National College of Ireland

Empirical Study and Forecasting Tesla Stock prices using Sentimental analysis and deep learning methods

MSc Research Project
Data Analytics

Jorden Anthon Lopes
Student ID: x19213344

School of Computing
National College of Ireland

Supervisor: Dr.Bharathi Chakravarthi

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Jorden Anthon Lopes
Student ID:	X19213344
Programme:	Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Dr.Bharathi Chakravarthi
Submission Due Date:	31/01/2022
Project Title:	Empirical Study and Forecasting Tesla Stock prices using Sentimental analysis and deep learning methods
Word Count:	XXX
Page Count:	13

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:	
Date:	30th 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):	

Empirical Study and Forecasting Tesla Stock prices using Sentimental analysis and deep learning methods

Jorden Anthon Lopes
X19213344

1 Environment Requirements

This configuration manual discusses the hardware and the software required to implement the research work. The steps taken are mentioned so it will be easy for anyone replicating the experiments.

2 System Specification

2.1 Hardware Requirements

The system specification where all experiments implemented are discussed below:

- **Processor:** Intel Core i5.
- **System Memory:**1TB Hard disk, 256GB SSD.
- **RAM:**20GB.

2.2 Software Requirements

The software requirements are discussed below:

- **Windows Edition:** Windows 10
- **Integrated Development Environment:**Google Colab.
  Colab reference figure 1

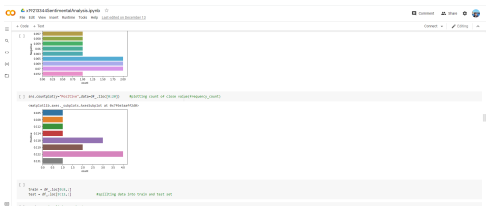


Figure 1: Google colab

- **Scripting Language:** Python 3.
- **Cloud Storage:** Google Colab.

2.3 Libraries

This project is combination of two part

2.3.1 Time series Implementation

first part is time series model implementation and required libraries are mentioned in Figure 2

```
import numpy as np                    #importing librarires
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from google.colab import drive
drive.mount('/content/drive',force_remount=True)
data_dir = 'drive/My Drive/Colab Notebooks/ThesisImplementation/TSLA.csv'
from scipy import stats
from pandas.plotting import table
import datetime as dt
import matplotlib.gridspec as gridspec   #importing libraries for deep learning models
import seaborn as sns
import tensorflow as tf
import tensorflow.keras as ks
from sklearn.model_selection import train_test_split
from tensorflow.keras.layers import Dense, LSTM, Conv1D, Bidirectional, Lambda, Input
from tensorflow.keras.models import Model
plt.style.use('default')
plt.style.use('ggplot')
from keras.layers import SimpleRNN
```

Figure 2: Time series model libraries

2.3.2 Sentimental analysis

Second part is includes sentimental analysis implementation and required libraries are mentioned in Figure 3

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from google.colab import drive
drive.mount('/content/drive',force_remount=True)
data_dir = 'drive/My Drive/Thesis folder/tweet_tesla_data.csv'
import nltk
nltk.download('vader_lexicon')
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import unicodedata
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import classification_report,confusion_matrix
!pip3 install catboost
from catboost import CatBoostClassifier
from sklearn.metrics import mean_absolute_error
import tensorflow as tf
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
```

Figure 3: Sentiment Anlysis model implementation libraries

2.4 Google colab setup

If any user does not have colab setup on google drive then

- go to google drive.
- click on new.
- click on More
- check for Google Co-laboratory
- if you do not have there plese go to more App and search for Colab.

3 Time Series Implementation

3.1 Experiment 1: Simple model implementation

- Upload TSLA.csv on google drive
- Open Google colab
- Upload x19213344Simple.ipynb file on google drive
- Double click on File on drive to open in google colab.
- change the path according to your drive location where TSLA.csv file has been stored.set path in below cell.

```
from google.colab import drive
drive.mount('/content/drive',force_remount=True)
data_dir = 'drive/My Drive/Colab Notebooks/ThesisImplementation/TSLA.csv'
Data=pd.read_csv("drive/My Drive/Colab Notebooks/ThesisImplementation/TSLA.csv")
```

Figure 4: Path to be set according to drive location

- Post opening file runtime set to GPU
- Till Drive mounting Execute one-one steps
- post mounting to drive Click on Runtime and select Run after.
- Analys the results of Experiment 1 which is implementation of Simple models.

3.2 Experiment 2: Adding convolution Layer

- Upload TSLA.csv on google drive
- Open Google colab
- Upload x19213344CNNMOdelAddition.ipynb file on google drive
- Double click on File on drive to open in google colab.

```
from google.colab import drive
drive.mount('/content/drive',force_remount=True)
data_dir = 'drive/My Drive/Colab Notebooks/ThesisImplementation/TSLA.csv'
Data=pd.read_csv("drive/My Drive/Colab Notebooks/ThesisImplementation/TSLA.csv")
```

Figure 5: Path to be set according to drive location

- change the path according to your drive location where TSLA.csv file has been stored.set path in below cell.
- Post opening file runtime set to GPU
- Till Drive mounting Execute one-one steps
- post mounting to drive Click on Runtime and select Run after.
- Analys the results of Experiment 2 which is implementation of CNN-Bi-LSTM,CNN-LSTM and CNN-RNN model

3.3 Experiment 3: Removing Outliers from data-set

- Upload TSLA.csv on google drive
- Open Google colab
- Upload x19213344CNNMRemmoveOutlier.ipynb file on google drive
- Double click on File on drive to open in google colab.
- change the path according to your drive location where TSLA.csv file has been stored.set path in below cell.

```
from google.colab import drive
drive.mount('/content/drive',force_remount=True)
data_dir = 'drive/My Drive/Colab Notebooks/ThesisImplementation/TSLA.csv'
Data=pd.read_csv("drive/My Drive/Colab Notebooks/ThesisImplementation/TSLA.csv")
```

Figure 6: Path to be set according to drive location

- Post opening file runtime set to GPU
- Till Mounting execute one -one step
- post mounting to drive Click on Runtime and select Run after.
- Analys the results of Experiment 3 which is implementation of CNN-Bi-LSTM,CNN-LSTM and CNN-RNN model without outliers in dataset.

4 Sentimental Analysis implementation

- Upload tweet_tesla_data.csv on google drive
- Upload TSLA_updated.csv on google drive
- Open Google colab
- Upload x19213344SentimentAnalysis.ipynb file on google drive
- Double click on File on drive to open in google colab.
- change the path according to your drive location where tweet_tesla_data.csv and TSLA_updated.csv file has been stored.set path in below cell.

```
from google.colab import drive
drive.mount('/content/drive',force_remount=True)
data_dir = 'drive/My Drive/Thesis folder/tweet_tesla_data.csv'
data=pd.read_csv('drive/My Drive/Thesis folder/tweet_tesla_data.csv')
stock_data=pd.read_csv("drive/My Drive/Thesis folder/TSLA_updated.csv")
```

Figure 7: Path to be set according to drive location

- Post opening file runtime set to GPU
- Till Drive mounting Execute one-one steps.
- post mounting to drive Click on Runtime and select Run after.
- Analys the results of sentiment analysis process , this process will take couple of hours to execute.

5 Important codes

Figure 8 describes how data is distributed
Figure 9 describes Graphical representation of data
Figure 10 describes Correlation between features
Figure 11 describes outlier detection process
Figure 12 describes how year wise data has been distributed
Figure 13 describes total volume distribution by year
Figure 14 describes total training loss
Figure 15 describes CNN-LSTM model
Figure 16 describes CNN-RNN model
Figure 17 discribes how data can be imported from drive
Figure 18 describes How outlier can be removed from dataset
Figure 19 describes configuration of CNN-BI-LSTM model
Figure 20 describes configuration of SGD and Huberloss function
Figure 21 describes the code use for concatenating tweets day wise
Figure 22 describes calculation of sentiment score

```
for feature in stock_features:
    Data=Data.copy()
    sns.distplot(Data[feature])
    plt.xlabel(feature)
    plt.ylabel("Count")
    plt.title(feature)
    plt.figure(figsize=(15,15))
    plt.show()
```

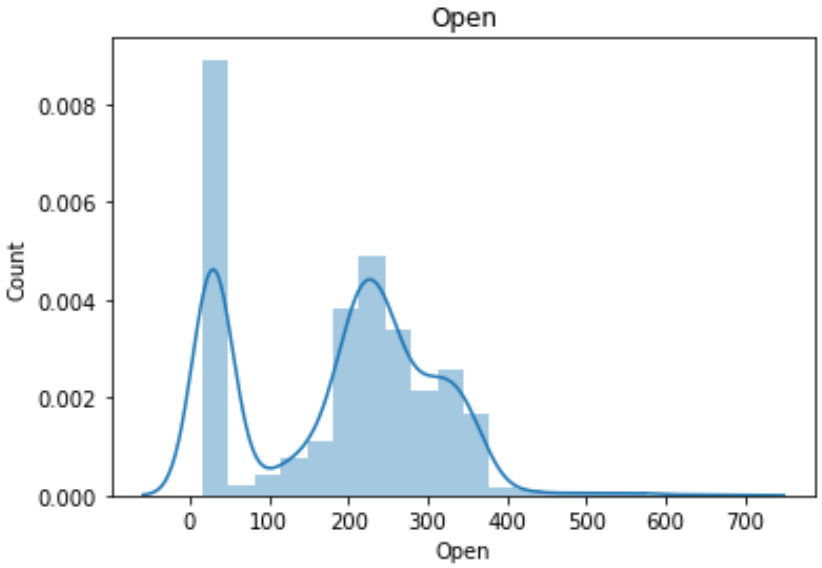


Figure 8: Data distribution

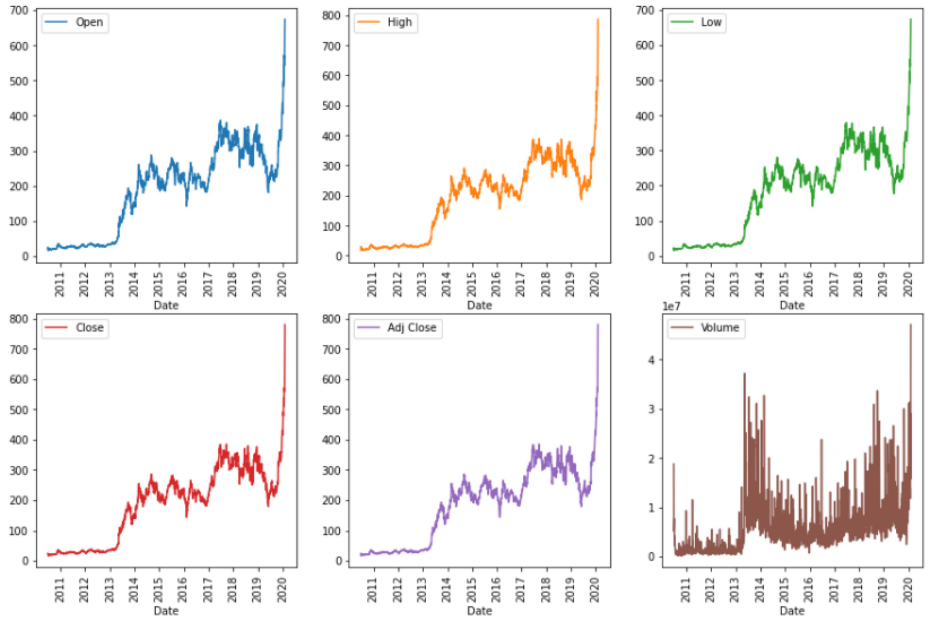


Figure 9: Stock price value graphs

```
correlationMatrix = Data.corr(method = "spearman")        #plotting correlation matrix
plt.figure(figsize=(15,5))
plot=sns.heatmap(correlationMatrix,annot=True)
```

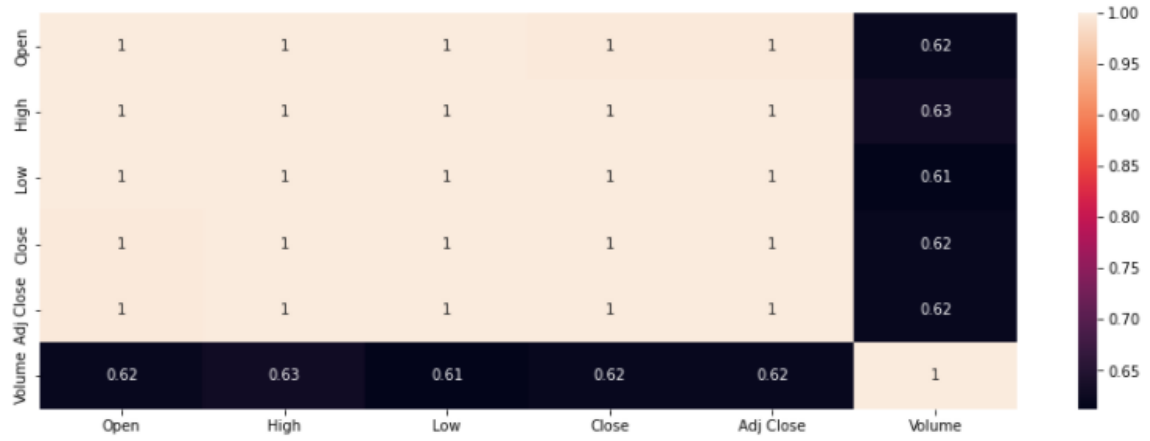


Figure 10: Correlation matrix

```
fig, ax = plt.subplots (len(Stock_Column_List), figsize = (10, 20))  # plotting box plots for outliers

for i, col_list in enumerate(Stock_Column_List):
    sns.boxplot(Data[col_list], ax = ax[i], palette = "winter", orient = 'h')
    ax[i].set_title(" Plot for Outlier Detection on" + " " + col_list, fontsize = 10)
    ax[i].set_ylabel(col_list, fontsize = 8)
    fig.tight_layout(pad = 1.1)
```

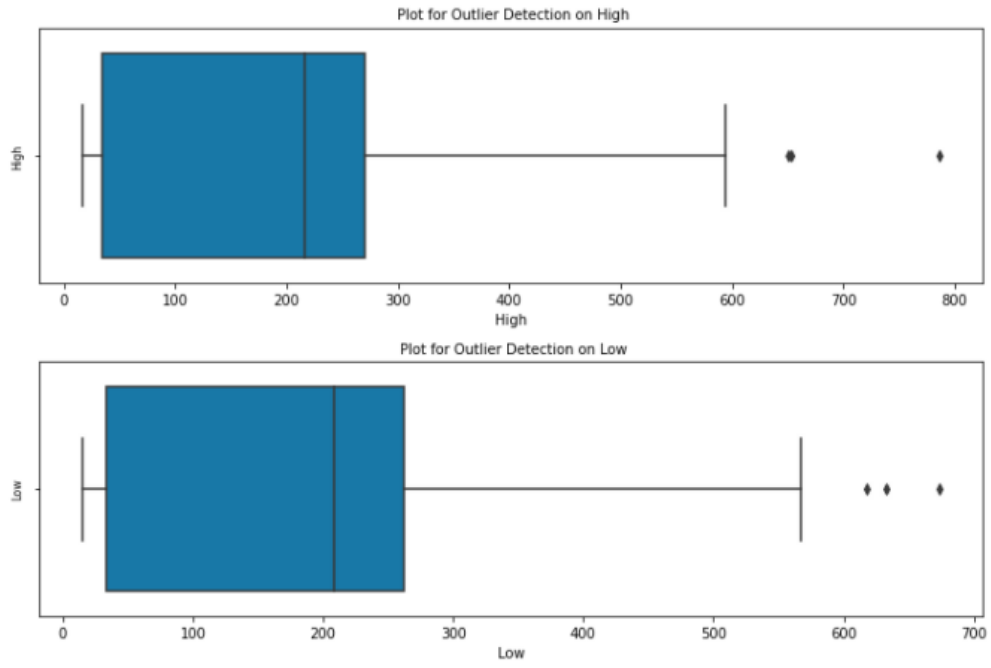


Figure 11: Outlier detection process

```
fig, ax = plt.subplots(len(Stock_Column_List), figsize = (10, 20))

# Group the data by year and plot
for i, col_list in enumerate(Stock_Column_List):
    Data.groupby('Year')[col_list].plot(ax = ax[i], legend = True)
    ax[i].set_title("Stock Price Movement Grouped by Year on" + " " + col_list, fontsize = 10)
    ax[i].set_ylabel(col_list + " " + "Price", fontsize = 8)
    fig.tight_layout(pad = 1.1)
    ax[i].yaxis.grid(True) # To enable grid only on the Y-axis
```

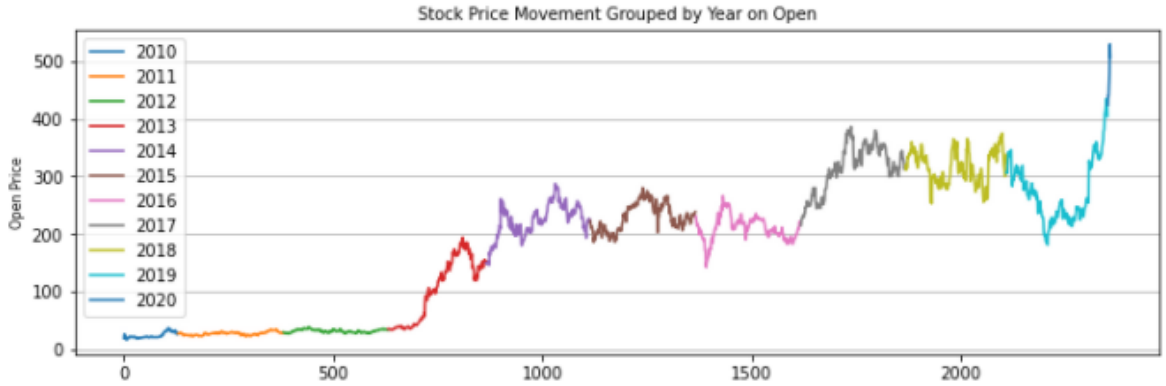


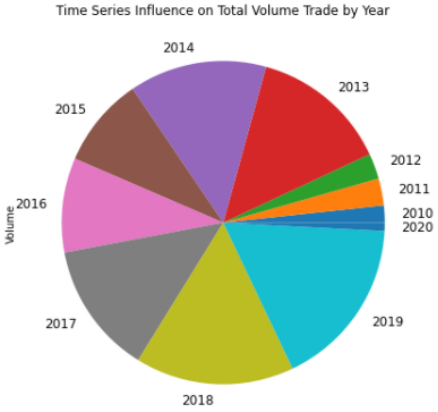
Figure 12: Year wise data distribution

```
for i, col_list in enumerate(Stock_Column_List):
    var = Data.groupby('Year')[col_list].sum()

# Convert the variable into a pandas dataframe
var = pd.DataFrame(var)

# Plot to understand the trend
plt.figure(figsize = (16, 7))
ax1 = plt.subplot(121)
var.plot(kind = "pie", y = "Volume", legend = False, fontsize = 12, sharex = False, title = "Time Series Influence on Total Volume Trade by Year", ax = ax1)

# Plot the table to identify numbers
ax2 = plt.subplot(122)
plt.axis('off') # Since we are plotting the table
tbl = table(ax2, var, loc = 'center')
tbl.auto_set_font_size(False)
tbl.set_fontsize(12)
plt.show()
```



	Volume
2010	205369100
2011	325235100
2012	307449000
2013	1657231300
2014	1657428000
2015	1088217800
2016	1138619300
2017	1590031400
2018	1913442600
2019	2060649800
2020	99283100

Figure 13: Year wise volume distribution

```
loss = history.history['loss']
epochs = range(len(loss))
plt.style.use('default')
plt.style.use('ggplot')
plt.figure(figsize=(9, 5))
#plt.plot(epochs[400:], mae, color='blue', label='trainig_mae')
plt.plot(epochs, loss, color='red', label='training_loss', linewidth=0.8)
plt.title('Training Loss')
plt.xlabel('epochs')
plt.xlim(0, 1000)
plt.ylim(0, 200)
plt.legend()
plt.show()
```

Figure 14: Training loss

```
def base_model(inputs):
    x = Conv1D(filters=32,kernel_size=4, activation='relu', name='conv_1D')(inputs)      #adding conv layer
    x = (LSTM(units=64, return_sequences=True, name='lstm_1'))(x)                        #adding lstm layer
    x = ks.layers.Dropout(0.2)(x)                                                        # adding dropout layer
    x = (LSTM(units=16, return_sequences=True, name='lstm_2'))(x)
    return x

def final_model(inputs):
    x = base_model(inputs)
    x = Dense(units=43,activation='relu', name='dense_1')(inputs)            #adding dense layers
    x = Dense(units=64, name='dense_2')(x)
    x = Lambda(lambda x:x*16, name='lambda_1')(x)                           #adding lamda layer for 4 outputs
    high_stock = Dense(1, name='high_stock')(x)
    low_stock = Dense(1, name='low_stock')(x)
    open_stock = Dense(1, name='open_stock')(x)
    close_stock = Dense(1, name='close_stock')(x)

    model = Model(inputs=inputs, outputs=[high_stock, low_stock, open_stock, close_stock])         #compiling whole model
    return model
```

Figure 15: CNN-LSTM model

```
def base_model(inputs):
    x = Conv1D(filters=256,kernel_size=4, activation='relu', name='conv_1D')(inputs)          #adding conv layer
    x = (SimpleRNN(units = 50, activation='linear', return_sequences = True))(x)              # adding RNN layer
    x = ks.layers.Dropout(0.2)(x)                                                             #adding dropout layer
    x = (SimpleRNN(units = 16, activation='linear', return_sequences = True))(x)
    return x

def final_model(inputs):
    x = base_model(inputs)
    x = Dense(units=43, activation='relu', name='dense_1')(inputs)                        #adding dense layers
    x = Dense(units=64, name='dense_2')(x)
    x = Lambda(lambda x:x*16, name='lambda_1')(x) #adding lamda layer
    high_stock = Dense(1, name='high_stock')(x)
    low_stock = Dense(1, name='low_stock')(x)
    open_stock = Dense(1, name='open_stock')(x)
    close_stock = Dense(1, name='close_stock')(x)

    model = Model(inputs=inputs, outputs=[high_stock, low_stock, open_stock, close_stock])
    return model
```

Figure 16: CNN-RNN model

```
[ ]  import numpy as np                          #importing librarires
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     import warnings
     warnings.filterwarnings('ignore')
```

```
[ ]  from google.colab import drive
     drive.mount('/content/drive',force_remount=True)
     data_dir = 'drive/My Drive/Colab Notebooks/ThesisImplementation/TSLA.csv'
     Data=pd.read_csv("drive/My Drive/Colab Notebooks/ThesisImplementation/TSLA.csv")

     Mounted at /content/drive
```

```
[ ]  Data=pd.read_csv("drive/My Drive/Colab Notebooks/ThesisImplementation/TSLA.csv")
```

Figure 17: Importing data from Drive

```
from scipy import stats
from pandas.plotting import table
import datetime as dt
# Remove the variables either using IQR technique or Z-Score
Descriptive_Statistics = Data.describe()
Descriptive_Statistics = Descriptive_Statistics.T # Convert into a dataframe

# Extract the IQR values
Descriptive_Statistics['IQR'] = Descriptive_Statistics['75%'] - Descriptive_Statistics['25%']

# In this scenario, the outliers are removed using Z-Score due to the variability in historical data
Data = Data[(np.abs(stats.zscore(Data[['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume']])) < 3).all(axis = 1)] #
Data = Data.reset_index() # Due to elimination of rows, index has to be reset
```

Figure 18: Outlier Removal code

```
def base_model(inputs):
    x = Conv1D(filters=256,kernel_size=4, activation='relu', name='conv_1D')(inputs)    #adding  conv layer
    x = Bidirectional(LSTM(units=64, return_sequences=True, name='b_lstm_1'))(x)        #adding  bilstm layer
    x = ks.layers.Dropout(0.2)(x)                                                       #adding dropout layer
    x = Bidirectional(LSTM(units=64, return_sequences=True, name='b_lstm_2'))(x)
    x = ks.layers.Dropout(0.2)(x)
    return x

def final_model(inputs):
    x = base_model(inputs)
    x = Dense(units=128, activation='relu', name='dense_1')(inputs)                      #adding dense layer
    x = Dense(units=64, name='dense_2')(x)
    x = Lambda(lambda x:x*16, name='lambda_1')(x)                                        #adding lamda layer for
    high_stock = Dense(1, name='high_stock')(x)
    low_stock = Dense(1, name='low_stock')(x)
    open_stock = Dense(1, name='open_stock')(x)
    close_stock = Dense(1, name='close_stock')(x)

    model = Model(inputs=inputs, outputs=[high_stock, low_stock, open_stock, close_stock])   #compiling whole mode
    return model
```

Figure 19: Code of CNN-BI-LSTM

```
lr_schedule = tf.keras.callbacks.LearningRateScheduler(
    lambda epoch: 1e-8 * 10**(epoch /22))
optimizer = tf.keras.optimizers.SGD(learning_rate=1e-8, momentum=0.9)
model.compile(loss={
    'high_stock': ks.losses.Huber(),
    'low_stock': ks.losses.Huber(),
    'open_stock': ks.losses.Huber(),
    'close_stock': ks.losses.Huber()
    },
     optimizer=optimizer,
     metrics={
         'high_stock': ['mae'],
        'low_stock': ['mae'],
        'open_stock': ['mae'],
        'close_stock': ['mae']
        })
history = model.fit(x=x_train, y=y_train, epochs=100, callbacks=[lr_schedule], verbose=0)
```

Figure 20: Code of SGD and Huber loss function

```
indx=0                                  #concatenating all tweets of same date
get_tweet=""
for i in range(0,len(data)-1):
    get_date=data.Date.iloc[i]
    next_date=data.Date.iloc[i+1]
    if(str(get_date)==str(next_date)):
        get_tweet=get_tweet+data.Tweets.iloc[i]+" "
    if(str(get_date)!=str(next_date)):
        temp_df = pd.DataFrame([[get_date,
                                 get_tweet]], columns = ['Date','Tweets'])
        con_data = pd.concat([con_data, temp_df], axis = 0).reset_index(drop = True)
        get_tweet=" "
```

Figure 21: Tweet concatenation

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer     # finding values for newly added columns from sentiment
#from nltk.sentiment.vader import SentimentIntensityAnalyzer
import unicodedata
sentiment_i_a = SentimentIntensityAnalyzer()
for indexx, row in con_data.T.iteritems():
    try:
        sentence_i = unicodedata.normalize('NFKD', con_data.loc[indexx, 'Tweets'])
        sentence_sentiment = sentiment_i_a.polarity_scores(sentence_i)
        con_data['Comp'].iloc[indexx] = sentence_sentiment['compound']
        con_data['Negative'].iloc[indexx] = sentence_sentiment['neg']
        con_data['Neutral'].iloc[indexx] = sentence_sentiment['neu']
        con_data['Positive'].iloc[indexx] = sentence_sentiment['pos']
    except TypeError:
        print (stocks_dataf.loc[indexx, 'Tweets'])
        print (indexx)
```

Figure 22: Calculation of sentiment score

6 Data set Links

for this implementation data has been covered from Yahoo finance and Tweeter API

1.Link for Yahoo Fiance: `https://finance.yahoo.com/quote/TSLA?p=TSLA&.tsrc=fin-srch`

2.Figure 23 Tweeter data Download

```
import tweepy
import csv
import codecs
from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream
#Access tokens
auth = tweepy.auth.OAuthHandler('lfVYDsjPpjBC1qmB9dVYGTM6b', 'T9qdLlw30X5WzZrX1W0Q46YYfKyqupAJ0LNG8vuVCTQ6XKLlJy')
auth.set_access_token('1467437206449000450-A6ehqxN8t6OMAcYvC2ry5V5olrks5T', '066BM1Y0PiHf5cauFZ3FWgaoDW4wFDTshaC6W
api = tweepy.API(auth)
# Open/Create a file to append data
csvFile = open('result1234.csv', 'a')
#Use csv Writer
csvWriter = csv.writer(csvFile)
#e=csvFile.encode('abc.csv')
f = codecs.open('abc.csv',encoding='utf-8', mode='a')
for tweet in tweepy.Cursor(api.search,
q="cool",
since="2021-11-30",
until="2021-12-01",
lang="en").items(100):
#Write a row to the csv file/ I use encode utf-8
  print (tweet);
csvWriter.writerow([tweet])
```

Figure 23: Tweeter data Download

References