Udacity Machine Learning Nanodegree Capstone Project Submission

Ben Wong
Calgary, Alberta
August 2018

Definition

Project Overview

I work in an investment firm that manages investment portfolios for institutions and high net-worth individuals. The company has a research department that chooses equity and fixed-income investments for the portfolios of our clients.

I would like to identify economic indicators that drive stock market returns and develop a machine learning model based on the indicators to predict market returns to help my company make money. I will be using the S&P 500 as the targeted stock market to research as it is the largest stock market in the world and has the most widely available information.

Problem Statement

The purpose of this project is to develop a machine learning algorithm that can predict stock market returns, specifically the S&P 500 index as it is the largest stock market with the most available information of all stock markets.

Stock market analysis is generally divided into two categories – 1) fundamental analysis and 2) technical analysis.

Fundamental analysis is predicated on the idea that the price of a stock should reflect the value of the individual company's financial fundamentals, including future growth, balance sheet financial position, current and future profitability and outlook, and its ability to generate a stable stream of cash flow.

Technical analysis is based on the idea the price of a stock follows historical price pattern and the pattern results from economic cycles of expansion and contractions, general business cycle, and the nature of the market forces.

The industry jargon for fundamental analysis is micro or bottom-up strategy, and for technical analysis is macro-strategy. Both strategies have successful adherents. The most successful practitioner of fundamental analysis is Warren Buffet, the 3rd world's richest man on Forbes wealth list, and the most famous adherent of technical analysis is Ray Dalio who ranks 67th on the same list.

This problem using machine learning algorithms to predict stock market return has previously been researched and the following are academic papers relating to this problem:

1. http://cs229.stanford.edu/proj2017/final-reports/5234854.pdf

Smarth Behl (smarth), Kiran Tondehal (kirantl), Naveed Zaman (naveedz) used economic indicators to identify buy and sell triggers

2. https://arxiv.org/pdf/1603.00751.pdf

Milosevic used financial indicators to predict stock prices.

3. https://etd.auburn.edu/xmlui/bitstream/handle/10415/5652/Application%20of%20machine%20learning%20techniques%20for%20stock%20market%20prediction.pdf?sequence=2&isAllowed=y

Weng used various machine learning techniques to predict stock prices.

My analysis is a form of technical analysis whereby economic indicators are used to predict future price returns.

Tools and Technologies

This project is conducted in Python2, Jupyter Notebook, the Sci-Kit Learn, Numpy and Panda libraries.

Metrics

Evaluation metrics that will be used to evaluate the success and performance of my model are:

1. **Mean absolute error** (MAE) - This is the absolute differences between the predicted values and the actual values. The lower the value, the better the model. This model is optimal because it is robust to outliers and is applicable to this project because it helps to explain how error-prone is the model.
2. **Root Mean squared error** (RMSE) - Like Mean absolute error (MAE), this metrics takes the squared-error between each point in the dataset and the regression line. The lower the value, the better the model. This metrics is optimal because it corrects for MAE by cancelling the positive and negative error values and is applicable to this project because it helps to explain how error-prone is the model.

3. **Explained variance score** - This is a metrics that measures the variance in the regression. The higher the score, the more of the target variable can be explained by the predictive variables. This is an optimal metric in this project because it helps to explain how much of the variation of the S&P 500 return is explained by the economic indicators.
4. **Coefficient of determination (r2 score)** – This is a metric that measures the total variation in the target variable that is captured by the model. This ranges from 0 to 1 where 0 means none of the variance is captured and 1 where all the variance is capture, therefore a high number is what we want in a good model. This is an optimal metric in this project because it helps to explain how much of the variation of the S&P 500 return is explained by the economic indicators.

Analysis

Data Exploration

The dataset, https://www.kaggle.com/robertnolan/economic-indicators, that will used for this project is from the machine learning competition website, Kaggle, and it is a collection of economic indiators from January 2007 to September 2017 by month. The dataset has 15 features and 129 data points. The

The attribution information of the dataset is as follows:

1) PeriodMonthly data in format: YYYY-MM

2) ConsConfThe consumer confidence index (CCI) is based on households' plans for major purchases and their economic situation, both currently and their expectations for the immediate future.

3) CompLeadThe composite leading indicator (CLI) is designed to provide early signals of turning points in business cycles showing fluctuation of the economic activity around its long term potential level. CLIs show short-term economic movements in qualitative rather than quantitative terms.

4) BusConfThe business confidence index (BCI) is based on enterprises' assessment of production, orders and stocks, as well as its current position and expectations for the immediate future. Opinions compared to a “normal” state are collected and the difference between positive and negative answers provides a qualitative index on economic conditions.

5) EmpEmployment rates are defined as a measure of the extent to which available labour resources (people available to work) are being used.

6) InvToSalesInventory turnover is a ratio showing how many times a company's inventory is sold and replaced over a period of time. The days in the period can then

be divided by the inventory turnover formula to calculate the days it takes to sell the inventory on hand. It is calculated as sales divided by average inventory.

7) PMIThe Purchasing Managers' Index (PMI) is an indicator of the economic health of the manufacturing sector. The PMI is based on five major indicators: new orders, inventory levels, production, supplier deliveries and the employment environment.

8) SP500 The Standard & Poor's 500, often abbreviated as the S&P 500, or just the S&P, is an American stock market index based on the market capitalizations of 500 large companies having common stock listed on the NYSE or NASDAQ.

9) MfgOrdDur"The Advance Report on Durable Goods Manufacturer's Shipments, Inventories and Orders," or the Durable Goods Report, provides data on new orders received from more than 4,000 manufacturers of durable goods, which are generally defined as higher-priced capital goods orders with a useful life of three years or more, such as cars, semiconductor equipment and turbines.

10) BldgPermA type of authorization that must be granted by a government or other regulatory body before the construction of a new or existing building can legally occur.

11) SalesAdvance Retail Sales: Retail (Excluding Food Services).

12) FedFundsFederal funds interest rate at which depository institutions (banks and credit unions) lend reserve balances to other depository institutions overnight, on an uncollateralized basis.

13) DJI The Dow Jones Industrial Average (DJIA) is a price-weighted average of 30 significant stocks traded on the New York Stock Exchange (NYSE) and the NASDAQ.

14) PayrollAll Employees: Total Nonfarm, commonly known as Total Nonfarm Payroll, is a measure of the number of U.S. workers in the economy that excludes proprietors, private household employees, unpaid volunteers, farm employees, and the unincorporated self-employed.

15) PersConsPersonal consumption expenditures (PCE) is the primary measure of consumer spending on goods and services in the U.S. economy. It accounts for about two-thirds of domestic final spending, and thus it is the primary engine that drives future economic growth.

Algorithms and Techniques

The technique we will approach this problem is a form of supervised machine learning called regression analysis. The approach is fitting a line through a data set of observations and using this line to predict unobserved values. The idea is that using a line that minimizes the sum of squared errors can maximize the probability of unobserved and future data.

The types of regression algorithm we will use to in our analyses are:

1. Linear regression – This method takes the best-fit line (sum of least squared) of the training data set
2. Polynomial regression – This method is similar to linear regression, but instead of fitting a straight line, it fits a curved line through the data points
3. Ridge regression – This method is used when the data set has multicollinearity (where the predictive variables are highly correlated). This model reduces the standard errors by adding a degree of bias (shrinkage parameter) to the model.
4. Lasso regression – This method is similar to Ridge regression, but instead of using squared values in the penalty function, it uses the absolute values.
5. ElasticNet regression – This method is related to both Ridge and Lasso and generates zero-valued coefficients.
6. Stochastic Gradient Descent regression – This method applies stochastic gradient descent to fit linear models by minimizing a cost function.
7. Decision tree regression – This method breaks down a dataset into branches and is split by the differences and eventually to the root node, the purest child node.
8. Multi-layer Perceptron (MLP) regression – This method is a form of deep learning that tries to mimic the biological brain where it is divided into three parts – 1) the input layer (the biological sensory layer that receives the information), 2) the hidden layer (the biological layer computes (trains) the data from the input layer, and 3) the output layer that makes the final decision.

Benchmark

The benchmark model used will a simple linear regression to get the baseline score because it is easy to understand because it is simply finding the best fit line across all the data points that minimize the total distances between the line and all the data points.

Methodology

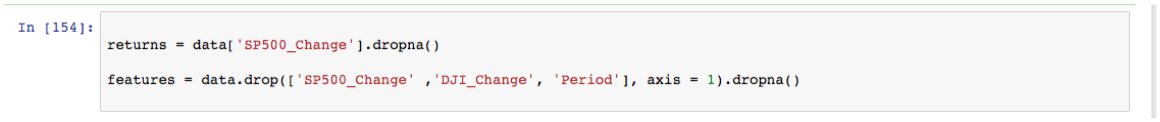
Data Preprocessing, exploratory visualization, refinement

The data was initially preprocessed to obtain the month-to-month change to regress the change in the predictor variables with the change in market return.

In [160]:

```python
# Import libraries necessary for this project
import numpy as np
import pandas as pd
from sklearn.cross_validation import ShuffleSplit

# Import supplementary visualizations code visuals.py
import visuals as vs

# Pretty display for notebooks
%matplotlib inline

# Load the economic indicator dataset
data = pd.read_csv('econ_change.csv', error_bad_lines=False)

print "Economic indicator dataset has {} data points with {} variables each.".format(*data.dropna().shape)
```

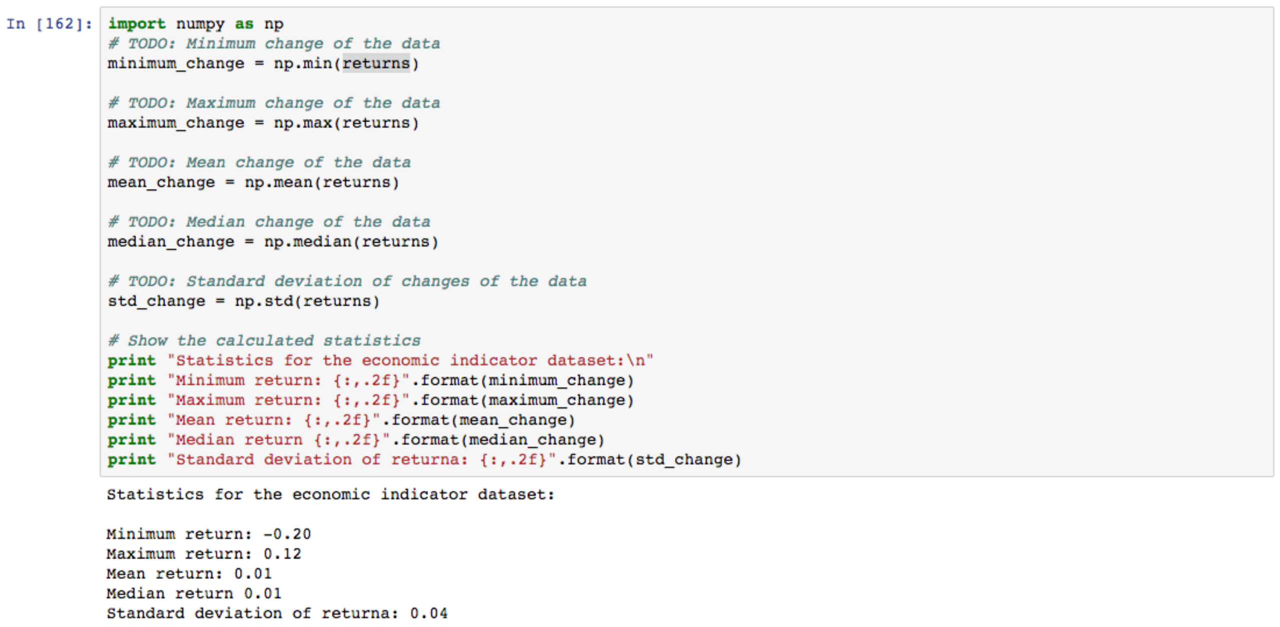
```
Economic indicator dataset has 128 data points with 15 variables each.
```



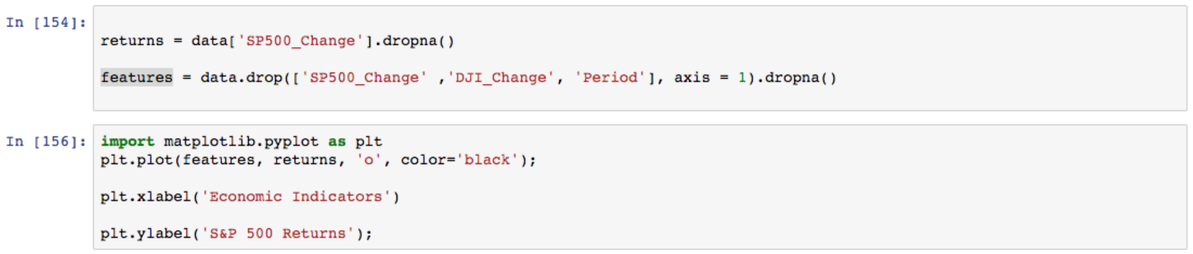
```
In [154]:
returns = data['SP500_Change'].dropna()

features = data.drop(['SP500_Change' ,'DJI_Change', 'Period'], axis = 1).dropna()
```

Dow Jones Industrial Average (DJI) is an index composed of the 30 largest stocks on the New York Stock Exchange (NYSE). It will be excluded from our dataset because we would like to assess the economic indicators on the overall market, which the S&P 500 is a more apt measure because it is composed of more companies (500), whereas the DJI (30) is biased towards big companies.

We are also dropping the first row of the dataset because the change in value is calculated by dividing the next period's value by the last period. Since the first row has no previous value, it is NAN and will be excluded. With the 1 row of the dataset excluded, we will have 128 data points (129 - 1).
Implementation.

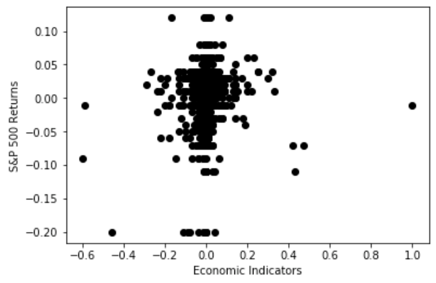


```
In [162]: import numpy as np
          # TODO: Minimum change of the data
          minimum_change = np.min(returns)

          # TODO: Maximum change of the data
          maximum_change = np.max(returns)

          # TODO: Mean change of the data
          mean_change = np.mean(returns)

          # TODO: Median change of the data
          median_change = np.median(returns)

          # TODO: Standard deviation of changes of the data
          std_change = np.std(returns)

          # Show the calculated statistics
          print "Statistics for the economic indicator dataset:\n"
          print "Minimum return: {:,.2f}".format(minimum_change)
          print "Maximum return: {:,.2f}".format(maximum_change)
          print "Mean return: {:,.2f}".format(mean_change)
          print "Median return {:,.2f}".format(median_change)
          print "Standard deviation of returna: {:,.2f}".format(std_change)

          Statistics for the economic indicator dataset:

          Minimum return: -0.20
          Maximum return: 0.12
          Mean return: 0.01
          Median return 0.01
          Standard deviation of returna: 0.04
```



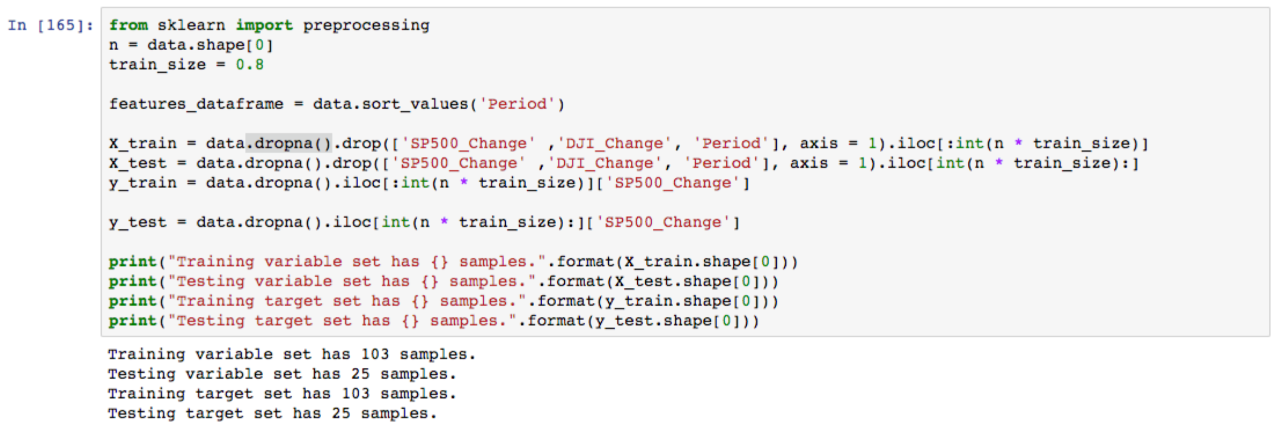
```
In [154]:
returns = data['SP500_Change'].dropna()

features = data.drop(['SP500_Change' ,'DJI_Change', 'Period'], axis = 1).dropna()
```

```
In [156]: import matplotlib.pyplot as plt
plt.plot(features, returns, 'o', color='black');

plt.xlabel('Economic Indicators')

plt.ylabel('S&P 500 Returns');
```



Next, we examined the statistics for the dataset. We see that the mean and median movement of the S&P 500 is 1%. This is a very low number as the market generally gyrate much from month-to-month. We can see the in the scatterplot that because of the small movement in such small time increment, all the data points are concentrated around 0 and would result in a poor predictive model. However, we will proceed to assess how the model performs.

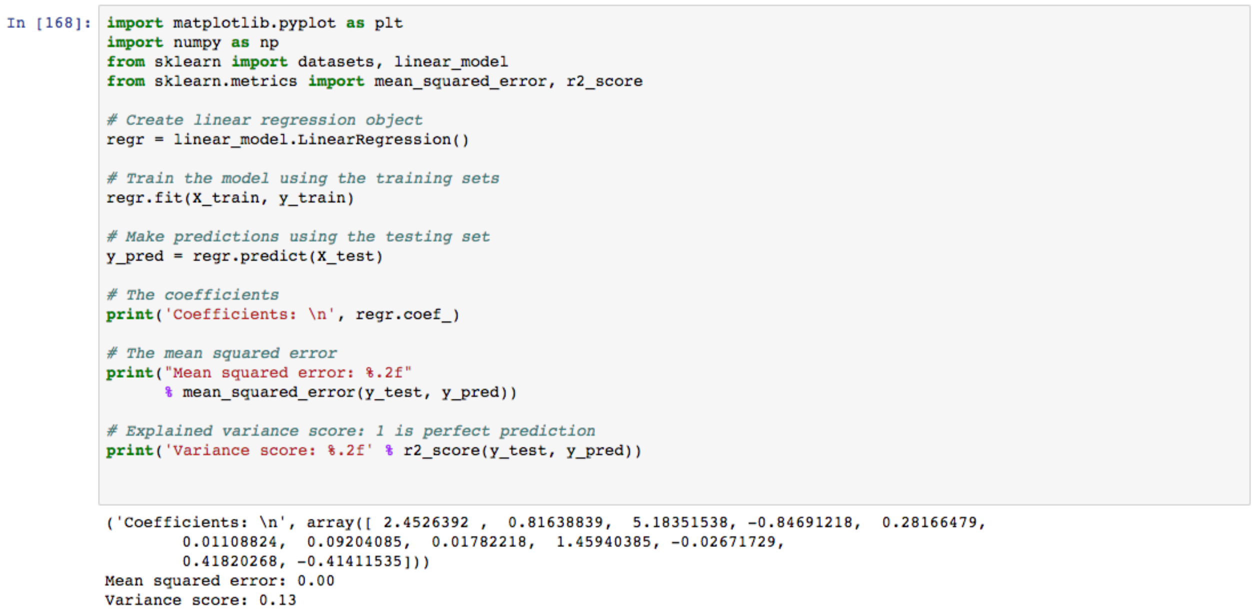
Next, I sort the data by time period, and then split it into 80% for training and 20% for testing.



```
In [165]: from sklearn import preprocessing
n = data.shape[0]
train_size = 0.8

features_dataframe = data.sort_values('Period')

X_train = data.dropna().drop(['SP500_Change' ,'DJI_Change', 'Period'], axis = 1).iloc[:int(n * train_size)]
X_test = data.dropna().drop(['SP500_Change' ,'DJI_Change', 'Period'], axis = 1).iloc[int(n * train_size):]
y_train = data.dropna().iloc[:int(n * train_size)]['SP500_Change']

y_test = data.dropna().iloc[int(n * train_size):]['SP500_Change']

print("Training variable set has {} samples.".format(X_train.shape[0]))
print("Testing variable set has {} samples.".format(X_test.shape[0]))
print("Training target set has {} samples.".format(y_train.shape[0]))
print("Testing target set has {} samples.".format(y_test.shape[0]))

Training variable set has 103 samples.
Testing variable set has 25 samples.
Training target set has 103 samples.
Testing target set has 25 samples.
```

Then we fit the training data to the Linear Regression model and run a basic prediction on it. The results were quite abysmal. The R2 score is 0.13.

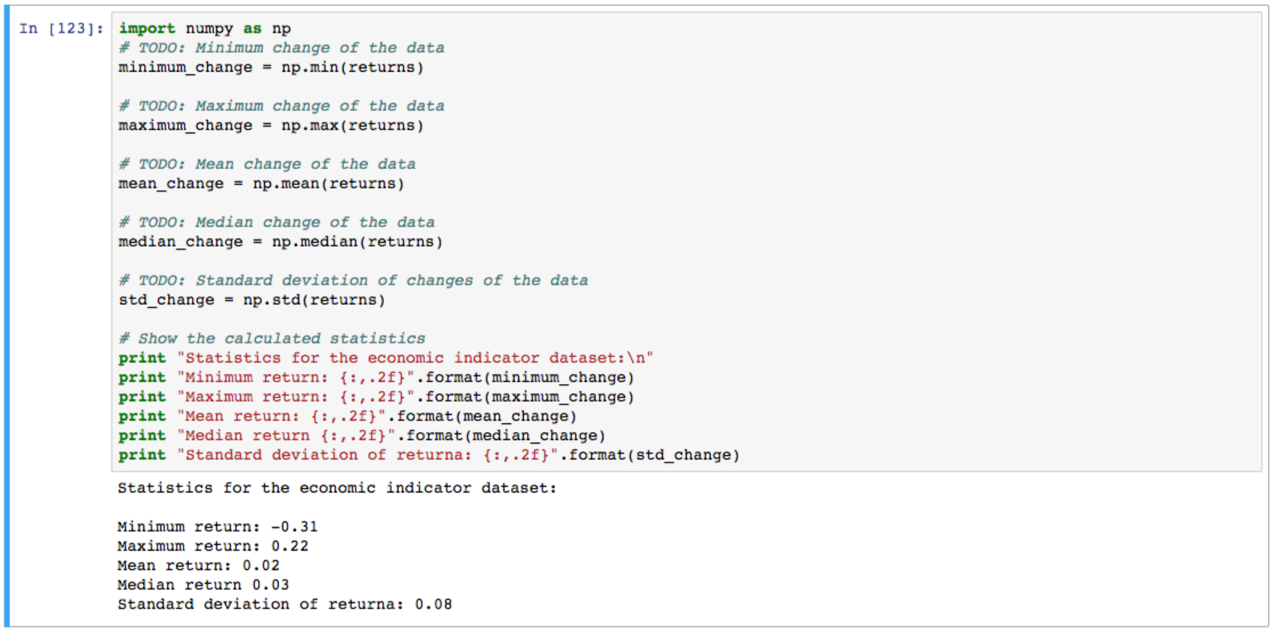
Step 3: Choosing a Model



```
In [168]: import matplotlib.pyplot as plt
          import numpy as np
          from sklearn import datasets, linear_model
          from sklearn.metrics import mean_squared_error, r2_score

          # Create linear regression object
          regr = linear_model.LinearRegression()

          # Train the model using the training sets
          regr.fit(X_train, y_train)

          # Make predictions using the testing set
          y_pred = regr.predict(X_test)

          # The coefficients
          print('Coefficients: \n', regr.coef_)

          # The mean squared error
          print("Mean squared error: %.2f"
                % mean_squared_error(y_test, y_pred))

          # Explained variance score: 1 is perfect prediction
          print('Variance score: %.2f' % r2_score(y_test, y_pred))
```

```
('Coefficients: \n', array([ 2.4526392 ,  0.81638839,  5.18351538, -0.84691218,  0.28166479,
        0.01108824,  0.09204085,  0.01782218,  1.45940385, -0.02671729,
        0.41820268, -0.41411535]))
Mean squared error: 0.00
Variance score: 0.13
```

To address the inherent issue with the small movement in the market on a monthly basis, I preprocessed the data so to obtain the change on a quarterly basis to expand the movement of the dataset.



```
In [123]: import numpy as np
          # TODO: Minimum change of the data
          minimum_change = np.min(returns)

          # TODO: Maximum change of the data
          maximum_change = np.max(returns)

          # TODO: Mean change of the data
          mean_change = np.mean(returns)

          # TODO: Median change of the data
          median_change = np.median(returns)

          # TODO: Standard deviation of changes of the data
          std_change = np.std(returns)

          # Show the calculated statistics
          print "Statistics for the economic indicator dataset:\n"
          print "Minimum return: {:,.2f}".format(minimum_change)
          print "Maximum return: {:,.2f}".format(maximum_change)
          print "Mean return: {:,.2f}".format(mean_change)
          print "Median return {:,.2f}".format(median_change)
          print "Standard deviation of returna: {:,.2f}".format(std_change)
```

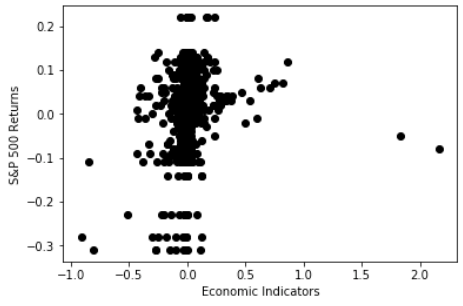
```
Statistics for the economic indicator dataset:

Minimum return: -0.31
Maximum return: 0.22
Mean return: 0.02
Median return 0.03
Standard deviation of returna: 0.08
```

Looking at the new statistics, the mean and median, and standard deviation more than doubled.

```
In [124]: from sklearn import preprocessing
          n = data.shape[0]
          train_size = 0.8

          features_dataframe = data.sort_values('Period')

          X_train = data.dropna().drop(['SP500_Change' ,'DJI_Change', 'Period'], axis = 1).iloc[:int(n * train_size)]
          X_test = data.dropna().drop(['SP500_Change' ,'DJI_Change', 'Period'], axis = 1).iloc[int(n * train_size):]
          y_train = data.dropna().iloc[:int(n * train_size)]['SP500_Change']

          y_test = data.dropna().iloc[int(n * train_size):]['SP500_Change']

          print("Training variable set has {} samples.".format(X_train.shape[0]))
          print("Testing variable set has {} samples.".format(X_test.shape[0]))
          print("Training target set has {} samples.".format(y_train.shape[0]))
          print("Testing target set has {} samples.".format(y_test.shape[0]))

          Training variable set has 100 samples.
          Testing variable set has 26 samples.
          Training target set has 100 samples.
          Testing target set has 26 samples.
```

Visualizing the data, we can see that the data became less concentrated and there is more movement.

```
In [122]: import matplotlib.pyplot as plt

          plt.xlabel('Economic Indicators')

          plt.ylabel('S&P 500 Returns');

          labels = ['text{}'.format(i) for i in range(len(features))]

          plt.plot(features, returns, 'o', color='black');
```



We now have 100 training samples compared to 103, and 26 testing samples compared to 25.

```
In [134]: import matplotlib.pyplot as plt
          import numpy as np
          from sklearn import datasets, linear_model
          from sklearn.metrics import mean_squared_error, r2_score

          # Create linear regression object
          regr = linear_model.LinearRegression()

          # Train the model using the training sets
          regr.fit(X_train, y_train)

          # Make predictions using the testing set
          y_pred = regr.predict(X_test)

          # The coefficients
          print('Coefficients: \n', regr.coef_)

          # The mean squared error
          print("Mean squared error: %.2f"
                % mean_squared_error(y_test, y_pred))

          # Explained variance score: 1 is perfect prediction
          print('Variance score: %.2f' % r2_score(y_test, y_pred))
```

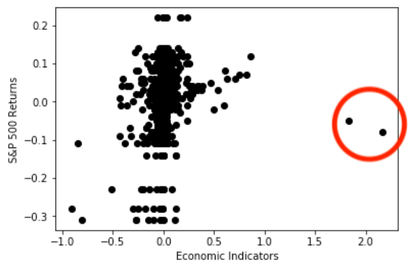
```
('Coefficients: \n', array([ 3.1346045 ,  3.58543975,  5.76091645, -1.47530833,  0.77128406,
       -0.23972218,  0.04165347,  0.03545784,  1.47202505, -0.03014659,
        0.63604057,  1.51943897]))
Mean squared error: 0.00
Variance score: 0.30
```

With the adjustment from monthly change to quarterly change, the r2 score improved significantly from 13% to 30%.

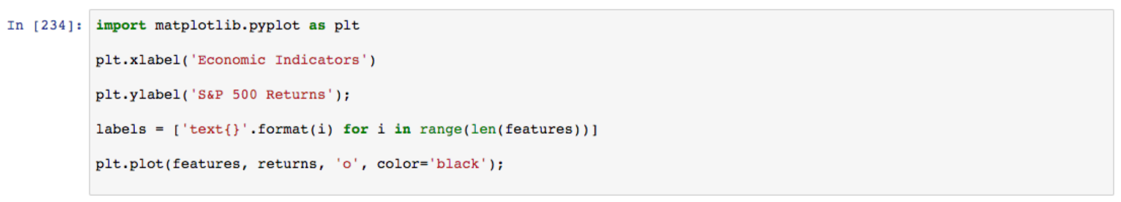
With the improved score, we now would like to identify outliers that may skew the model.

From the visualization, these data points are outliers and need to be removed.

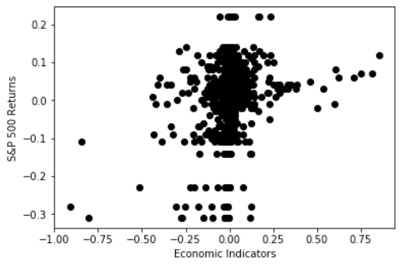
```
In [219]: import matplotlib.pyplot as plt

          plt.xlabel('Economic Indicators')

          plt.ylabel('S&P 500 Returns');

          labels = ['text{}'.format(i) for i in range(len(features))]

          plt.plot(features, returns, 'o', color='black');
```



The following visualization is the result of removing these data points.



```
In [234]: import matplotlib.pyplot as plt

          plt.xlabel('Economic Indicators')

          plt.ylabel('S&P 500 Returns');

          labels = ['text{}'.format(i) for i in range(len(features))]

          plt.plot(features, returns, 'o', color='black');
```



We recomputed the r2 on the linear regression model and the score plummeted to -2%.

I did not encounter any coding complication in this project as I was trying to mimic the Boston Housing Project as closely as possible in this program because it is based on supervised learning regression analysis. Therefore, I had a good template to work off of.

The only slight challenge is the train/split step where I did not want to split the data randomly, but would like the data to be ordered chronologically. Therefore, I could not use the train_test_split function built into scikit-learn. Therefore, I had to sort and split the data manually in Python. But this was not too much difficulty.

Results

Model Evaluation and Validation

The following summarizes the results of all the models.

	Linear	Polynomial	Ridge	Lasso	ElasticNet
Mean Absolute Error	3.200%	3.200%	2.230%	0.000%	3.180%
Mean Squared Error	0.140%	0.140%	0.090%	0.150%	0.150%
Variance Score	9.360%	9.360%	39.280%	3.180%	0.000%
R^2 Score	2.110%	2.110%	35.910%	-7.390%	-7.390%

	Stochastic Gradient Descent	Decision Tree	Multi-Layer Perceptron
Mean Absolute Error	3.130%	3.800%	3.800%
Mean Squared Error	0.140%	0.230%	0.230%
Variance Score	6.030%	-31.510%	-31.510%
R^2 Score	-2.480%	-66.440%	-66.440%

Model Justification

Ridge regression scores the best in all four evaluation metrics, with the lowest MAE and RMSE, and the highest variance and r2 score. Therefore, Ridge regression is the best model for this project.

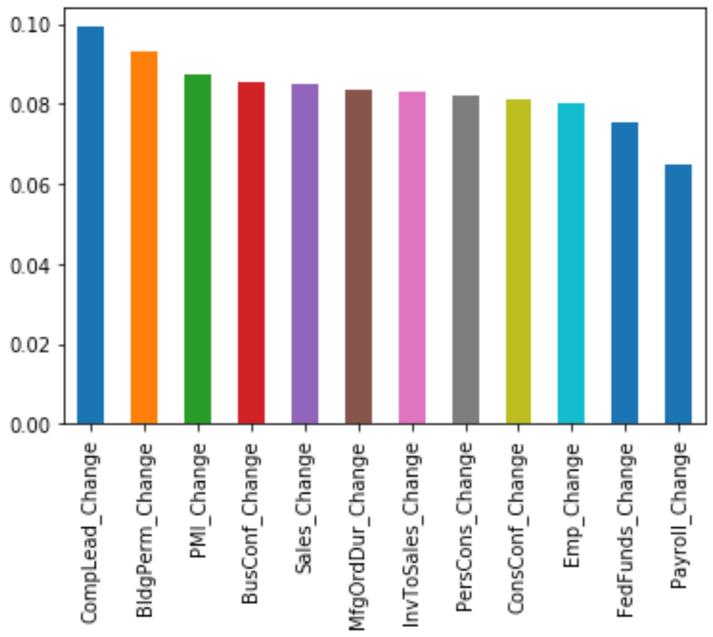
Despite excelling over all other models in our evaluation metrics, the model doesn't generalize well to the 20% of the testing data with a r^2 score of only 36%. A good regression model would be one with r^2 of over 80%.

Conclusion

Reflection and Improvement

In this project, we took 12 predictive economic variables (15 - 3 (DJI, Period, S&P500) and built a model to predict the S&P 500 return based on 10 years on data changed on a quarterly basis.

The following is a feature importance plot using the ExtraTreesClassifier from *sklearn* that shows the importance of the economic indicators in the order of the most importance to the least. The top 3 most important indicators are the composite leading indicator, the building permit indicator, and the Purchasing Managers' Index (PMI) indicator.



We applied the data set to 8 different regression models. The model that scored the best based on our metrics is the Ridge model. It has the lowest MAE, lowest RMSE, highest R2 score and the highest explained variance.

Despite of scoring the best in all the evaluation metrics, the R2 score of 36% is still quite low. It is preferred that we have over 80%, and ideally, over 90% in our R2 score.

The most important part of this project is to analyzing the different economic indicators and seeing which indicator can explain the S&P 500 the most. The other interesting aspect is seeing how the different regression machine learning models perform on the data set.

In this project, I learned that executing the machine learning algorithm is very easy once the data is analyzed, and preprocessed and ready to be applied to an algorithm. Running any of the algorithms in the project took less than 3 lines of codes.

Major improvements that may contribute to further analysis are adding more economic indicators and adding more years of data that include more economic and business cycles.

Sources and References

- *https://towardsdatascience.com/selecting-the-best-machine-learning-algorithm-for-your-regression-problem-20c330bad4ef*
- *https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/*
- *https://www.mathworks.com/help/stats/lasso-and-elastic-net.html*
- *https://stackoverflow.com/questions/44511636/matplotlib-plot-feature-importance-with-feature-names*
- *https://medium.com/usf-msds/choosing-the-right-metric-for-machine-learning-models-part-1-a99d7d7414e4*