**Homework Report**

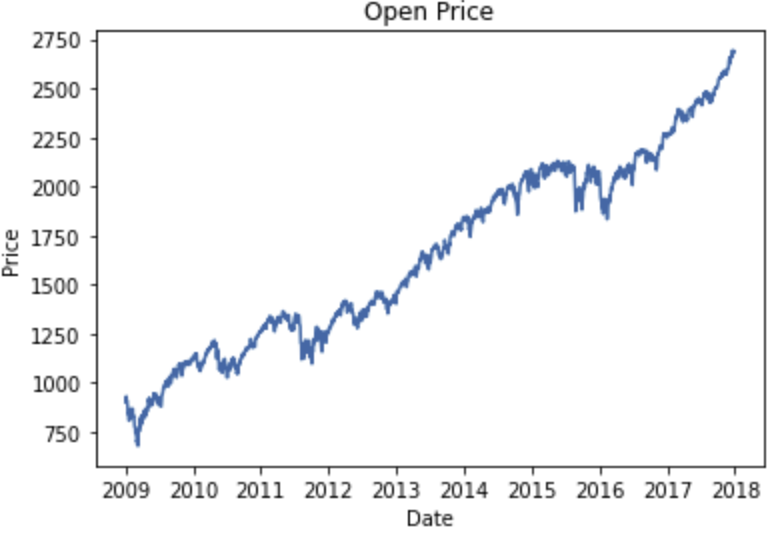
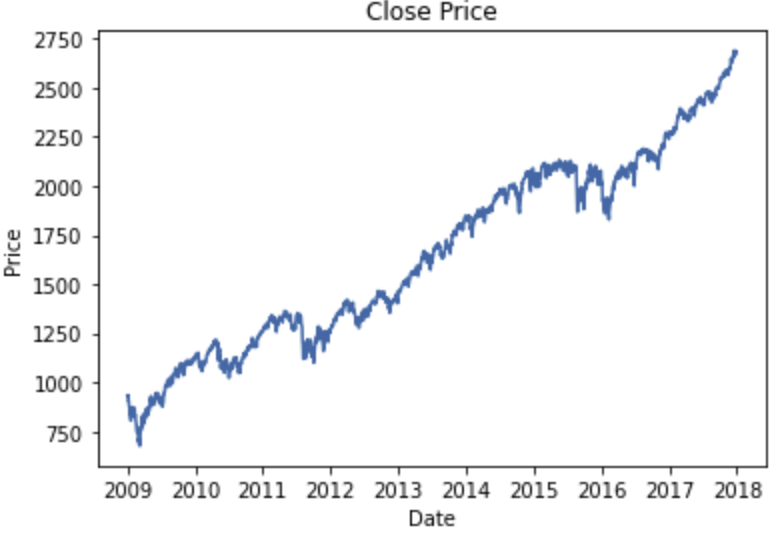
**Stock Movement Prediction (Open Price)**

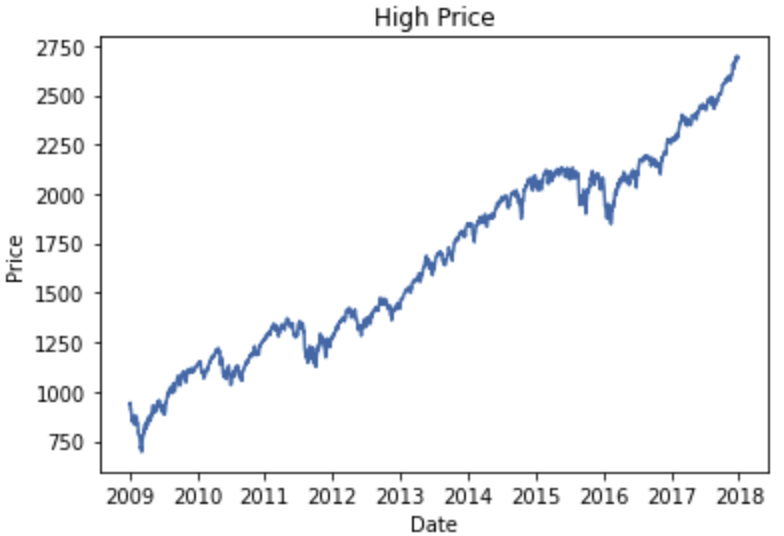
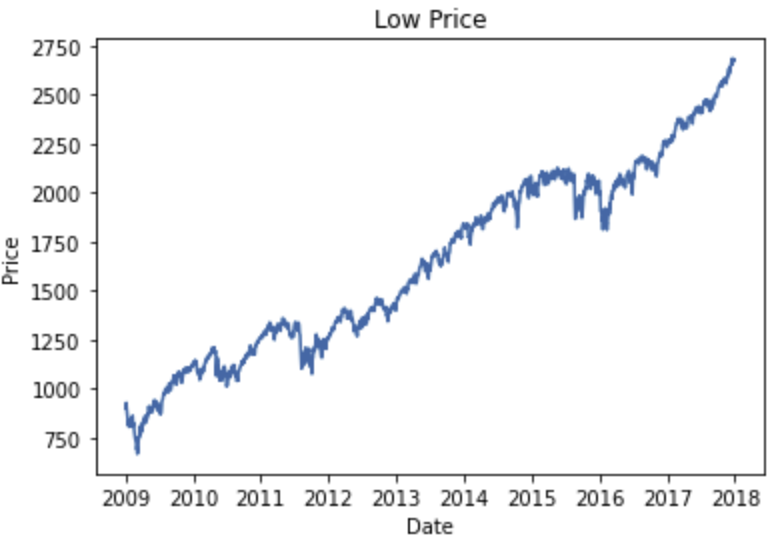
Classifiers used:

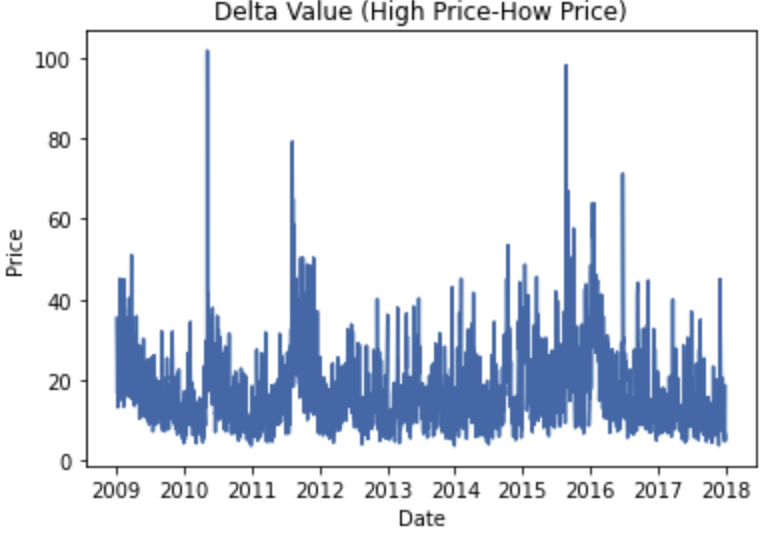
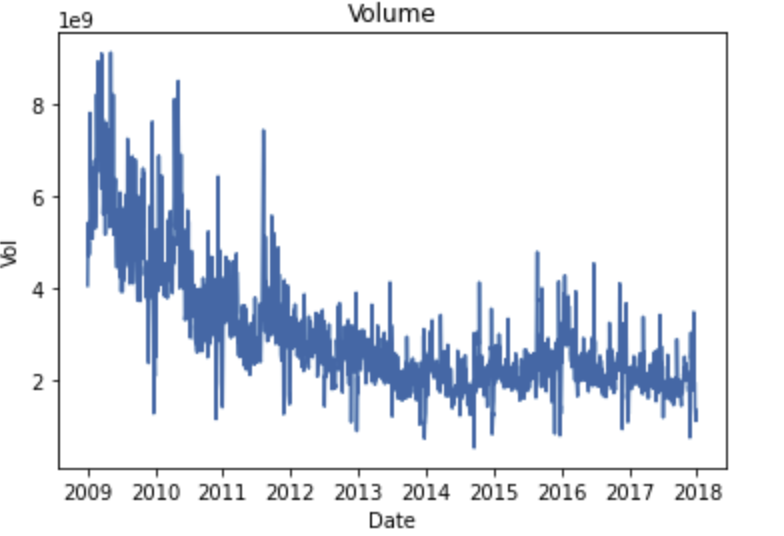
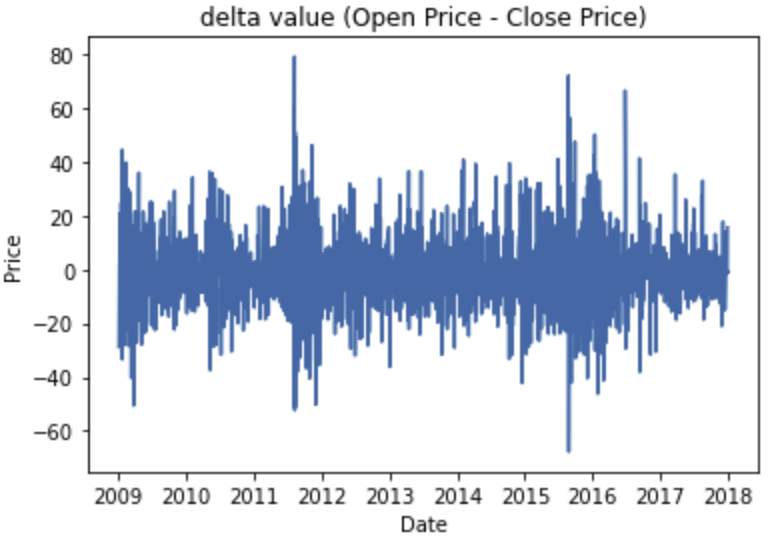
1. Facebook Prophet
2. LSTM
3. Stacked LSTM

**Data Visualization – understanding**

First, I plotted the training data to see if there is some kind of trend occurring in the dataset.

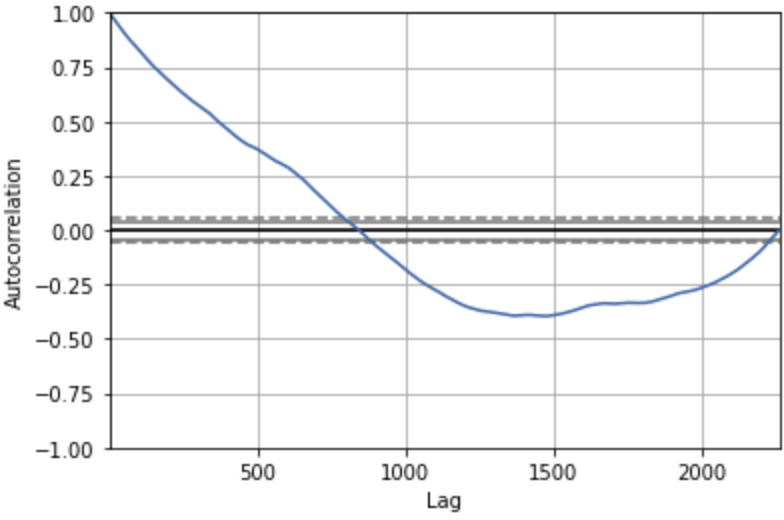
 

Observations:

* Here we can notice that Open Price & Close Price are very similar and diference in stock price would not drop or increase for more than $100 in a single day.
* The dataset has a upward trend
* Volume seems to be inversely proportional to Stock price ( each stock gets more expensive so with the same ammount of money you will only afford for few stock)

**Autocorrelation**



* This tells us that the closest 100 data values has very high correlation, meaning that if we choose the look back numbe < 100 the results may be optimal.

**Metrics:**

* For this homework I will use Root Mean Square Error as Metric for all the models
* Since RMSE is one of the mostly used to represent time series forecasting model performance

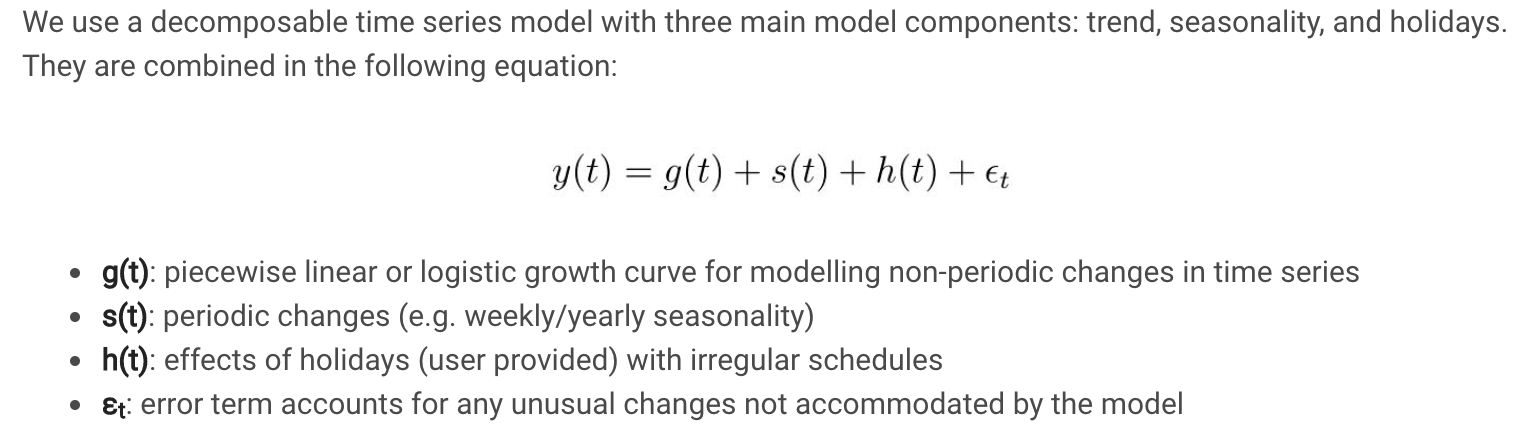
**Facebook Prophet (regression model)**

**Why Facebook Prophet?**

After surveying many statistical models such as Moving Average, Auto Regression, ARIMA, Exponential Smoothing. Facebook prophet seems to be the optimal, it takes into account all the components mentioned above.

Another aspect is that prophet is easy to use, only need to provide a pandas data frame with specific column names.

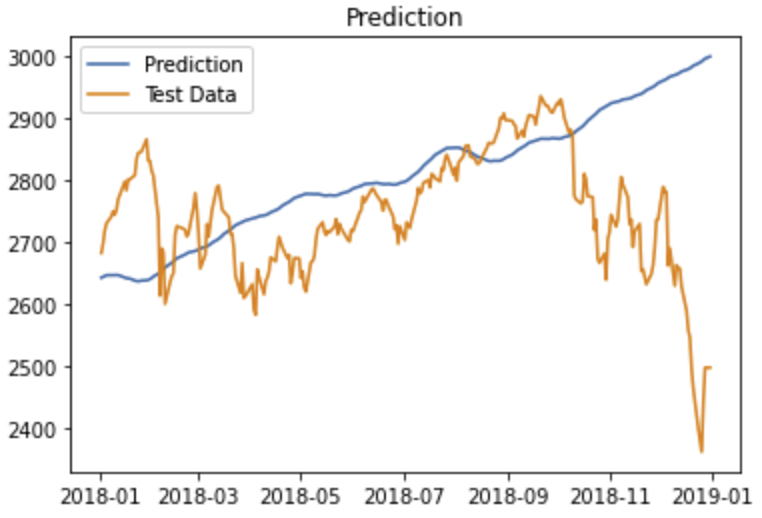




[link](https://www.analyticsvidhya.com/blog/2018/05/generate-accurate-forecasts-facebook-prophet-python-r/#:~:text=Prophet%20is%20an%20open%20source,of%20custom%20seasonality%20and%20holidays!)-> link of the images clipped

**Results:**

* RMSE: 155.72



Comments:

* we can see that the model has learn the upward trend of the data, but it hasn’t learned the details of the dataset.

**Long Short-Term Memory (LSTM)**

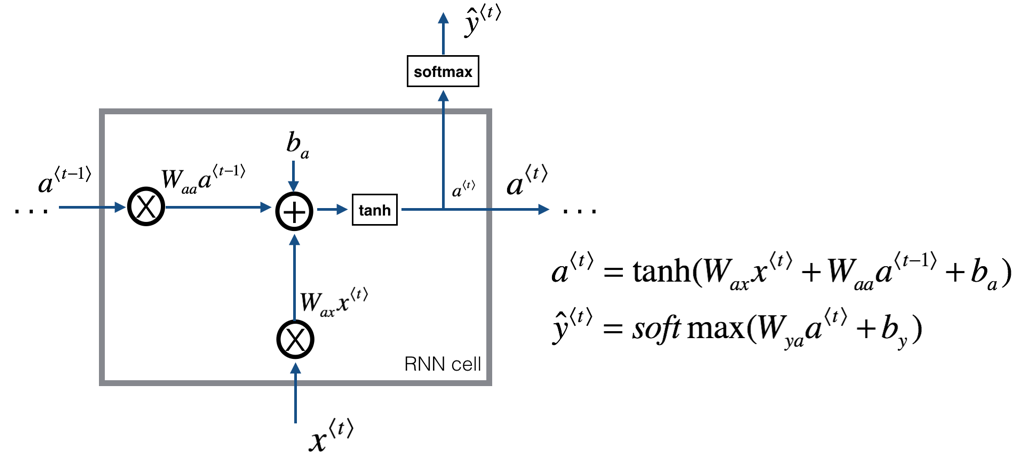
**Why LSTM?**

* On the RNN family LSTM is mostly known to have top performance on time series data.
* It solves the problems RNN has, RNN only looks or recalls the most recent data but forgets about data seen before. LSTM solves this problem by adding some submodules to choose which data to remember and which to forget.
* Objective is to input some past days data to predict the price of the coming day,  
  look back day number can be adjusted

Data Preprocessing:

* Normalizing data
* Creating look back sequence by sliding the input array

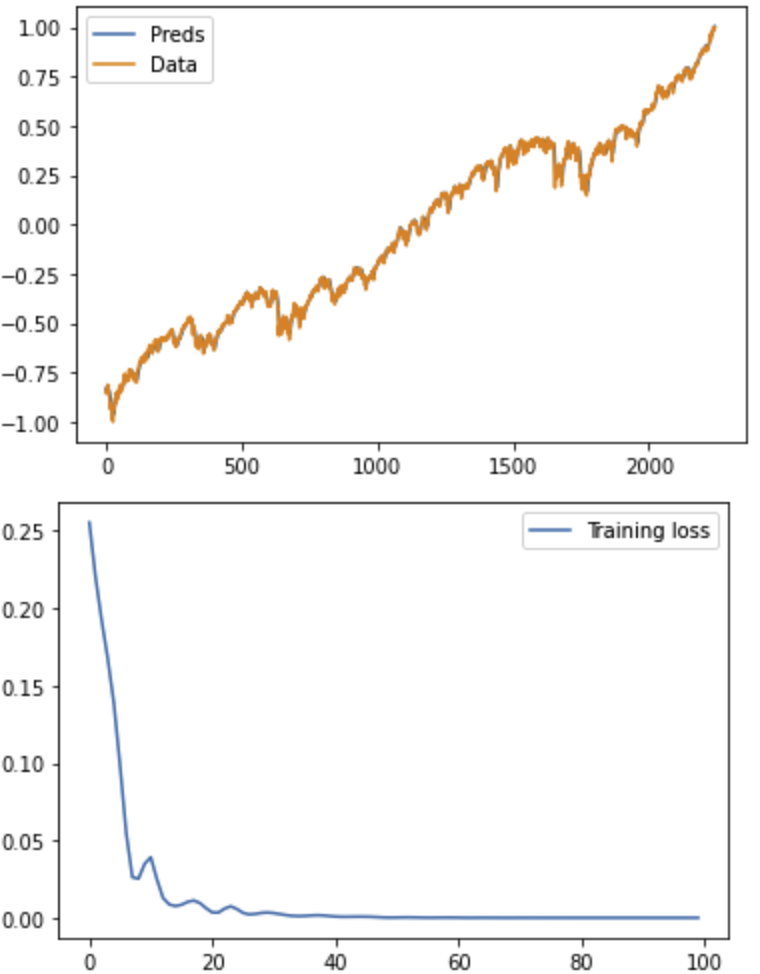
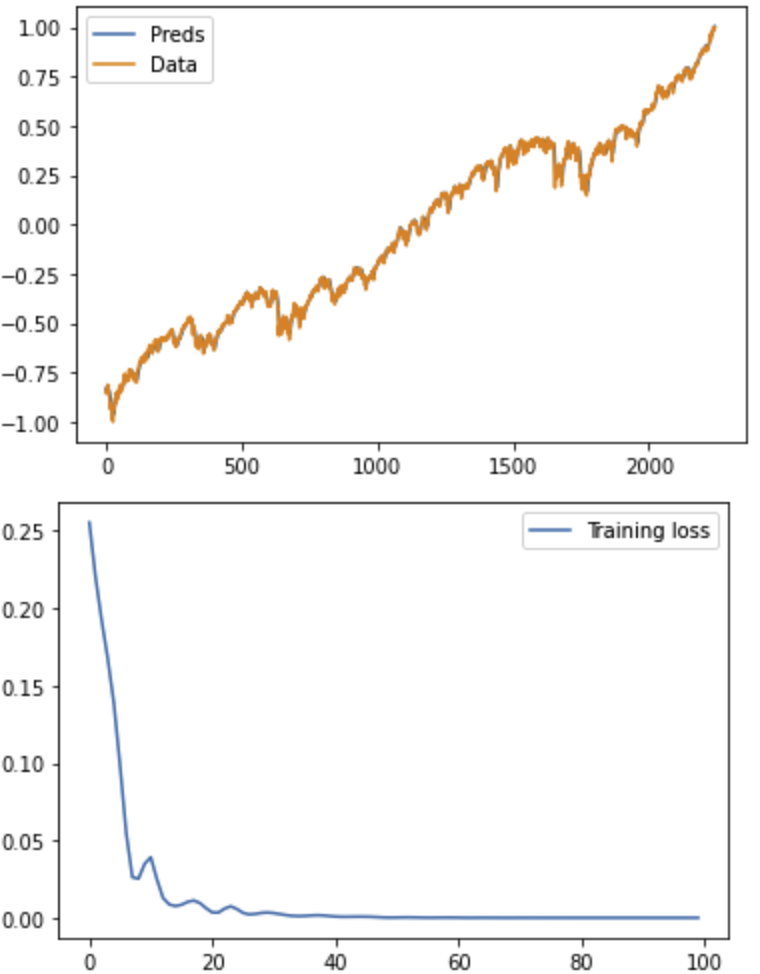
Model



* Single layer LSTM with 32 outputs followed by linear layer (32,1 )

Results:

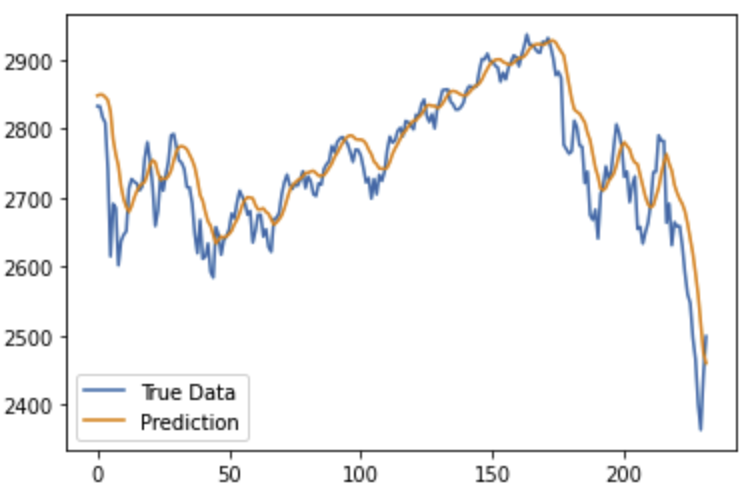
Training



Training data completely fits the labels, loss converges around epoch 50 can be adjusted during training can still notice loss dropping by very few.

Testing

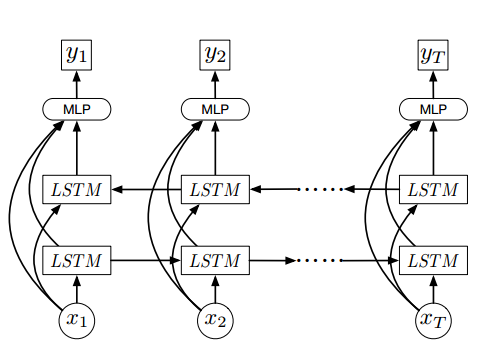
RMSE: 48.26



The predictions are pretty accurate, we can notice some horizontal shifts.

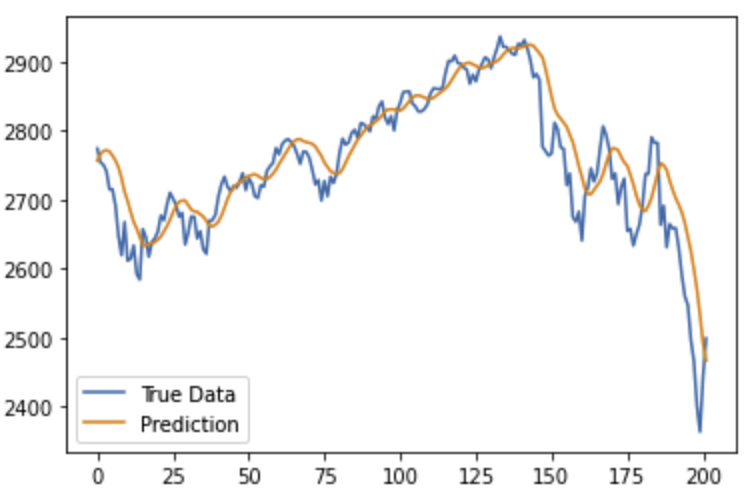
**Stacked LSTM**

Data preprocessing is very similar to LSTM, we just modify the model to have multiple LSTM layers



Results:

RMSE: 47.55



Model Performance

|  |  |  |  |
| --- | --- | --- | --- |
|  | Prophet | LSTM | Stacked LSTM |
| RMSE | 155.72 | 48.26 | 47.55 |

* It is expected that Stacked LSTM outperform the previous models
* Stacked LSTM has more layers so it can predict more complex behaviors
* Facebook prophet is a regression model it would perform better in a more stationary and cyclical data.
* This result may not vary much changing the dataset since prophet isn’t as complex as LSTM models it is expected that LSTM models have a better performance.

Data Processing and Hyper Parameters tuning

Here I will list the most crucial data preprocessing and hyper parameters tuned that can affect the model accuracy the most:

* Data normalization (can be (0,1) or (-1,1)) both works
* look back (LB), if LB is set too small the predicted values are noisier, on the other hand if LB is set too large the predicted value will be stationary close to moving average
* layers stacked, too much will cause model to overfit
* hidden dimensions, too much will cause model to overfit
* number of epochs, we can see that loss converges around epoch 50 but when I set the epoch around 50 the prediction trend seems to go the other way.