Thesis

USING MACHINE LEARNING AND GOOGLE SEARCHES TO PREDICT STOCK MARKET CHANGES

Submitted by

Jarred Glaser

Department of Economics

In partial fulfillment of the requirements

For the Degree of Master of Arts in Economics

University of Wisconsin – Milwaukee

Milwaukee, Wisconsin

Spring 2018

Abstract:

One of the most difficult entities to forecast is the stock market. Traditionally, stock market forecasting has been governed by the random walk theory and efficient market hypothesis, leading many researchers to believe that the stock market cannot be accurately predicted. However recent literature in forecasting and machine learning is beginning to challenge these beliefs. In this paper, I build on this research by using two machine learning models, SVM and Random Forest, with a set of technical indicators as well as Google search volume data as features to predict directional stock market closing price changes for several companies. I find prediction accuracy up to 89.9% and significant accuracy improvements provided by the Google search volume features.

1. Introduction

One of the most long-standing challenges of time series forecasting that exists today is the forecasting of stock prices. It makes sense that many statisticians and econometricians associate a large part of time series analysis with stock price forecasting. Not only is the ability to forecast stock prices extremely profitable to investors, but it remains one of the most challenging economic entities to predict.

Traditional studies of the stock market point to its characteristics being that of a random walk process. One of the most influential books in stock market prediction, by Princeton economist Burton Malkiel (1973), popularized the theory that stock prices take the form of a random walk model. He claimed that stock prices follow a "random walk" process, and therefore, cannot be predicted by past information.

Other research at the time encouraged this theory of an "unpredictable market". For instance, Fama (1965) explains the Efficient Market Hypothesis (EMH) in relation to stock markets. The EMH says that the vast availability of stock price information leads to a situation where many rational, profit maximizers, are actively competing to predict stock prices. Under these circumstances, the stock market acts as an efficient market, where, at any given point in time, current prices always reflect the actual, or intrinsic, value of a given stock. Fama points out that while prices may not always represent the exact intrinsic value, the forces of the many competing actors will tend to cause the price to hover randomly around its true intrinsic value. According to this theory, it is impossible for any one actor to "beat" the market. This is because every profit maximizing investor is attempting to make predictions with the same information that is freely available.

Traditional approaches to stock market prediction have, for the most part, been based on these theories. However, an increasing number of studies are beginning to prove that the stock market can in fact be more accurately predicted (Weng, 2017; Choudhry and Garg 2008; Preis, Moat, & Stanley 2013). These studies take advantage of new, advanced statistical techniques that borrow methods from other fields such as computer science and machine learning. With advancements in computational efficiency and artificial intelligence, as well as the availability of new data based on online interactions, the use of stock price forecasting methods that challenge the efficient market hypothesis and random walk theory have risen dramatically.

This paper attempts to build upon this research using some of these newer machine learning methods to create a better stock price forecast that can beat more traditional models. In this paper, I attempt to predict the direction of a change in a specific company's stock price. This transforms the prediction problem from a regression into classification (the stock price went up: 1, the stock price did not go up: 0). I use two different machine learning models, Support Vector Machines (SVM) and Random Forests (RF), for the predictive analysis, and examine which model performs best using the given parameters.

To improve the predictability of the model I use Google search data related to specific stocks as a parameter. The idea to use this information as a model parameter builds on previous research from Weng (2017) and Pries et al. (2013). These papers show that the use of online data sources as forecasting model parameters vastly improves the prediction accuracy of machine learning models. In addition to Google search data, I also use a few popular financial technical indicators as parameters. Research shows that technical indicators are a very important input for Machine Learning models when predicting stock prices (Xi, 2014; Chavan and Patil, 2013).

After collecting the data and calculating the technical indicators, I run recursive feature elimination (RFE) to pick only parameters that are most important to the model. To compare the strength of the models, and to avoid overfitting, I use a type of cross validation for time series data called recursive forecasting. Finally, model performance is evaluated using several different model evaluation parameters. Additionally, all of the R and Python code used for this paper can be found here https://github.com/jdglaser/mathesis.

2. Literature Review

Traditionally, stock market prediction has relied on the idea that stock markets follow a random walk model, and therefore cannot be accurately predicted. However, new research shows that taking advantage of new machine learning methods and data collection techniques can lead to an improvement in stock market forecasting. This research has been made possible by recent advancements in computing power in the past decade.

2.1 Prediction with Technical Indicators

Much of the recent research has focused on using machine learning models with technical indicators as inputs to predict stock market changes. Technical indicators are simply different equations that use historical prices as inputs to try and predict future trends and price movements in stocks (Investopedia 2018). Using these indicators to predict stock prices is nothing new (Vaiz and Ramaswami, 2016), however using them as inputs for machine learning models has shown to be worthwhile in some recent literature.

Madge (2015) finds prediction accuracy improvements over longer forecast horizons with a support vector machine model using technical indicators as inputs. However, these improvements diminish as the forecast horizon shortens. Choudhry and Garg (2008) find stock price forecast prediction accuracy of up to 61% using a hybrid Genetic Algorithm – Support Vector Machine model with technical indicators as inputs. They find that this hybrid model shows significantly better results than just a standalone SVM model. Di (2014) uses a radial basis function SVM algorithm to predict stock price trends for Apple, Amazon, and Microsoft, achieving prediction accuracy up to 77% for 3-10 day forecast horizons. While SVM's have been proven to be very effective in predicting stock price changes, Khaidem, Saha, and Dey (2016) show that using a tree based, random forest model for predicting 30, 60, and 90-day stock price directional changes can lead to prediction accuracy of as much as 96%.

2.2 Prediction with other Variables

While technical indicators can be very useful tools in predicting stock prices, recent research reveals that other variables can be used alongside technical indicators as inputs for machine learning models. These variables often come from a wide range of newly available information from online resources. This internet-based information can be a useful predictor for stock price changes as more of society transfers its daily activity to online interactions.

Preis, Moat, and Stanley (2013) show that by using Google search query volumes from Google Trends, trading behavior can be quantified to reveal early warning signs in stock market movements. Weng (2016) finds that Google news results and Wikipedia page hits for Apple's stock can be used as inputs alongside technical indicators to achieve stock price prediction accuracy up to 85%. Varian (2014) discusses several machine learning methods and how they can be used with large amounts of data, including data gathered from online interactions, to create powerful forecasting models.

3. Data Collection

3.1 Google Search Data

For this paper, I use data collected from Google searches as a parameter in forecasting stock price changes. The data is collected from Google Trends, a free service provided by Google, that tracks relative changes in the popularity of a search item. I use search terms related to the stocks of several different companies. The company stocks this paper attempts to forecast are: Apple (AAPL), Alphabet Inc. (GOOGL), Amazon (AMZN), and Facebook (FB). These companies were chosen because they are all very large, publicly traded, and closely tied to the internet.

While Google Trends is a very useful tool, there are a few kinks in the service that need to be addressed before the data can be used properly in the forecasting models. Firstly, Google Trends data only allows daily time series search data to be collected in 90-day periods. Any time frame longer than 90 days will result in the data being transformed into weekly time series data. To gather as much data as possible it is preferable to have daily time series data on the search terms. To adjust for this, I first collect the data in 90-day increments and append it together. While this could be done manually, to avoid human error, and to speed up the process, I wrote a program in Python, to complete the task. The program uses a Python module called *pytrends* that connects to an unofficial Google trends API (Application Programming Interface). Data is collected from a user defined number of years back to present day and returned in CSV format.

Another problem with Google trends data, is that, rather than raw search numbers for a given term, the search data is returned as relative popularity. This means that for each selected set of time for a given search term, the term is assigned a search frequency "score" relative to the day in that time frame that it was most searched. This score lies between 0 and 100 and is recalculated for every time frame. So, for example, if the term "Amazon stock" is selected in

Google Trends from the time frame of January 1st, 2017 to April 1st, 2017, the results will be daily time series data of the term's search popularity *relative* to its most searched day in that time frame. If we run another Google Trends query from April 1st, 2017 to July 1st, 2017 the relative popularity will reset and the terms' popularity for each day will be relative to the most searched day in that time frame. This means that it is impossible to get smooth daily search term data by appending many 90-day time increments together.

To get around this problem, the Python program uses some data manipulation to rescale the daily data. First, weekly data is acquired from Google Trends for the entire time period. The weekly data is then matched to the daily data, and the daily data is normalized based on the matched numbers (Johansson, 2014). Once the data is normalized, the entire adjusted daily time series data can be viewed.

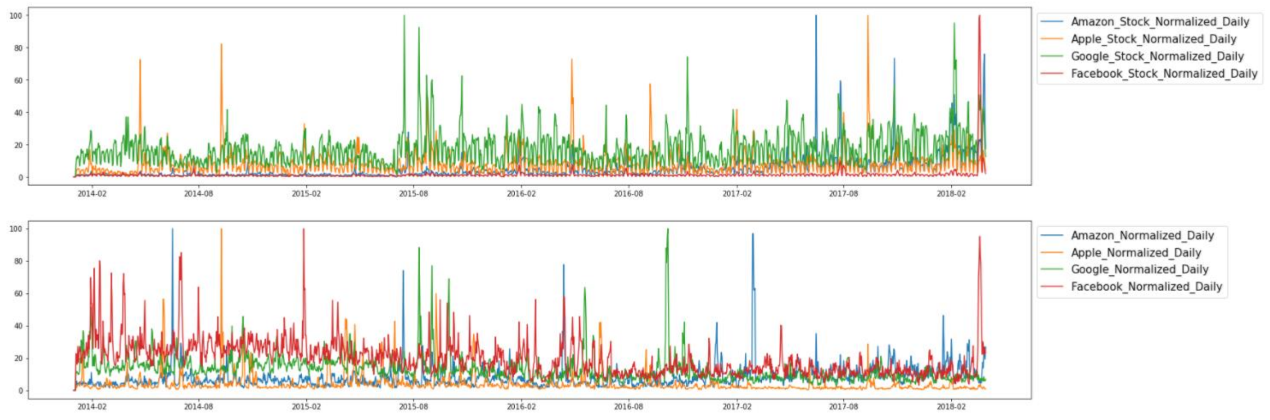


Figure 1: Google Web and News Search Terms

The first panel shows the daily relative web search frequency (how often the term was searched on Google) from Jan 1, 2014 to March 31, 2018 for the name of the select company plus the word "Stock" ("Amazon Stock" for example). The second panel, shows daily relative news search frequency (how often the term was searched on Google News), for the name of the select company. The data is normalized using the aforementioned method so that it is on a scale relative to the entire time period. It should be noted that the above plots show the entire set of time series data collected, but non-trading days are removed when the data is used in the models.

3.2 Technical Indicators

Financial technical indicators are statistical measures that use historical market activity to predict changes in prices or trends. While there are countless different technical indicators, most can be grouped into four main categories: Trend indicators, Momentum indicators, Volatility indicators and Volume indicators (Vaiz and Ramaswami, 2016). For the purposes of this paper, I use four momentum indicators (RSI, Stochastic Oscillator, SMI, and WPR), one trend indicator (MACD), and one volume indicator (OBV) as features for the models. More details about these technical indicators are listed below.

3.2.1 Relative Strength Index (RSI)

The RSI was developed by technical analyst Welles Wilder (Investopedia, 2018) and it calculates a ratio of recent upward price movements to absolute movements. The measure is used to indicate overbought and oversold market signals (Vaiz and Ramaswami, 2016).

$$RSI = 100 \times \frac{upavg}{upavg + dnavg}$$

$$\text{where } upavg = \frac{\mathrm{upavg} \times (\mathrm{n}-1) + \mathrm{up}}{\mathrm{n}} \quad \text{and} \quad downavg = \frac{\mathrm{downavg} \times (\mathrm{n}-1) + \mathrm{dn}}{\mathrm{n}}$$

3.2.2 Stochastic Oscillator (%K)

The Stochastic Oscillator was developed by George C. Lane in 1950, and it shows how the current price of a stock compares to the past high and past low price over a predefined period (Mitchell, 2018). The values of the Stochastic Oscillator range between 0 and 100 and follows the momentum of the price (Vaiz and Ramaswami, 2016). The stochastic oscillator (%K) is calculated as:

$$\%K = 100 \times \frac{\text{Close} - \text{LowestLow (last n periods)}}{\text{HighestHigh (last n periods)} - \text{LowestLow (last n periods)}}$$

3.2.3 Stochastic Momentum Index (SMI)

The stochastic momentum index was developed by William Blau and is an improved version of the Stochastic Oscillator (Vaiz and Ramaswami, 2016). The SMI is a more reliable indicator and is less subject to false swings.

$$\text{SMI} = 100 \times \frac{\text{cm}}{\frac{\text{hl}}{2}}$$

3.2.4 Williams %R (Williams Percentage Range)

The Williams Percent Range was developed by Larry Williams. The indicator ranges from 0 to -100 and indicates overbought and oversold values (Vaiz and Ramaswami, 2016). When the value is above -20, it indicates sell, and when the value is below -80 it indicates buy.

$$\%R = 100 \times \frac{\text{HighestHigh}_{\text{(last n periods)}} - \text{clos e}}{\text{HighestHigh}_{\text{(last n periods)}} - \text{LowestLow}_{\text{(last n periods)}}}$$

3.2.5 Moving Average Convergence Divergence (MACD)

The Moving Average Convergence Divergence indicator is a momentum oscillator that indicates trend (Vaiz and Ramaswami, 2016). The indicator oscillates around the 0 line and indicates to sell when the MACD crosses below the zero line or buy when the MACD crosses above the zero line. For this paper, I use both the MACD line, and signal as features.

$$\text{MACD line} = 12 \text{ day EMA} - 26 \text{ day EMA}$$
$$\text{Signal line} = 9 \text{ day EMA of MACD line}$$

3.2.6 On Balance Volume (OBV)

Developed by Joseph Granville, OBV measures the cumulative total of the up and down volume (Vaiz and Ramaswami, 2016).

$$\text{IF Close} > \text{Close}_{[-1]} \text{thenOBV} = \text{OBV}_{[-1]} + \text{Volume}$$
$$\text{Else If Close} < \text{Close}_{[-1]} \text{thenOBV} = \text{OBV}_{[-1]} - \text{Volume}$$
$$\text{ElseOBV} = \text{OBV}_{[-1]}$$

3.3 Search Data Features

Rather than using just the relative search score collected on the Google search terms as features, I calculate several measures based on this data as well. These measures help the models capture larger trends in the search data, to increase overall prediction accuracy. These measures are listed below. The inspiration for most of these features came from Weng (2017).

3.3.1 Search Term Simple Moving Average

This is the simple moving average of each search term. Moving averages are calculated at periods 6, 8, 10, and 20. This allows the models the capture average changes within subsets of time over the course of the time series. The first difference of the MA for each search term is also calculated.

$$\text{SMA} = \frac{\sum \text{Term}}{\text{n}}$$

$$\text{First Difference} = \text{Y}_\text{t} - \text{Y}_{\text{t}-1}$$

3.3.2 Search Term Exponential Moving Average

This is the exponential moving average of each search term. The exponential moving average puts more weight on more recent values than past values. The first difference of the EMA for each search term is also calculated.

$$\text{EMA} = \text{EMA}_{(-1)} + \text{K x } \left(\text{input} - \text{EMA}_{(-1)}\right) \text{ where } \text{K} = \frac{2}{(\text{n}+1)}$$

3.3.3 Search Term Disparity

This is simply the ratio of the Google relative search score for the term to its x day moving average. The first difference of the search term disparity is also calculated.

3.3.4 Change of RSI

The first difference of the Relative Strength Index (RSI) for each search term is calculated. See the calculation for the RSI, in the previous section.

3.4 Target

The target I am predicting is the directional change of Apple's closing stock price for a 3, 5, 15, and 30 day forecast horizon. The equation for the target is,

$$\text{Target}_{\text{d}} = \text{Sign}(\text{close}_{\text{t+d}} - close_t)$$

Where *t* is the current period and *d* is the days-ahead for the target. The target for each forecast horizon, will take a value of 1 if the closing price in period *t+d* is greater than the closing price in period *t*, and -1 if the closing price in period *t+d* is less than the closing price in period *t*. I then convert the numbers so that the target takes the value 1 if the price goes up and 0 if the price does not go up.

4. Algorithms

4.1 Support Vector Machines (SVM)

Support vector machines are a popular machine learning method that uses a maximal margin classifier to separate data into groups for classification. Data is separated by some p-dimensional hyper-plane defined by,

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0 \tag{4.1}$$

There are obviously an infinite number of possible separating hyperplanes, so to decide which hyperplane to use a maximal margin hyperplane is created. The maximal margin hyperplane is the hyperplane that is furthest from the closest observations (James, Witten, Hastie & Tibshirani 2013). A graphical example of the maximal margin hyperplane can be seen in figure 2.

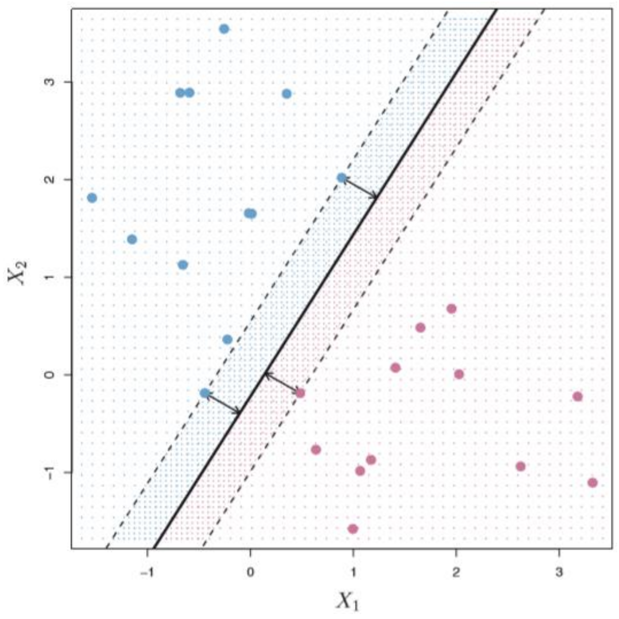


Figure 2: Maximal Margin Hyperplane (James et al., 2013)

The above 2-dimmensional hyperplane is an optimal separating hyperplane because it maximizes the distance between a few closest points on either side of it.

To construct the maximal margin hyperplane for a set of *n* observations $\mathrm{x}_1, \mathrm{x}_2, \dots, \mathrm{x}_\mathrm{n}$ with class labels $\mathrm{y}_1, \dots, \mathrm{y}_\mathrm{n} \in \{-1,1\}$ we create the below maximization problem.

$$\underset{\beta_0,\beta_1,\dots,\beta_p,M}{maximize}\ \mathrm{M} \tag{4.2}$$

$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 = 1 \tag{4.3}$$

$$y_i\left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}\right) \geq \mathrm{M}\ \forall\ \mathrm{i} = 1, \dots, \mathrm{n} \tag{4.4}$$

Equation (4.3) and (4.4) guarantees that all observations will be on the correct side of the hyperplane, and that each observation is at least distance *M* from the hyperplane. Therefore, we choose values for $\beta_0, \beta_1, \dots, \beta_p$ that maximizes the margin of the hyperplane *M* (James et al., 2013).

This works well for separable data, but of course would be implausible for data where we cannot separate all the observations perfectly with a maximal margin classifier. To adjust for this, we introduce the support vector classifier. The support vector classifier allows for some observations to be on the wrong side of the hyperplane. The maximization problem for the support vector classifier is shown below.

$$\underset{\beta_0,\beta_1,\dots,\beta_p,\epsilon_1,\dots,\epsilon_n,M}{maximize}\ M \tag{4.5}$$

$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 = 1 \tag{4.6}$$

$$y_i\left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}\right) \geq M(1 - \epsilon_i)\ \forall\ i = 1, \dots, n \tag{4.7}$$

$$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C \tag{4.8}$$

In the above maximization problem, we have a very similar setup as in (4.2) – (4.4). However, we now allow some observations to fall on the wrong side of the margin and even on the wrong side of the hyperplane. The leniency of which we allow this misclassification to occur for each variable x_i is controlled by the slack variable ϵ_i, which is in turn set by the tuning parameter *C*. If *C* is larger, then more observations can be misclassified. If C is smaller, less observations are allowed to be misclassified. If C = 0, then we have the same maximization problem as in (4.2) – (4.4) (James et al., 2013). Figure 3 shows an example of a support vector classifier.

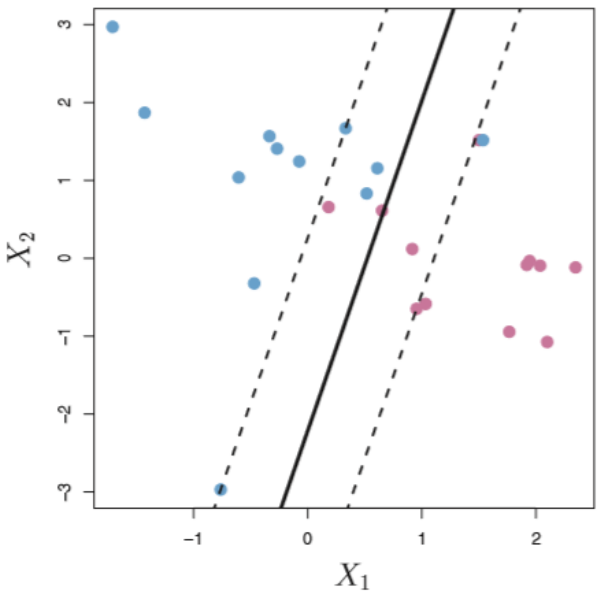


Figure 3: Support vector classifier (James et al., 2013)

Now that we have solved for the possibility of the data being non-separable, we must also allow for the possibility that the data is non-linear. To accommodate for this possibility, we may want to enlarge our feature space (James et al., 2013). However, enlarging the feature space with a large number of variables could quickly lead to a very complicated maximization problem when calculating the support vector classifiers. We can instead use support vector machines to enlarge the feature space in a computationally efficient way (James et al., 2013). The support vector machine uses kernels that find the inner product between observations. For example, the radial kernel, which is the kernel used in this paper, is calculated as shown in equation (4.9).

$$K(x_i, x_{i'}) = \exp\left(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2\right) \tag{4.9}$$

where γ is some positive constant. In the above equation the Euclidian distance $(x_{ij} - x_{i'j})^2$ is used to determine how test observations are classified based on how close or near they are to the training observation. This means that, in a radial kernel, nearby training observations have a much stronger effect in classifying test observations (James et al., 2013). A graphical example of an SVM with a radial kernel can be seen in figure 4.

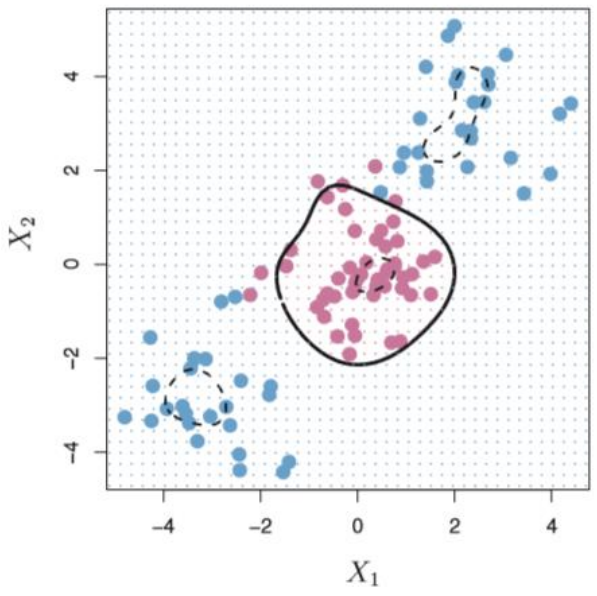


Figure 4: SVM with Radial Kernel (James et al., 2013)

This section only covers the basics of support vector machines. For further information on the mathematics behind support vector machines the reader is encouraged to reference, James et al. (2013); Hastie, Tibshirani, and Friedman (2008); and Kowalczyk (2017).

4.2 Random Forests (RF)

Random forests are another very popular and powerful model used in machine learning. They are often used for classification problems but can also be used to fit regression problems. Before explaining random forests, I first give a brief introduction to decision trees, a simpler form of random forests.

Decision trees are a rather simple, but very powerful, machine learning algorithm. Like random forests, they can be used to solve both regression or classification problems. A decision tree works by splitting the predictor space into various points, thereby creating separate groups to use for prediction. A simple graphical depiction of a decision tree can be seen in figure 5 where a decision tree model is used to predict whether a passenger of the titanic would have lived or died.

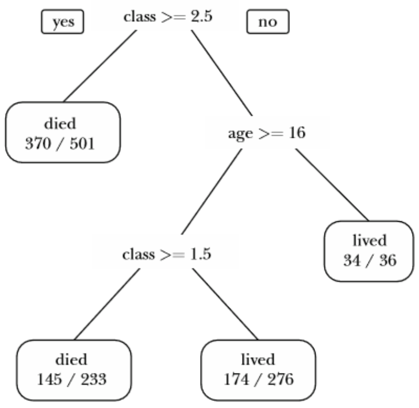


Figure 5: Decision Tree for Titanic Survival (Varian, 2014)

In the above example, the decision tree uses different predictors to split the data into several, non-overlapping regions (James et al., 2013). Those splits are then used to classify out of sample data and make predictions.

To create the splits, the tree uses a top-down, greedy process called recursive binary splitting (James et al., 2013). The initial set of observations is first split into two regions (m_1 and m_2). In the above example this would be the class being greater than or equal to 2.5 or less than 2.5. This split is made, by deciding on a predictor variable X_j and a split s that minimizes some objective function. For regression the RSS is usually minimized. For classification problems several objective functions can be minimized, most commonly the *Gini Index* (equation 4.10) or *Entropy* (equation 4.11).

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{4.10}$$

$$\mathrm{D} = -\sum_{\mathrm{k}=1}^{\mathrm{K}} \hat{\mathrm{p}}_{\mathrm{mk}} \log \hat{\mathrm{p}}_{\mathrm{mk}} \tag{4.11}$$

where $\hat{\mathrm{p}}_{\mathrm{mk}}$ is the proportion of observations that are in the *m*th region in the *k*th class. After the 1st split is made, another split is created on one of the two newly created regions, minimizing the same objective function. This creates a 3rd region. This process continues until there are enough regions that hold some small number of observations (James et al., 2013).

While decision trees can prove to be very useful in making in-sample predictions, they are very prone to overfitting, and their prediction accuracy greatly decreases in out of sample predictions (Varian, 2014). This is often due to the fact that decision trees tend to have very low bias but very high variance. That is, a tree may fit one training set very well, but slight changes in the observations of that training set could produce a very different model from the original (James et al., 2013). This is especially true as trees become more complicated with many branches.

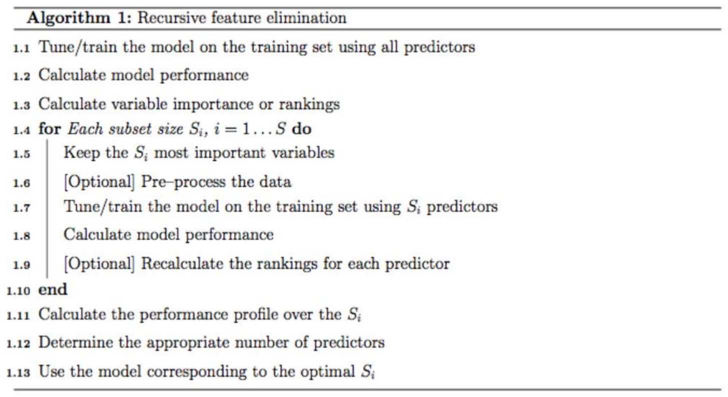
We can reduce this high variance problem in decision trees, by using random forests. Random forests use a method called bagging to create a model for prediction. Bagging takes random samples of the data and builds several models on each separate sample (James et al., 2013). For regression the new prediction for a given out of sample observation is the average of the prediction of each model. For classification we can simply take a majority vote to make our prediction. This averaging over many models, greatly diminishes the high variance problem, and improves out of sample prediction.

The random forests method takes the process of bagging a step further by also decorrelating the different trees. To do this each tree in the forest is only allowed to consider a random subset of the total predictors each time a split in a tree is made. (James et al., 2013). This keeps every tree in the random forest model from selecting the same highly important predictors, and thus decreases the variance even further.

5. Recursive Feature Elimination (RFE)

A common problem among many forecasting models, is overfitting of the data. Overfitting is when a model created for a sample of data has excellent predictions for the sample but performs very poorly for any out of sample predictions. One of the main causes of overfitting is including too many variables, or features, in the model. Oftentimes, when there are too many features included in a model, the features that are less important or correlate too strongly with other features, cause the model to perform poorly in out-of-sample predictions. To eliminate this problem, I use a method called Recursive Feature Elimination (RFE).

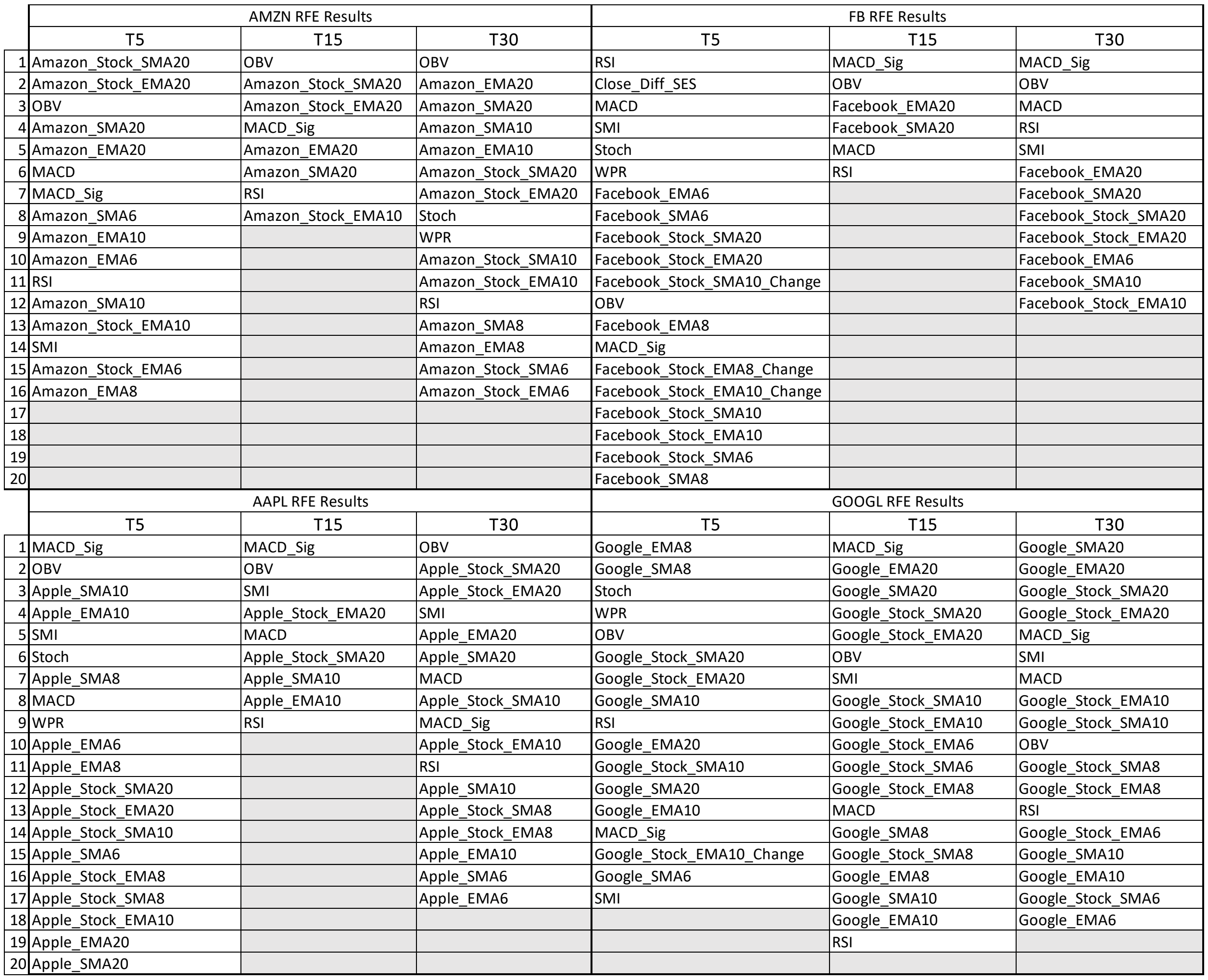
Recursive feature elimination uses a process called backward selection. The process essentially computes a model that includes all the variables first, and then drops the least important variable. The model is then recalculated with all the variables except the dropped variable. This process continues until enough features are dropped to meet either an optimal or user defined stopping point (Kuhn, 2017). A detailed depiction of the process can be seen in figure 6.



```
Algorithm 1: Recursive feature elimination
1.1  Tune/train the model on the training set using all predictors
1.2  Calculate model performance
1.3  Calculate variable importance or rankings
1.4  for Each subset size $S_i$, $i = 1 \ldots S$ do
1.5      Keep the $S_i$ most important variables
1.6      [Optional] Pre–process the data
1.7      Tune/train the model on the training set using $S_i$ predictors
1.8      Calculate model performance
1.9      [Optional] Recalculate the rankings for each predictor
1.10 end
1.11 Calculate the performance profile over the $S_i$
1.12 Determine the appropriate number of predictors
1.13 Use the model corresponding to the optimal $S_i$
```

Figure 6 Recursive Feature Elimination (Kuhn, 2017)

The Recursive feature elimination process is run 12 different times total: once for each period ahead forecast horizon (5, 10, and 15) and for each company (GOOGL, AMZN, FB, AAPL). There were 59 predictor variables in the final data set for each company. I set the subset size to a range of 1-20, so that each round of RFE returned an optimal number of features between 1 and 20, for each respective company and forecast horizon. The algorithm used for RFE was a random forest algorithm. The final selected features from running RFE for each company and forecast horizon can be seen in figure 7.



	AMZN RFE Results			FB RFE Results		
	T5	T15	T30	T5	T15	T30
1	Amazon_Stock_SMA20	OBV	OBV	RSI	MACD_Sig	MACD_Sig
2	Amazon_Stock_EMA20	Amazon_Stock_SMA20	Amazon_EMA20	Close_Diff_SES	OBV	OBV
3	OBV	Amazon_Stock_EMA20	Amazon_SMA20	MACD	Facebook_EMA20	MACD
4	Amazon_SMA20	MACD_Sig	Amazon_SMA10	SMI	Facebook_SMA20	RSI
5	Amazon_EMA20	Amazon_EMA20	Amazon_EMA10	Stoch	MACD	SMI
6	MACD	Amazon_SMA20	Amazon_Stock_SMA20	WPR	RSI	Facebook_EMA20
7	MACD_Sig	RSI	Amazon_Stock_EMA20	Facebook_EMA6		Facebook_SMA20
8	Amazon_SMA6	Amazon_Stock_EMA10	Stoch	Facebook_SMA6		Facebook_Stock_SMA20
9	Amazon_EMA10		WPR	Facebook_Stock_SMA20		Facebook_Stock_EMA20
10	Amazon_EMA6		Amazon_Stock_SMA10	Facebook_Stock_EMA20		Facebook_EMA6
11	RSI		Amazon_Stock_EMA10	Facebook_Stock_SMA10_Change		Facebook_SMA10
12	Amazon_SMA10		RSI	OBV		Facebook_Stock_EMA10
13	Amazon_Stock_EMA10		Amazon_SMA8	Facebook_EMA8		
14	SMI		Amazon_EMA8	MACD_Sig		
15	Amazon_Stock_EMA6		Amazon_Stock_SMA6	Facebook_Stock_EMA8_Change		
16	Amazon_EMA8		Amazon_Stock_EMA6	Facebook_Stock_EMA10_Change		
17				Facebook_Stock_SMA10		
18				Facebook_Stock_EMA10		
19				Facebook_Stock_SMA6		
20				Facebook_SMA8		

	AAPL RFE Results			GOOGL RFE Results		
	T5	T15	T30	T5	T15	T30
1	MACD_Sig	MACD_Sig	OBV	Google_EMA8	MACD_Sig	Google_SMA20
2	OBV	OBV	Apple_Stock_SMA20	Google_SMA8	Google_EMA20	Google_EMA20
3	Apple_SMA10	SMI	Apple_Stock_EMA20	Stoch	Google_SMA20	Google_Stock_SMA20
4	Apple_EMA10	Apple_Stock_EMA20	SMI	WPR	Google_Stock_SMA20	Google_Stock_EMA20
5	SMI	MACD	Apple_EMA20	OBV	Google_Stock_EMA20	MACD_Sig
6	Stoch	Apple_Stock_SMA20	Apple_SMA20	Google_Stock_SMA20	OBV	SMI
7	Apple_SMA8	Apple_SMA10	MACD	Google_Stock_EMA20	SMI	MACD
8	MACD	Apple_EMA10	Apple_Stock_SMA10	Google_SMA10	Google_Stock_SMA10	Google_Stock_EMA10
9	WPR	RSI	MACD_Sig	RSI	Google_Stock_EMA10	Google_Stock_SMA10
10	Apple_EMA6		Apple_Stock_EMA10	Google_EMA20	Google_Stock_EMA6	OBV
11	Apple_EMA8		RSI	Google_Stock_SMA10	Google_Stock_SMA6	Google_Stock_SMA8
12	Apple_Stock_SMA20		Apple_SMA10	Google_SMA20	Google_Stock_EMA8	Google_Stock_EMA8
13	Apple_Stock_EMA20		Apple_Stock_SMA8	Google_EMA10	MACD	RSI
14	Apple_Stock_SMA10		Apple_Stock_EMA8	MACD_Sig	Google_SMA8	Google_Stock_EMA6
15	Apple_SMA6		Apple_EMA10	Google_Stock_EMA10_Change	Google_Stock_SMA8	Google_SMA10
16	Apple_Stock_EMA8		Apple_SMA6	Google_SMA6	Google_EMA8	Google_EMA10
17	Apple_Stock_SMA8		Apple_EMA6	SMI	Google_SMA10	Google_Stock_SMA6
18	Apple_Stock_EMA10				Google_EMA10	Google_EMA6
19	Apple_EMA20				RSI	
20	Apple_SMA20					

Figure 7: Recursive Forecast Elimination Results

Each target (T5, T15, and T30), represent the 3 forecast horizons respectively. A full list of each feature name, and its corresponding meaning can be found in the appendix. The features for each forecast horizon and company are also ranked in descending order from most important to least important, as decided by the RFE algorithm. Some interesting findings came from the results of the RFE. Firstly, for every RFE result, there were features relating to the Google search terms chosen. This shows that the Google search data contains valuable predictive power for the target for every company and forecast horizon. Secondly, some of the technical indicators show up much more than others, specifically the SMI, MACD, and OBV indicators. These indicators are also usually higher up on each list. This suggests SMI, MACD, and OBV are much more powerful in predicting changes in closing price than other technical indicators.

Another interesting finding is that for every company except AAPL, there are very few variables selected for the 15-step ahead forecast horizon. This suggests that because the RFE algorithm finds the optimal number of variables between the specified range, many of the variables were not providing any extra forecasting power for the 15-day forecast horizon. A graphical depiction of the 15-step ahead RFE for FB can be seen in figure 8.

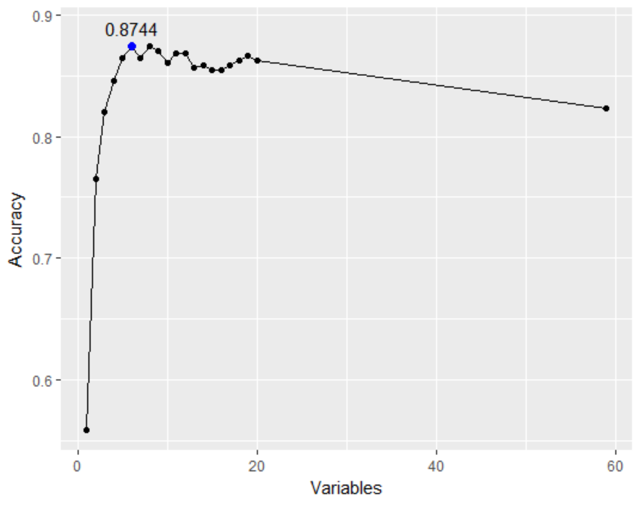


Figure 8: FB 15-step ahead RFE

Figure 8 shows that for forecasting Facebook's stock price, prediction accuracy is highest at the six variables shown in figure 7, after that point there is little to no value gained in including more variables. The predictive power of those six variables is also stronger than when all 59 variables are included.

Finally, when RFE was run for Google, at all forecast horizons, there were many variables kept, and a large number of those were Google search variables. This shows that not only are more predictors better for predicting Google's stock price, but that Google searches are very good predictors for Google's stock price as well. This is expected, since Google's business is almost entirely based on human interaction with the internet. A large uptick in searches for "Google" and "Google Stock", would provide a good indicator for a change in Google's stock price. Figure 9 shows the RFE path for each forecast horizon for Google.

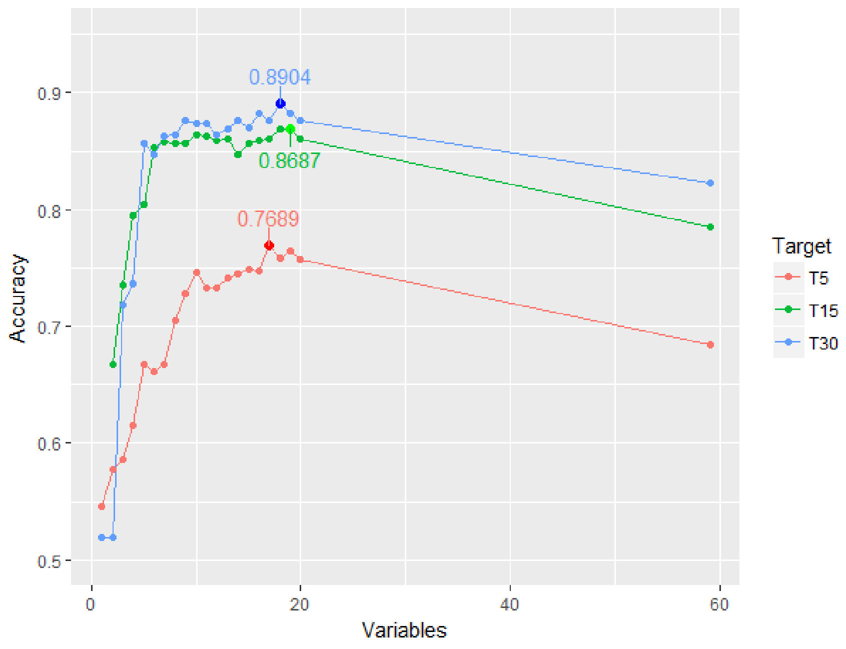


Figure 9: Google RFE

6. Results

The selected features from RFE for each company and each forecast horizon are first run through a support vector machine model and then a random forests model. The total sample size for running the model is from Feb 2, 2014 to Feb 14, 2018, for a total of 1,005 observations. The process for testing the performance of the model was a train/test validation process. A common practice in machine learning is to use k-fold cross validation to avoid overfitting when testing model performance. However, this method of cross validation requires the order of the data to be randomized, and several random samples to be drawn from the dataset. Since this cannot be done for time series data, I instead use a process called recursive forecasting.

In recursive forecasting, the data is split into an initial train and test set. For this paper, I split each dataset (for each respective company) into a train and test set at the half way point. The model is then run on the initial training data. Using the model trained on the initial training data, the next step ahead is forecasted, and the accuracy of the prediction is saved. The process then repeats itself adding one more observation to the data. This process of training and testing continues until the end of the dataset is reached, and the final model evaluation can be completed. A figure from a blog post made by Rob Hyndman (2016), illustrates this process well.

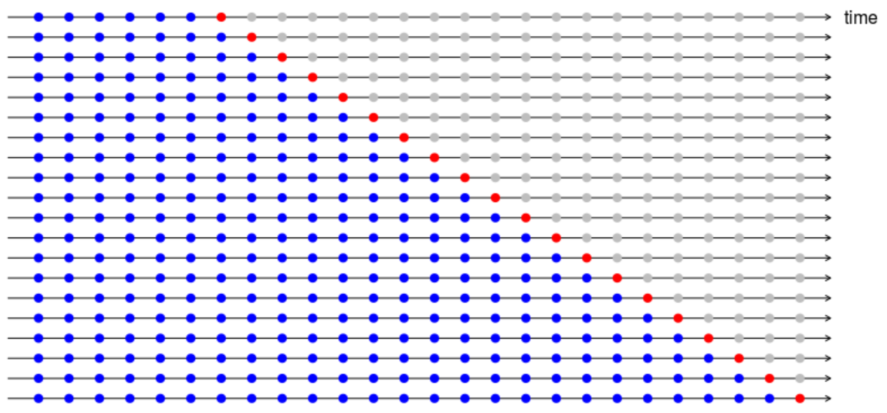


Figure 10: Recursive forecast illustration (Hyndman 2016).

In the above example, each set of training data is shown in blue, while the test sets are in red. As the process repeats, the training set moves forward one and tests the next observation.

For the SVM model, the kernel selected was a radial-basis function, and the type was classification. The random forest model was set to use 25 trees. The results for each model can be seen in figures 11 and 12.

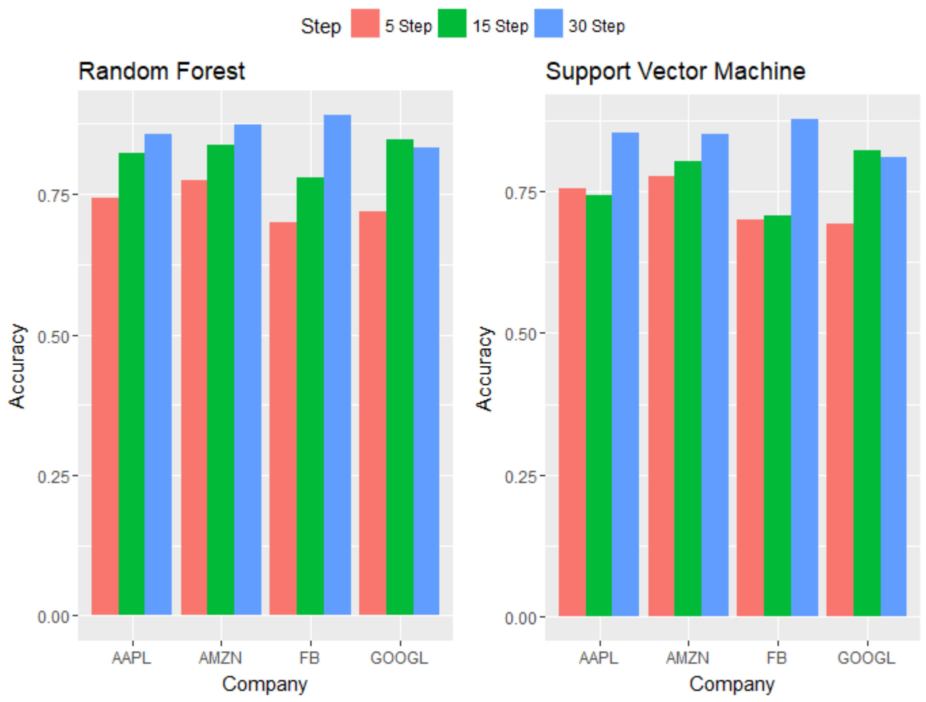


Figure 11: Random Forest and SVM Results

	Support Vector Machine							
	GOOGL SVM Results				AAPL SVM Results			
	Accuracy	Precision	Recall	Specificity	Accuracy	Precision	Recall	Specificity
5 Step	0.693676	0.746711	0.744262	0.616915	0.754941	0.775385	0.831683	0.640394
15 Step	0.822134	0.84593	0.887195	0.702247	0.743083	0.766423	0.902579	0.388535
30 Step	0.810277	0.8625	0.841463	0.752809	0.853755	0.88102	0.906706	0.742331
	AMZN SVM Results				FB SVM Results			
	Accuracy	Precision	Recall	Specificity	Accuracy	Precision	Recall	Specificity
5 Step	0.77668	0.812689	0.840625	0.666667	0.701581	0.754266	0.736667	0.650485
15 Step	0.804348	0.894444	0.840731	0.691057	0.70751	0.78187	0.795389	0.515723
30 Step	0.851779	0.936224	0.880096	0.719101	0.87747	0.909814	0.924528	0.748148
	Random Forest							
	GOOGL RF Results				AAPL RF Results			
	Accuracy	Precision	Recall	Specificity	Accuracy	Precision	Recall	Specificity
5 Step	0.719368	0.74184	0.819672	0.567164	0.743083	0.74928	0.858086	0.571429
15 Step	0.84585	0.867647	0.89939	0.747191	0.822134	0.845333	0.908309	0.630573
30 Step	0.832016	0.883281	0.853659	0.792135	0.855731	0.859043	0.941691	0.674847
	AMZN RF Results				FB RF Results			
	Accuracy	Precision	Recall	Specificity	Accuracy	Precision	Recall	Specificity
5 Step	0.774704	0.78453	0.8875	0.580645	0.699605	0.722892	0.8	0.553398
15 Step	0.835968	0.878788	0.908616	0.609756	0.778656	0.834758	0.84438	0.63522
30 Step	0.873518	0.907621	0.942446	0.550562	0.889328	0.902813	0.951482	0.718519

Figure 12: SVM and RF Results

The above table shows the corresponding results for the support vector machine and random forest model in predicting stock price directional changes for each company at each forecast horizon (5 step, 15 step, and 30 step). Accuracy measures the overall proportion of correct predictions made by the model. Precision measures the ratio of true positives to the total true and false positive predictions. Recall measures the ability for the model to predict positive labels (an increase in price), and specificity measures the ability for the model to predict negative labels (a decrease in price). A list of the formulas for these calculations can be found in the appendix.

The results show that, overall, both models perform very well in predicting directional changes in the stock prices of each company. In many cases, the random forest model seems to perform slightly better than the support vector machine model. In almost all cases the specificity was significantly lower than the other measurements, showing that the models were particularly weak in predicting when the stock price would go down. The best performance comes from the random forest model, which was able to obtain an accuracy of 88.9% for a 30-day ahead forecast for Facebook's stock.

It is also important to see whether the models performed better when given the information from the Google Search data. The results in figure 13 show a comparison between the support vector machine and random forest results when given all the RFE selected variables, as previously shown (RF - All and SVM - All), versus the same models when only given the technical indicators as features (RF - Technical Indicators and SVM - Technical Indicators). This comparison only predicts AAPL's 15-day ahead stock price changes as an example.

	Accuracy	Precision	Recall	Specificity
RF - Technical Indicators	0.693676	0.769444	0.793696	0.471338
SVM- Technical Indicators	0.626482	0.732558	0.722063	0.414013
RF - All	0.853755	0.87062	0.925501	0.694268
SVM - All	0.843874	0.866848	0.91404	0.687898

Figure 13: Search data vs Technical Indicators for AAPL 15-day forecast

The results show that the accuracy of the predictions improve quite significantly for both models when the Google Search data features are added. This suggests that the Google search data does in fact have an effect in improving predictions on stock price changes.

7. Conclusion

Predicting the stock market is an extremely challenging task due to its random nature. However recent advancements in machine learning have given way to a surge in opportunities to more accurately predict stock prices. When given both technical indicators and more recently available internet related features, these models prove to have a significant ability in forecasting stock price changes.

In this paper, I am able to use these models and features to achieve prediction accuracy results as high as 89.9%, which either beats or comes close to results from recent literature in the field. I am also able to show that the inclusion of features generated from Google search data provides significant improvement in prediction accuracy. These results show possible investment opportunities using similar data sets generated from the Python program created for this paper.

While the this paper revealed impressive results, there are a few limitations. For one, only the directional changes are forecasted. It would be more beneficial for an investor to know not only the direction of a stock price change, but the magnitude as well. Additionally, only the few selected companies were used in this example. The predictive power for the stocks of other companies, especially those in other sectors, not closely related to the internet, may be weaker.

This paper also opens the door for further research in this area. There could be benefit gained from adding more Google search terms related to the company as features in the models. As previously mentioned, one could also adjust the model into a regression forecast, which would allow investors to forecast the magnitude of a price change. Finally, there may also be other related internet-based features that could prove useful in forecasting, such as sentiment analysis on Google news article headlines or twitter feed data on a specific company.

While still stochastic and difficult to predict in nature, many recent developments in machine learning, computational efficiency, and statistical forecasting allow for far more accurate predictions of changes in stock market prices. As these advancements continue and as societal infrastructure moves more processes towards online based interactions, increasing the amount and value of available data, forecasting ability will more than likely remain on the current upward path. This will in turn provide a great reward for investors, and an even greater accomplishment for researchers in the field of forecasting.

References

Choudhry, R., & Garg, K. (2008). A hybrid machine learning system for stock market forecasting. *World Academy of Science, Engineering and Technology*, *39*(3).

Di, X. (2014). Stock Trend Prediction with Technical Indicators using SVM.

Fama, E. F. (1995). Random walks in stock market prices. *Financial analysts journal*, *51*(1).

Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning*. New York: Springer series in statistics.

Hyndmann, R. (2016, December 05). Cross-validation for time series. Retrieved May 10, 2018, from https://robjhyndman.com/hyndsight/tscv/

Investopedia (2016, November 02). Relative Strength Index (RSI). Retrieved May 10, 2018, from https://www.investopedia.com/terms/r/rsi.asp

Investopedia (2018, March 18). Technical Indicator. Retrieved May 10, 2018, from https://www.investopedia.com/terms/t/technicalindicator.asp

James, G., Tibshirani, R., Witten, D., & Hastie, T. (2013). An introduction to statistical learning-with applications in R. New York: Springer series in statistics.

Johansson, E. (2014, December 07). Global Payment Trends. Retrieved May 10, 2018, from http://erikjohansson.blogspot.com/2014/12/creating-daily-search-volume-data-from.html

Khaidem, L., Saha, S., & Dey, S. R. (2016). Predicting the direction of stock market prices using random forest. *arXiv preprint arXiv:1605.00003*.

Kowalczyk, A. (2017). Support Vector Machines Succinctly. Morrisville: Syncfusion Inc.

Kuhn, M. (2017, September 04). The caret Package. Retrieved May 10, 2018, from https://topepo.github.io/caret/index.html#

Madge, S., & Bhatt, S. (2015). Predicting Stock Price Direction using Support Vector Machines. *Independent Work Report Spring*.

Malkiel, B. G., & McCue, K. (1985). *A random walk down Wall Street* (Vol. 8). New York: Norton.

Mitchel, C. (2014, June 25). Ultimate Guide to the Stochastic Oscillator. Retrieved May 10, 2018, from https://traderhq.com/stochastic-oscillator-ultimate-guide/

Nau, R. (2014). Notes on the random walk model. *Fuqua School of business, Duke University*.

Preis, T., Moat, H. S., & Stanley, H. E. (2013). Quantifying trading behavior in financial markets using Google Trends. *Scientific reports*, *3*, srep01684.

Ulrich, J. (2018). TTR: Technical Trading Rules. R package version 0.23-3. https://CRAN.R-project.org/package=TTR

Vaiz, J. S., & Ramaswami, M. Forecasting Stock Trend Using Technical Indicators with R.

Varian, H. R. (2014). Big data: New tricks for econometrics. *Journal of Economic Perspectives*, *28*(2), 3-28.

Weng, B. (2017). Application of machine learning techniques for stock market predictions. *Doctor of Philosophy Dissertation Auburn University*.

Appendix

Appendix I – List of All Features

Close_Diff_SES	The 1st difference of the exponentially smoothed closing price.
RSI	Relative Strength Index
SMI	Stochastic Momentum Index
Stoch	Stochastic Oscillator
WPR	William %R
MACD	Moving Average Convergence Divergence
MACD_Sig	Moving Average Convergence Divergence Signal
OBV	On Balance Volume
X_Stock_RSI_Change	1st difference of RSI of search term “X Stock” where X is the company name
X_RSI_Change	1st difference of RSI of search term “X” where X is the company name
X_Stock_SMAY	Y(6,8,10,or 20) day simple moving average of search term “X Stock” where X is the company name
X_Stock_SMAY_Change	1st difference of Y(6,8,10,or 20) day simple moving average of search term “X Stock” where X is the company name
X_SMAY	Y(6,8,10, or 20) day simple moving average of search term “X” where X is the company name
X_SMAY_Change	1st difference of Y(6,8,10, or 20) day simple moving average of search term “X” where X is the company name
X_Stock_EMAY	Y(6,8,10,or 20) day exponential moving average of search term “X Stock” where X is the company name
X_Stock_EMAY_Change	1st difference Y(6,8,10,or 20) day exponential moving average of search term “X Stock” where X is the company name
X_EMAY	Y(6,8,10, or 20) day exponential moving average of search term “X” where X is the company name
X_EMAY_Change	1st difference Y(6,8,10, or 20) day exponential moving average of search term “X” where X is the company name

X_Stock_DisparityY	Ratio of “X Stock”(Where X is the company name) search term volume to its Y day moving average
X_Stock_DisparityY_Change	1st difference of Ratio of “X Stock”(Where X is the company name) search term volume to its Y day moving average
X_DisparityY	Ratio of “X”(Where X is the company name) search term volume to its Y day moving average
X_DisparityY_Change	1st difference Ratio of “X”(Where X is the company name) search term volume to its Y day moving average

Appendix II – Calculation of Accuracy, Precision, Recall, and Specificity

$$Accuracy = \frac{\mathrm{tp} + \mathrm{tn}}{tp + tn + fp + fn}$$

$$Precision = \frac{\mathrm{tp}}{tp + fp}$$

$$Recall = \frac{\mathrm{tp}}{tn + fp}$$

$$Specificity = \frac{\mathrm{tn}}{tn + fp}$$

Where,

tp = Number of true positives (prediction is 1 and actual value is 1)

tn = Number of true negatives (prediction is 0 and actual value is 0)

fp = Number of False positives (prediction is 1 and actual value is 0)

fn = Number of False negatives (prediction is 0 and actual value is 1)

Appendix III – Link to Code

All of the R and Python code used in this paper can be found on my GitHub:

https://github.com/jdglaser/mathesis