Machine Learning Engineer Nanodegree

Capstone Project

Junghoon Lee

December 14, 2017

I. Definition

Project Overview

Machine Learning is using everywhere. There are few things we cannot do with Machine Learning in these days. Stock price prediction is the most interesting issue with this technology. Many hedge fund companies are using machine learning for stock prediction and keeping the best portfolio.
Prediction methodologies fall into two broad categories. They are fundamental analysis and technical analysis.
Fundamental Analysts are concerned with P/E ratio, validity of stock, evaluation of company past performance. Warren Buffett is the most famous fundamental investor. He did not trade stock frequently, but keep long time. (Murphy, John J. (1999). *Technical analysis of the financial markets*.)
However, technical analyst are not concerned with any of the company's fundamental, but rely on chart information. For example, exponential moving average, candlestick pattern.
This, on the other hand, is solely based on the study of historical price fluctuations. Practitioners of technical analysis study price charts for price patterns and use price data in different calculations to forecast future price movements (Turner, 2007).
Since I have been investing money on stock market, this project result will be my best information for future investing. I want to see how much machine learning algorithm can predict real stock price.
So, in this project, prediction of "adj close" stock price for next week is the goal. I have implemented Linear regression, polynomial regression, KNN regression and ensemble(polynomial + KNN regression) algorithm to predict S&P 500 ETF(IVV).
I have used rolling mean value(10, 20, 40 days), stock trading volume on each day, Adj Close price of each day.

Problem Statement

There are a lot of input data to predict future stock price. It will be quite difficult to make a special set or a equation for all stocks in stock market(Nasdaq). Some stock (S&P 500 ETF) is quite converged market signal(interest rate, oil price and so on), but some small stocks(special market) are not follow market signal. So, this domain is really fit to machine learning algorithm.
So , in this project, I will predict “adj close” stock price for a day, week, month time frame. Volume, 20-days mean stock value, 10-days mean stock value are used to input feature.

Metrics

I have used three machine learning algorithms to predict S&P 500 (IVV) etf.

- The first machine learning algorithm is linear regression. (http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html)

Input features are

- Change (ivv['Close'] - ivv['Open']) / ivv['Open']
- Maxofday (ivv['High'] - ivv['Low']) / ivv['Low']
- Adj Close ivv['Adj Close'] / ivv['Adj Close'][0]
- Volume ivv['Volume'] / ivv['Volume'].mean()
- GLD gold index price
- RM10 10 days rolling mean price
- RM20 20 days rolling mean price
- RM40 40 days rolling mean price

For performance evaluation, r2_score function is used for all algorithm (http://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html)

It provides a measure of how well future samples are likely to be predicted by the model. Best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y, disregarding the input features, would get a R^2 score of 0.0.

If $\hat{y}_i$ is the predicted value of the i-th sample and y_i is the corresponding true value, then the score R² estimated over n_{samples} is defined as

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

where $\bar{y} = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} y_i$.

So, this provides meaningful information about each algorithms performance to predict future stock price.

Class definition is in sp_linear_reg.py.

- The second machine learning algorithm is polynomial regression. (http://scikit-learn.org/stable/auto_examples/linear_model/plot_polynomial_interpolation.html)

I have created make pipeline with PolynomialFeatures(), Ridge(). In order to find the best degree value, I have tried 1, 2, 3, 4, 5, 6, 7, 8 degree values.

The same input features are used as linear regression case.

Input features are

- Change (ivv['Close'] - ivv['Open']) / ivv['Open']
- Maxofday (ivv['High'] - ivv['Low']) / ivv['Low']
- Adj Close ivv['Adj Close'] / ivv['Adj Close'][0]
- Volume ivv['Volume'] / ivv['Volume'].mean()
- GLD gold index price
- RM10 10 days rolling mean price
- RM20 20 days rolling mean price
- RM40 40 days rolling mean price

For performance evaluation, r2_score function is used for all algorithm (http://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html)

Class definition is in sp_poly_reg.py.

- The third machine learning algorithm is KNN regression. (http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html)

Since this is different from linear and polynomial regression, this get values from near input feature, I have used different input feature values.

If I use the same input feature value, KNN can not predict reasonable price.

For example, Adj Close, this is increasing value. So, from historical training value, there is no near value of current Adj close price. So, prediction from this algorithm cannot generate higher price than before.

So, I have used different input feature set here.

Main different is 'Adj Close' price. I have change this price to each day's change rate.

That is, (today close price - previous day's close price) / previous day's close price .

Then, this is each day's change rate.

Input feature set is

- Change (ivv['Close'] - ivv['Open']) / ivv['Open']
- Maxofday (ivv['High'] - ivv['Low']) / ivv['Low']
- Chg perc (today close price - previous day's close price) / previous day's close price
- Volume ivv['Volume'] / ivv['Volume'].mean()
- GLD gold index price
- RM10 10 days rolling mean price
- RM20 20 days rolling mean price
- RM40 40 days rolling mean price

Class definition is in sp_knn_reg.py.

II. Analysis

Data Exploration

From yahoo finance, S&P 500 etf(IVV) data can be attained(https://finance.yahoo.com/quote/IVV/history?p=IVV). That format is as below.

“Date” column explains the day of information gathered

“Open” column explains start stock price at that day.

“High” column explains highest stock price at that day.

“Low” column explains lowest stock price at that day.

“Close” column explains the last stock price at that day.

“Adj Close” column explains Adjusted Close price stock price at that day. This includes dividends and stock split if it happened.

Date	Open	High	Low	Close*	Adj Close**	Volume
Dec 15, 2017	268.70	270.25	268.61	269.79	269.79	7,657,234
Dec 14, 2017	268.93	269.07	267.44	267.53	267.53	5,997,300
Dec 13, 2017	268.90	269.41	268.51	268.57	268.57	3,387,200
Dec 12, 2017	269.05	269.17	268.19	268.63	268.63	3,373,400

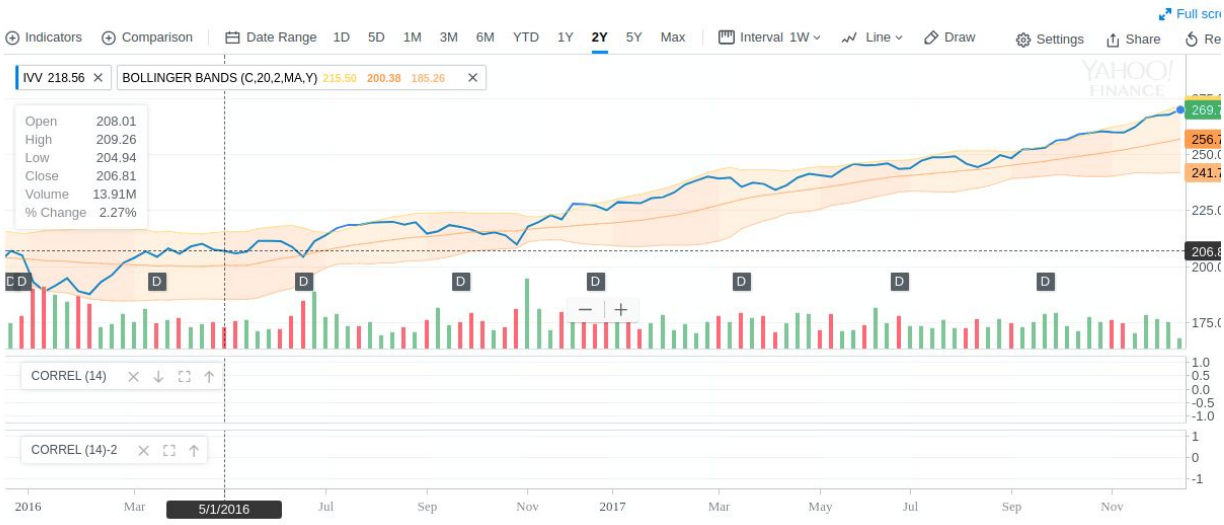
I have used gold price from GLD yahoo ticker. (https://finance.yahoo.com/quote/GLD/history?p=GLD)

Data format is the same.

Date	Open	High	Low	Close*	Adj Close**	Volume
Dec 15, 2017	119.41	119.50	118.97	119.18	119.18	7,531,964

Dec 14, 2017	119.10	119.29	118.71	118.93	118.93	6,992,900
Dec 13, 2017	118.19	119.35	118.01	119.17	119.17	10,086,700
Dec 12, 2017	117.65	118.17	117.40	118.15	118.15	8,020,100

IVV “adj close” price graph is as below.

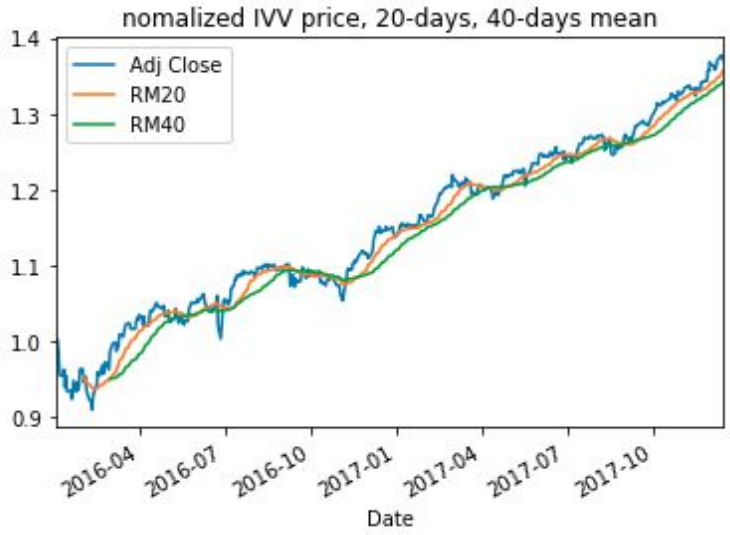


The price is increasing continuously and looks like linear graph. So, linear regression will be quite fit this stock prediction. In addition, polynomial regression will provide better performance. In addition, 20 day rolling mean(like bollinger band) is quite related well.

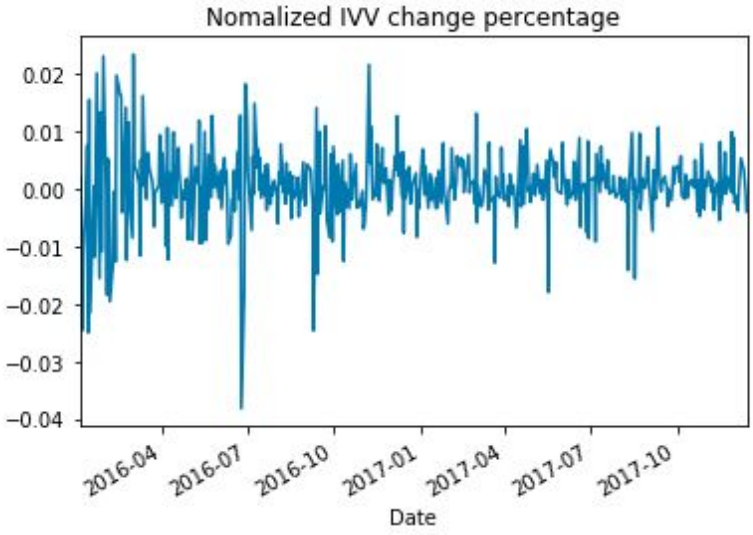
In order to fit this feature to KNN, I need to change major input value, “Adj Close” value to change rate of each day. If I use “Adj Close” value, KNN cannot find near value of days which is near from today. Because, price is increasing and that was not exist before. KNN will output the highest value of it’s table, then, prediction cannot be higher than before.

Exploratory Visualization

Input feature for linear regression and polynomial regression is as below.



For KNN, IVV adjusted close price change to each day's change rate.



I have choose about 2 year's history data for machine learning algorithm. If I select too long, machine learning algorithm may not focus on the current days. Since our most interest on these days.

In addition, duration of training data is from 2016-01-01 to 2017-10-11.

Since the first 40 days of rolling mean value were nan, the first 40 days are removed, since we need 40 days rolling means.

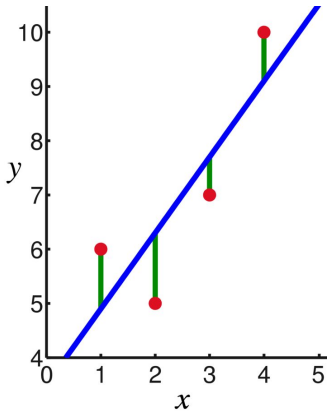
Test dataset is from 2017-10-12 to 2017-12-14.

Training set has 378 samples. Testing set has 45 samples.

Algorithms and Techniques

Linear regression algorithm is for benchmark test. This data is reference for the other two algorithms. So, I focus on default configuration of linear regression.

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

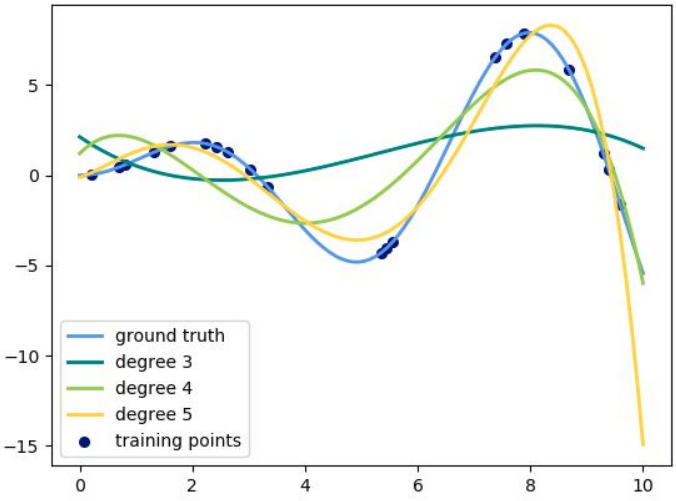


Linear regression example graph
(https://en.wikipedia.org/wiki/File:Linear_least_squares_example2.png)

Polynomial regression, PolynomialFeatures() function is used to make feature set of polynomial. Features are selected as this. if an input sample is two dimensional and of the form [a, b], the degree-2 polynomial features are [1, a, b, a^2, ab, b^2].

polynomial regression is a form of regression analysis in which the relationship between the independent variable x and the dependent variable y is modelled as an nth degree polynomial in x. Polynomial regression fits a nonlinear relationship between the value of

x and the corresponding conditional mean of y, denoted E(y |x), and has been used to describe nonlinear phenomena such as the growth rate of tissues, the distribution of carbon isotopes in lake sediments, and the progression of disease epidemics. Although polynomial regression fits a nonlinear model to the data, as a statistical estimation problem it is linear, in the sense that the regression function E(y | x) is linear in the unknown parameters that are estimated from the data. For this reason, polynomial regression is considered to be a special case of multiple linear regression. (https://en.wikipedia.org/wiki/Polynomial_regression)



Polynomial regression example graph (http://scikit-learn.org/stable/auto_examples/linear_model/plot_polynomial_interpolation.html)

Ridge function, Ridge Regression or Tikhonov regularization, was used to solve a regression model where the loss function is the linear least squares function

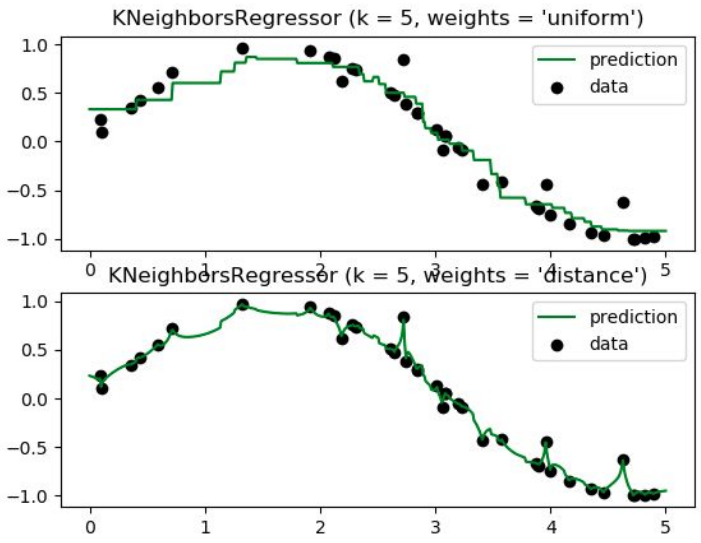
Ridge regression addresses some of the problems of ordinary least square by imposing a penalty on the size of coefficients. The ridge coefficients minimize a penalized residual sum of squares,

$$\min_{w} ||Xw - y||_2^2 + \alpha ||w||_2^2$$

Here, $\alpha \geq 0$ is a complexity parameter that controls the amount of shrinkage: the larger the value of α, the greater the amount of shrinkage and thus the coefficients become more robust to collinearity.
(http://scikit-learn.org/stable/modules/linear_model.html#ridge-regression)

KNN regression, I choose weights value to 'distance' which weight points by the inverse of their distance. in this case, closer neighbors of a query point will have a greater influence than neighbors which are further away.

Demonstrate the resolution of a regression problem using a k-Nearest Neighbor and the interpolation of the target using both barycenter and constant weights.

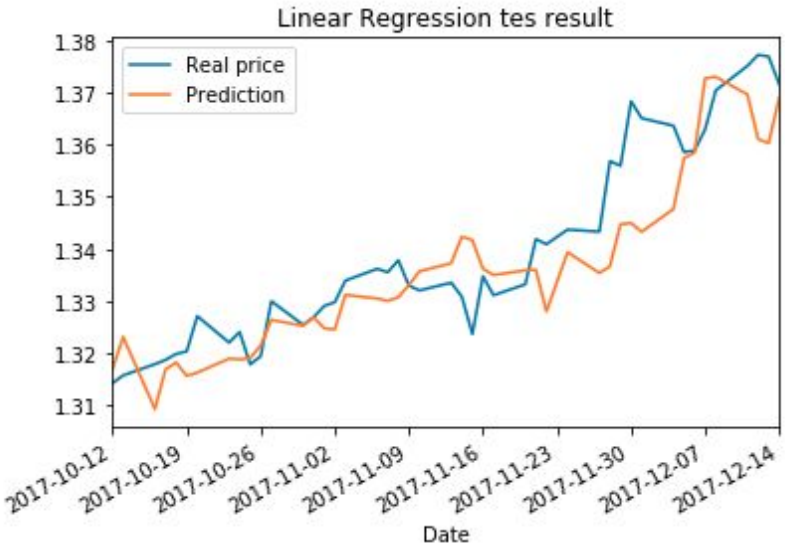


KNN regression sample graph (
http://scikit-learn.org/stable/auto_examples/neighbors/plot_regression.html)

Benchmark

From linear regression, this result was attained.

After training with training dataset, test result is below. Blue line is real price and yellow line is prediction price.



Mean square error is 3.32, variance score is 0.76.

- Mean squared error: 3.32
  Variance score: 0.76

Based on this result, linear regression seems reasonable prediction.

Feature input are

Change Maxofday Adj Close Volume GLD RM10 RM20 RM40
And coefficent value is
('Coefficients: \n', array([0.01253308, 0.04866664, 0.**87020545**, 0.00674234, -0.00435119,
0.01598344, 0.01444126, 0.10520062]))
The third input feature has most high coefficient value. So linear regression rely on 5 day's previous adj close price to predict future's adjusted close price.

III. Methodology

Data Preprocessing

For linear regression and polynomial regression has the same preprocessing method.

Which are defined at class SpPolyReg and class SpLinearReg, preprocess_fetures method.

Preprocess_features method does normalize input features and add normalized gold index ETF price to feature set.

Input features are

- Change
  normalized each day's close - open price : (ivv['Close'] - ivv['Open']) / ivv['Open']
- Maxofday
  Normalized each day's high - low price : (ivv['High'] - ivv['Low']) / ivv['Low']
- Adj Close
  Normalized Adj close price : ivv['Adj Close'] / ivv['Adj Close'][0]
- Volume
  Normalized each day's trade stock volume with mean :
  ivv['Volume'] / ivv['Volume'].mean()
- GLD
  Normalized gld Adj close price : gld['Adj Close'] / gld['Adj Close'][0]
- RM10 10 days rolling mean price with Normalized Adj close price
- RM20 20 days rolling mean price with Normalized Adj close price
- RM40 40 days rolling mean price with Normalized Adj close price

For KNN regression defines preprocess function in class SpKnnReg, preprocess_fetures function.

Preprocess_features method does normalize input features and add normalized gold index ETF price to feature set.

Input feature set is

- Change
  normalized each day's close - open price : (ivv['Close'] - ivv['Open']) / ivv['Open']
- Chg perc
  Adj Price change rate of each day : (today close price - previous day's close price) / previous day's close price
- Maxofday
  Normalized each day's high - low price : (ivv['High'] - ivv['Low']) / ivv['Low']
- Adj Close
  Normalized Adj close price : ivv['Adj Close'] / ivv['Adj Close'][0]
- Volume
  Normalized each day's trade stock volume with mean :

ivv['Volume'] / ivv['Volume'].mean()

- GLD
  Normalized gld Adj close price : gld['Adj Close'] / gld['Adj Close'][0]
- RM10 10 days rolling mean price with Normalized Adj close price
- RM20 20 days rolling mean price with Normalized Adj close price
- RM40 40 days rolling mean price with Normalized Adj close price

Implementation

Python and sklearn are used for this project.

4 classes are defined.

- SpMl class - Stock Prediction Machine Learning class.

This is base class. Two methods were defined to get stock price data from yahoo finance. Any general method for stock prediction will be written to here.

Define method to get stock information from yahoo finance.

- SpLinearReg class - Stock prediction Linear Regression class.

Define methods for preprocess, regression, prediction, display.

- SpPolyReg - Stock prediction Polynomial Regression class.

Define methods for preprocess, regression, prediction, display.

- SpKnnReg - Stock prediction KNN Regression class.

Define methods for preprocess, regression, prediction, display.

Preprocess method defined as below.

```
def preprocess_fetures(self):


    nm_ivv = self.ivv.copy(deep=True)

    nm_ivv['Adj Close'] = nm_ivv['Adj Close']/nm_ivv['Adj Close'][0]
```

```
  nm_ivv['Volume'] = nm_ivv['Volume']/nm_ivv['Volume'].mean()

  nm_ivv['Open'] = (nm_ivv['Close']-nm_ivv['Open'])/nm_ivv['Open']

  nm_ivv['High'] = (nm_ivv['High']-nm_ivv['Low'])/nm_ivv['Low']

  nm_ivv = nm_ivv.rename(columns = {'Open':'Change'})

  nm_ivv = nm_ivv.rename(columns = {'High':'Maxofday'})


  nm_gld = self.gld.copy(deep=True)

  nm_gld['Adj Close'] = nm_gld['Adj Close']/nm_gld['Adj Close'][0]

  nm_gld['Volume'] = nm_gld['Volume']/nm_gld['Volume'].mean()


  self.prices = nm_ivv['Adj Close']

  self.features = nm_ivv.drop(['Close', 'Low'], axis = 1)


  # Rolling Mean 10 days

  rm10_ivv = pd.rolling_mean(nm_ivv['Adj Close'], window=10)

  rm10_ivv_ = pd.DataFrame(rm10_ivv)

  rm10_ivv_ = rm10_ivv_.rename(columns = {'Adj Close':'RM10'})

  # Rolling Mean 20 days

  # Rolling Mean 40 days

…..

  nm_gld = nm_gld.drop(['Open', 'High', 'Close', 'Low', 'Volume'], axis = 1)

  nm_gld = nm_gld.rename(columns = {'Adj Close':'GLD'})


  self.features = self.features.join(nm_gld['GLD'], how='inner')

  self.features = self.features.join(rm10_ivv_, how='inner')

  self.features = self.features.join(rm20_ivv_, how='inner')
```

```
        self.features = self.features.join(rm40_ivv_, how='inner')


        return
```

Do_regression method does fit and predict

```
    def do_regression(self):
        X_train = self.features[65:-50]
        y_train = self.prices[70:-45]
        X_test = self.features[-50:-5]
        y_test = self.prices[-45:]

        # Show the results of the split
        print "Training set has {} samples.".format(X_train.shape[0])
        print "Testing set has {} samples.".format(X_test.shape[0])

        regr = linear_model.LinearRegression()
        regr.fit(X_train, y_train)

        # Make predictions using the testing set
        y_pred = regr.predict(X_test)

        LR_result = pd.DataFrame(y_test)

        LR_result = LR_result.assign(Prediction = y_pred)
        LR_result = LR_result.rename(columns = {'Adj Close':'Real price'})
        LR_result[['Real price','Prediction']].plot(title='Linear Regression test result')
        plt.show()
```

```
        # The coefficients

        print('Coefficients: \n', regr.coef_)

        # The mean squared error

        print y_test, y_pred

        print("Mean squared error: %.2f" % mean_squared_error(y_test * self.ivv['Adj Close'][0], y_pred *
self.ivv['Adj Close'][0]))

        # Explained variance score: 1 is perfect prediction

        print('Variance score: %.2f' % r2_score(y_test, y_pred))



        return
```

Refinement

I had a difficulty to get mean square error result with prediction and rear values. That was from normalization of output value(adj close price). Since I normalized this value from around 200 to 1, MSE result goes to 0.

So, I make input parameters for MSE as real value, and I can get meaningful result.

mean_squared_error(y_test * self.ivv['Adj Close'][0], y_pred * self.ivv['Adj Close'][0]))

IV. Results

Model Evaluation and Validation

From polynomial regression, I can get better performance than linear regression.

Degree values from 1 to 8, I can see overfitting from higher degree values.

degree : 1
Mean squared error: 22.72
Variance score: -0.59

degree : 2
Mean squared error: 2.36
Variance score: 0.84

degree : 3
Mean squared error: 7.51
Variance score: 0.48

degree : 4
Mean squared error: 7.23
Variance score: 0.50

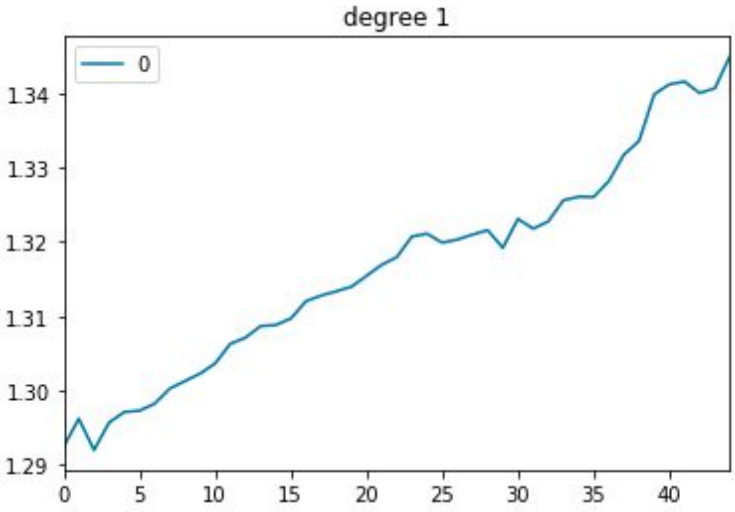
degree : 5
Mean squared error: 3.06
Variance score: 0.79

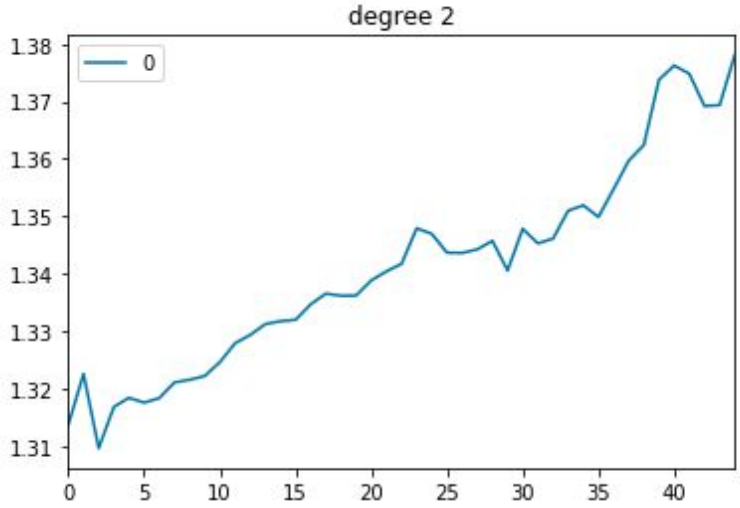
degree : 6
Mean squared error: 5.07
Variance score: 0.65

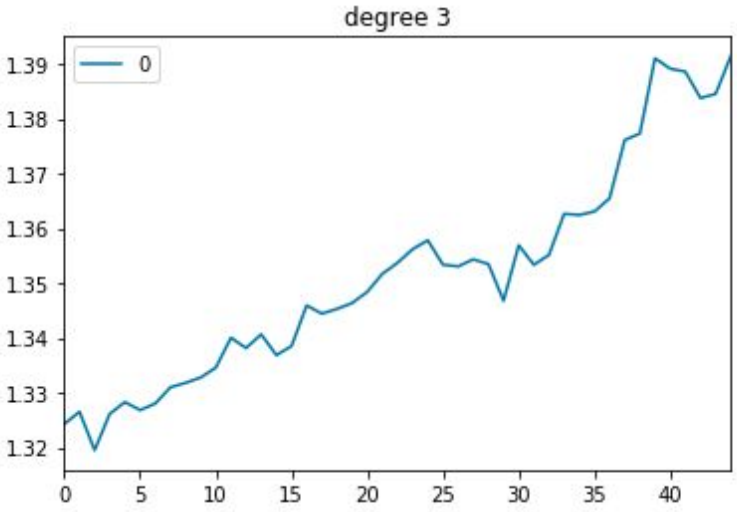
degree : 7
Mean squared error: 11.66
Variance score: 0.19

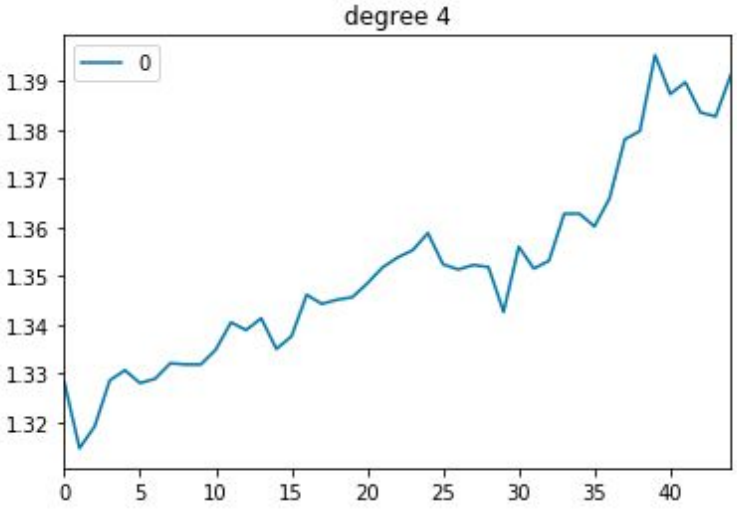
degree : 8
Mean squared error: 22.79
Variance score: -0.59

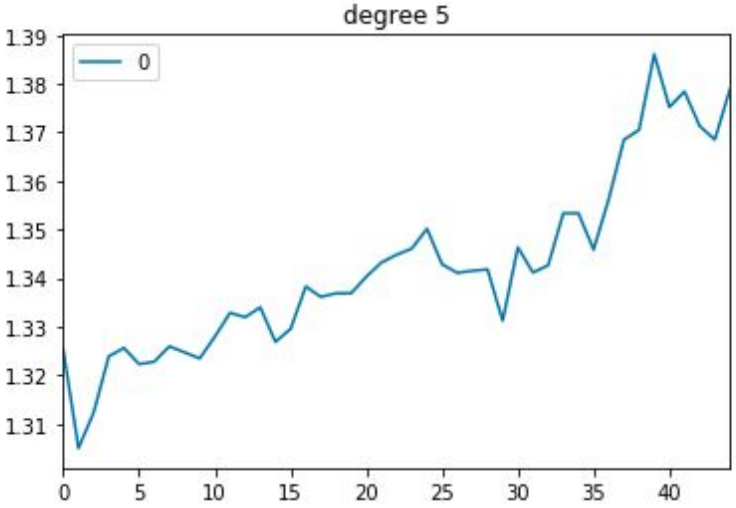
From graph, this became more clear.

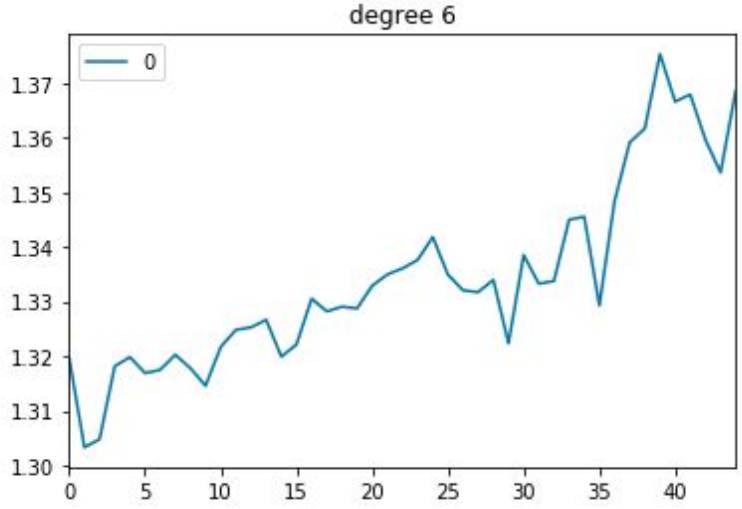

degree 1
0
1.34
1.33
1.32
1.31
1.30
1.29
0
5
10
15
20
25
30
35
40

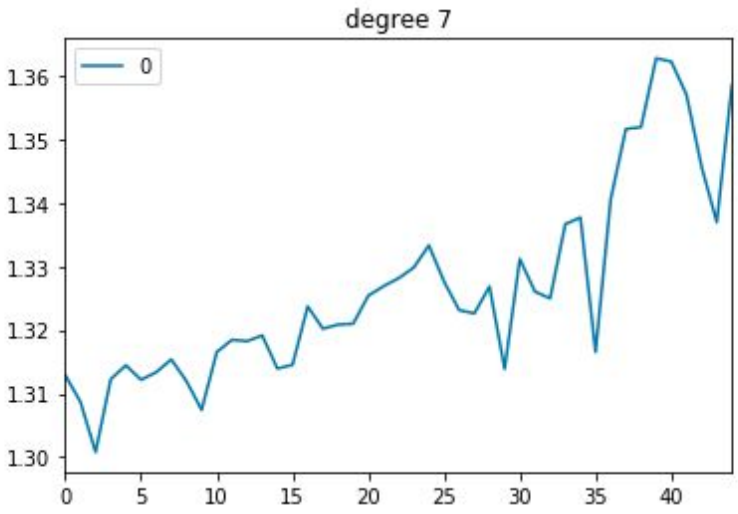

degree 2
0
1.38
1.37
1.36
1.35
1.34
1.33
1.32
1.31
0
5
10
15
20
25
30
35
40

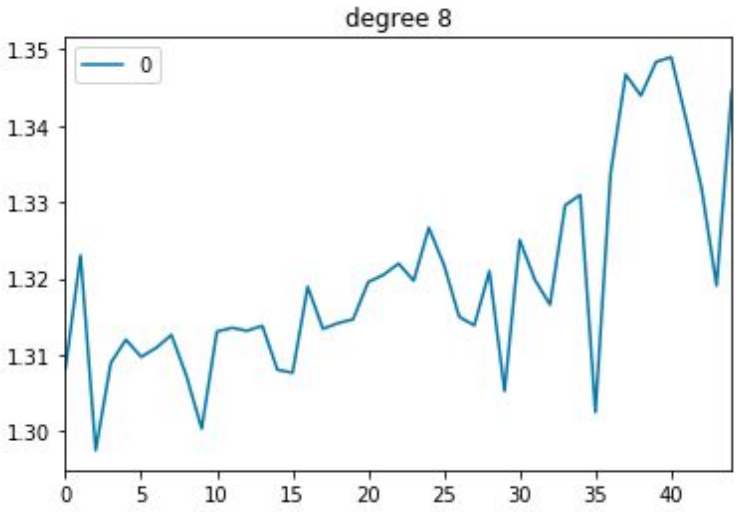

degree 3
0
1.39
1.38
1.37
1.36
1.35
1.34
1.33
1.32
0
5
10
15
20
25
30
35
40


degree 4
0
1.39
1.38
1.37
1.36
1.35
1.34
1.33
1.32
0
5
10
15
20
25
30
35
40


degree 5
0
1.39
1.38
1.37
1.36
1.35
1.34
1.33
1.32
1.31
0
5
10
15
20
25
30
35
40


degree 6
0
1.37
1.36
1.35
1.34
1.33
1.32
1.31
1.30
0
5
10
15
20
25
30
35
40

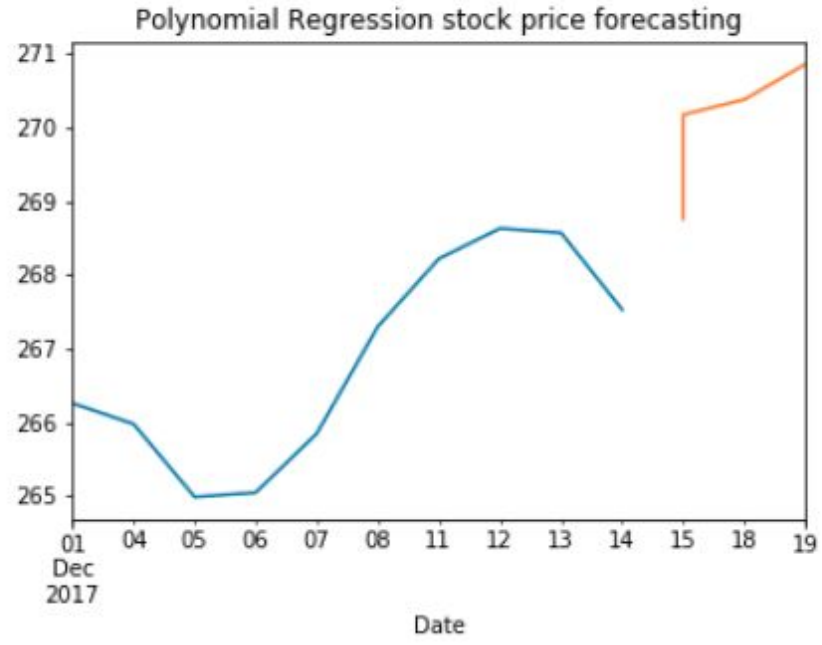




Best degree is 2. R2 Variance score is 0.82. This is better than linear regression.

So, I can predict next week's stock price with degree value 2.

```
Name: Adj Close, dtype: float64
            Prediction
Date
2017-12-15  268.759163
2017-12-16  269.761541
2017-12-17  270.171978
2017-12-18  270.377735
2017-12-19  270.856349
```



I have executed this on 2017-12-14. So, I predicted 5 days from 2017-12-15.

From KNN, I tried multi neighbor values as below.

n_neighbors : 5
Mean squared error: 0.19
Variance score: -0.40

n_neighbors : 7
Mean squared error: 0.17
Variance score: -0.25

n_neighbors : 10
Mean squared error: 0.17
Variance score: -0.22

n_neighbors : 15
Mean squared error: 0.16

Variance score: -0.18

n_neighbors : 20
Mean squared error: 0.15
Variance score: -0.11

n_neighbors : 25
Mean squared error: 0.15
Variance score: -0.07

n_neighbors : 30
Mean squared error: 0.15
Variance score: -0.06

n_neighbors : 35
Mean squared error: 0.14
Variance score: -0.04

n_neighbors : 40
Mean squared error: 0.14
Variance score: -0.04

n_neighbors : 50
Mean squared error: 0.14
Variance score: -0.03

n_neighbors : 60
Mean squared error: 0.14
Variance score: -0.03

n_neighbors : 70
Mean squared error: 0.14
Variance score: -0.03

n_neighbors : 80
Mean squared error: 0.14
Variance score: -0.03

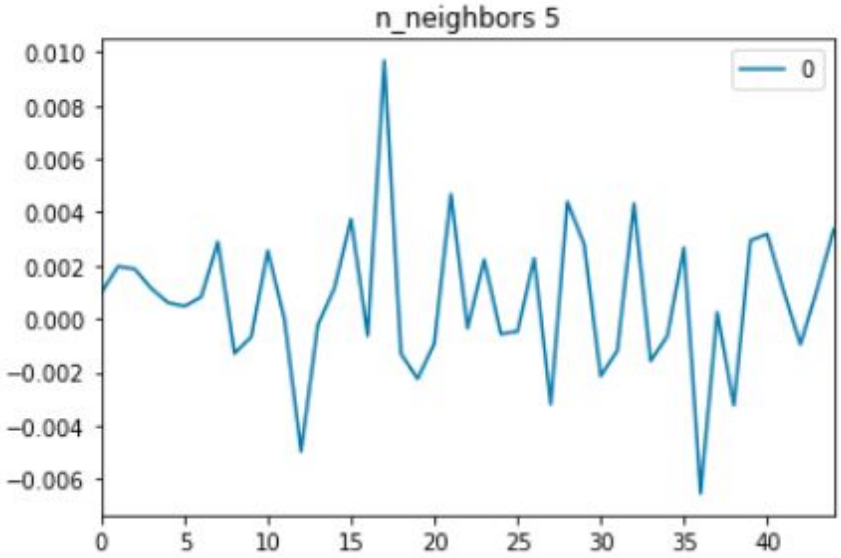
n_neighbors : 90
Mean squared error: 0.14
Variance score: -0.03

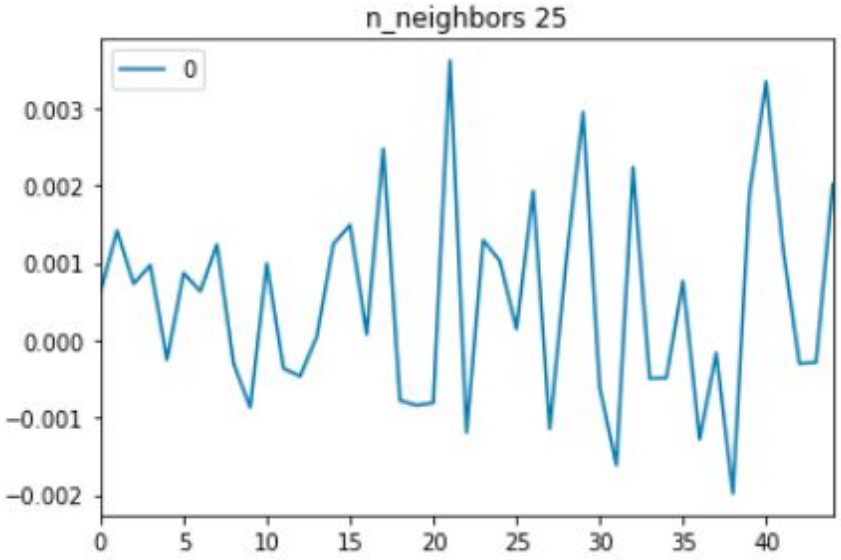
n_neighbors : 100
Mean squared error: 0.14
Variance score: -0.01

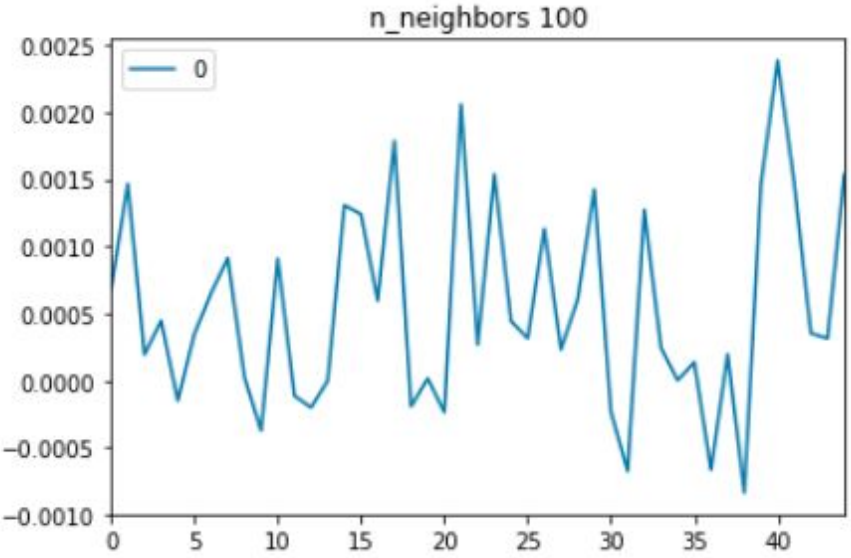
n_neighbors : 200
Mean squared error: 0.14
Variance score: -0.01

R2 Variance score goes to 0, but that needs too many neighbor. This is better result when I used original adjusted close price. However, this is not that reasonable result.

From graphs, x-axis is days, y-axis is change rate of adjusted close price.

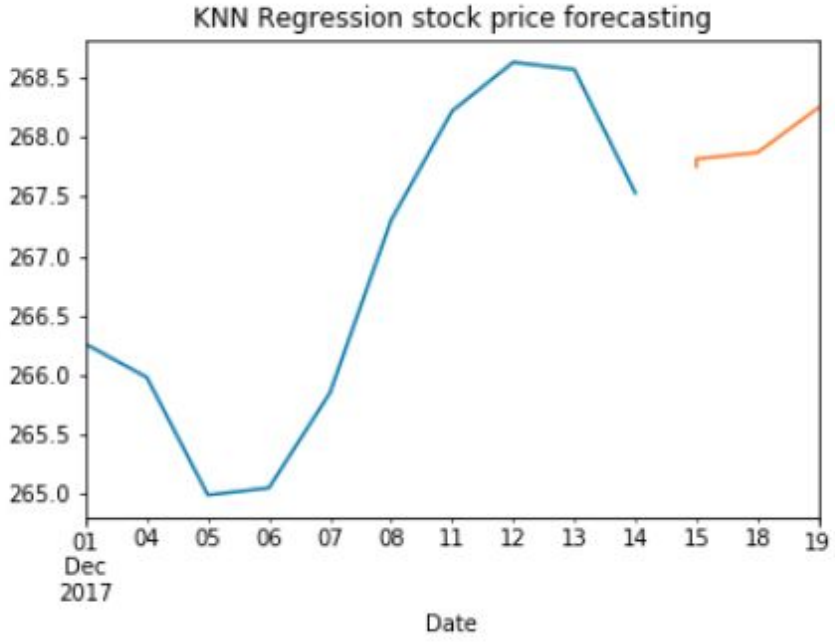






I predicted next five days stock price with this parameter value as below.

Date	Prediction
2017-12-15	267.753224
2017-12-16	267.807902
2017-12-17	267.818023
2017-12-18	267.870588
2017-12-19	268.248877

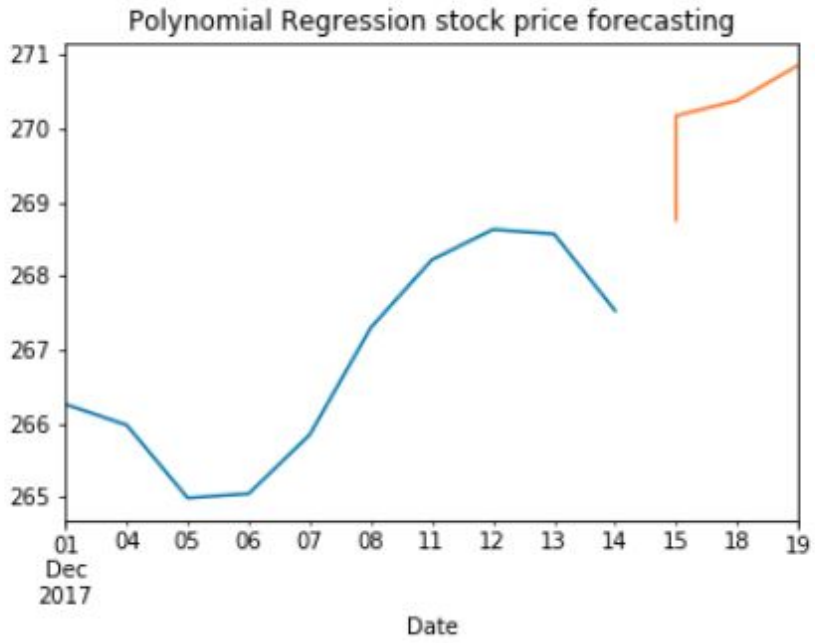


Finally, I have merged polynomial regression and KNN regression prediction result.

`ensemble = (kr_task.week_forecast + pr_task.week_forecast) /2`

Then, ensemble result is

Date	Prediction
2017-12-15	268.759163
2017-12-16	269.761541
2017-12-17	270.171978
2017-12-18	270.377735
2017-12-19	270.856349



Since KNN regression prediction performance is not that good to use, this ensemble machine learning prediction is also not meaningful.

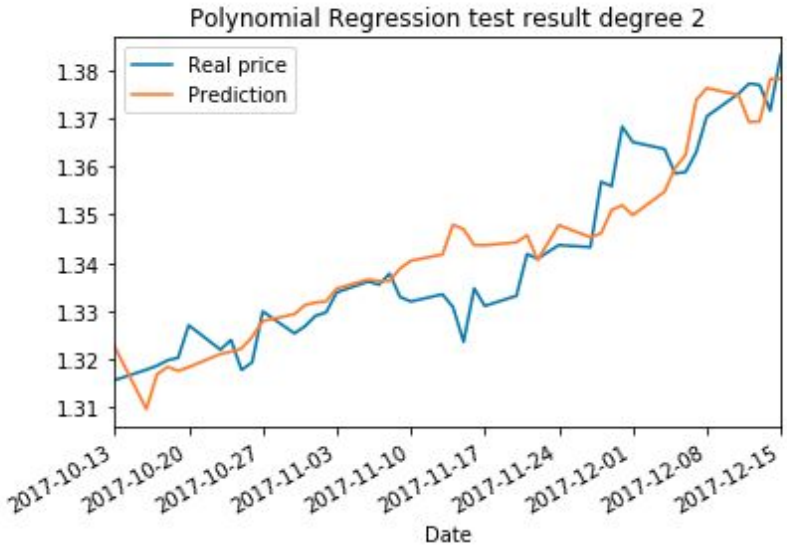
Justification

Polynomial regression could provide better performance than linear regression as I expected. However, KNN does not show meaningful prediction.

V. Conclusion

Regression algorithms are nice to use for stock prediction area. Linear regression provide quite nice performance, and polynomial regression provide better result. However, this is from S&P 500 ETF property. Since this ETF has 500 best company stocks in it, this has

low volatility. With rolling mean and short term before value have good relation with near future price.



Although linear & polynomial regression shows good performance, KNN's result was not acceptable. This will be future investigation to make this project better.
So, in order to use normal stock for each company, we need to gather more precise data, for example, earning result for every quarter, earning per share value. More fundamental data should be used. In addition, more machine learning algorithm can be implemented to predict future stock price include deep learning.