Machine Learning Project

Stock Analysis using LSTM Model

Due on May 15th, 2021

CS559A – Spring

Prof. Xinchao Wang

Ameya Yadav
(10459869)
Omkar Sinha
(10468312)

ABSTRACT

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) capable of learning order dependence in sequence prediction problems. This is a behavior required in complex problem domains like machine translation, speech recognition, and more. LSTMs are a complex area of deep learning. In this project, we will get insight into LSTMs using the stock analysis of Microsoft Company.

CASE STUDY

We are implementing Stock Analysis using the LSTM model and using 80 percent of the data as training data. We will be plotting and visualizing the graph of prediction versus actual closing values of stock prices. As for model evaluation, we will be using RMSE (Root Mean Squared Error) for calculating accuracy of the model and also showing a table of Predictions against Closed Values.

1. DATA CLEANING AND PREPARATION

A. Data Preprocessing

We will take the Microsoft stock data of the previous 5 years and take 80% training data (approx. 1283 rows). Then we will scale the data using fit_transform. We will then create a scaled training dataset, scaled test dataset and convert NumPy arrays of the training and test dataset.

We will be reshaping the data into 3D data as the LSTM model needs a 3D model to work. Example: (1283,6) indicates 1283 rows and 6 columns. We will be converting it to (1283,6,1)

B. Dataset

We are using Yahoo finance website and web scrape the stock data from the website. We will be using Microsoft ticker ('MSFT') and taking the start date from 01^{st} Jan 2015 to 06^{th} May 2021 (approximately 5 years of data).

Please refer to the below image:

```
72] df = web.DataReader('MSFT', data_source='yahoo', start='2015-01-01', end='2021-06-05')
    df.tail(10)
```

Date	High	Low	Open	Close	Volume	Adj Close
2021-05-03	254.350006	251.119995	253.399994	251.860001	19626600.0	251.860001
2021-05-04	251.210007	245.759995	250.970001	247.789993	32756100.0	247.789993
2021-05-05	249.500000	245.820007	249.059998	246.470001	21901300.0	246.470001
2021-05-06	249.860001	244.690002	246.449997	249.729996	26491100.0	249.729996
2021-05-07	254.300003	251.169998	252.149994	252.460007	27010100.0	252.460007
2021-05-10	251.729996	247.119995	250.869995	247.179993	29299900.0	247.179993
2021-05-11	246.600006	242.570007	244.550003	246.229996	33641600.0	246.229996
2021-05-12	244.380005	238.070007	242.169998	239.000000	36684400.0	239.000000
2021-05-13	245.600006	241.419998	241.800003	243.029999	29624300.0	243.029999
2021-05-14	249.179993	245.490005	245.580002	248.149994	23868600.0	248.149994

C. Data Preparation

Dependencies Setup: Python dependencies, the mentioned packages are imported: math, pandas, NumPy, sklearn, keras.models, keras.layers, matplotlib.

2. LSTM PREDICTION ALGORITHM

Long short-term memory (LSTM) units are units of a recurrent neural network (RNN). An RNN composed of LSTM units is often called an LSTM network. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.
LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the exploding and vanishing gradient problems that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications.

3. MODEL EVALUATION

As stated earlier, we are using RMSE to calculate the accuracy of the model (RMSE is inversely proportional to accuracy). If the epoch is set to one, we may get 0.33 RMSE but the value of the stock prices may be very far apart.

Setting the EPOCH to ten to train the model:

```
[82] model.fit(x_train, y_train, batch_size=1, epochs=10)

     Epoch 1/10
     1223/1223 [==============================] - 37s 28ms/step - loss: 0.0044
     Epoch 2/10
     1223/1223 [==============================] - 33s 27ms/step - loss: 2.2910e-04
     Epoch 3/10
     1223/1223 [==============================] - 31s 26ms/step - loss: 1.4450e-04
     Epoch 4/10
     1223/1223 [==============================] - 31s 25ms/step - loss: 2.2105e-04
     Epoch 5/10
     1223/1223 [==============================] - 31s 25ms/step - loss: 2.0766e-04
     Epoch 6/10
     1223/1223 [==============================] - 31s 25ms/step - loss: 1.3841e-04
     Epoch 7/10
     1223/1223 [==============================] - 31s 25ms/step - loss: 1.4575e-04
     Epoch 8/10
     1223/1223 [==============================] - 31s 25ms/step - loss: 9.1692e-05
     Epoch 9/10
     1223/1223 [==============================] - 30s 25ms/step - loss: 1.2061e-04
     Epoch 10/10
     1223/1223 [==============================] - 30s 25ms/step - loss: 1.1166e-04
     <tensorflow.python.keras.callbacks.History at 0x7f64d3395710>
```

For the following model with 10 epochs we get RMSE as:

Model Evaluation

Using root mean squared error - RMSE

Accuracy of model

A lower RMSE value/score means more accurate model

```
[89] rmse = np.sqrt(np.mean(predictions - y_test)**2)
     print ("Error rate: " + str(rmse))

     Error rate: 4.818627214431762
```

Actual closing values versus Predictions:

Showing a table of close prices against Predictions

```
[88] valid
```

Date	Adj Close	Predictions
2020-02-07	181.544250	183.883698
2020-02-10	186.292862	184.468048
2020-02-11	182.087234	188.468384
2020-02-12	182.353775	185.586197
2020-02-13	181.366547	185.210480
...	...	...
2021-05-10	247.179993	259.605621
2021-05-11	246.229996	254.585892
2021-05-12	239.000000	252.808640
2021-05-13	243.029999	245.949509
2021-05-14	248.149994	249.288651

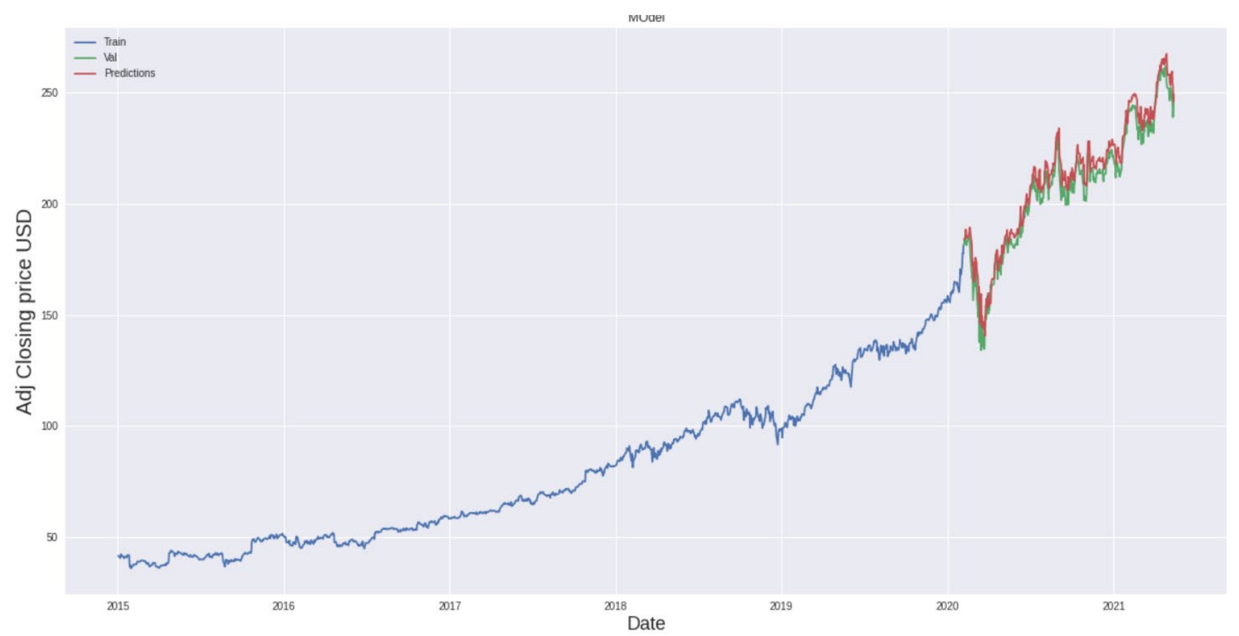
320 rows × 2 columns

4. CONCLUSIONS

This system can be improved by setting the number of epochs greater than one, this will train the model but consume comparatively more time to compile the model. In our project we have set the epoch to 10 to train the model and this has given us around 4.8186 RMSE (Root Mean Squared Error).

Visualization:


Adj Closing Price
250
200
150
100
50


MOdel
Train
Val
Predictions
Adj Closing price USD
250
200
150
100
50
2015
2016
2017
2018
2019
2020
2021
Date