Machine Learning Engineer Nanodegree
Capstone Project
Kailin Huang
May 14th, 2017

I. Definition
Project Overview
According to the market efficiency theory, share prices always incorporate and reflect all relevant information, thus it's very difficult, if not impossible to outwit the market and make substantial gains. But people have been interested in predicting stock prices for the potential of lucrative return. As technology brings innovation to all industries imaginable, machine learning has been influencing the financial industry, including stock trading. Indeed, there are many hedge funds increasingly rely on machine learning techniques that are optimized for speed and reliability [1]. Some strategies use natural language processing and sentiment analysis to mine tweets and news feed. In conclusion, machine learning is a promising technique to predict stock prices.

This project focuses on using machine learning to predict stock prices, with historical stock price data from online stock price database (Yahoo finance). The prediction implemented here doesn't use fundamental analysis, which focuses on underlying factors, such as company financial report, but focus on potential movement of past price. The feature space is derived from time series of the stock itself. As each single stock is subject to more noise compared to the noise of market indices (SP500, NASDAQ), and each stock is susceptible to the overall economic performance of the industry it belongs to, so this project will focus on one industry, the tech industry.

Problem Statement
The goal of the project is to build a stock price predictor and predict Adjusted Close price (the price has been amended to include any distributions and corporate actions that occurred at any time prior to the next day's open, examples are stock splits and dividends) [2]. The predictor takes daily trading data over a certain date range as input, and outputs projected estimates for near future dates. Inputs will contain multiple metrics, such as opening price, highest price the stock traded at, how many stocks were traded and closing price adjusted for stock splits and dividends.

Since the goal is to predict continuous values (stock adjusted close prices), regression and time series forecasting are best suited for this task. The key factors

of the solution strategy include selection of features, which is fundamental to the success of regression predictor, and determination of autoregressive factor and moving average factor. In linear regression, since any stock price is affected in some degree by macro market trend, which is susceptible to major economic and political events, so major market indices (SP500, interest rate, etc.) will be considered in model training as well. U.S. 10 year bond will be considered as well, because it is an important benchmark of interest rate, which affects investment activity [6]. After taking the market trend into consideration, I will proceed to consider unique property of the target stock, such as the daily price fluctuation (daily high – daily low), and Beta, a finance measurement indicating whether a stock is more or less volatile than the market.

This project focuses on the stock prices of ten tech companies: Google, Facebook, Tesla, Amazon, Apple, IBM, Nvidia, Microsoft, Cisco and Yahoo.

Metrics

To measure the success of stock predictor, mean square error is used. The equation is MSE = $(1/n)*\Sigma(y\_pred_i - y\_true_i)^2$, where y_pred is the predicted price of stock, y_true is the true price of stock .MSE is the average of squares of the prediction deviations. MSE is a good metric for predictor performance because it measures how accurately the predictor predicts the true values overall. MSE is always non-negative and the close the MSE is to zero, the better the estimator performs, while a big value signals poor fit. [7]

II. Analysis

Data Exploration

We have a well–defined input space: given a time range and a stock symbol, we can retrieve those historical stock data. For this project, the target stock is 10 tech companies (Google,Facebook, Tesla, Amazon, Apple, IBM, Nvidia, Microsoft, Cisco and Yahoo) . The time range is 2012-06-01 to present, this date is chosen so that all target companies had their IPO (Initial Public Offering). For example, Facebook held its IPO on May 18th 2012 [8]. The total data frame contains 1245 rows and 60 columns, and they are all numerical values.

The following is all features of a stock during five trading days:

Date	Open	High	Low	Close	Volume	Adj Close
2017-05-01	68.680000	69.550003	68.500000	69.410004	31789300	69.410004
2017-05-02	69.709999	69.709999	69.129997	69.300003	23519500	69.300003
2017-05-03	69.379997	69.379997	68.709999	69.080002	28751500	69.080002
2017-05-04	69.029999	69.080002	68.639999	68.809998	21502600	68.809998
2017-05-05	68.900002	69.029999	68.489998	69.000000	18882800	69.000000

Here, ‘Open’ means the price of the stock when market opens, ‘High’ means

the highest stock price during the trading day, 'Low' means the lowest stock price during the trading day, 'Close' means the price of the stock when market closes, 'Volume' means the number of shares traded on that trading day, 'Adj Close' is the adjusted close price, adjusted for dividends and company events.

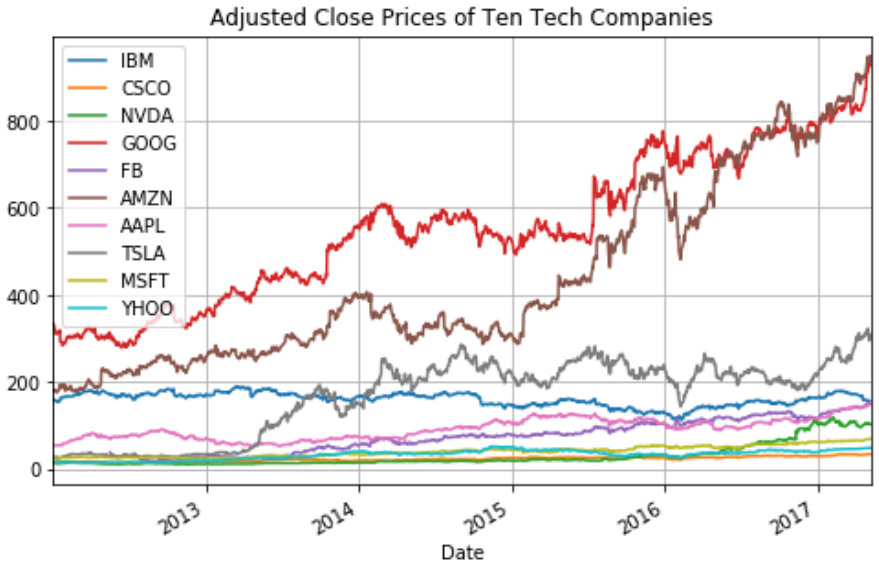
A statistical summary of the Adjusted Close price all then stocks of looks like the following:

	GOOG	FB	AMZN	AAPL	TSLA	MSFT	YHOO	IBM	CSCO	NVDA
count	1245.000000	1245.000000	1245.000000	1245.000000	1245.000000	1245.000000	1245.000000	1245.000000	1245.000000	1245.000
mean	575.463687	75.648875	456.836716	93.537599	178.769599	41.719142	34.372145	159.424751	23.678791	31.02861
std	154.772112	37.233110	206.595609	23.537848	80.379304	12.154704	9.537858	15.583139	4.960217	27.67531
min	279.246220	17.730000	208.220001	51.132658	26.100000	23.463078	14.650000	112.683210	13.014102	10.57822
25%	444.666094	41.279999	295.859985	72.328045	138.949997	31.041564	27.969999	148.317687	20.044570	14.14839
50%	558.172777	76.290001	370.559998	94.299041	206.270004	41.599548	36.169998	162.632789	23.357459	19.10176
75%	717.000000	107.910004	626.549988	111.201059	234.240005	50.913885	41.410000	171.147216	27.061911	32.39728
max	934.299988	152.779999	961.349976	156.100006	325.220001	69.410004	52.369999	190.226456	34.389999	127.8899

As we can see, no stocks has missing value as they all have 1245 counts, which is the same count before dropping null values. There aren't any abnormalities of the data, because of its public, timely and well-recorded nature. Outliers, if any, usually reflect significant market trends or major company events Google stock has the highest mean and second highest standard deviation, Amazon stock has the highest standard deviation and second highest standard deviation, which means these stocks has been going through rapid growth. While Cisco stock has the lowest standard deviation, reflecting stable stock price over the past few years.

Exploratory Visualization

The following graph shows the original Adjusted Close price of ten tech stocks(Google, Facebook, Tesla, Amazon, Apple, IBM, Nvidia, Microsoft, Cisco and Yahoo). over the past five years:



As we can see from the graph, Google and Amazon indeed has been going through exceptionally fast growth starting 2015, which is consistent with the conclusion from the statistical summary above. We can also learn that stock prices have different range, some kind of normalization will be needed in order to implement prediction algorithm.

Algorithms and Techniques

Given the input stock of interest, start day and end day, historical stock prices data will be retrieved through Yahoo Finance API. Two main algorithms will be considered: Linear Regression and ARIMA (Autoregressive Integrated Moving Average model). Because they are widely used, they are more interpretable compared to other methods, and their relative simplicity also makes it more appealing for statistical arbitrage. Other possible techniques include Random Forest Regressor, SVM Regressor and Neural Network, these are all valid, widely used models, but they are not as interpretable. For the purpose of interpretability, they are not discussed in this project.

Linear Regression

Linear regression has been used extensively in practical applications. Its goal is to model the relationship between a scalar dependent variable y and a vector of independent variables, the common method is to fit a line into the data point such that square error between line and data is minimized. One of its main benefits is its interpretability, by looking at the coefficients in the model, we can tell which independent variable has a big or small impact on the outcome, and which independent variable is positively correlated or negatively correlated with the

dependent variable.

R^2(coefficient of determination) will be used to evaluate the prediction power of model. R^2 ranges from 0 to 1, the closer the value is to 1, the better the model fits the data.

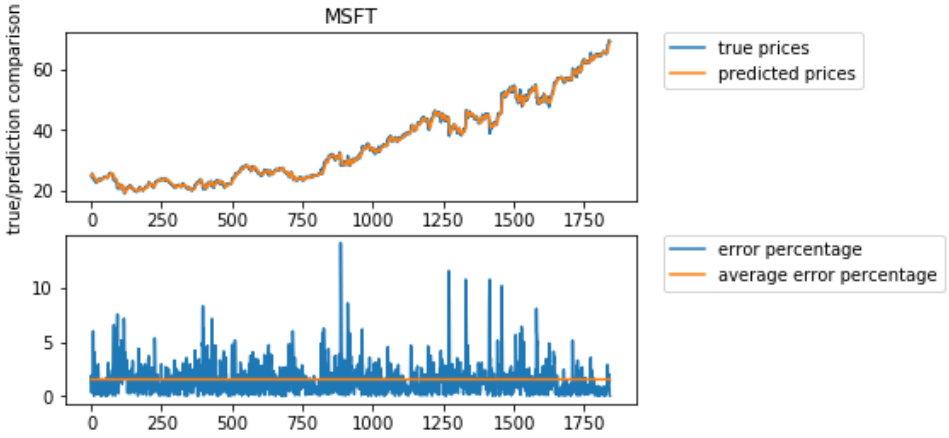
ARIMA Model

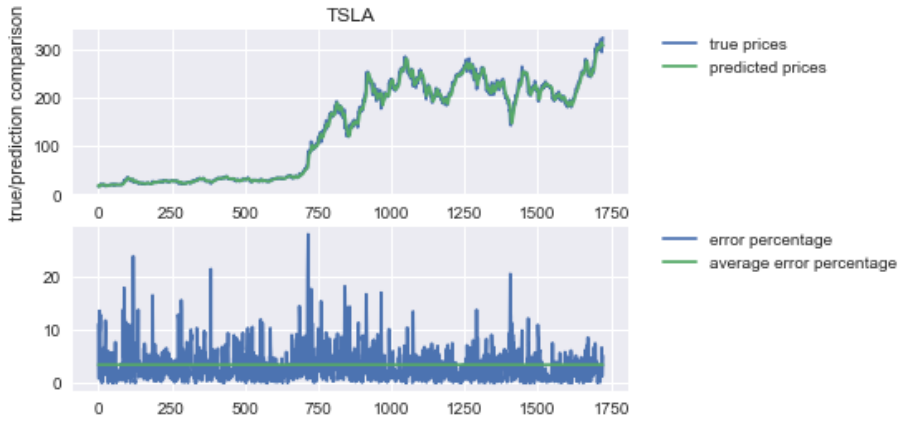
Time series is widely used to understand trend in stock market. ARIMA model can help better understand historic data and predict future points in the series. The idea of ARIMA model is that our variable of interest is regressed on its own lagged values and regression error is a linear combination of past error terms [5]. The parameters are (p, d, q), which are respectively autoregressive order, degree of differencing and moving-average order.

Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) are used for tuning parameters and model selection, the smaller AIC and BIC are, the better the model captures the trend of data.

Benchmark

The benchmark model is the rolling mean of the past five days. That is, we predict the adjusted close price to be the rolling mean of the previous 5 days. It is a naive predictor because we don't construct more insightful features, and we assume that stock price will be similar to the 5 closest data points. Results of the benchmark model can be visualized below, as we can see, the predicted prices are very close to true prices, although there are a couple times where the absolute error of prediction is more than 10%. The following graphs demonstrate the benchmark model performances on Microsoft stocks and Tesla Stocks, where 'average error percentage' is the average prediction error percentage, that is, the average of |y_true-y_pred|/y_true.

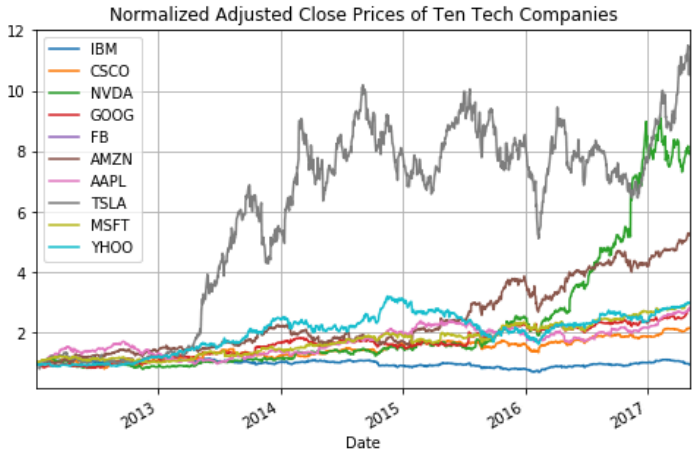




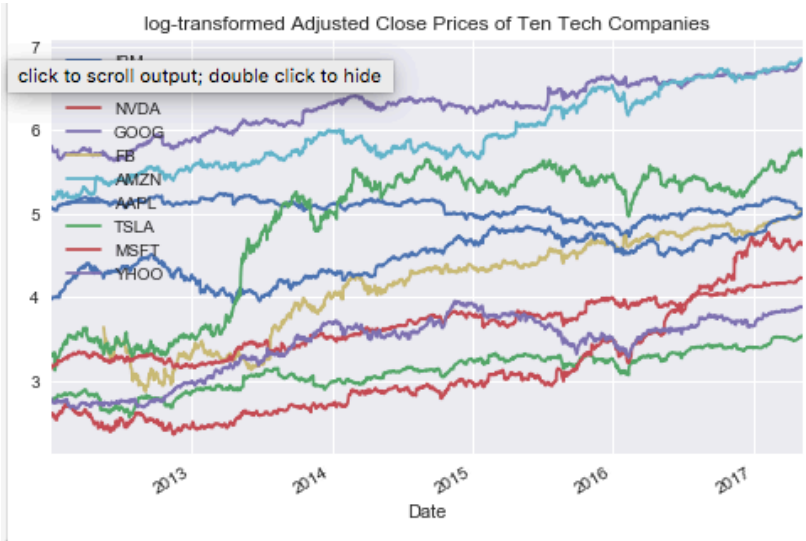
III. Methodology

Data Preprocessing

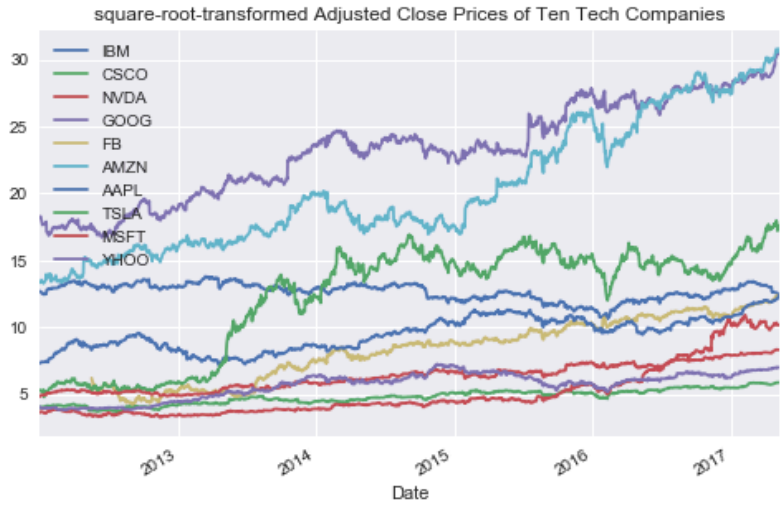
For stock prices datasets, no heavy preprocessing is needed, they are very public and well-documented data set as demonstrated in Data Exploration. However, there are two things that need to be considered: First, the possibility of missing data as the stock might not be trading on a specific day, here I used the technique forward fill, which propagates last valid observation forward to next valid backfill. Another common technique for time series missing data is backward fill, which uses next valid observation to fill gap. I chose forward fill because it aligns better with the goal of the project, which is to predicts the future with historical data. Second issue here is normalization, since different stock prices vary, for example, some stock can be as high as 800 per share, some can be 50 per share, even if they follow the same market trend, so normalization is necessary here. Each stock price will be divided by the price of training start day, so we can make sure that all stock prices start with 1 for training.
Below is a graph of normalized data:



A normalized plot gave us more information of a company's stock growth. We can see that Apple, Facebook, Google, Cisco and Microsoft stocks have very similar growth rate, while Tesla has the biggest jump and appears to be the most volatile, as it is a new industry. Amazon stock values experience a hike since the middle of 2015, which sets its growth rate apart from the other companies. Nvidia also experiences big growth starting 2016.
A common method to transform the data for time series forecasting is log transformation, it has the effect of stabilizing the time series when it is volatile. Below is a graph of log-transformed data:



Another common method to transform the data for time series forecasting is square-root-transformation, it also has the effect of stabilizing the time series. Below is a graph of square-root -transformed data:



Comparing the above two graphs we can see that log transformation does a better job stabilizing our data overall, so it will be used in model fitting.

Implementation

To implement linear regression model, we need to first select features and measure the model with R^2 and mean square error. Besides the original features ('Open', 'High', 'Volume', etc.), extra features will be constructed. Multiple global industrial indices, such as SP500, Dow Jones and Nikke, will be considered. To avoid the issue of multicollinearity, we don't use variables that are highly correlated with each other, so I will select variables by looking at their correlations with each other. Variables with high correlation with others will be removed. As we can see from the following correlation matrix, SP500, Dow Jones and Nikke are all highly correlated with each other. So the model will use SP500 and Ten-year bond only. It makes economic sense that SP500 AND Ten-year bond have negative correlation, as higher interest rate tends to dampen investment activities.

Correlation Table:

	SP500	DowJones	ten_yr_bond	nikke
SP500	1.000000	0.993728	-0.389822	0.935202
DowJones	0.993728	1.000000	-0.393801	0.918477
ten_yr_bond	-0.389822	-0.393801	1.000000	-0.199054
nikke	0.935202	0.918477	-0.199054	1.000000

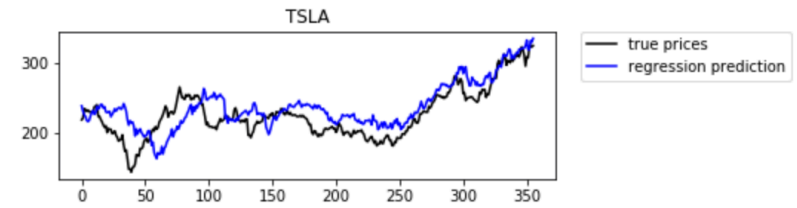
Besides macroeconomic features, features that capture the inherent volatility of the stocks should be included as well. These features include daily return and daily fluctuation, stock beta (data from Yahoo Finance, reflect how volatile a stock is

compared to market as a whole)[10] and a binary indicator of whether the stock price goes up or not. Daily return = (adjusted close price of day i – adjusted close of day (i-1))/ adjusted close of day (i-1).
Daily fluctuation = 100*(daily_high – daily_low)/daily_low.
The final feature vector is ['Open', 'Volume', 'go_up', 'daily_return', 'daily_fluctuation', 'beta','SP500', 'ten_yr_bond']

Some visual results of linear regression are displayed in the following:


```
('YHOO', 'mean square error:', 15.18314092958856, 'R^2:', 0.46686540120013764)
('coefs:', array([  1.55276317e+02,  -1.48928050e+00,   6.01810393e+00,
       -1.76120347e+02,   4.23262153e+00,  -3.46870751e-28,
        1.46353729e-01,   1.63271555e+01]))
```


```
('FB', 'mean square error:', 78.90549122609096, 'R^2:', 0.508795689363849)
('coefs:', array([  4.81464085e+00,  -1.82222814e-01,   8.07582751e-02,
       -3.17903931e+00,   1.73266291e-01,   2.77555756e-14,
        7.91824014e-02,  -8.14271222e-01]))
```




```
('TSLA', 'mean square error:', 627.6376949112508, 'R^2:', 0.49665586519063376)
('coefs:', array([  3.70955113e+00,  -3.16505840e+00,  -6.06752810e-01,
       -2.95553179e+01,   2.69639002e-01,   2.30926389e-14,
        8.22136596e-01,  -3.19648098e+01]))
```

The coefficients suggest that 'beta' has little predictive power to stock price, while 'Open' has dominant predictive power. Interestly, Ten-year bond is negatively related with Facbook stock and Tesla stock while it is positively related with Yahoo stock.

One way to determine how well the regression model fits the data is R^2, which is the coefficient of determination, and it is ranged from 0 to1. 1 means the model fits the data perfectly, while 0 means the model doesn't explain any variance in the model. In python, an output of negative R^2 means the model doesn't fit the data.

The following table is the R^2 for our target stocks.

	GOOG	IBM	TSLA	AAPL	CSCO	FB	NVDA	YHOO	AMZN	MSFT
R^2	-9	0.59	0.50	-0.81	0.41	0.51	-0.19	0.47	-0.32	-0.25

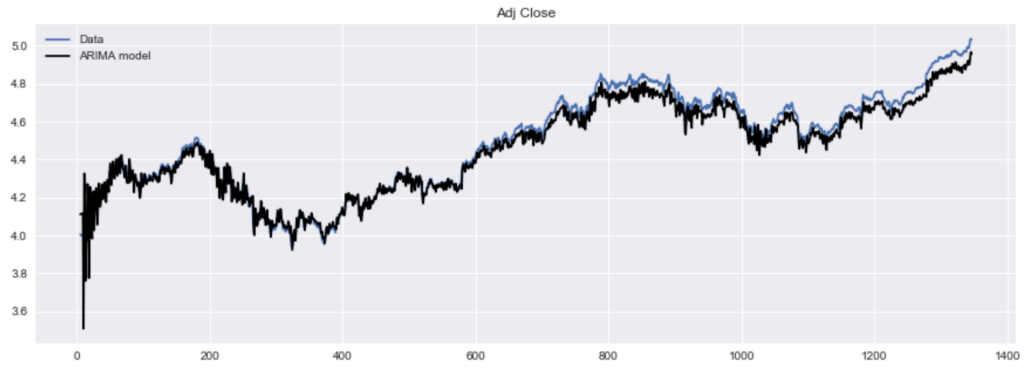
We see that the linear regression model does a moderate job modeling IBM, Tesla, Cisco, Facebook and Yahoo stock prices, but the model fits other stocks poorly. Since stock market follows a 'random walk', trying to model it long-term with linear regression is inherent fraudulent. Hence, I decided to go with ARIMA model, which shows more promise when the data shows evidence of non-stationarity[3]. I chose to set I to be 0 as log transformation, as a log transformation has been used to stabilize prices.

Refinement

To improve the ARIMA model, one way is to adjust two parameters: p and q, where p refers to the order of autoregressive model, q refers to moving-average model. To evaluate the parameters, we use AIC(Akaike Information Criterion) and BIC(Bayesian Information Criterion). The smaller the AIC and BIC values, the better [5].

Take Apple stock for example, when we set parameters (p, q) to (1,1), the AIC and BIC are -995 and -974 respectively. And the overall fit visualized is:

Increasing p and q tends to decrease AIC and BIC, but it adds model complexity by augmenting parameter space, and after a certain point, the model stops improving significantly. For example, when p and q are set to be 30, AIC and BIC are respectively -6010 and -5689, but the graph picks up too much noise in the model as visualized below:



IV. Results

Model Evaluation and Validation

The final parameters (p, d) is set to be (20,20), they are chosen after trying different parameters and choose by comparing AIC and BIC. The AIC and BIC for current models of the ten tech stocks are in the following table, as we see, they are all low values, signaling a well-fit model.

	GOOG	FB	AMZN	AAPL	TSLA	MSFT	YHOO	IBM	CSCO	NVDA
AIC	-5036	-3746	-4100	-5789	-3821	-5499	-2563	-7037	-3570	-3932
BIC	-4821	-3532	-3885	-5574	-3597	-5285	-2348	-6822	-3355	-3718

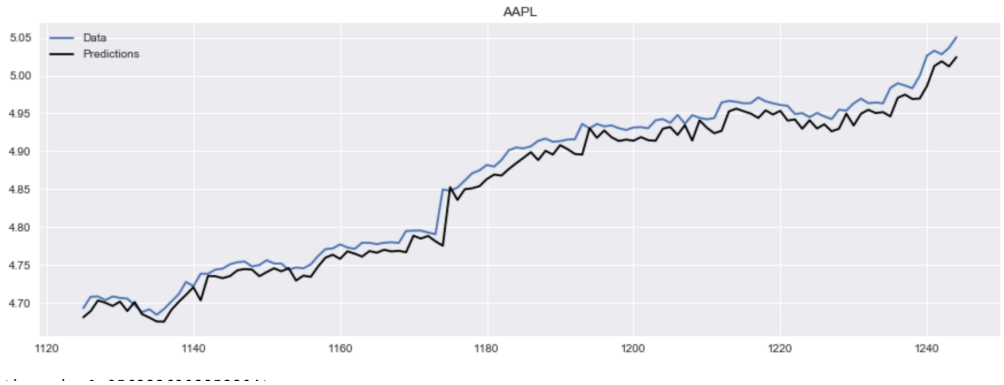
To check the robustness of the model, we use back testing, which is a special type of cross-validation applied to time series data. In this project, I remove the last 120 trading days in model training, and use the fitted model to predict those left out 120 days, then compare the prediction with real values.

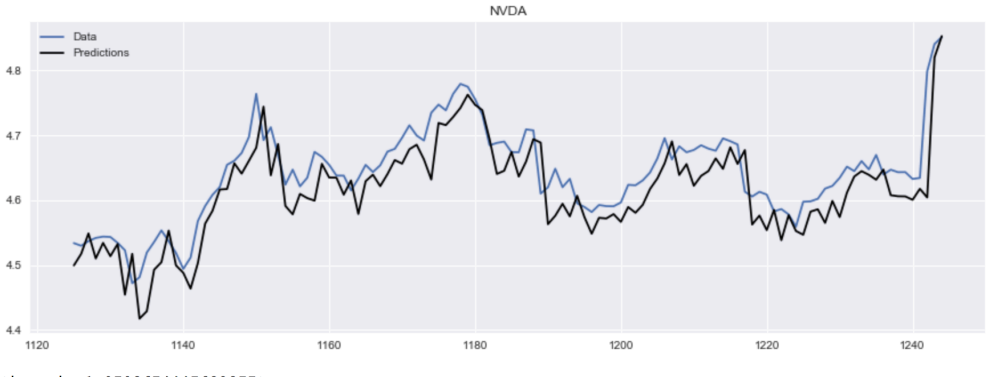
The following table is the 120-day back testing MSE:

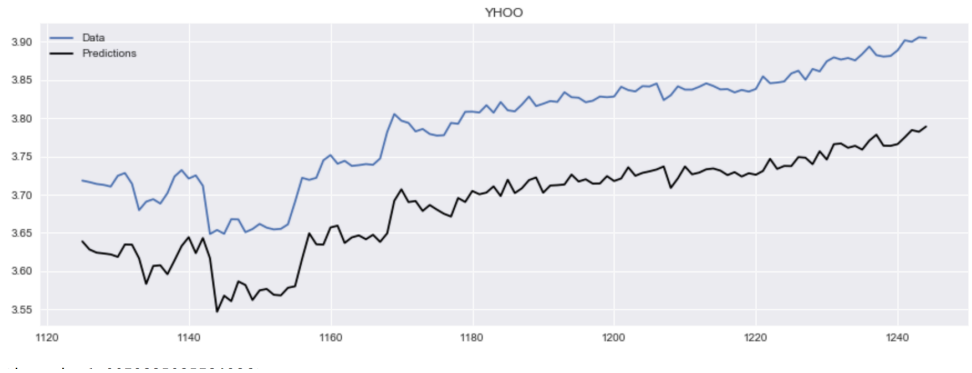
	GOOG	IBM	TSLA	AAPL	CSCO	FB	NVDA	YHOO	AMZN	MSFT
MSE	1.27	1.00	1.52	1.03	1.11	1.02	1.06	1.23	1.14	1.05

We see that MSE values are relatively stable for all ten stocks, they are all in the range of 1.0 to 1.27, such small MSE values suggest good fit of model.

The following graphs illustrate the fit of the 120-day back testing for Apple stock, Nvidia stock and Yahoo stock:







We can see that the data fits the past 120-day closely, especially for Apple and Nvidia stocks. Though there is a gap between prediction and data for Yahoo stock, the prediction still captures the movement of the stock, hence there exists possible remedy for the gap, but it is not discussed in this project.

Justification

The MSE of benchmark model:

	GOOG	IBM	TSLA	AAPL	CSCO	FB	NVDA	YHOO	AMZN	MSFT
MSE	155	7.52	61.9	4.51	0.215	4.4	3.29	0.83	155	0.779

The MSE of linear regression model:

	GOOG	IBM	TSLA	AAPL	CSCO	FB	NVDA	YHOO	AMZN	MSFT

MSE	28493.56	105.29	627.64	498.85	5.47	78.9	1076.40	15.18	15413.02	43.66

The MSE of ARIMA model:

	GOOG	IBM	TSLA	AAPL	CSCO	FB	NVDA	YHOO	AMZN	MSFT
MSE	1.29	1.002	1.15	1.06	1.06	1.08	1.06	1.21	1.07	1.04

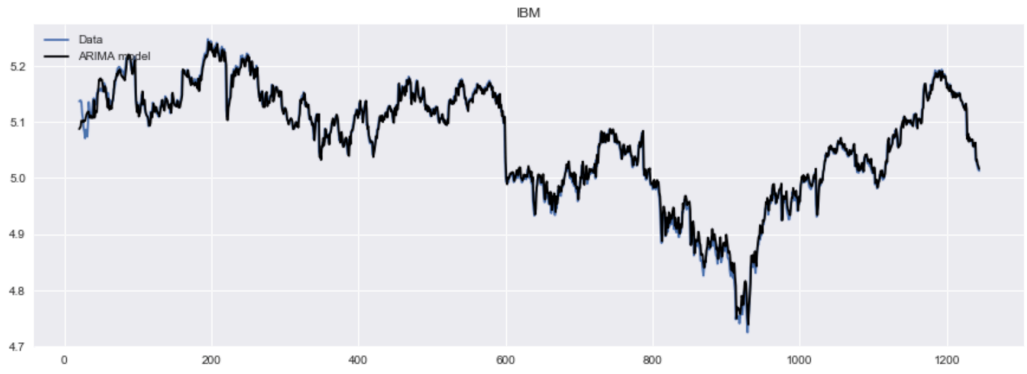
The MSE of the benchmark model fluctuate a lot: our highest growth stocks, Amazon and Google, have high MSE (both 155), while some relatively stable stocks (Microsoft and Cisco) have MSE less than 1. The MSE of linear regression model fluctuate even more, we can see that our highest growth stocks have MSE as high as 28493. While the stable stocks still have relatively high MSE, say, MSE of Cisco equals 5.4 and MSE of Microsoft is 43.66.

The MSE of ARIMA model is the most stable and the smallest of all three models, they are all in the range of 1 to 1.29. So ARIMA is indeed a better model for modeling financial time series.

V. Conclusion

Free-Form Visualization

The following graphs are the ARIMA model of IBM, Amazon and Apple stock prices. They are the overall fit of the ARIMA model of all historic prices from 2012-06-01. X-axis means the number of trading dates from 2012-06-01, while y-axis is the natural log of stock prices. We can see that the time series prediction overall captures the movement of stock prices.



Reflection

I started the project by collecting stock prices through 06-01-2012, so all target companies already had their IPO (Initial Public Offering). Then I tried a benchmark model, a linear regression model and ARIMA model to compare the performances. One difficult part of the project is that different stocks subject to different volatility. For stocks that are experiencing stellar growth, the model fails to predict the fast growth and tends to undervalue the stocks. For the purpose of devising a trading strategy, a more reasonable technique would be to predict whether the stock go up on a future date.

But overall, ARIMA model is promising in modeling finance time series data, as we can see from the lower Mean Squared Error values.

Improvement

There definitely exists a better solution, as sentiment analysis and news can be incorporated to improve the power of prediction. For example, the biotech industry would require specific knowledge in order to understand whether the tech employed by certain company is solid, and often time the industry's fate is controlled by the FDA (Food Drug Administration). For this industry, the safer bet to invest would be news mining and sentiment analysis [4]. For our target ten stocks, we can track news about quarterly corporate earnings, new products release and

news about merger and acquisition. These new are very important factors affecting stock prices. The combination of time series analysis and news/sentiment analysis calls for more complicated and sophisticated models.

Reference:

[1] https://www.bloomberg.com/news/articles/2017-03-27/hedge-fund-quants-close-in-on-designing-ultimate-trader-s-brain
[2] http://www.investopedia.com/terms/a/adjusted_closing_price.asp
[3] https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average
[4] http://www.cnbc.com/2017/05/10/a-moment-of-truth-for-biotech-stocks.html
[5] https://methodology.psu.edu/node/504
[6] http://www.investopedia.com/investing/how-interest-rates-affect-stock-market/
[7] https://en.wikipedia.org/wiki/Mean_squared_error
[8] https://en.wikipedia.org/wiki/Initial_public_offering_of_Facebook
[9] https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average
[10] http://www.investopedia.com/terms/b/beta.asp