Winter 3-14-2013

A Survey of Systems for Predicting Stock Market Movements, Combining Market Indicators and Machine Learning Classifiers

Jeffrey Allan Caley
Portland State University

10.15760/etd.2000

A Survey of Systems for Predicting Stock Market Movements, Combining Market Indicators and Machine Learning Classifiers

by

Jeffrey Allan Caley

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science
in
Electrical and Computer Engineering

Portland State University
2013

Abstract

In this work, we propose and investigate a series of methods to predict stock market movements. These methods use stock market technical and macroeconomic indicators as inputs into different machine learning classifiers. The objective is to survey existing domain knowledge, and combine multiple techniques into one method to predict daily market movements for stocks. Approaches using nearest neighbor classification, support vector machine classification, K-means classification, principal component analysis and genetic algorithms for feature reduction and redefining the classification rule were explored. Ten stocks, 9 companies and 1 index, were used to evaluate each iteration of the trading method. The classification rate, modified Sharpe ratio and profit gained over the test period is used to evaluate each strategy. The findings showed nearest neighbor classification using genetic algorithm input feature reduction produced the best results, achieving higher profits than buy-and-hold for a majority of the companies.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

The Efficient Markets Hypothesis (EMH) [1] states that the price of a stock reflects all known information, therefore only new information can cause a change in price. Since the arrival of new information is unpredictable, prediction of market prices is only possible when nonpublic information is known by the trader. This hypothesis also states that timing the market is not possible, which supports the well known investment strategy buy-and-hold, buying a stock and never selling it.

The EMH faces resistance from researchers and traders alike. Behavioral finance suggests that cognitive biases cause inefficiencies, letting rumor and speculation trump the importance of the known information [2] [3]. The abundance of highly successful investors, such as Warren Buffet and George Soros, suggests the existence of inefficiencies that can be exploited for investment success.

Three approaches are commonly used to predict price movements for a given stock: fundamental analysis, technical analysis and quantitative analysis. Fundamental analysis involves analyzing the financial statements, management, competitive advantage, competitors and markets of a company to determine the fair market value of a business [4]. Fundamental economic indicators are metrics that give a view of the economic health of a nation or global economy in which the business participates [5]. Using these factors, an investor can determine a fair value for a business and make an educated prediction of the future stock price of a company.

Technical analysis uses past market data, mainly price and volume information, to make predictions of future market movements [6]. Price and volume is put through mathematical transformations to convert the data into easily graphed and understood indicators. These indicators are thought to show graphical patterns and trends that can help predict future price. Trading decisions can then be made based on these trends.

Quantitative analysis uses numerical or quantitative techniques to derive an answer for most areas of finance. Trading strategy development, portfolio optimization, derivatives pricing and hedging, risk management and credit analysis [[7] are some of the areas quantitative analysis is used. Using techniques such as statistical arbitrage, automated trading and electronic market making, quantitative analysis removes the human element of trading and replaces it with statistical market models [8].

The purpose of this work was to explore multiple techniques for daily stock market predictions using a combination of technical, fundamental and quantitative analysis. Combining common technical and macroeconomic indicators with statistical techniques used in quantitative analysis, we looked to predict daily market movements using machine learning classifiers. Using these predictions as a guide, a trading strategy was developed to buy and sell stocks.

The results of this work show that using features comprised of technical indicators can be used as inputs into machine learning classifiers to create a trading strategy that outperforms the buy-and-hold strategy. Using a portfolio of 10 stocks as an evaluation set, the proposed trading strategy uses a genetic algorithm to select a subset of 52 features to be used as inputs into a K-nearest neighbors classifier. Performing this feature optimization on each stock in the evaluation set resulted in an average profit of 106.1%, 97.5% more than achieved using the buy-and-hold strategy.

1.1 Problem Statement

Where and how to invest one's money is a problem that many individuals face. Despite the large amount of money and interest embedded into this problem, there is still no definitive answer on when and where money should be invested. Due to this, most people either keep their money in the bank or hand it off to someone else to manage. This work looks to develop an automated trading system that can be used to make more money off trading securities than the traditional buy-and-hold strategy and to reduce the risk involved in making these investments.

1.2 Objective

The objective of this work is to develop a market trading model that can successfully trade market securities for a profit, beating buy-and-hold. This was accomplished through the construction of a market trading simulator and the exploration of many different trading models. The models will be evaluated by classification rate, profitability compared to buy-and-hold and the modified Sharpe ratio.

1.3 Thesis Format

The following includes a literature survey of background information and related techniques of market data classification (Chapter 2); a description of goals, hypothesis and evaluation methods (Chapter 3); an explanation of design (Chapter 4); and a thorough description of all experimental trading strategies with results (Chapter 5).

Chapter 2: Background Information and Literature Overview

There are many different techniques used to predict stock market movements. This chapter serves as an overview of stock market terminology, definitions of common macro and technical indicators, and previous research done in the field of stock market prediction using computational intelligence techniques.

2.1 Technical Analysis

In finance, technical analysis is the study of financial market movements. A technician is someone who performs technical analysis. They use past price and volume data displayed in graphical form to make decisions about where financial markets will move in the future [6]. Technicians believe that all information related to the stock is reflected in the price. Over the years, technicians have found many different ways to represent price and volume data to help simplify decision points. These representations are referred to as technical indicators.

This work uses many common technical indicators as inputs to classifiers. Technical indicators have been shown to be a valid representation of price and volume data that computational classification techniques can use to successfully classify stocks movements [9].

2.1.1 Moving Average [10]

There are two common types of moving averages; simple moving average and exponential moving average. A simple moving average (SMA) is calculated by computing the average price of a security over a specified number of periods. This process is repeated for each day, forming its own time series. The resulting formula is:

$$SMA = \frac{P_c(t) + P_c(t-1) + ... + P_c(t-n-1)}{n}$$

Where P_c is the closing price of a given day and n is the period in which the moving average is being calculated. Exponential moving average (EMA) is calculated by taking a weighted average of past prices. As prices get farther into the past, they are weighted less and less. This means more recent values of the security effect the EMA result more than past values. EMA is calculated by:

$$EMA = K(P_c - EMA(t-1)) + EMA(t-1)$$

Where,

$$K = \frac{2}{1+n}$$

n is the period of the EMA and P_c is the close price of the current day. To calculate the first EMA value, a SMA is used.

2.1.2 Moving Average Convergence/Divergence (MACD) [11]

The MACD is a collection of three signals that are calculated using exponential moving averages. First, the MACD, which is the difference between a 12 day and a 26 day exponential moving average. Second, the signal, which is the 9 day EMA of the MACD. Finally there is the histogram, which is the difference between the MACD and the signal. It is also referred to as Price Oscillator when the MACD has different values for the moving averages. It is used to identify changes in the strength, direction and momentum of a security.

$$MACD = EMA(12) - EMA(26)$$

$$Signal = EMA(MACD, 9)$$

$$Histogram = MACD - Signal$$

2.1.3 Relative Strength Index (RSI) [12]

The RSI is a momentum indicator which is between values of 0 and 100. RSI is used to measure the velocity and direction of price movements. It is calculated using the equations below.

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

Where,

$$RS = \frac{EMA(U, n)}{EMA(D, n)}$$

For trading periods that experience an upward change use

$$U = P_c(t) - P_c(t-1)$$

$$D = 0$$

Conversely, for a period that swings downward, use

$$D = P_c(t-1) - P_c(t)$$

$$U = 0$$

2.1.4 Bollinger Bands [13]

Bollinger Bands are bands relating to volatility placed below and above a moving average. Stock price interaction with these bands is thought to reveal information about price movements in the future. Bollinger Bands consist of a simple moving average, an upper Bollinger Band and a lower Bollinger Band.

$$MA = SMA(n)$$

$$BB_{Upper} = MA + K\sigma$$

$$BB_{Lower} = MA - K\sigma$$

Where σ is the standard deviation of P_c over the past n days and K is a selected multiple. Commonly used values for K and n are 2 and 20, respectively.

2.1.5 Stochastics [14]

The Stochastic Oscillator is a momentum indicator that refers to the current price in relation to its price range over a period of time. It is composed of two lines, the %K and the %D. These lines represent predicted turning points for the price of a security[14]. They are calculated:

$$\%K = 100\frac{P_c - P_L(n)}{P_h(n) - P_L(n)}$$

$$\%D = EMA(\%K,3)$$

Where P_c is the close price of the current day and P_L and P_h are the low and high price over the past period n.

2.1.6 Momentum and Rate of Change [15]

The rate of change is a simple technical indicator calculated by taking the difference between today's closing price and the price of the same security n days ago. This is then scaled by the older closing price. Without this scaling, it is called momentum.

$$Momentum = P_c - P_c(t-n)$$

$$ROC = \frac{P_c - P_c(t-n)}{P_c(t-n)}$$

2.1.7 Moving Variance [16]

Variance is a measure of the amount of variation from the mean in a time series. Moving variance is a moving average of variance values. To calculate variance, first calculate the mean of n number of points. Then find the difference between each point n and the mean, squaring the result. Finally, average these differences to find the variance.

2.1.8 Commodity Channel Index (CCI) [17]

Commodity Channel Index measures a security variation from its statistical mean which is used to identify cyclical trends. It is calculated by taking the difference between the current price of a security and its SMA, divided by the mean absolute deviation of the price.

$$CCI = \frac{P_t - SMA(P_t)}{(0.015)\sigma_{mad}(P_t)}$$

Where,

$$P_t = \frac{P_h + P_L + P_c}{3}$$

σ_{mad} is the mean absolute deviation of P_t.

2.1.9 Chaikin Oscillator [19]

The Chaikin Oscillator is a technical indicator that is used to relate price and volume data for a particular security. It is defined as the difference between the 3 day EMA of the Accumulation Distribution Line and the 10-day EMA of the Accumulation Distribution Line. Defined as the following:

$$Chaikin - Oscillator = EMA(ADL,3)$$

Where,

$$Accumulation - Distribution - Line(ADL) = ADL(t-1) + MFV$$

$$Money - Flow - Multiplier(MFM) = \frac{(P_c - P_L) - (P_h - P_c)}{(P_h - P_L)}$$

$$Money - Flow - Volume(MFV) = MFM * v$$

2.1.10 Disparity Index [20]

The disparity index is a percentage of the latest closing price compared to a specific moving average.

$$DI = \frac{100(P_c - SMA(P_c, n))}{SMA(P_c, n)}$$

2.1.11 Williams %R [21]

The Williams %R shows the current price of a security in relation to its high and low of the past n periods. It is used to see where the current price is in relation to its recent high or low. It is calculated by:

$$\%R = 100\frac{P_{h(n)} - P_c}{P_{h(n)} - P_{L(n)}}$$

where n is the period in which to move backwards in time to find the specific high or low price, denoted $P_{h(n)}$ or $P_{L(n)}$.

2.1.12 Volatility

Volatility is a measure of variation of price over a period of time. It is calculated by taking the standard deviation of n closing prices divided by a simple moving average of n days. In this study, 21 is used for n.

$$Volatility = \frac{\sigma(P_c, n)}{SMA(P_c, n)}$$

Where σ is the standard deviation.

2.2 Macroeconomic Data

Macroeconomic indicators are statistics that are used to evaluate the current state of an economy. These statistics are used to track the health of the economy overall and can help guide economic policy decisions, such as changing interest or tax rates [[22]. The overall health of an economy can have significant impacts on specific securities prices, making macroeconomic data an invaluable big picture resource.

Macroeconomic indicators:

2.2.1 United States (US) Federal Reserve Interest Rate

The interest rate at which depository institutions trade with each other. This rate is set by members of the Federal Open Market Committee [22].

2.2.2 Canada - US Exchange Rate

The rate in which Canadian currency is exchanged for US currency.

2.2.3 Japan - US Exchange Rate

The rate in which Japanese currency is exchanged for US currency.

2.2.4 Swiss - US Exchange Rate

The rate in which Swiss currency is exchanged for US currency.

2.2.5 US - Euro Exchange Rate

The rate in which the Euro is exchanged for US currency.

2.2.6 US - Pound Exchange Rate

The rate in which the British Pound is exchanged for US currency.

2.2.7 Three Month T-Bill Rate

The interest rate associated with a three month debt obligation backed by the US government.

2.2.8 Six Month T-Bill Rate

The interest rate associated with a six month debt obligation backed by the US government.

2.2.9 Trade Weighted US Dollar Index, Broad

A weighted average of foreign exchange values of the US dollar against the currencies of a broad group of US trade partners [23].

2.2.10 Trade Weighted US Dollar Index, Major

A weighted average of foreign exchange rates; the US dollar and a subset of the countries included in the broad [24]. Currencies included are: Euro Area, Canada, Japan, United Kingdom, Switzerland, Australia and Sweden [24].

2.3 Classifiers

There are many well known machine learning algorithms that can be used to classify a problem given a set of features. This work looks to survey some of these algorithms to see if any are particularly useful in classifying stock market data into ‘up’ or ‘down’ periods given a set of inputs generated through macroeconomic data and technical analysis. Classifiers investigated are:

2.3.1 K-nearest Neighbor (KNN)

The K-nearest neighbor classifier is one of the simplest machine learning algorithms. An object is classified by looking to its nearest examples, measured using any distance metric, and using a majority vote [9] to decide what class to assign the object. K states how many neighbors will be used in this vote. K=1 simply states that an object will be assigned the same class as its nearest example. This work implements the KNN using the function ‘knnclassify’ supplied by the Bioinformatics Toolbox in Matlab.

Simple Example:

Given the training set seen in table 2.1, a two dimensional graph is constructed. The + represents class 1 while the O represents class 0 of the binary classification problem. When new points are looking to be

Training Observations		
X	**Y**	**Class**
4	3	0
1	4	0
6	6	0
2	10	1
9	8	1
0	6	1
10	4	0
1	1	1

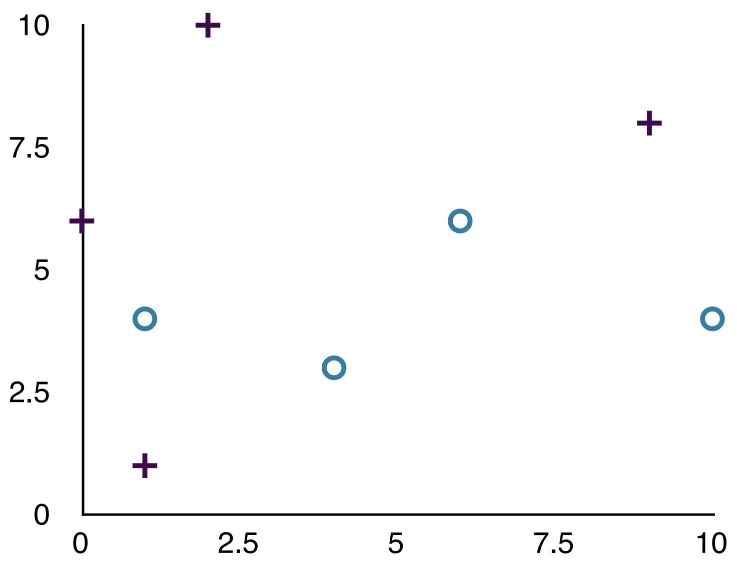

Table 2.1: Example Training observations graphed in XY plane

classified, represented as a square in Table 2.2, their K nearest points are looked to for identification. In this example, Euclidian distance will be used with K = 3. For test observation (4,4), graphically it can be seen in Table 2.2 that two of its three nearest neighbors are points (4,3) and (6,6). The final nearest neighbor could either be point (1,1) or (1,4). Euclidian distance can be measured for both of these points to determine which is closest. For point (1,1), the distance is calculated as $\sqrt{3^2+3^2}=4.24$ distance. For point (1,4), the calculation is $\sqrt{3^2+1^2}=3.16$ distance. Point (1,4) is therefore the 3rd nearest neighbor. Because all 3 of the neighbors to point

Test Observation s	
X	**Y**
4	4
2	8

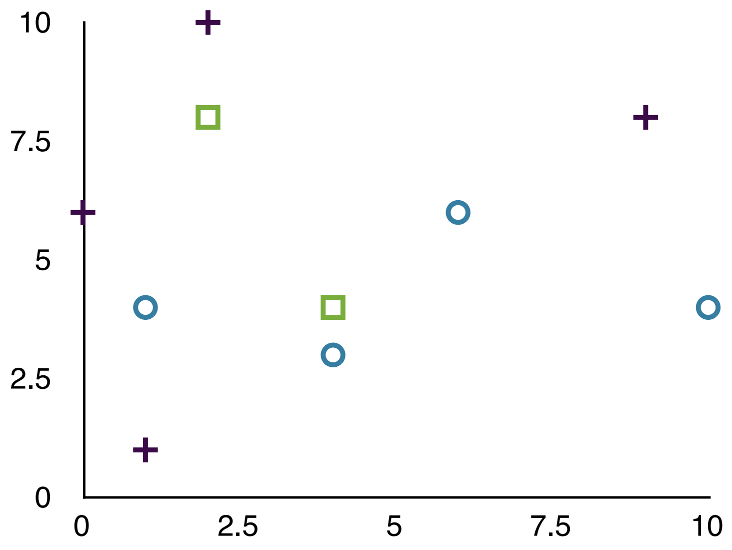

Table 2.2: Example test observations graphed in XY plane

(4,4) are of class 0, point (4,4) will be assigned to class 0. For point (8,9), a similar technique can be used to find its nearest neighbors, which are (2,10), (0,6) and (1,4). Two out of the 3 points are in class 1, therefore point (2,8) will be assigned class 1.

2.3.2 K-Means Clustering

K-means Clustering is an unsupervised training method that looks to partition the data space into K clusters. Once the data space has been partitioned into K clusters, each cluster can be assigned a class based on the majority vote of the observations within the cluster. There are many clustering algorithms, but the one commonly associated with K-Means Clustering is an iterative refinement technique [26]. Starting with an initial randomly generated set of K means, each observation is associated to its nearest mean, forming K clusters [25]. The observations of each cluster are then averaged to find a new cluster mean [25]. The observations are then reassigned to their nearest mean. This process is repeated until convergence is reached. Convergence is defined as an iteration where no means are moved. Once convergence is reached, each cluster is assigned its representative class based on a majority vote of all the observations associated with that cluster [25]. When a new observation is looking for a classification, it is measured against all the means to determine which cluster it belongs to and is assigned to the same class as the cluster.

In this work, K-means clustering is implemented using the function ‘kmeans’ provided in the Statistics Toolbox in Matlab. This function outputs the cluster center coordinates for each mean and the assigned mean for each data point in the training set. With this data, the associated class for each mean can be calculated, a ‘1’ or a ‘0’. The test data is then classified by comparing the distance of each point to the means to find its assigned class.

Simple Example:

Given the same training set as above, and a K=3, K-Means Clustering starts by randomly placing three means, represented by an X in table 2.4. Each observation is then assigned to the mean it is closest to, in this example, using Euclidian distance. The average of all points associated with each each mean is then calculated. Observations 1, 2 and 8 are associated to mean 1. So points

Initial Means		
mean#	**X**	**Y**
1	3	3
2	7	9
3	2	8

Table 2.3: Location of initial cluster centers

(4,3), (1,4) and (1,1) are averaged to find a new mean at (2,2.67). Observations 3, 5, and 7 are associated with mean 2, so the new mean 2 is (8.33,6). Observations 4 and 6 are associated with mean 3 resulting in a new mean 3 at (1,8). Now the process repeats itself with the new means. All observations are

Training Observations			
Observation#	**X**	**Y**	**Class**
1	4	3	0
2	1	4	0
3	6	6	0
4	2	10	1
5	9	8	1
6	0	6	1
7	10	4	0
8	1	1	1

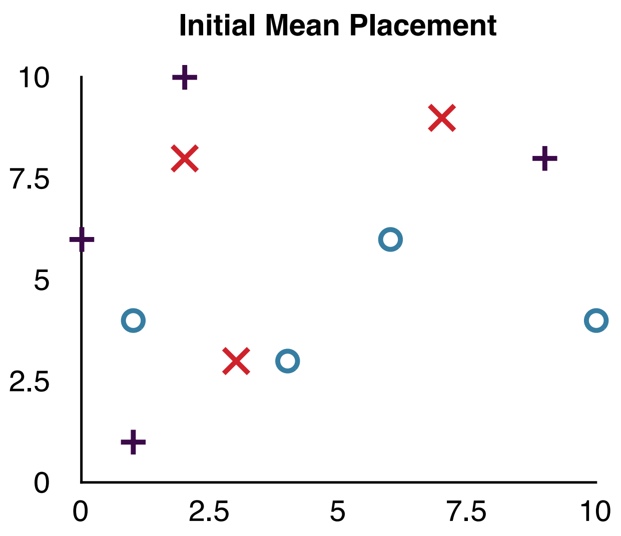

Table 2.4: Example training observations graphed in XY plane along with location of 3 cluster centers

reassigned to the new means. In this example, none of the observations are assigned to a different mean, meaning convergence has been reached. Now a majority vote is taken to assign a class to each cluster. Cluster 1 has two 0's and

1st iteration Means		
mean#	X	Y
1	2	2.67
2	8.33	6
3	1	8

Table 2.5: The location of the Means after 1 iteration through the training set

one 1 associated with it, so it is assigned class 0. Cluster 2 also has two 0's and one 1, so it also assigned to class 0. Cluster 3 has two 1's associated with it, so it is assigned to class 1. The K-means classifier is now trained, and can be used to classify new observations. Using the same test observations as

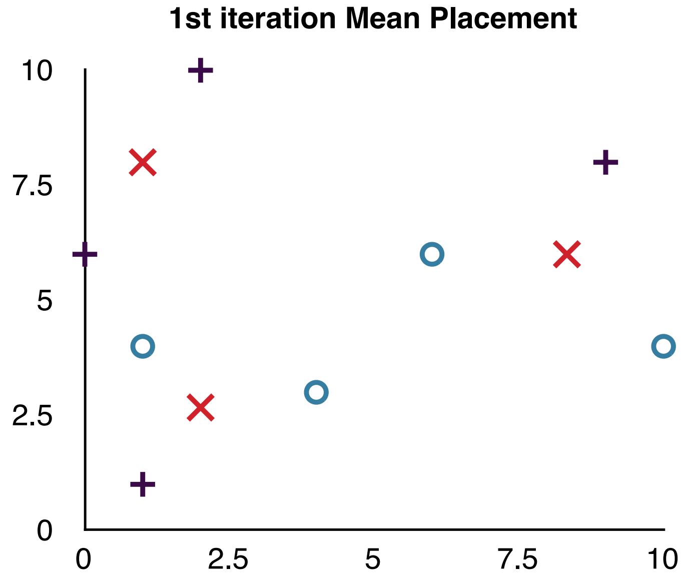

Figure 2.1: The location of the means after one iteration through the training data

1st iteration Observation assignment	
Observation#	**Mean Assigned**
1	1
2	1
3	2
4	3
5	2
6	3
7	2
8	1

Table 2.6: The assignment of each observation to a mean.

before, points (4,4) and (2,8) are assigned to the closest mean. Point (4,4) is found to be nearest mean 1, while point (2,8) is nearest to mean 3. Mean 1

Test Observation s	
X	**Y**
4	4
2	8

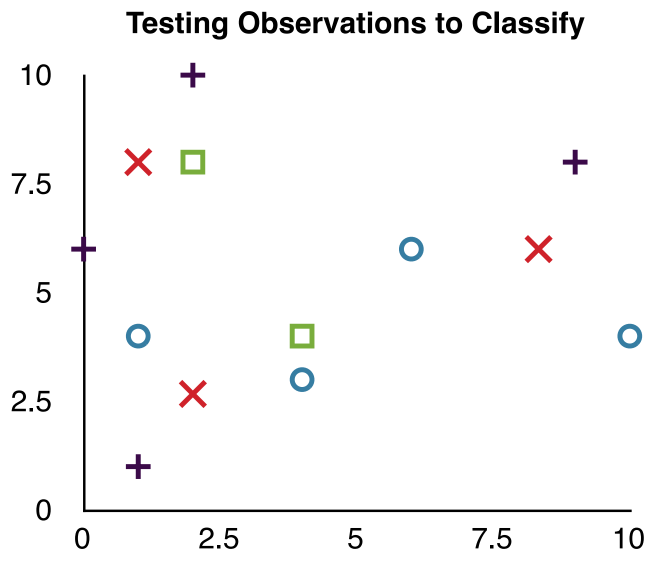

Table 2.7: Example test observations and final cluster centers graphed in XY plane

is associated with class 0, so test observation (4,4) is assigned to class 0. Observation (2,8) is assigned to class 1.

2.3.3 Support Vector Machine (SVM)

Support Vector Machines are supervised learning models that can be used for classification machine learning problems. The basic SVM takes a set of input observations and associated binary outputs and constructs a model that can classify new observations into one class or the other. The model consists of a mapping of the training observations as points in space, linearly separating the observation sets [27]. The margin around this linear separation is calculated to be as large as possible. A partitioning in higher dimensional space by a linear hyperplane corresponds to a nonlinear partition in the output space [28]. This higher dimensional partitioning is known as the SVM kernel, and can be defined by any mathematical surface [28]. Some of the more common kernels are linear, quadratic, polynomial and Gaussian radial basis function [27].

This work uses the ‘svmtrain’ and ‘svmclassify’ functions provided in the Bioinformatics Toolbox by Matlab to perform SVM classification. A polynomial kernel function is used to map the data. Because the training data set is large, the ‘MaxIter’ option is set to its maximum to allow for the training and classification to take place.The Matlab implementation is available in Appendix A.

Simple Example

Given a set of training observations, a support vector machine looks to linearly separate the data using a hyperplane. In this case, because we are only dealing in two dimensions, a simple line can be drawn to separate the data. In the figure 2.2, a possible solution has been shown. The line selected linearly separates the data. The blue area around the line shows its margin, defined as the distance between the line and the nearest observation. A support vector

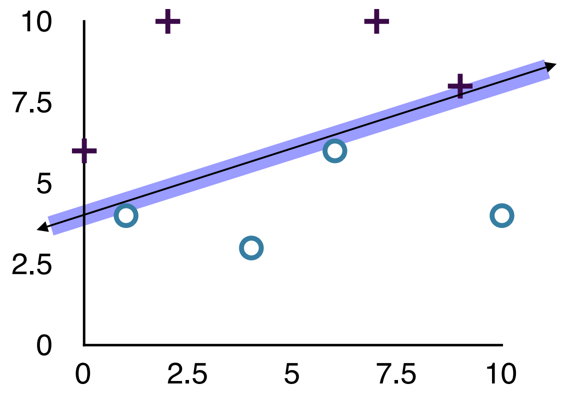

Figure 2.2: A linear line separating the data types

machine looks to place this line as far away as possible from the nearest observations, thus maximizing its margin. To do this a continuous constrained optimization problem needs to be solved. Each observation becomes a constraint that needs to be considered when looking for the line solution. In the

Training Observations			
Observation#	X	Y	Class
1	4	3	0
2	1	4	0
3	6	6	0
4	2	10	1
5	9	8	1
6	0	6	1
7	10	4	0
8	7	10	1

Table 2.8: Example training observations

end, the correct solution is found solving for a maximized margin in the continuously constrained optimization problem.

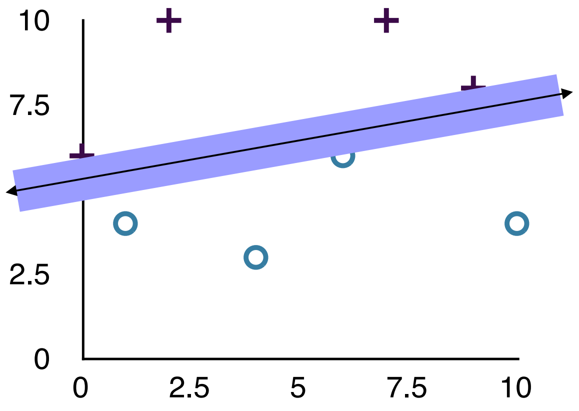

Figure 2.3: A linear separation with the margin maximized

If there is no hyperplane that can separate the example, then the soft margin method can be used [9]. The soft margin method allows for points to appear on the incorrect side of the margin. These points have a penalty associated with them that increases the farther from the margin they appear. The hyperplane separation looks to minimize the penalty of incorrectly labeled points, while maximizing the distance between the remaining examples and the margin.

1 Dimensional Data	
X	**Class**
1	1
2	1
4	1
6	0
7	0
9	0
11	1
15	1

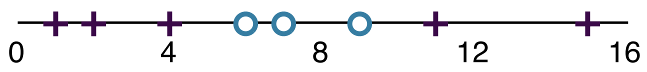

Table 2.9: Example observations graphed on the X plane

A second technique can be employed to separate data that isn't linearly separable. The process is to map the data into a higher dimensional space using an SVM kernel. Using the set of 1 dimension data shown in table 2.9, separation of the data linearly is impossible. This changes if the data is mapped into two dimensional space using the mapping $x = (x, x^2)$. When this two dimensional mapping is graphed, an obvious linearly separable line appears. The mapping

2 Dimensional Data		
X	Y	Class
1	1	1
2	4	1
4	16	1
6	36	0
7	49	0
9	81	0
11	121	1
15	225	1

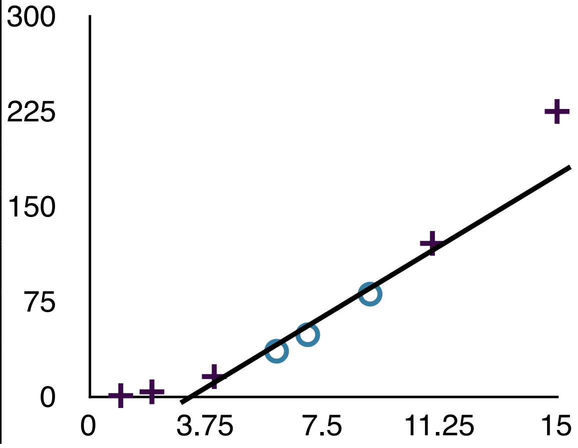

Table 2.10: Example observations mapped to a higher dimensional space. Then graphed in the XY plane to find a linear separation

used to increase the dimensionality of the problem is dependent on the data space being investigated.

2.4 A Literature Overview

This study performs a survey of existing techniques for stock market classification using technical analysis and machine learning classifiers. The following is an overview of the papers that served as guides to this work.

2.4.1 KNN trading system

In the paper “A Method of Automatic Stock Trading Combining Technical Analysis and Nearest Neighbor Classification,” the authors lay out a very straight forward implementation of market data classification [9]. Using four different stock market technicals; moving averages, Relative Strength Index, Stochastic and Bollinger Bands, 22 input features are constructed. These 22 inputs are then used to classify daily stock market data. The data is broken up into training and test data sets. A nearest neighbor classifier is used. Using 10 years of data, 15 different stocks from the Sao Paulo stock exchange are run through a simulated trading system. This system uses the KNN classifiers prediction of the next days stock price (buy, sell, keep) to execute a trade of 100 shares. The profit from these trades is compared to the buy and hold profit over the same period. The results show that this strategy outperformed the buy-and-hold strategy for 12 of the 15 test stocks.

2.4.2 SVM trading system

"Financial Time Series Forecasting Using Support Vector Machines" by Kyoung-jae Kim uses a similar approach to the KNN trading system method, replacing KNN with a support vector machine for classification [27]. Using the Korea composite stock price index, 12 technical indicators are generated to be used as input variables. Support vector machines (SVM) have parameters that can be tuned to optimize performance. Using a Gaussian radial basis function for the SVM's kernel, Kim explores which parameters perform best for his stock data [27]. A comparison between the SVM classifier, a back propagation Neural Network (BPN) and a KNN is performed. The results show that SVM's are sensitive to the value of its parameters and that SVM was able to outperform the

BPN and KNN classifiers in experimental tests when the correct parameters were selected [27].

2.4.3 Evolving SVM's

In "Evolving Least Squares Support Vector Machines for Stock Market Trend Mining", Lai, Wang and Chen use a similar approach to Kim [27] [28]. They use a SVM to classify data using indicators as inputs. These indicators are a combination of technicals and macroeconomic data, such as the interest rate on a one-month T-bill or the Consumer Price Index (CPI). The number of features to be used as inputs to the SVM is determined by a genetic algorithm (GA) [28]. The GA is setup to select a subset of possible inputs and uses them to train a SVM. The trained SVM is used to generate predicted stock movements for a test set. The classification rate is used to evaluate the success of the test. The fitness function for the GA is a combination of the classification success rate and the complexity of the inputs. Ultimately, the more inputs, the more complex the system. This technique was able to drive down the number of features used from 26 to 6 for certain data and improve classification performance. A second GA was also used to select parameters for the SVM [28].

2.4.4 Using Economic and Technical Indicators

"A Comparison of PNN and SVM for Stock Market Trend Prediction Using Economic and Technical Information" by Lahmiri, looks to use macroeconomic data features along with technical indicators as inputs to a probabilistic neural network and an SVM classifier [37]. Granger causality is used to identify the causal relationships between inputs and output in an effort to reduce the feature space [37]. The paper finds that the SVM performs best when only macroeconomic data is used, while the PNN performs best when only technical indicators are used [37]. The combination of both technical and macroeconomic data reduces the classification rate of the classifiers.

The paper also runs an experiment where it alters the trading strategy. Instead of training the classifier to identify days where the Standards and Poor's index (S&P) moves up, it alters the training data to detect days where the S&P moves up by at least 0.5%. This alternative creates a significant increase in classification rate [37].

Chapter 3: Goals Hypothesis and Evaluation Method

3.1 Goals

The main goal of this work was to create an automated trading method that could outperform the buy-and-hold strategy and lower risk. This strategy should have been effective for any market security, given a reasonable amount of historical training data.

The second goal of this work was to learn about computational finance techniques. As the world becomes more connected due to globalization and the Internet, the ability to track and measure these connections increases. In today's society more data is being recorded then ever, yet the information being gleaned from these records is in its infancy. Learning the techniques that can be used to parse and classify big data sets was the ultimate objective, due to the importance it will play in the future of the world.

3.2 Hypothesis

The hypothesis for this work was the following:

Historical stock market data contains information about what direction the market will move in the future. This information can be encoded into technical indicators and be classified into 'up' or 'down' days using known classification techniques such as KNN's or SVM's. These predictions can then be used to make market purchases that will be more profitable than a buy-and-hold strategy.

3.3 Evaluation method

To test this hypothesis, a simulated trading system will be constructed within Matlab. This system will allow a user to specify what technical indicators, classifier, frequency of trades, frequency of retraining, along with other options are used. The system will then simulate trading over a given time period and evaluate the performance using four performance criteria; classification percentage, false positive percentage, profit versus buy and hold and a modified Sharpe ratio.

3.3.1 Classification Percentage

Classification percentage is an evaluation of the results in terms of the percentage of correctly classified instances compared to the total number of instances. In this work, classification percentage is measured on testing data by totaling the number of correctly predicted price movements and dividing it by the total number of days in the set.

3.3.2 False Positive Percentage

Money can only be lost in the stock market when stocks are bought. Minimizing the number of times a classifier falsely predicts a ‘up’ day, minimizes the number of losing trades. By tracking the percentage of losing trades to total trades is another criteria in which to evaluate a strategy’s success. This is done by totaling the number of stock purchases that lost money and dividing it by the total number of stock purchases.

3.3.3 Profit Versus Buy and Hold

With financial data being used, another possible performance criteria is profit made due to the specific trading strategy. Profit is calculated as a percentage gain for each trade. The product of all trades profits gives a total profit percentage for the trading strategy. This can then be compared to the most common investment strategy, buy-and-hold, to see how it performs against a set baseline. Buy-and-hold is calculated the same way, with only one trade being made.

3.3.4 Modified Sharpe Ratio [42]

The Sharpe ratio measures the reward to risk ratio of a trading strategy. It is calculated by subtracting the risk free rate from the average return of the portfolio, divided by the standard deviation of the portfolio returns. The Sharpe ratio looks to show if a portfolio's returns are worth the risk associated with them. By having one number that combines portfolio returns and risk, comparisons between portfolios can be made. The modified Sharpe ratio, used in this work, removes the risk free rate from the equation. The modified Sharpe ratio is calculated by taking the average return, measured in percentage, and dividing it by the standard deviation of the percent daily movements of the trading strategy. A description of the implementation can be found in Appendix B.

Chapter 4: Design

Matlab was selected as the tool to design the trading simulator because of its ease of use. Most features needed for this simulator were already available inside existing toolboxes or readily available off the Internet. The simulator was designed to be flexible, allowing the user to change what stock to simulate, the time period of the stock being tested, how to classify the days, what type of classifier to use, the frequency of trades, the rate of retesting and a selection of multiple data reduction techniques. Using this simulator, a trading model was created to explore how to maximize classification percentage and trading strategy profit.

4.1 Trading models

4.1.1 Data

All data used in this work was pulled from Yahoo! finance's database of historical stock data [31]. Yahoo! provides the open, close, high, low, date, volume and adjusted close for each day of a security. To import this data into Matlab, the "Historical Stock Data downloader" was used [41]. Using the provided method 'hist_stock_data', any stock for any time period available on Yahoo! was imported into Matlab as a structure, broken down into the different categories. The time period investigated in this work was from January 1, 2001 through January 1, 2010. The first 70% of the data was used as training data while the last 30% was saved for testing.

In total, 52 technical indicators and 10 macroeconomic data indicators were constructed. Technical indicator construction was done using the TA-lib Library, an open source set of Matlab functions to calculate many common technicals [32]. For example, making the function call ‘TA_EMA(x,y)’, an exponential moving average can be calculated on x with a look back period of y. The 10 macroeconomic data indicators were pulled from the Federal Reserve Economic Database (FRED) [33]. Because some of these indicators require a long look back period, data as far back as January 1, 2000 was used. The inputs are reported in Table 4.1.

Nine stocks and one index ETF were used to test any given simulated trading strategy. The companies selected; Apple (AAPL), Coke (KO), Dupont (DD), Merck (MRK), McDonalds (MCD), Ford (F), Walmart (WMT), Bank of America (BAC), Disney (DIS) are all high market cap companies, with 8 of the 9 belonging to the Dow Jones Industrial Average. Over the time period selected for testing, 5 of the stocks gained in value, while 4 stocks lost value. The 1 index, the S&P 500 exchange-traded fund (ETF)(SPY), lost value. These companies were selected because they all had a long history and are all large capitalization companies. These factors reduce susceptibility to large swings due to news and provide a large quantity of historical data to use. Each trading simulation run was evaluated on the success over these 10 stocks.

Technical Indicators	
Input	**Definition**
1	$RSI(P_c, 14)$
2	$\frac{P_c - BB_{high}}{BB_{high}}$
3	$\frac{P_c - BB_{low}}{BB_{low}}$
4	$\%K(P, 2)$
5	$\%D(P, 2)$
6	$\%K - \%K(t-1)$
7	$\%D - \%D(t-1)$
8	$\frac{P_c - P_c(t-1)}{P_c(t-1)}$
9	$\frac{P_c - P_L}{P_h - P_L}$
10	$\frac{SMA(P_c, 20) - SMA(P_c(t-1), 20)}{SMA(P_c, 20)}$
11	$\frac{SMA(P_c, 200) - SMA(P_c(t-1), 200)}{SMA(P_c, 200)}$

Technical Indicators	
Input	**Definition**
12	$\dfrac{SMA(P_c,20)-SMA(P_c(t-1),200)}{SMA(P_c,200)}$
13	$\dfrac{P_c-SMA(P_c,200)}{SMA(P_c,200)}$
14	$\dfrac{P_c-\min(P_c(t-1),.....,P_c(t-5)}{\min(P_c(t-1),.....,P_c(t-5)}$
15	$\dfrac{P_c-\max(P_c(t-1),.....,P_c(t-5)}{\max(P_c(t-1),.....,P_c(t-5)}$
16	$\dfrac{V-V(t-1)}{V(t-1)}$
17	$\dfrac{SMA(V,20)-SVM(V(t-1),20)}{SVM(V(t-1),20)}$
18	$\dfrac{SMA(V,200)-SVM(V(t-1),200)}{SVM(V(t-1),200)}$
19	$\dfrac{SMA(V,20)-SVM(V(t-1),200)}{SVM(V(t-1),200)}$
20	$\dfrac{V-SVM(V,200)}{SVM(V,200)}$

Technical Indicators	
Input	**Definition**
21	$\frac{V - \min(V, V(t),....,V(t-5)}{\min(V(t),....,V(t-5)}$
22	$\frac{V - \max(V, V(t),....,V(t-5)}{\max(V(t),....,V(t-5)}$
23	$ROC(P_c, 2)$
24	$Momentum(P_c, 2)$
25	$EMA(P_c, 7)$
26	$EMA(P_c, 50)$
27	$EMA(P_c, 200)$
28	$Moving - Variance = SMA(Variance(P_c, 10), 7)$
29	$Moving - Variance = SMA(Variance(P_c, 10), 50)$
30	$Moving - Variance = SMA(Variance(P_c, 10), 200)$
31	$\frac{Variance(P_c, 10)}{Variance(P_c, 10) - Variance(P_c, 10)(t-1)}$
32	$EMA(P_c, 2) - EMA(P_c, 10)$
33	MACD
34	CCI
35	Linear Regression Line

Technical Indicators	
Input	Definition
36	Chaikin Oscillator
37	$\frac{P_c}{EMA(P_c,5)}$
38	$\frac{P_c}{EMA(P_c,10)}$
39	%R
40	%D
41	$P_c(t-1)$
42	$P_c(t-2)$
43	$P_c(t-3)$
44	$P_c(t-4)$
45	$P_c(t-5)$
46	$P_c(t-6)$
47	$P_c(t-7)$
48	$P_c(t-8)$
49	$P_c(t-9)$
50	$P_c(t-10)$

Technical Indicators	
Input	Definition
51	$\frac{(P_h(t) - P_L(t))^2}{(P_h(t-2) - P_L(t-2)) * V(t)}$
52	$P_c(t) - Max(P_c(t-1),...,P_c(t-5))$

Table 4.1: Full set of technical indicators and their formulas

MacroEconomic Data inputs	
Input	Definition
1	Federal Reserve Rate
2	Canada to US exchange rate
3	Japan to US exchange rate
4	Switzerland to US exchange rate
5	US to EURO exchange rate
6	US to Pound exchange rate
7	Three month T-Bill interest rate
8	Six month T-Bill interest rate
9	Trade Weighted U.S. Dollar Index: Broad
10	Trade Weighted U.S. Dollar Index: Major

Table 4.2: MacoEconomic indicators used

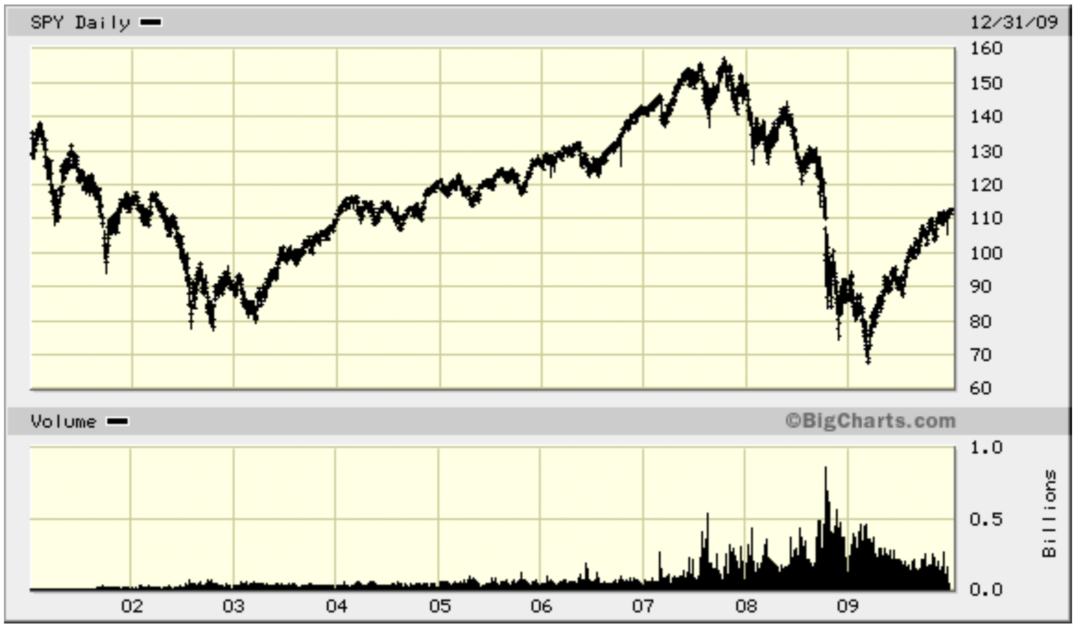

Figure 4.1: Daily movements of SPY over the training and testing period

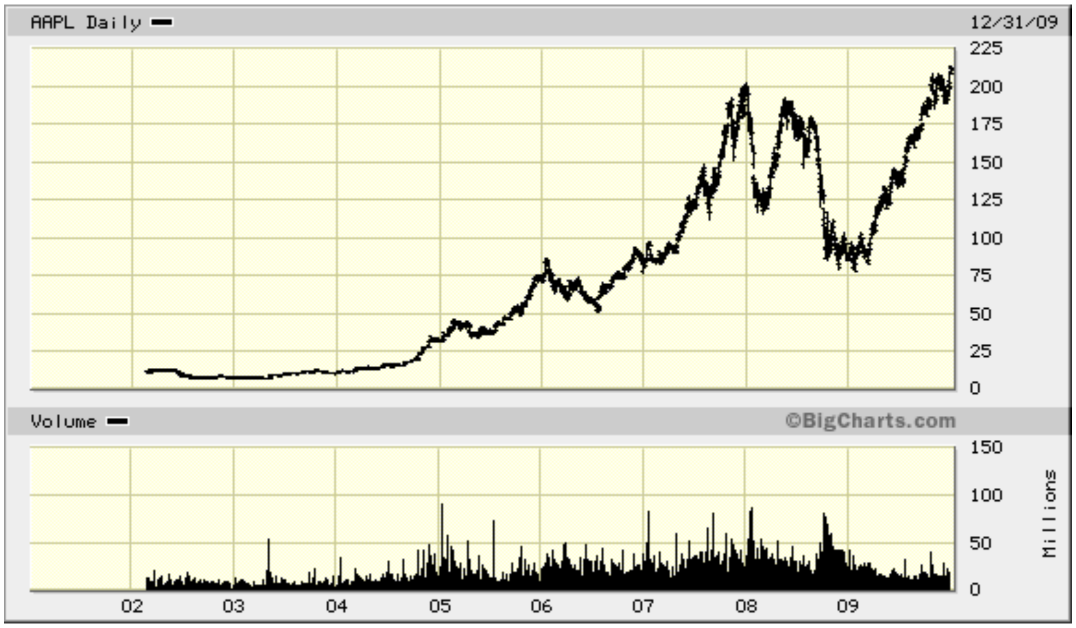

Figure 4.2: Daily movements of AAPL over the training and testing period

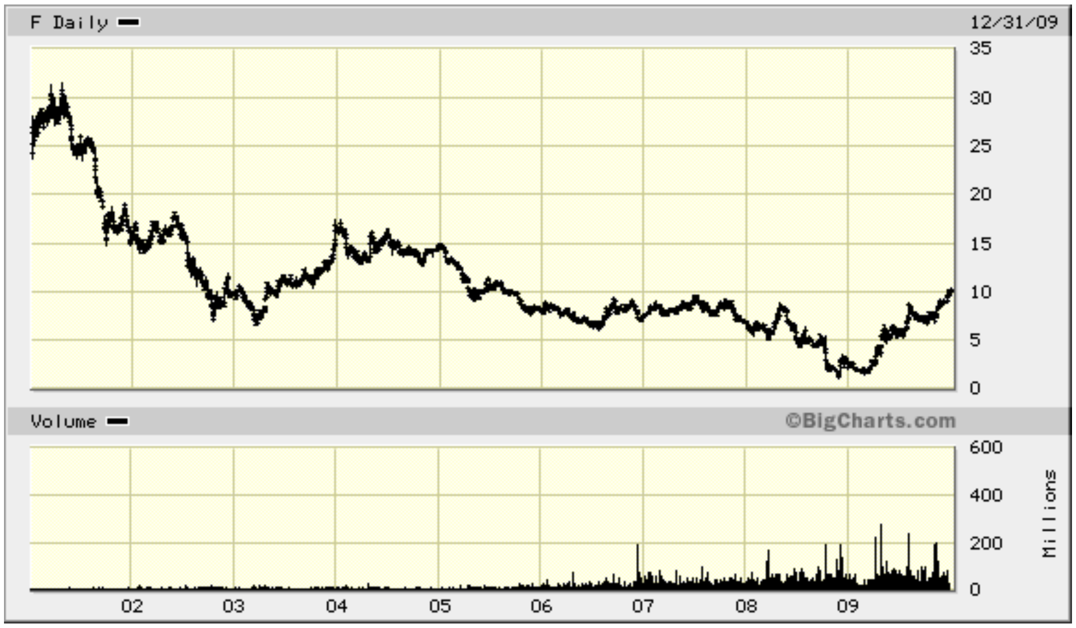

Figure 4.3: Daily movements of F over the training and testing period

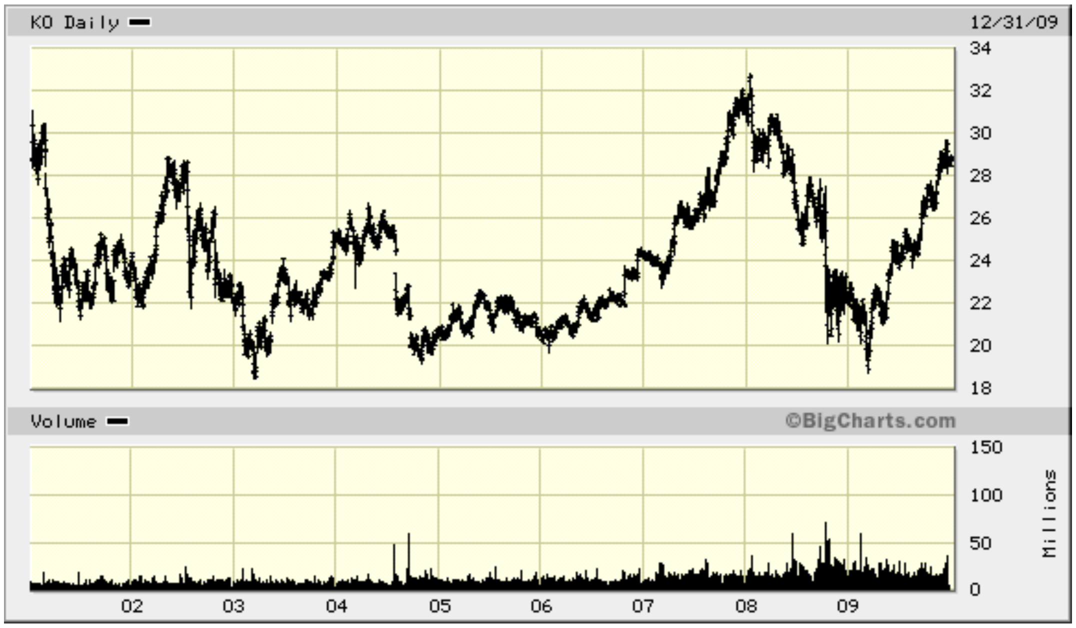

Figure 4.4: Daily movements of KO over the training and testing period

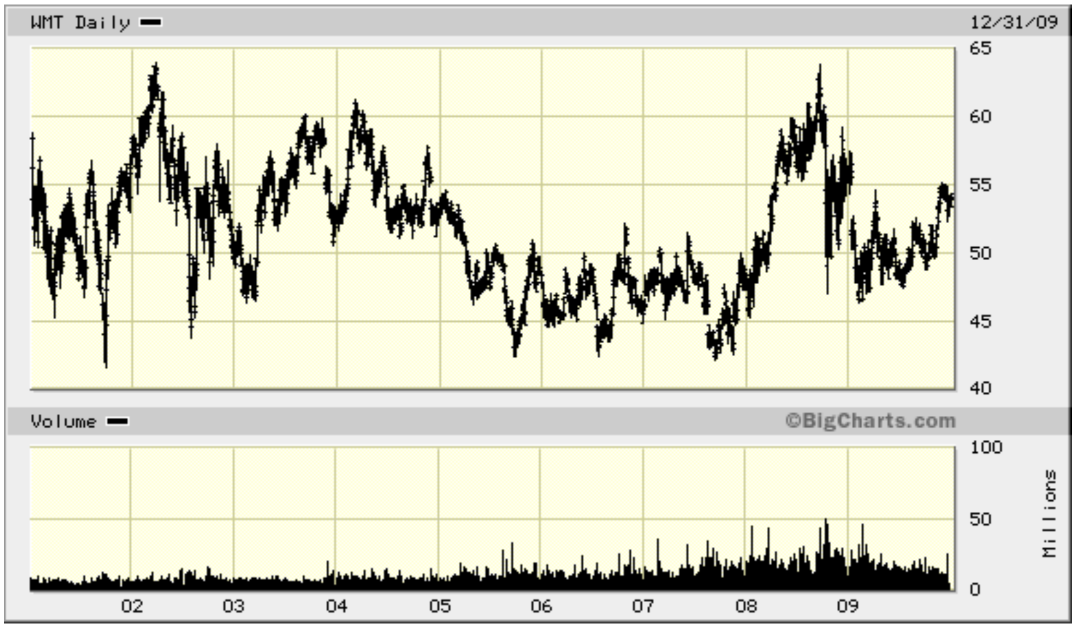

Figure 4.5: Daily movements of WMT over the training and testing period

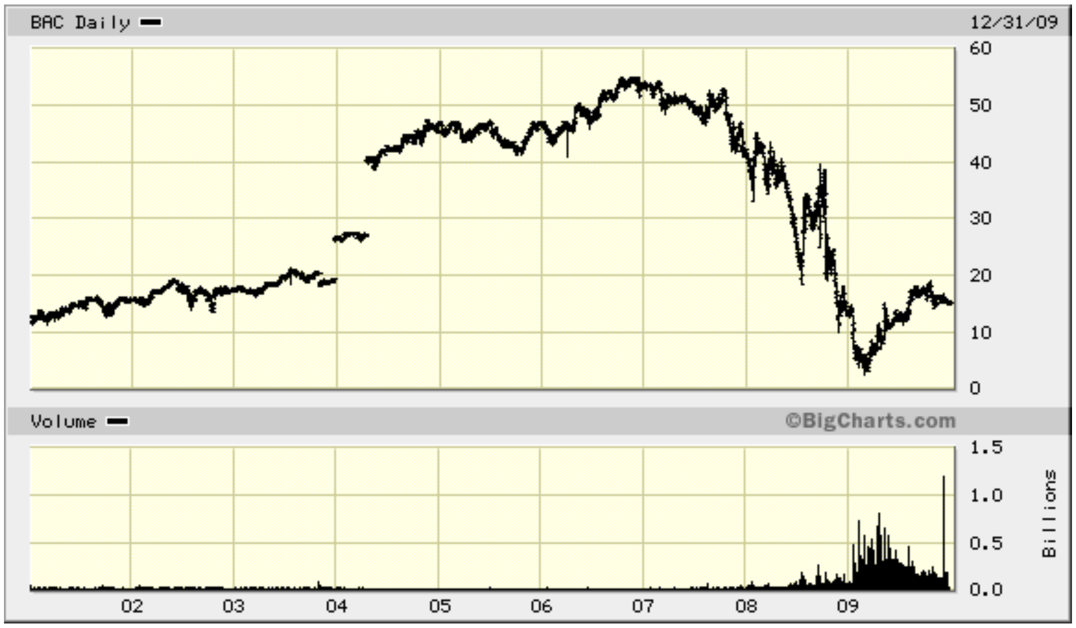

Figure 4.6: Daily movements of BAC over the training and testing period

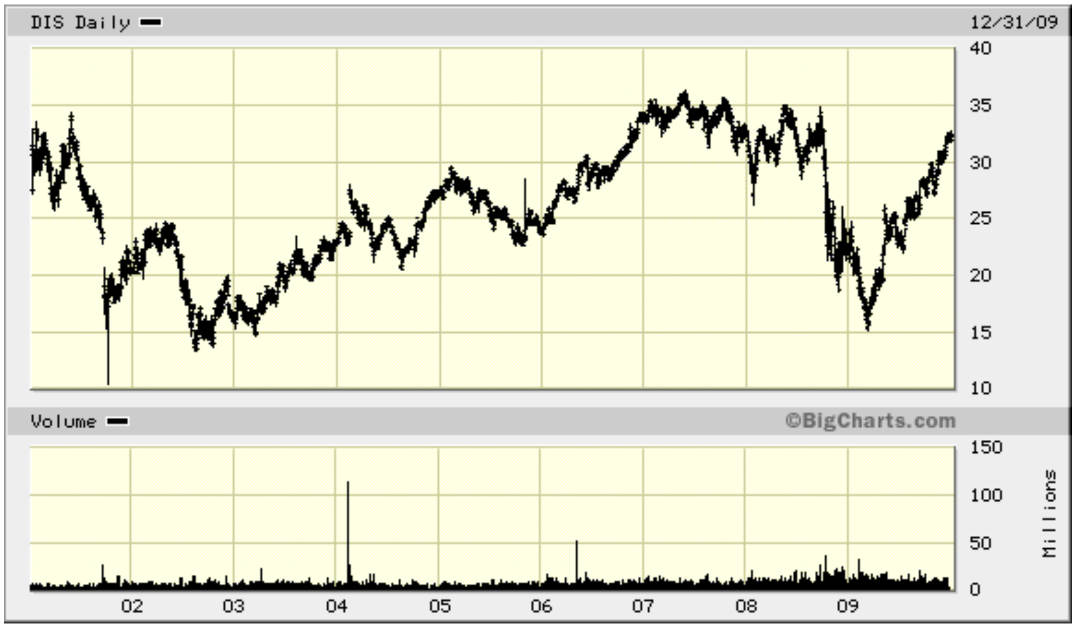

Figure 4.7: Daily movements of DIS over the training and testing period

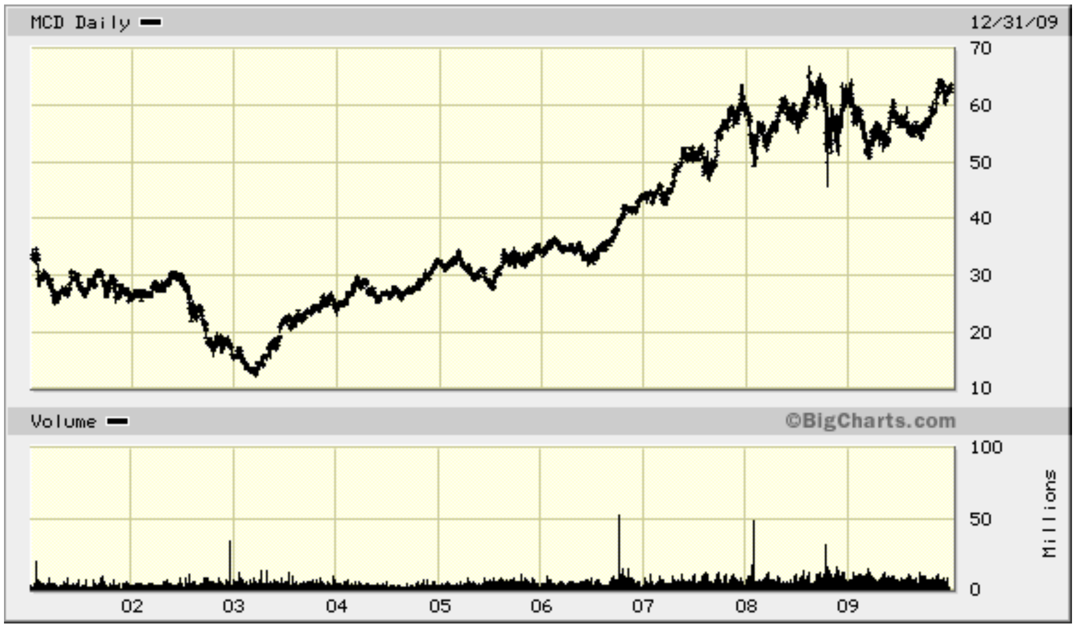

Figure 4.8: Daily movements of MCD over the training and testing period

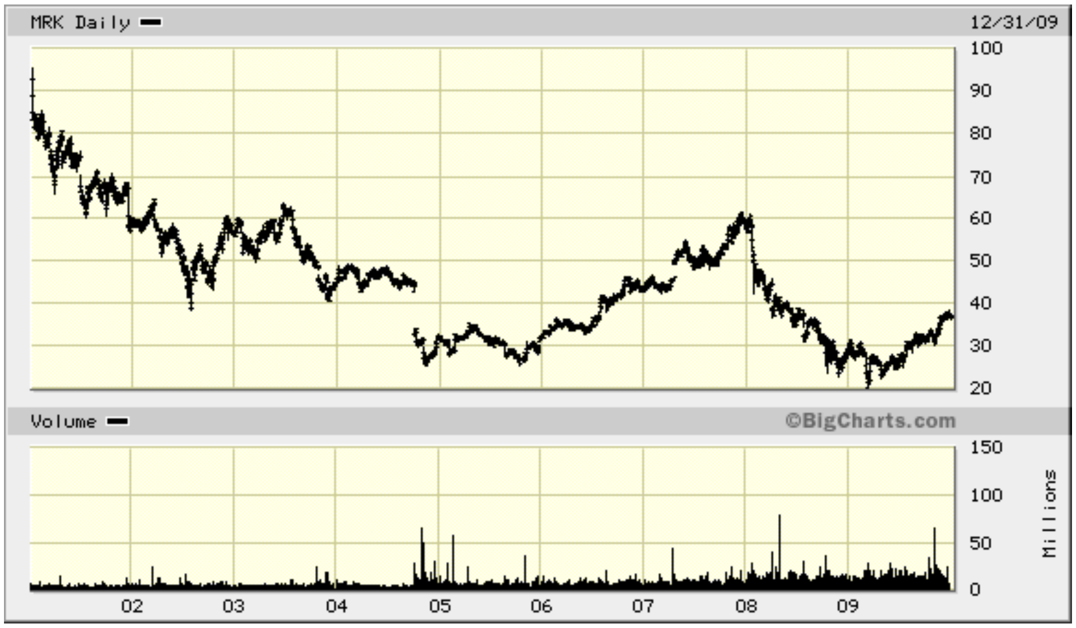

Figure 4.9: Daily movements of MRK over the training and testing period

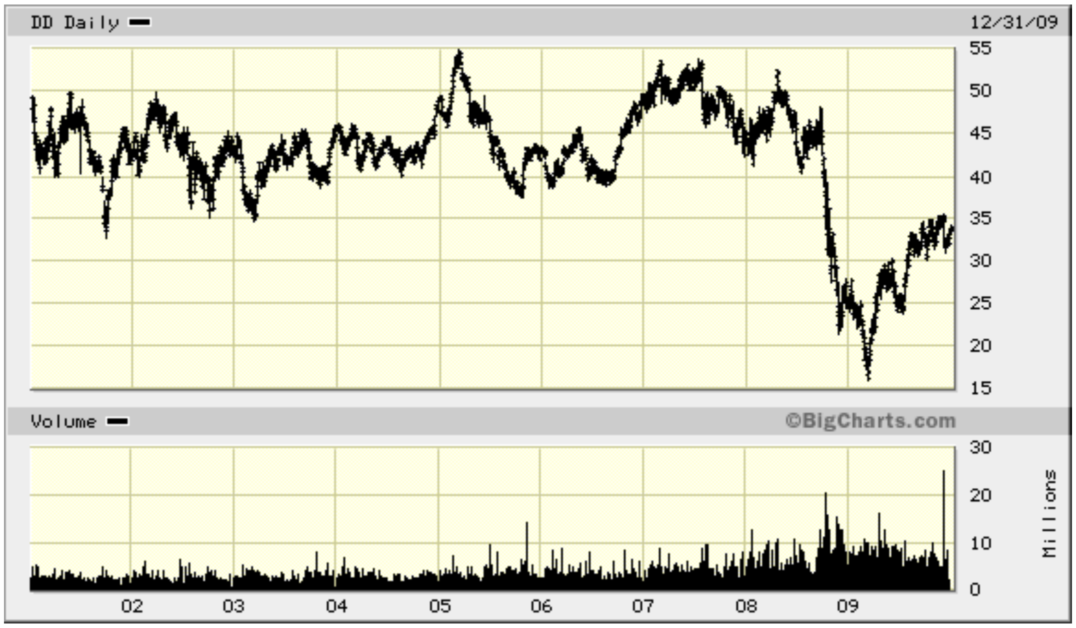

Figure 4.10: Daily movements of DD over the training and testing period

4.1.2 Binary Classification

The price of a specific security on the stock market is simply a dollar value representation of the price to purchase one share. This work deals with time series data that is expressed in these dollar values. Binary classification was selected as a way to represent this data in a more simplified form. This simplification improves the classification process. The binary classification was calculated using two different equations. First, a simple difference was taken; future data point minus current data point. For example, if the time series data is daily stock price, then the classification would state whether tomorrow's closing price is higher or lower than today's closing price. Higher was represented by a ‘1’, lower was represented by a ‘0’. Secondly, a set percent gain is required before classifying a point as ‘1’. For instance, if a 1% gain is required with daily data, only days where the close is up at least 1% from the previous day’s close will be classified as ‘1’, otherwise it is assigned ‘0’.

4.1.3 Trade Frequency

In this work, the trading frequency was daily trades at the closing price of the security. At the end of each day, the classifier outputs a predicted direction for tomorrow’s closing price. If this close price is ‘up’, then a purchase is made at the close price of the current day. If the close price is ‘down’, then no purchase of securities is made. In addition, any positions that are currently open, are closed. All technical indicators used as inputs to the classifier were calculated using the discrete measurement of one day.

4.1.4 Walk Forward Testing

When training and testing the classifiers, a walk-forward testing model was used. Walk forward testing uses the initial training data set to perform a test on a subset of the testing data. Once this subset has been evaluated, it is included in the next iteration of training data. The system is then retrained with the additional training data and tested over the next subset of testing data in the time series. This repeats until all test data has been tested. Walk forward testing's retraining of data can improve accuracy of test results due to the more recent data being included in the training set. Retraining in between each data point theoretically achieves the most realistic simulation of a daily trading model, but is not practical due to the time intensive nature of doing thousands of training cycles. Although a variable system was built which allows for any number of test window sizes, a period of 50 days between retraining was used for most tests.

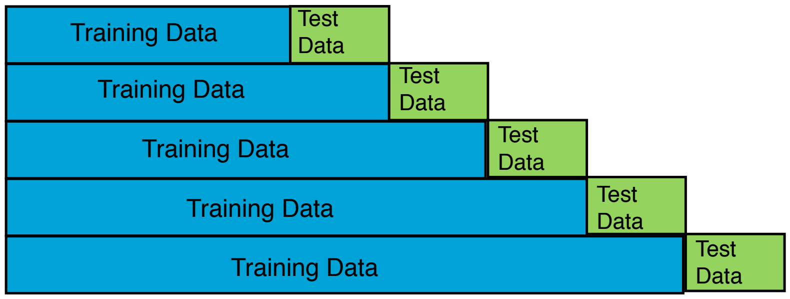

Figure 4.11: Scheme of data set division for training and testing

4.1.5 Data Normalization

Most classifiers use distance measurements to compare examples. To do this effectively, each classifier input needs to have a similar scale. The indicators being constructed in this work have values that range anywhere from below 1 to millions. To scale these inputs, the vector of all examples for each input are normalized to a length of 1 [34]. This is done using the 'normc' function within Matlab, seen in Appendix A.

4.2 Data Reduction

In this work, classifiers with up to 62 inputs are used, 52 technical and 10 macroeconomic. As each feature adds another dimension to the search space, higher dimensional problems get exponentially harder to solve. By reducing the number of redundant features, we can simplify the job for the classifier. Two different ways of reducing the search space were pursued; principal component analysis and genetic algorithm selection.

4.2.1 Principal component analysis

Principal component analysis (PCA) is a useful statistical technique that implements an orthogonal transformation to convert correlated data sets of high dimensionality into a set of linearly uncorrelated data of equal or lower dimensionality, called principal components [36]. This analysis is able to reduce the number of dimensions without much loss of information. The penalty associated with the loss of information can be outweighed by the simplification of a lower dimensional search space [36].

In this work, PCA is performed by using the functions ‘compute_mapping’ and ‘intrinsic_dim’ out of the Matlab Toolbox for Dimensional Reduction [43]. The ‘intrinsic_dim’ function performs an intrinsic dimensionality estimation using Eigen values that returns an estimated number of features that PCA can reduce the feature set to. The ‘compute_mapping’ is used to perform PCA to reduce the dimensionality of the feature set to the estimated dimensionality found using ‘intrinsic_dim’.

4.2.2 Genetic Algorithm selection

Genetic algorithms (GA) are an optimization tool that mimics natural evolution. Starting with an initial population of randomly generated candidate solutions, the solutions are evaluated to determine the best solutions. The evaluation metric used to compare the candidate solutions is called the fitness function. The candidates with the best solutions are cross bred and mutated to produce offspring. The offspring are again evaluated and then the process is repeated until an acceptable solution is found. Through this process, the search space is explored and the optimal solution is found.

This work uses genetic algorithms to select a subset of the initial 52 technical indicators. A binary string of length 52 was created, randomly assigning 1’s or 0’s to each bit within the string. Each of these bits represented an input of the classifier. If the bit was a ‘1’, then that technical was used as an input, if it was a ‘0’, it would not. Using this new subset of inputs, the classifier was trained and tested, with the classification rate being used as the fitness function. The intent is to maximize the classification rate. Because genetic algorithms are designed to minimize fitness, the classification rate is simply negated when fed into the GA. Once a candidate solution has had its fitness calculated, it enters the breeding and mutation step. Breeding was implemented using roulette wheel selection and two point crossover. The offspring of crossover are called children. Mutation occurs next, to all children of the crossover. Mutation is important to GA’s because it allows the search to avoid local minima within the search space. The mutation for these children was done with a simple random bit flip. The length of the binary string is traversed, applying a 2% chance of a bit flip to each bit. If a bit was selected for mutation, then the bit’s value is changed. If it was a ‘1’, it becomes a ‘0‘ and vis versa. Once a mutated child is created, it is tested for fitness and then compared to its unmutated parent. The candidate solution with the better fitness was moved to the next generation. This process was repeated for a set amount of generations. In the end, a candidate solution was evolved that selects a subset of the input feature set and performed an optimized classification.

This work uses the function ‘ga’ from the Global Optimization Toolbox to perform the genetic algorithm optimization. The fitness function ‘stockGA.m’, seen in Appendix A, was used to minimizing the negated classification rate for each instance of the classification. If not otherwise specified, the default Matlab parameters for the ‘ga’ function were used.

Chapter 5: Experiments and Results

5.1 KNN Classification

The first goal was to recreate the work done in “A Method for Automatic Stock Trading Combining Technical Analysis and Nearest Neighbor Classification” [9]. Because this work is from Brazil and uses Brazilian stocks, a different set of US stocks were selected for testing. The same features seen in [9], the first 22 features shown in Table 4.1, were used as inputs to a KNN classifier. Ten stocks were selected to test our trading method. No stop losses or data reduction was used and purchases were made when the classifier predicted the future closing price to be higher than the current closing price. Ten nearest neighbors were used for classification and an Euclidean distance metric was used. Daily trades were performed.

Stock	Classification %	False Positive %	profit %	Buy And Hold %	Sharpe Ratio	BuyAndHold Sharpe Ratio	# of Trades
SPY	52.95	45.3	5.7	-20.8	0.013	-0.0076	119
AAPL	46.8	49.1	32.5	121	0.0301	0.0549	149
F	50.7	50.9	15.6	26.9	0.023	0.0309	136
KO	54.5	39.7	115.4	19.4	0.1178	0.0239	136
WMT	49.3	50.2	8.6	15.5	0.0166	0.021	131
BAC	49.2	53.7	-62.3	-66.8	-0.0136	0.0039	89
DIS	53.3	47.2	48.5	-3.9	0.0417	0.0102	117

Stock	Classification %	False Positive %	profit %	Buy And Hold %	Sharpe Ratio	BuyAndHold Sharpe Ratio	# of Trades
MCD	52.7	44.2	60.1	40.4	0.0683	0.038	134
MRK	51.3	50.8	-0.8	-21.5	0.012	-0.0023	126
DD	51.4	47.1	62.5	-24	0.0465	-0.0024	124
Average	51.2	47.8	28.6	8.6	0.0355	0.0171	126.1

Table 5.1: Results for the KNN Classifier

The results for this test can be seen in Table 5.1. Average classification was 51.2%. This number isn't comparable to [9] because it did not use a binary classification, rather a ternary classification. The definition of the three classification values, 'buy, 'hold' and 'sell' were not stated in the paper, thus it is not being used here. What can be compared is the profit via the KNN trading strategy vs the buy-and-hold strategy. 7 of the 10 stocks outperformed buy-and-hold, with only 3 (AAPL, F and WMT) underperforming. The 3 underperforming stocks still saw gains during the test period, just not as large as those using buy-and-hold. This is comparable to [9], where 12 of 15 stocks outperform buy-and-hold. The number of trades made during the test period was also comparable to [9], seeing around 40 trades per year. This simulation made around 126 trades, on average, for a period of 3 years. The Sharpe ratio shows a reduction of risk for 6 of the 10 stocks using the KNN trading strategy. WMT joined the 3 stocks that underperformed in profit as the 4 stocks that had a greater Sharpe ratio for the buy-and-hold strategy. Looking at figure 5.1, the equity curve for the stock

SPY during the test period, we see the trading strategy closely mirrors buy-and-hold, with occasional areas of outperformance. Around day 150 in the test set, the stock drops around 10%, with the KNN trading strategy avoiding most of that loss. This occurs again at around day 400, where the KNN strategy dropped with the market, but quickly recovers when the buy-and-hold strategy doesn't. Just a few of these loss avoidances seems to make the difference between the 20% loss seen by the buy-and-hold strategy and the 5% gain seen by the KNN trading strategy.

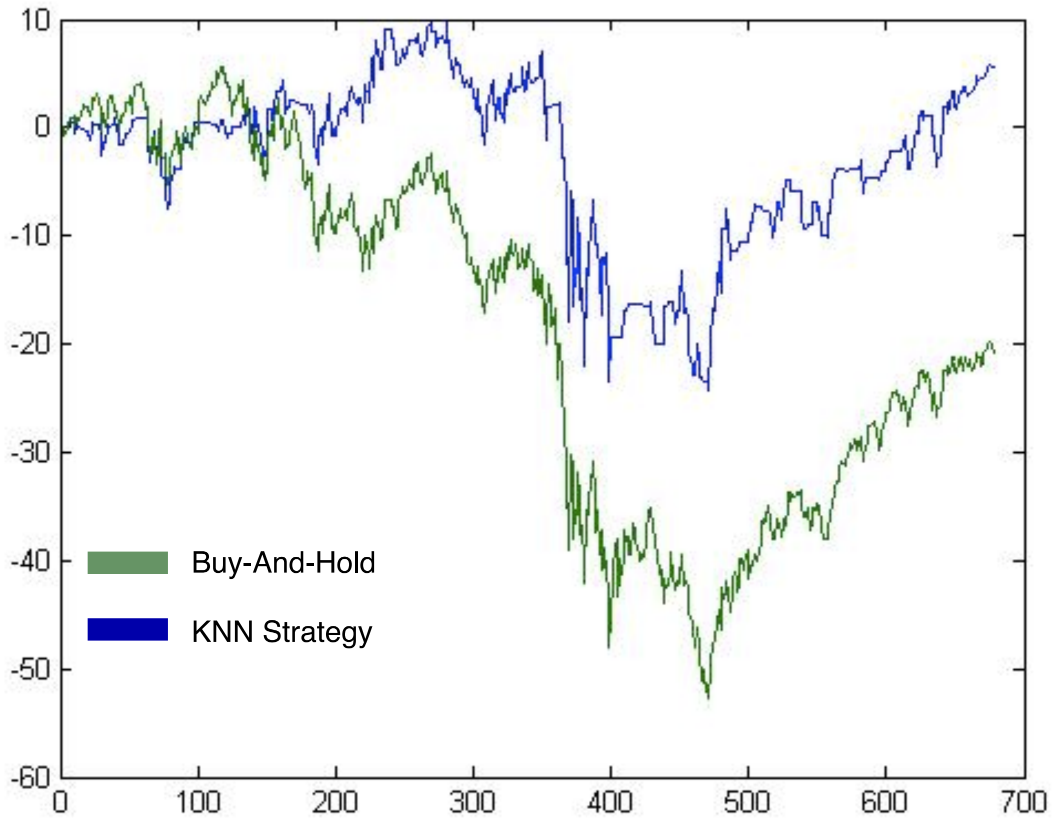

Figure 5.1: Equity curve of trading strategy and buy-and-hold for stock SPY in the out-of-sample period

5.2 Other Classifiers

Following the work, "A Comparison of PNN and SVM for Stock Market Trend Prediction using Economic and Technical Information", the next step in this exploration was to substitute the KNN classifier with other well known classifiers. A SVM classifier was chosen because it is a commonly used classifier for market classification, along with K-means [27] [28] [30] [37]. Tables 5.2 and 5.3 show the results for classification using an SVM and K-means, respectively. These tests used the same setup as the KNN classification. The SVM used a polynomial kernel. This was chosen after experimental evidence showed that the linear and radial basis kernels were unable to separate the data effectively. Both kernels resulted in single digit trades, often just 1 or 2 that would last for long periods of time. Because the optimal trading system performs trades frequently, the linear and radial basis kernels were not used. The polynomial kernel was able to achieve a trading number on par with KNN, so it was used. The K-means classifier uses 10 clusters, also experimentally found to yield best classification percentage.

Stock	**Classification %**	**False Positive %**	**profit %**	**Buy And Hold %**	**Sharpe Ratio**	**BuyAndHold Sharpe Ratio**	**# of Trades**
SPY	55	43.6	51.7	-20.8	0.0582	-0.0076	101
AAPL	48.5	46.9	78.1	121	0.0579	0.0549	109
F	48	52.5	11.6	26.9	0.0243	0.0309	92
KO	53.8	45.1	61.2	19.4	0.0597	0.0239	100

Stock	Classification %	False Positive %	profit %	Buy And Hold %	Sharpe Ratio	BuyAndHold Sharpe Ratio	# of Trades
WMT	51.7	47.3	36.9	15.5	0.0481	0.021	75
BAC	52.1	51.8	5.4	-66.8	0.0221	0.0039	40
DIS	49.6	51.8	-12.6	-3.9	-0.0012	0.0102	90
MCD	50.4	48.1	30.7	40.4	0.0367	0.038	71
MRK	50.5	52.3	5.4	-21.5	0.0127	-0.0023	77
DD	46.8	52	-54.9	-24	-0.0468	-0.0024	76
Average	50.6	49.1	21.4	8.6	0.02717	0.01705	83.1

Table 5.2: Results for the SVM Classifier

The SVM performed worse in every category when compared to the KNN classifier. Classification percentage dropped 0.6% and profit dropped from 28.6% to 21.4%. Five of the 10 stocks were able to outperform buy-and-hold, with F and AAPL being the two stocks that underperformed for both the KNN and SVM strategy. BAC saw the biggest change, moving from a -62.3% profit using the KNN strategy to a 5.4% profit using the SVM strategy. Figure 5.3 shows the equity curve of BAC for the SVM, KNN and buy-and-hold strategies. The SVM spends over 200 days without making a single trade, avoiding a significant down period, while KNN is able to make a profit over this same period. Around day 450 of the test period, the stock drops over 50% over a few week period.

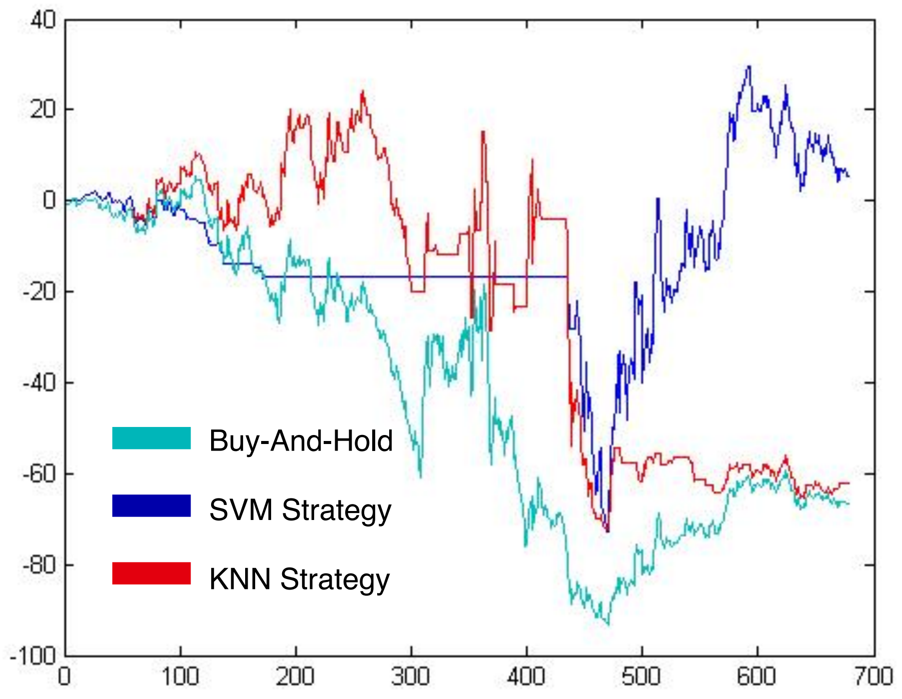

Figure 5.2: Equity curve of SVM and KNN trading strategies and buy-and-hold for stock BAC in the out-of-sample period

This drop isn't avoided by any strategy, but SVM makes a quick rebound, making it back into positive profits while KNN maintains a flat trajectory and buy-and-hold makes a 20% gain from its low.

Stock	Classification %	False Positive %	profit %	Buy And Hold %	Sharpe Ratio	BuyAndHold Sharpe Ratio	# of Trades
SPY	49.9	47.9	-39	-20.8	-0.0482	-0.0076	57
AAPL	50.1	46.9	1.0	121	0.0128	0.0549	67
F	52.7	48.5	302	26.9	0.0992	0.0309	63

Stock	Classification %	False Positive %	profit %	Buy And Hold %	Sharpe Ratio	BuyAndHold Sharpe Ratio	# of Trades
KO	53	42.1	26.5	19.4	0.0531	0.0239	86
WMT	51.8	47	55.4	15.5	0.0664	0.021	88
BAC	49.8	54.1	-15.3	-66.8	0.0127	0.0039	66
DIS	49.5	54	-23.1	-3.9	-0.035	0.0102	66
MCD	49.3	48.5	-4.7	40.4	0.00069	0.038	113
MRK	47.3	54.5	-42.6	-21.5	-0.0391	-0.0023	96
DD	47.6	52	-11.7	-24	-0.0082	-0.0024	66
Average	50.1	49.6	24.9	8.62	0.01144	0.01705	76.8

Table 5.3: Results for the K-Means Classifier

The K-means classifier performance also performed worse than the KNN in all of the performance metrics. Classification rate dropped down to 50.1, while profit dropped to 24.85%. One stock that performed significantly better was F. Using K-means, a 302% profit was achieved, compared to 26.9% for buy-and-hold and a 15.6% using KNN. Figure 5.4 shows the equity curve of F for the K-means and the buy-and-hold strategies. The K-Means classifier is able to do a good job of predicting the 200 day period in which F trends downward, and then is able to capitalize greatly on the period of growth following.

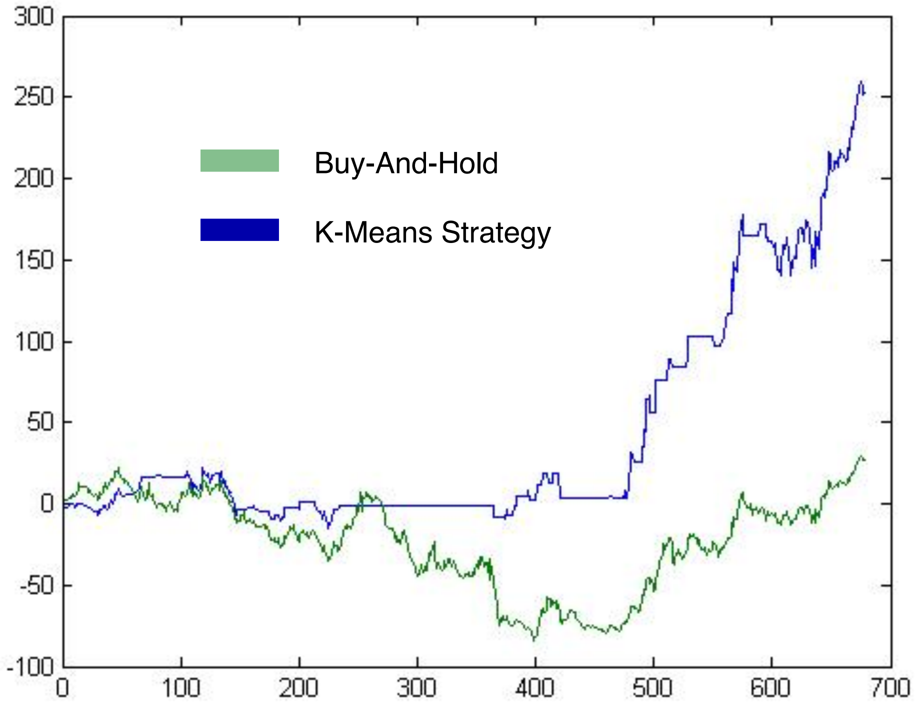

Figure 5.3: Equity curve of K-Means trading strategy and buy-and-hold for stock F using an SVM classifier in the out-of-sample period

5.3 Redefining the Classification Rule

Binary classification was defined as a day's closing price being at least one cent greater than the previous day's closing price for that day to be classified as an 'up' day. With this definition, the distinction between an 'up' and 'down' day can be very small for a great many data points. The next experiment explored changing the binary classification rule to require a certain threshold of percentage price movement to occur to classify a particular day as an 'up' day [37]. The goal of this is to help the classifier correctly classify big 'up' days with more precision. The results are shown in tables 5.4, 5.5 and 5.6 for differing threshold values.

Stock	Classification %	False Positive %	profit %	Buy And Hold %	Sharpe Ratio	BuyAndHold Sharpe Ratio	# of Trades
SPY	49	46.8	35	-20.8	0.0404	-0.0076	88
AAPL	47.9	47.4	70	121	0.0508	0.0549	125
F	50.1	53.1	7.7	26.9	0.0167	0.0309	112
KO	49.5	35.7	26	19.4	0.0622	0.0239	42
WMT	48.34	55	6	15.5	0.0175	0.021	68
BAC	50.8	53.7	-49.4	-66.8	-0.0046	0.0039	108
DIS	52.6	46.9	75.1	-3.9	0.06	0.0102	68
MCD	48.6	47.9	16.7	40.4	0.0349	0.038	75
MRK	54.2	45.8	54.6	-21.5	0.05	-0.0023	89
DD	52.75	41	84.2	-24	0.064	-0.0024	83
Average	50.4	47.3	32.6	8.6	0.0392	0.0171	85.8

Table 5.4: Results for the KNN Classifier with a 0.5% threshold values

With a threshold values of 0.5%, we see average classification go down slightly, but profits advance from 28.6% to 32.6%. This can be explained through the reduction of the false positive percentage. The number of stock purchases has decreased for the test period but the rate in which these trades results in a positive gain has increased slightly. The total number of trades also decreased by an average of 40 trades.

Stock	Classification %	False Positive %	profit %	Buy And Hold %	Sharpe Ratio	BuyAndHold Sharpe Ratio	# of Trades
SPY	49.2	43.7	56.6	-20.8	0.0632	-0.0076	65
AAPL	47.7	43.6	7.5	121	0.0149	0.0549	61
F	50.1	55.7	25.2	26.9	0.0266	0.0309	64
KO	48.9	31.3	22.3	19.4	0.0666	0.0239	22
WMT	50.1	46.9	9.8	15.5	0.0464	0.021	24
BAC	52.1	53.1	-14.2	-66.8	0.0126	0.0039	99
DIS	52.7	44.8	28.1	-3.9	0.0337	0.0102	57
MCD	48.3	46	-2.3	40.4	-0.0106	0.038	29
MRK	52.6	48.6	-0.9	-21.5	0.0037	-0.0023	47
DD	51.3	40	75.5	-24	0.074	-0.0024	49
Average	50.3	45.4	20.8	8.6	0.0331	0.0171	51.7

Table 5.5: Results for the KNN Classifier with a 1.0% threshold values

With a threshold value of 1.0%, we saw a decrease of profits, down to 20.8%. The classification still hovered around 50%, but the false positive percentage continued to drop, down to 45.4%. This is expected as changing the classification rule to look for more obvious positive stock movements would likely reduce the number of total trades made, but increase the number of successful trades. 4 stocks saw their profit gain compared to a 0.5% classification rule; SPY , F, WMT and BAC. When volatility is compared for the stocks that saw greater performance at a higher classification rule to those with worse performance, we see the higher the volatility, the more success at a higher classification rule. As figures 5.5 and 5.6 show, F volatility ranged from 0.02 to

0.12 during the test period and saw great improvement when the classification rule 1.0% was used. MRKs volatility ranged from 0.01 to 0.06, a volatility half of F's, and saw a decrease in profit using the increased classification rule.

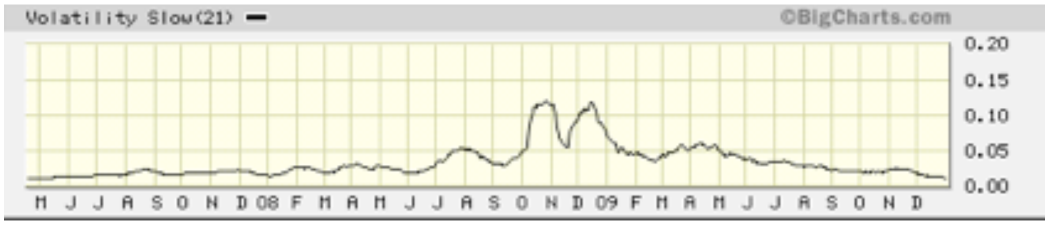

Figure 5.4: Volatility for stock F during test period

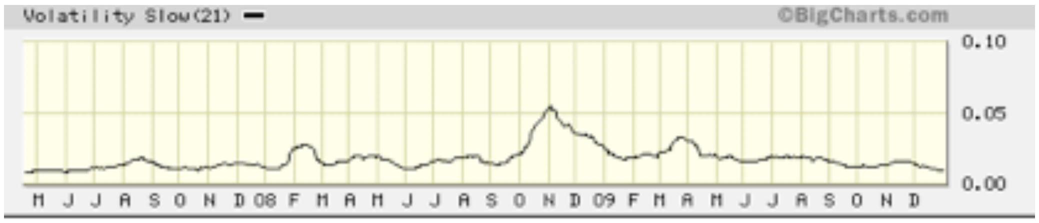

Figure 5.5: Volatility for stock MRK during test period

Stock	Classification %	False Positive %	profit %	Buy And Hold %	Sharpe Ratio	BuyAndHold Sharpe Ratio	# of Trades
SPY	47.4	49.3	18.9	-20.8	0.0296	-0.0076	44
AAPL	46.4	47.4	4	121	0.0217	0.0549	15
F	50.2	57.8	30.6	26.9	0.0377	0.0309	44
KO	48	25	3.9	19.4	0.0394	0.0239	10
WMT	50.4	37.5	6.9	15.5	0.0535	0.021	13
BAC	53.3	51.9	17.4	-66.8	0.0226	0.0039	81
DIS	52.4	44.3	27.9	-3.9	0.0388	0.0102	41
MCD	48	44.4	1.7	40.4	0.0183	0.038	8

Stock	Classification %	False Positive %	profit %	Buy And Hold %	Sharpe Ratio	BuyAndHold Sharpe Ratio	# of Trades
MRK	52.1	60	1.4	-21.5	0.0127	-0.0023	5
DD	50.1	40	44.3	-24	0.0644	-0.0024	30
Average	49.83	45.76	15.7	8.62	0.03387	0.01705	29.1

Table 5.6: Results for the KNN Classifier with a 1.5% threshold values

A threshold values of 1.5% caused a significant decrease in the number of trades, averaging only 29 trades during the 679 day test period. MRK and MCD only saw 5 and 8 trades respectively, making most data gathered for this period very susceptible to large swings due to 1 or 2 bad trades. BAC was an outlier, having made 81 trades during the period, almost twice as many trades as the next closest stock. This can be attributed to it having the highest sustained volatilities of any stock during the test period. Due to this, it also saw the biggest increase in profits for all stocks.

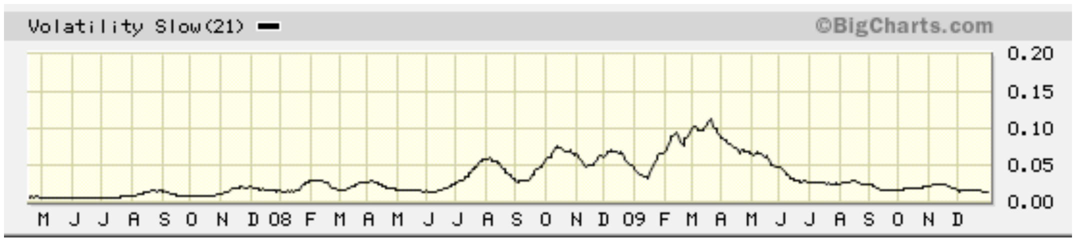

Figure 5.6: Volatility for stock BAC during test period

Overall, redefining the classification rule was an effective technique to decrease false positive percentage and boost profits for stocks experiencing high volatility. Actively selecting what threshold value to use per stock based on volatility could show increased performance. The modified Sharpe ratio remained high for all threshold values tried, even as average profits fell. This showed that the thresholding decreased risk in the trading strategy.

5.4 Adding Additional Technicals

The next step was to increase the information being used by the classifiers. This was done by increasing the number of technicals being used as inputs. From papers, “Evolving Least Squares Support Vector Machines for Stock Market Trend Mining” and “A Comparison of PNN and SVM for Stock Market Trend Prediction using Economic and Technical Information” an additional 30 technical features and 10 macro economic data features are found to help classify the data [28] [37]. With these 52 features being used as inputs for a KNN classifier, the results in Table 5.7 are gathered.

Stock	**Classification %**	**False Positive %**	**profit %**	**Buy And Hold %**	**Sharpe Ratio**	**BuyAndHold Sharpe Ratio**	**# of Trades**
SPY	51.3	46.2	-5.3	-20.8	0.0023	-0.0076	156
AAPL	52.4	42.5	283.2	121	0.1118	0.0549	143
F	51.6	50.4	31.1	26.9	0.0285	0.0309	151
KO	50.2	46.3	39.4	19.4	0.0578	0.0239	165
WMT	49.5	50	-20.8	15.5	-0.0269	0.021	161
BAC	51.8	51.7	15.8	-66.8	0.0251	0.0039	131
DIS	52.9	48.1	64	-3.9	0.0512	0.0102	159
MCD	51.7	45.8	19.2	40.4	0.028	0.038	150
MRK	49.8	52.3	-35.1	-21.5	-0.0245	-0.0023	157
DD	51.1	47	85.5	-24	0.0590	-0.0024	151

Stock	Classification %	False Positive %	profit %	Buy And Hold %	Sharpe Ratio	BuyAndHold Sharpe Ratio	# of Trades
Average	51.2	48.0	47.7	8.6	0.03123	0.0170	152.4

Table 5.7: Results for the KNN Classifier Using 52 Technical Indicators as Inputs

An increase in the number of technicals yielded no change in average classification percentage, but increased profits to 47.7% from 28.6%. The average Sharpe ratio saw a decrease to 0.03123. Before this, the Sharpe ratio had been tightly coupled with profit. When profit increased, so did the Sharpe ratio. This change states that while profits increased with 52 technicals, the risk associated with this strategy also increased.

In "Evolving Least Squares Support Vector Machines for Stock Market Trend Mining" [28], technical indicators, along with macro economic indicators taken from the Federal Reserve website, were used as inputs. Adding the same 10 macroeconomic indicators to our pool of features yield the results in table 5.8.

Stock	Classification %	False Positive %	profit %	Buy And Hold %	Sharpe Ratio	BuyAndHold Sharpe Ratio	# of Trades
SPY	51.7	45.7	20.2	-20.8	0.0256	-0.0076	157
AAPL	49.5	45.9	128.6	121	0.0711	0.0549	157
F	53.6	47.9	130.8	26.9	0.0613	0.0309	149
KO	54.1	41.6	52.8	19.4	0.0725	0.0239	154
WMT	50.2	49.3	-7.4	15.5	-0.0052	0.021	160
BAC	52	51.6	58	-66.8	0.0369	0.0039	142

Stock	Classification %	False Positive %	profit %	Buy And Hold %	Sharpe Ratio	BuyAndHold Sharpe Ratio	# of Trades
DIS	52	49.3	41.7	-3.9	0.0379	0.0102	154
MCD	50.5	47.2	16.2	40.4	0.0248	0.038	143
MRK	49.8	52.3	-29.8	-21.5	-0.019	-0.0023	159
DD	49.8	48.2	42.3	-24	0.038	-0.0024	151
Average	51.3	47.9	45.3	8.62	0.03439	0.01705	152.6

Table 5.8: Results for the KNN Classifier Using 52 Technical Indicators and 10 Macroeconomic Indicators

The addition of 10 macroeconomic indicators appeared to have little effect on the total trading strategy results. Most of the metrics saw little to no change when viewing the stocks as a group. However, individual stocks saw large swings in profit. AAPL saw its profits drop from 283% to 128.6% while F saw its profits gain 89.8%. It appears the additional information of macroeconomic indicators was beneficial to some stocks and detrimental to others. The increase to classification complexity could be blamed for the decrease in performance.

5.5 Reduction of Problem Dimensionality

With an input space of 52 dimensions, the ability for the classifier to build a good predictive model can be hurt by the complexity associated with a large search space. This is known as the “curse of dimensionality” [39]. To shrink the input space, two tools were explored; principal component analysis and a genetic algorithm to optimize input selection.

5.5.1 Principal Component Analysis (PCA)

Principal component analysis was applied to the 52 input vectors and reduced it down to its prime components. Intrinsic dimensionality estimation was performed on the 52 inputs to determine the minimum dimensionality reduction possible for PCA. Two linearly independent vectors were found to be all that was needed to store the information in the original 52 indicators. Using the 2 vectors created by the PCA reduction as inputs into a KNN classifier, the trading system was simulated.

Stock	Classification %	False Positive %	profit %	Buy And Hold %	Sharpe Ratio	BuyAndHold Sharpe Ratio	# of Trades
SPY	53.6	44.2	29	-20.8	0.0341	-0.0076	88
AAPL	52	44.7	103.7	121	0.063	0.0549	92
F	47.6	55	-46.2	26.9	-0.0078	0.0309	101
KO	49.8	47.6	6.2	19.4	0.0138	0.0239	94
WMT	46.4	53.5	-31.4	15.5	-0.049	0.021	90
BAC	51	54.3	-61.5	-66.8	-0.0223	0.0039	91
DIS	49.2	52.3	-23.4	-3.9	-0.0162	0.0102	87
MCD	49.78	48.4	43.8	40.4	0.0483	0.038	102
MRK	49.5	52	4.3	-21.5	0.0128	-0.0023	102
DD	51.3	47.9	-10.1	-24	0.0037	-0.0024	72
Average	50.0	50.0%	1.4	8.6	0.00804	0.01705	91.9

Table 5.9: Results for the KNN Using PCA to reduce the number of inputs

Table 5.9 shows the results for the classification of the PCA reduced inputs. Both average classification percentage and false positive percentage dropped to exactly 50%, showing that the two vector inputs into the classifier provided no better classification then a coin flip. The Sharpe ratio saw a big drop, now less then half the value of the buy-and-hold Sharpe ratio. Profits dropped to 1.4%. Overall, PCA seemed to reduce the amount of useful information provided to the classifier. The reduction in dimensionality did not make up for the loss in data.

5.5.2 Genetic Algorithm Dimensional Reduction

When using statistical technique to determine independence between the input vectors, the vectors value in classification is not considered. A vector that is determined to be redundant by PCA may turn out to be a great differentiator when applied to a classification mapping such as KNN or K-means. Because of this, a technique that applies the classification success of a vector to the dimensional reduction problem could yield positive results. To do this experiment, a genetic algorithm was used to do just that. A GA was constructed to select a random subset of the 52 technical indicators being used in the KNN classifier. The classification rate was used as the GA's fitness function and a population of 300 children was bred for 50 generations. The top children were bred using two point crossover, slowly improving the classification rate as the generations were created. Each child's classifier was trained and tested on data spanning from Jan 1, 2001 through Jan 1, 2006. 70% of the total data was used as training data while the remaining 30% was used for the test set. The classification percentage of the test data was used as the fitness function for the GA. The reduced data set allowed the final selected inputs to be tested over the same data as all classifiers, without involving a bias. Figure 5.8 shows the progress of the classification using a KNN classifier on Apple stock. The classification rate is negated for GA execution, so the number appears negative.

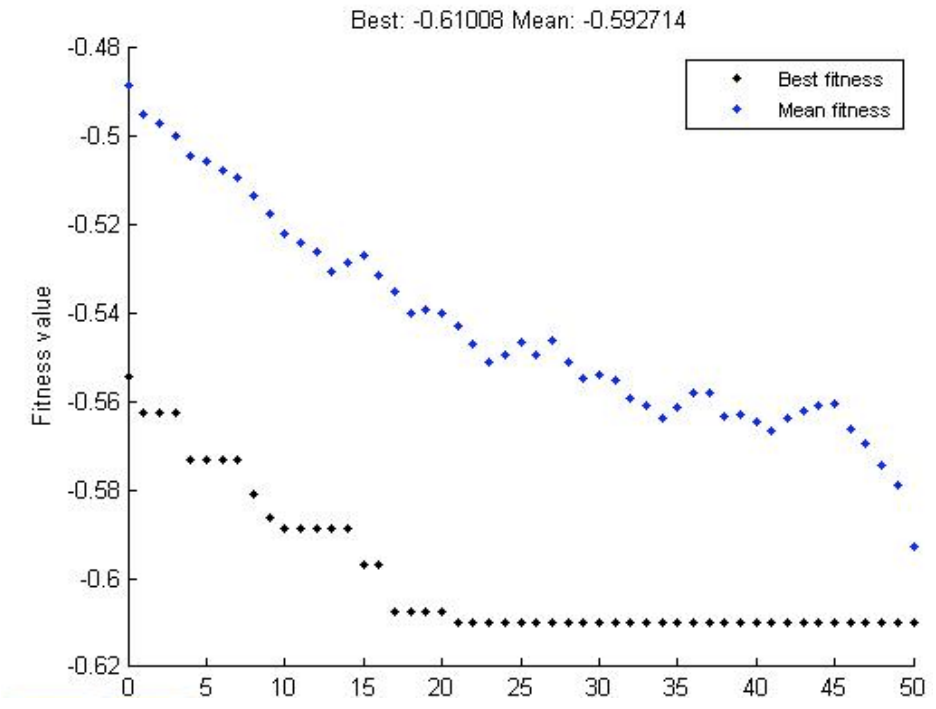

Figure 5.7: A graph of both the GA's best and mean fitness per generation

Overall, the results were quite promising. The GA made progress through the generations, resulting in a best classification rate of 61.1% by the 21st generation. Even though the average classification score continues to fall, finally reaching 59.3% by the 50th generation, the best fitness remains the same for the final 29 generations.

To achieve maximized fitness, the GA selected 18 technicals;1, 2, 4, 5, 6, 13, 19, 21, 28, 30, 32, 36, 38, 40, 41, 42, 46 and 47 from table 4.1. While a 61.1% classification percentage seems like a vast improvement from previous techniques, it needs to be remembered that this is with in-sample data. The GA performed a search, just like the KNN, but the GA uses the training and test data for its training data. To get a real measure of the performance of this system, a test needs to be run over the full data set, spanning Jan 1, 2001 to Jan 1, 2010. The results of this test are shown in Table 5.10.

Stock	Classification %	False Positive %	profit %	Buy And Hold %	Sharpe Ratio	BuyAndHold Sharpe Ratio	# of Trades
SPY	51.4	46.4	-28.6	-20.8	-0.0285	-0.0076	150
AAPL	56.1	40.8	219.5	121	0.0954	0.0549	153
F	53.5	48	178.4	26.9	0.066	0.0309	150
KO	51.8	44.9	43	19.4	0.0539	0.0239	150
WMT	51.6	47.9	62.6	15.5	0.0689	0.021	143
BAC	50.5	52.9	1.3	-66.8	0.0227	0.0039	140
DIS	50.8	50.6	31.4	-3.9	0.0317	0.0102	158
MCD	50.5	47.6	43.3	40.4	0.0474	0.038	146
MRK	50.2	51.8	-7.2	-21.5	0.0012	-0.0023	147
DD	52.3	45.9	107.2	-24	0.068	-0.0024	163
Average	51.9	47.7	65.1	8.6	0.04267	0.01705	150

Table 5.10: Results for the KNN classifier using a reduced set inputs. The reduction was done using a GA optimizing a classification of Apple stock.

Overall, we see a slight improvement in classification percentage, up 0.7% compared to when 52 technicals were used. Profits increased to 65.1%, the highest average profit achieved using any technique. F stock saw a huge boost in profits using the reduced feature set, jumping its profits to 178.4%.   The average Sharpe ratio also saw an increase in performance with the input reduction. However, some stocks performed worse with the input subset. SPY, DIS and BAC saw decreases in performance. The variability of success between stocks using this technique suggests that each stock may have a unique subset of technicals that improve performance.

Next, a GA optimization was performed for every stock in an attempt to optimize input selection for each stock. Once again, in-sample classification rate was very good, an average of 62.4%. Out of sample classification rate didn't perform nearly as well, coming it at just 50.7%. Profit gain however was quite impressive, with the average gain being 106.1% over the test period. This is over twice the gain seen before input reduce is performed. When examining stocks individually, it is seen that a few stocks saw huge profit gains, while most stocks profit decreased using the reduced feature set. F for instance, saw its profit gain go from 26.9% when using 52 technicals to 816.7 when using the 30

Stock	Classification %	False Positive %	profit %	Buy And Hold %	Sharpe Ratio	BuyAnd Hold Sharpe Ratio	# of Trades	Technicals Used
SPY	50.7	46.9	-8.9	-20.8	-0.002	-0.0076	152	2,3,5,7,8,12,14,19,20,21,23,24,25,28,31,34,40,41,48,50,51,52
AAPL	56.1	40.8	219.5	121	0.0954	0.0549	153	1,2,4,5,6,13,19,21,28,30,32,34,38,40,42,43,44,48,49,
F	54.5	47	816.7	26.9	0.1133	0.0309	153	2,6,12,13,14,15,16,18,21,24,25,26,27,28,29,30,32,33,34,35,36,37,38,39,41,42,43,50,51,52
KO	47.4	49.3	17.2	19.4	0.0285	0.0239	153	1,3,5,6,9,11,12,14,16,18,19,20,21,22,25,28,29,31,33,34,35,37,40,42,45,47,50,52
WMT	51.5	47.2	47.9	15.5	0.0573	0.021	150	2,4,5,8,9,10,13,18,20,22,27,28,29,30,41,42,43,44,45,46,47
BAC	48.8	54.2	-77.6	-66.8	-0.0274	0.0039	146	2,4,5,6,7,14,15,16,21,22,23,25,28,31,36,39,41,44,46

Stock	Classification %	False Positive %	profit %	Buy And Hold %	Sharpe Ratio	BuyAnd Hold Sharpe Ratio	# of Trades	Technicals Used
DIS	48.9	52.8	8	-3.9	0.0151	0.0102	162	1,6,8,9,13,15,21,24,27,28,30,33,37,38,39,44,46,50
MCD	51	47.1	19.9	40.4	0.028	0.038	165	1,8,9,15,16,20,24,27,28,33,38,39,43,45,48,52
MRK	46.1	55.6	-22.9	-21.5	-0.0145	-0.0023	158	2,8,10,14,15,17,18,19,21,22,25,26,32,33,34,35,36,37,39,41,42,43,47,49,50
DD	51.7	46.6	41.2	-24	0.036	-0.0024	149	1,2,5,7,9,12,13,16,17,18,19,20,24,25,28,29,32,33,37,38,39,41,43,51
Average	50.67	48.75	106.1	8.6	0.033	0.01705	154.1	

Table 5.11: Results for the KNN classifier using a reduced set inputs. The reduction was done using a GA optimizing for each stock

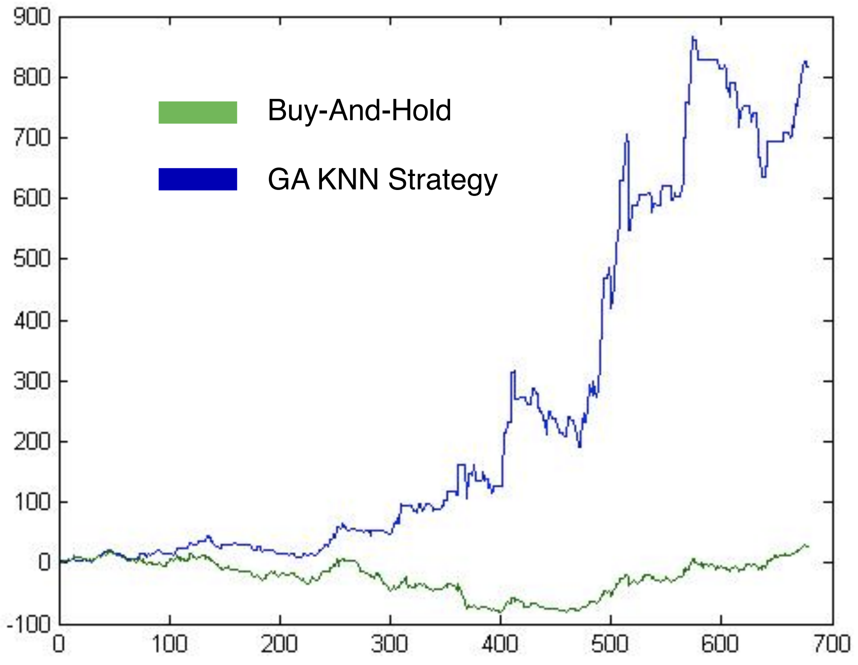

Figure 5.8: Fords 800% gain using GA selected indicators

technicals selected by the GA. This huge gain has a powerful impact on the average of all stock performance. If excluded from the average calculation, performance sees a decrease when individual technicals are selected.

The technicals selected by the GA varied from stock to stock, with no one group of technicals outperforming another. Table 5.12 shows that most technicals were selected on average just over 4 times. Input number 11, a technical derived from a 200 day moving average, was selected the least, only used by one stock. Input number 28, the 50 day moving variance, was the most common, used in 9 of the 10 stocks.

Input	Used	Input	Used	Input	Used	Input	Used	Input	Used
1	5	**12**	4	**23**	3	**34**	6	**45**	3
2	7	**13**	4	**24**	5	**35**	3	**46**	3
3	2	**14**	5	**25**	6	**36**	3	**47**	2
4	3	**15**	5	**26**	2	**37**	5	**48**	3
5	6	**16**	4	**27**	4	**38**	5	**49**	2
6	5	**17**	2	**28**	9	**39**	6	**50**	5
7	3	**18**	5	**29**	4	**40**	3	**51**	3
8	5	**19**	5	**30**	3	**41**	6	**52**	4
9	5	**20**	5	**31**	3	**42**	5		
10	2	**21**	7	**32**	4	**43**	5	Average	4.2
11	1	**22**	4	**33**	6	**44**	4		

Table 5.12: How many of each input was selected by the GA when optimizing the inputs for each stock

Chapter 6: Conclusions and Future Work

6.1 Conclusion

This work performed a survey of trading strategies involving technical indicators as inputs to machine learning classifiers in an attempt to classify stock market data into ‘up’ and ‘down’ days. The first strategy involved using 22 technical indicators as inputs into a KNN classifier. Results showed a 20% increase in average profits compared to buy-and-hold. Next, a SVM and K-means classifier were used in place of the KNN. The results showed average profits dropping 7.2% and 3.7% for SVM and K-means, respectively, when compared to the KNN strategy. This was in contrast to some of the surveyed work that showed SVM to be a superior classifier for this kind of data [27]. The classification rule was then altered to train the classifier to predict ‘up’ days for when the stock increased in value by at least 0.5%, 1.0% and 1.5%. This resulted in greater profits for stocks that had large volatility in the test period. Stocks with low volatility saw profits decrease.

An introduction of 30 more technical indicators as inputs saw average profits nearly double, from 28.6% to 47.7%. The introduction of 10 macroeconomic indicators added as inputs showed a slight decrease in performance. PCA was able to combine the 52 technical indicators into 2 inputs to be used in classification. This reduction caused a dramatic drop in average profits, down to 1.4% gain over the test period. The genetic algorithm input reduction reduced the number of inputs to an average of 22 inputs per stock.

This reduction saw average profits increase to 106.1%, a 97.5% increase over buy-and-hold.

Genetic algorithm optimization for feature selection yielded far better results than PCA. PCA ordered the eigenvalues of the co-variance matrix calculated on the training data and showed that two eigenvalues were dominant, while the remaining 50 had magnitudes too small to contribute. This result indicated that two features were sufficient as inputs. However, the GA indicated 22 features on average were needed. The GA's significant out-performance of PCA indicated that the genetic algorithms choice of an order of magnitude more features was the correct one. Additionally, the GA indicated feature selection was data dependent, where PCA suggested the same number of features could be used for all data. Both techniques resulted in unique features being selected per data set.

The techniques surveyed show a trading strategy using stock market technicals and machine learning classifiers can be created that significantly outperformed buy-and-hold. Performing preprocessing on the inputs, either with input selection through genetic algorithms, or adjusting the classification rule on the training set showed they can have a positive effect on profits and remain an interesting topic for further study.

Based on the conclusions of this work, I believe that using technical indicators as inputs into machine learn classifiers is a valid trading strategy that can beat buy-and-hold. Using a GA to select a subset of inputs for each stock in a portfolio of stocks, a KNN classifier can be trained to make profitable predictions. This work shows that not all stocks over any period of time will be successful using this strategy, so a portfolio of 10 to 20 stocks should be traded in parallel. The amount of money being invested in each stock should be large, as the high trade frequency will reduce margins due to the cost associated with buying and selling stocks. Using a classification rule that looks at the volatility of each stock and adjusts the threshold value could be an effective way to reduce trading costs through reducing the trade frequency.

6.2 Future work

There are several more variations of the current implementation that could be experimented with:

- Research what technicals/macroeconomic indicators have been shown to yield good predictive qualities. Use these as inputs to classifiers.
- In this work, only default or a limited selection of classifier parameters were used. Some classifiers, such as SVM, have been shown to outperform the KNN if the parameters are optimized for the data being used. Using a GA or other techniques to select the correct parameters for a classifier could greatly improve classification rates.
- Only a handful of basic classifiers were used in this work. Using different classifiers could yield interesting results.
- To trade a stock, a brokerage fee is applied. This fee can be anywhere from $5.00 to $20.00 per stock purchase. Integrating these fees into the simulator would increase the accuracy of trading profit per strategy.

- investigate the classification rule and its tie to market volatility in the training set.

References

[1] Malkiel, Burton G. "The efficient market hypothesis and its critics." *Journal of Economic Perspectives* (2003): 59-82.

[2] Francis Nicholson. Price-Earnings Ratios in Relation to Investment Results. *Financial Analysts Journal*. Jan/Feb 1968:105-109.

[3] Dehnad , Kosrow. "Behavioral Finance and Technical Analysis." The Capco Institute Journal of Financial Transformation. 32. no. Aug (2011): 107-111. http://www.capco.com/sites/all/files/journal-32_article-10.pdf (accessed October 2, 2012).

[4] Abarbanell, Jeffrey S., and Brian J. Bushee. "Fundamental analysis, future earnings, and stock prices." Journal of Accounting Research 35, no. 1 (1997): 1-24.

[5] Gupta, Mansi. fundamentalfinance.com, " Economic Indicators." Accessed October 27, 2012. http://economics.fundamentalfinance.com/macroeconomics/economic-indicators.php.

[6] Kirkpatrick and Dahlquist. *Technical Analysis: The Complete Resource for Financial Market Technicians*. Financial Times Press, 2006, page 3.

[7] http://www.investopedia.com/, "Quantitative Analysis." Accessed October 5, 2012. http://www.investopedia.com/terms/q/quantitativeanalysis.asp

[8] Moving markets Shifts in trading patterns are making technology ever more important, The Economist, Feb 2, 2006

[9] Teixeira, Lamartine Almeida, and Adriano Lorena Oliveira. "A method for automatic stock trading combining technical analysis and nearest neighbor classification." Expert Systems with Application. Vol 37. Oct (2010): 6885-6890.

[10] StockCharts.com - ChartSchool , "Moving Averages - Simple and Exponential." Accessed October 5, 2012. http://stockcharts.com/help/doku.php?id=chart_school:technical_indicators:moving_averages.

[11] StockCharts.com - ChartSchool , "Moving Average Convergence-Divergence (MACD)." Accessed October 5, 2012. http://stockcharts.com/help/doku.php?d=chart_school:technical_indicators:moving_average_conve.

[12] StockCharts.com - ChartSchool , "Relative Strength Index (RSI)." Accessed October 5, 2012. http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:relative_strength_index_rsi.

[13] StockCharts.com - ChartSchool , "Bollinger Bands." Accessed October 5, 2012. http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:bollinger_bands.

[14] StockCharts.com - ChartSchool , "Stochastic Oscillator." Accessed October 5, 2012. http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:stochastic_oscillator.

[15] StockCharts.com - ChartSchool , "Rate of Change (ROC)." Accessed October 5, 2012. http://stockcharts.com/help/doku.php?id=chart_school:technical_indicators:rate_of_change_roc_a.

[16] http://www.mathsisfun.com/, "Standard Deviation and Variance." Accessed October 5, 2012. http://www.mathsisfun.com/data/standard-deviation.html.

[17] StockCharts.com - ChartSchool, "Commodity Channel Index (CCI)." Accessed October 5, 2012. http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:commodity_channel_index_cci.

[18] http://www.onlinetradingconcepts.com/, "Linear Regression Line." Accessed October 5, 2012. http://www.onlinetradingconcepts.com/TechnicalAnalysis/LinRegLine.html.

[19] StockCharts.com - ChartSchool , "Chaikin Oscillator." Accessed October 5, 2012. http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:chaikin_oscillator.

[20] http://www.forexrealm.com/, "Disparity Index." Accessed October 27, 2012. http://www.forexrealm.com/technical-analysis/technical-indicators/disparity-index.html.

[21] StockCharts.com - ChartSchool , "William %R." Accessed October 5, 2012. http://stockcharts.com/school/doku.phpid=chart_school:technical_indicators:williams_r.

[22] The Federal Reserve System: Purposes and Functions, "The Implementation of Monetary Policy." Last modified 24 August 2011. Accessed October 9, 2012. http://www.federalreserve.gov/pf/pdf/pf_3.pdf.

[23] Federal Reserve Bank of St. Louis, "Trade Weighted U.S. Dollar Index: Broad (TWEXB)." Accessed October 9, 2012. http://research.stlouisfed.org/red2/ series/TWEXB.

[24] Federal Reserve Bank of St. Louis, "Trade Weighted U.S. Dollar Index: Major Currencies (TWEXM)." Accessed October 9, 2012. http://research.stlouisfed.org/fred2/series/TWEXM/.

[25] MacQueen, J. B. (1967). "Some Methods for classification and Analysis of Multivariate Observations". **1**. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. University of California Press. pp. 281–297.

[26] Greg Hamerly and Charles Elkan. Alternatives to the k-means algorithm that find better clusterings. In Konstantinos Kalpakis, Nazli Goharian, and David Grossmann, editors, Proceedings of the Eleventh International Conference on Information and Knowledge Management (CIKM-02), pages 600–607, New York, November 4–9 2002. ACM Press

[27] Kim, K. J. (2003). Financial time series forecasting using support vector machines. Neurocomputing, 55, 307–319.

[28] Yu, Lean, Huanhuan Chen, Shouyang Wang, and Kin Keung Lai. "Evolving Least Squares Support Vector Machines for Stock Market Trend Mining." IEEE Transactions on Evolutionary Computation. Vol 13. no 1. Feb (2009): 87-102.

[29] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” J. Comput. Syst. Sci., vol. 55, pp. 119–139, 1997.

[30] Lasky, Paul. "Predicting the E-mini - A new approach."Futures, July 2003, 40-43.

[31] Yahoo!, "Yahoo! Finance." Accessed October 11, 2012. http://finance.yahoo.com/.

[32] TicTacTec, "TA-Lib : Technical Analysis Library." Accessed October 11, 2012. http://www.ta-lib.org/index.html.

[33] Federal Reserve Bank of St. Louis, "Federal Reserve Economic Data." Accessed October 7, 2012. http://research.stlouisfed.org/fred2/.

[34] http://www.mathworks.com/, "Normc." Accessed October 27, 2012. http://www.mathworks.com/help/nnet/ref/normc.html.

[35] Yuille, Brigitte . http://www.investopedia.com/, "Short Selling: What Is Short Selling?." Accessed October 11, 2012. http://www.investopedia.com/ university/shortselling/shortselling1.asp

[36] Smith , Lindsay. University of Otago, Department of Computer Science , "A tutorial on Principal Components Analysis." Last modified February 26, 2002. Accessed October 2, 2012. http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf.

[37] Lahmiri, Salim. "A Comparison of PNN and SVM for Stock Market Trend Prediction using Economic and Technical Information." International Journal of Computer Applications (0975 – 8887). Vol 29, No 3 Sept (2011): 24-30.

[38] Magdon-Ismail M, Nicholson A, Abu-Mostafa YS. Financial markets: very noisy information processing. Proc IEEE 1998;86(11): 2184–95.

[39] Houle, M. E.; Kriegel, H. P.; Kröger, P.; Schubert, E.; Zimek, A. (2010). "Can Shared-Neighbor Distances Defeat the Curse of Dimensionality?" Scientific and Statistical Database Management. Lecture Notes in Computer Science. 6187. pp. 482

[40] Tsai, Shu-Jen Steven. ETDS, "Power Transformer Partial Discharge (PD) Acoustic Signal Detection using Fiber Sensors and Wavelet Analysis, Modeling, and Simulation." Last modified 2002. Accessed October 30, 2012. http://scholar.lib.vt.edu/theses/available etd-12062002-152858/ unrestricted/Chapter4.pdf.

[41] Renfree , Josiah. matlabcentral, "Historical Stock Data downloader." Last modified 2008. Accessed November 10, 2011. http://www.mathworks.com/matlabcentral/fileexchange/18458.

[42] Sharpe, W., 1994, “The Sharpe Ratio", Journal of Portfolio Management 21, 49-58.

[43] Van der Maaten, Laurens. Laurens van der Maaten, "Matlab Toolbox for Dimensionality Reduction." Last modified 2012. http://homepage.tudelft.nl/19j49/Home.html.

[44] Sui, Xueshen, Qinghua Hu, Daren Yu, Zongxia Xie, and Zhongying Qi. "A Hybrid Method for Forecasting Stock Market Trend Using Soft- Thresholding De-noise Model and SVM." *Rough Sets, Fuzzy Sets, Data Mining and Granular Computing* (2007): 387-394.

Appendix A: Code

The code that implements the trading strategies is done through class abstraction. The base class is defined as the base classifier (baseclassifier.m). Nearly all computation is done in the run command of baseclassifier. Extended off the base classifier is each of the classifiers used in this work; KNN(knn.m), K-Means(km.m) and SVM(svm.m). In each of this these, the implementation of each classifier is specified.

The properties defined in Baseclassifier are used to record data to be outputted to the console and to determine which trading strategy is used. The following properties are toggles that can be turned on by setting the value to ‘1’ or off by setting the value to ‘0’:

- ‘UseFiltering’ - filters the stock data, uses filtered data to create technicals
- ‘UseMacroEconomicData’ - include macroeconomic data as inputs into classifier
- ‘AddFilteringToIndicators’ - filters stock data, creates filtered technicals and adds them to already unfiltered list of technicals.
- ‘UseStatisticalReduction’ - reduces the input vector using statistical reduction techniques.

Other properties are used to set variable values:

- ‘PercentTestData’ - what percent of the data will be used for test
- ‘StopGain’ - sets the size of StopGain to use
- ‘StopLoss’ - sets the size of StopLoss to use

- ‘StatisticalReductionMethod’ - set the statistical reduction method to use. example being ‘PCA’
- ‘PercentGainNeededToQualifyAsABuyDay’ - set the threshold value for classification rule
- ‘ClassifierStepSize’ - set how often the classifier should retrain the classifier for walk forward testing

To run a simulated trading strategy, the first thing to do is to instantiate a classifier object. An example of instantiating a KNN classifier is seen below:

```
knn1 = knn('01012001','01012010','d','spy');
```

The first parameter sets the start date for the stock market data, while the second parameter sets the end date for the market data. The ‘d’ specifies the frequency of data and ‘spy’ is the stock that is being investigated.

Next, the percentage of data to be used as test data can be set by using the command below, the default value is specified in baseclassifier, but can be overwritten by:

```
knn1.PercentTestData = .3;
```

To run the simulation, call the run method on your object.

knn1.Run

This will run the simulation and output all the results to the console of Matlab.

experiment1.m is a script running many different simulations in one file.

All the code, in its respective files is below:

****Baseclassifier.m****

```
classdef BaseClassifier < handle
        properties
      StartDate
      EndDate
      Frequency
      Ticker
      StockData
      PercentTestData
      TestSetSize
      TrainingSetSize
      CorrectDailyTrades
      DateOffset
      FilteredStockData
      TestTechnicals
      TrainingTechnicals
      Result
      ClassificationPercentage
      UseFiltering
      TestSetAnswers
      TradingStrategyProfit
      StopGain
      TradingStrategyProfitWithStops
      StopLoss
      FalsePositivePercentage
      BuyAndHoldProfit
      PercentGainNeededToQualifyAsABuyDay
      FrequencyOffset
      CorrectDailyTradesAdjustedForPercentageGain
```

```
    Equity
    TechnicalsUsed
    ClassifierStepSize
    UseMacroEconomicData
    MacroEconomicData
    TestIndicators
    TrainingIndicators
    AddFilteredToIndicators
    StockTechnicals
    TechnicalsUsedInDE
    TechnicalNames
    TechnicalNamesUsedByDE
    StatisticalDimensionEstimate
    UseStatisticalReduction
    StatisticalReductionMethod
    UseRankFeatureReduction
    FeatureRanks
    NumberOfFeaturesToReduceTo
    PercentProfit
    BuyAndHoldProfitPercentage
    BuyAndHoldEquity
    Trades
    SharpeRatio
    SharpeRatioBuyAndHold
    DailyPercentMovementBuyAndHold
    DailyPercentMovement
    AdjSharpeRatio


end
      methods
    function obj = BaseClassifier(StartDate, EndDate, Frequency, Ticker)
        obj.StartDate = StartDate;
        obj.EndDate = EndDate;
        obj.Frequency = Frequency;
        obj.Ticker = Ticker;
        obj.StockData = [];
        obj.PercentTestData = .8;
        obj.TestSetSize = 0;
        obj.TrainingSetSize = 0;
        obj.CorrectDailyTrades = [];
        obj.DateOffset = 1;
        obj.FilteredStockData = [];
        obj.TrainingTechnicals = [];
        obj.TestTechnicals = [];
obj.StockTechnicals = [];
obj.Result = [];
obj.ClassificationPercentage = 0;
obj.UseFiltering = 0;
obj.TestSetAnswers = [];
obj.TradingStrategyProfit = 0;
obj.StopGain = .5;
obj.StopLoss = .5;
obj.TradingStrategyProfitWithStops = 0;
obj.FalsePositivePercentage = 0;
obj.BuyAndHoldProfit = 0;
obj.PercentGainNeededToQualifyAsABuyDay = .00;
obj.FrequencyOffset = 0;
obj.CorrectDailyTradesAdjustedForPercentageGain = [];
obj.Equity = [];
obj.TechnicalsUsed = ones(1,200);
obj.TechnicalsUsedInDE = ones(1,200);
obj.ClassifierStepSize = 1;
obj.UseMacroEconomicData = 0;
obj.TestIndicators = [];
obj.TrainingIndicators = [];
obj.MacroEconomicData = [];
obj.AddFilteredToIndicators = 0;
obj.TechnicalNames = ['RSI                                          ';...
               'BBandsHigh                                   ';...
               'BBandsLow                                    ';...
               'Stochastic %K                                ';...
               'Slow %D                                      ';...
               'StochK One Day Gain                          ';...
               'StochD One Day Gain                          ';...
               'One Day % gain                               ';...
               '(Close-Low)/Close                            ';...
               '20 Day Moving Average One Day Gain Percentage ';...
               '200 Day Moving Average One Day Gain Percentage';...
               '(20DayMA-200DayMA)/200DayMA                  ';...
               '(Close-200DayMA)/200DayMVA                   ';...
               '(Close-min5DayLag)/min5DayLag                ';...
               '(Close-max5DayLag)/max5DayLag                ';...
               'Volume One Day % Gain                        ';...
               '20 day VolumeMA one Day Gain Percentage      ';...
               '200 day VolumeMA one Day Gain Percentage     ';...
               '(20DayVMA-200DayVMA)/200DayVMA               ';...
               '(Volume-200DayVMA)/200DayVMA                 ';...
               '(Volume-min5DayVolumeLag)/min5DayVolumeLag)  ';...
               '(Volume-max5DayVolumeLag)/max5DayVolumeLag)  ';...
```

```
          'ROC                                    ';...
          'One Day Momentum                       ';...
          '7 day moving average                   ';...
          '50 day moving average                  ';...
          '200 day moving average                 ';...
          '7 day moving variance                  ';...
          '50 day moving variance                 ';...
          '200 day moving variance                ';...
          'moving variance ratio                  ';...
          'moving average convergence/divergence 20/40   ';...
          'Price Oscillator                       ';...
          'commodity channel index                ';...
          'Linear regression                      ';...
          'accumulation/distribution oscillator   ';...
          'disparity 3                            ';...
          'disparity 7                            ';...
          'Williams %R                            ';...
          '%D                                     ';...
          '1 day lag                              ';...
          '2 day lag                              ';...
          '3 day lag                              ';...
          '4 day lag                              ';...
          '5 day lag                              ';...
          '6 day lag                              ';...
          '7 day lag                              ';...
          '8 day lag                              ';...
          '9 day lag                              ';...
          '10 day lag                             ';...
          'EMV                                    ';...
          '(close-max(5previousDayCloses)         '];
obj.TechnicalNames = cellstr(obj.TechnicalNames);
obj.TechnicalNamesUsedByDE = [];
obj.StatisticalDimensionEstimate = 5;
obj.UseStatisticalReduction = 0;
obj.StatisticalReductionMethod = 'PCA';
obj.UseRankFeatureReduction = 0;
obj.FeatureRanks = [];
obj.NumberOfFeaturesToReduceTo = 10;
obj.PercentProfit = 1;
obj.BuyAndHoldProfitPercentage = 1;
obj.BuyAndHoldEquity = 0;
obj.Trades = 0;
obj.SharpeRatio = 0;
obj.SharpeRatioBuyAndHold = 0;
obj.DailyPercentMovementBuyAndHold = [];
```

```
            obj.DailyPercentMovement = [];
            obj.AdjSharpeRatio = 0;


            determineFrequencyOffset(obj);
            getStockData(obj);
            reverseDateOrder(obj);

        end

        function DEMacro(obj)

            if obj.UseMacroEconomicData == 1
                getMacroEconomicData(obj);
                %obj.TechnicalsUsed = [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1];
            end

        end

        function SetTechnicalArrays(obj)
           getTestSetSize(obj);
           filteredTestTechnicalsTotal = [];
           testTechnicals = [];
           obj.StockTechnicals = calculateTechnicals(obj, obj.StockData);
           if obj.AddFilteredToIndicators == 1
               obj.UseFiltering = 1;
               testTechnicals = obj.StockTechnicals(end-obj.TestSetSize+1:end,:);
           end
           if obj.UseFiltering == 1
              for i=1:obj.TestSetSize
                  testingStockData = getTestStockData(obj, i);
                  testingStockDataFiltered = waveletFilter(testingStockData);
                  %testingStockDataFiltered = TechnicalHelper(obj,
testingStockDataFiltered);
                  filteredTestTechnicals = calculateTechnicals(obj,
testingStockDataFiltered);
                  filteredTestTechnicalsTotal =
[filteredTestTechnicalsTotal;filteredTestTechnicals(end:end,:)];
              end
              obj.TestTechnicals = [testTechnicals filteredTestTechnicalsTotal];
           else
               obj.TestTechnicals = obj.StockTechnicals(end-obj.TestSetSize+1:end,:);
           end
        end



        function TechnicalsUsedNamed(obj)
            b = find(obj.TechnicalsUsedInDE > 0);
            for i=1:size(b,2)
                obj.TechnicalNamesUsedByDE = [obj.TechnicalNamesUsedByDE;
obj.TechnicalNames(b(i))];
            end
        end



        function Run(obj)
            getCorrectDailyTrades(obj);
            i = obj.ClassifierStepSize;
            while i <= obj.TestSetSize
                trainingMacroEconomicData = [];
                testingMacroEconomicData = [];
                trainingStockData = getTrainingStockData(obj, i);

                if obj.UseFiltering == 1;
                    if obj.AddFilteredToIndicators == 1
                        trainingStockData2 = waveletFilter(trainingStockData);
                        obj.TrainingTechnicals = calculateTechnicals(obj,
trainingStockData);
                        TrainingTechnicals2 = calculateTechnicals(obj,
trainingStockData2);
                        obj.TrainingTechnicals = [obj.TrainingTechnicals
TrainingTechnicals2];
                    else
                        trainingStockData = waveletFilter(trainingStockData);
                        obj.TrainingTechnicals = calculateTechnicals(obj,
trainingStockData);
                    end
                else
                    obj.TrainingTechnicals = calculateTechnicals(obj, trainingStockData);
                end


                if obj.UseMacroEconomicData == 1
                    trainingMacroEconomicData =
getTrainingMacroEconomicData(obj,i);
                    testingMacroEconomicData = getTestingMacroEconomicData(obj,i);
            end
            obj.TrainingIndicators = [obj.TrainingTechnicals
trainingMacroEconomicData];
            obj.TestIndicators = [obj.TestTechnicals(end-obj.TestSetSize-
obj.ClassifierStepSize+i+1:end-obj.TestSetSize+i,:) testingMacroEconomicData];

            if obj.UseRankFeatureReduction == 1;
               obj.FeatureRanks = rankfeatures(obj.TrainingIndicators',
obj.CorrectDailyTradesAdjustedForPercentageGain(i-obj.ClassifierStepSize
+1:end-obj.TestSetSize-obj.ClassifierStepSize+i,:));
               for j=1:size(obj.FeatureRanks)
                  if(obj.FeatureRanks(j) > obj.NumberOfFeaturesToReduceTo)
                     obj.TechnicalsUsedInDE(j) = 0;
                  else
                     obj.TechnicalsUsedInDE(j) = 1;
                  end

               end
            end

            RemoveTechnicals(obj);
            RemoveTechnicalsUsedInDE(obj);


            if obj.UseStatisticalReduction == 1;
               if i == obj.TestSetSize
                  obj.StatisticalDimensionEstimate =
round(intrinsic_dim(obj.TrainingIndicators, 'EigValue'));
               end
               [a b] = compute_mapping(obj.TrainingIndicators,
obj.StatisticalReductionMethod, obj.StatisticalDimensionEstimate);
               %[obj.TrainingIndicators mapping] =
compute_mapping(obj.TrainingIndicators, obj.StatisticalReductionMethod,
obj.StatisticalDimensionEstimate, 7);
               obj.TrainingIndicators = a;
               mapping = b;
               obj.TestIndicators = OUT_OF_SAMPLE(obj.TestIndicators,mapping);
               %obj.TestIndicators =
OUT_OF_SAMPLE(obj.TestIndicators,mapping);
            end

            %normalize indicators
            obj.TrainingIndicators = normc(obj.TrainingIndicators);
            obj.TestIndicators = normc(obj.TestIndicators);
            temp = Classify(obj,i);
            obj.Result = [obj.Result; temp];

            if obj.TestSetSize < i + obj.ClassifierStepSize && i ~= obj.TestSetSize
               i = obj.TestSetSize;
               obj.ClassifierStepSize =
mod(obj.TestSetSize,obj.ClassifierStepSize);
            else
               i = i + obj.ClassifierStepSize;
            end
         end
         evaluate(obj)
         %calculateTechnicals(obj);
         %GA(obj);
      end

      function reverseDateOrder(obj)
         obj.StockData.Date = obj.StockData.Date(end:-1:1);
         obj.StockData.Open = obj.StockData.Open(end:-1:1);
         obj.StockData.High = obj.StockData.High(end:-1:1);
         obj.StockData.Low = obj.StockData.Low(end:-1:1);
         obj.StockData.oldClose = obj.StockData.Close(end:-1:1);
         obj.StockData.Close = obj.StockData.AdjClose(end:-1:1);
         obj.StockData.Volume = obj.StockData.Volume(end:-1:1);
      end

      function determineFrequencyOffset(obj)
         if obj.Frequency == 'd'
            obj.FrequencyOffset = 252;
         end
         if obj.Frequency == 'w'
            obj.FrequencyOffset = 52;
         end
      end

      function getStockData(obj)
         paddedStartDate = str2num(obj.StartDate);
         paddedStartDate = paddedStartDate-obj.DateOffset;
         paddedStartDate = num2str(paddedStartDate);
         paddedStartDate = strcat('0',paddedStartDate);
         %get stock data
         obj.StockData =
hist_stock_data(paddedStartDate,obj.EndDate,obj.Ticker,'frequency',obj.Frequen
cy);
      end
```

```
        function getTestSetSize(obj)
           obj.TestSetSize =
round((size(obj.StockData.Close(obj.DateOffset*obj.FrequencyOffset+1:end-1),
1))*obj.PercentTestData);
        end

        function getCorrectDailyTrades(obj)
           obj.CorrectDailyTrades =
[obj.StockData.Close(obj.DateOffset*obj.FrequencyOffset+2:end) >
obj.StockData.Close(obj.DateOffset*obj.FrequencyOffset+1:end-1);0];
           obj.CorrectDailyTradesAdjustedForPercentageGain =
[obj.StockData.Close(obj.DateOffset*obj.FrequencyOffset+2:end)-
(obj.PercentGainNeededToQualifyAsABuyDay*obj.StockData.Close(obj.DateOffs
et*obj.FrequencyOffset+1:end-1)) >
obj.StockData.Close(obj.DateOffset*obj.FrequencyOffset+1:end-1);0];
        end

        function getMacroEconomicData(obj)
           FedRate = macroEconImport('DFF.csv',obj);
           CanToUS = macroEconImport('DEXCAUS.csv',obj);
           JapToUS = macroEconImport('DEXJPUS.csv',obj);
           SwitzToUS = macroEconImport('DEXSZUS.csv',obj);
           USToEuro = macroEconImport('DEXUSEU.csv',obj);
           USToPound = macroEconImport('DEXUSUK.csv',obj);
           ThreeMonthTBill = macroEconImport('DTB3.csv',obj);
           SixMonthTBill = macroEconImport('DTB6.csv',obj);
           TradeWeightXBroad = macroEconImport('DTWEXB.csv',obj);
           TradeWeightXMajor = macroEconImport('DTWEXM.csv',obj);
           obj.MacroEconomicData =
[FedRate,CanToUS,JapToUS,SwitzToUS,USToEuro,USToPound,ThreeMonthTBil
l,SixMonthTBill,TradeWeightXBroad,TradeWeightXMajor];
        end

         function evaluate(obj)
          obj.TestSetAnswers = obj.CorrectDailyTrades(end-obj.TestSetSize
+1:end);
          successes =(obj.TestSetAnswers==obj.Result);
          successSum = sum(successes);
          obj.ClassificationPercentage = successSum/obj.TestSetSize;
          obj.BuyAndHoldProfit = obj.StockData.Close(end)-
obj.StockData.Close(end-obj.TestSetSize+1);
          %obj.BuyAndHoldProfitPercentage = (obj.StockData.Close(end)-
obj.StockData.Close(end-obj.TestSetSize+1))/obj.StockData.Close(end-
obj.TestSetSize+1);
          falsePositiveRate =0;
        for i=1:obj.TestSetSize-1

            if obj.Result(i)== 1
                obj.TradingStrategyProfit = obj.TradingStrategyProfit+
(obj.StockData.Close(end-obj.TestSetSize+i+1)-obj.StockData.Close(end-
obj.TestSetSize+i));
                obj.PercentProfit = obj.PercentProfit*(obj.StockData.Close(end-
obj.TestSetSize+i+1)/obj.StockData.Close(end-obj.TestSetSize+i));
                obj.DailyPercentMovement(i) = ((obj.StockData.Close(end-
obj.TestSetSize+i+1)/obj.StockData.Close(end-obj.TestSetSize+i))-1)*100;
                if obj.Result(i+1) == 0
                    obj.Trades = obj.Trades + 1;
                end
            end

            obj.DailyPercentMovementBuyAndHold(i) =
((obj.StockData.Close(end-obj.TestSetSize+i+1)/obj.StockData.Close(end-
obj.TestSetSize+i))-1)*100;


            obj.BuyAndHoldProfitPercentage =
obj.BuyAndHoldProfitPercentage*(obj.StockData.Close(end-obj.TestSetSize+i
+1))/obj.StockData.Close(end-obj.TestSetSize+i);


            obj.Equity(i) = (obj.PercentProfit-1)*100;
            obj.BuyAndHoldEquity(i) = ((obj.StockData.Close(end-obj.TestSetSize
+i+1)-obj.StockData.Close(end-obj.TestSetSize+1))/obj.StockData.Close(end-
obj.TestSetSize+1))*100;

            if obj.Result(i)== 1
                if (obj.StockData.High(end-obj.TestSetSize+i+1)-
obj.StockData.Close(end-obj.TestSetSize+i))/obj.StockData.Close(end-
obj.TestSetSize+i) > obj.StopGain
                    obj.TradingStrategyProfitWithStops =
obj.TradingStrategyProfitWithStops+obj.StockData.Close(end-obj.TestSetSize
+i)*obj.StopGain;
                elseif (obj.StockData.Close(end-obj.TestSetSize+i)-
obj.StockData.Low(end-obj.TestSetSize+i+1))/obj.StockData.Close(end-
obj.TestSetSize+i) > obj.StopLoss
                    obj.TradingStrategyProfitWithStops =
obj.TradingStrategyProfitWithStops-obj.StockData.Close(end-obj.TestSetSize
+i)*obj.StopLoss;
                else
```

```
                    obj.TradingStrategyProfitWithStops = 
obj.TradingStrategyProfitWithStops+(obj.StockData.Close(end-obj.TestSetSize+i
+1)-obj.StockData.Close(end-obj.TestSetSize+i));
                end
            end

            if obj.Result(i)==1 && obj.TestSetAnswers(i)==0
                falsePositiveRate = falsePositiveRate+1;
            end

        end

        obj.FalsePositivePercentage = falsePositiveRate/(sum(obj.Result==1));

        plot(obj.Equity)
        hold all
        plot(obj.BuyAndHoldEquity)
        obj.PercentProfit = obj.PercentProfit-1;
        obj.BuyAndHoldProfitPercentage=obj.BuyAndHoldProfitPercentage-1;
        stdProfit = std(obj.DailyPercentMovement);
        stdBuyAndHold = std(obj.DailyPercentMovementBuyAndHold);

        obj.SharpeRatio = (mean(obj.DailyPercentMovement))/stdProfit;
        obj.SharpeRatioBuyAndHold = 
(mean(obj.DailyPercentMovementBuyAndHold))/stdBuyAndHold;

        j=1;
        for i=1:length(obj.DailyPercentMovement)-1
            if obj.DailyPercentMovement(i) ~= 0
                dailyPercentMovement(j) = obj.DailyPercentMovement(i);
                j = j+1;
            end
        end
        obj.AdjSharpeRatio = (mean(dailyPercentMovement))/
std(dailyPercentMovement);

    end

    function GA(obj)
       Classify(obj)
    end

    function RemoveTechnicals(obj)
        i = 1;
        j = 1;
```

```
            iternants = size(obj.TestIndicators,2);
            while j <= iternants
                if obj.TechnicalsUsed(j) <= 0
                    obj.TestIndicators(:,i) = [];
                    obj.TrainingIndicators(:,i) = [];
                else
                    i=i+1;
                end
                j=j+1;
            end
        end

        function RemoveTechnicalsUsedInDE(obj)
            i = 1;
            j = 1;
            iternants = size(obj.TestIndicators,2);
            while j <= iternants
                if obj.TechnicalsUsedInDE(j) <= 0
                    obj.TestIndicators(:,i) = [];
                    obj.TrainingIndicators(:,i) = [];
                else
                    i=i+1;
                end
                j=j+1;
            end
        end

end

    methods(Abstract)
        Classify(obj)
    end
end
```

calculateTechnicals.m

```
function[technicals] = calculateTechnicals(obj, stockData)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


ema7 = TA_EMA(stockData.Close, 7);  %7 day moving average
```

```
ema50 = TA_EMA(stockData.Close, 50);   %50 day moving average
ema200 = TA_EMA(stockData.Close, 200);  %200 day moving average
[macd, macdsig, macdHis]= TA_MACD(stockData.Close, 20, 40, 9);  %
[macd2, macdsig2, macdHis2]= TA_MACD(stockData.Close, 3, 9, 3);
rsi = TA_RSI(stockData.Close, 14);
adx = TA_ADX(stockData.High, stockData.Low, stockData.Close, 14);
%closeGTema200 = stockData.Close > ema200;
[bBandsHigh, bBandsMid, bBandsLow] = TA_BBANDS(stockData.Close,7,20,2);
[stochK,stochD] = TA_STOCH(stockData.High,stockData.Low,stockData.Close);
volume = stockData.Volume;
VMAs = TA_MA(stockData.Volume,20);
VMAl = TA_MA(stockData.Volume,200);
PMAs = TA_MA(stockData.Close,20);
PMAl = TA_MA(stockData.Close,200);
I6temp = stochK(2:end)-stochK(1:end-1);
I7temp = stochD(2:end)-stochD(1:end-1);
I8temp = (stockData.Close(2:end)-stockData.Close(1:end-1))./
stockData.Close(1:end-1);
I10temp = (PMAs(2:end)-PMAs(1:end-1))./PMAs(1:end-1);
I11temp = (PMAl(2:end)-PMAl(1:end-1))./PMAl(1:end-1);
I12temp = (PMAs(2:end)-PMAl(1:end-1))./PMAl(1:end-1);
I16temp = volume(2:end)-volume(1:end-1)./volume(1:end-1);
I17temp = VMAs(2:end) - VMAs(1:end-1)./VMAs(1:end-1);
I18temp = VMAl(2:end) - VMAl(1:end-1)./VMAl(1:end-1);
I19temp = VMAs(2:end) - VMAl(1:end-1)./VMAl(1:end-1);

%A Comparison of PNN and SVM for Stock Market Trend Prediction using
Economic and Technical Information
MPM = [0;0;((stockData.High(3:end)-stockData.Low(3:end))./
(stockData.High(1:end-2)-(stockData.Low(1:end-2))))/2;];
BR = (stockData.Volume./(stockData.High-stockData.Low))./2;
EMV = MPM./BR;
X4 = [0;0;0;0;0];
for i=6:size(stockData.Close,1)
temp = [stockData.Close(i-1) stockData.Close(i-2) stockData.Close(i-3)
stockData.Close(i-4) stockData.Close(i-5)];
X4 =  [X4;stockData.Close(i)-max(temp)];
end



ema5 = TA_EMA(stockData.Close, 5);
ema10 = TA_EMA(stockData.Close, 10);
disp5 = stockData.Close ./ ema5;
```

```
disp10 = stockData.Close ./ ema10;


ema3 = TA_EMA(stockData.Close, 3);
ema30 = TA_EMA(stockData.Close, 30);
disp3 = stockData.Close ./ ema3;
disp7 = stockData.Close ./ ema7;
disp30 = stockData.Close ./ ema30;
disp200 = stockData.Close ./ ema200;

I1 = rsi;  %RSI
I2 = (stockData.Close - bBandsHigh)./bBandsHigh;
I3 = (stockData.Close - bBandsLow)./bBandsLow;
I4 = stochK;  % Stochastic %K
I5 = stochD;  % slow %D
I6 = [0;I6temp];
I7 = [0;I7temp];
I8 = [0;I8temp];
I9 = (stockData.Close - stockData.Low)./(stockData.High-stockData.Low);
I10 = [0;I10temp];
I11 = [0;I11temp];
I12 = [0;I12temp];
I13 = (stockData.Close - PMAI)./PMAI;
I14 = (stockData.Close - TA_MIN(stockData.Close,5))./TA_MIN(stockData.Close,
5);
I15 = (stockData.Close - TA_MAX(stockData.Close,5))./
TA_MAX(stockData.Close,5);
I16 = [0;I16temp];
I17 = [0;I17temp];
I18 = [0;I18temp];
I19 = [0;I19temp];
I20 = (volume - VMAI)./VMAI;
I21 = (volume - TA_MIN(volume,5))./TA_MIN(volume,5);
I22 = (volume - TA_MAX(volume,5))./TA_MAX(volume,5);

%inputs from paper "Evolving Least Squares Support Vector Machines for
%Stock Market Trend Mining"

I23 = TA_ROC(stockData.Close);  %Rate of change (1 day look back)
I24 = [0;stockData.Close(2:end)-stockData.Close(1:end-1)];  %Momentum (1 day
look back)
I25 = ema7;  %7 day moving average
I26 = ema50;   %50 day moving average
I27 = ema200;   %200 day moving average
I28 = movingvar(stockData.Close,7);  % 7 day moving variance
```

```
I29 = movingvar(stockData.Close,50);   %50 day moving variance
I30 = movingvar(stockData.Close,200);   %200 day moving variance



I31 = I28.^2./[0;I28(2:end)-I28(1:end-1)].^2;   %moving variance ratio
for i=1:size(I31,1)
   if(I31(i) == inf)
      I31(i) = 0;
   end
end



I32 = macd;  %moving average convergence/divergence 20/40
I33 = TA_APO(stockData.Close);  %Price Oscillator
I34 = TA_CCI(stockData.High,stockData.Low,stockData.Close);  % commodity
channel index
I35 = TA_LINEARREG(stockData.Close);  % Linear regression
I36 =
TA_ADOSC(stockData.High,stockData.Low,stockData.Close,stockData.Volume);
% accumulation/distribution oscillator
I37 = disp5;  %disparity 3
I38 = disp10;  %disparity 7
I39 = TA_WILLR(stockData.High,stockData.Low,stockData.Close);  %Williams
%R
[placeHolder I40] =
TA_STOCHF(stockData.High,stockData.Low,stockData.Close);  % %D
I41 = TA_LAG(stockData.Close,1); %lagging indicator, 1-10 days
I42 = TA_LAG(stockData.Close,2);
I43 = TA_LAG(stockData.Close,3);
I44 = TA_LAG(stockData.Close,4);
I45 = TA_LAG(stockData.Close,5);
I46 = TA_LAG(stockData.Close,6);
I47 = TA_LAG(stockData.Close,7);
I48 = TA_LAG(stockData.Close,8);
I49 = TA_LAG(stockData.Close,9);
I50 = TA_LAG(stockData.Close,10);
I51 = EMV;
I52 = X4;
```

```
technicals =
[I1,I2,I3,I4,I5,I6,I7,I8,I9,I10,I11,I12,I13,I14,I15,I16,I17,I18,I19,I20,I21,I22,I23,I24,I
25,I26,I27,I28,I29,I30,I31,I32,I33,I34,I35,I36,I37,I38,I39,I40,I41,I42,I43,I44,I45,I4
6,I47,I48,I49,I50,I51,I52];
technicals = technicals(obj.DateOffset*obj.FrequencyOffset+1:end,:);
```

****experiment1.m****

```
clear
addpath 'ta-lib-0.0.3_ml2007a-mex_w32';
addpath 'drtoolbox';
knn1 = knn('01012001','01012010','d','spy');
knn1.TechnicalsUsed = [0 1 1 0 1 0 1 1 0 0 0 1 0 1 0 0 0 0 1 1 1 0 1 1 1 0 0 1 0 0
1 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0];
knn1.PercentTestData = .3;
knn1.PercentGainNeededToQualifyAsABuyDay = .000;
knn1.ClassifierStepSize = 377;
knn1.UseMacroEconomicData = 0;
knn1.UseFiltering = 0;
knn1.AddFilteredToIndicators = 0;
knn1.UseRankFeatureReduction = 0;
knn1.UseStatisticalReduction = 0;
%knn1.DEMacro;
knn1.SetTechnicalArrays
knn1.Run


knn2 = knn('01012001','01012010','d','aapl');
knn2.TechnicalsUsed = [1 1 0 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 1
0 1 0 0 0 1 0 1 0 1 1 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0];
knn2.PercentTestData = .3;
knn2.PercentGainNeededToQualifyAsABuyDay = .000;
knn2.ClassifierStepSize = 377;
knn2.UseMacroEconomicData = 0;
knn2.UseFiltering = 0;
knn2.AddFilteredToIndicators = 0;
knn2.UseRankFeatureReduction = 0;
knn2.UseStatisticalReduction = 0;
%knn2.DEMacro;
knn2.SetTechnicalArrays
knn2.Run

knn3 = knn('01012001','01012010','d','f');
```

```
knn3.TechnicalsUsed = [0 1 0 0 0 1 0 0 0 0 0 1 1 1 1 1 0 1 0 0 1 0 0 1 1 1 1 1 1 1
0 1 1 1 1 1 1 1 1 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0];
knn3.PercentTestData = .3;
knn3.PercentGainNeededToQualifyAsABuyDay = .000;
knn3.ClassifierStepSize = 377;
knn3.UseMacroEconomicData = 0;
knn3.UseFiltering = 0;
knn3.AddFilteredToIndicators = 0;
knn3.UseRankFeatureReduction = 0;
knn3.UseStatisticalReduction = 0;
%knn3.DEMacro;
knn3.SetTechnicalArrays
knn3.Run

knn4 = knn('01012001','01012010','d','ko');
knn4.TechnicalsUsed = [1 0 1 0 1 1 0 0 1 0 1 1 0 1 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 0
1 0 1 1 1 0 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 1 0 0 0];
knn4.PercentTestData = .3;
knn4.PercentGainNeededToQualifyAsABuyDay = .000;
knn4.ClassifierStepSize = 377;
knn4.UseMacroEconomicData = 0;
knn4.UseFiltering = 0;
knn4.AddFilteredToIndicators = 0;
knn4.UseRankFeatureReduction = 0;
knn4.UseStatisticalReduction = 0;
%knn4.DEMacro;
knn4.SetTechnicalArrays
knn4.Run

knn5 = knn('01012001','01012010','d','wmt');
knn5.TechnicalsUsed = [0 1 0 1 1 0 0 1 1 1 0 0 1 0 0 0 0 1 0 1 0 1 0 0 0 0 1 1 1 1
0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0];
knn5.PercentTestData = .3;
knn5.PercentGainNeededToQualifyAsABuyDay = .000;
knn5.ClassifierStepSize = 377;
knn5.UseMacroEconomicData = 0;
knn5.UseFiltering = 0;
knn5.AddFilteredToIndicators = 0;
knn5.UseRankFeatureReduction = 0;
knn5.UseStatisticalReduction = 0;
%knn5.DEMacro;
knn5.SetTechnicalArrays
knn5.Run

knn6 = knn('01012001','01012010','d','bac');
```

```
knn6.TechnicalsUsed = [0 1 0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 0 1 0 0 1 0 0
1 0 0 0 0 1 0 0 1 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0];
knn6.PercentTestData = .3;
knn6.PercentGainNeededToQualifyAsABuyDay = .000;
knn6.ClassifierStepSize = 377;
knn6.UseMacroEconomicData = 0;
knn6.UseFiltering = 0;
knn6.AddFilteredToIndicators = 0;
knn6.UseRankFeatureReduction = 0;
knn6.UseStatisticalReduction = 0;
%knn6.DEMacro;
knn6.SetTechnicalArrays
knn6.Run

knn7 = knn('01012001','01012010','d','dis');
knn7.TechnicalsUsed = [1 0 0 0 0 1 0 1 1 0 0 0 1 0 1 0 0 0 0 0 1 0 0 1 0 0 1 1 0 1
0 0 1 0 0 0 1 1 1 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0];
knn7.PercentTestData = .3;
knn7.PercentGainNeededToQualifyAsABuyDay = .000;
knn7.ClassifierStepSize = 377;
knn7.UseMacroEconomicData = 0;
knn7.UseFiltering = 0;
knn7.AddFilteredToIndicators = 0;
knn7.UseRankFeatureReduction = 0;
knn7.UseStatisticalReduction = 0;
%knn7.DEMacro;
knn7.SetTechnicalArrays
knn7.Run

knn8 = knn('01012001','01012010','d','mcd');
knn8.TechnicalsUsed = [1 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0
0 0 1 0 0 0 0 1 1 0 0 0 1 0 1 0 0 1 0 0 0 1 0 0 0];
knn8.PercentTestData = .3;
knn8.PercentGainNeededToQualifyAsABuyDay = .000;
knn8.ClassifierStepSize = 377;
knn8.UseMacroEconomicData = 0;
knn8.UseFiltering = 0;
knn8.AddFilteredToIndicators = 0;
knn8.UseRankFeatureReduction = 0;
knn8.UseStatisticalReduction = 0;
%knn8.DEMacro;
knn8.SetTechnicalArrays
knn8.Run

knn9 = knn('01012001','01012010','d','mrk');
```

```
knn9.TechnicalsUsed = [0 1 0 0 0 0 0 1 0 1 0 0 0 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 0 0
0 1 1 1 1 1 0 1 0 1 1 1 0 0 0 1 0 1 1 0 0 0 0 0];
knn9.PercentTestData = .3;
knn9.PercentGainNeededToQualifyAsABuyDay = .000;
knn9.ClassifierStepSize = 377;
knn9.UseMacroEconomicData = 0;
knn9.UseFiltering = 0;
knn9.AddFilteredToIndicators = 0;
knn9.UseRankFeatureReduction = 0;
knn9.UseStatisticalReduction = 0;
%knn9.DEMacro;
knn9.SetTechnicalArrays
knn9.Run

knn10 = knn('01012001','01012010','d','dd');
knn10.TechnicalsUsed = [1 1 0 0 1 0 1 0 1 0 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 0 1 1
0 0 1 1 0 0 0 1 1 1 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0];
knn10.PercentTestData = .3;
knn10.PercentGainNeededToQualifyAsABuyDay = .000;
knn10.ClassifierStepSize = 377;
knn10.UseMacroEconomicData = 0;
knn10.UseFiltering = 0;
knn10.AddFilteredToIndicators = 0;
knn10.UseRankFeatureReduction = 0;
knn10.UseStatisticalReduction = 0;
%knn10.DEMacro;
knn10.SetTechnicalArrays
knn10.Run
```

gaOutputFunct.m

```
function [state,options,optchanged ] = gaOutputFunct(options,state,flag)
PopulationTemp = state.Population;
for i = 1:size(state.Population,1)
   for j =1:(size(state.Population,2))
      if randi([1, 100]) < 6  % 5% chance to swap bit.
         if state.Population(i,j) < 0.5
            PopulationTemp(i,j) = 1;
         else
            PopulationTemp(i,j) = 0;
         end
      end
   end
```

```
    if stockGA(PopulationTemp(i,:)) < state.Score(i)
        state.Population(i,:) = PopulationTemp(i,:);
    end
end

optchanged = [];


end
```

getTestingMacroEconomicData.m

```
function[macroEconomicData] = getTestingMacroEconomicData(obj,i)
macroEconomicData = obj.MacroEconomicData(end-obj.TestSetSize-
obj.ClassifierStepSize+i+1:end-obj.TestSetSize+i,:);
end
```

getTestStockData.m

```
function[TestStockData] = getTestStockData(obj,i)
  TestStockData.Date = obj.StockData.Date(1:end-obj.TestSetSize+i);
  TestStockData.Close = obj.StockData.Close(1:end-obj.TestSetSize+i);
  TestStockData.Low = obj.StockData.Low(1:end-obj.TestSetSize+i);
  TestStockData.High = obj.StockData.High(1:end-obj.TestSetSize+i);
  TestStockData.AdjClose = obj.StockData.AdjClose(1:end-obj.TestSetSize+i);
  TestStockData.Volume = obj.StockData.Volume(1:end-obj.TestSetSize+i);
  TestStockData.Open = obj.StockData.Open(1:end-obj.TestSetSize+i);
end
```

****getTrainingMacroEconomicData.m*****

```
function[macroEconomicData] = getTrainingMacroEconomicData(obj,i)
macroEconomicData = obj.MacroEconomicData(i-obj.ClassifierStepSize+1:end-
obj.TestSetSize-obj.ClassifierStepSize+i,:);
end
```

getTrainingStockData.m

```
function[TrainingStockData] = getTrainingStockData(obj,i)
  TrainingStockData.Date = obj.StockData.Date(i-obj.ClassifierStepSize+1:end-
obj.TestSetSize-obj.ClassifierStepSize+i);
```

```
  TrainingStockData.Close = obj.StockData.Close(i-obj.ClassifierStepSize
+1:end-obj.TestSetSize-obj.ClassifierStepSize+i);
  TrainingStockData.Low = obj.StockData.Low(i-obj.ClassifierStepSize+1:end-
obj.TestSetSize-obj.ClassifierStepSize+i);
  TrainingStockData.High = obj.StockData.High(i-obj.ClassifierStepSize+1:end-
obj.TestSetSize-obj.ClassifierStepSize+i);
  TrainingStockData.AdjClose = obj.StockData.AdjClose(i-obj.ClassifierStepSize
+1:end-obj.TestSetSize-obj.ClassifierStepSize+i);
  TrainingStockData.Volume = obj.StockData.Volume(i-obj.ClassifierStepSize
+1:end-obj.TestSetSize-obj.ClassifierStepSize+i);
  TrainingStockData.Open = obj.StockData.Open(i-obj.ClassifierStepSize+1:end-
obj.TestSetSize-obj.ClassifierStepSize+i);
end
```

km.m

```
classdef KM < BaseClassifier
   methods
      function obj = KM(StartDate, EndDate, Frequency, Ticker)
         obj = obj@BaseClassifier(StartDate, EndDate, Frequency, Ticker);
      end

     function result = Classify(obj, i)
         result = [];
         clustersContainer = zeros(1,obj.numberOfClusters);
         totalClustersContainer = zeros(1,obj.numberOfClusters);
         [clusters,centers] =
kmeans(obj.TrainingIndicators,obj.numberOfClusters,'EmptyAction','singleton');
         trainingSetAnswers =
obj.CorrectDailyTradesAdjustedForPercentageGain(i-obj.ClassifierStepSize
+1:end-obj.TestSetSize-obj.ClassifierStepSize+i,:);
         for j=1:length(trainingSetAnswers)
            totalClustersContainer(1,clusters(j)) =
totalClustersContainer(1,clusters(j)) + 1;
            if trainingSetAnswers(j) == 1
               clustersContainer(1,clusters(j)) = clustersContainer(1,clusters(j)) + 1;
            end
         end
         clusterPercentage = clustersContainer./totalClustersContainer;
         clusterLabel = clusterPercentage > .5;

         for j=1:length(obj.TestIndicators(end-obj.ClassifierStepSize+1:end))
```

```
            distance = pdist([obj.TestIndicators(end-obj.ClassifierStepSize
+j,:);centers],'euclidean');
            [~,clusterIndex] =  min(distance(1:obj.numberOfClusters));
            result = [result;clusterLabel(1,clusterIndex)];
         end

      end
   end
   properties
      numberOfClusters = 10;
   end
end
```

****knn.m****

```
classdef KNN < BaseClassifier
   methods
      function obj = KNN(StartDate, EndDate, Frequency, Ticker)
         obj = obj@BaseClassifier(StartDate, EndDate, Frequency, Ticker);
      end

      function result = Classify(obj, i)
         result = knnclassify(obj.TestIndicators(end-obj.ClassifierStepSize+1:end,:),
obj.TrainingIndicators, obj.CorrectDailyTradesAdjustedForPercentageGain(i-
obj.ClassifierStepSize+1:end-obj.TestSetSize-obj.ClassifierStepSize
+i,:),obj.nearestNeighborCount, obj.measurementType, obj.tieBreaker);

      end
   end
   properties
      nearestNeighborCount = 10;
      measurementType = 'euclidean';
      tieBreaker = 'nearest';
   end
end
```

****macroEconImport.m****

```
function[macroEconData] = macroEconImport(fileName,obj)

fileData = importdata(fileName);
```

```
macroEconData = [];
j = 1;
i = 1;
for i=1:length(obj.StockData.Date)-obj.DateOffset*obj.FrequencyOffset
   if j == length(fileData.textdata) && ~isempty(macroEconData)
      j = 1;
      macroEconData = [macroEconData;macroEconData(end)];
   end
   if j == length(fileData.textdata) && isempty(macroEconData)
      j = 1;
      macroEconData = [macroEconData;0];
   end
   while j < length(fileData.textdata)
      if strcmp(obj.StockData.Date(i
+obj.DateOffset*obj.FrequencyOffset),fileData.textdata(j))
         macroEconData = [macroEconData;fileData.data(j)];
         break;
      end
      j = j + 1;
   end
end
```

stockGA.m

```
function [a] = stockGA(technicalsUsed)
   svm1 = DEClustering('01012001','01012006','d','f');
   svm1.PercentTestData = .3;
   svm1.PercentGainNeededToQualifyAsABuyDay = .000;
   svm1.ClassifierStepSize = 377;
   svm1.UseMacroEconomicData = 0;
   svm1.UseFiltering = 0;
   svm1.AddFilteredToIndicators = 0;
   svm1.UseRankFeatureReduction = 0;
   svm1.SetTechnicalArrays
   svm1.TechnicalsUsedInDE = round(technicalsUsed);
    svm1.Run
   a = -svm1.ClassificationPercentage;
   end
```

SVM

```
classdef SVM < BaseClassifier
   methods
      function obj = SVM(StartDate, EndDate, Frequency, Ticker)
          obj = obj@BaseClassifier(StartDate, EndDate, Frequency, Ticker);
      end

     function result = Classify(obj, i)
         options =  statset('maxiter',1000000000000000);
         svmStruct = svmtrain(obj.TrainingIndicators,
obj.CorrectDailyTradesAdjustedForPercentageGain(i-obj.ClassifierStepSize
+1:end-obj.TestSetSize-obj.ClassifierStepSize
+i,:),'Kernel_Function',obj.kernel_function,'options',options);
         %svmStruct = svmtrain(obj.TrainingIndicators,
obj.CorrectDailyTradesAdjustedForPercentageGain(i-obj.ClassifierStepSize
+1:end-obj.TestSetSize-obj.ClassifierStepSize
+i,:),'Kernel_Function',obj.kernel_function,quadprog_opts,options);
         %svmStruct =
svmtrain(obj.TrainingTechnicals,obj.CorrectDailyTrades(i:end-
obj.TestSetSize-1+i,:),'Kernel_Function',obj.kernel_function,'quadprog_opts',optio
ns);
         result = svmclassify(svmStruct,obj.TestIndicators(end-
obj.ClassifierStepSize+1:end,:));

     end
   end
   properties
      kernel_function = 'polynomial';
      RBF_Param = 1;
   end
end
```

waveletFilter.m

```
function[filteredStockData] = waveletFilter(stockData)

[THR,SORH,KEEPAPP] = ddencmp('den','wv',stockData.Close);
filteredStockData.Close = wdencmp('gbl',stockData.Close,'db7',
2,THR,SORH,KEEPAPP);

[THR,SORH,KEEPAPP] = ddencmp('den','wv',stockData.Open);
filteredStockData.Open = wdencmp('gbl',stockData.Open,'db7',
2,THR,SORH,KEEPAPP);

[THR,SORH,KEEPAPP] = ddencmp('den','wv',stockData.High);
```

```
filteredStockData.High = wdencmp('gbl',stockData.High,'db7',
2,THR,SORH,KEEPAPP);

[THR,SORH,KEEPAPP] = ddencmp('den','wv',stockData.Low);
filteredStockData.Low = wdencmp('gbl',stockData.Low,'db7',
2,THR,SORH,KEEPAPP);

[THR,SORH,KEEPAPP] = ddencmp('den','wv',stockData.Volume);
filteredStockData.Volume = wdencmp('gbl',stockData.Volume,'db7',
2,THR,SORH,KEEPAPP);

[THR,SORH,KEEPAPP] = ddencmp('den','wv',stockData.AdjClose);
filteredStockData.AdjClose = wdencmp('gbl',stockData.AdjClose,'db7',
2,THR,SORH,KEEPAPP);

filteredStockData.Date = stockData.Date;
```

****runGA.m****

```
addpath 'ta-lib-0.0.3_ml2007a-mex_w32';
addpath 'drtoolbox';
options = gaoptimset(@ga);
%options.TimeLimit = 60;
options.Generations = 30;
options.CrossoverFcn = @crossovertwopoint;
options.SelectionFcn = @selectionroulette;
options.MigrationFraction = 0;
options.PopulationSize = 30;
options.PlotFcns = @gaplotbestf;
%options.OutputFcns = @gaOutputFunct;
[poop pee] = ga(@stockGA,52,options);
%svm1.TechnicalsUsedNamed
```

Appendix B: Calculation of the Modified Sharpe Ratio

The modified Sharpe ratio is calculated in the ‘evaluate’ function in ‘baseclassifier.m’. The calculation is performed by storing the percentage gain of each day in the test period in an array. Days in which the trading strategy is not in the market, a ‘0’ is recorded in the array. Once the entire test set has been recorded, the average of this array is divided by the standard deviation of the array to calculate the modified Sharpe ratio. The buy-and-hold modified Sharpe ratio is calculated the same way, the only difference being buy-and-hold is in the market every day.

Appendix C: Data Filtering

Stock market data is generally considered noisy data[38]. Minimizing this noise could be an effective tool to simplify and improve the data being classified. One potential way of removing, or at least dampening, the noise in the data is to run a smoothing filter over the data. Wavelet de-noising through coefficient thresholding is a well known way of removing unwanted noise[40][44]. This de-noising can be applied to the waveforms that is our pricing data; high, low, volume, open and close.

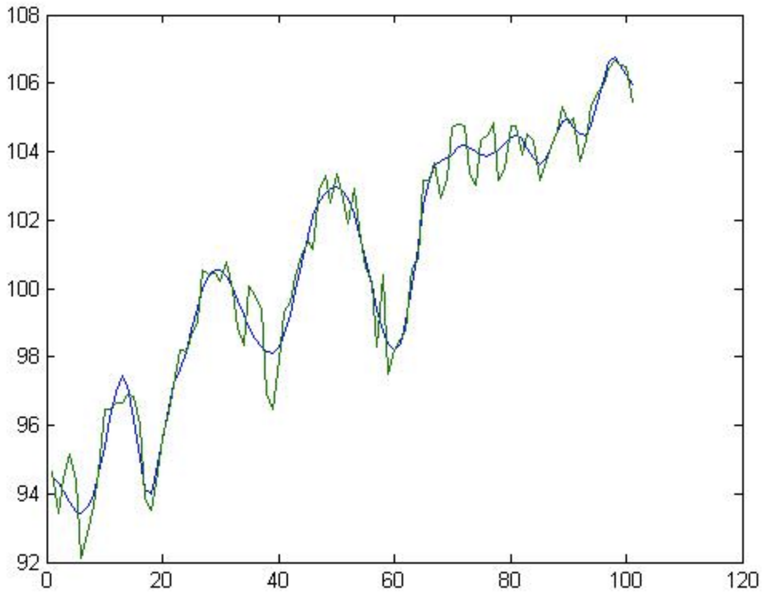

Figure A.1: Filtered SPY closing price data, last 100 days of training set

With the filtered data, the technicals are reconstructed and used as inputs into a KNN classifier. Because wavelet filtering uses the whole waveform when performing its smoothing, the filtered waveforms had to be built up day by day to prevent future data points from being encoded into current data. Performing this function, a progressive filtering loop was employed. The training data plus 1 day of test data was initially filtered. The last data point of the filtered data was the first point of the filtered test data. To get the second point, all the training data plus 2 days of test data was filtered, with the final data point used as the 2nd filtered test data point. This loop is continued until all the test data had been filtered.

The wavelet filtering was done in Matlab using the ‘ddencmp’ and ‘wdencmp’ function out of the Wavelet Toolbox. ‘ddencmp’ returns the default values for denoising a critically-sampled discrete wavelet transform. ‘den’ for denoising, ‘wv’ for wavelet transform and the stock data were used a inputs into ‘ddencmp’. The outputed values of ‘ddencmp’ were used as input values to ‘wdencmp’, the function used to de-noise a signal. Wavelet Daubechies 7, ‘db7’, was the wavelet filter used. Figure A.1 shows the a smoothed closing price signal.

Stock	**Classification %**	**False Positive %**	**profit %**	**Buy And Hold %**	**Sharpe Ratio**	**BuyAndHold Sharpe Ratio**	**# of Trades**
SPY	47.1	50.1	-44.7	-20.8	-0.0607	-0.0076	105
AAPL	48.6	47.6	25.8	121	0.0268	0.0549	115
F	52.9	49.1	324.5	26.9	0.0798	0.0309	92
KO	48.2	48.8	4.5	19.4	0.0114	0.0239	102
WMT	45.4	54.6	-49.8	15.5	-0.0856	0.021	98
BAC	49.8	53.5	-62.8	-66.8	-0.0222	0.0039	91
DIS	51.6	49.6	-2.9	-3.9	0.0076	0.0102	116

Stock	Classification %	False Positive %	profit %	Buy And Hold %	Sharpe Ratio	BuyAndHold Sharpe Ratio	# of Trades
MCD	51.4	46.9	24	40.4	0.03	0.038	105
MRK	49.9	51,6	-18.6	-21.5	-0.0031	-0.0023	94
DD	50.5	47.1	-0.8	-24	0.0063	-0.0024	100
Average	49.54	49.7	19.92	8.62	-0.001	0.01705	101.8

Table A.1: Results for the KNN Classifier Using Filtered Data

The results from table A.1 show wavelet filtered data ran through a KNN classifier. All evaluation metrics see a drop in performance. Classification rates drop below 50% for the first time. Sharpe ratio also fell to its lowest level of all tried techniques, dropping below zero.

This was a first attempt at using filtering techniques on market data to improve classification. Papers such as [44] have shown filtering can be an effective tool in preprocessing data. Even though these results did not show promise, further investigation into filtering techniques is an area of interest for future study.