lstm.add(Dense(50,activation='relu'))
model_lstm.add(Dense(20,activation='relu'))
model_lstm.add(Dense(1,activation='sigmoid'))

filepath=r"lstm_model.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='val_acc', verbose=1, save_best_only=True, mode='auto')
callbacks_list = [checkpoint]
model_lstm.compile(loss = 'binary_crossentropy', optimizer='adam',metrics = ['accuracy'])
model_lstm.summary()
history = model_lstm.fit(X_train, y_train, epochs=2,validation_data=(X_test, y_test),callbacks=[checkpoint])

#Graph
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)
#Train and validation accuracy
plt.plot(epochs, acc, 'b', label='Training accurarcy')
plt.plot(epochs, val_acc, 'r', label='Validation accurarcy')
plt.title('Training and Validation accurarcy')
plt.legend()

plt.figure()
#Train and validation loss
plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and Validation loss')
plt.legend()
plt.show()

#Testing and Graphs
y_pred_keras = model_lstm.predict(X_test).ravel()
fpr_keras, tpr_keras, thresholds_keras = roc_curve(y_test, model_lstm.predict(X_test))
from sklearn.metrics import roc_auc_score
from matplotlib import pyplot
pyplot.plot([0, 1], [0, 1], linestyle='--')
pyplot.plot(fpr_keras, tpr_keras, marker='.')
pyplot.show()
```

Figure 15: LSTM Model

- Implementation of Multi-layer Perceptron classifier (MLP) in figure 16.

```
#MLP ML
from sklearn.neural_network import MLPClassifier
def MLP(X_train,y_train,X_test,y_test):
  dt = MLPClassifier()
  dt.fit(X_train,y_train)
  y_pred = dt.predict(X_test)
  print("MLP:Confusion Matrix: ", confusion_matrix(y_test, y_pred))
  print ("MLP:Accuracy : ", accuracy_score(y_test,y_pred)*100)
  #confusion Matrix
  matrix =confusion_matrix(y_test, y_pred)
  class_names=[0,1]
  fig, ax = plt.subplots()
  tick_marks = np.arange(len(class_names))
  plt.xticks(tick_marks, class_names)
  plt.yticks(tick_marks, class_names)
  sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="YlGnBu" ,fmt='g')
  ax.xaxis.set_label_position("top")
  plt.tight_layout()
  plt.title('Confusion matrix', y=1.1)
  plt.ylabel('Actual label')
  plt.xlabel('Predicted label')
  plt.show()
  #ROC_AUC curve
  probs = dt.predict_proba(X_test)
  probs = probs[:, 1]
  auc = roc_auc_score(y_test, probs)
  print('AUC: %.2f' % auc)
  le = preprocessing.LabelEncoder()
  y_test1=le.fit_transform(y_test)
  fpr, tpr, thresholds = roc_curve(y_test1, probs)
  plot_roc_curve(fpr, tpr)

  #Classification Report
  target_names = ['Yes', 'No']
  prediction=dt.predict(X_test)
  print(classification_report(y_test, prediction, target_names=target_names))
  classes = ["Yes", "No"]
  visualizer = ClassificationReport(dt, classes=classes, support=True)
  visualizer.fit(X_train, y_train)
  visualizer.score(X_test, y_test)
  g = visualizer.poof()
MLP(X_train,y_train,X_test,y_test)
```

Figure 16: MLP Model

• CNN and LSTM models outperform well with an accuracy of 66Percent in Figures 17 and 18 respectively.

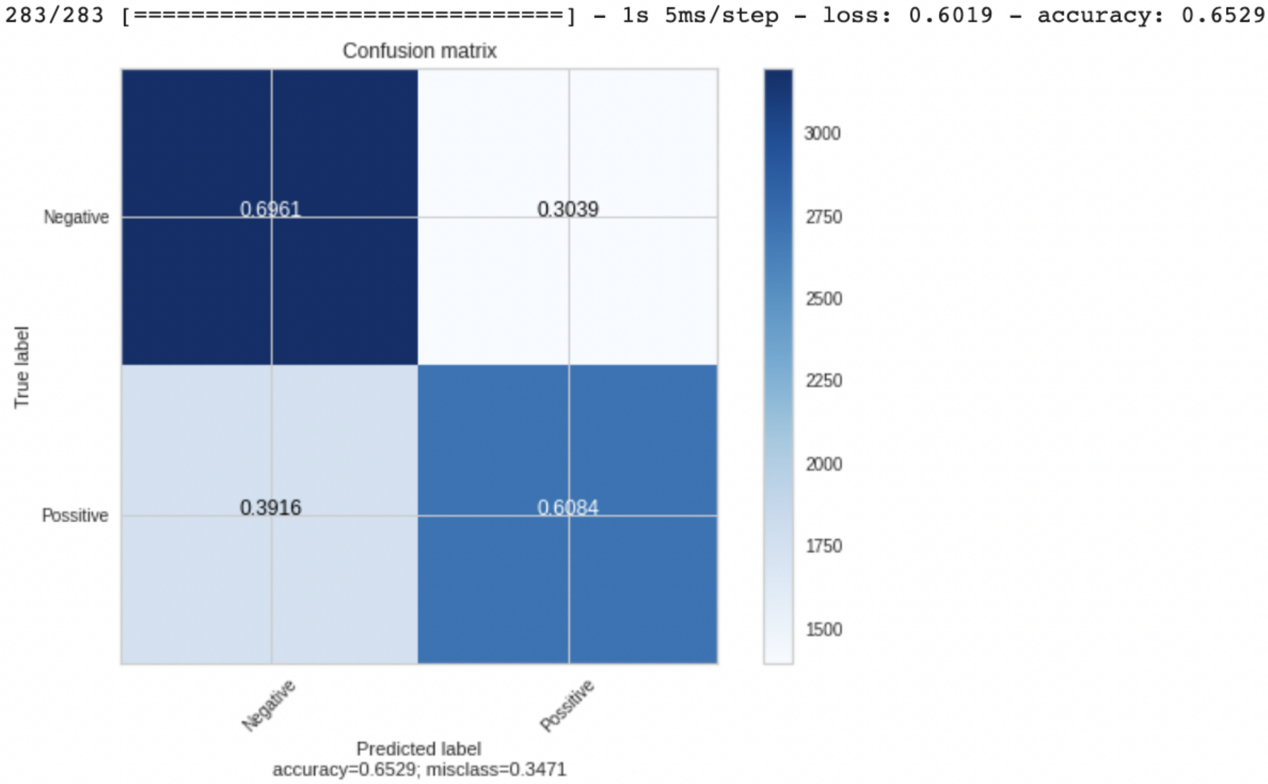

Figure 17: Accuracy of CNN model

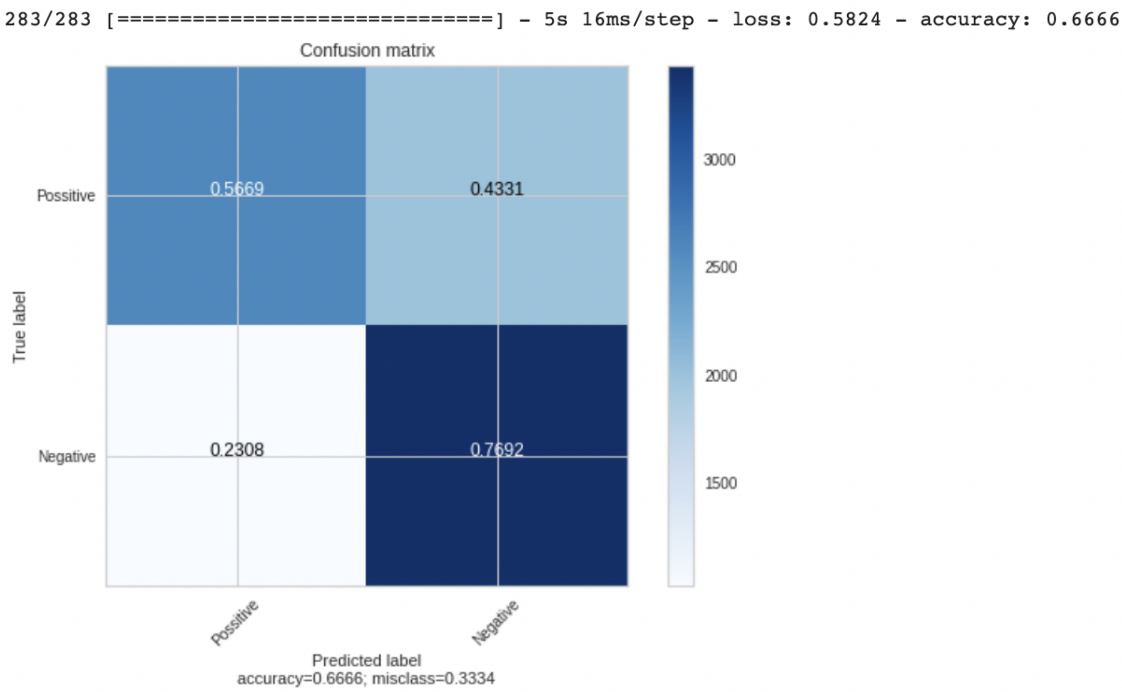

Figure 18: Accuracy of LSTM model

- Test the tweet to find the sentiment in Figure 19.

```
text = "it is a bit sad to claim the fame for success of #vaccination on patriotic competition between USA, Canada, UK and â€¦ https://t.co/IfMrAyGyTP"
preprocessed_text = [preprocess(text)]
data_words_test = list(sent_to_words(preprocessed_text))
data_cleaned_test = []
data_cleaned_test_final = []
data_cleaned_test.append(detokenize(data_words_test[0]))
data_cleaned_test_final.append(stopWords(data_cleaned_test[0]))
print(data_cleaned_test_final)
sequence = tokenizer.texts_to_sequences(data_cleaned_test_final)
test = pad_sequences(sequence, maxlen=max_len)
pred = model_lstm.predict(test)
print(pred)
if pred > 0.5:
  print('Positive')
else:
  print('Negative')
```

```
['bit sad claim fame success vaccination patriotic competition usa canada uk']
[[0.56019914]]
Positive
```

Figure 19: Testing the sentiment of the tweet

References

Yang, X. and Sornlertlamvanich, V. (2021). Public perception of covid-19 vaccine by tweet sentiment analysis, *2021 International Electronics Symposium (IES)*, pp. 151–155.