

Using Support Vector Machines to Create Profitable Trading Strategies

Name: Ciaran Martin

Student Number: 14338426

Final Year Project – **2017/2018**

B.Sc. Double Honours in Computer Science
(5 Credits)



**Maynooth
University**
National University
of Ireland Maynooth

Department of Computer Science

Maynooth University

Maynooth, Co. Kildare

Ireland

A thesis submitted in partial fulfilment of the requirements for the B.Sc.

Double Honours in Computer Science.

Supervisor: Dr. Phil Maguire

Contents

Declaration.....	i
Abstract.....	ii
List of Figures	iii
List of Tables	iii
Chapter one: Introduction	1
1.1 Motivation.....	1
1.2 Problem statement	1
1.3 Approach.....	2
1.4 Metrics	2
1.5 Project.....	2
Chapter two: Technical Background	3
2.1 Capial Asset Pricing Model.....	3
2.2 Long Short Equity Theory.....	3
2.3 Mean Reversion	4
2.4 Momentum	4
2.5 Support Vector Machines	4
Chapter three: The Problem.....	5
3.1 Attributes of a Successful Algorithm.....	5
3.2 Automated Stock Selection.....	6
3.3 Project UML Documentation	7
Chapter four: The Solution.....	7
4.1 Stock Feature Selection.....	7
4.2 The Quantopian Pipeline API	9
4.3 Machine Learning Approach	10
4.4 Model Selection	10
4.5 Refining the Model.....	14
Chapter five: Evaluation	15
5.1 Full Backtest	15
5.2 Algorithm testing	16
5.3 Algorithm Comparisons	16
5.4 Live-Testing	17
Chapter six: Conclusion	18

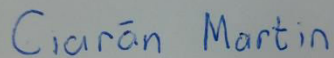
Appendices	21
Appendix 1 Code developed for this project.	21

Declaration

I hereby certify that this material, which I now submit for assessment on the program of study as part of the B.Sc. Double Honours in Computer Science qualification, is *entirely* my own work and has not been taken from the work of others - save and to the extent that such work has been cited and acknowledged within the text of my work.

I hereby acknowledge and accept that this thesis may be distributed to future final year students, as an example of the standard expected of final year projects.

Signed:

A rectangular box containing a handwritten signature in blue ink that reads "Ciarán Martin".

Date: 27/03/2018

Abstract

With the increase in popularity of automated algorithms on the stock market this paper discusses how statistical and machine learning methods can be combined to improve the performance of Long-Short Equity algorithms. Support Vector Machines (SVM's) are used in this paper as a form of non-linear regression to aid in predicting the movements of stocks and are the key component in the Long-Short Equity algorithm developed. All research and algorithm testing discussed in this paper was performed on Quantopian [1]. Quantopian is a free online Python-based platform which allows quantitative analysts to create their own trading algorithms assisted by their large repository of libraries and up-to-date data.

From working on several different algorithms it became clear that Support Vector Machines alone, are not sufficient in providing accurate predictions on what the next price of a stock will be. However when combined with sound finance principles such as Relative Strength Index (RSI) and Volume, a profitable algorithm was created which provided high returns on investment when back-tested and live-tested.

List of Figures

Figure 1	Support Vector Machine Separation.	5
Figure 2	Monthly investments in Amazon.com.	6
Figure 3	UML Sequence Diagram of Project.	7
Figure 4	Momentum vs Price Graph.	9
Figure 5	Good vs Bad SLR scenario.	11
Figure 6	SVR vs KRR.	12
Figure 7	SVR vs SLR vs Polynomial Regression.	13
Figure 8	Effect of SVR on Apple Stock Investment.	13
Figure 9	Backtest of Final Algorithm.	15
Figure 10	Full Backtest.	16
Figure 11	Results of Live-Testing.	17
Figure 12	Results of Only Other SVR Algorithm Found.	18

List of Tables

Table 1	Comparison of Algorithm Results.	16
---------	---------------------------------------	----

Chapter one: Introduction

“Electronic and algorithmic trading has become a significantly larger focus for financial institutions, securities regulators, and different exchanges” (Kim, Kendall, 2007, p. 1) [2]. Algorithmic trading opens up a whole new area of high frequency, complex trading which would be impossible for brokers. The ability to automatically analyse huge datasets can lead to much more secure investments and hedge funds which in turn is much more appealing to investors. The main trading strategy investigated in this paper is the Long-Short Equity strategy.

In a Long-Short Equity trading strategy the investor can take out long positions and short positions on securities. A security is a form of stock which can be traded in some exchange market. A long position is the simpler of the two, this is where you own the security and expect its price to rise. A short is more complicated but is essentially seen as the opposite of a long. A short position is the sale of a stock you don't actually own, when you believe the price will drop and you can then buy it at a later date for the reduced price. The idea is to take these types of positions out simultaneously to lower the overall exposure to the market trend. This type of strategy has been heavily employed since its inception in 1949 by A. W. Jones [3].

1.1 Motivation

The area of algorithmic trading is only increasing in size but is still in its early days with many traders still relying on media such as the news, Twitter, Facebook and company reports [4]. However it is my belief that a well-constructed algorithm, correctly utilizing statistical and machine learning techniques could outperform traders a vast majority of the time. The S&P 500 (Standard & Poor's 500) is an American stock market index of 500 large companies listed on the NASDAQ or NYSE [5]. In this paper I will be mainly focussing on stocks in the S&P 500 due to its consistent performance and readily available data. The goal of this paper is to create a trading algorithm which utilizes the available data in order to make predictions about future prices of stocks.

1.2 Problem statement

Over the years machine learning has come to solidify its position in the technology industry, including the quantitative finance sector. In this paper a number of machine learning algorithms are explored most significantly in the area of regression. Regression is a form of data analysis in which the data is represented by variables in an equation where the variable of interest (response variable) is expressed as a function of one or more predictors [6]. However regression is seldom used to predict out of sample data, thus several other factors are used in conjunction with the regression algorithms such as relative strength index (RSI), standard deviation, the momentum of a stock and the amount of the stock traded (volume). These factors combined will be used to filter the stock data and select an appropriate subset of stocks from which we intend to make predictions about using our regression algorithms.

1.3 Approach

Before finding success with regression algorithms many other algorithms were tested such as Pairs-Trading, where highly correlated stocks would be traded in pairs when one acted unlike the other. This type of trading didn't quite suit the S&P 500 companies as this window for arbitrage did not occur regularly enough for the highly correlated stocks. Mean reversion was also investigated. Mean reversion is the tendency for a variable to revert towards its mean over time or over repeated measurements. Many physical and economic variables are mean-reverting [7]. However working with a mean reversion strategy a higher frequency of transactions was encouraged which meant that there was less data to make predictions from between consecutive transactions, as well as heightened transaction fees. Regression algorithms took up most of the work on this project. Scikit learn was my choice of machine learning platform as it was supported by Quantopian [8]. Using various libraries in Scikit-learn, many different types of regression algorithms were tested such as: Simple Linear Regression, Kernel Ridge Regression, Autoregression (time series) and most successfully Support Vector Machine Regression. Each of these algorithms, where possible, were then rigorously back-tested on Quantopian. Back-testing on Quantopian is a form of simulation which tests how your algorithm would have performed in the stock market over a given window of time. The window of time that was chosen was from January 2012-Present. This back-testing was used to quantify the performance of each algorithm.

1.4 Metrics

Quantopian back-tests provide a wide variety of metrics which describe how your algorithm performed over the selected time period. These are: Total Returns, Benchmark Returns (S&P 500 is set as the benchmark to beat), Alpha (a measure of a portfolio's returns on investment), Beta (measure of how a portfolio moves compared to the benchmark), Sharpe (measure of algorithms performance accounting for its risk), Sortino (a variation of Sharpe that only includes negative volatility), Volatility (the degree of variation of a trading price over time) and Max Drawdown (maximum loss from a peak to trough of a portfolio, before a new peak). The goal is to maximise: Total Returns, Alpha, Sharpe and Sortino, while minimizing: Volatility and Max Drawdown, and to achieve a Beta value of close to 0 (+/-0.3).

1.5 Project

In the paper the final proposed algorithms all outperform the S&P 500 benchmark in returns by quite a large margin, one of which achieving >600% returns with promising trends for further growth. This high returns was also coupled with a high Sortino Ratio of 2.2 and an almost ideal Sharpe Ratio of 0.85 (1 or more is ideal). The Beta of the algorithm was 0.3, which lies just inside the bounds of what is considered acceptable by Quantopian. The Alpha value was quite strong giving a value of 0.36 (the algorithm over-performed the benchmark by 36%). However the max draw down for the algorithm was quite high at 33.56% but this appears to be due to an isolated case.

The algorithm also performed well during the S&P 500 crash at the start of February 2018, (where the S&P500 reported its worst monthly performance in 2 years) during which the algorithm turned a large profit.

The algorithm was live-tested from December to March during this time it achieved a strong performance with returns of 55% at the end of the 3 month period and a Sharpe ratio of 1.56 and Sortino Ratio of 3.2. Values such as Alpha and Beta require longer periods of live-testing to get a reliable value for so these are omitted here.

As the core of this final algorithm uses support vector machines extensively it is clear that there is merit to using this form of machine learning algorithm in conjunction with other methods to accurately determine the future value of stocks.

Chapter two: Technical Background

2.1 Capital Asset Pricing Model

The Capital Asset Pricing Model (CAPM) [9] gives a linear relationship between the mean return on a security or portfolio and the risk associated with it. It is defined as:

$$\mu_i = r + (\mu_m - r)\beta_i + \alpha$$

Where

μ_i = The mean return of security i ,

μ_m = The mean return of the market portfolio,

r = The risk free interest rate,

$$\beta_i = \frac{\text{Cov}(R_i, R_m)}{\sigma_m^2}$$

Where R_i is the rate of return of security i , R_m is the rate of return of the market portfolio and σ_m^2 is the variance of the market portfolio.

α = The excess returns generated by the model

Often an alpha value (α) is included as a constant in this model, this denotes the extra returns that are generated from correctly selecting long and short positions to increase the portfolios returns.

A strong characteristic for any portfolio is to have an overall β of ≈ 0 . This means that the mean return of the portfolio will be generated by the risk free interest rate and the alpha value. A good method of ensuring a β of ≈ 0 is by structuring your portfolio based on ideas from Long-Short Equity Theory.

2.2 Long Short Equity Theory

Long-Short Equity (LSE) Strategies [3] are typically good at reducing volatility in a portfolio as well as aiding it in remaining “market neutral” (β of ≈ 0). This is done by diversifying the portfolio and taking out an approximately equal number of long and short positions across different sectors thus mitigating the effect of the overall market movement on the portfolio. The use of a LSE strategy works well with the CAPM as LSE strategies tend to achieve beta values of close to 0 allowing us to discount most of the effect of the beta from our CAPM model leaving us with r and α .

2.3 Mean Reversion

Mean reversion as stated in section 1.4 is the tendency of a variable to revert to its mean. A specific example in the context of the stock market is a security with a high beta value in one time period tends to have a lower beta in the next time period [7]. This is also applicable to the price of stocks. A common technique is to get a moving average over a certain time window along with other descriptive statistics e.g. RSI or dollar volume, and if the stock price is trending below the average it is grouped with similar acting stocks and long positions are taken out on a selection of these. Similarly if the stock is trending above average we expect it to return to the mean so it is attractive for taking out short positions on. Ultimately after testing several algorithms, mean reversion was decided against in favour of a momentum strategy.

2.4 Momentum

Momentum measures the rate of increase or decrease in the price of a stock. Momentum is simply calculated as the current close price divided by the close price of a certain day in the past or a certain number of days in the past. The latter of these was used in the final algorithm as it gave data which was more suitable to use in regression models for prediction.

2.5 Support Vector Machines

The Support Vector Machines for regression library provided by Scikit Learn [10] was used as the main form of prediction in the final algorithm.

SVMs try to separate the data by support vectors, these are vectors which minimize a distance ϵ from some target y whilst minimizing the costs δ_i where higher costs are incurred by data points being outside these support vector boundaries.

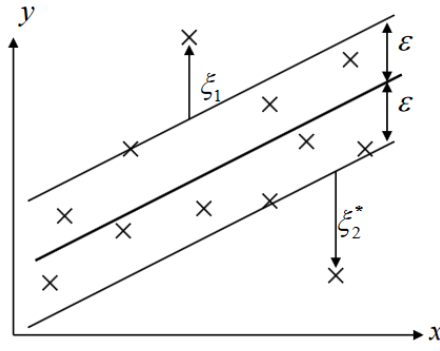


Figure 1. Support Vector Machine separation

As the data in this situation was non-linear a kernel function was used to transform the data into a higher dimension where it was separable and from here the linear separation needed for the support vector regression was performed. The kernel function used was the Gaussian Radial Basis Function (RBF). Altogether it produces the following model [11]:

$$\hat{Y} = \sum_{i=1}^N (\alpha_i - \alpha_i^*) \cdot K(x_i, x)$$

Where $K(x_i, x)$ denotes the Gaussian RBF below:

$$K(x_i, x) = \exp\left(-\frac{\|x_i - x\|^2}{2\sigma^2}\right)$$

Chapter three: The Problem

3.1 Attributes of a Successful Algorithm

Quantopian [1] outlines a list of attributes that they look for in successful algorithms. I created my algorithm with this in mind with the intention on having the algorithm realistic and applicable to the world of algorithmic trading. These attributes are as follows:

1. Beta

The algorithm firstly has it's correlation to the market calculated. On Quantopian this market is set as the S&P500. A portfolio with high a very high or very low value of Beta is highly correlated with the market and is as such more susceptible to market fluctuations. The ideal value of Beta according to Quantopian is in the range [-0.3,0.3].

$$\beta = \frac{\text{Covariance}(p_i, p_b)}{\text{Variance}(p_b)}$$

2. Hedging

The algorithm should be hedged to the market. This means that both long and short positions should be taken out. This leads to a lower Beta value and lower market risk.

3. Positive Returns

The algorithm must generate profit in trading. This can be tested two ways on the Quantopian platform, back-testing and live-testing. Back-testing is when the algorithm is placed in the market during a fixed time period in the past and its performance is recorded. Live-testing is when the algorithm is testing live on the day to day markets.

4. Activity

The algorithms should reweight their positions no more than twice a day and no less than once a month.

5. Volatility

Volatility is the amount of variation of a stock or portfolio's value over a period of time. It is measured using the standard deviation of the logged returns.

6. Maximum Drawdown

A portfolio with low volatility usually has a corresponding low maximum drawdown. This is the largest drop from the peak to trough of a portfolio's value.

7. Sharpe Ratio

The Sharpe ratio is a measure of volatility of the returns. The Sharpe ratio is calculated as the difference between Mean Portfolio Return and Risk Free Rate of Return all divided by the standard deviation of the portfolio's return.

$$Sharpe = \frac{\mu_p - r_f}{\sigma_p}$$

8. Sortino Ratio

The Sortino Ratio is calculated in a similar manner to the Sharpe ratio and also is a measure of volatility but it only accounts for bad risk. It is calculated as the difference between Mean Portfolio Return and Risk Free Rate of Return, all divided by the standard deviation of negative returns.

$$Sortino = \frac{\mu_p - r_f}{\sigma_n}$$

3.2 Automated Stock Selection

Quantopian states that all algorithms must be built on sound investment principles and not using any form of random selection or hindsight. Abusing hindsight can lead to algorithms that appear to be incredibly powerful, while in fact they are simply investing in securities that the programmer knows do well over the testing window.

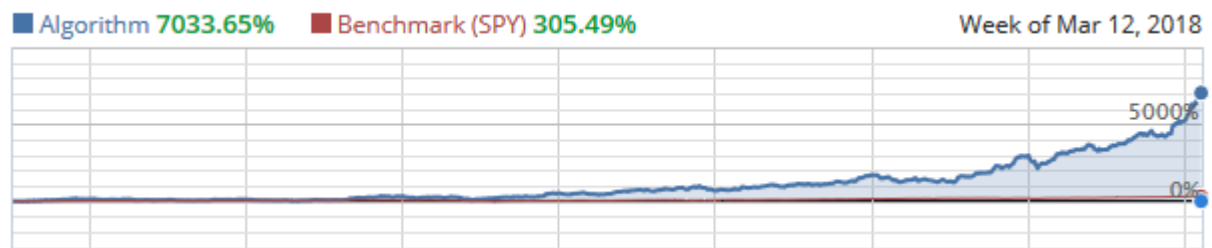


Figure 2. Resulting graph of investing in Amazon.com, Inc. every month from 2003-2018

In the above figure the algorithm was set up so that it invested its capital into Amazon every month for 15 years. This decision was made entirely due to hindsight to highlight the effect of selecting a subset of stocks manually to invest in and the deceptively successful results it provides.

To avoid this, the ideal algorithm should not use a small subset of stocks to invest in but rather a larger universe of stocks and by some characteristics calculated by the algorithm model a subset of this universe should be chosen and long or short positions taken out on each member of the subset. This approach will lead to better, more useful results as they are much more relevant to out of sample data when the algorithm begins the live-testing phase of development.

3.3 Project UML Documentation

The sequence diagram below describes a brief outline of how the algorithm will function on each monthly iteration. A large library of functions and APIs are provided by Quantopian. The Pipeline (discussed further later) is an API provided by Quantopian which is vital to any algorithm developed on the platform. It enables handling of data and heavy calculations to be performed at a much higher rate. This allows the use of computationally intense machine learning algorithms even for minutely trading. Quantopian also provides a wealth of data to its users, many of these are wrapped up nicely in functions which allow access too many variables such as close price, daily minimum value, daily maximum, volume of stock traded and more.

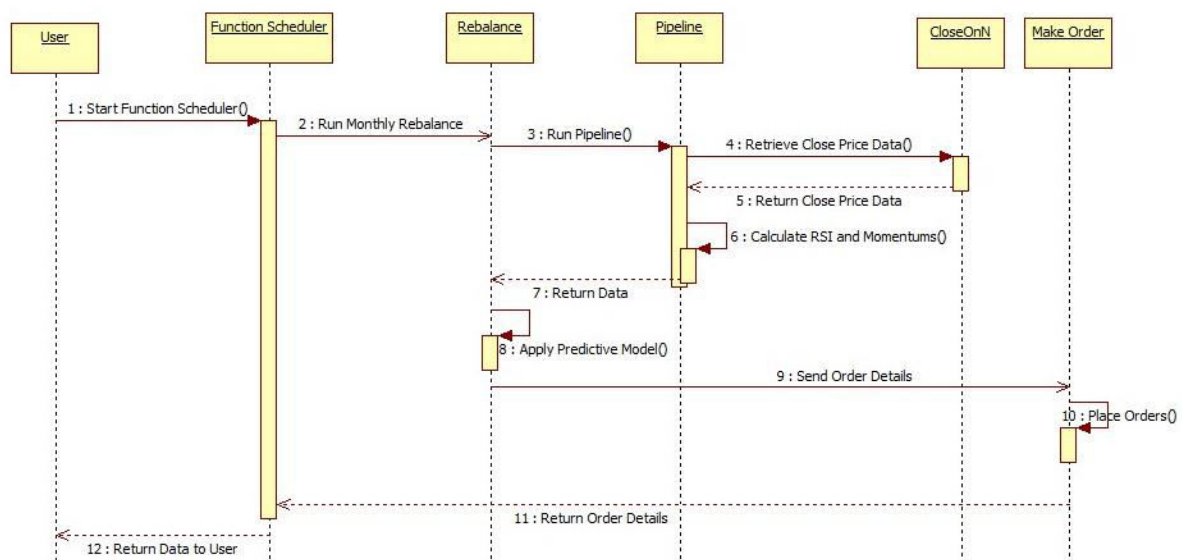


Figure 3. Sequence diagram of the algorithm running each iteration.

Chapter four: The Solution

4.1 Stock Feature Selection

Quantopian provides its users with many libraries of functions to derive descriptive statistics from the stock data. The simplest of these descriptive statistics are: Open price, Close price, Minimum price, Maximum price and Volume traded. Initially trends in the closing price of stocks were examined as a potential method of predicting where the stock price would move to next. Regression methods were applied to the closing price of stocks but it was found that often it would not be able to correctly identify which stocks would increase or decrease at a high enough rate to turn a

consistent profit, leading to many regression algorithms crashing at different points of the back-testing time period and not recovering.

An issue that arose regularly was attempting to purchase or sell more of a stock than was available to buy. Quantopian provides a function which allows you to invest a percentage of your portfolio into a certain stock, this is:

$$\text{order_target_percent}(\text{stock}, \text{value})$$

Where the stock denotes the stock of interest and value takes a number between -1 and 1. For positive numbers this would state how much of a portfolio is being invested in a long position in the respective stock and for negative numbers it indicates how much of the portfolio is being invested in a short position of the respective stock. Often this would lead to scenarios in which too high of a proportion of the portfolio would be invested into long or short positions of stocks as their volume tradable was just too low to fill the large order placed. As Quantopian requires all algorithms to run with \$10 million of capital for live-tests it was necessary to only trade in stocks that had a sufficiently high value of volume traded.

During the process of researching mean reversion algorithms a term that was seen on a number of occasions was “Relative Strength Index” (RSI). Relative Strength Index, developed by J Welles Wilder [13] is used as a measure of how quickly prices are moving. RSI is a scoring system between 0 and 100. At an RSI of less than 30 a stock is considered oversold, and at an RSI of over 70 a stock is considered over bought.

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

Where RS (Relative Strength) is,

$$RS = \frac{\text{Average } N \text{ Days Gains}}{\text{Average } N \text{ Days Losses}}$$

Overbought and oversold are often both consequences of news about some stock which causes a large spike or drop in interest. Both overbought and oversold stocks tend to recover their losses after a while. Overbought stocks are expected to decrease again in value whereas oversold stocks are expected to increase in price again. However activity in this area is unpredictable and due to restrictions on Quantopian RSI is only calculable from the current day and back N days as opposed to being able to calculate the RSI between two points in the past. This made it so the RSI could not be modelled by some form of predictive model so it wasn't possible to tell if the RSI was above 70 and increasing or decreasing and similarly when the RSI is below 30. This lead to too many incorrect long and short positions being taken out incorrectly.

After departing from mean reversion algorithms the next set of algorithms investigated were momentum trading algorithms [14]. Momentum is a measure of the rate of increase or decrease of a stock's value. It is a strong indicator of strength of a stock as you can model the momentums just as easily as you can model close prices however momentums might be less sensitive to random dips and spikes as it is calculated over a time period as follows:

$$\text{Momentum} = \frac{V}{V_{\mu}}$$

$V = \text{Current Day Price}$

$V_\mu = \text{Mean price over } N \text{ days } (N \approx 10)$

Momentum gives a more generalised idea of how the stock is performing as opposed to tracking close prices. Momentum lines, when plotted, appear smoother than then a line following close price trends (seen in the figure below). This smoother line allows us to take better inference from the data as it is more generalizable. For this reasons the regression models in this project are trained on past momentum values as opposed to past close prices.

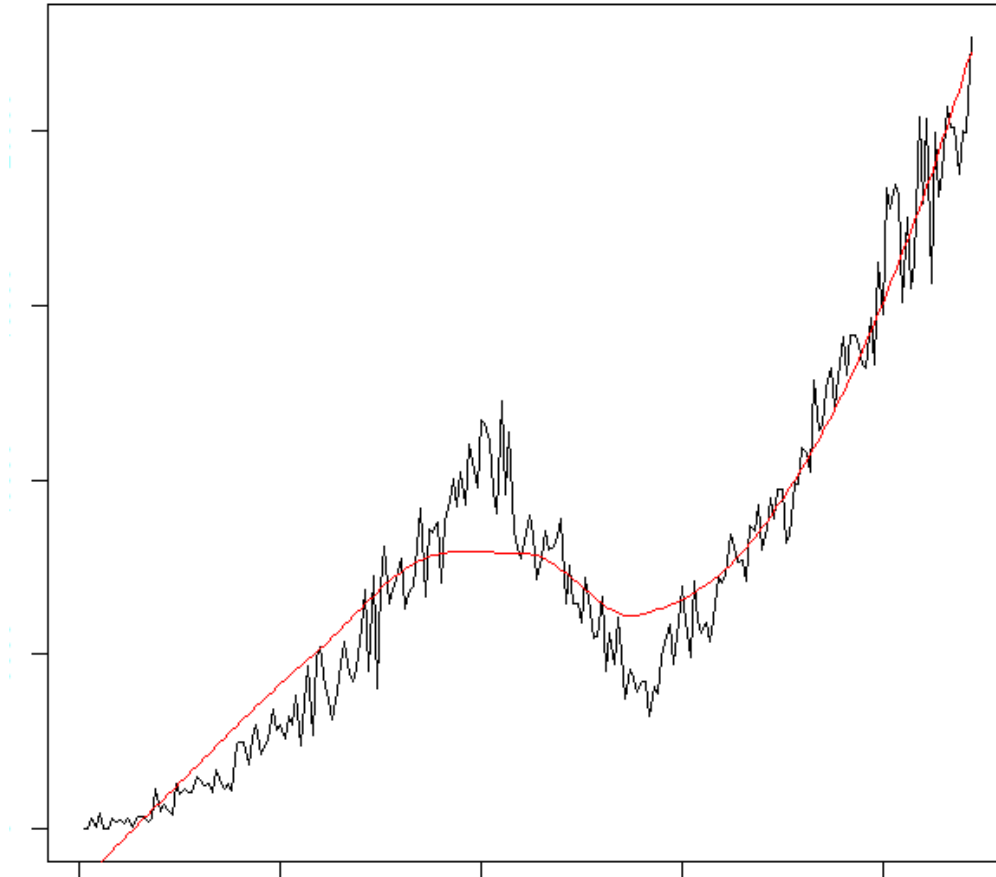


Figure 4. Line following close price (black), line following momentum (red).

4.2 The Quantopian Pipeline API

A large part of constructing any trading algorithm is data handling. Quantopian provides a great API for handling this called Pipeline [15]. This allows you to dynamically select stocks from a total universe of over 8000 stocks. A pipeline does 3 main types of computation: filters, factors and classifiers.

Filters return a boolean value so they handle certain functions such as, checking if a stock has a high enough average dollar volume. These filters are used to control the size of the output from the pipeline by only including those that pass the filter condition.

Factors are any function that returns an integer value. For example a factor could be simply returning the previous close price of a stock. There is a large library of factors provided to the coders by Quantopian these include functions like *Average_Dollar_Volume*, which finds the average volume of a stock traded over the past N days. Quantopian also allows users to create custom factors using the *CustomFactor* class, this gives a high level of customizability to the coder to make any form of factor. In the scope of this project a custom factor was used called *CloseOnN*, this custom factor would return the close price of a stock on a specific day in the past as opposed to a list of values from all the previous days. Each factor function also takes a *window_length* as input. This represents the period of time to look back on whilst making calculations

Classifiers take in the stocks and produce some form of categorical output. The main use of classifiers in pipeline is to group different stocks together for some larger computation (factor/filter). An example classifier would be labelling stock by sector i.e. label all stocks in technology with a 'T' and healthcare with 'H'.

The pipeline API uses Pandas Dataframes and Pandas Series to store the output [16]. Pandas is a library optimised to handle large datasets, it is built on top of the NumPy package [17]. NumPy is a popular package for python which is used to optimise data handling by efficiently and simply masking the map function in python to speed up applying functions to every variable in a python list. This is quite convenient as the machine learning library used in this project, Scikit-Learn, expects data in the form of 1-Dimension or 2-Dimensional NumPy arrays.

4.3 Machine Learning Approach

From the get go it was clear that machine learning methods would have to be used to optimise the results of this project. As a wealth of data and an API for optimising calculations is provided a machine learning approach is perfectly suited to handle this situation. The goal of the machine learning algorithm will be to interpret the current market trend and then act appropriately on this to maximise the returns of the algorithm.

Due to the highly variable nature of the stock market there are often small windows in which to exploit a trend or pattern in the market before it quickly disappears. The goal was to correctly tune and train a machine learning algorithm so that it will become proficient in noticing the correct changes in the market so it can exploit the trend before the window to do so closes.

The plan for the machine learning algorithm is to initially run the pipeline to generate the appropriate data and to cut down on the size of the dataset being used via filters and factors. Then taking this cleaned up data set apply a form of regression algorithm to approximate the next movement of the stock value. From there apply a weighting to each of these predictions between -1 and 1 which will be the proportion of the portfolio to invest in that stock.

4.4 Model Selection

When applying machine learning to a problem one of the biggest challenges is first selecting the type of algorithm to use and then selecting a suitable model. Originally regression machine learning was not going to be used in the project but after failing to successfully profit from a purely classification algorithm a regression approach was considered. Regression is a statistical method for estimating

relationships between variables [12], here the relationship of interest is between the past performance and the predicted future performance of stocks. The format of the data that was being passed to the regression algorithms was the momentum values over the past two month period.

Several regression algorithms were explored during the process of creating the final algorithm. Initially a Simple Linear Regression (SLR) approach was considered. This did not work very well however as there are essentially equally many situations where it is and isn't applicable to the data.

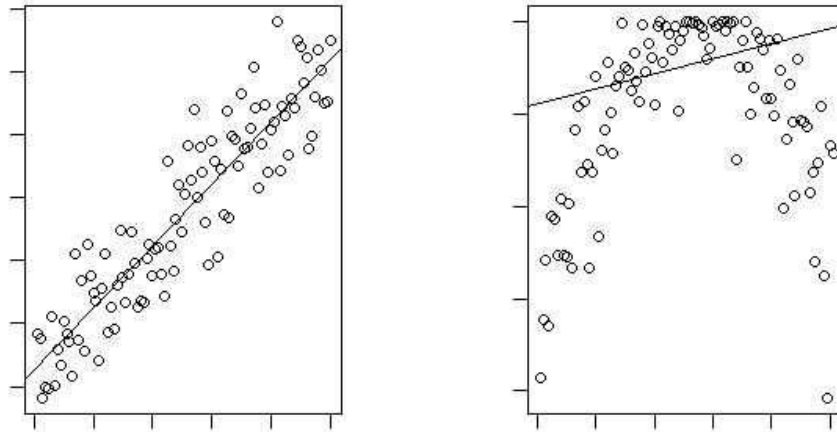


Figure 5. Left, good scenario for SLR. Right, bad scenario for SLR.

In figure three, it can be seen that if a SLR line was used to predict if the value of a stock was increasing or decreasing we would run into many scenarios when the value is actually decreasing but due to an ill-fitting model it would appear to the user as though it is increasing.

Since linear methods seemed inappropriate, other non-linear methods were investigated. Firstly multiple regression was looked at as it is known for handling non-linear situations with multiple predictors. The multiple regression model is as follows:

$$\hat{Y} = \beta_0 + \beta_1 \cdot X_1 + \dots + \beta_n \cdot X_n$$

Where each β_i is the coefficient corresponding to each predictor X_i . The approach was soon dropped as the strongest predictor was simply the previous momentum value of stocks and this lead to the multiple regression model behaving much the same as the simple linear regression model discussed above.

It was clear at this point that an algorithm better equipped to deal with this non-linear, time-series like data would be required. After researching regression models it seemed the best candidates were Support Vector Machines for regression and Kernel Ridge Regression. Both of these machine learning algorithms are supported by Scikit-learn but unfortunately both could not be tested as the version of Scikit-learn used by Quantopian did not include the Kernel Ridge Regression library. This would not be much of an issue however as the performance of the two algorithms is almost identical (see figure below).

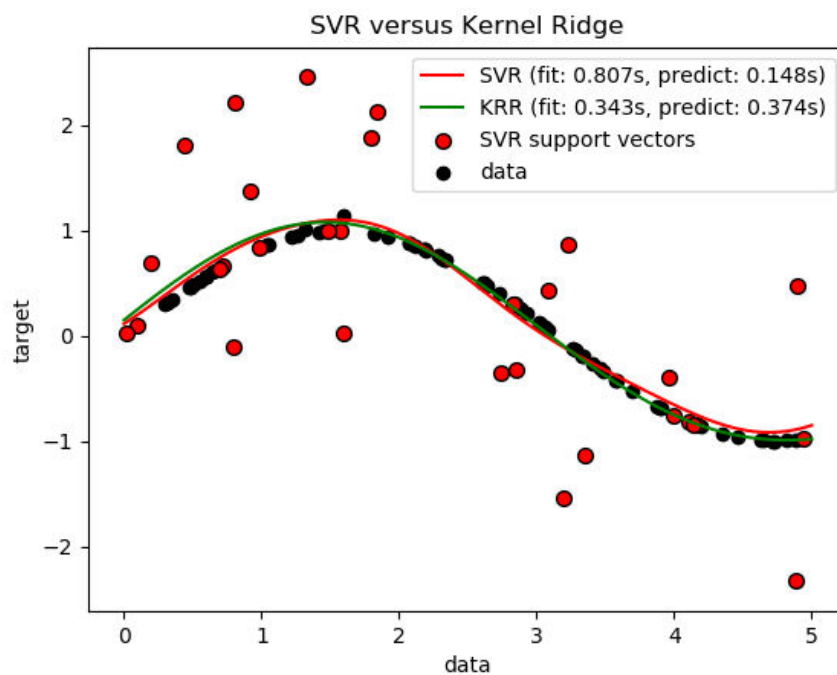


Figure 6. Scikit-Learn: Comparison of kernel ridge regression and SVR [8]

Support Vector Regression (as discussed above) uses Support Vector Machines and Kernel functions to make the data linearly separable in a higher dimension and transforming this back to the lower dimensions. This produces a non-linear regression line which appears to be suitable for modelling time series data. A SVR model outperforms simple linear and polynomial regression on non-linear test data by a large margin (see figure below).

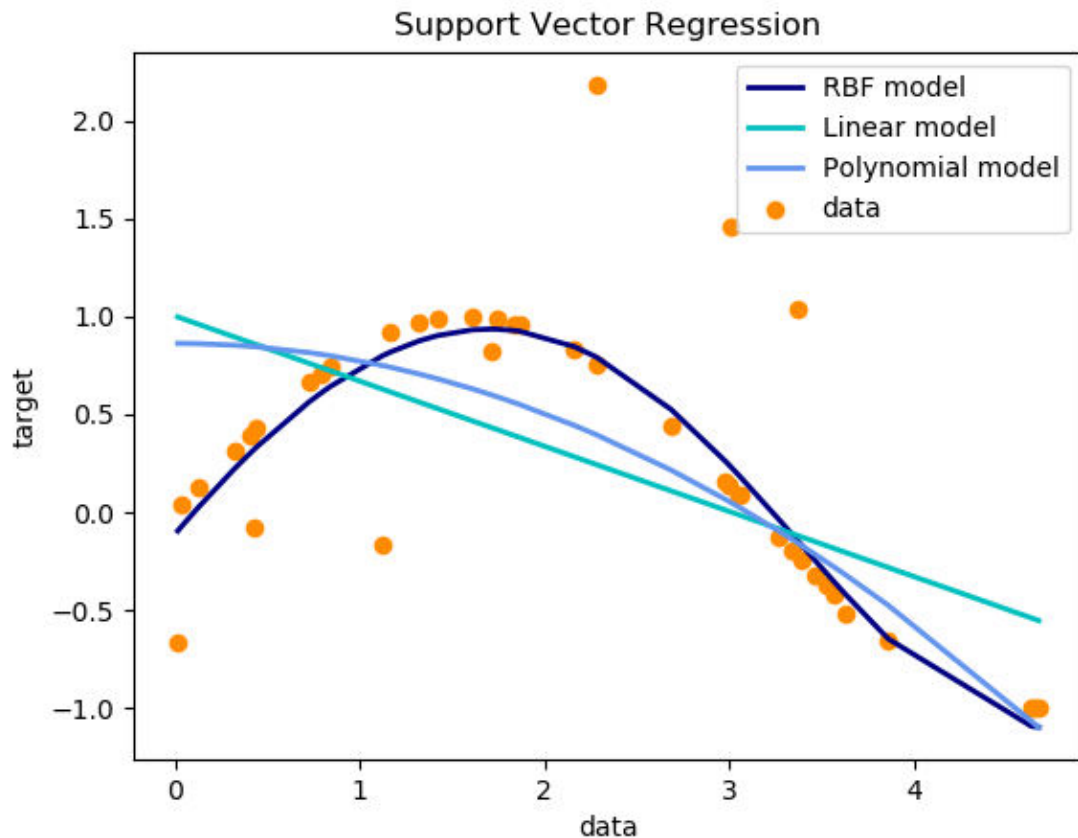
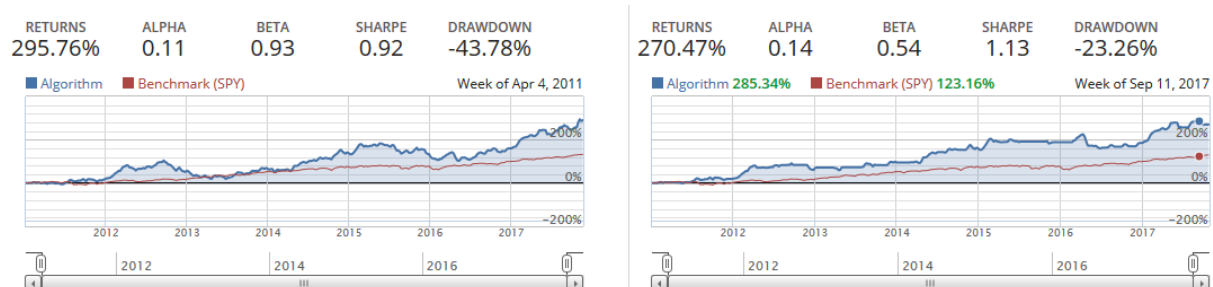


Figure 7. Scikit-Learn: Comparison of SVR vs Linear and Polynomial models [8]

Seeing the above performance a SVR model was created and tested on Apple to see how it performed on out of sample data. A test was set up where in one case Apple stock was purchased at the start vs. the second case where monthly the previous data would be fed into a SVR model and the position on the stock would be updated to reflect the result from the model. This test produced



the following results:

Figure 8. Apple stock only (Left) vs. Apple stock in SVR (Right).

The results of this test indicated that there is potential in a SVR model for predicting out of sample data at a reasonable level as we can see in the SVR results we have slightly lower returns but this is compensated by a higher alpha, a lower beta, a higher Sharpe ratio and a much lower maximum drawdown.

4.5 Refining the Model

Initially the algorithm did not perform to the standards I had hoped it would. However at this stage of development it traded weekly as well as using momentum measures that were calculated by $\frac{\text{Current Day Price}}{\text{Mean(previous 3 days prices)}}$. This was then changed to trade monthly and each value of momentum had a 2 weeks (10 trading days) worth of data to calculate a mean for the denominator. This provided us with more data to train our model on which is very appealing for a regression based model. The next step was refining the universe of stocks into a smaller subset that would perform better in the SVR model. The main issue that was happening was many stocks were acting unpredictably and the model was performing badly on the out of sample data of these stocks.

The first step in trying to fix this issue was tackled by sorting the stocks from the pipeline output by the standard deviation of their close prices in ascending order and testing the model when it only has to deal with low standard deviation stocks. This led to results that didn't make much returns. During this testing period it was noticed that Apple was among the stocks with high standard deviation of returns so following the earlier test on low standard deviation stocks the model was testing on high standard deviation stocks to see if this behaviour continued. In this case the returns were higher but there was huge max drawdown values >80% and the Sharpe Ratio took a large hit because of high level of volatility.

Relative Strength Index (RSI) was reintroduced at this stage as a potential fix to the model. A filter was added to the pipeline which would remove any stocks with an RSI of greater than 70 or less than 30 (i.e. remove any overbought or oversold stocks). Stocks outside of these levels of RSI often behave unpredictably so they did not seem suited to a regression model for prediction. After this alteration there was a good improvement in performance, returns were now beginning to surpass the benchmark S&P500 but the Sharpe ratio and max drawdown were still quite high.

A double model approach was tested to try mitigate some of the losses. This model involved the SVR model and a SLR model, the SVR model would make its prediction for the next momentum value of the stocks and the SLR model would predict the approximate next close price of the stock and if both the models predicted improvements or deterioration of a stock value then a corresponding long or short position would be taken on it. This model however was highly over-fitted and led to an overall algorithm that didn't trade very often as it was hard to satisfy both models simultaneously.

The next approach tested was the use of the RSI filter in the pipeline followed by the SVR model for predicting the next value but also include a factor of standard deviation of the momentum of the stock. This factor would be used in the following way:

1. Make the next prediction for each of the stocks
2. Calculate the standard deviation for each of the stocks
3. Check if the next prediction minus the standard deviation of the momentum was greater than the previous momentum value, if so, append that stock to the list of stocks for long positions.
4. Check if the next prediction plus the standard deviation of the momentum is less than the previous momentum value, if so, append that stock to the list of stocks for short positions.

Weighting the stocks was the next step in tuning the algorithm. After doing some research and reading it appeared as though an equal weighting strategy would be effective. In a Quantopian

forum post, a user Thomas Weicki performed extensive research on different weighting strategies and compared them on three main points: Returns, Sharpe Ratio and Volatility [18]. Equal weighting, when compared to other methods such as minimum variance and mean variance weighting, provides a high annual returns and one of the top Sharpe Ratios but has one of the higher values in volatility. This was reflected in the results on the algorithm where the Sharpe Ratio was acceptable but not great of approximately 0.76, returns were high at 415.44%, a good alpha of 0.38, a beta value well within Quantopian's acceptable range of 0.1 but a very high max drawdown of 52.11%.

The final hurdle in the tuning of this algorithm was to minimize this max drawdown and increase the Sharpe ratio. The first step taken in reducing this was to limit the amount of the portfolio that any single stock could take up. This limit was set at 20% of the portfolio. This change reduced the max drawdown to 41.51% which is still very high and reduced the returns to 330.84%. This change also altered the beta value, the beta after the backtest was now -0.23 while the alpha and Sharpe values stayed the same. The new beta value opened up the possibility of investing in the S&P500 to increase the beta value. This is appealing as the S&P500 is known as a very safe investment for the most part. Investing the remainder of the portfolio into the S&P500 whilst also limiting all stocks to a maximum of 20% of the portfolio had a strong effect on the results (see figure below).

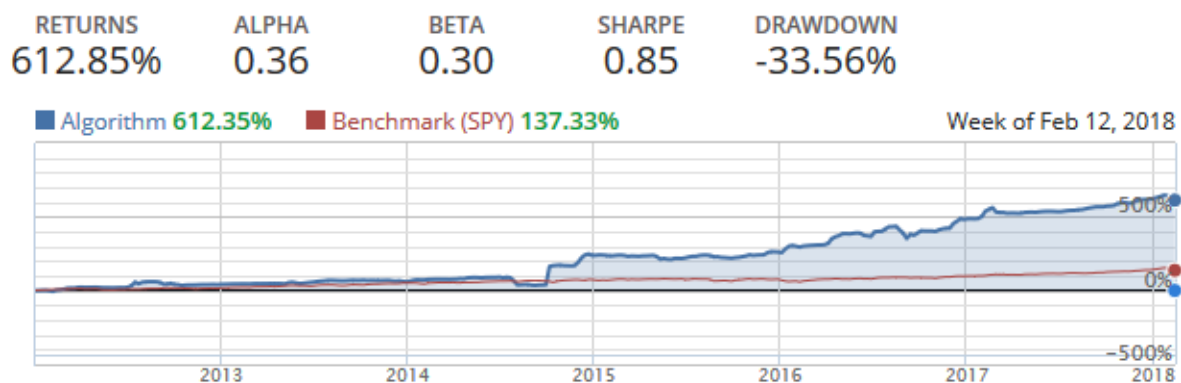


Figure 9. Results of a normal backtest of final algorithm.

The Sharpe ratio is now quite close to 1 which is a good Sharpe ratio, the beta value is just inside Quantopian's acceptable range and the drawdown decreased again and is now on 33.56%. However it appears that the drawdown is primarily caused by an isolated incident towards the end of 2014 there are no other comparatively large dips in the curve of returns.

Chapter five: Evaluation

5.1 Full Backtest

Quantopian provides its users with the ability to run full back-tests, this provides a wealth of detail on the performance of an algorithm as well as facilitating debugging and analysis of the algorithm. Day to day values for each descriptive statistic are provided as well as 1 month, 3 month, 6 month and 12 month highs. In order to add an algorithm to the Quantopian live-testing platform the algorithm requires at least one full backtest. For the full backtest the final algorithm discussed in section 4 was chosen. The results of the full backtest are displayed below.



Figure 10. Results of the full back-test from 2012-Present.

The final results of the algorithm came out quite strong with a very high returns compared to the benchmark returns, an alpha value of 0.36, which implies the portfolio outperformed the benchmark by 36%. A beta value of 0.3 which is within Quantopian's range of acceptable values indicates that the portfolio is very weakly correlated to the market and will not respond to swings in the market (i.e. a portfolio with a beta value of 1 moves in tandem with the market). A Sharpe ratio of 0.85 which is quite close to 1 which was the goal however I feel this could be improved upon by making changes to the model which are discussed later. A Sortino ratio of 2.2 which indicates that the algorithm returns are more than high enough to compensate for the volatility in negative returns. The value of volatility is somewhat high and possible steps to reduce this are discussed later. The maximum drawdown is 33.56% which occurs in the isolated incident visible in the graph above during August of 2014. Outside of the obscure 2014 dip the next highest drawdown is 20.43% which is a much more acceptable number.

5.2 Algorithm Testing

Testing for the algorithm had to take place on Quantopian's platform. Most of the verification testing was simply running back-tests to ensure it was doing as it should. Quantopian's platform however does not support any testing software or libraries, however the machine learning algorithm accepts only numbers as inputs and these numbers can be of any value. Each backtest run generates millions of data points so each backtest can be viewed as a very large test case. Due to the limitations in testing, a majority of the testing was bug fixing and validation testing on the live platform. One issue that was encountered was NA values appearing in pandas data frames generated by the pipeline output occurring due to missing values that would cause the support vector regression algorithm to crash. This was handled by excluding rows from the pipeline output which contained at least one NA value.

5.3 Algorithm Comparisons

Algorithm	Returns	Alpha (S&P500)	Beta (S&P500)	Sharpe	Max Drawdown
Benchmark	131.03%	0	1	1.2	12.14%
1	415.44%	0.38	0.1	0.76	52.11%
2	350.5%	0.36	-0.19	0.73	42.36%
3	612.85%	0.36	0.3	0.85	33.56%

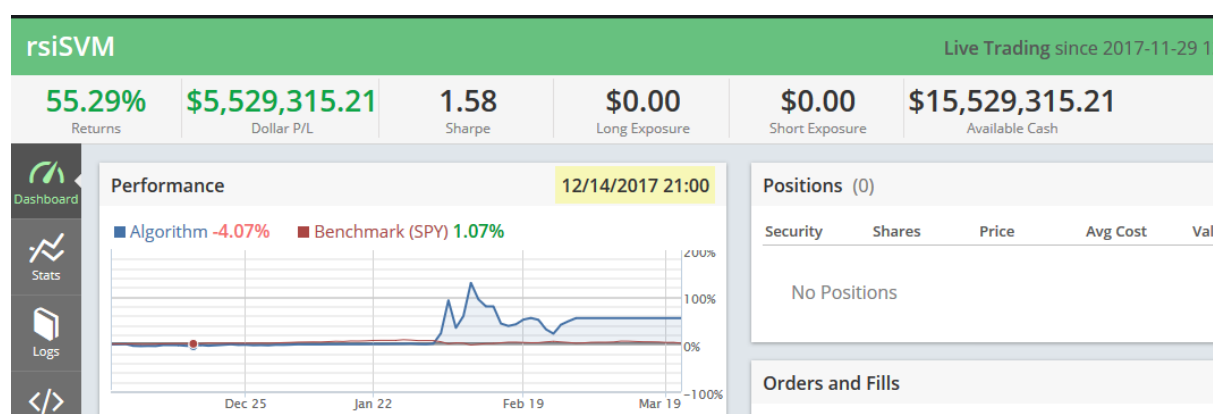
Table 1. Descriptive statistics of algorithms 1, 2 and 3 (described below)

1. This algorithm applies the machine learning model to the data without any of the later steps taken to lower the drawdown and increase the Sharpe Ratio.
2. This algorithm is the same as algorithm 1 but employs a limit on all stocks so that none of them can have more than 20% of the portfolio invested in them.
3. This algorithm is the final algorithm discussed above, it employs all of algorithms 1 and 2 but also uses the fact that the beta value was negative in algorithm 2 and as such an investment into the S&P500 was included.

After testing and comparing all the algorithms it was clear that algorithm 3 was most suited to live-testing.

5.4 Live-Testing

Quantopian provides a live-testing platform where it simulates an algorithm on the live market and produces up to date live results and descriptive statistics on a minutely basis as well as up to date graphics on a daily basis, updated at market close. The live-testing platform provides the following descriptive statistics of the portfolio: Alpha, Beta, Sharpe Ratio, Sortino Ratio and Volatility. The live-testing platform allows the users to perform software validation. However the only issue with the live-testing platform as a form of validation for the algorithm is that in order to get a realistic descriptive statistics you would require a long live-testing period. The algorithm was live-tested from



December 2017 till March 2018 with the following results:

Figure 11. Live-testing algorithm results, December 2017 – March 2018

From the live-testing we can see that the algorithm did in fact perform well during the live-testing period achieving very high returns for a three month period as well as a strong Sharpe ratio of 1.58. However the Sharpe ratio is likely swollen due to the high performance in February and a more realistic figure would be achieved over a longer live-testing period.

From this testing it appears as though the algorithm works and does turn a consistent profit which was the main goal of this project.

Chapter six: Conclusion

Previously there had been very little research into support vector machines for use on the stock market. Searching on the Quantopian forums only a single post in which a user used a support vector regression model was found. However this algorithm was not very successful (see below).

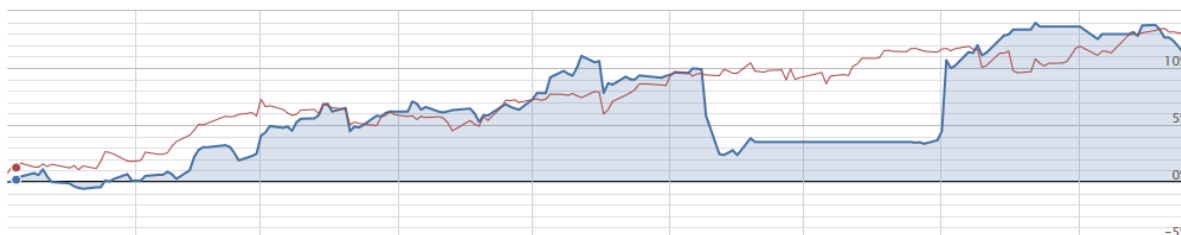


Figure 12. Only other SVR work found on Quantopian forums [19]

It appears as though any investor would be better off investing in the benchmark S&P500 here as opposed to using the provided algorithm. It appears as though the addition of finance principles such as relative strength index and momentum works very well with the statistical machine learning approach taken in this project. This project shines a good light on support vector regression as a predictive model for the stock market as the results of this model dwarfed some other algorithm's performances.

The live-testing of the algorithm provides promise that the algorithm isn't too overfitted as the performance in the 3 month period looked very strong. The algorithm even safely negotiated around the large dip in the market during the early weeks of February where the S&P500 fell 4.1% in one day (the largest fall since 2011) [20]. In fact the algorithm acted correctly and made profit during this turbulent time.

In order to be applied to other markets the algorithm most likely would have to be retuned in certain areas such as the factor of standard deviation to include in predictions and perhaps the tuning parameters of the SVR model. These parameters had to be tuned to optimise results in the test set and in every market the test data will be different so an exact copy may not be applicable to every market.

There algorithm has promise but one of the goals that it failed to reach was attaining a high Sharpe ratio. The goal was to reach 1 at least but it fell short at just 0.85. Future work in the algorithm would mainly be increasing the Sharpe ratio or reducing the maximum drawdown, in most circumstances improving one of these usually will have a positive effect on the other. A step that could improve the model could be to loosening it slightly more so it takes in a wider range of stocks when investing.

The model may be slightly overfitted but not by much as it is still performing well on out of sample data, however if it is correctly loosened and retuned to handle this new set up then it would allow greater diversification of the stocks invested in each more leading to a stronger hedged position and hopefully that would lead to a more secure investment. Hedging portfolios more often has the effect

of lowering returns but in this case however the returns are already quite high so there is definitely space to lower the returns to improve the stability of the algorithm as a whole.

As the stock market is highly variable and seeing as that characteristic will never change I feel that a statistical machine learning approach backed up by solid financial principles is definitely a sound approach to tackling investment strategies. As times move forward and further research is done in the area of support vector regression for predicting stock market movements, this algorithm only stands to improve as it already contains a solid foundation of a regression model for identifying stock market trends.

References

- [1] Quantopian.com, 2011. Retrieved from <https://www.quantopian.com>.
- [2] Kim, Kendall. Electronic and Algorithmic Trading Technology: The Complete Guide, Elsevier Science, 2007. ProQuest Ebook Central, p.1. Retrieved from <https://ebookcentral.proquest.com/lib/nuim/detail.action?docID=305560>.
- [3] Fung, W., & Hsieh, D. A. (2011). The risk in hedge fund strategies: Theory and evidence from long/short equity hedge funds. *Journal of Empirical Finance*, 18(4), pp. 547-569. Retrieved from <https://www.sciencedirect.com/science/article/abs/pii/S0927539811000211>
- [4] Renee Caruthers (2014). Twitter Trading Looks for Action, Traders, June 2014. Market Media.
- [5] *Standard & Poor's S&P 500 factsheet*, 2018. S&P Dow Jones Indices LLC. www.spindices.com/idsenhancedfactsheet/file.pdf?calcFrequency=M&force_download=true&hostIdentifier=48190c8c-42c4-46af-8d1a-0cd5db894797&indexId=340.
- [6] Black, J., Hashimzade, N., & Myles, G.(2017). regression. In A Dictionary of Economics. : Oxford University Press. Retrieved 3 Mar. 2018, from <http://www.oxfordreference.com.jproxy.nuim.ie/view/10.1093/acref/9780198759430.001.0001/acref-9780198759430-e-2639>.
- [7] Black, J., Hashimzade, N., & Myles, G.(2017). mean reversion. In A Dictionary of Economics. : Oxford University Press. Retrieved 3 Mar. 2018, from <http://www.oxfordreference.com.jproxy.nuim.ie/view/10.1093/acref/9780198759430.001.0001/acref-9780198759430-e-4135>.
- [8] Scikit-learn.org, "scikit-learn: machine learning in Python," 2016. Retrieved from: <http://scikit-learn.org/stable/>.
- [9] Levy, H. (2011). The Capital Asset Pricing Model. In *The Capital Asset Pricing Model in the 21st Century: Analytical, Empirical, and Behavioral Perspectives* (pp. 117-155). Cambridge: Cambridge University Press. https://www.cambridge.org/core/services/aop-cambridge-core/content/view/F3087AB0D77C44EABCCCFBED64C92472/9781139017459c5_p117-155_CBO.pdf/capital_asset_pricing_model.pdf
- [10] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011
- [11] Li, J., Li, G., Sun, D., & Lee, C. (2012). Evolution strategy based adaptive L q penalty support vector machines with gauss kernel for credit risk analysis. *Applied Soft Computing Journal*, 12(8), 2675-2682. [10.1016/j.asoc.2012.04.011](https://doi.org/10.1016/j.asoc.2012.04.011)
- [12] Yan, X. (2014). *Linear regression analysis : theory and computing*. Retrieved from

<https://ebookcentral.proquest.com>

- [13] N.I. Rudik (2011). *The Encyclopedia of the Indicator RSI (Relative Strength Index)*. Corporate Governance: The international journal of business in society, Vol. 13 Issue: 2.
<https://doi.org/10.1108/14720701311316698>
- [14] Badrinath, S. G., and Sunil Wahal. "Momentum Trading by Institutions." *The Journal of Finance*, vol. 57, no. 6, 2002, pp. 2449–2478. JSTOR, JSTOR, www.jstor.org/stable/3094533.
- [15] Quantopian.com, "Pipeline API", (2015). Retrieved from:
<https://www.quantopian.com/tutorials/pipeline>
- [16] Wes McKinney. "Data Structures for Statistical Computing in Python", Proceedings of the 9th Python in Science Conference, 51-56 (2010), pandas.pydata.org.
- [17] Travis E, Oliphant, "A guide to NumPy", USA: Trelgol Publishing, (2006),
<http://www.numpy.org/>.
- [18] Thoman Weicki (2016), "Hierarchical Risk Parity: Comparing various Portfolio Diversification Techniques", Quantopian Community. Retrieved from
<https://www.quantopian.com/posts/hierarchical-risk-parity-comparing-various-portfolio-diversification-techniques>
- [19] Luc Prieur (2017), "Algo with Support Vector Machine in Pipeline", Quantopian Community. Retrieved from <https://www.quantopian.com/posts/algo-with-support-vector-machine-in-pipeline>
- [20] Rapoza, K. (2018), "Here's how much more the S&P500 has to fall before we're in a bear market". Retrieved from <https://www.forbes.com/sites/kenrapoza/2018/02/08/heres-how-much-more-the-sp-500-has-to-fall-before-were-in-a-bear-market/#1c4e127e3991>

Appendices

Appendix A: Source code for the project