Development part 1

Stock price prediction

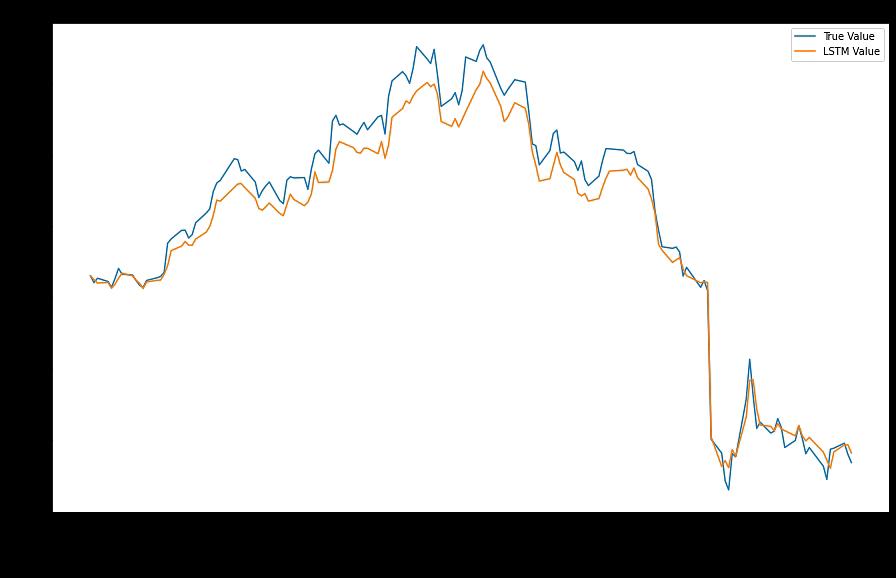
Introduction:

Stock market prediction is the act of trying to determine the future value of a company stock or other financial instrument traded on an exchange. The successful prediction of a stock’s future price could yield significant profit.

Machine Learning Stock Prediction Project

Nevertheless, we can check the performance of our model on a test set as below.

Y\_pred = 1stm.predict(x\_test)

 To evaluate, first, we plot the curve for true values and overlap it with that for the predicted values.

Thus, we can see that LSTM can emulate the trends of the stock prices to a certain extent. Based on the recent dip in prices, it has also fit the dropping curve well.As we decided earlier, we can also check the RMSE and MAPE values to evaluate the performance. We will use these values for future comparison.

1. rsme = mean\_squared\_error(y\_test, y\_pred, squared-False)
2. mape = mean\_absolute\_percentage\_error(y\_test, y\_pred)
3. print("RSME: ", rmse)
4. print("MAPE: ", mape)

RSME: 0.16808551269723027

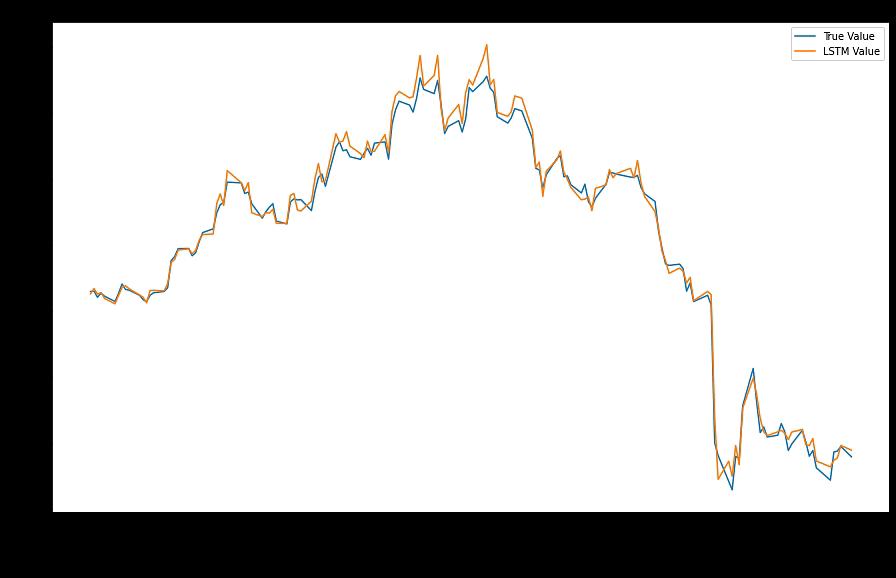
MAPE: 0.12614061132643917

L et’s try to get better results with the same dataset but a deeper LSTM model.

1. 1stm= Sequential()
2. 1stm.add(LSTM(50, input\_shape=(x\_train.shape[1], X\_train.shape[2]),activation='relu', return\_sequences=True))
3. 1stm.add(LSTM(50, activation='relu'))
4. 1stm.add(Dense(1))
5. 1stm.compile(loss='mean\_squared\_error', optimizer='adam')
6. 1stm.summary()

We added another LSTM layer and increased the number of LSTM units per layer to 50.

While the loss still converges early, the curve is better fitted to the true value.



Moreover, the RMSE and MAPE values are better too.

1. rmse = mean\_squared\_error(y\_test, y\_pred, squared=False)
2. mape = mean\_absolute\_percentage\_error(y\_test, y\_pred)
3. print(“RSME: “, rmse)
4. print(“MAPE: “, mape)

RSME: 0.0710094097230204

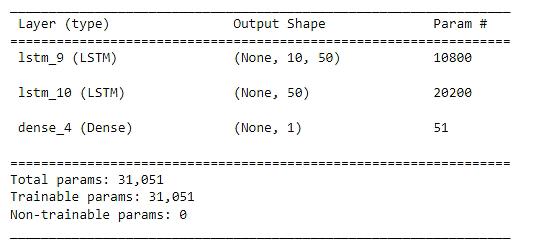
MAPE: 0.058775797917331896

Thus we observe substantial improvement by adding another LSTM layer to the model. However, further adding even more layers would not be fruitful as the model might overfit or stagnate during training.

Now we will try fitting the same model but with increased time steps. We’ll try for n\_steps=10.We change the value in the block below and rerun the entire process with the same model as before.

1. n\_steps=10
2. X1, y1 = 1stm\_split(stock\_data\_ft.values, n\_steps=n\_steps)
3. train\_split-0.8
4. split\_idx= int(np.ceil(len(X1)\*train\_split))
5. date\_index = stock\_data\_ft.index
6. X\_train, X\_test = X1[:split\_idx], X1[split\_idx:]
7. y\_train, y\_test = y1[:split\_idx], y1[split\_idx:]
8. X\_train\_date, X\_test\_date = date\_index[:split\_idx], date\_index[split\_idx:-n\_steps] 11
9. print(X1. Shape, X\_train. Shape, X\_test.shape, X\_test\_date.shape, y\_test.shape)

(747, 10, 3) (598, 10, 3) (149, 10, 3) (149,) (149,)

The model we used above