University of BRISTOL

DEPARTMENT OF COMPUTER SCIENCE

A Comprehensive Study of the Ensemble Learning in Predicting the Stock Market

Lim Hao Yee

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Bachelor of Science in the Faculty of Engineering.

Monday 9th May, 2022

Abstract

This project is to study the effect of applying different ensemble learning models in predicting the stock market. The implemented ensemble learning models in this project are the majority voting, bagging, boosting and stacking. Besides that, this project also further analyses the effect of using different base learners, which are the decision tree, support vector machine and multilayer perceptron for each of the ensemble learning models. Two market indexes are used for this project, which are the S&P500 and NIFTY50. Furthermore, the simple moving average, stochastic oscillator, relative strength index, moving average convergence divergence, williams percent range, momentum, weighted moving average, commodity channel index are used as the input features for the models. The results obtained from this project showed that the stacking and boosting ensemble learning methods are not suitable for predicting the stock market due to the nature of the dataset. Overall, ensemble learning methods still showed a positive effect to be applied on predicting the stock market as the majority voting model has improved 11.9% and 3.3% in terms of accuracy compared to the best individual machine learning algorithm for S&P500 and NIFTY50 respectively. Lastly, the choice of base learner for the ensemble learning models played a vital role in determining whether the model does improve in terms of performance.

Dedication and Acknowledgements

This project is able to complete due to the support of my supervisor, Rami Cehab that regularly provides suggestions and advices throughout the whole project period.

Declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Taught Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, this work is my own work. Work done in collaboration with, or with the assistance of others, is indicated as such. I have identified all material in this dissertation which is not my own work through appropriate referencing and acknowledgement. Where I have quoted or otherwise incorporated material which is the work of others, I have included the source in the references. Any views expressed in the dissertation, other than referenced material, are those of the author.

Lim Hao Yee, Monday 9th May, 2022

Contents

List of Figures

List of Tables

Ethics Statement

This project did not require ethical review, as determined by my supervisor, Rami Cehab.

Supporting Technologies

- I used the Scikit-Learn library to implement decision tree, support vector machine, adaboost, bagging and majority voting models.
- I used the Keras library to implement the multilayer perceptron model.
- The stacking model implemented in this study was highly inspired by one of the contributor from StackOverflow (`https://datascience.stackexchange.com/questions/41378/how-to-apply-stacking-cross-validation-for-time-series-data`).
- I used the Yahoo Finance API to collect market data for the S&P500 and NIFTY50.
- The Numpy and Pandas library are used to perform data-preprocessing.
- The mlxtend library are used to conduct an analysis on the models' variance.

Notation and Acronyms

MLP	:	MultiLayer Perceptron
DT	:	Decision Tree
SVM	:	Support Vector Machine
BAG	:	Bagging
AdaBoost	:	Adaptive Boosting
ANN	:	Artificial Neural Network
RNN	:	Recurrent Neural Network
SMA	:	Simple Moving Average
WMA	:	Weighted Moving Average
RSI	:	Relative Strength Index
MACD	:	Moving Average Convergence Divergence
EMH	:	Efficient Market Hypothesis
LSTM	:	Long Short Term Memory
SVR	:	Support Vector Regression
MOM	:	Momentum
CCI	:	Commodity Channel Index
Williams%R	:	Williams Percent Range
S&P500	:	Standard and Poor's 500 Index
NIFTY50	:	National Stock Exchange FIFTY

Chapter 1

Introduction

The stock market has always been regarded as an important element in the 21st century gold rush, where investors are investing their hard-earned money into different securities in the hopes of gaining financial success. This is evident in a report from the World Bank [2]at the end of 2020, which shows that the stock market capitalisation worldwide has reached a whopping 93.686 trillion USD. To mitigate the risk of loss and maximise profits on return, investors are facing an uphill battle to predict the market trend as there are many different factors that could impact the price of a security at any time, such as financial news, financial reports of a security, traders' behaviour etc. Therefore, the prediction of market trends has become a complicated puzzle which many are rushing to solve.

With regard to the prediction of the stock market trend, there are two traditional approaches that were commonly used by researchers from the financial field back in the early days. The first being technical analysis, now frequently used by day traders, which analyses historical stock data such as price and volume to forecast a security's future price. Whereas the second approach, fundamental analysis, studies the security's financial report such as revenue, profit margin, etc and is popular among long-term investors.

However, these two approaches have undergone harsh criticism by many as being unreliable in the prediction of a security's future price. This is supported by Eugene Fama's theory of Efficient Market Hypothesis(EMH)[5] in the 70s, which proposes that the analysis brought out through technical or fundamental analysis is considered redundant as all known information about a security is already reflected in the current price of the security, thus, defeating the purpose of looking through historic market data. Although there are winners and losers in the market, this theory believes that the majority of the investors in general did not make a gain or loss in the stock market and those who did, by pure luck. Therefore, the theory concluded that it is impossible to predict the future price of a security through any kind of analysis.

The Random Walk theory[11] proposed by American economist, Burton Malkiel, supports EMH by stating that the price movement of a security follows a random walk and is considered unpredictable. In his book, he stated that "a blindfolded monkey throwing darts at a newspaper's financial pages could select a portfolio that would do just as well as one carefully selected by experts". This theory was supported by an interesting experiment conducted by the Wall Street Journal[1] where professional investors were gathered to pick the securities that they believed would generate investment profit based on their analysis, which will then be compared with securities randomly picked by using darts. At the start, the securities chosen by the experts outperform the randomly picked securities. Over the years of the experiment, the investment profit generated by both varying methods were surprisingly roughly the same.

Nonetheless, due to the advancement of technology, researchers shifted their attention to utilising machine learning algorithms in predicting the stock market. It is believed that machine learning algorithms are able to spot the underlying hidden pattern on the historic market data which can ultimately be used to predict the market trend to a certain extent. Artificial neural netowrk[17] and support vector machine[7] have proven to be useful in accomplishing this goal. Commonly, researchers used classical approaches such as technical and fundamental analysis to produce informative input features for the predictive models.

Currently, Deep Learning is considered as the state-of-the-art method in utilising machine learning algorithms for making predictions on the price movement of the stock market. This is because the recurrent neural network (RNN) is equipped with an internal memory, allowing the output from the previous state to be used as the input for the current state. This function of the RNN made it best suited for time-series data, which is the case for the stock market data. Furthermore, deep learning models such as the convolutional neural network(CNN) and long short-term memory(LSTM) are also used to perform sentiment analysis based on financial news to predict the trend of the stock market[16].

Concurrently, ensemble learning has also gained a lot of popularity as an effective and efficient technique, as it combines multiple individual predictive models and yields a better performance than an individual machine learning model. This is evidently shown by the fact that it is heavily utilised in various Data Science competitions. For example, at the Netflix tournament, a team led by researchers from AT&T had found success in improving the Netflix movie recommendations to pass the 10% mark by utilising the blending ensemble learning model. Besides that, ensemble learning has also found success in improving the model's performance in numerous fields such as Finance[6], Agriculture[4], Healthcare[15], etc.

1.1 Motivation

Although ensemble learning has been widely utilised in various fields and proven to be very useful in producing a higher quality predictive model, it is not always guaranteed, due to multiple factors such as the choice of base learner used. Currently, there are lots of ensemble learning methods that have been developed such as bagging, blending, boosting, etc., with each having its own unique way of combining predictions from multiple individual machine learning algorithms, in order to produce a better overall prediction.

Despite ensemble learning models being used in various fields, it is seldom used by researchers to predict the trend of the stock market. Due to the lack of experimentation on the effect of different ensemble learning models with regards to market trend prediction, the general guidelines for using it to predict the stock market is unclear. Consequently, the majority of the researchers tend to lean towards other machine learning algorithms like RNN instead of ensemble learning models. Although the RNN model has achieved good performance in predicting the stock market, it is also known to be difficult to implement and requires more computational cost compared to ensemble learning models.

Therefore, the aim of this study is to provide a comprehensive analysis on the effect of different ensemble learning algorithms in predicting the stock market. Besides that, this project will also provide an analysis on the effect of using different base learners on the ensemble learning models. Hopefully at the end of this project, it will provide readers a good understanding on the utilisation of ensemble learning methods in predicting the price trend of the stock market.

1.2 Ideas, Aims and Objectives

The overall objective for this project is to provide a thorough analysis on the effect of ensemble learning algorithms in predicting market trends. This study will further analyse the effects of various ensemble learning models and base learners used for each model in predicting the stock market. The idea is to first implement three individual machine learning models which are the support vector machine, decision tree and multilayer perceptron which will be used as the baseline and base learner for the ensemble learning models. Next, four different types of ensemble learning models which are stacking, boosting, majority voting and bagging will be implemented. While the boosting and bagging models will be experimented with different base learners, the stacking and majority voting models will be experimented with different combinations of base learners. Since this study is about the prediction of the market trend, all the implemented models are classifiers which are used to predict the uptrend or downtrend price movement of a security.

The aims and objectives for this project are the following:

- Design and implement the individual machine learning models

- Design and implement the ensemble learning models
- Evaluate the performance of ensemble learning methods
- Evaluate the performance of machine learning methods
- Analyze whether the ensemble learning methods can outperform individual machine learning models in predicting the trend of the price movement for securities
- Analyze the effect of different base learners used for the ensemble learning methods
- Analyze the effect of ensemble learning models when different stocks dataset are used

Chapter 2

Literature Review

Since this project focused on using individual machine learning and the ensemble learning models to predict the price trend of the stock market. Therefore, this chapter is served to review the techniques used and results obtained by other related studies that utilised machine learning and ensemble learning algorithms in predicting the stock market.

The study done by Yakup Kara et al[9] proposed the idea of using ten different technical indicators such as simple moving average, relative strength index, etc to use as the input variables for the artificial neural network and support vector machine. The dataset used for the research is the Istanbul Stock Exchange(ISE) National 100 index. At the end of the study, two main findings were found, the first one is that the ANN model outperformed the SVM model by 4.02% in terms of accuracy and the second finding is that the ten technical indicators have proven to be useful in predicting the direction of stock price movement.

Besides that, Yuxuan Huang et al[8] proposed the idea of using information obtained from fundamental analysis as the input features for the feed-forward neural network, random forest and adaptive neural fuzzy inference system models to predict the Standard and Poor's 100(S&P100) index. The change of price per earning, change of total assets, change of book value and etc were used for the input features. Furthermore, the models is used to make long-term stock performance prediction and 22 years' worth of stock data is used in the study. The study concluded that all three machine learning models are capable of outperforming the market, provided enough data is used.

In Bin Weng et al study[18], the decision tree, support vector machine and artificial neural network is used to predict the price trend movement of Apple stock. The study used four different sources, which are the stock market data, technical indicators, Wikipedia hits and Google news data as the input features for the models. The study was focused on examining the importance of the extra data source and the results showed that the online resources from Wikipedia and Google are useful for the models. Furthermore, the results showed that combining disparate sources also improved the models' performance. However, the result might be invalid as the models are only applied to an individual stock and K-fold cross validation is used to obtain the result of the models' performance.

Shikha Mehta et al[12] proposed the idea of using a weighted ensemble model by combining support vector regression, multiple regression and long short-term memory to predict the stock price. The Yahoo stock is used as the dataset for the project. Overall the result from the study showed that the deviation between the predicted price and actual price has a significant reduction in ensemble model compared to any of the three individual models. Therefore, the study has proven that ensemble models do produce a more effective and efficient model compared to single model.

In [10], Yang Li et al proposed the idea of using the blending ensemble method to combine predictions from deep learning models, which are the LSTM and gated recurrent units(GRU). The stock data such as the adjusted closing price and financial news are used as the input features for the models. The news data was collected from CNBC, Reuters, Wall Street Journal and Fortune and the dataset used in the study was the Standard and Poor's 500(S&P500) index. The result showed that the ensemble learning method produced a better performance model compared to the individual models used in the project.

Chapter 3

Technical Background

This chapter is intended to provide technical background for this project. This includes a basic introduction to the technical analysis(Section 3.1) and a detail explanation of the technical indicators(Section 3.2) that are used as the input features for the models in this project. Besides that, this chapter also introduce the technical background of the machine learning(Section 3.3) and ensemble learning(Section 3.4) models used in this project. The introduction for all these models will be highly technical and considered as a low-level introduction.

3.1 Technical Analysis

Technical indicator is a technical analysis tool that is applied by traders who believe that using a security's historic market data such as price and volume can be used to judge the security's future price movement. It is one of the traditional approaches alongside with fundamental analysis which traders normally use to predict the future price of a security. Technical analysis was first introduced by Charles Dow in the late 1800s and lots of researchers such as William P. Hamilton, Robert Rhea, etc, have also contributed to the development of technical analysis.

Technical indicators are pattern-based signals produced by price and volume of a security. Currently, there are more than a hundred technical indicators available in the financial field. Out of the hundred technical indicators, the popular indicators which are commonly used by traders includes simple moving average, bollinger bands, relative strength index, etc. The common strategy applied by traders is to use a bag of different technical indicators instead of a single one. This is because each technical indicator has its own functionality and can be separated into four different categories such as trend indicators, volume indicators, momentum indicators and volatility indicators. Besides that, each category of indicators can be further separated into either leading or lagging indicators.

3.1.1 Trend Indicators

These indicators study the historic price of a security by smoothing out the price in a given time period to determine the direction and strength of security's price trend. The time period for trend indicators can be set for months or even to minutes depending on traders' preferences. The advantage of using trend indicators is because it smoothed out the price in a given period by cutting down the noise, which allows traders to get an idea on the direction on where the security's price is moving. Examples of trend indicators are simple moving average(SMA), moving average divergence convergence(MACD) and parabolic SAR.

3.1.2 Momentum Indicators

Momentum indicators study the rate of change of a security's price in a given time period. It measures the strength or weakness of a security's price trend and is often used with trend indicators to identify the strength of the trend. Momentum indicators can also be used to spot trend reversal by determining the overbought and oversold condition for a security. Historically, it is more common to use momentum indicators when the market is bullish than bearish because a bullish market tends to last longer than a bearish market. Examples of momentum indicators are relative strength index(RSI), average directional index (ADX) and stochastic oscillator.

3.1.3 Volatility Indicators

Volatility indicators measure how much a security deviates away from its mean price. It is a useful indicator for traders to understand the degree of volatility of a stock in order to decide their trading strategy. Some traders might prefer to trade their stock when the volatility is high since the potential investment return is higher. This is because when a security experiences high volatility, the price movement tends to be noisy and produces unpredictable price swings. On the other hand, some traders might prefer to enter the market when the volatility is low since the price movement is steady and the risk of making a loss is lower. However, unlike trend indicators, the volatility indicators do not provide any information on the direction of the security's price. Examples of volatility indicators are bollinger bands, average true range(ATR) and keltner channel.

3.1.4 Volume Indicators

Trading volume of a stock is the total number of transaction occurred on a given trading day for a security. Trading volume often reflects how traders in the market are perceiving the stock which is an important information for traders. This is because when a stock is experiencing a high trading volume and an increase in price might indicates there is a uptrend movement for the stock's price. Volume indicators is the technical analysis tool that use this information to help traders to validate a stock's price trend or to spot trend reversal. It studies the effect of trading volume in a given time period. Examples of volume indicators are on-balance volume(OBV), volume relative strength index(Volume RSI) and money flow index(MFI)

3.2 Technical Indicators

This section is served to explain the technical background of the technical indicators used in this project. The technical indicators used in this project are the simple moving average, weighted moving average, momentum, moving average convergence divergence, commodity channel index, relative strength index, stochastic oscillator and williams percent range. The description and the interpretation of the technical indicators will be clearly explains in this section.

3.2.1 Simple Moving Average

Simple moving average(SMA) is a technical analysis tool that averages the prices of a security, which is usually the closing price, in a predefined time period. It is a trend indicator that helps traders to determine the direction of price trend for a security. The time period of SMA is adjustable depending on the traders' preferences. It is common for short-term traders to use a time period of 20, 10 or 7 days to build their SMA, this is because these shorter-term SMA generate more trading signals as it is more sensitive to the price changes of a security. However, the signals generated by these shorter-term SMA are more unreliable and might generate false signals to the traders. Whereas, a longer-term SMA with time period such as 50, 100 or 200 days normally generate a more reliable trading signal to traders but the cons of these longer-term SMA is that they introduced more lag, which means these longer-term SMA will react slower to a price trend reversal than a shorter-term SMA.

There are a lot of different ways on how traders interpret the values of simple moving average to assist their trading decision. Price crossover is one of the most common technique used by traders to interpret SMA. It is done by assessing crossover between a security's current price and the SMA. When the security's current price is above the SMA, it indicates the trend of the security is up. Conversely, when the security's current price is below the SMA, it indicates the price trend of the security is moving down. Furthermore, some traders will construct two SMA that has different time periods and the crossover between these two SMA can be used to generate trading signal.

This project used a simple moving average(SMA) with a time period of 10 days and the formula is given by:

$$SMA = \frac{C_1 + C_2 + ... + C_{10}}{10}$$

(3.1)

Where:

C: The closing price of a security

3.2.2 Weighted Moving Average

Weighted moving average(WMA) is another type of moving average that is commonly used by traders. WMA is also used to determine the direction of the price trend and trend reversal for a security, which is essentially the same functionality as SMA. However, some traders believed that WMA is a better version of moving average compared to SMA. The limitation of SMA is that it responses slower in the recent price changes compared to WMA. This is because WMA assigns heavier weights to the more current prices and lesser weights on the past prices for a security. Hence, some traders believed that WMA provides a more reliable trading signal compared to SMA. Since weighted moving average is just an another version of moving average, the interpretation of the value of WMA is the same as the interpretation for the SMA.

This project used a weighted moving average with a time period of 10 days and the formula is given by:

$$WMA = \frac{C_1 * 10 + C_2 * 9 + ... + C_{10}}{55} \tag{3.2}$$

Where:

C: The closing price of a security
W: The weighting factor

3.2.3 Relative Strength Index

Relative strength index(RSI) is the most widely used momentum indicator by traders. It measures the speed of price changes over a given time period and is used to determine the overbought or oversold condition for a security. RSI is a bounded oscillator which means the value of RSI is bounded from 0 to 100. Traditionally, when the value of RSI exceeds 70, it assumes that it is overbought in the price of a security. Conversely, if the value of RSI is below 30, it assumes that the price of the security is under oversold condition and advises traders to buy.

This project used a 14-days time period for RSI and the formula is given by:

$$RS = \frac{AvgGain}{AvgLoss} \tag{3.3}$$

$$RSI = 100 - \frac{100}{1 + RS} \tag{3.4}$$

Where:

- AvgGain = Average percentage gain in 14-days
- AvgLoss = Average percentage loss in 14-days

3.2.4 Momentum

Momentum(MOM) is a momentum indicator that measures the change of price movement in a given time period. It is the simplest momentum indicator that simply subtract a stock's current price and the previous price, which is normally the closing price, from a predefined time period. It provides trading signals to traders as the value of MOM is used to determine on whether a security is overbought or oversold. Unlike other momentum indicators such as RSI, the value of MOM is unbounded.

The simplest way to interpret the value of MOM is to observe on whether the value of MOM is positive or negative. In other words, if the value of MOM is above 0, it indicates a bullish price movement for a security. On the contrary, if the value of MOM is below 0, it indicates a bearish price movement for the security. However, the calculation of MOM is fairly simple and is commonly combined with other momentum indicators to validate the trading signals.

This project used a 10-days momentum and the formula is given by:

$$MOM = C_t - C_{t-9} \tag{3.5}$$

Where:

- C_t: Current closing price
- C_{t-9}: Closing price 9 days ago

3.2.5 Moving Average Convergence Divergence

Moving average convergence divergence(MACD) is a trend-following momentum indicator that is used to provide buy and sell signals to the traders. MACD is different from other momentum indicators as it uses two trend indicators, moving averages, into a momentum indicator. This is done by subtracting the longer exponential moving average(EMA) from the shorter exponential moving average to form the MACD line. Exponential moving average is very similar to WMA as it also gives more weights to the current closing price of a security, the only difference is that the assignment of the weights grow exponentially to the most current price. Commonly, a 26-days time period is used for the longer EMA and 12-days time period is used for the shorter one.

Traders often construct another line, which is called signal line, to generate trading signal by observing the crossover between the MACD and signal line. The signal line is a 9-days EMA of the MACD line. When the MACD line is below the signal line, it indicates the price of a security is likely to fall. Conversely, when the MACD line is above the signal line, it indicates the price of a security is likely to rise. Furthermore, some traders treat the MACD line same as the interpretation of MOM, where positive value indicates an uptrend price movement for the security and negative value indicates a downtrend price movement.

This project used the common time period for the two EMA, which are 12-days and 26-days and the formula of MACD is given by:

$$k = \frac{2}{N+1} \tag{3.6}$$

$$EMA = k * C_{current} + EMA_{previous} * (1 - k) \tag{3.7}$$

$$MACD = EMA_{12} - EMA_{26} \tag{3.8}$$

Where:

- N = The size of time period
- k = The weighting factor for EMA
- $C_{current}$ = The current closing price

3.2.6 Commodity Channel Index

Commodity channel index(CCI) is another momentum-based indicator that provides buy and sell signals to traders. It is an oscillator and the value normally ranges from -100 to +100 and the value outside of this range normally reflects whether there is a strong or weak price movement for a security. The percentage of CCI values that falls between -100 to +100 is dependent on the given time period. When a longer time period is given, a higher percentage of CCI values falls between -100 to +100. Conversely, when a shorter time period is given, a lower percentage of CCI values falls between -100 to +100.

Normally, traders will interpret a security as an overbought condition when the value of CCI exceeds +100 and an oversold condition when the value of CCI is below -100. However, the threshold value is adjustable based on traders' preference, a threshold value [-200,200] is commonly used by traders if they prefer a safer trading options as the signals are more reliable and accurate.

The formula of CCI is given by:

$$TP = \frac{H + L + C}{3} \tag{3.9}$$

$$SMA_{TP} = \sum_{i=1}^{12} TP \tag{3.10}$$

$$MD = \sum_{i=1}^{12} \frac{\mid TP - SMA \mid}{12} \tag{3.11}$$

$$CCI = \frac{TP - SMA_{TP}}{0.015 * MD} \tag{3.12}$$

Where:

- TP = Typical price
- H = Highest price
- L = Lowest price
- C = Closing price
- MD = Mean deviation

3.2.7 Stochastic Oscillator

Same as other momentum-based indicators, stochastic oscillator also provides overbought and oversold trading signals to traders. It is used to compute %K, which is known as fast stochastic indicator, by comparing a security's most recent closing price and the high-low price range over a predefined time period, which is usually 14-days. Besides that, the %D, which is known as slow stochastic indicator is also used by traders to determine a safer trading option. The %D indicator is the moving average of the value of %K over a given period, which is usually 3. Traders believed that the slow stochastic indicator provides a more accurate trading signal compared to %K. This is because %K reacts to the security's price changes too fast and often produce false trading signals.

Traders often use %K to identify the overbought or oversold condition of a stock. A stock is said to be overbought when the value of %K exceeds 80 and oversold when the value of %K is below 20. However, some traders prefer to use the crossover between %K and %D to identify trading signals. The trading signal occurs when there is a crossover between %K and %D in the overbought or oversold region. If the %K crosses above the %D in the oversold region, the security is believed to be oversold. Likewise, the stock is believed to be overbought and suggests traders to sell if the %K crosses below the %D in the overbought region.

This project used a 14-days time period for the %K and a 3-period of moving average of %K for the slow stochastic indicator. The formula for %K and %D is given by:

$$\%K = \frac{C - L_{14}}{H_{14} - L_{14}} * 100 \tag{3.13}$$

$$\%D = \frac{\sum_{i=12}^{14} \%K_i}{3} \tag{3.14}$$

Where:

- C = Current closing price
- L_{14} = The lowest price in 14 days
- H_{14} = The highest price in 14 days

3.2.8 Williams Percent Range

William percent range, also known as Williams %R, is a momentum indicator. It is very similar to the stochastic oscillator as William %R also compares a security's most recent closing price with the high-low price range over a predefined time period to determine the overbought or oversold condition for the security. The value of Williams %R is bounded, which ranges from 0 to -100.

Typically, traders observe the value of Williams %R to decide their trading strategy. When the value of Williams %R is higher than -20, it indicates the price of a security is overbought and suggests traders to sell the security. Conversely, if the value of Williams %R is lower than -80, it indicates the price of the security is oversold and suggests traders to sell the security. However, same issue with the stochastic oscillator, the Williams %R is highly responsive to price changes and the trading signals can be unreliable. As a result, the trading signals are commonly further validate by using other technical indicators.

This project use a 14-days time period for William %R and the formula is given by:

$$Williams\%R = \frac{H_{14} - C}{H_{14} - L_{14}} \tag{3.15}$$

Where:

- H_{14} = Highest price in 14 days
- L_{14} = Lowest price in 14 days
- C = The current price

3.3 Machine Learning Algorithms

This section is to provide a complex and low-level technical background of Multilayer Perceptron, Support Vector Machine and Decision Tree to the readers. These machine learning algorithms will be used to predict the trend of the price movement for stocks and also applied as the base learners for the Ensemble Learning models in this study. Besides that, the performance of these predictive models are used as the baseline when evaluating on whether ensemble learning methods improve model's performance.

3.3.1 Multilayer Perceptron

MLP is a higher level of artificial neural network compared to perceptron as it is able to deal with non-linearly separable data. It is a feed-forward artificial neural network. A feed forward neural network means the information only travels forwards in the network. The architecture of MLP consists of an input layer, one or more hidden layers and an output layer. Each layer can be explained as follow:

- Input layer: The input layer is responsible to bring the input data through the input neurons to the network and the number of input neurons are determined by the number of input features.
- Hidden layer: This layer is able to perform nonlinear transformations of the inputs and higher number of hidden layers are commonly used to solve more complex data. The hidden layer contains a bias node and an activation function

- Output layer: This layer is responsible to produce the final output from the network. It also contains a bias node and an activation function

The reason why MLP is capable of dealing with non-linearly separable data is because of the activation functions used. When neurons, z_i computes the value of weighted sum and applied it with a non-linear activation functions, it produces a non-linear output. Hence, when a MLP has several hidden layers and if each layer uses a non-linear activation functions, it increases the complexity of the computation and allows the network to handle complex data. Some commonly used activation function are the rectified linear units(ReLu; Equation 3.16), hyperbolic tangent(Tanh; Equation 3.17) and the sigmod/logistic activation function(sig; Equation 3.18). The softmax activation function(Equation 3.19) is also a non-linear activation function but it is only applied to the output layer in the case of multi-class classification.

$$ReLu_{z_i} = \begin{cases} z_i, & \text{if } z_i > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{3.16}$$

$$tanh(z_i) = \frac{2}{e^{-2z_i} + 1} - 1 \tag{3.17}$$

$$sig(z_i) = \frac{1}{1 + e^{-}z_i} \tag{3.18}$$

$$softmax(z_i) = \frac{e_i^z}{sum_{j=1}^{K} e_j^z} \tag{3.19}$$

Where:

- K = number of classes

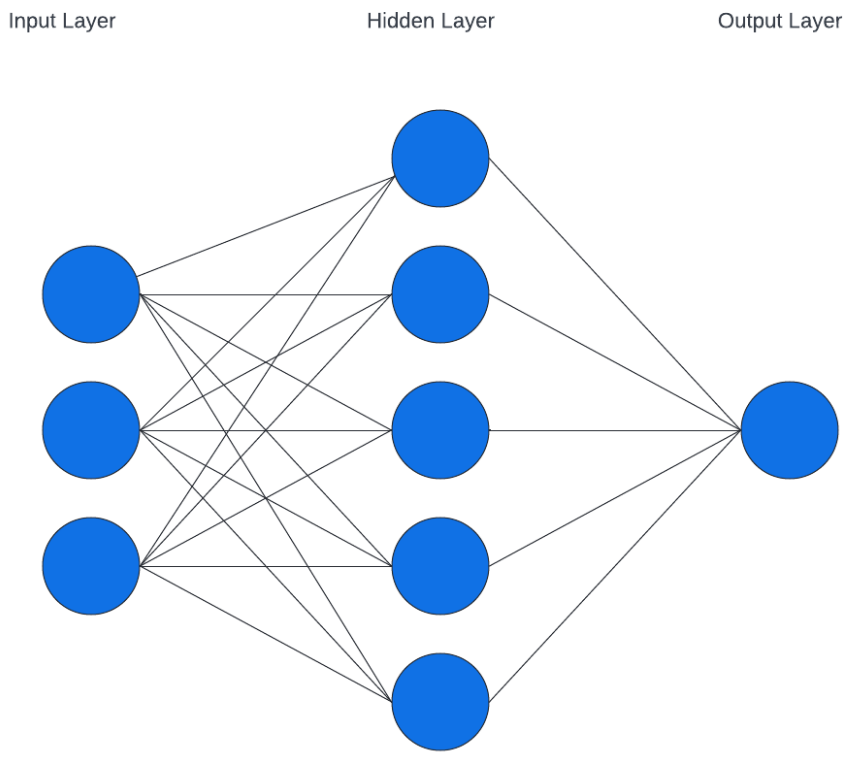

Figure 3.1: Feed-forward Artificial Neural Network

In order to clearly explain how MLP compute the prediction, $\hat{y}$, an example of a three-layer of MLP for a binary classification is shown in Figure 3.1. Based on the figure, the number of input neurons is 3 which indicates the dimension of the input vector x must be 3 as well. The computations of MLP take place in every neurons in the hidden and output layers. At the hidden layer, the neuron first computes the weighted sum by multiplying the weight matrix, w_1 and the input vector and added up with the bias vector, b_1. The value of the weighted sum is then applied with an activation function, Φ_1 to produce the final output, h for the hidden layer. The equation for calculating the value of h is given by:

$$h = \Phi_1(x * w_1 + b_1) \tag{3.20}$$

The output vector, h from the hidden layer are used in the output layer to produce the predictions. The dimension of the weight matrix, w_2 is 5x1 as the number of neurons in the hidden layer is 5 and the number of neurons in the output layer is only 1. In the output layer, the neuron also computes the value of weighted sum by multiplying h and the weight matrix, w_2 and added up with the bias vector, b_2. Since this example is a binary classification problem, the value of the weighted sum is then applied to the sigmoid activation function, Φ_2 to compute $\hat{y}$. The equation for computing the predictions is given by:

$$\hat{y} = \Phi_2(h * w_2 + b_2) \tag{3.21}$$

Since the value of weights and biases in MLP are initialised randomly, the accuracy of the predictions made by the MLP without any training will not produce a good result. Therefore, the MLP has to be trained by adjusting the value of the network parameter in each layer, which are the weights and biases, iteratively in order to minimise the loss function through a gradient descent algorithm. In each iteration of the training process, the input x is fed into the network to produce prediction output $\hat{y}$. With the prediction output, the network is able to compute the loss function L. The loss function for regression task is mean squared error(Equation 3.22) and cross entropy for classification task(Equation 3.23)

$$L = MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2 \tag{3.22}$$

$$L = \text{Cross Entropy} = -\frac{1}{N}\sum_{i=1}^{K}\sum_{j=1}^{N} y_{ij}\log(\hat{y_{ij}}) \tag{3.23}$$

Where:

- M = Number of Classes
- N = Number of predictions

The back-propagation algorithm is used to calculate the gradients of the loss function with respect to parameter via chain rule for each layer . For a single parameter θ, the equation to calculate its gradient is given below:

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial h} * \frac{\partial h}{\partial \theta} \tag{3.24}$$

Where:

- h = The weighted sum

Lastly, the value of the gradient is used to update the parameter and the equation is shown as below:

$$\theta = \theta - \eta * \frac{\partial L}{\partial \theta} \tag{3.25}$$

Where:

- η = The learning rate, which determines the influence of the gradient

3.3.2 Decision Tree

Decision tree is a supervised machine learning algorithm that is widely used in the industry to produce a predictive model. There are different algorithms for decision tree such as Iterative Dichotomiser 3(ID3), C4.5, C5 and Classification and Regression Trees(CART). The CART algorithm is able to deal with continuous and categorical input features, whereas an algorithm such as ID3 is only capable of handling categorical input features.

Before explaining the CART decision tree, some terminology of the CART decision tree will first be introduced as below:

- Root Node: The starting node of the tree and represents all samples
- Decision Node: It is an internal node and can be split further
- Leaf Node: The final node and will not be split further
- Branch: The connection between nodes
- Splitting: The process of dividing a node into two or more sub-nodes

The decision tree starts from the root node which contains the whole training data. In order to split the root node, the best split is found by finding the split has the lowest cost function. The cost function for a regression model is the sum squared error(SSE; Equation 3.26) and for a classification model, the cost function is the Gini index (Gini; Equation 3.27). The CART algorithm find the best split using the greedy algorithm which calculates the cost function for all the possible splits across all the possible input features, the split with the lowest cost function is chosen as the best split. This process is repeated at every single node until a stopping criterion is met such as a the maximum depth of a tree is reached. This algorithm is said to be greedy because it makes the locally optimal choice at each node instead of a global optimum choice.

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y_i}) \tag{3.26}$$

Where:

- n = The number of observations
- y = The observed value
- $\hat{y}$ = The predicted value

$$Gini = 1 - \sum_{i=1}^{C} (P_i)^2 \tag{3.27}$$

Where:

- C = Number of classes
- P = the probability of an object being classified to a particular class

3.3.3 Support Vector Machine

Support vector machine(SVM) is a machine learning algorithm that uses hyperplane to classify data points with different classes. SVM can be used for both regression and classification task. The main objective for SVM is to find the optimal hyperplane that could perfectly separate the data points that are different classes. The dimension of the hyperplane is dependent on the dimension of the input feature. For a two-dimensional input data, the hyperplane is essentially a line. Whereas for a three-dimensional input data, the hyperplane is a two-dimensional plane. Since this project used SVM for classification only, therefore the technical background of the classification SVM will only be discussed. In this part, the SVM will be explained for two different cases, which are the case for linearly separable dataset and linearly inseparable dataset.

Linearly Separable Dataset

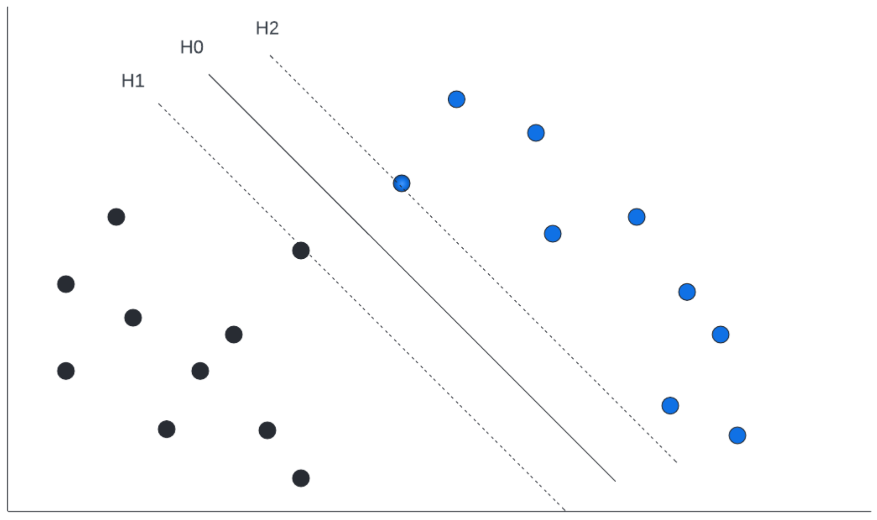

Figure 3.2: Hyperplanes in SVM

Assuming a Z number of n-dimensional linearly separable data with only two different classes, the dataset can be formalised as x_i, y_i, where $i = 1, 2, ..., Z$, $y_i \in -1, 1, x \in \Re^N$. For simplicity purpose, the dimensional of the dataset is considered as 2. Therefore, the hyperplane, H_0 is essentially a straight line and the formula is given by:

$$H_0 : w \cdot x + b = 0 \tag{3.28}$$

Where:

- w : The vector normal to the hyperplane
- b : The offset

Since the dataset is linearly separable, there are several hyperplanes(Figure 3.2) that could perfectly separate the two classes. However, the purpose of SVM is to maximise the margin, which is essentially the distance between the marginal hyperplanes, H_1 and H_2, in order to find the optimal hyperplane, H_0. The hyperplanes H_1 and H_2 is formed based on the support vector for the class -1 and 1, where support vector is the data points that are closest to the hyperplane, H_1 and H_2. These hyperplane is shown on Figure **??** and the equations for H_1 and H_2 is given by:

$$H_1 : w \cdot x + b = -1 \tag{3.29}$$
$$H_2 : w \cdot x + b = 1 \tag{3.30}$$

Where:

- H_1: The hyperplane for class -1
- H_2: The hyperplane for class 1

Before introducing the optimasation problem for SVM, the value of the margin, M, has to be calculated and it is essentially the distance between the marginal hyperplane, H_1 and H_2, and the formula is given by:

$$M = (1 - b)/|w| - (-1 - b)/|w| = \frac{2}{|w|} \tag{3.31}$$

Since the value of M is the sum of the distance between the marginal hyperplane, H_1 and H_2. Therefore, the margin, M can be rewritten as $\frac{1}{|w|}$ and the main goal of SVM is to maximise the value of M which can be formalised into $\min_{w,b} \frac{|w|^2}{2}$ and this value can be solve by using the Lagrange Multiplier. This type of SVM is called hard margin SVM as there is no data point appears between the marginal hyperplanes, H_1 and H_2.

Linearly Inseparable Dataset

Most of the real-world dataset are considered to be linearly inseparable, therefore the soft margin SVM is used to address this problem. The soft margin SVM loosen the constraint of not allowing any data point appear between the marginal hyperplanes, H_1 and H_2 by introducing the hyperparameter C and slack variables, ξ. The slack variables are used to measure the distance between the data points and the marginal hyperplanes and the hyperparameter C defines the tolerance of misclassification. Therefore, the objective function for the soft margin SVM is formalised as $\min_{w,b,\xi} \frac{|w|^2}{2} + C\sum_{i=1}^{L} \xi_i$ and can be solved by using the Lagrange Multiplier.

Besides using the hyperparamter C and slack variables to overcome the issue from linearly inseparable dataset. The SVM used "kernel trick" to map the data to a higher dimensional feature space. After the transformation, the data will now become linearly separable. Some commonly used kernel functions for SVM are the Gaussian radial basis (RBF; Equation 3.32) and polynomial(poly; Equation 3.33) kernel functions.

$$K(x_i, x_j) = \exp(-\gamma|x_i - x_j|^2) \tag{3.32}$$

$$K(x_i, x_j) = (x_i \cdot x_j + 1)^d \tag{3.33}$$

Where:

- d : The degree of the polynomial function
- γ : The constant of radial basis function

3.4 Ensemble Learning Algorithm

Ensemble learning is a machine learning technique that is used to combine multiple machine learning models, also known as base learners in order to produce a better predictive model. This section is used to explain the technical background of the ensemble learning methods used in this project, which are the majority voting, bagging, boosting and stacking ensemble learning methods.

3.4.1 Majority Voting

Majority voting, also known as voting ensemble is an ensemble machine learning model that combines predictions from multiple machine learning models to produce a better predictive model. It is a heterogeneous ensemble method as the voting ensemble allows different types of base learners to be used.

There are two different method used by the majority voting to combine predictions for classification, which are the hard voting and soft voting methods. The hard voting method(Equation 3.34) produces the final prediction based on the class label that has the majority votes from the base learners. Whereas for the soft voting method(Equation 3.35), the base learners compute the prediction probability value for each class and the voting ensemble model averages the probabilities for each class. The class with the highest average probability is chosen as the predicted class for the voting ensemble.

$$\text{Hard Voting} = mode(C_1, C_2, ..., C_n) \tag{3.34}$$

Where:

- C_i = The class label predicted by ith base learner

$$\text{Soft Voting} = argmax_i \frac{1}{m}\sum_{j=1}^{m} p_{ij} \tag{3.35}$$

Where:

- p_{ij} = The predicted probability for i class from jth base learner

This study will only used the soft voting method for the voting ensemble model. This is because when combining the predictions from only two base learners and in the case of a tie, the voting ensemble model which uses the hard voting approach will simply select the class based on the ascending order. Therefore, the final predicted outputs might not be useful.

3.4.2 Bagging

Bagging, also known as bootstrap aggregation is another type of ensemble machine model. It is a homogeneous ensemble method as it only allows the same type of classifier to be used. Bagging is used to improve a model's performance by combining the predictions made from multiple versions of a classifier. It is believed that the bagging ensemble method can help a classifier in reducing the variance while maintain the model's bias, which in turn improve the model's performance.

As mentioned before, bagging combines predictions from multiple versions of a classifier. Firstly, the bagging algorithm creates n number of bootstrapped samples depending on the number of base learners by using a technique called bootstrapping. Bootstrapping is a statistical resampling technique that randomly samples a dataset with replacement. Lastly, for classification, the bagging algorithm makes predictions by either using the hard or soft voting approach. For regression, the bagging algorithm averages the value produced by the base learners.

The reason why the bagging algorithm is able to improve the accuracy and stability of a model is because of the bootstrapped samples. The bootstrapped samples are independent from each other, therefore the base learners have not seen the full training dataset which in turn prevent the issue of overfitting. Besides that, the bagging algorithm trains the base learners in parallel which results in a decrease of the computational time.

Random forest is actually a type of the bagging algorithm and is commonly used as a predictive model instead of a decision tree. This is because the random forest is able to solve the issue of overfitting that is commonly occurred in the decision tree. However, a traditional bagging model only performs bootstrapping whereas the random forest performs bootstrapping and uses different input features for each base learners.

3.4.3 Boosting

Boosting is an ensemble learning algorithm which uses multiple weak learners in order to produce a strong predictive model. Commonly, a weak learner is defined as a machine learning algorithm that is only able to produce predictions that are slightly better than random guessing. The boosting algorithm has a lot of variations such as AdaBoost, Gradient Boosting, XGBoost and Light GBM. This study will only use the AdaBoost algorithm for the boosting ensemble model.

AdaBoost also known as adaptive boosting is an adaptive algorithm because it increases the weightage of the misclassified training instances from the previous weak learner and allows the next weak learner to focus more on correcting the misclassified training instances. For classification problem, the process of the AdaBoost algorithm can be explained in a few steps. Firstly, it initialises an equal weight to every training instances. The training process starts when the first weak learner is trained using the training dataset that has equal weight and the total error is calculated. The total error is the summation of the weights of the misclassified training instances. The total error is used to calculate the value of alpha(α; Equation 3.36), which also know as "Amount of Say". The value of *alpha* determines the importance of a weak learner when is used to make final predictions.

$$\alpha = \frac{1}{2} \log \frac{1 - \text{total error}}{\text{total error}} \tag{3.36}$$

After that, the weights of the misclassified and correctly classified training instances are adjusted(Equation 3.37) by increasing the weights for the misclassified training instances and decreasing the weights for the correctly classified training instances. The adjustment for the weights of the training instances allow the next weak learner to pay more attention in classifying the misclassified training instances and less attention to the correctly classified training instances.

$$w_i = w_{i-1} * \exp^{\pm\alpha} \tag{3.37}$$

Where:

- w_i = The adjusted weight for the next weak learner

This process will be repeated for every weak learners until the predifined number of weak learners is reached. Lastly, the AdaBoost algorithm is able to produce the prediction $\hat{y}$ using the equation below:

$$\hat{y} = \sum_{i=1}^{N} \alpha_i c_i \tag{3.38}$$

Where:

- α_i = The alpha value for the ith weak learner
- c_t = The prediction made by ith weak learner

The purpose of the boosting algorithm is to reduce bias in order to produce a better predictive model. Boosting basically tries to reduce the bias error which arises when models are not able to identify relevant trends in the data.

3.4.4 Stacking

Stacking, also known as stack generalization is heterogeneous ensemble learning method that combines prediction from multiple machine learning models that are not the same type. It is one of the most famous ensemble machine learning algorithm that is commonly used in Data Science competitions to improve model's accuracy. Unlike boosting which combines predictions from multiple version of weak learners, stacking uses machine learning models that are effective in solving the predictive modeling task but effective in different ways.

The simplest architecture of a stacking model consists of two levels, the level-0 consists of multiple predictive model, which is also known as the base learners. These base learners are used to predict the validation sets and the predictions made by the base learners are used as the train set for the next level model. In level-1, it consists of only one predictive model, which is known as the meta-model. This meta-model is used to combines the predictions made from the base learners and linear model such as logistic regression and linear regression models are commonly used. It is not necessary for the stacking model to consists only two levels as it is often to construct a multiple levels of base learners to produce a stronger stacking model.

The training process of a two levels stacking algorithm is given below:

1. The training data is splits by using the K-fold cross validation, where the K subsample is used as the validation set and the K-1 subsamples are used as the training data
2. A base learner is trained on the K-1 subsamples and made predictions on the K subsample which is the validation set
3. The process of step 2 is repeated for K times with each of the K subsample is used only once as the validation data.
4. The base learner is trained on the original training data and made predictions on the test set to evaluate its performance
5. The steps from step 2 to step 4 is repeated for other base learners
6. The predictions made by the base learners on the validation sets are used to train the meta-model
7. Lastly, the trained meta-model is used to make predictions on the test set.

Chapter 4

Project Execution

This chapter will first introduce the basic description of the project workflow in Section 4.1 to provide readers the context surrounding the project execution. The Section 4.2 and 4.3 provides the description the dataset and evaluation metrics used in this project. The data pre-processing techniques used in this project will be explains in details on Section 4.4. Lastly, Section 4.5 provides the implementation and the training process, which is mainly the hyperparameter tuning process, for the models used in this project.

4.1 Project Framework

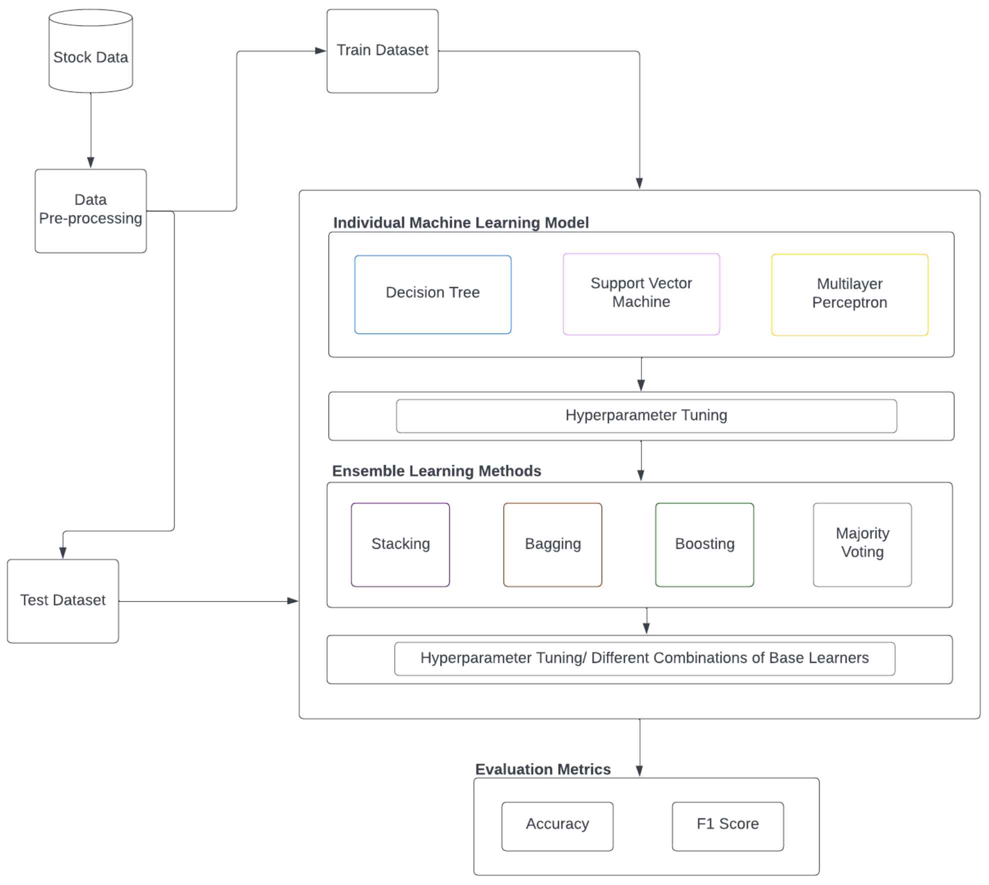

Figure 4.1: Project Framework

Figure 4.1 shows the basic framework of this project. Firstly, the stock data will go through the process of data pre-processing in order to produce a better quality and acceptable format for the predictive models. Secondly, the three individual machine learning models, which are the decision tree, support vector machine and multilayer perceptron, will first be implement and perform hyperparameter tuning to produce the optimal models. Thirdly, the individual machine learning models are used as the base learner to implement the ensemble learning models. Besides that, some of the ensemble learning such as the bagging and boosting consists of hyperparameter. Therefore, these ensemble models also required to perform the process of hyperparameter tuning to find the optimal combination of hyperparameter in order to produce the optimal model. Lastly, all the implemented models are evaluated by using the evaluation metrics and the obtained results will be used for an in-depth analysis which is the main goal of this project.

4.2 Project Dataset

Based on the literature review in Chapter 2, all of the studies focused on making predictions on two types of market data. Some of the studies used market indexes such as the S&P100[8], ISE National 100[9] and S&P500[12]. Other studies used individual company market data such as Apple[18]. Currently, there is no general guideline on which type of market data is better to apply on machine learning algorithms. However, this study used stock indexes as the dataset because most of the stock indexes such as Standard and Poor's 500(SP500) consists of the top 500 companies in the United State. Essentially, the price movement of SP500 can used to represent the whole market movement in the United State.

The stock indexes used in this project are the S&P500 and the National Stock Exchange Fifty(NIFTY50). The NIFTY50 is a stock index which consists of the top 50 listed companies on the India National Stock Exchange. The market data for S&P500 and NIFTY50 are collected from 01-11-2004 to 31-01-2021 from Yahoo Finance. However, only the dataset from 01-01-2005 to 31-12-2020 will be used as the dataset for the predictive models in this study. This is because the extra data is used to construct the technical indicators which require the historic market data. Lastly, the description of the imported market data is explained below:

1. Open : It is the first traded price for a security when trading begins.
2. Close : It is the last traded price for a security in a trading day.
3. High : It is the highest price traded for a security in a trading day.
4. Low : It is the lowest price traded for a security in a trading day.
5. Adj Close : It is the adjusted closing price for a security which provides a more accurate representation of the security's value.
6. Volume : It is the number of transaction made for a security in a trading day

4.3 Evaluation Metric

In order to provide an analysis on the predictive models in this project, an evaluation metric has to be used to evaluate the models' performance. Since the models implemented in this study are all classifiers, therefore the evaluation metric used in this study is the accuracy. Essentially, the accuracy calculates the fraction of correct predictions made by the model and the formula is given by:

$$accuracy(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^{N-1} 1(\hat{y}_i = y_i) \tag{4.1}$$

Where:

- y = The observed value
- = The predicted value by the predictive model
- N = The number of data points

Besides accuracy, the F1-score is also used as the evaluation metric for this project. Commonly, accuracy is not enough to determine the performance of a classifier. This is because the result obtained from the accuracy can be misleading especially in the case of class imbalance. A high value of accuracy might be achieved by a classifier due to the fact that one class has a higher ratio compared to other classes in the unseen dataset. Whereas, the F1-score is able to address this issue as the calculation of the F1-score takes into account of the other evaluation metrics such as precision and recall which are also used to address the issue of class imbalance in the unseen dataset. For a binary dataset, the formula of precision and recall is given by:

$$\text{precision} = \frac{TP}{TP + FP} \tag{4.2}$$

$$\text{recall} = \frac{TP}{TP + FN} \tag{4.3}$$

Where:

- TP = Number of True Positive samples
- FP = Number of False Positive samples
- FN = Number of False Negative samples

Lastly, the formula of the F1-score is given by:

$$\text{F1-score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.4}$$

4.4 Data Preprocessing

The dataset for S&P500 and NIFTY50, collected from Yahoo Finance is still in a raw form. Raw data, without any data preprocessing, will render the information collected from the dataset to be less effective when it comes to training the individual machine learning and ensemble learning models in this project. Hence, the purpose of this section is to demonstrate the technique used to transform the raw market data into a dataset of better quality before using it to train the predictive models in this study.

4.4.1 Feature Generation

The initial dataset for the S&P500 and NIFTY50 only provides very basic information and using said information for the input features of the predictive models will most likely guarantee poor performance. Most studies reviewed on Chapter 2 used, either technical or fundamental analysis to improve the raw market data to a higher standard. Since this study is about making price trend predictions on short-term markets, technical analysis will be a better choice compared to fundamental analysis, the reason being that the technical indicators used in technical analysis are usually more flexible in regard to the time period used to analyse the historic market data, which can be hours, days or even months. Whereas, fundamental analysis studies the quarterly or yearly financial report of a security to determine the price movement of the security, making it more suitable for long-term predictions. Therefore, for this project, technical analysis will be the better choice between the two classical approaches.

As mentioned in Section 3.1, technical analysis employs the usage of technical indicators for price trend predictions of a security. Naturally, technical indicators will be used as the input features in this study. Out of all the hundred technical indicators that have currently been developed, some have proven to be more effective than others. However, due to time constraints, building all the developed technical indicators might not be realistic for this project. Hence, using the already proven technical indicators as the input features for predictive models might be the better approach for this study. This study will utilise the technical indicators deployed in Yakup Kara's study[9], which includes the simple moving average (SMA), weighted moving average (WMA), momentum (MOM), moving average convergence divergence (MACD), commodity channel index (CCI), relative strength index (RSI), stochastic oscillator %K(Stochastic %K), stochastic oscillator %D(Stochastic %D) and williams percent range (Williams %R). The technical background of these nine technical indicators are explained in Section 3.2.

Feature Transformation

Previously mentioned in Section 3.2, the function of the technical indicators is to provide trading signals for traders and there are several unique methods to interpret the value of each type of technical indicator with the purpose of predicting a security's price movement. With regards to this function, Jigar Patel et al[13] proposed the idea of transforming the continuous value of the technical indicators into discrete form, [-1,+1], which represents the price trend of a security, where '-1' indicates a downtrend price movement of a security, whereas '+1' indicates an uptrend price movement of a security. This idea found success in improving the performance of the predictive models as the newly formed input features were easier for the machine learning algorithms to interpret, which ultimately improved its performance. Therefore, this study will be using the same strategy used by Jigar Patel by transforming the continuous value of the technical indicators into discrete form that represents the price trend of a security. The method used to transform each of the technical indicators will be further explained below.

With regards to SMA and WMA, both of the technical indicators are a moving average which is used to determine the direction of the price trend for a security. Therefore, the value of SMA and WMA can be interpreted the same way. If the current price is above the value of SMA and WMA, the price trend is considered as an uptrend and will be denoted as '+1'. Conversely, if the current price is below the value of SMA and WMA, the price trend is considered as a downtrend and will be denoted as '-1'.

Since Williams %R is very similar to Stochastic %K and %D as all three study the relationship between the closing price of a security and the high-low price range over a given time period. Therefore, the same strategy will be used to interpret the value of all these three technical indicators. If the value of the technical indicator at time 't' is higher than the value at time 't-1', the price of a security is experiencing an uptrend and is represented as '+1' and vice-versa. Similarly, the same strategy will be also used for MACD.

In terms of RSI, the value is used by traders to determine the overbought and oversold condition of a security. If the value of RSI is above 70, the security is considered to be overbought and if the value of RSI is below 30, the security is considered to be oversold. Hence, when the value goes above 70, it is denoted as '-1' which indicates the price of a security will be dropping and if the value of RSI goes below 30, it is denoted as '+1' which indicates the price of a security will be rising. In the case where the value of RSI is between 30 to 70 and the value of RSI at time 't' is greater than the value at time 't-1', the trend is represented as '+1' and vice-versa.

The strategy used to interpret the value of CCI is similar to the one used for RSI as the former is also used to determine the overbought and oversold condition for a security. However, unlike RSI, the value of CCI is unbounded and the value of [-100,100] or [-200,200] commonly used by traders as the boundary to determine the overbought and oversold condition for a security. Hence, if the value of CCI is above 200, the security is considered to be overbought and it is represented as '-1' and if the value of CCI is below -200, the security is considered to be oversold and it is represented as '+1'. For values in between -200 and 200 and the value of CCI at time 't' is higher than the value at 't-1', the price of a security is considered to be in uptrend and represented as '+1' and vice-versa.

Lastly, for the MOM technical indicator which calculates the rate of change of a security's price in order to determine the momentum of the stock price. When the value of MOM is positive, it is denoted as '+1' to indicate uptrend movement for the security price and if the value of MOM is negative, which indicates the security price is experiencing a downtrend movement, it is denoted as '-1'.

4.4.2 Target Variable

Since this study is to evaluate the predictive models in making price trend prediction in the stock market, the value of the target variable has to be either '1' or '0' to represent uptrend and downtrend price movement respectively. The target variables for the dataset in this study are constructed using a 3-days time horizon. Thus, the idea is to first observe the technical indicator at time 't' and compares the difference between the opening price at time 't+4' and the closing price at time 't+1'. If the opening price of a security at time 't+4' is higher than the closing price at time 't+1', the target variable is denote as '1'. Conversely, if the opening price at time 't+4' is lower than the closing price at time 't+1', the target variable is denotes as '0'.

To provide readers a clearer idea on the construction of the target variable, a real-world point of view is that if a trader looks at the technical indicator on Monday and it provides a strong buying signal, the trader will buys the security on the Tuesday and sells the security on Thursday.

4.4.3 Train-Test Split

The machine learning and ensemble learning algorithms used in this study are all supervised machine learning algorithms. Hence, the models will first be trained on the training set and evaluate the models' performance using the test set. For SP500 and NIFTY50, the training set contains of the dataset from 01-01-2005 to 31-12-2015 and the test set is from 01-01-2016 to 31-12-2020.

Class Imbalance

Based on the training sets for S&P500 and NIFTY50, the distribution of the training sets have shown the issue of class imbalance. In S&P500, the percentage for class labels '1' and '0' are 56% and 44% respectively. Whereas in NIFTY50, the percentage for class labels '1' and '0' are 52% and 48%. Even though both the training sets for S&P500 and NIFTY50 only suffer a slight imbalance on the class distribution, it has shown a negative impact on the predictive models' performance.

Imbalanced dataset has always been one of the major issue that influenced the generalisation of a predictive model in classification. This is because when an imbalance dataset is used to train a predictive model, the model only learns from the majority class and ignores the minority class. As a result, when the model is used to make predictions on unseen dataset, it will classifies all the unseen dataset to the majority class.

To mitigate this issue, the first attempt approach is to use the technique of random undersampling to address this issue. Random undersampling involves randomly selecting data points from the majority class and deletes them from the training set. With this technique, the ratio of the class distribution for SP500 and NIFTY500 are 50:50. However, the limitation of this approach is that deleting the samples from the training set causes loss of information. Besides that, the size of the training set for S&P500 and NIFTY50 are less than 3000, which is considered a small training set. Therefore, random undersampling technique might not be the best method to mitigate this issue.

The best method to mitigate this issue is by adjusting the class weight in the machine learning algorithms. Most of the machine learning algorithms are capable to let users defined the class weights for each classes and is commonly used to mitigate the issue of class imbalance. For example, if a dataset consists of 90% of data points labelled as class "1" and only 10% of data points labelled as class "-1". By setting the class weights for the class "-1" nine times larger than the class weight of class "1", the machine learning algorithm is able to learn the from the two classes equally.

4.5 Model Implementation and Training

This section is to discuss the implementation and the training process for each of the predictive models used in this study. The training process for each of the predictive models requires the process of hyperparameter tuning to find the optimal combination of the hyperparameter values. Hence, the selection of the hyperparameter to tune for each models will also be explained. Besides that, this section will also discuss the traditional approach used for hyperparameter tuning and the limitation of it when dealing with the dataset used in this study.

4.5.1 Hyperparameter Tuning

Hyperparameter tuning involves tweaking the hyperparameters in the machine learning models in order to find the optimal combination of hyperparameters for the models. Hyperparameter tuning not only helps a predictive model to produce a better accuracy, but it also helps a model to avoid overfitting and underfitting. This study used GridSearchCV, which is a library function by the Scikit-Learn library to perform hyperparameter tuning for all the models except the stacking and majority voting ensemble models. This is because the only hyperparameter for these two models are the choices of the base learners.

Given a predefined hyperparameter values, the GridSearchCV tries all the combinations of the hyperparameter values and evaluates the model for each combination through cross-validation. After evaluate all the combination of the hyperparameter values, it returns the best combination that produced the best result. Traditionally, K-fold cross validation is the default approach used in GridSearchCV for model evaluation. The K-fold cross validation first splits the dataset into K number of folds. The first fold is used as the validation set to evaluate the model and the remaining folds are used to train the predictive model. This process repeats for k iterations in order to allow each fold to be used as the validation set. Lastly, the performance of the model is evaluated by averaging the results obtained from the validation sets.

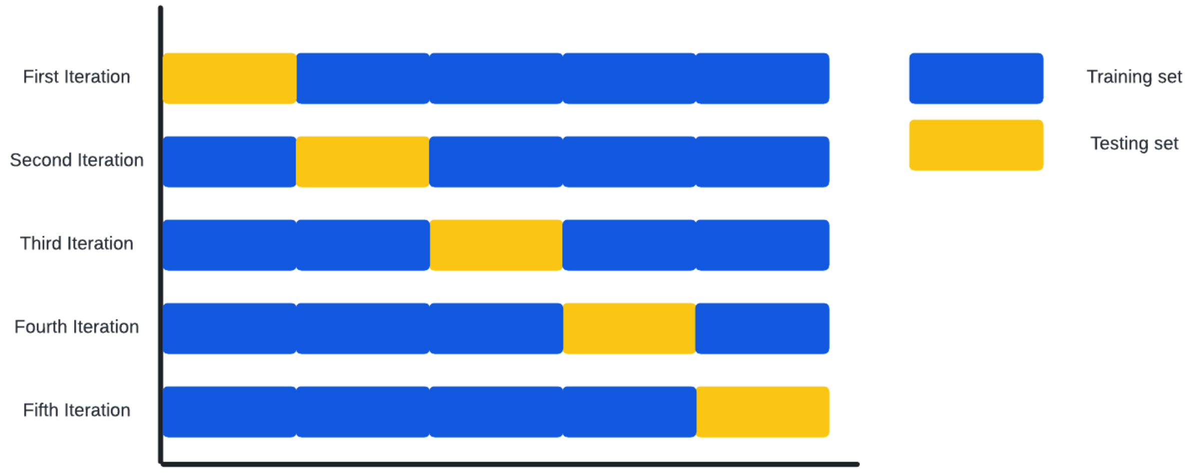

Figure 4.2: K-fold Cross Validation

However, the K-fold cross validation might introduces the issue of data leakage when is used on time-series dataset and produce an inaccurate result of the model. This is because the data points in a time-series dataset follows a chronological order. Based on Figure 4.2, the training data occurs after the validation set. This is problematic as it allows the model to learn from future and make predictions on the past which ultimately produce invalid result of the model's performance. Since the dataset in this study is also a time-series dataset, an alternative approach will be used to replace the K-fold cross validation.

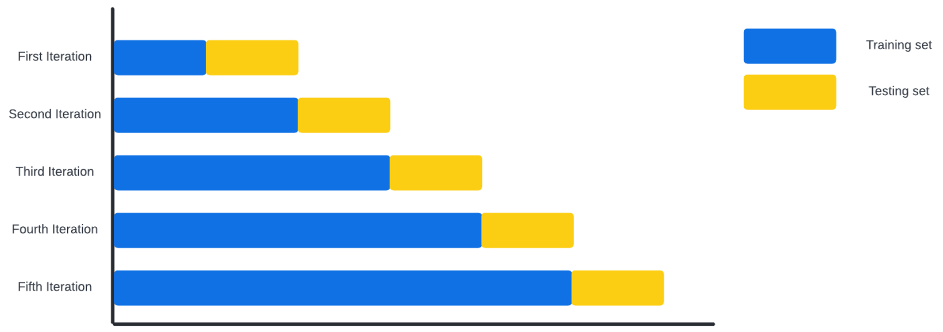

Figure 4.3: Growing-Window Forward Validation

To mitigate the issue caused by K-fold cross validation, this study will another cross-validation method which is called growing-window forward validation. This approach is recommended by Matthias Schnaubelt[14] as the study pointed out that the growing-window forward validation is best suited for time-series dataset. The growing-window forward validation maintains the temporal order of the dataset as shown in Fig 4.3. It ensures that the validation sets always appear after the training sets in every iterations which prevent the issue of data leakage. The validation set will always have the same size, whereas the size of the train set increases in each time.

4.5.2 Multilayer Perceptron

The MLP is built by using the Keras library, which is a library used to build artificial neural network. For simplicity purpose, the MLP consisted only a hidden layer and the activation function used for the hidden layer is the rectified linear unit function(ReLu). Besides that, the MLP uses the stochastic gradient descent(sgd) as the solver for weight optimisation.

There are plenty of hyperparameters in the neural network and each of the hyperparameters have its own functionality. However, this study only tuned the most important hyperparameters, which are the number of neuron in the hidden layer, epochs, batch size and the learning rate. These hyperparameters are extremely important because a high value of these hyperparameters might causes overfitting to happen. Conversely a low value of these hyperparamter might causes underfitting to happen as well. The values tested to these hyperparameters are shown in Table 4.1 and below are the short description of the hyperparameters:

- neurons : To determine the number of neurons in the hidden layer
- batch size : To determine the number of samples that will propagated through the network
- epochs : To determine the number of epochs, where an epoch is made up of one or more batches.
- learning rate : To control the step size in updating the weights

Hyperparameter	Value
Neurons	20, 40, 60, 80, 100
Batch Size	20, 40, 60, 80, 100
Epochs	20, 40, 60, 80, 100
Learning Rate	0.002, 0.004, 0.006, 0.008, 0.01

Table 4.1: Hyperparameter values tested for Multilayer Perceptron

4.5.3 Decision Tree

Through Scikit-learn Library, this project used the DecisionTreeClassifier model to implement the DT. As mentioned in Section 3.3.2, the decision tree stops building the tree until a stopping criterion is reached. This is because without the stopping criterion, the tree will continues to grow until each leaf node represents each observation of the training data and ultimately result in overfitting. Hence, the hyperparameters max_depth, max_features, min_samples_leaf and the min_samples_split are used to prevent the decision tree from overfitting. The values tested on these hyperparameter are shown in Table 4.2 and below are the short description of the hyperparameter for DT:

- max_depth : The maximum depth of the tree
- max_features: The number of features to considers when looking for the best split
- min_samples_leaf : The minimum number of samples required to be at a leaf node
- min_samples_split : The minimum number of samples require to split an decision node

Hyperparameter	Value
max_depth	6, 8, 10, 12, 14, 16, 18, 20
max_features	1, 2, 3, 4, 5, 6, 7, 8, 9
min_samples_leaf	2, 4, 6, 8, 10
min_samples_split	1, 2, 4, 6, 8, 10

Table 4.2: Hyperparameter values tested for Decision Tree

4.5.4 Support Vector Machine

The support vector machine is implemented by using the SVC model from the Scikit-Learn Library and the radial basis function is used as the kernel for the SVM. Commonly, the hyperparameters C and gamma are tuned for SVM because these hyperparamters influenced the generalisation of the model. For the hyperparameter C, it is a regularisation parameter that determines the cost of the misclassification data points. If the value of C is small, it will results in a decision boundary with a large margin as the model allows more misclassification. Conversely, if the value of C is high, the margin of the decision boundary is small as the model has to minimize the number of misclassified data points. For the hyperparameter gamma, it is actually the parameter of the kernel and defines how far the influence of a single data point reaches. A high value of gamma often results in a more curvature of the decision boundary and vice-versa. To produce the optimal SVM model, the optimal combinations of these hyperparameter have to be found and the values tested for the hyperparameter C and gamma in this study are shown in Table 4.3.

Hyperparameter	Value
C	0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.5, 2
Gamma	0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.5, 2

Table 4.3: Hyperparameter values tested for Support Vector Machine

4.5.5 Majority Voting

The majority voting ensemble method is implemented by using the VotingClassifier model from the Scikit-Learn library. As mentioned in Section 3.4.1, there are two approaches used by the majority voting ensemble learning method, which are the hard voting and the soft voting methods. This project will only used the soft voting methods for the model. This is because when combining the predictions from only two base learners and in the case of a tie, the model which uses the hard voting method will simply select the class based on the ascending order. Therefore, the final predicted outputs will not be useful to assess whether there is an improvement of the model's accuracy. The voting ensemble does not require any usage of the GridSearchCV as the only hyperparameter is the choices of the base learners. There are four voting ensemble models to implement in this study and each of the models consist of different combination of base learners. The four different combination of the base learners for the majority voting models are shown in Table 4.4.

Model	Base Learner Combination	Abbreviation
1	Decision Tree & Support Vector Machine & Multilayer	Vot-ALL Perceptron
2	Decision Tree & Support Vector Machine	Vot-DTSVM
3	Decision Tree & Multilayer Perceptron	Vot-DTMLP
4	Support Vector Machine & Multilayer Perceptron	Vot-SVMMLP

Table 4.4: Base Learner Combination for the Majority Voting Model

4.5.6 Bagging

The bagging ensemble learning model is implemented by using the BaggingClassifier model from the Scikit-Learn library and three bagging models are implemented by using different base learners which are the DT, MLP and SVM. The hyperparameters tuned for the bagging models in this study are the n_estimators and max_sample. The max_sample determine the number of samples draw from the training set and n_estimators determines the number of base learners used in the bagging model. Table 4.5 shows the value tested for the hyperparameter of the bagging model and Table 4.6 shows the base learner used for each of the bagging model.

Hyperparameter	Value
n_estimators	20, 40, 60, 80, 100
max_sample	0.2, 0.4, 0.6, 0.8, 1.0

Table 4.5: Hyperparameter values tested for Bagging

Model	Base Learner	Abbreviation
1	Decision Tree	Bag-DT
2	Multilayer Perceptron	Bag-MLP
3	Support Vector Machine	Bag-SVM

Table 4.6: The Base Learner in Bagging Model

4.5.7 Boosting

As mentioned in Section 3.4.3, this study used the AdaBoost algorithm for the boosting ensemble method and is implemented by using the AdaBoostClassifier model from Scikit-Learn library. This study implemented three boosting models which used different weak learners for each dataset, which are the DT, MLP and SVM. The hyperparameters to tune in this study in order to produce the optimal boosting model are the learning_rate and n_estimators. The learning_rate controls the amount of contribution of each weak learner and the n_estimators determines the number of weak learners used in the model. Table 4.7 shows the value tested for the learning_rate and n_estimators and Table 4.8 shows the base learners used for the boosting models in this project.

Hyperparameter	Value
n_estimators	20, 40, 60, 80, 100
learning_rate	0.2, 0.4, 0.6, 0.8, 1.0

Table 4.7: Hyperparameter values tested for Boosting

Model	Base Learner	Abbreviation
1	Decision Tree	Ada-DT
2	Multilayer Perceptron	Ada-MLP
3	Support Vector Machine	Ada-SVM

Table 4.8: The Base Learner in Boosting Model

4.5.8 Stacking

The stacking model is built from scratch in this project as the StackingClassifier model from the Scikit-Learn does not support of using the growing-window forward validation as the cross-validation method for the model. As explained in Section 4.5.1, the K-fold cross validation might introduce the issue of data leakage in this project. In this project, the stacking model only consists of two levels, where the level-0 is the individual machine learning models and level-1 consists of the meta-model. The logistic regression is used as the meta-model in this project as the predictions made by the logistic regression is essentially the weighted average of the predictions made by the models in level-0. Lastly, there are four different stacking model implemented in this project where each of the stacking model used a different combination of machine learning models in level-0 as shown in Table 4.9.

Model	Base Learner Combination for level-0	Abbreviation
1	Decision Tree & Support Vector Machine & Multilayer	Stack-ALL Perceptron
2	Decision Tree & Support Vector Machine	Stack-DTSVM
3	Decision Tree & Multilayer Perceptron	Stack-DTMLP
4	Support Vector Machine & Multilayer Perceptron	Stack-SVMMLP

Table 4.9: Base Learner Combination for the Stacking Model

Chapter 5

Critical Evaluation

This chapter is used to provide a critical evaluation of the models built and study the effect of ensemble learning method in predicting the price trend of the stock market. This project implemented 17 different models for each of the dataset used in this project. As a result, evaluating 17 different models at the same time might not be to useful in terms of the visualisation. Therefore, this chapter is separated into 6 different parts. The first 5 parts is used to evaluate the performance of the individual machine learning, majority voting, bagging, boosting and the stacking models. In the last part of this chapter, it is used to analyse the final comparison between all the best models in the individual machine learning and ensemble learning models.

5.0.1 Evaluation on the Individual Machine Learning Model

Individual Machine Learning Model	Accuracy	F1 Score
Decision Tree	0.508	0.538
Multilayer Perceptron	0.522	0.573
Support Vector Machine	0.513	0.559

Table 5.1: Results for Individual Machine Learning Models:S&P500

Individual Machine Learning Model	Accuracy	F1 Score
Decision Tree	0.509	0.503
Multilayer Perceptron	0.512	0.464
Support Vector Machine	0.518	0.530

Table 5.2: Results for Individual Machine Learning Models:NIFTY50

Table 5.1 and 5.2 shows the accuracy and F1-score for the individual machine learning models when used to predict the S&P500 and NIFTY50 dataset. In both cases, the performance of all the individual machine learning models did not performed well as the accuracy for all of the models are only performed slightly better than random guessing. This showed that the task of predicting the trend of the stock market is extremely difficult as these machine learning models are only capable of making correct predictions slightly more than half. One of the possible reason to explain on why the models could not achieve good performance is because the size of the train set in this project is small. Therefore, all the models are trained with insufficient amount of data and could not learn the hidden pattern from the data.

Besides that, the SVM and MLP have shown a better performance compared to the DT in both dataset. This is because SVM and MLP are considered a more complicated and complex models compared to the DT. Overall, the MLP model performed the best compared to DT and SVM by having the highest accuracy score, 0.522 and F1-score, 0.573 when used for the SP500. For NIFTY50, the SVM performed the best compared to MLP and DT by having the highest accuracy score, 0.518 and F1-score, 0.530. These two models will be used as the baseline models in the final comparison in order to evaluate the effect of ensemble learning models.

5.0.2 Evaluation on the Majority Voting Model

Majority Voting Models	Accuracy	F1 Score
Vot-ALL	0.556	0.660
Vot-DTMLP	0.532	0.585
Vot-DTSVM	0.560	0.670
Vot-SVMMLP	0.584	0.724

Table 5.3: Results for Majority Voting Models–S&P500

Majority Voting Models	Accuracy	F1 Score
Vot-ALL	0.527	0.540
Vot-DTMLP	0.523	0.514
Vot-DTSVM	0.510	0.523
Vot-SVMMLP	0.535	0.563

Table 5.4: Results for Majority Voting Models–NIFTY50

Table 5.3 and 5.4 shows the accuracy and F1-score for the majority voting ensemble models for S&P500 and NIFTY50. In both dataset, the majority voting models showed the capabilities of predicting the trend of the stock market especially in the case for S&P500. The Vot-SVMMLP model showed a very high accuracy and F1-score for the S&P500. Besides that, when the combination of base learner used for the majority voting model is SVM and MLP, the accuracy and F1-score achieved the highest compared to other models for S&P500 and NIFTY50. This result is expected as the SVM and MLP have shown a better performance compared to the DT.

Based on the tables, the performance of majority voting models varied quite a lot as the final predictions made by the models are high dependent on the combination of base learner. This is because the majority voting model computes the final prediction just by picking the target class that has the highest sum of probabilities of the predictions from the base learner. As a result, if a model misclassified a data point with high prediction probability, the model will most likely dominates the result of the majority voting model. Therefore, the majority voting model will also computes the wrong prediction.

Overall, the Vot-SVMMLP achieved the best performance among all of the other majority voting models for S&P500 and NIFTY50 and will be used for the final comparison.

5.0.3 Evaluation on the Bagging Model

Bagging Model	Accuracy	F1 Score
Bag-DT	0.542	0.602
Bag-SVM	0.588	0.740
Bag-MLP	0.541	0.605

Table 5.5: Results for Bagging Models–S&P500

Bagging Model	Accuracy	F1 Score
Bag-DT	0.534	0.552
Bag-SVM	0.534	0.696
Bag-MLP	0.520	0.504

Table 5.6: Results for Bagging Models–NIFTY50

Table 5.5 and 5.6 shows the accuracy and F1-score for the bagging models for S&P500 and NIFTY50. Based on the table, the Bag-SVM models showed a very good performance for S&P500 and NIFTY50. However, when an in-depth analysis was done on the models, it showed that the predictions made from the models were only a single class. Therefore, the performance of the models is considered invalid as the models lost its generalisation on unseen dataset. One of the main reason that this issue occurred to the models is because the primarily focus of the bagging method is to reduce the model variance. When conducting an analysis on the model variance(Table 5.7 and 5.8), the SVM models have the lowest variance compared to DT and MLP. Thus, the SVM models already have a very low variance which result in the bagging method did not work when applied to it.

Overall, when the bagging model used DT as the base learner, it significantly improved the performance of the model. The Bag-DT achieved the best performance among other models in S&P500 and NIFTY and will be used for the final comparison.

Model	Variance
Decision Tree	0.273
Support Vector Machine	0.210
Multilayer Perceptron	0.264

Table 5.7: Model Variance–S&P500

Model	Variance
Decision Tree	0.275
Support Vector Machine	0.238
Multilayer Perceptron	0.277

Table 5.8: Model Variance–NIFTY50

5.0.4 Evaluation on the AdaBoost Model

AdaBoost Model	Accuracy	F1 Score
Ada-DT	0.521	0.552
Ada-SVM	0.588	0.740
Ada-MLP	0.412	0.000

Table 5.9: Results for AdaBoost Models–S&P500

AdaBoost Model	Accuracy	F1 Score
Ada-DT	0.521	0.535
Ada-SVM	0.534	0.696
Ada-MLP	0.466	0.000

Table 5.10: Results for AdaBoost Models–NIFTY50

Table 5.9 and 5.10 shows the accuracy and F1-score for the boosting models for S&P500 and NIFTY50. The issue of the bagging model that used SVM as base learner also happened in the AdaBoost models. In S&P500 and NIFTY50, the Ada-SVM and Ada-MLP models suffered the problem of model's performance degradation as the generalisation of the models on unseen dataset are worsen. One of the possible reason is because the boosting model is commonly used with weak learner that only perform slightly better than random guessing. When it is used with a strong learner that has achieved a good performance, degradation of the model performance often happened[19]. As pointed out in Section 5.0.1, the MLP and SVM models have shown better performance in the SP500 and NIFTY50 compared to DT. Therefore, this might explain the reason on why the performance of SVM and MLP has became worsen when used in the boosting models. However, this explanation might not be valid because even though the MLP and SVM models performed better than DT, the accuracy for MLP and SVM are only slightly better than random guessing. Another possible reason that might be able to explain this issue is that boosting does not work well with noisy dataset[3]. The dataset used in this project are stock market data which is known to be random, volatile and noisy and ultimately caused boosting models to not work well.

Overall, only the result from the Ada-DT models in S&P500 and NIFTY50 are considered valid. Although the boosting technique does improved the performance of the DT but the models only showed a slight improvement. Therefore, boosting technique can be concluded to be unhelpful when used to predict stock market.

5.0.5 Evaluation on the Stacking Model

Stacking Models	Accuracy	F1 Score
Stack-ALL	0.586	0.721
Stack-DTMLP	0.588	0.740
Stack-DTSVM	0.588	0.740
Stack-SVMMLP	0.588	0.740

Table 5.11: Results for Stacking Models–S&P500

Stacking Models	Accuracy	F1 Score
Stack-ALL	0.537	0.687
Stack-DTMLP	0.534	0.696
Stack-DTSVM	0.534	0.696
Stack-SVMMLP	0.534	0.696

Table 5.12: Results for Stacking Models–NIFTY50

Table 5.11 and 5.12 shows the accuracy and F1-score for the stacking models for S&P500 and NIFTY50 Based on Figure 5.9 and 5.10, the stacking models achieved a very high accuracy score even if the combination of base learners are different. However, when performing an in-depth analysis on the predicted class by the stacking models, it showed that all of the stacking models are predicting one class only. Therefore, even though the accuracy score for the stacking models are high, it is not suitable to be used in real-world application as the results are considered invalid.

There are two main reason to explain this issue, firstly, the stacking models usually used strong learners as the model in level-0. The definition of strong learners are the machine learning algorithms that able to predict well on the unseen dataset. However, in this study, all the models used for the stacking models have a slightly better performance than random guessing. The second reason is because the cross validation technique used in this study , which is the growing-window forward validation. The main limitation of this cross validation technique is that in the first few iterations, the size of the training set is very small. This issue is problematic for the stacking model as the meta-model is trained by using the predictions made by the models in level-0. If the size of the training set is small, the quality of the predictions made by the models in level-0 will be poor as well. As a result, the stacking model could not provide a better performance.

Therefore, stacking might not be a suitable ensemble learning method used for predicting the trend of the stock market as K-fold cross validation creates bias to the final output and the growing-window forward validation causes degradation in the performance of the stacking model.

5.0.6 Final Comparison

Figure 5.1 and Figure 5.2 shows the best performance model from the individual machine learning and ensemble learning models for S&P500 and NIFTY50. The final comparison do not consider the stacking models because all of the models' result are considered to be invalid. Overall, it shows that ensemble learning methods always outperformed the individual machine learning model. For S&P500, the majority voting which used SVM and MLP performed the best by having 11.9% increased in term of accuracy compared to the MLP. Whereas, the majority voting model which used SVM and MLP as the base learner also has the best performance by having 3.3% increased in term of accuracy compared to SVM in NIFTY50. Besides that, the boosting models performed the worst among the ensemble learning models which showed that it is not suitable to be used on stock market dataset compared to other ensemble learning methods.

Furthermore, the accuracy score for the models in the NIFTY50 is lower than the model in S&P500. This indicates that even though ensemble learning methods will always outperform single machine learning model, the performance of the ensemble learning models are highly dependent to the performance of the individual machine learning model.

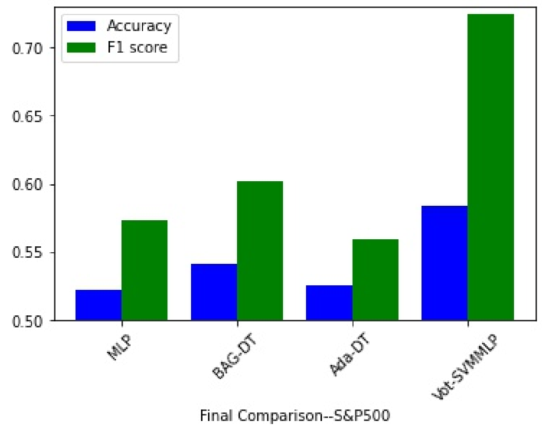

Figure 5.1: Final Comparison for S&P5000

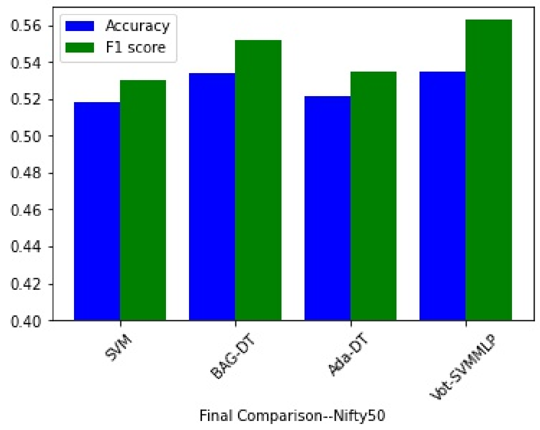

Figure 5.2: Final Comparison for NIFTY50

Chapter 6

Conclusion

This project was successful in performing a comprehensive study on the effect of ensemble learning methods for stock market predictions, by the implementation of the majority voting, bagging, boosting and stacking models. Furthermore, this project also provided an in-depth analysis on the effect of using different base learners for the bagging and boosting models and different combinations of base learners for the majority voting and stacking models. Last but not least, this project has outlined the best approach to implement machine learning algorithms for stock market prediction. For example, the problem of using K-fold cross validation and the issue of class imbalance was promptly explained in Section 4.5.1 and Section 4.4.4 respectively.

Three main findings were gathered after analysing the models built in this project. First, the stacking and boosting ensemble learning methods might not be suitable to be used for predicting the trend of the stock market due to the nature of the stock market dataset. The second finding is that the choice of base learner for the ensemble learning methods is vital, which directly impacts ensemble model's performance. Especially for the case of the bagging and majority voting models. Lastly, this project has concluded that ensemble learning models often yielded a better performance when compared to individual machine learning models.

This project serves as a study to understand the effectiveness of various ensemble learning methods for predicting the market trend and also to provide analysis on the choice of the base learner for said ensemble models. Hopefully, readers will find this project helpful in understanding the utilisation of ensemble learning methods for the purpose of stock market predictions.

Limitation and Further Work

One of the limitation of this project is that it did not make a comparison with other advanced machine learning algorithms such as recurrent neural network which proved to be highly effective in making predictions on the trend of the stock market. The training process of the recurrent neural network required a lot of time. Therefore, due to the time constraint of this project, simpler machine learning algorithms such as SVM, MLP and DT are used in this project. Besides that, an alternative approach for the implementation of the stacking could not be discovered to address the issue of the growing-window forward validation. As a result, all the stacking models implemented in this project could not be used for the final comparison. Lastly, the technical indicators used in this project have a very low correlation with the target variable, which might be the main reason on why all of the models only performed slightly better than random guessing.

One of the possible further work for this project is to implement other ensemble learning method such as blending and weighted majority voting models. The blending model might be a better choice compared to the stacking model in predicting the trend of the stock market. This is because instead of using cross validation for the level-0 models to make predictions, the blending model used holdout dataset for the models to make predictions. As a result, the blending model allows the models in level-0 to train on a sufficient amount of training dataset. For weighted majority voting, it is most likely a better model compared to the majority voting models used in this project. This is because the weighted majority voting is capable to place a higher weightage on better performance models which most likely yield a better result compared to the majority voting.

As mentioned before, the technical indicators used in this project have a very low correlation with the target variable. Therefore, another possible extension of this project can be performing a feature selection on all the technical indicators that have been developed currently in order to pick the most effective technical indicators for the target variable. Whereas, another option might be using a variety of input feature such as technical indicators, information from fundamental analysis and market news which was used in Bin Weng et al's study[18].

Another possible extension of this project is to construct better features for the machine learning algorithms through feature selection. This is because some of the technical indicators might works better when applied on different stock. This project did not perform feature selection to pick the best technical indicators as the number of developed technical indicators are more than a hundred. Due to the time constraint for this project, it is not possible to build all of the technical indicators.

Bibliography

[1] Dartboard investments: Contests, monkeys, and random stock picks. https://prosperityeconomics.org/dartboard-investment-contests/, Lastt accessed on May 1, 2022.

[2] Market capitalization of listed domestic companies (current us$), 2022. https://data.worldbank.org/indicator/CM.MKT.LCAP.CD, Last accessed on May 1, 2022.

[3] Oza Nikunj C. Aveboost2: Boosting for noisy data". pages 31–40, 2004.

[4] Yan-An Chen, Wen-Hao Hsieh, Yu-Shuo Ko, and Nen-Fu Huang. An ensemble learning model for agricultural irrigation prediction. pages 311–316, 2021.

[5] Eugene F Fama. Efficient Capital Markets: A Review of Theory and Empirical Work. *Journal of Finance*, 25(2):383–417, 1970.

[6] Shihao Gu, Bryan Kelly, and Dacheng Xiu. Empirical Asset Pricing via Machine Learning. *The Review of Financial Studies*, 33(5):2223–2273, 02 2020.

[7] Bruno Miranda Henrique, Vinicius Amorim Sobreiro, and Herbert Kimura. Stock price prediction using support vector regression on daily and up to the minute prices. *The Journal of Finance and Data Science*, 2018.

[8] Yuxuan Huang, Luiz Fernando Capretz, and Danny Ting-Yi Ho. Machine learning for stock prediction based on fundamental analysis. *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 01–10, 2021.

[9] Yakup Kara, Melek Boyacioglu, and Omer Baykan. Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the istanbul stock exchange. *Expert Systems with Applications*, 38:5311–5319, 2011.

[10] Yang Li and Yi Pan. A novel ensemble deep learning model for stock prediction based on stock prices and news. 2020.

[11] Burton. G. Malkiel. *A Random Walk Down Wall Street.* Norton, New York, 1973.

[12] Shikha Mehta, Priyanka Rana, Shivam Singh, Ankita Sharma, and Parul Agarwal. Ensemble learning approach for enhanced stock prediction. pages 1–5, 2019.

[13] Jigar Patel, Sahil Shah, Priyank Thakkar, and K Kotecha. Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert Systems with Applications*, 42(1):259–268, 2015.

[14] Matthias Schnaubelt. A comparison of machine learning model validation schemes for non-stationary time series data. 2019.

[15] Sidey-Gibbons and Jenni A. M. Machine learning in medicine: a practical introduction. *BMC Medical Research Methodology.*

[16] Sahar Sohangir, Dingding Wang, Anna Pomeranets, and Taghi M. Khoshgoftaar. Big data: Deep learning for financial sentiment analysis. *Journal of Big Data*, 5:1–25, 2017.

[17] B. W. Wanjawa and L. Muchemi. ANN Model to Predict Stock Prices at Stock Exchange Markets. 2015.

[18] Bin Weng, Mohamed A. Ahmed, and Fadel M. Megahed. Stock market one-day ahead movement prediction using disparate data sources. *Expert Syst. Appl.*, 79:153–163, 2017.

[19] Wickramaratna, Jeevani, Holden, and Sean. Performance degradation in boosting. 2001.