Capstone Project :
Build a Stock Price Indicator
Shu Huang
Mar, 2018

Definition:

Project Overview:

The purpose of this project is to apply fundamental machine learning techniques to stock market to investigate whether machine learning algorithm could be utilized for prediction of stock price. Although the market is complicated and influenced by many factors and research of machine learning algorithm has been applied to stock market, Kim et al, 2003 has conducted research on prediction of the direction of stock on daily time series using SVM, Tsai and Wang, 2009 research on forecasting of stock price using ensemble learners. In this capstone project, I chose to investigate prediction of stock price using machine learning models.

The main data set I used in this project is Google's historical dataset from 2006 to 2016 downloaded from yahoo finance (https://finance.yahoo.com/quote/GOOG?p=GOOG). In terms of background knowledge to understand this project, people with basic machine learning concepts and stock trading experience would be easy to understand this problem and the approach to solve it.

Problem Statement:

The main goal of stock market is to choose right stocks which might have an increasing price in the short time or long run and therefore gain profit.

I will treat this problem as a regression problem and choose different regressors to solve it. In this project, it targets at a prediction of google's price of 7 days later using historical data since 2006. The features would include all the columns in the original data set and additional technical indicators Bollinger Bands and Simple moving average (sma). I will try different models from simple one to complex model and also investigate the size of training data influence on model performance.

The expected solution would be a selection of a model with good metrics valuation. Although this project use google's historical data, other stock historical data can also be predicted by the same approach.

Metrics:

The metrics I would like to use is root of mean square error (RMSE), which measures the vertical distance between the data and fitted line. It's the square root of the average of squared

differences between prediction and actual observation. I will use metrics to compare model's performance.

I will also show R2 score of each model for individual training quality.
R squared measures how close the data is fitted to the regression line. It is coefficient of determination and always between 0 and 1. 0 indicates the model explains none of the variance of data around mean and always predict the y value disregarding the input features and 1 indicates the model explains all the variance of data around mean. In general , the higher r2 coefficient, the better the model fits the data.

Analysis:

Data Exploration:

I used the stock data of Google from Jan. 1 2006 to Dec. 31 2016 downloaded from Yahoo Finance.
There are 7 columns in the data set, including :

Date: trading date of the stock
Open : opening price of the stock
High : highest price of the stock in the trading day
Low : lowest price of the stock in the day
Close : closing price of the stock
Adj Close : adjusted price of the stock, it would be different from close price is splitting or dividends happened.
Volume : trading volume of the stock

Sample data of 'GOOG' is shown below with 7 features and data is indexed from 1. There are 2769 rows in total: no missing data/null data point in the dataset, no ZERO values in the dataset.

	Date	Open	High	Low	Close	Adj Close	Volume
1	2006-01-04	220.515762	223.029404	218.454163	221.181427	221.181427	30771600
2	2006-01-05	221.558975	224.316040	219.323517	224.162048	224.162048	21757200
3	2006-01-06	226.958847	233.729813	225.155579	231.325455	231.325455	35744800
4	2006-01-09	231.698029	235.170441	228.980698	231.941452	231.941452	25750100
5	2006-01-10	230.709457	233.605621	229.527145	233.362198	233.362198	18312400

6	**2006-01-11**	**234.112320**	**236.019913**	**233.074081**	**234.291168**	**234.291168**	**18131900**
7	**2006-01-12**	**235.329407**	**235.960312**	**229.258896**	**230.317017**	**230.317017**	**20382200**
8	**2006-01-13**	**230.654816**	**231.936478**	**229.313538**	**231.618546**	**231.618546**	**15412700**
9	**2006-01-17**	**230.033844**	**233.431747**	**229.770569**	**232.045761**	**232.045761**	**16648100**

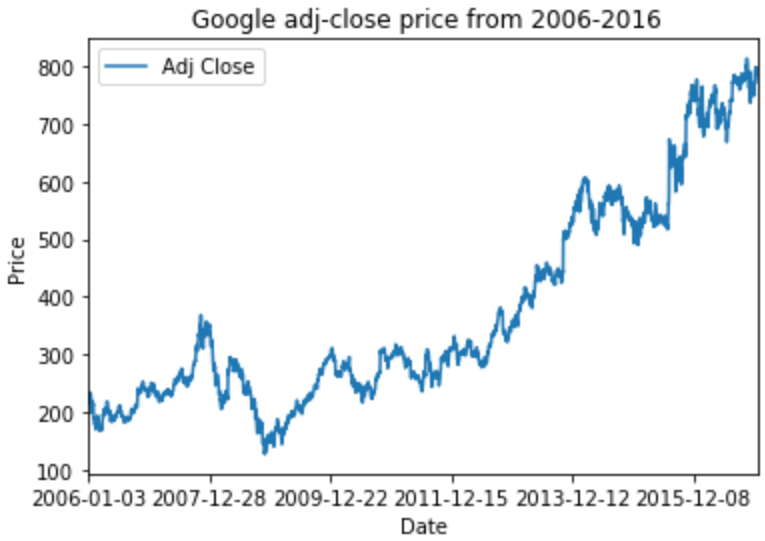
This table below shows the statistics of data, there is not much meaning could be taken from this table. It just gives an range of the price.

	Open	High	Low	Close	Adj Close	Volume
count	**2769.000000**	**2769.000000**	**2769.000000**	**2769.000000**	**2769.000000**	**2.769000e+03**
mean	**375.902446**	**379.276280**	**372.115031**	**375.714687**	**375.714687**	**6.931651e+06**
std	**179.439162**	**180.423646**	**178.249847**	**179.377893**	**179.377893**	**6.539445e+06**
min	**130.406830**	**133.814667**	**122.850975**	**127.888214**	**127.888214**	**7.900000e+03**
25%	**240.778961**	**243.709900**	**238.588211**	**240.858444**	**240.858444**	**2.821100e+06**
50%	**298.727081**	**301.851746**	**296.461792**	**298.558167**	**298.558167**	**5.028200e+06**
75%	**528.318787**	**532.658752**	**524.247375**	**528.444336**	**528.444336**	**8.763800e+06**
max	**816.679993**	**816.679993**	**805.140015**	**813.109985**	**813.109985**	**8.276810e+07**

Exploratory Visualization:

Google's stock price

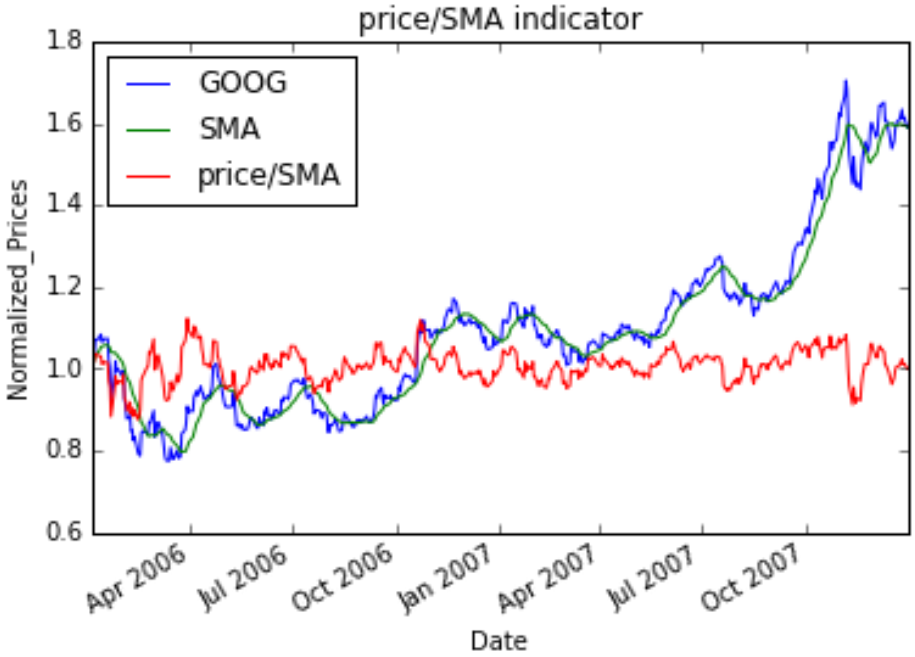
The plot of google's price from 2006 to 2016 is shown below. It shows the price changes with date to get an overview of historical data. The price is increased from 200 to 600 dollar per share.

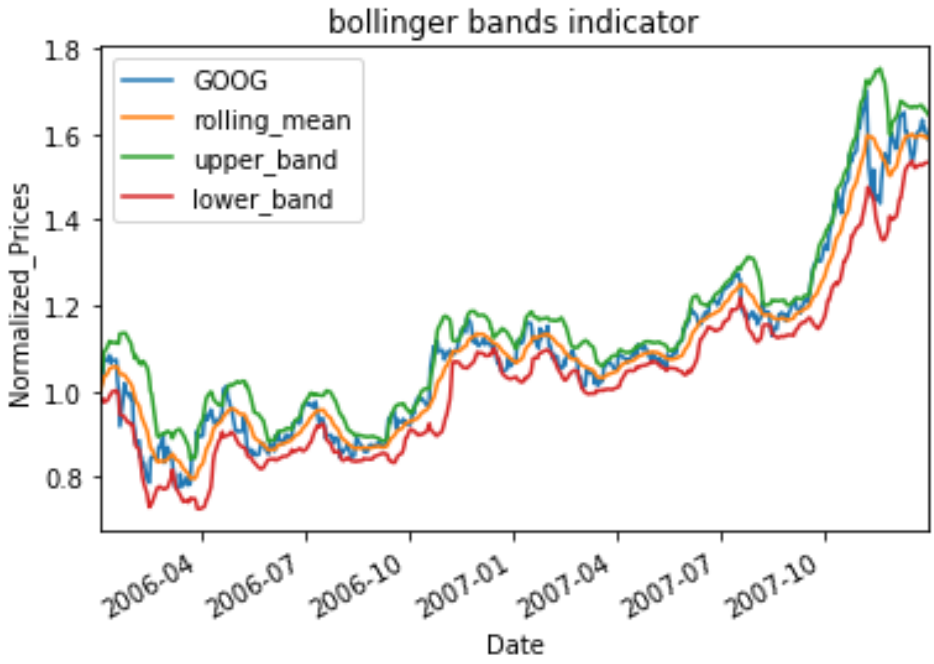


Technical Indicators' plots

Two technical indicators' plots are also shown below. Technical indicators are metrics derived from past price activity in a stock and used to predict the general price direction by looking at the past patterns.

I am using three technical indicators: price/SMA (simple moving average), Bollinger Bands and SMA is calculated by mean of rolling price of a 7-day window. Price/SMA shows price rising or falling tendency. Momentum = price/SMA -1, which distribute around -0.5 to 0.5. In this figure, I just present the value of price/SMA without subtracting 1.





Algorithms and Techniques:

For this stock project, I will treat it as a regression problem. So I would like to use different training algorithms (**Linear regression , KNN, Decision Tree, Random Forest**) and compare their performance.

Linear regression:
Since I choose to solve this problem in a regression mode, the first model come to my mind is linear regression. Linear regression fits the input data into a straight line , the equation is y = ax + b where x is independent variable , features in this project, y is dependent variable , prediction of price of 7 days later. Fitting a regression line usually employ a least squares method, which calculate the best fitting line for input data by minimizing the sum of the squares of training y to the fitting line. In this project, I use linear model from sklearn.linear_model, the methods below are mostly used :
fit(X, y)

fit(X, y[, sample_weight])	Fit linear model.
predict(X)	Predict using the linear model
score(X, y[, sample_weight])	Returns the coefficient of determination R^2 of the prediction.

K-nearest neighbours :
KNN is an instance based algorithm. The target y value is predicted by local interpolation of the targets associated of the nearest neighbors in the training set. It gives a weighted average of regression function in a local space. I use sklearn KNeighborsRegressor(). The methods are similar to linear regression.

Decision Tree :
Decision Tree is a logic-based algorithm. It can be applied both categorical or discrete data. I will use decisionTreeRegressor in this case. It fits a sine curve with additional observation and learns from local line regression, which approximates the sine curve. The maximum depth of tree controls the learning process. A higher depth will result in learning in noise data and overfitting. The pros of decision tree include high training speed, high intolerance to missing data and noise data and easy to understand the logic and to visualize The cons are low accuracy and overfitting.

Random Forest:
A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

The data is time series data, shuffling is not good for data set split. Because after shuffling, the time ordering is not observed and the training data may seem future dataset and will result in overfitting.

Benchmark
The benchmark model would be a KNN model , which is trained on first 1000 rows and prediction of test data (1000 :1252 rows). Its root mean squared error (rmse) is 0.18897.

Methodology:

Data Preprocessing

The dataset is a CSV file format, have been processed as following steps :

- First of all, the original data set is indexed by number, starting from 0. The dataset quality is good and no missing data/ nan or zero value data is found in the dataset.
- Adj close price is normalized before calculation for technical indicators.
- Indicators calculation :
  Bollinger bands and simple moving average(SMA) of 7 day are calculated by normalized adj-close price, upper and lower bands have been calculated separately.
  Bollinger bands $(n)_t = (y_t - sma(n)_t) / 2 * std(n)_t$ where y is the price at date t , n is for past n trading days and sma is moving average, std is standard deviation.

- After Calculation and feature appended to the original dataset, this processed dataset is used for training and testing.

- training/testing data split : Since the data is time series data, shuffling is not good for data set split. Because after shuffling, the time ordering is not observed and the training data may seem future dataset and will result in overfitting. I choose training size (1000 rows) and testing data set is 252 rows following the training data.
- target value : price of 7 days later is appended to the last column of the original dataset as targeted y value.

Implementation

For this stock project, I use Linear regression , KNN, Decision Tree, Random Forest algorithm to train the data. These models are imported from SKlean and predict and fit methods are mostly used.

The first step is simply applying those models and no tuning the parameters yet and just get a basic sense of how these models will train the data and their corresponding performance and get an initial result , which is displayed at refinement section. For Initial implementation , the training/testing split ratio is 5:1 .

The second step is to refine the algorithms and all the details are in the refimentment section. Third, I will validate the final model to test it robustness for the whole dataset (2200 rows in total) and then average the results to show how it fits to unseen data.

Refinement

Initial Results : I choose the first 1000 rows as training dataset and 252 rows after that as testing dataset. Initial results are summarized as below:
Linear regression model has the best RMSE, which is 0.058. KNN in the initial training will serve as bench model.

	RMSE
Linear Regression	0.05831284563185253
KNN	0.18897031188369667
Decision Tree	0.08891343634683321

Random Forest	0.07352555771905804

The process of refinement :

Linear Regression : I refined Linear Regression model by dropping some features(drop close price, since I already use adj-close price, drop price/sma) to see if it could have better performance.

I use GridSearchCV from SKlearn for refinement of Linear Regression , KNN , Decision Tree and Random Forest.
KNN : I varied k value from 1 , 5 to 30 at a step of 5 and leaves as the same range as k .
Decision Tree: change max depth from 1 to 7, and best estimator shows the best max depth is 5.
Random Forest : The tuning parameter n_estimators , which is the number of trees in the forest from 10 to 70 denotes the number of trees in the forest.

Refined Results:
Linear Regression : not much improvement.
KNN : After refinement, the rmse of knn improved a lot, decreased from 0.19 to 0.07
Decision Tree : the rmse of decision tree improved from 0.089 to 0.067 at training size 1000
Random Forest : it improved slightly with a score from 0.073 to 0.067 at training size 1000

	RMSE
Linear Regression	0.0583506683985424
KNN	0.07136776233426978
Decision Tree	0.06770349406705777
Random Forest	0.06795244820852127

In summary :

- Linear regress show the best rmse (0.058) among the four models.
- The other three models (KNN, decision tree, random forest) has a similar score l after refinement at training size 1000.
- Decision Tree and Random Forest have similar rmse after refinement.
- In summary , Linear regression model is final model chosen to solve this problem , not just because it has the least rmse. According to r2 score, it also shows it fits the model well.

Results

Model Evaluation and Validation

I applied 4 training algorithms to google's historical data. Linear Regression performs better than other algorithms, which is not what I expected at the beginning. I thought a appropriate model would be a more complex model . I did feature turning of refinement for linear regression model , which didn't have no significant improved results.

I think linear regression model is robust enough for a prediction of a price several days later. It can provide a good estimate of future. I did k-fold validation for linear regression model , following this method (http://francescopochetti.com/pythonic-cross-validation-time-series-pandas-scikit-learn/). The data (2200 rows in total) are divided into 10 folds, each fold has 220 rows. The split ratio of training/testing increased from 1/1 to 9/1. The process is as follows : first I trained on first fold and tested on the second fold. Then on the second iteration , trained on the first and second, and tested on the third fold.. and so on .In such sequences, the time order is observed. Finally I average the rmse of this 9 splits results. The average rmse is 0.067.

The linear model can be trusted because it provides general estimation of price prediction and price trend in a short term. Although real trading strategy may not be decided based on this simple model.

Justification

The benchmark in this project is KNN model, which is trained on first 1000 rows and prediction of test data (1000 :1252 rows). Its rmse is 0.189. The linear regression model beats the benchmark with a 0.058 rmse.

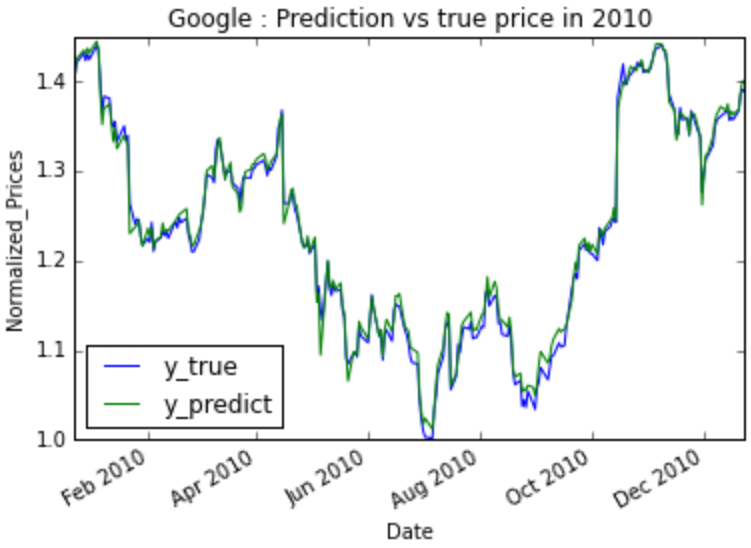
I analyzed and discussed the final results in refinment section.

I think the final model - linear regression model solved the problem since my target is to predict a future price of 7 days later. The model works well for prediction .

Conclusion

Free-Form Visualization

The following plot is the normalized price of prediction vs test price in 2010. It shows the prediction line match closely to the real price line. I would like to display how the prediction price looks like visually.



Reflection

Summary of the entire process:

I analyzed the google's stock price from 2006 to 2016. In this ten years, the price increased from 200 to 800 dollar , there are up and downs though. I chose this stock as an example for solving price prediction problem and didn't do anything specific for this particular stock , so this solving approach still applied to other stock as well.

The data processing involved dropping missing data and appending technical features that might help for predictions. In this particular case, I added Bollinger bands and Simple moving average indicators.

I chose four algorithms , linear regression , knn , decision tree and random forest to train the data. Linear regression performs way better than other models with a 0.058 rmse value and it is simple and generalized well. After the initial valuation of the four models , I did refinement for them individually. I tuned features for linear regression model , unfortunately, it didn't improve much. I also used gridsearch for knn , decision tree and random forest , tuning their paraments such as , leaves number, max depth, number of trees, which improved their results significantly.

A k-fold validation is also conducted on the final model - linear regression model . The process is summarized in the validation section.

The interesting aspect of the project is about models complexity. In this stock problem, it seems simple model work better than complex models, which may due to its regression property. I was expecting a more complicated model at the beginning. I did learned a lot by completing this project and understood more when applying those models to dataset.

The difficult aspects of this project is how to frame the exploration of this stock problem. Since this project is open ended, first of all I should define a problem - what I want to solve? and then the approaches and techniques to solve the problem. I chose a simple and straightforward problem- prediction of price to define the goal of this project. Second, do you want to treat this problem as regression problem or classification problem ? which will determine your metrics to evaluate and methods to use. I chose to consider it as a regression problem since it is numeric and I am not just consider that price is increasing or decreasing, I would like to know a estimate of a number. Third, tuning parameters is also not easy, for example, in decision tree model , I tried to loop for more max depth, but it turns out a worse performance. However, the tuning process did help me to understand the model more than just call the functions.

The final model, linear regression model is not what i initially guessed about model selection. But It does show a good performance on historical data and the approach is very straightforward without any refinement. I think it can be applied to other stocks as well.

Improvement

There is definitely improvement can be made on algorithm and techniques. For example, add more technical indicators to help with prediction, try reinforcement learning, Since in modeling experiment, we could know the performance by the metrics (R2 score), we know how to tune training size and other parameters to get a higher score as possible. But in real life, we have no idea how the real price will be . A good model perfectly predicted the past may not work in the future. There adding more influential factors and ensemble learners may be more effective in prediction.

I would like to apply deep learning to stock market if I knew how, which might involve other factors , other just historical data points for predictions.

If I use my final solution as new benchmark, I think there are better solution exist because my model is relative simple and I didn’t do much feature selections for better results.