**Capstone Project** Theo Lekkas

Machine Learning Engineer Nanodegree 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]](#footnote-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]](#footnote-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 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:



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]](#footnote-3) or exponential moving average[[4]](#footnote-4). Below is formula for the RS using a smoothed average:



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]](#footnote-5).

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**

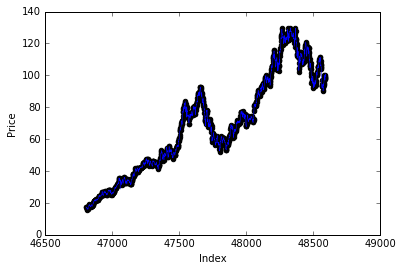
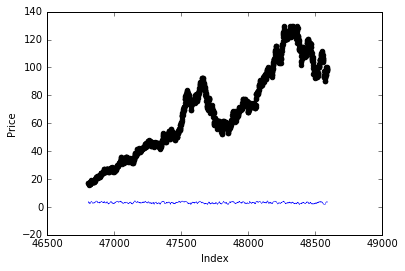
The data set used is taken from the Quandl WIKI data[[6]](#footnote-6). This is a community curated set of 3179 stocks (as of 06/08/2016, the date the dataset used in this project was downloaded.) The data set consists of the entire[[7]](#footnote-7) price history of the stocks included in the WIKI data 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.)

****

The chart on the left comes from the second regression model (adjusted high, low, open, volume, & RSI against the adjusted close.)

**Algorithms and Techniques**

**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]](#footnote-8) (the Wilshire 5000[[9]](#footnote-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**

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]](#footnote-10) attribute to assign to some var name, and finally add 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:

1. 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]](#footnote-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 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]](#footnote-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]](#footnote-13) & decision tree regression[[14]](#footnote-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 these 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.

**Refinement**

**Results**

**Model Evaluation and Validation**

**Justification**

**Conclusion**

**Free-Form Visualization**

**Reflection**

**Improvement**

1. Relative Strength Index - RSI <http://www.investopedia.com/terms/r/rsi.asp> [↑](#footnote-ref-1)
2. 1996 Chairman's Letter - <http://www.berkshirehathaway.com/letters/1996.html> [↑](#footnote-ref-2)
3. [https://en.wikipedia.org/wiki/Moving\_average - Simple\_moving\_average](https://en.wikipedia.org/wiki/Moving_average#Simple_moving_average) [↑](#footnote-ref-3)
4. [https://en.wikipedia.org/wiki/Moving\_average - Exponential\_moving\_average](https://en.wikipedia.org/wiki/Moving_average#Exponential_moving_average) [↑](#footnote-ref-4)
5. [https://en.wikipedia.org/wiki/Relative\_strength\_index - Calculation](https://en.wikipedia.org/wiki/Relative_strength_index#Calculation) [↑](#footnote-ref-5)
6. Quandl WIKI data: <https://www.quandl.com/data/WIKI> [↑](#footnote-ref-6)
7. For certain stocks not the entire stock history is available in this dataset. [↑](#footnote-ref-7)
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. [↑](#footnote-ref-8)
9. <https://en.wikipedia.org/wiki/Wilshire_5000> [↑](#footnote-ref-9)
10. <http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.shift.html> [↑](#footnote-ref-10)
11. [http://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.LinearRegression.html - sklearn.linear\_model.LinearRegression.score](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_model.LinearRegression.score) [↑](#footnote-ref-11)
12. <http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.iterrows.html> [↑](#footnote-ref-12)
13. <http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html> [↑](#footnote-ref-13)
14. <http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html> [↑](#footnote-ref-14)