# Capstone Project

Machine Learning Engineer Nanodegree

Theo Lekkas

July 2016

# Definition

## Project Overview

In this project I look to determine whether or not the Relative Strength Index (RSI) is an effective tool for buying & selling stocks. First I begin with a definition of the RSI:

> "The relative strength index (RSI) is a technical momentum indicator that compares the magnitude of recent gains to recent losses in an attempt to determine overbought and oversold conditions of an asset."[1]

The RSI is supposed to be able to determine both good buy & sell points for stocks (it is also used in trading futures, but I will restrict my analysis to stocks for this project.) The purpose of this project is to determine whether or not buying a stock when it's oversold & then selling the stock when it's overbought is a profitable strategy. Below, I will more clearly state the parameters of this problem & further clarify various definitions.

## Problem Statement

This project intends to determine whether a trading strategy of going long (buying) a stock when it is considered oversold using the RSI is a viable (profitable) strategy. Profitability, however, is not enough for a viable strategy. We will also want a comparison against some benchmark; the reason for this is to determine whether or not it was better to simply invest in the broader market rather than deploying a specific trading/investing strategy. For this project I will use the returns to the S&P 500 over the time period used for this strategy. There are other more sophisticated metrics that look at the volatility of the strategy (such as Sharpe ratios), but for two reasons I will not be looking at these: 1) for the purposes of this project I am only interested whether or not RSI is really predictive and nominally better than the broader market 2) I don't much care about volatility, to quote Warren Buffet "I would much rather earn a lumpy 15% over time than a smooth 12%."[2]

The buy signal for a stock will be when the RSI indicates an oversold threshold has been reached & the sell signal will be when the RSI indicates an overbought threshold has been reached. I will use adjusted daily prices. Both the purchases & sales will be made on the adjusted closing price. In the initial attempt I will not

---

[1] Relative Strength Index - RSI http://www.investopedia.com/terms/r/rsi.asp

[2] 1996 Chairman's Letter - http://www.berkshirehathaway.com/letters/1996.html

consider commissions, slippage, or other market frictions; these can be added later if the strategy actually seems viable. Furthermore, I will also look at various moving averages, which the RSI is dependent upon. I will also look at different levels of oversold & overbought (these will become clearer below when the RSI formula is broken down.)

I will use two different machine learning models, linear regression & decision trees, to see if there is predictive value to this strategy.

Below I will more clearly define RSI and various elements/concepts that will be used throughout the project.

**Metrics**

The RSI is a momentum indicator that attempts to determine short-term bullish (positive) & bearish (negative) positions for stocks. The RSI looks at the price action of a stock and does not consider other elements (such as sentiment, news, accounting metrics, etc.) The RSI is a pure "technical" indicator, its only concerned with the price movement of the instrument being analyzed.

The formula for the RSI is as follows:

$$RSI = 100 - \frac{100}{1 + RS}$$

Created with http://www.sciweavers.org/free-online-latex-equation-editor

As can be interpreted from the formula above, the RSI is an index between 0 & 100.

The RS is the relative strength metric, it is defined as average of up days over n periods divided by the average of down days over n periods. There are a couple of ways to calculate the averages; smoothed moving average[3] or exponential moving average[4]. Below is formula for the RS using a smoothed average:

$$RS = \frac{SMMA(U, n)}{SMMA(D, n)}$$

Created with http://www.sciweavers.org/free-online-latex-equation-editor

The U & D in the above formula represent 'Up' & 'Down' which are calculated by subtracting the current close with the previous days close[5].

---

[3] https://en.wikipedia.org/wiki/Moving_average - Simple_moving_average

[4] https://en.wikipedia.org/wiki/Moving_average - Exponential_moving_average

[5] https://en.wikipedia.org/wiki/Relative_strength_index - Calculation

Finally, we need to define the meaning of Oversold & Overbought. These are somewhat arbitrary, but for the initial runs I will use the traditional definitions. Oversold on the RSI is < 30 and Overbought > 70. These two definitions are definitely ripe for fine-tuning, but we will use the traditional definitions at first.

# Analysis

## Data Exploration

The data set used is taken from the Quandl WIKI data[6]. This is a community curated set of 3179 stocks (as of 06/08/2016, the date the dataset used in this project was downloaded.) The data set consists of the entire[7] price history of the stocks included in the WIKI data set. The price history includes the open, high, low, & close as well as adjusted prices (adjusted prices take into account stock splits & dividends and update the historical prices in order to have continuous prices where there are no large changes to prices that had nothing to do with daily changes.) The data also includes the ticker symbol, the date, and the volume (as well as adjusted volume.) The dataset itself is freely available from the Quandl website (linked to in the footnotes), but requires an API key (which is free, but requires registration.) The WIKI dataset, as of download date, is a csv file with 14,150,092 rows of data with 14 columns to each row.

The subset of data used consists of the adjusted price/volume data, the ticker symbol, and the date. Furthermore, there will be a subset of the entire WIKI dataset included in the repository. The entire WIKI dataset is ~1.6 gb in size. The subset will include the stocks that were used in the Jupyter notebook that accompanies this report as well as well number of other stocks randomly chosen in case some additional analysis/testing is desired on the part of the reader.

All stock datasets are in chronological order and then organized alphabetically. The stocks used are taken from the larger dataset using a ticker_gettr() function that is defined in the notebook.

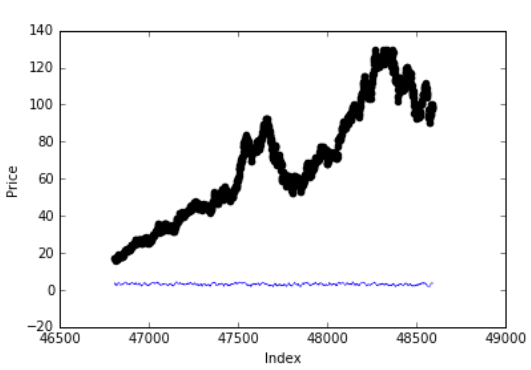
## Exploratory Visualization

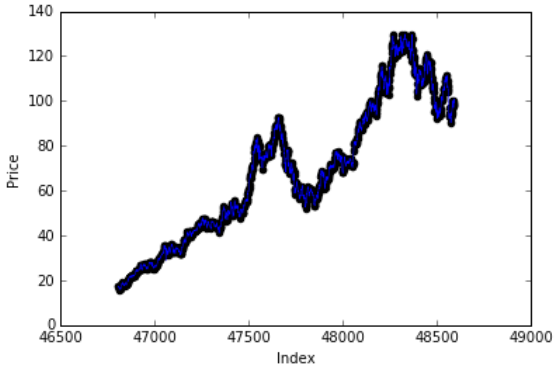
The first set of visualizations come from the initial regression models, which are discussed more in depth in the Implementation section. The chart on the right comes from the first regression model (RSI against adjusted closing prices.)

---

[6] Quandl WIKI data: https://www.quandl.com/data/WIKI

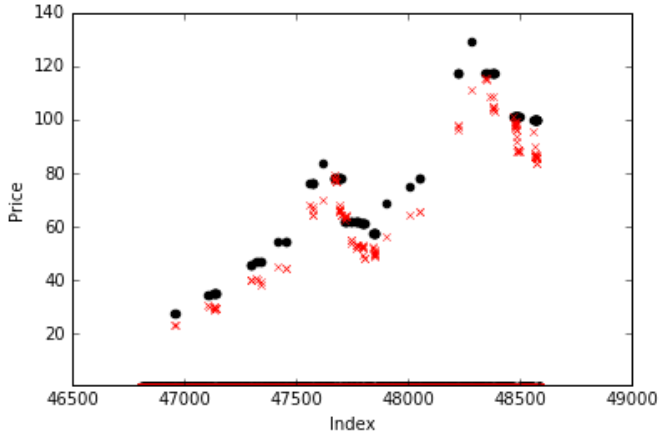
[7] For certain stocks not the entire stock history is available in this dataset.



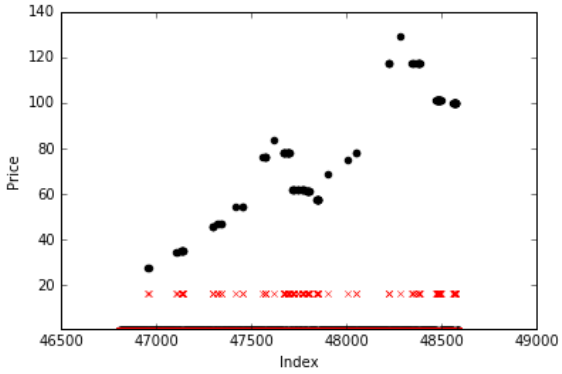



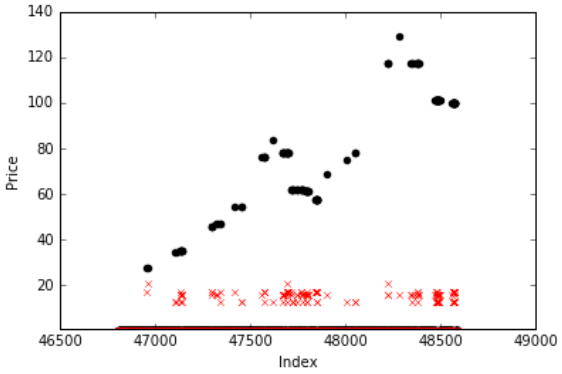

The chart on the left comes from the second regression model (adjusted high, low, open, volume, & RSI against the adjusted close.) The blue line in both charts represents the predicted prices. In the chart on the right the line overlaps the actual price data almost perfectly.

This chart shows the first linear regression model (rsi_30_model.) The red x's represent the predicted prices while the black dots represent actual prices.

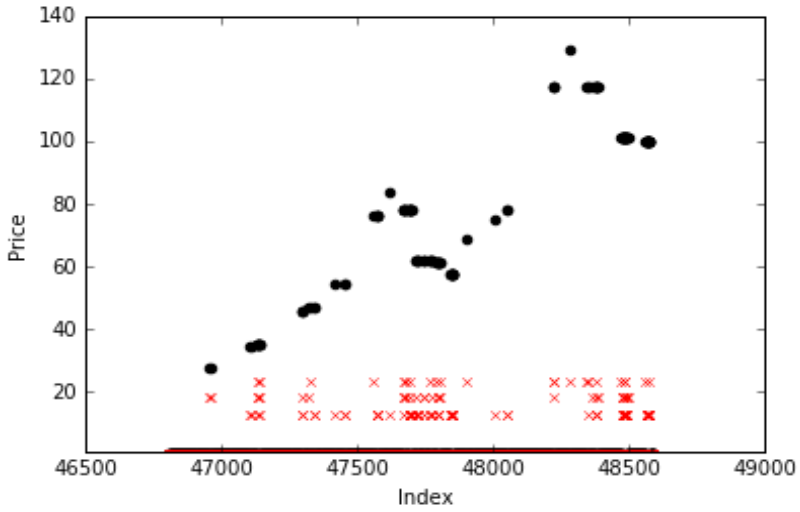



Next we have the first decision tree regression charts:

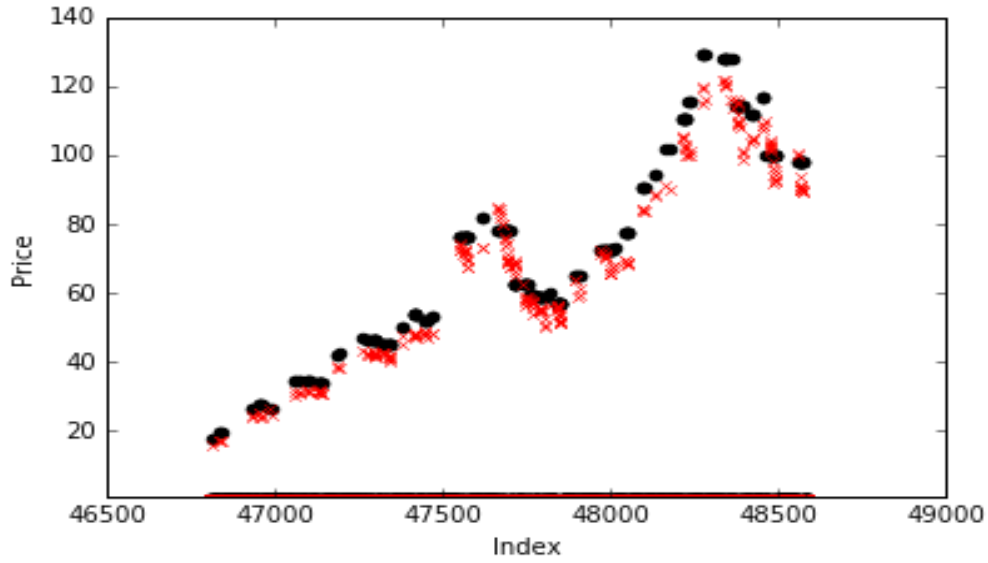



The chart on the left has a max depth of 2 and the chart on the right has a max depth of 5. The chart below has a max depth of 20.




This final chart shows the most accurate model (linear regression model rsi_309_model.)




All of these charts of Apple, Inc.

**Algorithms and Techniques**

In this project I will be using two different types of regression models. My intent here is to determine whither or not I can predict prices accurately with the RSI and whether or not that is effective to determine a trading strategy.

I will be using linear regression & decision tree regression. For decision tree regression I will be using three different max depths (2, 5, & 20.)

For the data I will be splitting the data manually in order to maintain the chronology of the data.

### Benchmark

As with most trading strategies the benchmark to compare the strategy to is usually the S&P 500 returns. In this instance, not only do I want to compare the strategy of buying a stock below RSI 30 & closing the position (i.e. selling the stock) above RSI 70 to the S&P 500, but I also want to compare this strategy to simply going long the stock over the time frame the strategy was being executed. By doing so I can make two necessary comparisons. The first comparison is whether or not the strategy had better returns than simply going long the individual stock ("buy and hold".) This is an important comparison to make since the buy & hold strategy has significantly lower transactions/slippage costs than any trading strategy does. So if a trading strategy can perform better than simply buying & holding the security trading, then the strategy has some viability (since the returns would be higher.)

The second comparison, to the S&P 500[8] (the Wilshire 5000[9] index is sometimes used for this type of comparison as well as the index represents the all publicly traded U.S. equities), is an important comparison because buying the S&P 500 represents going long the U.S. markets and diversifying away any industry or company specific risk. The argument for using the S&P 500 as a benchmark is that if a strategy can't return more than a simplistic, low risk buy & hold strategy using the S&P, then the trading/investing strategy is not worth the additional costs (i.e., management fees, commissions, slippage, etc.) associated with an active strategy.

A benchmark that I did not discuss above is one of holding periods. I wanted to include simple holding periods to see if they produced better price predictions than that traditional go long below a certain RSI threshold & close position above a certain RSI threshold. This benchmark does not exist to necessarily invalidate whether or not using the RSI is a good strategy, but it is used to see if there are better exit points for an open position after a buy signal is generated. So this is a quasi-benchmark, but it is necessary to discuss the purpose of this since it plays a role in my analysis of the RSI.

## Methodology

### Data Preprocessing

---

[8] It is important to note that this is U.S. biased research, but the principles of the strategy can be translated to any financial market.

[9] https://en.wikipedia.org/wiki/Wilshire_5000

A number of preprocessing steps were taken in order to build a dataset that would be used to build the machine learning models. The preprocessing steps are clearly shown in the accompanying notebook, but will be discussed here in order to make certain that reasoning behind each step is clear.

The first step was to create a new dataframe from the dataset that was opened initially. The purpose of this step was to make use of all the adjusted data. The reason I chose the adjusted data is relatively straightforward. The adjusted data adjusts for changes in price that are due to splits or dividends. Therefore, adjusted datasets for stock prices are fairly continuous, in that these datasets only show prices fluctuations that are due to market changes.

The next step was to add a column for the RSI data that is calculated in the notebook. This step also requires the removal of some the earliest rows depending on what moving average is being used (e.g. the first 14 rows are removed if the 14 day moving average is used.)

The next step adds two new columns. These columns are the 'Buy Price' & 'Sell Price' columns. The purpose of these columns is to get the adjusted closing price of the stock when the stock is less than the low threshold (the 'Buy Price') or greater than the high threshold (the 'Sell Price') for the RSI.

The next step was to add a number of columns that represent holding periods. Rather, these columns have the adjusted closing price a number of days in the future. The purpose of these columns was to see if there was a simple holding period strategy that was/is better than using the upper RSI threshold.

The final step was to add a column for the 'Sell Signal Price'. This column represents the adjusted closing price for the first > RSI 70 day after the initial buy signal. This is place to discuss what is going on in this column. There are many instances of the same price being used in the in the 'Sell Signal Price' column. The reason why there is often the same price is used is because I am looking for the first instance *after* of sell signal the buy signal is generated. Often times there will be multiple instances of the buy signal being generated before a sell signal being generated. All of the buy signals are still valid, they just happen to have the same sell signal.

**Implementation**

The final implementation relied upon a number of preprocessing functions that were discussed in the previous section. In this section I want to clarify the why of these functions and discuss some other aspects of the implementation. Many of the functions I wrote are automation functions that were written with the intent of automatically repeating some process so I do not exclude any step if doing it by

hand. The remaining functions were written in order to calculate some value and add it to the dataset that is eventually used during the analysis.

I will start this discussion with what I termed the "simple, naïve regression" in the accompanying notebook. As discussed above I created two functions (rs() and rsi()) whose purpose is to calculate the relative strength and the relative strength index level. Since relative strength is a straightforward calculation it is more effective to calculate these values with code than to attempt to get the data from an outside source. My reasoning here is fairly simple, by constructing functions to make the calculations I can directly discuss what assumptions & methods I use to make the necessary calculations. I discussed the relative strength calculations I used in detail in the 'Metrics' section. The functions that I wrote are in the 'Basic Calculations' section of the notebook. These two functions are used to derive the values that exist in the 'RSI' column of the final datasets.

An important element of both the rs() & rsi() functions are the parameters. For both of these functions a look_back parameter exists. The purpose of the look_back parameter is to allow a user to determine what the length of the moving average will be. The moving average is something is very prone to fine-tuning so I decided it would be best to make that something that is easily updated as opposed to a parameter that is hardcoded.

Two more functions that I created are price_columns() & rsi_add(). The price_columns() functions purpose is to create two new columns and populate them with data. These columns are 'Sell Price' and 'Buy Price'. The values for these two columns are calculated by filtering the 'Adj-C' columns (adjusted close) by the RSI value. If the RSI value is greater than 70, the Adj-C value will be added to the 'Sell Price' column. If the RSI is lower than 30, the Adj-C value will be added to 'Buy Price' column. The purpose of these columns is to ease the process of finding the buy and sell prices.

rsi_add() is a more straightforward function. The rsi_add() function is more of an automation function, its purpose is to call the rsi() function and add an 'RSI' column to a dataset that is passed into the function. The rsi_add() function also takes the ave_length parameter that serves the same purpose as the look_back parameter discussed above for the rs() & rsi() functions.

One other implementation feature before I began to run the models was to add holding period columns. As discussed in the Benchmark & Data Preprocessing sections, I wanted to add holding periods for comparison purposes. Shifting the 'Adj-C' column back by an arbitrary number created the holding periods. The process of how this was done was to create a copy of the data's Adj-C column, and then use pandas shift[10] attribute to assign to some var name, and finally add

[10] http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.shift.html

a new column (something like 'Day 10') and pass the var's values to that new column.

Before I ran the naïve regression models I had to split my data into training/test sets. In order to do this I created a function (train_test()) to split in the dataset. The reason I wrote this function from scratch was because it necessary to preserve the order of the data. This function takes in the dataset (with the added columns), the desired features, the desired prediction data, and the % of data to be used as the training set. Then creates two separate new dataframes from the dataset passed into it. From there the function proceeds to split the data at the % desired by running the following line of code:

```
train_percent = int(round(len(data) * train))
```

The function then returns four new dataframes that contain the training features/predictions and the testing features/predictions. From here I can now build my models.

With all the preprocessing & train/test splits dine I decided to run a linear regression model to see if the raw (without being filtered with any specific levels) RSI had any predictive value. The first attempt was simply the 'RSI' as my feature set against the adjusted closing price data (Adj-C). This model produced, as seen in the notebook, produced some very poor results. From the results it is clear that using just the raw RSI number is fairly useless for predicting prices. This shouldn't really come as a surprise, although I was expecting the results to be so bad. This outcome makes sense since the RSI is only supposed to be predictive in the extremes.

After this initial attempt I decided to use multiple features to see what would happen. At this point I am still only using the raw RSI value, but I also include adjusted high, low, open, & volume data. I decided to basically throw everything at the regression function and see what happens. Unsurprisingly the outcome was almost perfect, I overfit the data! The outcomes of both of the regression models can be seen in the Visualizations section, but I wanted to include some metrics here. For the first model the model score[11] is `-4.4864101023722318` and its RSS is `9.356749e+06` (the model score can be as high as 1.0; the higher the score the 'better' the predictive value of the model and the converse is true for RSS, where an RSS of 0 would be a perfect prediction. Perfect predictions, however, most likely mean the data has been overfit.)

Now that I ran some simple models I wanted to test for predictive value the RSI using the oversold & overbought thresholds discussed above. In order to do this

---

[11] http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html - sklearn.linear_model.LinearRegression.score

I created another function called sell_signal_prices(). This function takes in three parameters, the dataset, the overbought threshold, and the oversold threshold. The output of this function is the dataset with a new column, 'Sell Signal Price'. The values in this column represent the first adjusted closing price when the RSI exceeds the overbought threshold after the buy signal (which is less than the oversold threshold) is generated.

A couple of ***caveats*** need to be pointed about the sell_signal_prices() function. Although the function works consistently, it relies on some less than optimal implementations. First, there is nested for loop that is used, although not a very big deal given the size of the datasets used this can represent a problem if the datasets are too big. Second, the values in the columns are updated while iterating over the rows. This manner of adding values to a column goes against what the core maintainers of pandas recommend[12]. Although this functions works, it is not ideal.

Now that the dataset is cleaned up and all the needed values have been added I began to run my regressions. The regressions that I chose were linear regression[13] & decision tree regression[14]. For both of these regressions I am running the RSI & Buy Price against the Sell Signal Price. I am using the first 80% of the dataset (in this case Apples stock history to build the model) to train the model and the remainder as the test sets.

The linear regression model (rsi_30_model) turned out to be very predictive, in fact this is the model that will be repeatedly used throughout. The decision tree regressions, however, weren't so effective. For the decision tree regressions I chose three different max depths: 2, 5, & 20.

A final implementation detail before I move on to the refinements. I created a back of the envelope calculation to sum up the difference between the 'Sell Signal Price' value and the 'Buy Signal' value. The purpose of this is to give simplistic back test (although it is essential to point that this is not a real back test, merely a simplistic calculation to see if there is a positive or negative outcome from mechanistically buy @ < 30 & selling @ > 70.) Towards the end of the notebook I write another function, back_envelope(), to generate the back of the envelope calculations automatically. This function takes a list of datasets and generates the number of transactions and the sum of the difference. The thinking behind this is that the higher the summed value the better the result. Whether or not this calculation is a good back of the envelope version of a back test is a good question, but it did lead to some interesting insights that will be discussed later.

---

[12] http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.iterrows.html

[13] http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

[14] http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html

One last implementation detail to discuss is how the holding periods were used. For the holding periods I wrote a function, predictor(), to automate the training/testing for each holding period. The predictor() function uses a linear regression model and takes in, as parameters, the dataset to be used, periods (which is a list of strings), features to be trained on (again a list of strings), and the % of the data to be used as training data. The linear regression model in predictor() is training the 'RSI' & 'Buy Price' data against the holding period data. As discussed above, the holding period data is the adjusted closing price for the session x holding periods in the future.

**Refinement**

After running the initial regressions and finding a model that seems to be predictive, I needed to test out some assumptions. As discussed early on, the traditional thresholds & moving average used are < 30 for oversold, > 70 for overbought, and the 14 day moving average. There are few things that I wanted to know in order to find a good model. I wanted to see whether or not a lower threshold for oversold had a positive impact on the prediction results. I also wanted to see whether or not a different moving average had an impact on the results. Finally, I wanted to make use of the holding period columns to see if simply holding the security after a buy signal was superior to using the overbought threshold.

The first thing I attempted was to see what effect using a different moving average had on my results. I decided to use a 9-day moving average to see if a shorter time frame had an affect on the accuracy of the model. I constructed a new dataframe and went through the same process discussed above to construct a dataset that would be used in the analysis. The only difference was that I used 9 days for the RSI look_back. I once again used Apple (the Apple dataset is want I decided to use as the dataset to train all the models) as my dataset.

I first ran the linear regression model since the original regression model was very accurate. I then trained the decision tree models, once again using the 2, 5, and 20 max_depth parameters. As with the 14-day regression, the decision tree models were not very accurate. I will discuss the decision tree regressions more in the Results section, but at this point I decided to dispense with the decision tree models.

The next refinement was to see if a lower threshold for the RSI buy signal would see any improvements over the 30 RSI. I once again used Apple, but I also used the 14-day & 9-day datasets to see if there was any improvement.

# Results

**Model Evaluation and Validation**

After the testing some refinements I came upon the best model. Model rsi_309_model had the most accurate predictions. Model rsi_309_model is a linear regression model that uses the 9-day moving average and a buy signal threshold of 30 RSI. My expectation here was that this model would not be significantly better than the 14-day model, but I was wrong. My assumption was there would be more transactions and that would have a negative effect, however, I did not take into account how far back this dataset goes (back to 1980.) Typically the more transactions that occur the worse the results are as slippage, commissions, & other frictions impact results. The reality is that the number of transactions over the period observed were not significant enough to generate the friction costs that often have an impact on active trading strategies. Simply, the strategy doesn't over trade. Model rsi_309_model only had 1436 transactions over a 36 year period, or ~40 trades a year.

Now I want to include the model score & RSS for each of the models.

rsi_30_model, this is the first model using a 14-day moving average (M.A.) & a RSI buy signal of 30.
Score: 0.98512976758219584
RSS: 11671.091729

trereg_2, this is a decision tree regression model using a max_depth of 2, a 14-day M.A., & a RSI buy signal of 30.
Score: 0.29166041143263166
RSS: 555949.3679156045

trereg_5, this is a decision tree regression model using a max_depth of 5, a 14-day M.A., & a RSI buy signal of 30.
Score: 0.27080168559486484
RSS: 572320.60514729063

trereg_20, this is a decision tree regression model using a max_depth of 20, a 14-day M.A., & a RSI buy signal of 30.
Score: 0.28727977271820282
RSS: 555949.3679156045

rsi_309_model, a linear regression model that uses a 9-day M.A. and RSI buy signal of 30 (this is the best model.)
Score: 0.99313047238565977
RSS: 9260.092403

trereg_2309, this is a decision tree regression model using a max_depth of 2, a 9-day M.A., & a RSI buy signal of 30.
Score: 0.286652608931
RSS: 961589.082591

trereg_5309, this is a decision tree regression model using a max_depth of 5, a 9-day M.A., & a RSI buy signal of 30.
Score: 0.286379440629
RSS: 961957.312237

trereg_20309, this is a decision tree regression model using a max_depth of 20, a 9-day M.A., & a RSI buy signal of 30.
Score: 0.313487783451
RSS: 925415.387739

rsi_1425_model, a linear regression model that uses a 14-day M.A. and RSI buy signal of 25 (this is the best model.)
Score: 0.976905234058
RSS: 9384.264266

rsi_925_model, a linear regression model that uses a 9-day M.A. and RSI buy signal of 25 (this is the best model.)
Score: 0.989580230991
RSS: 9533.049306

Finally, I will discuss the holding periods. The holding periods proved to be very inaccurate in predicting prices. This can be perceived as a positive for the RSI, since it is an indicator that attempts to understand whether or not a security is overbought/oversold rather than how long the security is held. Since these were such inaccurate results I only tested the holding periods with a linear regression model using the 14-day, RSI 30 dataset. The resulting model scores & RSS:

The score, -4.05448784097, and RSS, 8630669.29191, of the model for 1 day holding
The score, -4.04152152132, and RSS, 8629283.47635, of the model for 3 day holding
The score, -4.03151988386, and RSS, 8632681.31584, of the model for 5 day holding
The score, -4.00502160549, and RSS, 8636099.24864, of the model for 10 day holding

Now that I have found the most accurate model, rsi_309_model, what happens when I use the model on data it has not been trained? Is rsi_309_model generalizable? In order to answer these questions I chose four other stock datasets to run predictions on. Two of these stocks are in similar industries as Apple, one older (IBM) and one around the same age (MSFT.) The other two come from other industries, ZUMZ & X. Of these, the one with the most interesting results is X, which will be discussed in greater detail the Conclusion section.

I constructed the dataframes that rsi_309_model made the predictions on the same way as I did with the Apple dataframes. I even kept the holding periods to have consistent datasets. In short, the rsi_309_model does a very good job in generalizing to data it's never seen before. X, however, makes demonstrates a very clear problem with RSI models in general. Below I have the scores & RSS for each of the four new stocks.

IBM model score, 0.98963700485044326, and RSS, 89627.127300015272
MSFT model score, 0.99088780111471553, and RSS, 4059.2439652341886
ZUMZ model score, 0.9696304006226093, and RSS, 6900.9056025502259
X model score, 0.83073419185596586, and RSS, 173604.18356860543

Here we can see that X falls short of the other stocks, below I will include the back of the envelope results which further demonstrates how poor X does.

For X: # of transactions: 1222. Total value of transactions: -1552.02945296.
For MSFT: # of transactions: 1041. Total value of transactions: 474.602933111.
For ZUMZ: # of transactions: 455. Total value of transactions: 104.58.
For IBM: # of transactions: 2231. Total value of transactions: 1140.24345053.

**Justification**

I now know that the RSI is a predictive indicator, but the real question remains. Is using the RSI better than buying & holding the S&P 500 and, more importantly, is it better than buying & holding the stocks that its used on.

I also want to quickly discuss building the model on a specific stocks dataset as opposed to a more general dataset (the S&P 500 or the Wilshire 5000.) My purpose here was to determine whether or not the calculations presented at the beginning of this report were generalizable. From that perspective, it should not matter the dataset that a model is constructed from, what *should* matter is whether or not that formulas presented could create a predictive model that can be used on other stock datasets. In that respect, I believe that the model is in fact predictive, although it has some limitations (which I will discuss more in depth when I discuss X.)

What I want to really know is whether or not this strategy beats simple buy & hold strategies. In order to do this I will use Quantopian's[15] platform to conduct the backtest for RSI 9-day strategy. First I want to discuss some necessary caveats. First, that data source used in the Quantopian backtest is not the same as the Quandl dataset; however, the data is similar enough that I believe the impact on the comparison will not be affected. Second, the backtest only go back to

[15] https://www.quantopian.com/home

01/01/2003 (except for ZUMZ, which has the same timeframe) because of certain limitations to Quantopian's dataset.

As repeatedly mentioned, the two benchmarks I will use are the S&P 500 and the stock itself. Below is a table with the results of the each stock chosen:

| | Total Returns | | | |
|---|---|---|---|---|
| **Stock** | **RSI 9** | **Vs. S&P 500** | **RSI 9** | **Vs. Buy & Hold** |
| Apple | 913% | 204.19% | 913% | 10008.19% |
| MSFT | 87.40% | 204.19% | 87.40% | 179.90% |
| X | -93.20% | 204.19% | -93.20% | 55.10% |
| IBM | 143.30% | 204.19% | 143.30% | 147.40% |
| ZUMZ | 9.30% | 120.10% | 9.30% | 22.10% |

As we can see from the results, even though the RSI 9 model was predictive as a strategy it failed to consistently out perform the S&P 500 and consistently underperformed the a simple buy & hold of the stocks tested.

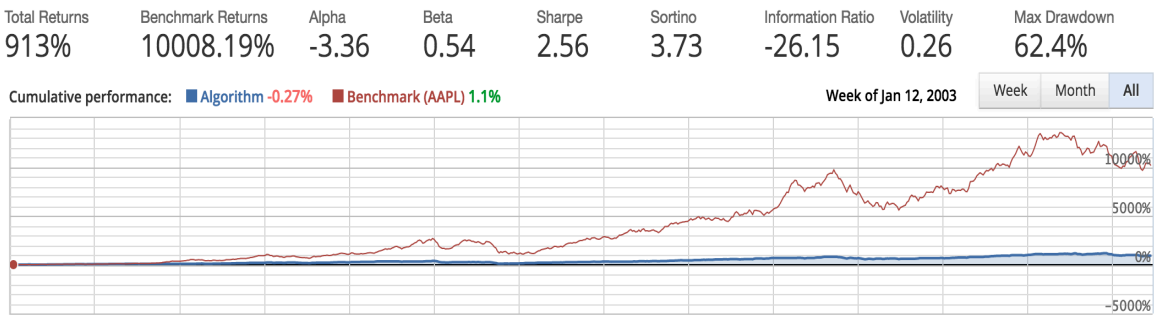
A note here, I script is included with the code that was run on the Quantopian platform that can be copy & pasted to be tested. This is not a stand-alone script and requires the Quantopian environment to be run.

# Conclusion

**Free-Form Visualization**

Here I will include some charts generated on the Quantopian platform that shows the returns for Apple against itself & the S&P 500. The other stock comparison charts are included in an images directory.

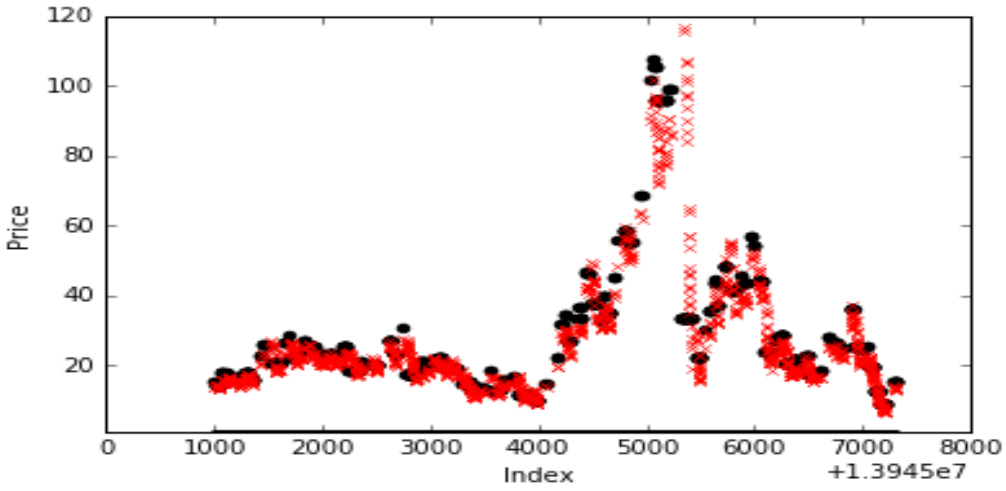
Apple RSI 9 Vs. Apple Buy & Hold



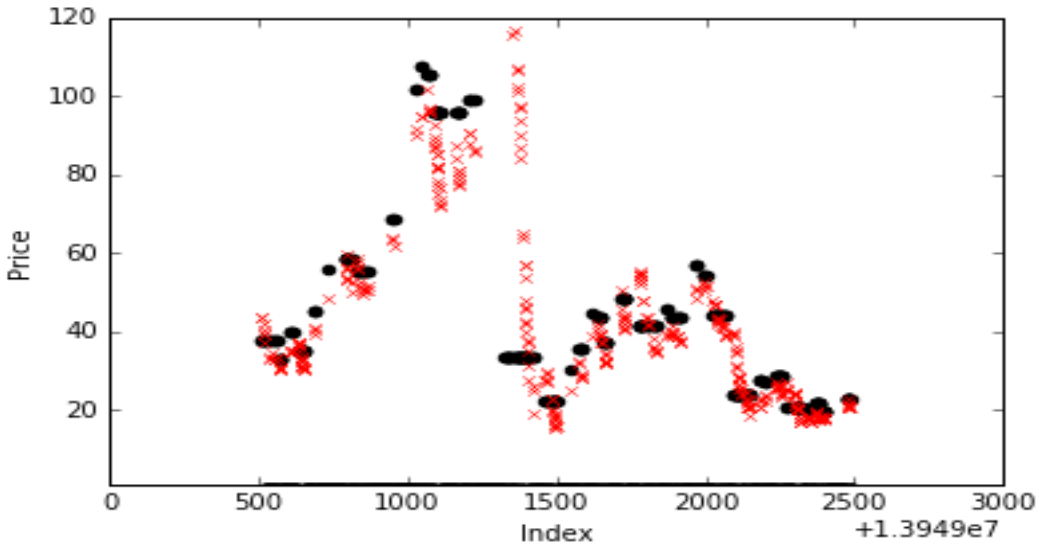

Apple RSI 9 Vs. S&P 500 Buy & Hold


I also want to include some charts of X that shows the predictions versus actual prices. I will discuss X in more detail below. These charts show how far off the mark the predictions were in certain type of environment.

All X data:




Next, I zoomed in on the area where the bad predictions mostly concentrated. This chart shows the data leading up to, including, and shortly after the financial crisis of 2009:




**Reflection**

Not discussed in depth were the decision tree regression models. As seen the charts towards the beginning of this paper the decision tree models were very poor predictors of prices. It is abundantly clear that decision tree models are greatly impacted by the earliest data. Since stock prices tend to increase in value over time, it is clear that the decision tree regression was not the appropriate algorithm to use. I was hoping to have more than one algorithm generating fairly accurate predictions, but the decision tree regressors proved to be overlay impacted by the early data and unable to update the predictions as more data was taken in.

As mentioned a number of times throughout this paper, I was not just looking to see if the RSI was predictive indicator, but whether or not it can prove to be a more effective strategy than simple buy & hold. On the predictive front, RSI proves to be fairly decent. The RSI, however, seems to have some strong weaknesses (which I will explore more with the discussion on X.) As a trading strategy, the RSI (in this case the RSI 9-day) falls far short of simply just owning the underlying securities or the S&P 500 as a whole. Although a bit disappointing, it is not entirely surprising that an indicator that has been around for decades no longer (if it ever did) performs better than the market. Over time trading strategies seem to loose their edge as more & more market participants take advantage of any edge these strategies provide.
Disappointment of decision tree models (reference how they are often touted in finance.)

X was a spectacular losing trading vehicle for the RSI 9 strategy and some hints as to why can be seen in the two charts above. It is clear that when prices moved rapidly in the opposite trend, rsi_309_model continued to predict prices

that were significantly different from the actual closing prices. This was consistent with prediction results, the back of the envelope calculation, and the results of implementing the strategy in the real back test. What is clear is that when extreme changes happen in the data, the RSI misses the information in that data. This is a great weakness in the strategy. As we saw with X, when the stock lost upwards to 90% of its value, the RSI model was predicting much higher prices than the actual closing prices. This was confirmed with the 93.2% loss that execute this strategy on X over the 13 year period represented by the back test conducted.

It's clear that using this model without some risk management & position sizing will lead to negative relative to basic benchmarks (and in some instances in absolute terms.)

**Improvement**

Where can improvements in the model be made in the use of linear regression for the RSI trading strategy? First, more fine-tuning can be made. More overbought & oversold thresholds can be used to see if there are more accurate models to be uncovered. Also, different moving averages can also be used. Second, different types of moving averages can also be used to see if there is a positive effect. Third, introducing some risk management & position sizing would have a beneficial effect on the performance of any models produced.

Beyond linear regression, it would also be beneficial to see whether or not trades can be classified as buys or sells using the price data and the RSI data. It might be much more beneficial to classify buys or sells as opposed to attempting to predict prices outright. The ability to determine whether or not a buy or sell signal is actually a good signal would be tremendously beneficial, especially given how poorly the linear regression model performed on the backtests.