----------Report on----------

“STOCK PREDICTIVE SENTIMENT ANALYSIS”

BACHELOR OF ENGINEERING IN
Artificial Intelligence and Machine Learning Engineering

Name: Hemanth RangeGowda S P

(Mangalore Institute of Technology & Engineering)

ABSTRACT

Prediction in machine learning **refers to the output of an algorithm after it has been trained on a historical dataset and applied to new data when forecasting the likelihood of a particular outcome.** In this project, we investigate the impact of sentiment expressed through Twitter tweets on stock price prediction. Twitter is the social media platform which provides a free platform for each individual to express their thoughts publicly. Specifically, we fetch the live twitter tweets of the particular company using the API. All the stop words, special characters are extracted from the dataset. The filtered data is used for sentiment analysis using Naïve bayes classifier. Thus, the tweets are classified into positive, negative and neutral tweets. To predict the stock price, the stock dataset is fetched from yahoo finance API. The stock data along with the tweets data are given as input to the machine learning model to obtain the result. LSTM deep learning algorithm is used as a model to predict the stock market price. The obtained prediction value is compared with the actual stock market value. The effectiveness of the proposed project on stock price prediction is demonstrated through experiments on several companies like Apple, Amazon, Microsoft using live twitter data and daily stock data.

CHAPTER 1.

INTRODUCTION

Introduction to ML

In this model, we test a hypothesis based on the premise of behavioral economics, that the emotions and moods of individuals affect their decision making process, thus, leading to a direct correlation between “public sentiment and market sentiment”. We perform sentiment analysis on publicly available Twitter data to find the public mood and the degree of membership into 3 classes- Positive, Negative, Neutral. We use these moods and previous days’ Stock price data to predict future stock movements and then use the predicted values in our portfolio management strategy.

Problem Statement

Stock exchange is a subject that is highly affected by economic, social, and political factors. There are several factors e.g. external factors or internal factors which can affect and move the stock market. Stock prices rise and fall every second due to variations in supply and demand. Various Data mining techniques are frequently involved to solve this problem. But techniques using machine learning will give a more accurate, precise and simple way to solve such issues related to stock and market prices. “Stock Price Prediction Using Twitter Sentiment Analysis” a method for predicting stock prices is developed using news articles. The changes in stock prices of a company, the rises and falls, are correlated with the public opinions being expressed in tweets about that company. Understanding people's opinion from a piece of text is the objective of sentiment analysis. Positive news and tweets in social media about a company would definitely encourage people to invest in the stocks of that company and as a result the stock price of that company would increase. A prediction model for finding and analyzing correlation between contents of tweets and stock prices and then making predictions for future prices can be developed by using machine learning.

CHAPTER 2.

SYSTEM ANALYSIS

1. Existing system

In the last decade, sentiment analysis has carried great importance because of the huge amount of textual data on news and social media platforms. There is a significant amount of research on mining opinions of users for different application areas. Ali performed sentiment analysis to detect transportation entities in the large corpus and to detect traffic accidents to reduce serious injuries [43]. Basiri proposed a sentiment analysis method on the Twitter dataset using the attention-based bidirectional CNN-RNN deep model. Li proposed a bidirectional emotional recurrent unit for exploring the emotion in conversation. Using the same trend for stock prediction, some researchers have started to utilize sentiment analysis to analyze the stock market's movements. In 2011, J. Bollen proposed a sentiment analysis method to predict the movement of the Dow Jones Industrial Average (DJIA) stock market based on the famous microblogging site Twitter. Khatri and Srivastava explored the relationship between text sentiment from tweets, comments on the Stock Twist website, and the stock price of Facebook, Apple, Google, Oracle, and Microsoft stocks from Yahoo Finance. In 2017, Urolagin used an SVM classifier to perform sentiment classification and predict stock market status based on stock price data from Yahoo Finance and social-media data from Twitter. He explored an association between tweet features and the stock prices of a company. Not long after that, Chakraborty proposed a stock-market-movement prediction model based on tweets sentiment of NETFLIX stock. The SVM classifier was used for sentiment classification, while the stock movement prediction model was trained by applying the boosted regression tree. In 2020, Khan utilized algorithms on social media and financial news data to discover the impact of both data on stock movement.

2.Proposed System

This section explains the proposed methodology, the standard dataset from, LSTM model.

A. Transfer Learning

Transfer learning is one of the machine learning models which uses the knowledge gained from solving one problem is incorporated to solve another problem. It is evident that Transfer learning solves many problems within short intervals of time. Transfer Learning is incorporated whenever there is any need to reduce computation cost, achieve accuracy with less training

B. Preparation of Training Dataset

Stock data extracted is not completely understandable because of public holidays and weekends where the stock market does not function. There are missing in the stock value. These empty values can be approximately used in a simple way. Consider, the stock value on a day is x and the next value present is y with some missing in between. So, the first value is estimated as (y+2)/2 and the same method is used to fill the missing values. Extracted tweets contain many stop words, unnecessary data like special characters, URLs, pictures. These tweets are pre-processed to obtain the emotion of the public. For pre-processing of data, we employ three steps of filtering: Tokenization: Each tweet is split into individual words called tokens. This process is done to break the text, separated by whitespace characters. Removal of stop words: Words like "a", "an", "the", "he", "she", "by", "on" are not required for sentiment analysis. These are called stop words, which is removed before the sentiment analysis process. Regex Matching: Special characters such as "URL", "!", "#", "@" are all removed and replaced by whitespaces.

2. Objective of the System

In the past decade, a lot of research has gone into Stock Price Prediction. The primary objective of Stock Price Prediction is to improve the predictions made by Machine learning models. Predicting the value based on past data will not return an effective prediction. Therefore we are using Twitter tweets sentiment analysis so that the effectiveness in the prediction can be achieved

CHAPTER 3.

REQUIREMENT ANALYSIS

Hardware Requirement Specification

- Processor: Minimum 1 GHz; Recommended 2GHz or more
- Ethernet connection (LAN) OR a wireless adapter (Wi-Fi)
- Hard Drive: Minimum 32 GB; Recommended 64 GB or more
- Memory (RAM): Minimum 1 GB; Recommended 4 GB or above
- Sound card w/speakers
- Some classes require a camera and microphone

Software Requirement Specification

- Platform: Anaconda
- Environment: Python 3
- Modules: Matplotlib, NumPy, Pandas, Keras, Tensorflow.

CHAPTER 4.

DESIGN ANALYSIS

Logistic Regression

Linear Regression is a linear approach to modeling the relationship between a scalar response and one or more explanatory variables (also known as dependent and independent variables). A stock's price and time-period determine the system parameters for linear regression, making the method universally applicable. Gradient Descent algorithm is used for tuning the weights of the model to attain less

Support Vector Machines

Support Vector Machines (SVM) are supervised algorithm that works for both classification and regression problems. Support vectors are coordinate points in space, formed using the attributes of a data point. Briefly, for an N-dimensional dataset, each data point is plotted on an N-dimensional space using all its feature vector values as a coordinate point. Classification between the classes is performed by finding a hyperplane in space that clearly separates the distinct classes. SVM works best for high-dimensional data. The important aspect of implementing the SVM algorithm is finding the hyperplane. Two conditions are to be met in the order given while choosing the right hyperplane.

- The hyperplane should classify the classes most accurately.
- The margin distance from the hyperplane to the nearest data point must be maximized

For a low dimensional dataset, the method of kernel trick in SVM introduces additional features to transform the dataset to high dimensional space and thereby make identifying the hyperplane achievable. The linear solver-based SVM is a better implementation of SVM in terms of time complexity. The complexity scales between O(n samples X n2 samples) and O(n samples X n3 samples).

LSTM

Long short-term memory (LSTM) is an artificial neural network used in the fields of artificial intelligence and deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. Such a recurrent neural network (RNN) can process not only single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition, machine translation, robot control, video games, and healthcare. LSTM has become

the most cited neural network of the 20th century. The name of LSTM refers to the analogy that a standard RNN has both "long-term memory" and "short-term memory". The connection weights and biases in the network change once per episode of training, analogous to how physiological changes in synaptic strengths store long-term memories; the activation patterns in the network change once per time-step, analogous to how the moment-to-moment change in electric firing patterns in the brain store short-term memories. The LSTM architecture aims to provide a short-term memory for RNN that can last thousands of timesteps, thus "long short-term memory".

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate.The cell remembers values over arbitrary time intervals and the three *gates* regulate the flow of information into and out of the cell.

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications.

An LSTM module (or cell) has 5 essential components which allows it to model both long-term and short-term data. Hidden state - This is output state information calculated w.r.t current input, previous hidden state and current cell input which you eventually use to predict the future stock market prices.

CHAPTER 5.

IMPLEMENTATION

Implementation is the stage where the theoretical design is turned into a working system. The Most crucial stage in achieving a new successful system and in giving confidence in the new system to the users that it will work efficiently and effectively.

The system can be implemented only after thorough testing is done and if it is found to work according to the specification. It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the changeover, and an evaluation of change over methods apart from planning.

Two major tasks of preparing the implementation are education and training of the users and testing of the system. The more complex the system being implemented, the more involved will be the system analysis and design effort required just for implementation.

The implementation phase comprises several activities. The required hardware and software acquisition is carried out. The system may require some software to be developed. For this, programs are written and tested. The user then changes over to his new fully tested system and the old system is discontinued.

Data Collection

The first step in implementing the Stock Price Prediction is to collect stock data. This data includes the historic data of the company. Along with the stock data it is important to collect twitter data which provide insights on how the market opened on a particular day based on any positivity or negativity in twitter. We used Kaggle Netflix dataset which has already collected the historic twitter data for a given span and company stock data for the same span of time

Python Library

The python libraries which were used in the model building are:

```
import numpy as np
import tensorflow as tf
from keras.callbacks import ModelCheckpoint
from keras.models import Sequential
from keras.layers import
LSTM,Conv1D,Conv2D,MaxPooling2D,MaxPooling1D,Flatten
```

```
from keras.layers import Dense, Dropout
import pandas as pd
# from keras.optimizers import Adam
from matplotlib import pyplot as plt
from sklearn.preprocessing import StandardScaler,MinMaxScaler
import seaborn as sns
#from datetime import datetime

from tensorflow.compat.v1.keras.layers import CuDNNLSTM,Bidirectional
```

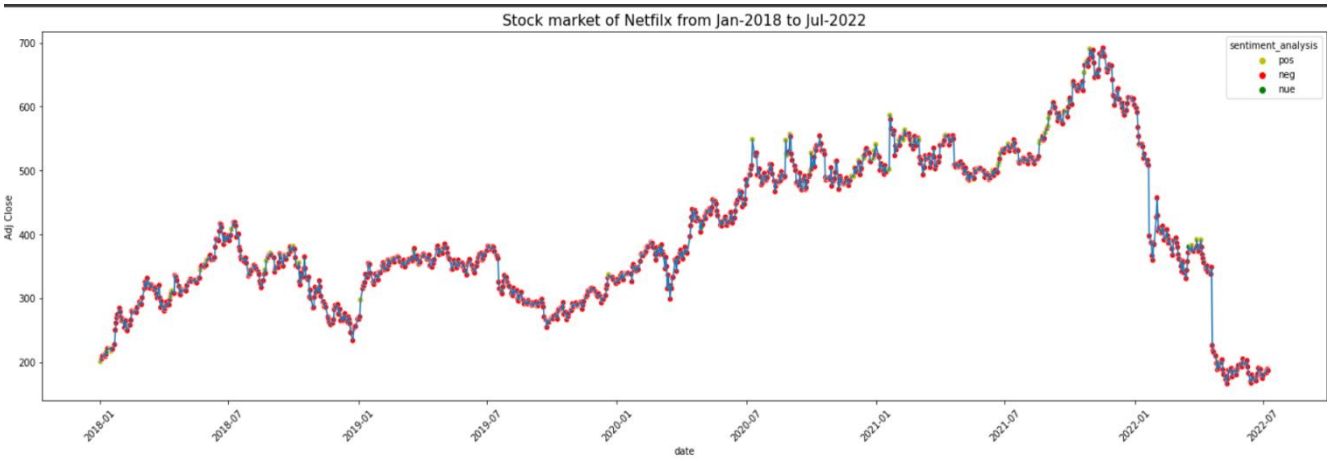
Data Visualization

Visualizing the data gives more understanding of the problem and the type of solution to be built. The distribution of classes, the number of instances under each category, the spread of the data, and the correlation between the features and clustering are a few methods to visualize the data. Python and R provide statistical functions for data visualization.

Feature Analysis

Primarily, we are importing the required modules for the model to work,

```
import seaborn as sns
plt.figure(figsize=(25,7));
sns.lineplot(x=df["date"],y=df["Adj Close"])
df['sentiment_analysis']=df['P_mean']
df['sentiment_analysis']=df['sentiment_analysis'].apply(lambda x: 'pos'
if x>0 else 'nue' if x==0 else 'neg')
sns.scatterplot(x=df["date"],y=df['Adj
Close'],hue=df['sentiment_analysis'],palette=['y','r','g'])
plt.xticks(rotation=45);
plt.title("Stock market of Netfilx from Jan-2018 to Jul-
2022",fontsize=16);
```



And the result is in the Table 1.1

Table 1.1

Algorithm	**Accuracy**
LSTM (without twitter data)	45.52%
LSTM (with twitter data)	67.16%

TESTING

Once training of the model is done,it is important to test the model's accuracy of the data which is not used while training.This broad purpose of this exercise has been to arrive at trading strategies which could help in the real-world application of the models developed. The study could not approach to those levels due to several constraints and limitations as described above. Two Regressor that were used viz LSTM and Random Forest have been used to predict the next day value and the RMSEs are considered as the evaluation metrics. Since this was a regression exercise prediction with certain confidence levels couldn't be achieved. Thus, a simple intuitive analysis of RMSE values was carried out to gauge the appropriateness of the values predicted by our models. The results as achieved from the models are indicated below

Algorithm for different scenario	**MSE error**
LSTM-open (without tweets)	162.333
LSTM-open (with tweets)	87.6900
LSTM-close (without tweets)	175.9912
LSTM-close (with tweets)	122.3798

CHAPTER 6.

SNAPSHOTS

Feature Scaling:

LSTM Model

Data scaling for LSTM because uses sigmoid and tanh that are sensitive to magnitude

```
scaler = MinMaxScaler()
scaler = scaler.fit(df_for_training)
df_for_training_scaled = scaler.transform(df_for_training)

scaler_for_inference = MinMaxScaler()
scaler_for_inference.fit_transform(df_for_training.loc[:,['Open','Adj Close']])

df_for_training_scaled
# df_for_training_scaled=df_for_training.copy()
# df_for_training_scaled=df_for_training_scaled.to_numpy()
```

Model building:

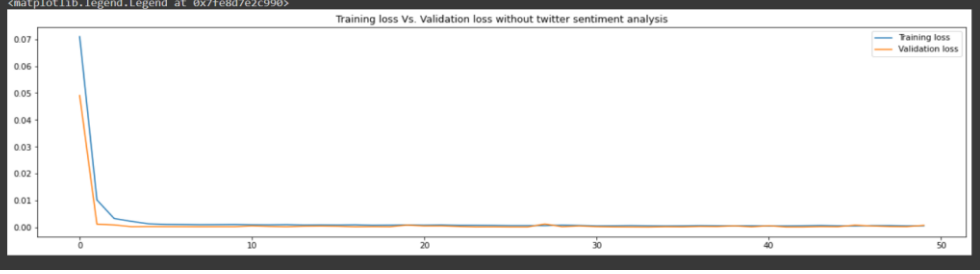
```
def build_model(input_shape):
    tf.random.set_seed(seed)
    cnn_lstm_model = Sequential()

    cnn_lstm_model.add(Conv1D(filters=128, kernel_size=2, strides=1, padding='valid', input_shape=input_shape))
    cnn_lstm_model.add(MaxPooling1D(pool_size=2, strides=2))

    cnn_lstm_model.add(Conv1D(filters=64, kernel_size=2, strides=1, padding='valid'))
    cnn_lstm_model.add(MaxPooling1D(pool_size=1, strides=2))
    # cnn_lstm_model.add(MaxPooling1D(pool_size=1, strides=2))

    cnn_lstm_model.add(Bidirectional(LSTM(256, return_sequences=True)))
    cnn_lstm_model.add(Dropout(0.2))
    cnn_lstm_model.add(Bidirectional(LSTM(256, return_sequences=True)))
    cnn_lstm_model.add(Dropout(0.2))

    cnn_lstm_model.add(Dense(32, activation='relu'))


    cnn_lstm_model.add(Dense(trainY.shape[2], activation='relu'))

    # cnn_lstm_model.build(input_shape=(trainX.shape[0], trainX.shape[1], trainX.shape[2]))

    cnn_lstm_model.compile(optimizer='adam', loss='mse')
    cnn_lstm_model.summary()
    return cnn_lstm_model
```

Validating the error on training data without twitter data :

Plotting Training and validation loss

```
plt.figure(figsize=(20,5))
plt.plot(history_without_twitter.history['loss'], label='Training loss')
plt.plot(history_without_twitter.history['val_loss'], label='Validation loss')
plt.title('Training loss Vs. Validation loss without twitter sentiment analysis')
plt.legend()
```

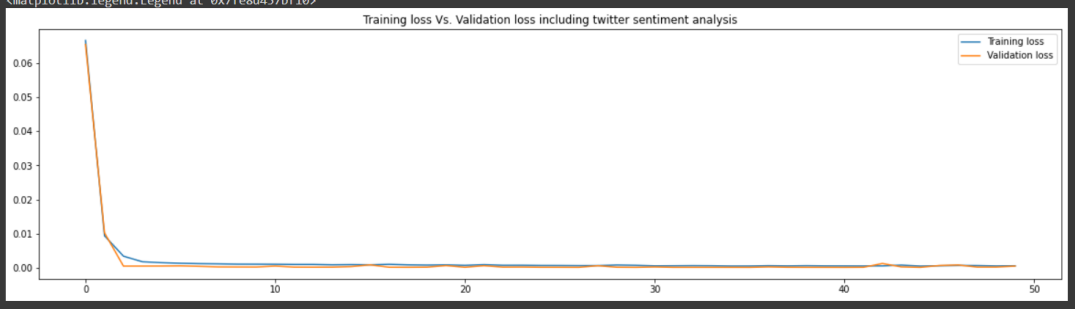
```
<matplotlib.legend.Legend at 0x7fe8d7e2c990>
```



Validating the error on training data with twitter data :

```
plt.figure(figsize=(20,5))
plt.plot(history_twitter.history['loss'], label='Training loss')
plt.plot(history_twitter.history['val_loss'], label='Validation loss')
plt.title('Training loss Vs. Validation loss including twitter sentiment analysis')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7fe8d457bf10>
```



Computing Accuracy:

Computing training accuracy

```
training_dates= df_for_training.index[:X_train_lstm_without_twitter.shape[0]]
#Make prediction
training_prediction_without_twitter = cnn_lstm_model_without_twitter.predict(X_train_lstm_without_twitter)

training_prediction_twitter = cnn_lstm_model_twitter.predict(X_train_lstm_twitter)

training_prediction_without_twitter=training_prediction_without_twitter.reshape(training_prediction_without_twitter.shape[0], training_prediction_without_twitter.shape[2])

training_prediction_twitter=training_prediction_twitter.reshape(training_prediction_twitter.shape[0], training_prediction_twitter.shape[2])

y_train_pred_lstm_without_twitter = scaler_for_inference.inverse_transform(training_prediction_without_twitter)

y_train_pred_lstm_twitter = scaler_for_inference.inverse_transform(training_prediction_twitter)

y_train_lstm_reshaped_without_twitter=y_train_lstm_without_twitter.reshape(y_train_lstm_without_twitter.shape[0], y_train_lstm_without_twitter.shape[2])

y_train_actual_lstm = scaler_for_inference.inverse_transform(y_train_lstm_reshaped_without_twitter)
```

Feature Scaling:

LSTM Model

Data scaling for LSTM because uses sigmoid and tanh that are sensitive to magnitude

```
scaler = MinMaxScaler()
scaler = scaler.fit(df_for_training)
df_for_training_scaled = scaler.transform(df_for_training)

scaler_for_inference = MinMaxScaler()
scaler_for_inference.fit_transform(df_for_training.loc[:,['Open','Adj Close']])

df_for_training_scaled
# df_for_training_scaled=df_for_training.copy()
# df_for_training_scaled=df_for_training_scaled.to_numpy()
```

Model building:

```
def build_model(input_shape):
    tf.random.set_seed(seed)
    cnn_lstm_model = Sequential()

    cnn_lstm_model.add(Conv1D(filters=128, kernel_size=2, strides=1, padding='valid', input_shape=input_shape))
    cnn_lstm_model.add(MaxPooling1D(pool_size=2, strides=2))

    cnn_lstm_model.add(Conv1D(filters=64, kernel_size=2, strides=1, padding='valid'))
    cnn_lstm_model.add(MaxPooling1D(pool_size=1, strides=1))
    # cnn_lstm_model.add(MaxPooling1D(pool_size=1, strides=2))

    cnn_lstm_model.add(Bidirectional(LSTM(256, return_sequences=True)))
    cnn_lstm_model.add(Dropout(0.2))
    cnn_lstm_model.add(Bidirectional(LSTM(256, return_sequences=True)))
    cnn_lstm_model.add(Dropout(0.2))

    cnn_lstm_model.add(Dense(32, activation='relu'))


    cnn_lstm_model.add(Dense(trainY.shape[2], activation='relu'))

    # cnn_lstm_model.build(input_shape=(trainX.shape[0], trainX.shape[1], trainX.shape[2]))

    cnn_lstm_model.compile(optimizer='adam', loss='mse')
    cnn_lstm_model.summary()
    return cnn_lstm_model
```

Creating training data:

Train test split for LSTM

```
from sklearn.model_selection import train_test_split

X_train_lstm_without_twitter, X_test_lstm_without_twitter, y_train_lstm_without_twitter, y_test_lstm_without_twitter = train_test_split(trainX[:,:,1:-1], trainY, test_size=0.2, shuffle

X_train_lstm_twitter, X_test_lstm_twitter, y_train_lstm_twitter, y_test_lstm_twitter = train_test_split(trainX, trainY, test_size=0.2, shuffle=False)

X_train_lstm_without_twitter.shape,X_train_lstm_twitter.shape

((8008, 5, 6), (8008, 5, 7))
```

Train validation split for LSTM

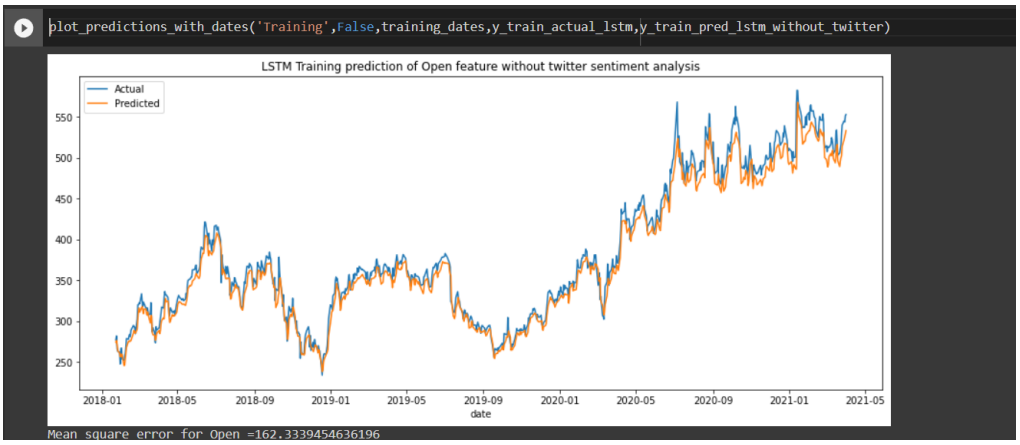
```
from sklearn.model_selection import train_test_split

X_train_lstm_without_twitter, X_val_lstm_without_twitter, y_train_lstm_without_twitter, y_val_lstm_without_twitter = train_test_split(X_train_lstm_without_twitter, y_train_lstm_withou

X_train_lstm_twitter, X_val_lstm_twitter, y_train_lstm_twitter, y_val_lstm_twitter = train_test_split(X_train_lstm_twitter, y_train_lstm_twitter, test_size=0.1, shuffle=False)

X_train_lstm_without_twitter.shape,X_train_lstm_twitter.shape
```

Fit the model:

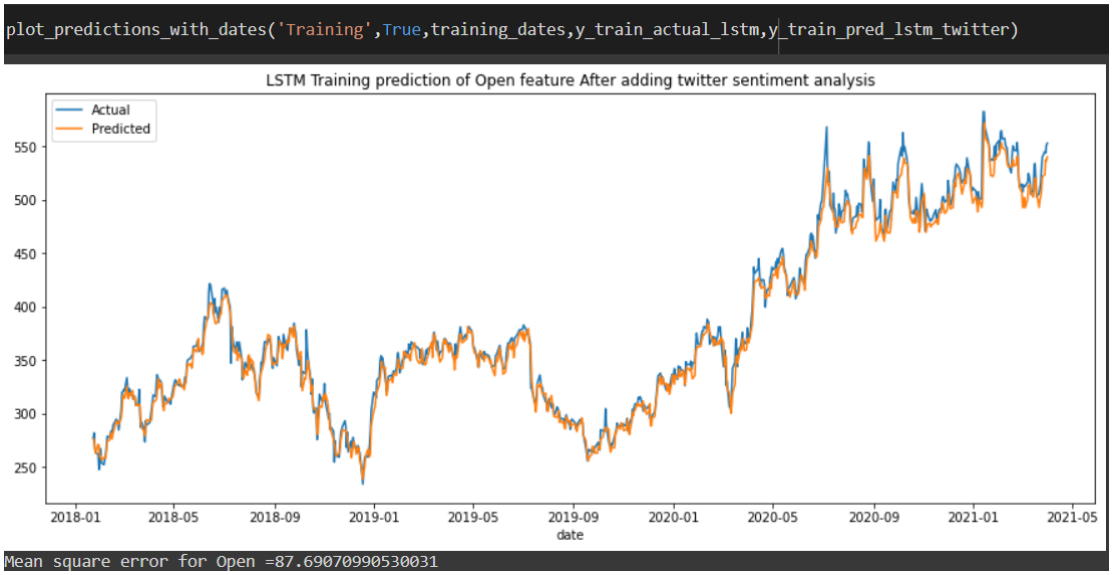
```
# fit the model

cnn_lstm_model_without_twitter=build_model((X_train_lstm_without_twitter.shape[1],X_train_lstm_without_twitter.shape[2]))
cnn_lstm_model_twitter=build_model((X_train_lstm_twitter.shape[1],X_train_lstm_twitter.shape[2]))

history_without_twitter = cnn_lstm_model_without_twitter.fit(X_train_lstm_without_twitter, y_train_lstm_without_twitter, epochs=50, batch_size=64, validation_data=(X_val_lstm_without

history_twitter = cnn_lstm_model_twitter.fit(X_train_lstm_twitter, y_train_lstm_twitter, epochs=50, batch_size=64, validation_data=(X_val_lstm_twitter, y_val_lstm_twitter), verbose=1
```

Final Outputs:

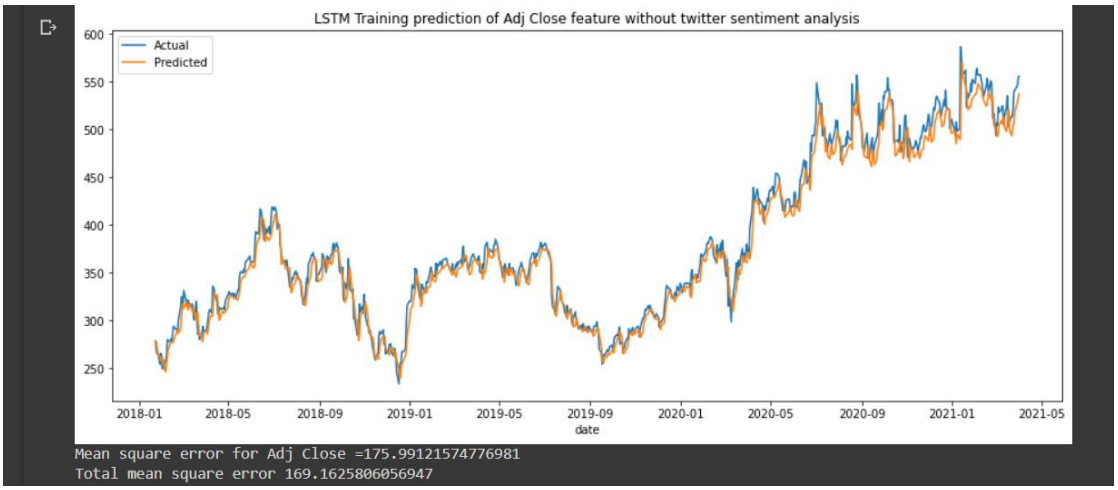
- **Open without twitter data**



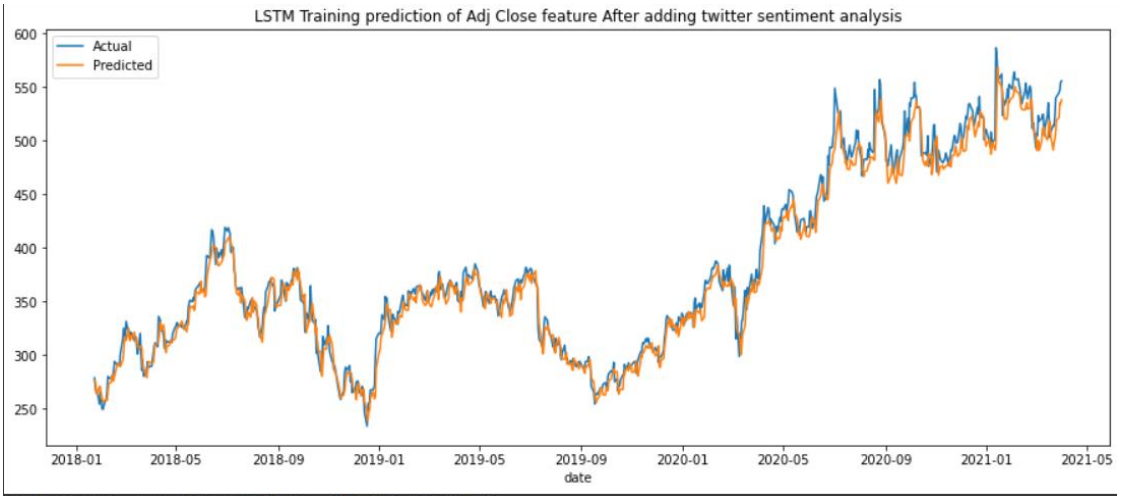
- **Open with twitter data**



- **Close without twitter data**



- **Close with twitter d**




```
Mean square error for Adj Close =157.0689983743797
Total mean square error 122.37985413984003
```

CHAPTER 7.

CONCLUSION

In this study, we presented a new multichannel network for predicting stock prices based on high-demand stock (NETFLIX) selected by the size of their market capitalization. We first employed the Natural Language Toolkit to perform sentiment analysis for text content in Twitter tweets to obtain insights into users' sentiments about a particular stock. From classifying sentiment analysis results, we obtained an overall sentiment score for a given day for the stock. We then generated candlestick chart images based on historical time series data by using computer graphic techniques to visualize the daily price and stock movement for a given stock. These two types of data were input into our joint multi channel network. Our joint network consisted of two branches: (1) the first branch included a LSTM without twitter data performing classifications based on sentiment analysis. (2) The second branch contained a LSTM with twitter data performing stock price prediction based on given data. The outputs of the two branches were concatenated and fed into a standard set of dense layers where the last layer computed the data and predicted the stock movement for the near future. The experiment results 23.53% 34.78% 62.31% 66.67% indicated that our proposed model achieved promising results for stock prediction and outperformed networks using either single sentiment data or candlestick charts alone. This indicates that using both types of data is more effective in the prediction of stock trends because both types of data can change and affect a stock's price movement and traders' decisions. The experiment results also indicated that the performance of stock prediction over longer periods of time achieved a more favorable result than did predictions made over shorter periods of time. Our proposed method achieved the most favorable performance with 75.38% accuracy for NETFLIX stock over a 10-day time period

CHAPTER 10.

REFERENCE

[1] A.E. Stefano Baccianella and F. Sebastiani. Sentiwordnet 3.0: An enhanced lexical resources for sentiment analysis and opinion mining. *In LREC. LREC.*

[2] A. Lapedes and R. Farber. Nonlinear signal Processing using neutral network: Prediction and System modeling. *In Los Alamos National Lab Technical Report.*

[3] G.P. Gang Leng and T. M. McGinnity. An on-line algorithm for creating self – organizing fuzzy neural networks. *Neural Networks, 17(10):1477-1493.*

- *https://www.sciencedirect.com/science/article/abs/pii/S187775031100007X* , By J Bollen.
- *https://www.geeksforgeeks.org/twitter-sentiment-analysis-using-python/*
- *https://towardsdatascience.com/sentiment-analysis-for-stock-price-prediction-in-python-bed40c65d178* - *By James briggs*