Stock Market Trend Prediction

- Phase 2 -

B.E Final Year Project Report-2

Sudhakar Mishra | 1SG17CS091

Final Year

Sapthagiri College of Engineering

Computer Science and Engineering

SAPTHAGIRI
College of Engineering
Creating Tommorow

Computer Science and Technology
Sapthagiri college of Engineering
https://www.sapthagiri.edu.in/

Title:
Stock market Trend Analysis

Theme:
Computer Science

Project Period:
Odd Semester 2020

Project Group:
Individual Project

Participant(s):
Sudhakar Mishra

Supervisor(s):
Dr. Vivek Chaturvedi

Link to project: `https://github.com/sudhakarmishra2011/STOCK-MARKET-TREND-PREDICTION`

Page Numbers: 34

Date of Completion:
sep 6, 2020

Problem Statement:

Analyzing the stock market data and try to fit them in the machine learning models in a way that it outperforms the result produced by other contemporary researchers in this area.

Contents

Preface

The project titled **Stock Market Trend Prediction** is aimed to predict the trends in the stock market using statistical tools and machine learning models in order to reduce the risk involved in the business.
Predicting stock market trend is quite a difficult job considering all the factors that could affect it. In this report, I would walk through all the steps required to analyze the data and pick a model based on the type of data. I would also point out the problems with existing research results and propose solutions to that. I would also discuss what should be the parameters for accuracy, are the traditional parameters fine enough to work with? A comparison of models like Random Forest, Support Vector Machine, Logistic Regression, Auto Regressive Integrated Moving Average etc. on various data available in context to Indian Market is also included in this report.

SCE Bangalore, Sep 6, 2020

Sudhakar Mishra

Acknowledgement

I am thankful to SCE Bangalore for providing me the opportunity to do this project.

I extend my heartiest thanks to Dr. Vivek Chaturvedi (Asst. Professor, Computer Science and Technology SCE Bangalore) for supporting me to achieve this goal. I would also like to acknowledge Ashutosh Upadhye(Final Year, CSE UG, SCE Bangalore) and Vishal Kr. Chaudhari(Final Year, CSE UG, SCE Bangalore) for their help for formulation of models and statistical tools. For helping me understand economics associated with this project better, I would like to thank Shweta Suman(B.Com., LNMU).

Chapter 1

Introduction

The prediction of stock market trend is a fairly complex task, because the data are inherently noisy, non-stationary, and deterministically chaotic. The noisy characteristic refers to the unavailability of complete information from the past behavior of stock markets to fully capture the dependency between future and past prices. The information that is not included in the model is considered as noise. The non-stationary characteristic implies that the distribution of data is changing over time. By deterministically chaotic, one means that financial time series are short-term random but long-term deterministic. Many factors and unexpected events or incidents like economic or political situation, trader' s expectations, catastrophe or war may cause the change of a financial time series such as stock market index and exchange rates. At the same time the relationship of any financial time series with the other related data series may also change with time. Therefore, predicting the financial market' s movements is quite difficult.

Why is this area becoming a hot topic?
Trading in stock market indices has gained popularity in major financial markets around the world. Accurate predictions of stock market indexes are important for many reasons. Chief among these are the need for the investors to hedge against potential market risks, and opportunities for speculators and arbitrageurs to make profit by trading in stock index. Clearly, being able to accurately forecast the stock market index has profound implications and significance for both researchers and practitioners. Evidently, some of the researchers came up with really good results which could theoretically give some good profit.

Chapter 2

Background

First, we discuss some of the terminologies and concepts which are necessary to understand the scope of this work.

2.1 Data

The data we are collecting from stock market contains daily opening price(O), daily closing price(C), daily high(H), daily low(L) and volume traded on that day(V). Apart from OHLCV, some stock market data have adjusted closing price(refer to appendix [A] for details) too, to accommodate corporate actions.

2.2 How do traders deal?

Investopedia, one of the most followed website for stock market related things, in a series of blogs(`https://www.investopedia.com/university/stocks/stocks3.asp`) suggest that, traders do not rely much on machine learning for prediction. They use their own conscience for buying and selling stocks. They use their understanding of markets to make the deal. Often global influence travels slowly to the local market, so that can also be used as an indicator to buy or sell. Political stability, war conditions, corporate actions, climatic catastrophes, and many other things affect the way traders deal, they care for all these things. Some traders also use statistical tools discusse in chapter [3] to understand the markets.

2.3 Types of prediction

In this project, we have proposed methods to predict the direction of price. Based on the OHLCV(open, high, low, close, volume) data we have, we need to predict the direction of price. The direction could be either upward or downward(rarely same), so this narrows our prediction to binary classification. We could predict the direction of price intra-day or inter-day, either the direction of today's closing with respect to today's opening(O/C) or the direction of tomorrow's closing with respect to today's closing(C/C). The second one(C/C) is better because it captures the pattern of both during work hour change and after work hour change. In inter-day prediction, we may target for single day interval or for multiple day intervals like a week or two. The prediction results get better and better for longer intervals because daily noise has less role to play when we go for prediction over longer intervals. We would come to these things once more while presenting the result. amsmath

Chapter 3

Technical Indicators

Technical Indicators are important parameters that are calculated from time series stock data that aim to forecast financial market direction. These are mathematical formulas in terms of stock market price and volumes. These are the tools which are widely used by investors to check for bearish or bullish signals. Investors use these tools directly to buy and sell stocks. I have used these indicators to generate features for machine learning models. We shall see few popular indicators which will help us understand better the role of indicators in predicting the direction of stocks price.

- **Relative Strength Index(RSI)**

$$RSI = 100 - \frac{100}{\frac{AverageGain}{AverageLoss}}$$

RSI is a popular momentum indicator which determines whether the stock is overbought or oversold. A stock is said to be overbought when the demand unjustifiably pushes the price upwards. This condition is generally interpreted as a sign that the stock is overvalued and the price is likely to go down. A stock is said to be oversold when the price goes down sharply to a level below its true value. This is a result caused due to panic selling.

RSI ranges from 0 to 100 and generally, when RSI is above 70, it may indicate that the stock is overbought and when RSI is below 30, it may indicate the stock is oversold. The same can be seen in [3.1]

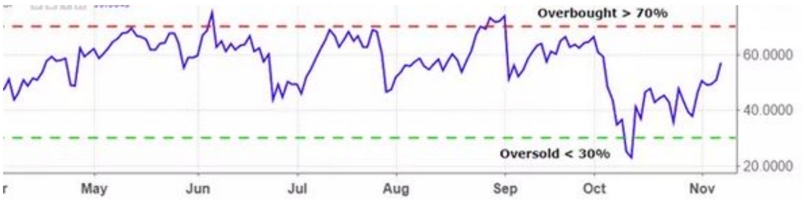


Figure 3.1: RSI

- **Moving Average Convergence Divergence(MACD)**

$$MACD = 12dayEMA - 26dayEMA$$

$$Signalline = 9dayEMA$$

where EMA is exponential moving average

$$EMA(today) = Price(today) * k + EMA(yesterday) * (1 - k)$$

where k is multiplier for weighting

$$k = \frac{2}{days + 1}$$

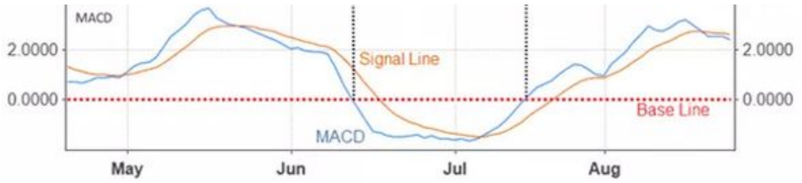


Figure 3.2: MACD

When the MACD goes below the SingalLine, it indicates a sell signal. When it goes above the SignalLine, it indicates a buy signal.

- **Williams %R**

$$\%R = \frac{H14 - price}{H14 - L14} * (-100)$$

where, L14 = Lowest Low over the past 14 days
H14 = Highest High over the past 14 days
Williams %R ranges from -100 to 0. When its value is above -20, it indicates a sell signal and when its value is below -80, it indicates a buy signal.

- **Stochastic Oscillator**

  $$\%K = \frac{price - L14}{H14 - L14} * 100$$

  where, L14 = Lowest Low over the past 14 days
  H14 = Highest High over the past 14 days
  Stochastic Oscillator follows the speed or the momentum of the price. As a rule, momentum changes before the price changes. It measures the level of the closing price relative to low-high range over a period of time.

- **Price rate of change**

  $$PROC(t) = \frac{C(t) - C(t-n)}{C(t-n)}$$

  where, PROC(t) = Price Rate of Change at time t
  C(t) = Closing price at time t
  It measures the most recent change in price with respect to the price in n days ago.

Several other indicators like Bollinger Bands Double, Exponential Moving Average, Moving average, Simple Moving Average, Triple Exponential Moving Average, Weighted Moving Average, On Balance Volume, Chaikin A/D Oscillator, Average True Range, Stochastic Relative Strength Index, Average Directional Movement Index etc are also used. They are calculated over varied number of intervals and it is left upon feature selection to see which one is better.

Apart from these traditional indicators I have added a new indicator named **Lag** which is nothing but a normalized comparison of today's closing vs n days earlier closing. The lag is calculated over various period of time and evidently they are useful for prediction. The feature selection supports the claim. We would discuss it again once we define our model.

Coming back to question posed earlier, how these indicators are useful for our prediction? There are various reasons for using them, in the next chapter [4] we can see several literatures endorse the use of indicators. When we train our models using the OHLCV data directly as features, the accuracy while testing the model is quite low. But when we train the models with technical indicators as features the accuracy while testing is much better. This suggests that using technical indicators as features is helping our models learn better. However, few neural network models are capable enough to generate these kind of technical indicators on their own, so they don't need to be trained by these indicators. Models like Random Forest, on a broader note are voting technique over several features and as most of the features has some importance associated with them, training random forest with these features essentially becomes voting over several features. However, the most important reason that these indicators help because they are in some sense trying to pick long term trends over some interval and averaging out the daily noise. So, there might be the chance that daily noise gets cancelled or some how decreased over long period of time. The graph of Opening price vs ADX [3.3] is evidence for same as the curve for ADX is much smoother than the opening value itself. Considering all these reasons it would be obvious to say that these indicators help.

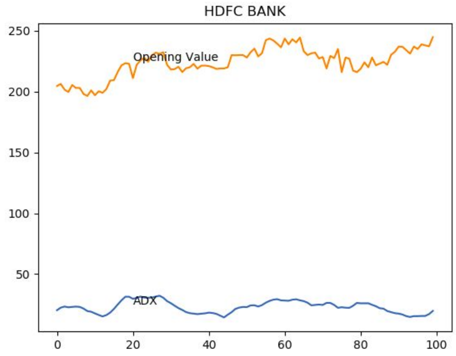


Figure 3.3: Opening price vs ADX

The detailed description of indicators could be found in [4], [1], [6], `https://mrjbq7.github.io/ta-lib/` and `https://www.investopedia.com/`. There are a lot other sources also available on Google.

Chapter 4

Literature Survey

There are abundant of literature in this area claiming very good accuracy however they are not still commercially used. We now present a brief summary of a few of the good literature and then come to the flaw due to which they are of no commercial use.

In [4], authors have used Random Forest and Support Vector Machine with some modification to achieve a considerably good result over a very noisy stock data of NIFTY 50 SP. The paper suggests to tune the hyper-parameters of Random Forest and Support Vector Machine on trail and hit basis over the training data to achieve the best accuracy while testing. They also suggest the use of few of the technical indicators mentioned in chapter [3]. In [1], authors have reported very good accuracy on AMAZON, APPLE, MICROSOFT data again using SVM with same technical indicators as features. But this time instead of predicting daily trend, the paper suggests predicting over 15 days interval or 30 days interval or more for better accuracy. The model has been trained regularly over a moving window and the updated model is used to test the data ahead. Also, this paper uses a lot of indicators and leaves up to feature extraction to select the best. In [6], authors suggest exponential smoothening of data and then performing the experiment ahead with same method as the one proposed by researchers in [4].

Apart from these simple model, in [3], authors suggest use of multi-layered LSTM and claims that the memory holding capacity of LSTM helps to pick repeating patterns may be of some seasonal festival that repeat every year or so. In [2] and [5], authors endorse Auto Regressive Integrated Moving Average(ARIMA) over these models because of its persistent accuracy over any several interval of time. Some literature suggest use of Ensemble model as it improves confidence of true positives(refer to appendix [A] to understand true positives).

4.1 Pros

The main points to note from the above discussed literature are:-

- Tuning of hyper-parameters
- Updating models with moving window
- Predicting over longer interval of times
- Using all indicators and let feature selection do the task
- Exponential smoothening of data
- Use of LSTM might pick festival trends
- ARIMA may be better because of the time series characteristics of data
- Ensemble models may impove confidence of true positives

4.2 Cons

Reasons why even after claiming such good results there is almost no commercial use of these surveys:-

- Use of impure data :- Not many of the papers publish their codes as well but after looking on few of the implementation like one in `http://ahmedas91.github.io/blog/2016/07/01/predicting-us-equities-trends-using-random-forests/`, it seems some people are actually using impure data. By impure data I mean they are using the the data to predict the same data. For instance in figure 4.1, we could see that after shuffling the training set gets indicator 22 which is composed of data from day 18 and day 19. Predicting for day 18 or day 19 in test set would yield good result because it is already used in training the model.
  Solution :- Write clean code to avoid such anomaly.

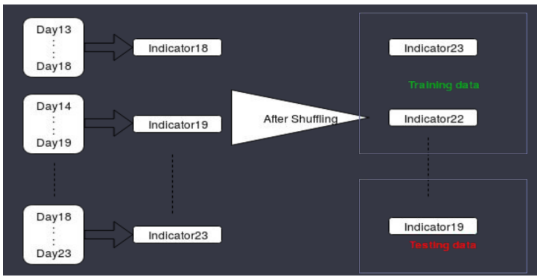


Figure 4.1: Impure Data

- Use of less noisy data :- The data from Japan's stock, Switzerland's stock and few other are less noisy data. There is not a lot of daily noise in those data. So, predicting trends in such data becomes easy. The good result doesn't mean that the model was good.
  Solution :- Use data that has some noise to check robustness of model like NIFTY 50 etc.

- Use of biased data :- During some periods some stocks really do well continuously or do bad continuously. Picking such period for training and testing is really bad because if we predict the trend to be upward every day where actually the stock price was going good, we would get a good accuracy. Suppose, in actual price out of 100 days, if on 70 days the stock makes profit. Now let us say we have a very bad model which predicts every thing to be positive, that means it would predict profit for all 100 days. Even then we would get a accuracy of 70% which is really great.
  Solution :- Check if data is biased or the prediction is biased. I would present how I tackled this problem in result section.

- Ill representation of accuracy :- High accuracy doesn't mean certain profit. It might be the case that on the day we correctly predicted the trend, had lower returns while the day where we failed had much higher return. Thus we may end up in total loss. Another thing is that if we predicted the trend to be causing loss, and day ends up making profit is not as bad as we predicted it to be making profit and it made loss. Think twice about that, if we predict loss, no one would put money in that stock, even the stock makes profit there is nothing to loose. But think of the other scenario where we said that today is a profitable day for a particular stock and people invested their money but they all end up loosing, they would incur a total loss. Thus it is better to make false negative than false positive. So, there should be lesser false positives.

Solution :- While printing the result print the accuracy of false positives, true positives, false negatives and true negatives. Look in the result section for a basic idea.

Chapter 5

Data

The data we are going to use for our experiments are going to be from Indian markets. The Indian market is a perfect balance to check the relevance of models, it is enough rough to check the robustness of models and enough stable so that the models don't get underestimated. The markets like Japan and Switzerland are stable enough to make any model look good. The Indian market brings a good amount of challenge to the models. India being a developing nation have many big economic reforms going on like GST, Demonetisation and many more, which makes the business even more challenging.

5.1 Data used

The data which is used for experiments in this project are data from following companies taken from the period of 2005 to 2018:-

- BSE Sensex
- NIFTY 50
- YES Bank
- HDFC Bank
- AXIS Bank
- ICICI Bank
- HSBC India
- JET Airways
- TATA Motors
- Reliance Inds. Lim.
- APPLE(USA)

These all data are available at `in.investing.com`, free for any commercial and non-commercial use. The data are also available on Yahoo-Finance and other websites. If it is required to add new data-set to the tool, they need to be added rightly

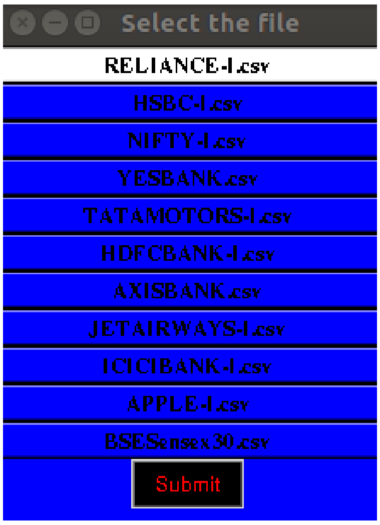


Figure 5.1: Data Selection Prompt

under data/raw_data folder in csv format. The same would be available as 5.1 once the tool is run. Each data have some unique characteristics attached with it.

5.1.1 Nifty 50

The NIFTY 50 index is National Stock Exchange of India's benchmark broad based stock market index for the Indian equity market. Full form of NIFTY is National Stock Exchange Fifty . It represents the weighted average of 50 Indian company stocks in 12 sectors and is one of the two main stock indices used in India. The opening and closing price value patterns over 12 years can be seen in figure 5.2 and figure 5.3 The drop from 2008 to 2009 is result of **Global Recession**.

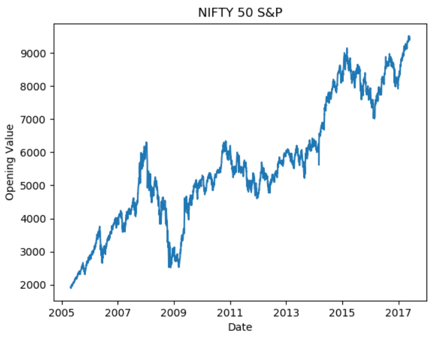


Figure 5.2: Opening price pattern

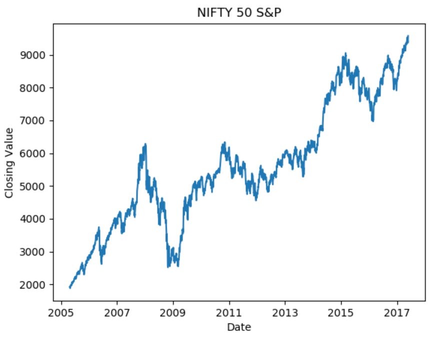


Figure 5.3: Closing price pattern

The mean over the years is **5580.03** and the variance is **3513247.95**. But it is not wise to consider mean and variance of data over such a long period of time because as we could see the prices started from **2,000** and went above **10,000**. It would be better if we analyze the data over short period of time rather than such a long interval. So, the mean and variance plotted over a window of 50 days could be seen in figure 5.4 and figure 5.5.

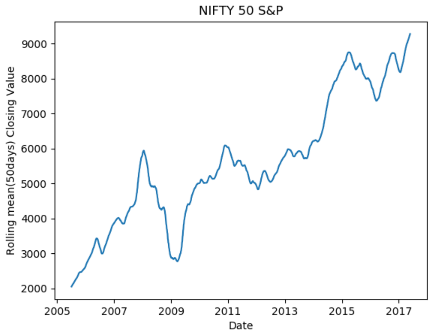


Figure 5.4: Mean in rolling window of 50

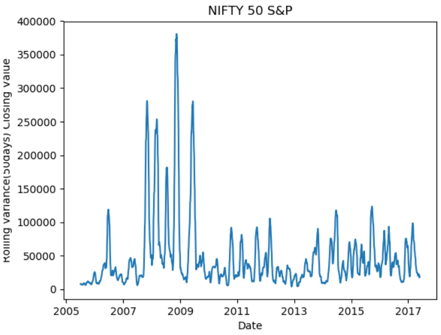


Figure 5.5: Variance in rolling window of 50

Certain peaks in the Variance graph show major festivals like **Diwali, Holi** etc.

5.1.2 HDFC Bank

DFC Bank Limited (Housing Development Finance Corporation) is an Indian banking and financial services company headquartered in Mumbai, Maharashtra.

The opening and closing price value patterns over 12 years can be seen in figure 5.6 and figure 5.7 The mean over the years is **553.79** and the variance is **131847.11**.

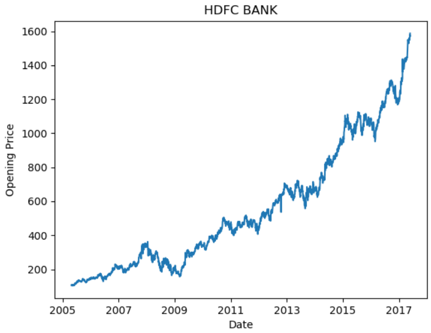


Figure 5.6: Opening price pattern

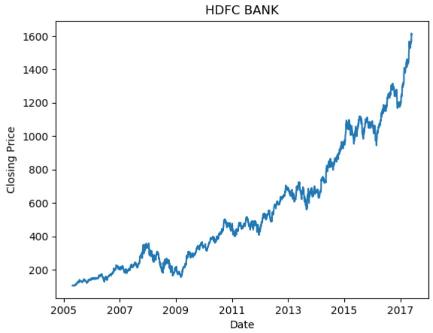


Figure 5.7: Closing price pattern

The mean and variance plotted over a window of 50 days could be seen in figure 5.8 and figure 5.9.

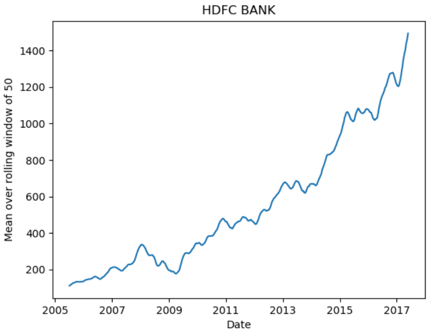


Figure 5.8: Mean in rolling window of 50

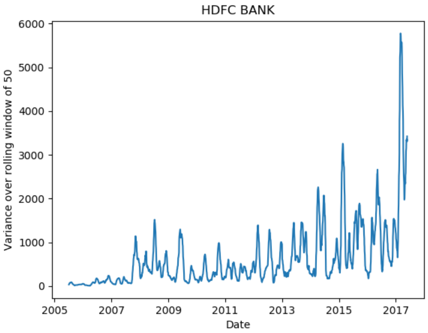


Figure 5.9: Variance in rolling window of 50

5.1.3 Reliance Inds. Ltd.

Reliance Industries Limited (RIL) is an Indian conglomerate holding company headquartered in Mumbai, Maharashtra, India. Reliance owns businesses across India engaged in energy, petrochemicals, textiles, natural resources, retail, and telecommunications. The opening and closing price value patterns over 12 years can be seen in figure 5.10 and figure 5.11

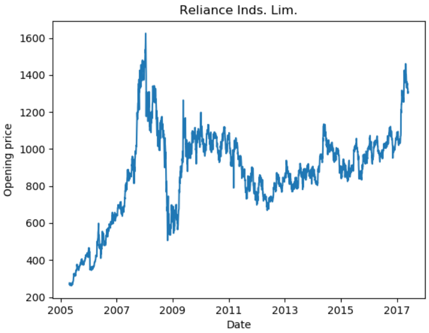


Figure 5.10: Opening price pattern

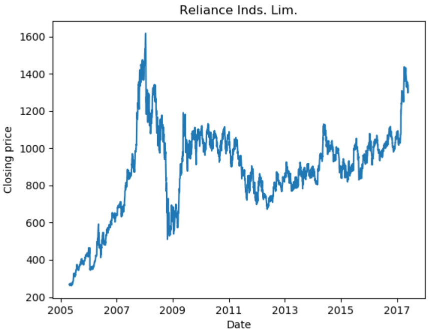


Figure 5.11: Closing price pattern

The mean over the years is **879.19** and the variance is **58031.30**. The mean and variance plotted over a window of 50 days could be seen in figure 5.12 and figure 5.13.

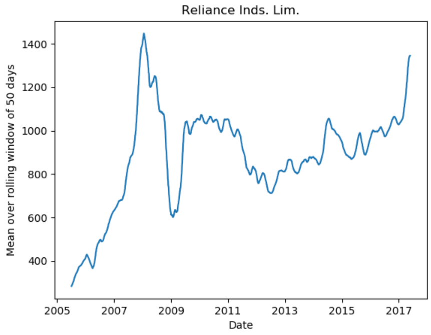


Figure 5.12: Mean in rolling window of 50

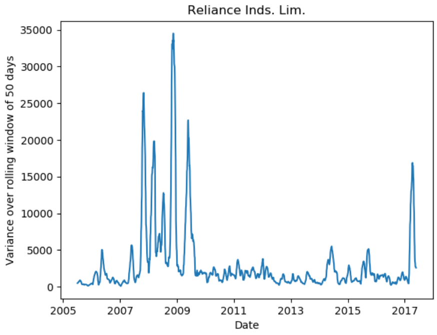
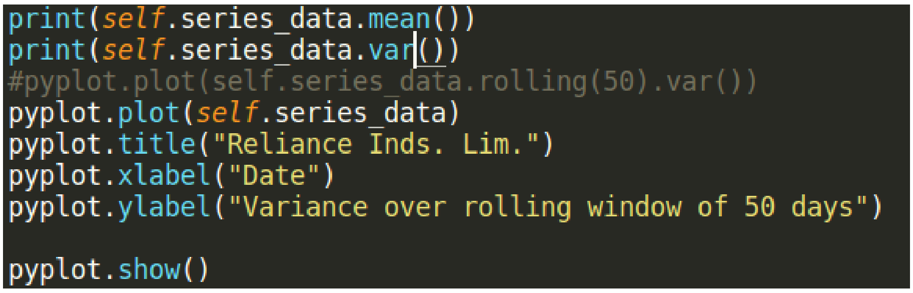


Figure 5.13: Variance in rolling window of 50



```
print(self.series_data.mean())
print(self.series_data.var())
#pyplot.plot(self.series_data.rolling(50).var())
pyplot.plot(self.series_data)
pyplot.title("Reliance Inds. Lim.")
pyplot.xlabel("Date")
pyplot.ylabel("Variance over rolling window of 50 days")

pyplot.show()
```

Figure 5.14: Code snippet for data property related plots

Similarly, other data could also be analyzed like the above three. The code in figure 5.14 could be used as a reference.

Chapter 6

Tools Used

- Python :- Language of coding for all purpose in this project.
- NumPy :- NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. For processing CSV files and dealing with matrices, NumPy has been used in this project.
- Pandas :- In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. Similar job as NumPy but it helps better with time series data.
- Matplotlib :- A python library used to draw plots.
- TaLib :- TA-Lib is widely used by trading software developers requiring to perform technical analysis of financial market data. Includes 150+ indicators such as ADX, MACD, RSI, Stochastic, Bollinger Bands, etc.
- SkLearn :- Scikit-learn is probably the most useful library for machine learning in Python. It is on NumPy, SciPy and matplotlib, this library contains a lot of effiecient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction. I have used it to create models like Random Forest, Support Vector Machine, Logistic Regression etc.

- Keras :- Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. In this project it is used for applying Neural Network like LSTM etc.
- StatsModels :- statsmodels is a Python module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration. The same has been used to create ARIMA model.
- Tkinter :- Used to create basic interface.

Chapter 7

Functionality

The data files containing the OHLCV data are added to `data/raw_data folder`. On running the `main.py`, the tool provides an option to select the data file as in figure 5.1. After selecting the desired data, the data is pre-processed by removing the NA or empty values by mean value of that column by using Sk-learn and the pre-processed data is added to `data/proc_data`. For pre-processing the data, a prompt appears for the users to select the columns numbers of respective data columns as in figure 7.1.

After that the technical indicators as mentioned in chapter 3 are calculated using Talib library and the indicators are stored in `indicators/` folder. There are two indicators file generated one for Opening vs Closing prediction with name `yoc` and other for Closing vs Closing with name `ycc`. The `yoc` uses one extra column that is today's opening as one of the features. After all these calculations, user is asked to select the split of data and up to how many more windows user wants to predict. The figure 7.3 shows the same.

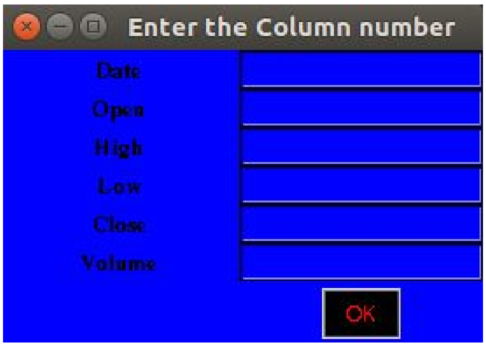


Figure 7.1: Select column numbers

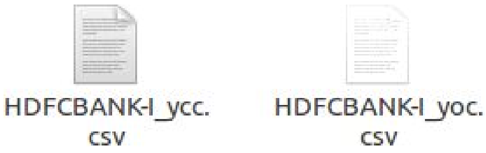


Figure 7.2: Select the split size

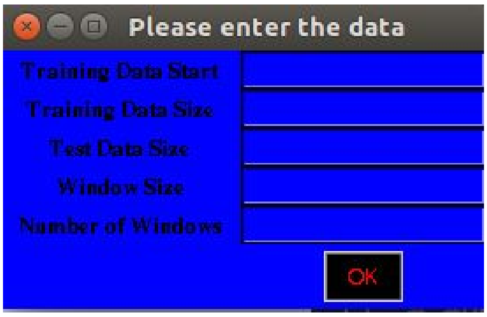


Figure 7.3: Data Split Size

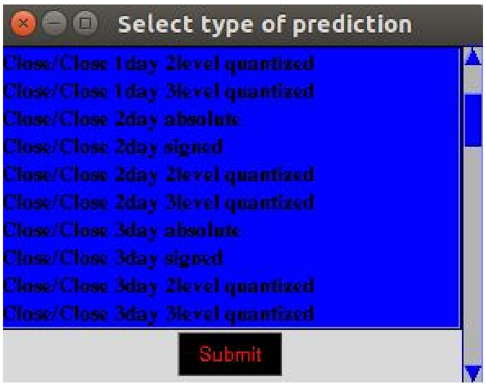
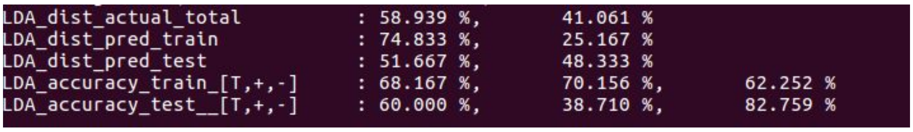

Select type of prediction
Close/Close 1day 2level quantized
Close/Close 1day 3level quantized
Close/Close 2day absolute
Close/Close 2day signed
Close/Close 2day 2level quantized
Close/Close 2day 3level quantized
Close/Close 3day absolute
Close/Close 3day signed
Close/Close 3day 2level quantized
Close/Close 3day 3level quantized
Submit

Figure 7.4: Select the prediction type

Once the user enters the desired value in prompt, he/she is asked for type of prediction as in figure 7.4. Then the models are run over the data and the result is published in an elaborated fashion like figure 7.5. The representation of results are a reflection of flaws presented in chapter 4. The result contains distribution of both classes before and after prediction for both training and testing data. Printing result in this way helps to find over-fitting, biased data and biased result.



```
LDA_dist_actual_total         : 58.939 %,       41.061 %
LDA_dist_pred_train           : 74.833 %,       25.167 %
LDA_dist_pred_test            : 51.667 %,       48.333 %
LDA_accuracy_train_[T,+,-]    : 68.167 %,       70.156 %,       62.252 %
LDA_accuracy_test__[T,+,-]    : 60.000 %,       38.710 %,       82.759 %
```

Figure 7.5: Result

Chapter 8

Models and results

While creating models during the early phase of this project we created models considering the economic and mathematical significance of our data as most of the papers suggest but later we acknowledged that this is also a time series data rather than just a normal piece of data. Also, even in time series, it is specifically non-stationary data. Dickey-Fuller (refer to Appendix A) test suggests the same. We shall discuss the meaning of non-stationary data when we come to ARIMA model creation. But, first of all we need to develop a feature selection method.

As suggested by authors, in [1], we generated a lot of features and left it to Recursive Feature Elimination(refer to Appendix A) to select features from it. In chapter 3 we discussed about the **Lag** technical indicator that we created, so how can we say it is relevant? So, here is the answer, after running RFE on all the indicators few of the lags appear quite often like lag1 and lag5. That is justified by the dependency of today's price on yesterday's price and the price a week before on some day. While the same doesn't occurs for all data during all period of time but quite sure some of the lag feature appear after RFE.

8.1 The case of simple data

For the first part we used the technical indicators as feature vector to train the models and predicted the test data. Before moving ahead let us clear the way of data splitting. Most of the time we used 500 days for training and 100 days for testing and a window of size 100 rolling 10 times. By window we mean, for the first time predict by training on first 500 days and testing on next 100 days, for next windows use the 100 days from test data as a part of training data(basically add those 100 days to training data) and update the model to predict next 100 days and do so for 10 times.

Models like Lasso Regression, Logistic Regression, Ridge Regression, Naive Bayes, Linear Discriminant Analysis, Support Vector Machine and Random Forest were created during first phase(before mid-term). Most of the models mentioned above except for Random Forest and Support Vector Machine don't have much to tune. So, we used them directly to train and test. But for Random Forest and Support Vector Machine there are few parameters to tune.

8.1.1 Support Vector Machine(SVM)

With linear SVM we just have the penalty constant to tune. Penalty constant essentially determines, how much loss could we allow to classify a data point correctly. But, we have used Kernelized SVM, here we also have the kernel coeffiecient to tune.Kernel coefficient on a much broader sense determines the influence radius of data points. The kernel coefficient and the penalty constant effect the accuracy in some random fashion. So, we used Grid Search Cross Validation(refer to Appendix A) to pick a best value of the two from some values that were giving good accuracy over repeated experiments. Keeping the penalty constant fixed and varying the kernel coefficient, one could see the varying accuracy in figure 8.1. Again see figure 8.1 for impact of constant kernel coefficient and varying penalty constant on accuracy.

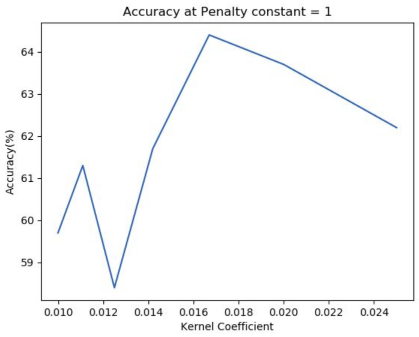


Figure 8.1: Varying kernel coefficient

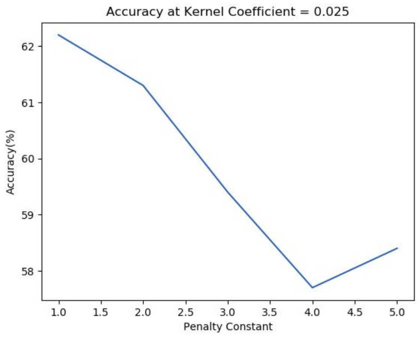


Figure 8.2: Varying Penalty constant

8.1.2 Random Forest

In Random Forest we have a lot of hyper-parameters to tune like n_estimators, min_samples_split, min_samples_leaf, min_weight_fraction_leaf=, max_features, max_leaf_nodes(refer to A). But while doing several experiments we noticed most of the hyper-parameters except number of estimators attain some fix value for each data and for that value the average accuracy is considerably better. The varying accuracy for same data by changing the number of estimators could be seen in figure 8.3.

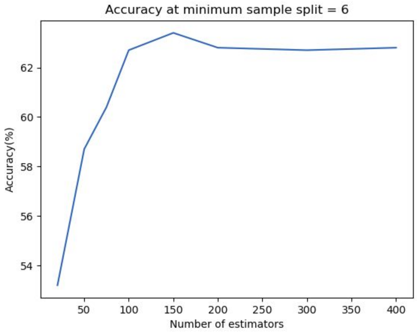


Figure 8.3: Result with minimum sample spilt fixed at 6

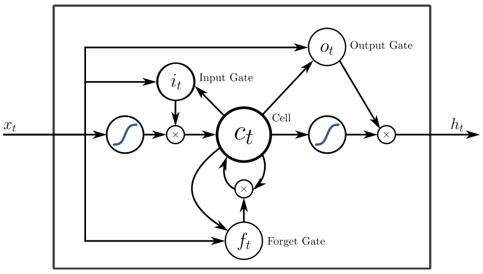


Figure 8.4: A simple LSTM cell

8.1.3 Long Short-Term Memory(LSTM)

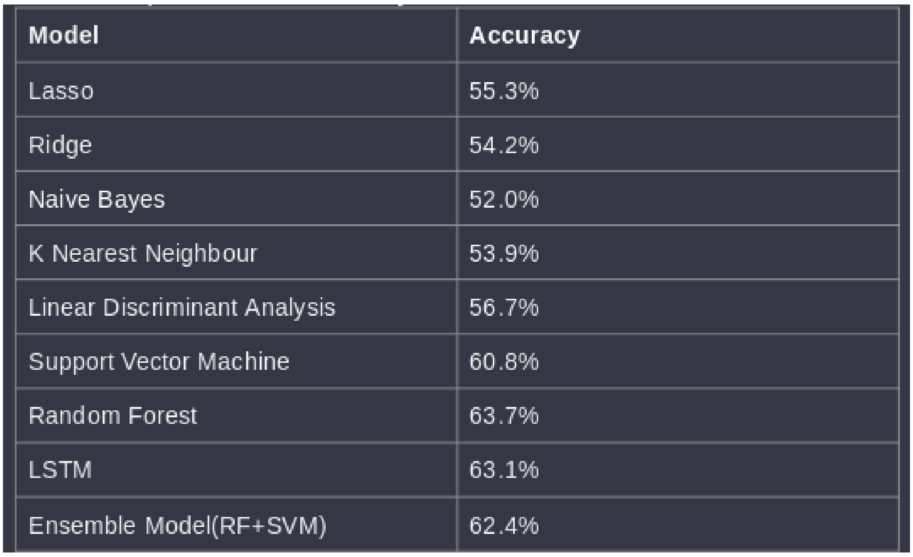
We used LSTM over any other neural networks because LSTM has memory associated with it and it overcomes the vanishing gradient problem. A simple LSTM cell has option to store or forget a piece of information. The graphic representation of an LSTM cell can be seen in figure 8.4. We have used the LSTM model with two layer of 300 LSTM cell and a single node of Dense as output layer.

8.1.4 Ensemble Model

We tried several Ensemble model like Logistic Regression + Random Forest, Logistic Regression + Support Vector Machine, Random Forest + Support Vector Machine and many others. But result didn't improve, the result was between the accuracy of individual models. The interesting fact here is that if we look closely into the models basically what it does is that it adds the probability of getting classified in a class by all the models and classifies it in the class with maximum added probability. If we set a threshold over the probability like classify into a class only if all the models suggest the same class, then we could get a good accuracy. One of the best ensemble model out of all others I tested was SVM + Random Forest. When we put Hard voting method(refer to Appendix A) for classifying, out of 100 only 78 were classified but 62 were accurately predicted.

8.1.5 Result-1

When the aforementioned models were run of NIFTY 50 data SP. The results obtained were like 8.5.



Model	Accuracy
Lasso	55.3%
Ridge	54.2%
Naive Bayes	52.0%
K Nearest Neighbour	53.9%
Linear Discriminant Analysis	56.7%
Support Vector Machine	60.8%
Random Forest	63.7%
LSTM	63.1%
Ensemble Model(RF+SVM)	62.4%

Figure 8.5: Result

8.2 The case of time series data

As we have already discussed, the data we are using is a time series data. It is not stationary data i.e. the mean is not zero and the variance is not constant, this could be evidently seen in chapter 5 where we discussed about the data. The same is suggested by Dickey-Fuller test A. So, the first step is to make the data stationary, one of the most popular way is using first or second difference. When working with Nifty 50 SP data, the first difference itself becomes stationary as suggested by Dickey-Fuller test. First difference the value of current data point minus value of previous data point. The figure 8.6 shows the mean is zero and the variance is also almost constant with reasonably big fluctuations on some festivals like Diwali or some other special occasions.

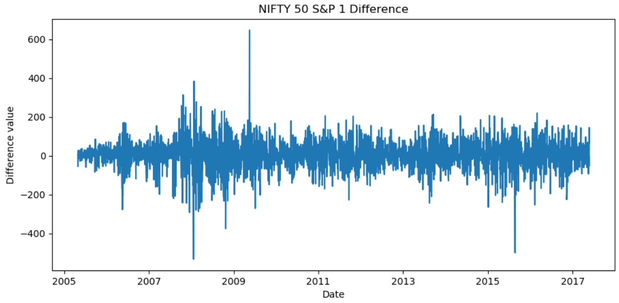


Figure 8.6: Nifty First Difference

Once we have got a stationary data we are ready to apply ARIMA model over the stationary data. First, we shall discuss about ARIMA.
An autoregressive integrated moving average, or ARIMA, is a statistical analysis model that uses time series data to either better understand the data set or to predict future trends. n autoregressive integrated moving average model is a form of regression analysis that gauges the strength of one dependent variable relative to other changing variables. The model's goal is to predict future securities or financial market moves by examining the differences between values in the series instead of through actual values.

An ARIMA model can be understood by outlining each of its components as follows:

- Autoregression (AR) refers to a model that shows a changing variable that regresses on its own lagged, or prior, values.
- Integrated (I) represents the differencing of raw observations to allow for the time series to become stationary, i.e., data values are replaced by the difference between the data values and the previous values.
- Moving average (MA) incorporates the dependency between an observation and a residual error from a moving average model applied to lagged observations.

The ARIMA model has three parameters (p, d, q) one corresponding to each above three point respectively. The d part is pretty obvious and the value could be decided by the number of time we need to difference the data to make it stationary. For our case it is 1. For p and q most of the researcher simple trial and hit with different values and pick the best on basis of accuracy. But the best p and q value might not continue to give good results every time as the selection of p and q is not mathematically justified, the better accuracy might just be a coincidence. The way to select p and q is to draw Auto-correlation Function(ACF) and Partial Auto-correlation Function(PACF) and based on that (p,q) value shall be decided. The ACF plot is merely a bar chart of the coefficients of correlation between a time series and lags of itself. The PACF plot is a plot of the partial correlation coefficients between the series and lags of itself. In general, the partial correlation between two variables is the amount of correlation between them which is not explained by their mutual correlations with a specified set of other variables. For example, if we are regressing a variable Y on other variables X1, X2, and X3, the partial correlation between Y and X3 is the amount of correlation between Y and X3 that is not explained by their common correlations with X1 and X2. If we look closely on plot of ACF 8.7 and PACF 8.8 for Nifty data using StatsModel, we can see blue zone which basically represents the critical zone. (p,q) value can be determined as number of bars above blue zone before the first bar inside the blue zone, which is basically (1,1) here but for the HDFC data it was (2,1). So, it is totally dependent on data type. The accuracy using ARIMA model was best of all other models and was **67.6%**. The better result is the reflection of fact that we are considering one of the most special character of this data i.e time series.

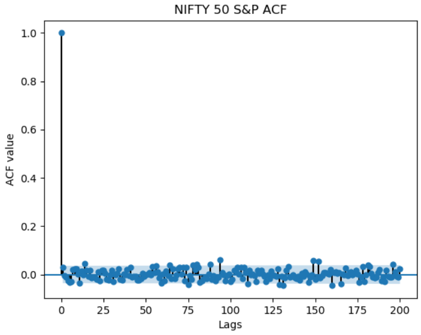


Figure 8.7: Auto-correlation Function

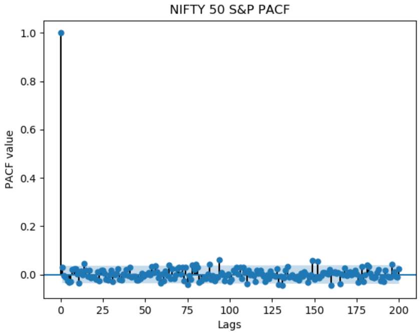


Figure 8.8: Partial Auto-correlation Function

Chapter 9

Conclusion and Future work

After completing the project, I found few things that must be said which currently the researchers are avoiding.

- Few researchers are making some mistakes to get good accuracy.
- Most of them are ill-interpreting the results.
- No model is perfect. Claiming a model to be best is a total lie.
- Selection of model totally depends on type of data.
- No single model is universal. It might fail on some data.

The way I have created the models I feel there is no more chance of improvement in them. I have tried to fill all possible gaps from where we could boost accuracy. So, it would be better to develop some new techniques and use some hybrid models rather than toying these models. It would be great if someone comes up with a good interface and dedicated database to do real-time prediction. That thing would certainly have commercial use. Another idea could be developing a business from small money investors using these models to minimize the risks.

Bibliography

[1] Xienji Di. *Stock Trend Prediction with Technical Indicators using SVM*. SCPD Apple. 2011.

[2] Jia-Yann Leu Jung-Hua Wang. *Stock Market Trend Prediction Using ARIMA-based Neural Networks*. International Conference on neural network. 1996.

[3] Fangyan Dai Kai Chen Yi Zhou. *A LSTM-based method for stock returns prediction : A case study of China stock market*. IEEE International Conference on Big Data. 2015.

[4] M. Thenmozhi Manish Kumar. *FORECASTING STOCK INDEX MOVEMENT: A COMPARISION OF SUPPORT VECTOR MACHINES AND RANDOM FOREST*. Indian Institute of Capital Markets 9th Capital Markets Conference Paper. 2006.

[5] M. Thenmozhi Manish Kumar. *Stock Index Return Forecasting and Trading Strategy Using Hybrid ARIMA-Neural Network Model.*

[6] Luckyson Snehanshu Sudeepa. *Predicting the direction of stock market prices using random forest*. Applied Mathematical FInance. 2016.

Appendix A

Appendix

Adjusted closing price :- An adjusted closing price is a stock's closing price on any given day of trading that has been amended to include any distributions and corporate actions that occurred at any time before the next day's open.
True Positive and True Negative :- A true positive is an outcome where the model correctly predicts the positive class. Similarly, a true negative is an outcome where the model correctly predicts the negative class.
False Positive and False Negative :- A false positive is an outcome where the model incorrectly predicts the positive class. And a false negative is an outcome where the model incorrectly predicts the negative class.

Dickey-Fuller Test :- In statistics, the Dickey–Fuller test tests the null hypothesis that a unit root is present in an autoregressive model. The alternative hypothesis is different depending on which version of the test is used, but is usually stationarity or trend-stationarity. It is bit difficult to understand it by definition, so visit `https://www.youtube.com/watch?v=2GxWgIumPTA` to understand it more clearly.

Recursive Feature Elimination :- Given an external estimator that assigns weights to features (e.g., the coefficients of a linear model), the goal of recursive feature elimination (RFE) is to select features by recursively considering smaller and smaller sets of features. First, the estimator is trained on the initial set of features and the importance of each feature is obtained either through a coef_ attribute or through a feature_importances_ attribute. Then, the least important features are pruned from current set of features. That procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached.
Grid Search Cross Validation :- Grid search is an approach to parameter tuning that will methodically build and evaluate a model for each combination of algorithm parameters specified in a grid. Refer `https://scikit-learn.org/stable/`

modules/generated/sklearn.model_selection.GridSearchCV.html for more detail.

Hyper-parameters random forest :- Refer to https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html to understand all the parameters well.

Voting :-

If 'hard', uses predicted class labels for majority rule voting. Else if 'soft', predicts the class label based on the argmax of the sums of the predicted probabilities, which is recommended for an ensemble of well-calibrated classifiers.