Investment and Trading Capstone Project

Build a Stock Price Indicator

Machine Learning Engineer Nanodegree

Jie Hou
August 1st, 2016

I. Definition

Project Overview

Investment firms. Hedge funds and even individuals have been using financial models to better understand market behavior and make profitable investments and trades. A wealth of information is available in the form of historical stock prices and company performance data, suitable for machine learning algorithms to process.

In this project, I will build a stock price predictor that takes daily trading data over a certain date range as my data source, and produce projected estimates for given query dates. As enhancement, this predictor tool also allows users to choose their stocks and at the end, it would generate the best suggested buy point and date, and best suggested sell point and date.

Problem Statement

The goal is to create a general stock price predictor for a certain range of future dates. The tasks involved are as following:

- Download and preprocess the stock raw data
- Pick and choose the suitable features and label to be trained
- Train a regressor that learns the behavior of the stock price change
- Make prediction about a certain range of future dates
- Make suggestion on best sell and buy dates and points(price)

The final product would be considered successful if it can predict the stock prices and make correct suggestion on when and at what price is the best sell or buy points for the future dates with expected accuracy. (see benchmark below for each period time expectation)

Metrics

The R-squared function is used to provide a measure of how well future samples are likely to be predicted by the model. Here is the formula:

$$R^2(y,\hat{y}) = 1 - \frac{\sum_{i=0}^{n_{\text{samples}}-1}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n_{\text{samples}}-1}(y_i - \bar{y})^2}$$

Where:

$\sum_{i=0}^{n_{\text{samples}}-1}(y_i - \hat{y}_i)^2$ is the regression sum of squares (((true result value – predict result value)^2).sum()).

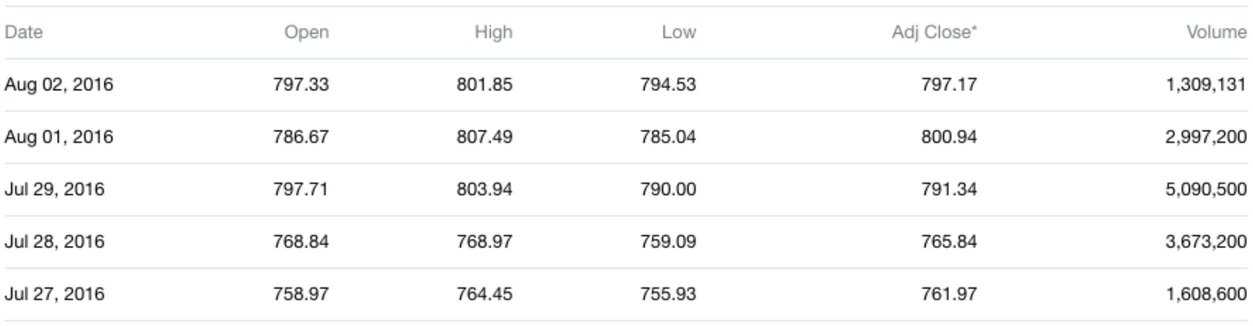
$\sum_{i=0}^{n_{\text{samples}}-1}(y_i - \bar{y})^2$ is the residual sum of squares(((true result value – the mean of the true result value)^2).sum()).

The best possible score is 1.0 and also it can be negative when the model performs poorly. A constant model that always predicts the expected value results, disregarding the features, would get a R-squared score of 0.0.

II. Analysis

Data Exploration

The data source used for this project is the datasets downloaded from Yahoo.Finance(*http://finance.yahoo.com/*). The dataset contains a list of the historical stock information for a specific stock name. In this project, only Google and IBM stocks are supported. There are 757 daily records(from 07-25-2013 to 07-25-2016) for each dataset(2 datasets in total). Here is an example of how the dataset looks like:



Date	Open	High	Low	Adj Close*	Volume
Aug 02, 2016	797.33	801.85	794.53	797.17	1,309,131
Aug 01, 2016	786.67	807.49	785.04	800.94	2,997,200
Jul 29, 2016	797.71	803.94	790.00	791.34	5,090,500
Jul 28, 2016	768.84	768.97	759.09	765.84	3,673,200
Jul 27, 2016	758.97	764.45	755.93	761.97	1,608,600

Figure 1 (http://finance.yahoo.com/quote/GOOGL/history?p=GOOGL)

As the picture showed above, the dataset has six columns.

- **“Date”:** shows the row data is for that certain date.
- **“Open”:** shows the open price for that stock
- **“High”:** shows the highest point the stock price reached
- **“Low”:** shows the lowest point that stock price reached
- **“Adj Close”:** shows the adjusted close price of that stock at the end of day
- **“Volume”:** shows how many shares has been traded during the day

Here is the definition of the adjust close price: An adjusted closing price is a stock's closing price on any given day of trading that has been amended to include any distributions and corporate actions that occurred at any time prior to the next day's open. The adjusted closing price is often used when examining historical returns or performing a detailed analysis on historical returns.

(*http://www.investopedia.com/terms/a/adjusted_closing_price.asp*) Since we are predicting the price, the “Adj Close” value would become the goal for our prediction.

Features play significant role to determine if the whole project would be successful or not. So carefully pick and choose useful features would become the key of this project.

Let's walk through all those existing features. “Date” feature is a continuous weekday-values, since it represents a continuous value of our weekday data, it would be the perfect candidate for our data index. “Open” and “Close” are also two very important features we have in the table. They represent the beginning price and the ending price for a “given date”. Because we have a limit feature here, meaning we don't have more information to understand what caused the beginning price, and what happened during the day to cause the close price, we can't see a clear picture of relations between these two features. So we can't use them as good feature candidates. Instead, with “Open” and “Close”, we would be able to figure out what the percent of the change between the beginning and the ending. So the change of “Open” and “Close” well represents the stock change during the day. So we would pick change as a good feature for this project. We will name it as “openCloseChg%”.

Same idea here for “High” and “Low”, we won't use “High” and “Low” directly here as our feature candidates, instead, the percentage of the change between “High” and “Low” would nicely represents the change between the highest price and the lowest price of the day. So we would pick high and low change as a good feature for this project. We will name it as “highLowChg%”.

Since “Adj Close” is our target, we would keep it unchanged in the table.

“Volume” is a very important figure in this project, it give us the number of the shares being traded during the day. It plays a key role of determine the stock price. So we would also keep it unchanged in the table.

To summarize here, the new features for this project would be:

Date	Volume	HighLowChg%	OpenCloseChg%	Adj Close

Exploratory Visualization

The plot below shows what would each feature be look like when put them all together. With features plotted, we would have a more direct feel about how our feature data would look like. This would also be very help later to determine what machine learning algorithms would be a good choice to learn and predict our future price.

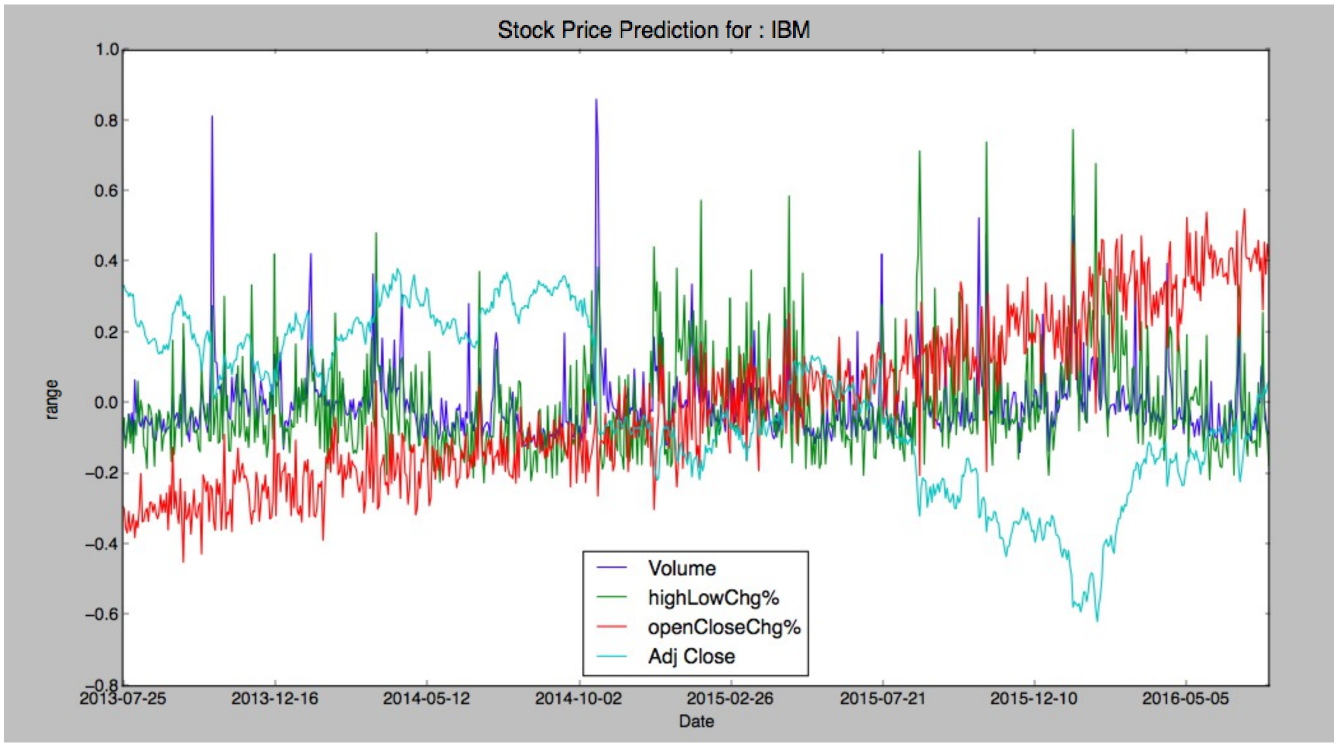


Figure 2

The X-axis in the picture above is the Date, the index of our data. Y-axis is a value range between -1 and +1. The reason why it is showed like this is because in our data feature set, we have highLowChg% and openCloseChg%, which are the percentage of high-low, and open-close. The value of that percentage is between -100%(open value might be lower than close value) and +100%. The Adjust Close price in this Google stock example is between $500-$700, and the Volume is a set of pretty big number(7 digits). When put all of them together without feature scaling, those two percentage values and the Adj Close value would be all stick around y-axis 0 line(because of the huge volume value, the y-axis unit would be 1e7, other feature values compared with 1e7 would become almost 0-ish). The before the plot, I chose to normalized the data to make all of them in between -1 and 1. which could give a more meaningful picture about how those features look like. After investigated the feature picture, it looks like a good linear would well describe the data.

Algorithms and Techniques

Based on the visualization of the features table, a Linear Regression algorithm would be a good choice in this project. Linear Regression algorithm is an approach for modeling the relationship between a scalar dependent variable y and one or more explanatory variables denoted X. In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. It has many practical uses nowadays. Especially in finance industry. As the picture showed above, all the features look like fit well linearly.

Also, there are lots of other Regressions algorithms available, such as Support Vector Regression and Decision trees. They all can be used as an algorithm in this project. For the Support Vector Regression, the model is produced by support vector regression. It depends only on a subset of the training data, because the cost function for building the model ignores any training data close to the model prediction. In the stock trading market, there are some stock indexes which prices change very “gentle” (meaning they don't change too much from time to time, such as some mutual funds index). The every day “close” price might be very similar to each other,, which might also be true for the future prediction. As the SVR definition says that it would ignore some of the training data close to the model prediction. If there is only a slightly change for everyday's “close” price, we might lose lots of training data if we use the SVR algorithm. Decision Tree algorithm is also a powerful algorithm for this regression projects. It's easy to implement, and very straightforward. And very little data preparation is needed if use Decision Tree. But the reason that Decision Tree is not considered as an option here is because the Decision Tree is unstable. The reliability depends on how the data is fed. Even a small change in input data can cause a large changes in the tree. And even we don't need to worry about the data feeding, we might still need to worry about how deep we want the tree to be in order to avoid the possibility overfitting issue.

Based on all the facts we have in the project, the LinearRegression algorithm would be the most appropriate option for this project. To apply the LinearRegression algorithm, all the features “date”, “highLowChg%”, “openCloseChg%” and “Volume” would be used as feature input denoted in X, and the prediction of “Adj Close” in the future would be our target denoted as y. After the successful prediction, the R^2 score would be calculated which would tell the performance of our machine learning algorithm.

Benchmark

The purpose of this project is only to create a tool to predict the stock price based on machine learning algorithm and the limit information I have. In real world stock trade practice, there are lots of other features/information should be considered to make a more accurate prediction. This tool is NOT made for real world stock trade price prediction.

My goal of this project is with the data I got from yahoo.finance, make the prediction of next 7 days, 14 days and 28 days (after the datasets' dates) of the stock “Adj Close” price prediction for two listed companies “Google” and “IBM”.

For 7 days prediction, the goal of accuracy is to reach a score point at 0.85.

For 14 days prediction, the goal of accuracy is to reach a score point at 0.75

For 28 days prediction, the goal of accuracy is to reach a score point at 0.65

III. Methodology

Data Preprocessing

The original data downloaded from yahoo.finance includes the past three years daily trading history. There are 7 features in that table: “Date”, “Open”, “High”, “Close”, “Low”, “Adj Close” and “Volume”. In order to make all the features useful, meaningful and more relevant to each other, I created 4 new features:

“highLowChg%”: represents the percentage change of high price and low price

“OpenCloseChg%”: represents the percentage change of the open price and close price

“Volume”: remains as it is

“Date”: remains as it is

Since “date” is a continuous value that represents every weekday, it is picked as the index of our data table. “Adj Close” is marked as our target (denoted y in the Linear Regression Algorithm). So it is set aside separately.

The value of both “highLowChg%” and “openCloseChg%” are between -1 and 1, and the “Volume” has a 7 digits value, so a feature scaling is needed. Also, there might be a chance that the company's stock exchange is closed while the market is still running, we might end up with naN(no data in that date), those days would be considered as outliners, and a special value -9999999 would be filled up in those fields. After the scaling process, all the values for all the feature are between -1 and 1. All the outliners would be ignored.

The scaled feature dataset would look like the Figure 2 above.

Implementation

The implementation can be split into 4 steps:

1. The dataset preparation(split to training and testing datasets) and regressor training stage
2. The regressor predict stage
3. make suggestion of best sell/buy stage
4. plot final result stage

Stage1: In this stage, based on the user's request (how many days of period to be predicted in the future), the new predict labels for future prediction need to be created. The shift() is used to shift the label(“Adj Close”) data up to the future dates. Meaning we will always have the current features to predict certain days period in the future of the “Adj Close” value. After that, we need to resize the feature dataset by separating the training set and predict set. After that, a cross_validation.train_test_split() will shuffle all the data records in the dataset and split the dataset in to two different datasets called training dataset and testing dataset. the training, testing datasets split ratio is 8:2. The final step in Stage 1 is to create a linearRegression classifier, and pass in the training dataset for training.

Stage2: In this stage, things are all taken care of by the python. What the code does is to pass the predicting dataset to the classifier.predict() method. The final predict results would be returned back. In the stage, we also want to know how well the algorithm performed, so the R^2 score is used here to report how well the algorithm worked.

Stage3: In this stage, a best buy/sell point would be made as a suggestion to our users. In order to figure out the best buy point, we need to run through the predict result to figure out what's the lowest value in the certain period of future days. The lower the price, the better the buy point would be. After the lowest price was found, we return the index value which is the “date” correspond to the lowest price back to the user. Similar work is done for best sell point. We go through the predict result and figure out the highest price, together with the “date” and return it to the user.

Stage4: In this stage, it would be nice to show a professional price trending picture of a certain stock in a period of time, and mark down the best sell/buy point in the picture. So I implemented plot() with a “ggplot” style, “Date” as X-axis and “Price” as Y-axis to give a closer picture look of the data.

Refinement

In the benchmark showed above, the goal is set 0.85 accuracy for 7 days prediction, 0.75 accuracy for 14 days prediction and 0.65 accuracy for 28 days prediction. I was struggling in the beginning of the project. After I finished the data scaling, dataset splitting, training and predicting, the score I got was -0.3 for 7 days, even worse for 14 days and 28 days, which means my algorithm performed extremely poorly. At that point, I doubted if the LinearRegression is the best fit for my project. I tried other supervised regression algorithms such as Lasso, Bayesian Regression. None of these would help to bring the score back to positive number(even with the tune). Couple days later after I looked back to what might cause the issue, I figured that I didn't feed enough training data to the algorithm. The data I gave was only one year period of weekly data. The reason I chose weekly data in the beginning is because since I will provide 7 days, 14 days and 28 days prediction, which are just equal to one week, 2 weeks and 4 weeks. Which was proved by myself a bad decision. Reasons for this is one year period of weekly data, would only have 52 data record, which might not be good enough to predict. Also, stock trade market open 5 days a week, excluding weekend and holidays. For 7 days prediction, it would be one week of trading data plus another two days. So it won't fit in a weekly category(this might be the main reason).

Few days later, I went back and downloaded the new dataset with 3 years period of daily data, and passed into the algorithm for training and testing, I got the result as I expected.

IV. Results

Model Evaluation and Validation

The LinearRegression has a good performance in this model.

The total dataset has 756 data records for training and testing. The training time when 7 days is required is 0.069 seconds, and the predict time when 28 days is required is 0.00013 sec.

The model was built based on using Google as the stock model. Because during the period of day I picked, Google has a strong growth performance in the real stock market. It would be more Linear fit to my model. After the training with the Google data, I got:

	7 days (0.85)	14 days (0.75)	28 days (0.65)
Google	0.93	0.91	0.83

The result is much better than I expected. In order to make the model for general, I picked another stock IBM as a test validation example. Because IBM didn't have a good performance as Google does in that period of time, meaning it won't be that linear. After train the same amount of data and make prediction, here is what I got:

	7 days (0.85)	14 days (0.75)	28 days (0.65)
IBM	0.9	0.8	0.66

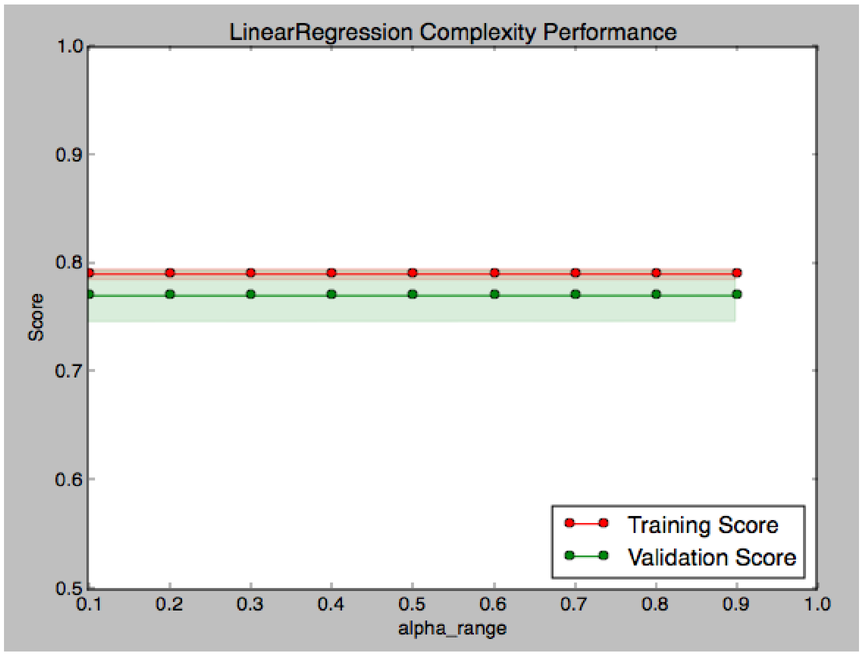
Although the model didn't perform as good as when testing google, but the result is still better than the goal.

The intercept I found for this LinearRegression is 158.442289216, and the estimated coefficient is [-0.32222803, -0.58809079, -0.60068231, 14.32715632].

In regularization term, I used the RidgeCV Regression. The Algorithm is tuned by using different Alpha values([0.1, 0.5, 1, 10]). The result shows the best fit is when alpha = 0.5. Here is the result with different alpha values:(note: the Google dataset and 28 period days are used in this example)

Alpha	**Coefficient**	**intercept**
0.1	[-0.3225398 0.5871149 -0.60397826 14.32218436]	158.442267178
0.5	[-0.32378048 0.58322029 -0.61711387 14.30234931]	158.442179682
1.0	[-0.32531702 0.578372 -0.63342559 14.27767365]	158.442071774
10.0	[-0.35046004 0.49474205 -0.90776911 13.85471125]	158.440389467

The training score and test score are validated by using validation_Curve(). Here is the calculation of the performance:



Based on the picture shows, because the training score and the cross validation score are pretty close to each other and running almost parallel, we would be able to say that the model is robust.

On the enhancement side, the model performed great for making suggestion for best sell/buy point. This is validated by manually looking at the predict result and date, and comparing with the suggest result generated by the model. The final result matches what we expected for both Google and IBM stocks.

The plot() function at the end of the model gives a closer look at the price change trend by using the different color to illustrate the current “Adj Close” price and predict “Adj Close” price. Here is the snapshot for that plot result.

Justi fication

The Accuracy for the model training/predicting is:

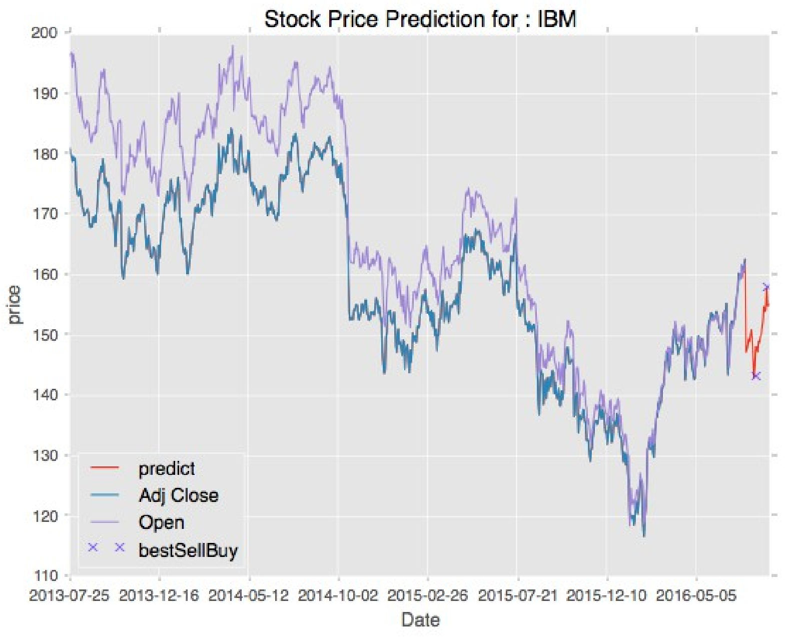
	7 days (0.85)	14 days (0.75)	28 days (0.65)
Google	0.93	0.91	0.83
IBM	0.9	0.8	0.66

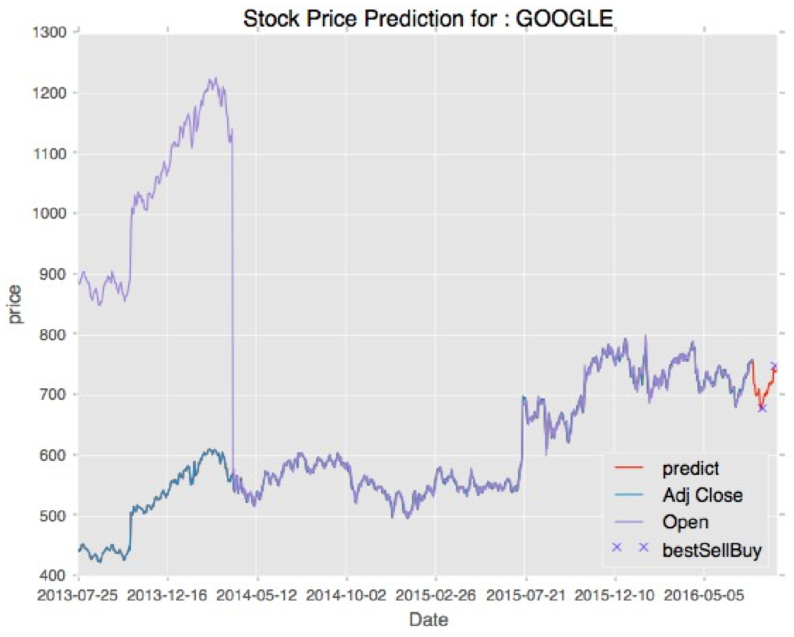
As the table above showed, the accuracies for these two stocks meet the goal setup in the benchmark. Also the model makes correct suggestions on the best sell/buy points.

V. Conclusion

Free-Form Visualization

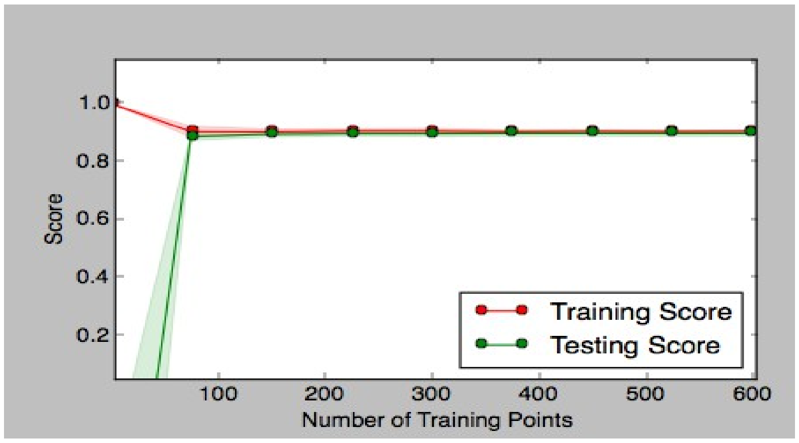
Here is the final result visualization.





As showed in the previous picture, in the X-axis(Date) Y-axis(price), the blue line shows the current “Adj Close” price for each stock price. The red line at the end shows the predict price for the predict dates. Also, the purple line shows the open price for each date and the line follows the “Adj Close” and predict pretty well. (Open price equals to the previous day's close price) The blue “X” marks point out the best Sell/Buy points. The X-axis corresponds tot he “X” mark is the date for best Sell/Buy points.

Here is the training and testing performance comparison:



The comparison picture shows the performance of training and testing. When we have more than 100 training examples, the training score almost equals to testing score. Which means we don't suffer the overfitting problem.

Also, the model prints out three messages:

example: Google, 28 days

output:

predict accuracy is: 0.83

best sell point is: 746.79 at 2016-08-25

best buy point is: 677.64 at 2016-08-05

Reflection

The process used for this project can be summarized as:

1, Initial problem and think of possible solutions. In this case, what algorithm should be tried first.

2, getting public data, understanding and analyzing each features

3, setup benchmark for the model

4, perform any necessary data preprocessing

5, try and determine the algorithms

6, tune if necessary the algorithms to meet the goal

7, additional enhancements

Throughout the whole project. I found the most challenging parts are step 2. There are actually only certain number of machine learning algorithms. And they are also separated to two categories- Supervised and Unsupervised Learning. Deciding which category the problem falls into is not that difficult. Once we decide which learning it is, it would be simple to pick some of the popular algorithms to try it on. So I think choosing algorithms and following that(meaning tune to meet the goal) is not the tough steps. On the other hand, the key of making a good machine learning model is the data. How to understand and analyze data is the most tough steps. Like in this project, in the beginning, every feature in that dataset seems important, but if we just choose to use those features directly, it would be so hard to understand the relationships between those features and the final predict(y). In step 2, after I got the public raw data, I spent half of the project time to understand the data, and find out the relationships between them and finally create new reasonable features. Then everything else would become much smoother.

Improvement

This is a stock investment predictor tool. If someone wants to use it, then it has to be well trusted(money related thing is a serious thing to most of the people). This project is based on the current public available data we can found on the internet. Although the prediction is pretty accurate(for 7 days), I would still say “don't use it to try to make money on it”, because predicting a stock price is not only a case like this project, it also requires lots of professional finance knowledge. One improvement I can think of is to get more features such as the other related company's stock performance. For example, let's say Google, if one of the biggest google advertisement sponsors left google, that would affect a lot on google's stock performance. Etc. in general term, lots of other features related to the company's performance would affect the stock performance. So if we want to improve the predictor tool, more useful features would help to make the tool more trusted.

Reference:

http://finance.yahoo.com/
http://finance.yahoo.com/quote/GOOGL/history?p=GOOGL
http://www.investopedia.com/terms/a/adjusted_closing_price.asp