Stock Price Trend Prediction Based on Machine Learning

Project Overview

In our economic society, the investors always want to predict future stock price trend, because he successful prediction of a stock's future price could yield significant profit. However, it is usually difficult to predict the trend of the stock market.
Too many factors can cause the fluctuation of the stock price, involving business fundamentals, world events and so on. For example, the stock prices of airline company will be influenced by oil prices, terrorist attacks and so on. What's more time-effectiveness of prediction, the complexity of selection of features and many other factors also lead to the difficulty in market prediction.
There are two broadcast categories of approaches for use to choosing stock to buy or sell. One is fundamental analysis, another is technical analysis. Fundamental analysis involving looking at aspects of a company in order to estimate its value. Technical analysis don't care the value of a company, they look for patterns or trends in a stock's price. Since technical analysis is efficient and convenient, I will focus on the technical analysis.
The dataset in this project comes from Yahoo Finance. Since there are too many stocks in the stock market, I will pick one of them to analysis.

Problem Statement

The prediction of stock price trend is a supervised learning question. In this project, the target is to predict future close price of a stock, which also reflects whether the price will go up or down.
The prediction will be made based on historical data. The S&P 500 trust is an exchange-traded fund which trades on the NYSE Arca under the symbol. It is designed to track the S&P 500 stock market index. For a long time, this fund was the largest ETF in the world. It is reasonable to consider S&P 500 to be the reflection of the stock market. In this project, I also pick Google's stock to make prediction.

The dataset will be divided into two groups. One is used for training and the other one is used for testing. I will explore the data and do some preprocessing job, and then I will apply different algorithms to build models on the dataset.

Metrics

1. R^2 (coefficient of determination)

In statistics, the coefficient of determination, denoted R2 and pronounced "R squared", is a number that indicates the proportion of the variance in the dependent variable that is predictable from the independent variable.
R^2 is a regression score function. It provides a measure of how well future samples are likely to be predicted by the model. Best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y, disregarding the input features, would get a R^2 score of 0.0.
The higher R^2 score, the more likely the model can predict future stock price.

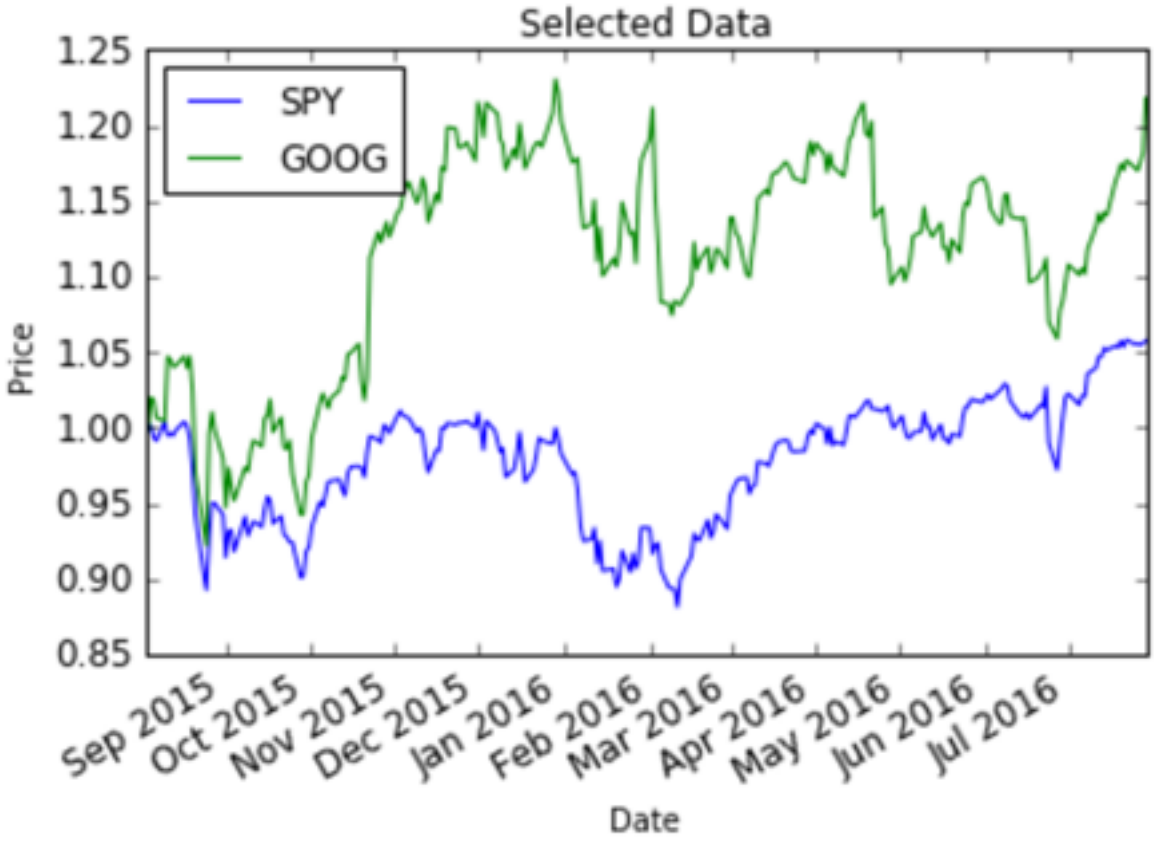
2. Time

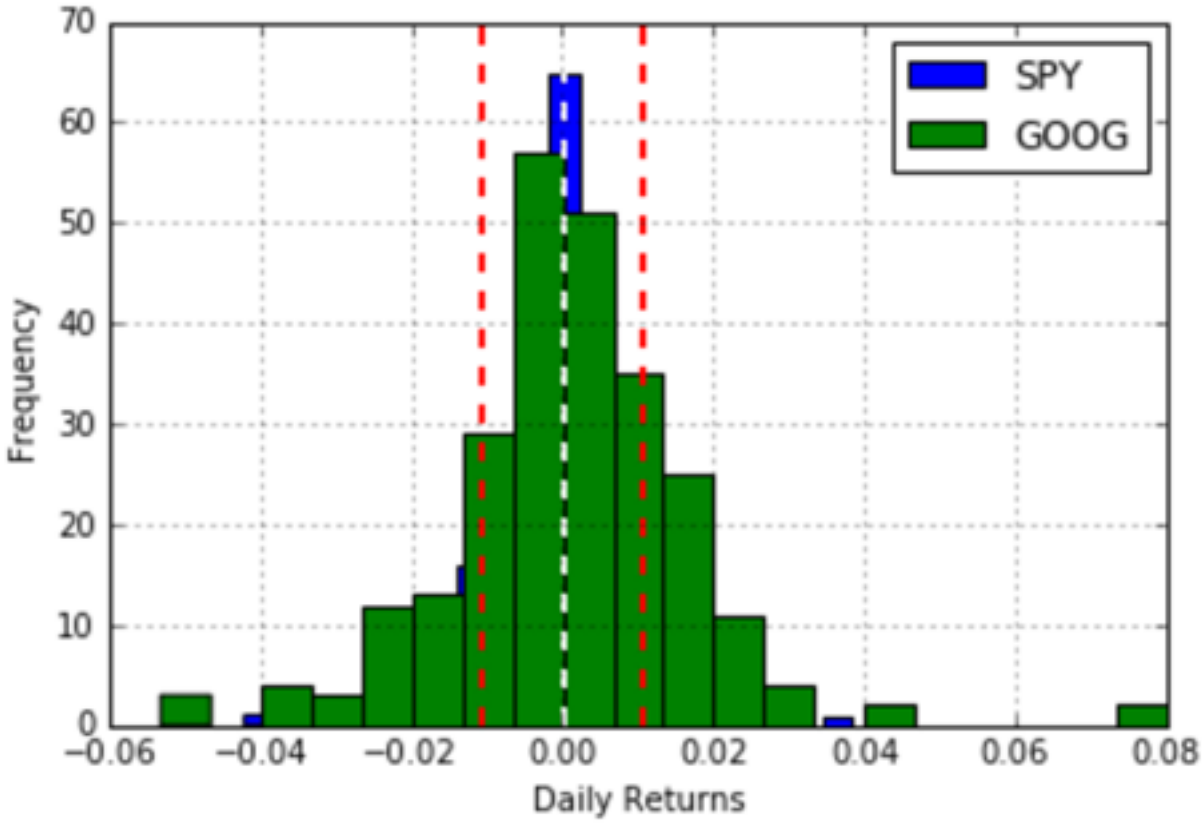
The time the model takes to do classifications is also an important criterion. A good model should work efficiently so that large dataset can be processed in a fast speed. I will use the testing time as one of the metrics. The less the time used, the better the performance of the model.

Analysis

Data Exploration

I have checked some statistical characteristics of the data through table and bar plots. Let's compare the close price of SPY and Google first.


Selected Data
SPY
GOOG
Price
1.25
1.20
1.15
1.10
1.05
1.00
0.95
0.90
0.85
Sep 2015
Oct 2015
Nov 2015
Dec 2015
Jan 2016
Feb 2016
Mar 2016
Apr 2016
May 2016
Jun 2016
Jul 2016
Date


SPY
GOOG
Frequency
70
60
50
40
30
20
10
0
−0.06
−0.04
−0.02
0.00
0.02
0.04
0.06
0.08
Daily Returns

From the plot and bar chart above, I find that the trend of Google's stock and the trend the market which is reflected by SPY are similar in general. As a result, the close price of SPY can be considered as an important factor to predict the close price of Google.

I sliced the data from 2015-08-03 to 2016-07-29, which contains 251 samples.The statistical description can be seen from the following table.

	SPY	GOOG
count	251.000000	251.000000
mean	0.978929	1.113789
std	0.040145	0.070917
min	0.881380	0.922134
25%	0.948778	1.081368
50%	0.990767	1.129640
75%	1.005409	1.169072
max	1.057964	1.230335

Since the stock price of SPY is very different from that of Google, we can not compare these two price directly. So the data was normalized first and then we can compare them in an impartial way. The mean of the normalized stock price of SPY and Google are both very approximate to 1.0 and the standard deviation is very low which means

these two stocks are relatively stable.

Algorithms and Techniques

First, I realize that the number of the data is not quite large, and we are doing regression. I want to find a linear relationship between the target stock price and historical stock price. So linear regression is the algorithm I choose.

Next, since the dataset is not large, I perform a model which can reduce the variance. Bagging does very well in improving the stability and accuracy of models used in statistical regression. It also reduces variance and helps to avoid overfitting. So bagging is a good choice.

Linear Regression

Linear regression is an approach for modeling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X.

The relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data.

Positive:

1. It's simple and convenient to use.
2. It works well in practice.

Negative:

1. It's easy to cause under fitting.

Bagging

Bagging, also called bootstrap aggregating, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression.

Given a standard training set D of size n, bagging generates m new training sets Di, each of size n′, by sampling from D uniformly and with replacement. By sampling with replacement, some observations may be repeated in each Di. If n′=n, then for large n the set Di is expected to have the fraction (1 - 1/e) (≈63.2%) of the unique examples of D, the rest being duplicates. This kind of

sample is known as a bootstrap sample. The m models are fitted using the above m bootstrap samples and combined by averaging the output (for regression) or voting (for classification).

Positive:

1. It can improve the accuracy and stability of the algorithm.
2. It also reduces variance and avoid overfitting.

Negative:

1. There is no weight for the predict function.

Benchmark

The situation of the stock market is very complicated and the information we have is limited. However, technical analysis only cares about the price and volume. Here I only analysis the close price of Google's stock, the model is simpler than the one used in real stock market. I use the function in scikit-learn module of python to do experiments, and the best performance algorithm's R^2 score is 0.83. I also read some other thesis about stock prediction, none of the R^2 score of the experiment reaches 0.85.

So I will set 0.83 as benchmark, and I hope to raise the R^2 score to 0.85.

Data Preprocessing

As long as the stock market allows trading, SPY won't be null. There is no missing value in SPY and I will use SPY as reference. If there are some missing values in Google, I will use the forward price to fill the missing values.

Bollinger Bands is a very import feature in stock market. It reflects the fluctuation and confident interval of stock price. Bollinger Bands includes mean and standard deviation. Bollinger Bands mean is the mean of the stock price in some periods. Bollinger Bands standard deviation of the stock price in some periods. Bollinger Bands standard deviation can be calculated from Bollinger Bands standard

deviation mean, here I only take Bollinger Bands mean into consideration.

I use the historical SPY and Google close price, historical volume of Google and Bollinger Bands mean as the features to predict future price trend. So we have 251 samples with 4 features for each.

All categorical features are numeric values. Next, I will split the dataset into training and test sets. Use 75% of the dataset as training data and remaining parts as testing data. The split results are stored in X_train, X_test, y_train, and y_test.

I will explain the procedure of the prediction by the following example:

I have the SPY close price, Google close price , Google volume and Bollinger Bands mean of SPY on 2016-06-01. Then I will put these values into the model I have trained and get the Google close price (output of the model) on 2016-06-08. The time difference between current price and future price is seven days.

Implementation

I take the Bollinger Bands mean as one of the features, however, I can not calculate the Bollinger Bands mean for all the data points. I need several data points before the current data point that I am calculating Bollinger Bands mean for. Otherwise it will cause NaN values in Bollinger Bands mean for the first 30 data points in the dataset. So I have to extract a slice of the dataset (etc: 2015-09-01 ~ 2016-07-29) and use the data before that slice (etc: 2015-08-01 ~ 2016-06-30) to calculate the Bollinger Bands mean (for data in 2015-09-01 ~ 2016-07-29).

I used linear regression and bagging algorithms to train the data, the metrics were R^2 score and testing time. First I imported LinearRegression from sklearn.linear_model, trained them on the extracted feature matrix and predicted the stock price on testing feature matrix. Next,I imported BaggingRegressor from sklearn.ensemble, trained them on the extracted features and predicted the stock price on testing feature matrix.

I have some difficulties when perform the experiment. At first, no matter what I do, the R^2 score is always very low and cannot even reach 0.4. Since I know that technical analysis is only related to the stock price and volume, I think that perhaps the samples used for training is too few, which caused the bad performance of the model. Then I tried to adjust the number of training data and added more samples, the result improved a lot!

Results

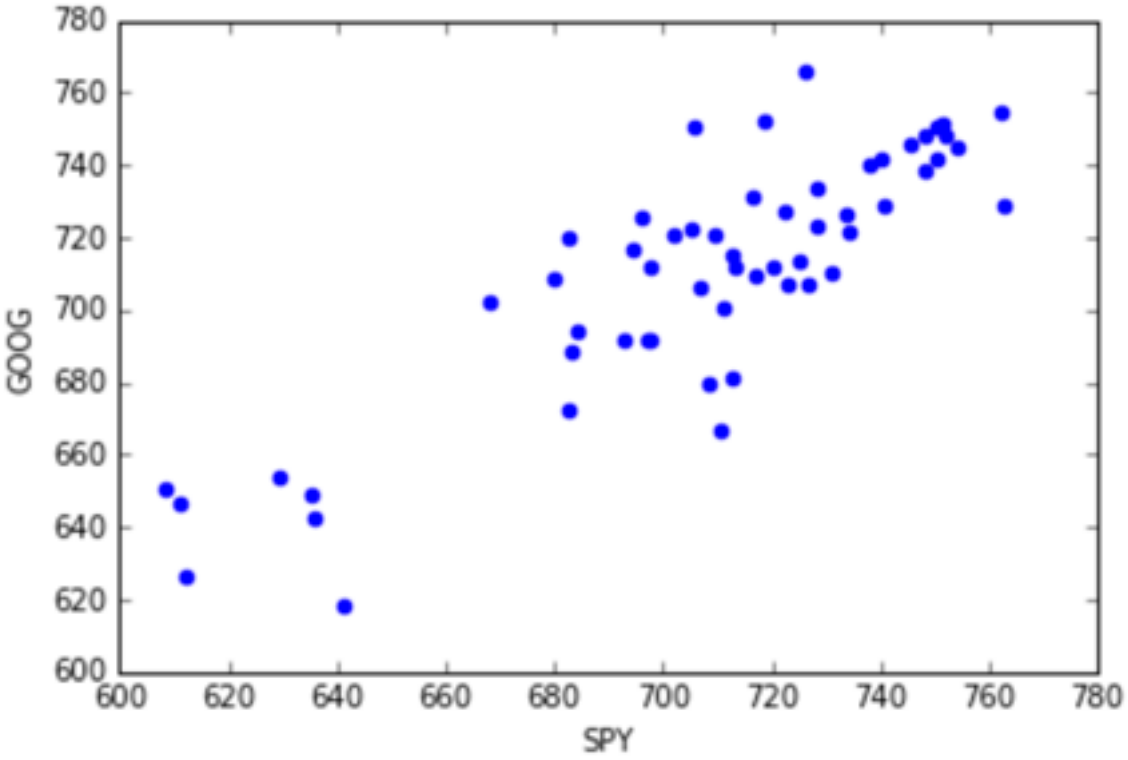
All the R^2 score and testing time of the experiments were recorded. The results can be shown as the following:

LinearRegression:

1. Train time: 0.000925064086914

2. Test time: 0.000237941741943

3. R^2 Score: 0.73

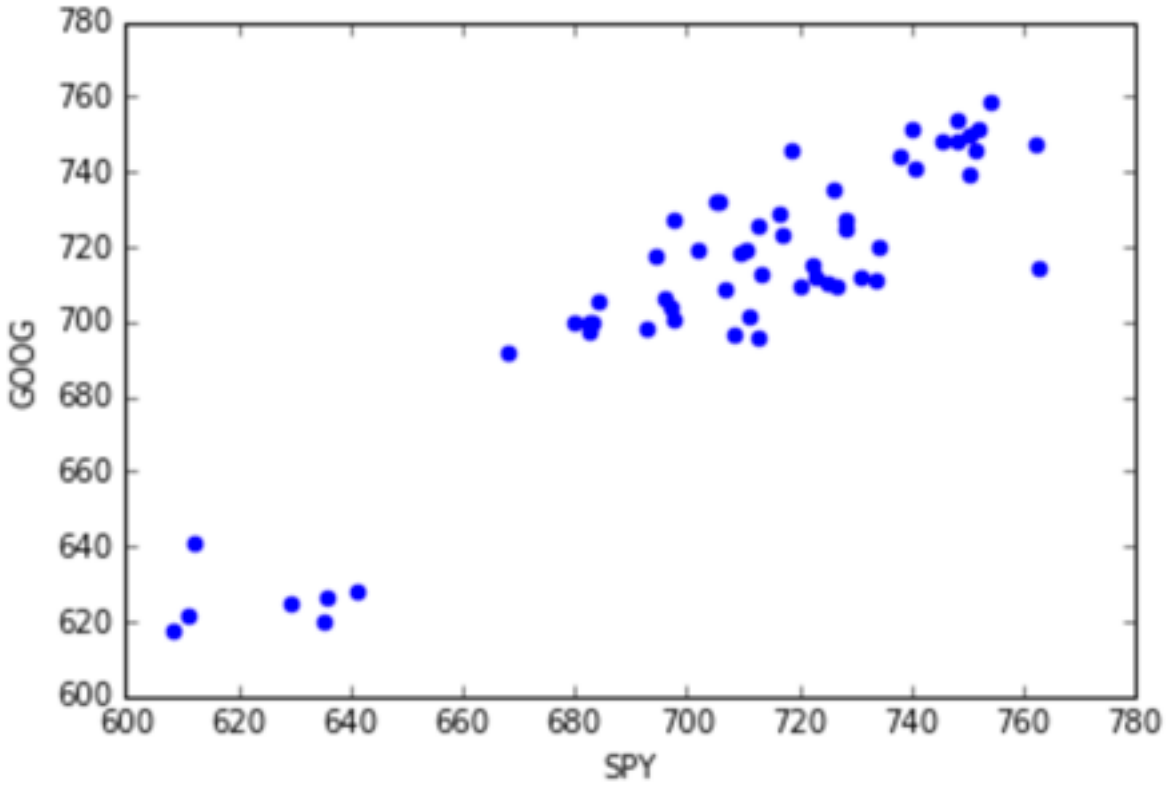


BaggingRegression:

1. Train time: 0.0224959850311

2. Test time: 0.000209093093872

3. R^2 Score: 0.83



From the experiment results, bagging spent less time on test than linear regression. What's more, bagging has higher R^2 score than linear regression. That is to say, bagging performs better linear regression. So I choose the bagging model to be the best model to do further improvement.

Robust Validation

I choose another stock price to validate the robustness of the model. This time I picked the stock of IBM to do the validation. The test result can be shown as the following plot.

	Predict Stock Price	True Stock Price
5	142.448256	144.153017
7	143.225609	140.364876
8	141.259493	142.278369
12	139.900933	141.248777
15	140.025249	140.811687

The R^2 score even reached 0.9 this time. It's reasonable to think that this model is robust.

Refinement

I ran GridSearch on my final chosen model of BaggingRegression to improve my model. The data is in a time sequence, so I can't disturb the order of the data while do cross validation. As a result, I choose 10-fold cross-validation and set random_state to 0. I adjusted parameters like max_features, n_estimators for bagging model to acquire improved solutions. Since I have four features in my training set, I tested [1,2,3,4] on max_features to see how many of these features are helpful to predict the stock price. The number of estimators will also influence the accuracy of the prediction. So I set [200,300,500] on n_estimators to see what's the most suitable value for n_estimators. It turns out that when the max_features is 3 and the n_estimators is 300, the model

performs best. Technical analysis only cares about the price and volume of a stock, maybe Bollinger Bands is not necessary here. It is entirely reasonable to set the max_features to 3. More n_estimator means more calculating, which consumes lots of time and resource. If the accuracy can't be further improved, it's better to use less estimators to improve the efficiency. 300 is a suitable value for n_estimator. At this time, the R^2 score finally raised from 0.83 to 0.85, which is above the benchmark.

Conclusion

There is some interest thing in getting target data. Since the data is in a time sequence, I adjust the sequence of the training data, and then I just got the target data. For example, I slice the data from 2015-09-01 to 2016-05-29, and I used samples from 2015-09-01 to 2016-05-22 as the training data and extracted the column of Google stock price from 2015-09-08 to 2016-05-29 as the target data. The way to pick target data in this project is tricky and I think this is really interesting.

The following are some predictions made by the refined model. I also made a plot to visualize the predicting result.

	Pridict Stock Price	True Stock Price
5	626.810003	635.979980
7	625.361502	629.250000
8	623.530567	635.440002
12	629.558419	611.969971
15	619.772295	608.419983

From the above results, the predict stock prices are pretty close to the true stock prices. With the help of this model, I can predict the future trend of Google stock price confidently.

To sum up, I analyzed the performance of linear regression and bagging algorithm on the prediction of future stock price. The training time of linear regression is less than bagging, but the testing time is almost the same. The R^2 score of bagging is 0.85, while the R^2 score of linear regression is 0.83. We care most about the R^2 score and the testing time, so we can conclude that bagging is more suitable than linear regression in predicting future stock price. The result beat the benchmark and it is reasonable to consider this model as a good tool to predict the stock price.

Improvement

I introduced the price and volume of Google stock and only introduced the price of SPY to as the reflection of the stock market. If the volume of SPY was also included in the dataset, the situation of the stock market can be reflected more accurate. This is helpful to predict the price of the stock. More training data will also leads to an accurate prediction. If I use more data and include the volume of SPY to train the model , I believe the model can be improved a lot.

Reference

[1]Bias–variance tradeoff : en.wikipedia.org/wiki/Bias–variance_tradeoff

[2]Linear Regression: en.wikipedia.org/wiki/Linear_regression

[3]Bagging: en.wikipedia.org/wiki/Bagging

[4]scikit-learn userguide: scikit-learndevelopers, Release0.17.1