# Machine Learning Engineer Nanodegree

## Capstone Project

Youness Assassi  
February 2nd, 2018

## I. Definition

### Project Overview

# Using machine learning to solve equity trading problems is something that I am highly interested in. According to JPMorgan[[1]](#footnote-1), computer generated trades account for almost 90% of all trading volume as of February 2018. Where does this leave the small investor who does not have access to the same resources as large hedge funds and investment banks do? Resources such as computers capable of high frequency trading and expert statisticians that use quantitative analysis to beat the market.

# 

# After the market crash of 2008, I decided to get an MBA with focus on finance in order to understand why the markets fluctuate the way they do. I believed that it was important for me to gain more knowledge about the financial market so I do not make the same mistake that others made when they suffered major losses during the recession. My takeaway at the time was to not try to play the market as no one beats it in the long run. Now, with the advent of machine learning, I am not so sure that this theory holds anymore. The goal of this research is to find out if the market can be beat using some of the new theories in conjunction with machine learning and automation. This will be a breakthrough for someone like me as I may be able to invest a small amount of money and let an automated system trade on its own with the goal of maximizing profits with lower risk.

# There is ample amount of research being done in this field, but much of it is not published for the obvious reason that the sponsors of the research would like to keep the information to themselves. Luckily, Udacity with the help of professor Tucker Balch of Georgia Tech has published a free course online called Machine Learning for Trading.[[2]](#footnote-2) In this course, the professor explains how hedge funds use different machine learning methods to devise strategies that can beat the market. Another research paper with the same name was published by Gorden Ritter[[3]](#footnote-3), a professor at NYU where he shows how machine learning, specifically reinforcement learning or Q learning can be applied in long term portfolio management problems.

### Problem Statement

Given only the freely available historical stock market information such as adjusted stock price and volume, can a stock portfolio be actively managed for a period of time that will beat the returns of the S&P 500? The goal of this project is to build an application that can generate an optimized portfolio of stocks given a certain amount of money, then adjust the portfolio as the market changes. The portfolio returns will have to beat the market returns within a defined time period of no more than 1 day in the future through a simulation using real market data. This is regression type of a problem, where I will use the S&P500 statistics as input and individual stock price as output. Some of the input variables I will be trying out will be the sharpe ratio, bolinger bands and volatility.

### Datasets and Inputs

I will be using the S&P 500 historical stock prices[[4]](#footnote-4) that includes the high, low, open, close and adjusted close of each stock. Out of the 505 stocks, my application will generate a portfolio of about 10 stocks. This portfolio will be optimized for its high Sharpe ratio value with a percentage allocation for each of the stocks. The idea is to then analyze these stocks by generating a number of statistics that can be used as input variables or features. Some of these statistics include the Bolinger Bands, momentum and volatility, while the future stock price will be the label or output variable. The model will be trained using a number of days in the past with a target price of 1 day in the future. I will be using TimeSeriesSplit from Sklearn to split the data between training and testing. This will ensure the proper evaluation of the model as it will not get access to future data.

The goal is to eventually be able to train the model with the most up-to-date information on stock performance so that it can be able to generate recommendations around necessary adjustments to the portfolio in order to maximize its return.

According to Tucker Balch[[5]](#footnote-5), technical analysis can be used to build a short term strategy (hours to a handful of days) that is able to beat the market which I plan to use here. The opposing strategy called fundamental analysis would be more appropriate for a long term strategy (months to years).

**Solution Statement**

The plan is to start with a portfolio of the entire S&P 500, then based on the historical performance of the stocks that make up the S&P 500, generate a portfolio of about 10 stocks that is optimized using the Sharpe ratio. This would ensure that we have a portfolio with high return and low volatility. The next step is to generate other statistics for each of these stocks, such as the Bolinger Bands, momentum and volatility that can be used as features for our learning algorithm. The label of course will be the stock price 1 day in the future.

The plan is to try different algorithms to generate the model, including Linear Regression, KNN and Random Forests. Once the model is trained with the features from each of the stocks, I will use the model to predict the portfolio price 1 day into the future and then compare it with the actual price. The total portfolio value will then be compared to the value of S&P index.

### Metrics

### The benchmark model that I will be using is the performance of the S&P 500 during the same period used for the generated portfolio. Since models built based on technical analysis do not perform well in the long term, the comparison will be limited to 1 day in the future.

After generating the portfolio of stocks, the application will perform a comparison of the generated portfolio with the returns of the S&P 500 in the same period. The formula used will be:

P1 = Total portfolio value at beginning of testing period

P0 = Total portfolio value at end of testing period

The application will also utilize the Root Mean Squared Error of Predictions as a metric to evaluate the performance of each of the models given that this is a regression problem.

$y_i$ \begin{displaymath}RMS Errors= \sqrt{\frac{\sum_{i=1}^n (\hat{y_i}-y_i)^2}{n}}\end{displaymath}

$\hat{y}_i$is the actual stock price value

is the predicted value

n is the number of test data rows

## II. Analysis

### Data Exploration

The data used for this project was retrieved from yahoo finance. The module ‘processing.start’ handles the automatic retrieval all of S&P 500 historical stock information since 2006. New data can be retrieved by typing ‘python –m processing.start’ in the command line from the root of the project.

The stock daily information initially includes the following parameters:

* Date: the date the was traded on
* Open: The price of the stock at the market open
* High: The highest value the stock reached that day
* Low: The lowest value the stock reached that day
* Close: The price of the stock at the market close
* Adj Close: The closing price with consideration of other factors such as stock splits and dividends.
* Volume: The trading volume of the stock on that day.

In order to achieve the first goal of finding an optimal portfolio based on historical data, the first step the application performs is to collect The Date and Adjusted close values from all of the stocks that make up the S&P 500. Unfortunately, not all of the tickers will have a complete set of data from the desired start date. For that reason, we apply a number of techniques to fill the missing data as much as possible. The forward fill methodology is the first one I used so that the last known value is available at every time point. If the last known value is not available then I applied the backward fill methodology, only this time we borrow the value from the future incase forward fill fails. If any cells are still left empty then we drop them at this point.

The next step is to generate the optimal portfolio combination of stocks. For this step, I used the entire set of S&P stocks as input, then applied scipy.optimize which tries to reduce the error by finding the highest Sharpe ratio of the stock combination.

Sharpe ratio =

The next step is to apply machine learning to try and predict the value of the suggested portfolio. This would allow us to decide whether we should invest in the portfolio or not. The data I used as features for machine learning algorithm includes:

* Bollinger Bands®: two standard deviation away from the stock simple moving average
* Volatility: standard deviation of the stock
* Momentum: the rate of acceleration of a stock’s price.
* 5 day Simple Moving Average
* Daily Returns

### Exploratory Visualization



PGR feature Correlation

The figure above is a representation of the correlation between the different features for the PGR Stock Symbol data for year 2017. The adjusted close and next day adjusted close prices are left here for display only and will not be used as features to train the regression models.

As can be seen on the heatmap figure, there is a strong negative correlation between the Simple Moving average and the lower Bolinger Band. There is also a high correlation between Momentum and the Simple Moving average. I will most likely be removing some of these highly correlated features from the model training as they will only introduce noise and complexity to the model.

### Algorithms and Techniques

The first technique that I will be using is to generate a portfolio of stocks with the goal of maximum return at a lower risk. To do that I will first generate a portfolio composed of all 505 equally weighted stocks from the S&P 500. Then use an optimizer called scipy.optimize that will try to reduce the error in the portfolio by maximizing its Sharpe ratio. I will be using the following parameters for the minimize function of scipy.optimize:

* Error function that optimizes for the Sharpe ratio
* Method name: SLSQ
* Bounds: between 0 and 1 for weights so that we do not end up with more than 100% allocation for a single stock

What we end up with is a portfolio of 10 stocks or less, with various weights allocated to each one of these stocks. The next step is to predict the future price of each of stock from the new portfolio. This will help determine when it would make sense to buy and sell the individual equities making up the portfolio. For this step, I will be training and testing the models using various regression algorithms with different parameters including:

* Linear Regression
  + No parameter setting
* Support Vector Machines Regression
  + Kernel: rbf, poly and linear
  + C: 1e3
  + Gamma: 0.1
  + Degree: various numbers
* Nearest Neighbor Regressor
  + n\_neighbors: 3, 5, 7
* Random Forest Regressor:
  + Max\_depth: various numbers
  + Random\_state: 0 to ensure we get the same result every time.

The Support Vector Machine algorithms perform poorly when trained against the original distribution of the data. For that, I will be using StandardScaler to preprocess the data so that it is well normalized.

Also, since this is a time series regression problem, I will have an issue when splitting the data for testing and training using the standard cross validation techniques. Instead, I will be using TimeSeriesSplit which always provides testing data that the model has never been trained on. This will ensure that the results we get are not tainted by the fact that the model was able to peak into the future.

### Benchmark

The benchmark model that I will be using is the performance of the S&P 500 during the same period used for the generated portfolio. Since models built based on technical analysis do not perform well in the long term, the comparison will be limited to 1 day in the future. I will also use a secondary model as a benchmark to see if the selected model performs better or worse.

The formula used will be: (P1 – P0)/P0 where P1 represents the total value at the end of the test period and P0 represents the total value at the beginning of the testing period. I will also use the Root Mean Squared Error of Predictions as a metric to evaluate the performance of each model. The model with the lowest RMSE will be the one I use for predicting the portfolio value.

## III. Methodology

### Data Preprocessing

After scraping the stock information from Yahoo Finance, the next step was choosing the relevant information. The downloaded data from yahoo finance includes the following columns:

* Date: the day of the trading
* Open: the value at the beginning of the day
* High: the highest value the stock traded at during the day
* Low : the lowest value the stock traded at during the day
* Close: the value at the end of the day
* Adjusted Close: the close value adjusted for stock splits and dividends
* Volume: the number of times the stock was traded that day

Only the Adjusted Close and Date will be necessary from this data set for the next step which is to generate the rest of the features we will be using. One of the issues I faced here is that some of the stocks either did not exist from the day we need the information, or for some reason had values during the weekend when the stock market was closed. So before we do any calculation, we need to make sure we have the correct the dataset.

By using Pandas Dataframe, it is easy to fill out empty cells with relevant information or drop certain rows completely. In this case, we combine all of the stocks data in a Dataframe and then add S&P index data. The next step is to drop the dataframe rows where we have empty S&P cells. So that solves the issue with stocks with values that do not correspond to the S&P trading dates. We still have other stocks that did not trade at all in certain periods, and for those we use a forward fill methodology in which we fill the adjusted close values of a certain stock with the last known value. Now if those stocks do not have any last known value, we use a backward fill methodology where we use the first value we have to fill the past dates with missing information.

If for any reason we still have empty cells, then we just drop the rows entirely.

The next step is to generate the various features we need. I created Ticker and TickerAnalysed classes for this reason. Ticker is responsible for collecting the ticker data and cleaning it up, while TickerAnalysed which is a sub-class of Ticker is responsible for calculating the different statistics and that includes:

* Momentum: a trend in the direction of the price in a 5 day window
* Simple Moving Average: The price average over a 5 day window
* Upper Bollinger Band: Two standard deviation above the mean in a 5 day window
* Lower Bollinger Band: Two standard deviation below the mean in a 5 day window

Because of the high correlation between some of the features described above, I will most likely be dropping some of them if I do not see any performance update. Also, since some of the regression algorithms, namely the Support Vector Machine Regression algorithms, require data to be normalized, so that will be done right before we run the data as normalization does not lead better results from other algorithms like K Nearest Neighbor and Linear Regression.

### Implementation

Now that we have collected and refined our features,

In this section, the process for which metrics, algorithms, and techniques that you implemented for the given data will need to be clearly documented. It should be abundantly clear how the implementation was carried out, and discussion should be made regarding any complications that occurred during this process. Questions to ask yourself when writing this section:

* Is it made clear how the algorithms and techniques were implemented with the given datasets or input data?
* Were there any complications with the original metrics or techniques that required changing prior to acquiring a solution?
* Was there any part of the coding process (e.g., writing complicated functions) that should be documented?

### Refinement

In this section, you will need to discuss the process of improvement you made upon the algorithms and techniques you used in your implementation. For example, adjusting parameters for certain models to acquire improved solutions would fall under the refinement category. Your initial and final solutions should be reported, as well as any significant intermediate results as necessary. Questions to ask yourself when writing this section:

* Has an initial solution been found and clearly reported?
* Is the process of improvement clearly documented, such as what techniques were used?
* Are intermediate and final solutions clearly reported as the process is improved?

## IV. Results

(approx. 2-3 pages)

### Model Evaluation and Validation

In this section, the final model and any supporting qualities should be evaluated in detail. It should be clear how the final model was derived and why this model was chosen. In addition, some type of analysis should be used to validate the robustness of this model and its solution, such as manipulating the input data or environment to see how the model’s solution is affected (this is called sensitivity analysis). Questions to ask yourself when writing this section:

* Is the final model reasonable and aligning with solution expectations? Are the final parameters of the model appropriate?
* Has the final model been tested with various inputs to evaluate whether the model generalizes well to unseen data?
* Is the model robust enough for the problem? Do small perturbations (changes) in training data or the input space greatly affect the results?
* Can results found from the model be trusted?

### Justification

In this section, your model’s final solution and its results should be compared to the benchmark you established earlier in the project using some type of statistical analysis. You should also justify whether these results and the solution are significant enough to have solved the problem posed in the project. Questions to ask yourself when writing this section:

* Are the final results found stronger than the benchmark result reported earlier?
* Have you thoroughly analyzed and discussed the final solution?
* Is the final solution significant enough to have solved the problem?

## V. Conclusion

(approx. 1-2 pages)

### Free-Form Visualization

In this section, you will need to provide some form of visualization that emphasizes an important quality about the project. It is much more free-form, but should reasonably support a significant result or characteristic about the problem that you want to discuss. Questions to ask yourself when writing this section:

* Have you visualized a relevant or important quality about the problem, dataset, input data, or results?
* Is the visualization thoroughly analyzed and discussed?
* If a plot is provided, are the axes, title, and datum clearly defined?

### Reflection

In this section, you will summarize the entire end-to-end problem solution and discuss one or two particular aspects of the project you found interesting or difficult. You are expected to reflect on the project as a whole to show that you have a firm understanding of the entire process employed in your work. Questions to ask yourself when writing this section:

* Have you thoroughly summarized the entire process you used for this project?
* Were there any interesting aspects of the project?
* Were there any difficult aspects of the project?
* Does the final model and solution fit your expectations for the problem, and should it be used in a general setting to solve these types of problems?

### Improvement

In this section, you will need to provide discussion as to how one aspect of the implementation you designed could be improved. As an example, consider ways your implementation can be made more general, and what would need to be modified. You do not need to make this improvement, but the potential solutions resulting from these changes are considered and compared/contrasted to your current solution. Questions to ask yourself when writing this section:

* Are there further improvements that could be made on the algorithms or techniques you used in this project?
* Were there algorithms or techniques you researched that you did not know how to implement, but would consider using if you knew how?
* If you used your final solution as the new benchmark, do you think an even better solution exists?

**Before submitting, ask yourself. . .**

* Does the project report you’ve written follow a well-organized structure similar to that of the project template?
* Is each section (particularly **Analysis** and **Methodology**) written in a clear, concise and specific fashion? Are there any ambiguous terms or phrases that need clarification?
* Would the intended audience of your project be able to understand your analysis, methods, and results?
* Have you properly proof-read your project report to assure there are minimal grammatical and spelling mistakes?
* Are all the resources used for this project correctly cited and referenced?
* Is the code that implements your solution easily readable and properly commented?
* Does the code execute without error and produce results similar to those reported?

# Sentdex Python Programming for Finance

<https://www.youtube.com/watch?v=W4kqEvGI4Lg>

Udacity Machine learning for trading

<https://github.com/arwarner/machine-learning-for-trading>

Siraj- Predicting Stock Prices

<https://www.youtube.com/watch?v=SSu00IRRraY>

Ticker class

<https://github.com/voyageth/trading/blob/master/capstone.ipynb>

<https://github.com/jessicayung/machine-learning-nd/blob/master/p5-capstone/report.md>

1. https://www.cnbc.com/2017/06/13/death-of-the-human-investor-just-10-percent-of-trading-is-regular-stock-picking-jpmorgan-estimates.html [↑](#footnote-ref-1)
2. https://www.udacity.com/course/machine-learning-for-trading--ud501 [↑](#footnote-ref-2)
3. Gordon Ritter <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3015609> [↑](#footnote-ref-3)
4. <https://github.com/younessassassi/machine-learning-udacity-capstone/tree/master/data/stock_dfs> [↑](#footnote-ref-4)
5. https://www.cc.gatech.edu/~tucker/ [↑](#footnote-ref-5)