# Machine Learning Engineer Nanodegree

## Capstone Project

Sachin Guliani
February 1st, 2018

## Definition

### Project Overview:

Warren Buffett and many other investors say that the best way to save your money is to invest in the S&P 500 [2]. S&P index is a representation of the weighted average of the stock prices of the top 500 companies in the U.S. S&P from the period of 1950-2009 on average returned 11% a year. Since average inflation was 3.8%, money that is kept in S&P increased on average 7% each year. Keeping cash is bad because it loses its value from inflation. Interest from bank accounts often is below or barely above the inflation rate, so it is not the best way to keep money long term. Bonds are often good investments, but their return is usually lower than S&P. Keeping your money in S&P allows you to consistently make the most money over time. The question I want to answer is if timing the market allows a person to make more money in S&P. Basically, given data of S&P for a given month, can we use a classification model to accurately decide whether to take money out of S&P or keep money in S&P [3]?

### Problem Statement:

Can we predict when S&P goes down and when S&P goes up and use that information to make more money? Here are the tasks:

1. Download and pre-process Shiller S&P 500 data
2. Calculate Base Case
3. Train on data from first 70% of dates
4. Test on data from last 30% of dates
5. Compare Test and Base Case

### Metrics:

Every month, $1000 dollars is saved (in today's dollar). The program has to decide whether to invest the $1000 into stocks or convert all the existing stocks into cash. The Base Case is when the $1000 is always converted into stocks based on the current stock price.

Base Case Method:

1. Start with Shares and Money set to 0
2. Money = Shares*(Price + Dividend/12) + 1000 (Dividends are reinvested in S&P)
3. Shares = Money/Price
4. Loop over Steps 2 and 3 using the new Shares price
5. Return Money

Percentage Difference:

$$Percentage = 100 \bullet \frac{Money_{Trained} - Money_{Base\ Case}}{Money_{Base\ Case}}$$

How well the program learns is the percentage difference from how much money is made in comparison to the base case. If the percentage difference is positive, then it has succeeded. If it is negative, then it has performed worse than holding money in S&P.

# Analysis

## Data Exploration:

The Data Set is U.S. Stock Markets 1871-Present and Cape Ratio.
Here is a some of the data:

| Date | P | D | E | CPI | Date Fraction | LRI | Real P | Real D | Real E | CAPE |
|---|---|---|---|---|---|---|---|---|---|---|
| **1881.01** | 6.19 | 0.27 | 0.49 | 9.42 | 1881.04 | 3.7 | 162.1 | 6.94 | 12.72 | 18.47 |
| **1881.02** | 6.17 | 0.27 | 0.48 | 9.51 | 1881.12 | 3.69 | 159.96 | 7 | 12.49 | 18.15 |
| **1881.03** | 6.24 | 0.28 | 0.48 | 9.51 | 1881.21 | 3.69 | 161.78 | 7.13 | 12.38 | 18.27 |
| **1881.04** | 6.22 | 0.28 | 0.47 | 9.61 | 1881.29 | 3.68 | 159.66 | 7.19 | 12.15 | 17.95 |
| **1881.05** | 6.5 | 0.29 | 0.47 | 9.51 | 1881.37 | 3.67 | 168.52 | 7.39 | 12.16 | 18.87 |
| **1881.06** | 6.58 | 0.29 | 0.47 | 9.51 | 1881.46 | 3.67 | 170.59 | 7.52 | 12.06 | 19.03 |

1. The Date goes from 1871.01 to 2018.01. The decimal indicates the month. S&P Data is given every month from January of 1871 to January of 2018
    a. .01 indicates January
    b. .02 indicates February
    c. ….
    d. .12 indicates December
2. P represents the price. It is the price of the S&P Index. If you wanted to buy one share of S&P, that is how much it would cost in dollars. The S&P Index is the weighted average of all the stock prices based on the companies worth. For example, today Google and Amazon prices are going to be weighted higher in the S&P Index since it has a bigger percentage of the market. The S&P is just a stock index that represents the top 500 companies [8].
3. D represents the dividend. Some companies offer dividends, while others don't have them. Dividends give you money for owning the stocks. "For the issuing company, they are a way to re-distribute profits to shareholders as a way to thank them for their support and to encourage additional investment. Dividends also serve as an announcement of the company's success." A 50-cent dividend means that for each share you have, you will receive 50 cents. In this case we are looking at the weighted average of all the dividends based on stock weightage. This is the amount you would get per share if you owned the stock for the last year. As an approximation I divided it by 12 to make it the

amount earned each month. I can use this approximation since the dividend's don't vary greatly from one another [1].

4. E represents the Earnings. The earnings is the profit divided by the stock price. It's how much money a company made per share. In this case it is the weighted average of earnings so the earnings the top top 500 companies made per S&P share [4].
5. "The Consumer Price Index (CPI) is a measure that examines the weighted average of prices of a basket of consumer goods and services, such as transportation, food and medical care. It is calculated by taking price changes for each item in the predetermined basket of goods and averaging them. Changes in the CPI are used to assess price changes associated with the cost of living; the CPI is one of the most frequently used statistics for identifying periods of inflation or deflation [5]."
6. The Date Fraction is used to plot the date on a number line.
7. "Long-term interest rates refer to government bonds maturing in ten years [6]."
8. The Real Price is the S&P index price based on the value of today's dollar. Due to inflation, the value of the dollar today is drastically different from the value of the dollar in 1881. The regular S&P doesn't tell us anything about how it is compared to today (due to inflation).
9. The Real Dividend is inflation adjusted Dividend rate.
10. Real Earnings is inflation adjusted Earnings price.
11. CAPE is the cyclically adjusted P/E Ratio. The Price per Earning's Ratio where the Earnings are averaged over the last 10 years.

Data up till December of 1880 is not used in the experiment because The CAPE is not available for that period of time. The CAPE needs the earnings for the last 10 years and the data set starts in 1870, so that is why it's not available. After June of 2017, the earnings are not available so we don't use any of that data.

## Exploratory Visual:

Here is a visualization of the S&P price index over time [7]:

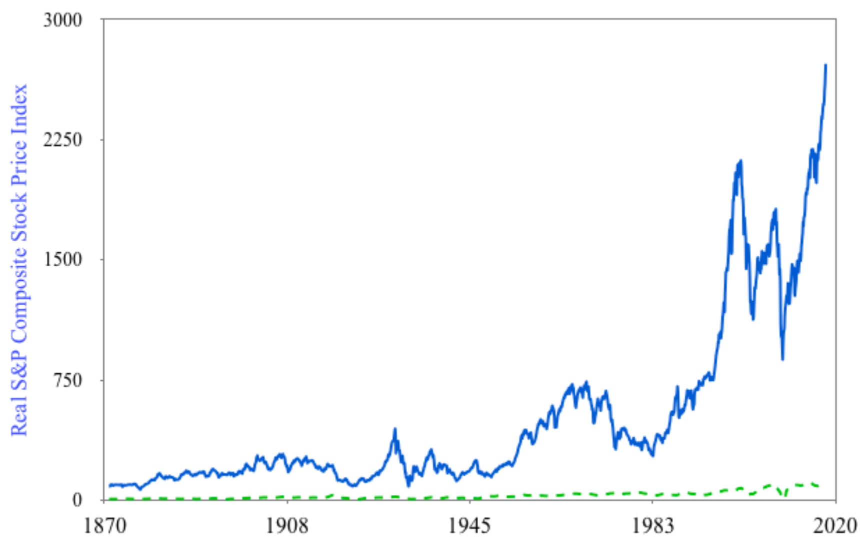

Figure 1.

You can see how the price has increased over time. After 1983 the S&P prices have skyrocketed. The dips in the markets coincide with recessions/depressions. It is a representation of the U.S. economy. That's why a lot of investors say to invest in S&P Index. Another interesting graphic is the graph of CAPE and Long Interest Rate over time [7]:

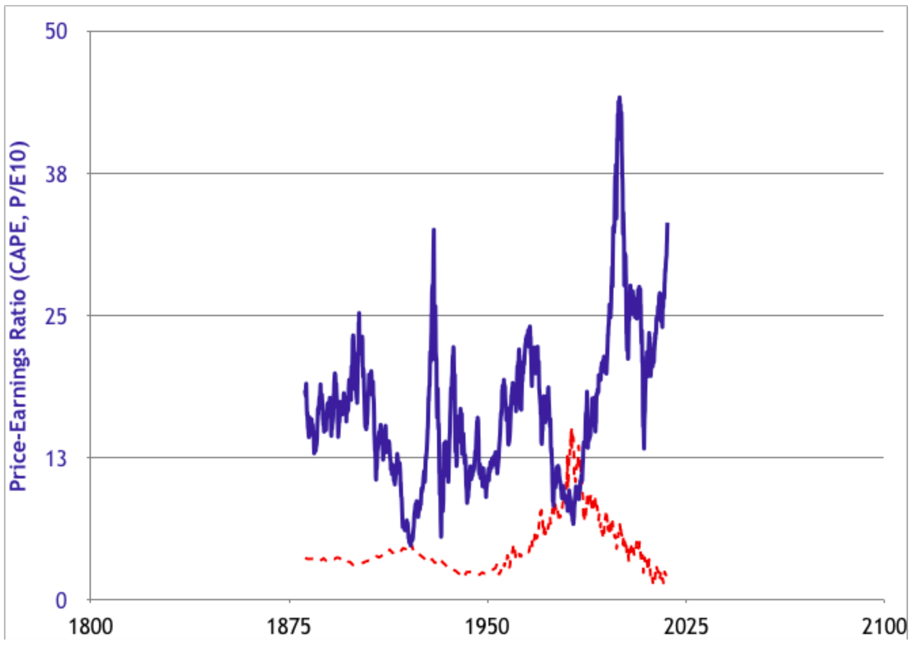

Figure 2.

This visual really shows the power of CAPE and Long interests Rates. High Price-Earning's Ratio leads to drastic falls in the market. If the price varies a lot compared to the earnings it tends to lead to market drops. Right now the Price-Earning's ratio is really high, so it will be interesting to see what happens to the market. High Interest Rates also tend to coincide with low Price-Earning's Ratio's.

## Algorithms and Techniques:

The algorithm that will be used is the epsilon greedy Q-Learning algorithm. The overview of the algorithm is that a random action can be chosen with an epsilon probability or an action can be picked based on what action has the highest score. First it is trained on the data from 1881-1975. The result will be a dictionary of scores based on adjusted parameters and actions. That dictionary will be used on a testing set from 1976-2017. Decisions on whether to buy or sell will be based on it. The Q-Learning algorithm:

$$Q(s_t, a_t) = (1 - \alpha) \bullet Q(s_t, a_t) + \alpha \bullet r_t$$

This is the Q-Learning algorithm with the discount factor set to 0. I chose to set it to 0 because I wanted the program to care only about immediate rewards at each time step. A discount factor could be set to maximize profits over time giving future rewards a bigger weightage, but I wanted to program to be able to predict even smaller market drops not just overall market trends.

Actions:

1. Hold: Keep money in the stock market and invest new $1000 in the market
2. Sell: Trade shares for money. $1000 in addition is added to money.

Parameters:

1. Rewards
2. CAPE
3. Long Interest Rate
4. Real Price
5. Real Dividend
6. $\alpha$ - Learning Rate (constant that affects rate of learning)
7. $\varepsilon$ - Decay Rate

Real Price, Real Dividend, Long Interest Rate, and CAPE will be the information used to learn. Real Price and Real Dividend will be used to calculate the money made. Real Price will also be involved in the assignment of rewards. Decisions will be made based on Long Interest Rate and CAPE. They will represent the states in the Q-Learning formula.

Robert Shiller shows that there is a correlation between future S&P and the cyclically adjusted price per earnings ratio (CAPE) [7].

"Low long-term interest rates encourage investment in new equipment and high interest rates discourage it. Investment is, in turn, a major source of economic growth. [6]"
Both of these factors are related to economic growth so that is why they will be used to decide whether to buy or sell shares.

## BenchMark:

The benchmark that the results will be compared to is based on the money made from holding in the market based on the holding algorithm from the 1976 to 2017. The result produced is $3,351,912.48 given that $1000 is invested every month.

# Methodology

## Data Preprocessing:

Since there are abnormalities in the data set, only 1881 to 2017 is considered. Not all the data is relevant. I copied and pasted The Date, S&P index, Real Price, Real Dividend, Long Interest Rate, and CAPE into a separate document and converted it to a .csv file. This data isn't easy to access  in the form of the .csv so I created a dictionary based on the month number with all the data previously mentioned. It is numbered by:

1. Data at 1881.01 is numbered as 1
2. Data at 1881.12 is numbered as 12
3. Data at 1882.01 is numbered as 13
4. Data at 1882.12 is numbered as 24
5. …..

October for each year is listed as exp: 1881.1 so it needed to be converted to a string and corrected to 1881.10

The dictionary looks like: {1: {'Real Dividend': 6.94, 'S&P': 6.19, 'Real Price': 162.1, 'Long Interest Rate': 3.7, 'Date': '1881.01', 'CAPE': 18.47}, 2: {'Real Dividend': 7.0, 'S&P': 6.17, 'Real Price': 159.96, 'Long Interest Rate': 3.69, 'Date': '1881.02', 'CAPE': 18.15}, …}

## Implementation:

The CAPE and Long Interest Rate will be used as state pairs. This means that the program will decide a result based on the CAPE and Long Interest Rate. These are continuous values so they are converted into discrete ranges. The values need to be based on the training set data. If a pair is in the testing set data, but not in the training set then it's useless. I set the bottom of the ranges for cape and long interest rate based on the minimum values of each. I set the max for long interest rate to be 8.0. If you look at figure 2, the interest rate in the data set doesn't really go beyond 8.0 so I set that to be the upper bound. For cape I made the max value to be 32, since it only momentarily goes beyond that point in the training set. Based on the max and minimum values, ranges are evenly made. The ranges have to end with negative infinity and positive infinity since it is not known what the minimum and max values are of the testing set. A parameter that can be adjusted determines the number of ranges made for CAPE and Long Interest Rate.

Based on the number of ranges, tuples will be created for each range. A list of tuples for the long interest rate and a list of tuples for CAPE. These will be combined into another tuple representing the state. For each state a score will be given for sell and hold. At first everything will be set to 0. Using the Q-Learning equation the scores will be iteratively calculated. Training Steps:

1. Set Rewards: I experimented with different rewards. Originally I made the reward values constant and the program didn't perform that well. With experimentation, I realized the best method was basing the rewards on the price difference from the current months price to the previous months price. Rewards:
    a. If money is kept in the stock market and the price goes up. The reward is 1*(Price Difference) from current month and previous month.
    b. If money is kept in the stock market and the price goes down. The reward is -1*(Price Difference) from current month and previous month.
    c. If money is taken out of the market and the price goes up. The reward is -1*(Price Difference)
    d. If money is taken out of the market and the price goes down. The reward is 1*(Price Difference)
2. With epsilon probability, hold or sell will randomly be picked. With (1 - epsilon) probability a state will be picked based on the scores of the built dictionary. In the beginning the program will be random and as times goes on it will rely more on the dictionary. I experimented with different decay functions to determine how epsilon will decay. None of them really greatly affected the results. I tried function such as $\varepsilon = \alpha^t$ and $\varepsilon = cos(\alpha t)$. The results didn't greatly differ from one another so I opted for the simplest one of the all which was $\varepsilon = 1 - .001 * t$.

3. Each month the action picked the month before is judged based on whether the stock price went higher or lower. The rewards are picked in that format. Using the state from the previous month, the score for the state, and the reward, the dictionary is updated.

An example dictionary:
Learning_dictionary = {((3.0, 3.75), (14.89, 19.945)): {'sell': -7.098516397196308, 'hold': 13.565831608341679}, ((-inf, 2.25), (14.89, 19.945)): {'sell': -8.090746249999997, 'hold': 17.78906006853524}, ((2.25, 3.0), (14.89, 19.945)): {'sell': -14.705331760710935, 'hold': 6.660773877428409}, …. }

Testing:

1. Use Learning Dictionary from Learning phase for testing.
2. Loop from 1976 to 2017
3. Check state in dictionary and pick the action with max value

### Refinement:

One of the main things that needed to be refined was the max for ranges on CAPE and Long Interest Rate. Originally I didn't set those and I got a learner dictionary with many empty values. Each time you run the learning phase will get you different results because of the nature of randomness in Q-Learning algorithm, but then I noticed a weird thing occurring. Each time the same dictionary was used on testing, different results would occur. Untouched dictionary entries had a score of 0 for actions 'sell' and 'hold' leading to a random selection on the action. This leads to drastically different results. Eventually I noticed that the ranges for CAPE and Long Interest Rates primarily affected this. This is due to learning CAPE and Long Interest Rates varied from testing CAPE and Long Interest Rates.

Originally I used the best value produced on the learning set from multiple runs. I would get answers like 150 million dollars made during the learning set which was almost triple what was made by holding in the market. Those dictionaries applied on the testing set didn't perform well so I realized that in these cases it was fitting to the learning set well, but it wasn't generalizing. That's why I ran 10 runs to find the parameters with the highest average.

Rewards had to be changed. At first I used constant rewards for each of the four outcomes. For the positive ones I used +1 and the negative ones I used -1. I tried varying these, but it just made the program perform worse. The problem with using constant rewards is that making $1 should be rewarded differently than $10. Then I made the rewards based on the price difference from month to month and it made the program perform a lot better.

# Results

## Model Evaluation and Validation:

In order to determine what parameters had the best model, all combinations of alpha from .01 to 1.0, long interest rate from 2 to 10 ranges, and cape from 2 to 10 ranges are applied to Learning. The parameters which made the most money on the Learning set are then applied to Testing. I looked at the results for 10 different runs. They varied a lot so then I decided to determine the parameters by looking at all combinations and finding what has the highest average on the Learning set. Now using this parameter out of 10 additional runs I take the best Learning Dictionary produced. The learning dictionary produced made $43,893,335.40 which is less than $55,323,777.79 that was made just by holding all the money in the market. This isn't necessarily a bad thing because it may have made the wrong decisions in the beginning as it was learning giving a low amount of money earned in the learning set.

Now running the learning dictionary on the testing set makes a wide range of results so I decided to run it 10 times and look at the best result which is $2,588,032.21

## Justification:

Using the percentage difference formula, the result is -22.79% from the bench mark. That means that it performed worse than the benchmark results. Most of the results performed in the 2 million range, but once in a while it performs better. When I started the project I thought that the market could be timed, but during the project I started realising that it was not a good idea. Earlier I talked about when making the ranges for the long interest rates and cape I had to limit the max value for each of them. Before I did that I noticed a lot of states that were untouched in the learning dictionary and it wasn't performing all that well. The CAPE values for the training set are mostly below thirty two, while the cape values in the training set go really high. This can be seen in figure 2. During the testing it's not really using information that it has learned from. Similarly long interest rates go up really high in the 1980's and in the training set it never reaches that point. The testing set is not really using accurate information. Since I set the max values to a lower limit, it's using information from lower values to make decisions for it, which may not be accurate. Actually given the results, it's not accurate. In figure 1 you can see that the training set has S&P growth rates that are significantly lower. The S&P values in the testing set sky rocket after the 80's. A period of less growth is being used to predict a period of high growth. In order for a machine learning algorithm to accurately work, it needs to look at data similar to what it has seen before.

Sometimes the data performed well and I originally I was trying to optimize results to bring out values in the 3-4 million range. I started to realize that the reason it sometimes performed well was that the learning dictionary would have states that were untouched. Sell and Hold had a value of 0 for many pairs. An action is randomly picked if values for them are tied in the dictionary. That means that due to randomness it performs well. Using one set of parameters you can optimize it so that there are minimal ties in the dictionaries, but as soon as the parameters change that is not the case. Going through all the parameters and choosing the best does not allow optimization for maximal values in the dictionary.

A valid question is why did I organize the training and testing sets like this? Why not take random intervals of the data and train on them? I wanted to make it more accurately represent real life. In real life we want to use past data to predict future events. I wanted to create a program that could tell me whether to sell or buy based on monthly S&P prices. I realized in this project that it is a lot better to follow the advice of the investors and just hold your money in the index.

## Conclusion:

Here is what the results look like:

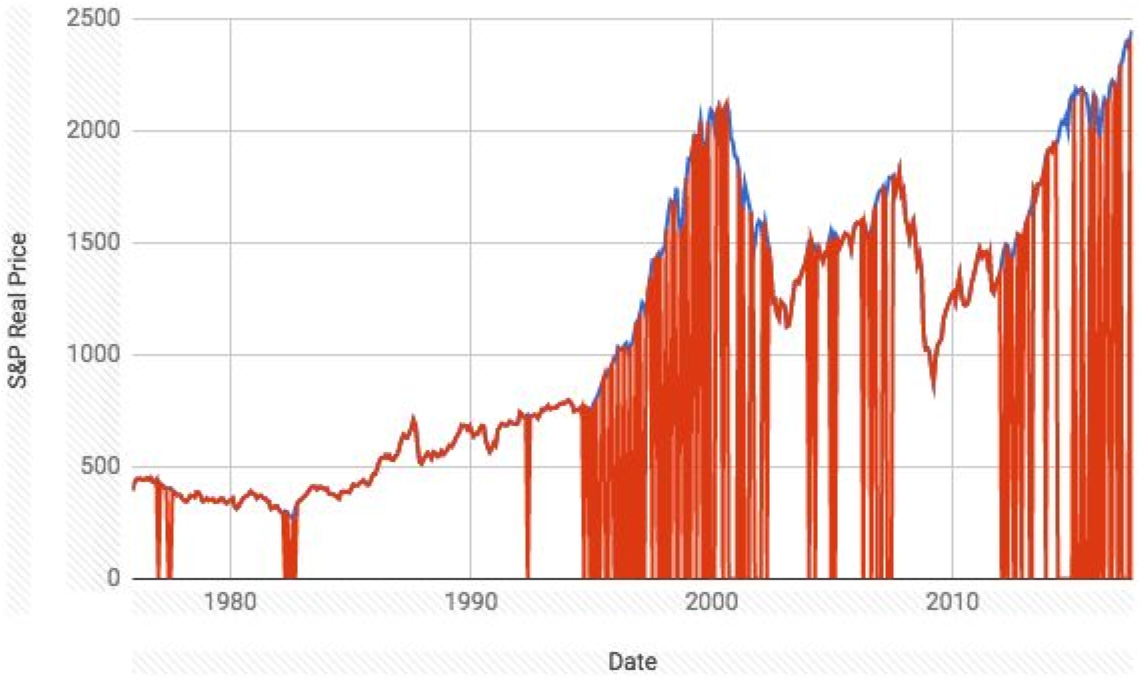

Figure 3.

The areas where the graph is blue are where stock is sold. The blue lies in between vertical lines. Let's zoom in after 1990 to where most of the vertical lines begin:

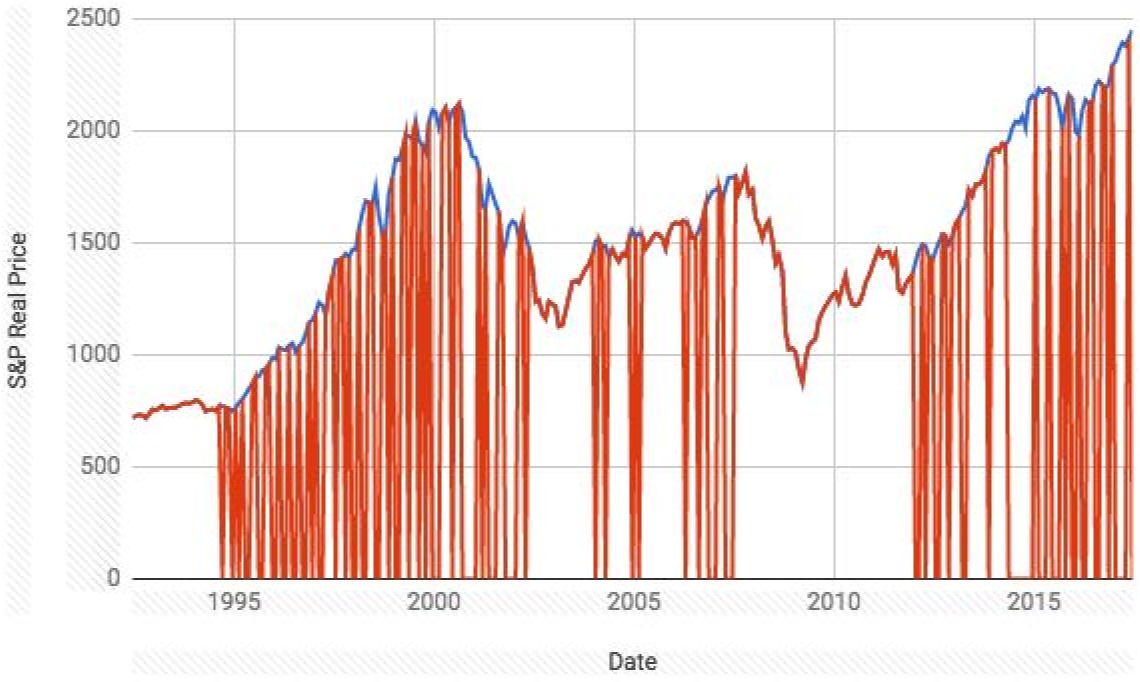

Figure 4.

We can see that it mostly sold up to the dot com boom in 2000. Now although an optimal program would have sold close to 2000 it may not be such a bad result. High CAPE values are associated with instability in the market so the program selling during the boom may not be such a bad thing. It starts selling again in the last couple of years of the testing set. The market is going exceedingly high which could be a sign of a coming crash. The point is that it did learn from the data, but it is not able to accurately determine the best times to buy and sell. It missed out on a lot of money and performed less than holding money in the market.

For one thing is that 100 years of data is not enough to learn off S&P. Enough time has not passed to fully see how the market performs. The other thing is that there are so many factors that can affect the market. The 2008 housing crisis was due to the banks and can't be predicted by market data. At this point the best thing to do is hold your money in the market.

## Reflection:

The process involved using the Q-Learning algorithm to see if it was possible to use the algorithm to time the market. Using Long Interest Rates and Cyclically Adjusted P/E Ratio can predictions be made for when the market will go up or down. Can this information be used to come out with more money than keeping stocks in the market. Iterative Q-Learning algorithm learns on the first 100 years of market data to come up with a dictionary based on Long Interest Rate and CAPE pairs to use on the last 40 years of market data. Based on the Long Interest Rate and CAPE for each month, the dictionary will tell whether to sell shares or hold.

The difficult part of this project was to try to get the best results. The more I looked at the results, the more I realized weird things that were occuring. One thing I noticed originally was that a lot of the dictionary entries were untouched and so I had to set limits to CAPE and Long Interest Rates. This took me a while to figure out. Another big thing was that results would change when testing everytime I ran the program even though the testing dictionary was the same. The reason was that untouched dictionary entries would lead to random selection of hold or sell which would greatly change results. A big realization was how much randomness was affecting the results. Not just from this, but the nature of the Q-Learning algorithm. Randomness is used to search the field. Really there wasn't enough data to work with, leading results to be drastically different.

## Improvement:

I would use a function approximator rather than a dictionary state pairs for state action scores. The problem was that I converted continuous values into discrete values and a function would work a lot better for this. I would be able to plug in values of long interest rates and cape in a formula and get values from it, rather than mapping specific ranges of long interest rates and cape together.

Overall, I don't think that Q-Learning works well on this given the scarcity of data. The main thing is that the data from the first 100 years is so different from the last 40 years. The U.S. economy and the stock market have grown so fast in the last 40 years, nothing like before. I think machine learning might not be the best method for this. If I were to still use machine learning learning on this topic, I think I would use supervised learning instead. That might help deal with the lack of data better than Q-Learning.

## Sources:

[1]
https://www.investopedia.com/articles/investing/091015/how-dividends-affect-stock-prices.asp

[2]
https://www.cnbc.com/2017/05/12/warren-buffett-says-index-funds-make-the-best-retirement-sense-practically-all-the-time.html

[3]
http://www.simplestockinvesting.com/SP500-historical-real-total-returns.htm

[4]
https://www.investopedia.com/articles/basics/03/052303.asp

[5]
https://www.investopedia.com/terms/c/consumerpriceindex.asp

[6]
https://data.oecd.org/interest/long-term-interest-rates.htm

[7]
http://www.econ.yale.edu/~shiller/data/peratio.html

[8]
https://www.investopedia.com/terms/s/sp500.asp