Machine Learning Engineer Nanodegree
Capstone Project – Predicting the Direction of Monthly S&P 500 Returns
Jack St. Clair
February 16, 2018

I. Definition

a. Project Overview

At the highest level, this project is a binary classification problem. The target class is the direction of the after-cash return of the S&P 500 in any given month with a positive return denoted as a '1', and a negative return as a '0'. Now, there are countless many ways to attempt this problem ranging from pure technical analysis using only prior levels of the S&P 500 to fundamental analysis using metrics such as the market's P/E level, among others. The features I intend to use are inspired by the macroeconomy. These will include factors such as inflation, the fed funds rate, and the strength of the USD, among others. In macroeconomics, these factors help define what is known as the short-run model of the economy summarized by GDP. For this project, the monthly return of the S&P 500 will be used as a proxy for the monthly change in GDP.

b. Problem Statement

The general framework for the machine learning model is that it will receive inputs that are available at the beginning of the month, such as the month-over-month change in inflation, and generate a prediction from them. Data going back to 1986, over 350 months, from the St. Louis Federal Reserve (for the macroeconomic data) and Yahoo! Finance (for the S&P 500) are used to train the different machine learning techniques that will be tested.

As stated above, the goal of this project is to correctly classify winning months from losing months of the S&P 500's after-cash return based on several different macroeconomic indicators such as inflation and the fed funds rate. Each sample point will consist of a label, either '1' or '0' and four pieces of macroeconomic data that would have been available at the *beginning* of the month. These sample points will create a scatter plot in multi-dimensional space and the goal of the machine learning algorithms will be to separate the positively labeled classes from the negatively labeled classes based on the macroeconomic indicators, and then to generate predictions based on unseen data for other months.

c. Metrics

Given the project's goal is to predict the after-cash return of the S&P 500, it only makes sense to have a metric that captures the purpose of the end user, to make money trading the S&P 500. An important distinction is that losing money is more frustrating than not making money. Incorrectly predicting a winning month is worse than incorrectly predicting a losing month, all else equal. The metric used should reward models that are more conservative than other models as opposed to simply using the accuracy of each model. Thus, a F-score will be used to measure the performance of the models. More specifically, a F-0.5 score will be used so that more emphasis is placed on precision as compared to recall.

II. Analysis

a. Data Exploration

The labels of the training data, as stated earlier, are whether or not the S&P 500 exhibited positive after-cash returns going back to February of 1986. A very important feature of this dataset is that it is not balanced. Roughly 60% of the months in the entire sample yield positive after-cash returns as you might expect given the S&P 500's historical success. This is yet more motivation to consider an alternative metric to accuracy, as a naïve predictor (always predicting a '1') will have about 60% accuracy.

Now, let's discuss the features being used in the study. First, a measure of the strength of the US Dollar. One can expect that when the strength of the US dollar appreciates, foreign investment into the US goes down[1]. To see why this is the case, assume that 1 USD currently can be exchanged for 1 Euro, and that the US Dollar strengthens so that 1 USD can now be exchanged for 2 Euros. When this happens, foreigners would have to pony up double the Euros for a good made in the US now than they had to prior to the appreciation of the US Dollar, and thus, foreign investment can be expected to fall in the short term. This decrease, all else equal, would lead to a short-term decline in US output. For this reason, the change in strength of the US Dollar will be included as a feature. But, what about the overall level of the US Dollar one might ask. According to Jones' "Macroeconomics", the total level of US Dollar strength is not part of the short-term model because the economy adjusts and reacts which motivates the inclusion of only the change in its level[2].

Another important feature that will be used in the study is the Fed funds rate. This is the interest rate charged on overnight loans between banks and is by and large determined by the Fed itself. It is one lever the Fed can pull to either stimulate or slow down the economy. As a brief corollary, it is important to understand the anatomy of interest rates. The nominal interest rate (the most common rate that is quoted) is equal to a combination of the real interest rate and inflation. For simplicity, the nominal interest rate is most intuitively understood as the real interest rate plus inflation (although it is slightly different mathematically). How the Fed comes into play is through setting the overnight nominal interest rate (the Fed funds rate). Now, because changes in inflation take a while to percolate through the economy, when the Fed changes the Fed funds rate, inflation is by and large unchanged[3]. This means that the real interest rate tracks the change in the Fed funds rate in the short term. Putting this all together implies that a rise in the Fed funds rate results in a rise in the real interest rate which in turn slows the economy[4]. To see why this is the case, consider a simple example of a pizza shop who has the option of investing $100 into a new oven which would return 5%, or saving the money and having it return 4%. Before the change, the pizza shop would obviously invest in the oven, increasing short-term GDP. After the change in interest rates however, the oven would still return 5% after one year but now the investment in the bank could return 6% in which the shop would then save the money and thereby foregoing a short-term increase in GDP. This simple example shows what happens on a grander scale when interest rates rise, the economy slows, all else equal. Given the above logic, the change in the Fed funds rate will be used as a feature in this classification problem.

---

[1] "The Open Economy in the Short-Run Model." Macroeconomics: Economic Crisis Update, by Charles I. Jones, 2nd ed., Norton, 2011.

[2] "The Open Economy in the Short-Run Model." Macroeconomics: Economic Crisis Update, by Charles I. Jones, 2nd ed., Norton, 2011.

[3] "Monetary Policy and the Phillips Curve." Macroeconomics: Economic Crisis Update, by Charles I. Jones, 2nd ed., Norton, 2011.

[4] "Monetary Policy and the Phillips Curve." Macroeconomics: Economic Crisis Update, by Charles I. Jones, 2nd ed., Norton, 2011.

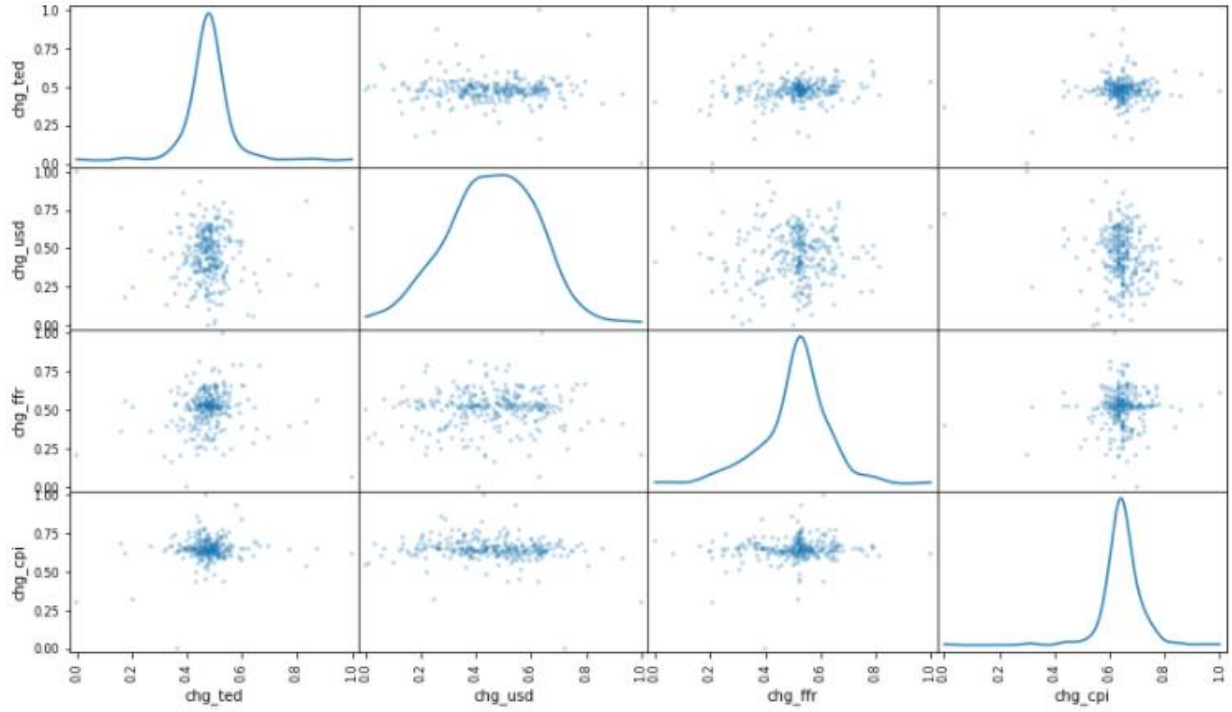
The third feature that will be used for training is inflation. As a note, CPI (consumer price index) will be used as the proxy for inflation. How inflation comes into the short-run model incorporates the Fed. In the short-run model of the economy, there is an expectation for what inflation will be, typically some complex average of recent CPI data and some expectation for how the Fed will act[5]. It is therefore how actual inflation comes in relative to what was expected that impacts the economy. However, to simplify this, last month's inflation number will be used as the expectation for inflation. This difference will be the feature used in the model. As an example of how inflation can affect the economy, let's assume inflation has been rising faster than expected, the Fed would raise the Fed Funds rate which in turns reduces short-run output due to the lack of spending and investment caused by the increase in the cost to borrow.

Last on the list of features that will be used is what is known as the TED spread which is a proxy for the level of the risk premium in an economy[6]. As an example, if the current nominal interest rate is 5%, but some company is at risk of going bankrupt, a bank will charge that company more than 5%, say 8% (the difference, 3%, is the "risk premium"), because the bank must be compensated for taking on the risk that the company would default, not paying back its loan! Therefore, in bad economic times, a high level of the TED spread is often observed. Mathematically, the effective interest rate between banks is the nominal interest rate plus the risk premium (the TED spread)[7]. This implies that at times, the Fed cannot even control the economy in its normal ways because if the risk premium is high, it largely determines the nominal interest rate[8]. For these reasons, the TED spread will be included. To be exact, the TED spread is the difference between the interest rate banks borrow and lend from each other, 3-month LIBOR, and the interest rate offered on short-term treasury bills, 3-month T-bills, which is considered risk-free. Much like the other features, it is the change in the TED spread that will be used as a feature.

b. Exploratory Visualization

These four features, once scaled to a range between zero and one, all share a fairly normal-like distribution. Visualization of the distributions for the four features, change in USD strength, change in the Fed Funds rate, change in the TED spread, and change in Inflation after scaling is below.

[5] "Stabilization Policy and the AS / AD Framework." Macroeconomics: Economic Crisis Update, by Charles I. Jones, 2nd ed., Norton, 2011.
[6] "The Great Recession and the Short-Run Model." Macroeconomics: Economic Crisis Update, by Charles I. Jones, 2nd ed., Norton, 2011.
[7] "The Great Recession and the Short-Run Model." Macroeconomics: Economic Crisis Update, by Charles I. Jones, 2nd ed., Norton, 2011.
[8] "The Great Recession and the Short-Run Model." Macroeconomics: Economic Crisis Update, by Charles I. Jones, 2nd ed., Norton, 2011.



c. Algorithms and Techniques

K-Nearest Neighbors, KNN for short, will be one of the models tested. KNN is nice because it has a very intuitive form and can be easily applied to this economic setting. The basis for KNN is that given a bunch of data, when asked for a prediction, KNN will look at the K closest samples, average them, and then output a prediction. In terms of this economic setting, when trying to predict a given month's class, the algorithm will look to other examples of months that had similar data and use them to make a prediction. However, with a dataset of this size (roughly 350 samples), I would expect KNN to struggle given the lack of coverage the data has over the possible feature values. That is the use of four features, each taking values over six different ranges (i.e., 1-z above, 2-z above, etc.), there would be six to the power of four different possible combinations, 1296, of them to be exact, roughly three and a half times larger than the number of samples we actually have. This issue, an example of the curse of dimensionality, means that it will be very difficult for KNN to perform well since it will be making predictions based on the average of samples that may not actually be very close to the testing sample.

Another neatly intuitive algorithm that will be used is a decision tree classifier. Decision trees essentially ask many questions of the samples features (i.e., whether there was a change in the fed funds rate or not) in such a way that best differentiates the two classes. This setup will eventually keep asking questions until the samples are correctly classified and thus can be very prone to overfitting using an out-of-the-box parameter setting. Therefore, parameters such as 'min_samples_split', the allowable number of samples remaining after a given question is asked, will likely require attention. While the intuition and application of a decision tree seems like it could work very well, the small dataset is concerning as enough meaningful questions may not be able to be asked due to the lack of data and thus hitting the 'min_samples_split' limit.

The third algorithm that will be trained is a support vector machine, SVM for short. This model will do its best to separate the data based not on probabilities, but on the distance between distinct points belonging to separate classes. In this project, distance will be measured using the 'rbf' kernel, the

default setting. Like the other models, the SVM model will require parameter tuning. Both the 'C' and 'gamma' may need tuning. 'C' controls the degree of regularization whereas 'gamma' controls the impact of a given single point on the decision boundary.

d. Benchmark

For this project, a logistic regression model will be used as the benchmark for this paper. The reason is that logistic regression is a very simple, yet powerful model that works well with linearly separable data and has a very intuitive output, a probability that a given input belongs to the target class. For future reference, below is the performance of the benchmark model on the training and cross-validation sets.

	LogisticRegression
acc_cv	0.645833
acc_train	0.599303
f_cv	0.695067
f_train	0.651515

III. Methodology

a. Data Preprocessing

First, macroeconomic data was gathered from the FRED database and month-over-month changes were taken[9]. CPI and USD data were lagged an extra month as the data for the month gets published around two weeks after the end of each month. S&P 500 data was downloaded from Yahoo! Finance[10]. Monthly percent returns were taken and then adjusted by an estimate for the cash rate (the 3-month treasury bill rate) to give an after-cash return. If the after-cash return was positive, it was assigned a '1', otherwise, a '0'. This data was then read into Python using pandas 'read_csv' function. Then, the data was split into a training set, a cross-validation set, and a testing set. Because there are only roughly 380 samples in the entire set, 75% were allocated to the training set, and 12.5% to each the

[9] U.S. Bureau of Labor Statistics, Consumer Price Index for All Urban Consumers: All Items [CPIAUCSL], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/CPIAUCSL, February 16, 2018.
Board of Governors of the Federal Reserve System (US), Effective Federal Funds Rate [FEDFUNDS], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/FEDFUNDS, February 16, 2018.
Federal Reserve Bank of St. Louis, TED Spread [TEDRATE], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/TEDRATE, February 15, 2018.
Board of Governors of the Federal Reserve System (US), Trade Weighted U.S. Dollar Index: Major Currencies [TWEXMMTH], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/TWEXMMTH, February 16, 2018.
Board of Governors of the Federal Reserve System (US), 3-Month Treasury Bill: Secondary Market Rate [TB3MS], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/TB3MS, February 16, 2018.

[10] "Historical Prices | S&P 500." Yahoo! Finance, Yahoo!, 16 Feb. 2018, finance.yahoo.com/quote/%5EGSPC/history?period1=504939600&period2=1518757200&interval=1mo&filter=history&frequency=1mo.

cross-validation and testing sets. To be precise, data from 2/1/1986 until 12/1/2009 was used as training, data from 1/1/2010 until 12/1/2013 was used as cross-validation and data from 1/1/2014 until 12/1/2017 was used as testing. Once the data had been read into pandas, a 'MinMaxScaler' object was fit to the training data and used to transform the training, cross-validation, and the testing set.

Before scaling:

date	chg_ted	chg_usd	chg_ffr	chg_cpi	target
2/1/1986	-0.13	-1.2149	-0.28	0.4	1
3/1/1986	0.05	-4.4735	-0.38	-0.2	1
4/1/1986	-0.12	-2.4337	-0.49	-0.6	0
5/1/1986	-0.04	-1.0595	-0.14	-0.4	1
6/1/1986	0.02	-2.6492	0.07	0.3	1

After scaling:

date	chg_ted	chg_usd	chg_ffr	chg_cpi	target
2/1/1986	0.430769	0.348515	0.371585	0.648426	1
3/1/1986	0.500000	0.009822	0.316940	0.556710	1
4/1/1986	0.434615	0.221835	0.256831	0.495567	0
5/1/1986	0.465385	0.364667	0.448087	0.526139	1
6/1/1986	0.488462	0.199437	0.562842	0.633140	1

Next, correlations and distributions were analyzed using pandas' methods 'corr' and 'scatter_matrix'. There was no correlation higher than 0.25 in absolute value and each distribution displayed normal-like characteristics (as shown earlier) thus no further preprocessing was needed.

Correlation Matrix and Visualization:

	chg_ted	chg_usd	chg_ffr	chg_cpi
chg_ted	1.000000	-0.051625	0.100365	0.143114
chg_usd	-0.051625	1.000000	0.043917	-0.164027
chg_ffr	0.100365	0.043917	1.000000	0.095044
chg_cpi	0.143114	-0.164027	0.095044	1.000000

b. Implementation

The first step in the implementation of the machine learning algorithms was to create a training and predicting pipeline (the same thing done in the 'finding donors' project) that took as input the classifier to be used, as well as the data and a scorer. The three models were then fed into the pipeline and ultimately outputted both an accuracy and an F-score on the training and cross-validation sets. To be specific, each algorithm took as input the after-scaling pandas data frame using the default parameters of each algorithm.

Results of the default classifiers:

	DecisionTreeClassifier	KNeighborsClassifier	SVC
acc_cv	0.479167	0.395833	0.645833
acc_train	1.000000	0.721254	0.599303
f_cv	0.582524	0.534591	0.695067
f_train	1.000000	0.751029	0.651515

c. Refinement

For each of the three models above, there was parameter tuning that was executed through a 'GridSearchCV' object. For the decision tree classifier, the 'min_samples_split' parameter was tuned as it clearly overfit the training set. The 'n_neighbors' parameter for the KNN classifier and both the 'C' and 'gamma' parameters were also tuned for the support vector machine classifier.

First, for KNN, the 'n_neighbors' parameter was allowed to range from 1 to 30 because any number higher than 30 would likely eventually always output '1' due to the 60/40 split of '1's and '0's in the target class. Grid search determined that a 'K' value of 26 was optimal and below is the improvement it made on the cross-validation set. The decision tree classifier was the next to be tuned. Here, the 'min_samples_split' parameter was allowed to range from 2 to 50 for similar reasons as above. The best classifier among these values turned out to be the decision tree with 13 as it's value for 'min_samples_split' and yielded a modest improvement noted below. Lastly, the support vector classifier was tuned using grid search across both its 'C' and 'gamma' parameters with 'C' ranging over the python list [.01,.25,.05,.1,.5,1.,2.,4.,8.] and 'gamma' ranging over the same values. However, no improvement was made over the default values. Below is a table of the before and after tuning results.

	KNN	Decision Tree	SVC
Unoptimized			
Accuracy on CV set	0.3958	0.4792	0.6458
F-score on CV set	0.5346	0.5825	0.6951
Optimized			
Accuracy on CV set	0.6042	0.5208	0.6458
F-score on CV set	0.6744	0.6303	0.6951
Change			
Accuracy on CV set	0.2084	0.0416	0
F-score on CV set	0.1398	0.0478	0

.

IV. Results

a. Model Evaluation and Validation

With these tuned models from above, they and our benchmark (Logistic Regression) were put to work on the test set which ranges from 2014-2017. Unfortunately, none of the tuned models were able to outperform the benchmark.

```
Classifier : LogisticRegression
Accuracy on the test set : 0.645833333333
F-score on the test set : 0.695067264574

Classifier : KNeighborsClassifier
Accuracy on the test set : 0.604166666667
F-score on the test set : 0.678391959799

Classifier : DecisionTreeClassifier
Accuracy on the test set : 0.4375
F-score on the test set : 0.546218487395

Classifier : SVC
Accuracy on the test set : 0.645833333333
F-score on the test set : 0.695067264574
```

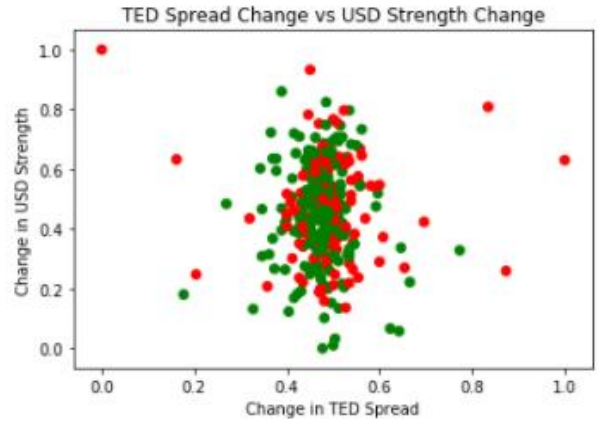
b. Justification

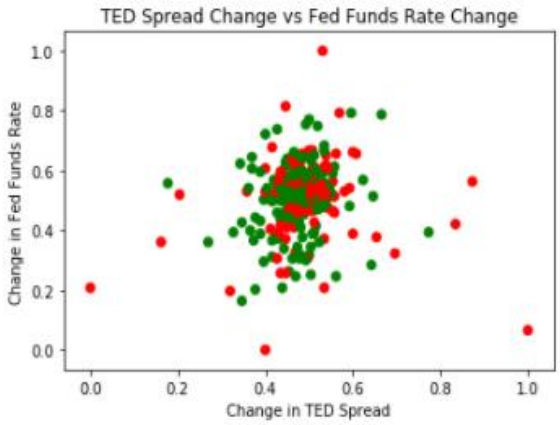
In fact, the highest F-score came from predicting a '1' on all 48 samples in the test set, the same performance a naïve predictor would accomplish. While this is certainly a disappointing result, predicting the stock market is undoubtedly one of the most challenging ventures one can undertake. And, if it were easy as taking these four major macroeconomic factors to be able to predict the movements in the stock market, everyone would do it until it became priced in and would no longer work.

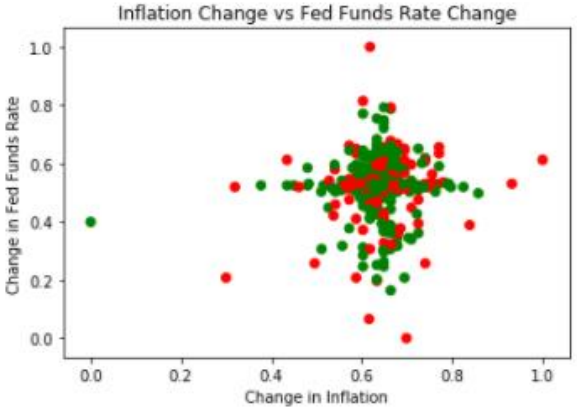
V. Conclusion

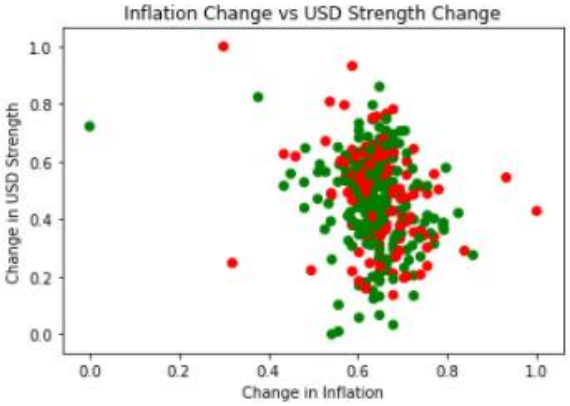
a. Free-Form Visualization

In order to see how hard of a problem this is, the following four plots show just how closely-packed the data is to one another illustrating the difficulty in separating the two classes. These illustrate to us the dynamics of the stock market and just how "random" it may be. Green data points are months in which the market returned a positive percentage after-cash and red data points are negative yielding months.









b. Reflection

Predicting stock returns is no easy task. There is an entire wealth of information out there and people have spent their lives trying to model and predict the market correctly. This project was most interesting as it was direct application of macroeconomic theory onto the short-run, monthly, returns of the US stock market. However, as can be shown mathematically, the excess return of assets is how economic conditions unfold relative to what is priced in and simply taking the conditions from the beginning of the month and using them to predict the return of the entire month does a disservice to the complexity of the market. That being said, every model starts somewhere and this project shows that improvement can be made and perhaps a more robust version could yield some good results.

c. Improvement

Some thoughts for improving results could be using PCA to reduce dimensionality or, conversely, adding features such as market lags or including other shocks to the economy (i.e., oil prices) would benefit the model. Other improvements could also include more parameter tuning that searches across a larger range. All in all, while I was unable to outperform the benchmark, this project was a great experience in practicing machine learning and am hopeful to continue down this track.