A Machine Learning Approach to Automated Trading

Boston College Computer Science Senior Thesis

Ning Lu

Advisor: Sergio Alvarez

May 9, 2016

Abstract

The Financial Market is a complex and dynamical system, and is influenced by many factors that are subject to uncertainty. Therefore, it is a difficult task to forecast stock price movements. Machine Learning aims to automatically learn and recognize patterns in large data sets. The self-organizing and self-learning characteristics of Machine Learning algorithms suggest that such algorithms might be effective to tackle the task of predicting stock price fluctuations, and in developing automated trading strategies based on these predictions. Artificial intelligence techniques have been used to forecast market movements, but published approaches do not typically include testing in a real (or simulated) trading environment.

This thesis aims to explore the application of various machine learning algorithms, such as Logistic Regression, Naïve Bayes, Support Vector Machines, and variations of these techniques, to predict the performance of stocks in the S&P 500. Automated trading strategies are then developed based on the best performing models. Finally, these strategies are tested in a simulated market trading environment to obtain out-of-sample validation of predictive performance.

Contents

1. Introduction

As of 2012, a report from Morgan Stanley showed that 84% of all stock trades in the U.S. Stock Market were done by computer algorithm, while only 16% were by human investors [10]. The explosion of algorithmic trading, or automated trading system, has been one of the most prominent trends in the financial industry over recent decade. An automated trading system utilizes advanced quantitative models to generate certain trading decisions, automatically submits orders, and manages those orders after submission. One of the biggest attractions of automated trading is that it take the emotion out of human traders since everything is systemized, also, this technology can reduce the costs of trading and improve liquidity in the Market.

Three steps are needed in order to build a complete automated trading system. First, the system has to have some models generating Stock Market predictions. Second, a trading strategy that takes the model predictions as inputs and outputs trade orders needs to be specified. Last, backtesting is essential to evaluate the trading system's performance on historical market data and thus determine the viability of the system. Current research has been focused largely on market prediction accuracy, but tends to ignore the second and third steps which are very important for building a profitable and reliable trading system.

In this paper, we first focus on forecasting stock price movements using Machine Learning algorithms. We explore variations of basic ML algorithms such as Logistic Regression, Decision Tree, Naive Bayes and Support Vector Machine, and also various ensemble methods to optimize the prediction accuracy. We also look at different ways of building the dataset which algorithms are trained on. Then we customize a trading strategy to take full advantage of the best prediction models. Finally, we use a market simulator to backtest our proposed automated trading system and obtain out-of-sample validation of the system's performance as well.

The rest of this paper is organized as follows: Section 2 gives an overview of automated trading and explains why it has been growing in such significant pace. Section 3 introduces the Machine Learning algorithms we used in this paper and describes how they are optimized. Section 4 explores the "individual approach" to build the prediction model while Section 5 investigates the more sophisticated "sector approach". In Section 6 we show a customized trading strategy and report its out-of-sample performance. Conclusions and future suggested works are discussed in Section 7 at last.

2. Automated Trading

2.1 Overview and Terms

The Stock Market is a marketplace where shares of public companies are traded. A company becomes public when it makes an Initial Public Offering, or commonly known as IPO, which means that investors worldwide are able to buy and trade shares of stock in the company. These shares represent part ownership in the company, and their prices represent what investors believe a piece of the company, or a stock will be worth in the future. Hence, stock prices are determined purely by supply and demand in the market.

Generally speaking, an investor has two choices: if he believes the shares of a company will rise in value some time in the future, he can place a 'buy' order for the stock in the market, and when the order is executed he owns the stock, also known as entering a long position. Then, if more and more people believe the same way, demand for this company's stock goes up and therefore the stock price will increase and investors with long positions enjoy profit. Otherwise, if more people believe the company will worth less in the future, demand drop and the stock price will decrease and these investors will suffer lost. However, in this case, an investor who also decide the shares of a company will depreciate in value, he can place a 'sell' order to short the stock, thus enters a short position. If he already owns this company's stock, the sell order will sell the desired amount of his long position. If he does not own the stock previously, this is called 'short selling', which means that he will borrow someone else's stock and sell them immediately, and when he wants to 'buy cover', he buys back the same amount of shares and return them to the borrower. Therefore, short selling allows investors to profit from a price drop.

The participants in the Stock Market is heterogeneous, meaning there are many different types of investors, each with different return goals and risk-taking levels. Individual investors, institutional investors such as mutual funds, ETFs, and hedge funds, and computer trading algorithms all compete in the same market with the same goal: profit from making the right bet on future stock prices, buy low sell high or the opposite accordingly.

All investors, especially computer trading algorithms, need to specify two things: trading frequency and the "universe" they trade on. Trading frequency refers to how often one makes a trading decision. For human investors their trading frequency may be more flexible, though an investor like Warren Buffett is likely to trade in very low frequency, and a day trader places many

orders everyday to seek intraday profit. For automated trading systems, because of their systematic nature, the trading frequency needs to be specified by their developers before even designing the them. Trading frequency has very wide range: from once a lifetime, which means buy and hold forever, to once every nanosecond for High-Frequency Trading algorithms. As a result, model selection or strategy design can be very different depends on trading frequency.

A universe is the range of stocks an investor chooses to trade on. For example, an investor whose universe is “global market” means that he does not limit his stock-picking to any geographic constraints. A universe can also be the U.S. Stock Market, the S&P 500, or just one sector within the S&P 500. The Standard & Poor’s 500 Index (S&P 500) is one of the most commonly used benchmarks for the overall U.S. Stock Market. It is an index of 500 stocks chosen for market size, liquidity and industry grouping, among other factors. The S&P 500 is meant to reflect the risk/return characteristics of the U.S. large cap universe [16]. SPY is the ticker of the first and most popular ETF in the U.S. whose objective is to duplicate as closely as possible the total return of the S&P 500 [14]. So investors who want to buy the U.S. market can buy the SPY ETF.

Moreover, the S&P 500 can be broken down into different sectors including Consumer Discretionary, Consumer Staples, Energy, Financials, Healthcare, Industrials, Information Technology, Materials and Utilities [36]. The automated trading system proposed in this paper uses the S&P 500 universe, and in particular the Energy, Information Technology and Utilities sectors. The system trades on a daily frequency, and a buy-and-hold strategy on SPY acts as a benchmark to compare our system’s performance.

2.2 Motivation

The biggest advantage of algorithmic trading is that it makes trading more systematic. Human investors are very emotional. One can experience euphoria of having a position go right, or can experience the “fight or flight” reaction when a position is losing money. Not only that this type of reactions shut down the parts of the brain responsible for logic and reasoning [2], but also one can become more greedy or fearful and therefore loses his trading discipline. Having a discipline, or a trading plan, is very important to achieve profit and consistent profit in the Stock Market. However, there is no such plan that can generate positive returns all the time, and when facing these temporarily drawdowns, emotional factors described above can easily destroy the trading discipline. Therefore, because trading plan is followed systematically, an automated trading

system minimizes emotions throughout the trading process, ensures that discipline is maintained through volatile market conditions, and allow us to achieve consistency.

The other advantage of algorithmic trading over human traders is the ability to backtest the strategy. Backtesting refers to applying a trading system to historical data to verify how a system would have performed during the specified time period. It helps the system developers learn how a trading strategy would performed in certain situations in the past, and is likely to perform in the future. It also provides the opportunity to optimize a trading strategy [26], by tweaking model parameters over each iteration. The predictive power of backtesting rests on the important assumption that the statistical properties of the price series are unchanging, so that the trading rule that were profitable in the past will be profitable in the future [5]. However, such assumption is subject to invalidated risk, like changes in the macroeconomic prospect, fundamental of the company or structure of the financial market. Also, dividing historical data into in-sample and out-of-sample sets during backtesting can provide traders a practical and efficient means for evaluating the system [13]. In-sample data can be used to optimize the trading strategy, but it is important to then evaluate the system on clean out-of-sample data to determine its viability and eliminate overfitting.

Moreover, an automated trading system can time the market well, which is extremely hard for human investors. The legendary investor Warren Buffett has a famous quote: “We simply attempt to be fearful when others are greedy and to be greedy only when others are fearful.” This is one form of an attempt to time the market when applying to every day trading. When people are greedy, prices are overvalued thus a trader should sell the stock, and vice versa. But it is extremely hard to consistently identify when people are being greedy or fearful, as Buffett himself also said “our favorite holding period is forever”, which is a warning of do not try to time the market. On the other hand, another form of timing the market is what High-Frequency Trading firms does: find arbitrage opportunities and trade them within microseconds. At this frequency, it is impossible for human to do so. However, automated trading systems can do a better job timing the market, and even do so in a multi-tasking way, which means that a system can time thousands of stocks simultaneously. This is why HFT firms all use computerized trading algorithms, and with consistent good financial time-series forecasting, or sentiment analysis (analysing news articles or tweets), trading systems can also do a decent job at identifying greeds and fears in the market.

2.3 Past Approaches

In the domain of using machine learning techniques to forecast stock market movements, *Financial Time Series Forecasting with Machine Learning Techniques: A Survey* [17] provides a cohesive overview of recent developments: Out of 46 publications this survey covers, 21 of them use Artificial Neural Network based technology. 10 of them explore evolutionary or optimisation techniques and the rest combine different algorithms into hybrid systems. 31 of them use a daily frequency, and 35 of them only use market index as the target. For evaluation methods, majority of them use only forecast errors as the evaluation metric, and only three papers test their models in a real, or simulated trading environment, where one needs to concern with margin requirements, trading cost and risk exposure.

Stickel [31] tries to predict individual analyst's forecast of corporate earning using the change in the mean consensus forecast of other analysts. Huang [12] and Lu's papers mainly focus on establishing Support Vector Machine models, or enhancing them with new training algorithms to forecast major stock index. In particular, Lu [21] implements a regression model of SVMs called Support Vector Regression, with the help of Vapnik's epsilon insensitivity loss function. In *The Performance of Several Combining Forecasts for Stock Index* [34], Wang and Nie implements a SVM-based forecasts model, which combines the Grey model, BP neural networks and SVM, to predict the ShangHai Stock Index. Li [18] provides a different approach: he uses a Naive Bayesian machine learning approach to classify the information content of forward-looking statements. He claims the Bayesian learning algorithm is better than a traditional dictionary-based approach, in terms of the correlations between the predicted tones of forward-looking statements and current earnings.

3. Machine Learning Methods

3.1 Brief Introduction

Alan Turing, one of the greatest computer scientists in human history and father of artificial intelligence, in his paper “Computing Machinery and Intelligence” [33] asks this famous question: “Can machines think?” Until today, we as human still don’t know the answer to this question, but variations of it has been answered. One of them is: Can machines learn? The answer is yes, supervised or unsupervised. And Machine Learning is the field of study which studies how machines learn.

According to Wikipedia, Machine Learning is a subfield of computer science that evolved from the study of pattern recognition and computational learning theory in artificial intelligence [35]. Arthur Samuel, a pioneer in artificial intelligence, defined machine learning as a “field of study that gives computers the ability to learn without being explicitly programmed” [30]. Tom Mitchell, who teaches at Carnegie Mellon University and is a prominent figure in the field of machine learning, defines it as “the study of computer algorithms that improve automatically through experience” [25]. In this book, Professor Mitchell provides a formal definition: A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P is its performance at tasks in T, as measured by P, improves with experience E [25].

This interdisciplinary field has given birth and provided the theoretical foundation to the “Big Data” boom. One application of machine learning is “Data Mining”, which uses enormous historical data sets to improve decisions. Examples include turning medical records into medical knowledge, or turning consumer transaction histories into targeting promotion advertisements. The other application is to develop softwares that we cannot program by hand, such as autonomous driving or speech recognition. Moreover, Deep Learning, which is again an interdisciplinary field of machine learning and cognitive science, has made huge progress on imitating human brain activities. AlphaGo, which is the first computer program to beat a 9-dan human world Go champion, is a product of deep learning techniques.

There are many aspects of algorithmic trading, such as High-Frequency Trading, Statistical Arbitrage and Financial Time Series Forecasting. In this paper, we focus on the forecasting realm, and examine what can machine learning algorithms achieve, not playing the game of chess, but the game of forecasting stock prices.

3.2 Basic Models & Variations

3.2.1 Logistic Regression

Logistic regression was developed by statistician David Cox in his 1958 paper [8]. It is a special case of generalized linear model, and thus analogous to linear regression. The idea is to estimate the conditional probability of a binary response based on one or more predictor variables by using the cumulative logistic distribution.

If we have a binary output variable Y, and we want to model the conditional probability P(Y=1|X=x) as a function of a vector x. Notice that log p(x) is a linear function of x, which has an unbounded range, but a easy modification, the logistic transformation log p/(1-p) is bounded. Therefore, the logistic regression model is

$$log\frac{p(x)}{1-p(x)} = \beta_0 + x \cdot \beta \tag{1}$$

And solving for p, gives a form of the sigmoid function.

$$p(y|x) = \frac{1}{1+e^{-(\beta_0+x\cdot\beta)}} \tag{2}$$

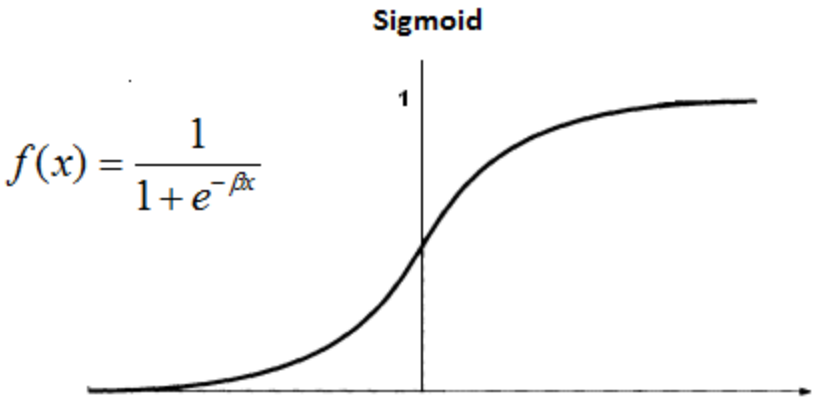

Figure 1: The logistic, or sigmoid, function.

Two variations of logistic regression are implemented for the purpose of this paper. Logistic regression with a ridge penalty is used for the individual approach, and the lasso logistic

regression is used for the sector approach. The applications will be discussed in greater detail later.

Ridge regression minimizes

$$\sum_{i=1}^{N}\left(y_i - \sum_j \beta_j x_{ij}\right)^2 + \lambda \sum_j {\beta_j}^2 \tag{3}$$

Where N is the number of observations, y_i is the response at observation i, x_i is the vector of independent variables at observation i, β_j are the coefficients of the original logistic regression, and lambda is the positive regularization parameter, also known as the ridge penalty. The second term shrinks the coefficients to prevent any one of the them being too large and cause overfitting.

Similarly, the lasso regularization [32] minimizes

$$\sum_{i=1}^{N}\left(y_i - \sum_j \beta_j x_{ij}\right)^2 + \lambda \sum_j |\beta_j| \tag{4}$$

Instead of having the square term, the absolute value term is known as a L_1-norm penalty. By increasing lambda, the lasso penalty also force the coefficients to shrink, but in this case they tend to be truncated at 0. Therefore, the lasso regularization can work as a feature selection process.

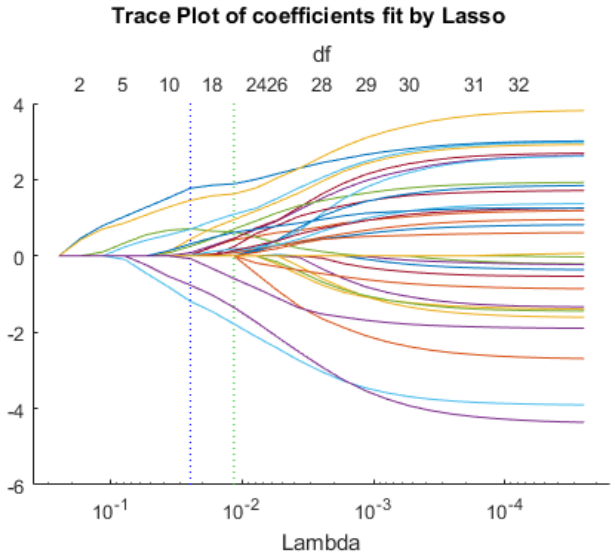

Figure 2: Example of the lasso logistic regression selecting meaningful coefficients

3.2.2 Decision Tree

A decision tree is a hierarchical data structure implementing the divide-and-conquer strategy, whereby the local region is identified in a sequence of recursive splits in a smaller number of steps [1]. Therefore, a decision tree is composed of internal decision nodes and terminal leaves, see figure 3. Given an input, at each node, a test is applied and one of the branches is taken depending on the outcome. This process starts at the root and is repeated recursively until a leaf node is hit, at which point the value written in the leaf constitutes the output.

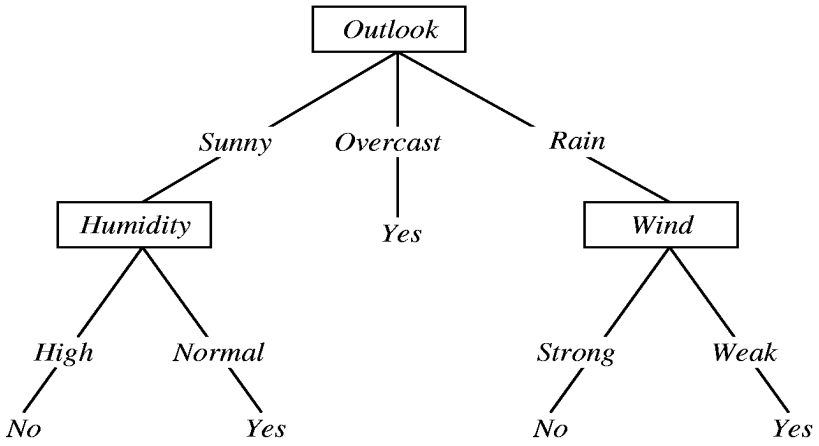

Figure 3: An example of a decision tree determining whether one can play outdoor sports based on weather

In the case of a decision tree for classification, namely, a classification tree [4], the goodness of a split is quantified by an impurity measure. One example of such measure is the Gini index [4], which has the formula:

$$1 - \sum_i p^2(i) \tag{5}$$

Where the sum is over the classes i at the node, and p(i) is the observed fraction of classes with class i that reach the node. A pure node, which is a node with just one class, has Gini index 0, otherwise the Gini index is positive. If a node is not pure, then the instances should be split again to decrease impurity, and there are multiple possible attributes on which we can split. Among all, we look for the split that minimizes impurity after the split because we want to generate the smallest tree.

However, a deep tree with many leaves still tends to overfit and therefore its test accuracy is often far less than its training accuracy. In contrast, a shallow tree can be more robust, that its training accuracy could be more close to that of a test set [3]. The decision tree model we use in this paper is trimmed by minimize leaf size, that is, each leaf has at least a number of observations we set, and more observations per leaf means less leaves and shallower tree.

3.2.3 Naive Bayes

Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with naive independence assumptions between the features. In other words, a Naive Bayes classifier assigns a new observation to the most probable class, assuming the features are conditionally independent given the class value [25].

The Naive Bayes classifier is,

$$V_{NB} = arg \max_{v_j \in V} P(v_j) \prod_i P(a_i|v_j) \tag{6}$$

Where v_{NB} denotes the target value output by the Naive Bayes classifier, $P(v_j)$ is the prior, or class probability and the last term is the sample likelihood, which tess us how likely our sample a_i is if the parameter of the distribution takes the value v_j.

Therefore, Naive Bayes classifies data in two steps: the training step, which uses the training data to estimate the parameters of a probability distribution, the prior probability and the sample likelihood. Then, the prediction step is, for any unseen test data, the method computes the posterior probability of that sample belonging to each class, and this data is classified as the class with the largest posterior probability.

In this paper, we pre-assume that the data comes from a normal distribution, which makes sense for stock returns [5] as they are generally randomly distributed with a mean of zero. Therefore, the Naive Bayes model estimates a separate normal distribution for each class by computing the mean and standard deviation of the training data in that class.

Moreover, we use the ROC Curve analysis to optimize our Naive Bayes classifier. A Receiver Operating Characteristic (ROC) curve shows true positive rate versus false positive rate (equivalently, sensitivity versus 1-specificity) for different thresholds of the classifier output [23].

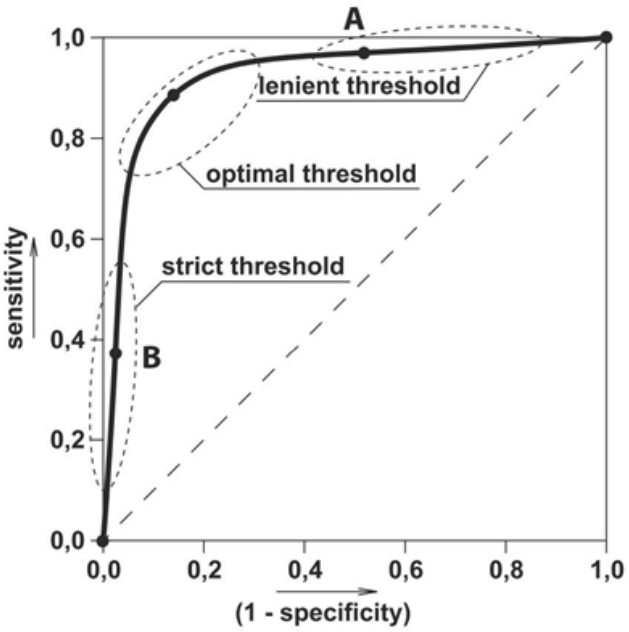

Figure 4: An example of ROC curve and its optimal threshold

ROC curve in general is used to compare performances across different classifiers, by comparing each classifier's AUC (area under curve) of its ROC curve. But in this case, we use the ROC curve to find the threshold that maximizes the classification accuracy for the Naive Bayes classifier. To do this, we find the slope S by:

$$S = \frac{Cost(P|N) - Cost(N|N)}{Cost(N|P) - Cost(P|P)} * \frac{N}{P} \tag{7}$$

Where Cost(N|P) is the cost of misclassifying a positive class as a negative class, vice versa, and P = TP+FN and TN+FP. The optimal operating point is found by moving the straight line with slope S from the upper left corner of the plot down and to the right, until it intersects with the ROC curve, and the optimal threshold is the one associated with this operating point.

3.2.4 Support Vector Machine

Support Vector Machine, which was introduced by Vapnik [7], maps the data into a higher dimensional input space and constructs an optimal separating, often linear hyperplane in this space. For separate classes, the optimal hyperplane maximizes a margin surrounding itself, which creates boundaries for the positive and negative classes, in a binary case. SVM imposes a penalty

on the length of the margin for every observation that is on the wrong side of its class boundary when classes are inseparable.

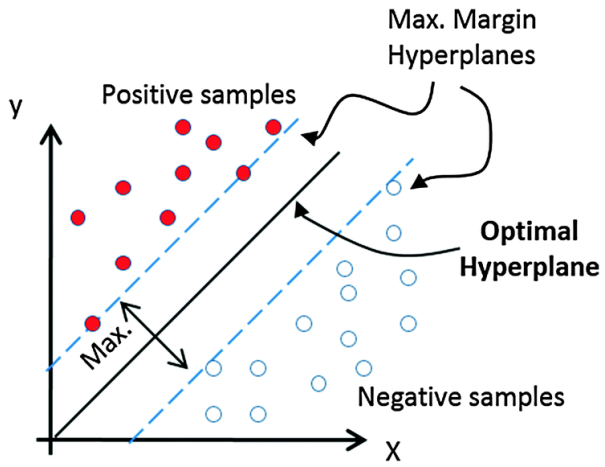

Figure 5: an example of a Support Vector Machine in a separable case

In addition to performing linear classification, SVMs can efficiently perform a nonlinear classification using the "kernel trick", which implicitly maps the inputs into higher-dimensional feature spaces. Without going into details, the SVM model in this paper uses the radial basis kernel, and the Matlab function automatically find an optimal scale value for the kernel function [24].

3.3 Ensemble Methods

Each Machine learning algorithms we discussed above has many parameters to optimize for and also many ways to perform the optimization. The "No Free Lunch" Theorem states that, there is no single learning algorithm that in any domain always induces the most accurate learner [38]. Also, each learning algorithm dictates a certain model that comes with a set of assumptions. This inductive bias leads to error if the assumptions do not hold for future data. In other words, learning is an ill-posed problem, and with finite data, each algorithm converges to a different solution and fails under different circumstances [1]. Therefore, these problems raise the incentive to explore models that composes of multiple base-learners that complement each other by combining them. The idea is that, there may always be another learner that is more accurate, and

by suitably combining multiple base-learners, overall accuracy can be improved. In general, this approach is called the Ensemble Method.

We explore two ideas of ensemble method in this paper. The first one is Voting, which is the simplest way to combine multiple classifiers.

$$y_i = \sum_j w_j d_{ji}, where\ w_j \geq 0, \sum_j w_j = 1 \quad (8)$$

Where w_j is the weight of each base-learner and d_{ji} is the classified output of each learner given a data point. In our case, all base-learners are given equal weight, and the votes are combined linearly by simply adding them up. The final decision is made based on the majority of the votes.

The second ensemble takes the idea of the Random Subspace Method [11], which randomly chooses different input representations from the original data input, and then combines classifiers using different input features. This has the effect that different learners will look at the same problem from different points of view and will be robust. It can also help reduce the curse of dimensionality because each input data set is fewer dimensional.

Other common ensemble methods that are not examined in this paper includes Error-Correcting Output Codes [9] (ECOC), Bagging, and Boosting [27].

4. Prediction Model: Individual Approach

4.1 Methodology

The first method to develop a stock price prediction model is what we called an individual approach, which means that the model uses historical performance of a stock itself to predict its future price movements. The idea is that, looking at how the stock historically moves in N-day windows, there are patterns useful to predict the future price given a new N-day window. Notice that this approach ignores important factors such as general market information and performances of competing companies, but we believe this simplified model can provide a good starting point and a benchmark for comparison with complex models later.

Some details about implementing this model needs to be addressed. First, we want to make this a classification problem instead of a regression problem because we can use probability models like logistic regression to control the “confidence level” of the predictions and moreover, it is not necessary to predict the right price as long as we get the direction right. To see this, let’s assume a regression problem setting and suppose a stock with price $100 and $101 for today and tomorrow. Two models give different predictions of $99 and $110 for tomorrow. The first one is close to the true price but it would suggest a sell and we lose money, while the second one, though has higher error, gets the right direction and we would profit from it. Second, we use daily returns instead of prices when building the dataset. Since most price series are geometric random walks, it violates the econometrics requirement that the expected value of error in a regression needs to be zero.but the returns are usually randomly distributed around a mean of zero [5].

On top of logistic regression, we added a ridge penalty to prevent overfitting. A ridge penalty can be used in logistic regression to improve the parameter estimates and to diminish out-of-sample error, by imposing a restriction on the parameters.

By tweaking lambda we can restrict any one of the estimated coefficients to be too large. To see why this would help our model, suppose we want to predict tomorrow’s return by using returns from last seven days. The coefficient for yesterday’s return would be significantly larger than the rest of them, because yesterday is the most relevant to today. However, one parameter being dominate significantly increase the chance to overfit, that is, the model would give too much weight on yesterday while we want to consider the whole week. By applying the penalty, we can prevent this situation from happening and give more weights to the “further back” parameters.

4.2 Data

The data we use is web scraped from Yahoo Finance, which contains adjusted closing price for all the stocks in S&P 500 from Jan 2,2014 to Feb 1, 2016. We did not go further back because prices more than 2 years old is likely to be irrelevant. Adjusted closing price is the daily closing price, which is the last trading price before market closes at 4 pm, adjusted for dividends and stock splits. It there was a non-trading day for any stock, I put the adjusted close from the previous day.

The following table demonstrate how the dataset looks like by showing an example of two observations.

X	Y
R_{A1} R_{A2} R_{A3} … R_{AN}	Y_{AN+1}
R_{A2} R_{A3} R_{A4} … R_{AN+1}	Y_{AN+2}

Table 1: Example of two observations in the individual approach training set

Where R_{A1} is the return of stock A from day 1 to day 2. Y_{AN+1} is a binary response variable, which equals to 1 if the return of stock A at day N+1 is positive and 0 otherwise. By constructing the dataset this way, we use the returns of the stock itself from day 1 to day N to predict its sign of return at day N+1, so N represents how far we want the model to look back. As a result, a logistic regression model that runs on this dataset will have the characteristics of an autoregression.

For this paper, we pick N=4 based on running an iteration from two to ten and choosing the N with the best in-sample performance. There are five trading days in a week, so it makes sense to say that looking back a week gives a somewhat optimal performance. we would use this value of N throughout this paper. Though it may not be optimal when applying for the models using the sector approach, we believe that it is still a reasonable choice.

4.3 Results

4.3.1 Evaluation Metric

There are two ways that we use in this thesis to evaluate the performance of a prediction model. The first one is what we call the “true rates”, and the second one follows a traditional statistical approach, that is, we conduct a hypothesis test to see whether the true rates are significantly different than random guessing.

For probabilistic model like logistic regression, the model may not make a prediction for every observation because we can define threshold for its confidence level. For example, we may only count those predictions of Y equals 1 with probability higher than 0.6. Therefore, we only care about the “true positive rate”, which calculates out of the times that the model predicts “up” given a threshold, how many times did the stock price actually went up. Similarly, we define “true negative rate” and “true rate”, which is just a weighted average of both positive and negative rates.

Each stock in the S&P 500 has its own trained model and a set of three true rates defined above. Then, to valuate statistical significance, we conduct a t-test with a null hypothesis that the true rates are from a Gaussian distribution with mean equals to 0.5, which represents just random guessing. If we can successfully reject the null hypothesis, we have evidence that our model’s prediction performs better than random guessing at a 5% significance level.

4.3.2 True Rates for Individual Approach

For each stock, the logistic regression model is trained on the first 80 percent and test on the rest 20 percent. For the application of building an automated trading system that would work on future unseen data, we don’t care about how small the training error gets, but we want to see whether the prediction accuracy on the test set can get significantly better than random guessing. The following tables show the average true rates over all the stocks for both logistic regression and ridge logistic regression with different thresholds.

	Logistic Regression			Ridge Logistic Regression		
Threshold	TP	TN	TR	TP	TN	TR
0.5	0.5002(0.0552)	0.5118(0.1051)	0.5067(0.0421)	0.5032(0.0590)	0.5086(0.0560)	0.5050(0.0362)
0.55	0.5192(0.1765)	0.5006(0.2767)	0.5161(0.1447)	0.5680(0.3346)	0.4998(0.3501)	0.5289(0.3141)
0.6	0.5374(0.3255)	0.4898(0.3800)	0.5219(0.3050)	0.5035(0.4171)	0.4584(0.4074)	0.4761(0.3710)

Table 2: True rates of logistic regression and ridge logistic regression

Since we only have two classes, a threshold of 0.5 means that the model is making prediction every time, or more precisely, we are counting it's prediction every time. At this threshold, logistic regression gives an average of 0.5002 true positive rate, with a standard deviation of 0.0552, 0.5118 true negative rate with standard deviation 0.1051 and in aggregate a true rate of 0.5067 with standard deviation 0.0421 over this testing period. Ridge logistic regression had similar performance. None of the true rates for neither model tested significant against random guessing, using the 0.5 threshold.

As we can see from the above table, as we set more restrictive thresholds, the standard deviation gets a lot larger. The reason is that we are counting less predictions, so either those predictions we counted are very right or they are very wrong. However, the average true rates do not move away from 0.5. For all the true rates, only the true positive rate for ridge logistic regression passes the t-test that it is significantly better than random guessing at 5% significance level, with p-value equals 0.0024.

This result suggests that we need to be careful when picking higher thresholds. Although the model is more "sure" when we count its prediction as a trade signal, it may not actually help the overall performance. Also, when we only count a small number of predictions one wrong prediction would have bigger impact.

5. Prediction Model: Sector Approach

5.1 Methodology

The idea of the sector approach is to allow our model to look at a bigger picture, that is, it accounts for the information of all the stocks within the same sector when predicting for a particular stock's price. This has a large advantage comparing to the individual approach. For example, if we are to predict tomorrow's stock price for Apple, the logistic regression model above does not know what has been going on outside of Apple's own stock price. However in reality, price movements of other technology stocks can affects Apple's stock price: if Google's stock rockets amid the news of it entering the smartphone business at full force, the price of Apple's stock most likely would fall. By looking at the returns of the stock of companies within the same sector, though the model cannot identify and analyze those events like this one, but it can take advantage of existed correlations between the stock price of a company and the prices of others within the same industry.

In terms of machine learning algorithms we used to build our model for the sector approach, we tried four basic techniques: lasso logistic regression, decision tree, naive bayes and support vector machine(SVM). Each one of them has been slightly optimized by ways we have discussed in Section 2. On top of it, we applied two ensemble methods: majority vote and a customized random forest.

Majority vote is a simple ensemble method which can take advantage of all four of the basic algorithms. For each observation, lasso, decision tree, naive bayes and SVM each votes for its prediction, and the majority vote model makes a prediction only when at least three of the votes agree. This way we can be more confident on the predictions from the majority vote model.

The second ensemble method we designed, called Random Subset, is based on the concept of a random forest. The idea is to randomly pick different subsets of the stocks in the same sector as predictors. In particular, we choose one basic algorithm, for example naive bayes, as the underlying model. Then, we build one hundred training sets, each randomly picks ten percent of stocks in the sector and let naive bayes train on all of them. In the end we would have one hundred votes, and Random Subset would make a prediction when at least half of the votes agree on a direction. Note that each underlying model can predict up, down or none depends on threshold of our choice, Random Subset also may not give a prediction for every observation.

From these six models, four basic and two ensemble, we will pick the top performing ones to backtest on real out-of-sample data in the next section.

5.2 Data

We choose the utility, energy and information technology sectors to be examined. Out of the 484 stocks from the S&P 500 index that have traded under the same ticker from Jan 2, 2014 to Feb 1, 2016, there are 29 stocks in the utility sector, 39 stocks in the energy sector and 61 stocks in the information technology sector.

We use the same autoregression technique as with the individual approach, but the data set is built in a different way to incorporate the information of other companies within a sector. The following table provides an example of what the first two observations look like:

X	Y
$R_{A1}\ R_{A2}\ R_{A3}\ R_{A4}\ R_{B1}\ R_{B2}\ R_{B3}\ R_{B4}\ \ldots\ R_{Z1}\ R_{Z2}\ R_{Z3}\ R_{Z4}$	Y_{A5}
$R_{A2}\ R_{A3}\ R_{A4}\ R_{A5}\ R_{B2}\ R_{B3}\ R_{B4}\ R_{B5}\ \ldots\ R_{Z2}\ R_{Z3}\ R_{Z4}\ R_{Z5}$	Y_{A6}

Table 3: Example of two observations in the sector approach training set

Where R_{A1} is the return of stock A from day 1 to day 2. Y_{A5} is a binary response variable, which equals to 1 if the return of stock A at day 5 is positive and 0 otherwise. A to Z represent companies in the same sector. We fix N, which is how many days we want to look at at 4 as we discussed in the last section. To predict return of stock A, the model have the returns of all the companies within the same sector going back 4 days. Also, the target stock in Y column can be any company from those appeared in X.

If we were to use all 484 stocks from the S&P 500 as predictors, the data set would have 1938 predictive attributes. For the smallest sector, a data set using only stocks from the utility sector has only 116 attributes, which represents a much smaller and workable dimension.

5.3 Results

5.3.1 True Rates

Similar to the approach in the previous section, we split the dataset into 80 percent training and 20 percent testing. Since training error means nothing when implementing the trading strategy, the true positive rate, true negative rate and overall ture rate from only the testing set are reported. Also, we loop through all stocks within a sector and use each one of them as the target stock to get a list of true rates. This way we can compute means and standard deviations and therefore conduct statistical tests to justify overall performance.

	True Positive	True Negative	True Rate
Utility	0.5595 (0.0481)	0.4507 (0.0641)	0.5235 (0.0332)
Energy	0.4653 (0.0406)	0.5369 (0.0563)	0.5047 (0.0341)
Information Technology	0.5244 (0.0478)	0.5031 (0.0558)	0.5102 (0.0356)

Table 4: Average true rates of Lasso Logistic Regression on different sectors

	True Positive	True Negative	True Rate
Utility	0.5699 (0.0635)	0.4624 (0.0418)	0.5179 (0.0375)
Energy	0.4524 (0.0426)	0.5320 (0.0536)	0.5042 (0.0339)
Information Technology	0.5075 (0.0519)	0.4966 (0.0653)	0.5052 (0.0363)

Table 5: Average true rates of Decision Tree on different sectors

	True Positive	True Negative	True Rate
Utility	0.5949 (0.0623)	0.4957 (0.0577)	0.5495 (0.0458)
Energy	0.4797 (0.0467)	0.5812 (0.0656)	0.5193 (0.0326)
Information Technology	0.5115 (0.0523)	0.5048 (0.0594)	0.5091 (0.0345)

Table 6: Average true rates of Naive Bayes on different sectors

	True Positive	True Negative	True Rate
Utility	0.5804 (0.0545)	0.5081 (0.0741)	0.5562 (0.0416)
Energy	0.4818 (0.0728)	0.6049 (0.0897)	0.5201 (0.0369)
Information Technology	0.5142 (0.0472)	0.5040 (0.1254)	0.5149 (0.0379)

Table 7: Average true rates of Support Vector Machine on different sectors

The average true rates for each of the four basic models are reported for all three selected sectors, while the standard deviations are shown in parenthesis. In general, models under the sector approach outperforms the logistic regression models in the individual approach as we get some true rates which are 1 unit standard deviation away better than 50% random guessing.

Sector-wise, the utility sector has the best predictive performance. Naive Bayes obtains an impressive almost sixty percent average true positive rate for this sector. Also, the average overall true rate of the utility sector outperforms energy and information technology, no matter which of the basic model. One thing worth to notice is that overall true rate seems to be decreasing as the size (number of stocks) of the sector increases. This may be due to the “curse of dimensionality”, which means that higher the dimension, or the number of attributes, of the data set, the chance to overfit increases, so we may get better in-sample performance but out-sample performance falls toward random guessing.

Model-wise, lasso-logistic regression and decision tree shares similar performance both got outperformed by naive bayes and SVM. Between this two better models, SVM in general gives slightly higher standard deviation. As a result, we decide that naive bayes seems to be the best basic model among the four.

Moreover, these models are better predicting upside than downside. True positive rates are in general higher than true negative rates, with the exception of the energy sector. But even with the energy sector, only naive bayes and SVM seems to give downside prediction significantly better than random guessing. This different behavior obtained from the energy sector, we believe is due to the free-falling oil price during the time of testing: this macroeconomic factor boosts downside prediction accuracy simply because there are way more down days than up days. But we are glad to see that naive bayes and SVM still maintains good overall true rates for the energy sector.

To improve downside accuracy, we introduced a two-step decision-making rule. To do this we optimize Naive Bayes by using ROC curve analysis. In detail, we pick the optimal threshold which associated with the optimal operating point of the ROC curve. The two-step process goes

as follows: if the original Naive Bayes predicts up, the final model takes this prediction, but if the original model predicts down, the final model only makes the same prediction if the probability of predicting a negative return given by the original model exceeds the optimal threshold from the ROC curve.

	True Positive	True Negative	True Rate
Utility	0.5949 (0.0623)	0.5023 (0.0623)	0.5513 (0.0456)
Energy	0.4797 (0.0467)	0.5884 (0.0671)	0.5210 (0.0335)
Information Technology	0.5115 (0.0523)	0.5057 (0.0595)	0.5096 (0.0346)

Table 8: Average true rates of Naive Bayes with ROC analysis on different sectors

The true negative rates improve slightly using the two-step rule. Although we believe this improvement may not be significant, it is good to see no signs of overfitting.

Next, we look at the true rates from running the two ensemble methods. Again, Majority Vote takes the votes from SVM, Naive Bayes, Decision Tree and Lasso Logistic. Random Subset incorporates Naive Bayes with ROC analysis as the underlying model. The following table reports the true rates for these two model:

	True Positive	True Negative	True Rate
Utility	0.5807 (0.0610)	0.4921 (0.0836)	0.5573 (0.0443)
Energy	0.4753 (0.0675)	0.6003 (0.0963)	0.5192 (0.0394)
Information Technology	0.5133 (0.0389)	0.5055 (0.1407)	0.5233 (0.0419)

Table 9: Average true rates of Majority Vote on different sectors

	True Positive	True Negative	True Rate
Utility	0.6021 (0.0556)	0.5187 (0.0845)	0.5779 (0.0504)
Energy	0.4574 (0.0905)	0.5871 (0.1053)	0.5108 (0.0526)
Information Technology	0.5348 (0.0860)	0.5317 (0.1311)	0.5382 (0.0679)

Table 10: Average true rates of Random Subset on different sectors

The performance of Majority Vote is very similar to those of SVM and Naive Bayes. We believe that this is because those models' predictions do not differ from each other for most observations, especially the two best performing basic models agree with each other a lot, and

since there are only four votes, the Majority Vote does not provide meaningful improvements for prediction accuracy.

However, we do see gladful improvements from the true rates given by Random Subset as both upside and downside contributes positively. Utility sector has an 58% overall accuracy compares to 55% from Naive Bayes. Information Technology sector also sees a 3% overall accuracy improvements. It suggests that this ensemble method helps to boost performance.

5.3.2 Statistical Testing

We want to carefully examine whether the higher true rates above can withstand statistical tests that prove them better than random guessing. Before conducting the student-t hypothesis test, we need to make sure that we can reasonably assume these true rates come from the normal distribution family. Therefore, we introduce a non-parametric normality testing called the Lilliefors test.

Lilliefors test uses the Kolmoforov-Smirnov statistic for testing whether a set of observations is from a normal population when the mean and variance are not specified but must be estimated from the sample [19].

$$D = \max_{x} |\hat{F}(x) - G(x)| \tag{9}$$

D is the maximum discrepancy between the empirical distribution function (F(x)) and the CDF of the normal distribution with the estimated mean and standard deviation (G(x)), and then uses the Lilliefors distribution table to see whether the maximum discrepancy is large enough to be statistically significant.

	True Positive	True Negative	True Rate
Utility	0	0	1
Energy	0	0	0
Information Technology	0	0	0

Table 11: Lilliefors test results of Naive Bayes

	True Positive	True Negative	True Rate
Utility	0	0	1
Energy	0	0	0
Information Technology	0	1	0

Table 12: Lilliefors test results of SVM

0 means the Lilliefors test fails to reject the null hypothesis that the sample data comes from a normal distribution with unknown parameters, while 1 means the test rejects the rates are normally distributed at the 5% significance level. The tests suggest that almost all of the true rates are normally distributed, so we can conduct t-test on them to see whether they are significantly better than random guessing. For the overall true rates of the utility sector which fail the normality test, a different kind of statistical test is needed to address significance.

	True Positive	True Negative	True Rate
Utility	1 (6.2644e-09)	0 (0.6922)	1 (2.9240e-06)
Energy	1 (0.0097)	1 (2.6383e-09)	1 (7.0202e-04)
Information Technology	0 (0.0907)	0 (0.5323)	1 (0.0429)

Table 13: Student-t test results of Naive Bayes

	True Positive	True Negative	True Rate
Utility	1 (1.0683e-08)	0 (0.4064)	1 (7.7875e-08)
Energy	1 (0.0137)	1 (6.9060e-08)	1 (0.0020)
Information Technology	1 (0.0154)	0 (0.5159)	1 (0.0013)

Table 14: Student-t test results of SVM

The values in the above tables are the results of conducting t-tests with the null hypothesis that the true rates are from a Gaussian distribution with mean equal to 0.5, and the p-values are reported in parenthesis. 1 indicates that the test rejects the null hypothesis at a 5% significance level, which means that the predictive performance is different than random, while 0 means that the observed effect is no better than merely random guessing.

For all three sectors, both Naive Bayes and SVM's overall true rates are significantly better than random guessing at the 5% significance level. This shows that our prediction models can in fact withstand statistical testing, that the predictive accuracies are not results of pure luck.

However, some of the true rates with p-value larger than 0.01 cannot reject the null hypothesis at 1% significance level. Also, the t-tests confirm our previous finding that SVM and Naive Bayes have better upside predictions than downside, since the true negative rates from both utility and information technology sectors fail to be different than random guessing.

Although the overall true rates of the utility sector strongly reject the t-test, it is not trustworthy since earlier we show that they fail to pass the Lilliefors normality test. Therefore, we introduce the Mann-Whitney U test, or equivalent to the Wilcoxon Rank Sum test [37], which is a non-parametric statistical hypothesis test for equality of population medians of two independent samples [22]. This test is an alternative to the paired student's t-test when the population does not come from the normal distribution family.

	True Positive	True Negative	True Rate
Utility	1 (1.5135e-05)	0 (0.6891)	1 (1.3329e-04)

Table 15: Wilcoxon Rank Sum test results of Naive Bayes

	True Positive	True Negative	True Rate
Utility	1 (8.9780e-06)	0 (0.6394)	1 (2.1910e-05)

Table 16: Wilcoxon Rank Sum test results of SVM

The table shows the result of conducting a Wilcoxon rank sum test, with the null hypothesis that the true rates come from the same population as a sample whose median equals to 50 percent. 1 means that we reject the null hypothesis at the 5% significance level that the true rate has a median significantly higher than 50%, while 0 means that we fail to reject otherwise. The p-values are reported in parenthesis. The results confirm that Naive Bayes and SVM have overall true rates significantly better than random guessing.

At last, we examine the statistical significance of the results obtained from Random Subset, using Naive Bayes with ROC analysis as the underlying model. Same procedure is to be followed: first conduct the Lilliefors test for normality, then based on the result decide whether to use t-test or the Wilcoxon rank sum test.

	True Positive	True Negative	True Rate
Utility	0	0	0
Energy	0	0	0
Information Technology	0	0	0

Table 17: Lilliefors test results of Random Subset using Naive Bayes with ROC analysis

	True Positive	True Negative	True Rate
Utility	1 (1.2603e-10)	0 (0.2433)	1 (4.7571e-09)
Energy	1 (0.0055)	1 (7.9446e-06)	0 (0.2069)
Information Technology	1 (0.0025)	0 (0.0638)	1 (4.2767e-05)

Table 18: Student-t test results of Random Subset using Naive Bayes with ROC analysis

Table 17 shows that all of the true rates obtained by Random Subset pass the Lilliefors test, that is, we can conduct the t-test and assume the true rates come from the normal distribution family.

Table 18 shows the result of the t-tests. All the true positive rates for the three sectors are still statistically significant better than random guessing at the 5% significance level. Although Random Subset has better accuracy for downside prediction, the true negative rates for utility and information technology sector still fail the t-test. Therefore, the downside accuracy improvements may not be significant, and we need be aware of this fact when designing our trading strategy. Also, the overall true rate for the energy sector fails to reject the null hypothesis, compares to it given by Naive Bayes or SVM alone which passes the test. We cannot conclude that the ensemble method outperforms the best basic models, so it is necessary to test both in the next section, where we will build our trading strategy and validate it using truly out-of-sample data.

6. Trading Strategy Implementation

6.1 Strategy

The second part of building an automated trading system is to specify a trading strategy that use our machine learning model predictions as input and outputs actual buy/sell orders. It is not enough to simply buy every time the model signals upward motion and sell every time it signals down. A good trading strategy should not only take full advantage of the model's predictions by understanding the essence of the prediction, but also consider the existing positions when generate trade orders.

The models proposed in this paper are designed to make prediction on next-day returns, so the essence is that we can only trust the model predictions one-day ahead and the strategy we designed need to take this in consideration. To illustrate this point, assume that our model has given upward predictions for the past five days and we bought 1 unit of 100 shares of the target stock each day. The position in this stock we have at the end of the fifth day is 500 shares, but now the model gives us a sell signal for day six. A question the strategy needs to answer is how much we want to sell given this sell signal. If we only sell 1 unit, we will still have 400 shares tomorrow when we know, based on our model, that we will lose money on this long position. In another word, the upward predictions have carried over for more than 1 day until we sell all the shares. The example strategy does not take full advantage of the model predictions since they do not tell us what will happen more than one day ahead.

```
If model predicts up
        If at long position or no position
                Buy one unit (100 shares)
        If at short position
                Buy back all short position and buy one unit

If model predicts down
        If at short position or no position
                Short one unit (Buy -100 shares)
        If at long position
                Sell all long position and short one unit
```

Figure 6: Detail illustration of our customized trading strategy

The above figure demonstrates the strategy we designed specifically for the models proposed in this paper. This strategy takes in consideration the existing portfolio positions before submitting a trade order. Let's say the model gives a buy signal, which means that it suggest the stock price is going up tomorrow. If we already hold a long position, which cumulates from previous buy signals, or have no position at all, it makes sense to buy one more unit or enter a long position. But if we are shorting the stock, we know that the short position will suffer lost tomorrow when price goes up, so we need to exist all short position to avoid this lost, and, we want to go aggressive and even start entering a new long position to profit from the price increase. This aggressiveness depends highly on our model's accuracy and as a result, the returns generated by this strategy are meant to have higher volatilities.

In Section 4 and 5, we train the Machine Learning prediction models on the first 70 percent of the data and calculate the true rates on the rest 30 percent. In real world situation, however, new data point comes in every day so we can also take advantage of this fact. Instead of building just one training data set, we build it dynamically, that is, our trading system update the train set everyday by adding the new entry.

Moreover, for each sector, we pick the top 10 performers from the validation period to trade on. The table below shows the tickers for these best performing stocks. Only using the top 10 stocks may cause overfitting thus hurt the out-sample results. In real situation, however, we can manage the cash proportion of our portfolio easier compare to trading every stock in the sector. Of course, this list of top performers should be updated constantly, like once a month.

Naive Bayes with ROC	
Utility	AEP, D, CMS, SO, PNW, ES, PCG, DTE, XEC, WEC
Energy	VLO, PXD, MPC, NBL, PSX, TSO, DVN, RIG, OKE, OXY
IT	FISV, INTU, EMC, CA, ADP, AMZN, MU, TXN, WU, HPQ
Random Subset	
Utility	XEL, D, CMS, WEC, PNW, AEP, DTE, NI, PEG, ES
Energy	ESV, PXD, EOG, MPC, XOM, AES, OKE, DO, RIG, DVN
IT	NVDA, CA, MU, FISV, EBAY, AMHT, ADP, EMC, PAYX, KLAC

Table 19: Top 10 performers of each sector for Naive Bayes and Random Subset

6.2 Market Simulator

We develop a market simulator to provide an environment where the trading strategies can be tested. The simulator uses real-work stock prices, and inherits a simplified version of real market mechanism. It allows strategies to place buy/sell orders, executes the orders at market price including trading cost, and evaluates portfolio value at daily frequency.

A key advantage of having a market simulator is that we can backtest a trading strategy to see how it would have performed in the past. Backtesting is a key difference between a traditional investment management process and a quantitative investment process [6]. Not only we can derive complete detail of the performance of a strategy such as daily returns, position changes and cash changes, but also backtesting allows s to experiment with variations to the original strategy to refine and improve it.

However, this simplified version of market is subject to many limits that may cause the test results to be less significant or unrealistic. The biggest concern is that the prices which orders are executed on are different than they would be in actual trading. The real Market uses a bid/ask spread mechanism, where buy order is fulfilled at the lowest possible ‘ask’ price and sell order is fulfilled at the highest possible ‘bid’ price. Without going into more details, usually the market price, which the simulator uses, is the average of the lowest ask price and highest bid price. Also, our prediction model needs to wait for the closing price of today in order to make the prediction for tomorrow. This means that our strategy can only place orders after the Market is closed for the day so that the orders in real life would be executed at the opening price on the next day, but our simulator assume that the order is executed at the closing price today. In the two situations described above, usually the difference between market price and bid/ask price is tiny, and except for some special events the difference between today’s closing price and tomorrow’s opening price is also small, but keep in mind that these differences cumulated can potentially hamper our backtesting result.

Backtesting is done in two parts. We train and optimize the trading strategies on data between Jan 2, 2014 to Jan 31, 2016. Then we run truly out-of-sample tests from Feb 1, 2016 to May 1, 2016 to obtain validation of our strategies’ historical performances

6.3 Out-of-Sample Live Simulation

6.3.1 Sharpe Ratio

In order to measure a strategy's performance, looking at just the historical returns the strategy generates is not enough. We need a performance measure that considers the consistency of the returns generated by a strategy. In William F. Sharpe's 1994 original paper [28], He introduced the Sharpe Ratio which is a measure for calculating risk-adjusted return. The Sharpe ratio equals mean of portfolio return minus risk-free rate and divided by standard deviation of portfolio return.

The Sharpe Ratio is the average return earned in excess of the risk-free rate per unit of volatility [15], where the risk-free rate usually is the federal interest rate in the U.S. It has become a industry standard for measuring strategy performances and it is also great at comparing performances across different strategies. In practise, people annualize the Sharpe ratio. In our case where the Sharpe ratio is calculated at daily frequency, we annualize it by multiplying by square root of 252, since there are 252 trading days in a year.

As a rule of thumb [6], a strategy that has an annualized Sharpe ratio of at least 1 is suitable as a stand-alone strategy, that is, the returns generated by it have statistical significance. A strategy that can achieve profitability almost every month typically has its Sharpe ratio greater than 2, and a strategy that is profitable almost every day usually has Sharpe ratio greater 3. For the purpose of this thesis, we aim for Sharpe ratio above 1.

Notice that although the Sharpe ratio offers a standard and convenience solution for measuring strategy performance, it has certain limits worthy for us to bear in mind. William Sharpe in a reprinted version of his paper in 1996 [29] acknowledges that "any measure that attempts to summarize even an unbiased prediction of performance with a single number requires a substantial set of assumptions for justification". In particular, as Andrew Lo points out [20], the calculation to annualize the Sharpe ratio, especially for the standard deviation part, relies on the assumption that the daily returns are i.i.d and serially uncorrelated, so the result is subject to estimation errors.

6.3.2 Individual Approach

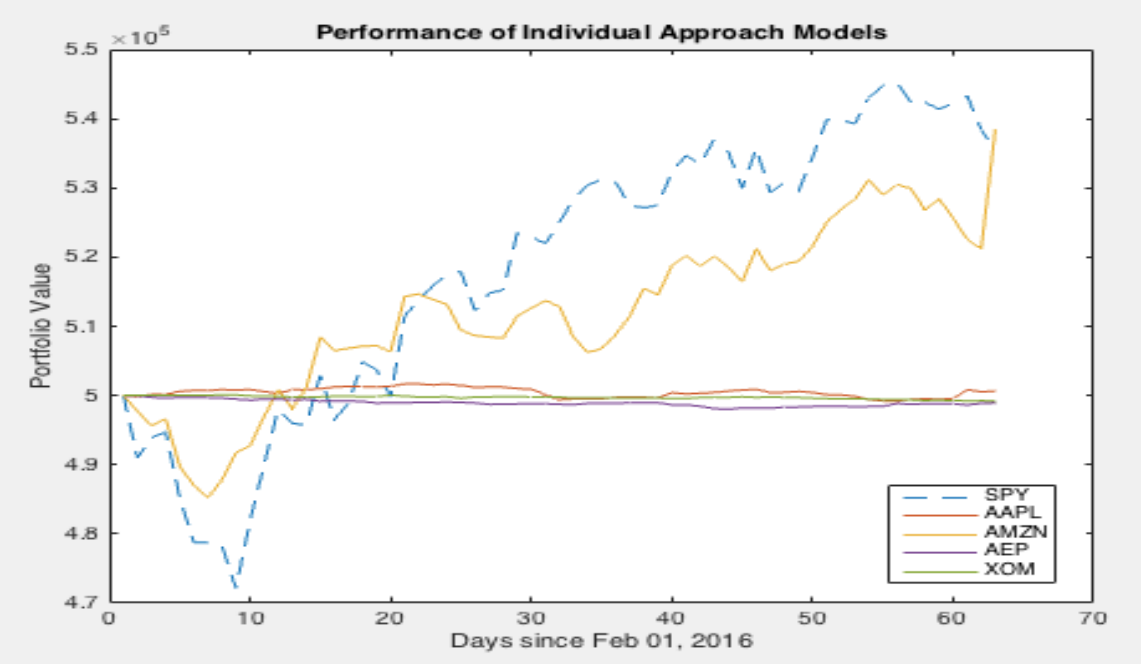

Figure 7: Three months out-of-sample simulation results for individual approach models

	SPY	AAPL	AMZN	XOM	AEP
Sharpe	2.0738	0.5343	2.8183	-2.8736	-2.0579

Table 20: Sharpe ratio of ridge logistic regression trading on different stocks

Figure 7 shows the performances of our trading system trading on Apple, Amazon, Exxon Mobil and American Electric Power, using the ridge logistic regression with 0.55 threshold. The system trades from February 1st to April 29th, which represents a two-month out-of-sample test. These target stocks are chosen as “representatives” because there are the largest market-cap companies in their corresponding sectors.

The dotted line is the performance of the S&P 500 in the same period, or strictly speaking is the performance of a buy-and-hold strategy trading on SPY. The SPY index achieves a 2.07 sharpe ratio during this two month, but only the trading system which trades on Amazon beats the Market with a 2.82 sharpe ratio. This system outperforms Market for the first 20 days, but keeps underperforming in the next 50 days only to catch up at the very end.

Other Amazon, systems that trades on Apple, XOM and AEP all underperform the Market in this out-sample test. Only the Apple system achieves positive sharpe ratio but an insignificant 0.53, while systems on XOM and AEP have negative -2 sharpe ratios, that is, these two systems consistently lose money every month. However, the good news is that even with such large negative sharpe ratios, both systems do not lose a lot portfolio value, which shows that our strategy does have certain degree of downside protection.

6.3.3 Sector Approach

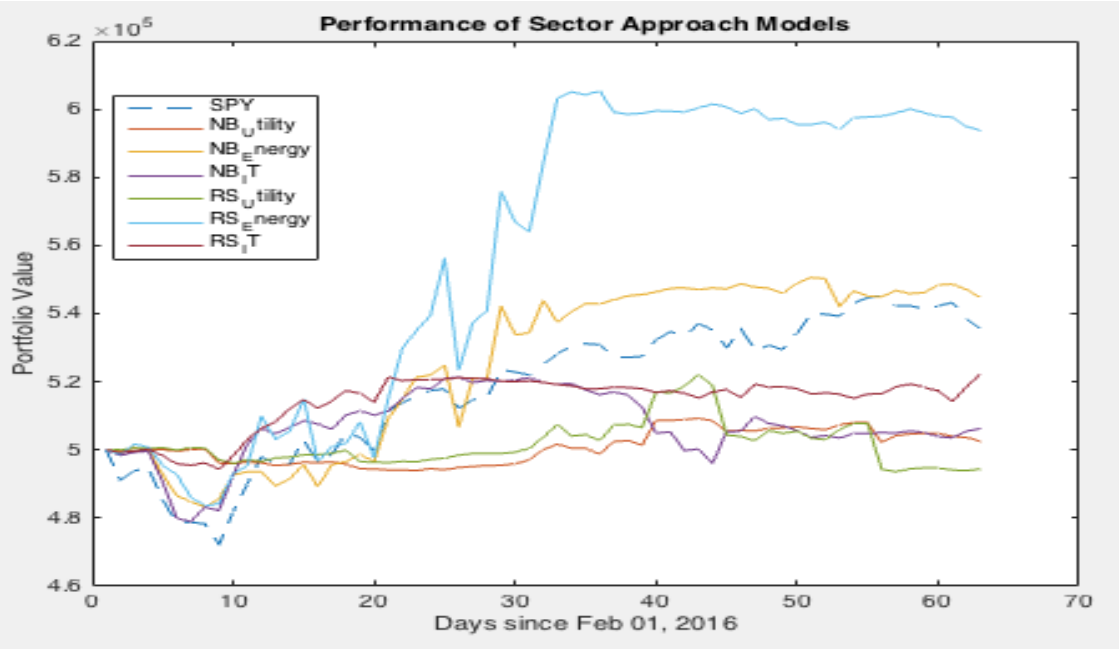

Figure 8: Three months out-of-sample simulation results for sector approach models

		Naive Bayes with ROC			Random Subset		
	SPY	Utility	Energy	IT	Utility	Energy	IT
Sharpe	2.0738	0.4344	2.1362	0.5069	-0.4024	2.6898	2.8393

Table 21: Sharpe ratio of sector approach trading systems trading on different sectors

Here, ‘Naive Bayes’ refers to a trading system which use Naive Bayes with ROC optimization as the underlying model, and similarly with ‘Random Subset’, which refers to a system which employs the Random Subset technique with ROC-optimized Naive Bayes as the underlying model.

Naive Bayes on utility, energy and information technology sectors all achieve positive returns during the testing period. In particular Naive Bayes trading on the top 10 stocks in the energy sector has a promising 2.14 sharpe ratio, though the other two do not have sharpe ratios above 1. The Random Subset systems trading on energy and IT sector both have strong out-sample performances, with promising 2.69 and 2.84 sharpe ratio respectively. On both sectors, Random Subset outperforms Naive Bayes.

One interesting thing to notice is that, in section 6, the ML models run on utility sector have higher true rates than those on energy and IT, but the overall true rates for the utility sector do not pass the normality test. We would have expected that the systems which trade on utility sector to have the best portfolio performance, however, they are the worst performers during the testing period, and Random Subset on utility even has a negative sharpe ratio. This might because of the true rates do not come from a normal distribution, therefore skewed and subject to bigger risk of overfitting.

Moreover, five out of the six trading systems achieve positive sharpe ratio, only two trading systems, Naive Bayes and Random Subset trading on energy sector, outperform the S&P 500 index during the testing period. But although is common to compare the performance trading strategies with the SPY, long-short trading strategies like our trading systems are dollar neutral,!!! and therefore more appropriate to compare with the risk-free interest rate. In this case then, since the risk-free interest rate is close to zero under current economic policy, all five systems with positive sharpe ratio generate higher return than the risk-free rate.

7. Conclusion

In this thesis, we built an automated trading system based on Machine Learning algorithms. Based on historical price information, the machine learning models forecast next-day returns of the target stock. A customized trading strategy then takes the model prediction as input and generate actual buy/sell orders and send them to a market simulator where the orders are executed. After training on available data from January 2, 2014 to January 31, 2016, our system is back-tested on out-of-sample data from February 1, 2016 to May 1, 2016.

We found that only looking at a company's past stock price itself was not sufficient enough to predict its future returns. A better way to do so was to look at the entire sector which the target company was part of, and used historical price information of all companies within the sector to predict the target's next-day return. In a three month in-sample validation period, some models achieved almost 60 percent accuracy with statistical significance. Also, we found that simple variations of machine learning basic algorithm, like the ROC curve optimization, or Ensemble Methods which combine basic algorithms can improve accuracy and provide significant results.

A customized trading strategy that utilizes our model predictions showed signs of successfully timing the market. Our trading systems achieved promising sharpe ratios with nearly 20 percent return, after transaction cost, during the out-of-sample testing period, and was able to beat the risk-free interest rate and some even outperformed the S&P 500 index. The backtesting on unseen data validated that our automated trading system was able to consistently generate positive returns.

For future works, there are so many things worth to try. This thesis only focused on technical analysis, so an obvious thing to do is to incorporate some fundamental analysis. We tried to add daily trading volume to our data set, but it doubled the dimension and did not improve our model predictions, but with careful feature selection, we believe there should be information gain by adding trading volumes. Variables about company fundamentals such as revenues and earnings, and about macroeconomic issues such as interest rates, exchange rates and unemployment reports should also help predicting stock prices. Moreover, we remain curious how more sophisticated machine learning algorithms or deep learning techniques can be meaningfully implemented in the application of financial forecasting, with caution for overfitting drawbacks.

After all, automated trading should not be just about algorithms, programing and mathematics [5]: an awareness of fundamental market and macroeconomic issues is also needed

to help us decide whether the backtest is predictive and the automated trading system will continue to be predictive.

Acknowledgment

First of all, I want to sincerely thank my parents. Without their unconditional support, I would not have the opportunity to receive high quality education in the United States and complete a thesis at the amazing Boston College. Thank you Professor Alvarez, my thesis advisor, and the entire Computer Science department for providing me resources and giving me very helpful advices not only for this thesis work, but throughout my four years as a Computer Science major. Thank you Professor Lowrie for being an amazing academic advisor and congratulation on your retirement. I want to also thank Professor Law from the economic department, his new course Intro to Computational Investing really gave me a good start on this project, and I borrowed his idea of building a market simulator to build mine. Yang Zhou, my partner in this class has also contributed to the development of the logistic regression in this paper. Thank you Boyao Sun for always counter arguing me and helped me with the idea of the random subset method. Thank you Cydgg, Emma and Congcong for always reminding me to be a kind person. Also, thank you Professor Sadka from CSOM who also gave me advice on this thesis, and I want to thank Carnegie Mellon University for giving me the opportunity to lean more about the field of Quantitative Finance in the MSCF program. Thank you Alan Turing, and everyone who has contributed to the field of Machine Learning and continue to push the advancement of modern technology. Last and specially, I want to thank Summer Li for your immense support and understanding. I sacrificed a lot of time that belongs to you for this thesis, and I want to repay you for the rest of my life.

References

[1] Alpaydin, Ethem. *Introduction to Machine Learning*. Cambridge, MA: MIT, 2010. Print.

[2] Beder, Tanya S., and Cara M. Marshall. *Financial Engineering: The Evolution of a Profession*. Hoboken, NJ: John Wiley & Sons, 2011. Print.

[3] Bradford, Jeffrey P et al. "Pruning decision trees with misclassification costs." *Machine Learning: ECML-98* (1998): 131-136.

[4] Breiman, Leo et al. *Classification and regression trees*. CRC press, 1984.

[5] Chan, Ernest P. *Algorithmic Trading: Winning Strategies and Their Rationale (Wiley Trading Series)*. John Wiley & Sons, 2013. Print.

[6] Chan, Ernest P. *Quantitative Trading: How to Build Your Own Algorithmic Trading Business*. Hoboken, NJ: John Wiley & Sons, 2009. Print.

[7] Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." *Machine learning* 20.3 (1995): 273-297.

[8] Cox, David R. "The regression analysis of binary sequences." *Journal of the Royal Statistical Society. Series B (Methodological)* (1958): 215-242.

[9] Dietterich, Thomas G. and Ghulum Bakiri. "Solving multiclass learning problems via error-correcting output codes." *Journal of artificial intelligence research* (1995): 263-286.

[10] Financial Times. "'Real' investors eclipsed by fast trading" 2012. http://www.ft.com/cms/s/0/da5d033c-8e1c-11e1-bf8f-00144feab49a.html

[11] Ho, Tin Kam. "The random subspace method for constructing decision forests." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 20.8 (1998): 832-844.

[12] Huang, Wei, Yoshiteru Nakamori, and Shou-Yang Wang. "Forecasting stock market movement direction with support vector machine." *Computers & Operations Research* 32.10 (2005): 2513-2522.

[13] Investopedia. "Backtesting And Forward Testing: The Importance Of Correlation" 2010. http://www.investopedia.com/articles/trading/10/backtesting-walkforward-important-correlation.asp

[14] Investopedia. "Exchange-Traded Funds: SPDR S&P 500 ETF" 2008. http://www.investopedia.com/university/exchange-traded-fund/etf3.asp

[15] Investopedia. "Sharpe Ratio Definition" 2013 http://www.investopedia.com/terms/s/sharperatio.asp

[16] Investopedia. "Standard & Poor's 500 Index (S&P 500) Definition" 2016. http://www.investopedia.com/terms/s/sp500.asp

[17] Krollner, Bjoern, Bruce Vanstone, and Gavin Finnie. "Financial time series forecasting with machine learning techniques: A survey." (2010).

[18] Li, Feng. "The information content of forward-looking statements in corporate filings—A naïve Bayesian machine learning approach." *Journal of Accounting Research* 48.5 (2010): 1049-1102.

[19] Lilliefors, Hubert W. "On the Kolmogorov-Smirnov test for normality with mean and variance unknown." *Journal of the American Statistical Association* 62.318 (1967): 399-402.

[20] Lo, Andrew W. "The statistics of Sharpe ratios." *Financial Analysts Journal* 58.4 (2002): 36-52.

[21] Lu, Chi-Jie, Tian-Shyug Lee, and Chih-Chou Chiu. "Financial time series forecasting using independent component analysis and support vector regression." *Decision Support Systems* 47.2 (2009): 115-125.

[22] Mann, Henry B, and Donald R Whitney. "On a test of whether one of two random variables is stochastically larger than the other." *The annals of mathematical statistics* (1947): 50-60.

[23] MathWorks. "Performance Curves - MATLAB & Simulink." 2012. http://www.mathworks.com/help/stats/performance-curves.html

[24] MathWorks. "Train binary support vector machine classifier - MATLAB fitcsvm." 2014. http://www.mathworks.com/help/stats/fitcsvm.html

[25] Mitchell, Tom M. *Machine Learning*. New York: McGraw-Hill, 1997. Print.

[26] Ni, Jiarui, and Chengqi Zhang. "An efficient implementation of the backtesting of trading strategies." *Parallel and Distributed Processing and Applications* (2005): 126-131.

[27] Schapire, Robert E. "The strength of weak learnability." *Machine learning* 5.2 (1990): 197-227.

[28] Sharpe, William F. "The sharpe ratio." *The journal of portfolio management* 21.1 (1994): 49-58.
[29] Sharpe, WF. "The Sharpe Ratio - Stanford University." 2015. https://web.stanford.edu/~wfsharpe/art/sr/sr.htm
[30] Simon, Phil. *Too Big to Ignore: The Business Case for Big Data*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2013.
[31] Stickel, Scott E. "Predicting individual analyst earnings forecasts." *Journal of Accounting Research* (1990): 409-417.
[32] Tibshirani, Robert. "Regression shrinkage and selection via the lasso." *Journal of the Royal Statistical Society. Series B (Methodological)* (1996): 267-288.
[33] Turing, Alan M. "Computing machinery and intelligence." *Mind* 59.236 (1950): 433-460.
[34] Wang, Weihong, and Shuangshuang Nie. "The performance of several combining forecasts for stock index." *Future Information Technology and Management Engineering, 2008. FITME'08. International Seminar on*. IEEE, 2008.
[35] Wikipedia. "Machine learning" 2011 https://en.wikipedia.org/wiki/Machine_learning
[36] Wikipedia. "List of S&P 500 companies" 2011. https://en.wikipedia.org/wiki/List_of_S%26P_500_companies
[37] Wilcoxon, Frank. "Individual comparisons by ranking methods." *Biometrics bulletin* 1.6 (1945): 80-83.
[38] Wolpert, David H. "The lack of a priori distinctions between learning algorithms." *Neural computation* 8.7 (1996): 1341-1390.