A Framework on Automated Trade Systems using TimeSeries Forecasting Algorithms and Machine Learning Classifiers

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering

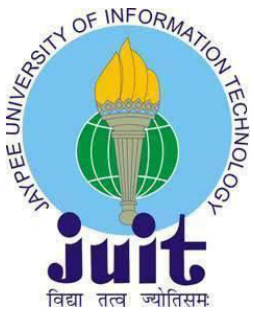
By

Arun Sharma (151310)

Under the supervision of

Dr. Rajinder Sandhu

to



Department of Computer Science & Engineering and Information Technology

Jaypee University Of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

CERTIFICATE

Candidate's Declaration

I hereby declare that the work presented in this report entitled **Algorithmic Trading using classifiers and time series forecasting algorithms** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** submitted in the department of Computer Science &Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2018 to May 2019 under the supervision of Dr. Rajinder Sandhu Assistant Professor (Senior Grade).The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Arun Sharma, 151310

This is to certify that above statement made by the candidate is true to the best of my knowledge.

Dr. Rajinder Sandhu

Assistant Professor (Senior Grade)

Computer Sciences Department

Dated:

Acknowledgement

I would like to express my deepest appreciation to all those who provided me the possibility to complete this report. A special gratitude I give to our final year project supervisor, Dr. Rajinder Sandhu, whose contribution in stimulating suggestions and encouragement, helped me and my partner to coordinate our project especially in writing this report. Furthermore I would also like to acknowledge with much appreciation the crucial role of Jaypee University Of Information Technology, who gave the permission to use all required equipment and the necessary materials to complete the project "Algorithmic Trading". A special thanks goes to my supervisor, Dr. Rajinder Sandhu, who help me to assemble the parts and gave suggestion about the project "Algorithmic Trading" he have invested his full effort in guiding us for achieving the goal. I have to appreciate the guidance given by other supervisor as well as the panels especially in our project presentation that has improved our presentation skills thanks to their comment and advices.

Abstract

Main idea here is to implement a model which will predict the stock price(trend) for the future. Here data is gathered from the yahoo finance and data of stock is saved locally in csv format. Data then fed to the classifiers to predict the condition for **Algorithmic Trading using classifiers and time series forecasting algorithms**. Data is also fed to few time series forecasting algorithm to predict the trend and seasonality for the forecasting which will help the traders to invest wisely.

Combined result of classifiers and time series algorithms is taken into consideration for the conditions buy/sell/hold of algorithmic trading. Data is first preprocessed so that only relevant data is present and rest of data can be dropped. Here we are only interested in finding the Close/AdjClose price of the MMM.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Statement:

The Stock market prediction is a very interesting task which involves high knowledge of how the stock market works, and what deviations can be caused in a market due to various conditions. While some researchers might argue that the market itself is efficient, that if there is new information or any deviation in a market it absorbs it by correcting itself, thus making no room for predictions, while some researchers may argue that if we train the data well and can come up with some sort of strategy that is effective. Idea here is to fed the data of stock for a company to the different classifiers and time series forecasting algorithms to make prediction for the condition of buy/sell/hold the stock price over a period of time

1.2 Objective

Develop a model for algorithmic trading to help in prediction of buy/sell/hold condition
To develop the time series forecasting model to predict the future trend of MMM`s stock price

1.3 Methodlolgy

- Tickers of SNP 500 companies scrapped and are stored into a file locally.

- Using yahoo finance and tickers obtained data of stock of first 20 companies is stored locally in csv format.
- Data stored locally preprocessed and splitted into training and test data
- Trained and test data of MMM is fed to the classifiers and time series forecasting algorithms

- Conditions of buy/sell/hold are obtained using the trend result obtained in above step.

- Algorithms are then compared based upon their result.

1.4 Tools and Modules Used

- *Python Idle 3.6.5* - It is a python editor in which we will actually implement the algorithm to extract data sets, perform manipulations, and predict the expected results with accuracy.

- *NumPy* - Numpy is a python library which helps to store complex, multidimensional arrays and provide a vast variety of methodical function which can be operated on these array.

- *Matplotlib* - Matplotlib is a python library for the Python .it is used for plotting the graphs and it also provide Object oriented API which can be used to embed plots into its application.

- *Pandas* - It also a Python library used for manipulating the data and provides data structures and operations For manipulating the data which can be in the form of numerical table or time series data.

- *Pandas-DataReader* - Used to extract data from a large number number of internet sources.

- *BeautifulSoup4* - BeautifulSoup4 is a parsing python package used for parsing HTML and XML documents. A parse tree is created and it is used to extract data from different web pages.

- *Scikit-Learn/SkLearn* - A ML python library consists of classification, regression, and many clustering algorithm like support vector machine , knn, random forest.

- *Satsmodel* – A python module used to import simplexponential, holt and many others.

Chapter 2

Literature Survey

2.1 Algorithmic Trading Using Classifiers

Decision making in stock market is a difficult task and it is challenging too, This project focus on the methods and techniques which can be implemented To predict the stock price ,building such model is not an easy task and it has to be accurate Investors wants to invest in those stocks only which have a tendency to go up in its stock price. Here the goal is to develop such model which will predict the stock price in order to decide Whether to : 1.buy 2.hold 3.sell

The goal here is to provide an output (buy/hold) as per customer requirements like amount to invest, time duration ,max profit, max loss. This was done by making use iof various data mining and ML techniques

[1] Predicting the direction of stock prices is a widely studied subject in many fields including-trading, finance, statistics and computer science. Investors in the stock market can maximize their profit by buying or selling their investment if they can determine when to enter and exit position. Professional traders typically use fundamental and/or technical analysis to analyze stocks in making investment decisions. Fundamental analysis involves a study of company fundamentals such as revenues and profits, market position, growth rates, etc. Technical analysis, on the other hand, is based on the study of historical price fluctuations. Due to the nature of market forces, economies tend to follow a pattern of expansion and contraction, over long periods of time. The stocks trade within an overarching environment where economy moves from one phase of the business cycle to the next. Compared to the existing work, this project analyses the stocks trading decisions utilizing the technical behavior of the trading patterns within the context of the fluctuating economic and business environment. The objective function is to maximize medium to longer term profits based on S&P500 stock market index. The inputs are the technical indicators data and the economic indicators data. Three models (neural network, soft max logistic regression, decision forest) are then used to predict the buy/sell decisions.

2.1.1 Proposed Systems

As discussed above stock market prediction is a vast topic and has a lot areas on which we can research upon, but one thing all models have in common is their check on

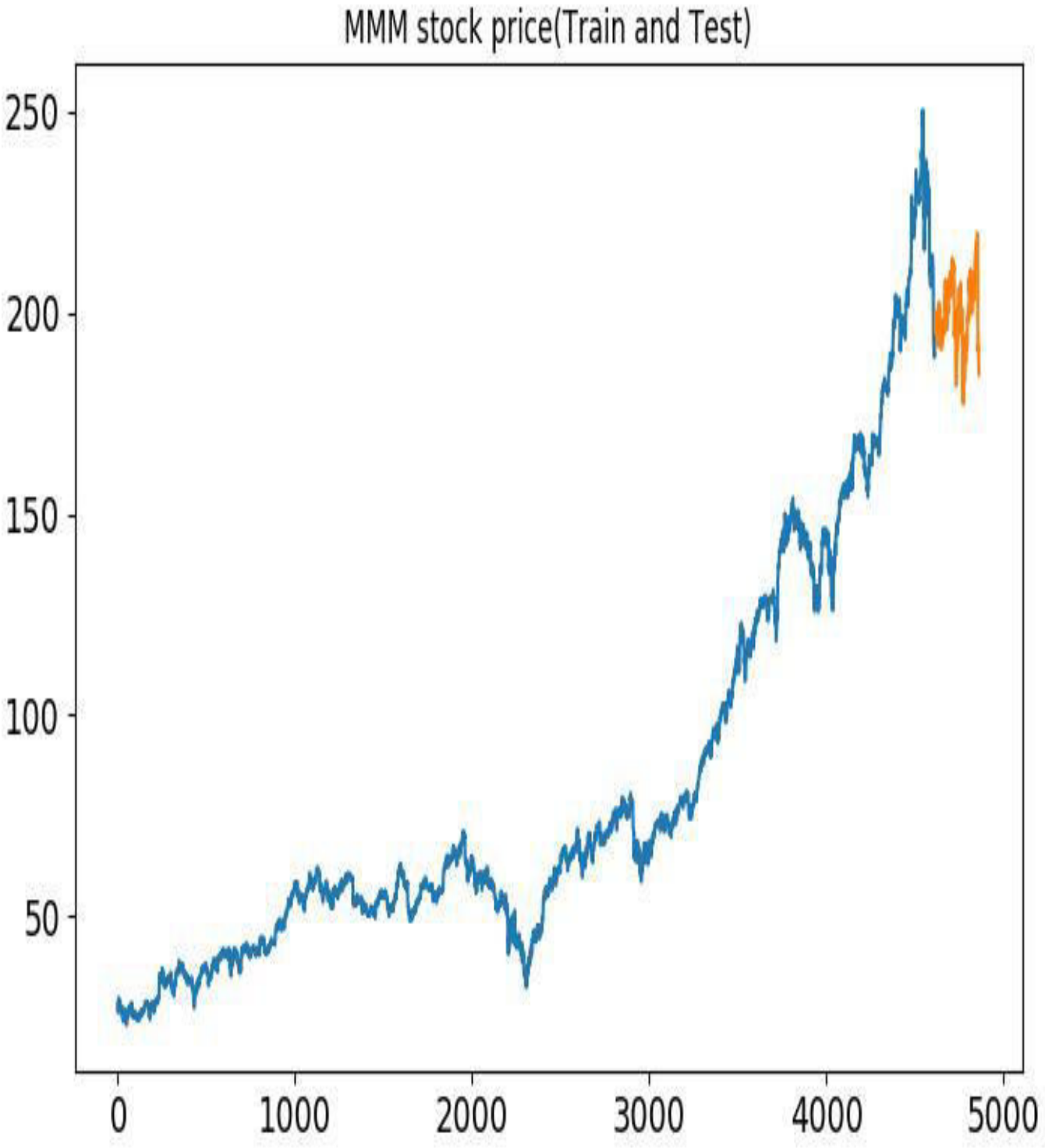
accuracy of how well the models applied can fit to a given dataset and is it matching the results and predicting correctly or not. Still each model has a few things in common, they all need a list of companies of any stock exchange to predict upon the three basic conditions of market buy, hold, and sell and to do this we need the stock market data for each company against their tickers stored in our machine to avoid larger accessing time and performing data manipulations in order to prepare the dataset for further machine learning classifiers which will eventually predict the results and provide the output.

2.1.2 Feasibility Study

To check the feasibility of the above model we will check the given output and match it against the graph of the actual company and observe the patterns. Another way (explained below)to check the feasibility is checking the predicted trend by holt winters method and ARIMAX time series forecasting algorithms with the actual trend observed in the market.

2.2 Using Time-Series Forecasting algorithms

[2] What is a time series data? It is very important to understand time series data in order to understand the methodology followed here. A data which follows a trend and some seasonality over a period of time is called as a time series data. Seasonality is defined as how the data is repeating its trend over a period of time. Hence time series data is periodic in nature. Below is the example of time series data of MMM from 2000 to 2019


MMM stock price(Train and Test)
250
200
150
100
50
0
1000
2000
3000
4000
5000

Chapter 3

System Analysis & Design

3.1 Requirement Analysis

Here we make use of technical analysis to predict the future stock price movement using firms historical Data. Technical parameter on their own don't predict stock price but historical data can be used by technical Parameter to predict the stock price on current market situation. Technical analysis helps the investor to have a idea of predicted the stock price We will save the company tickers of S&P 500 list from Wikipedia and extract stock data against every company ticker. Then we will take the close index of each company and put it into one data frame and we will try to find a correlation between each company and then we will preprocess the data and establish different technical parameters based on stock price, volume and close price and based on the movement of prices will develop technical indicators that will help us set a target percentage to predict buy, sell,, hold.

3.2 Data Extraction

[3] To begin with we need a list of companies for which we could develop statistical models and predict future conditions of stock. Each company will have its own record of stock data from 2000 to 2017/2019. Firstly we need a list of companies, so we take the S&P 500 list, we extract the data from Wikipedia, there the S&P list is in a table format which is easy to manipulate. First we visit and hit the Wikipedia page, and we are given the response, which contains our source code, to treat it how we want we access the .text attribute of it which we turn into a soup object using BeautifulSoup4 which is HTML parsing library. Once we've had the soup we found the table of stock data by simply searching for 'wiki table sortable classes' and accessing the table 0th column and first row of every entry to get the ticker symbol for 500 companies arranged according to their total market cap.For each table row we say that, ticker is the table data, we grab the .text of it and we append this ticker to our list, to save the list we use pickle and if the list changes we can modify it to check for specific periods of time. We are saving the list of tickers, so not to hit Wikipedia again and again every time we run the script. Now that we

have tickets of 500 companies, we need the stock pricing data of each company. We extract the stock pricing data of the first 20 companies; each company has stock data to around 6000 entries for each company. The companies which were started after 2000 and has empty values they are replaced by zero.We extract the data using pandas-datareader, a python extracting library.We extract the data from Morningstar and saved the data locally in .csv format and the data is used for further manipulations. Now that we have stock data and company tickers we can move further to data manipulation and what indexes do we have in our data

```
def save_sp500_tickers():
    resp = requests.get('https://en.wikipedia.org/wiki/List_of_S%26P_500_companies')
    soup = bs.BeautifulSoup(resp.text, "lxml")
    table = soup.find('table',{'class':'wikitable sortable'})
    tickers = []
    for row in table.findAll('tr')[1:]:
        ticker = row.findAll('td')[0].text
        ticker=ticker.rstrip("\n\r")
        tickers.append(ticker)

    with open("sp500tickers.pickle","wb") as f:
        pickle.dump(tickers, f)

    print(tickers)

    return(tickers)
```

Fig3.2.1

```
arun@arun-inspiron-3521:~/Desktop/stocks$ python gettingsnp500.py
['MMM', 'ABT', 'ABBV', 'ABMD', 'ACN', 'ATVI', 'ADBE', 'AMD', 'AAP', 'AES', 'AMG', 'AFL', 'A', 'APD', '
', 'AGN', 'ADS', 'LNT', 'ALL', 'GOOGL', 'GOOG', 'MO', 'AMZN', 'AEE', 'AAL', 'AEP', 'AXP', 'AIG', 'AMT'
', 'ADI', 'ANSS', 'ANTM', 'AON', 'AOS', 'APA', 'AIV', 'AAPL', 'AMAT', 'APTV', 'ADM', 'ARNC', 'ANET', '
VB', 'AVY', 'BHGE', 'BLL', 'BAC', 'BK', 'BAX', 'BBT', 'BDX', 'BRK.B', 'BBY', 'BIIB', 'BLK', 'HRB', 'BA
R', 'BF.B', 'CHRW', 'COG', 'CDNS', 'CPB', 'COF', 'CPRI', 'CAH', 'KMX', 'CCL', 'CAT', 'CBOE', 'CBRE', '
'CF', 'SCHW', 'CHTR', 'CVX', 'CMG', 'CB', 'CHD', 'CI', 'XEC', 'CINF', 'CTAS', 'CSCO', 'C', 'CFG', 'CTX
SA', 'CMA', 'CAG', 'CXO', 'COP', 'ED', 'STZ', 'COO', 'CPRT', 'GLW', 'COST', 'COTY', 'CCI', 'CSX', 'CMI
, 'XRAY', 'DVN', 'FANG', 'DLR', 'DFS', 'DISCA', 'DISCK', 'DISH', 'DG', 'DLTR', 'D', 'DOV', 'DOW', 'DWD
TN', 'EBAY', 'ECL', 'EIX', 'EW', 'EA', 'EMR', 'ETR', 'EOG', 'EFX', 'EQIX', 'EQR', 'ESS', 'EL', 'EVRG',
 'FFIV', 'FB', 'FAST', 'FRT', 'FDX', 'FIS', 'FITB', 'FE', 'FRC', 'FISV', 'FLT', 'FLIR', 'FLS', 'FLR',
'FOX', 'BEN', 'FCX', 'GPS', 'GRMN', 'IT', 'GD', 'GE', 'GIS', 'GM', 'GPC', 'GILD', 'GPN', 'GS', 'GWW',
'HCP', 'HP', 'HSIC', 'HSY', 'HES', 'HPE', 'HLT', 'HFC', 'HOLX', 'HD', 'HON', 'HRL', 'HST', 'HPQ', 'HUM
'IR', 'INTC', 'ICE', 'IBM', 'INCY', 'IP', 'IPG', 'IFF', 'INTU', 'ISRG', 'IVZ', 'IPGP', 'IQV', 'IRM', '
 'JPM', 'JNPR', 'KSU', 'K', 'KEY', 'KEYS', 'KMB', 'KIM', 'KMI', 'KLAC', 'KSS', 'KHC', 'KR', 'LB', 'LLL
, 'LIN', 'LKQ', 'LMT', 'L', 'LOW', 'LYB', 'MTB', 'MAC', 'M', 'MRO', 'MPC', 'MAR', 'MMC', 'MLM', 'MAS',
 'MRK', 'MET', 'MTD', 'MGM', 'MCHP', 'MU', 'MSFT', 'MAA', 'MHK', 'TAP', 'MDLZ', 'MNST', 'MCO', 'MS', '
, 'NTAP', 'NFLX', 'NWL', 'NEM', 'NWSA', 'NWS', 'NEE', 'NLSN', 'NKE', 'NI', 'NBL', 'JWN', 'NSC', 'NTRS'
XY', 'OMC', 'OKE', 'ORCL', 'PCAR', 'PKG', 'PH', 'PAYX', 'PYPL', 'PNR', 'PBCT', 'PEP', 'PKI', 'PRGO', '
```

Fig 3.2.2

```
def get_data_from_morningstar(reload_sp500=False):
    if reload_sp500:
        tickers = save_sp500_tickers()
    else:
        with open("sp500tickers.pickle","rb") as f:
            tickers = pickle.load(f)

    if not os.path.exists('stock_dfs'):
        os.makedirs('stock_dfs')

    start = dt.datetime(2000,1,1)
    end = dt.datetime.now()

    for ticker in tickers[:20]:
        print(ticker)
        str1 ="new"
        ticker_append = ticker + str1
        if not os.path.exists('stock_dfs/{}.csv'.format(ticker_append)):
            df = web.DataReader(ticker,'yahoo',start,end)
            df.to_csv('stock_dfs/{}.csv'.format(ticker))
            df=pd.read_csv('stock_dfs/{}.csv'.format(ticker))
            df_col =['Date','Open','High','Low','Close','Volume','Adj Close']
            new_df=df[df_col]
            print(new_df.head())
            new_df.to_csv('stock_dfs/{}.csv'.format(ticker_append),index=False)
            print(df.tail())


        else:
            df=pd.read_csv('stock_dfs/{}.csv'.format(ticker_append))
            print(df.tail())
            print('Already Have{}'.format(ticker_append))
```

Fig 3.2.3

```
MMM
           Date        Open        High         Low       Close     Volume  \
4864  2019-05-03  185.809998  186.690002  184.089996  185.220001  4747700.0
4865  2019-05-06  182.039993  183.110001  180.130005  183.039993  6517600.0
4866  2019-05-07  181.809998  181.899994  177.809998  179.119995  5415600.0
4867  2019-05-08  179.509995  180.529999  178.500000  178.589996  2981500.0
4868  2019-05-09  177.300003  177.399994  174.186493  176.425003  2933190.0

      Adj Close
4864  185.220001
4865  183.039993
4866  179.119995
4867  178.589996
4868  176.425003
Already HaveMMMnew
ABT
           Date       Open       High        Low      Close     Volume  \
4864  2019-05-03  78.830002  79.129997  78.260002  78.690002  5452700.0
4865  2019-05-06  77.419998  79.169998  77.349998  79.070000  4006700.0
4866  2019-05-07  78.500000  78.870003  76.199997  76.910004  5322500.0
4867  2019-05-08  76.930000  76.970001  76.139999  76.220001  3959300.0
4868  2019-05-09  75.660004  75.940002  74.879997  75.940002  1828528.0
```

Fig 3.2.4

3.3 Data Manipulation

Now that we have stock pricing data of companies, we are going to bring this data together in one data frame. Now we have to access the data together. To access the data together we will join all of the dataset together. Every data set contains: Open, High, Low,Close,Volume. For now we are interested in close/AjClose for now. We make use of list of tickers and start with empty data frame, now the data has to be read in each stock data frame .We will majorly focus in the Close columns. Now we intend to rename the column name to the ticker name and we will build a shared data frame. We use pandas python librarry to outer join the data frame and if the main_df is empty, then we start with current df, else we will use join. Now we will visualize the correlation between the companies ,we can make use of matplotlib with Numpy to visualize it. Mapping style 'ggplot' was used, to build a correlation with panda we will add 'main_combine_dataframe.corr()' and it will automatically look in the entire dataframe And it will find every correlation between columns Now we will make a heat map. To do this we need actual data to graph This will give a numpy array of just values, which are basically correlation numbers Then we build our figure and axes. By observing the correlations, we se that we have many relationships. The majority of companies are positively related. There are a few that are strongly connected there are some that are negatively correlated and some are not correlated at all

By observing the correlations, we se that we have many relationships.

The majority of companies are positively related. There are a few that are strongly connected there are some that are negatively correlated and some are not correlated at all.

```python
def compile_data():
    with open("sp500tickers.pickle","rb") as f:
        tickers = pickle.load(f)

    main_df = pd.DataFrame()
    count_ = 0
    for count,ticker in enumerate(tickers):
        count_ = count_+1;
        if count_ >=20:
            break;
        else:
            str1 = "new"
            ticker_append = ticker + str1
            #print(ticker_append)
            df = pd.read_csv('stock_dfs/{}.csv'.format(ticker_append))
            df.set_index('Date', inplace=True)

            df.rename(columns = {'Adj Close':ticker}, inplace=True)
            df.drop(['Open','High','Low','Close','Volume'], 1, inplace=True)

            if main_df.empty:
                main_df = df
            else:
                main_df = main_df.join(df, how='outer')

            if count % 10 == 0:
                #print(count)

                #print(main_df.head())
                main_df.to_csv('sp500_joined_closes.csv')
```

Fig3.3.1

	Date	MMM	ABT	ABBV	ABMD	ACN	\
4864	2019-05-03	185.220001	78.690002	78.709999	271.750000	176.979996	
4865	2019-05-06	183.039993	79.070000	79.260002	268.950012	176.259995	
4866	2019-05-07	179.119995	76.910004	77.949997	261.980011	173.940002	
4867	2019-05-08	178.589996	76.220001	77.989998	260.269989	173.820007	
4868	2019-05-09	176.425003	75.940002	77.879997	260.290009	173.830002	

	ATVI	ADBE	AMD	AAP	AES	AMG
4864	47.150002	285.579987	28.219999	163.270004	16.930000	110.820000
4865	48.160000	283.660004	27.420000	161.990005	16.809999	97.959999
4866	46.680000	277.070007	26.660000	160.660004	16.530001	94.089996
4867	46.820000	276.769989	27.090000	158.619995	15.980000	93.970001
4868	46.680000	275.329987	27.139999	159.000000	16.084999	93.699997

Fig 3.3.2

```python
def visualize_data():
    df = pd.read_csv('sp500_joined_closes.csv')
    print(df.tail())
    df_corr=df.corr()

    print(df_corr.head())

    data = df_corr.values
    fig = plt.figure()
    ax = fig.add_subplot(1,1,1)

    heatmap = ax.pcolor(data , cmap=plt.cm.RdYlGn)
    fig.colorbar(heatmap)
    ax.set_xticks(np.arange(data.shape[0]) + 0.5, minor=False)
    ax.set_yticks(np.arange(data.shape[1]) + 0.5, minor=False)
    ax.invert_yaxis()
    ax.xaxis.tick_top()

    column_labels = df_corr.columns
    row_labels = df_corr.index

    ax.set_xticklabels(column_labels)
    ax.set_yticklabels(row_labels)
    plt.xticks(rotation=90)
    heatmap.set_clim(-1,1)
    plt.tight_layout()
    plt.show()
```

Fig 3.3.3

```
          MMM       ABT       ABBV      ABMD       ACN       ATVI      ADBE
MMM   1.000000  0.953933  0.929524  0.814625  0.977191  0.944400  0.896382
ABT   0.953933  1.000000  0.873466  0.854767  0.977803  0.905836  0.924142
ABBV  0.929524  0.873466  1.000000  0.889956  0.924985  0.907073  0.910206
ABMD  0.814625  0.854767  0.889956  1.000000  0.880147  0.895268  0.969495
ACN   0.977191  0.977803  0.924985  0.880147  1.000000  0.947388  0.935070

          AMD       AAP       AES       AMG
MMM  -0.173949  0.870515 -0.209017  0.786405
ABT  -0.173541  0.878077 -0.156174  0.750746
ABBV  0.739828  0.237890  0.393745 -0.211328
ABMD  0.148422  0.645603  0.002311  0.405965
ACN  -0.014707  0.874624  0.165083  0.714480
```

Fig 3.3.4

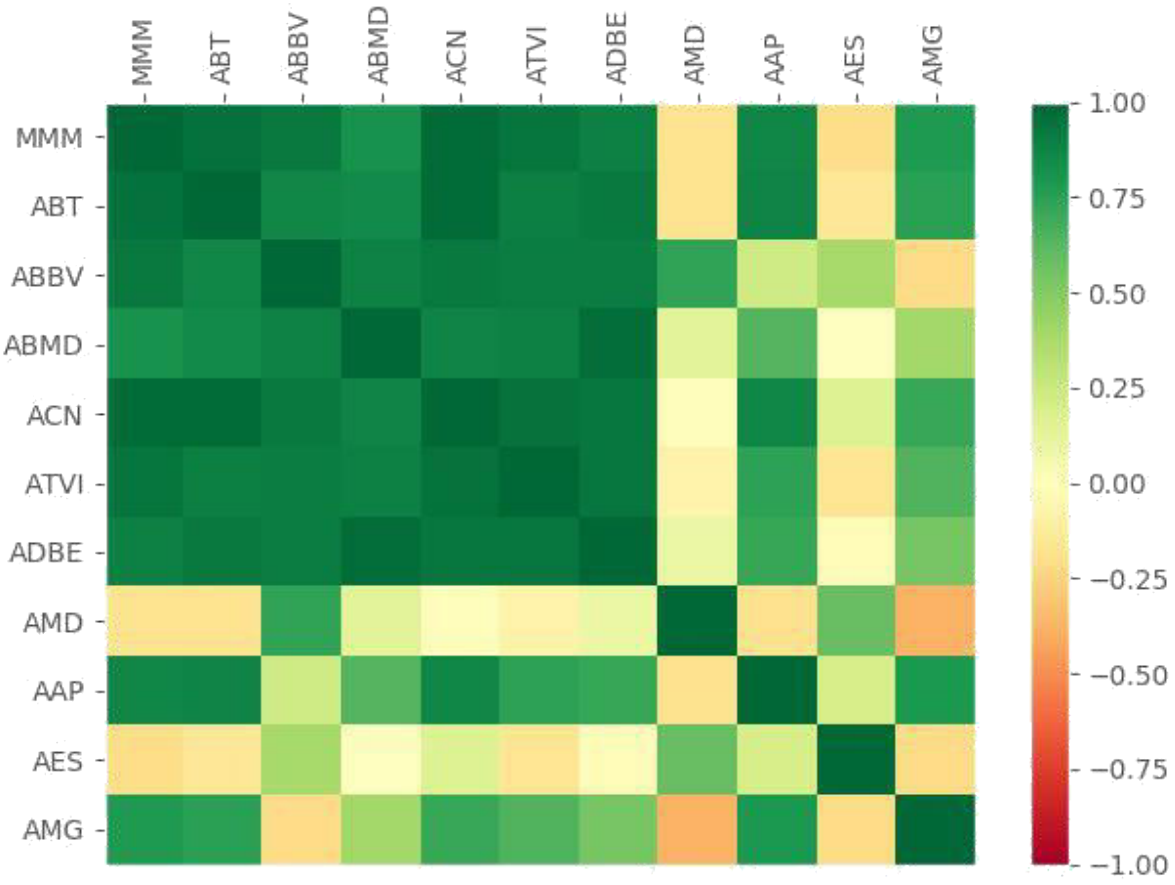

MMM
ABT
ABBV
ABMD
ACN
ATVI
ADBE
AMD
AAP
AES
AMG
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

Fig 3.3.5

3.4 Preprocessing the Data to prepare for Machine Learning

Lets take the data of different companies and apply the required ML classifiers .as we have seen. Before that some companies have relationship with other companies , the goal here is that to predict.Stock price of tommorrow for a company by observing the changes in price today .This can only be done if machine can identify and fit these relationship between companies We start with labelling using machine learning ,thus we need to convert our dataset to feature and label Features can be pricing data of companies but we will consider features as pricing data of all of the companies and the label here is to buy or dont buy any company stock. Let's consider3M (MMM). Feature set takes into account that all companies % changes that day and those will be our features.label here is that whether or not 3M (MMM) rises more than x% in next x days, X can have any value as required . Start with a company is a buy only if, within the next 7 working days, price goes up more than 2% and sell the stock if the price goes down more than 2% within those 7 days.

When the algorithm suggests to buy a company stock investor can place a 2% drop stop loss.If the company has risen by 2% investor can sell the stock or hold, investor can make a strategy by it.A same strategy model can be built by training data which will give price in the future.A function is created which will take 1 parameter, the ticker in question. every model will be trained on A single company, suppose we want to predict the price for the next 7 days. we will observe the close Price for all companies and grab the list of all the tickers of the companies, fill the missing values with 0 No we have to find the % change in the price for next 7 days. We observe that new data frame columns were created for a specific ticker in question, We will use string formatting to create the custom name as required. We are getting the future values by. shift which is basically a shift in the column up or down . Here we observe a negative amount i.e. that column is shifted UP by i rows this is how we get the future value of i days % change is calculated. We have successfully created the function which will create label. We have a factor that can tell buy/sell/hold. If the prices changes in future by 2% it will give an output to buy or sell accordingly.
X stores our feature sets (daily % changes for every company in the S&P 500).

Y is "target" or "label." now we map our feature sets to.

```
ef process_data_for_labels(ticker):
    hm_days = 7
    df = pd.read_csv('sp500_joined_closes.csv', index_col=0)
    print(df.head())
    tickers = df.columns.values.tolist()
    df.fillna(0,inplace=True)
    return tickers, df
```

```
def buy_sell_hold(*args):
    cols = [c for c in args]
    requirement = 0.02
    for col in cols:
        if col > requirement:
            return 1
        elif col < requirement:
            return -1
        elif (col == requirement):
            return (0)
```

Fig 3.4.1

```
def extract_featuresets(ticker):
    tickers, df = process_data_for_labels(ticker)

    df['{}_target'.format(ticker)] = list(map(buy_sell_hold,
                                              df['{}_1d'.format(ticker)],
                                              df['{}_2d'.format(ticker)],
                                              df['{}_3d'.format(ticker)],
                                              df['{}_4d'.format(ticker)],
                                              df['{}_5d'.format(ticker)],
                                              df['{}_6d'.format(ticker)],
                                              df['{}_7d'.format(ticker)]
                                              ))

    vals = df['{}_target'.format(ticker)].values.tolist()
    str_vals = [str(i) for i in vals]
    print('Data Spread:', Counter(str_vals))

    df.fillna(0, inplace = True)
    df = df.replace([np.inf, -np.inf], np.nan)
    df.dropna(inplace = True)

    df_vals = df[[ticker for ticker in tickers]].pct_change()
    df_vals = df_vals.replace([np.inf, -np.inf], 0)
    df_vals.fillna(0, inplace = True)

    X = df_vals.values

    y = df['{}_target'.format(ticker)].values

    return X, y, df
```

Fig 3.4.2

3.5 Introduction To Classifiers

Classification is supervised here it predic the class of given data sets.Classification predictive Models maps the function from input variable X to output Varible y. The classifier uses the dataset as training dataset and testing datasetIf the model is trained properly it can classify the output effectively. As discussed before classification falls into the category of supervised learning where the targets is given an input data e.g. credit approval, heart diseases learners in classification are of 2 types namely lazy learners and eager learners.

1.Lazy learners

2.eager learners

1.Lazy learners : it stores the data in training and halts untill a testing data occured Hence classification is done , it have less training time and more testing time e.g. k-nn, Case-based reasoning

2. Eager learners

Classification model is given training data before giving it order to classification Here it have more training time and less testing time Ex. Decision Based Tree, Naive Bayes, ANN

3.5.1 Different Types Of Classifier Used

K Nearest Neighbours - K-nn is a non parametric method of classification in recognizing the pattern .Input is given as k closet of training example in feature space Output gives the classification. In k-NN classification, the output is given as class membership.

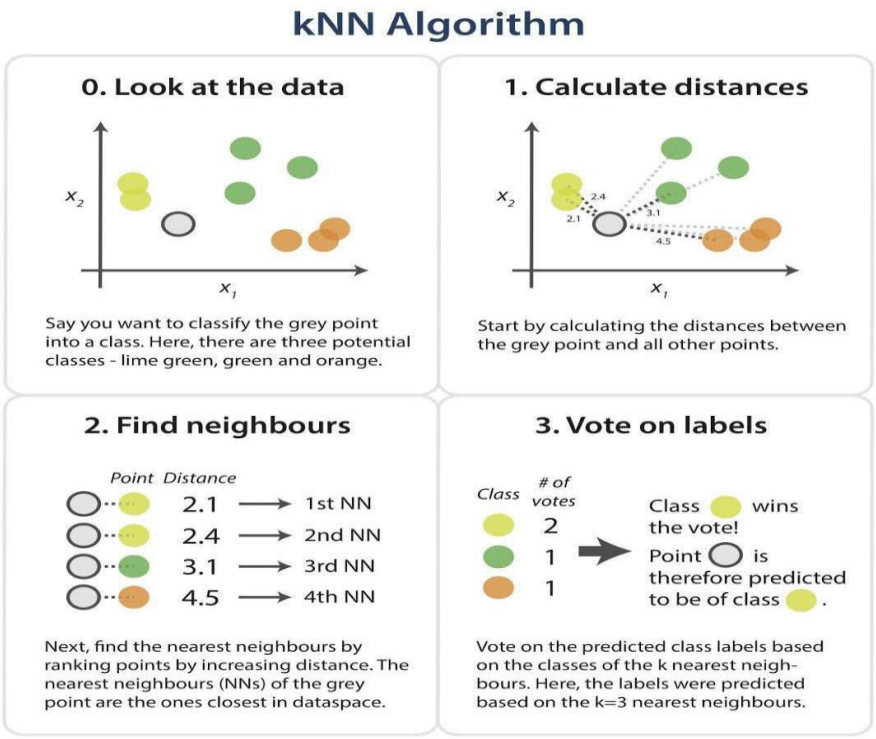


Figure 3.5.1 [4]

Support Vector Machines - Support vector machines (SVMs, also support vector networks) are One of the supervised learning models which analyze the data to give classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model

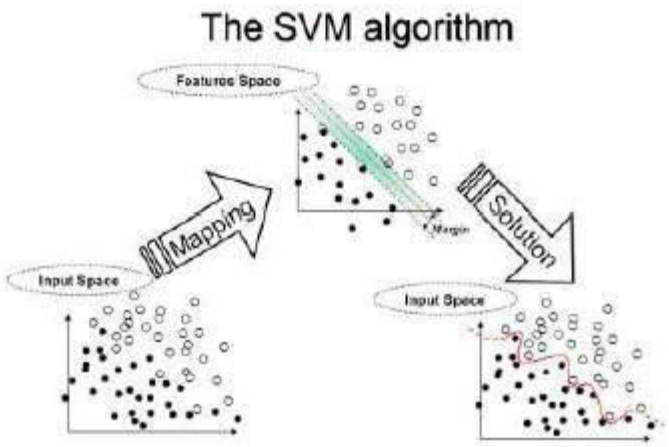


Figure 3.5.2 [5]

Random Forest Classifier - it is used to classify the conditions of buy sell and hold based on the predicted value and the market conditions. In random foret classification is done by making multitude of different decision trees but in training.

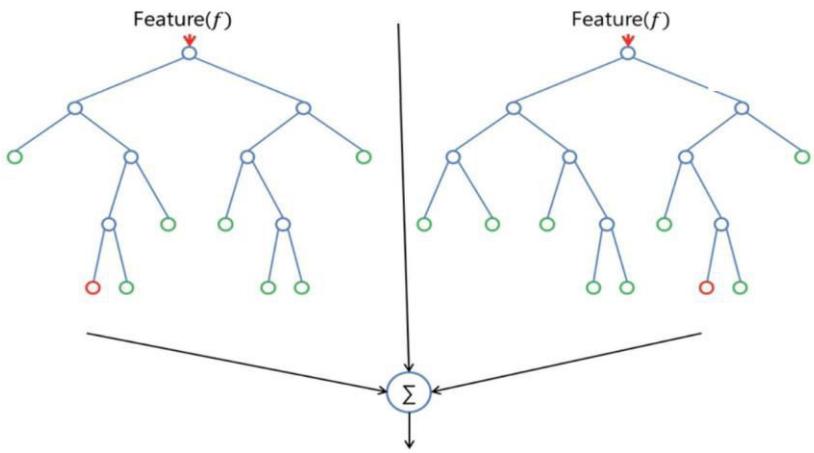


Figure 3.5.3 [6]

3.6 Performing Machine Learning

```
def do_ml(ticker):
    X, y, df = extract_featuresets(ticker)

    X_train, X_test, y_train, y_test = cross_validation.train_test_split(X, y, test_size=0.25)

    clf = VotingClassifier([('lsvc', svm.LinearSVC()),
                            ('knn', neighbors.KNeighborsClassifier()),
                            ('rfor', RandomForestClassifier())])

    clf.fit(X_train, y_train)
    confidence = clf.score(X_test, y_test)
    print('Accuracy:', confidence)
    predictions = clf.predict(X_test)
    print('Predicted Class Counts:', Counter(predictions))
    return confidence
do_ml('MMM')
```

Figure 3.6.1

3.7 Time Series Forecasting Algorithms

- Stock Data of MMM is stored into a dataframe called “df”.
- Data is then subnetted and splitted into traing and testing data.
- Different time series algorithms is trained and tested.
- based upon the result of the prediction in trend and error algorithms ae compared .

Different types of algorithm that are used are explained below:

3.7.1 Simple Exponential Smoothning [7] It comes from the extension of simple average and weighted moving average. Here more weightage to the latest data rather than old data over time. i.e. more weight is given to the new data. Forecasting is done here by calculating average weight of exponentially decreasing weights into the past as defined by the formula below.

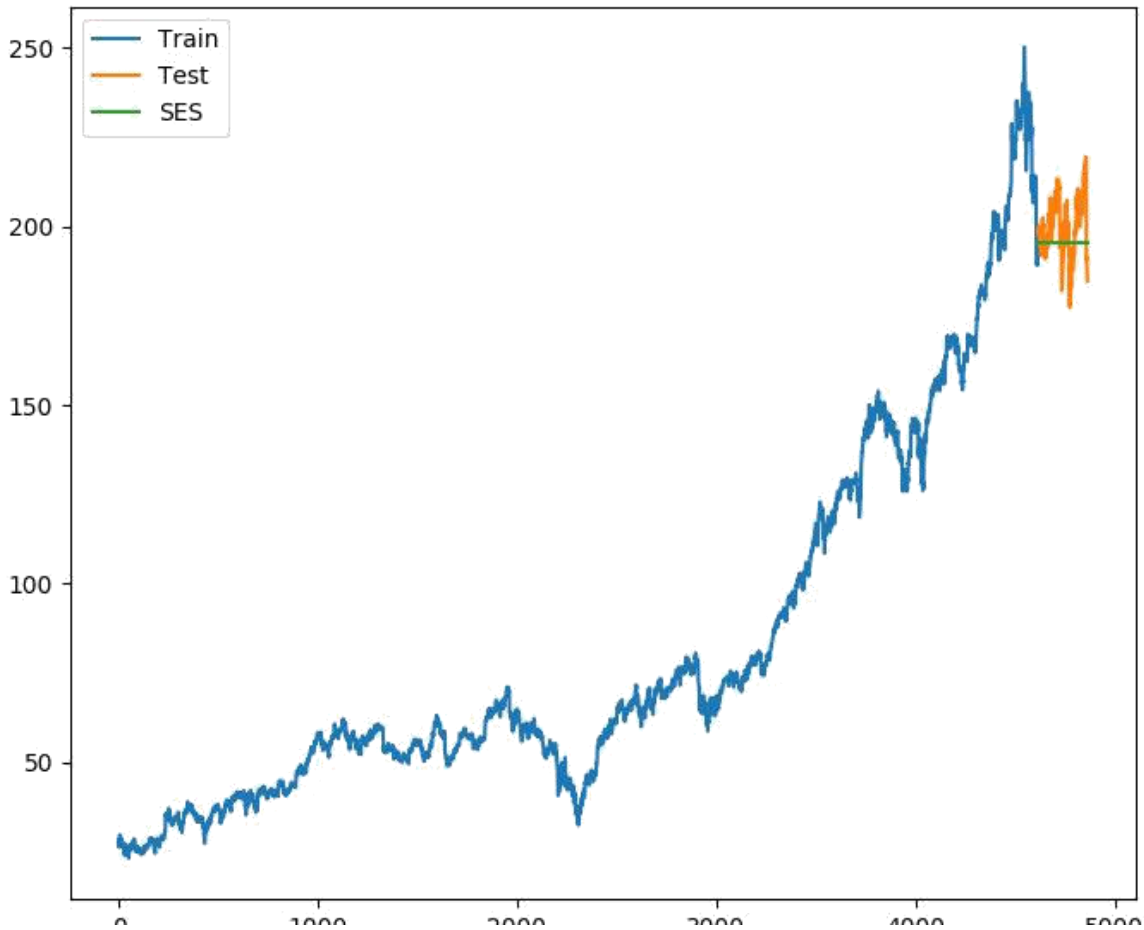
$$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \cdots$$

where $0 \leq \alpha \leq 1$ and its is called the smoothening parameter and here we take its value 0.6

```
def simple_exponentially_smoothening():

    y_hat_avg = test.copy()
    fit2 = SimpleExpSmoothing(np.asarray(train['AdjClose'])).fit(smoothing_level=0.6,optimized=Fals
    y_hat_avg['SES'] = fit2.forecast(len(test))
    plt.figure(figsize=(8,8))
    plt.plot(train['AdjClose'], label='Train')
    plt.plot(test['AdjClose'], label='Test')
    plt.plot(y_hat_avg['SES'], label='SES')
    plt.legend(loc='best')
    plt.show()
```

Figure 3.7.1


Train
Test
SES
250
200
150
100
50

3.7.2 Holt winters linear

[8] Holt winters linear model takes ina cooount that time series data have a trend between a point a and b. Trend can be increasing and decreasing depending upon the trend and exponentially decreasing weights for past data can help in better prediction of trend in stock price. Model finds the prameters like residual,level and trend in the data. This can be used to predict the future trend of stock. Forecast is now done by level and trend also along with the forcast equation.

$$\text{Forecast equation : } \hat{y}_{t+h|t} = \ell_t + h\, b_t$$

$$\text{Level equation : } \ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1}+b_{t-1})$$

$$\text{Trend equation : } b_t = \beta*(\ell_t - \ell_{t-1})+(1-\beta)b_{t-1}$$

```
def Holt_linear():

    y_hat_avg = test.copy()

    fit3 = Holt(np.asarray(train['AdjClose'])).fit(smoothing_level = 0.3,smoothing_slope = 0.001)
    y_hat_avg['Holt_linear'] = fit3.forecast(len(test))

    plt.figure(figsize=(8,8))
    plt.plot(train['AdjClose'], label='Train')
    plt.plot(test['AdjClose'], label='Test')
    plt.plot(y_hat_avg['Holt_linear'], label='Holt_linear')
    plt.legend(loc='best')
    plt.show()

    rms1 = sqrt(mean_squared_error(test.AdjClose, y_hat_avg.Holt_linear))
    print('\nroot mean square error for holt linear ',rms1)
```

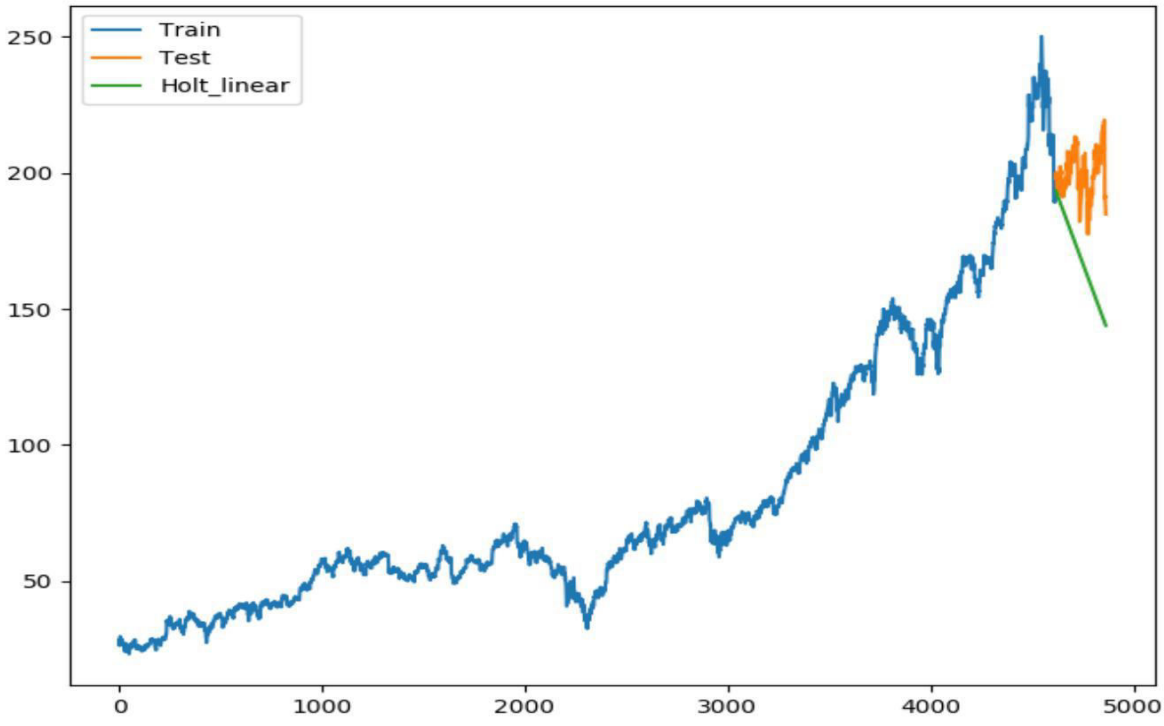


Figure 3.7.3

3.7.3 Holt winters Method(with seasonality)

[9] Both of the above method have taken into account the importance of exponentially decreasing weight and trend of the time series data. Here both of the approaches are valid in addition one more property is added to the list and it is seasonality (defined as the how the data repeats itself periodically over a period of time)

$$
\begin{aligned}
\text{level} \quad L_t &= \alpha(y_t - S_{t-s}) + (1-\alpha)(L_{t-1} + b_{t-1}); \\
\text{trend} \quad b_t &= \beta(L_t - L_{t-1}) + (1-\beta)b_{t-1}, \\
\text{seasonal} \quad S_t &= \gamma(y_t - L_t) + (1-\gamma)S_{t-s} \\
\text{forecast} \quad F_{t+k} &= L_t + kb_t + S_{t+k-s},
\end{aligned}
$$

where s is the length of the seasonal cycle, for $0 \leq \alpha \leq 1$, $0 \leq \beta \leq 1$ and $0 \leq \gamma \leq 1$.

we have taken the seasonal lenght here as 12 because it is observed that behaviour of t the sock repeats itself after every year to some extend.

```
ef Holt_Winters_method():
   from statsmodels.tsa.api import ExponentialSmoothing
   y_hat_avg = test.copy()
   fit1 = ExponentialSmoothing(np.asarray(train['AdjClose']) ,seasonal_periods=12,trend='add', /
   seasonal='add',).fit()
   y_hat_avg['Holt_Winter'] = fit1.forecast(len(test))
   plt.figure(figsize=(8,8))
   plt.plot( train['AdjClose'], label='Train')
   plt.plot(test['AdjClose'], label='Test')
   plt.plot(y_hat_avg['Holt_Winter'], label='Holt_Winter')
   plt.legend(loc='best')
   plt.show()
   rms3 = sqrt(mean_squared_error(test.AdjClose, y_hat_avg.Holt_Winter))
   print('\nroot mean square error for holt linear ',rms3)
```

Figure 3.7.4

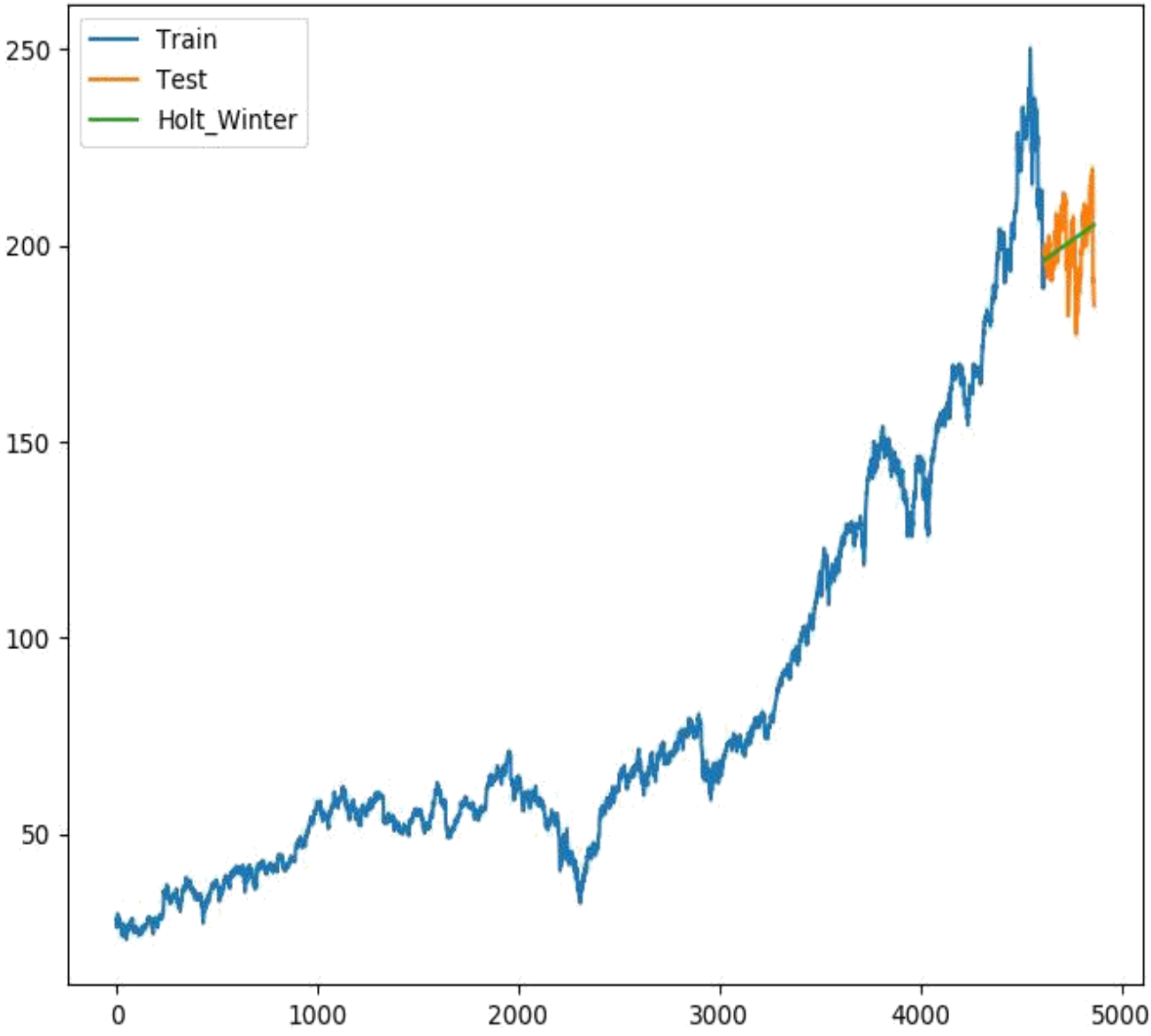

Train
Test
Holt_Winter
250
200
150
100
50
0
1000
2000
3000
4000
5000

Figure 3.7.5

3.7.3 ARIMA(Autoregressive Integrated Moving Average)

[10] ARIMA finds the correlations in the data with each other rather than just finding the seasonality and tred in the data. Athis algorithm have an extension which uses seasonality concept it is more accurate than holt winters smoothening method.

```
def SARIMAX():
    y_hat_avg = test.copy()
    fit4 = sm.tsa.statespace.SARIMAX(train.AdjClose, order=(2, 1, 4),/
    seasonal_order=(0,1,1,12)).fit()
    y_hat_avg['SARIMA'] = fit4.predict(start=4617, end=4864, dynamic=True)
    plt.figure(figsize=(8,8))
    plt.plot( train['AdjClose'], label='Train')
    plt.plot(test['AdjClose'], label='Test')
    plt.plot(y_hat_avg['SARIMA'], label='SARIMA')
    plt.legend(loc='best')
    plt.show()

    rms4 = sqrt(mean_squared_error(test.AdjClose, y_hat_avg.SARIMA))
    print('\nroot mean square for ARIMA:',rms4)
```

Figure 3.7.6

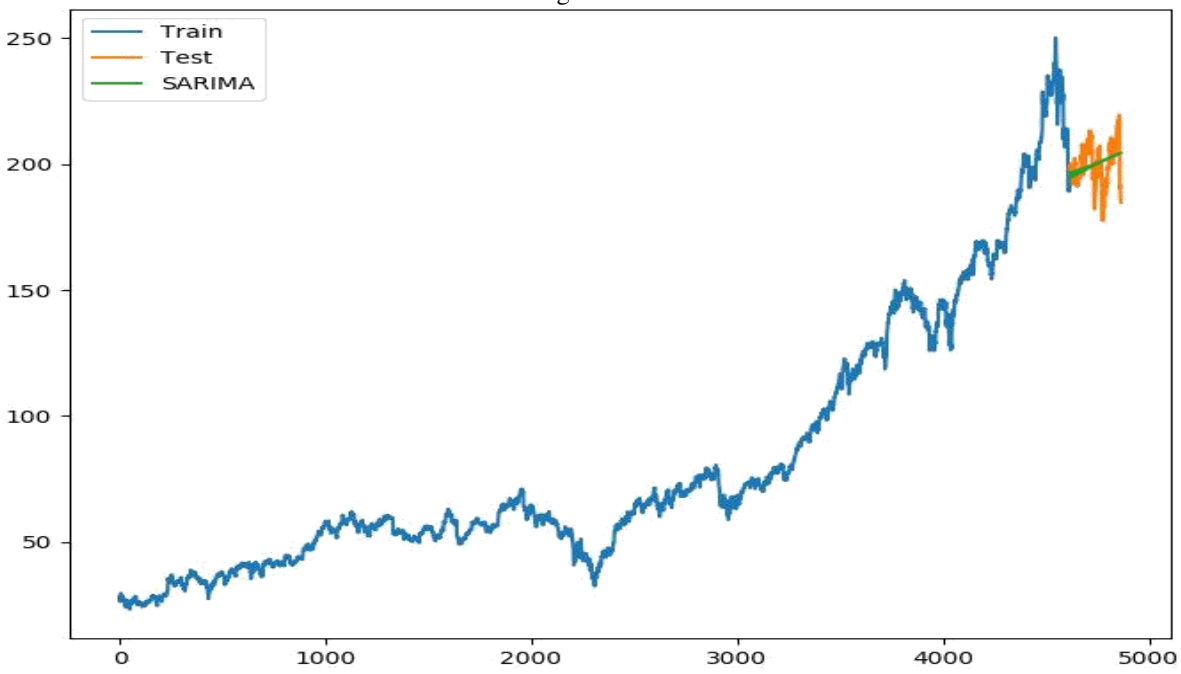


Figure 3.7.7

Chapter 4

Results and Outputs

4.1 Using Classifier

As we can see in the figure given below, one side it shows the predicted counter spread of the company future prices, and another figure shows the graph of the company at that particular time of year in terms to our prediction and we can observe that much of the results are accurate. As we can see that the data spread is mostly saying buy the stock, we can be wrong on the hold condition because the training data will never be perfectly balanced ever, so supposedly if the model predicted buy then this would be 1722 correct out of 4527 which is still good and actually a better score than we got, and we still are reaching the above accuracy mark of 33% which is good in a stock market. Many conditions will still be there which machines can miss out, supposedly we have our conditions to buy, sell, hold and sometimes the model can be penalised, say the model expected a 2% rise in the next seven days, but the rise only went up to 1.5% and went 2% the next day, then the model will predict buy, hold according to the 1.5% rise in the seven days and give the expected spread. A model can also be penalised if supposedly the rise went 2% up and then suddenly drops 2% low the next day, this sort of outcomes in actual trading would be considerate, and same goes for the model of it turns out to be highly accurate. Now looking at the spread and the graph of the company notice around the period of 2017 the company was rising in the market so therefore there were actually more buys, which rapidly dropped in 2018, but the data we extracted was till 31, December 2017 and it shows that at the starting of the year it had lot of buys, hence 1722 out of 4527 which rapidly was sold just in a short time hence a lot of sells more than the holds, giving 1424 out 4527, the model may not be perfectly accurate but has a very close range of decisions which can be accepted in real trading or using algorithms to trade.

```
/usr/local/lib/python3.6/dist-packages/sklearn/cross_validation.
odel_selection module into which all the refactored classes and
 from that of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
Data spread: Counter({'1': 1722, '-1': 1424, '0': 1381})
/usr/local/lib/python3.6/dist-packages/sklearn/preprocessing/lab
g False, but in future this will result in an error. Use `array.
  if diff:
accuracy: 0.3654867256637168
/usr/local/lib/python3.6/dist-packages/sklearn/preprocessing/lab
g False, but in future this will result in an error. Use `array.
  if diff:
predicted class counts: Counter({1: 531, -1: 425, 0: 174})
```

Figure 4.1

4.2 Using Time Series Forecasting Algorithm

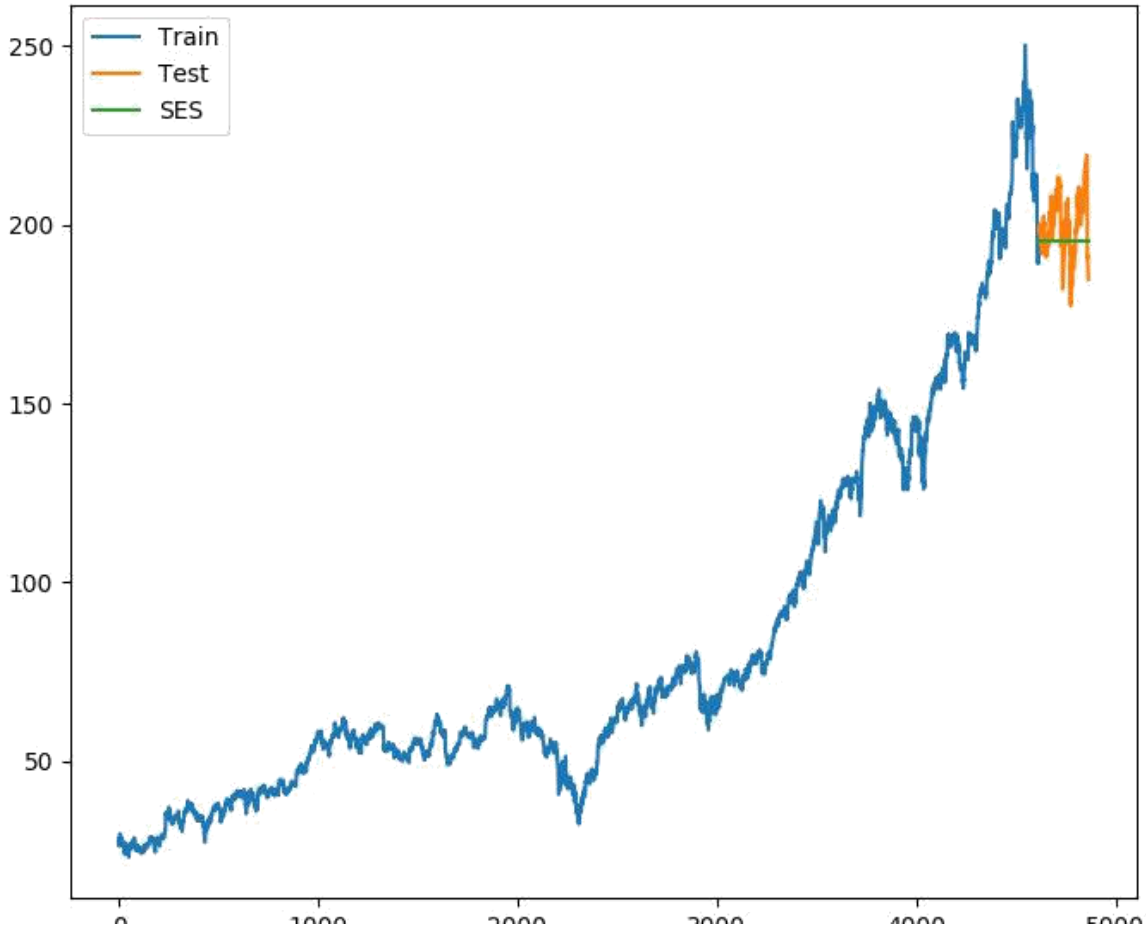


Figure 4.2

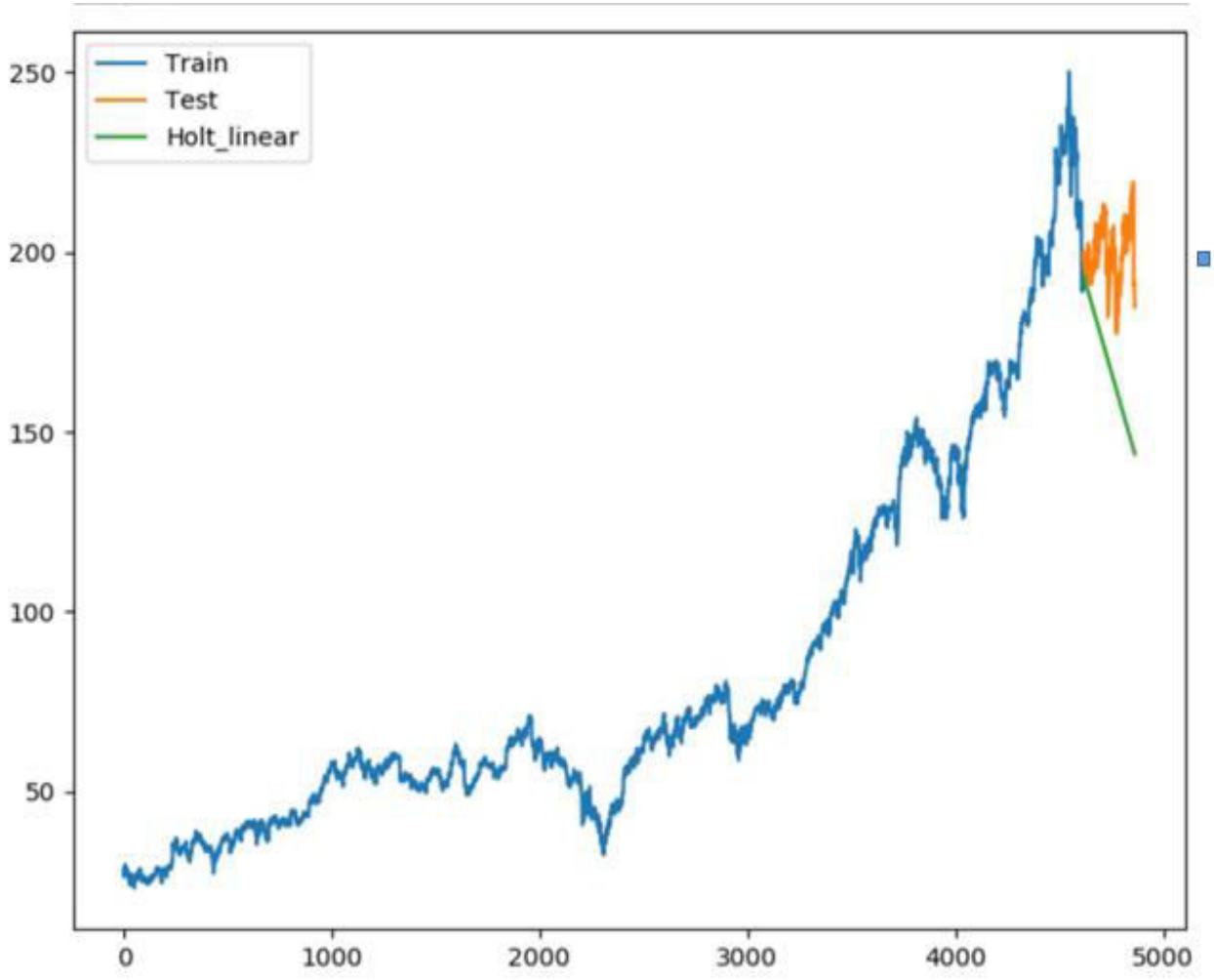

Train
Test
Holt_linear
250
200
150
100
50
0
1000
2000
3000
4000
5000

Figure 4.3

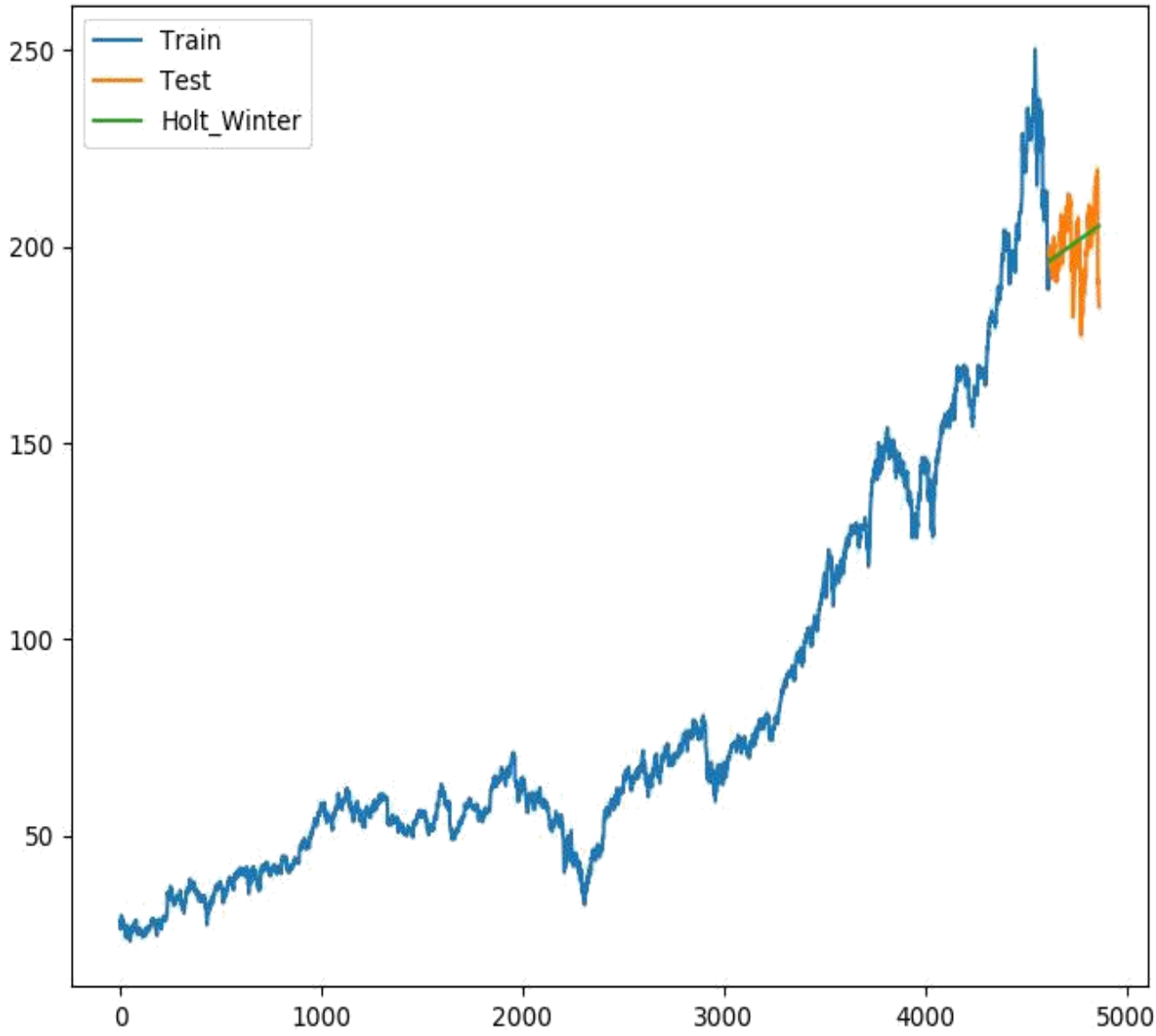

Train
Test
Holt_Winter
250
200
150
100
50
0
1000
2000
3000
4000
5000

Figure 4.4

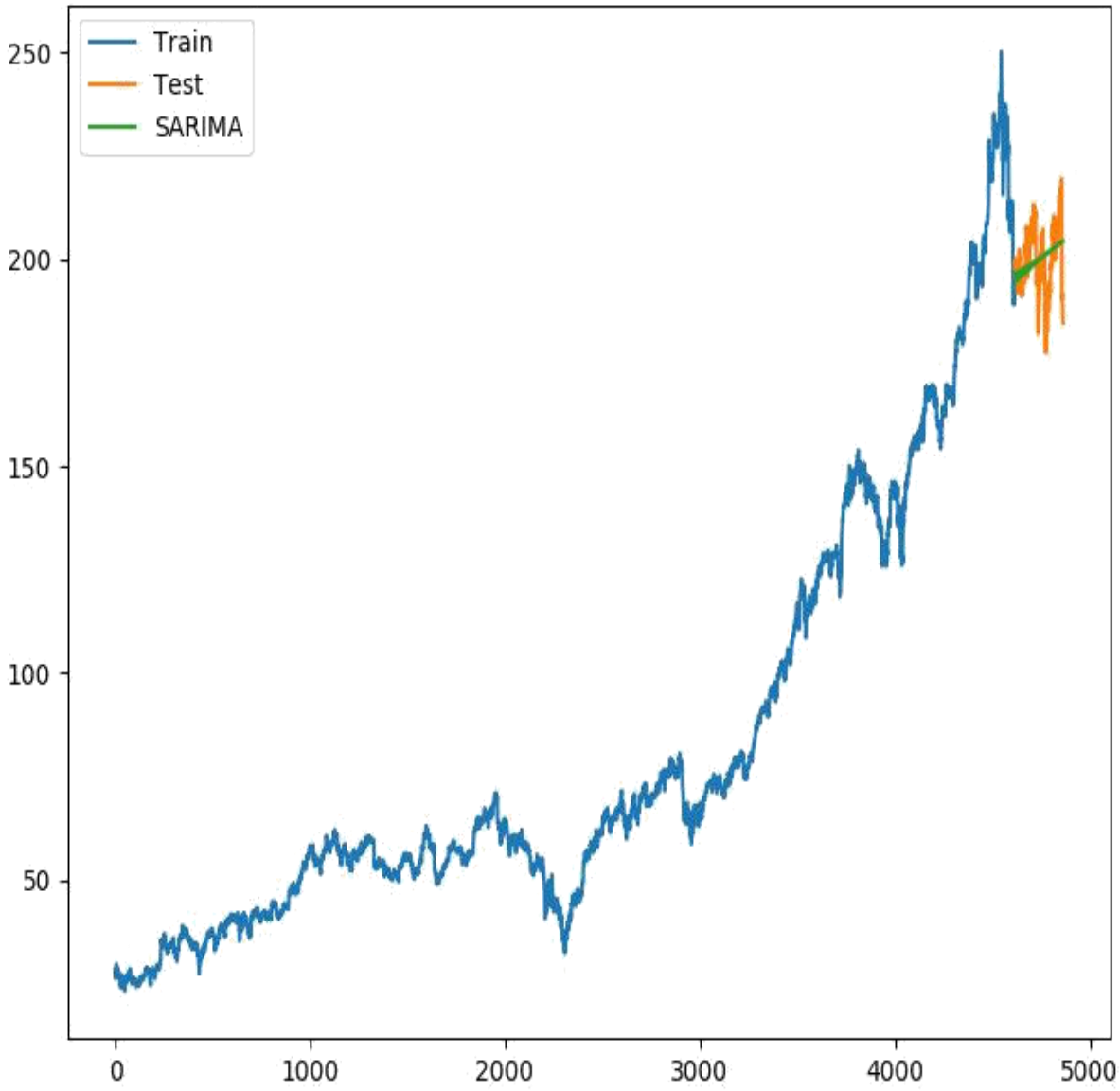

Train
Test
SARIMA
250
200
150
100
50
0
1000
2000
3000
4000
5000

Figure 4.5

Chapter 5

Conclusions

5.1 Using Classifiers

We hereby concluded that no trading algorithm can be 100% efficient, not only 100%, it will mostly never be close to 70% but to achieve even an accuracy of 40% or 35% is still good enough to get a good predicted spread. Although our maximum achieved accuracy was 39%, we were still able to closely predict the expected outcome and have matched against the company graph. We can make our prediction more efficient by including large data sets that have millions of entries and could train the machine more efficiently. Different movements of stocks can lead to different raises or lows in the predicted price, we can use these movements to judge whether a company should be traded in or not. No training Data can ever be balanced, hence there are always some imbalance which can be seen in the above data spread, but to still predict close to an outcome will also lead to a good strategy if it has higher than 33% accuracy. We should be, while devising a strategy should always think to always have minimal imbalance while still being above 33% accurate.

We can also conclude that in a stock market, there is possible that some companies might not be correlated at all, but mostly can be correlated, and can help judge movements of stock accordingly, we can scale correlations and see how much in percentages they are correlated.

We can include massive data sets, to increase more efficiency, and in our data set we had nan values in tables because of two simple reasons either a particular company wasn't opened during that time of year, or the data is not readily available in both the cases we replaced the nan values with 0 which is something that we might want to change while developing a trading strategy.

5.2 Using Time Series Forecasting Algorithm

Following time series algorithms have following root mean square errors based upon the error best one is holt winters method and ARIMA

Algorithm	Root Mean Square Error
Simple exponential smoothening	9.53
Holt linear method	36.39
Holt winters method(with seasonality)	8.05
ARIMA(best)	8.04

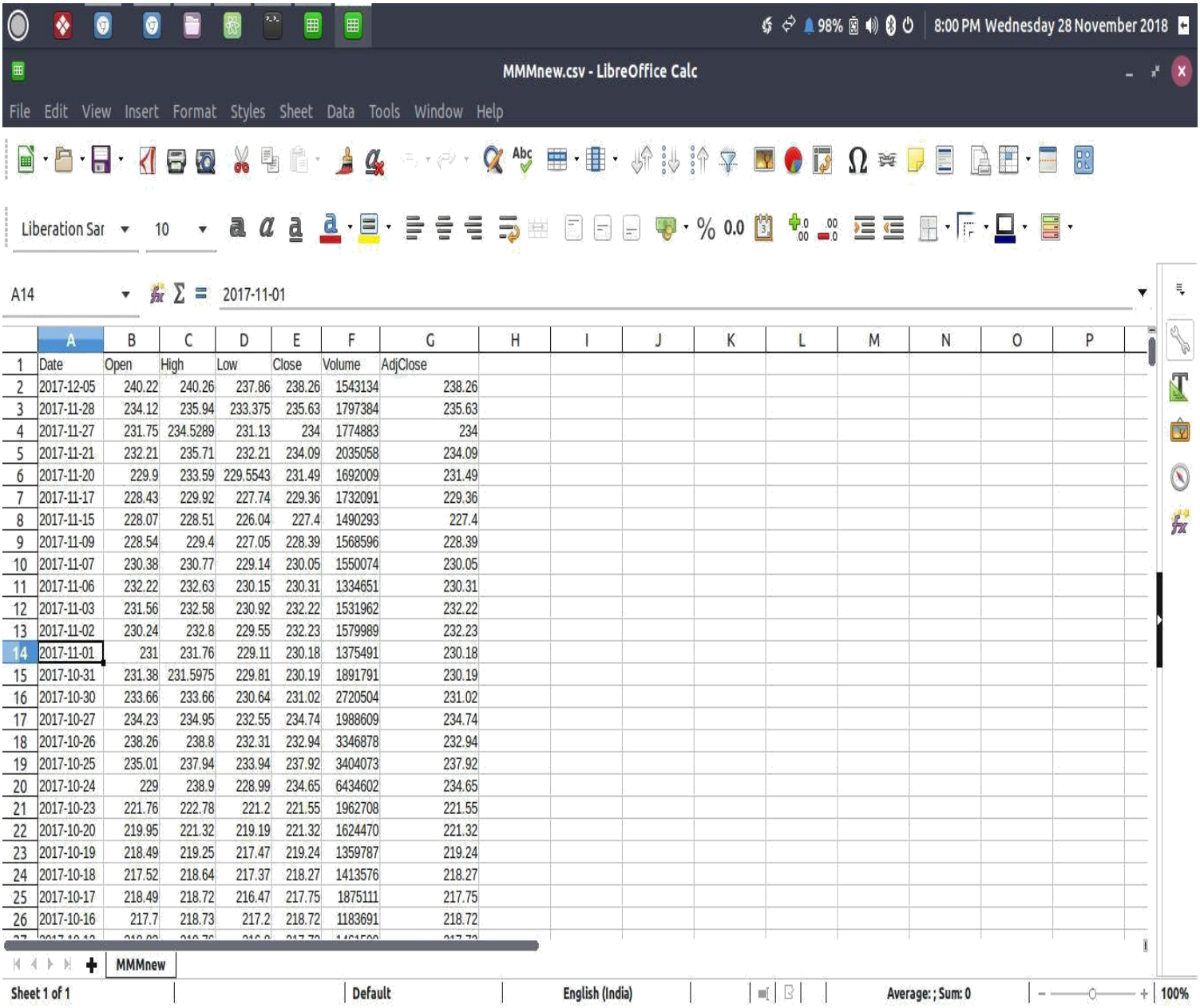
It is Cleary visible above that ARIMA and holt winters method with seasonality is best forecasting algorithms which was discussed and forecasting was implemented

Chapter 6

References

- [1]https://www.researchgate.net/publication/278937384_Automated_Stock_Market_ Trading_Sy stem_using_Machine_Learning [accessed Nov 21 2018])
- [2] https://www.analyticsvidhya.com/blog/2018/02/time-series-forecasting-methods/
- [3] https://www.pythonprogramming.net [methods to parse tickers] [accessed on Dec 2018]
- [4] Fig 3.5.1 - https://www.google.com [accessed on march 2019]
- [5] Fig 3.5.2 https://www.google.com [accessed on march 2019]
- [6] Fig 3.5.3 https://www.google.com [accessed on march 2019]
- [7] https://www.analyticsvidhya.com/blog/2018/02/time-series-forecasting-methods/ [accessed on march 2019]
- [8] https://www.analyticsvidhya.com/blog/2018/02/time-series-forecasting-methods/ [accessed on march 2019]
- [9] https://www.analyticsvidhya.com/blog/2018/02/time-series-forecasting-methods/ [accessed on march 2019]
- [10] https://www.analyticsvidhya.com/blog/2018/02/time-series-forecasting-methods/ [accessed on march 2019]

Chapter 7
Appendix(DataSet SnapShot)



	A	B	C	D	E	F	G
1	Date	Open	High	Low	Close	Volume	AdjClose
2	2017-12-05	240.22	240.26	237.86	238.26	1543134	238.26
3	2017-11-28	234.12	235.94	233.375	235.63	1797384	235.63
4	2017-11-27	231.75	234.5289	231.13	234	1774883	234
5	2017-11-21	232.21	235.71	232.21	234.09	2035058	234.09
6	2017-11-20	229.9	233.59	229.5543	231.49	1692009	231.49
7	2017-11-17	228.43	229.92	227.74	229.36	1732091	229.36
8	2017-11-15	228.07	228.51	226.04	227.4	1490293	227.4
9	2017-11-09	228.54	229.4	227.05	228.39	1568596	228.39
10	2017-11-07	230.38	230.77	229.14	230.05	1550074	230.05
11	2017-11-06	232.22	232.63	230.15	230.31	1334651	230.31
12	2017-11-03	231.56	232.58	230.92	232.22	1531962	232.22
13	2017-11-02	230.24	232.8	229.55	232.23	1579989	232.23
14	2017-11-01	231	231.76	229.11	230.18	1375491	230.18
15	2017-10-31	231.38	231.5975	229.81	230.19	1891791	230.19
16	2017-10-30	233.66	233.66	230.64	231.02	2720504	231.02
17	2017-10-27	234.23	234.95	232.55	234.74	1988609	234.74
18	2017-10-26	238.26	238.8	232.31	232.94	3346878	232.94
19	2017-10-25	235.01	237.94	233.94	237.92	3404073	237.92
20	2017-10-24	229	238.9	228.99	234.65	6434602	234.65
21	2017-10-23	221.76	222.78	221.2	221.55	1962708	221.55
22	2017-10-20	219.95	221.32	219.19	221.32	1624470	221.32
23	2017-10-19	218.49	219.25	217.47	219.24	1359787	219.24
24	2017-10-18	217.52	218.64	217.37	218.27	1413576	218.27
25	2017-10-17	218.49	218.72	216.47	217.75	1875111	217.75
26	2017-10-16	217.7	218.73	217.2	218.72	1183691	218.72

Figure 5