

# STOCK MARKET ANALYSIS THROUGH MACHINE LEARNING ALGORITHMS

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

in

**Computer Science and Engineering**

by

Akshay Sharma(151236)

Vaibhav Singh (151246)

Under the supervision of Mr. Arijit Das

to



Department of Computer Science & Engineering and Information  
Technology

**Jaypee University of Information Technology Wanknaghat, Solan-173234,  
Himachal Pradesh**

# CERTIFICATE

## Candidate's Declaration

I hereby declare that the work presented in this report entitled “Stock Market Analysis through Machine learning algorithms” in partial fulfilment 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, Wazirpur, is an authentic record of my own work carried out over a period from July 2018 to May 2019 under the supervision of Mr. Arijit Das.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Akshay Sharma, 151236

Vaibhav Singh, 151246

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

Mr. Arijit Das

Assistant Professor(Grade-II)

Computer Sciences Department

Dated:

## **Acknowledgement**

We would like to express our deepest appreciation to all those who provided us the possibility to complete this report. A special gratitude we give to our final year project supervisor, Mr. Arijit Das, whose contribution in stimulating suggestions and encouragement, helped me and my partner to coordinate our project well especially in writing this report.

Furthermore we would also like to acknowledge with much appreciation the crucial role of Jaypee University of Information Technology, who gave us the permission to use all required equipment and the necessary materials to complete the project “Stock Market Analysis through Machine learning algorithms ”. A special thanks goes to my supervisor, Mr. Arijit Das, who help me to assemble the parts and gave suggestion about the project, he have invested his full effort in guiding us for achieving the goal. We 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**

Machine learning and its role in the future price determination or stock prediction is being done here. The data is of SNP 500 list which is a list of top companies. Company tickers are extracted and also the respective stock data. We study about various machine learning classifiers and how they are being used in the process.

This field of research is also referred to as Algorithmic Trading.

It strategically focuses on monitoring the market for any changes ie the deviation of the parameter ( the value who's prediction is to be made ) . In case of stock market with the help of proper algorithms we can decide whether to buy, sell or hold a stock.

Various data manipulations are done and the feature sets made were assigned to different labels and the after this classifiers were used to predict the three conditions specified above.

For a layman predicting stock prices is very difficult because not only the dataset is very huge but it is very complex at the same time . A deep knowledge of how these stocks change and move overtime is required. Also the extent of the movement is to be determined ie if a stock moves up the how much is the increase and vice versa.

Also how much change is to be ignored etc. The main question is whether a machine can store and analyse this large extent of data and also devise relationships between various parameters in the dataset .

One basic way of analysing how market and its companies work is to determine how these companies are interrelated. Trick is to find out the correlation between different pairs.

Those companies that are highly correlated are supposed to have a large impact on each other's stock prices while those with less correlation are supposed to be slightly independent of each other. Also this can be used to find out the pair of companies that are not at all related to each other. This is followed by selecting an appropriate feature set and applying ML algorithm to this feature to determine whether to sell or buy, or hold.

# Table of Contents

Title Page

Declaration of the Student Certificate of the Guide .....i

Acknowledgement .....ii

Abstract .....iii

## 1. Introduction

1.1.1 ProblemDefinition .....1

1.1.2 Project Overview andSpecification.....1

1.1.3 HardwareSpecifications .....2

1.1.4 SoftwareSpecifications .....2

## 2.LiteratureSurvey

2.1.1 ExistingSystems .....4

2.1.2 ProposedSystem .....5

2.1.3 FeasibilityStudy .....5

## 3.System Analysis &Design

3.1.1 RequirementAnalysis .....6

3.1.2 Extraction of Data .....6

3.1.3 Manipulation of the Data .....10

3.2.1 Analyzing data in machine learning .....13

3.2.2 Need for preprocessing.....14

|   |           |
|---|-----------|
| 3.2.3 Preprocessing data for MachineLearning .....  | 15        |
| 3.2.4 Visualizing data(for multivariate plots)..... | 19        |
| 3.2.5 Classifiers in machine Learning .....         | 19        |
| 3.2.6 Different classifiers used .....              | 20        |
| 3.3Performing MachineLearning .....                 | 23        |
| <b>4.Results and Outputs .....</b>                  | <b>25</b> |
| <b>5. Conclusions and Recommendations .....</b>     | <b>27</b> |
| <b>6. References .....</b>                          | <b>29</b> |
| List of Figures .....                               | 30        |

# Chapter 1

## Introduction

### **Problem Definition:**

The prediction of the market stock prices is a very absorbing task that requires deep knowledge of the stocks and how they work. It is must to determine what are the various conditions on which the stock prices depend and how largely do they depend on these conditions. Although it is said and argued by various data scientists that the market is self-correcting which means if by any means there is any disturbance or deviation then the market acts on its own to negate or absorb it and thus does not spare any room for any kind of predictions to be made.

Although some might also argue that if a machine is given enough data and is trained well on this dataset then it is possible to make slight predictions and determine a general trend. This is where the concept of HFT (High frequency prediction).

### **Project Overview and Specifications:**

AI (Artificial Intelligence) has become a very important aspect of everyday life applications. A known example is siri, alxa, bixbie which is a virtual assistant created by apple, amazon and Samsung respectively. It is also a very integral part of the financial applications and widely used in stock market prediction. The main aim of our project is to make fast and accurate decisions regarding selling buying or holding a stock.

Machine learning is a subset or a part of Artificial Intelligence and provides us with appropriate algorithms to deal with a large amount of data also referred to as big data.

In other words all the data is fed into a MLA. This model overtime adjusts its parameters and improves its results.

## **Hardware Specifications**

Since it requires us to deal with a large amount of data , we are supposed to have good hardware specifications. Good storage , high RAM and decent GPU is must.

Also a good editor is required, for instance, spyder, visual studio,etc.

## **Software Specifications**

1. Python Idle 3.7:The machine learning algorithms need to be implemented in a editor. Python idle is the editor that we are using. All the tasks like extraction , manipulation and the final predictions are done using the editor only.
2. Numpy Lib: This is one of the python libraries .It has a collection of complex mathematical functions and operations used to deal with big data. It has support for large , multidimensional arrays also.
3. Matplotlib: This is also a library in python and is used for the purpose of plotting as the name suggests. It is an extension of above stated numpy library. It is based on object oriented programing.
4. Pandas: It is a library in python for manipulating and analysing data. All the calculations related to time series and numerical table.



5. BeautifulSoup: It is a python package. The set of XML and HTML documents are parsed using this package. It creates a parsed tree for parsed pages used to extract the data from web pages. This is also called web scraping.
6. Pandas Data Reader: It is a component of pandas library and hence is used to fetch data from internet.

## Chapter 2

### Literature Survey

#### Existing Systems

The task of stock market decision making is very cumbersome and absorbing at the same time. Data scientists have huge pressure on them as a lot of money is on stake. An accurate prediction can lead to large profits and a little mistake can cost the company a large fortune also. Stock market is very complex and intertwined .

Because of the same reason it becomes very tough task for the scientists to design a model that can accurately predict the movement of stock values. The field of computer science that is used in designing such models is (Artificial Intelligence) .

AI models offer us the capability to learn which is otherwise not found in general computers. General computers are designed to take inputs and then deal with these inputs in a prescribed way to give us desired outputs. This is how a general program works. But AI-computers are different in a way that they can learn from the data that they are fed just like humans. Machine learning is a branch of AI that offers various algorithms to extract and learn from the data and also improve its results overtime.

Some of the machine learning techniques that are widely used for making financial predictions are :-

- Naïve byes theorem
- K-NN (k- Nearest neighbours)
- SVM (Support Vector Machine)
- ANN (Artificial Neural Network)
- Random Forest

Usually the scientists go for two kinds of analysis ie.,:

Technical and Fundamental analysis and these both are quite different from each other.

The market position, revenues and profits , growth rate etc are all a part of the fundamental analysis of the company while the technical analysis is mainly related to the

stock price and its fluctuations. Market is very volatile can cause an economy to rise and fall at any time or instant.

All the current systems focus at understanding this volatile behaviour of the market and designing models that can most accurately make predictions

## **Proposed Systems**

The data that we are training our model with is a list of top 500 companies along with their respective financial details. The list is called SNP500 and our objective is to make decisions regarding holding selling or buying a company's stock

## **Feasibility Study**

To check the feasibility of the above model the given output will be checked and matched against the graph of the actual company for that period of time and observe the patterns.

# Chapter 3

## System Analysis & Design

### Requirement Analysis

Predicting future stock prices by studying the past trends in the movement of the stock value is done with the help of technical analyses. We have to keep in mind that the technical parameters don't directly predict or give us the future value but with the help of past inputs and their respective outputs, these technical parameters can determine a general trend in the movement of the stock value so they basically give the investor an idea about whether a stock is going up or is going to go down.

The database in our case i.e., SNP500 has each company's ticker name and corresponding to this ticker name all the other stock details are saved so extraction is done on the basis of these tickers.

The parameter that we have taken into consideration first is the close price of the stock. This is then saved in a data-frame. To get a general idea of the interdependence of the various companies we have also calculated their correlation which is depicted using a heat map.

The next goal is to set a target percentage. This target percentage decides whether a stock is put on hold or we buy more of a stock or even if we need to sell the stock. For this we pre-process the data and establish various other parameters like volume price etc.

## Extraction of the Data

- The first step would be locating the data on the internet. We found the SNP500 list on Wikipedia in a table format.
- This data dates from 1/1/2000 to 31/12/2017 so the model we make gets trained on a certain percentage of this data
- We visit the source code of the Wikipedia page and try to access the .text attribute.
- This attribute is then converted into a SOUP object using the BeautifulSoup library.
- Next step is to get the ticker value for all the companies. As we see in the table, tickers are stored in the 1<sup>st</sup> row of the 0<sup>th</sup> column. SO the corresponding datais extracted using the wiki table sortable classes.
- Now we need to save this data in the form of a list in a pickle file which will update itself by checking for any changes on regular intervals. This is done by taking the .text part and appending this ticker it into a list.
- Now we have a pickle file with the list of tickers of 500 companies stored in this pickle file. This is followed by extracting their corresponding stock details from the table. This is done using a .csv file in which the data of first 15 companies is stored as dealing with a larger data becomes a storage problem.
- This extraction is done using Pandas Data Reader of the Pandas library.

- There are approximately 6000 entries for each company and for all those companies which started between 2000 and 2017 null values before their start are replaced by a zero

So now we have a .csv file that contains the stock data of 15 companies along with their respective tickers This data is ready for manipulation in the next step.

```

1 import bs4 as bs
2 import datetime as dt
3 import matplotlib.pyplot as plt
4 from matplotlib import style
5 import numpy as np
6 import os
7 import pandas as pd
8
9 import pickle
10 import requests
11
12 style.use('ggplot')
13
14 def save_sp500_tickers():
15     resp = requests.get('https://en.wikipedia.org/wiki/List_of_S&#26P_500_companies')
16     soup = bs.BeautifulSoup(resp.text, 'lxml')
17     table = soup.find('table',{'class':'wikitable sortable'})
18     tickers = []
19     for row in table.findAll('tr')[1:]:
20         ticker = row.findAll('td')[1].text
21         tickers.append(ticker)
22
23     with open("sp500tickers.pickle","wb") as f:
24         pickle.dump(tickers, f)
25
26     print(tickers)
27
28     return(tickers)
29
30
31 save_sp500_tickers()
32
33
34 def get_data_from_morningstar(reload_sp500=False):
35     if reload_sp500:
36         tickers = save_sp500_tickers()
37     else:
38         with open("sp500tickers.pickle","rb") as f:
39             tickers = pickle.load(f)
40
41     if not os.path.exists('stock_dfs'):

```

The screenshot shows the Spyder Python IDE interface. The main editor window displays the Python code for fetching and saving stock tickers. The code uses BeautifulSoup to parse a Wikipedia page and save the tickers to a pickle file. The IPython console on the right shows the execution output, including a warning about the parent poll failing and the kernel being left running.

Fig 3.1

```

Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Akshay sharma\Desktop\stock

Editor - C:\Users\Akshay sharma\Desktop\stock\gettingsnp500.py
IPython console
result.py.py Console 1/A Console 2/A

In [13]: runfile('C:/Users/Akshay sharma/Desktop/stock/gettingsnp500.py', wdir='C:/Users/Akshay sharma/Desktop/stock')
[["MM", "ABT", "ABBV", "ABMD", "ACN", "ATVI", "ADBE", "AMD", "AAP", "AES", "AMG", "AFL", "A", "APD", "AKAM", "ALK", "ALB", "ARE", "ALXN", "ALGN", "ALLE", "AGN",
"ADS", "LNT", "ALL", "GOOGL", "GOOG", "NO", "AMZN", "AEE", "AAL", "AEP", "AKX", "ALG", "ANT", "AWK", "AMP", "ABC", "AME", "ANGN", "APH", "APC", "ADI", "ANSS", "ANTH",
"ADM", "AOS", "APA", "AVI", "AAPL", "AMAT", "APTV", "ADM", "ARIC", "AMET", "ADG", "AIZZ", "ATO", "T", "ADSK", "ADP", "AZO", "AVB", "AVY", "BHGE", "BLL", "BAC", "BK",
"BA", "BBT", "BDX", "BRK.B", "BBY", "BIIB", "BLK", "HRB", "BA", "BKNG", "BWA", "BXP", "BSX", "BMY", "AVGO", "BR", "BF.B", "CHRW", "COG", "CDS", "CPB", "COF",
"CPRI", "CAH", "KMX", "CCL", "CAT", "CBOE", "CBRE", "CBS", "CE", "CELG", "CNC", "CNP", "CTL", "CERN", "CF", "SCHW", "CHTR", "CVX", "CMG", "CB", "CHD", "CL", "XEC",
"CTIN", "CTAS", "CSCO", "C", "CFG", "CTXS", "CLX", "CHE", "CMS", "KO", "CTSH", "CL", "CMCSA", "CNA", "CAG", "CXO", "COP", "ED", "STZ", "COO", "CPRT", "GLW", "COST",
"COTY", "CCI", "CSX", "CMI", "CVS", "DHI", "DHR", "DRI", "DVA", "DE", "DAL", "XRAY", "DVN", "FANG", "DLR", "DFS", "DISCA", "DISCK", "DISH", "D", "DLTR", "D", "DOV",
"DOX", "DNDP", "DTE", "DRE", "DUK", "DXC", "ETFC", "EWM", "ETN", "EBAY", "ECL", "EIX", "EW", "EA", "EHR", "ETR", "EOG", "EFX", "EQIX", "EQR", "ESS", "EL", "EVRG",
"ES", "RE", "EXC", "EXPE", "EKPD", "EMR", "XOM", "FFLV", "FB", "FAST", "FRT", "FDX", "FIS", "FITB", "F", "FRC", "FISV", "FLT", "FLIR", "FLS", "FLR", "FMC", "FLI",
"FF", "FTNT", "FTV", "FBHS", "FOXA", "FOX", "BEN", "FCX", "GPS", "GRW", "IT", "GO", "GE", "GIS", "GPI", "GPC", "GLD", "GPI", "GS", "GM", "HAL", "HBI", "HOG", "HRS",
"HIG", "HAS", "HCA", "HCP", "HP", "HSIC", "HSY", "HES", "HPE", "HLT", "HFC", "HOLX", "HD", "HON", "HRL", "HST", "HPO", "HUM", "HBAN", "HII", "IDXX", "INFO", "ITM",
"ILMN", "IR", "INTC", "ICE", "IBM", "INCY", "IP", "IPG", "IFF", "INTU", "ISRG", "IVZ", "IPGP", "IQV", "IRM", "JKHY", "JEC", "JBHT", "JEF", "JSM", "JNJ", "JCI", "JPM",
"JNPR", "KSU", "K", "KEY", "KEYS", "KMB", "KIM", "KMI", "KLAB", "KSS", "KHC", "KR", "LB", "LL", "LH", "LRGX", "LW", "LEG", "LEN", "LLY", "LNC", "LIN", "LKQ", "LMT",
"L", "LOW", "LYB", "MTB", "NAC", "M", "MRO", "MPC", "MAR", "MNC", "MLM", "MAS", "MA", "MAT", "NKC", "MXIM", "MCD", "MCK", "MDT", "MRK", "MET", "MTD", "NGH", "MCHP",
"NU", "MSFT", "MAA", "MK", "TAP", "MDLZ", "MNST", "MCO", "MS", "MOS", "MSI", "MSCI", "MYL", "NDAQ", "NOV", "NTR", "NTAP", "NFLX", "MIL", "NEM", "NMSA", "NUS",
"NEE", "NLSN", "NKE", "NI", "NBL", "JUN", "NSC", "NTRS", "NOC", "NCLH", "NRG", "NUE", "NVDA", "ORLY", "OXY", "OMC", "OKE", "ORCL", "PCAR", "PKG", "PH", "PAYX",
"PYPL", "PWR", "PCT", "PEP", "PKI", "PRGO", "PFE", "P", "PSX", "PXD", "PKI", "RL", "PPG", "PPL", "PFG", "PG", "PGR", "PLD", "PRU", "REG", "PSA", "PDM",
"PVH", "ORV", "PWR", "QCOM", "DGG", "RIF", "RTN", "O", "RHT", "REG", "REGN", "RF", "RSG", "RWD", "RHI", "ROK", "ROL", "ROP", "ROST", "RCL", "CRN", "SBAC", "SLB",
"STX", "SE", "SRE", "SHL", "SPG", "SKKS", "SLG", "SMA", "SO", "LUV", "SPGI", "SKK", "SBUX", "STT", "SYK", "STI", "SIVB", "SYMC", "SYF", "SNPS", "SY", "TROW",
"TTWO", "TPR", "TGT", "TEL", "FTI", "TFX", "TXN", "TXT", "TMO", "T", "TWR", "TJX", "TK", "TSS", "TSCO", "TDG", "TRV", "TRIP", "TSN", "UDR", "ULTA", "USB", "UAA",
"UA", "UNP", "UAL", "UNH", "UPS", "URI", "UTX", "UHS", "UMH", "VFC", "VLO", "VAR", "VTR", "VRSN", "VRSK", "VZ", "VRTX", "VIAB", "V", "VNO", "VMC", "WAB", "WMT",
"WSA", "DIS", "WM", "WAT", "WEC", "WGI", "WFC", "WELL", "WDC", "WU", "WRK", "WY", "WHR", "WMB", "WLTH", "WYNN", "XEL", "XRX", "XLNX", "XYL", "YUM", "ZBH", "ZION",
"ZTS"]

Date MMM ABBT ABBV ... AMD AAP AES AMG
4522 2017-12-22 NaN 56.93 98.21 ... 10.54 100.55 10.71 204.20
4523 2017-12-26 NaN 57.00 97.75 ... 10.46 101.96 10.64 204.06
4524 2017-12-27 NaN 57.47 98.09 ... 10.53 99.77 10.67 204.67
4525 2017-12-28 NaN 57.46 97.79 ... 10.55 99.71 10.76 206.51
4526 2017-12-29 NaN 57.07 96.71 ... 10.28 99.69 10.83 205.25

[5 rows x 12 columns]
MMM ABBT ABBV ... AMD AAP AES AMG

History log
history.py
python gettingsnp500.py
runfile('C:/Users/Akshay sharma/Desktop/stock/gettingsnp500.py', wdir='C:/Users/Akshay sharma/Desktop/stock')
runfile('C:/Users/Akshay sharma/Anaconda3/lib/site-packages/pandas/core/frame.py', wdir='C:/Users/Akshay sharma/Anaconda3/lib/site-packages/pandas/core')
runfile('C:/Users/Akshay sharma/Desktop/stock/result.py', wdir='C:/Users/Akshay sharma/Desktop/stock')

Permissions: RW End-of-lines: LF Encoding: ASCII Line: 7 Column: 20 Memory: 68%

```

Fig 3.2

```

Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Akshay sharma

Editor - C:\Users\Akshay sharma\Desktop\stock\gettingsnp500.py
IPython console
temp.py.py gettingsnp500.py result.py.py Console 1/A Console 2/A

34 def get_data_from_morningstar(reload_sp500=False):
35     if reload_sp500:
36         tickers = save_sp500_tickers()
37     else:
38         with open("sp500tickers.pickle", "rb") as f:
39             tickers = pickle.load(f)
40
41     if not os.path.exists('stock dfs'):
42         os.makedirs('stock dfs')
43
44     start = dt.datetime(2000,1,1)
45     end = dt.datetime(2017,12,31)
46
47     for ticker in tickers[:20]:
48         print(ticker)
49         str1 = "new"
50         ticker_append = ticker + str1
51         if not os.path.exists('stock dfs/{}.csv'.format(ticker_append)):
52             df = web.DataReader(ticker, 'morningstar', start, end)
53             df.to_csv('stock dfs/{}.csv'.format(ticker))
54             df=pd.read_csv('stock dfs/{}.csv'.format(ticker))
55             df_col = ['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'AdjClose']
56             new_df=df[df_col]
57             print(new_df.head())
58             new_df.to_csv('stock dfs/{}.csv'.format(ticker_append), index=False)
59
60     else:
61         dfpd=pd.read_csv('stock dfs/{}.csv'.format(ticker_append))
62         print(df.head())
63         print('Already Have{}'.format(ticker_append))
64
65
66
67
68 get_data_from_morningstar()
69
70
71 def compile_data():
72     with open("sp500tickers.pickle", "rb") as f:
73         tickers = pickle.load(f)
74         print(df.to_csv('stock dfs/{}.csv'.format(ticker)))

Python 3.7.0 (default, Jun 28 2018, 08:04:48) [MSC v. 1912 64 bit (AMD64)]
Type "copyright", "credits" or "license()" for more
information.

IPython 6.5.0 -- An enhanced Interactive Python.
C:\Users\Akshay sharma\Anaconda3\lib\site-packages\ipykernel\parent\poller.py:116: UserWarning: Parent poll failed. If the frontend dies, the kernel may be left running. Please let us know about your system (bitness, Python, etc.) at ipython-dev@scipy.org ipython-dev@scipy.org

In [1]:
In [1]:

Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 67 Column: 1 Memory: 68%

```

Fig 3.3

```

415 ]
      Date MMM   ABT   ABBV   ...   AMD   AAP   AES   AMG
4522 2017-12-22 NaN  56.93 98.21   ...   10.54 100.55 10.71 204.20
4523 2017-12-26 NaN  57.00 97.75   ...   10.46 101.96 10.64 204.06
4524 2017-12-27 NaN  57.47 98.09   ...   10.53 99.77 10.67 204.67
4525 2017-12-28 NaN  57.46 97.79   ...   10.55 99.71 10.76 206.51
4526 2017-12-29 NaN  57.07 96.71   ...   10.28 99.69 10.83 205.25

[5 rows x 12 columns]
      MMM   ABT   ABBV   ...   AAP   AES   AMG
MMM  1.000000 0.955076 0.936969   ...   0.891890 -0.219049 0.848106
ABT  0.955076 1.000000 0.936257   ...   0.900704 -0.176817 0.903079
ABBV 0.936969 0.936257 1.000000   ...   0.259915 -0.072084 0.124418
ABMD 0.871920 0.809805 0.883543   ...   0.719013 -0.006226 0.575262
ACN  0.975235 0.965740 0.889958   ...   0.896404 0.145614 0.833832

[5 rows x 11 columns]

```

```

File Edit Shell Debug Options Window Help
---
000
Symbol      Date      Close    High     Low     Open    Volume
1  MMM  2000-01-03  47.188  48.250  47.032  48.03  2173400
1  MMM  2000-01-04  45.313  47.407  45.313  46.44  2713800
1  MMM  2000-01-05  46.625  48.125  45.563  45.57  3699400
1  MMM  2000-01-06  50.375  51.250  47.157  47.16  5975800
1  MMM  2000-01-07  51.375  51.907  49.969  50.57  4101200
Already HaveMMM
ABT
Symbol      Date      Close    High     Low     Open    Volume
1  ABT  2000-01-03  15.4639 16.1114 15.5520 15.7753 10667432
1  ABT  2000-01-04  15.3003 15.5520 15.1045 15.4121 9315154
1  ABT  2000-01-05  15.1886 15.4361 15.0208 15.0208 11758286
1  ABT  2000-01-06  15.4401 15.7758 15.1324 15.1324 13389878
1  ABT  2000-01-07  16.0555 16.2233 15.4401 15.4401 14315159
Already HaveABT
ABBV
Symbol      Date      Close    High     Low     Open    Volume
1  ABBV 2012-12-10  35.00  37.00  34.91  37.00  749378
1  ABBV 2012-12-11  35.00  35.00  35.00  35.00  0
1  ABBV 2012-12-12  33.36  35.25  33.09  35.25  2530442
1  ABBV 2012-12-13  33.80  34.24  32.61  33.50  4253303
1  ABBV 2012-12-14  33.00  34.08  32.65  33.90  4006673
Already HaveABBV
ABMD
Symbol      Date      Close    High     Low     Open    Volume
1  ABMD 2000-01-03  18.250  18.657  18.250  18.52  185400
1  ABMD 2000-01-04  17.813  18.500  17.000  18.50  34400
1  ABMD 2000-01-05  18.000  18.188  16.938  17.07  122800
1  ABMD 2000-01-06  18.032  18.063  17.625  17.63  84200
1  ABMD 2000-01-07  17.938  18.250  17.563  18.00  69000
Already HaveABMD
ACN
Symbol      Date      Close    High     Low     Open    Volume
1  ACN  2001-07-19  15.17  15.29  15.00  15.10  33703000
1  ACN  2001-07-20  15.01  15.05  14.80  15.05  9238500
1  ACN  2001-07-23  15.00  15.01  14.55  15.00  7501000
1  ACN  2001-07-24  14.86  14.97  14.70  14.95  3537300
1  ACN  2001-07-25  14.95  14.95  14.65  14.70  4208200
Already HaveACN
ATVI
Symbol      Date      Close    High     Low     Open    Volume
1  ATVI 2000-01-03  1.3699  1.3749  1.1664  1.3149  7226760

```

Fig 3.4



## Manipulation of the Data

Now that we have the Open, High, Low and close value for the required companies the next step is to manipulate this data to analyse the relationship between various companies. We put the data in one data-frame and only consider the close values for the time being.

This is followed by making an empty data-frame in which we save the previously stored tickers along with the corresponding close values. We rename this column as per the ticker name and make a sharing data-frame.

First step is to check if the `main_df` is empty or not. If it is then we start with the `current_df` otherwise we use pandas outer-join to combine the data-frames.

An interesting way of visualising the correlation between the close stock values of the various companies would be to make a heat map with is available in `thematplotlib` library.

The mapping style used is “`ggplot`”. There is a function in `matplotlib` which is used to find the correlation b/w data

for e.g. “`dataframe-name.corr()`”

Using this function we are able to calculate the correlation from every column to column. We have the values now, next step is to graph this data and make a heat map. There is no built-in heat map in `matplotlib` but some tools have been provided which help in building one. A numpy array of correlation values is fed to the graph itself and then we build the figure and axes.

This heat map can be made using various different colours the range can also vary depending on the values that we are dealing with. `Cmap` is used to decide the colour scale.

We use `RdYlGn` which describes a heat map that has three colours namely- red, yellow and green. The way this heat map works and the large amount of information that it reflects without using much space is very interesting. The colour goes from red for the lowest values to yellow for the mid-range values and then finally green colour for high correlation values. Also we add a side bar to understand how the colour scale works better. This is followed by adding labels to both “x”-axis and the “y” axis to understand which value belongs to which company.

We tilt the graph to make the graph more readable.

Next step is to set the mathematical range of the heat map. Since we know that the correlation can be positive, zero and negative. Positive means they are directly related to each other and vice-versa. Also zero correlation means that the two companies are not related at all.

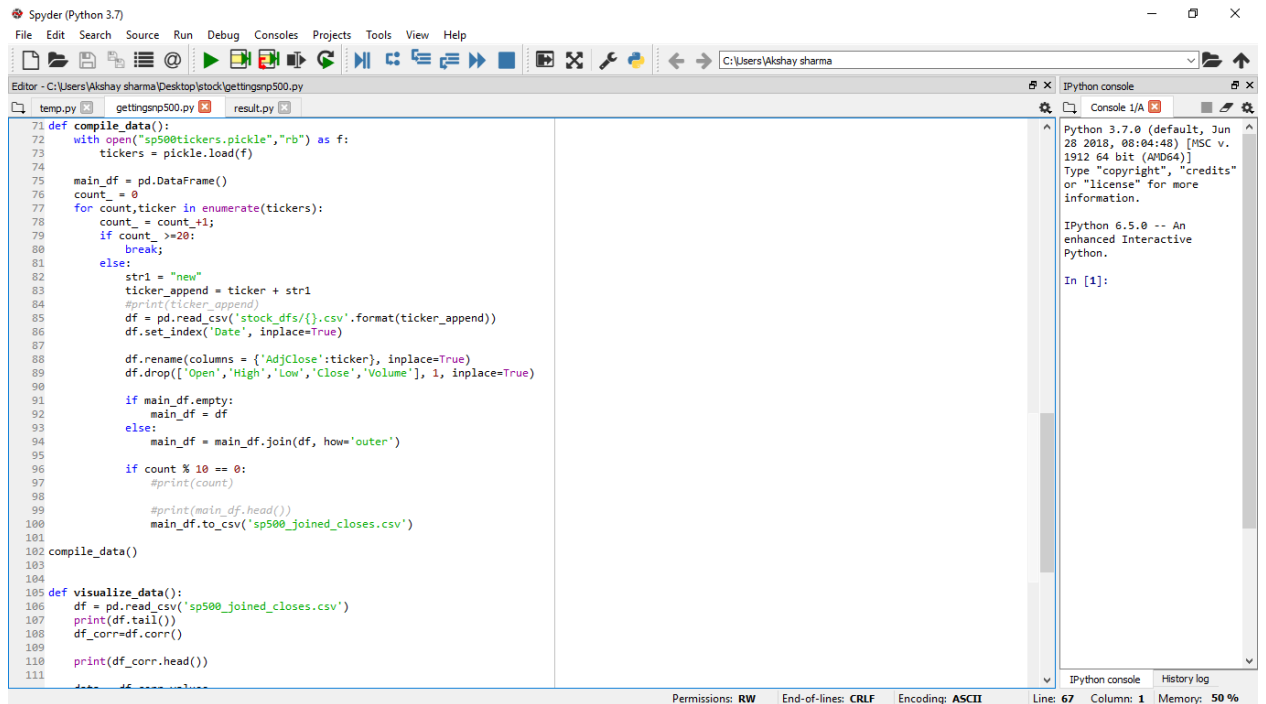


Fig 3.5

|                   |      |            |        |        |        |        |        |        |       |        |  |
|-------------------|------|------------|--------|--------|--------|--------|--------|--------|-------|--------|--|
| 3                 | ALXN | 2000-01-06 | 7.7500 | 7.7500 | 7.2500 | 7.2500 | 800400 |        |       |        |  |
| 4                 | ALXN | 2000-01-07 | 8.2500 | 8.3125 | 7.3750 | 7.5000 | 749200 |        |       |        |  |
| Already Have ALXN |      |            |        |        |        |        |        |        |       |        |  |
|                   |      | Date       | MMM    | ABT    | ABBV   | ...    | AMD    | AAP    | AES   | AET    |  |
| 4690              |      | 2017-12-25 | 234.73 | 56.93  | 98.21  | ...    | 10.54  | 100.55 | 10.71 | 179.96 |  |
| 4691              |      | 2017-12-26 | 235.45 | 57.00  | 97.75  | ...    | 10.46  | 101.96 | 10.64 | 180.42 |  |
| 4692              |      | 2017-12-27 | 236.20 | 57.47  | 98.09  | ...    | 10.53  | 99.77  | 10.67 | 180.85 |  |
| 4693              |      | 2017-12-28 | 235.72 | 57.46  | 97.79  | ...    | 10.55  | 99.71  | 10.76 | 181.23 |  |
| 4694              |      | 2017-12-29 | 235.37 | 57.07  | 96.71  | ...    | 10.28  | 99.69  | 10.83 | 180.39 |  |

Fig 3.6

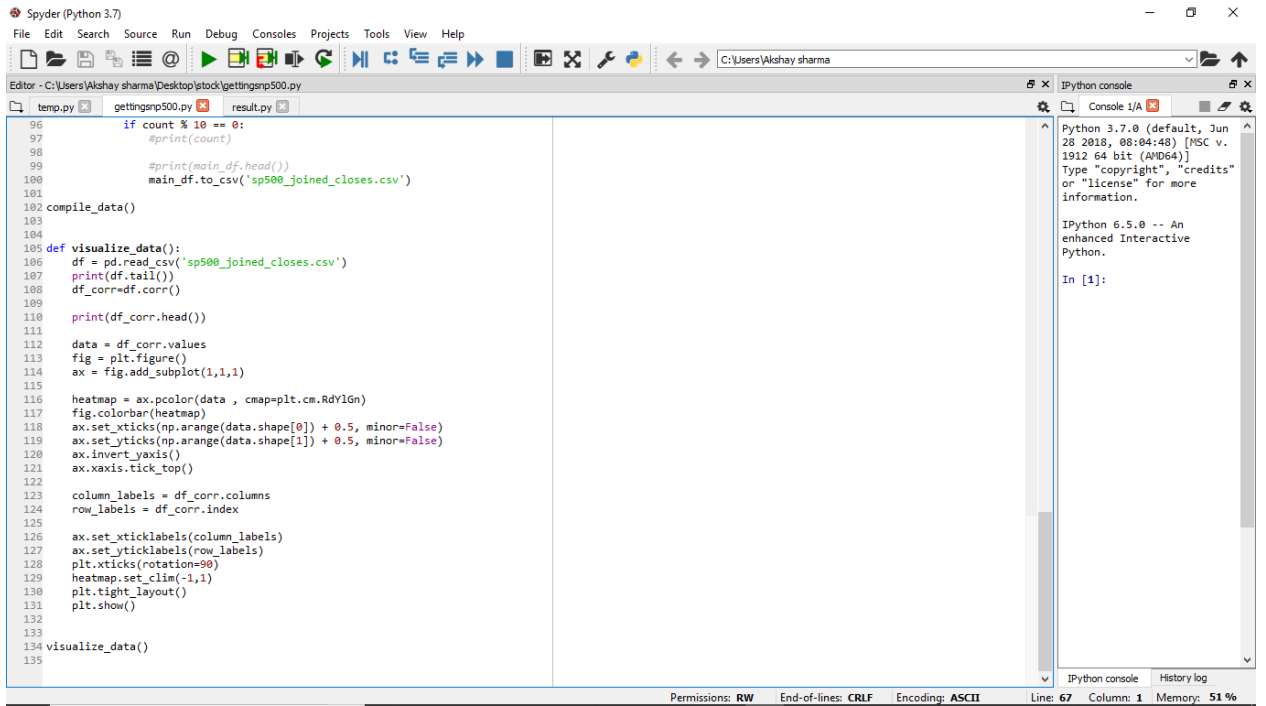


Fig 3.7

```

4690 2017-12-25 234.73 56.93 98.21 ... 10.54 100.55 10.71 179.96
4691 2017-12-26 235.45 57.00 97.75 ... 10.46 101.96 10.64 180.42
4692 2017-12-27 236.20 57.47 98.09 ... 10.53 99.77 10.67 180.85
4693 2017-12-28 235.72 57.46 97.79 ... 10.55 99.71 10.76 181.23
4694 2017-12-29 235.37 57.07 96.71 ... 10.28 99.69 10.83 180.39

[5 rows x 12 columns]
      MMM      ABT      ABBV      ...      AAP      AES      AET
MMM  1.000000  0.925710  0.921983  ...  0.859452 -0.304913  0.965608
ABT  0.925710  1.000000  0.914364  ...  0.879624 -0.217926  0.921008
ABBV 0.921983  0.914364  1.000000  ...  0.289796 -0.253843  0.907987
ABMD 0.866284  0.795963  0.834205  ...  0.715834 -0.074593  0.885329
ACN  0.953973  0.942296  0.839797  ...  0.898378 -0.043713  0.945445

[5 rows x 11 columns]

```

Fig 3.8



Fig 3.9

## Analysing data in machine learning

Now that, we have loaded the dataset using pandas, let's find out more about our data.

- Describing the Dataset  
Using the method describe(), we can find out parameters like count, mean, std, and max.  
[df.describe()]
- Shape of the dataset  
The shape tuple simply, will give us the dimensions of the dataset.  
[df.shape()]
- Extracting the data from the dataset  
Here, according to our requirement, if we want only the first ten rows from the dataset, we can call the head() method on it. To this, we can pass it the argument 10, for printing the first 10 rows from the head.  
[df.head(10)]
- Performing operations around a variable  
We can perform certain operations on a variable. For instance, here, we demonstrate how to group data on a variable. For this, we use the groupby() function.

## Need for Preprocessing data

In Machine Learning, the technique of organizing the data for an ML algorithm consists of the following phases:

- Data Selection
- Data Preprocessing
- Data Transformation

The data format should be in an appropriate manner, for attaining better yield from the model applied in Machine Learning projects. Some predefined Machine Learning model needs data in a prearranged format. For instance, Random Forest classifier doesn't support null values, in this way to implement random forest algorithm, null values must be managed from the original raw data set.

Data selection comprises of numerous stages as subsequent:

There is an immense amount of variety, quickness, and size of the accessible information for a Machine Learning problem. In this step, we only select a subset of the figures that we are available with.

The chosen section must be a precise and significant representation of the full sample. A few data can be replicated or derived from the available information whenever essential. Information not applicable to the issue in hand can be evaded.

After the data selection procedure, the data needs to be preprocessed/cleaned out using following steps:

The data should be formatted in order to make it suitable for ML. Then, cleaning of the data is done to remove incomplete/null variables. Data cleaning/refining involves filtering the data, based on a no. of variables, as follows :

#### Insufficient Data:

The extent of information essential for ML algorithms can shift starting from thousands to millions, based on the complex nature of the issue and the chosen algorithm.

#### Non-Representative Data:

The instance that is selected must be a precise depiction of the whole data information, as non-representative info from the data may train an algorithm such that it won't function actually well on fresh test information.

#### Substandard Data:

Outliers, errors, and noise can be eradicated to acquire a better fitting of the selected model. Absent features such as age, for 15 percent of the spectators may be ignored

entirely, or an average value can be presumed for the missing constituent.

Selecting the correct size of the sample is a key step in data preparation. Samples that are too big or too minor, might give skewed results.

Smaller samples result in sampling noise since they get prepared on non-representative info. For instance, testing the voter sentiments from an tremendously little subset of voters.

Bigger examples function splendidly, if there is no sampling bias, that is, then the correct information is picked. For instance, sampling bias would happen while checking voter sentiment just for the technically sound subset of voters, while overseeing others.

## **Preprocessing the Data to prepare for Machine Learning**

In Machine learning, ML algorithms don't toil very good with the treating of raw/unprocessed data. And, beforehand we can feed this raw data into a machine learning algorithm, we must preprocess this information, or in other arguments, we must put on some transformations on it. By carrying out data preprocessing, we transform the raw data into a suitable and clean data set because, whenever the data is collected from numerous sources, it is collected in a raw layout, which is not conceivable for the analysis. Some ML models need information to be in a definite format.

The task here is to see what occurs, if data from all of the companies is taken, and fed through an ML classifier. Over time, diverse companies have different relationships with each other, so if the mechanism can identify and fit these relations, it's possible this could calculate from changes in prices at present, what will occur tomorrow with a particular company.

To initiate with, the foremost job of machine learning is to take "feature sets" and tries to map them to "labels." Whether it is K- Nearest Neighbors or deep learning with the neural networks, the manner remains the same for all. Thus, the chief task is to change

our existing data to feature sets and labels . In general,Pre-processing can be mentioned to as the transformations applied to our facts,beforehand feeding it into the ML algorithm.

Features can also be the other firm's rates, but in its place, say that the features are the pricing variations for the day for all of the companies. And, the label will be whether or not we essentially want to buy a particular corporation. Consider a company for an illustration, for the feature sets, we take into reason all company's %age changes for that day, and those will be the features. Label will be whether or not,that company rose more than x percent within the next x days, where it is conceivable to pick whatsoever is you want for x. To start, let's say that a company is a buy if, inside the following 7 days, its value goes up more than 2 percent and it is a sell if the price goes down more than 2 percent.

If the procedure says buy, then we perform buy, place a 2 percent fall stop-loss (essentially something that tells the interchange is price decreases below this number / or goes overhead if there is shorting of the firm, then exit the position). Otherwise, sell the company once it has risen 2 percent, or could be conservative and sell at 1 percent growth etc. Irrespective of this fact, this could reasonably easily build an approach from this classifier. In order to begin, the prices into future for our training statistics is needed.

This function will take a single parameter, the particular ticker being referred to. Each model will be organized on a single company. Next, know how stretched into the future the prices are required for. Here we are considering for 7 days. Now, read in the info for the close prices for all organizations that have been kept previously, take a list of all the present tickers, and initiate filling any misplaced with 0 for the time being. This may be something someone need to modify later on, yet how about we go with 0 for now. Now, take the % change values for the following 7 days.



This generates a new data frame columns for a particular ticker in question, using string formatting to generate the custom names. This way, for receiving future values is with shift, which mainly will shift a column up or down. Here, shift a negative quantity, which will take that column and, if somebody could tell it visually, it would shift that column up by  $i$  rows. This gives the future prices  $i$  days in advance, which can be calculated for % age variation.

Create a function, that generates our label. There are a multiple of choices. There is a function that dictates buy/ sell/hold, or maybe just buy/sell, therefore using the previous spread.

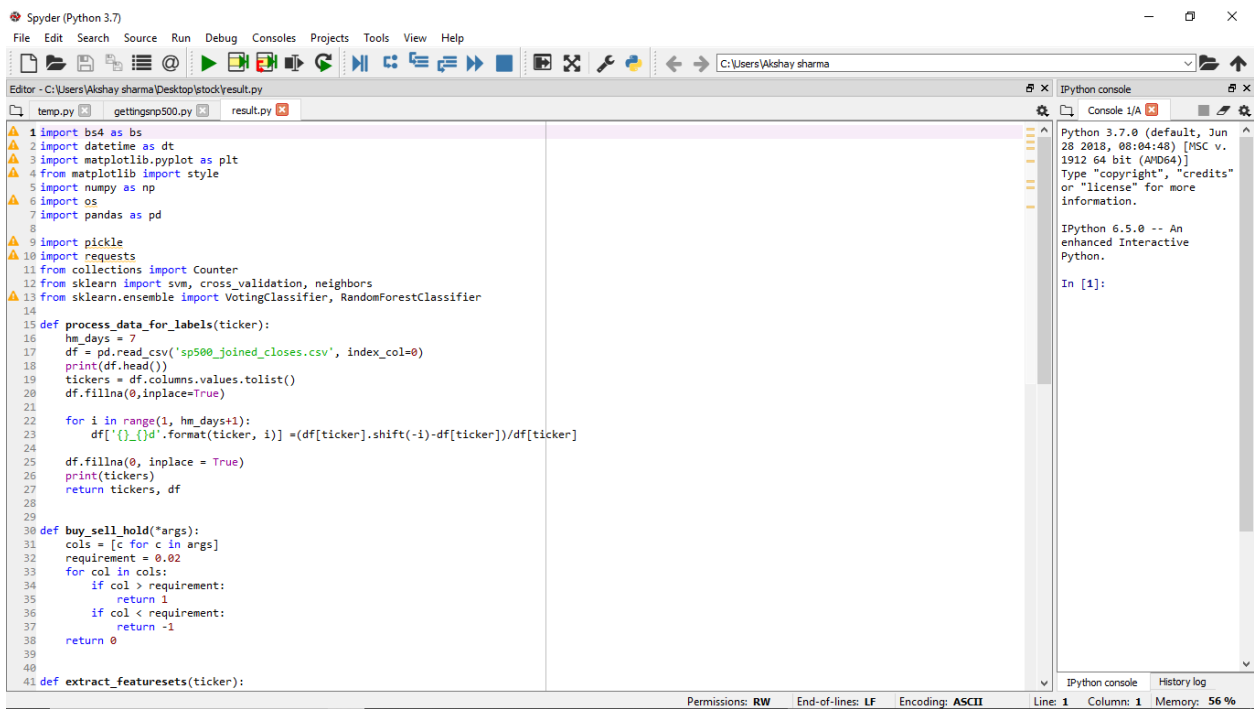
Principally, if the price increases more than 2 percent in the next 1 week, forecast that it's a buy. If it drops more than 2% in the following week, that is a sell. If it doesn't do either of those, then it is moving insufficiently, and this would expect to just hold whatever our position is. If there are shares in that firm, do nothing, but keep the current place. If there aren't any shares in the enterprise, do nothing, just hold

Used args here, so to take any no. of columns here that are required. The concept here is that to plot this function to a Pandas-DataFrame column, & the column would act as "label." If it is -1, it's a sell, 0 is a hold, and a 1 is buy. The "\*args" will be those upcoming price change columns, and if there is movement that exceeds 2 percent in either direction that's what dealers should be concerned in. This is not a completely faultless function. For example, price might go up by two percent, & then fall 2 percent, and it mightn't be ready for that, but it will do just well still.

There is some fully absent information, which we will exchange with 0. Then it possibly have some of infinite data, specially if in the algorithm there is a percent variation from 0 to anything. So transform infinite values to NaNs, then drop NaNs. Presently, the "features" are that day's values for stocks. Just static numbers, really specifying nothing at all.

In its place, a superior metric would be every single company's % change that day. The idea here being that some companies will change in price before others, and we can profit maybe on the laggards. So convert the stock values to percentfluctuations.

The capital X contains the feature sets (daily percentage variations for every firm in the S&P 500). The lowercase y is the "target" or the "label." Basically, those which are going to map our feature sets to.



```
1 import bs4 as bs
2 import datetime as dt
3 import matplotlib.pyplot as plt
4 from matplotlib import style
5 import numpy as np
6 import os
7 import pandas as pd
8
9 import pickle
10 import requests
11 from collections import Counter
12 from sklearn import svm, cross_validation, neighbors
13 from sklearn.ensemble import VotingClassifier, RandomForestClassifier
14
15 def process_data_for_labels(ticker):
16     hm_days = 7
17     df = pd.read_csv('sp500_joined_closes.csv', index_col=0)
18     print(df.head())
19     tickers = df.columns.values.tolist()
20     df.fillna(0, inplace=True)
21
22     for i in range(1, hm_days+1):
23         df['{}_{}_d'.format(ticker, i)] = (df[ticker].shift(-i)-df[ticker])/df[ticker]
24
25     df.fillna(0, inplace = True)
26     print(tickers)
27     return tickers, df
28
29
30 def buy_sell_hold(*args):
31     cols = [c for c in args]
32     requirement = 0.02
33     for col in cols:
34         if col > requirement:
35             return 1
36         if col < -requirement:
37             return -1
38     return 0
39
40
41 def extract_featuresets(ticker):
```

Fig 3.10

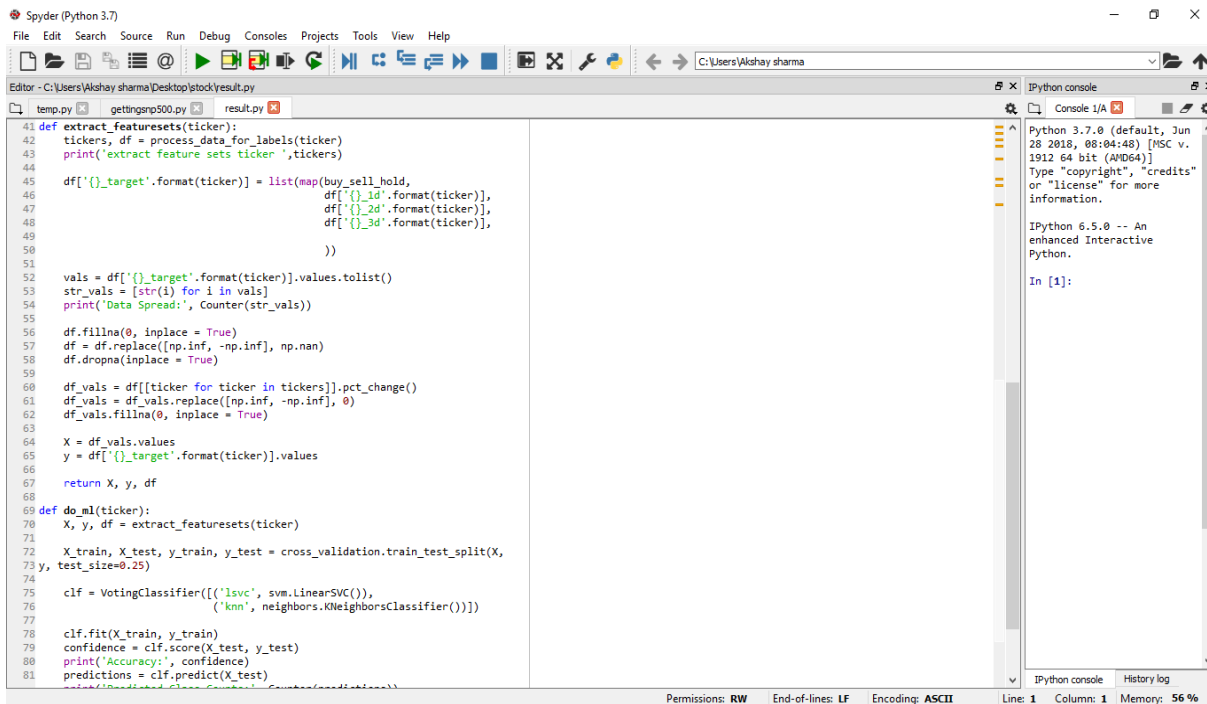


Fig 3.11

## Visualising data- Multivariate plots

A multivariate analysis examines more than two variables. For two variables, we call it bivariate.

Correlation Matrix Plot:

A correlation matrix plot demonstrates how changes between the two variables relate.

The two variables that change in the same direction are positively correlated. A change in opposite direction implies negative correlation between the variables.

- Correlations=df.corr()
- fig=plt.figure()

## **Classifiers in Machine Learning**

Classification can be classified as a major process that includes predicting the class of given data sets for prediction of different real time applications, classes are also sometimes referred to or known as “ targets” or “labels” or “categories”. Classification includes predictive Modelling, which is the job of approximating a given mapping function (f), via the input variables (X) to distinct yield variables(y). Ponder over an example, spam recognition in our email services suppliers can be recognized to as a classification problem, it comprises binary classification, as there are only two classes of spam and not-spam. A classifier will exploit certain training information in a particular %age so as to comprehend how the given input variables will relate to class. In this particular case, that is spam/ non-spam, e-mails have to be used to provide training information to the machine. When this classifier will be trained correctly it can be utilized to detect an unknown e- mail.

Classification generally belongs to the class of supervised machine learning, where the targets/ labels/ categories are also provided with the input data. There can be many uses in classification in various different domains, such as in stock market, recommender systems, target marketing etc.

### **Different Types Of Classifiers Used**

There are basically 2 types of learners in classification that are classified as lazy learners and eager learners.

#### **1.Lazy learners**

Lazy learners have a simple task to store the training data and wait until the testing data appears. When the testing data appears, classification is conducted that is based on the most of the associated information there is in the stored training data.As compared to the eager learners, the lazy learners have very less training time but has more time in forecasting.

Examples: k-nearest neighbor(knn), Case-based reasoning

## 2. Eager learners

Eager learners usually build a classification model that is based on the specified training data before getting the information for classification. It should be capable to obligate to a single hypothesis that will cover the entire instance space there is. As a result of this model building, the eager learners usually take an extended time for train and very little time to forecast.

## **Decision Tree, Naive Bayes, k-Nearest Neighbors**

### **K-nearest neighbors**

The k-nearest neighbors (k-nn) algorithm, also known as k-NN is a non-parametric process or approach that is utilized for classification & regression. In both the circumstances of the classification and regression, the input comprises of the k-closest training samples that are present in the feature space. And the output majorly depends on whether or not, the k-NN class is utilized for the classification or it is being used for regression.

In k-NN when it is used for classification, the output is generally a class association. An object is classified if there is a majority of vote from its neighbors, to it and then the object being allocated to the class that is most mutual among whole of its k-nearest neighbors, where k is a +ve integer characteristically minor. If  $k = 1$ , then object is purely allocated to the class of its single-nearest neighbor.

In k-NN, when it utilized for regression, the output is usually the property rate for the standardized object. This value contains the average of all the values of its k-nearest neighbors.

k-NN or the k nearest neighbor majorly is a type of instance based learning, or comes under a class of lazy learning, in which the function is only approximated locally and all other calculation is delayed until the time of the process of classification. The k nearest neighbor algorithm is the simplest among all of the ML algorithms.

For the both of the process of classification & regression, it is a very muchvaluablemethod that can be utilized to allot some of the weight to the offerings that is of the neighbors, so that the neighbors that are nearest additional to the calculated average than the more that are the distant ones. Consider an example, of a common weighting scheme that consists of providing each neighbor, a weight of some functions that contains distance like  $1/d$ , where  $d$  is the distance to the specified neighbor.

Instance of the k-NN classification: The test sample (green dot) should be classified either to the first class of blue squares or to the second class of red triangles. If  $k = 3$  (solid line circle) it is allocated to the second class because there are 2 triangles and only 1 square inside the inner circle. If  $k = 5$  ( the outer dashed line circle) it is allocated to the 1st class i.e., 3 squares vs. 2 triangles inside the outer circle.

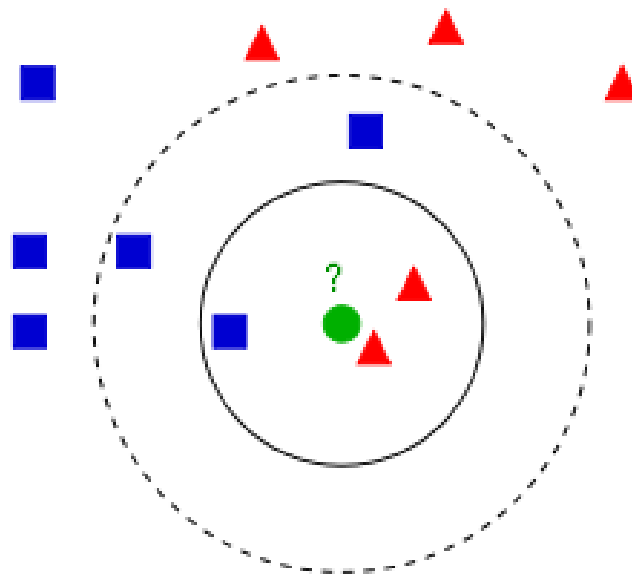


Fig 3.12

## Random Forest Classifier

“Random forests” also known as the “Random decision forests” is a tool to ensemble learning technique for classification, regression and other various tasks, that typically operate by building a multiple of decision trees at training period and outputting the class that is the way of the classes for classification or average calculation that is, regression, of the individual trees.

Random decision forests correct for decision trees' habit of overfitting to their training set.

The ever first algorithm created for random decision forests working was created by Tin Kam Ho using the method of random subspace, which according to Ho's construction, is an approach to implement the famous "stochastic discrimination" style, that is used for classification proposed by Eugene Kleinberg.

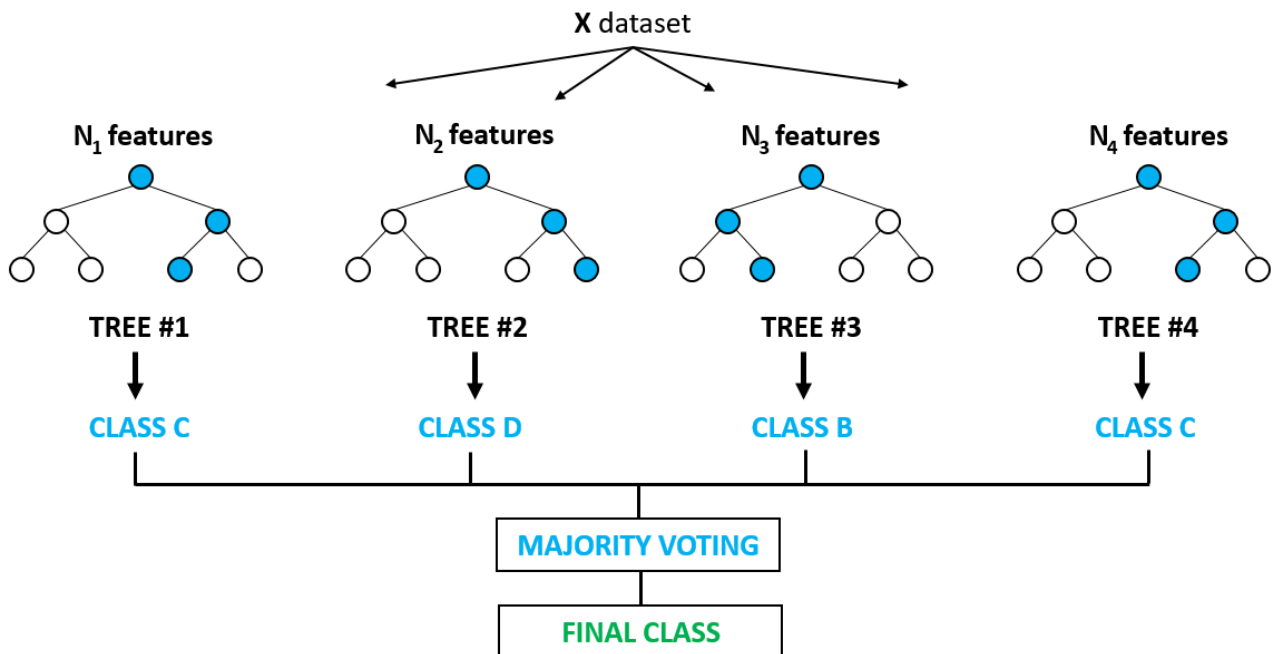


Fig 3.13

Leo Breiman and Adele Cutler developed a later version of the algorithm, and "Random Forests" is their trademark. The extension has arbitrary selection of the features, presented by Ho, later on self-sufficiently introduced by Amit and Geman in order to build a collection of decision trees with measured alteration.

## **Support Vector Machines**

Support vector machines (SVM's), also known as "Support vector networks" are the learning models that fall in the class of supervised learning models, with different related learning algorithms that are used to examine information that is used for the classification & regression analysis. If we have a set of training instances, in which each example is marked as belonging to one or another of the two classifications, then an SVM training algorithm constructs a model that allocates new samples to either of the category, which makes it a non-probabilistic binary linear classifier although some methods like Platt scaling exist that makes it a probabilistic approach. SVM model represents the instances in the form of the points in a space which are mapped so that the specimens of distinct classes are divided by a clear wide gap that is as much wide, as it can be possible. Further then, more new instances are mapped into that equivalent space and they are forecasted if they fit to a category based on the fact that which side of the gap they will fall in.

In addition SVMs can competently carry out a non-linear classification using the kernel trick and implicitly mapping their inputs into the high-dimensional feature spaces.

When the information that is present is unlabeled, then in that case supervised learning is unlikely to work and has no effect, and hence we require an unsupervised learning approach, it attempts to find a natural clustering of the data that is then grouped, and then it will map this fresh information to these earlier formed groups.

The clustering algorithm for support vectors, was firstly created by Hava Siegelmann



and Vladimir Vapnik, applied the whole statistics of support vectors that was established in the SVM algorithm, which helped to categorize unlabelled information, and it is one of the most extensively used clustering algorithms among all industrial applications.

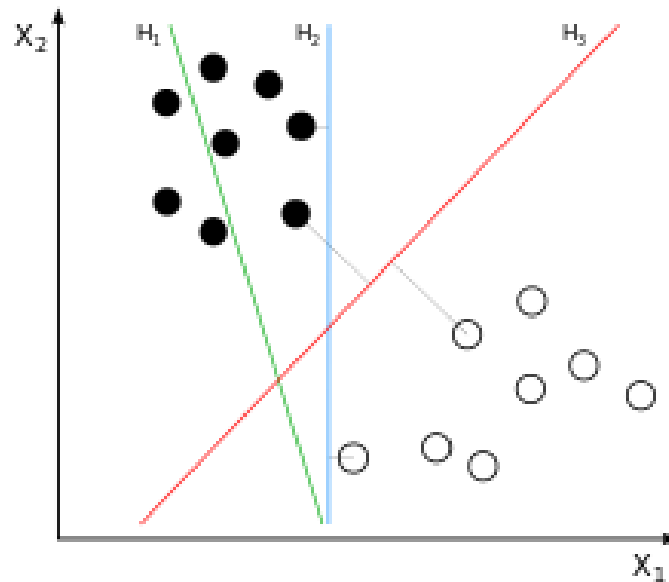


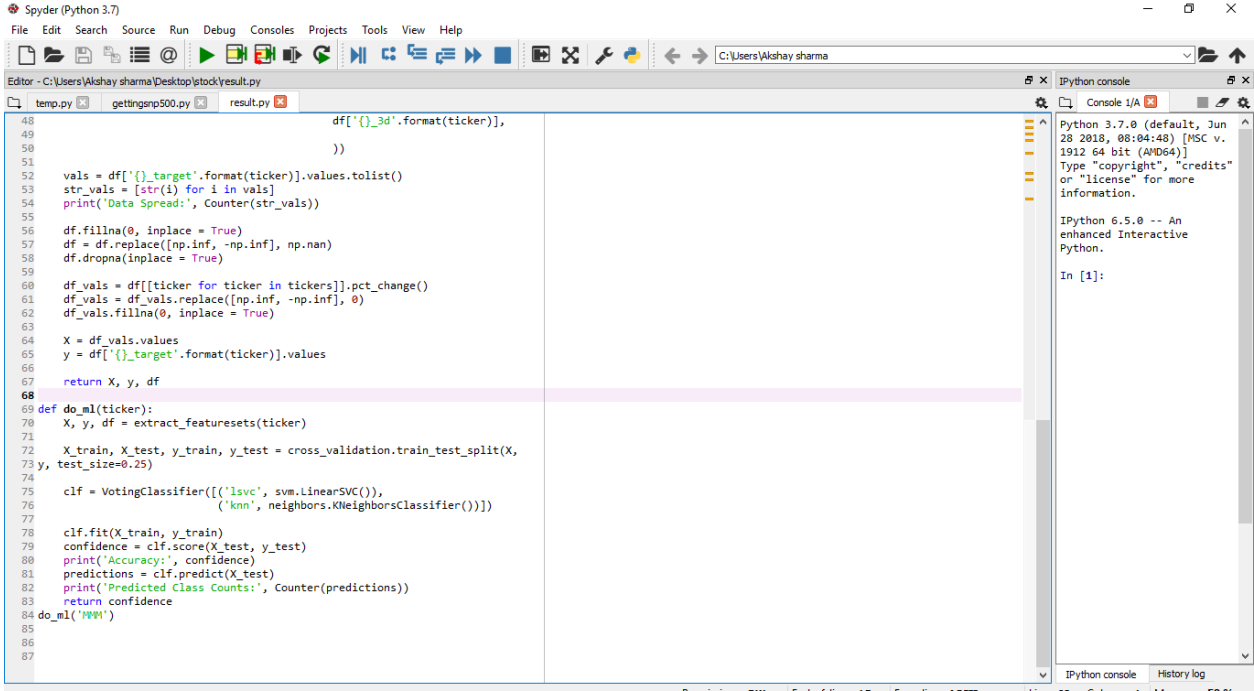
Fig3.14

Here,  $H_1$  does not isolate the classes.  $H_2$  does, but only with a minor margin.  $H_3$  splits them with the maximal margin.

## Performing Machine Learning

Sklearn is a machine learning framework. The svm is being imported for a Support Vector Machine classifier, cross\_validation will easily let us create shuffled training and testing samples for classification. The neighbours that is imported is for K Nearest Neighbors Algorithm. Then, there is also a Voting Classifier and RandomForest-Classifier. The voting classifier is just what it sounds like, basically, it's a classifier that will combine many classifiers, and allow them to each get a "vote" on what they think the class of the feature sets is. The random forest classifier is just another classifier. Used these three classifiers in the voting classifier. Now, the feature sets and labels are available, and it's time to shuffle them, train, and then test.

What this does, is shuffle our data (so its not in any specific order any more), and then create training and testing samples for prediction. Don't try to "test" this algorithm on the same data that is being trained against. If that happens, chances are it would do a lot



```
48         df['{}_3d'.format(ticker)],
49     ))
50
51
52     vals = df['{}_target'.format(ticker)].values.tolist()
53     str_vals = [str(i) for i in vals]
54     print('Data Spread:', Counter(str_vals))
55
56     df.fillna(0, inplace = True)
57     df = df.replace([np.inf, -np.inf], np.nan)
58     df.dropna(inplace = True)
59
60     df_vals = df[[ticker for ticker in tickers]].pct_change()
61     df_vals = df_vals.replace([np.inf, -np.inf], 0)
62     df_vals.fillna(0, inplace = True)
63
64     X = df_vals.values
65     y = df['{}_target'.format(ticker)].values
66
67     return X, y, df
68
69 def do_ml(ticker):
70     X, y, df = extract_featuresets(ticker)
71
72     X_train, X_test, y_train, y_test = cross_validation.train_test_split(X,
73 y,
74 test_size=0.25)
75
76     clf = VotingClassifier([('lsvc', svm.LinearSVC()),
77 ('knn', neighbors.KNeighborsClassifier())])
78
79     clf.fit(X_train, y_train)
80     confidence = clf.score(X_test, y_test)
81     print('Accuracy:', confidence)
82     predictions = clf.predict(X_test)
83     print('Predicted Class Counts:', Counter(predictions))
84     return confidence
85
86 do_ml('MMM')
```

Python 3.7.0 (default, Jun 28 2018, 08:04:48) [MSC v.1912 64 bit (AMD64)]  
Type "copyright", "credits" or "license" for more information.

IPython 6.5.0 -- An enhanced Interactive Python.

In [1]:

Permissions: RW End-of-lines: LF Encoding: ASCII Line: 68 Column: 1 Memory: 59 %

Fig 3.15

better than it would be in reality. Hence, test the algorithm on data that it has never seen before to see if there is actually a model that works.

The line `clf.fit()` will take the X data, and fit to the y data, for each of the pairs of X's and y's that are present. Once that's done, it can be tested.

If this model is indeed successful, this can be saved with pickle, and can be loaded at any time to feed it some feature sets and get a prediction out of it, with `clf.predict`, which will predict either a single value from a single feature set, or a list of values from a list of feature sets.



Fig 3.16

## Chapter 4

### Results and Outputs

It is noticeable from the above work that, it displays the forecasted counter spread of the company's upcoming prices, and another figure shows the graph of the company at that particular time of year in terms to the prediction. The output shows whether a stock is in condition of Buy, Hold or sell. It can be understood that the data spread is generally saying buy the stock, it can be incorrect on the hold state bcoz the training data will never be flawlessly balanced forever, so apparently if the model predicted buy, then this would be somehow correct, which is still worthy and actually a improved cut than it attained, & it still is getting the above accurateness mark of 33 percent, which is almost very good in a stock market. Many situations will still be there which machineries can slip out, supposedly this has circumstances to buy, sell or hold, and occasionally the model can be penalized, at a guess the model anticipated a 2 percent growth in the next seven days, but the rise only went up to 1.5 percent and went 2 percent the following day, then the model will forecast buy or hold, conferring to the 1.5 percent rise in the seven days & give the likely spread. A model can also be penalised if supposedly the rise went 2% up and then unexpectedly drops 2 percent low for the next day, this kind of consequences in real trading woul'd be thoughtful, and the same goes for the model, if it turns to be extremely correct. Now, observing the spread and graph of the company notice that around the period of the year 2017, the company was growing in the market. So, there were truly more buys, which swiftly fell in the year of 2018, but the data we extracted in the beginning was till 31 Dec, 2017 and it shows that at the beginning of the year, it had a lot of buys, henceforth 1722 out of 4527 which quickly was traded just in a short time. Hence a lot of sells more than the holds, the model might not be flawlessly accurate, but has a very close range of judgments, which can be recognized in the real trading or using algorithms to trade.

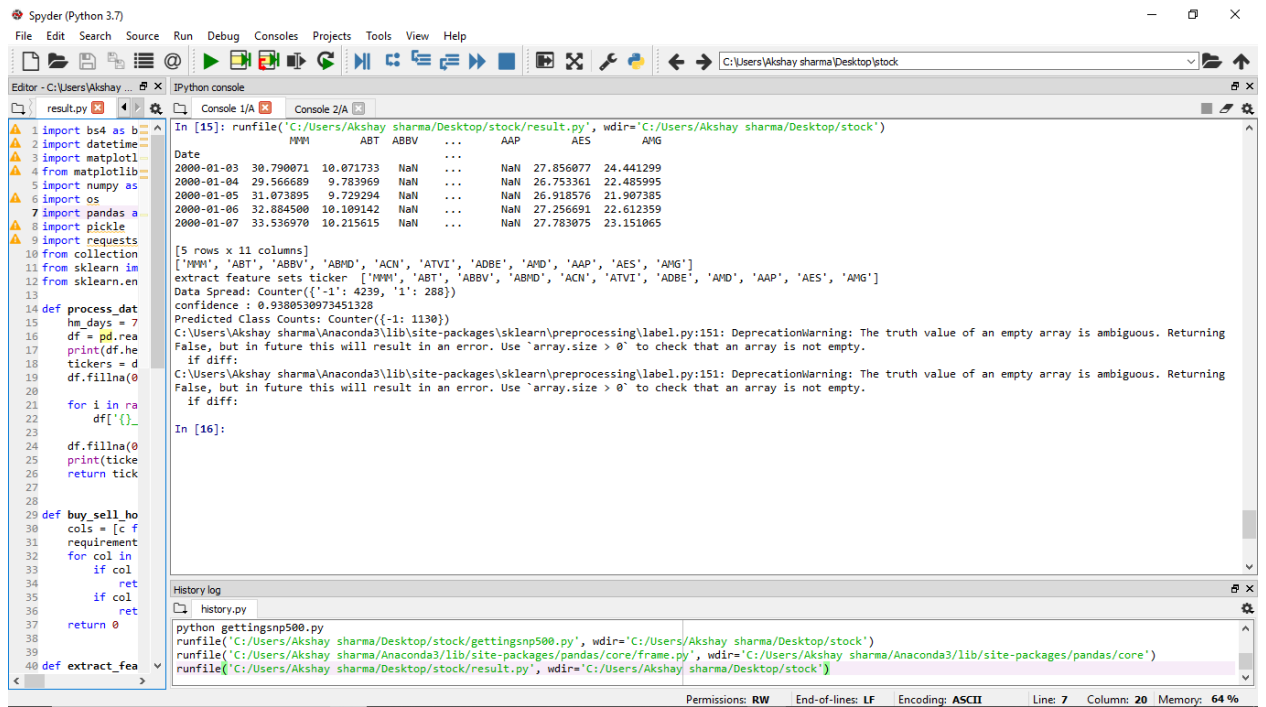


Fig 3.17

```

[5 rows x 11 columns]
['MMM', 'ABT', 'ABBV', 'ABMD', 'ACN', 'ATVI', 'ADBE', 'AMD', 'AAP', 'AES', 'AMG']
extract feature sets ticker ['MMM', 'ABT', 'ABBV', 'ABMD', 'ACN', 'ATVI', 'ADBE', 'AMD', 'AAP', 'AES', 'AMG']
Data Spread: Counter({'-1': 4239, '1': 288})
confidence : 0.9380530973451328
Predicted Class Counts: Counter({'-1': 1130})

```

Fig 3.18

## Chapter 5

### Conclusions and Recommendation

To make our prediction more efficient, it can be done 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, 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.

Thus, here it can be concluded that none of the machine learning trading algorithms can be hundred percent efficient, and not only 100%, it will rarely be close to about 70%, but to achieve even an accuracy of 40% or 35% is still good enough to get a good predicted spread. Although maximum achieved accuracy was 39%, it was still able to closely predict the expected outcome and have matched against the company graph.

It can also be determined that in a stock market, in the context of the correlation of companies with each other, there's a possibility that some firms might not be correlated at all, and generally can be correlated to one another, & can help determine and analyze the movements of stock consequently, we can scale the correlations and understand that as to how much in %ages they're correlated.

Including enormous information sets, to escalate more efficiency, and in data set if had non assigned or absent values in tables, because of two modest explanations either a particular enterprise was not released during that time of the year, or the information is not eagerly accessible, in both the circumstances replace the unassigned or absent values with 0, which is somewhat, dealer might need to modify while forming a trading approach.

Hereby, we can say, concluding from the above work and the research involved in it, that no trading algorithm can be 100% proficient. Most of the trading algorithms will generally never be close to even 70 percent. Though, supreme achieved accuracy was 39 percent, it was still able to closely foresee the anticipated result. To make our forecast more effective, it can be done by including huge information sets having around millions and millions of items and could train the machine more capably

Various developments of stocks can prompt various raises or lows in the anticipated value, utilize these developments to pass judgment on whether an organization ought to be traded or not. No training Data can ever be balanced, consequently there is in every case some imbalance which can be found in the above information spread, yet to in any case still predict closer to a result will likewise prompt a decent system on the off chance that it has higher than 33% precision. While, contriving a system merchant ought to dependably think to dependably have negligible unevenness while as yet being above 33% precise.

Also, it can be concluded that in a stock marketplace, there is a possibility that some enterprises might not be correlated at all, and mostly can be correlated to each other, and can help judge movements of stock consequently, we can scale correlations and see how much in %ages they are correlated.

Including gigantic information sets, to build more effectiveness, and in data sets whenever had not assigned or absent values in tables, due to two straightforward reasons either a specific organization wasn't opened amid that time of the year, or the information isn't promptly accessible, in both the cases replace the not assigned values with 0, which is something that trader should need to change while building up a trading methodology.

# Chapter 6

## References

- <https://medium.com/analytics-vidhya/using-machine-learning-to-predict-stock-prices-c4d0b23b029a>
- <https://www.simplilearn.com/data-preprocessing-tutorial>
- A machine learning based stock trading framework using technical and economic analysis Smarth Behl (smarth), Kiran Tondehal (kirantl), Naveed Zaman (naveedz)
- <http://cs229.stanford.edu/proj2017/final-reports/5234854>
- <https://data-flair.training/blogs/python-ml-data-preprocessing/>
- <https://ieeexplore.ieee.org/document/8212715>
- <https://medium.com/analytics-vidhya/using-machine-learning-to-predict-stock-prices-c4d0b23b029a>
- International Journal of Pure and Applied Mathematics: MACHINE LEARNING APPROACH IN STOCK MARKET PREDICTION  
Raut Sushrut Deepak , Shinde Isha Uday ,Dr. D. Malathi ,2 B.Tech Student, Professor  
Department of Computer Science and Engineering SRM University, Kattankulathur,  
Chennai.
- “Textual analysis of stock market prediction using financial news articles”, by Robert P. Schumaker and Hsinchun Chen February 2009.
- “Intelligent Stock Trading System based on SVM Algorithm” by Qinghua Wen, Zehong Yang, Yixu Song, Peifa Jia



- International Research Journal of Engineering and Technology (IRJET)  
“Stock Market Prediction Using Machine Learning”- V Kranthi Sai Reddy.
- “Using AI to Make Predictions on Stock Market”: Alice Zheng Stanford University  
Stanford, Jack Jin Stanford University Stanford.

## List of Figures

- Figure 3.1: Code for getting the company tickers
- Figure 3.2: Output Of the above collected 500 tickers
- Figure 3.3: To get stock data from Morningstar.
- Figure 3.4: Output Stock data of the companies.
- Figure 3.5: Compile all close index of company in one dataframe.
- Figure 3.6: Output close index of all companies together in one dataframe.
- Figure 3.7: Code to find and visualise correlations.
- Figure 3.8: Output Of the correlation table.
- Figure 3.9: The Heatmap of the correlations.
- Figure 3.10: Code to set trading conditions and data processing for labels.
- Figure 3.11: Code to extract feature sets and map them to labels.
- Figure 3.12: k-nearest neighbor illustration
- Figure 3.13: Random forest classifier illustration.
- Figure 3.14: Illustration of SVM
- Figure 3.15: Performing machine learning
- Figure 3.16: Graph Of MMM company that year.
- Figure 3.17: Output of the code.
- Figure 3.18: Output.