Statistical Learning in Financial Markets

Thesis Draft

CDT Joseph Schlessinger
COL Grover LaPorte, LTC Tim Sikora, & CPT Steven Morse

Department of Mathematical Sciences
United States Military Academy
West Point, New York
February 27, 2019

Contents

Abstract

The day-to-day changes in the stock market are very difficult to predict; as such, many amateur investors rely on simple techniques that make no effort to time the market, such as dollar cost averaging. The goal of this project was to outperform dollar cost averaging by using an investment strategy informed by stock changes predicted by statistical learning. Our model focused on predicting the performance of a single stock. Our model was generally unsuccessful in predicting day-to-day changes, but was successful under certain cases. It appears that the complexities of the stock market cannot be modeled by basic linear models, but more complicated neural networks did show some progress. While modeling a regression proved to be very challenging, it was possible to get reasonable accuracy by modeling it as a classification problem. Moreover, the informed investment strategy was able to outperform dollar cost averaging in stagnant and bearish markets. As long as the stock market remains very difficult to predict, dollar cost averaging remains a simple yet effective investment strategy.

1 Introduction

There is an old saying that history has a way of repeating itself. Many believe this holds in the stock market. This project seeks to leverage this idea: based on the past performance of the stock market, can we predict what will happen next. We hope to create an informed investment strategy to beat dollar cost averaging, a common investment strategy explained later, by applying traditional time series analysis as well as newer methods of machine learning.

In the following report, we present a literature review encompassing the basics of investing, traditional time series analysis and machine learning. All of these broad topics play a central role in this project.

We follow with an overview of the methods used in this project and the results. Finally, we present a plan for future work.

2 Literature Review

2.1 Investment Basics

Investing is a complicated subject. There are a number of different types of financial assets, trades, investment strategies, and risk measurements. For the purposes of this project, we will simplify financial markets significantly by focusing on a single type of security, stocks.

2.1.1 Stocks

Stocks are the most basic element of the financial market. By owning a stock, in some sense you own part of a company. A public company is divided into shares through an initial public offering. If you own one share of Google, you own some small but measurable percent of that company.

A stock is a piece of some public company. A public company means it is traded publicly; anyone may purchase a share giving them partial ownership of the company. The stock market is open from 8:00 AM to 4:00 PM, at which point differing volumes of trading occur.

How is the market price set? Most people have a vague idea of what determines the price of a stock at any given time: supply and demand. More precisely, the stock price is determined automatically by the exchange (e.g. NYSE) through a constant auction-like process consisting of "asks" and "bids." An ask is a potential seller dictating a price that they are willing to sell at. A bid is a buyer dictating a price that they are willing to buy at. A computer then works to facilitate trades between all of the asks and the bids. The most recent price of a transaction is then the market price. This has profound effects on predictability, since a single stock price is a result of millions of decisions and a lot of human behavior; namely, traders speculation informed by a variety of factors drives decisions. [Ken]

Different Markets Explored. Below, I have summarized the markets explored throughout the project.

- Bullish : in this market, the stock value increases.
- Bearish : in this market, the stock value decreases.
- Noisy neutral : in this market, the stock value ends at the same point it starts. Between then, the stock both moves up and down.

Although our study focuses on stocks, there are a number of other financial assets that make up the complicated financial markets. I have summarized a few of the most prominent ones below:

- Bonds : A bond is like a share of debt. When an investor makes a loan to a borrower, the investor can divide up the loan among bonds, which others can then purchase and receive the interest from payback of the loan.
- Commodities : Commodities are not strictly an investment term; they refer to anything that is an input in the production of something else. Some examples are oil, gold, and natural gas. Investors can trade on commodities by betting on whether the cost of such goods will go up or down.
- Options : A stock option gives an investor the option to buy or sell a stock at a certain price. A put option is a bet that the stock will fall or the right to sell at a certain price. A call option is a bet that the stock will rise or the right to to buy a stock at a certain price.
- Exchange-traded funds (ETFs) : An ETF tracks a stock index, commodity, bonds, or some combination of assets. Shares of ETFs are traded on an exchange like stocks.
- Mutual funds : A mutual fund is a pool of money collected for the purpose of investing in securities like stocks and bonds. The fund has a manage who allocates the funds money among different investments.

2.1.2 Dollar Cost Averaging

Dollar cost averaging (DCA) is a popular investment strategy. It comes from the wisdom that the stock market is unpredictable; in other words, the investor has no way of knowing whether the stock value will increase or decrease. By investing some fixed amount at a regular interval, the investor averages out the risk and will gain, on average, whatever the stock makes. This phenomenon can be seen in Figure 1. Dollar Cost Averaging is the standard of comparison for investment strategies.

Example. Consider you want to invest in some stock X and you have a principal of \$12,000. If you were using DCA, you would pick some interval to invest at. If you decided on an interval of one month over a year, you would invest \$1,000 at the start of each month. However, on months where the stock is lower in price, you will buy more shares and when the stock is higher in price, you will buy fewer shares. The alternative is investing all of your money on the first day of the first month. DCA results in a lower average purchase price. Given the following equation for profit

$$\text{profit} = \text{number of shares } * (\text{current price} - \text{purchase price})$$

it is obvious that a lower purchase price will increase profit. Figure 1 depicts this phenomenon. Obviously, the average purchase price with DCA is lower because the stock decreases after the first month. Note, DCA would be suboptimal if the stock only increased after the first month; in this case, it would be better to invest as a lump sum. Still, the unpredictability of the stock market and its tendency to fluctuate usually leads to DCA performing better than lump sum investing.

With its simplicity, independence of market trends, and performance, dollar cost averaging is often used as the standard of comparison for investment strategies.

2.2 Time Series Analysis

Stock data examined over time forms a time series. Explicitly, we have some series $\{X_t\} = \{x_1, x_2, \dots, x_n\}$ where x_t is some vector corresponding to the tth day of a stock. We are concerned with forecasting the price of a stock for the next day. Given $\{X_t\}$, can we predict x_{t+1}?

2.2.1 Stationary Models

A time series is stationary if it has similar statistical properties at $\{X_t\}$ and at $\{X_{t+h}\}$ for some h. For a more mathematically rigorous definition of stationary, we need to define some additional functions.

The mean function of $\{X_t\}$ is $\mu_X(t) = E(X_t)$. This refers to the average value of the time series or stock. The first requirement for stationarity is that $\mu_X(t)$ be independent of t. In the context of stocks, the time series around day 0 should have the same mean as the time series around day n. This rarely happens with financial data, which will be addressed later.

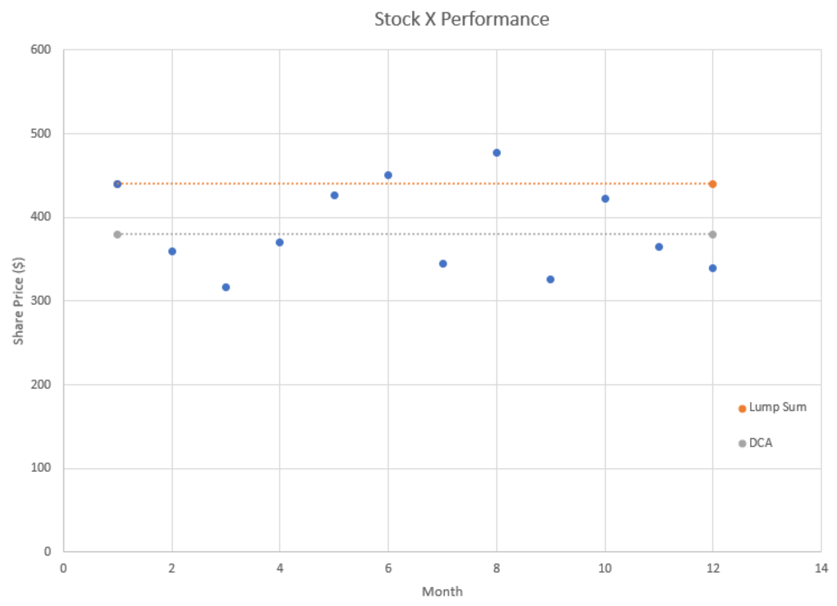


Figure 1: Stock price at the beginning of each month is shown in blue. The orange line shows the average purchase price for a stock if it is invested as a lump sum at the beginning of the investment period. The gray line shows the average purchase price if the principal is divided up and invested at the beginning of each month according to dollar cost averaging with an interval of one month. [Kan]

The second requirement has to do with the covariance of the time series. The covariance function of $\{X_t\}$ is defined as

$$\gamma_X(r,s) = \text{Cov}(X_r, X_s) = E[(X_r - \mu_X(r))(X_s - \mu_X(s))]$$

Our next requirement for a time series to be stationary is that $\gamma_X(t+h,t)$ is independent of t for each h.

2.2.2 Trends, Seasonality, and Noise

Stock data as a time series is not stationary. This should be obvious. A traditional way to decompose a time series is into a trend component, a seasonal component, and some random noise component. Thus, a time series can be decomposed in the following way

$$X_t = m_t + s_t + Y_t \qquad \text{additive decomposition}$$

where m_t is the trend component, s_t is the seasonal component, and Y_t is some random stationary noise component. In Figure 2, these components are on display. The plot titled purely random error is our random noise component; this is a stationary time series. The plots depict two types of trend, linear and nonlinear. A trend refers to some change to the data that occurs over time. As stocks generally increase overtime, there is clearly some trend involved in stock time series. Additionally, time series can have seasonality, seen in the bottom left of the figure. Anything that is cyclic can be represented as seasonality. As an example, if a stock increased at the beginning of the month, peaked in the middle, and then decreased before bottoming out at the end of the month, this would be a seasonality we would have to negate before applying time series analysis. The plot in the bottom right demonstrates an interaction of these additive terms, which is generally what financial data looks like. [BD96, 22]

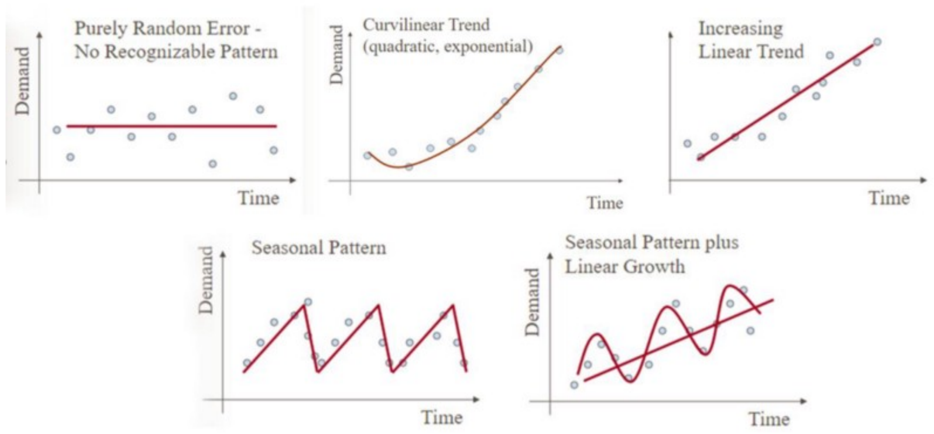


Figure 2: This figure shows the various components of a time series. [Rei]

2.2.2.1 Trend and Seasonality Elimination by Differencing

Many time series analysis models require a stationary time series. While we can account for this by attempting to estimate the trend and seasonality of the time series, in this case we will transform the data to eliminate trend and seasonality.

To eliminate trend, you can apply the difference operator ∇^d which lags the time series data by the past d days.

$$\nabla^1 X_t = X_t - X_{t-1} = (1 - B)X_t$$

where B is the backward shift operator such that $BX_t = X_{t-1}$.

Thus,

$$\nabla^t X_t = \nabla(\nabla(X_t)) = (1 - B)(1 - B)X_t = (1 - 2B + B^2)X_t = X_t - 2X_{t-1} + X_{t-2}$$

This applies for some general ∇^d. A similar technique can be used to negate seasonality by applying ∇_d, where subscript d instead refers to lagging by the value in the time series d periods ago.

$$\nabla_d = X_t - X_{t-d} = (1 - B^d)X_t$$

A more rigorous explanation can be found in [BD96, 22-32].

2.2.3 Moving Average

Moving average models use the past forecast errors to model the time series and forecast.

$$x_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}$$

θ_q represents the weight for ε_q. An MA(q) model considers the q most recent data points. [HA09, 8.4]

2.2.4 Autoregressive Models

Autoregressive models use past values of the variable to predict future values of the variable. With financial data, the implication in using autoregressive models is the stock market repeats itself, and we can learn from the past what may happen in the future. [HA09, 8.3] The autoregressive model can be written as

$$x_t = c + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p} + \varepsilon_t$$

where ε is white noise, c is some intercept and ϕ_n is the weight associated with data point y_n. An AR(p) model considers the most recent p data points in the time series.

2.2.5 ARIMA

AutoRegressive Integrated Moving Average (ARIMA) combines moving average and autoregression into one model.

$$x'_t = c + \phi_1 x'_{t-1} + \cdots + \phi_p x'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

x'_t refers to the differenced series. This comes from the "integrated" part of the model; it is differenced. The general ARIMA model is of form ARIMA(p, d, q) where p is the order of the autoregressive part, d is the degree of differencing, and q is the order of the moving average part. [HA09, 8.5]

3 Data

All data for the project comes through the Yahoo! finance API. At the beginning of this research, we pulled all of the available data from each stock in the S&P 500. The amount of data varies by stock. Older companies like GE have data going back to the late 1950s whereas newer companies like Google go back to the early 2000s. The code for this can be found in Appendix A.

3.1 Features

The available data for analysis is summarized below:

- Close price - for a given day, this is the price that the stock ended at.
- Open price - for a given day, this is the price that the stock started at.

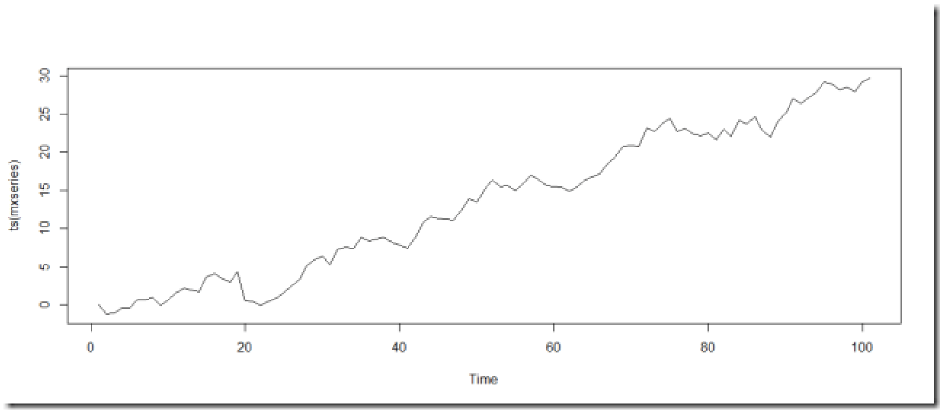


Figure 3: Here is a time series with a clear upward trend. Imagine we trained a model on this data from 0 to 50 seconds and used 50 to 100 seconds as our test set. The inputs for our test set would be data values that the model had not trained on because the upward trend means the magnitude of the data increases over time. Differencing removes this issue by effectively recasting each data point in the context of the data preceding it.

- Volume traded - for a given day, this is how many shares were traded
- High price - this is the daily high for the stock
- Low price - this is the daily low for the stock

3.2 Time Lag

The input vector was lagged at variable lengths. The number of days to lag was one of the many hyper-parameters in this process.

3.3 Differencing

Stock data has a noticeable trend, and it is difficult to train a model on data with a trend. See Figure 3 for a further explanation of the need for differencing.

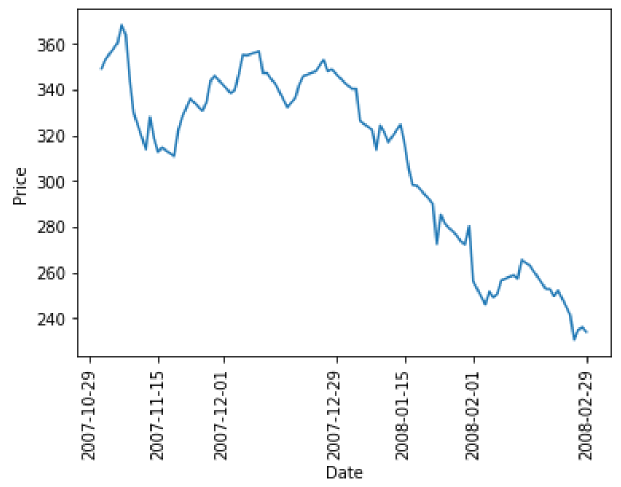
We used a time differencing of one, which was sufficient and optimal for stationarity. A before and after of differencing can be found in Figure 4.

3.4 Augmenting with Additional Stocks

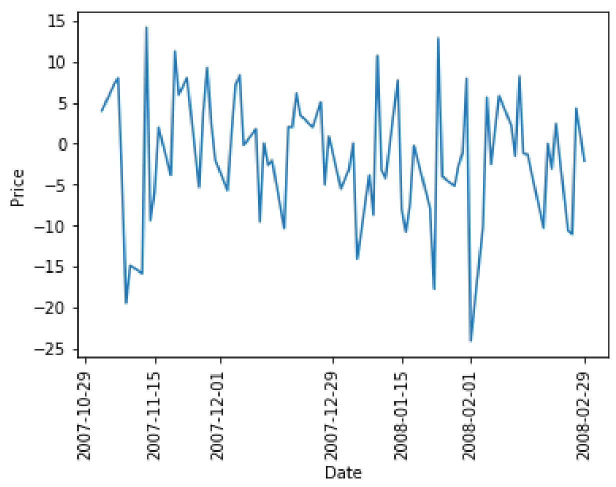
One of our intuitions was the correlation of stocks in the same sector or otherwise similar markets might be useful for our stocks. As such, our input vector also contained the lagged data for other stocks, to include closing price, opening price, volume traded, high and low.

4 Methods

The research framework can be summarized in the following few steps:



(a) The time series data without any differencing.



(b) The time series data after a differencing with time delta 1.

Figure 4: This figure highlights the effect of differencing on stationarity.

1. Using stock data, we build an input vector.
2. We choose a machine learning model to train on all available data up until the beginning of the investment period.
3. Using the trained model and the available data, predict the stock price of the next day.
4. Given the model's prediction, decide to sell, buy or hold according to a strategy.

4.1 Machine Learning

Machine learning is a type of spplied statistics that is often associated with mysterious terms like neural networks, artificial intelligence, and deep learning. However, it is important to note that these methods of machine learning are simply statistical models just as more rudimentary methods like linear regression are.

4.1.1 Regression v. Classification

Machine learning, in the context of prediction, breaks down into two main subsets: regression and classification. The distinction between the two subsets lies in the output type. Both share the same input of some set of data $\{(x_i, y_i)\}$. With regression $y_i \in \mathbb{R}$ whereas with classification, $y_i \in \{0, \ldots, N\}$, where N is the number of classifications or categories there are. [HTF01, 9-10] When looking at stock time series data, a regression model would seek to forecast what the price of the stock will be the following day, while a classification model might seek to determine whether the stock will go up, down or stay the same.

4.1.2 Linear Models

Linear models are one of the simplest approaches to prediction. Given a vector of inputs $X = (x_1, x_2, \ldots, x_n)^T$, a linear model predicts output by assuming the data is well predicted by a linear equation of the form

$$\hat{Y} = \hat{\beta}_0 + \mathbf{X}^T \hat{\beta}$$

where $\hat{\beta}_0$ is the intercept for the data and $\hat{\beta}$ is the vector of coefficients for the input matrix X. [HTF01, 11]

We can estimate the coefficients (or "weights") using the method of least squares, which seeks to minimize the residual sum of squared error.

With least squares, we select $\hat{\beta}$ as the solution to the following optimization problem:

$$\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{N} \left(y_i - \beta_0 - \mathbf{X}_i^T \hat{\beta} \right)^2$$

[HTF01, 42] This is a convex optimization problem and the solution can be written in closed form.

Ridge Method

One problem that can occur with least squares regression is poor prediction accuracy because of high variance. [HTF01, 55] This is also known as over-fitting. A solution is to shrink the size of the regression coefficients by imposing a penalty on their size. This is a simple modification of the least squares coefficient selection where we determine the coefficients according to the following formula:

$$\hat{\beta}_{\text{ridge}} = \text{argmin} \sum_{i=1}^{N} (y_i - \beta_0 - \mathbf{X}_i^T \hat{\beta})^2 + \gamma \sum_{j=1}^{p} \beta_j^2$$

[HTF01, 59]

Notice, the only difference between ridge and least squares is the penalty term for coefficient size, which is controlled by the tuning parameter γ. A large γ will place a large penalty on parameter size. As γ increases, the coefficients approach 0. Also, note that the following penalty represents the L_2 norm of β and it does not include β_0:

$$\sum_{j=1}^{p} \beta_j^2$$

LASSO Method

The LASSO (least absolute shrinkage and selection operator) method is another variant of least squares that employs a slightly different penalty. Coefficient selection goes according to the following equation:

$$\hat{\beta}_{\text{lasso}} = \text{argmin} \sum_{i=1}^{N} (y_i - \beta_0 - \mathbf{X}_i^T \hat{\beta})^2 + \gamma \sum_{j=1}^{p} |\beta_j|$$

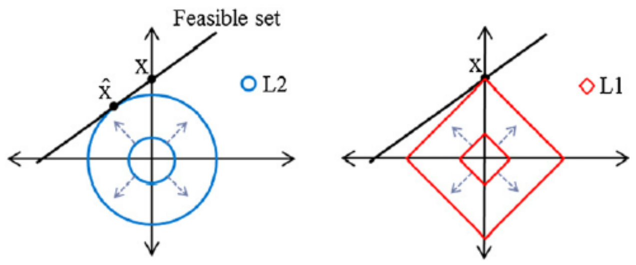


Figure 5: On the left, we have the L_2 norm used in Ridge. On the right we have the L_1 which is used in LASSO. The effect of using the L_1 norm leads to a greater likelihood of the coefficients β_j going to 0.

With lasso, the penalty is $|\beta|$ instead of β^2 (the L_1 norm instead of the L_2 norm). One effect of using this different penalty is a sufficiently large γ will lead to some coefficients equaling zero. The lasso method thus does variable selection in addition to fitting a linear model. [HTF01, 64]

Figure 5 explains the difference between the norms used as penalty in Ridge and Lasso. When optimizing the function, it is likely to intersect the L_1 norm on the axis, as seen on the right of Figure 5, which leads to a coefficient value of zero. However, the function being optimized is more likely to intersect the L_2 norm somewhere not on the axis, giving a nonzero value for the coefficient

4.1.3 Neural Networks

Neural networks are a type of nonlinear statistical model. Neural networks have three parts: an input layer, which takes the same vector X, a hidden layer which performs an activation function, and an output layer. Assume we have I input nodes, J hidden layer nodes, and K output nodes. The following explanation corresponds to Figure 6.

Each hidden layer node u_j learns a vector of weights α_j and some bias α_{0j}. Bias in neural networks is simply some intercept for the vector of weights α_j. Each weight α_{ji} corresponds to an input node X_i. Note, the hidden layer captures certain features of the model, but the weights and values do not have an obvious meaning. It is for this reason that neural networks are mysterious and often thought of as a black box. The value of u_j is simply some function f applied to a linear combination of the weights and input values:

$$u_j = f(\phi_j + \sum_{x=0}^{i} \alpha_{ji} X_i)$$

f is referred to as the activation function, which is often chosen to be the rectified linear unit (ReLU):

$$f(x) = max(0, x)$$

From this set of hidden layer values $U = \{u_j\}$, we can generate output.

Each output node u'_k has its own set of weights β_k for the hidden layer nodes as well as a bias μ_k. When generating output, the output node applies some function g_k to the linear combination of weights and hidden layer values.

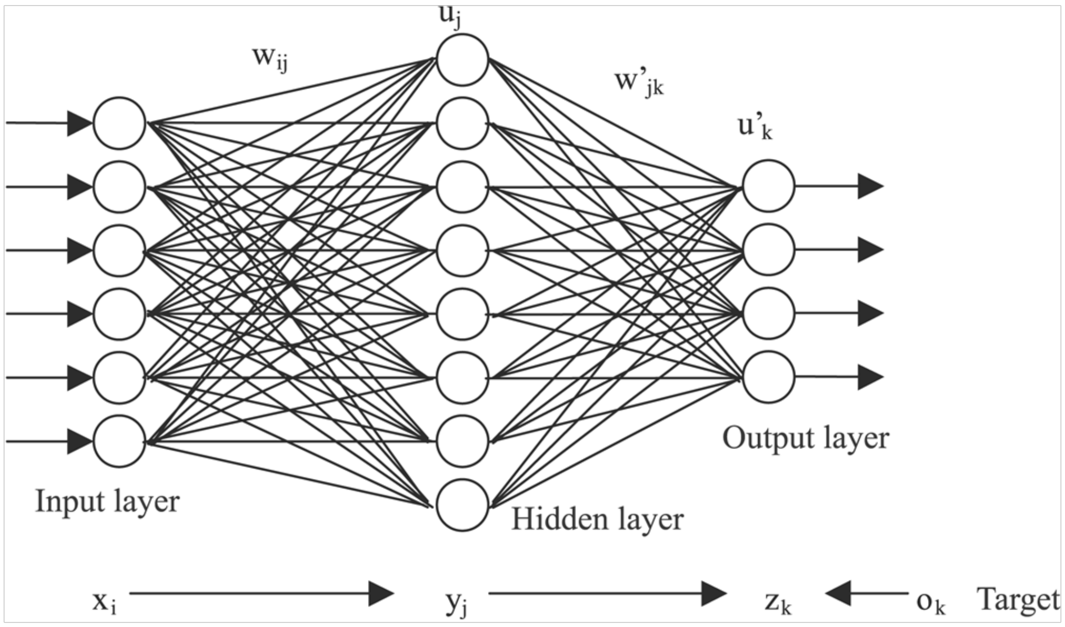


Figure 6: This is a visual depiction of a general neural net. Note, for regression, there is a single output node. The description in the text generally follows this diagram. [Tem]

$$u'_k = g_k(\mu_k + \sum_{i=0}^{J} \beta_j i u_j)$$

For regression, the output function is typically the identity function, so $g_k(a) = a$. With regression, there is also typically only one output function. It follows that g_k is the models prediction of output for some input X. [HTF01, 350-351] For classification, on the other hand, the sigmoid function is often used as defined below:

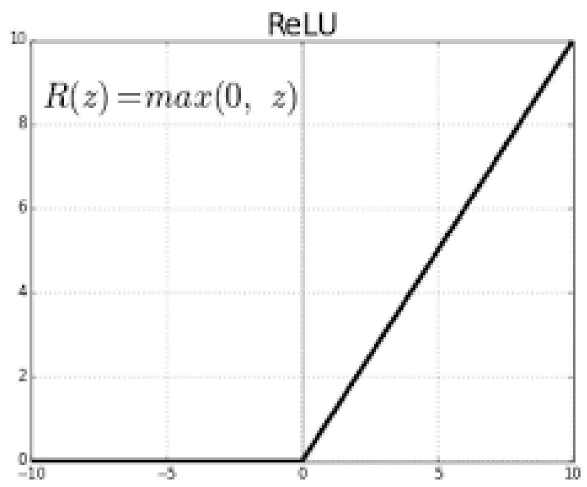
$$\sigma(v) = \frac{1}{1 + e^{-v}}$$

The sigmoid function simply takes an input v and maps it between 0 and 1, as seen in Figure 7b. For a binary classification, it follows that values above .5 would be classified as a 1 and values below .5 would be classified as a 0.

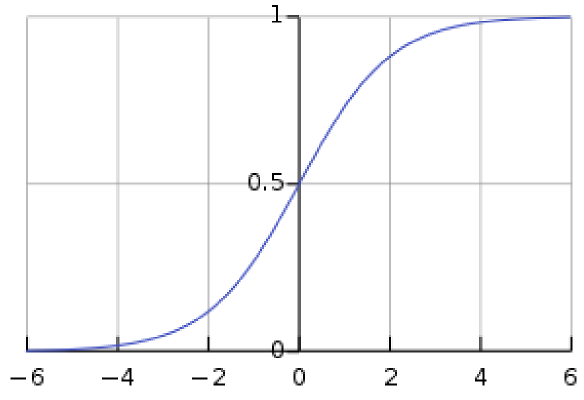
In summary, we take some input vector X. Each hidden layer node takes the linear combination of learned weights for each input node and the value of each input node and generates a number between 0 and 1. Finally, the output layer node computes the linear combination of hidden layer weights and the hidden layer values as the prediction for some input vector X.

4.1.4 Determining Coefficients

While we have outlined the general model for a neural network, there are some undefined parameters. These are the weights that the neural network "learns." We define the set of all weights as θ, which consists of



(a) The rectified linear unit is often used as the activation function in hidden layer nodes.



(b) The sigmoid function is often used as a function in output layer nodes.

$$\{\alpha_{0j}, \alpha_j : j = 1, 2, \ldots, J\}$$
$$\{\beta_{0k}, \beta_k : k = 1, 2, \ldots, K\}$$

With regression, we again use sum-of-squared errors to measure fit and determine coefficients. Note, this is the same measure of error used in least squares linear regression.

So, the squared error function for a neural network is

$$R(\theta) = \sum_{k=1}^{N} \sum_{i=1}^{N} (y_{ik} - f_k(x_i))^2$$

The task is thus to minimize $R(\theta)$ by back-propagation. It is less straight-forward to minimize this error function because it consists of many nested functions. Moreover, the activation function ReLU is not differentiable at 0, so it cannot be solved by simply taking the gradient. More information on back-propagation can be found in [HTF01, 354].

Using sum-of-squared errors leads to over-fitting in neural networks in the same way as least squares regression. Thus, neural networks can also include a penalty function in the same way that Ridge and Lasso do.

It is important to understand that many elements of the neural network are customizable You can have several hidden layers, you can add input nodes, hidden layer nodes, or output nodes. You can change the output function or the activation function.

4.2 Classification

Reformulating the problem as a classification changes the error function. We can no longer use squared error because are outputs are categories. Instead, cross-entropy is used and the following error function is minimized:

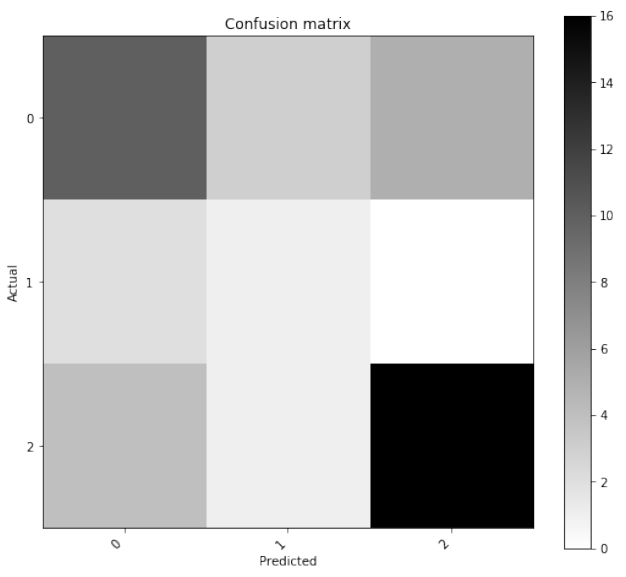


Figure 8: Here is the confusion matrix for our classification model. The cutoff between sell and hold was -0.2% yield and the cutoff between hold and buy was a 0.2% yieldsell.

$$R(\theta) = \sum_{k=1}^{N}\sum_{i=1}^{N}(y_{ik}\log f_k(x_i))^2$$

The challenge for reformulating the stock problem as a classification problem is tuning the categories so that the classifier can be successful in general cases. We settled on three classes: buy, hold and sell. We ran tests to try and find the optimal division in percent yield; in other words, what percent yield range should be a buy, sell, and hold?

4.2.1 Choosing Classes

We experimentally determined the best intervals for the classes by training the model on a number of different intervals and seeing which performed best. Initially, we looked at the confusion matrices and the accuracy rate. We determined that the following mapping worked well, where y is the percent yield:

$$\text{classify}(y) = \begin{cases} \text{sell} & y \leq -.2 \\ \text{hold} & -.2 \leq y \leq .2 \\ \text{buy} & .2 \leq y \end{cases}$$

The confusion matrix is depicted in Figure 8.

This is insufficient, however. Our overall goal is again the investment strategy. Overall accuracy may not be a tell-all because some mistakes are worse than others. Moreover, our investment strategy is not a direct reflection of the accuracy of the model. It is possible that a less accurate model holistically could give better results for the strategy.

4.2.2 Cross Validation

Often the challenge with learning through statistical models is a shortage of data. We first have to partition this data set into a training and test set, as depicted in Figure 9. Additionally, there are a number of parameters that need to be chosen, but cannot be optimally solved for. These parameters are chosen through the cross validation process.

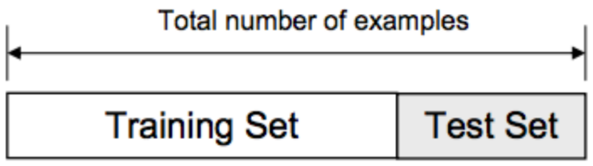


Figure 9: Training a model requires splitting data into a test set and a training set. [Bro]

This test set is untouched until the end when you evaluate the models performance on the data set. Thus, we already have a diminished data set when we are trying to train the model. There are useful methods to get the most out of your training set and tune hyper-parameters One such method is K-fold cross validation graphically depicted in Figure 10. K-fold could be used to select λ in Ridge regression, before training the model on the entire training set using the determined optimal λ.

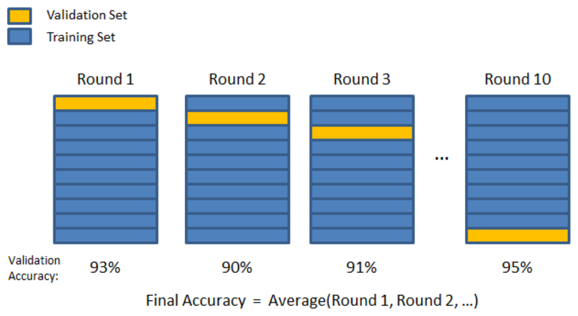


Figure 10: This figure depicts a K fold cross validation where $k = 10$. [Bro]

K-fold cross validation takes the training data and partitions it into K segments. It then trains the model on $K - 1$ segments, and evaluates the fit of the model on the remaining segment. It runs K rounds choosing a different validation segment each time. The average of the K rounds is taken as the score. The entire process of splitting data and validating can be seen in Figure 11.

4.3 The Strategy

The prediction engine is the important mathematical part of the project, but the investment strategy is similarly important because the final goal was to compare to dollar cost averaging.

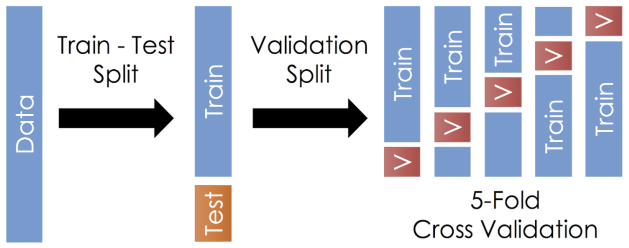


Figure 11: This figure depicts the entire process of training a model. It demonstrates a K-fold cross validation where $k = 10$. [Mar]

4.3.1 Practical Assumptions

In an effort to transform the financial world into a more reasonable object for academic research, we have made a few simplifying assumptions.

- Trades occur instantly and at the beginning of the day. The investment algorithm purchases at the exact price the stock closed at the previous day. This is unrealistic to some extent because in order to purchase some number of shares at some price, there has to be another owner willing to sell those shares at that price, which is not always the case. However, this assumption is consistent across both the statistical investment methods and dollar cost averaging standard of comparison.
- There is no transaction fee, which is not the case when trading. Typically, investors pay some fixed fee to the brokerage firm for completing the transaction. Again, because this is a consistent assumption in the dollar cost average baseline, this will not affect the comparison.
- We are assuming that the stock market has some degree of predictability. In particular, we assume that the future value of a stock can generally be predicted by the recent history of the stock.

4.3.2 Dollar Cost Averaging

The dollar cost averaging strategy is uninformed. Given an interval for investment, the dollar cost averaging strategy was programmed to divide the initial principal into the number of days of investment, and buy an equal amount each day of investment.

4.3.3 Yield Estimation Proportionality Investing

The basis of our investment strategy was to invest an amount of money that was proportional to the expected yield of the stock. If the stock is supposed to go up a lot, buy a lot. This is a notably short-sighted strategy, which has its pros and cons. In the case of the classification variation, the goal was to scale the investment amount with the classifer's confidence in its classification.

4.3.4 Short-Sighted Investment Strategies

The short-sighted nature of the investment strategy was founded in a number of assumptions summarized below.

- The goal of the project was to compete with dollar cost averaging, which basically follows the market. So, we knew it would be possible to beat dollar cost averaging in markets where the stock goes down or stays the same. Our strategy has the flexibility to sell stocks, while dollar cost averaging can only increase its stake. One consequence of the nature of dollar cost averaging is it is near optimal in a bullish market. In other words, when the stock is going up, dollar cost averaging is doing well because its position gets larger and larger. This leaves two options to outperform dollar cost averaging:

  1. Invest heavier than dollar cost averaging earlier in the investment period
  2. Capitalize on the local maximums and minimums in the stock.

  The first option was not possible because our method has some averaging as well; we reserve a maximum investment amount each day to avoid spending all the money early in the investment period. So, to beat dollar cost averaging, we would have to exploit the second option. These local maximums and minimums can be day to day; there is potential for earning every time the stock fluctuates. This was the motivation for the short-sighted strategy.

- Forecasting models typically get substantially worse as they are projected further out. Thus, we thought we would be most accurate at predicting the near future.

- The model, and limitations in compute power, forced us to focus on microtrends in the market, rather than macrotrends. In predicting the future, we would look on the scale of the past week or two, rather than the past few months. It would be difficult to project further out and capture long term trends with such a recent input vector.

- Lastly, we wanted to have a reasonably fair comparison. We tried to model the strategy after dollar cost averaging. If we gave our model the same amount of money to play with, but the option to invest less or potentially sell, would it be better than dollar cost averaging? It was difficult to make the investment strategy sufficiently comparable to dollar cost averaging specifically because our strategy had the unfair advantage of being able to sell. Any complexity in the strategy both made the comparison to dollar cost averaging difficult as well as confused the effects of the machine learning that was informing the model.

The strategy is summarized below.

1. Allocate some budget d out of your total principal to invest each day.[1]

[1] This is somewhat arbitrary. We found that without allocating a budget, the strategy invested a lot of money up front and did not invest later on. As our baseline for comparison is dollar cost averaging which invests a fixed amount at a certain interval, we allocated a daily amount to invest for a more equal comparison.

2. Estimate x_{n+1} using a model. Compute the estimated percent yield $\hat{p} = \frac{\hat{x_{n+1}} - x_n}{x_n}$. In the case of classification, convert the classification into a percentage of positive or negative confidence.

3. If $\hat{p} > 0$, purchase the following dollar amount worth of stock: $\alpha \cdot \hat{p} \cdot d$. α is a hyper-parameter that is set experimentally based on the most successful simulations.

   If $\hat{p} > 0$, then sell the following number of shares: $\beta \cdot \hat{p} \cdot$ shares owned

4.4 The Simulation

Below, we have listed the variables for the simulation:

- *stock* - the stock in the S&P 500 that will be modeled and invested in
- *timeframe* - the period of time in the past that the investment will take place during
- *principal* - the amount of money the simulation starts with
- *trainingData* - the amount of training data to consider. If left blank, the model will attempt to fit to all of the available data. Otherwise, it will used the preceding n days
- *validationFreq* - tells the model how often to retrain the model, reselect parameters, and cross validate.
- *parameterRange* - tells the model which parameter values to try when it is selecting through cross validation
- α - the tuning parameter for purchasing stocks
- β - the tuning parameter for selling stocks

5 Results

The models were compared based on a number of investment periods: bullish, bearish, and noisy neutral.

5.1 Regression

5.2 Classification

Ultimately, the performance of our strategy depended on certain hyper-parameters. The most important of these was the amount to invest. If the classifier predicted an increase or decrease, how much of the daily allotted amount should be spent. A more aggressive strategy was necessary to outperform dollar cost averaging in a bull market; however, this same aggressive strategy made our strategy perform worse than dollar cost averaging in a bearish market.

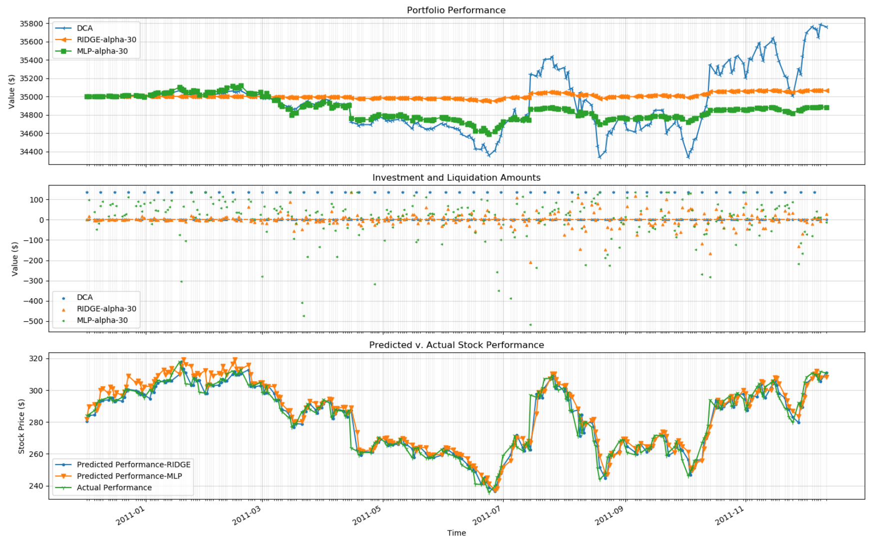


Figure 12: Here is an example of a time period in which the stock goes up slightly. Ridge and the multi-layer perceptron lose to dollar cost averaging, which mirrors the amrket that increases. Here, the regression based models clearly are unable to take advantage of small gains with basically no change.

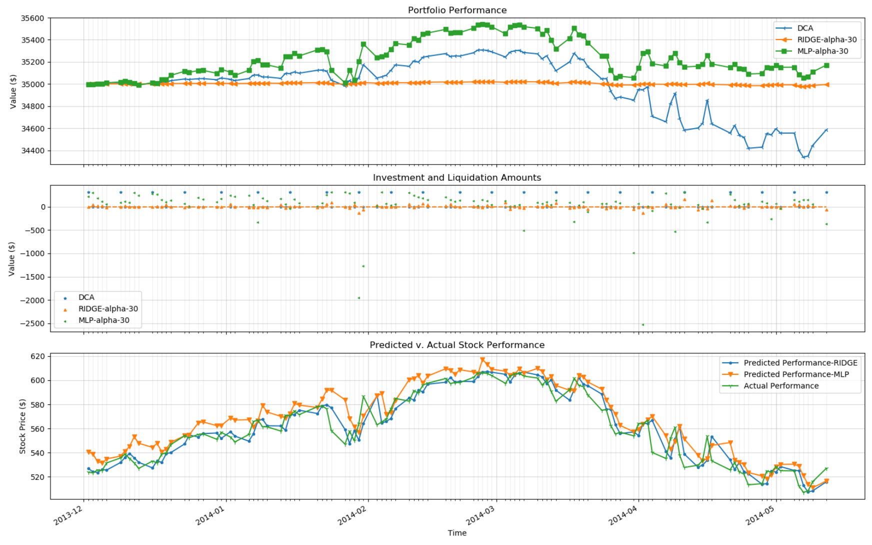


Figure 13: Here is an example of a noisy steady market. Dollar cost averaging performs worst here. Multi layer perceptron is able to take advantage of early rises, and by selling its position when the stock is going down, it is able to weather some of the markets fall.

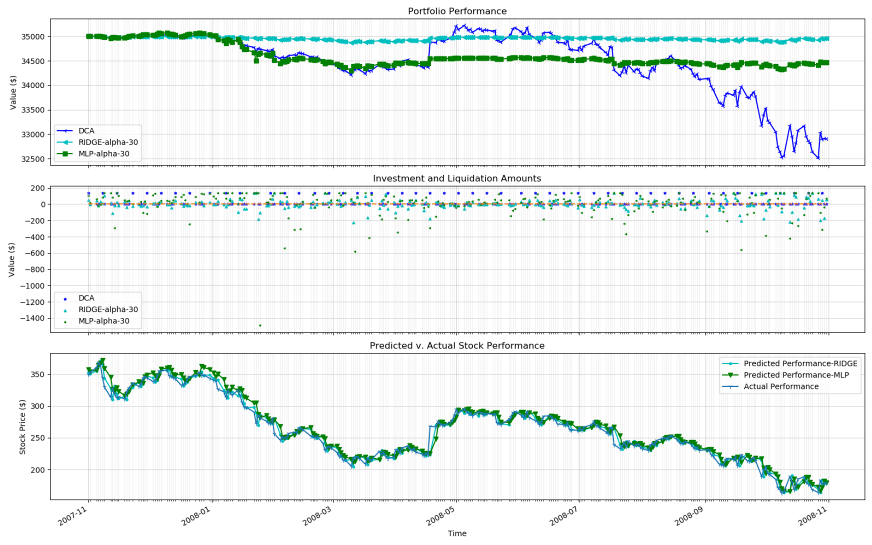


Figure 14: Here is a declining market. Dollar cost averaging consistently performs worse here because it is unable to sell and has a heavy position in a declining stock.

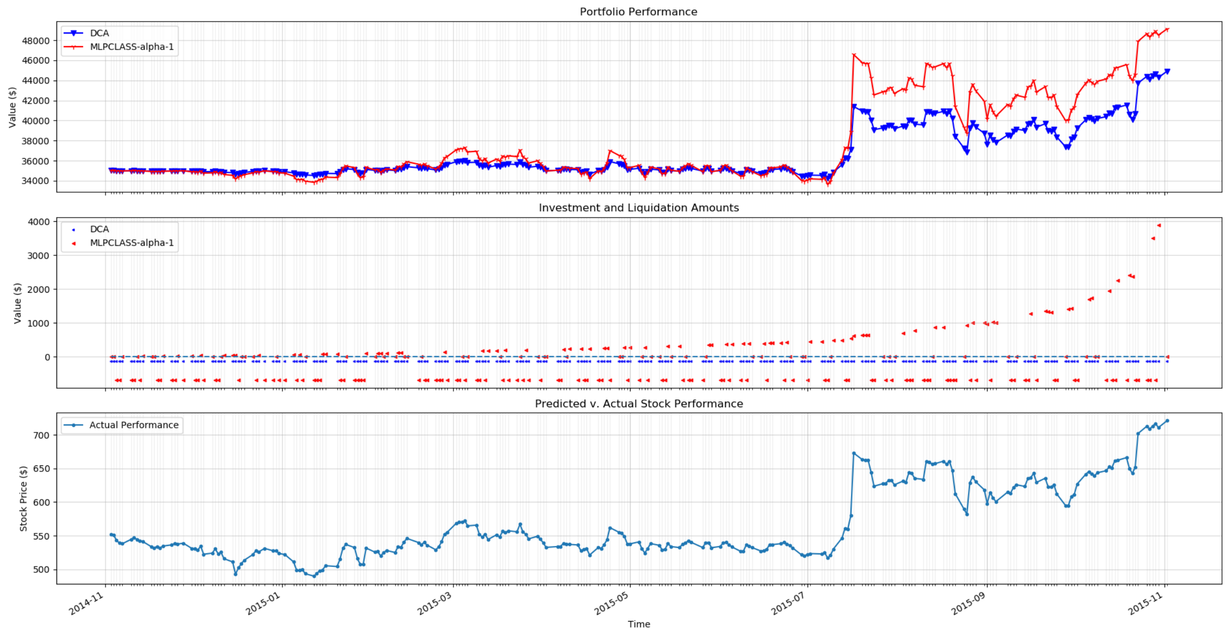


Figure 15: This shows the mlp classifier outperforming dollar cost averaging in a bull market, which was the goal at the start. Unfortunately, this result was not largely reproducible.

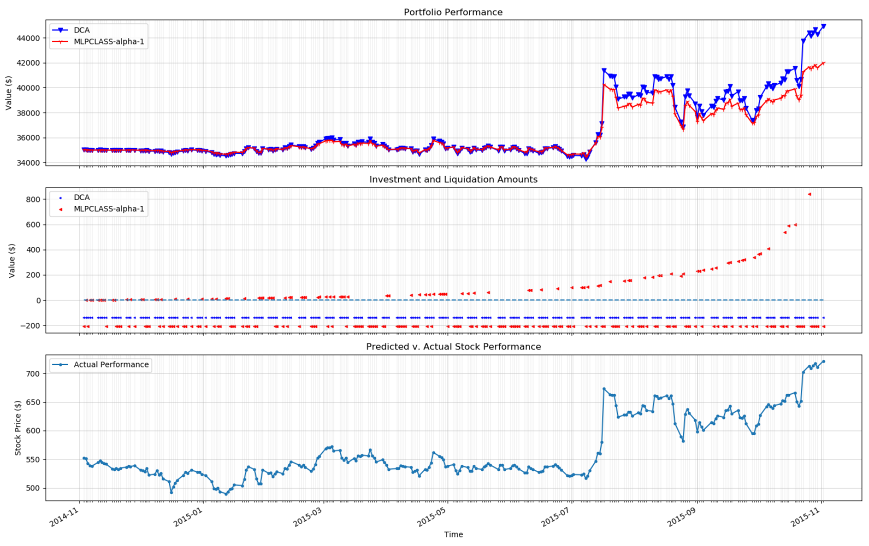


Figure 16: This shows the multi layer perceptron classifier in a bullish market losing to dollar cost averaging. This uses a more conservative investment strategies, where you invest or sell %30 of assets based on the classification.

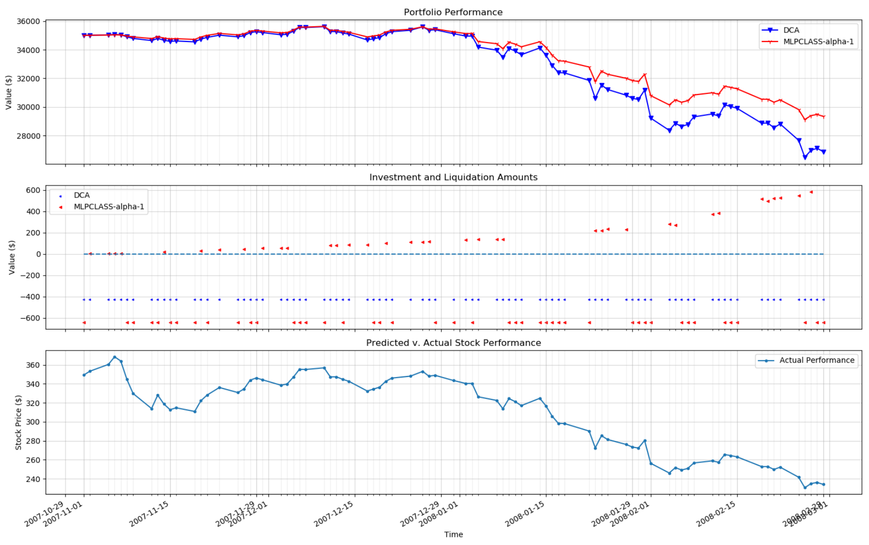


Figure 17: This shows a bearish market. With the conservative investment strategy, MLP outperforms DCA.

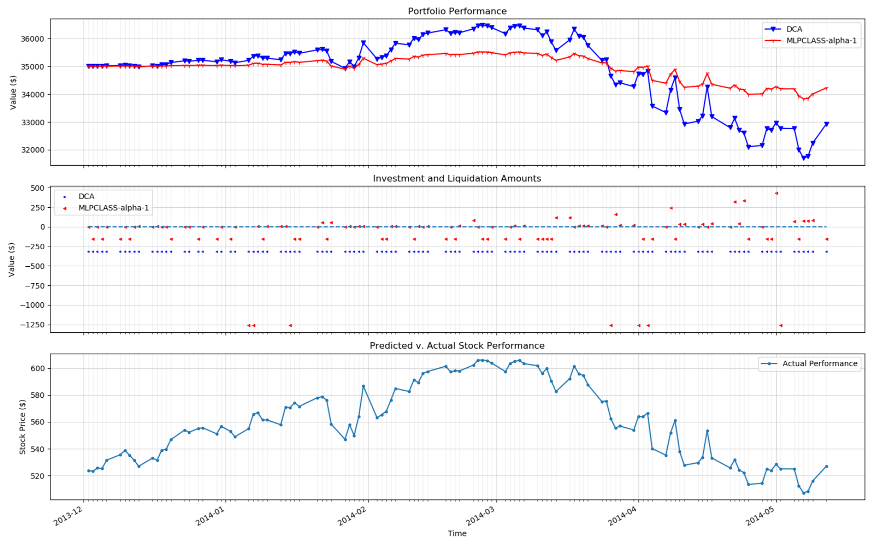


Figure 18: This shows the multi layer perceptron classifier run on Google's stock over a time period in which the stock started and ended at the same point. Both investment strategies lost money, but the classifier-informed strategy outperformed DCA by $10000 over the span of 5 months.

6 Future Work

TBD

A Pulling Data

```
# * coding: utf 8 *
"""
Created on Sat Oct 07 18:00:14 2017

@author: Grover.Laporte
"""

import re
import urllib.request
import calendar
import datetime
import getopt
import sys
import time

crumble_link = 'https://finance.yahoo.com/quote/{0}/history?p={0}'
crumble_regex = r'CrumbStore":{"crumb":"(.*?)"}'
cookie_regex = r'set cookie:␣(.*?);␣'
quote_link = 'https://query1.finance.yahoo.com/v7/finance/download/{}?period1

#link = crumble_link.format('KO')
#response = urllib2.urlopen(link)
#print str(response.info())
#print re.search(cookie_regex, str(response.info()))

def get_crumble_and_cookie(symbol):
    link = crumble_link.format(symbol)
    response = urllib.request.urlopen(link)
    print(str(response.info()))
    match = re.search(cookie_regex, str(response.info()))
    print(match)
    cookie_str = match.group(1)
    text = str(response.read())
    # print("text: ", text)
    match = re.search(crumble_regex, text)
    crumble_str = match.group(1)
    return crumble_str, cookie_str


def download_quote(symbol, date_from, date_to):
    time_stamp_from = calendar.timegm(datetime.datetime.strptime(date_from, "
    time_stamp_to = calendar.timegm(datetime.datetime.strptime(date_to, "%Y%
    attempts = 0
    while attempts < 5:
        crumble_str, cookie_str = get_crumble_and_cookie(symbol)
        link = quote_link.format(symbol, time_stamp_from, time_stamp_to, crun
        #print link
        r = urllib.request.Request(link, headers={'Cookie': cookie_str})

        try:
            response = urllib.request.urlopen(r)
            text = response.read()
            print("{}␣downloaded".format(symbol))
            return text
        except urllib.request.URLError:
            print("{}␣failed␣at␣attempt␣#␣{}".format(symbol, attempts))
            attempts += 1
            time.sleep(2*attempts)
    return ""
#python apiYahoo.py  symbol=IBM  from=2017 01 01  to=2017 05 25  o IBM.csv
#from ipython run apiYahoo.py  symbol=IBM  from=2017 01 01  to=2017 05 25
if __name__ == '__main__':
    print(get_crumble_and_cookie('KO'))
    from_arg = "from"
    to_arg = "to"
    symbol_arg = "symbol"
    output_arg = "o"
    opt_list = (from_arg+"=", to_arg+"=", symbol_arg+"=")
    try:
        options, args = getopt.getopt(sys.argv[1:],output_arg+":",opt_list)
    except getopt.GetoptError as err:
        print(err)

    for opt, value in options:
        if opt[2:] == from_arg:
            from_val = value
        elif opt[2:] == to_arg:
            to_val = value
        elif opt[2:] == symbol_arg:
            symbol_val = value
        elif opt[1:] == output_arg:
            output_val = value

    print("downloading␣{}".format(symbol_val))
    text = download_quote(symbol_val, from_val, to_val)
    with open(output_val, 'wb') as f:
        f.write(text)
    print("{}␣written␣to␣{}".format(symbol_val, output_val))

import csv
import sys
import time
symbols = []
industries = []
dates = []
with open("s&p500Cos.csv") as csvfile:
    data = csv.reader(csvfile, delimiter=',')
    #next(data) #skip header row
    for row in data:
        date = time.strptime(row[6], "%m/%d/%Y")
        dates.append(date)
        industries.append(row[3])
        symbols.append(row[0])

from subprocess import call
flag = False
for i in range(len(symbols)):
        if flag:
                call(["python", "apiYahoo3.py", "  symbol=" + symbols[i], "
        if symbols[i] == "MYL":
                flag = True
```

References

[BD96] Peter Brockwell and Richard Davis. *Introduction to Time Series and Forecasting.* Springer-Verlag, 175 Fifth Avenue, New York, New York, 1996.

[Bro] Adi Bronshtein. Train/test split and cross validation in python. https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6. Accessed: 2018-10-04.

[HA09] Rob Hyndman and George Athansopoulos. *Forecasting: Principles and Practice.* McGraw-Hill/Irwin, 1221 Avenue of the Americas, New York, New York, 2009.

[HTF01] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer-Verlag, 175 Fifth Avenue, New York, New York, 2001.

[Kan] Samudra Kanankearachchi. Time series forecasting in machine learning. https://medium.com/99xtechnology/time-series-forecasting-in-machine-learning-3972f7a7a467. Accessed: 2018-10-04.

[Ken] Joshua Kennon. How stock prices are determined. https://www.thebalance.com/how-stock-prices-are-determined-358144. Accessed: 2019-02-20.

[Mar] Hector Martinez. Train/test split, cross-validation and you. https://medium.com/@hi.martinez/train-test-split-cross-validation-you-b87f662445e1. Accessed: 2018-10-04.

[Rei] Jesper Reiche. Dollar cost averaging bitcoin and cryptocurrencies. https://blog.goodaudience.com/dollar-cost-averaging-bitcoin-and-cryptocurrencies-43815cf69b8c. Accessed: 2018-10-04.

[Tem] Graham Templeton. Artificial neural networks are changing the world. what are they? https://www.extremetech.com/extreme/215170-artificial-neural-networks-are-changing-the-world-what-are-they. Accessed: 2018-10-04.