Neural Network Prediction of Stock Market Indices:
A Sliding Window Approach

Stephen McGee
Abhilash Menon
Team 21

Final Report
CPSC 8650
Dr. Wang

04/20/2017

Stephen McGee, Abhilash Menon
Clemson University School of Computing, Clemson, SC, USA.
sjmcgee@g.clemson.edu, abhilas@g.clemson.edu

1. Abstract

The utility of Neural Networks has been proven many times over in a variety of applications ranging from speech processing to handwriting analysis[1]. The strength of these methods relies in the implementation, where underlying hidden layers exist between the input and output. It is within these intervening areas that most of the complex relationships between features are realized and explored. In our project, we aim to implement a neural network approach to predict the trend of stock market indices over a 4 year period.

Our project focuses on stock market index data ranging from 1/18/2013 to 4/3/2017, that was collected from Yahoo Finance[2]. Our analysis relied on a 30-day sliding window approach where the 30th day of the market was predicted using a neural network created using data from the previous 29 days. The window would then be advanced by one day and the process would be repeated. This allowed for a more accurate model of the volatile nature of the stock market. Using this approach, we were able to reach our initial goal of exceeding a prediction accuracy of 70% using the keras method, however, our R implementation fell short with an accuracy of ~62-68%. We approached this project using two software packages, R statistical software, & the Keras library with a Tensorflow backing.

2. Revision from Original Project Goals

2.1 Original Paper

Originally, our project focused on implementing the methods presented in a paper proposed by Dr. Luna Zhang[3] from Soft Tech Consulting, Incorporated, located in Virginia. In this paper, Dr. Zhang proposes a variant of Deep Neural Networks (DNN) known as Genetic Deep Neural Networks (GDNN), which she and her team implement to analyze stock market indices. Their method effectively integrates known genetic algorithms to select for a variety of optimal activation functions in developing the most efficient GDNNs. Ultimately, her paper provides a detailed comparison of GDNNs to standard DNNs based primarily on the analysis of the Dow Jones Industrial Average (DJIA), a price-weighted average of 30 of the most prominent businesses in the United States.

2.2 Contact with Author

Our team contacted Dr. Zhang on February 28th, 2017 asking for permission to implement the methods presented in her paper with several minor adjustments. We also asked her about the nature of her data in addition to any advice that she may have for us moving forward.

We received a response back from Dr. Zhang on March 5, 2017 thanking us for her interest in her work, and giving us permission to use her approach with any minor adjustments that are necessary. She also provided advice as to how to approach the methods she implemented in her paper.

2.3 Decision to revise project goals

We felt that our original goal of mimicking the results of the paper would be less beneficial for our own personal data mining experience, as the author more-or-less surveyed the efficacy of alternatives to an existing data mining technique. Her approach attempted to assess the differences between varying activation functions, while our goals were geared more towards the implementation of prediction. For this reason, we decided in February to change course and focus on predicting stock market indices. This decision was reached with the advice of Dr. Wang, who stated that utilizing Deep Neural Networks

(DNNs) on a semester-long project was risky, since deep learning methods could potentially demand more time than we have available; and any fault in planning or implementation of the algorithm may prove to be detrimental to our project as we might not have had enough remaining time in the semester to successfully complete our analysis. For this reason, we decided to change our approach to something more traditional.

3. Problem Introduction

3.1 Volatility of the Stock Market

The stock market has always served as an indicator of economic health. Though telling trends seem to emerge amidst the data of years past, predicting the future of stock prices with any certainty is still an astronomically hard task, one that many state has far too many variables[4]. Largely, this is due to the nature of human behavior, which fundamentally drives the economic market itself. Sometimes, even public perception can be attributed to gains or losses in the stock market. For example, in 2008, an incorrect rumor stating of Apple CEO Steve Jobs had suffered a severe heart attack caused the stock of Apple (APPL) to fall 10% in only 10 minutes[5]. So, whether we attribute economic gains and losses to the crash of the housing market, a new presidential administration, or even a nation-wide interest in organic food consumption, human behavior is simply too complex to accurately predict with many modern techniques.

3.2 S&P500 & NASDAQ

Though the daily stock quotes for publicly traded companies are largely dependent upon a wide variety of factors, we can still provide certain metrics that aggregate the behavior of the market into one place. The S&P 500 serves as "an index of 500 companies seen as the leading indicator of U.S. equities"[6]. This includes companies that are designated as "large cap" companies, or those which have a market value exceeding $10 billion. Among those listed are companies in industries ranging from Genomics to Gaming, and even Soft Drinks[7]. Often, an index such as this is observed to assess the nature of the economy.

The NASDAQ focuses mostly on companies based in the technology industry. It serves as a "benchmark index [of more than 3000] U.S. tech stocks"[8]. Many modern companies are mapped by this stock market index, including Google, Amazon, & Apple.

Each index has its strengths and weakness. However, regardless of which one is used, they both provide valuable insight into the nature of human behavior and the overall economy. In Figure 1 shown below, we can see two distinct drops in the overall trend of the market. The fist dip begins after the peak around 2001, with a second dip beginning to occur at around 2008. These can be attributed to the "dot-com bubble" and the "housing market crash," respectively. Interestingly, the difference between the two stocks is shown in looking at these market crises. The S&P500 appears to be affected similarly between the two market crises while the NASDAQ, a primarily tech-based index, was more severely affected by the dot-com bubble, a market crash resulting from the over investment of web-site startups in the early 2000s. It was for this reason that we decided to focus most of our efforts on the S&P500 index, as it appeared to be more well-rounded in predicting market behavior.

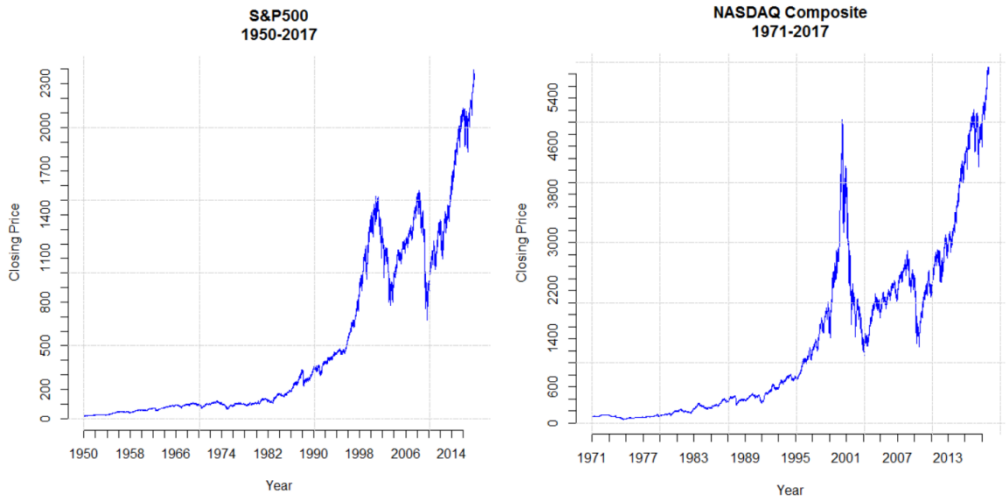

Figure 1: S&P500 Index & NASDAQ Composite

3.3 Sliding Window Approach

Originally, we intended to base our neural network around the entirety of the 4 years of data that we had collected. However, after reviewing market trends from the past decade, we determined that the volatility of the market over such a large time span was too great. So, after much trial and error, we decided to limit our analyses to an incremental sliding window of 30 consecutive "market days", where a "market day" refers to any day in which the stock market was open for trade.

4. Data Collection

4.1 Predictors

For the predictors used in our project, we decided to focus on a combination of freely available market quotes and calculated metrics commonly used in many prediction strategies. We split our list into the two generic groups: Market Data & Calculated Data.

4.2 Market Data

This data involved the historical prices for each predictor that were attained using Yahoo Finance. They were freely available and are based around the closing prices that were settled on by the end of any specific day. Below each predictor is a brief description of the predictor itself, as well as why it was chosen.

4.2.1 S&P500

This index was chosen as it is widely considered to be substantial indicator of economic health in the market place. For our analyses, we used the closing prices (or a calculation based on them) for any given day as the dependent variable in our vector of attributes. We also used the provided daily market measurements for the index including: High, Low, Volume, & Open.

4.2.2 Volatility Index (VIX)[9]

This index is a measure of the implied volatility of the S&P 500 which is released daily by the CBOE (Chicago Board Options Exchange). Often it is used to assess "market risk" with higher values (>30) tending to be associated with high market volatility [10]. This was chosen specifically to coincide with our S&P500 analysis as it directly reflects implied market turbulence. Our hope was that it would be instrumental as a predictor in our neural network analysis for deriving future values of the S&P500.

4.2.3 Adjusted U.S. Dollar Index (DX-Y.NYB)[11]

The US Dollar Index is a measure used to determine the strength of the US Dollar against 6 different currencies held by major players in global trade[12]. Values that are higher than 100 signify that the US Dollar has appreciated over other currencies during any specific period of time by a certain percentage. Higher values may indicate the strength of the US economy, while lower values may signify a downturn (or "bearish market"), potentially even a market crash.

4.2.4 Oil Prices [GCSI Crude Oil TR ETN (OIL)][13]

Often in economic discourse, the price of crude oil is often discussed. Due to the dependence upon the material in today's gasoline-driven world, it serves that oil prices would be an obvious indicator of market strength. When oil prices are high, many industries tend to suffer, whether they be the agricultural industry or even the tourism business. We expected that this predictor would be inversely related to higher closing prices in the market indices.

4.2.5 Corn Futures (ETRS Comm. Sec. Ltd ETFS CORN)[14]

Corn prices were considered in our analysis for the same reason as oil prices. Our assumption was that the S&P500 would be directly affected by the fluctuation of these prices due to the sheer number of companies measured by the index which are reliant upon the crop. Several of the companies listed within the S&P500, including Pepsi, Wal-Mart, and Tyson Foods have, at one point, if not currently, relied upon the production of high-fructose corn syrup or animal corn feed to aid in the production of their products.

4.3 Calculated Data

This data involved calculating commonly used prediction metrics on the index data itself to determine if these measurements would be beneficial in predicting future stock values. Their methods of calculations are publicly known. Below each predictor is a brief description of the predictor itself, as well as why it was chosen.

4.3.1 SMA (5, 30, & 200-Days)

Standing for "Simple Moving Average", the SMA is a basic average of prices over a given period of days. Our predictors involved three separate time frames of prediction, 5, 30, and 200 days. A visualization of this approach is given in the following graph below (Figure 2). Notice how the 200-day SMA shows a much more gradual slope of the market average, while the 5-day SMA yields a much more volatile prediction, albeit, less volatile than the actual prices themselves.

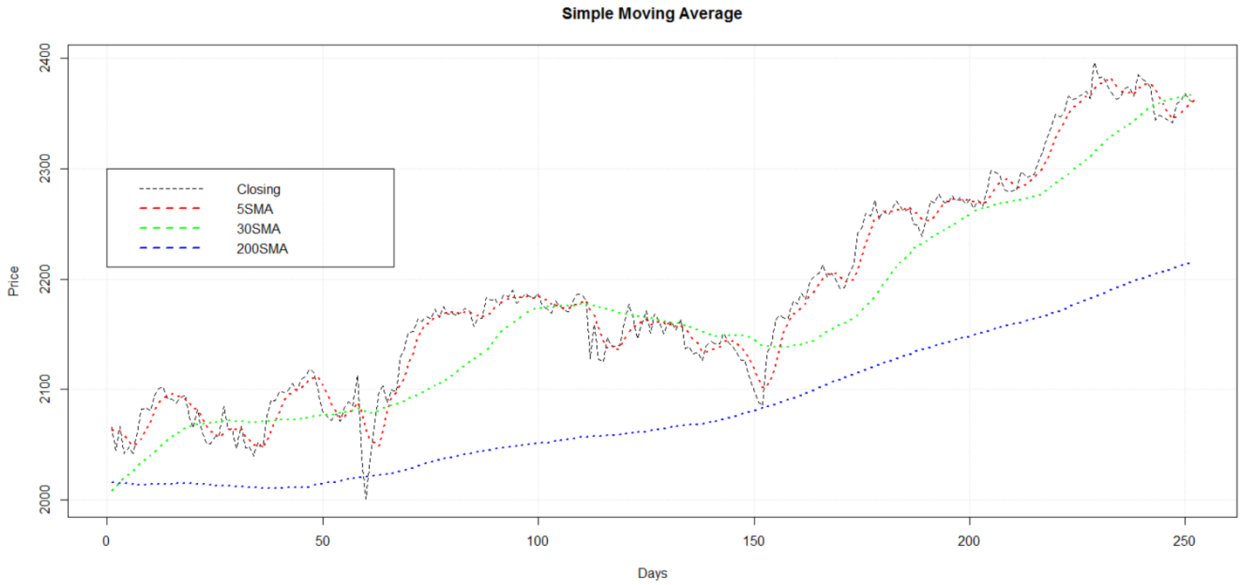

Figure 2: Simple Moving Average Comparison

4.3.2 MACD

An abbreviation for "Moving Average Convergence Divergence", MACD is a common prediction method used to determine when to sell and when to buy a certain stock. Though commonly used on specific stocks, we have implemented it here on the S&P500 index. The MACD value itself is usually calculated by subtracting the 26-day exponential moving average (EMA) from the 12-day EMA [15]. A "signal" line is then calculated as a 9-day moving average of the MACD values. The two lines are then plotted together to determine the correct time to invest in or withdraw from the market. In Figure 3 below, we have plotted the MACD and signal line for the last 60 days of our raw S&P500 data. Commonly, it is an indicator to "buy" more stocks if the MACD line crosses above the signal line. This can be shown around day 49 in the chart below. Conversely, if the MACD line falls below the signal line, it is time to sell (~day 55).

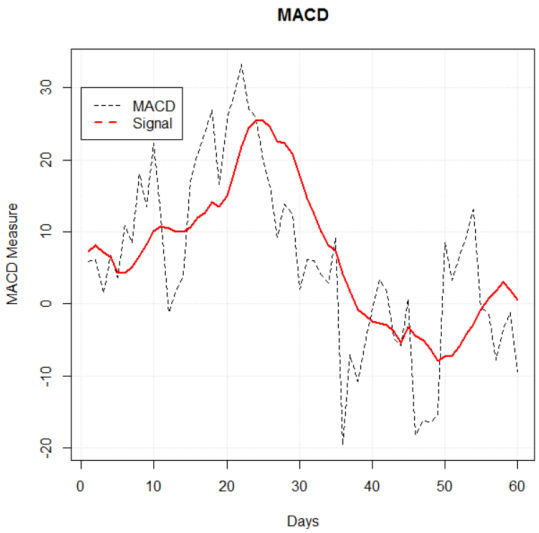

Figure 3: 60-day MACD Analysis

We decided to use the calculated MACD value itself as a measurement for our neural network models, without the aid of the Signal line. By itself, the MACD value can serve as a continuous predictor variable in our model. If we were to incorporate signal line crossovers into our analysis, then we would have been forced to consider the predictor as a discrete variable, since upward crossover would signify a time to buy (or "1"), and a downward crossover, time to sell (or "0"). For this reason, we decided to keep it simple and just utilize the MACD value itself.

4.3.3 Average True Range

The Average True Range (ATR) is a metric originating from the late-1970s which was used as a suggestive metric for market volatility [15]. The ATR is calculated by taking the minimum of a size 2 permutation of 3 separate values, including the current high, low, and previous days closing. In the Figure 4 shown below, we can see that there is a spike in the ATR with each dip in the normalized closing prices. Noticeably, the ATR line becomes steadier as the normalized market price remains stable. For this reason, we decided to use the ATR as a signifier of volatility in the market. We assumed that ATR values would allow us to better our prediction methods when creating our neural network models.

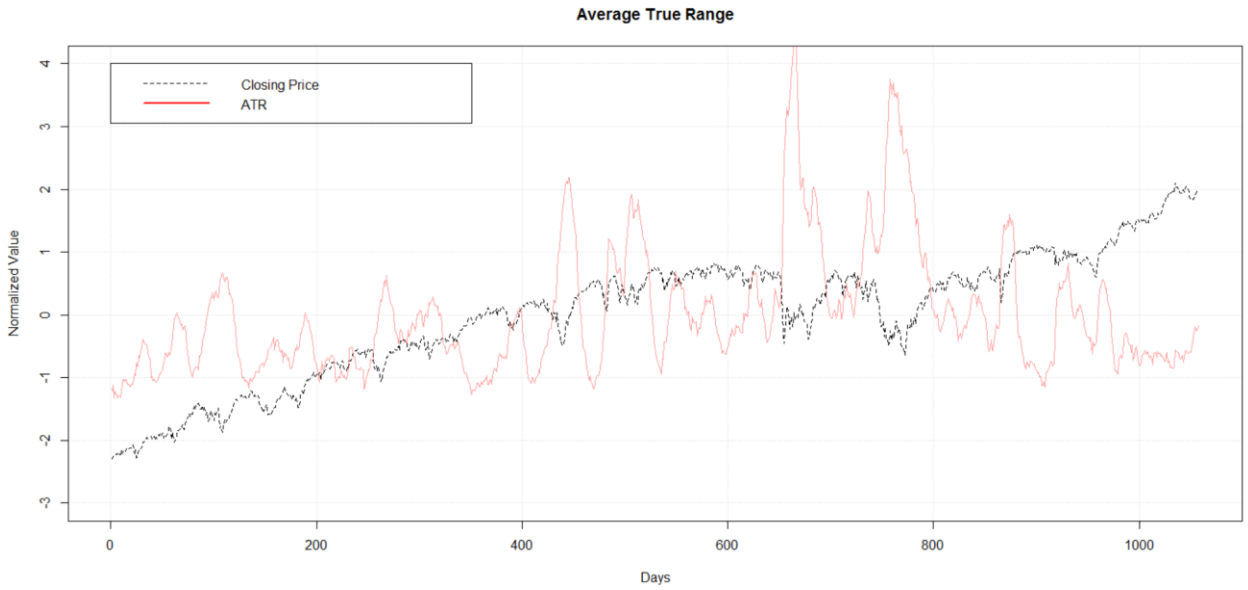

Figure 4: Normalized Average True Range

5. Neural Network Setup & Design

5.1 Missing Data

Missing data was not common in our data set. Many financial sites offer up-to-the-minute stock quotes with a veritable wealth of additional information to help with investment decisions. Historical data is easily accessible through sites such as Yahoo Finance. The only data we were missing was for 3-5 days in the Corn Futures and Oil Price data sets. These values were replaced by taking the average of the previous and following day surrounding the missing information.

5.1.2 Normalization

Originally, we had implemented a min-max normalization technique using a range of [0,1] using the formula below:

$$v' = \frac{v - \min_a}{\max_a - \min_a}$$

Where v= the current value, $\min_a$=the minimum of the range of attribute values, and max_a=the maximum of the range of values.

This was to allow for more simplified computation and to standardize any attributes collected from multiple databases. We had originally believed this to be the most optimal method for use in neural networks, however, we saw a few percentage points increase in accuracy upon switching to a z-score normalization method. These values were calculated using the following formula for each attribute:

$$v' = \frac{v - \bar{x}}{\sigma}$$

Where v= the current value, $\bar{x}$ = the mean of the attribute values, and $\sigma =$ the standard deviation of the attribute values.

5.1.3 Dependent Variables (Discrete vs. Continuous)

In trying to predict the market, we separately implemented two types of dependent variables for our models.

In our discrete models, the dependent variable was designated as a "1" if the closing price for the current day was higher than the closing price for the previous day. Alternately, a "0" was used to signify that the current day ended up being lower than the close of the previous day. Using this method, we were able to keep the analysis down to a simple question: "Could we predict, based on the previous 30 days, whether or not tomorrows prices would be higher than todays closing prices?"

In our continuous models, our dependent variable was simply established as the closing price for any specific day. Though the question that this analysis posed was not as basic, it allowed us to produce a linear visualization as to how our model predicted future stock prices. This method was used to assess how our model predicted actual stock prices over time, as opposed to just daily increases or decreases in the S&P500.

5.1.4 Neural Network Framework

Using our list of 14 predictors, we created a neural network topology of 14-28-1, utilizing twice as many hidden nodes as predictors for the discrete analysis. For the continuous analysis our topology was listed as 13-28-1, since the Closing price was now used as the dependent variable. An example of the topology used in our discrete model is shown below in Figure 5. The bias for the output node and each hidden node is highlighted in blue. All values for the weights and biases have been removed to provide a clearer visualization of the network.

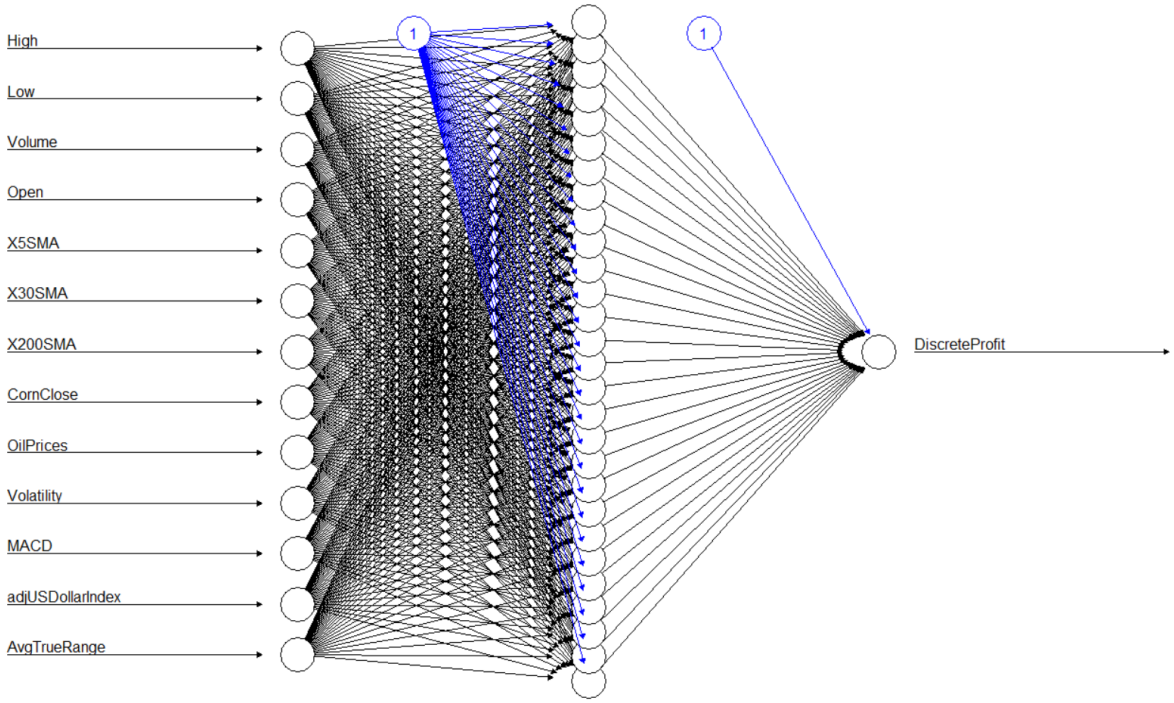

Figure 5: Neural Network Topology

6. Implementation & Efficiency

This section details out how the two separate software packages we used in implementing our neural network analyses.

6.1 R Approach

The R programming software is an open-sourced, statistical language commonly used for visualization and statistical analyses [16]. R was chosen as one of our approaches due to its ease of use in manipulating vectors, the data structure which served as the central format for representing our list of attributes for any specific day in the stock market.

In our analysis, we used R version 3.1.3 with the downloaded libraries for the "neuralnet" package [17]. In order to test out the functionality of this package, we created a neural network based around the examples provided in the package documentation. Our first example tested the discrete output functionality using the "XOR" example on page 3 of the .pdf, which returned an average accuracy of >90%.

This neural net utilized the standard backpropagation algorithm for calculations with a learning rate of 0.001. This rate was arrived at via trial and error and was used to help with convergence at each step in the network, ultimately allowing for a more accurate prediction. We established the output activation function to be sigmoid for the discrete prediction network. Using this information, we were able to achieve an average accuracy of approximately 62.47% using all 14 attributes across 1058 neural networks created in a sliding window approach over approximately 4 years worth of market data. Given that this

accuracy was not as high as we had hoped, we explored additional options using attribute filtering in subsequent sections.

6.2 Keras Approach

Keras is a high-level neural networks API, written in Python and capable of running on top of either TensorFlow or Theano. It was developed with a focus on enabling fast experimentation. In our case, we have been using tensorflow as the backend. We had installed it using bash on Windows. Some of the advantages of keras are as follows:

- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility)
- Runs seamlessly on CPU and GPU
- Compatible with: Python 2.7-3.5

Keras abstracts away from the TensorFlow providing us with a simpler API to code in Python while still making it possible to use the functionalities of TensorFlow. These advantages were the driving force behind our decision to use keras as a starting point in designing and coding our neural networks.

In order to familiarize ourselves more with keras and TensorFlow, we decided to attempt to create a neural network using publically available test datasets. After ensuring that the libraries and their dependencies were installed, we decided to test our installation using a test data pertaining to whether or not a patient had diabetes. This classification attempt yielded an accuracy of approximately 77%, which helped us make sure that the installation was capable of building a functional neural net.

The next dataset we used was more pertinent to our own project (in its preliminary stages). We had obtained historical stock data from Yahoo Finance pertaining to Apple (AAPL) dating back to 1980. Our accuracy yielded by this neural net was very low, which we first attributed to the fact that stock splits and dividends issued by the company had not been taken into account or eliminated from the dataset. To circumvent this, we reran this analysis starting from the day after the last Apple split (June 9, 2014) until early March of this year. However, the lack in accuracy persisted, which we ultimately attributed to a lack of substantial predictors. At the time, we were relying on adjusted close, open, high, low, close, and volume, which constitute the standard sock price reporting metrics. Once we moved our number of predictors up to 14, our keras implementation reached an accuracy of 88%. In order to determine that we weren't overfitting, we tried this approach with a variety of different splits of training and test data until we were satisfied with our overall outcome.

The loss function used was "binary_crossentropy" which was decided upon based on the nature of our data and its output in particular. The optimizer function we have used is "adam". After we read through some related work, we were able to find out that the typically use optimizer function for a backpropagation network is "gradient descent". Although, we decided to go ahead with "adam" as it yields a better performance and is more sophisticated compared to the standard gradient descent approach[18]. This was due to the method in which learning rate was controlled. Adam uses moving averages in its calculations, which in turn, enables it to use a larger effective step size. This is substantiated and discussed at length in section 3.1.1 of [19].

7. Regression comparison

This section shows how our R-based neural network model performed against standard regression techniques using the same normalized data. Like in the neural network model, our regression will be

based on a discrete dependent variable. For this, we introduce the concept of a probit link function which allows us to perform regression using a binary dependant variable. The probit link specifies that:

$$h(*) = \Phi^{-1}(*)$$

where Φ refers to the standard normal cumulative distribution function. This function allows us to turn our dependent dichotomous variable into a continuous variable by mapping it to the inverse-cdf (or normal pdf). Once we can map this value, we can assign all values below 0.5 to 0 and all values greater than or equal to 0.5 to 1. This was implemented in R using the "glm" function in R with "probit" specified as the binomial link function under the family variable.

Using Probit regression, we were able to derive an efficiency of around 68.58% using all 14 attributes. This shows a moderate increase in accuracy over our R neural network implementation. Perhaps, this points towards overfitting in our neural network. To determine this, we move our analysis towards modeling the most valuable attributes.

8. Data Mining Most Valuable Attributes

This section details out how we implemented a Random Forest algorithm to determine the top 5 most valuable attributes amongst our list of 14 predictors.

8.1 Random Forest Implementation

In determining the most influential variables within our analysis, we turned towards the R 'randomForest'[20] package. To prepare for this analysis, we changed each of our discrete binary dependent variables to a factor classification, where 1='y', representing a profit from one day's close over the previous, and 0='n', which represents a loss seen between consecutive days. Using the following line of R code, we can determine the distribution of "y" variables in the data set:

$$\gg table(snpData\$DiscreteProfit)/nrow(snpData)$$

This produced the following distribution (Table 1) of "profits" over the course of the data.

"n"	"y"
0.46214	0.5369

Table 1: Distribution of Profits

We then separated the data into two separates groups: training & testing. The groups were divided up by randomly selecting vectors from the data set without replacement. 85% of the vectors were placed into the training subset, while 15% of the vectors were used in the testing subset. Using these subsets, we trained the random forest using 3000 trees with a maximum height of 20.

We then incorporated the "importance" function to determine which of the 14 predictors were the most informative in determining the binary classification. This R command is shown below:

$$\gg importance(randomForestModel, type = 1)$$

This command returns the "Mean Decrease in Accuracy" for each attribute. This refers to the percentage of decrease in accuracy of our model that would result from removing any specific variable. Looking at

the table below (Table 2), if we were to remove "X5SMA" (5-day SMA) from our model, then the accuracy of our model would decrease by 41.18%. Also note that by keeping the MACD attribute as a predictor, we are actively decreasing the accuracy in our model slightly.

Attribute	MeanDecreaseAccuracy
High	29.87
Low	26.89
Volume	14.12
Close	52.29
Open	78.63
X5SMA	41.18
X30SMA	27.47
X200SMA	16.87
CornClose	11.73
OilPrices	24.31
Volatility	70.38
MACD	-0.49
AdjUSDollarIndex	16.80
AvgTrueRange	20.08

Table 2: Mean Decrease Accuracy of Predictor

According to these results, our top 5 variables, in decreasing order of importance are Open, Volatility, Close, X5SMA, & High.

The "best" tree which we derived, which was determined to be the most accurate in its classification is show below. The labeling was removed to allow it to be more visually appealing.

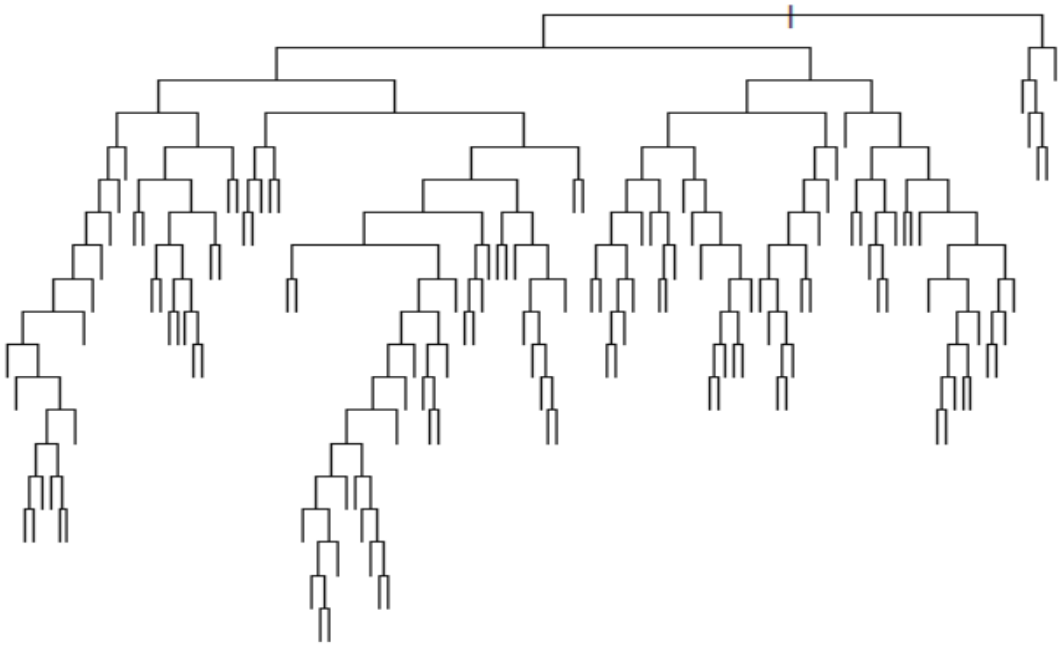

Figure 6: Best Performing Random Forest Tree

Zooming in near the top of the chart, we illustrate the classification pathways of a tree depth of 4, as shown below:

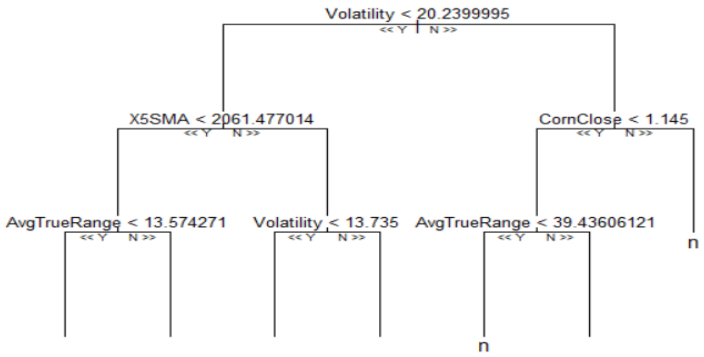

Figure 7: Best Random Forest Tree (depth=4)

With our forest created, we received an accuracy of 68.35%, with a 95% confidence interval of (60.49%, 75.51%). The following table (Table 3) shows the confusion matrix for the test data:

		Reference	
		N	Y
Predictions	N	40	21
	Y	34	58

Table 3: Confusion Matrix for Test Data

8.2 Discrete Prediction & Linear Regression of Selected attributes

In order to determine the effectiveness of the of the Random Forest algorithm in determining the most beneficial attributes, we re-ran our analysis using only the top 5 predictors from the S&P500 data. This was subsequently compared to the probit regression function using the same parsed data. With the neural network model, our accuracy decreased to 49.03% with the probit function having an accuracy of around 50.88%. This prediction accuracy is quite poor and on par with just flipping a coin. Therefore, we decided to use the attribute selection method in the continuous prediction setting.

8.3 Continuous Prediction

The attribute selection method did not prove to be as effective as originally thought for our discrete analyses. For this reason, we attempted to derive a continuous model using the same predictors (with the exception of discreteProfit), and a new neural network model topology 13:26:1. In this case, the previous days' Closing price for the S&P500 would now serve as the dependent variable for each vector in our model. In lieu of providing accuracy metrics, the mean and standard deviation of the distance between the actual and predicted values is provided, alongside a visualization of the prediction itself. In looking at the performance of our continuous stock prediction below, we can see that our model appears to much more erratic In looking at the last ~120 days of the market, we show the plot of the actual price versus our predicted continuous price below:

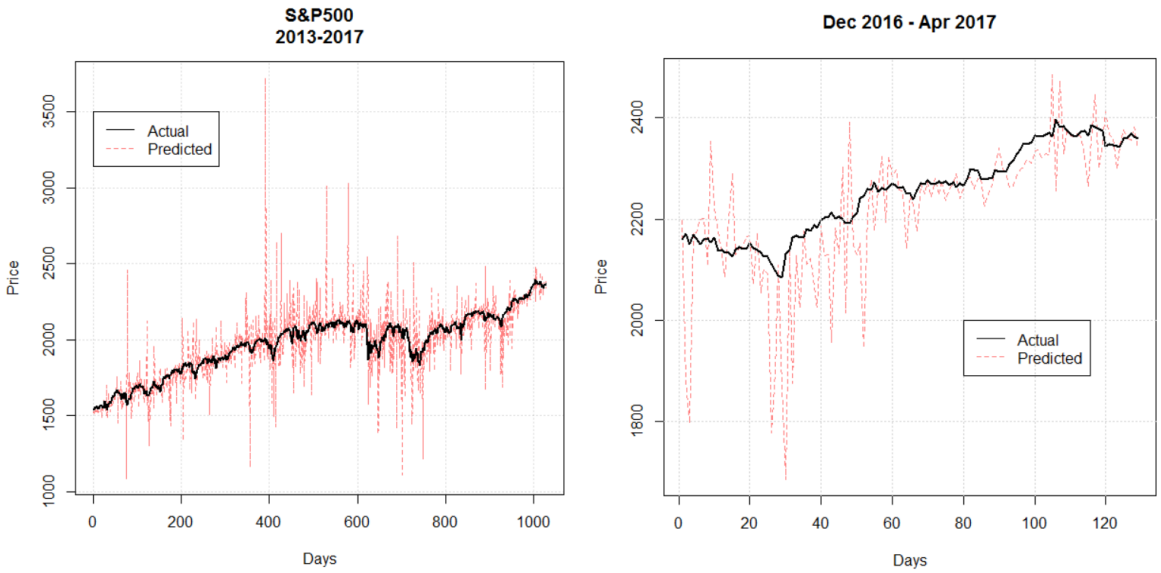

Figure 8: (left) Continuous prediction of 4 years worth of S&P500 data (right) Continuous prediction of 4 months worth of S&P500 data

It is apparent that our continuous neural network prediction is highly erratic in comparison to the actual price of the market, as shown on the left plot above. When we look at a zoomed-in portion of our prediction on the right in, in the graph above, our prediction appears to move in a steadier fashion when the actual price of the stock index is non-volatile over a period of days. In this way, our prediction appears to serve more as a volatility metric as opposed to a stock market price predictor. Perhaps we placed too many volatility metrics in our model, and they dominated the neural network itself. Regardless, the difference between the predicted and actual prices is normally distributed with a mean of -4.33 and a standard deviation of 156.55 or N~(-4.33,156.55).

Unfortunately, when we consider only the top 5 attributes as predictors for the continuous analysis, we see that the predicted line is even more erratic, to the point where there seems to be minimal gathering around the actual price. In this subanalysis, we see that the difference between the predicted price and the actual price is normally distributed ~ N(-11.70,307.73)

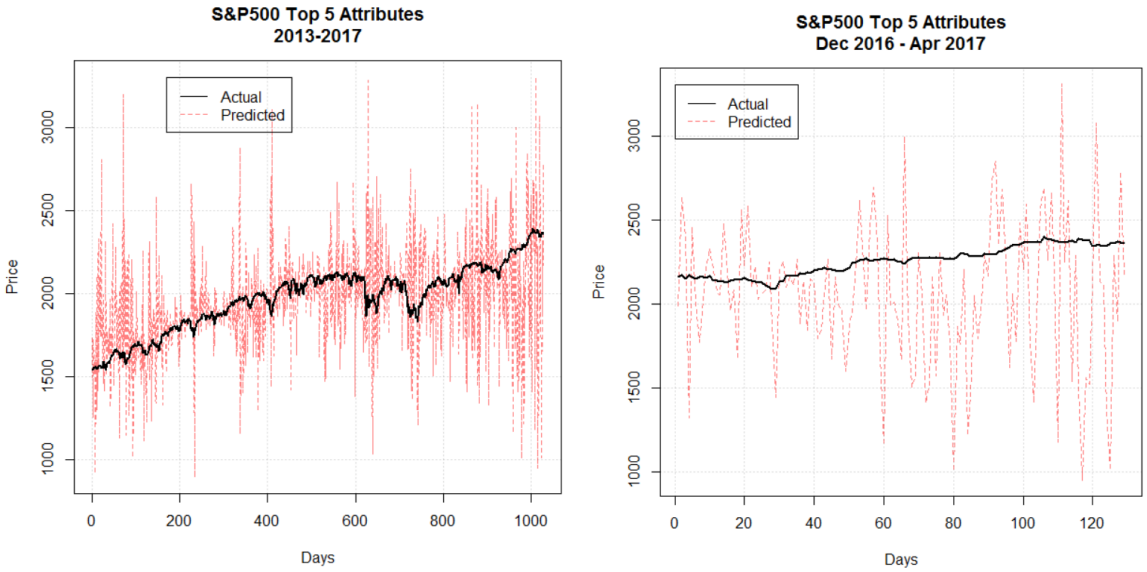

Figure 9: (left) Continuous prediction of S&P500 Data Using Top Attributes (right) Continuous prediction of 4 months of S&P500 Data Using Top Attributes

In analyzing the side by side boxplots of the two distributions of differences, we can see that the Full Model, with all 14 attributes, has the least variance within its distribution, thereby producing the more accurate predictions, on average.

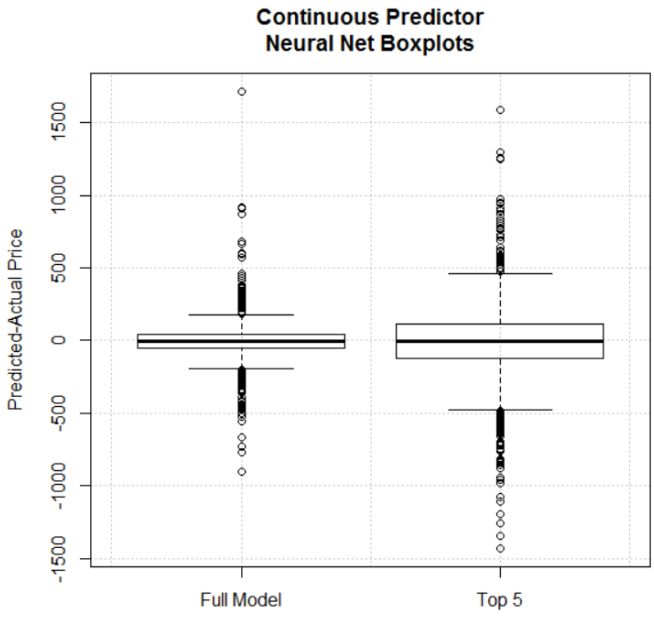

Figure 10: Boxplots of Continuous Prediction

9. Closing Thoughts

Neural Network analysis is a fascinating analytical tool, with deep roots in the field of traditional machine learning. Designed to operate like neurons in the brain, we can use it to effectively find associations between attributes of an analysis and derive knowledge about the underlying mechanics within a dataset. With our implementation, we were able to reach an accuracy of 62.47% with our R implementation and 88% with our Keras implementation. Though we were unable to reach the accuracy of our probit regression in our R implementation, we still were able to map meaningful connections between the attributes that allowed us to predict the trend of the S&P 500 within a mean difference of $4.33 using all 14 attributes. We also learned the power behind the functionality of keras, a software designed for Deep Neural Networks, which produced a much higher level of accuracy than our alternate method. Perhaps in future analyses, we base most of our methods on testing the limits of what keras has to offer.

We also were able to assess the most important attributes within our analysis using the random forest algorithm for attribute selection. This was pivotal in the earlier stages in determining which attributes to use in our analysis that would provide the most useful information. We also learned that filtering out any attribute with a positive value for mean decrease in accuracy, reduced the overall accuracy of our model. In our case, relying on only the most informative attributes produced highly erratic results, proving that in order to map a neural network to the sheer complexity of the stock market, one almost always requires more data.

Though we were able to reach a level of accuracy exceeding our initial expectations with keras, we acknowledge the fact that the attributes we decided upon in our model selection were by no means perfect, and that our timeframe of stock quotes was extremely narrow in comparison to the life of the economy. In fact, many online stock analysis sites state that predicting the stock market with high accuracy is a fruitless endeavor, and borders on the edge of gambling when the predictions are used to make large investments. The sheer complexity of human behavior is nearly incalculable with many modern techniques.

10. References

[1] "Neural Networks & Deep Learning": http://neuralnetworksanddeeplearning.com/chap6.html

[2] Yahoo Finance http://finance.yahoo.com

[3] Zhang, Luna M. "Genetic Deep Neural Networks Using Different Activation Functions for Financial Data Mining." 2015 IEEE International Conference on Big Data, pp.2849-2851, 2015.

[4] "Why We Can't Predict Financial Markets" https://hbr.org/2009/01/why-we-cant-predict-financial. 2009.

[5] "Apple's Stock Hit By Web Rumor" https://money.cnn.com/2008/10/03/technology/apple. 2008

[6] "Standard & Poor's 500 Index" http://www.investopedia.com/terms/s/sp500.asp

[7] "List of S&P 500 Companies" https://en.wikipedia.org/wiki/List_of_S%26P_500_companies

[8] "Nasdaq" www.investopedia.com/terms/n/nasdaq/asp

[9] "Volatility Index Data" http://finance.yahoo.com/quote/%5EVIX?ltr=1

[10] "Volatility Index" www.investopedia.com/terms/v/vix.asp

[11] "Adjusted US Dollar Index Data" http://finance.yahoo.com/quote/DX-Y.NYB/history?period1=1333857600&period2=1491624000&interval=1d&filter=history&frequency=1d

[12] "U.S. Dollar Index – USDX" http://www.investopedia.com/terms/u/usdx.asp

[13] "Oil Price Data" https://finance.yahoo.com/quote/OIL/history?period1=1333598400&period2=1491364800&interval=1d&filter=history&frequency=1d

[14] "Corn Futures Prices" https://www.google.com/finance/historical?cid=10946030&startdate=Apr+7%2C+2012&enddate=Apr+6%2C+2017&num=30&ei=F5_lWJCSFMHCeKbxlAg

[15] "Average True Range" http://www.investopedia.com/articles/trading/08/average-true-range.asp

[16] "The R Project for Statistical Computing" https://www.r-project.org/

[17] "R Package 'neuralnet'" https://cran.r-project.org/web/packages/neuralnet/neuralnet.pdf

[18] ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION Diederik P. Kingma* University of Amsterdam dpkingma@uva.nl Jimmy Lei Ba∗ University of Toronto jimmy@psi.utoronto.ca

[19] Practical Recommendations for Gradient-Based Training of Deep Architectures Yoshua Bengio Version 2, Sept. 16th, 2012

[20] "R Package 'randomForest'" https://cran.r-project.org/web/packages/randomForest/randomForest.pdf