Univariate Data Analysis Of Stock Prices using Facebook Prophet Model

By Foo ZheShen

Introduction

What is Univariate?

Univariate is a term commonly used in statistics to describe a type of data which consists of observations on only a single characteristic or attribute such as a time series data.

We can use stock prices as an example since the Closing price of tomorrow is independent and cannot be predicted using other attributes like on the day’s high and low, since those attributes hasn’t occur yet or are still unknown. That is why many of the prediction models of stock prices that can be found online are misleading since they require other independent attributes to make predictions.

What is Prophet?

Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects.

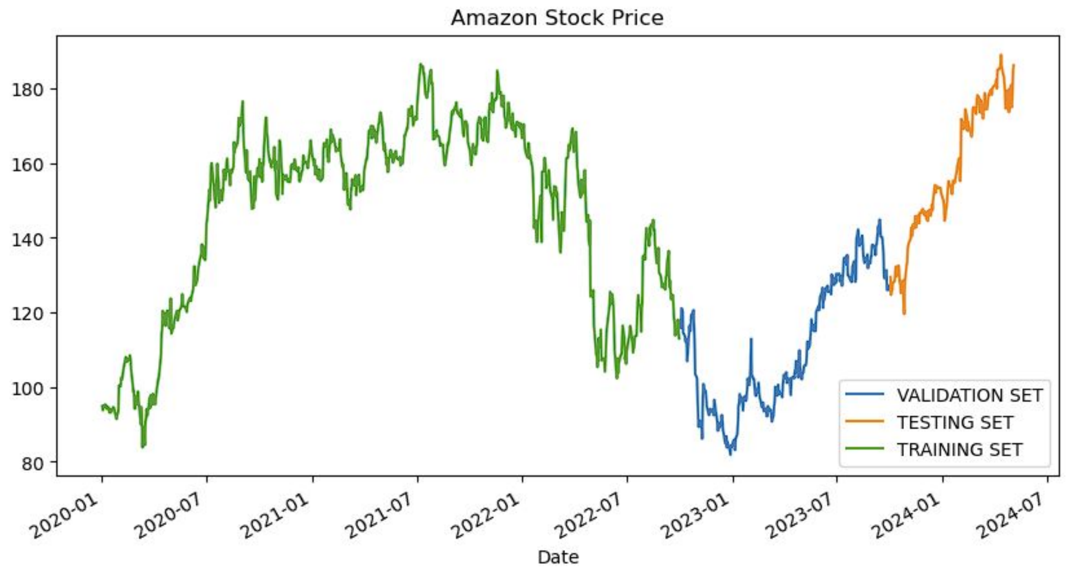
Hence, we will be using univariate analysis and Facebook’s Prophet model to understand the characteristics of the stock's price movements over a certain period of time.

Data Exploration

For this analysis we are going to split the data into 3 parts.

Training, Validation, and Testing sets.

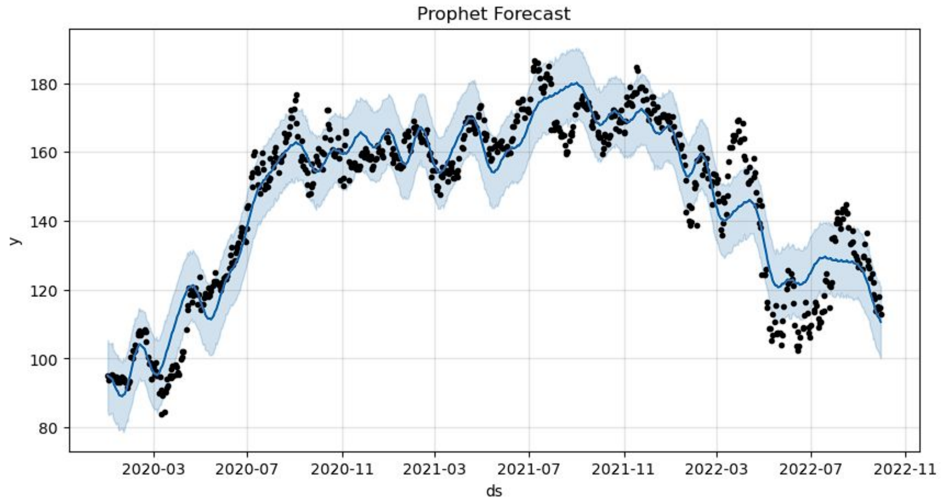
1. Training to fit the model
2. Validation to fine tune model parameters
3. Testing for our forecast analysis



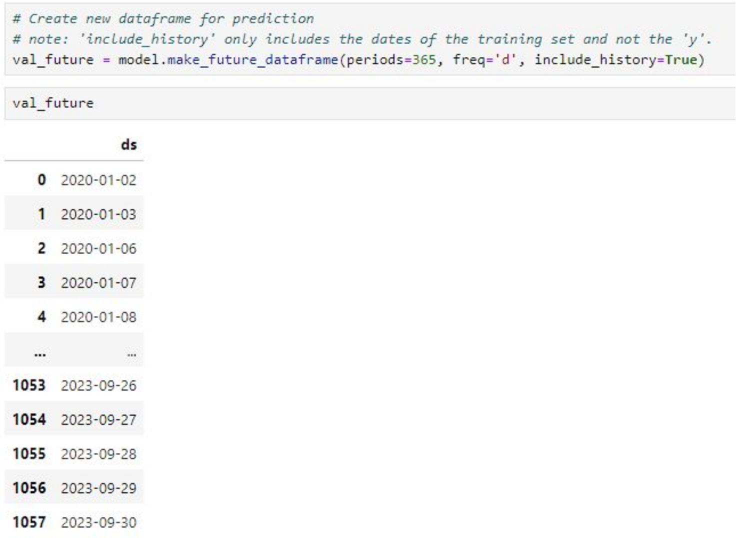
Fit and Training First Model

Default model without adjustments:

```
model = Prophet()
df_train_fcst = model.fit(df_train).predict(df_train)
```



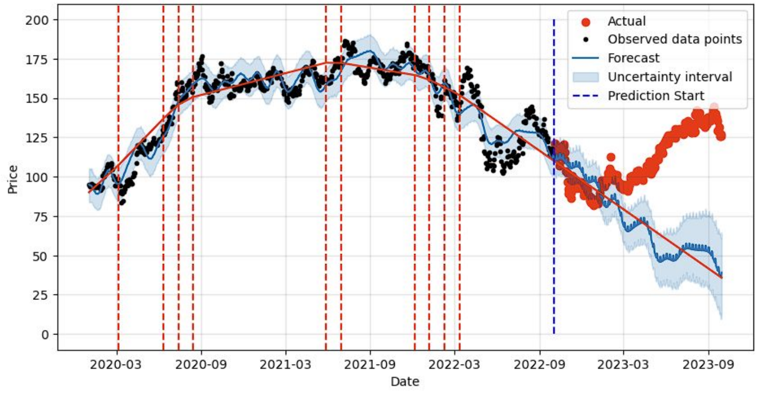
Validation



```
# Create new dataframe for prediction
# note: 'include_history' only includes the dates of the training set and not the 'y'.
val_future = model.make_future_dataframe(periods=365, freq='d', include_history=True)
```

```
val_future
```

	ds
0	2020-01-02
1	2020-01-03
2	2020-01-06
3	2020-01-07
4	2020-01-08
...	...
1053	2023-09-26
1054	2023-09-27
1055	2023-09-28
1056	2023-09-29
1057	2023-09-30



Changepoints

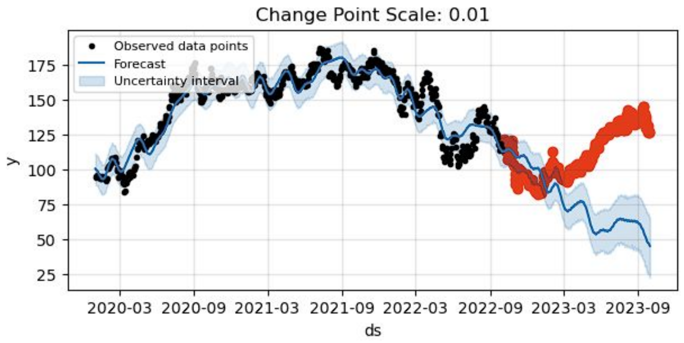
In real time series, we often find them frequently having abrupt changes in their trajectories.

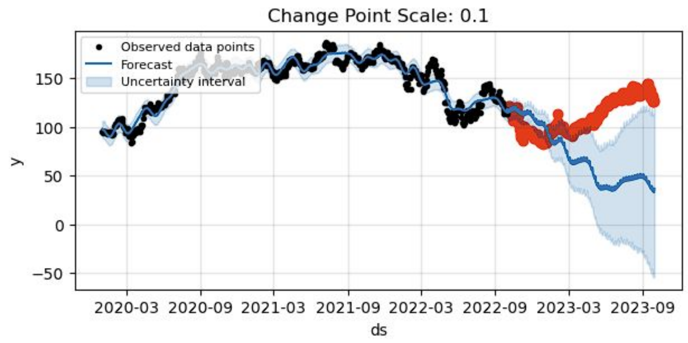
If the trend changes are being overfit (too much flexibility) or underfit (not enough flexibility), we can adjust the strength of the sparse prior using the input argument changepoint_prior_scale.

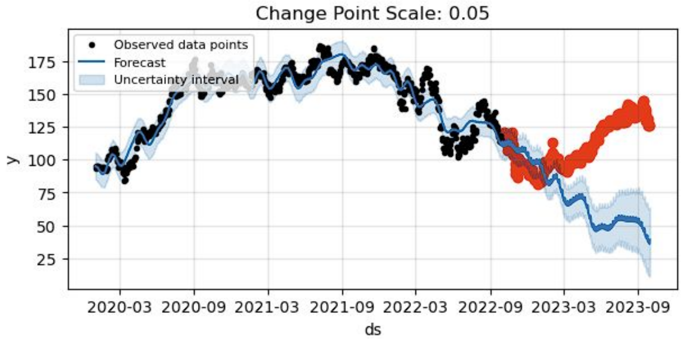
Prophet will automatically detect these changepoints and will allow the trend to adapt appropriately. By default, this parameter is set to 0.05. Increasing it will make the trend more flexible and vise versa.

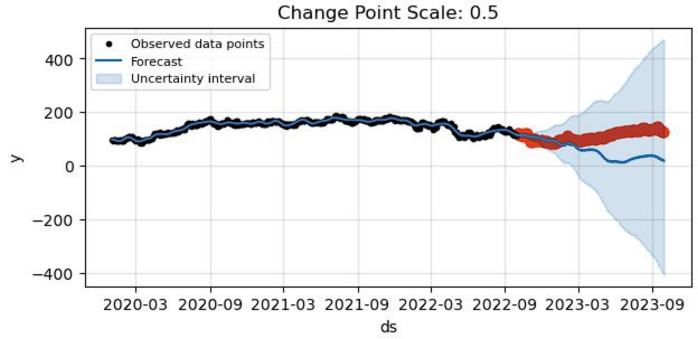
Changepoints Scales

We can visualize how different changepoint scales affect the model.









Other Parameters

Seasonality:

Seasonalities are estimated using a partial Fourier sum.

Just like changepoints, there is a parameter, seasonality_prior_scale, which can similarly adjusts the extent to which the seasonality model will fit the data.

By default this parameter is 10, which provides very little regularization. Reducing this parameter dampens the effects.

Additive & Multiplicative:

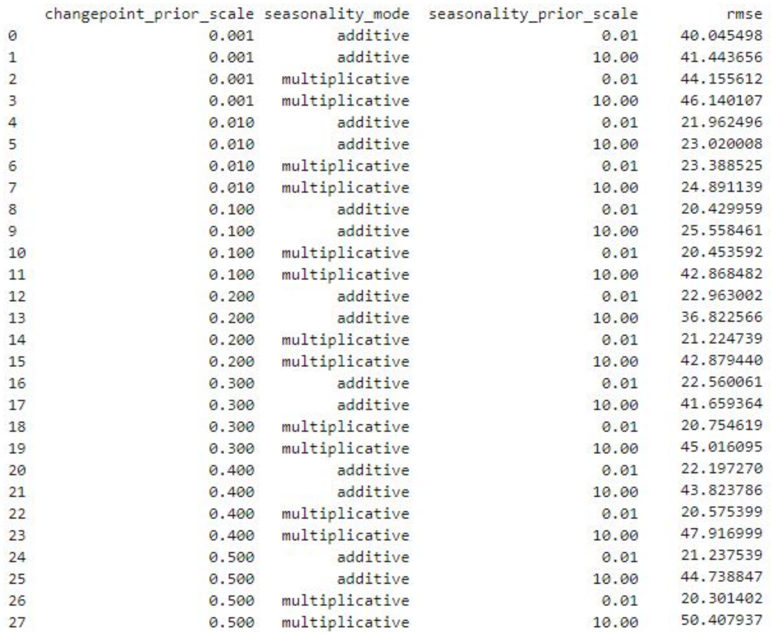
The additive model is useful when the seasonal variation is relatively constant over time.

The multiplicative model is useful when the seasonal variation increases over time.

Extra regressors are put in the cross validation of the model, so that we can find if the model for this time series depends on the extra regressor as either an additive or multiplicative factor.

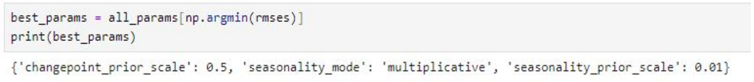
Cross Validation

For this project, cross validation is performed using 366 days initial training with 90 day horizons on 45 day intervals.



```
    changepoint_prior_scale seasonality_mode  seasonality_prior_scale       rmse
0                     0.001         additive                     0.01  40.045498
1                     0.001         additive                    10.00  41.443656
2                     0.001   multiplicative                     0.01  44.155612
3                     0.001   multiplicative                    10.00  46.140107
4                     0.010         additive                     0.01  21.962496
5                     0.010         additive                    10.00  23.020008
6                     0.010   multiplicative                     0.01  23.388525
7                     0.010   multiplicative                    10.00  24.891139
8                     0.100         additive                     0.01  20.429959
9                     0.100         additive                    10.00  25.558461
10                    0.100   multiplicative                     0.01  20.453592
11                    0.100   multiplicative                    10.00  42.868482
12                    0.200         additive                     0.01  22.963002
13                    0.200         additive                    10.00  36.822566
14                    0.200   multiplicative                     0.01  21.224739
15                    0.200   multiplicative                    10.00  42.879440
16                    0.300         additive                     0.01  22.560061
17                    0.300         additive                    10.00  41.659364
18                    0.300   multiplicative                     0.01  20.754619
19                    0.300   multiplicative                    10.00  45.016095
20                    0.400         additive                     0.01  22.197270
21                    0.400         additive                    10.00  43.823786
22                    0.400   multiplicative                     0.01  20.575399
23                    0.400   multiplicative                    10.00  47.916999
24                    0.500         additive                     0.01  21.237539
25                    0.500         additive                    10.00  44.738847
26                    0.500   multiplicative                     0.01  20.301402
27                    0.500   multiplicative                    10.00  50.407937
```

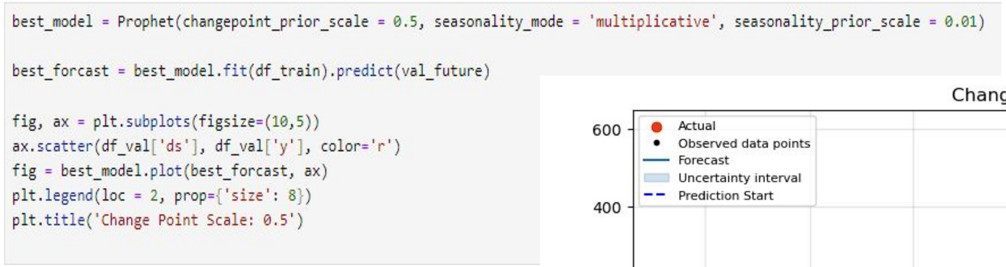
After performing cross validation checks for the parameters, we can start building our final model.



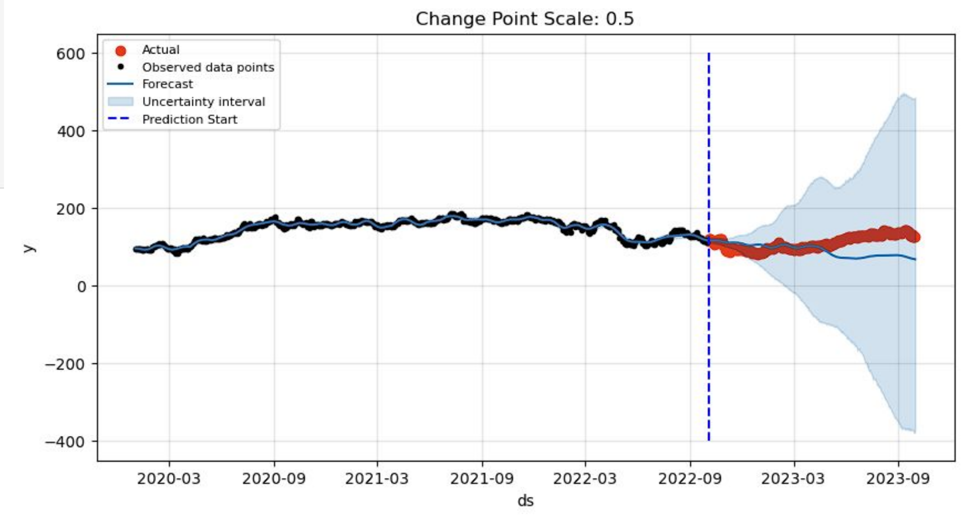
```
best_params = all_params[np.argmin(rmses)]
print(best_params)
```

```
{'changepoint_prior_scale': 0.5, 'seasonality_mode': 'multiplicative', 'seasonality_prior_scale': 0.01}
```

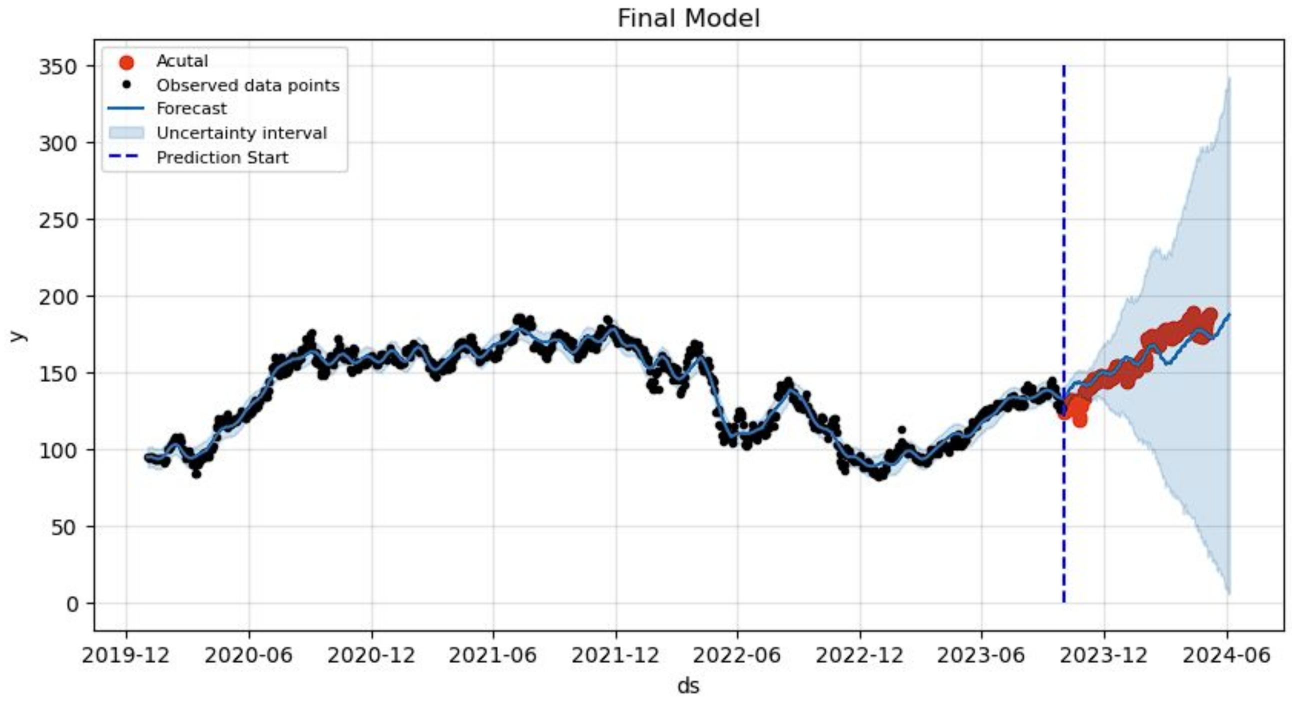
Building the Best Model



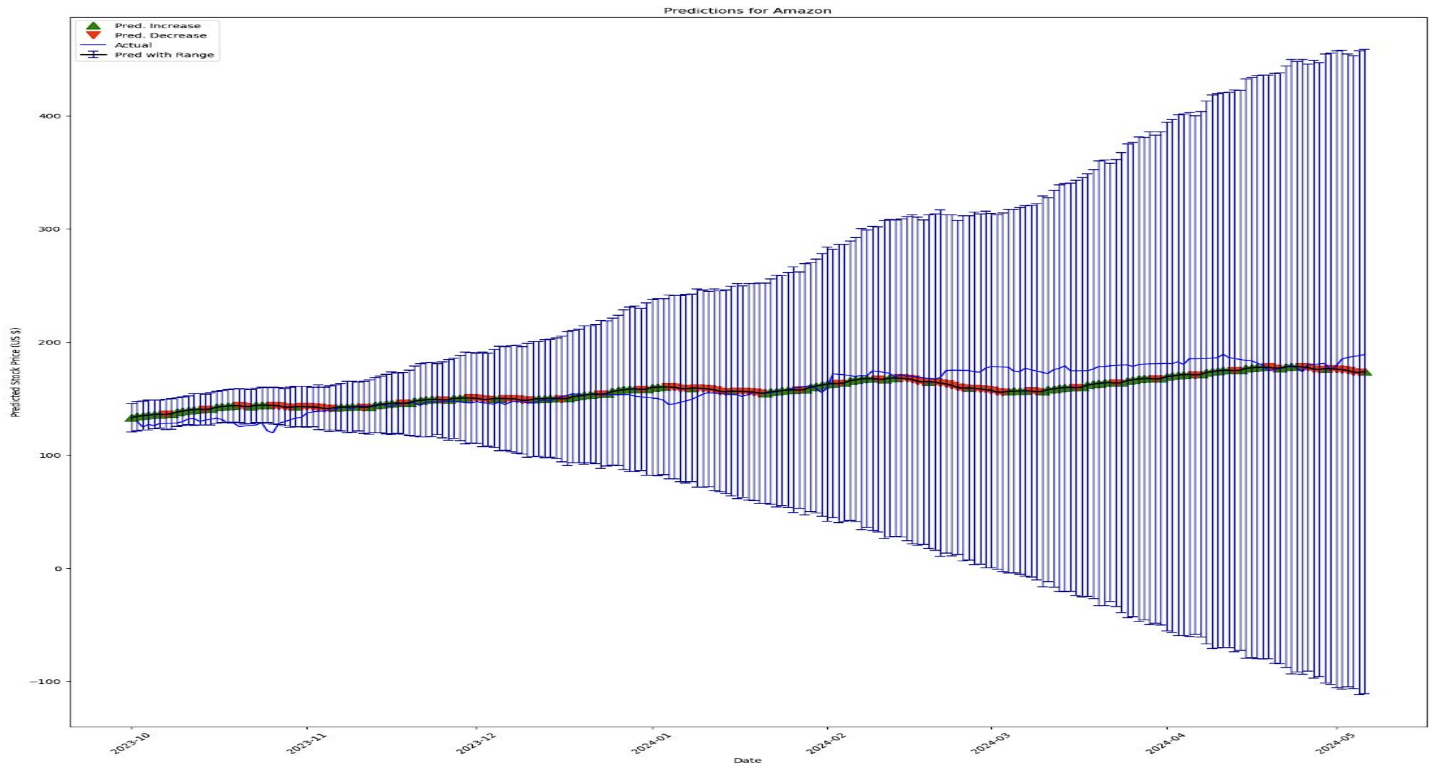
```
best_model = Prophet(changepoint_prior_scale = 0.5, seasonality_mode = 'multiplicative', seasonality_prior_scale = 0.01)

best_forcast = best_model.fit(df_train).predict(val_future)

fig, ax = plt.subplots(figsize=(10,5))
ax.scatter(df_val['ds'], df_val['y'], color='r')
fig = best_model.plot(best_forcast, ax)
plt.legend(loc = 2, prop={'size': 8})
plt.title('Change Point Scale: 0.5')
```



Re-Training Best Model



Final Prediction



Final Thoughts

From the charts, the model seems to be able to be able to detect the trend movement of the stock price well within the confidence interval.

The forecast for the final prediction and actual stock price has a mean error of 7.60 USD.

Depending on the kind of trading strategy, this could be a very impressive forecast for long-term holding strategy. However, for short-term trading, the accuracy error of predicted price, although small in number, could still pose a huge risk on investment.

Further model adjustments can be made to include global and local market news and earning reports.