stock-bharat

November 8, 2023

1 Stock Prediction

1.0.1 Taking a stock price of Netflix company and predicting its price by using LSTM.

Importing Libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
from sklearn.preprocessing import MinMaxScaler
```

loading the dataset

[2]:
```
df=pd.read_csv(r"C:\Users\RAMYA SRI\Downloads\archive (2)\NFLX.csv")
```

[3]:
```
df
```

[3]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2018-02-05	262.000000	267.899994	250.029999	254.259995	254.259995	11896100
1	2018-02-06	247.699997	266.700012	245.000000	265.720001	265.720001	12595800
2	2018-02-07	266.579987	272.450012	264.329987	264.559998	264.559998	8981500
3	2018-02-08	267.079987	267.619995	250.000000	250.100006	250.100006	9306700
4	2018-02-09	253.850006	255.800003	236.110001	249.470001	249.470001	16906900
…	…	…	…	…	…	…	…
1004	2022-01-31	401.970001	427.700012	398.200012	427.140015	427.140015	
1005	2022-02-01	432.959991	458.480011	425.540009	457.130005	457.130005	
1006	2022-02-02	448.250000	451.980011	426.480011	429.480011	429.480011	
1007	2022-02-03	421.440002	429.260010	404.279999	405.600006	405.600006	
1008	2022-02-04	407.309998	412.769989	396.640015	410.170013	410.170013	

```
…         …
1004  20047500
1005  22542300
1006  14346000
1007   9905200
1008   7782400

[1009 rows x 7 columns]
```

[4]:
```
df.head()
```

[4]:
```
         Date        Open        High         Low       Close   Adj Close  \
0  2018-02-05  262.000000  267.899994  250.029999  254.259995  254.259995
1  2018-02-06  247.699997  266.700012  245.000000  265.720001  265.720001
2  2018-02-07  266.579987  272.450012  264.329987  264.559998  264.559998
3  2018-02-08  267.079987  267.619995  250.000000  250.100006  250.100006
4  2018-02-09  253.850006  255.800003  236.110001  249.470001  249.470001

     Volume
0  11896100
1  12595800
2   8981500
3   9306700
4  16906900
```

[5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1009 entries, 0 to 1008
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       1009 non-null   object
 1   Open       1009 non-null   float64
 2   High       1009 non-null   float64
 3   Low        1009 non-null   float64
 4   Close      1009 non-null   float64
 5   Adj Close  1009 non-null   float64
 6   Volume     1009 non-null   int64
dtypes: float64(5), int64(1), object(1)
memory usage: 55.3+ KB
```

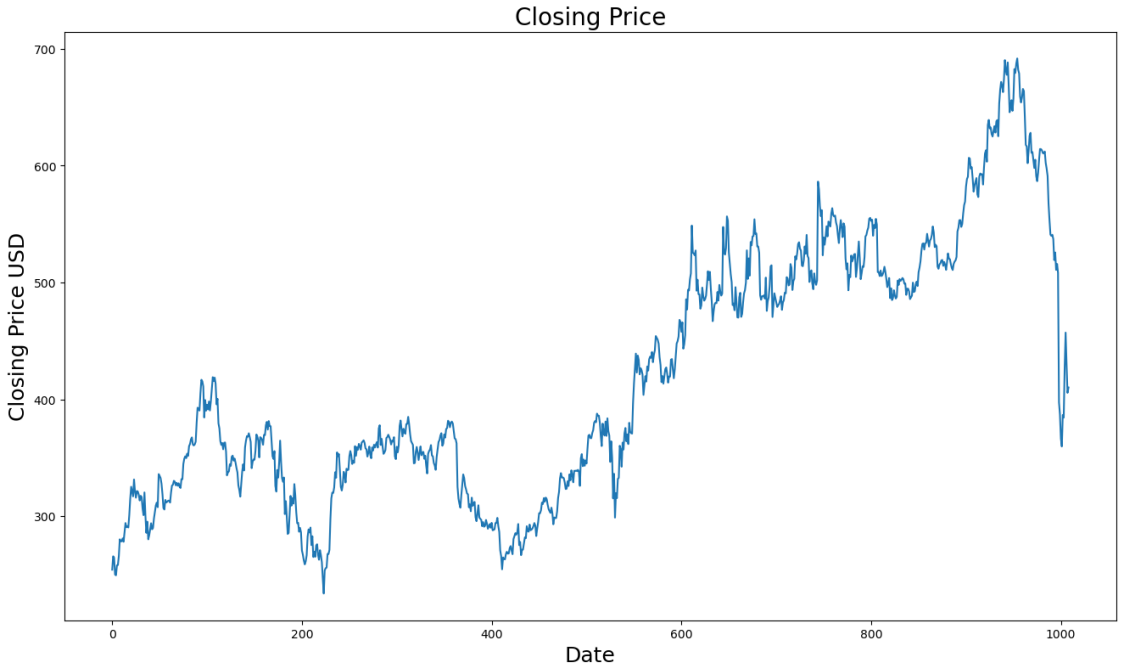
[6]:
```
df.shape
```

[6]:
```
(1009, 7)
```

Ploting the closing price of the stock to visualize the trend

[10]:
```
plt.figure(figsize=(16,9))
plt.title('Closing Price',fontsize=20)
plt.plot(df['Close'])
plt.xlabel('Date',fontsize=18)
plt.ylabel('Closing Price USD',fontsize=18)
```

[10]:
```
Text(0, 0.5, 'Closing Price USD')
```



Processing the data before feeding to LSTM

[11]:
```
data = df.filter(['Close']).values
```

normalizing the data between 0 and 1 using the MinMaxScaler

[12]:
```
scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(data)
```

Training the dataset

[13]:
```
train_data = scaled_data[:int(len(scaled_data)*0.8)]
x_train = []
y_train = []

for i in range(60, len(train_data)):
    x_train.append(train_data[i-60:i, 0])
    y_train.append(train_data[i, 0])
```

```
x_train, y_train = np.array(x_train), np.array(y_train)
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
```

Building the LSTM mosel

[14]:
```
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(x_train.shape[1], 1)))
model.add(Dropout(0.2))
model.add(LSTM(50, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(50))
model.add(Dropout(0.2))
model.add(Dense(1))

model.compile(optimizer='adam', loss='mean_squared_error')
```

[15]:
```
model.fit(x_train, y_train, epochs=50, batch_size=32)
```

```
Epoch 1/50
24/24 [==============================] - 15s 129ms/step - loss: 0.0319
Epoch 2/50
24/24 [==============================] - 3s 127ms/step - loss: 0.0059
Epoch 3/50
24/24 [==============================] - 3s 128ms/step - loss: 0.0052
Epoch 4/50
24/24 [==============================] - 3s 129ms/step - loss: 0.0047
Epoch 5/50
24/24 [==============================] - 3s 128ms/step - loss: 0.0043
Epoch 6/50
24/24 [==============================] - 3s 128ms/step - loss: 0.0042
Epoch 7/50
24/24 [==============================] - 3s 128ms/step - loss: 0.0036
Epoch 8/50
24/24 [==============================] - 3s 126ms/step - loss: 0.0037
Epoch 9/50
24/24 [==============================] - 3s 127ms/step - loss: 0.0036
Epoch 10/50
24/24 [==============================] - 3s 128ms/step - loss: 0.0039
Epoch 11/50
24/24 [==============================] - 3s 127ms/step - loss: 0.0040
Epoch 12/50
24/24 [==============================] - 3s 126ms/step - loss: 0.0036
Epoch 13/50
24/24 [==============================] - 3s 127ms/step - loss: 0.0035
Epoch 14/50
24/24 [==============================] - 2s 98ms/step - loss: 0.0034
Epoch 15/50
```

```
24/24 [==============================] - 1s 59ms/step - loss: 0.0030
Epoch 16/50
24/24 [==============================] - 1s 57ms/step - loss: 0.0032
Epoch 17/50
24/24 [==============================] - 1s 60ms/step - loss: 0.0035
Epoch 18/50
24/24 [==============================] - 1s 59ms/step - loss: 0.0033
Epoch 19/50
24/24 [==============================] - 1s 58ms/step - loss: 0.0030
Epoch 20/50
24/24 [==============================] - 1s 58ms/step - loss: 0.0029
Epoch 21/50
24/24 [==============================] - 1s 56ms/step - loss: 0.0027
Epoch 22/50
24/24 [==============================] - 1s 56ms/step - loss: 0.0028
Epoch 23/50
24/24 [==============================] - 1s 56ms/step - loss: 0.0030
Epoch 24/50
24/24 [==============================] - 1s 57ms/step - loss: 0.0027
Epoch 25/50
24/24 [==============================] - 1s 57ms/step - loss: 0.0028
Epoch 26/50
24/24 [==============================] - 1s 56ms/step - loss: 0.0027
Epoch 27/50
24/24 [==============================] - 1s 55ms/step - loss: 0.0024
Epoch 28/50
24/24 [==============================] - 1s 55ms/step - loss: 0.0027
Epoch 29/50
24/24 [==============================] - 1s 56ms/step - loss: 0.0026
Epoch 30/50
24/24 [==============================] - 1s 55ms/step - loss: 0.0025
Epoch 31/50
24/24 [==============================] - 1s 54ms/step - loss: 0.0024
Epoch 32/50
24/24 [==============================] - 1s 56ms/step - loss: 0.0021
Epoch 33/50
24/24 [==============================] - 1s 55ms/step - loss: 0.0024
Epoch 34/50
24/24 [==============================] - 1s 55ms/step - loss: 0.0023
Epoch 35/50
24/24 [==============================] - 1s 54ms/step - loss: 0.0024
Epoch 36/50
24/24 [==============================] - 1s 55ms/step - loss: 0.0023
Epoch 37/50
24/24 [==============================] - 1s 57ms/step - loss: 0.0021
Epoch 38/50
24/24 [==============================] - 1s 56ms/step - loss: 0.0023
Epoch 39/50
```

```
24/24 [==============================] - 1s 55ms/step - loss: 0.0024
Epoch 40/50
24/24 [==============================] - 1s 54ms/step - loss: 0.0023
Epoch 41/50
24/24 [==============================] - 1s 57ms/step - loss: 0.0026
Epoch 42/50
24/24 [==============================] - 1s 55ms/step - loss: 0.0020
Epoch 43/50
24/24 [==============================] - 1s 54ms/step - loss: 0.0019
Epoch 44/50
24/24 [==============================] - 1s 56ms/step - loss: 0.0019
Epoch 45/50
24/24 [==============================] - 1s 54ms/step - loss: 0.0020
Epoch 46/50
24/24 [==============================] - 1s 57ms/step - loss: 0.0022
Epoch 47/50
24/24 [==============================] - 1s 57ms/step - loss: 0.0023
Epoch 48/50
24/24 [==============================] - 1s 54ms/step - loss: 0.0017
Epoch 49/50
24/24 [==============================] - 1s 56ms/step - loss: 0.0019
Epoch 50/50
24/24 [==============================] - 1s 54ms/step - loss: 0.0018
```

[15]:
```
<keras.callbacks.History at 0x1c3ea3217f0>
```

predictions on the test data

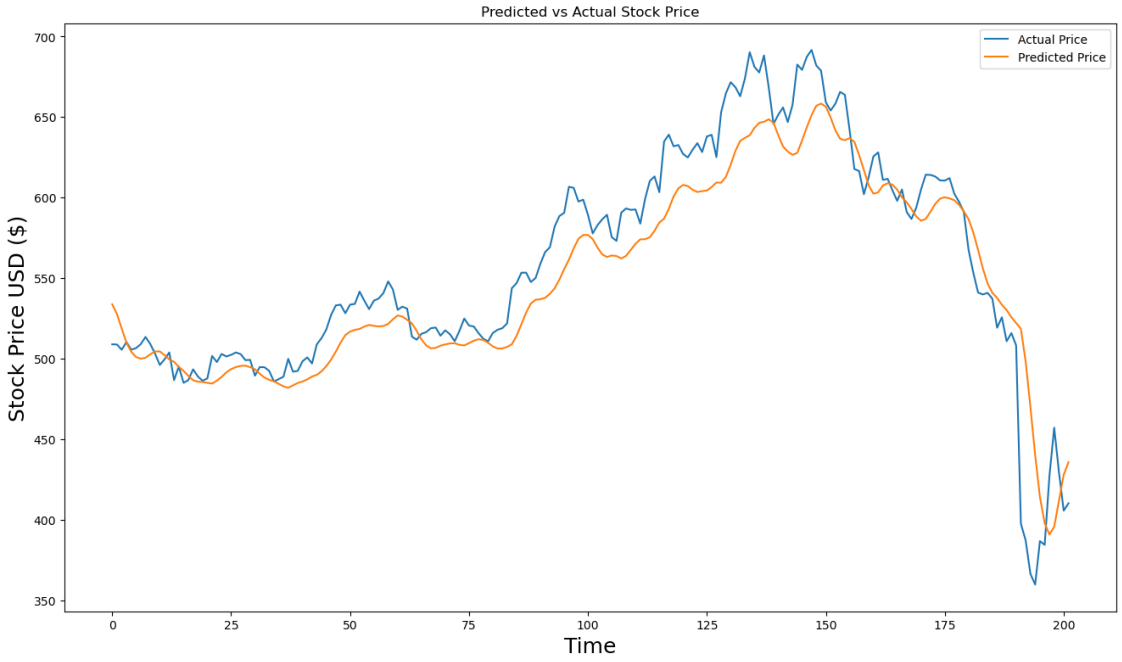
[16]:
```
test_data = scaled_data[int(len(scaled_data)*0.8) - 60:]
x_test = []
y_test = data[int(len(data)*0.8):, :]

for i in range(60, len(test_data)):
    x_test.append(test_data[i-60:i, 0])

x_test = np.array(x_test)
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))

predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)
```

```
7/7 [==============================] - 1s 17ms/step
```

Visualizing the Predicted price as compared to Actual price

[17]:
```
plt.figure(figsize=(16,9))
plt.title('Predicted vs Actual Stock Price')
plt.plot(y_test, label='Actual Price')
```

```
plt.plot(predictions, label='Predicted Price')
plt.xlabel('Time', fontsize=18)
plt.ylabel('Stock Price USD ($)', fontsize=18)
plt.legend()
plt.show()
```



[ ]: