VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgaum-590018

A PROJECT REPORT **(15CSP85)** ON

"STOCK MARKET PREDICTION USING ML"

Submitted in Partial fulfillment of the Requirements for the Degree of

Bachelor of Engineering in Computer Science & Engineering

By

RAHUL PRASAD MAJGI (1CR16CS123)

VIGNESH BALCHAND (1CR16CS178)

SUNAYANA V RAO (1CR16CS167)

Under the Guidance of,

Ms. Savitha S

Assistant Professor, Dept. of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI, BANGALORE-560037

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI, BANGALORE-560037

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the project work entitled **"STOCK MARKET PREDICTION USING ML"** carried out by **Mr. RAHUL PRASAD MAJGI**, USN **1CR16CS123**, **Mr. VIGNESH BALCHAND**, USN **1CR16CS178**, **Ms. SUNAYANA V RAO** , USN **1CR16CS167,** bonafide students of CMR Institute of Technology, in partial fulfillment for the award of **Bachelor of Engineering** in Computer Science and Engineering of the Visveswaraiah Technological University, Belgaum during the year 2019-2020. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library.

The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

.

Ms. Savitha S	**Dr. Prem Kumar Ramesh**	**Dr. Sanjay Jain**
Assistant Professor	**Professor & Head**	**Principal**
Dept. of CSE, CMRIT	**Dept. of CSE, CMRIT**	**CMRIT**

DECLARATION

We, the students of Computer Science and Engineering, CMR Institute of Technology, Bangalore declare that the work entitled "**STOCK MARKET PREDICTION USING ML**" has been successfully completed under the guidance of Ms. Savitha S, Asst. Prof., Computer Science and Engineering Department, CMR Institute of technology, Bangalore. This dissertation work is submitted in partial fulfillment of the requirements for the award of Degree of Bachelor of Engineering in Computer Science and Engineering during the academic year 2019 - 2020. Further the matter embodied in the project report has not been submitted previously by anybody for the award of any degree or diploma to any university.

Place:

Date:

Team members:

RAHUL PRASAD MAJGI(1CR16CS123) _____

VIGNESH BALCHAND(1CR16CS178) _____

SUNAYANA V RAO (1CR16CS167) _____

ABSTRACT

The project aims to predict the prices of a basket of stocks on the NSE/BSE with an acceptable degree of accuracy. This study, based on the demand for stock price prediction and the practical problems it faces, compared and analysed a variety of neural network prediction methods, and finally chose LSTM (Long Short-Term Memory, LSTM) neural network. Then, through in-depth study on how to predict the stock price by the LSTM, the feasibility of the method and the applicability of the model are analysed, and finally the conclusion is drawn. It is found that historical information is very important to investors as the basis of investment decisions. Past studies have used opening and closing prices as key new predicators of financial markets, but extreme maxima and minima may provide additional information about future price behaviour. Therefore, the index of three representative stocks in stock market are selected as the research objects, and the key data collected from them include the opening price, closing price, lowest price, highest price, date and daily trading volume. The results show that although LSTM neural network model has some limitations, such as the time lag of prediction, but with attention layer, it can predict stock prices. Its main principle is to discover the role of time series through analysing the historical information of the stock market, and to deeply explore its internal rules through the selective memory advanced deep learning function of LSTM neural network model, so as to achieve the prediction of stock price trend.

ACKNOWLEDGEMENT

We take this opportunity to express our sincere gratitude and respect to **CMR Institute of Technology, Bengaluru** for providing us a platform to pursue our studies and carry out our final year project

We have a great pleasure in expressing our deep sense of gratitude to **Dr. Sanjay Jain,** Principal, CMRIT, Bangalore, for his constant encouragement.

We would like to thank **Dr. Prem Kumar Ramesh,** Professor and Head, Department of Computer Science and Engineering, CMRIT, Bangalore, who has been a constant support and encouragement throughout the course of this project.

We consider it a privilege and honor to express our sincere gratitude to our guide, **Ms. Savitha S, Assistant Professor,** Department of Computer Science and Engineering, for the valuable guidance throughout the tenure of this review.

We also extend our thanks to all the faculty of Computer Science and Engineering who directly or indirectly encouraged us.

Finally, We would like to thank our parents and friends for all their moral support they have given us during the completion of this work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ANN	**Artificial Neural Network**
API	**Application Program Interface**
CNN	**Convolutional Neural Network**
EMA	**Exponential Moving Average**
LSTM	**Long Short-Term Memory**
ML	**Machine Learning**
SVM	**Support Vector Machines**

CHAPTER 1

INTRODUCTION

Stock markets in general have an air of un-predictability A stock market, equity market or share market is the aggregation of buyers and sellers (a loose network of economic transactions, not a physical facility or discrete entity) of stocks (also called shares), which represent ownership claims on businesses; these may include securities listed on a public stock exchange, as well as stock that is only traded privately. Examples of the latter include shares of private companies which are sold to investors through equity crowdfunding platforms. Stock exchanges list shares of common equity as well as other security types, e.g. corporate bonds and convertible bonds.

Participants in the stock market range from small individual stock investors to larger investors, who can be based anywhere in the world, and may include banks, insurance companies, pension funds and hedge funds. Their buy or sell orders may be executed on their behalf by a stock exchange trader.

1.1 Complex platform and multifaceted problem

They have been a complex platform and thus this problem is a multifaceted one. Stock markets in general are very complex in nature and The Stock Market itself is shaped by expectations. There are markets and derivatives markets (markets based on markets) that trade solely on the expected future of a firm, commodity or currency. These expectations change as rapidly as humans change their minds - which is pretty much infinite. Apart from this there are many more factors that influence the working of the stock market, some of which are investor sentiment, customer feedback and even behavioural aspects of the employees and company representatives.

1.2 Predictions – challenging, great financial potential

The Ability to perform accurate predictions on the stock market not only adds financial value but is also challenging from an academic standpoint. Stock market prediction is the act of trying to determine the future value of a company stock or other financial instrument traded on an exchange. The successful prediction of a stock's future price could yield significant profit. Information on insider trading is available on websites of stock exchanges and can be used to predict future prices.

Predicting how the stock market will perform is one of the most difficult things to do. There are so many factors involved in the prediction – physical factors vs. psychological, rational and irrational behaviour, etc. All these aspects combine to make share prices volatile and very difficult to predict with a high degree of accuracy. Can we use machine learning as a game changer in this domain? Using features like the latest announcements about an organization, their quarterly revenue results, etc., machine learning techniques have the potential to unearth patterns and insights we didn't see before, and these can be used to make unerringly accurate predictions

1.3 Relevance of the Project

One of the most important reasons for predicting the market's movement is that several investors believe that the only time to invest in the market is when it is going up. When the market falls, such investors would like to stay away and return only when they are confident that the market would rise again.

Machine Learning fundamentals are used to predict stock market prices and also some algorithms use social sentiment as well as historical data to perform predictions.

Predicting short-term movement of any stock or the market in general, not only calls for an ability to correctly predict all these parameters but also an ability to predict how the majority of investors would react to each of these events. It is beyond the scope of almost all investors to correctly and consistently predict these things.

1.4 Scope of the Project

The project aims to predict the prices of a basket of stocks on the NSE/BSE with an acceptable degree of accuracy. By having an idea about the price of a stock in the market prior to its sale, we will know beforehand which stock to purchase thus making a profit. The successful prediction of a stock's future price could yield significant profit. The efficient-market hypothesis suggests that stock prices reflect all currently available information and any price changes that are not based on newly revealed information thus are inherently unpredictable.

1.5 Chapter Wise Summary

Chapter 1:

This contains the introduction of the project. Importantly, it explains what stock markets are, talks about their nature and complexity and about predicting their prices and the respective impact of that.

Chapter 2:

Here we can see the objectives of our project and the methodologies and processes that we are going to use to be able to achieve our end result.

Chapter 3:

Contains the Literature survey of the papers we have referred to and whose concepts we will be modifying and implementing in our project.

Chapter 4:

Has a list of all the requirements of our project. The overview of the project and the goal and scope of our project. Who our consumers will be and other similar information will be found here.

Chapter 5:

Chapter 5 of this report contains our problem statement and information about what systems already exist and how our implementations are different from them.

Chapter 6:

Here we have explained our current progress with respect to the project along with the future procedures/course of plan we will be following to successfully complete the project we have.

CHAPTER 2

LITERATURE SURVEY

Literature survey is the process in which a complete and comprehensive review is conducted encompassing both the published and unpublished work from other alternative sources of information. This review is conducted in the domains of specific interest to the person or researcher. Further, the results of this process are documented.

This entire process comes in aid of the researcher to address the important and relevant aspects of the research that had not been addressed prior to the conduction of this research. Therefore, it can be understood that the conduction of literature survey is necessary for the process of gathering secondary data for the research which might prove to be extremely helpful in the research and also designing the architecture of the project. There can be multiple reasons behind the purpose of conducting literature survey.

2.1 Analysis of Investor sentiment and its effect on stock market volatility

Here mainly study the specific mechanism of investor sentiment affecting stock market volatility. With the help of Pollet and Wilson's theory of volatility decomposition, it performs a comparative analysis based on big data strategy and sources.

This paper collects the data of web news emotion index, web search volume, social network emotion index, social network heat index, and establishes corresponding analysis index. After correlation analysis and Granger causality tests, it extracts the indicators which have significant correlation with the financial market and brings them into forecasting analysis.

The model constructs market volatility index and analyses the correlation between investor sentiment and stock price changes. In empirical study, the deviation between stock price and value is introduced as an explanatory variable, and the logarithmic return of stock is used to measure the volatility of stock price.

Stock market volatility index is an index to measure the volatility of the stock market. As an important index to reflect investors' sentiment, it reflects the number of investors who are willing to pay to hedge the investment risk according to the trend of the stock market and the level of volume and price.

Market volatility index can effectively measure the level of market risk, and provide an effective reference for government departments and financial institutions to study and judge risks and make macroeconomic decisions.

At the same time, the derivatives based on volatility index can provide investors with a variety of investment and hedging tools.

- mechanism of investor affecting stock market volatility.
- Collection of data to form indexes.
- web news emotion index, web search volume, social network emotion index, social network heat index, corresponding analysis index.

Principles:

- Stock market volatility index – measure volatility of stock market.
- Can effectively measure level of market risk -effective reference for decision making.
- Relies on large data of Internet behavior.
- Can explain impact of market sentiment and investor behavior on the stock market.
- New perspective and has great research value.

Proposed Methodology:

- Data Collection.
- Feature Extraction Module.
- Training Module.
- Prediction Module.

Challenges:

- Difficulty in obtaining historical data from twitter.
- Filtering unwanted tweets.
- Authentication for accessing real time Twitter data.
- Support Vector Machine proved to be the most efficient and feasible
- Using Machine Learning techniques for prediction purposes is inexpensive compared to the ground survey
- Classification of tweets as positive, negative and neutral gives a good overview of public mood.

2.2 Using Social Media Mining Technology to Assist in Price Prediction of Stock Market

There are many factors that lead to the rise and fall of financial market movement. Predictions of stock market price and its direction are quite difficult.

Statistical analysis methods stand a good chance at finding the main factor that impacts the short term stock volatility. While data mining techniques have been profitably to generate high prediction accuracy of stock price movement.

A great number of financial analysts and stock market investors' are convinced that they can make profits by employing one of the technical analysis approaches to forecast stock market.

Data Fetching

- stock history data- Yahoo and Google.
- stock comment data from financial websites using topical crawler.
- All the data stored with Hadoop distributed file system (HDFS).

Sentiment Analysis

Feature Extraction

Text parsing : fetch a vector of words and phrases describing stock comment

Usage of HowNet+

HowNet+ is a sentiment dictionary used to explicitly divide the emotion words with negative types and positive types.

Calculate sentiment index and sentiment discrepancy Index

Using the below formulae, we can calculate the above-mentioned parameters.

$$SI = ln\frac{1+N_{positive}}{1+N_{negtive}} \quad (1)$$

$$SDI = \left|1 - \left|\frac{N_{positive}-N_{negtive}}{N_{positive}+N_{negtive}}\right|\right| \quad (2)$$

Results of the experiment

SVM model predicted price closer to the actual closing price with small bias.

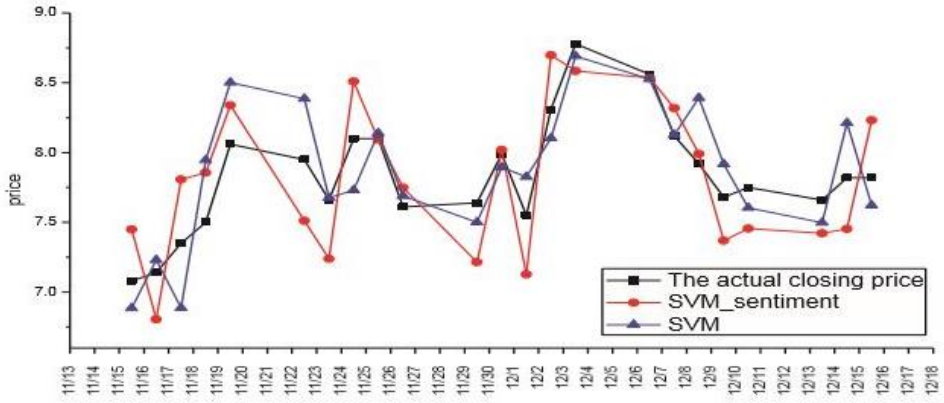

Fig 2.1 SVM model predictions

2.3 An Ensemble stock predictor and Recommender System

The model proposed takes into account the historical stock price time series data for a particular company, the tweets about that company and the overall business news headlines of the country. These parameters are defined in the form of a time series. Time series data can be defined as a chronological sequence of observations for a selected variable.The variable in this case in closing stock price. The proposed model can be classified under the non-linear time-series forecasting models.

The model aims to build on some of the previous works in this domain. Previous works include using techniques like SVM and ANN for analysing stock market trends.Apart from historical time series data, investor sentiment may cause the prices to deviate from underlying fundamentals.

The sentiment analysis is performed using Naïve Bayes algorithm to classify the sentiment as positive, negative or neutral. This investor sentiment is fed as input to LSTM model.

The model is built on data that was collected from three main sources. Historical stock price data on various companies over a period of 20 years was acquired from the official National Stock Exchange, India website. Tweets were scraped from www.twitter.com website using tweet-scraper module and filtered based on the company name. Historical news dataset was obtained from www.kaggle.com.

2.3.1 GENERIC SOLUTION

The system is built on a large dataset obtained from various sources like National Stock Exchange website (nseindia.com), company's twitter handle and various news sites. Using this data an ensemble machine learning model is built as the backbone of the application. The model contains 4 modules:

- LSTM module for capturing trends in historical closing stock prices.

- LSTM module for capturing sentiments in news headlines .

- Multivariate regression to combine LSTM (stock) + LSTM (news) + LSTM (tweets) into a single prediction module.

- CNN classifier module to generate recommendations by visualizing the trends in predicted stock prices within a specific time frame.

2.4 Stock Market Prediction based on Social Sentiments using Machine Learning

2.4.1 Proposed Methodology

Data Collection:

For tweet collection, Twitter provides robust API. There are two possible ways to gather tweets: streaming API and search API. To overcome the limitation of streaming API, we have used search API. The search API is REST API which allow users to request specific query of recent tweets. An API requires the user have an API key authentication. After authenticating using key, we were able to access through python library called Tweepy. The text of each tweet contains too much extraneous words that do not consider to its sentiment. Tweets include URLs, tags to others and many other symbols that do not have any sentiment value. To accurately obtain tweet's sentiment we need to filter noise from its original state.

First step is to split the text by space, forming a list of individual words per text which is called as list of words. We will use each word in tweet as feature to train our classifier.

Next, we remove stop words from list of total words. We have used python's Natural Language Toolkit (NLT) library to remove stopwords.Stopwords contains articles, punctuation and some extra words which do not have any sentiment value. There is a stop word dictionary which check each word in list of tokenized words against dictionary. If the word is stop word then it is filtered out.

Now, tweet contains extra symbols like "@", "#" and URLs. The word next to "@" symbol is always a username which does not add any sentiment value to text, but it is necessary to identify the subject of the tweet. Words following "#" are kept as they contain information about tweet. URLs are filtered out as they do not add any sentiment meaning to text. To accomplish all these processes, we use regular expressions that matches these symbols.

Feature Extraction Module:

After gathering large tweet corpus, we have built and train classifier for tweet sentiment analysis. We examine mainly two classifiers: Naïve Bayes and Support Vector Machine. For each classifier we extract the same features from the tweets to classify on it. To build feature set, we process each tweet and extract meaningful feature and create feature matrix by unigram technique.

For example, if positive tweet contains word "sorrow", a feature for classification would be whether or not a tweet contains the word "sorrow".

Training Module:

The generated data is used as training dataset to train the model for sentiment analysis. On inspecting the model on test dataset, we receive the tweet sentiment labels as an output. We will use this dataset for stock market prediction. We calculate total available stock tweets regarding each company and generate another dataset which contains positive, negative, neutral as well as total tweets of each day as a feature matrix. On other side we have taken stock market historical data for each day and have calculated market up as well as down direction and took it as label for dataset. In case of stock market historical data, we have used Python's yahoo-finance library.

Prediction Module:

After training our classifier, we move on to an application to look at correlation between tweet sentiment and stock market prices on each day scale. To do so, we have collected stock data as well as tweet data for same timeline as explained above. In addition, we focus on specific company stocks gathered daily data for each. After justifying a valid correlation, we are able to predict the stock value

2.4.2 Challenges

- Historical Twitter data cannot be obtained, unless it is saved by someone, so data has to be collected over a duration of a fixed number of months starting from the present date and time.
- It is necessary to filter out required data from the stream of unrelated tweets; and
- Authentication is required for accessing real time Twitter data.

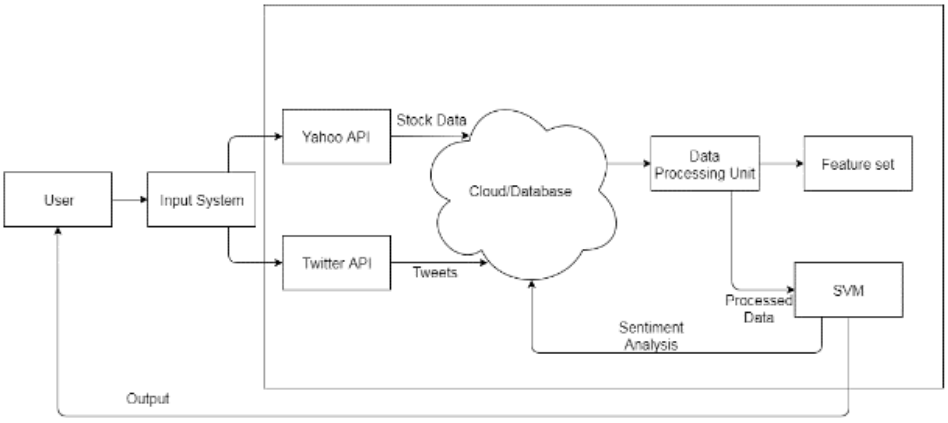

Fig 2.2 System Architecture

- Based on the comparative study that we performed, Support Vector Machine proved to be the most efficient and feasible model in predicting the stock price movement, in favor of the sentiments of the tweets.
- Using Machine Learning techniques for prediction purposes is inexpensive compared to the ground survey that would have been conducted otherwise to gauge the public mood.
- Cloud services will enable us to collect large amount of data and also store it in real time when we will get the data directly from the REST API.
- Classification of tweets as positive, negative and neutral gives a good overview of public mood. Other factors that are affecting the public mood

2.5 Stock Tracking : A New Multi-Dimensional Stock Forecasting Approach

2.5.1 Stock model

We suppose that the stock market is only weak form EMH (Efficient Market Hypothesis) that the price contains enough history information. This assumption makes it possible to infer the future price only from historical data. Generally speaking, every moving object of interest can be considered as a target, not excepting the stock. We treat the stock forecasting as a special target tracking. We say it is special because all the forecasting data can be obtained finally which implies the state space and observer space are the same.

2.5.2 Methodology

- Problem formulation
- Stock forecasting
- Transition matrix computation
- Stock tracking process
- Stock forecasting
- State vector construction and model training
- Forecasting results

Attempt to use the theory of information fusion for economic application-stock forecasting. Such efforts are present the multiresolution filter application on the typical macroeconomic a microeconomic system. The Stock Tracking approach treat the fluctuating stock as a moving target, and use a recursive method to predict its value. This approach is very simple and can be implemented easily. Its forecasting result is exciting. Using vectors to represent stock information makes it convenient to deal with multi-dimensional data and increases the flexibility of this approach greatly.

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATION

This chapter describes the requirements for the project. The Software Requirement Specification (SRS) is explained in detail, which includes overview of dissertation as well as the functional and non-functional requirements of this dissertation. An SRS document describes all data, functional and behavioural requirements of the software under production or development.

SRS is a fundamental document, which forms the foundation of the software development process. It is the complete description of the behaviour of a system to be developed. Requirement Analysis discusses the conditions to be met for a new or altered product. Requirement Analysis is critical to the successful development of a project. Requirement must be documented, measurable, testable, related to in identified business needs or opportunities, and defined to a level of detail sufficient for system design.

3.1 Functional Requirements

- Stockbrokers, also known as registered representatives in the U.S., are the licensed professionals who buy and sell securities on behalf of investors. The brokers act as intermediaries between the stock exchanges and the investors by buying and selling stocks on the investors' behalf. An account with a retail broker is needed to gain access to the markets.

- Portfolio managers are professionals who invest portfolios, or collections of securities, for clients. These managers get recommendations from analysts and make the buy or sell decisions for the portfolio. Mutual fund companies, hedge funds, and pension plans use portfolio managers to make decisions and set the investment strategies for the money they hold.
- Investment bankers represent companies in various capacities, such as private companies that want to go public via an IPO or companies that are involved in pending mergers and acquisitions. They take care of the listing process in compliance with the regulatory requirements of the stock market.

- Custodian and depot service providers, which are institution holding customers' securities for safekeeping so as to minimize the risk of their theft or loss, also operate in sync with the exchange to transfer shares to/from the respective accounts of transacting parties based on trading on the stock market.
- Market maker: A market maker is a broker-dealer who facilitates the trading of shares by posting bid and ask prices along with maintaining an inventory of shares. He ensures sufficient liquidity in the market for a particular (set of) share(s), and profits from the difference between the bid and the ask price he quotes.

3.2 Non-Functional Requirements

Non-functional requirements are the requirements which are not directly concerned with the specific function delivered by the system. They specify the criteria that can be used to judge the operation of a system rather than specific behaviours. They may relate to emergent system properties.

Non-functional requirements for this system are specified as follows:

- Data should be taken from authorized sources
- Data retrieved through api should be accurate
- Predictions made should be useful and worthwhile

3.3 Hardware Requirements

- Dedicated GPU
- 64 bit quad core processor
- 16 GB RAM

3.4 Software Requirements

- Python
- Spyder

CHAPTER 4

SYSTEM ANALYSIS AND DESIGN

This chapter gives an overview of the proposed system along with defining the system architecture and UML diagrams such as use-case and class diagrams. A UML diagram is a diagram based on the UML (Unified Modelling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

4.1 Overview of System Design

The system we are introducing will train a model using a training algorithm that we will be coming up with and based on the results of that prediction the individual can decide to purchase a stock or not. The profit or loss calculation is usually determined by the closing price of a stock for the day; hence we will consider the closing price as the target variable.

Broadly, stock market analysis is divided into two parts – Fundamental Analysis and Technical Analysis.

- Fundamental Analysis involves analysing the company's future profitability on the basis of its current business environment and financial performance.
- Technical Analysis, on the other hand, includes reading the charts and using statistical figures to identify the trends in the stock market.

Stock market prediction seems a complex problem because there are many factors that have yet to be addressed and it doesn't seem statistical at first. But by proper use of machine learning techniques, one can relate previous data to the current data and train the machine to learn from it and make appropriate assumptions.

4.2 System Architecture

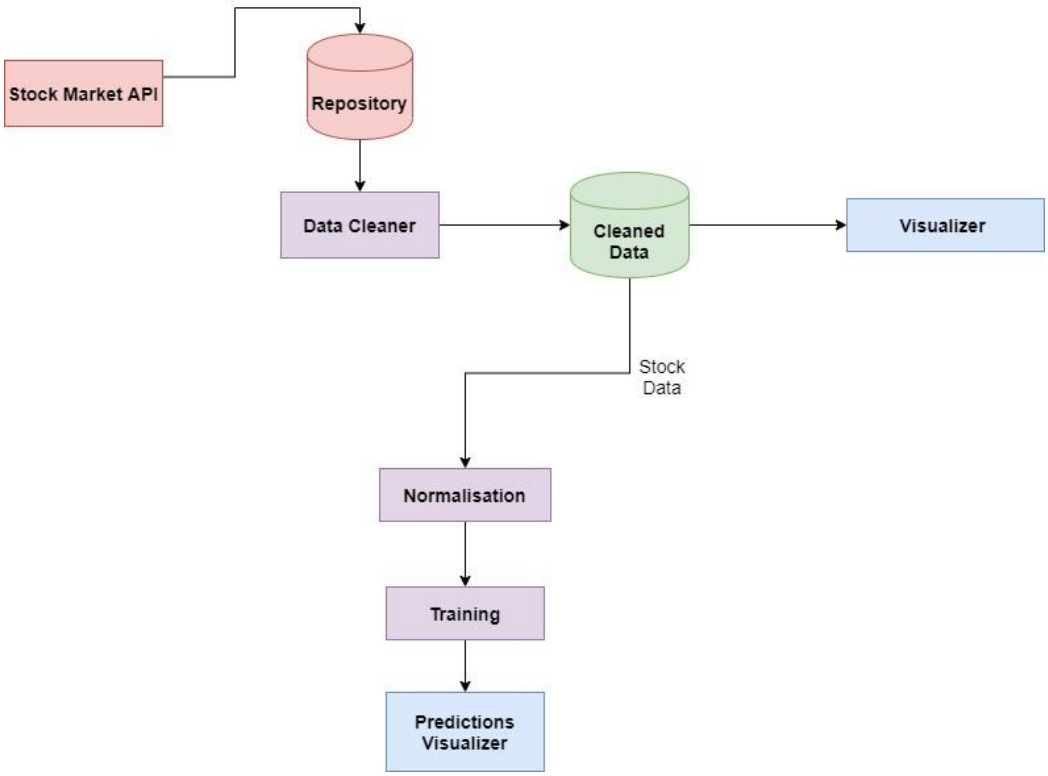

Fig. 4.1 System architecture

Once the program is started the data is automatically fetched from through a stock market API and that data is stored in a repository. The data which comes Through the API is already cleaned and filtered, this clean data is now represented visually through graphs with the help of Python visual representation Libraries.

Now, the next important process is the data normalization, here Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. After normalization, the training procedure begins where we train the model with the training algorithm and data and once that is completed, we finally visualize the predictions and the real time data side by side to cross verify our results.

4.3 Use-Case Diagram

A UML use case diagram is the primary form of system/software requirements for a new software program underdeveloped. Use cases specify the expected behaviour (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation (i.e. use case diagram). A key concept of use case modelling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behaviour in the user's terms by specifying all externally visible system behaviour.

A use case diagram is usually simple. It does not show the detail of the use cases:

- It only summarizes some of the relationships between use cases, actors, and systems.
- It does not show the order in which steps are performed to achieve the goals of each use case.

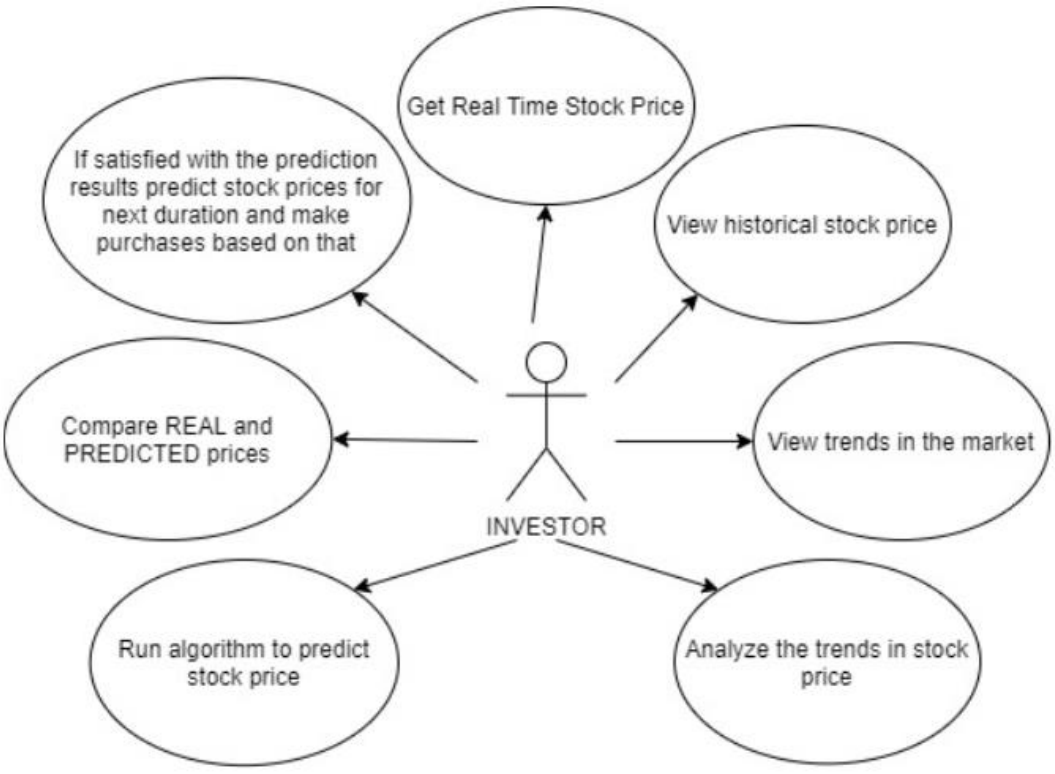

Fig. 4.2 Use case diagram

CHAPTER 5

IMPLEMENTATION

This chapter covers the stages of implementation for the proposed system to predict the stock price and movement of stocks.

5.1 Dataset

This dataset comprises of following five variables: open, close, low, high and volume. Open, close, low and high are different bid prices for the stock at separate times with nearly direct names. The volume is the number of shares that passed from one owner to another during the time period. The dataset can be acquired by different ways.

5.1.1 Getting data from Alphavantage

Since American Airlines Stock market prices are used to make the predictions, the ticker to “AAL” is set. Additionally, a url string is defined which will return a JSON file with all the stock market data for American Airlines within the last 20 years, and a file to save, which will be the file to which the data is saved.

Next, a condition is specified: if the data isn’t already saved, it is fetched from the URL stored in the url string. The date, low, high, volume, close, open values is stored to a pandas Data Frame df and is saved in a file. However, if the data is already there, it is just loaded from the CSV file.

#code

```
api_key = '<your API key>'

# American Airlines stock market prices

ticker = "AAL"

# JSON file with all the stock market data for AAL from the last 20 years

url_string =
"https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=%s
&outputsize=full&apikey=%s"%(ticker,api_key)
```

```
# Save data to this file
file_to_save = 'stock_market_data-%s.csv'%ticker


# If you haven't already saved data,
# Go ahead and grab the data from the url
# And store date, low, high, volume, close, open values to a Pandas DataFrame
if not os.path.exists(file_to_save):
    with urllib.request.urlopen(url_string) as url:
        data = json.loads(url.read().decode())
        # extract stock market data
        data = data['Time Series (Daily)']
        df = pd.DataFrame(columns=['Date','Low','High','Close','Open'])
        for k,v in data.items():
            date = dt.datetime.strptime(k, '%Y-%m-%d')
            data_row = [date.date(),float(v['3. low']),float(v['2. high']),
                    float(v['4. close']),float(v['1. open'])]
            df.loc[-1,:] = data_row
            df.index = df.index + 1
    print('Data saved to : %s'%file_to_save)
    df.to_csv(file_to_save)


# If the data is already there, just load it from the CSV
else:
```

```
print('File already exists. Loading data from CSV')

df = pd.read_csv(file_to_save)
```

5.1.2 Getting data from Kaggle

Data found on Kaggle is a collection of csv files and you don't have to do any pre-processing, so you can directly load the data into a Pandas DataFrame.

#code

```
df =
pd.read_csv(os.path.join('Stocks','hpq.us.txt'),delimiter=',',usecols=['Date','Open','High','Low','Close'])

  print('Loaded data from the Kaggle repository')
```

5.2 Data Processing

```
# First the mid prices from the highest and lowest is calculated

high_prices = df.loc[:,'High'].as_matrix()

low_prices = df.loc[:,'Low'].as_matrix()

mid_prices = (high_prices+low_prices)/2.0

train_data = mid_prices[:11000]

test_data = mid_prices[11000:]

# The data to be scaled between 0 and 1

# Both test and train data is normalized with respect to training data

scaler = MinMaxScaler()

train_data = train_data.reshape(-1,1)

test_data = test_data.reshape(-1,1)
```

```
# the Scaler is trained with training data and smooth data

smoothing_window_size = 2500

for di in range(0,10000,smoothing_window_size):

    scaler.fit(train_data[di:di+smoothing_window_size,:])

    train_data[di:di+smoothing_window_size,:] =
scaler.transform(train_data[di:di+smoothing_window_size,:])



# the last bit of remaining data is normalized

scaler.fit(train_data[di+smoothing_window_size:,:])

train_data[di+smoothing_window_size:,:] =
scaler.transform(train_data[di+smoothing_window_size:,:])

# Reshape both train and test data

train_data = train_data.reshape(-1)

# Normalize test data

test_data = scaler.transform(test_data).reshape(-1)

# Now perform exponential moving average smoothing

# So the data will have a smoother curve than the original ragged data

EMA = 0.0

gamma = 0.1

for ti in range(11000):

  EMA = gamma*train_data[ti] + (1-gamma)*EMA

  train_data[ti] = EMA

# Used for visualization and test purposes

all_mid_data = np.concatenate([train_data,test_data],axis=0)
```

5.3 Using Exponential Moving Average

In the exponential moving average method, you calculate x_{t+1} as,

$$X_{t+1} = EMA_t = \gamma \times EMA_{t-1} + (1-\gamma)X_t$$

where $EMA_0 = 0$ and EMA is the exponential moving average value, maintained over time.

The above equation basically calculates the exponential moving average from t+1time step and uses that as the one step ahead prediction. γ decides what the contribution of the most recent prediction is to the EMA. For example, a γ=0.1γ=0.1 gets only 10% of the current value into the EMA.

#code

```
window_size = 100

N = train_data.size

run_avg_predictions = []

run_avg_x = []

mse_errors = []

running_mean = 0.0

run_avg_predictions.append(running_mean)

decay = 0.5

for pred_idx in range(1,N):

    running_mean = running_mean*decay + (1.0-decay)*train_data[pred_idx-1]

    run_avg_predictions.append(running_mean)

    mse_errors.append((run_avg_predictions[-1]-train_data[pred_idx])**2)

    run_avg_x.append(date)

print('MSE error for EMA averaging: %.5f'%(0.5*np.mean(mse_errors)))
```

5.4 Introducing LSTMs

Long Short-Term Memory models are extremely powerful time-series models. They can predict an arbitrary number of steps into the future. An LSTM module (or cell) has 5 essential components which allows it to model both long-term and short-term data.

- Cell state (c_t) - This represents the internal memory of the cell which stores both short term memory and long-term memories
- Hidden state (h_t) - 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. Additionally, the hidden state can decide to only retrive the short or long-term or both types of memory stored in the cell state to make the next prediction.
- Input gate (i_t) - Decides how much information from current input flows to the cell state
- Forget gate (f_t) - Decides how much information from the current input and the previous cell state flows into the current cell state
- Output gate (o_t) - Decides how much information from the current cell state flows into the hidden state, so that if needed LSTM can only pick the long-term memories or short-term memories and long-term memories

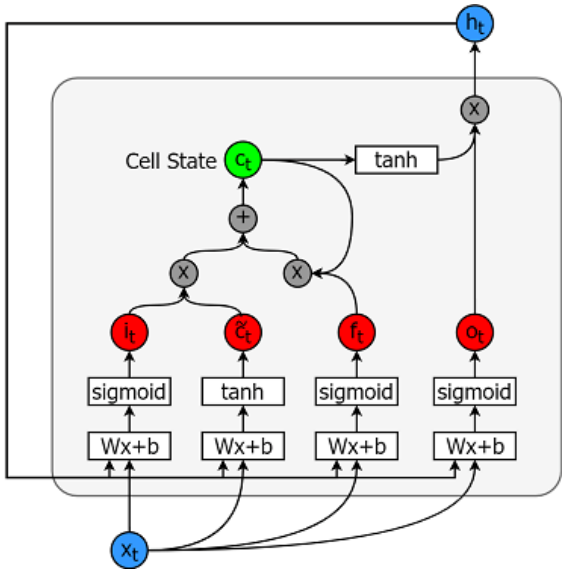

Fig 5.1 Cell state in LSTM

First a data generator is implemented to train the model. This data generator will have a method called. unroll_batches(...) which will output a set of num_unrollings batches of input data obtained sequentially, where a batch of data is of size [batch_size, 1]. Then each batch of input data will have a corresponding output batch of data. Fig 5.2 illustrates how a batch of data is created visually.

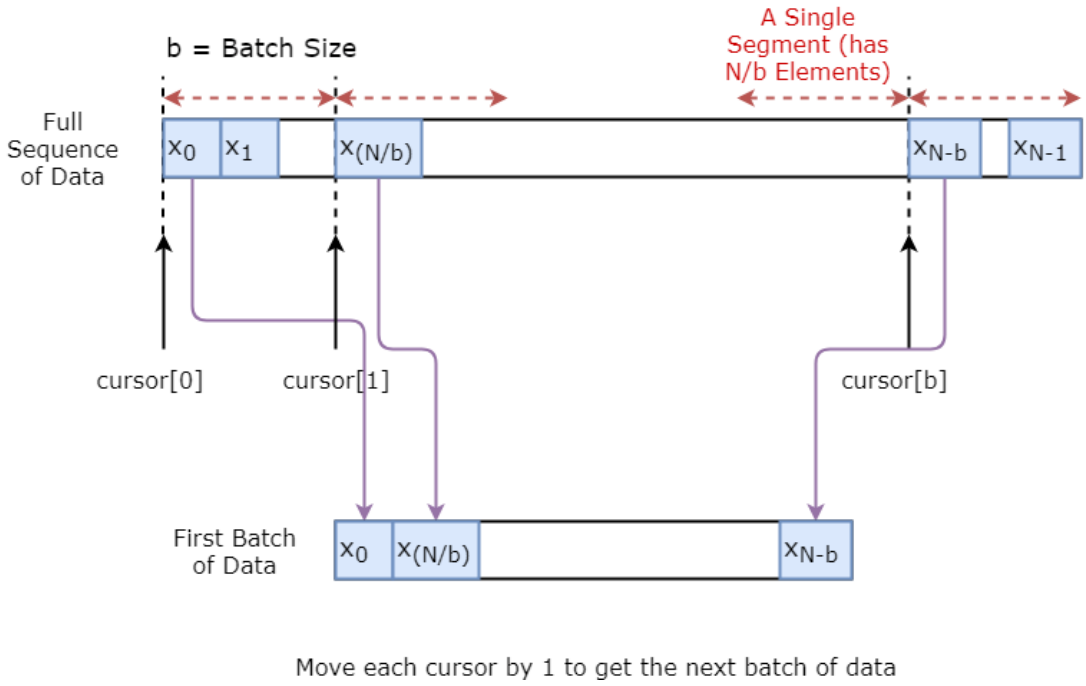

Fig. 5.2 Data generation

#code

```
class DataGeneratorSeq(object):

    def __init__(self,prices,batch_size,num_unroll):

        self._prices = prices

        self._prices_length = len(self._prices) - num_unroll

        self._batch_size = batch_size

        self._num_unroll = num_unroll

        self._segments = self._prices_length //self._batch_size

        self._cursor = [offset * self._segments for offset in range(self._batch_size)]
```

```
def next_batch(self):
        batch_data = np.zeros((self._batch_size),dtype=np.float32)
        batch_labels = np.zeros((self._batch_size),dtype=np.float32)
        for b in range(self._batch_size):
            if self._cursor[b]+1>=self._prices_length:
                self._cursor[b] = b * self._segments
                self._cursor[b] = np.random.randint(0,(b+1)*self._segments)
        batch_data[b] = self._prices[self._cursor[b]]
        batch_labels[b]= self._prices[self._cursor[b]+np.random.randint(0,5)]
        self._cursor[b] = (self._cursor[b]+1)%self._prices_length
       return batch_data,batch_labels
def unroll_batches(self):
        unroll_data,unroll_labels = [],[]
        init_data, init_label = None,None
        for ui in range(self._num_unroll):
            data, labels = self.next_batch()
            unroll_data.append(data)
            unroll_labels.append(labels)
       return unroll_data, unroll_labels
def reset_indices(self):
        for b in range(self._batch_size):
            self._cursor[b] =
np.random.randint(0,min((b+1)*self._segments,self._prices_length-1))
```

```
dg = DataGeneratorSeq(train_data,5,5)

u_data, u_labels = dg.unroll_batches()

for ui,(dat,lbl) in enumerate(zip(u_data,u_labels)):

    print('\n\nUnrolled index %d'%ui)

    dat_ind = dat

    lbl_ind = lbl

    print('\tInputs: ',dat )

    print('\n\tOutput:',lbl)

#Input data

train_inputs, train_outputs = [],[]

# unroll the input over time defining placeholders for each time step

for ui in range(num_unrollings):

    train_inputs.append(tf.placeholder(tf.float32,
shape=[batch_size,D],name='train_inputs_%d'%ui))

    train_outputs.append(tf.placeholder(tf.float32, shape=[batch_size,1], name =
'train_outputs_%d'%ui))

lstm_cells = [

    tf.contrib.rnn.LSTMCell(num_units=num_nodes[li],

                            state_is_tuple=True,

                            initializer= tf.contrib.layers.xavier_initializer()

                           )

 for li in range(n_layers)]

drop_lstm_cells = [tf.contrib.rnn.DropoutWrapper(
```

```
    lstm, input_keep_prob=1.0,output_keep_prob=1.0-dropout, state_keep_prob=1.0-
dropout

) for lstm in lstm_cells]

drop_multi_cell = tf.contrib.rnn.MultiRNNCell(drop_lstm_cells)

multi_cell = tf.contrib.rnn.MultiRNNCell(lstm_cells)

w = tf.get_variable('w',shape=[num_nodes[-1], 1],
initializer=tf.contrib.layers.xavier_initializer())

b = tf.get_variable('b',initializer=tf.random_uniform([1],-0.1,0.1))

#Calculating LSTM output and Feeding it to the regression layer to get final
prediction

# Create cell state and hidden state variables to maintain the state of the LSTM

c, h = [],[]

initial_state = []

for li in range(n_layers):

  c.append(tf.Variable(tf.zeros([batch_size, num_nodes[li]]), trainable=False))

  h.append(tf.Variable(tf.zeros([batch_size, num_nodes[li]]), trainable=False))

  initial_state.append(tf.contrib.rnn.LSTMStateTuple(c[li], h[li]))


# Do several tensor transformations, because the function dynamic_rnn requires the
output to be of a specific format.

all_inputs = tf.concat([tf.expand_dims(t,0) for t in train_inputs],axis=0)

# all_outputs is [seq_length, batch_size, num_nodes]

all_lstm_outputs, state = tf.nn.dynamic_rnn(

    drop_multi_cell, all_inputs, initial_state=tuple(initial_state),
```

```
    time_major = True, dtype=tf.float32)

all_lstm_outputs = tf.reshape(all_lstm_outputs,
[batch_size*num_unrollings,num_nodes[-1]])

all_outputs = tf.nn.xw_plus_b(all_lstm_outputs,w,b)

split_outputs = tf.split(all_outputs,num_unrollings,axis=0)

# The loss of all the unrolled steps is calculated at the same time. Therefore, the mean
error or each batch is taken and the sum of that over all the unrolled steps is used.

print('Defining training Loss')

loss = 0.0

with tf.control_dependencies([tf.assign(c[li], state[li][0]) for li in range(n_layers)]+

                             [tf.assign(h[li], state[li][1]) for li in range(n_layers)]):

  for ui in range(num_unrollings):

    loss += tf.reduce_mean(0.5*(split_outputs[ui]-train_outputs[ui])**2)

print('Learning rate decay operations')

global_step = tf.Variable(0, trainable=False)

inc_gstep = tf.assign(global_step,global_step + 1)

tf_learning_rate = tf.placeholder(shape=None,dtype=tf.float32)

tf_min_learning_rate = tf.placeholder(shape=None,dtype=tf.float32)

learning_rate = tf.maximum(

    tf.train.exponential_decay(tf_learning_rate, global_step, decay_steps=1,
decay_rate=0.5, staircase=True),

    tf_min_learning_rate)

# Optimizer.

print('TF Optimization operations')
```

```
optimizer = tf.train.AdamOptimizer(learning_rate)

gradients, v = zip(*optimizer.compute_gradients(loss))

gradients, _ = tf.clip_by_global_norm(gradients, 5.0)

optimizer = optimizer.apply_gradients(

   zip(gradients, v))

print('\tAll done')

print('Defining prediction related TF functions')

sample_inputs = tf.placeholder(tf.float32, shape=[1,D])

# Maintaining LSTM state for prediction stage

sample_c, sample_h, initial_sample_state = [],[],[]

for li in range(n_layers):

  sample_c.append(tf.Variable(tf.zeros([1, num_nodes[li]]), trainable=False))

  sample_h.append(tf.Variable(tf.zeros([1, num_nodes[li]]), trainable=False))


initial_sample_state.append(tf.contrib.rnn.LSTMStateTuple(sample_c[li],sample_h[li
]))


reset_sample_states = tf.group(*[tf.assign(sample_c[li],tf.zeros([1, num_nodes[li]]))
for li in range(n_layers)],

                    *[tf.assign(sample_h[li],tf.zeros([1, num_nodes[li]])) for li in
range(n_layers)])

sample_outputs, sample_state = tf.nn.dynamic_rnn(multi_cell,
tf.expand_dims(sample_inputs,0),

                    initial_state=tuple(initial_sample_state),

                    time_major = True,
```

```
                    dtype=tf.float32)

with tf.control_dependencies([tf.assign(sample_c[li],sample_state[li][0]) for li in
range(n_layers)]+

                    [tf.assign(sample_h[li],sample_state[li][1]) for li in
range(n_layers)]):

  sample_prediction = tf.nn.xw_plus_b(tf.reshape(sample_outputs,[1,-1]), w, b)

print('\tAll done')
```

Here the model is trained and stock price movements for several epochs is predicted and is seen whether the predictions get better or worse over time. The following procedure is followed.

- Define a test set of starting points (test_points_seq) on the time series to evaluate the model on
- For each epoch

  For full sequence length of training data

  - Unroll a set of num_unrollings batches
  - Train the neural network with the unrolled batches
  - Calculate the average training loss

  For each starting point in the test set

  - Update the LSTM state by iterating through the previous num_unrollings data points found before the test point
  - Make predictions for n_predict_once steps continuously, using the previous prediction as the current input
  - Calculate the MSE loss between the n_predict_once points predicted and the true stock prices at those time stamps

```
#code

epochs = 30

valid_summary = 1 # Interval you make test predictions

n_predict_once = 50 # Number of steps you continously predict for

train_seq_length = train_data.size # Full length of the training data

train_mse_ot = [] # Accumulate Train losses

test_mse_ot = [] # Accumulate Test loss

predictions_over_time = [] # Accumulate predictions

session = tf.InteractiveSession()

tf.global_variables_initializer().run()

# Used for decaying learning rate

loss_nondecrease_count = 0

loss_nondecrease_threshold = 2 # If the test error hasn't increased in this many steps, decrease learning rate

print('Initialized')

average_loss = 0

# Define data generator

data_gen = DataGeneratorSeq(train_data,batch_size,num_unrollings)


x_axis_seq = []

# test predictions

test_points_seq = np.arange(11000,12000,50).tolist()

for ep in range(epochs):
```

```
# ======================== Training=========================
    for step in range(train_seq_length//batch_size):
            u_data, u_labels = data_gen.unroll_batches()
            feed_dict = {}
    for ui,(dat,lbl) in enumerate(zip(u_data,u_labels)):
            feed_dict[train_inputs[ui]] = dat.reshape(-1,1)
            feed_dict[train_outputs[ui]] = lbl.reshape(-1,1)
            feed_dict.update({tf_learning_rate: 0.0001, tf_min_learning_rate:0.000001})
              _, l = session.run([optimizer, loss], feed_dict=feed_dict)
            average_loss += l
# ============================ Validation ======================
    if (ep+1) % valid_summary == 0:
            average_loss = average_loss/(valid_summary*(train_seq_length//batch_size))
            # The average loss
      if (ep+1)%valid_summary==0:
        print('Average loss at step %d: %f' % (ep+1, average_loss))
train_mse_ot.append(average_loss)
 average_loss = 0 # reset loss


 predictions_seq = []
mse_test_loss_seq = []
 # =============== Updating State and Making Predicitons ============
      for w_i in test_points_seq:
```

```
            mse_test_loss = 0.0

            our_predictions = []

            if (ep+1)-valid_summary==0:

            # Only calculate x_axis values in the first validation epoch

            x_axis=[]

            # Feed in the recent past behavior of stock prices

            # to make predictions from that point onwards

            for tr_i in range(w_i-num_unrollings+1,w_i-1):

            current_price = all_mid_data[tr_i]

            feed_dict[sample_inputs] = np.array(current_price).reshape(1,1)

                    _ = session.run(sample_prediction,feed_dict=feed_dict)


        feed_dict = {}

current_price = all_mid_data[w_i-1]

feed_dict[sample_inputs] = np.array(current_price).reshape(1,1)

# Make predictions for this many steps

        # Each prediction uses previous prediciton as it's current input

        for pred_i in range(n_predict_once):


          pred = session.run(sample_prediction,feed_dict=feed_dict)

our_predictions.append(np.asscalar(pred))

feed_dict[sample_inputs] = np.asarray(pred).reshape(-1,1)

if (ep+1)-valid_summary==0:
```

```
            # Only calculate x_axis values in the first validation epoch
            x_axis.append(w_i+pred_i)
mse_test_loss += 0.5*(pred-all_mid_data[w_i+pred_i])**2
session.run(reset_sample_states)
predictions_seq.append(np.array(our_predictions))
mse_test_loss /= n_predict_once
        mse_test_loss_seq.append(mse_test_loss)
  if (ep+1)-valid_summary==0:
       x_axis_seq.append(x_axis)
current_test_mse = np.mean(mse_test_loss_seq)
# Learning rate decay logic
if len(test_mse_ot)>0 and current_test_mse > min(test_mse_ot):
          loss_nondecrease_count += 1
  else:
          loss_nondecrease_count = 0
if loss_nondecrease_count > loss_nondecrease_threshold :
            session.run(inc_gstep)
            loss_nondecrease_count = 0
            print('\tDecreasing learning rate by 0.5')
test_mse_ot.append(current_test_mse)
      print('\tTest MSE: %.5f'%np.mean(mse_test_loss_seq))
      predictions_over_time.append(predictions_seq)
      print('\tFinished Predictions')
```

CHAPTER 6

RESULTS AND DISCUSSION

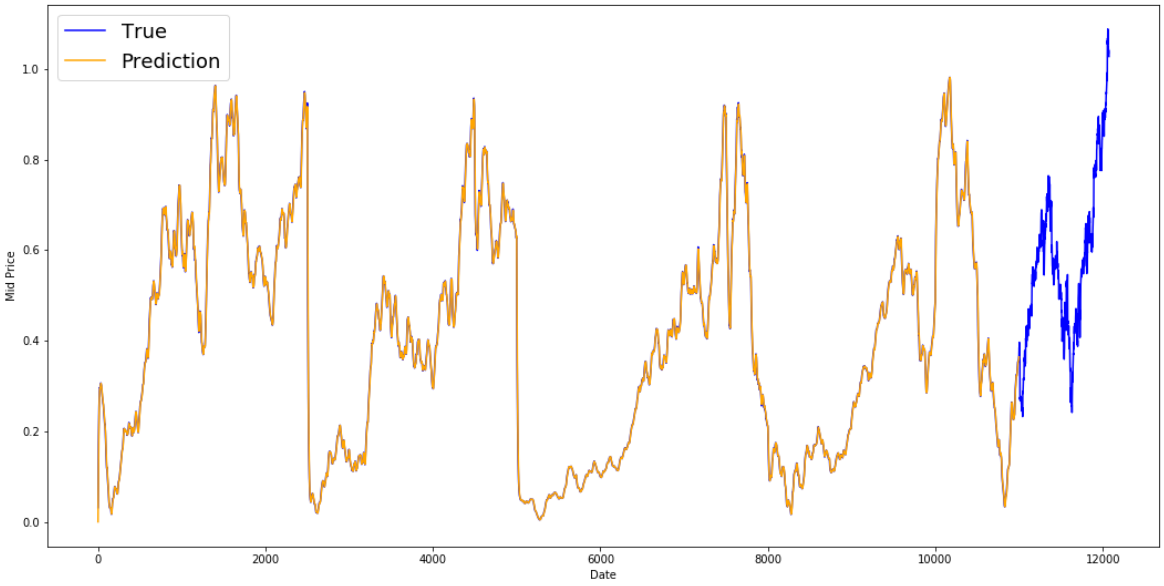

Fig 6.1 EMA implementation

Fig. 6.1 is obtained using an implementation of EMA. It is clear that the predictions are accurate with almost negligible error. EMA provides the best in class prediction for single step and therefore cannot be used for long term trading strategies.

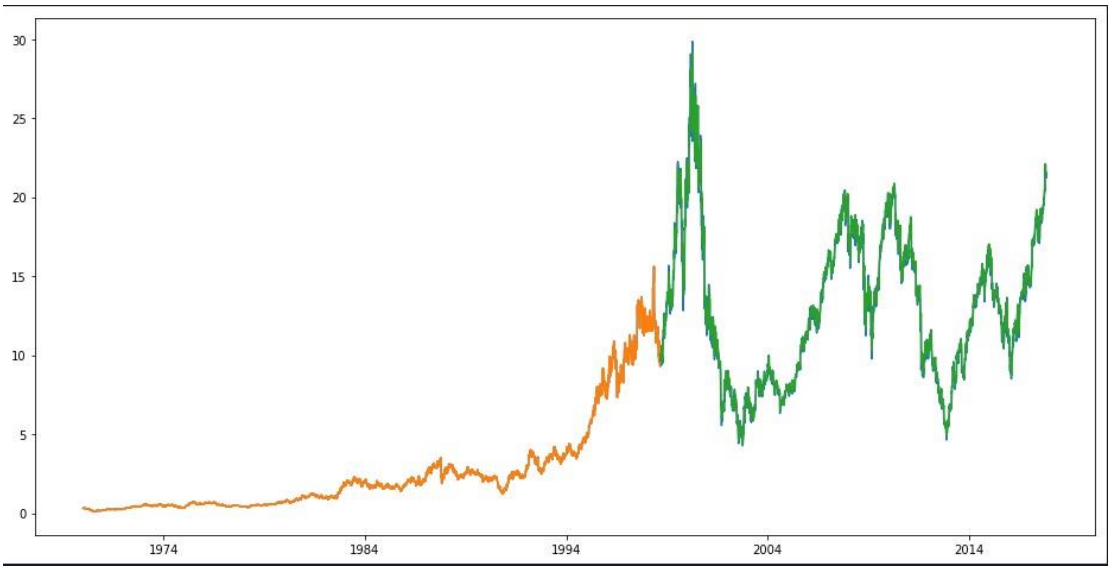

Fig 6.2 LSTM result 1

Fig. 6.2 depicts the scenario where we have:

Trained with : 7244 days

RMSE: 8%

Predicted for : 4830 days

We can see that the error is much greater when we train for a less duration and predict for a larger duration.

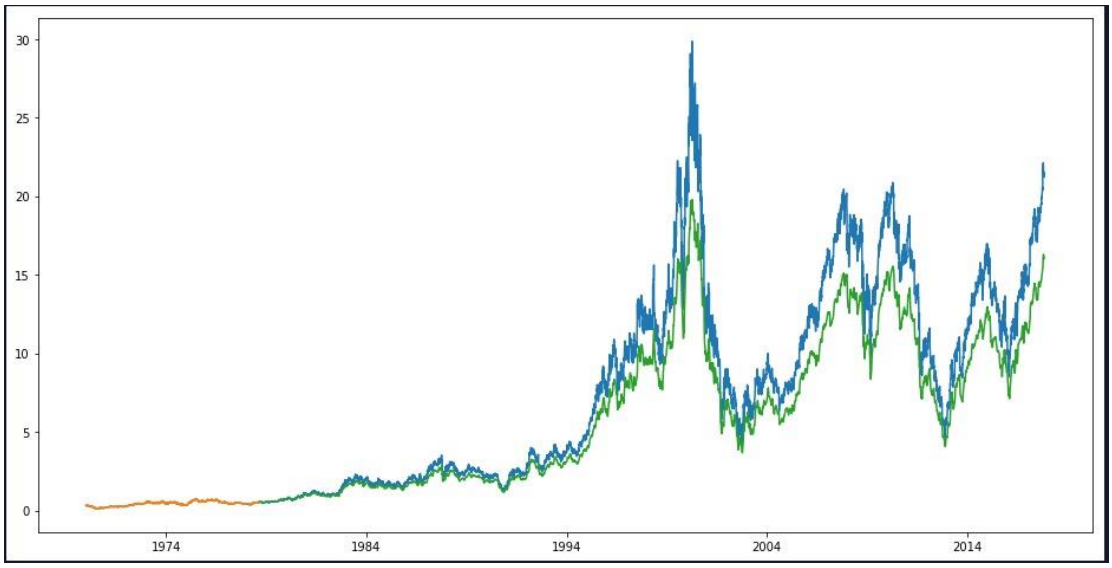

Fig 6.3 LSTM result 2

In Fig. 6.3 we can see that we have

Total: 12074 days

Trained with : 2190 days

RMSE: 29%

Predicted for : 9884 days

We have trained for a significant amount of days and predicted the prices, here we can see that the predictions are satisfactory.

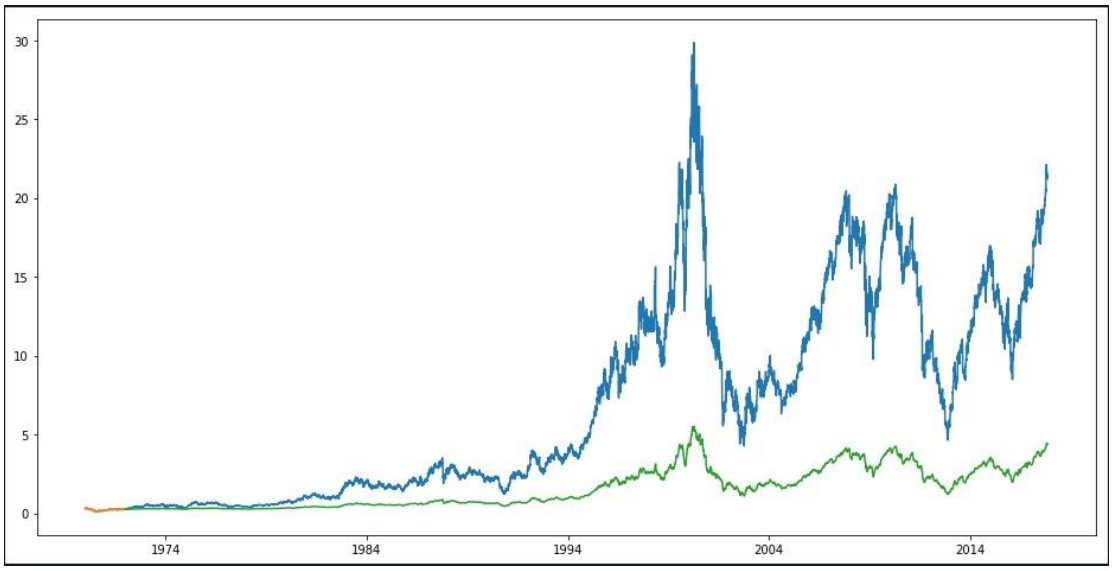

Fig 6.4 LSTM result 3

In Fig 6.4 this scenario we have

Total: 12074 days

Trained with : 500 days

RMSE: 78%

Predicted for : 11574 days

We can observe here that the predictions made are achieveing immense accuracy causing overfitting which is not favourable.

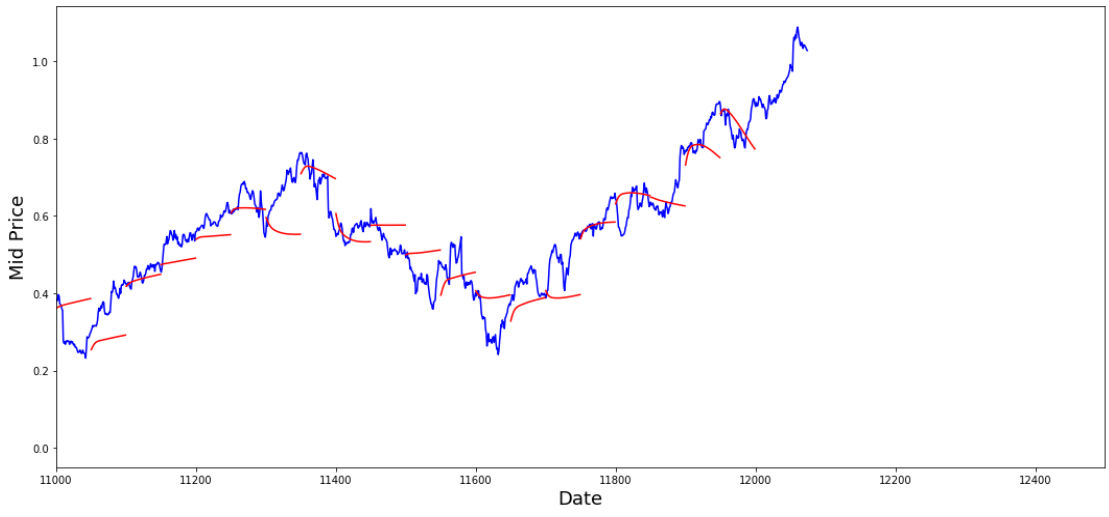

Fig 6.5 Best predictions over time

In Fig. 6.5, the red lines show the predicted stock movements, the length of the line is the duration for which the movement prediction is made and the shape represents the expected movement.

Fig 6.6 Evolution of predictions over time

In Fig 6.6, for cluster of lines, each line represents a movement prediction made by the system the densest areas represent the most probable prediction, out of which the most accurate one is selected.

CHAPTER 7

TESTING

This chapter gives an overview of the various types of testing incorporated during the entire duration of the project.

7.1 Unit Testing

Testing of an individual software component or module is termed as Unit Testing. It is typically done by the programmer and not by testers, as it requires detailed knowledge of the internal program design and code. It may also require developing test driver modules or test harnesses.

7.2 Component Testing

Component Testing is mostly performed by developers after the completion of unit testing. Component Testing involves testing of multiple functionalities as a single code and its objective is to identify if any defect exists after connecting those multiple functionalities with each other.

7.3 Integration Testing

Testing of all integrated modules to verify the combined functionality after integration is termed as Integration Testing. Modules are typically code modules, individual applications, client and server applications on a network, etc. This type of testing is especially relevant to client/server and distributed systems.

7.4 System Testing

Under System Testing technique, the entire system is tested as per the requirements. It is a Black-box type Testing that is based on overall requirement specifications and covers all the combined parts of a system.

7.5 Interface Testing

The objective of this Interface Testing is to validate the interface as per the business requirement. The expected interface of the application is mentioned in the detailed design document and interface mock-up screens. Checks if the application correctly connects to the server.

7.6 Compatibility Testing

Compatibility Testing checks whether the application is compatible with the specified software and hardware requirements and functions efficiently as expected.

7.7 Performance Testing

Performance testing is used to check for appropriate and efficient performance is shown by the system as per the requirements. The connection requirements are to be maintained to ensure efficient performance evaluation.

7.8 Usability Testing

Under Usability Testing, User-friendliness check is done. The application flow is tested to know if a new user can understand the application easily or not, proper help is documented if a user gets stuck at any point. Basically, system navigation is checked in this testing.

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

This chapter gives an overview of the conclusion, our contribution to the project, the future scope of the proposed system and our learnings incurred during the entire duration of the project

8.1 Conclusion

- Dilemma b/w overfitting and actual prediction.

  When predictions are made in fields like the stock market where the data is dynamic and will never be the same it is difficult to believe or depend on the predictions alone as there is so much uncertainty.

- Ability to predict the general trend of a given stock.

  With the help of stock market predictions, we can now understand the working of the market better and use it to our advantage to make smart and profitable choices.

- EMA predicts next step with negligible error.

  EMA provides the best in class prediction for single step and therefore cannot be used for long term trading strategies.

- Cannot predict uncertainties.

  There are many factors that affect the stock market as a whole, and there is a significant amount of uncertain and unpredictable factors as well. Unfortunately, we are unable to make predictions on the various uncertain factors that cause fluctuations in the stock market.

- Not investment viable.

  Due to many factors which are uncertain and those which are certain as well, it is not viable to invest in stock markets just based on predictions made based on historical and other data.

- Good learning experience.

  In order to be able to invest in the stock market or to be able to write code for stock market prediction, the amount of knowledge required is immense and vast, hence during that process so much learning happens.

8.2 Contribution

Stock market prediction is the act of trying to determine the future value of a company stock or other financial instrument traded on an exchange. The successful prediction of a stock's future price could yield significant profit.

With the ability to predict the stock prices or the trends alone, one is now able to make a very huge profit by investing in the stock market.

8.3 Future Scope

- Factoring relationships b/w stocks and different markets:

  Along with the basic factors, in the future models we can establish relationships between different markets and stocks to determine how they influence each other leading to more accurate predictions.

- Accounting for several organisational factors:

  Many factors like:

  - Market sentiment
  - Industry performance
  - Government policies
  - Company news
  - Market Capitalization
  - May now be taken into account while making predictions to improve accuracy

- Taking global economic trends into account:

  Trends like Global recession, climate change, rise of moral capitalism and others will influence stocks and hence can be considered while making predictions.

REFERENCES

[1] "Apache Spark - Lightning-Fast Unified Analytics Engine," 2020. Available: https://spark.apache.org/

[2] V. Atanasov, C. Pirinsky, and Q. Wang, "The efficient market hypothesis and investor behavior," 2018.

[3] S. Agrawal, D. Thakkar, D. Soni, K. Bhimani, and C. Patel, "Stock market prediction using machine learning techniques," International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 2019.

[4] M. Afeef, A. Ihsan, and H. Zada, "Forecasting stock prices through univariate arima modeling," 2018.

[5] P.-F. Pai and C.-S. Lin, "A hybrid arima and support vector machines model in stock price forecasting," Omega, vol. 33, no. 6, pp. 497–505, 2018.

[6] D. Karmiani, R. Kazi, A. Nambisan, A. Shah, and V. Kamble, "Comparison of predictive algorithms: Backpropagation, svm and lstm for stock market," in 2019 Amity International Conference on Artificial Intelligence (AICAI). IEEE, 2019, pp. 228–234.

[7] E. Ahmadi, M. Jasemi, L. Monplaisir, M. A. Nabavi, A. Mahmoodi, and P. A. Jam, "New efficient hybrid candlestick technical analysis model for stock market timing on the basis of the support vector machine and heuristic algorithms of imperialist competition and genetic," Expert Systems with Applications, vol. 94, pp. 21–31, 2018.

[8] S. Sharma and B. Kaushik, "Quantitative analysis of stock market prediction for accurate investment decisions in future," Journal of Artificial Intelligence, vol. 11, pp. 48–54, 2018.

[9] https://zerodha.com/varsity/

[10] https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21