Imperial College
Business School

Stock Market Prediction

Using Support Vector Machine

and

Modern Portfolio Theory

Lecturer: Professor Tarun Ramadorai

Course: BS1811 - Big Data in Finance

Robert Arnason CID: 01592337

30th April 2019

Contents

1 Introduction **1**

2 Background **1**

2.1 Data Set 1

2.2 Early Assumptions 2

3 Methodology **2**

4 Results and Analysis **6**

5 Conclusions **7**

5.1 Model Summary 7

5.2 Limitations and Improvements 8

6 Appendices **9**

6.1 Python Files 9

1 Introduction

The objective of predicting stock market movements is one that has challenged many specialists for years and will most likely continue to do so in the future. As new technology emerges, new opportunities come with it that can enable you to "beat" the market and profit. However, the market will quickly negate those technological advantages when sufficient amount of companies start implementing the same methods. Quick adaption and continuous improvement is therefore key to excelling when using technical approaches in stock market trading. The algorithms implemented in this report are in no way revolutionary, but a combination of two well established algorithms will hopefully deliver promising results.

The basic idea is to use a support vector machine algorithm to classify stock movement as either positive and negative. This is then fed into a modified version of the Markowitz portfolio optimisation model to get an optimal portfolio consisting of positively predicted stocks with a maximum variance allowance.

2 Background

2.1 Data Set

There are multiple companies that keep track of daily stock pricing with varying range of stocks that they cover. Initial search for data sets containing stock data returned multiple results to consider for this report. However, Most of them did not contain any data from this year and some consisted of so many stocks that the complexity of reading it in for processing would only be contributing to unnecessary extra work. After reading about other methods to access stock data, pandas datareader looked like a promising alternative that could access current and customised selections of stock data. Only problem with this is that most commonly used sources, Google Finance and Yahoo Finance, both changed their API recently and no open source python software was up to date for reading it. Tiingo, another provider of stock data was therefore chosen to be the best alternative with only a small subscription charge needed to gain access to the same detailed data through their API. One reason behind choosing Tiingo over other providers was the adjusted values provided in their database that accounts for both dividends and stock splits. This kind of adjusted data is often used when looking at historical stock data.

With this large selection of stock data now available, the subset chosen to use for this report was the S&P 500 companies. It was decided to extract ten years worth of stock data and use the adjusted closing prices to calculate the daily returns for each stock. These daily returns were then stored in a single data frame that would form the fundamental data for all further analysis.

2.2 Early Assumptions

After all the data had been gathered and processed, some assumptions were made before proceeding with the project. First of all, all assumptions related to modern portfolio theory have to be mentioned since this model is the foundation on which the investment strategy is built upon. These assumptions will however not be listed up here to save space for more relevant analysis.
Secondly, bid ask spread can vary greatly between stocks based on liquidity and popularity of the stock. Furthermore, it can vary on each individual stock based on the time of day. After some research into the behaviour and common characteristics of S&P 500 stocks, it was decided to assume that on an average the bid ask spread can be assumed to be 0.05% of the stock closing price. This would then be deducted from the daily profit of the portfolio on correct proportion to the portfolio changes made each day.
Finally, the transaction cost itself was assumed not to be relevant to the portfolio trading since they can be treated is practically nonexistent when sufficient amounts are being traded frequently enough. In other words, if a flat rate is paid on each transaction it can be lowered to a very low percentage if enough money is invested in the portfolio. Furthermore, some brokers offer deals to high frequency traders with large investment portfolios and this could lower the percentage to an even greater extent.

3 Methodology

The first required step to predict stock market direction was to calculate the daily returns of each stock. This meant taking the adjusted closing price each day and using it to calculate daily returns using the following formula:

$$\text{Daily Return}_i = \frac{\text{Adjusted Closing Price}_i - \text{Adjusted Closing Price}_{i-1}}{\text{Adjusted Closing Price}_{i-1}}$$

As mentioned before, the reason behind choosing adjusted closing prices is because it is more common when looking at historical data. The adjusted closing price accounts for the stock's dividends, stock splits and new stock offerings. So it provides analysts with a more accurate representation of the company's true equity value beyond the simple market price. This daily returns data frame was then used as the main input to predict the stock movement direction. It was decided to split it into training, validation and test sets to use in different phases of the project. These phases were tuning of hyper parameters for the support vector machine, tuning of custom parameters used in the portfolio optimisation and finally to test the final method on previously unseen data.
The algorithm used for this prediction is support vector regression or SVR. It was viewed to have some advantages over support vector classification even though the core problem is based on

classification into upwards and downwards stock movement. The reasoning behind choosing the regression version was to implement the regression outcome into the portfolio optimisation, but more on that later. Some work was put into feature engineering, to avoid only relying on the past stock returns for all stocks. The following attributes were constructed for the SVR model in hope of receiving a more accurate classification.

- Index Momentum: track overall movement of the S&P 500 index, 1 for up and -1 for down
- Index Volatility: calculated for N previous days to observe overall market volatility
- Stock Volatility: calculated for N previous days to observe the volatility of the stock currently being predicted

A label was also required for each stock prediction and was represented with 1 if the stock had positive returns and -1 if the return was negative. This would correctly represent the stock movement direction.
With the algorithm chosen, new attributes defined and the label created. Next step was to tune the hyper parameters, this was done using a rolling window validation and a grid search of each parameter combination. The parameters that had to be tuned were:

- window_size: size of past data window to be used to predict future return direction
- C: penalty parameter of the error term
- epsilon: defines a margin of tolerance where no penalty is given to errors
- hist: additional parameter that is not an input into the SVR function but a measure of how many previous days to use when calculating the index volatility and stock volatility mentioned above

The tuning of hyper parameters turned out to be the most computational heavy part of this project. The main cause of this is that each individual stock has its own tuning and training process. The rolling window validation is also very heavy to compute since there is an extremely high amount of models to fit and use to predict. To help with this, an AWS EC2 instance was used that could compute around the clock on a cloud server. It is worth mentioning that even using this cloud computing service, the parameters could not be tuned for every stock. The remaining stocks were assigned with the tuning parameters that had appeared most frequently. Some other methods might be better suited for the tuning phase if it would be conducted again, this will be further discussed later in this report.

With the hyper parameters tuned for every stock, or as many as possible, and stored in a data frame. They could then be used to predict the stock direction for each day in the validation set.

This is also done on a rolling basis where past data window is used to train and then a single day is predicted. This has to be done for every day in the validation set and for every stock that has future stock prices to compare with. These predictions are all gathered in a separate data frame.

The predictions are then fed into a modified Markowitz model that is used to determine the weights in the optimal daily portfolio. This has to be done using two models, the first model is used to initialise the portfolio on day one and the second model is used for all subsequent days. The first model has the following formulation.

$$\begin{aligned}
\text{maximise} \quad & \sum_{i=1}^{N} \mu_i SP_i w_i & & \\
\text{subject to} \quad & \sum_{i=1}^{N} w_i = 1, & & (1)\\
& w^T \Sigma w \le \overline{R} & & (2)\\
& w_i \le b_i, & \forall\ i \quad & (3)\\
& w_i \ge 0.0001 b_i, & \forall\ i \quad & (4)\\
& w_i \ge 0 \ \&\ b_i \in \{0,1\} & \forall\ i \quad &
\end{aligned}$$

Here N denotes the total number of stocks, so the summation covers all S&P 500 stocks. There are three parameters used in the model. $\overline{\boldsymbol{R}}$ is the maximum allowed portfolio variance and $\boldsymbol{SP}$ is a vector containing the stock predictions for a particular day that is being observed. The maximum variance was chosen by looking at the efficient frontier for the combined training and validation sets. Stock predictions are not percentages but a regression weights given when predicting the stock movement direction. $\boldsymbol{\mu}$ is a vector of expected returns for each stock and $\boldsymbol{\Sigma}$ is the covariance matrix. These values are calculated using past data and the validation set was used to determine how far back to observe that data. Finally there are two variables in the model, $\boldsymbol{w}$ is a vector of weights that will be assigned to each stock in the portfolio and $\boldsymbol{b}$ is a binary variable that triggers when the stock is to be included, the reason behind this will be covered in the constraints.
The objective function is based on the optimisation problem from portfolio theory where the objective is to maximise returns from the selected weights. There is however a small addition here where the objective is also multiplied with the regression weights. This means that given a choice between two stocks with the same expected return, a stock predicted to go up with a value of 0.99 is a more desirable stock to include in the portfolio over one that is predicted to go up at 0.1. The same works for negative numbers when predicting the stock to go down, so the optimisation model rewards and penalises in proportion with the confidence of that prediction. This will change the efficient frontier but will still deliver the best portfolio within a maximum risk limit.
The first two constraints are regular portfolio optimisation constraints, constraint 1 ensures that you always invest your total wealth and constraint 2 limits the total portfolio variance, $w^T \Sigma w$, at a predefined maximum allowed risk. Constraints 3 and 4 stops the solver from putting extremely low weights on some stocks, so the weights are either above 0.01% or 0%. This is mainly done for

convenience when reading the portfolio weights and to avoid rounding errors. The last constraints limit variables to be either non-negative or binary.

The second model is formulated to handle changes made to portfolio weights and tries to limit these changes. To do this some additional variables and constraints had to be introduced to the first model along with changes to the objective function. It has the following formulation.

$$\begin{aligned}
\text{maximise} \quad & \sum_{i=1}^{N} \mu_i SP_i w_i - \lambda(w_{ai} + w_{bi}) && \\
\text{subject to} \quad & \sum_{i=1}^{N} w_i = 1, && (1)\\
& w^T \Sigma w \leq \overline{R} && (2)\\
& w_i \leq b_i, \qquad \forall\, i && (3)\\
& w_i \geq 0.0001 b_i, \qquad \forall\, i && (4)\\
& w_{i-1} - w_i \geq w_{ai} - w_{bi}, \qquad \forall\, i && (5)\\
& w_i \; w_{ai} \; w_{bi} \geq 0 \quad b_i \in \{0,1\} \quad \forall\, i &&
\end{aligned}$$

One additional parameter has been added in this formulation, $\boldsymbol{\lambda}$, this parameter is a penalty term that lowers the objective function if changes are made in the portfolio weights between days. The changes are tracked by taking the absolute daily difference between every weight. This forces the program to make fewer changes to avoid losses from the bid ask spread cost. However, the objective function has to be linear. So to take an absolute value of the weights, two dummy variables had to be added into the formulation, $\boldsymbol{w_a}$ and $\boldsymbol{w_b}$. These dummy variables and constraint 5 ensure that the objective function will behave in the exact same way as if you were to take the absolute value between $\boldsymbol{w_{i-1}}$ and $\boldsymbol{w_i}$.

When these models had been properly set up and the code prepared, the predictions are fed into the model along with historical data. The resulting portfolio weights are then used to determine the success or failure of the model. This is done by calculating the profits or loss acquired over this period. The daily profit or loss from stock price changes was calculated separately from the bid ask spread transaction cost. The following formulas were used for these calculations. The profit formula is calculate for all days to get the daily profits and the cost formula is calculated from day 2 onwards to get the daily cost from changes made.

$$\text{Day k profit/loss} = \sum_{i=1}^{N} (\text{Stock Portfolio Weight}_{i,k} \cdot (1 + \text{Stock Daily Return}_{i,k}))$$

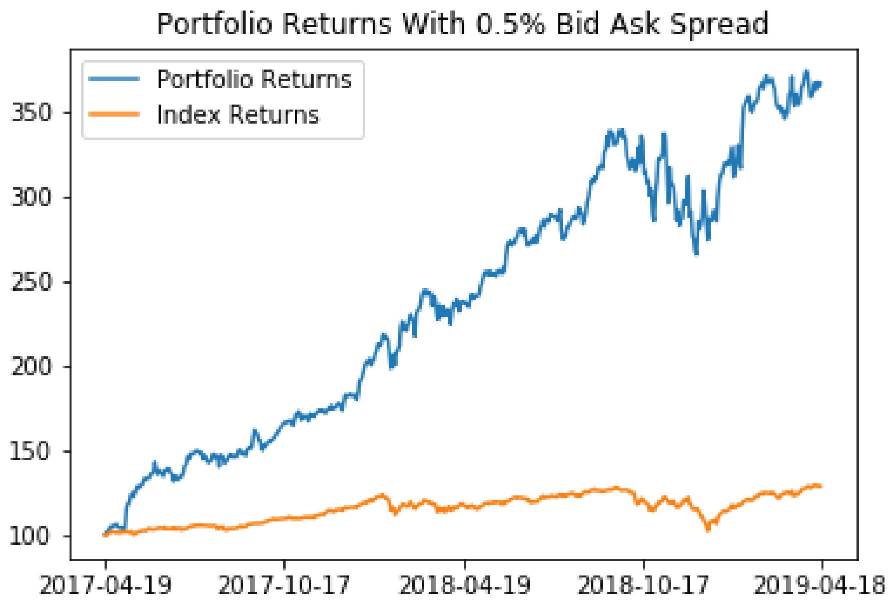
$$\begin{aligned}
\text{Day k Cost} = & \sum_{i=1}^{N} |\text{Stock Portfolio Weight}_{i,k} - \text{Stock Portfolio Weight}_{i,k-1}| \;/\; 2 \\
& - (\sum_{i=1}^{N} |\text{Stock Portfolio Weight}_{i,k} - \text{Stock Portfolio Weight}_{i,k-1}| \;/\; 2) \cdot 0.995^2
\end{aligned}$$

The net daily profit or loss is then accumulated over the whole prediction period to observe if the model has delivered over the entire period.

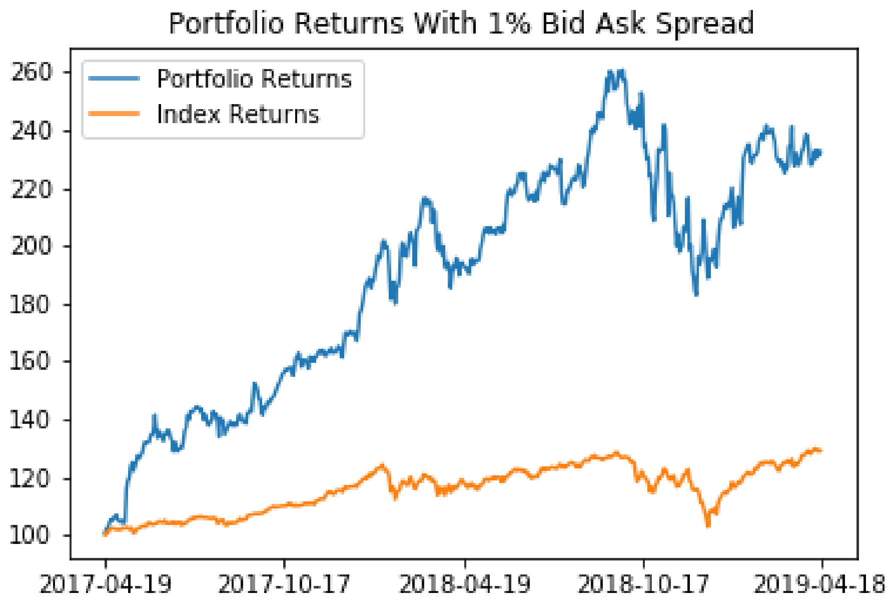
All these steps were first carried out on the validation set, with multiple runs of the optimisation model. Portfolio weights were extracted for different combinations of historical window size used to calculate expected return and the covariance matrix and the penalty term used in the objective function. These weights were then used to calculate profit or loss on every portfolio and the highest profiting combination of historical window and penalty term was kept to use on the test set.

4 Results and Analysis

The following results were generated from the test data after all models had been properly set up and parameters tuned based on the training and validation set. The parameters for the SVR algorithm were used to train one model for every stock, this model was then used to predict the daily stock movement direction of the test data. That data was fed into the portfolio optimisation model along with the best historical window size and penalty term. These parameters were previously tuned with the validation set and are 750 in window size and 0.001 in penalty. The resulting percentage return was 266.4% over the entire test period with the following graph showing the overall movement of the portfolio investment.



It can be seen from this graph that the profit turned out to be way above what could be considered normal, especially when compared to the index profit of 29.1%. This demanded further investigation into the whole method, but no cause for this abnormally high return was found. So most likely it can be explained with cost factors not being accounted for. The results are however very promising, with only a downturn in late 2018 when the overall index also took a dip. The return was also calculated with a higher bid ask spread to observe the behaviour with higher transaction cost. This resulted in an overall profit of 132.5% and can be seen in the following graph.



5 Conclusions

5.1 Model Summary

The aim of this project was to predict stock movement direction and create a market portfolio using these predictions. Like with all stock market predictions, the final objective is to maximise profits using the invested amount. The resulting portfolio showed great promise with an overall return of 266.4% and an approximate yearly return of 90%. This return is alarmingly good and must be taken with caution. To increase the confidence in this model, it was tested with a higher bid ask spread of 1% which resulted in an overall return of 132.5% or a yearly return of approximately 53%. This return is a bit lower but would still be considered a very good investment return.
Another point worth mentioning is the extreme difference in returns between the validation phase and the test phase. The historical window and penalty term chosen based on the validation set had a profit of over 3000% in the two year validation period. This should by all logic be an impossible return percentage, the reason is however based on a well known fact that wealth grows like a hockey stick. When you have higher capital, percentage increases start to have much greater effect on the increase in invested funds. The major problem is the wide gap between possible returns from the validation set and the final returns on the test set is so great that a bankruptcy is a possible outcome if you rely solely on the model in the current state.

5.2 Limitations and Improvements

There were many hours that went into tuning the hyper parameters for the support vector regression. To speed up this process, a cloud computing service from AWS was used. Initial tries with the cloud computing resulted in some time consuming and expensive lessons. However, the lessons learned in this process were many and good to have for future projects. Most of all that finding the tuning frame of the parameter can take a lot of time and is in truth tuning by itself. A different method that would be worth looking into is Bayesian optimisation, a short research into this method showed some promise but trials to implement this within the given time frame were not possible.

Another possible improvement is to look into increasing stability between validation profits and test profits. This could be made possible with some additional constraints or stability measures in the portfolio optimisation.

One financial limitation is that based on some online research, it can be difficult to buy and sell S&P 500 stocks directly. So having access to stocks with that high liquidity and in turn low bid ask spread, is maybe not a possibility. With higher transaction costs, it becomes harder to trade on a daily bases. This could however maybe be fixed by trading on a weekly bases using the SVR predictions.

6 Appendices

6.1 Python Files

- pre_process: notebook used to download and process stock data into the daily returns data frame.
- plot_EF: notebook used to plot up the efficient frontier. It was used to determine the maximum variance for the portfolio optimisation.
- final_project: notebook that includes the prediction of validation and test returns, portfolio optimisation and the final return plots.
- SVRtrain: Python script used to tune hyper parameters with cloud computing.