Completed as part of the Final Year Project for the Bachelor of Engineering in Computer Science 2017-18, University of Hong Kong.

Forecasting the S&P 500 Index using a Machine Learning Approach

Ismail Hossain
Supervisor: Dr YIP Chi Lap Beta

COMP4801: Final Report, 2013527186

Department of Computer Science

University of Hong Kong

April, 2018

Acknowledgements

I would like to thank Dr. Beta for his humble and kind effort to support me throughout this project, without whom completing this would be a mountainous effort. I am also sincerely grateful to my family, my late father Nurul Hoque and my friends who were always available through thick and thin.

Last but not the least, the University of Hong Kong, for giving me this incredible opportunity to thrive and look into life much strong much more confidently.

Abstract

Predicting the direction of stock prices is a widely studied subject in many fields including trading, finance, statistics and computer science. Investors in the stock market can maximize their profit by buying or selling their investment if they can determine when to enter and exit a position. Due to the non-linear, volatile and complex nature of the market, it is quite difficult to predict. The factors that influence stock prices are complex to model. Machine Learning algorithms have been widely used to predict financial markets with some degree of success. This project aims to study the application of these algorithms to predict one-day-ahead outcomes of the S&P500 Index with special emphasis on feature generation and analyzing the predictive ability of several algorithms. Three different outcomes based on Next Day Closing price, Next Day Opening Price and Next Day Returns were studied. The construction of the predictive models is based on historical information of the index extracted through Yahoo finance. In order to analyze the predictive quality of the algorithms, the final results were compared through the implementation of ROC curve.

Index

Chapter 1

Introduction

Forecasting is the process of predicting the future values based on historical data and analyzing the trend of current data. Processing powers of computers nowadays have become powerful enough to process large amount of data. Stock price time-series are often characterized by a chaotic and non-linear behavior which makes the forecast a challenging task. The factor that produces uncertainty in this field are complex and from different nature, from economic, political and investment decisions to unclear reasons that, somehow, produce effects and make hard to predict how the prices will evolve. The stock market attracts investments due to the ability of producing high revenues. However, owing to its risky nature, there is a need for an intelligent tool that minimizes risks and, hopefully, maximizes profits.

Predicting stock prices using historical data of the time-series to provide an estimate of future values is the most common approach among the literature. More recently, researchers have started to develop machine learning (ML) techniques that resemble biological and evolutionary process to solve complex and non-linear problems. This work contrasts the typical approach, where classical statistical methods are employed.

The application of ML algorithms can be helpful in various financial problems. It has already been applied successfully in financial forecasting, trading strategies optimization and financial modelling.

This project focus on forecasting stock prices time-series using a machine learning approach. Considering a short-term forecasting problem (one-day-ahead forecast), the objective is to predict the stock price in given day_{t+1} using a set of inputs variables that represents the past stock prices up to day_t .

The problem can be described as follows: given a set of inputs variables, x_t , x_{t-1} , ... , x_{t-n} we have:

$$x_{t+1} = f(x_t , x_{t-1}, \ldots, x_{t-n})$$

where x_{t+1} is the next value of the time-series and f is the forecast function.

ML algorithms play the role of finding the best forecast function, f, through the identification of hidden patterns and relations in the data either by parameter optimization, creation of expressions and variable selection.

1.1 Document Structure

The paragraphs bellow describes how the rest of the document is structured.

Chapter 2, Background, summarizes some past work done on this field and also the Scope and Objectives of this project.

Chapter 3, Data collection, describes the approach implemented during this project to collect the dataset, the preparation and preprocessing of the data.

Chapter 4, presents the list of technical indicators and target variables implemented in this project.

Chapter 5, discusses the machine learning models that are to be used in the project.

Chapter 6, Results and Discussion, presents the outcome of the project and an interpretation of the results provided.

Chapter 7, Conclusion and Future Work, summarizes the project results and proposes possible future developments.

Chapter 2

Background

2.1 Literature Review

Stock market prediction, which has the capacity to reap large profits if done wisely, has attracted much attention from academia and business. Due to the non-linear, volatile and complex nature of the stock market, it is quite difficult to predict.

For many years the following question has been a source of continuing controversy "To what extent can the past history of a common stock's price be used to make meaningful predictions concerning the future price of the stock?" (Fama, 1965).

Answer to this question has been provided on the basis of two assumptions. The first being the theory of random walk says that successive price changes are independent, identically distributed random variables, which would make prediction impractical since future price will be no more predictable than the path of a series of cumulated random numbers. On the other hand, against the random walk model, the other theory suggests stocks past price holds patterns that which can be used to predict future behavior. The arguments for and against random walk model based on past research are stated below.

2.1.1 Random Walks and the Efficient Market Hypothesis

The idea of stock prices following a random walk is connected to that of the EMH. The premise is that investors react instantaneously to any informational advantages they have thereby eliminating profit opportunities. Thus, prices always fully reflect the information available and no profit can be made from information based trading (Lo and MacKinley, 1999). This leads to a random walk where the more efficient the market, the more random the sequence of price changes. Fama (1970) stated that there are three versions of efficient markets:

1. Weak-form: An information set comprises of historical prices only, meaning that it is not possible to earn superior risk adjusted profits which are based on past prices (Shleifer, 2000). This leads to the random walk hypothesis.
2. Semi-strong form: The information set includes historical prices and all publicly available information as well.
3. Strong form: Information set is broadened still further to include even insider information (LeRoy, 1989)

2.1.2 Arguments for the Random Walk Model

Shleifer (2000) identified three main arguments for EMH:

1. Investors are rational and hence value securities rationally.

2. Some investors are irrational but their trades are random and cancel each other out.

3. Some investors are irrational but rational arbitrageurs eliminate their influence on prices.

If all these exist, then both efficient markets and stock prices would be very unpredictable and thus would follow a random walk

2.1.3. Arguments against the Random Walk Model

Market Over- and Under-reaction:

Fama (1998) argues that investors initially over or under-react to the information and the serial correlation explained above is due to them fully reacting to the information over time. The phenomenon has also been attributed to the 'bandwagon effect'.

Seasonal Trend:

Here, evidence is found of statistically significant differences in stock returns during particular months or days of the week. The 'January effect' is the most researched, but Bouman and Jacobsen (2002) also find evidence of lower market returns in the months between May and October compared with the rest of the year.

Size :

Fama and French (1993) found evidence of correlation between the size of a firm and its return. It appears that smaller, perhaps more liquid firms, garner a greater return than larger firms.

Dividend Yields:

Malkiel (2003) notes that generally a higher rate of return is seen when investors purchase a market basket of equities with a higher initial dividend yield.

Value vs. Growth Firms

It has been noted by Malkiel (2003) and Fama and French (1993) that in the long-term, value (low price to earnings (P/E) and price to book-value (P/BV) ratios) firms tend to generate larger returns than growth (high P/E and P/BV ratios) firms. In addition, Fama and French (1993) found there to be good explanatory power when the size and P/BV were used concurrently.

These arguments are powerful and could lead people to doubt the EMH and random walks. On the other hand, why are there investors with sophisticated tools if their efforts are futile?

2.1.4 Technical vs Fundamental Analysis

Investors and traders typically employ two classes of tools to decide what stocks to buy and sell; fundamental and technical analysis, both of which aim at analyzing and predicting shifts in supply and demand (Turner, 2007). Shifts in supply and demand are the basis of most economic and fundamental forecasting. If there are more sellers than buyers for a stock (i.e., increased supply), the theory states that the price should fall, and similarly, if there are more buyers than sellers (i.e., increased demand) the price should rise. Given the ability to foresee these shifts in supply and demand thus gives the trader the ability to establish profitable entry and exit positions, which is the ultimate goal of stock analysis.

While fundamental analysis involves the study of company fundamentals such as revenues and expenses, market position, annual growth rates, and so on, technical analysis is solely concerned with price and volume data, particularly price patterns and volume spikes (Turner, 2007). Price and volume data is readily available in real time, which makes technical analysis ideally suited for short-term swing trades. The underlying assumption in technical analysis is that stock prices evolve with certain regularity, forming reliable and predictable price and volume patterns that reveal market psychology which can be used to determine shifts in supply and demand (Turner, 2007).

2.1.5. Related Research

In this section several systems and research papers that have investigated the profitability of computerized technical analysis are presented.

Lo, Mamaysky and Wang (2000) found that "through the use of sophisticated nonparametric statistical techniques… [analysts] may have some modest predictive power" (Malkiel, 2003). Many within the business community also highlight Warren Buffet's ability to consistently beat the S&P index as an additional proof that the market can be predicted with an accuracy rate that exceeds the 50% threshold. Brock et al. (1992) describe an approach that employs two technical indicators (one of which is the moving average crossover rule discussed in detail in Section 3.2) to generate buy and sell signals. Profits generated by the signals are calculated by simulating transactions on the Dow Jones Industrial Average over 1897-1986. Their results shows that the system generate consistent returns.

Experienced analysts could apply some mathematical models that are proven based on the past data in order to evaluate company's intrinsic value, such as Graham number. Graham number and Graham's criteria is probably one of the most famous models (Graham, 1949). However, due to the increased volatility in the current market, it would be probably impossible to find a company that satisfies Graham's principles on today's stock exchanges. Because of these changes, the need for adjusted models rose. Also, stock market changes over time (Hendershott & Moulton, 2011). New investment strategies and new technology were introduced, which made some of the old models obsolete. Since financial literacy became higher, there are more market players than ever. However, for some of the old models cannot be easily adopted for the changes in stock market.

The introduction to algorithms in trading definitely changed the stock market. Algorithms made it easy to react fast to certain events on the stock market. Machine learning algorithms also enabled analysts to create models for predicting prices of stocks much easier. Introduction of machine learning caused that new models can be developed based on the past data. In this paper we will describe the method for predicting stock market prices using several machine learning algorithms.

2.1.6. Application of Machine Learning algorithms

Machine learning algorithms have been widely applied in many areas of finance. More specifically, ML techniques are common accepted to predict stock markets by means of a regression or classification problems. Usually, we have a quantitative output measurement (such as a stock price) or categorical (such as stock price goes up/down), that we wish to predict based on a set of variables, for example the stock prices of previous days or other indicators that could explain the final outcome. The use of ML algorithms allows us to build predictive models that can explain the relation between input and output variables on a set of training data. Considering a Supervised Learning approach, the agent is provided with known input-output values (labelled data) and tries to formulate a function to explain such relation.

Phua et al. performed a study predicting movement of major five stock indexes: DAX, DJIA, FTSE-100, HSI and NASDAQ. They used neural networks and they were able to predict the sign of price movement with accuracy higher than 60% by using component stocks as input for the prediction (Phua, et al., 2003).

Kim (Kim, 2003) trained SVM on daily time series from of Korean stock market. He reported a hit rate of around 56%.

Huang et al did a tried to use support vector machines in order to predict weekly movement of Japanese NIKKEI 225 index. Their approach achieved 73% hit rate with SVM and 75% with combined model. Also SVM outperformed in their approach backpropagation neural networks (Huang, et al., 2005).

2.2 Scope

The main objective of the project is to examine the important and potential factors/predictors that could impact the stock market by applying machine learning algorithms on historical past prices and implementation of various technical indicators as features.

Past research was based on various potential factors (e.g. economic, online news, social media) that could impact the stock market and acquired data from disparate data sources such as Yahoo Finance, Wikipedia, Google Trends.

To narrow our scope, we will only evaluate time-series data collected from Yahoo Finance and generate features such as technical indicators based on this data collected.

To predict stock movements we propose a data-driven approach that consists of three main phases. In Phase I, we scrape three sets of data from Yahoo Finance. These datasets include

(a) Historical prices collected from 2012-01-01 to 2017-12-17 of the S&P 500 index. S&P reflects US economy. Includes 500 top US companies in leading industries and captures 80% coverage of available market capitalization.

(b) The stock prices may be highly influenced by other major financial major indexes across the world. So we can include them as features and need to get data for them. Indexes considered are FTSE100, NIKKEI225, DJIA and NASDAQ.

(c) Exchange rates and commodity prices may also influence S&P500 so we take USD/CNY, USD/JPY,USD/EUR and USD/GBP, Gold, Oil.

In Phase II, commonly used technical indicators that reflect price variation over time are implemented and included as features to our training model.

In Phase III, we utilize 6 AI techniques to predict stock movement.

- Ada Boost
- Logistic Regression
- Support Vector Machine
- Random Forest
- K Nearest Neighbors
- Gaussian Naïve Bayes

In Phase IV, the models are compared and evaluated based on 10-fold cross validation sample using the area under the operating characteristics curve (AUC).

2.3 Tools required

The whole project is based will be implemented on Python 3.6 on Anaconda. For Data analysis, R and Python are the two most popular tools. However, Python is faster and much more comfortable in handling very large datasets by the recent inclusion of Panda's data frames. While using Python, the data analysts can use Pandas to aggregate, manipulate, and visualize relational data. Likewise, they can use Seaborn and Matplotlib to visualize statistical models.

Anaconda is a free and open source distribution of the Python and R programming languages for data science and machine learning related applications that aims to simplify package management and deployment.

Anaconda has built-in web based user interface for project such as Jupyterlab which is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.

To view and test the project it is recommended to first download Anaconda with the latest Python 3 version using Anaconda Distribution which includes Python, Jupyter Notebook and other commonly used packages for data science.

The important python libraries used in this project are:

pandas, numpy, data (from pandas_datareader import data as pdr) , datetime, fix_yahoo_finance (pip install fix_yahoo_finance), matplotlib, functools, pandas_talib (git clone https://github.com/femtotrader/pandas_talib.git) , seaborn and MinMaxScalar (from sklearn.preprocessing import MinMaxScaler).

2.4 Objectives

- Can S&P be predicted to some degree using Technical Analysis?
- Which target (Open, Close or Return) is most suitable for prediction?
- Which technological model is best at predicting the stock movement?
- Do the models predictions vary across different target?

Chapter 3

Data Collection

The S&P 500 Index data is first collected for the period of 2012-01-01 to 2017-12-17. To see how S&P is affected by other major world indexes we include them as feature as well. However it is important to ask which other world indexes should we include?

There are many works that deal with international financial markets and their relations to one another, and most of these are based on correlation. However, correlation is non causal and symmetric and is, therefore, not able to shed light on how Index fluctuation on one region of the world can effect on other part (Raddant, Kenett, 2017). Finding dependencies between world stock markets is a field that requires extensive statistical analysis which is not included in this project.

The correlation matrix on the following page compares the percentage change of the closing price of S&P Index with various other world indexes. It suggests that S&P Index is more positively correlated with other US indexes compared to stock exchanges across Asia and Europe.

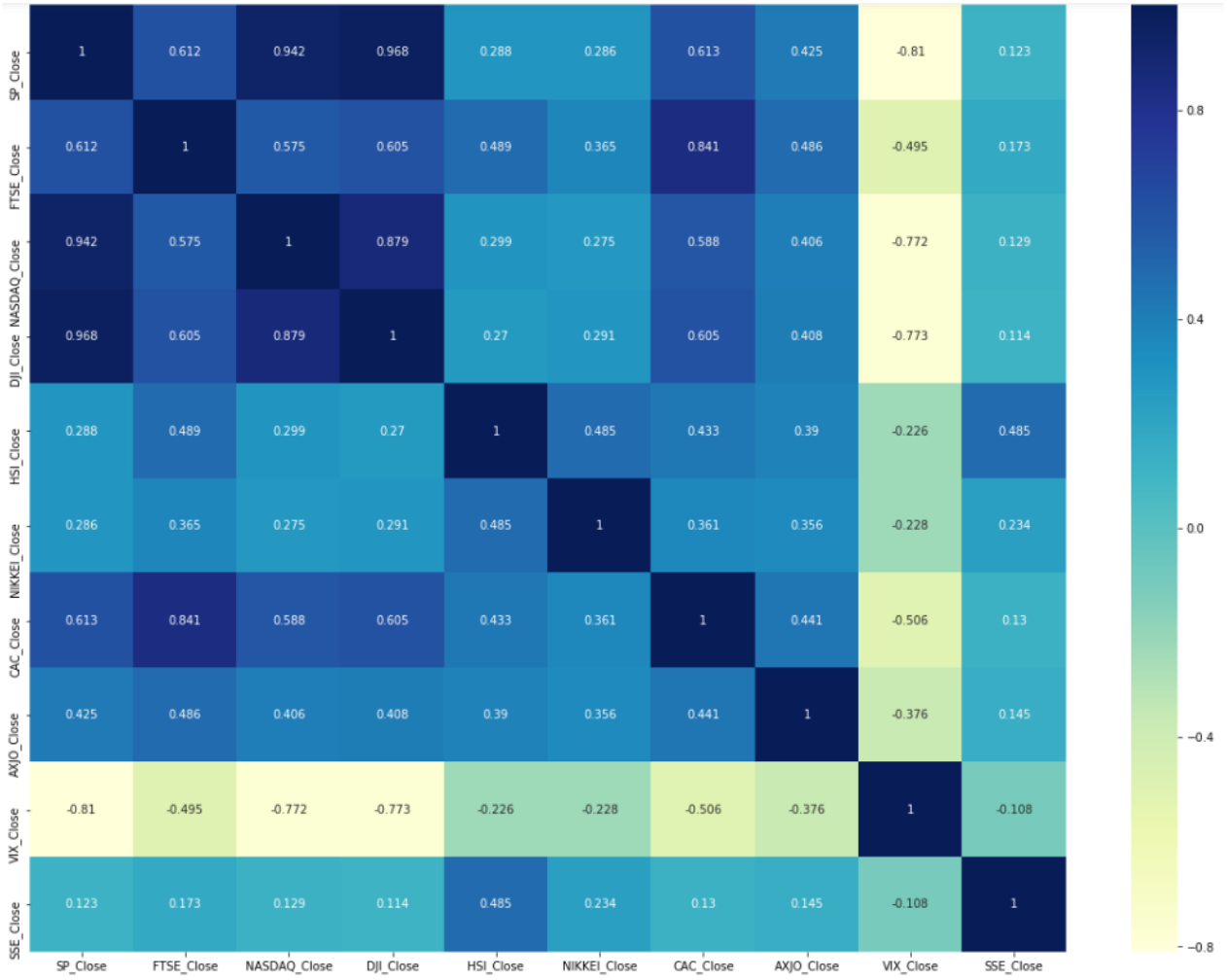

Figure 1: Percentage change of SP_Close is plotted with other major world indexes (FTSE:London, NASDAQ: US, DJI:US, HIS: Hong Kong, NIKKEI: Japan, CAC: France, AXJO: Australia, SSE: Shanghai) to show the correlation of their change. S&P is more positively correlated with other US indexes (closer to 1, dark blue) compared to Europe & Asian Indexes (lowest is Shanghai Stock Exchange followed by Hong Kong and Japan)

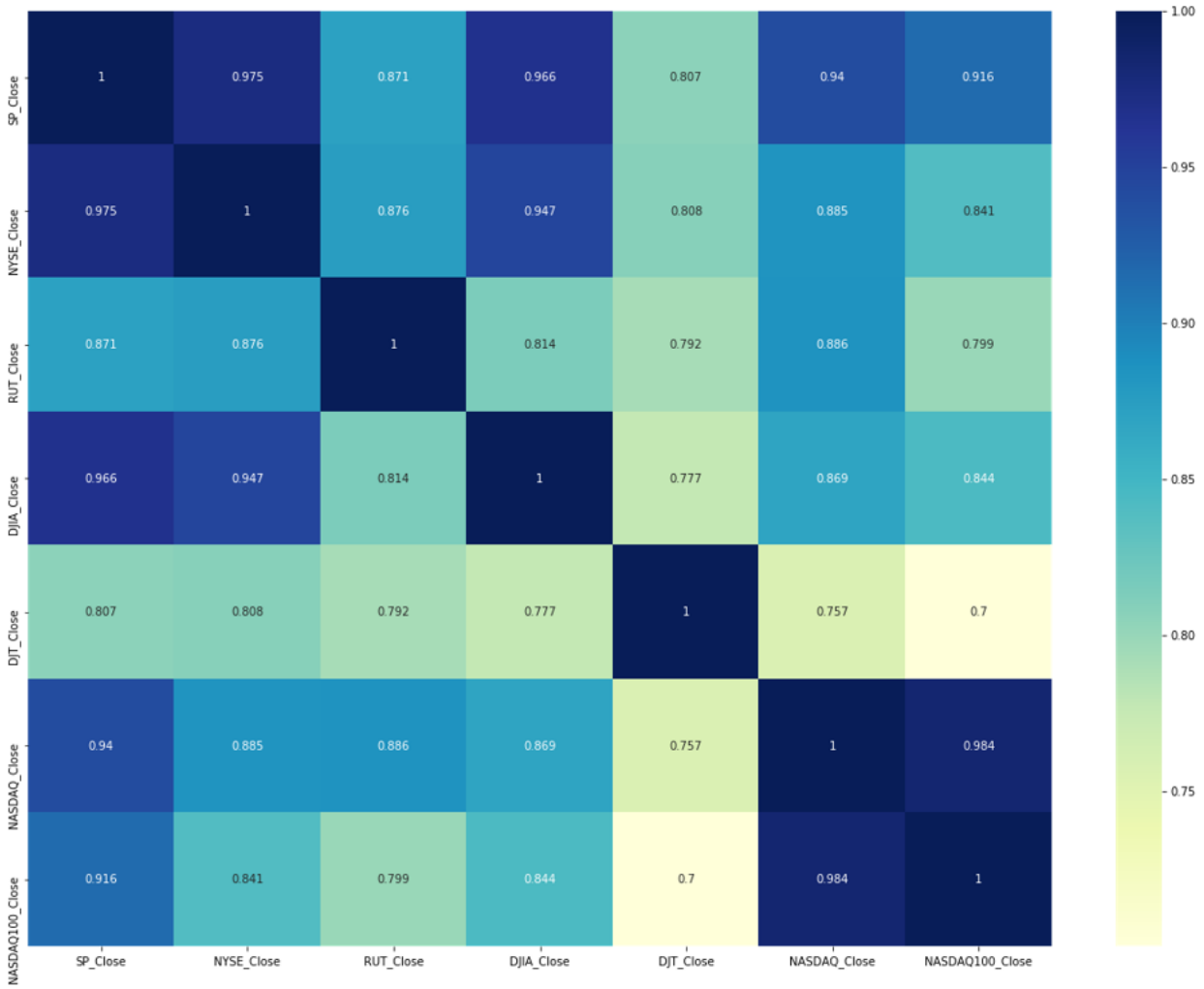

Figure 2: The correlation matrix of S&P Index with other US indexes only. This matrix shows higher correlation (lowest being 0.87 with DJT_Close) for S&P Index compared to Figure 1.

It is wise to include Indexes only from the US for our project. However from the above figure it suggests, other world indexes are still positively correlated with that of S&P. In the highly interconnected global financial system, local financial shocks and events can be easily amplified and turned into global events.

I decide to include the following world indexes for this project:

FTSE : The Financial Times Stock Exchange 100 Index is a share index of the 100 companies listed on the London Stock Exchange with the highest market capitalization.

NIKKEI : The Nikkei 225, more commonly called the Nikkei, the Nikkei index, or the Nikkei Stock Average, is a stock market index for the Tokyo Stock Exchange.

DJIA : The Dow Jones Industrial Average, or simply the Dow, is a stock market index that shows how 30 large, publicly owned companies based in the United States have traded during a standard trading session in the stock market.

NASDAQ : The NASDAQ Composite is a stock market index of the common stocks and similar securities listed on the NASDAQ stock market. Along with the Dow Jones Average and S&P 500 it is one of the three most-followed indices in US stock markets.

The data collected are each day Open, High, Low, Adjusted Close and Volume for the period 2012-01-01 to 2017-12-17.

Exchange rates may have an effect on the S&P Index as well. The S&P 500, the benchmark US share index, rose 10% to new all-time highs in the month following Trump's election on 9 November. There are many factors that can influence stock markets, but the strength of the US dollar, which gained 3% during the same period, was a key driver. (David Brett, 2017).

As the chart below illustrates, more than 70% of the revenues generated by US companies listed on the S&P come from the US. France is next biggest contributor, followed by the UK.

The inner ring illustrates that 99.8% of the companies on the S&P 500 have registered head offices (i.e. they are domiciled) in the US.

The outer ring shows the regions from which S&P 500 companies generate their revenues. All but 29.6% are generated inside the US.

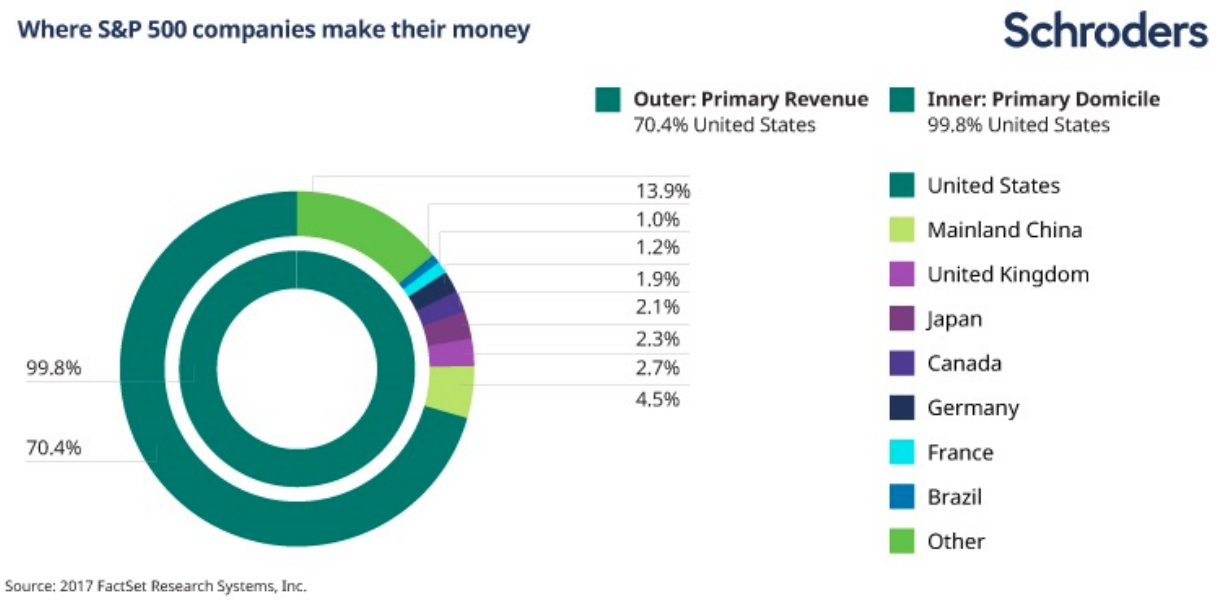

Figure 3: 29.6% of S&P Companies revenue comes outside US. Even though this may not seem like much it is inevitable that exchange rates of US plays a part in the prices of stock for S&P.

The following exchange rates data are collected from FRED (Federal Reserve economic data) to be added as features:
USDJPY, USDCNY, USDGBP, and USDEUR.

Recent study suggests that Oil and gold prices and price volatilities have extensive impacts on economic and financial activities of the US (Korhan, 2015). So the following commodity prices are also included in our dataset:
Gold, Oil.

All the datasets are collected by using the Pandas Datareader. It is now important to merge these datasets into one single data frame and also to clean the dataset by removing any row with missing places of data.

3.1 Data Merging

We merge closing values of other indexes, commodity and exchange rates with S&P Data in one single DataFrame. The closing values of other indexes are renamed according to their particular index/exchange rate/commodity so we can identify different Close when we merge the data.

3.2 Data Cleaning

Data rows with missing values are dropped.

	Open	High	Low	SP_Close	Adj Close	Volume	FTSE_Close	NIKKEI_Close	DJI_Close	NASDAQ_Close	USDJPY_Close
2012-01-04	1277.030029	1278.729980	1268.099976	1277.300049	1277.300049	-702387296	5668.500000	8560.110352	12418.419922	2648.360107	76.68
2012-01-05	1277.300049	1283.050049	1265.260010	1281.060059	1281.060059	20982704	5624.299805	8488.709961	12415.700195	2669.860107	77.18
2012-01-06	1280.930054	1281.839966	1273.339966	1277.810059	1277.810059	-638137296	5649.700195	8390.349609	12359.919922	2674.219971	77.06
2012-01-10	1280.770020	1296.459961	1280.770020	1292.079956	1292.079956	-73007296	5696.700195	8422.259766	12462.469727	2702.500000	76.84
2012-01-11	1292.020020	1293.800049	1285.410034	1292.479980	1292.479980	-326847296	5670.799805	8447.879883	12449.450195	2710.760010	76.90

Figure 4: Pandas dataframe showing the merged data with that of S&P after data cleaning.

Chapter 4

Feature Generation

4.1 Data Normalization

Data obtained from the above resources is the absolute daily price information. It is also important to note, stocks of NIKKEI and FTSE are traded in different time zones than that of DJIA, NASDAQ and S&P. To address the above issues, we transform daily data into daily return by calculating the percentage change using this formula.

$$r_n = \frac{P_n - P_{n-1}}{P_{n-1}} * 100$$

where r_n is the return price at day n and P_n and P_{n-1} is the closing price at day n and n-1 respectively.

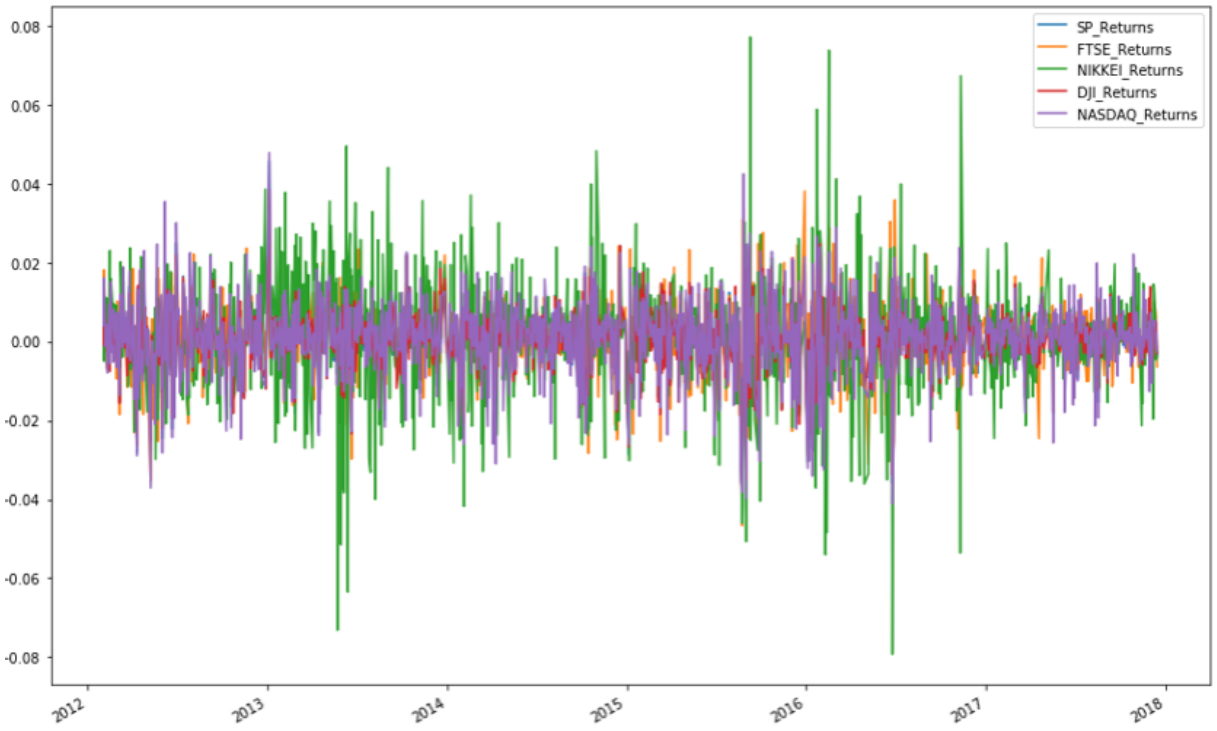

Figure 5: A matplotlib showing the daily percentage change of the Indexes included in this project

4.2 Technical Indicators

In addition to major indexes and commodities and exchange rate we introduce several technical indicators as features: Pandas.series API is used extensively for calculating the below well renowned technical indicators:

- **Relative Strength Index (RSI)**

It compares the magnitude of recent gains to recent losses in an attempt to determine overbought and oversold conditions of an asset. RSI ranges from 0 to 100. In practice, investors sell if its value is ≥ 80 and buy if it is ≤ 20.

The relative strength index is calculated using the following formula: (https://www.investopedia.com/terms/r/rsi.asp)

RSI = 100 - 100 / (1 + RS)

Where RS = Average gain of up periods during the specified time frame / Average loss of down periods during the specified time frame

The specified time frame used is 14 days.

In Python: (https://dataanalysiswithpandas.blogspot.hk/2016/08/technical-indicator-with-pandas-and.html)

- **Commodity Channel Index (CCI)**

The Commodity Channel Index (CCI) is a momentum based technical trading tool used most often to help determine when an investment vehicle is reaching a condition of being overbought or oversold. (https://www.investopedia.com/terms/c/commoditychannelindex.asp)

The Commodity Channel Index is computed with the following formula:

CCI = (Typical Price - 20-period Moving Average of TP) / (.015 x Mean Deviation)

Typical Price (TP) = (High + Low + Close)/3

(http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:commodity_channel_index_cci)

In python: (http://www.andrewshamlet.net/2017/07/08/python-tutorial-cci/)

- **Accumulation/Distribution (ACCDIST)**

Accumulation/distribution is a momentum indicator that attempts to gauge supply and demand by determining whether investors are generally "accumulating," or buying, or "distributing," or selling, a certain stock by identifying divergences between stock price and volume flow. The accumulation/distribution is calculated by first calculating the money flow multiplier, and then multiplying the money flow multiplier by the period's volume. (https://www.investopedia.com/terms/a/accumulationdistribution.asp)

Money Flow Multiplier = [(Close - Low) - (High - Close)] /(High - Low)

Money Flow Volume = Money Flow Multiplier x Volume for the Period

ACCDIST = Previous ACCDIST + Current Period's Money Flow Volume

(http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:accumulation_distribution_line)

In python: (https://github.com/Crypto-toolbox/pandas-technical-indicators/blob/master/technical_indicators.py)

- **Momentum (MOM)**

The Momentum (MOM) indicator compares the current price with the previous price from a selected number of periods ago. This indicator is similar to the "Rate of Change" indicator, but the MOM does not normalize the price, so different instruments can have different indicator values based on their point values.

MOM = Close - Close of n days ago

In my project I take 2 values of n = 3, 5.

In Python: (https://github.com/Crypto-toolbox/pandas-technical-indicators/blob/master/technical_indicators.py)

- **Rate of Change (ROC)**

The Rate-of-Change (ROC) indicator is a pure momentum oscillator that measures the percent change in price from one period to the next.

*ROC = [(Close - Close n days ago) / (Close n days ago)] * 100*

In my project I take 3 values of n = 3, 5, 7

In python: (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.pct_change.html)

- **Simple Moving Average (MA)**

A simple moving average is formed by computing the average price of the index over a specific number of periods.

In Python: (http://pandas.pydata.org/pandas-docs/version/0.17.0/generated/pandas.rolling_mean.html)

- **Exponential Moving Average (EMA)**

Exponential moving averages (EMAs) reduce the lag by applying more weight to recent prices. The weighting applied to the most recent price depends on the number of periods in the moving average.

Calculation:

Initial Simple Moving Average: 10-period sum / 10

Multiplier: (2 / (Time periods + 1)) = (2 / (10 + 1)) = 0.1818 (18.18%)

EMA: {Close - EMA(previous day)} x multiplier + EMA(previous day).

(http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:moving_averages)

In Python: (http://pandas.pydata.org/pandas-docs/version/0.17.0/generated/pandas.ewma.html)

- **Standard Deviation (STDDEV)**

Standard deviation is an indicator that measures the size of recent price moves of index, to predict how volatile the price may be in future.

In Python: (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.std.html)

In my project I use the requested axis as 10 and 20 days.

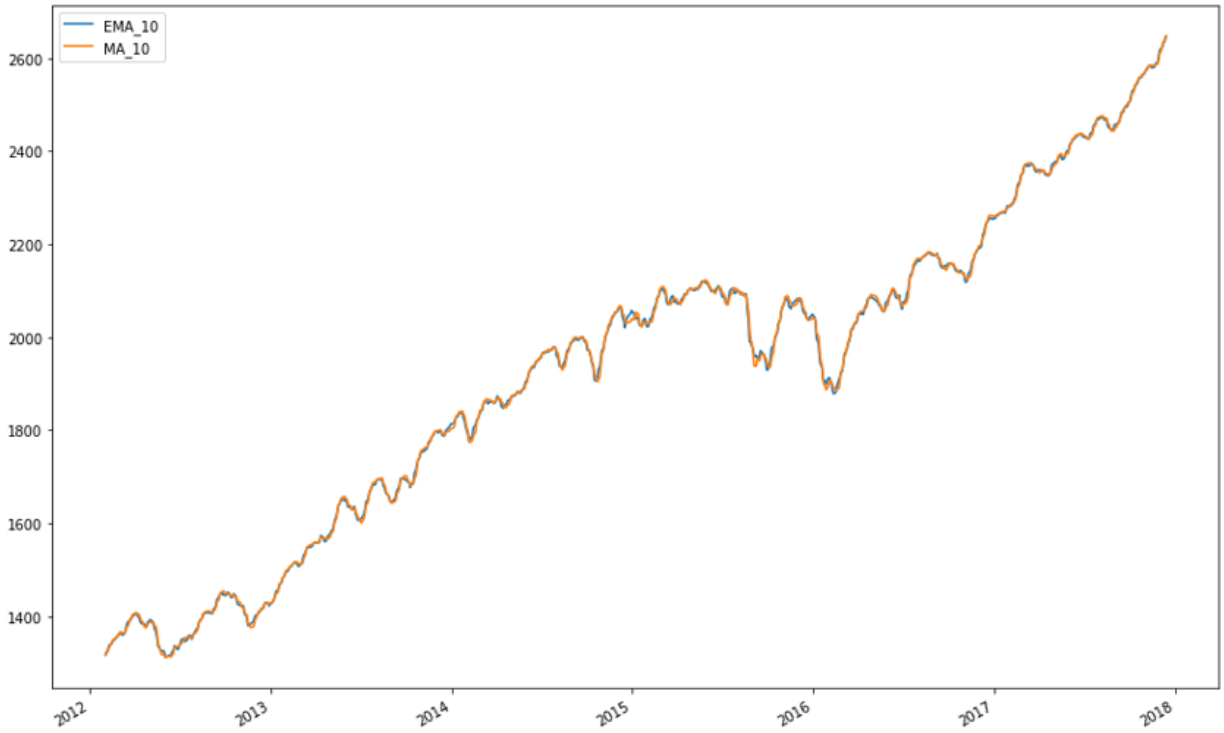

Figure 6: A matplotlib showing the EMA and MA over 10 days period for the S&P Index

4.3 Target Variable

The end goal of this phase is to have the data processed for the artificial intelligence models. This phase is ended by defining different one-day-ahead outcomes (hereafter targets).

There are several one-day-ahead outcomes that can be of interest to investors (Bin Weng, 2017). I define three target variables:

- Target 1: Close(i+1) - Close(i)
  It compares the closing price of day i+1 with the closing price of the previous day. If Close(i+1) > Close (i) -> +1 else -1

- Target 2: Open(i+1) - Open(i)
  It compares the opening price of day i+1 with the opening price of the previous day. If Open(i+1) > Open (i) -> +1 else -1

- Target 3: if Return > 0.0015 -> +1 else -1
  Target 3 will be explained further in the next section.

In summary, the Target variables will be added to the dataframe. The machine learning algorithms will then predict the Target variables for the Test dataset (explained in the next section) after given the features and Target variables of the Training dataset.

Machine learning algorithms are described as learning a target function (f) that best maps input variables (X) to an output variable (Y): ***Y = f(X)***
(X) = features
(Y) = Target 1, 2 or 3.

After Data processing, the data frame is transformed containing the daily returns and target variable assigned to -1 or 1 (downward state and upward state).

MOM_3	...	NASDAQ_Returns	USDJPY_Returns	USDCNY_Returns	USDGBP_Returns	USDEUR_Returns	Gold_Returns	Oil_Returns	Target1	Target2	Target3
12.530029	...	0.004006	0.000263	-0.000935	-0.002271	-0.000911	0.006322	-0.013008	1.0	1.0	1.0
32.489990	...	0.016079	0.005386	0.000159	-0.002150	-0.004633	-0.009709	0.014944	-1.0	1.0	-1.0
20.239990	...	-0.001263	0.000000	0.001476	0.003359	0.001831	-0.008651	-0.009305	1.0	1.0	1.0
21.510010	...	0.000720	0.004442	-0.001172	0.003727	0.008682	0.002909	0.017133	1.0	1.0	1.0
5.059937	...	0.004056	0.000260	-0.001713	-0.004468	0.000680	0.012761	0.002537	1.0	1.0	-1.0

Figure 7: Dataframe at the end of Phase II showing the Target variables and the returns.

Chapter 5

Model Construction

In this phase scikit-learn library is used to implement the Machine Learning models which features various classification and regression algorithms such as Support Vector Machine, Random Forest etc.

5.1 Defining the Machine Learning Models

5.1.1 Support Vector Machine

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection. SVM finds a "large margin" so that the decision boundary stays as far away from the closest samples as possible.

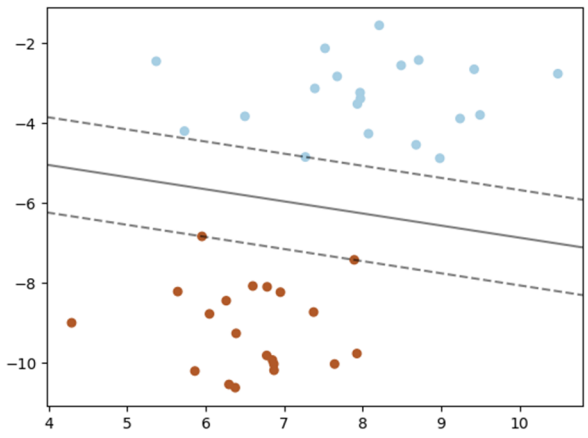

Figure 8: A Hyperplane showing the margin staying as far away from the closest samples as possible (Red = negative hyperplane or -1, Blue = positive hyperplane or +1). The aim is to maximize the margin.

The *Kernel* function to be used is 'Linear' since it has more flexibility in the choice of penalties and loss functions and should scale better to larger number of samples.

C value to be used is [0.01, 0.1, 1.0, 10.0, 100.0]. A lower value of C helps when there are noisy observations and it corresponds to regularize more the estimation.

5.1.2. Random Forest and Adaptive Boosting (AdaBoost) Classifier

Random Forest and AdaBoost are a set of ensemble learning method where the goal is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability over a single estimator.

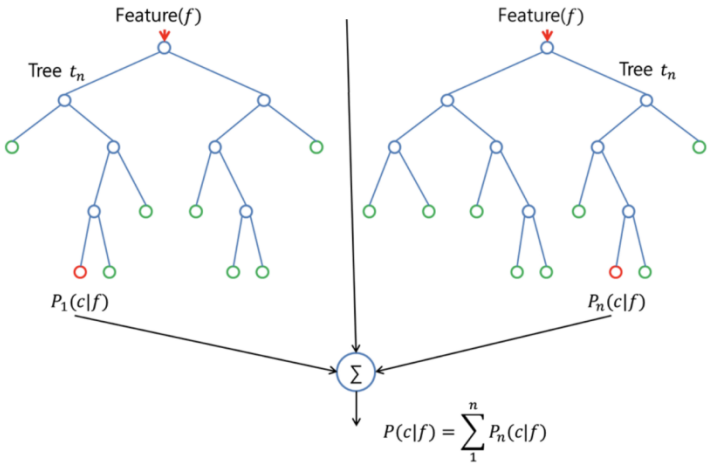

Figure 9: In Random Forest a multitude of decision trees at training time is constructed and outputting the class that is the mode of the classes of the individual trees. The idea is that no single classifier is best for all circumstances and there multiple classifiers are combined and taken a majority vote (P(c|f).

In Random forest, each tree in the ensemble is built from a sample drawn with replacement from the training set. When splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features. As a result of this randomness, the bias of the forest usually slightly increases but, due to averaging, its variance also decreases, usually more than compensating for the increase in bias, hence yielding an overall better model.

The main parameter to adjust when using Random Forest is *n_estimators* which is the number of trees in the forest. The larger the better but also the longer it will take to compute. The n-estimators to be used are [10, 20, 50, 100].

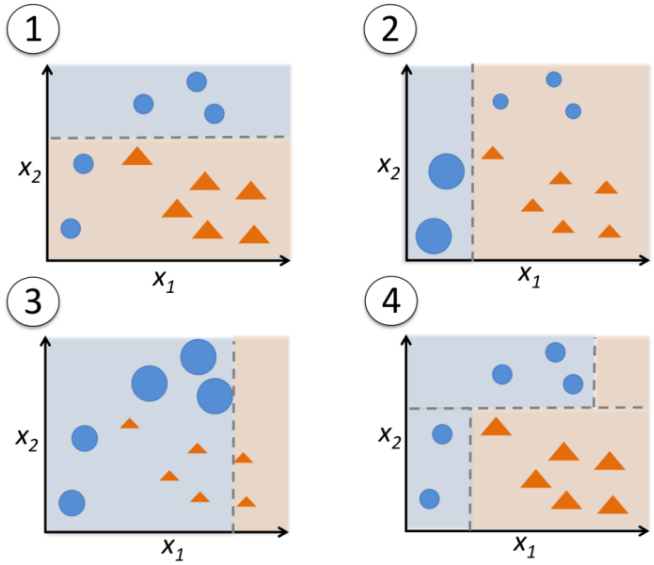

Figure 10: In AdaBoost, previously misclassified samples larger weights. Steps: 1) equal weight training of all samples by C1, two blue circles are misclassified. 2) larger/lower weights to wrongly/correctly classified samples, train C2. 3) larger/lower weights to wrongly/correctly classified samples, train C3. 4) combine C1, C2, C3 for weighted majority voting

The core principle of AdaBoost is to fit a sequence of weak learners (i.e., models that are only slightly better than random guessing, such as small decision trees) on repeatedly modified versions of the data. The predictions from all of them are then combined through a weighted majority vote (or sum) to produce the final prediction.

The number of weak learners is controlled by the parameter *n_estimators*.
The *learning_rate* parameter controls the contribution of the weak learners in the final combination.
n_estimators [10,20,50] and learning_rate [0.8,1.0,1.2] are used.

5.1.3. Logistic Regression

Logistic regression is a linear model for classification rather than regression. The target value is expected to be a linear combination of the input variables. In mathematical notion, if $\hat{y}$ is the predicted value.

$$\hat{y}(w, x) = w_0 + w_1 x_1 + ... + w_p x_p$$

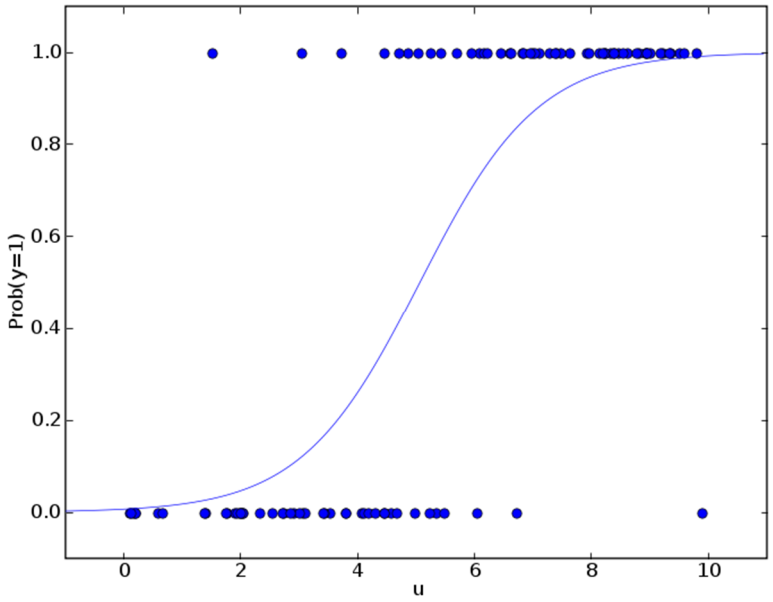

Figure 11: The Logistic regression model is used to estimate the probability of a binary response based on one or more predictor (or independent) variables (features). It allows one to say that the presence of a risk factor increases the odds of a given outcome by a specific factor. The model itself simply models probability of output in terms of input.

Like in support vector machines, smaller value of parameter *C* specifies stronger regularization. The *C* values used are [0.001, 0.01, 0.1, 1.0, 10.0, 100.0].

5.1.4. k Nearest Neighbour

Nearest neighbour (NN) classifiers assign an input example the class (label) of similar labelled examples. The algorithm requires a training dataset made up of examples that have been classified into several categories.

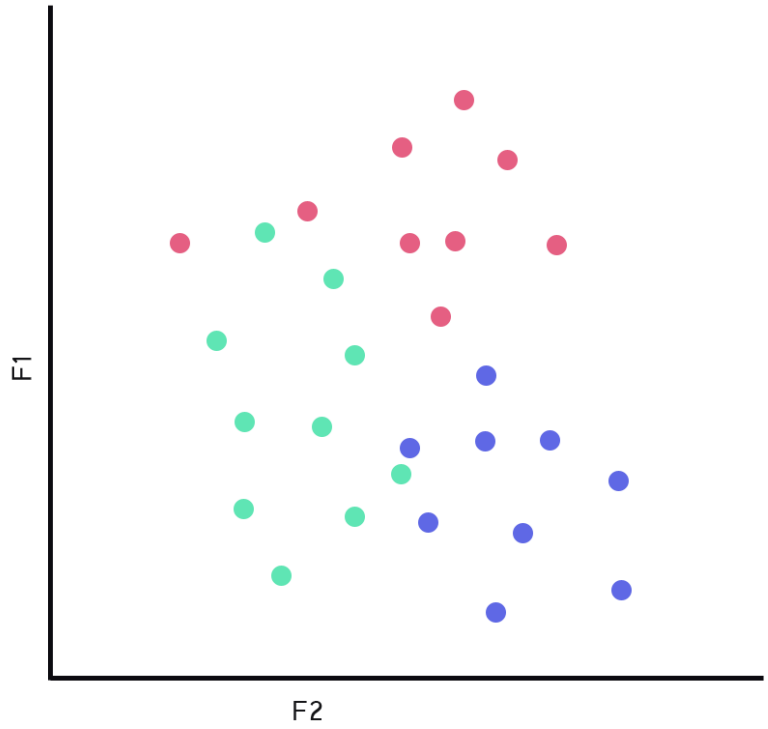

Figure 12: For each unlabelled record in the test dataset, k-NN identifies k records in the training data that are the "nearest". The unlabelled test instance is assigned the class of the "majority" of the k nearest neighbours.

The value of k determines how well the model will generalize to future data. Choosing a large k reduces the impact or variance caused by noisy data, but can bias the learner so that it runs the risk of ignoring small, but important patterns. The parameter *n_neighbors* defines the number of neighbors to use. Value of *n_neighbors* selected are [3,5,7,9,11,13].

5.1.5 Gaussian Naïve Bayes (GNB)

k-NN uses neighboring points to classify/label an input point . Sometimes it is not that clear cut that a data point falls entirely into one particular category. GNB assumes that all features in a dataset are independent and equally important.

5.2 Cross validation and Grid-Search

Each of our learning models split the dataset into Train and Test set. The *training set* is 2012-01-01 to 2016-12-31 while the *test set* is 2017-01-01 to 2017-12-17.

The following function separates the test set and train set:

```
def mask(df, features,y,start_test):
    '''
    To separate dataset and get X_train, X_test, y_train, y_test
    df=dataframe
    y = target 1 , 2 or 3
    start_test = 2017-01-01, i.e. split data until 2017-01-01 as training and the remaining as test
    '''
    X=df[features]
    y=df[y]
    X_train = X[X.index < start_test]
    y_train = y[y.index < start_test]
    X_test = X[X.index >= start_test]
    y_test = y[y.index >= start_test]
    return X_train, y_train, X_test, y_test
```

When evaluating different settings ("hyperparameters") for estimators, such as the C setting that must be manually set for an SVM, there is still a risk of overfitting *on the test set* because the parameters can be tweaked until the estimator performs optimally. This way, knowledge about the test set can "leak" into the model and evaluation metrics no longer report on generalization performance. To solve this problem, we implement Scikit-Learn's *cross-validation* feature.

K- fold cross validation: If K=10, randomly split the training set into K distinct subsets called folds, then train and evaluates the K times, each using a different fold for evaluation and training on the other 9 folds. In general K=10 is most widely used.

The following function is used to cross-validate the training set:

```
def CV(X_train, y_train, folds, method, parameter):
    '''
    To get the average score of cross validation

    '''
    k = int(np.floor(float(X_train.shape[0])/folds))
    acc = np.zeros(folds-1)
    for i in range(2, folds+1):
        split = float(i-1)/i
        data = X_train[:(k*i)]
        output = y_train[:(k*i)]
        index = int(np.floor(data.shape[0]*split))
        X_tr = data[:index]
        y_tr = output[:index]
        X_te = data[(index+1):]
        y_te = output[(index+1):]
        acc[i-2] = classifier(X_tr, y_tr, X_te, y_te, method, parameter)[1]
    return acc.mean()
```

Cross validation iterators can be used to directly perform model selection using Grid Search for the optimal hyperparameters of the model.

The below function searches for the best parameter:

```
def SearchGrid(X_train, y_train, folds, method, grid):
    '''
    To get the best parameters for a given classifier
    '''
    param = list(grid.keys())
    finalGrid = {}
    if len(param) == 1:
        for value_0 in grid[param[0]]:
            parameters = [value_0]
            accuracy = CV(X_train, y_train, folds, method, parameters)
            finalGrid[accuracy] = parameters
        final = sorted(finalGrid.items(), key=operator.itemgetter(0), reverse=True)
        return final[0]
    elif len(param) == 2:
        for value_0 in grid[param[0]]:
            for value_1 in grid[param[1]]:
                parameters = [value_0, value_1]
                accuracy = CV(X_train, y_train, folds,method, parameters)
                finalGrid[accuracy] = parameters
        final = sorted(finalGrid.items(), key=operator.itemgetter(0), reverse=True)
        return final[0]
```

Chapter 6

Results and Discussion

Back to our objectives:

- Can S&P be predicted to some degree using Technical Analysis?
- Which target (Open, Close or Return) is most suitable for prediction?
- Which technological model is best at predicting the stock movement?
- Is there one single ML model that successfully predicts across all target variable? One that outperforms all other?

To help get to our objectives; we implement a tool for classification called the Receiver Operating Characteristic curve or ROC. In the following section a short description to explain what ROC represents.

Receiver Operating Characteristic curve (ROC)

In our project we defined three Target variables holding the value of either -1 or 1 (downstate or upstate).

The algorithm maps our input variables to this target variable and that is the whole basis of prediction. There can only be two predicted condition: positive or negative.

If the predicted condition is positive it can be either a True Positive (predicted value is right) or False Positive (predicted value is wrong).

If the predicted condition is negative it can be either a True Negative (predicted value is right) or False Negative (predicted value is wrong).

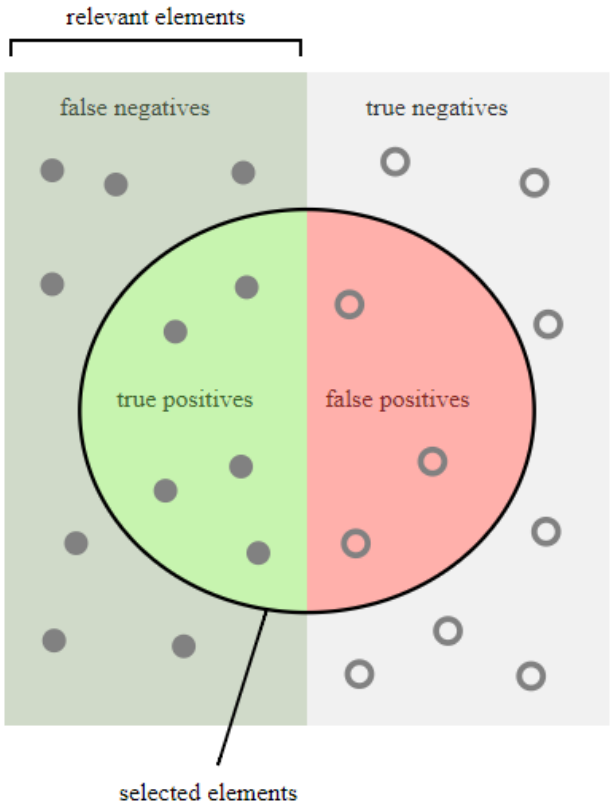

Figure 13: The possible instances of prediction of the ML algorithms

The ROC curve plots True positive rate vs. False positive rate at various threshold settings. Therefore, True Positive Rate (TPR) is the hit rate or sensitivity and False Positive Rate (FPR) is the fall-out or (1-specificity). The TPR defines how many correct positive results occur among all positive samples available during the test. FPR, on the other hand, defines how many incorrect positive results occur among all negative samples available during the test.

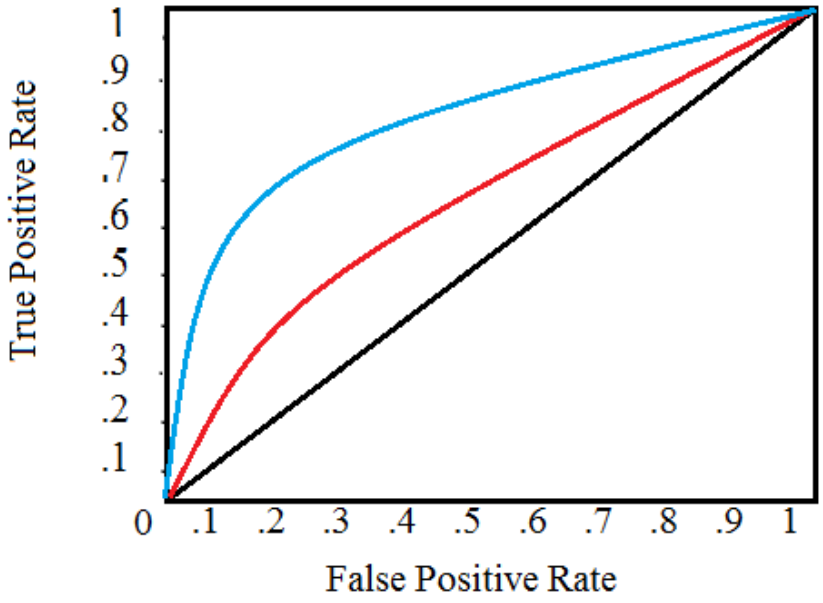

Figure 14: A sample ROC curve

From figure 14, a ROC curve demonstrates several things:

1. The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the test.
2. The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test.
3. The area under the curve is a measure of test accuracy. An area of 1 represents a perfect test; an area of 0.5 represents a test similar to a coin-flip.

The Area Under Curve (AUC) is equal to the probability that a classifier will rank a randomly chosen positive chance as positive, in other words, higher than a randomly chosen negative chance.

We implement ROC curve and find the AUC score for each of our target variables using varying instances of selected features:

- All variables are selected as features including technical indicators, index returns, stock price, etc
- Only technical indicators are selected as features
- Only Index Returns and Gold and Oil returns are selected as features.

The results are shown in the following section:

Target 1: Close(i+1) - Close(i)

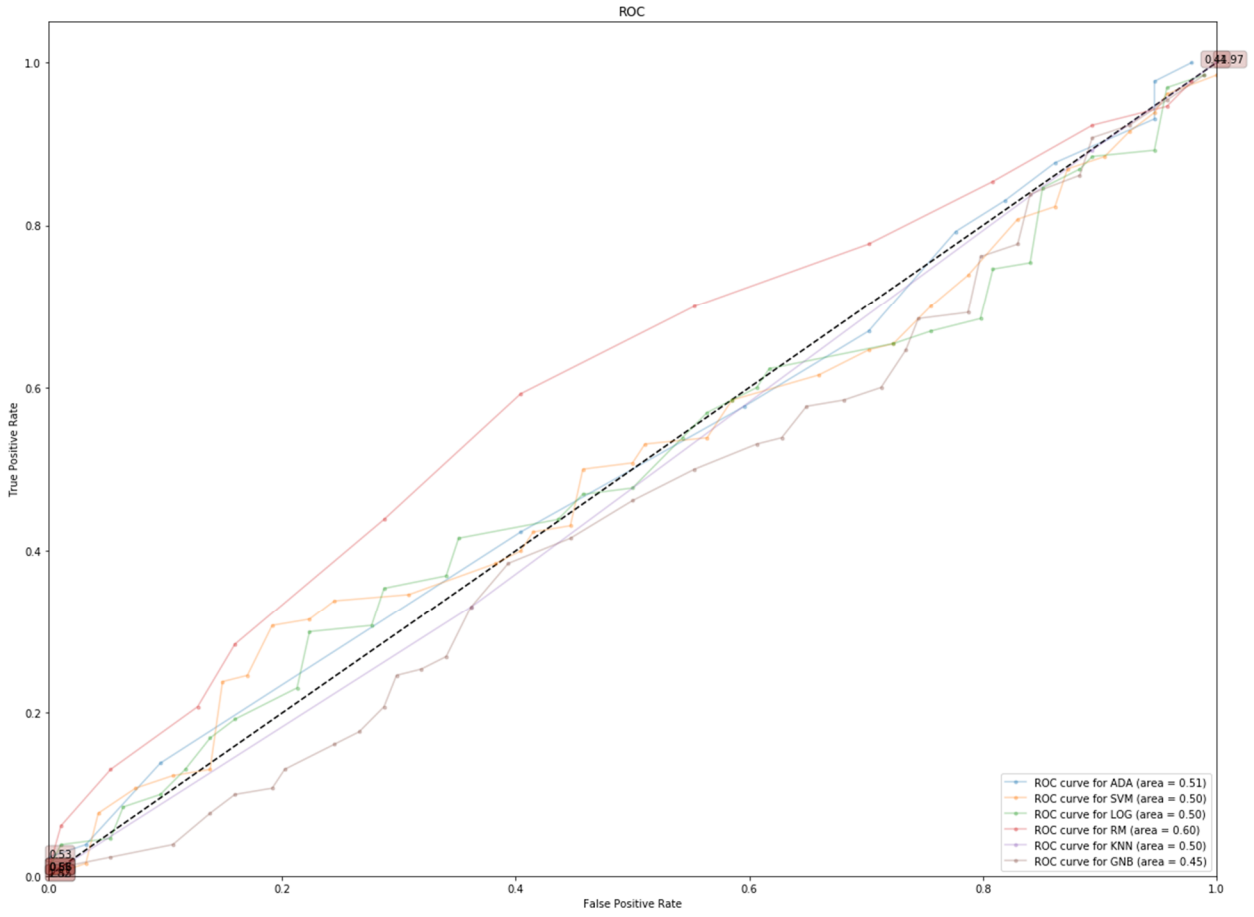

Figure 15: ROC Curve, Target 1, All features selected.

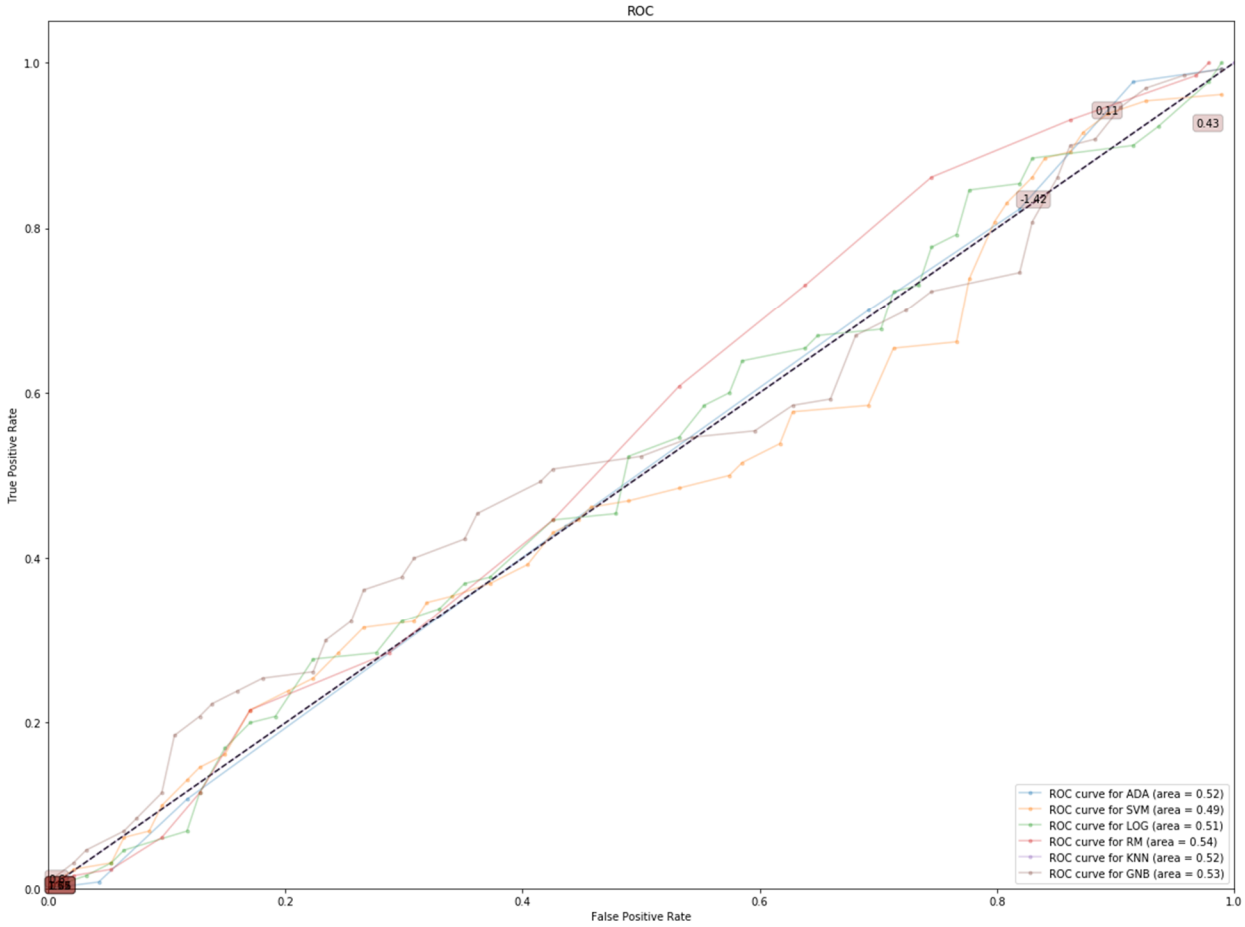

Figure 16: ROC Curve, Target 1, Technical Indicators as features.

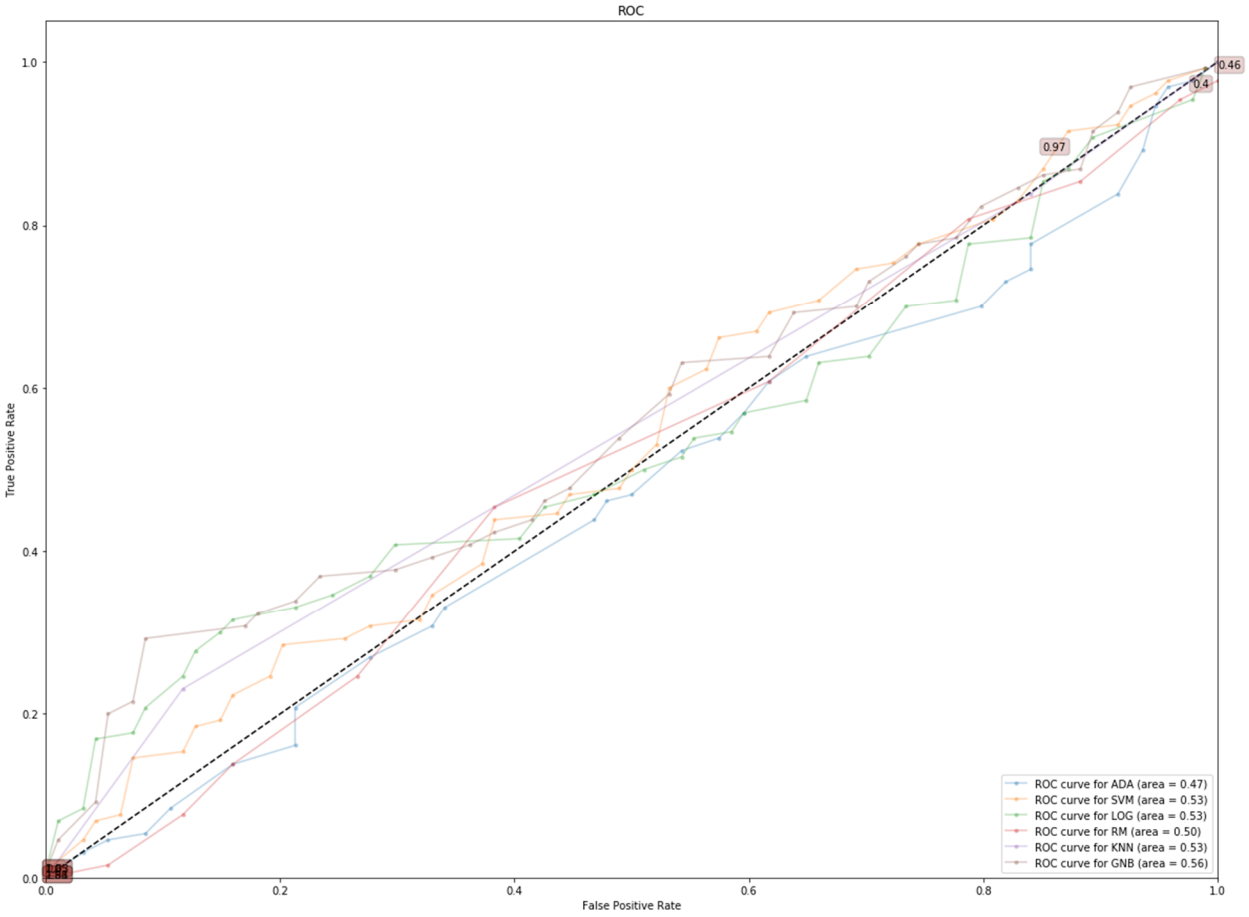

Figure 16: ROC Curve, Target 1, Index Returns and Gold and Oil Returns as features.

Target 2: Open(i+1) - Open(i)

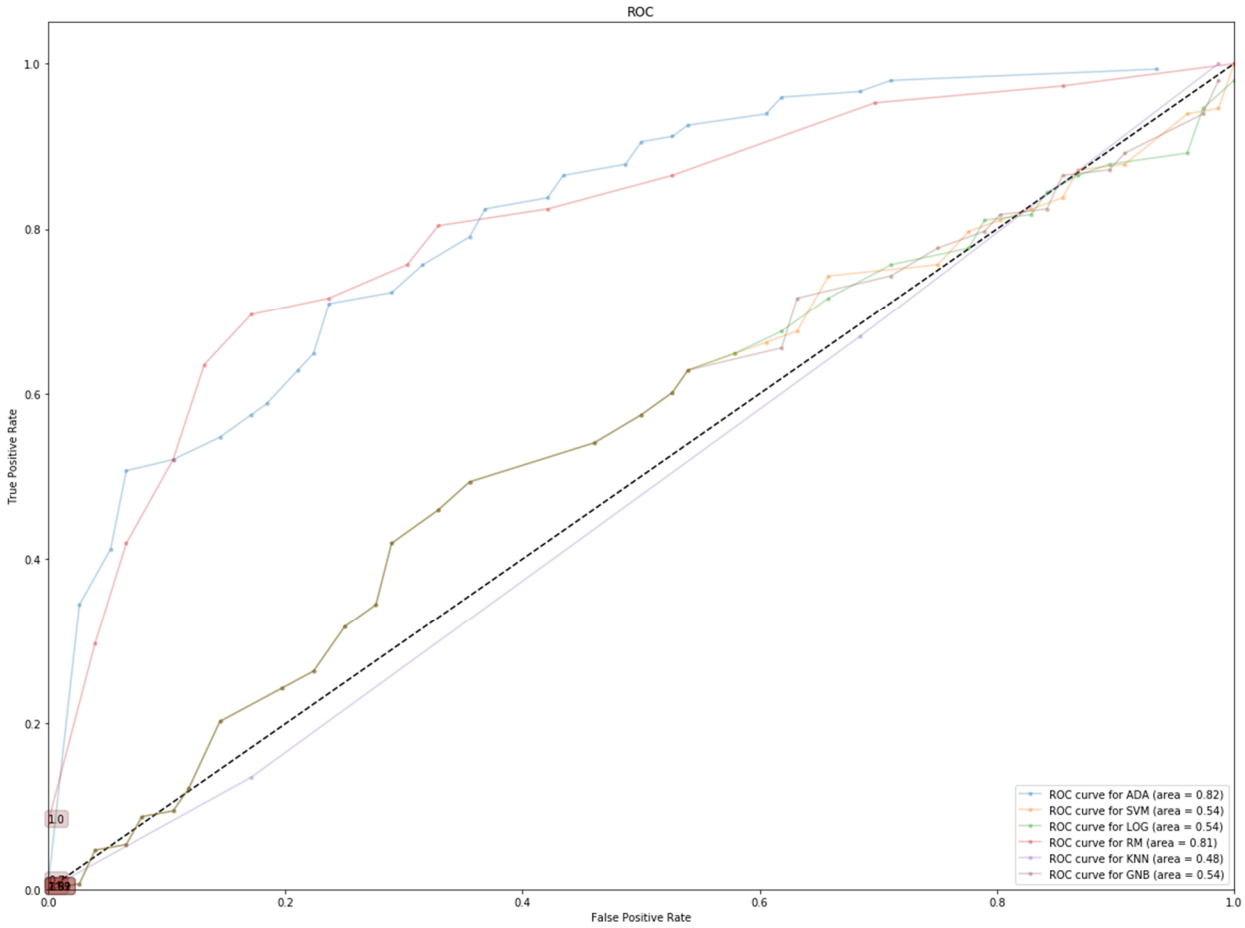

Figure 17: ROC Curve, Target 2, all features selected.

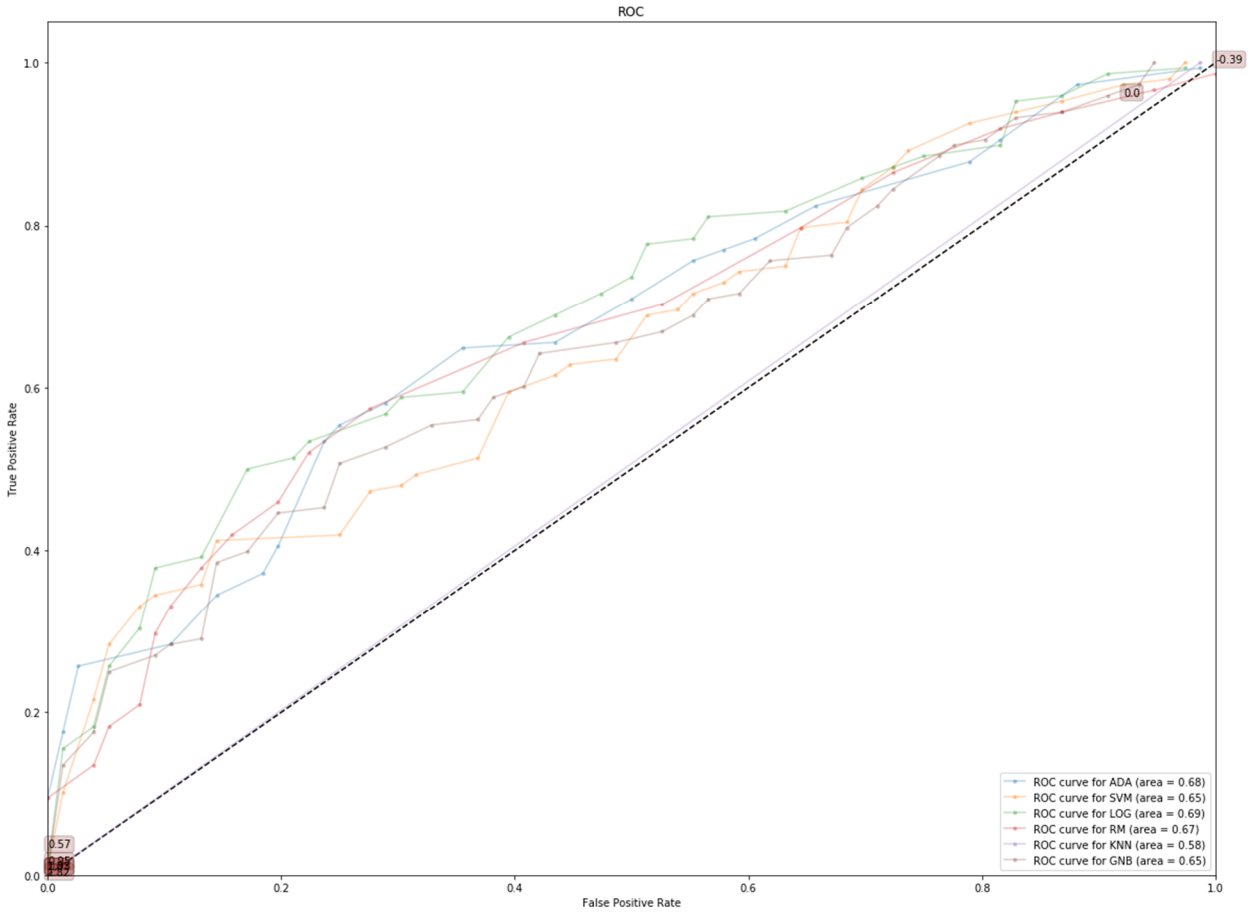

Figure 18: ROC Curve, Target 2, Technical Indicators as features.

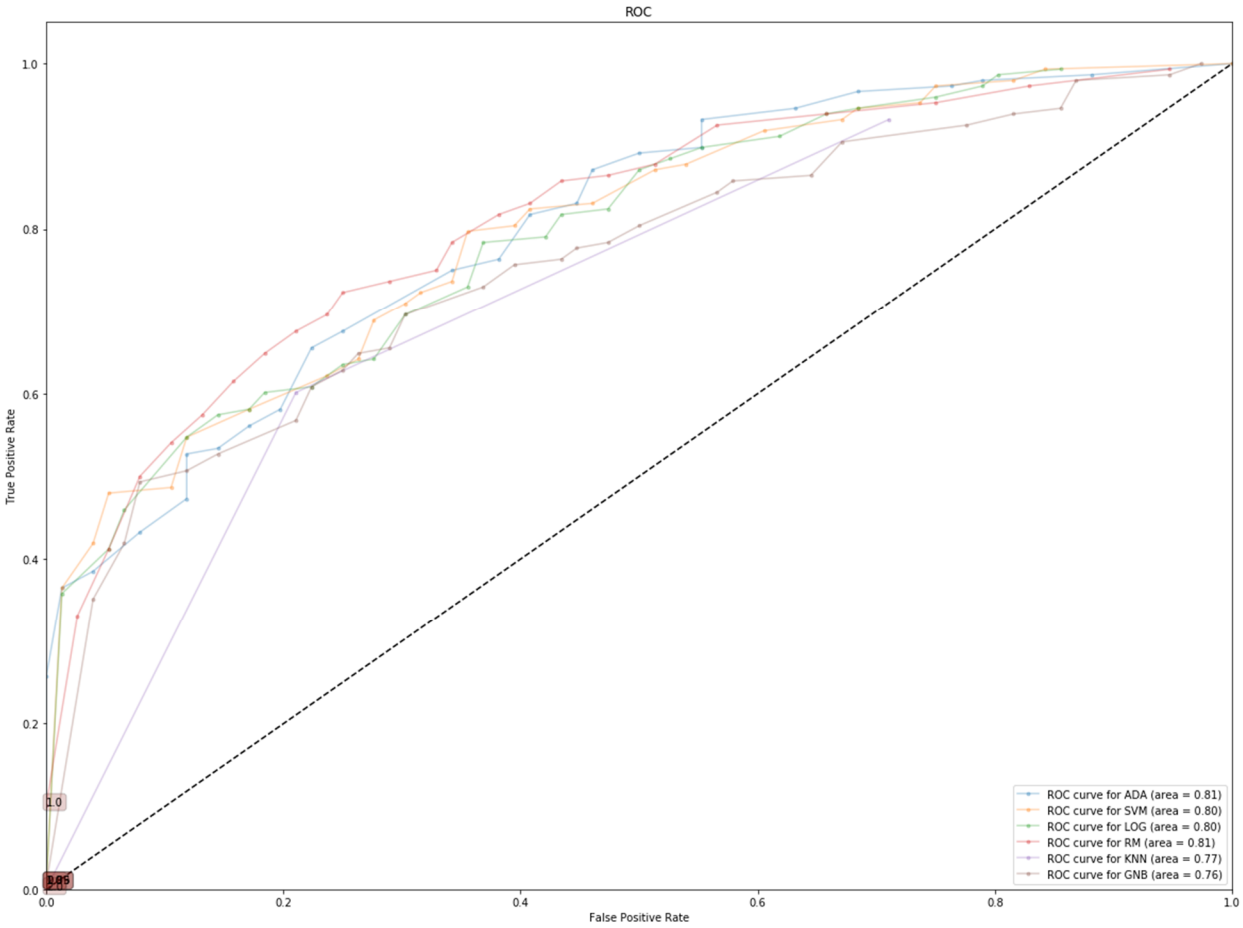

Figure 19: ROC Curve, Target 2, Index Returns and Gold and Oil Returns as features.

Target 3: if Return > 0.0015 -> +1 else -1

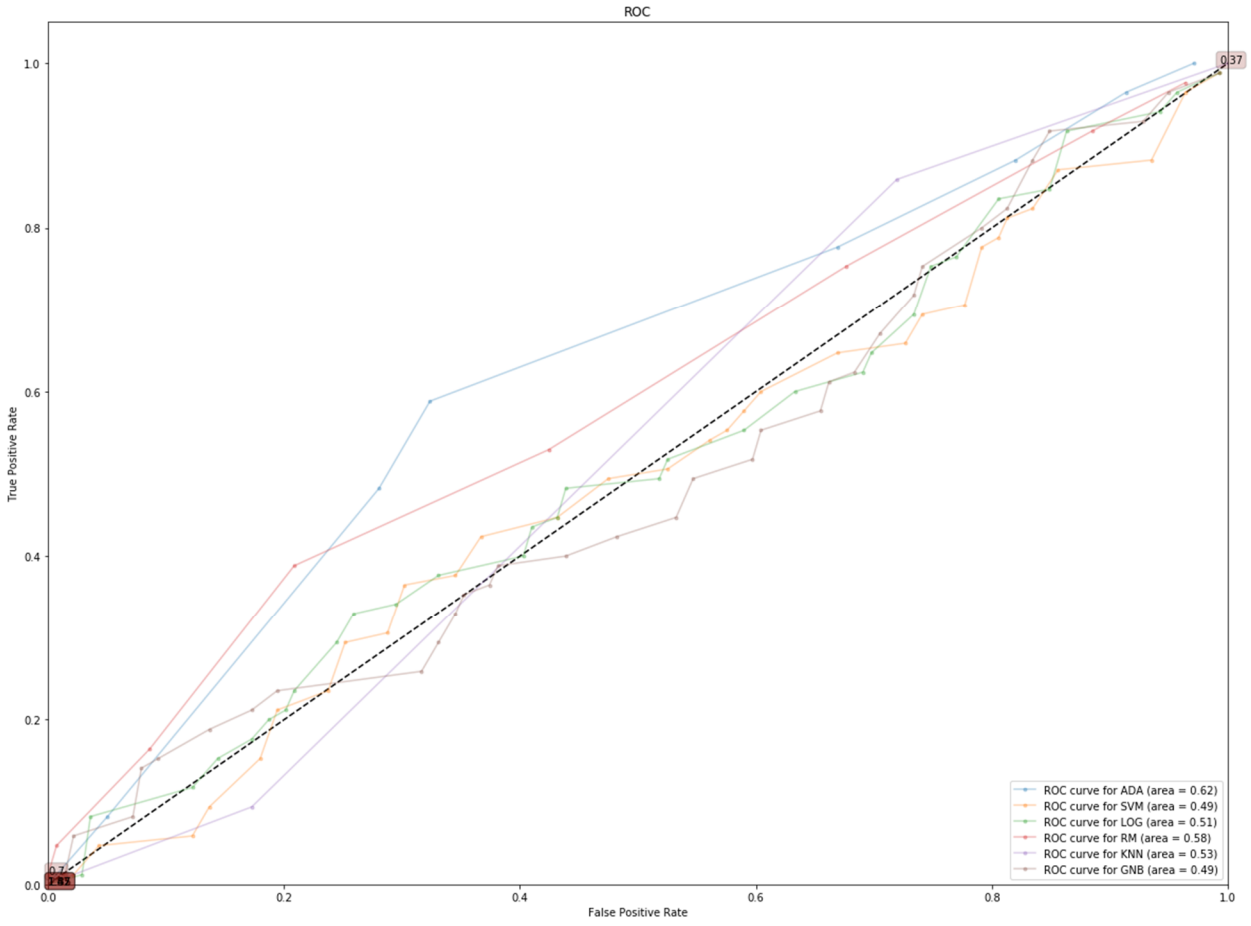

Figure 20: ROC Curve, Target 3, all features selected.

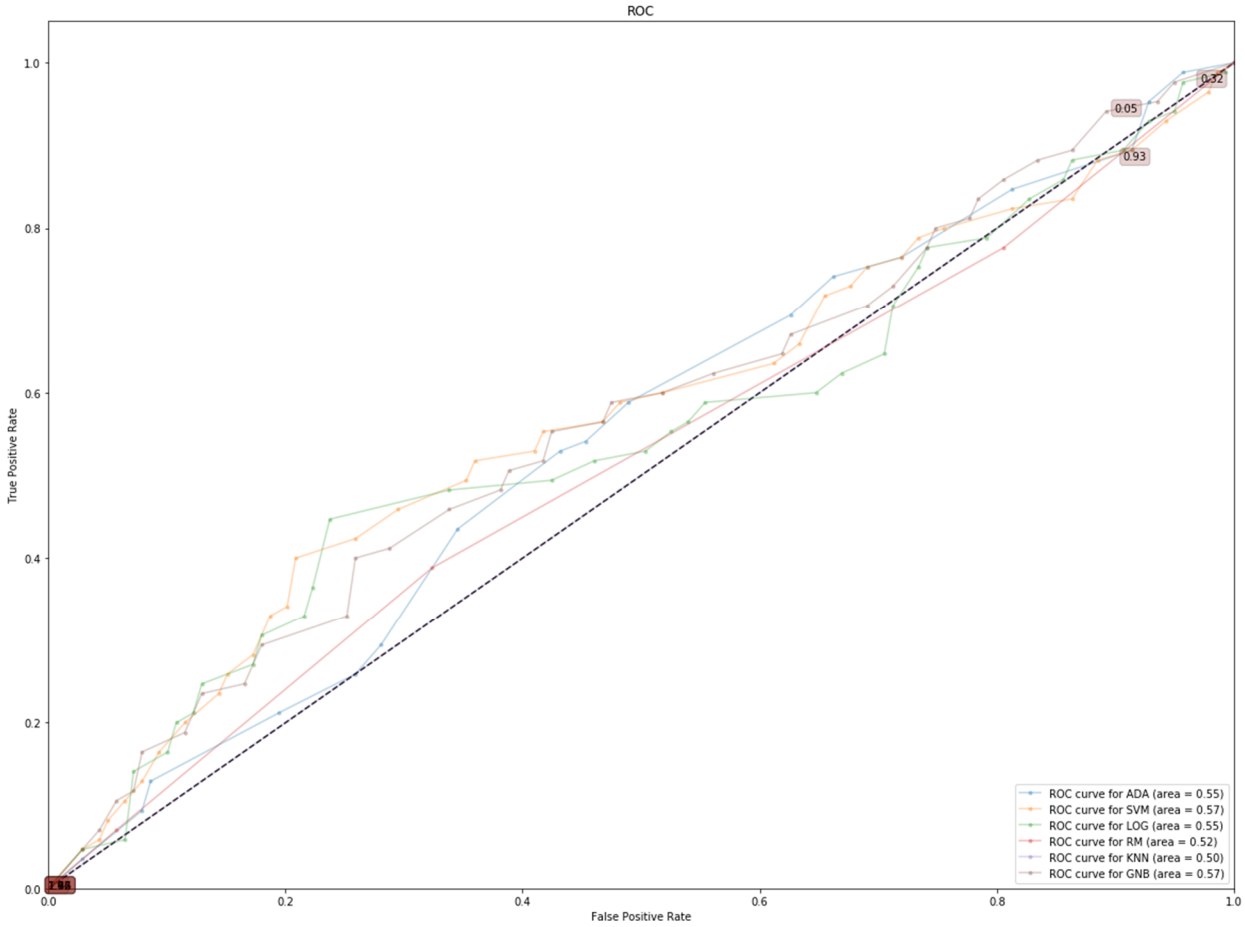

Figure 21: ROC Curve, Target 3, Technical Indicators as features.

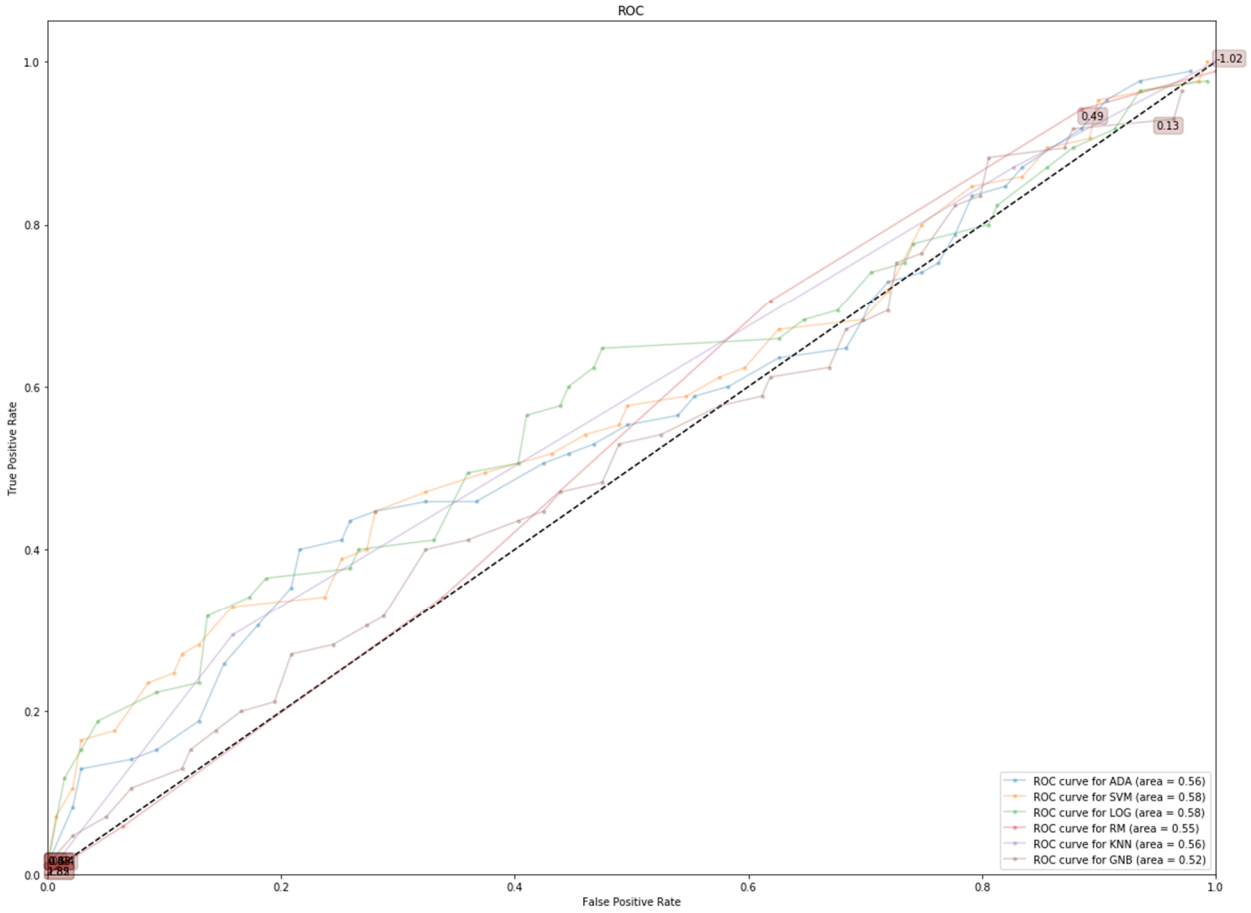

Figure 22: ROC Curve, Target 3, Index Returns and Gold and Oil Returns as features.

	All features	Technical Indicators	Index, Gold, Oil
Target 1 : Close			
ADA	0.51	0.52	0.47
SVM	0.50	0.49	0.53
LOG	0.50	0.51	0.53
RF	0.60	0.54	0.50
KNN	0.50	0.52	0.53
GNB	0.45	0.53	0.56
Target 2 : Open			
ADA	0.82	0.68	0.81
SVM	0.54	0.65	0.80
LOG	0.54	0.69	0.80
RF	0.81	0.67	0.81
KNN	0.48	0.58	0.77
GNB	0.54	0.65	0.76
Target 3 : Returns			
ADA	0.62	0.55	0.56
SVM	0.49	0.57	0.58
LOG	0.51	0.55	0.58
RF	0.58	0.52	0.55
KNN	0.53	0.50	0.56
GNB	0.49	0.57	0.52

Table 1: Summary of the AUC score of each Target with different selected features.

- **For Target 1, best AUC is achieved by Random Forest.**
- **For Target 2, best AUC is achieved by Random Forest and AdaBoost .**
- **For Target 3, almost all algorithms performed quite similar.**
- **Based on AUC, the algorithms can best predict Target 2.**
- **GNB's AUC significantly reduces by selecting all features in Target 1 and Target 3.**
- **For all Targets AUC improved when only Index Return and Gold and Oil returns were selected.**
- **The algorithms performed the least for Target 1.**
- **Index returns and gold and oil returns were significant predictors for Target 2.**

Chapter 7

Conclusion and Future Work

In this project, we developed an evaluation system based on ROC AUC for viewing the underlying performance of 6 different machine learning algorithms. The project was based on selecting intelligent time-series data and feature generation and more importantly, defining three different one-day ahead outcomes for a better overview of the Algorithm's performance.

Perhaps more importantly we have addressed the following theoretical questions that relate to the design of expert and intelligent systems:

(A) What is the value of using technical analysis when predicting one-day ahead stock movement?

(B) Which target is most suitable for prediction based on the historical prices and technical indicators?

(C) Which Machine learning model provides the best predictive performance for each of the three targets?

Our results provide prima-facie evidence that all the three targets can be predicted better than a coin-flip by selecting the best performing algorithm. These results have showed significant performance compared to just the investment strategy of buy and hold and may also give significant information to algorithmic traders and investors about the way around to tackle the Financial Data Forecasting problem.

As we mentioned in the beginning, the theory underlies that stock market hold the semi-strong form. This means if there is any successful trading strategy deployed in the market; the market will soon adjust and come to equilibrium in response to any new investment strategy. The trick lies both in speed and ability to foresee long-term movement of the market.

In regards to the application of Machine learning algorithms shown in this paper, there are multiple ways in which the result can be improved since the limited time was not enough to tackle this versatile problem by taking in all required to-dos. This project can be taken on to implement a trading strategy which could give us future absolute price or returns information but the following aspect should be taken in account to improve it further :

1) Disparate Data sources or Additional Domain Knowledge:

   Only numerical data was used. Recent studies suggest that news, media, rumors, political unrest can vastly influence the movement of stock or index prices. There can be additional data source from Google Trends, Wikipedia, Yahoo Finance news, etc. which holds these information and possibly will significantly improve the performance of the model.

2) Dimensionality Reduction:

   The number of variables can be reduced by implementing some scikit-learn functionalities like Principal Component Analysis or variables of similar structure can be clustered into sets using k-Means, Spectral clustering which has not been covered in this project.

3) Feature selection:

   In this project there was no technical methodology used to select features for each algorithms but only by categorization such as technical indicators, index returns etc. However recent studies suggest different algorithms can perform well in different features or technical indicators. A Recursive Feature Elimination (RFE) algorithm can be implemented which will do backwards selection of predictors or features based on predictor importance ranking. This will also help to remove any features that are not useful.

4) Standardization or Variance Scaling:

   In our project we have standardized features by calculating percentage returns and therefore the data may look like a normally distributed data. This idea can be taken further to Binarization of the features such that all features will only have a value from 0 to 1. Scikit-learn provides a MinMaxScalar utility which can be used.

5) Imputation of missing values and Exponential weighting of data:

   The data used in this project were not exponentially weighted such that data closer to

the present will have greater weight than data previous years ago. This may also improve the performance of the machine learning algorithms. Furthermore, we can also include data from a longer period of time to increase our sample set or impute missing values of data.

6) Implementation of Genetic Programming or Neural Networks:

   In addition to the algorithms discussed in this paper, we can take it further by implementing Neural Networks or Genetic Programming which in recent years is being popular and holds competitive performance record in predicting stock markets.

We still deem the results satisfactory more so than what was expected when the work was initiated. Studies on this topic are continuously evolving and it will be interesting to see the implication on society for its betterment that this vast topic holds.

Bibliography

1. *Eugene F. Fama, 1965. The Behavior of Stock-Market Prices.*
2. *Andrew W. Lo, A. Craig MacKinlay, 1999. A Non-Random Walk Down Wall Street.*
3. *Andrei Shleifer, 2000. Inefficient Markets: An Introduction to Behavioural Finance.*
4. *Stephen F. LeRoy, 1989. Efficient Capital Markets and Martingales.*
5. *Eugene F. Fama, 1998. Market efficiency, long-term returns, and behavioral finance.*
6. *Sven Bouman and Ben Jacobsen, 2002. The Halloween Indicator, "Sell in May and Go Away": Another Puzzle.*
7. *Burton G. Malkiel, 2003. The Efficient Market Hypothesis and Its Critics.*
8. *Fama and French, 1993. Common risk factors in the returns on stocks and bonds.*
9. *Turner, 2007. Human emotions: A sociological theory.*
10. *Andrew W. Lo, Harry Mamaysky, Jiang Wang, 2000. Foundations of Technical Analysis: Computational Algorithms, Statistical Inference, and Empirical Implementation.*
11. *Hendershott & Moulton, 2011. Automation, Speed, and Stock Market Quality: The NYSE's Hybrid.*
12. *Soyoung Kim, 2003. Monetary Policy Rules and Business Cycles.*