Machine Learning Engineer Nanodegree

Capstone Project

WONHO NA
October 22st, 2018

I. Definition

Project Overview

There has been a lot of tries to predict the stock market. As such, stock market prediction is a problem that so many people wants to solve. One of my desire using machine learning to address is also predicting stock market price. The stock price is very very hard to predict but also known that it is not entirely random walk. And it means there has been somehow patterns to tackle. So I thought that the problem could be solved by using a machine learning algorithm not even 100% accurate but closer to the actual price. It is also known that there are so many companies trying to predict stock price with machine learning. In the capstone project, I created a web application that is predicting stock market price with input values such as tickle, which is a symbol of a specific company, and the future days. And training the past stock price of given company, the application shows the predicted value of the given next day and the stock price graph.

Problem Statement

The problem to solve is obvious. The problem is to predict the future stock price and how close the expected amount to the actual cost. So I tried to predict the price of a specific next day as close as possible based on previous closing stock values. To solve this problem, I have to get the stock price data, and I got the data from 'Quandl' because they provide excellent API about the stock price. They also offer python API, so it's very convenient to get the stock price data. I don't need to save the stock data in CSV files and load the file to train the model or predict a future price. I only do the API call, and that's it.

Metrics

The model has to predict the specific value which means the model is a regression model. So the metrics are related to how close the predicted value to the actual value. So I use the 'mse' or mean squared error for metrics. Here's the definition of the MSE

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

https://en.wikipedia.org/wiki/Mean_squared_error

II. Analysis

Data Exploration

The Quandl API fetches the dataset. The size of the dataset varies from the company which to be predicted. For example, when predicting the company Amazon, I will use the recent five years of data for the training, validating, and testing The first four years dataset will be used training the predictor, and a year of the next data set will be used for confirming the model, and the dataset from the first of this year to about March will be used to test my model. The dataset is not included the Saturday, and Sunday, so the size of the one-year dataset is not like 365 days. And in this project, I only use one feature for training the predictor, Adj. Closing price. Here is the definition of the difference between Closing and Adj. Close price.

An adjusted closing price is a stock's closing price on any given day of trading that has been amended to include any distributions and corporate actions that occurred at any the time before the next day's open https://www.investopedia.com/terms/a/adjusted_closing_price.asp

Exploratory Visualization

Here is the stock price of the AMZN. The stock price has been increased since 1997. For the model to be fitted more accurately, the data has to be normalized. But the data has no lower bound and upper bound. So normalization for the whole dataset is not a good idea because it is not appropriate for normalizing future price data with current value. Because the price keeps increasing and when normalizing whole data, the early time's data would not affect much to the model training. So, I decide for the normalizing process to has been done for each window size or time step. As we'll see below, the model I chose to predict stock price is LSTM and window size or time step matters in the model. Through this normalization process, the predictor would learn the pattern no matter what the

absolute prices are.

Algorithms and Techniques

The LSTM is known for excellent performance in time series data prediction. So there have been many tries to predict stock price with LSTM model. The LSTM is the improved model of RNN. The RNN also used for time series prediction. But the RNN model has gradient vanishing problem. But the LSTM model improved and solved this issue. So the LSTM the model could train well with the data far from current time and has better performance for long-term time series data. So I thought that this model is best for predicting stock price.

Benchmark

I used Linear Regression model for a benchmark because this model is straightforward among regression models. This model predicts merely values based on the linear relation between feature and the target value. I don't think that this model has good performance because I don't believe that this model couldn't train the time series patterns. But as we'll see below, the Linear Regression model seems much more accurate than the LSTM model.

III. Methodology

Data Preprocessing

The preprocessing consist of the following steps: 1. take data to correspond to the input ticker from Quandl API 2. confine the data from step 1 starting from 4 years from the last date of the data. 3. pick up columns which are used to feature data. 4. split the data from step 3 into train set, validation set, and test set. At this time, each dataset should be saved in the form of 3D array consist of sequence, time step, and features because the LSTM model has one epoch with a time step. 5. each sequence has to be normalized for better performance. The normalization is proceeded by using this

$$\text{Normalization: } n_i = \left(\frac{p_i}{p_0}\right) - 1$$

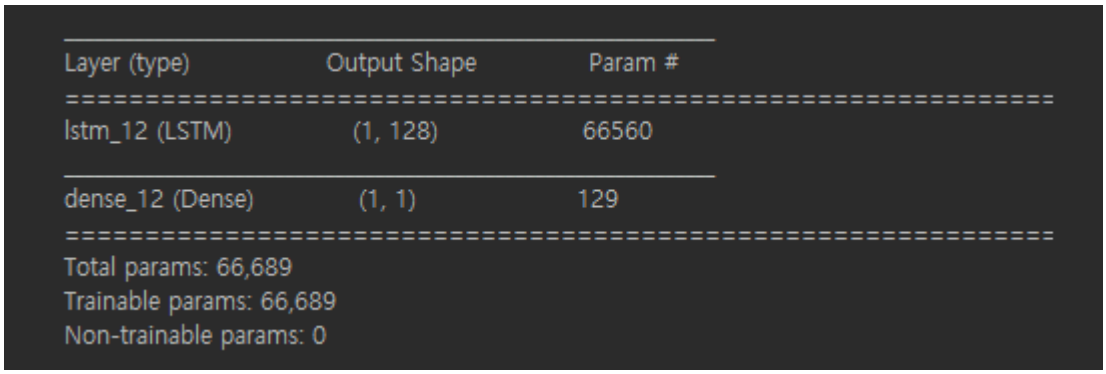
equation. Here, P0 is the first value of each, and the Pi is the ith value of the sequence. The denormalization is going to proceed when the model predicts sequences multiple.

Implementation

The implementation process can be split into two main stages: 1. The predictor training stage 2. The application development stage

During the first stage, The predictor was trained on the preprocessed training data. And I used the LSTM model which provided from Kears. The whole process was done by Jupyter notebook, and can be further divided into the following steps: 1. Define the network architecture and training parameters. 2. Define also the loss function and optimizer. the loss function is Mean Squared Error

as mentioned above, and the optimizer is 'Adam.' 3. training the LSTM model with the training set. 4. make LossHistory class for logging the loss of each epoch when training the model. 5. Plot the loss values versus epochs with training data. 6. bound the stock data to validation dataset range and evaluate the evaluation data with the trained model. 7. If the loss is not good enough, return to step 3 and increase the number of epochs or number of batch size or other parameters used in the model. 8. save the model and store the model at the saved_model directory so that every time the app is used to predict future price, the trained model can be used without long delay time.



The application development process can be split into three main stages: 1. make /index page. Here, the user input company ticker and days from now that he/she wants to know. 2. the input data is used in the /predict path. In the /predict path, the application load file which has the information about the trained model and uses the recent data of the size of the time step, and predicts the stock price of the future day which the user wants to know. When the model predicts the prices, first normalize the first sequence to predict the next value, and after predicting next value, the value

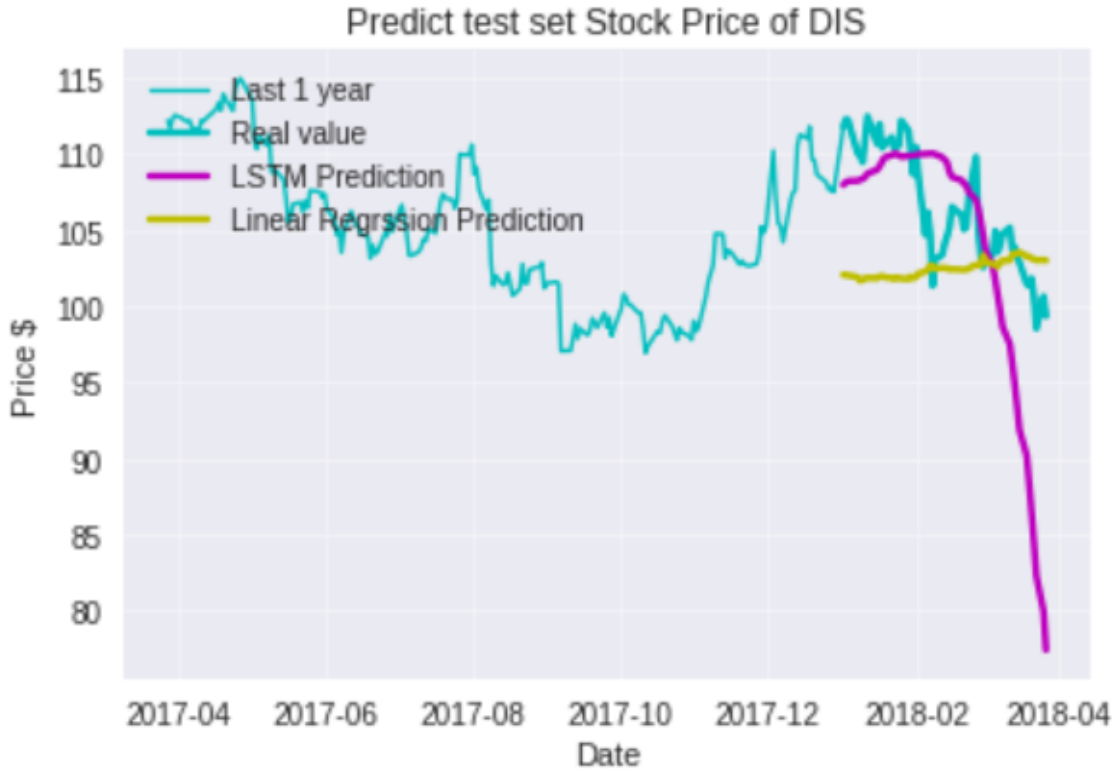
$$\text{De-Normalization: } p_i = p_0(n_i + 1)$$

should be de-normalized by the equation:

1. the predicted result is showing from the /predict path with the graph. To illustrate the figure, the figure which is made from the step 2 saved as the file in the pyplot directory, and there is the API for sending the graph image file for showing the the predicted graph on the /predict page.

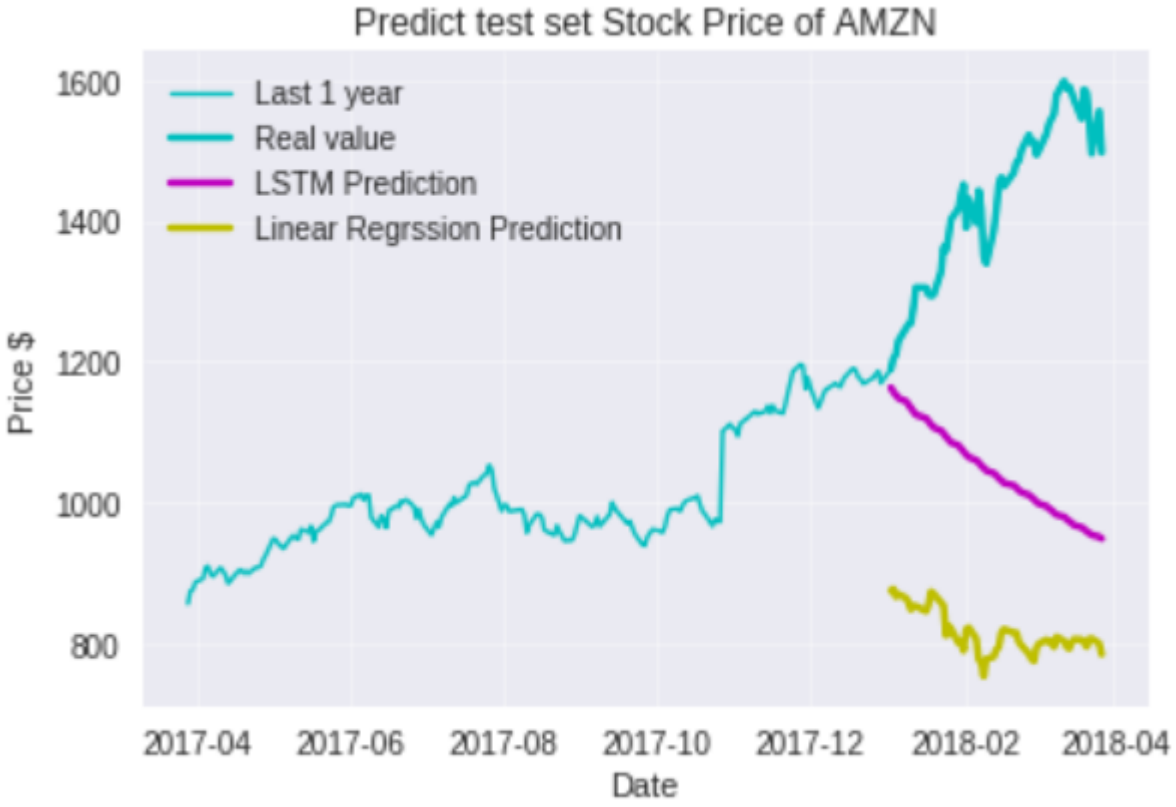
Refinement

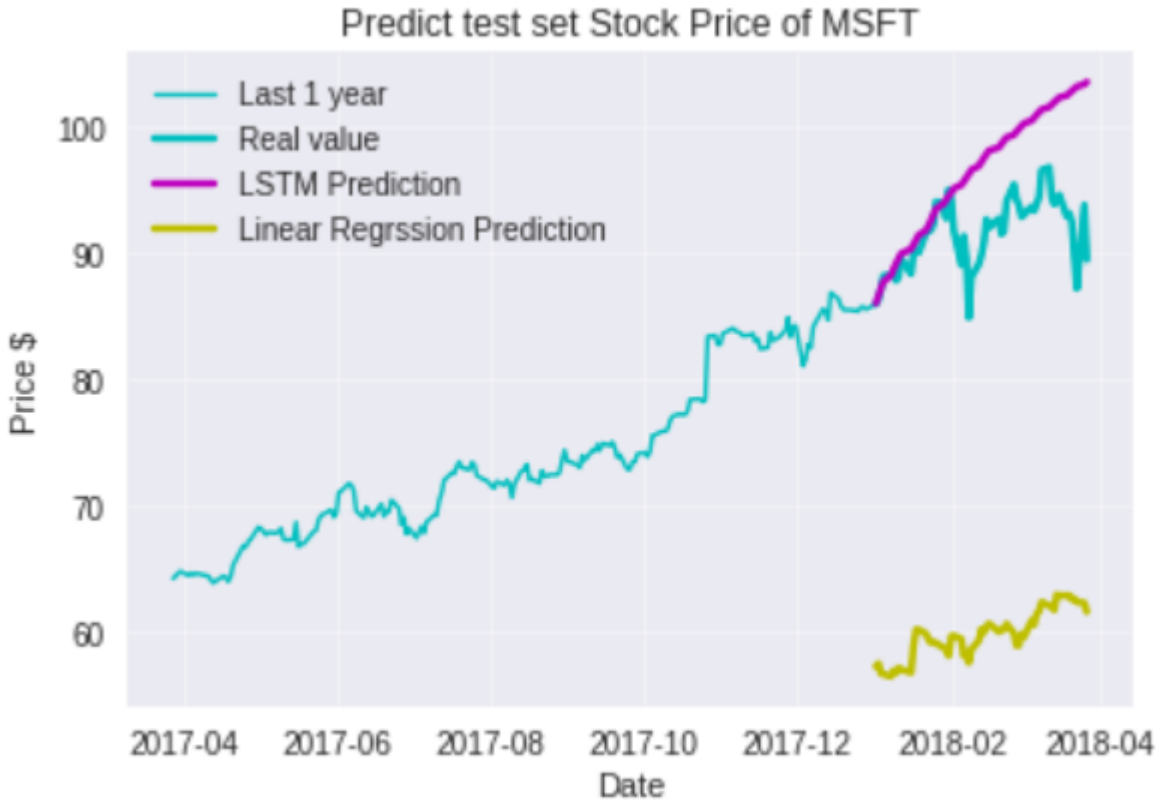
My initial solution was not good. The model seems to predict future prices, but it just looks like stock price graph Here is my first initial solution.

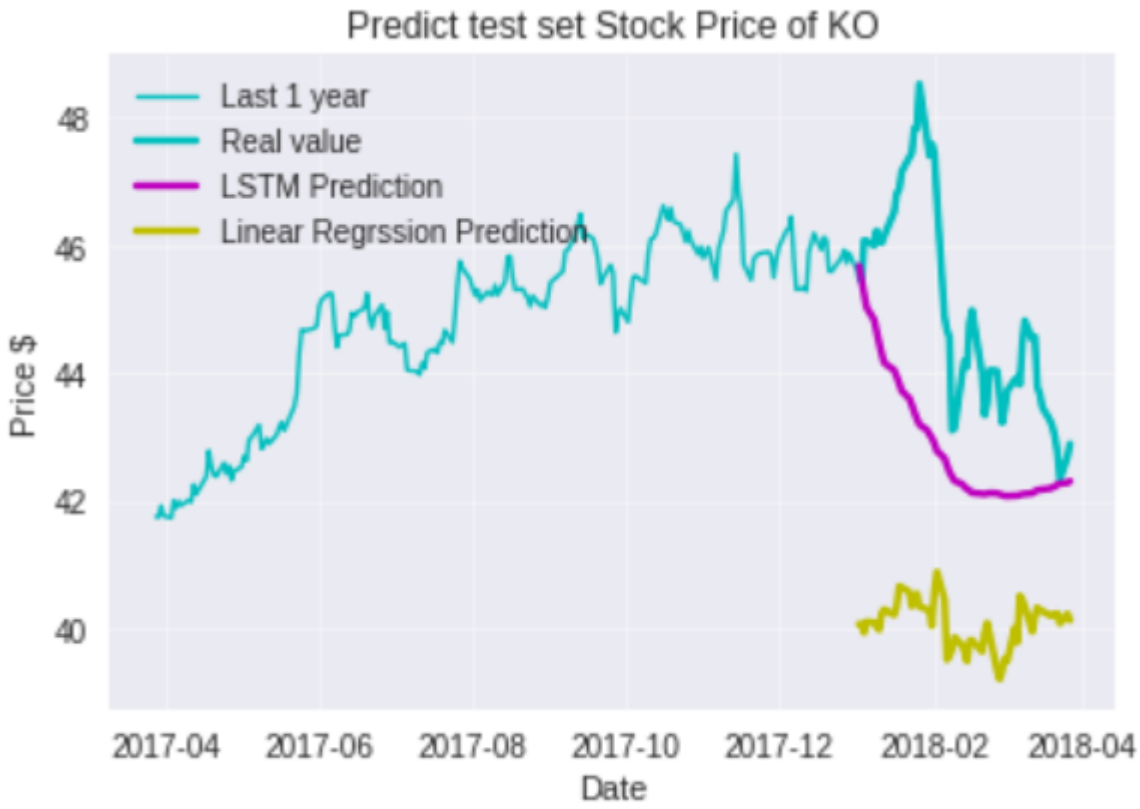


This was improved upon by using the following techniques and process - State
After googling to improve my LSTM model, I found that the stateful LSTM model would have much performance than the model that is not stateful. Here, the stateful means that the LSTM model remembers the current sample trained state and pass the state to the next sample become the initial state of the following sample. So, when using this technique, the model would be trained well. The only things to do is change the shape of batch input from (sequence, time step, features) to (1, time step, feature) and set the stateful parameter to 'True' and loop each training step of LSTM model over sequence size. - increasing window size or each time step of sequence from 50 to 249.
The 50 number is just a guess. And the 249 is some days of the stock market in a year. And I made a hypothesis that if each time step covers the year, then the model would train the pattern much more accurate than a just a random number. - increase the training data to 4 years.
The reason I have done this is similar to the action of increasing window size. I thought that as the training data size the increase, the model, would find the patterns much more comfortable.

After doing this, the LSTM model results like this: Also, here is the stock prediction for rest of the company which is Amazon, Microsoft, and Coca-Cola.


Predict test set Stock Price of AMZN
Last 1 year
Real value
LSTM Prediction
Linear Regrssion Prediction
1600
1400
1200
1000
800
Price $
2017-04
2017-06
2017-08
2017-10
2017-12
2018-02
2018-04
Date


Predict test set Stock Price of MSFT
Last 1 year
Real value
LSTM Prediction
Linear Regrssion Prediction
100
90
80
70
60
Price $
2017-04
2017-06
2017-08
2017-10
2017-12
2018-02
2018-04
Date



The plot shows past prices and both LSTM prediction prices and the benchmark prediction prices. The future values that the model predicted are not look like actual values. The trend of the prediction accuracy differs to the company which means that the dataset influence the efficiency of the LSTM model.

IV. Results

Model Evaluation and Validation

During development, a validation set was used to evaluate the model. Here is the final architecture and hyperparameters I chose: - The number of epochs: 8 - The training data: 4years (250*4 = 1000) - The window size or time step: 249 - The features: 'Adj. Closing' - The layers: one LSTM layer with stateful, 128 memory cell and one dense layer. - The user input ticker is restricted to the given company ticker ['AMZN', 'MSFT', 'DIS', 'KO'] - Likewise, the future days of user input are restricted to the 100 days from now.

Justification

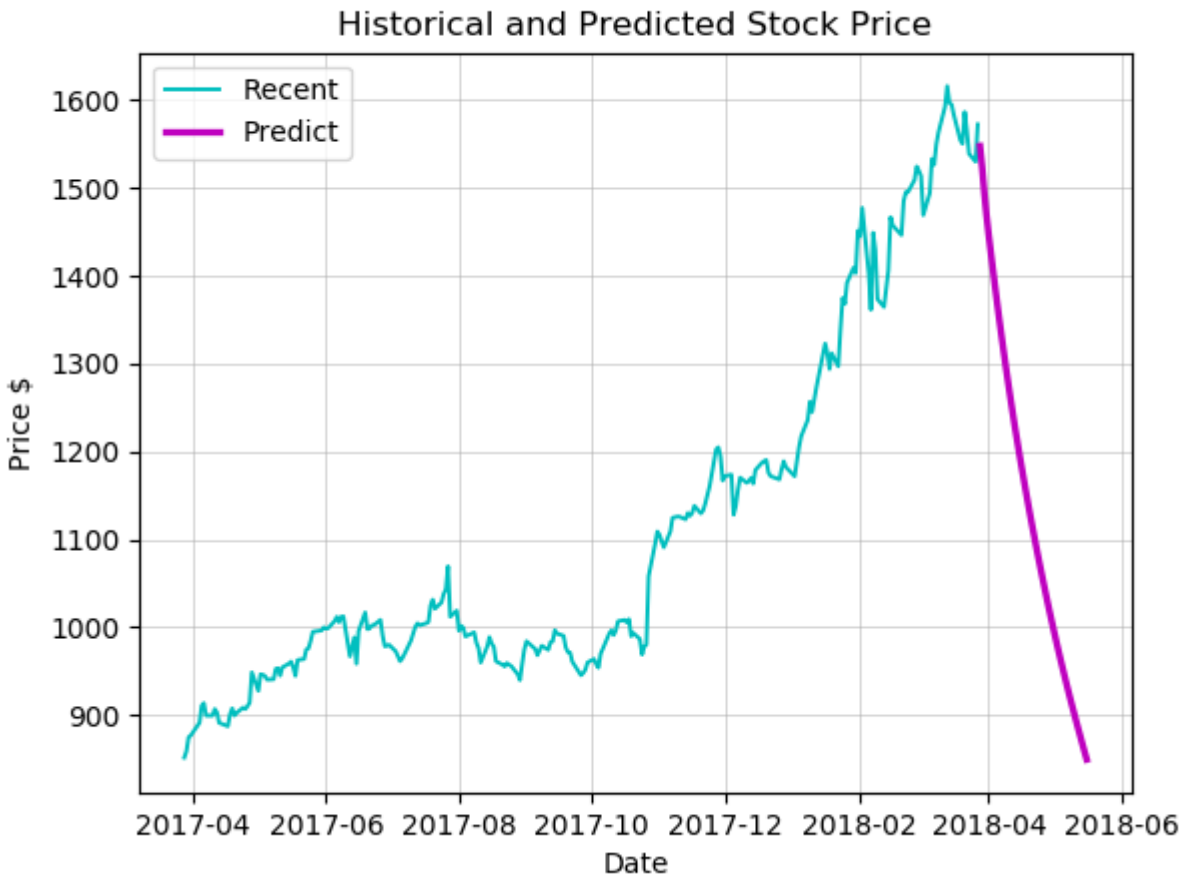
The LSTM model and the benchmark model has a difference. First, the LSTM model predicts rather exact stock price than the trend of the future price. The benchmark model was a Linear Regression model. The model seems like the more actual stock price. But, the absolute value of prediction price is far from the previous prices. I think the reason is that the stock prices keep going up, and the training set was same as the LSTM model, so the benchmark model trained far from the current price. So the predicted values were lower than current price but similar range from the past year data. The LSTM model is not very close to the actual price on the test set but if the model is

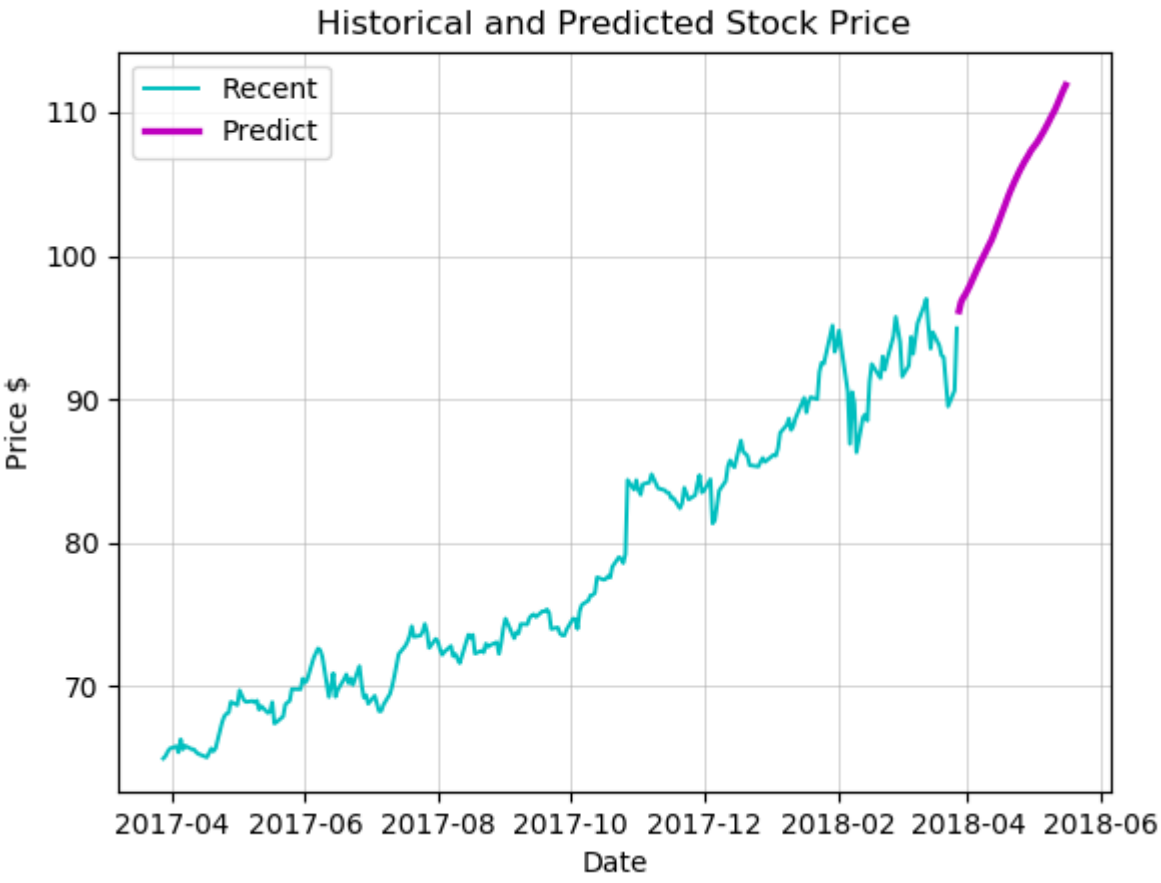
supplemented such as adding unique feature for training, the model prediction would be near perfection.

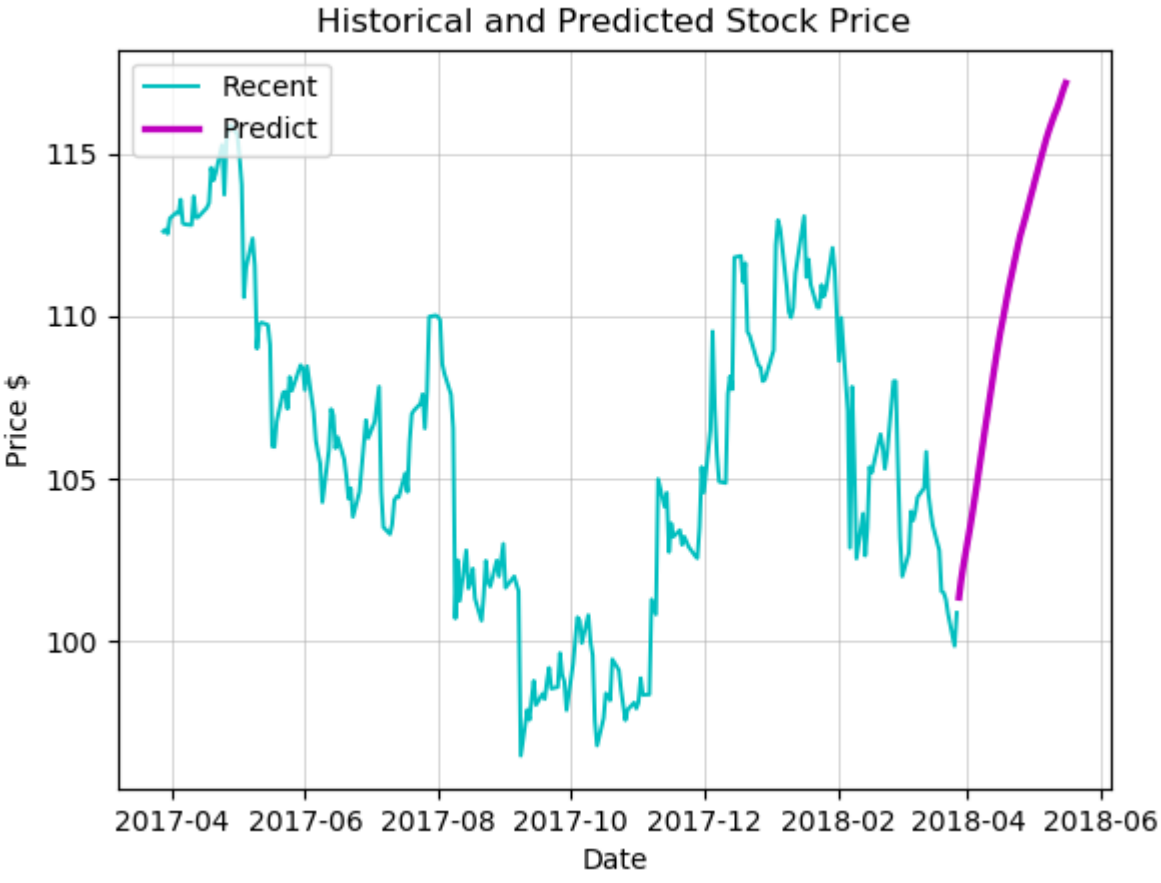
V. Conclusion

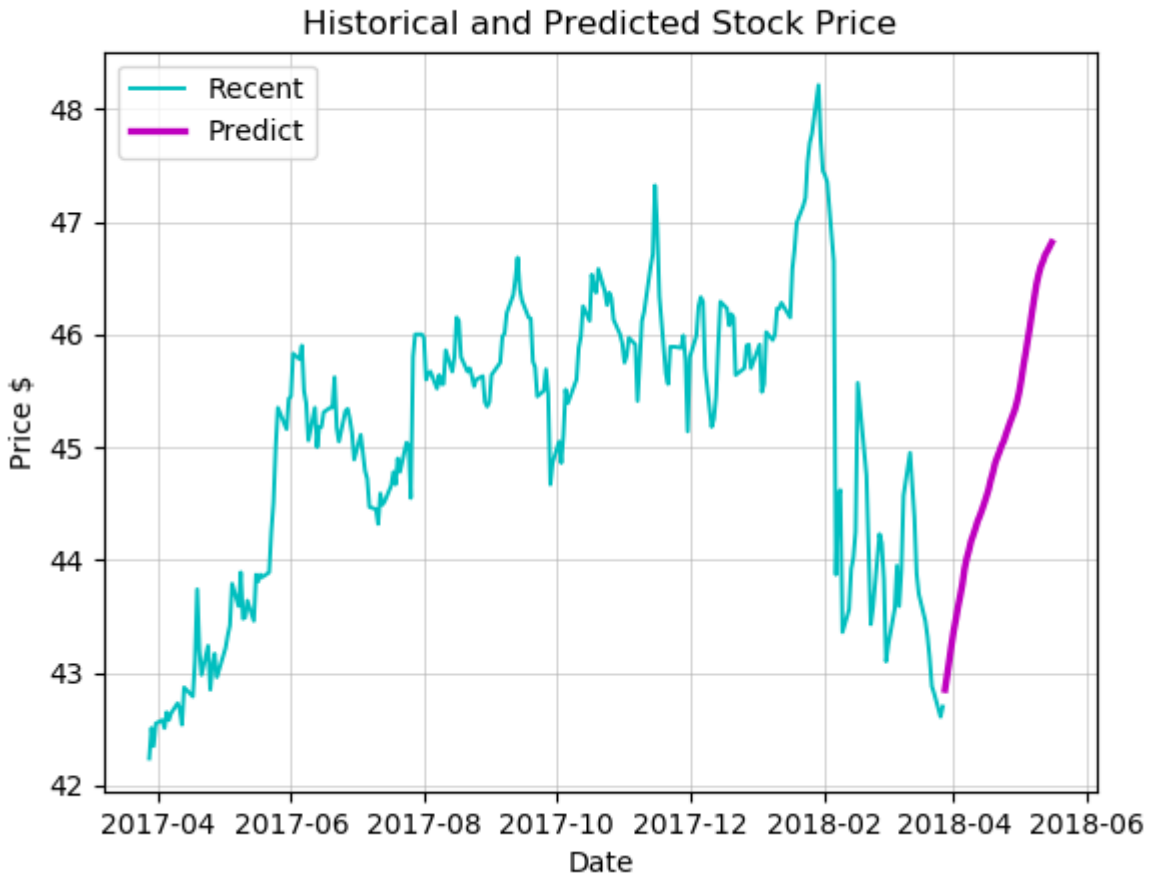
Free-Form Visualization

Here are the prediction graphs of each company on 50 days after the current date.




Historical and Predicted Stock Price
Recent
Predict
110
100
90
80
70
Price $
2017-04
2017-06
2017-08
2017-10
2017-12
2018-02
2018-04
2018-06
Date


Historical and Predicted Stock Price
Recent
Predict
115
110
105
100
Price $
2017-04
2017-06
2017-08
2017-10
2017-12
2018-02
2018-04
2018-06
Date



Reflection

First of all, the stock market prediction field keeps in my head while learning machine learning. That much, the stock market prediction is a fascinating domain for me. But, it was a very tough way for me that I didn't learn about the LSTM in the machine learning Nanodegree program. Since the stock prediction is hard for using a simple model such as Linear Regression, SVM, or just neural network that I learned from the lectures, I try to choose a well-known model for time series prediction which is LSTM model. Though I did not learn the LSTM model from the Nanodegree program, I have to spend most of the time researching the concept of the model, how to implement, etc. while doing my capstone project. That was not a good idea because the time was ticking and I rarely have time for finding more suitable hyperparameters, features, etc. The problem was very challenging, but It was also very proud that I implemented the model that predicts what I want to see. But there has left many things to improve.

Improvement

To achieve a more accurate model, it would be good to add or experiment below features: - Follow the change point of the given dataset. The change point matters because it shows the trend of the stock prices - try to use more feature to train the model. - do the sentimental analysis about the company that the user wants to predict price on and assist this result in predicting stock price.