Machine Learning for Stock Price

The Capstone Project for Udacity Machine Learning Engineer Nanodegree

Kailun Wang

Contents

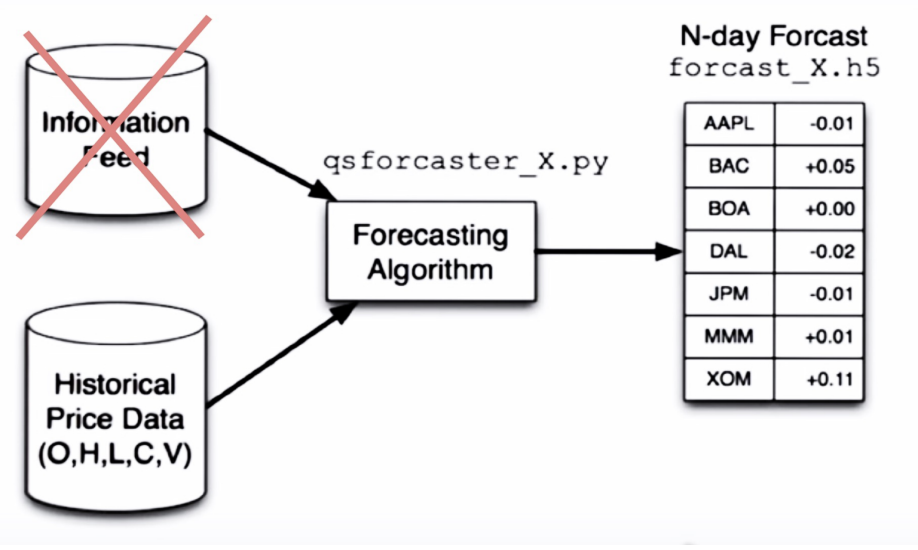
1. Introduction

1.1 Background information:

The fluctuations in stocks markets are unpredictable from the birth of the stock trading more than 200 years ago. However, the incrementally advancing in computers, algorithms and data recording makes predictions in financial markets gradually become possible. Financial sector is one of the earliest domains which aggressively attempt to adopt new technologies with the incentive of higher profits. Since the financial data on stocks, commodities or interest rates are abundant, labeled and well-documented, so with simple preprocessing on those data, they will be qualified fuel to drive artificial intelligence that serves as a necessarily complementary part of humans in the financial sector. Financial service companies like Goldman Sachs, Morgan Stanley have been using machine learning for years to better observe markets and organize inner resources. Hedge funds like Renaissance Technology, Citadel LLC or Bridgewater Associates have been intensively hiring software engineers, mathematicians and data scientists to improve their trading algorithms. In contrast to traditional floor trading or phone trading, this is a significant leap.

1.2 Problem statement

The goal of this project is to build a supervised regression model that takes historical stock trading data from the stock exchanges, trains through regression algorithms and outputs predicted next day stock prices for given queries of a certain company with a certain date. For the reason of simplicity, the information feed is not included in this project.



Individually speaking, if common people plan to invest in stock market in order to appreciate their cash assets, they would need an additional technical analysis tool for trading strategies. A machine learning algorithms backed stock price predictor will achieve this goal.

1.3 Project Origins:

Personally speaking, this project is a combination of my past investing-related dots and a expansion of new financial technical analysis I have just learned. I like spending time on things value more tomorrow than today and investing is a good field to practice it. I followed insights of value investing masters Warren Buffet and Charlie Munger and purchased stocks, such as Amazon, Nvidia, Facebook, that I think that they truly have brought irreplaceable values to their customers or shareholders. And I was rewarded fairly. Yet, I am still curious if machine can be the assistant on investing. Later on, I learned two legendary founders of the world top hedge funds, James Simon and Ray Dalio. Unlike Buffet's value investing, those hedge fund founders work closely with machine learning and focus intensively on technical analysis like momentum of stocks, volume of trading, trends of stock prices, etc.

Especially thank these resources, since with these academic and non-academic supports, this project has become real:

- https://www.omscs.gatech.edu/cs-7646-machine-learning-trading
- https://www.investopedia.com/articles/07/montecarlo.asp
  https://dspace.mit.edu/bitstream/handle/1721.1/105982/965785890-MIT.pdf
- https://www.quora.com/Which-are-the-best-models-for-stock-analysis-How-should-a-beginner-start

1.4 Metrics

Since this project is to solve regression problems, regression metrics r2 score and mean squared error are two useful candidates. (reference: https://en.wikipedia.org/wiki/Coefficient_of_determination)

R2 score, also known as coefficient determination, can be used to measure the difference between actual prices and predicted prices and widely used in regression problems.

$$R^2 \equiv 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

SSres is sum of squared errors of prediction, which equals the sum of all squared difference of the actual stock prices (yi) and the predicted stock prices (fi or y-hat).

SStot is the squared differences of each observation from the overall mean, which equals the sum of all squared difference of the actual stock prices (yi) and the average of the stock prices (y-bar)

$$SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2, \quad SS_{\text{res}} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2$$

The Reason of using r2 score as one of metrics is that it can indicate the proportion of variance in the dependent variables that is predictable from independent variables. Also, typically the trend of stock price can be regarded as linear lines, and the r2 score can give some information about the goodness of fit of a model, which statistically is how well is the regression line generated by the regression models approximates the real data points. If r2 score equals 1, then it perfectly fits, which means that the depend variable can be predicted from the independent variable without error. If r2 score equals 0, then it means that the independent variable cannot be predicted from dependent variables. If r2 score are negative, then those regression models are statistically worse. Therefore, r2 score is a useful metrics in this project given the dataset.

Mean squared error(MSE) is also a practical metric in regression problems. Unlike R2 score which has to predict multiple points and can vary statistically significant depending on the number of data points, MSE works effectively for evaluating the performance of the model on any number of data points.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} (\hat{Y}_i - Y_i)^2$$

The reason of using MSE rather than Mean Absolute Error(MAE) is that MSE considers both the variance of the prediction and its bias. MSE will give information about the spread of the predicted data points, because it penalizes outliers more heavily than what MAE does due the square property. Since the data points in this project are most likely linear, so the spread of the predicted data points around the linear line can be better represented by MSE. Therefore, MSE is another useful metric in this project given the dataset.

Reference:

- https://www.quora.com/What-is-the-difference-between-squared-error-and-absolute-error
- https://en.wikipedia.org/wiki/Coefficient_of_determination
- https://en.wikipedia.org/wiki/Mean_squared_error

1.5 Related data sets:

The candidate dataset for this project are the following stocks downloaded from Yahoo! Finance:

- S&P500 (SPY.csv)
- Apple (APPLE.csv)
- Microsoft (MSFT.csv)
- Amazon (AMZN.csv)
- Facebook (FB.csv)
- JP Morgan (JPM)
- Berkshire Hathaway (BRK-B.csv)
- Johnson & Johnson (JNJ.csv)
- Alphabet Inc C (GOOG.csv)
- Alphabet Inc A (GOOGL.csv)
- Exxon Mobil Corp (XOM.csv)

Explanation on SPY:

"SPDR® S&P 500 is an ETF (or exchange-traded fund) that tracks the S&P 500 index, and is itself listed on NYSE under the ticker symbol 'SPY'." Excerpt from

"S&P 500 is a stock market index based on 500 large American companies listed on the NYSE or NASDAQ. Think of it as a weighted mean of the stock prices of the companies, where the number of shares are used as weights (with some adjustment for any events that may affect apparent stock value, such as splits)." Excerpt from

SPY will be used as a reference for two reasons: first, stock in exchanges will open for trading when SPY opens for trading; second, the price of SPY can be regarded as the average market price, and it is a good reference for other stocks to compare with.

Reference:

- https://finance.yahoo.com/

- https://en.wikipedia.org/wiki/SPDR_S%26P_500_Trust_ETF
- https://en.wikipedia.org/wiki/S%26P_500_Index

2. Analysis

2.1 Data Exploration
It is necessary to choose one stock to develop deeper explorations, and the index of the market portfolio “SPY” is a comprehensive option.
This is the first five rows of the dataset SPY.

	Date	Open	High	Low	Close	Adj Close	Volume
0	1993-01-29	43.9687	43.9687	43.7500	43.9375	27.466780	1003200
1	1993-02-01	43.9687	44.2500	43.9687	44.2500	27.662146	480500
2	1993-02-02	44.2187	44.3750	44.1250	44.3437	27.720715	201300
3	1993-02-03	44.4062	44.8437	44.3750	44.8125	28.013771	529400
4	1993-02-04	44.9687	45.0937	44.4687	45.0000	28.130983	531500

This is the last five rows of the dataset.

	Date	Open	High	Low	Close	Adj Close	Volume
6296	2018-01-30	282.600006	284.739990	281.220001	281.760010	281.760010	131796400
6297	2018-01-31	282.730011	283.299988	280.679993	281.899994	281.899994	108364800
6298	2018-02-01	281.070007	283.059998	280.679993	281.579987	281.579987	90102500
6299	2018-02-02	280.079987	280.230011	275.410004	275.450012	275.450012	173174800
6300	2018-02-05	273.450012	275.850006	263.309998	263.929993	263.929993	285623500

These are the statistics of the dataset:

	Open	High	Low	Close	Adj Close	Volume
count	6301.000000	6301.000000	6301.000000	6301.000000	6301.000000	6.301000e+03
mean	125.935800	126.675043	125.107522	125.931212	104.406346	8.406350e+07
std	50.526929	50.648121	50.383799	50.530478	54.594421	1.005090e+08
min	43.343700	43.531200	42.812500	43.406200	27.134647	5.200000e+03
25%	94.500000	95.375000	93.500000	94.375000	70.730179	6.280900e+06
50%	121.500000	122.320000	120.779999	121.519997	94.472298	5.406440e+07
75%	145.440002	146.149994	144.149994	145.169998	119.688576	1.248421e+08
max	285.929993	286.579987	284.500000	286.579987	286.579987	8.710263e+08

According to the explorations, features in this dataset are:

- 'Date': date of trading in format of YYYY-MM-DD
- 'Open': the open price of the stock in the date of trading
- 'High': the highest price of the stock in the date of trading
- 'Low': the lowest price of the stock in the date of trading
- 'Close': the close price of the stock in the date of trading
- 'Adj Close': the close price of the stock after adjusted inflation and dividends in the date of trading
- 'Volume'": the transaction of the stock occurred in the date of trading

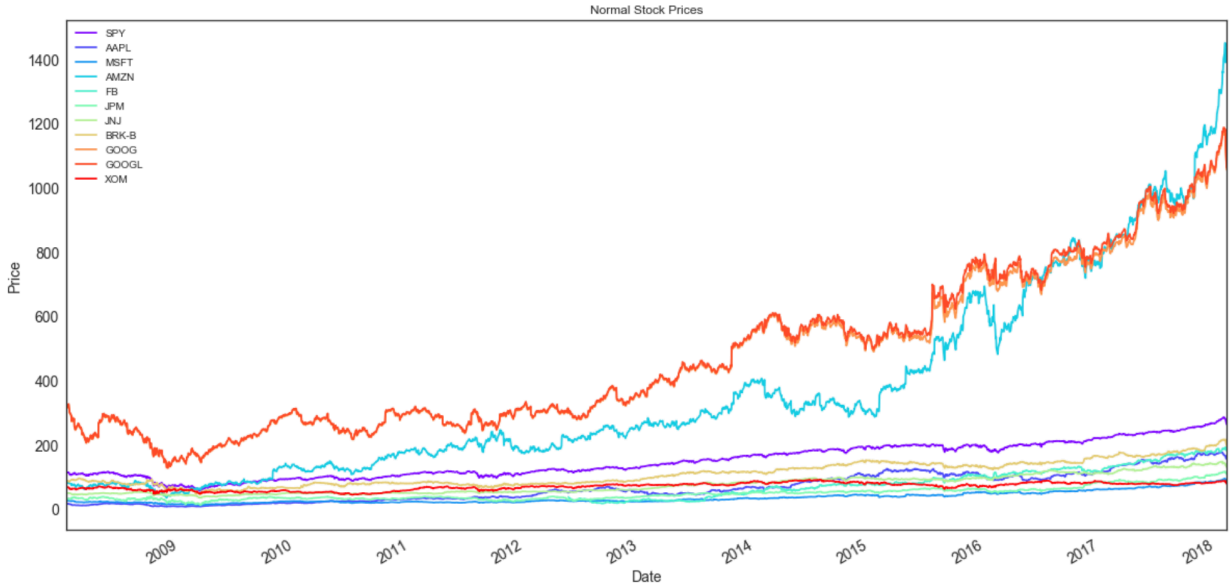
To build and train machine learning models, feature labels from the dataset are 'Date', 'Open', 'High', 'Low', 'Close' and 'Volume', and target label is numerical value 'Adj Close'.

For this project, the target date range is from Jan 5th 2009 to Feb 1st 2018. The reason that the date starts from the beginning of 2009 instead of 2008 or 1998 is because of economic cycle. The global economic crisis in 2008 is the end of the previous economic cycle, the stock prices were under drastic fluctuations. Those periods of data were approximately outliers and might mislead the regression models if those data were includedAlso, most of dataset downloaded end at early 2018, and there is no sign of severe fluctuation in economy. Therefore, it is a better and simpler way to use data start from 2009 to 2018.

2.2 data visualization

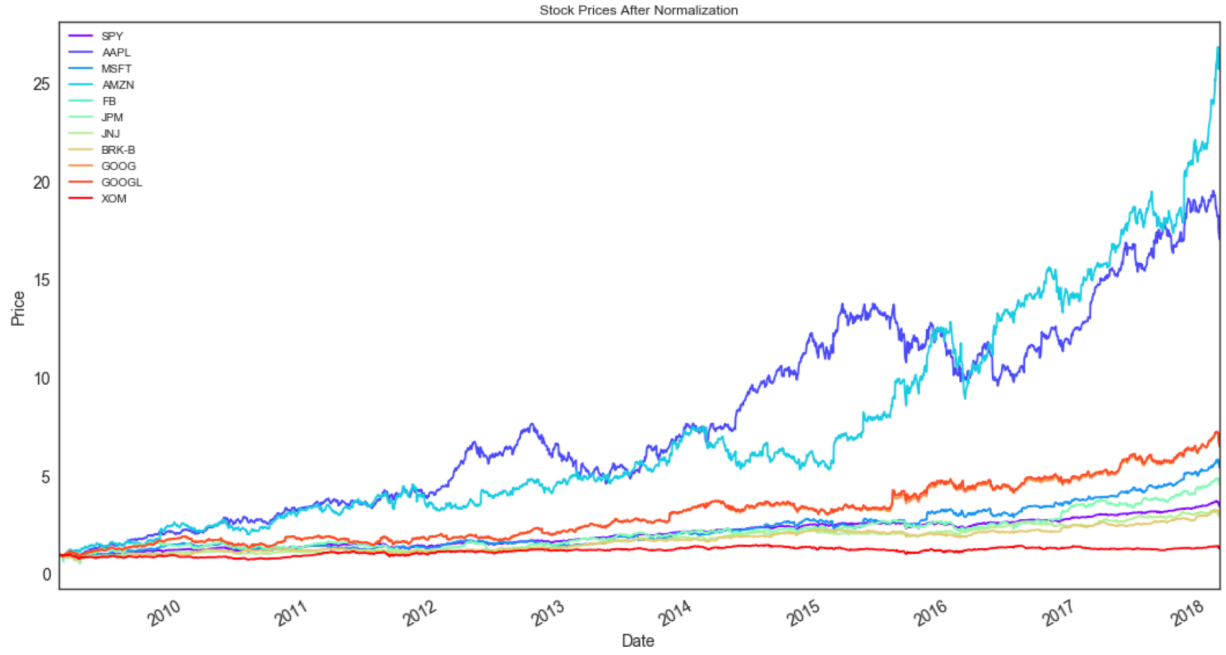
According to the Morningstar.com, The top ten largest weighted components of SPY are AAPL(3.79%), MSFT(3.04%), AMZN(2.48%), FB(1.81%), JPM(1.70%), BRK-B(1.68%), JNJ(1.53%), GOOG(1.41%), GOOGL(1.40%), XOM(1.39%) These components can largely affect the trend of the stock price of SPY. Therefore, it is important to perform more investigations on their relationships with SPY.

Here is the stock price trend of SPY and other top ten weighted stocks over last ten years.

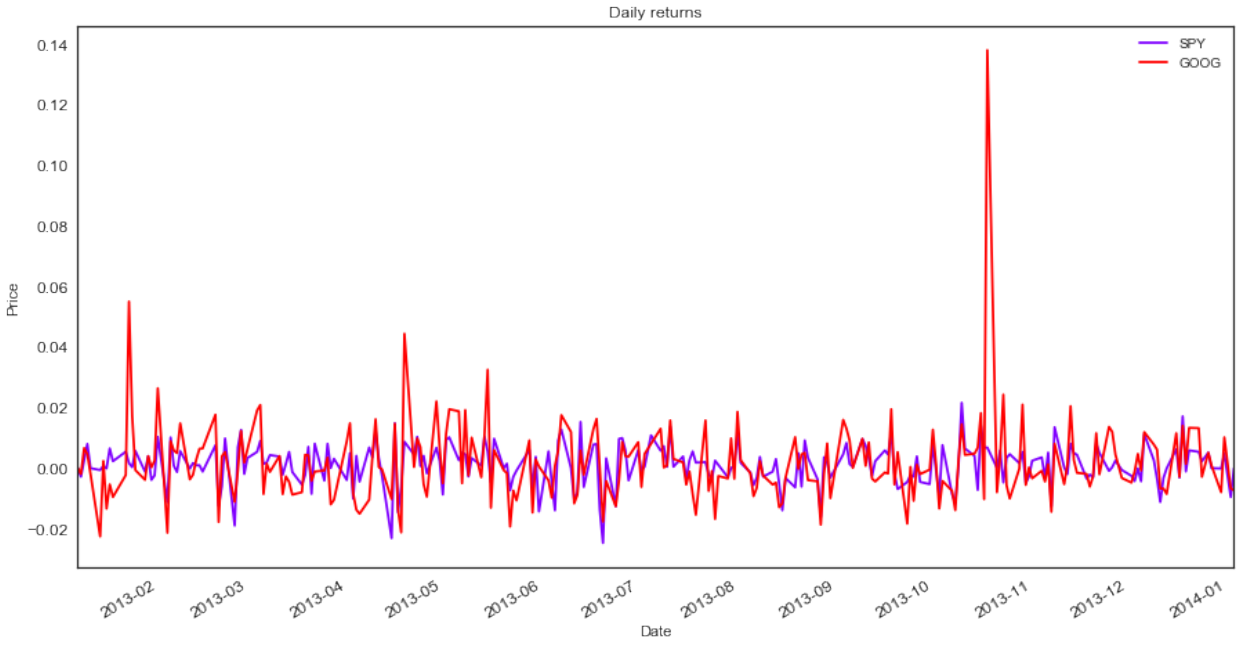


Comparing the stock prices among companies cannot provide enough information to judge the growth in terms of stock price. For example, in the graph above, Google stands at 2nd place with a greatly high price and Apple is just among the average in terms of stock price. However, in the graph below, Apple becomes the 2nd and Google becomes the 3rd in terms of growth in stock price.

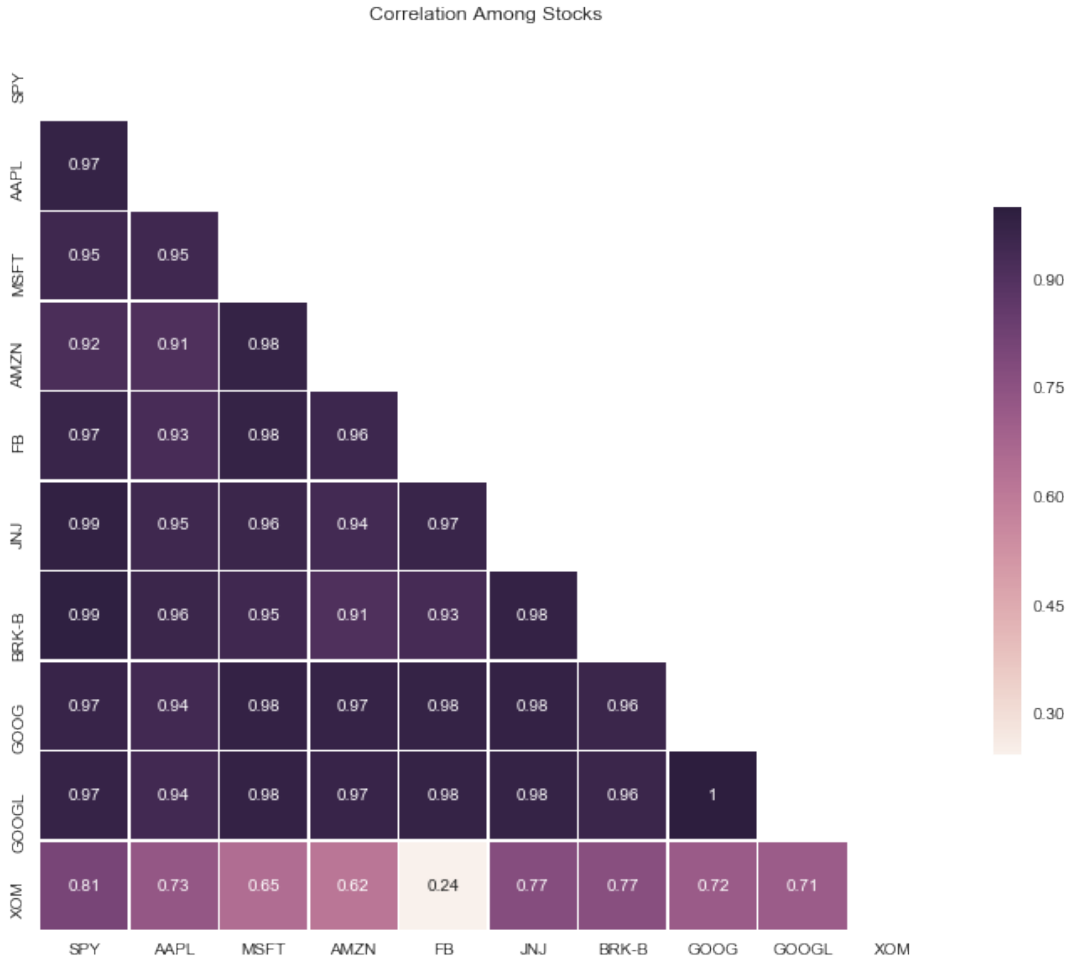
This is the stock price trend of those stocks over last ten years after normalization.



Regardless the annual growth, this is comparison of daily returns between SPY and one of the top ten weighted stocks GOOG over year 2014. The volatility or the spread of GOOG is notably greater than those of SPY.



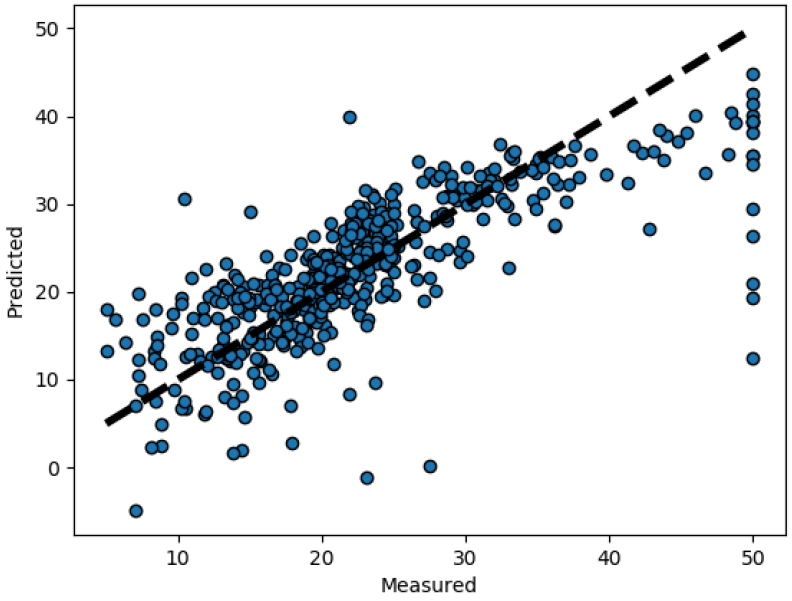
Since the SPY and its inside top ten weighted companies can influence each other, this graph shows how each of them correlated with each other. The darker a cube color is, a stronger correlation between two companies. From the graph, only the last row shows lighter. It is reasonable, because the top nice companies are from sectors of IT, the Internet, consumer electronics, finance and investment and they are closely connected. On the other hand, XOM, Exxon Mobile, is a traditional energy company, so it does not have strong common interest with the other companies, especially Facebook.



2.3 algorithms and methodology

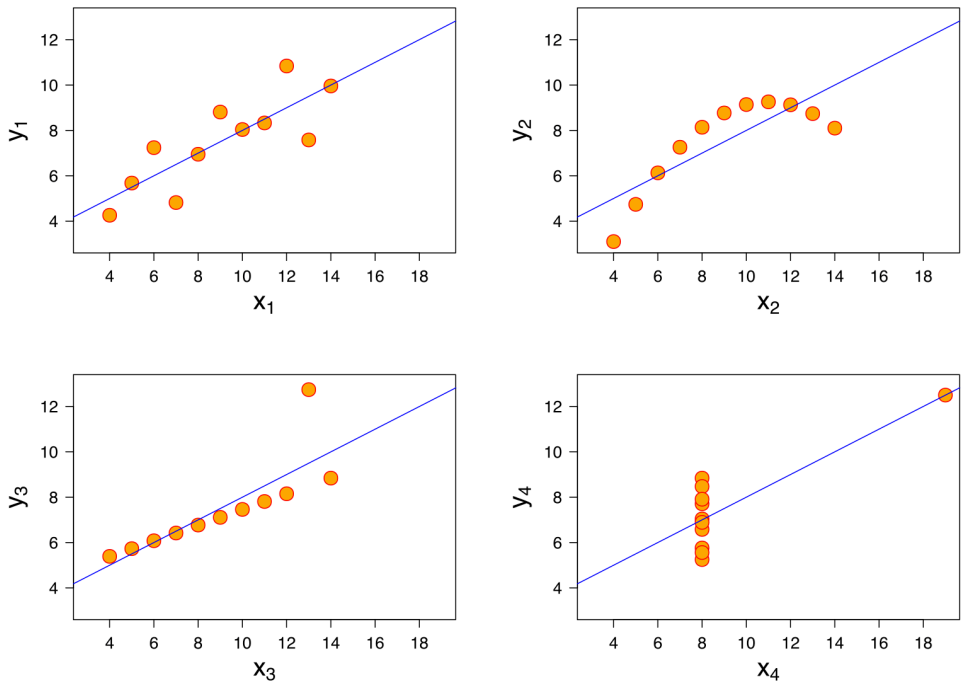
Linear Regression

Linear regression algorithm is a linear approach for modeling for modeling the relationship between a scalar depended variable y and one or more explanatory variables (independent variables) x. In this project, the stock price fluctuation is a typical linear regression problem since the overall movement likes a line and linear regression fits the best line to the given data.



The line or the model is the algorithm represented by the equation: y = ax + b. The algorithm training processing is to find a & b by massive pairs of (Xi, Yi). Yi in this project is 'Adj Close' price, Xi here is 'Open', 'High', 'Low', 'Close', 'Volume'. After finished the training, a model with two estimated a & b can be generated. This model uses testing test to get a prediction set. Finally use metrics to compare prediction set and actual set.

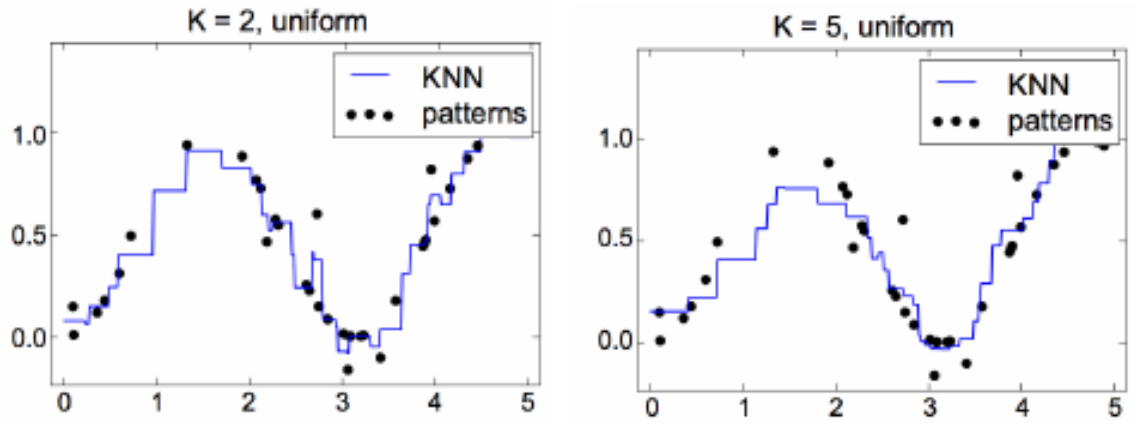
Admittedly, linear regression algorithm limitations. For example, one of major problem is showing below:



The four graphs have same linear regression lines: identical means, standard deviations and correlations. However, the distributions of the data points are fundamentally different. This damages the effectiveness of linear model. Therefore, to compare with, other regression algorithms will be used and tested in the next.

KNN

K-Nearest Neighbors(KNN) algorithm is a non-parametric method that can used for both regression and classification problems. In regression cases, he input consists of the k closest training examples (or aka 'neighbors') in the feature space. The output is the property value for the object and this value is the average values of those k closest training examples.

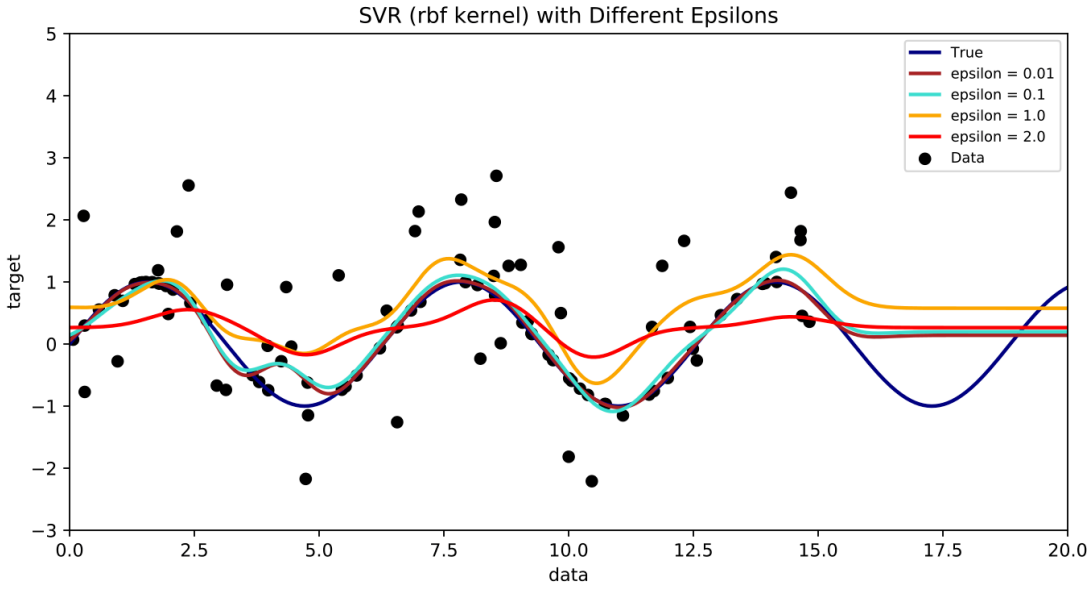


The data points of stock prices flow like points in the graphs above. Since KNN is an instance-based learning, KNN will teach the model to exact surrounding information and make predictions. Therefore, it is beneficial to use KNN.

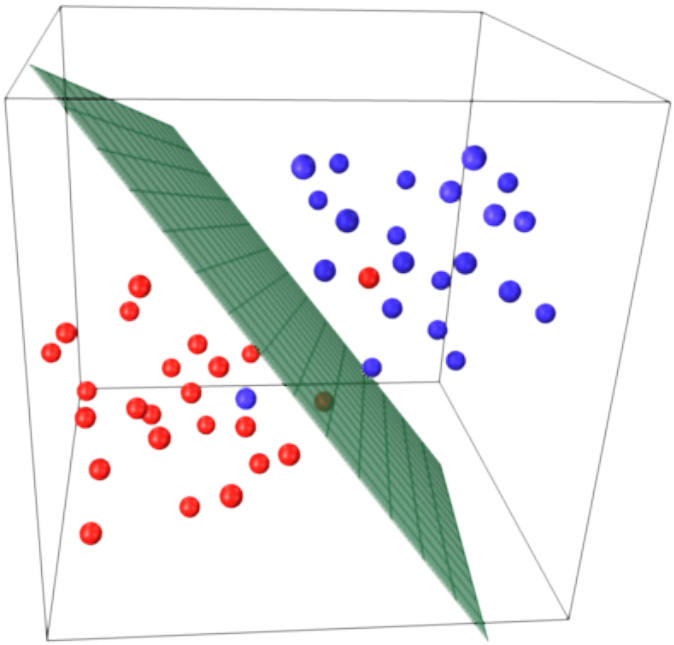
However, a flaw of KNN is that it is sensitive to the local structure or noise of the data. Moreover, KNN limits imagination of predictions, which means that KNN cannot predict a price greater than $500 if all k examples are less than $500.

Support Vector Machine – Support Vector Regression

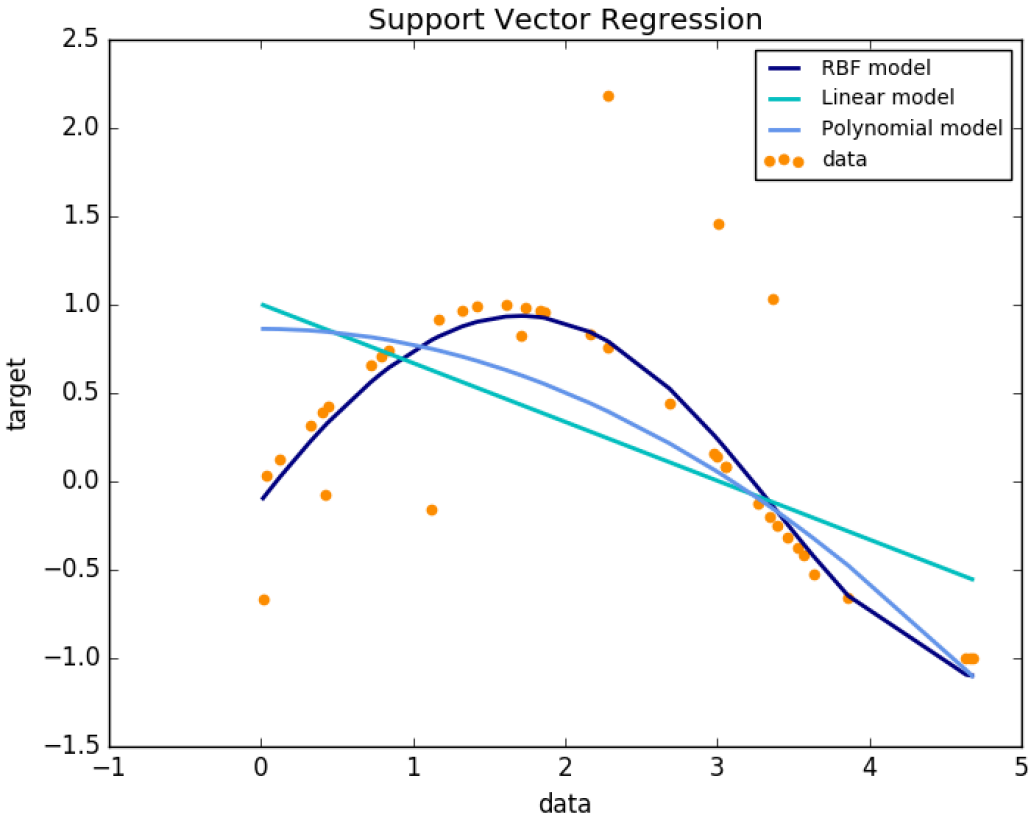
Support Vector Machine (SVM) is a powerful supervised learning models with associated learning algorithms that analyze the data used for classification and regression analysis. It is able to perform linear or non-linear classification, regression and outlier detection. A version of SVM for regression is called Support Vector Regression (SVR). This model is developed by SVM classification depends on only on a subset of data, because the cost function for building the model does not care about training points that lie beyond the margin(distance between two classes)



SVM can construct one more multiple hyperplanes in a high- or infinite- dimension space. The hyperplane makes a good separation on points that has the largest distance to the nearest training data point of any class.



Also, Support Vector Regression(SVR) provides useful tune options. For example, the kernel tricks. They are used to find a line that always fit the data points. Kernel tricks are Linear, Polynomial, RBF, Sigmoid, etc. The graph below shows how lines generated by different kernels fit to the data points. Due to the different spread of the data in this project, tune the kernel tricks along with other hyperparameters would be useful to find lines that make better predictions.



Reference:

- https://en.wikipedia.org/wiki/Linear_regression
- https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
- https://en.wikipedia.org/wiki/Support_vector_machine
- http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

2.4 Benchmark

The benchmark for linear regression, KNN and SVR models is an accuracy score of 50%. If it is greater than 50%, it means that the model is already better than gaming and will have a positive expected value in long term.

The benchmark model is linear regression model here, because it is simple and comparable.

3. Methodology

3.1 data preprocessing

Before the dataset can be used to train the regression models, several preprocessing steps need to be done.

a) Data are re-indexed by date not row number
Data frames generated by Pandas orderly start by 0, and the data downloaded from Yahoo! Finance are in the format.

	Date	Open	High	Low	Close	Adj Close	Volume
0	12/12/80	0.513393	0.515625	0.513393	0.513393	0.023357	117258400.0
1	12/15/80	0.488839	0.488839	0.486607	0.486607	0.022139	43971200.0
2	12/16/80	0.453125	0.453125	0.450893	0.450893	0.020514	26432000.0
3	12/17/80	0.462054	0.464286	0.462054	0.462054	0.021022	21610400.0
4	12/18/80	0.475446	0.477679	0.475446	0.475446	0.021631	18362400.0

For the purpose of slicing specific date range, the data need to be re-indexed. The built-in function 'pandas.read_csv' has a proper parameter 'index_col=' to manually change the way of indexing. After set "index_col='date'", the data now index based on date.

	Open	High	Low	Close	Adj Close	Volume
Date						
12/12/80	0.513393	0.515625	0.513393	0.513393	0.023357	117258400.0
12/15/80	0.488839	0.488839	0.486607	0.486607	0.022139	43971200.0
12/16/80	0.453125	0.453125	0.450893	0.450893	0.020514	26432000.0
12/17/80	0.462054	0.464286	0.462054	0.462054	0.021022	21610400.0
12/18/80	0.475446	0.477679	0.475446	0.475446	0.021631	18362400.0

b) Data are all converted into correct format.
One unexpected issue is that some of downloaded dataset was in string format, for example, the AAPL.csv:

```
: df_Apple.info()
```

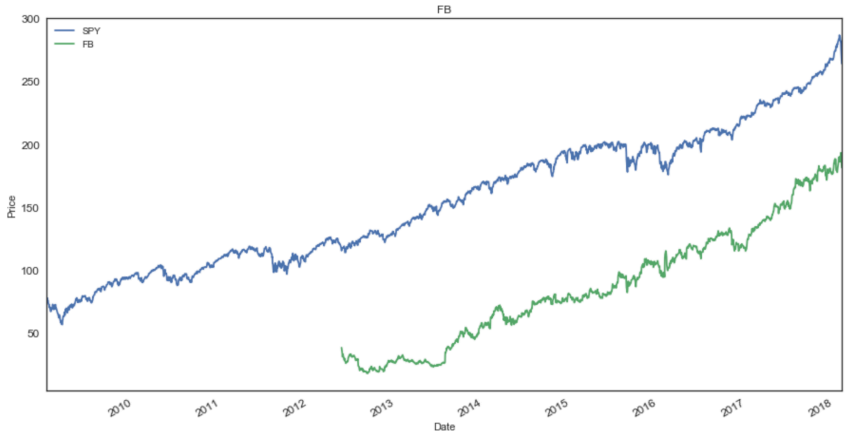
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9372 entries, 0 to 9371
Data columns (total 7 columns):
Date          9372 non-null object
Open          9372 non-null object
High          9372 non-null object
Low           9372 non-null object
Close         9372 non-null object
Adj Close     9372 non-null object
Volume        9372 non-null object
dtypes: object(7)
memory usage: 512.6+ KB
```

This incorrect format issue caused problems. The stock prices could not be plotted. Furthermore, computations could not be performed on those string data. One solution is the Pandas build-in function 'to_numeric'. After the commend "pd.to_numeric(df[symbol], errors = 'coerce')", the data have the proper format.

```
df_Apple = pd.read_csv('Dataset/AAPL.csv')
df_Apple.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9379 entries, 0 to 9378
Data columns (total 7 columns):
Date          9379 non-null object
Open          9376 non-null float64
High          9376 non-null float64
Low           9376 non-null float64
Close         9376 non-null float64
Adj Close     9376 non-null float64
Volume        9376 non-null float64
dtypes: float64(6), object(1)
memory usage: 513.0+ KB
```

c) The missing values are filled in.
Since Facebook had not gone IPO before 2012, it does not have the stock price data before 2012.



Since the target date range for this project starts from Jan 5th 2009 and the training for algorithms need continuous data, Pandas built-in function 'fillna' can solve this problem. Because Facebook lacks data in the front, so 'filla' needs to fill backward. The command "fillna(method = 'backfill', inplace = True)" shows the graph after filled in the missing values. The function fills backward from previous known values and this can provides objective results when missing data.

d) Drop null values

Some companies in some of days did not have stocks trading and dataset show 'NaN' (not a number) values in those days. In the function "pd.to_numeric(df[symbol], errors = 'coerce')", the inside parameter "error='coerce'" can set invalid parsing as NaN. Then the function "df.dropna()" can drop the rows contains NaN values.

e) Normalize the stock prices

To perform normalization, several procedures need to be done. Firstly, create an empty data frame indexed with date. Secondly, the dataset need to join in the empty data frame after read dataset. Before join in, only select values from 'Adj Close' column. Thirdly, build a helper function "normalize_data" to execute the commend "df/df.iloc[0,:]". In

Layman's term, it means that use every next day's stock price of those companies to divide the stock price in day Jan 5th 2009. Next, build another helper function "plot_data" to show the graph. The detailed code can be found in the file "helper.py".

3.2 Implementation

Three algorithms, Linear Regression, K-Nearest Neighbor and Support Vector Regression are chosen for this project.

Input features are 'Open', 'Close' 'High', 'Low', 'Volume', and target label is 'Adj Close'. After the data is cleaned, it can be put into use.

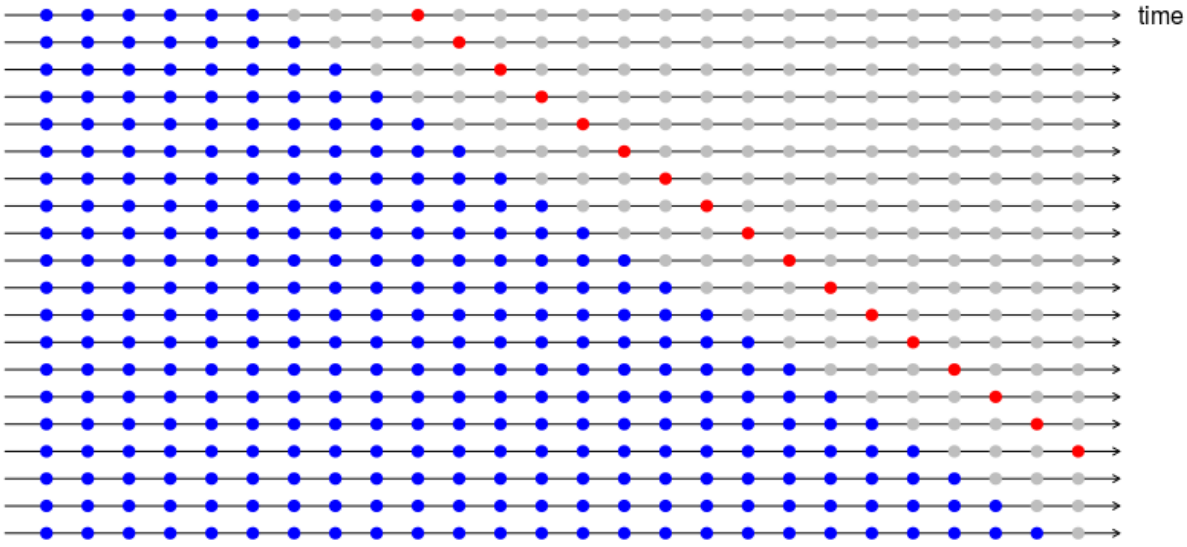
The greatest challenge for this project is to deal with time series data. The fact that the data is naturally ordered denies the possibility to apply general methods which by default tend to shuffle the entries losing the time information. Usually, in a common supervised learning problem, test and train split method can just easily split data to training and testing based on a ratio such as 0.8 or 0.75, and to get higher accuracy the cross-validation technique can be applied.

Time series data cannot use cross validation to improve accuracy, because time serious cannot be shuffled randomly. The reason time series data cannot be randomly shuffled is because a model will know the future after it is trained on a future dataset and it will be meaningless to make predictions about future.

There is a solution specifically for time series data problem. Here is the detailed explanation with a relevant graph.

1. First, clean the past 10 years of trading data and get only necessary features (dates, volume, adj close) into a new data frame.
2. Next, split data in test and train set given a date. (e.g. Feb 14th 2018). Train data is from Jan 5th 2009 to Feb 13th 2018, test data is Feb 14th 2018
3. Split train set in maybe 10 consecutive time folds
4. Train algorithms (linear regression, KNN, SVM and Random Forests) with train data.
5. Train on fold 1 → Test on the first day of fold 2
6. Train on fold 1+2 → Test on the first day of fold 3
7. Train on fold 1+2+3 → Test on the first day of fold 4
8. Train on fold 1+2+3+4 → Test on the first day of fold 5
9. Train on fold 1+2+3+4+5 → Test on the first day of fold 6
10. Train on fold 1+2+3+4+5+6 → Test on the first day of fold 7
11. Train on fold 1+2+3+4+5+6+7→ Test on the first day of fold 8

12. Train on fold 1+2+3+4+5+6+7+8 → Test on the first day of fold 9
13. Train on fold 1+2+3+4+5+6+7+8+9 → Test on the first day of fold 10
14. Compute the average accuracies of the 9 test folds.



This detailed code can be found at the file named “Project.ipynd”.

After followed the learning logic from the above, the results from three models before performing GridSearchCV come out are unexpected. Take sample of SPY as example.

In the case of Linear Regression, the r2 score is about 0.85, and the mean squared error is about 26.64

In the case of KNN, the r2 score is about -21.66, and the mean squared error is about 4481.36

In the case of SVR, the r2 score is about -24.91, and the mean squared error is about 17145.1.

Reference:

- http://francescopochetti.com/pythonic-cross-validation-time-series-pandas-scikit-learn/
- https://robjhyndman.com/hyndsight/tscv/

3.3 Refinement

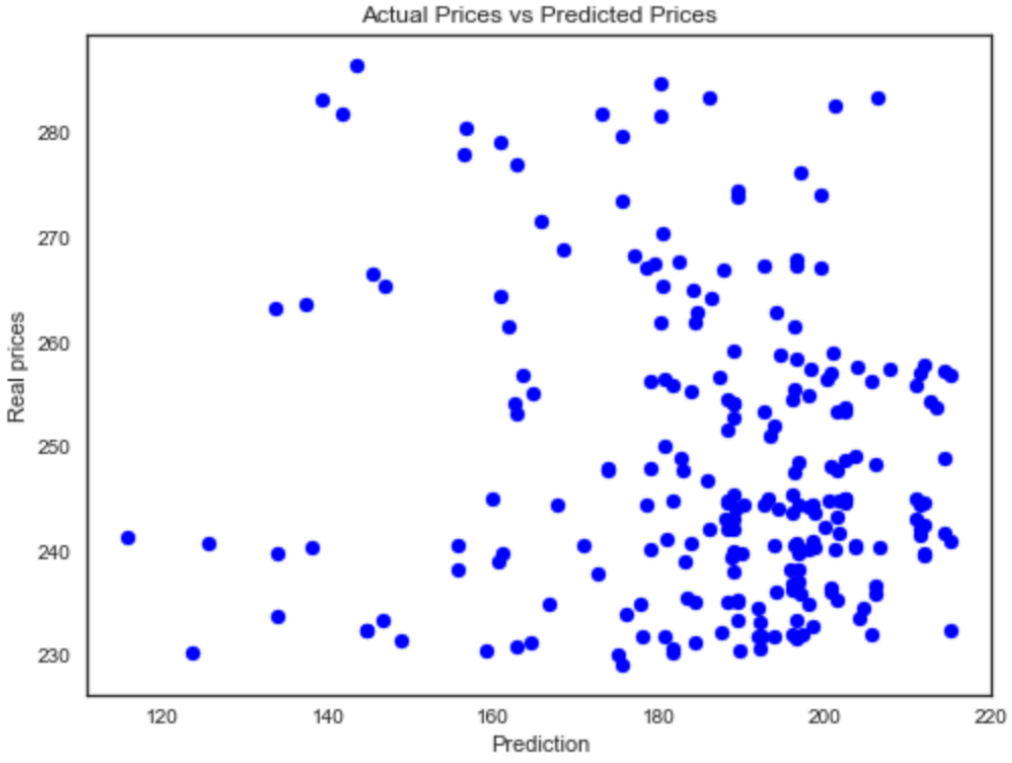
GridSearchCV is an effective way to tune the models. Specifying a parameter grid with different hyperparamters values can evaluate and find optimized parameter combinations.

In the case of KNN, three hyperparamters are turned: 'n_neighbors', which is the number of nearest points that need to be included into calculations; 'leaf_size', which is controls the number of samples at which a query switches to brute-force; ‘weights’, the value assigned to a query point is computed from a simple majority vote of the nearest neighbors. This is the result after implemented GridSearchCV:

```
best parameter:  {'n_neighbors': 30, 'weights': 'uniform', 'leaf_size': 1}
best score:  -35.5762346733


Score on training fold 8
regressor.score(X_trainFolds, y_trainFolds):  0.577586585714
Score on testing data
regressor.score(X_testFold, y_testFold): -21.66
Mean squared error:  4481.36477512
Average accuracy:  -21.6561842304
```

Unexpectedly, the result does not improve and the scatter points spread irregularly.

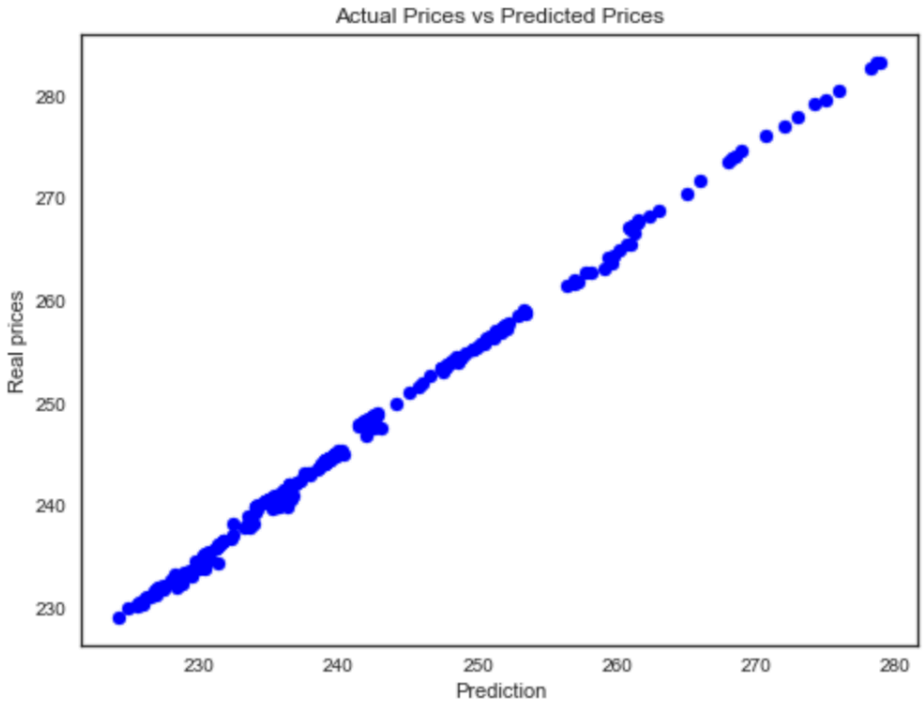


This detailed code can be found at the file named “Project.ipynd”.

4. Results

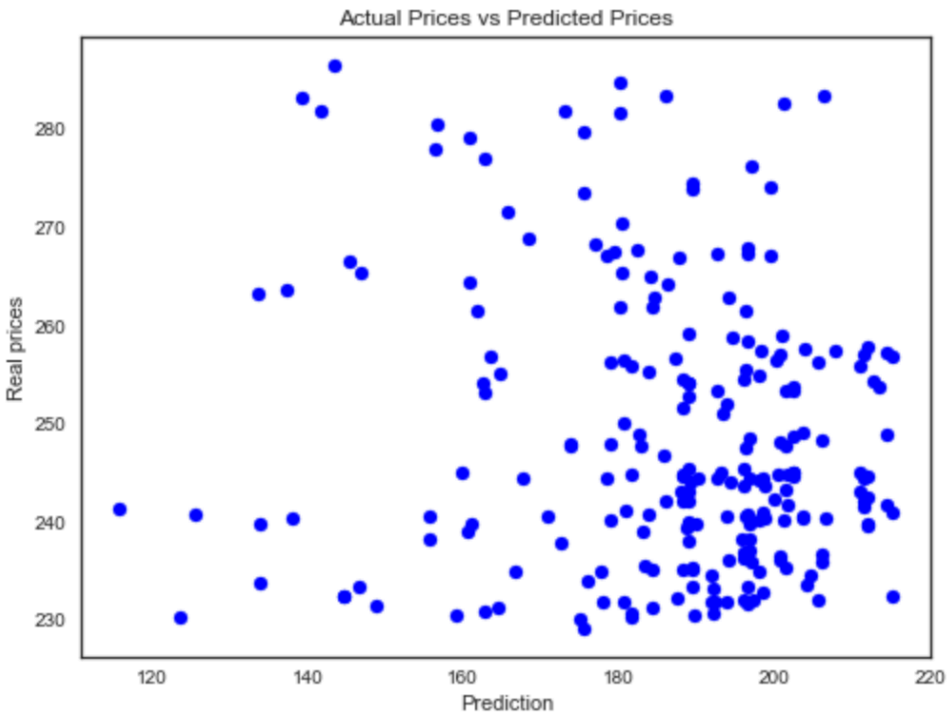
4.1 Model evaluation and metrics
This is the result generated by **Linear Regression** model with a plot chart.

```
Score on training fold 8
regressor.score(X_trainFolds, y_trainFolds):  0.997773877557
Score on testing data
regressor.score(X_testFold, y_testFold): 0.84
Mean squared error:  26.6395158578
Average accuracy for SPY: 0.852023613249
```



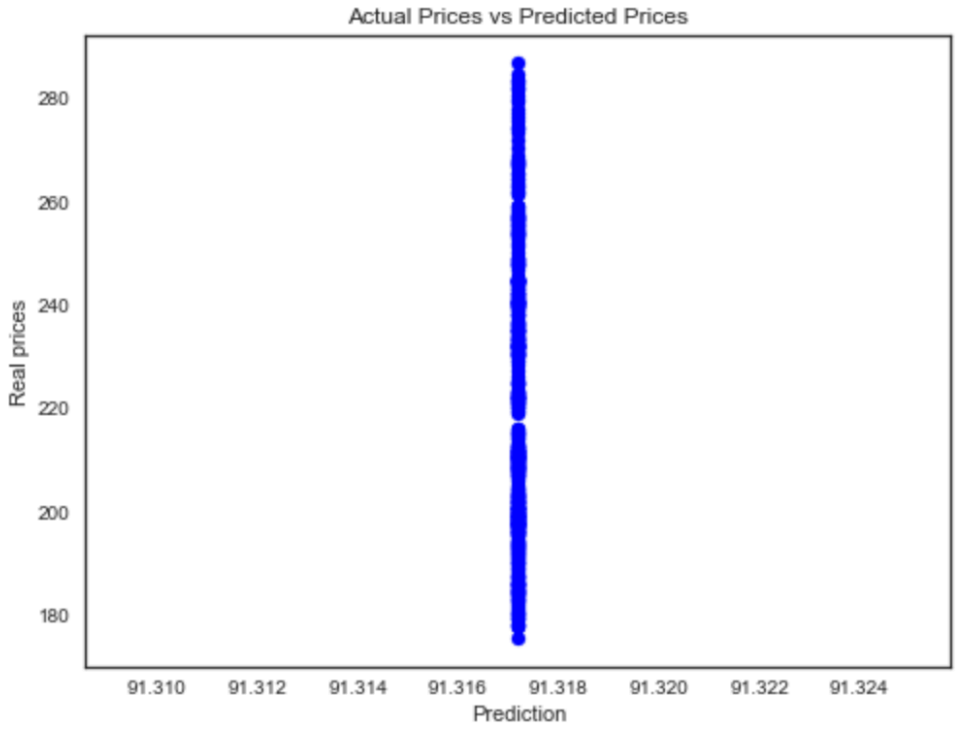
This is the result generated by **KNN** model with a plot chart.

```
Score on training fold 8
regressor.score(X_trainFolds, y_trainFolds):  0.577586585714
Score on testing data
regressor.score(X_testFold, y_testFold): -21.66
Mean squared error:  4481.36477512
Average accuracy:  -21.6561842304
```



This is the result generated by SVR with a plot graph.

```
Score on training fold 8
regressor.score(X_trainFolds, y_trainFolds):  0.0370121940688
Score on testing data
regressor.score(X_testFold, y_testFold): -24.91
Mean squared error:  17145.1009655
Average accuracy:  -24.9084354132
```



Observations after obtained results from three models are unexpected. The simple Linear Regression model, however, has the highest r2 score and smallest of mean squared error. The r2 score obtained from training SPY is not only greater than 0.5 but also actual high enough to show a strong correlation between predicted prices and actual prices. R2 sores and mean squared errors produced by KNN and SVR before performing GridSearchCV are unexpectedly low and high. R2 scores are even negative, which means that the model truly does not fit to the data point. The following refinement will verify the validation of KNN and SVR again by optimizing hyperparameters.

4.2 Justification

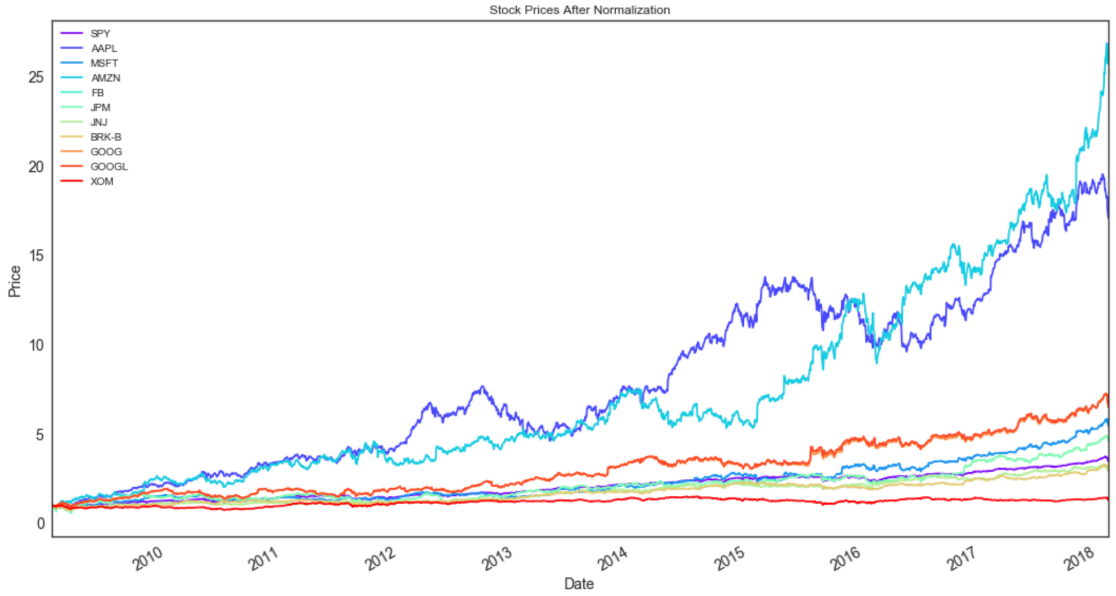
The benchmark is the model has an accuracy score that greater than 0.5, and the benchmark model is the Linear Regression. The Linear Regression model has an accuracy score 0.85 for the given dataset, and it passes the benchmark. KNN has an accuracy score of -21.66 after GridSearchCV, so it neither passes the benchmark nor benchmark model. The SVR has an accuracy score of -24.9 so it neither passes the benchmark nor benchmark model, either.

In terms of mean squared error, Linear Regress still has the smallest number of 26.64. KNN has a big number of 4481.36. SVR has the biggest number of 17145.1.

5. Conclusions

5. 1 Free-form visualization

The visualization of the stock prices after normalization is a good option. The reason that Linear Regression algorithm performs better than KNN or SVR is probably because of the linear property of stock price. The stock price movement patterns of some good companies from S&P500 can be detected more easily by this algorithm.



5.2 Improvements

Firstly, RNN should be adopted after future study on the topic.
Secondly, this project should continue to finish the goal of 7464 from Georgia Institute of Technology CS, which the final project can recommend people to optimize their investment portfolio after taking inputs including returns, alpha, beta, sharpe ratio, and moving average, etc.