Kleio Baxevani, Prem Chand, Desiderio Pilla
CISC 684 Machine Learning
Project Report

Title: Predicting Stock Market using Machine Learning Techniques

The Problem:

The problem we tried to solve in this project was to predict the trend of a specific stock (if it will increase or decrease) in the next month using the data from the previous year. To solve this problem, we use the following 7 attributes as inputs: Open, Volume, GISC Sector, Value, Price to Earnings Ratio, Debt to Equity Ratio, and Free Cash Flow Yield. Every attribute is a real value (except GISC Sector is a class). This is a Boolean classifier which returns 0, as an output, in case the stock price will decrease and 1 in case it will increase. For every instance, we will have to define the target classification based on the price of the stock one month prior. We can use a logistic regression or use a neural network to predict the classification based on these 7 attributes.

Datasets:

The datasets were chosen for this project can be found in the following link.
https://www.kaggle.com/dgawlik/nyse/download

The data was scraped from the internet for individual stocks. This includes daily price and trading data, as well as earnings data for every quarter since 1995.

The scraped data was manipulated to calculate a variety of value- and momentum-based metrics. For each trial, a testing date is given. The training data includes a 1 or 2 year period (chosen as a hyperparameter of the model) occurring at least *n* days before the testing date. The test data is comprised of the 6 month period following the testing date.
N corresponds the the classification of an instance. This data has a binary classification defined by whether or not the price of the security increased *n* business days after the current date. It is for this reason that there must be a gap of *n* days between the training and test data. This value is also selected as a hyperparameter of the model. Predicting the price of a stock 1 day into the future (which would practically yield the highest returns), is nearly impossible. It is unrealistic to do so given the feature spaced used in this project, as no geopolitical or media information was gleaned.

*Note: This model does not take dividend payouts into account, and all stock prices are split-adjusted.

Files:

Attributes

- Only the attributes that will be used are listed. The actual datasets contain more attributes.
- prices-split-adjusted.csv – date, symbol, open, close, volume
- fundamentals.csv - Ticker Symbol, Period Ending, Cash and Cash Equivalents , Common Stocks, Net Cash Flow-Operating, Total Assets, Total Liabilities, Earnings Per Share, Estimated Shares Outstanding
- securities.csv – Security, GICS Sector

The prices-split-adjusted.csv file has data for every stock with a distinct instance for each day.
The fundamentals.csv file has data for every stock with a distinct instance for each year (not starting on January 1st, but at some specified day every year).
The securities.csv file has data for every stock with a single instance for each stock.

The files were all joined such that each instance contains the daily prices-split-adjusted values, the most recent fundamentals values, and the securities data. Joins was on the date and symbol (for securities.csv, the "Security" label is the ticker symbol).

With the newly created dataset, the following attributes would be created for each instance:

- Value
- Price to Earnings Ratio
- Debt to Equity Ratio
- Free Cash Flow Yield

After these attributes are created, the only columns that will be saved are

- Open
- Volume
- GICS Sector
- Value
- Price to Earnings Ratio
- Debt to Equity Ratio
- Free Cash Flow Yield

Our data has 7 attributes: Open, Volume, GISC Sector, Value, Price to Earnings Ratio, Dept to Equity Ratio, and Free Cash Flow Yield. Every attribute is a real value (except GISC Sector is a class)

For every instance, we will have to define the target classification based on the price of the stock one month prior.

We can use a logistic regression or use a neural network to predict the classification based on the 7 attributes.

Derived Attributes:

- $Value = \frac{close}{\frac{TotalAssets - TotalLiabilities}{CommonStocks}}$
  - The value of a stock is the ratio between a single share's actual price and its valued price. A value greater than 1 indicates a share is trading above its inherent value, while a value less than 1 indicates a share is trading below its inherent value.
  - $Value = \frac{MarketValue}{BookValue}$
  - $BookValue = \frac{ShareholderEquity}{of commonshares}$
- $PriceEarningsRatio = \frac{close}{EarningsPerShare}$
  - The P/E shows whether a company's stock price is overvalued or undervalued compared to the earnings of a company. The value is almost always greater than 1, as earnings are expected to increase each year for a company. Higher P/E ratios indicate a stock is more expensive compared to other companies.
- $DebtEquityRatio = \frac{TotalLiabilities}{TotalAssets - TotalLiabilities}$
  - This ratio puts a company's debt into perspective. The smaller the value, the larger financial leverage a company has.
- $FreeCashFlowYield = \frac{Netcashflows(Operating) - CapitalExpenditures}{EstimatedSharesOutstanding * open + TotalLiabilities - Cash \wedge CashEquivalents}$
  - This is the amount of cash that a company has available to them. Higher values are desired.
  - $MarketCapitalization = EstimatedSharesOutstanding * open$
  - $EnterpriseValue = MarketCapitalization + TotalLiabilities - Cash \wedge CashEquivalents$
  - $FreeCashFlow = [Netcashflows(Operating)] - CapitalExpenditures$
  - $FreeCashFlowYield = \frac{FreeCashFlow}{EnterpriseValue}$

Algorithms:

This project utilizes two widely used algorithms for binary-class classification tasks. (i) Logistic Regression and (ii) Neural Network classifier (Multi-layer perceptron).

(i) Logistic Regression:

We use binary logistic regression to classify if the price of a security would increase or fall after n into the future. Here n is also selected a hyperparameter of the training model. Our predictProb function returns a score between 0 and 1. By selecting the threshold 0.5 we classify the predicted values in 2 classes. The goal of this algorithm is to solve an optimization problem which aims to minimize the cost associated with the misclassification of target values of the training data. Since we have a binary class classifier, we associate different cost for both classes and attempt to minimize a unified cost function by combining both cost functions according to target class values. We use gradient descent method to solve the optimization problem with some small learning rate. By running the gradient descent method for some number of iterations until the cost function decreases to a satisfactory level, then the learned weights of the classifier are readily used to predict the target class (0 or 1).

This algorithm can be summarized in a few steps:

Given: learning rate, dataset

Initialize: weights

Repeat until convergence:

$$l(\boldsymbol{\omega}) = \sum[y_i(\omega_0 + \textstyle\sum_j \omega_j \cdot x_{i,j}) - \ln(1 + \exp(\omega_o + \textstyle\sum_j \omega_j \cdot x_{i,j}))]$$

$$\frac{\partial\, l(\boldsymbol{\omega})}{\partial\, \omega_j} = \sum y_i x_{i,j} - \frac{x_{i,j}\exp(\omega_o + \sum_j \omega_j \cdot x_{i,j})}{1 + \exp(\omega_o + \sum_j \omega_j \cdot x_{i,j})} = \sum x_{i,j}(y_i - \Pr[y_i = 1|x_i])$$

for $j > 0$

- $\omega_j = \omega_j + \eta \frac{\partial\, l(\boldsymbol{\omega})}{\partial\, \omega_j}$
- η: learning rate

Return: weights

(ii) Neural Network classifier (Multi-layer perceptron):

Another algorithm used in this project uses the neural network to classify the binary class data. We use a single hidden layer in the structure of the neural network with sigmoid activation functions. Similar to logistic regression, neural network classifier also attempts to minimize the misclassification error. We again use the gradient descent method to solve the optimization problem by finding gradients of the cost function using backpropagation method. Again, suitable hyperparameters i.e. learning rate, number of iterations are explored for a particular dataset.

The neural network classifier can be summarized in the following pseudocode:

```
Given: learning rate, momentum, dataset
Initialize: weights, biases
Repeat until convergence:
        for all do
            (forward) calculate activations of hidden and output layers
            (backward) calculate deltas wrt hidden and output layers
            update weights using:
                $\frac{\partial E_x}{\partial \omega_i} = -(t_x - o_x)\frac{\partial o_x}{\partial \omega_i} = -(t_x - o_x) \cdot x_i \cdot o_x(1 - o_x)$
                $\omega \leftarrow \omega - \eta \frac{\partial E}{\partial \omega}$ (delta rule)
Return: weights
```

Results:

Amazon.com, Inc. (AMZN)

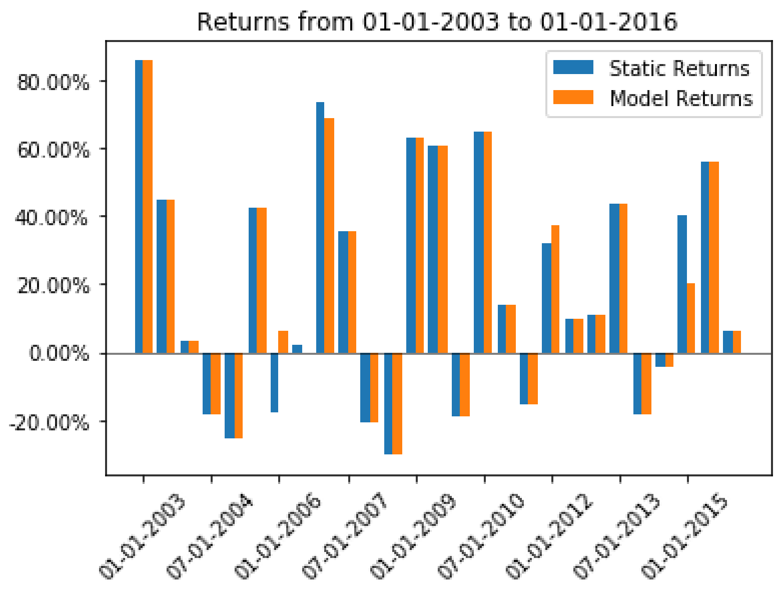
Static Returns: 30.55% / year

Logistic Regression Returns: 31.51% / year

Neural Network Returns: 21.95% / year

From the period 01-01-2003 to 07-01-2016, the total returns were:

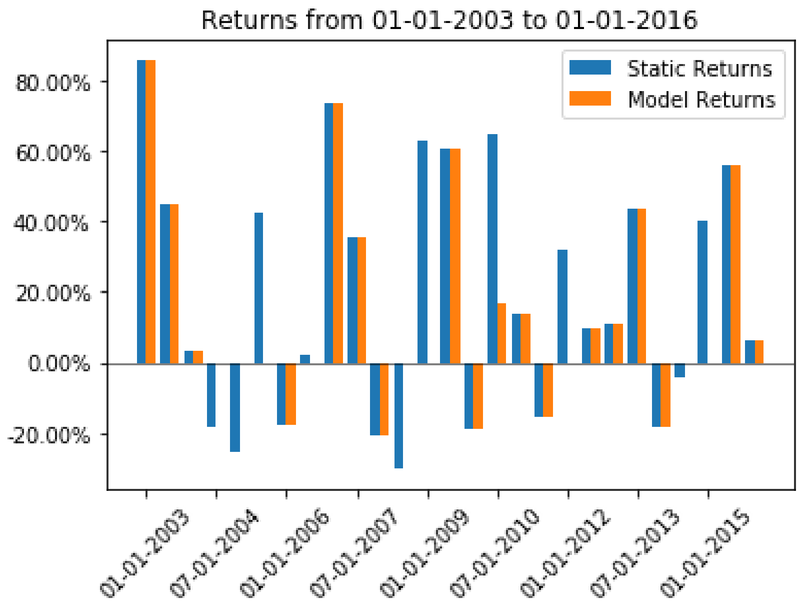
Static(real data) Returns : 3556.72% (30.55% /yr)
Model(Logistic Regression) Returns: 3936.21% (31.51% /yr)



Based on these results, an investor would have been better off **using the model to buy/sell the stock throughout the period.** The average accuracy of the model was 61% (min 30%, max 96%).

From the period 01-01-2003 to 07-01-2016, the total returns were:
Static(real data) Returns : 3556.72% (30.55% /yr)

Model(Neural Network) Returns : 1357.20% (21.95% /yr)



Based on these results, an investor would have been better off **holding onto the stock for the entire period.** The average accuracy of the model was 49% (min 0%, max 100%).

DuPont de Nemours, Inc. (DD)

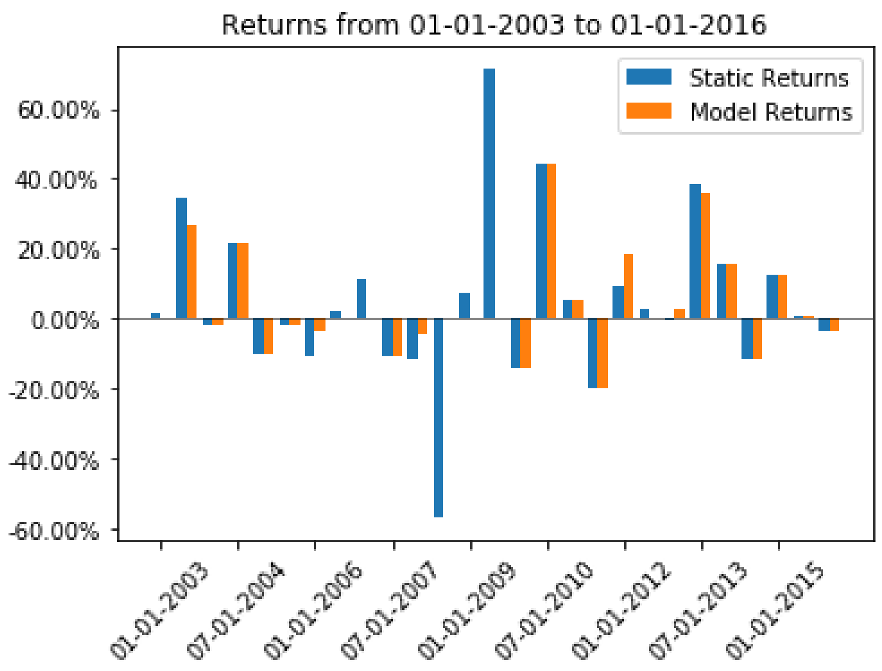
Static Returns: 3.69% /yr

Logistic Regression Returns: 5.66% /yr

Neural Network Returns: 7.93% /yr

From the period 01-01-2003 to 07-01-2016, the total returns were:

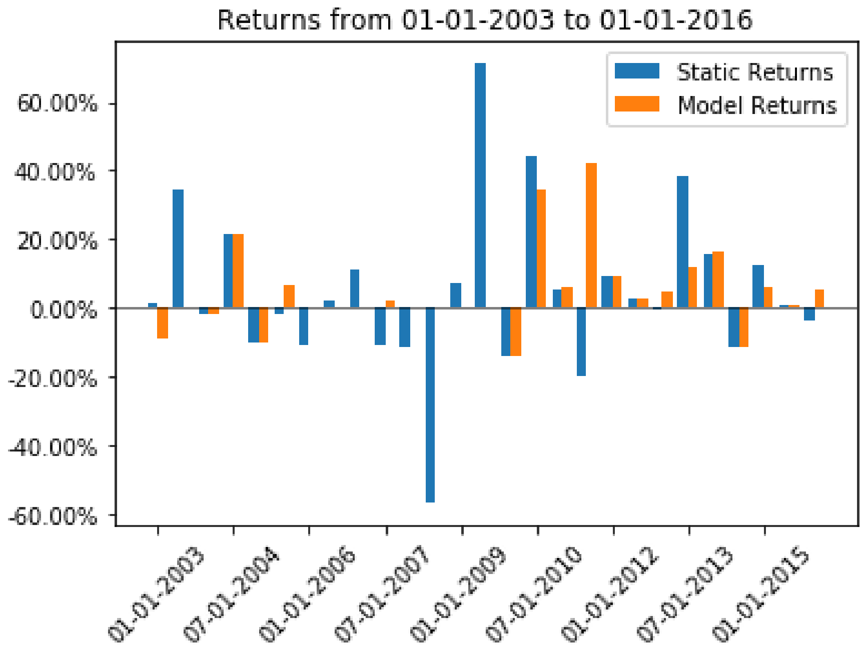
Static(real data) Returns : 63.20% (3.69% /yr)
Model(Logistic Regression) Returns: 110.36% (5.66% /yr)



Based on these results, an investor would have been better off **using the model to buy/sell the stock throughout the period.** The average accuracy of the model was 45% (min 2%, max 100%).

From the period 01-01-2003 to 07-01-2016, the total returns were:

Static(real data) Returns : 63.20% (3.69% /yr)
Model(Neural Network) Returns : 180.34% (7.93% /yr)



Based on these results, an investor would have been better off **using the model to buy/sell the stock throughout the period.** The average accuracy of the model was 51% (min 18%, max 86%)

<u>Exxon Mobil Corporation (XOM)</u>

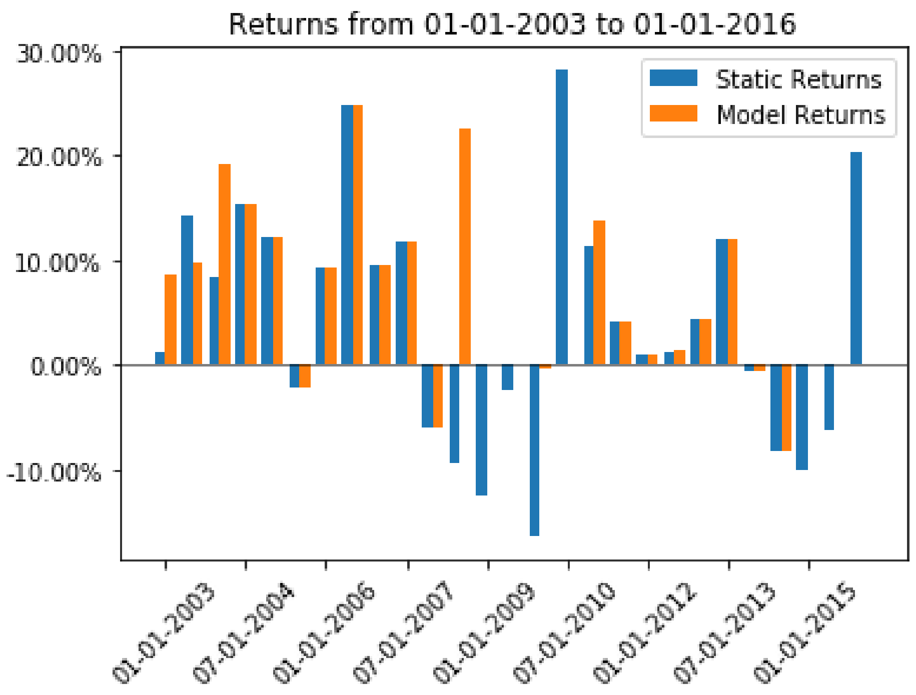
Static Returns: 7.46% / year

Logistic Regression Returns: 11.74% / year

Neural Network Returns: 13.47% / year

From the period 01-01-2003 to 07-01-2016, the total returns were:

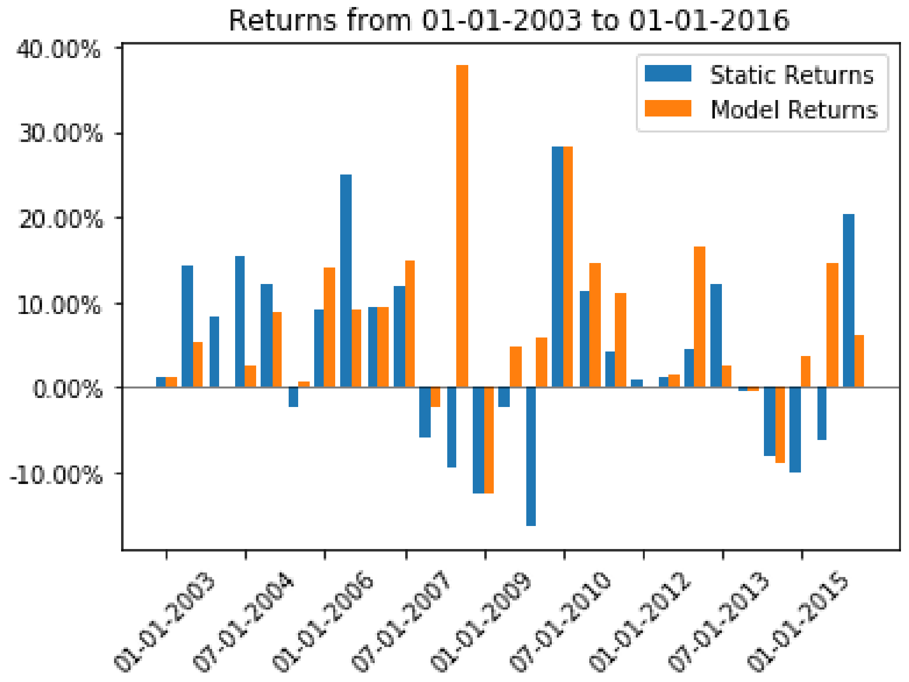
Static(real data) Returns : 164.28% (37.46% /yr)
Model(Logistic Regression) Returns: 347.67% (11.74% /yr)



Based on these results, an investor would have been better off **using the model to buy/sell the stock throughout the period.** The average accuracy of the model was 56% (min 9%, max 85%).

From the period 01-01-2003 to 07-01-2016, the total returns were:

Static(real data) Returns: 164.28% (7.46% /yr)
Model(Neural Network) Returns : 450.48% (13.47% /yr)



Based on these results, an investor would have been better off **using the model to buy/sell the stock throughout the period.** The average accuracy of the model was 55% (min 14%, max 91%)

IBM Corporation (IBM)

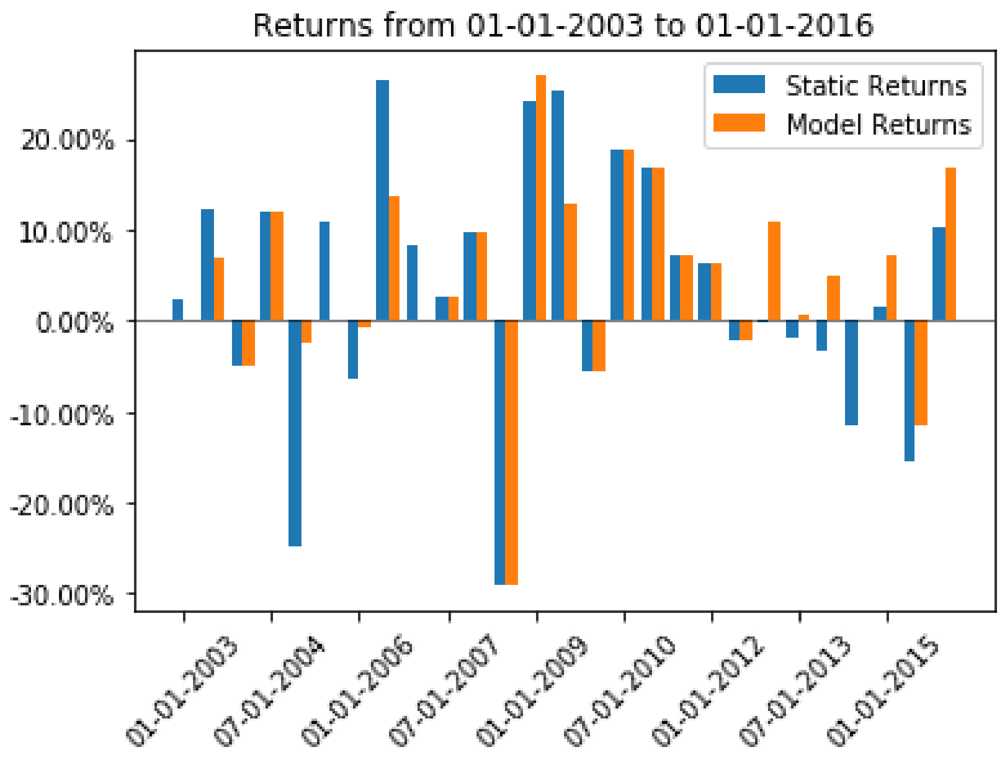
Static Returns: 4.80% /yr

Logistic Regression Returns: 7.66% /yr

Neural Network Returns: 8.31% /yr

From the period 01-01-2003 to 07-01-2016, the total returns were:

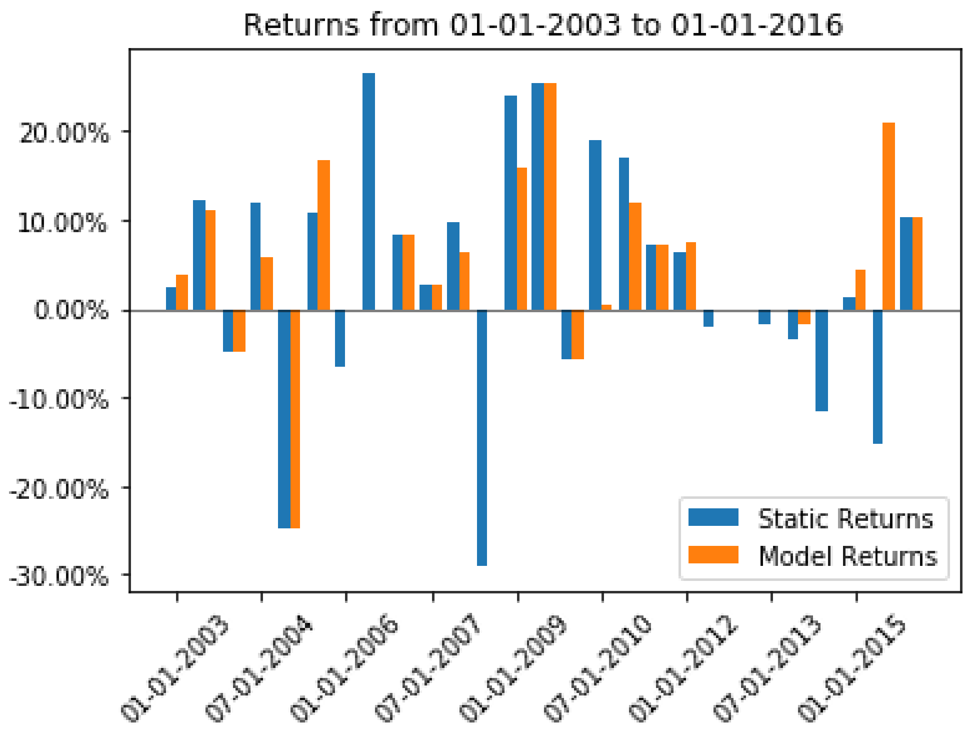
Static(real data) Returns: 88.38% (4.80% /yr)
Model(Logistic Regression) Returns: 170.95% (7.66% /yr)



Based on these results, an investor would have been better off **using the model to buy/sell the stock throughout the period.** The average accuracy of the model was 62% (min 7%, max 100%).

From the period 01-01-2003 to 07-01-2016, the total returns were:

Static(real data) Returns: 88.38% (4.80% /yr)
Model(Neural Network) Returns: 193.91% (8.31% /yr)



Based on these results, an investor would have been better off **using the model to buy/sell the stock throughout the period.** The average accuracy of the model was 57% (min 2%, max 100%)

Case Study: Johnson & Johnson (JNJ)

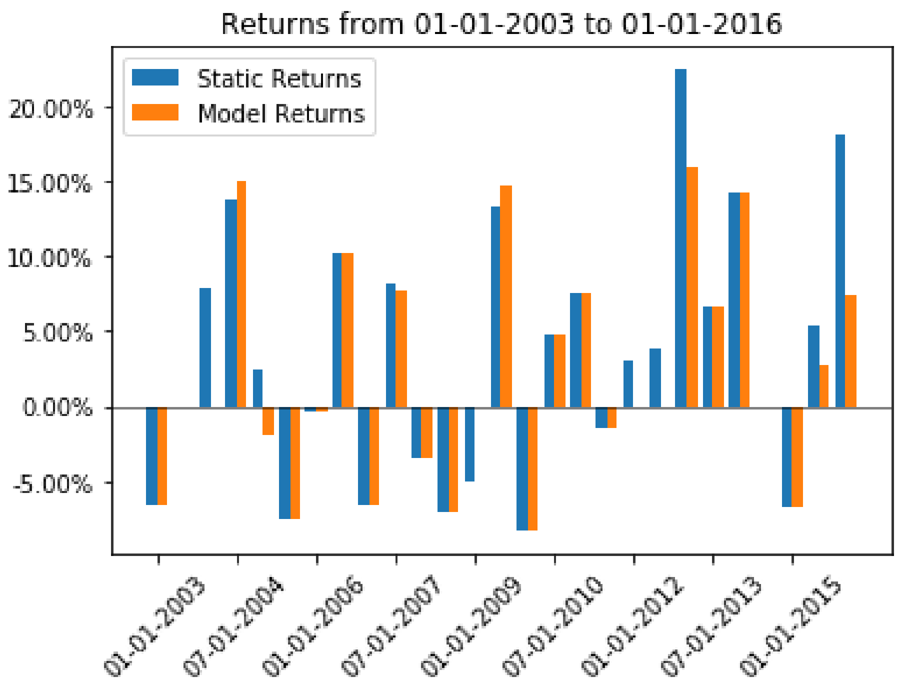
Static Returns: 5.99% /yr

Logistic Regression Returns: 3.70% /yr

Neural Network Returns: 7.25% /yr

From the period 01-01-2003 to 07-01-2016, the total returns were:

Static(real data) Returns: 119.23% (5.99% /yr)
Model(Logistic Regression) Returns: 63.39% (3.70% /yr)

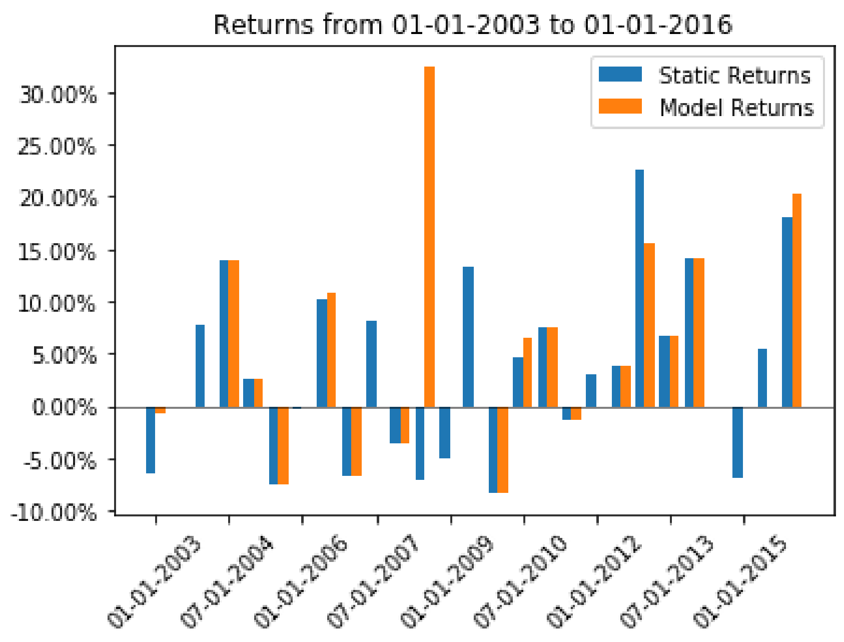


Based on these results, an investor would have been better off **holding onto the stock for the entire period**. The average accuracy of the model was 0.50% (min 0.05%, max 0.90%).

From the period 01-01-2003 to 07-01-2016, the total returns were:

Static(real data) Returns: 119.23% (5.99% /yr)

Model(Neural Network) Returns: 157.17% (7.25% /yr)



Based on these results, an investor would have been better off **using the model to buy/sell the stock throughout the period.** The average accuracy of the model was 43% (min 0%, max 100%).

J.P. Morgan Chase & Co. (JPM)

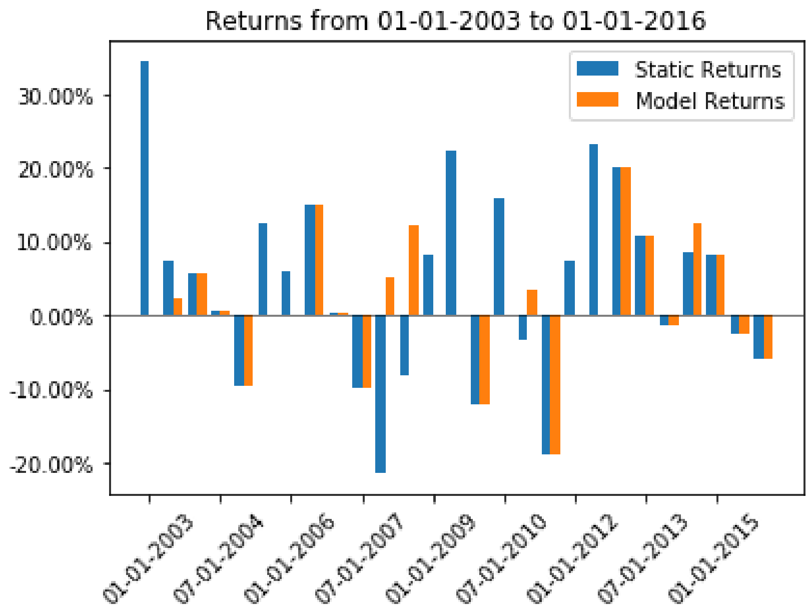
Static Returns: 6.84% /yr

Logistic Regression Returns: 1.95% /yr

Neural Network Returns: 8.92% /yr

From the period 01-01-2003 to 07-01-2016, the total returns were:

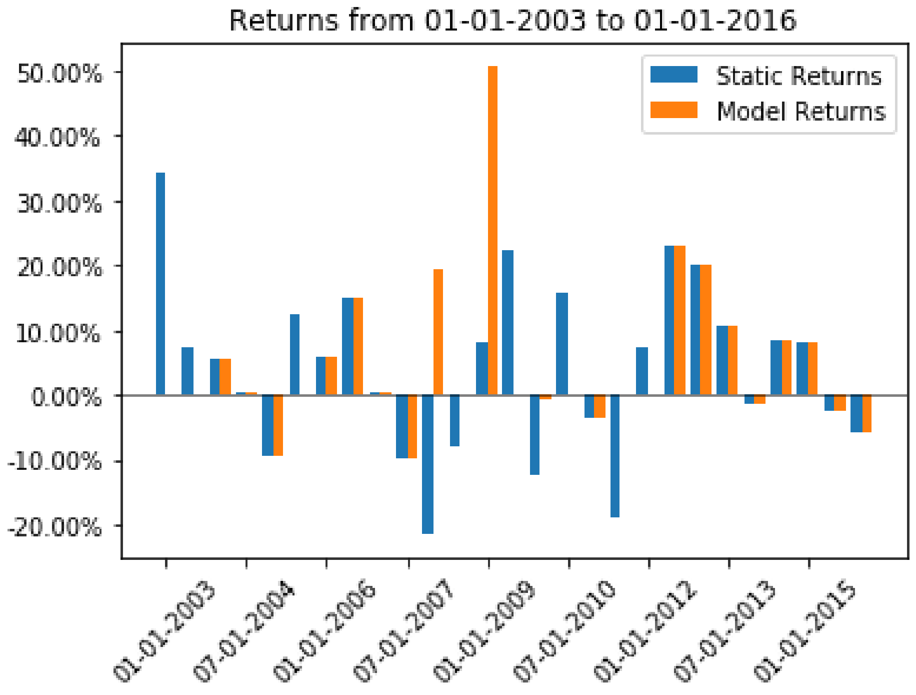
Static(real data) Returns: 144.26% (6.84% /yr)
Model(Logistic Regression) Returns: 29.86% (1.95% /yr)



Based on these results, an investor would have been better off **holding onto the stock for the entire period.** The average accuracy of the model was 50% (min 17%, max 88%).

From the period 01-01-2003 to 07-01-2016, the total returns were:

Static(real data) Returns: 144.26% (6.84% /yr)
Model(Neural Network) Returns: 216.81% (8.92% /yr)



Based on these results, an investor would have been better off **using the model to buy/sell the stock throughout the period.** The average accuracy of the model was 50% (min 0%, max 100%).

Microsoft Corporation (MSFT)

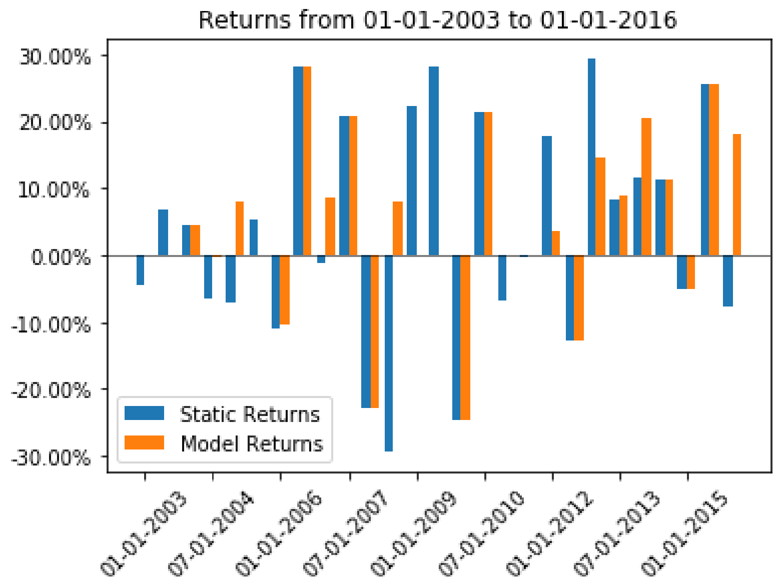
Static Returns: 4.89% /yr

Logistic Regression Returns: 7.79% /yr

Neural Network Returns: 6.55% /yr

From the period 01-01-2003 to 07-01-2016, the total returns were:

Static(real data) Returns: 90.51% (4.89% /yr)
Model(Logistic Regression) Returns: 175.13% (7.79% /yr)

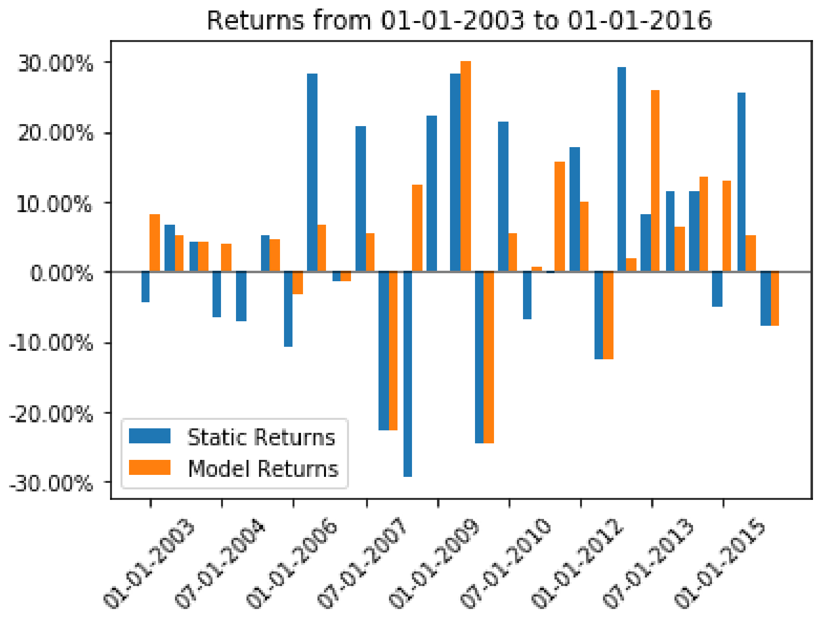


Based on these results, an investor would have been better off **using the model to buy/sell the stock throughout the period.** The average accuracy of the model was 51% (min 8%, max 97%).

From the period 01-01-2003 to 07-01-2016, the total returns were:

Static(real data) Returns: 90.51% (4.89% /yr)

Model(Neural Network) Returns: 135.58% (6.55% /yr)



Based on these results, an investor would have been better off **using the model to buy/sell the stock throughout the period.** The average accuracy of the model was 51% (min 16%, max 88%).

Walt Disney Co (DIS)

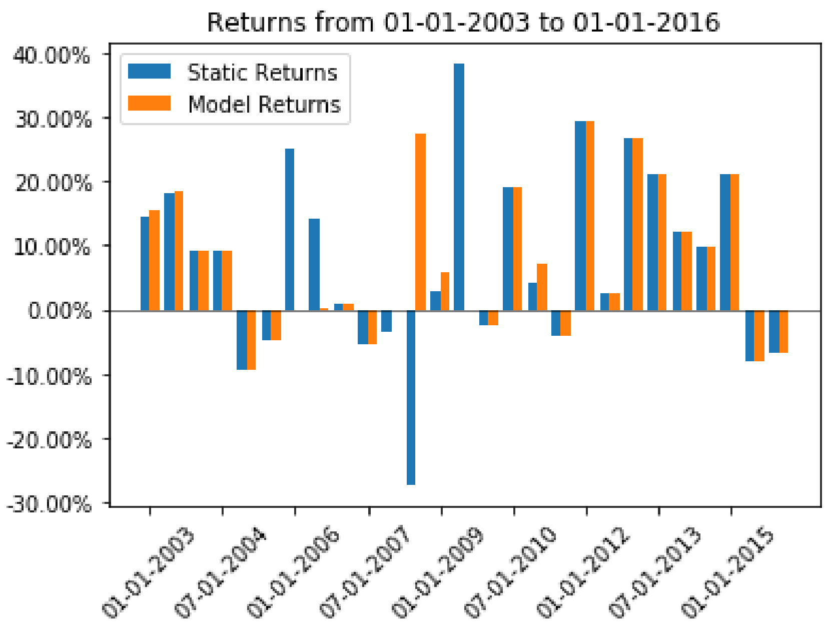
Static Returns: 13.83% /yr

Logistic Regression Returns: 13.68% /yr

Neural Network Returns: 16.11% /yr

From the period 01-01-2003 to 07-01-2016, the total returns were:

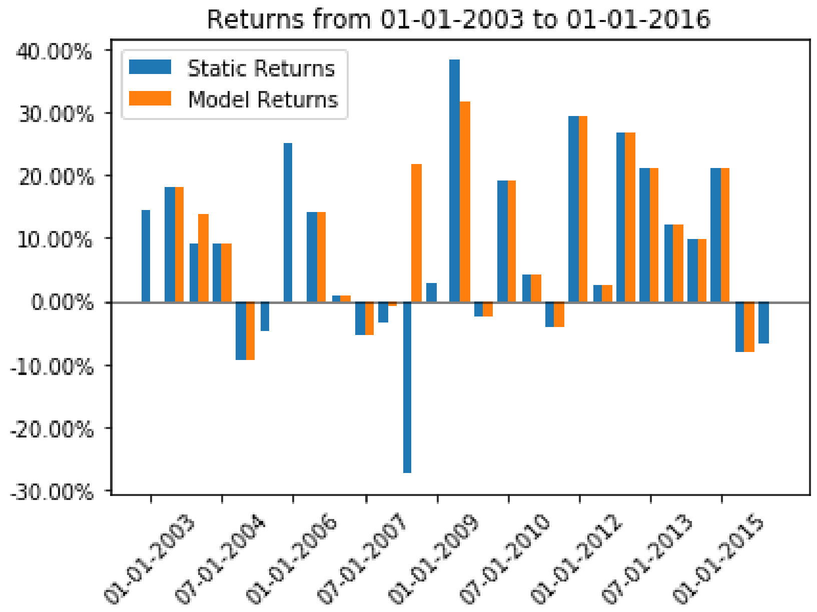
Static(real data) Returns: 474.50%   (13.83% /yr)
Model(Logistic Regression) Returns: 464.91%   (13.68% /yr)



Based on these results, an investor would have been better off **holding onto the stock for the entire period.** The average accuracy of the model was 56% (min 6%, max 93%).

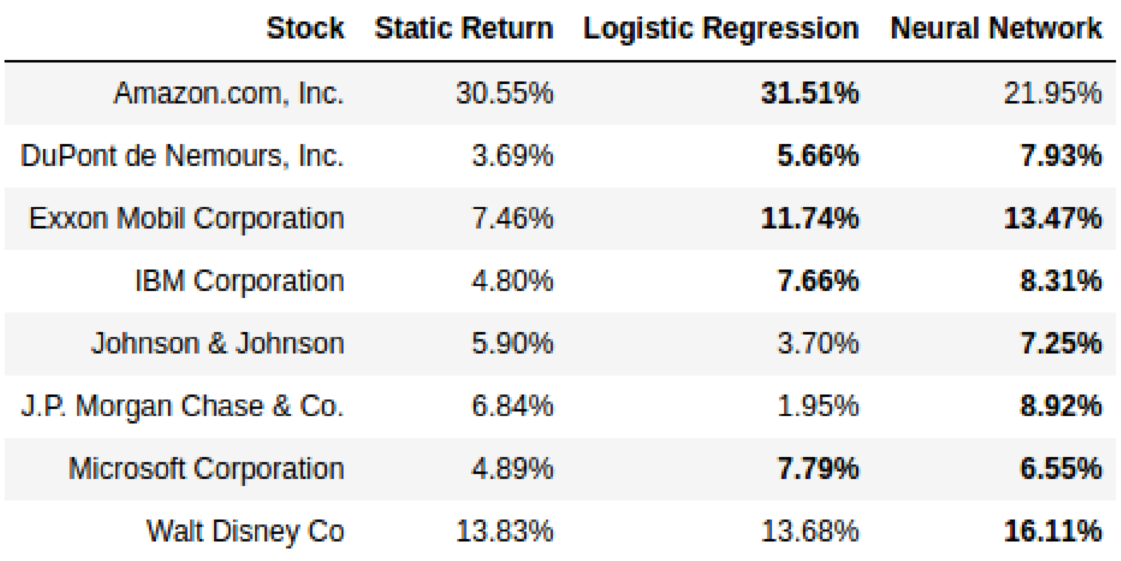
From the period 01-01-2003 to 07-01-2016, the total returns were:

Static(real data) Returns: 474.50%   (13.83% /yr)
Model(Neural Network) Returns: 651.33%   (16.11% /yr)



Based on these results, an investor would have been better off **using the model to buy/sell the stock throughout the period.** The average accuracy of the model was 58% (min 0%, max 100%).

Discussion:



Stock	Static Return	Logistic Regression	Neural Network
Amazon.com, Inc.	30.55%	**31.51%**	21.95%
DuPont de Nemours, Inc.	3.69%	**5.66%**	**7.93%**
Exxon Mobil Corporation	7.46%	**11.74%**	**13.47%**
IBM Corporation	4.80%	**7.66%**	**8.31%**
Johnson & Johnson	5.90%	3.70%	**7.25%**
J.P. Morgan Chase & Co.	6.84%	1.95%	**8.92%**
Microsoft Corporation	4.89%	**7.79%**	**6.55%**
Walt Disney Co	13.83%	13.68%	**16.11%**

The above chart shows that the model was able to successfully outperform the unengaged investor for all 8 securities tested. Companies were chosen from a broad range of sectors to test the robustness of the model. The logistic regression proved to be the more lucrative investing technique for 5/8 holdings, with an average return of 0.72% /yr greater than the static return. The neural network proved to be the more lucrative investing technique for 7/8 holdings, with an average return of 1.57% /yr greater than the static return. Among the 7 stocks in which the model performed better, the average return was 3.02% /yr greater than the static return.

Future Work:

While the model appears to have performed well, I would not recommend using this to guide any investment decisions. The use of averaging over many years and multiple economic cycles is naive, especially considering the lack of information used in the decision making process of these models. This model can serve as a guide, but it should not be used to pick which securities to buy. The purpose of this model is to maximize the returns of a holding, not to give input on which stocks will outperform others.

References:

[1] P. D. Yoo, M. H. Kim, T. Jan, "Machine Learning Techniques and Use of Event Information for Stock Market Prediction: A Survey and Evaluation", Proceedings of the International Conference on Computational Intelligence for Modeling, Control and Automation, and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC), 2005.

[2] J. Patel, S. Shah, P. Thakkar, K. Kotecha, "Predicting stock market index using fusion of machine learning techniques", Journal of Expert Systems and Applications, vol. 42, issue 4, pp. 2162-2172, 2015.

[3] C. F. Tsai, S. P. Wang, "Stock Price Forecasting by Hybrid Machine Learning Techniques", Proceedings of the International Multi Conference of Engineers and Computer Scientists, 2009.