Stock Price Predictor

August 15, 2021

1 Stock Price Predictor

1.0.1 Part 1 - Data Preprocessing

[1]:
```
!pip install --upgrade yfinance
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: yfinance in /home/mamba-
helisinki/.local/lib/python3.9/site-packages (0.1.63)
Requirement already satisfied: multitasking>=0.0.7 in /home/mamba-
helisinki/.local/lib/python3.9/site-packages (from yfinance) (0.0.9)
Requirement already satisfied: lxml>=4.5.1 in /home/mamba-
helisinki/.local/lib/python3.9/site-packages (from yfinance) (4.6.3)
Requirement already satisfied: numpy>=1.15 in /home/mamba-
helisinki/.local/lib/python3.9/site-packages (from yfinance) (1.19.5)
Requirement already satisfied: requests>=2.20 in /usr/lib/python3/dist-packages
(from yfinance) (2.25.1)
Requirement already satisfied: pandas>=0.24 in /home/mamba-
helisinki/.local/lib/python3.9/site-packages (from yfinance) (1.3.1)
Requirement already satisfied: pytz>=2017.3 in /usr/lib/python3/dist-packages
(from pandas>=0.24->yfinance) (2021.1)
Requirement already satisfied: python-dateutil>=2.7.3 in /home/mamba-
helisinki/.local/lib/python3.9/site-packages (from pandas>=0.24->yfinance)
(2.8.1)
Requirement already satisfied: six>=1.5 in /usr/lib/python3/dist-packages (from
python-dateutil>=2.7.3->pandas>=0.24->yfinance) (1.15.0)
WARNING: You are using pip version 21.1.2; however, version 21.2.3 is
available.
You should consider upgrading via the '/usr/bin/python3 -m pip install --upgrade
pip' command.
```

1.0.2 Importing the libraries

[2]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

1.0.3 Importing the training set

[3]:
```
import yfinance as yf
data = yf.download('BHARTIARTL.NS','2015-05-08')
plt.figure(figsize=(16,8))
plt.title("Opening Price History")
plt.plot(data["Open"])
plt.xlabel("Date")
plt.ylabel("Open Price INR($)")
plt.show()
dataset=data.filter(["Open"]).values
trainingDataLength = int(len(dataset)*(0.95))
training_set = dataset[0:trainingDataLength, :]
training_set
```

```
[*********************100%***********************]  1 of 1 completed
```

[3]:
```
array([[364.51681519],
       [364.51681519],
       [364.05773926],
       …,
       [529.        ],
       [530.        ],
       [529.        ]])
```

1.0.4 Feature Scaling

[4]:
```
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range = (0, 1))
training_set_scaled = sc.fit_transform(training_set)
```

1.0.5 Creating a data structure with 60 timesteps and 1 output

[5]:
```
X_train = []
y_train = []
for i in range(60, len(training_set_scaled)):
    X_train.append(training_set_scaled[i-60:i, 0])
    y_train.append(training_set_scaled[i, 0])
X_train, y_train = np.array(X_train), np.array(y_train)
```

1.0.6 Reshaping

[6]:
```
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
```

2 Part 2 - Building and Training the RNN

2.0.1 Importing the Keras libraries and packages

[7]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dropout
```

2.0.2 Initialising the RNN

[8]:
```
regressor = Sequential()
```

2.0.3 Adding the first LSTM layer and some Dropout regularisation

[9]:
```
regressor.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.
 ↪shape[1], 1)))
regressor.add(Dropout(0.2))
```

2.0.4 Adding a second LSTM layer and some Dropout regularisation

[10]:
```
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))
```

2.0.5 Adding a third LSTM layer and some Dropout regularisation

[11]:
```
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))
```

2.0.6 Adding a fourth LSTM layer and some Dropout regularisation

[12]:
```
regressor.add(LSTM(units = 50))
regressor.add(Dropout(0.2))
```

2.0.7 Adding the output layer

[13]:
```
regressor.add(Dense(units = 1))
```

2.0.8 Compiling the RNN

[14]:
```
regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')
regressor.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 60, 50)            10400
_________________________________________________________________
dropout (Dropout)            (None, 60, 50)            0
_________________________________________________________________
lstm_1 (LSTM)                (None, 60, 50)            20200
_________________________________________________________________
dropout_1 (Dropout)          (None, 60, 50)            0
_________________________________________________________________
lstm_2 (LSTM)                (None, 60, 50)            20200
_________________________________________________________________
dropout_2 (Dropout)          (None, 60, 50)            0
_________________________________________________________________
lstm_3 (LSTM)                (None, 50)                20200
_________________________________________________________________
dropout_3 (Dropout)          (None, 50)                0
_________________________________________________________________
dense (Dense)                (None, 1)                 51
=================================================================
Total params: 71,051
Trainable params: 71,051
Non-trainable params: 0
_________________________________________________________________
```

2.0.9 Fitting the RNN to the Training set

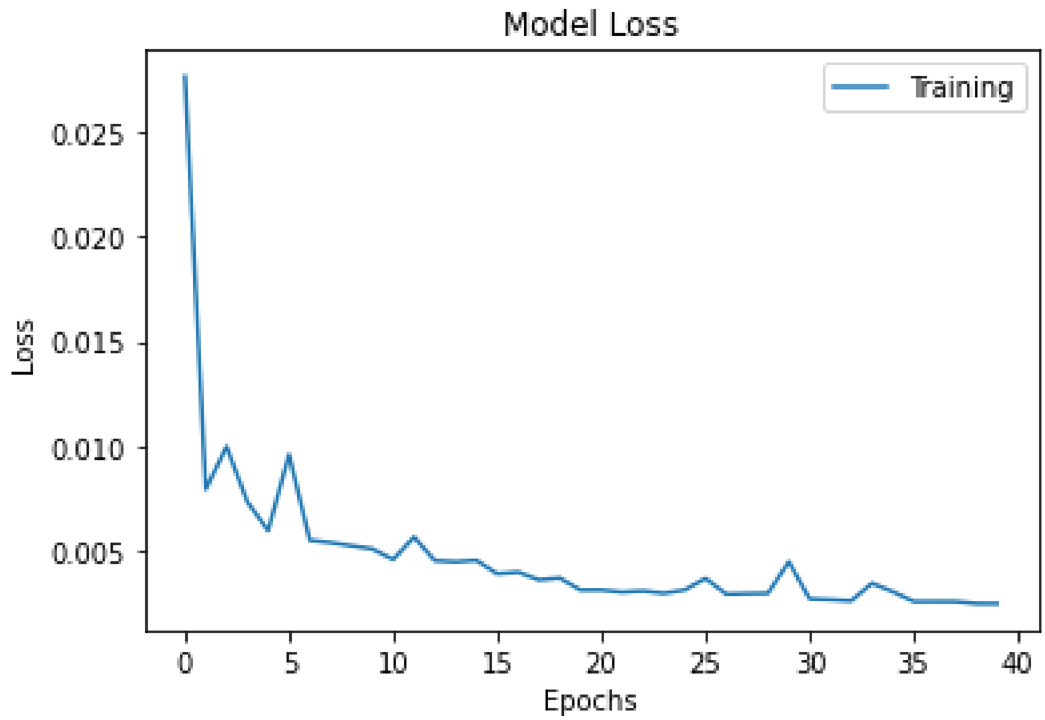
[15]:
```
history = regressor.fit(X_train, y_train, epochs = 40, batch_size = 32)
```

```
Epoch 1/40
45/45 [==============================] - 11s 97ms/step - loss: 0.0277
Epoch 2/40
45/45 [==============================] - 4s 96ms/step - loss: 0.0079
Epoch 3/40
45/45 [==============================] - 4s 98ms/step - loss: 0.0100
Epoch 4/40
45/45 [==============================] - 4s 97ms/step - loss: 0.0073
Epoch 5/40
45/45 [==============================] - 4s 96ms/step - loss: 0.0059
Epoch 6/40
45/45 [==============================] - 4s 96ms/step - loss: 0.0096
Epoch 7/40
45/45 [==============================] - 4s 96ms/step - loss: 0.0055
Epoch 8/40
45/45 [==============================] - 4s 96ms/step - loss: 0.0054
Epoch 9/40
45/45 [==============================] - 4s 96ms/step - loss: 0.0052
Epoch 10/40
45/45 [==============================] - 4s 96ms/step - loss: 0.0051
Epoch 11/40
45/45 [==============================] - 4s 96ms/step - loss: 0.0046
Epoch 12/40
45/45 [==============================] - 4s 96ms/step - loss: 0.0056
Epoch 13/40
45/45 [==============================] - 4s 96ms/step - loss: 0.0045
Epoch 14/40
45/45 [==============================] - 4s 96ms/step - loss: 0.0044
Epoch 15/40
45/45 [==============================] - 4s 97ms/step - loss: 0.0045
Epoch 16/40
45/45 [==============================] - 4s 96ms/step - loss: 0.0039
Epoch 17/40
45/45 [==============================] - 4s 99ms/step - loss: 0.0039
Epoch 18/40
45/45 [==============================] - 4s 95ms/step - loss: 0.0036
Epoch 19/40
45/45 [==============================] - 4s 96ms/step - loss: 0.0037
Epoch 20/40
45/45 [==============================] - 4s 95ms/step - loss: 0.0031
Epoch 21/40
45/45 [==============================] - 4s 96ms/step - loss: 0.0031
Epoch 22/40
45/45 [==============================] - 4s 95ms/step - loss: 0.0030
```

```
Epoch 23/40
45/45 [==============================] - 4s 96ms/step - loss: 0.0030
Epoch 24/40
45/45 [==============================] - 4s 95ms/step - loss: 0.0029
Epoch 25/40
45/45 [==============================] - 4s 96ms/step - loss: 0.0031
Epoch 26/40
45/45 [==============================] - 4s 96ms/step - loss: 0.0036
Epoch 27/40
45/45 [==============================] - 4s 96ms/step - loss: 0.0029
Epoch 28/40
45/45 [==============================] - 4s 96ms/step - loss: 0.0029
Epoch 29/40
45/45 [==============================] - 4s 96ms/step - loss: 0.0029
Epoch 30/40
45/45 [==============================] - 4s 96ms/step - loss: 0.0044
Epoch 31/40
45/45 [==============================] - 4s 98ms/step - loss: 0.0027
Epoch 32/40
45/45 [==============================] - 4s 96ms/step - loss: 0.0026
Epoch 33/40
45/45 [==============================] - 5s 110ms/step - loss: 0.0026
Epoch 34/40
45/45 [==============================] - 5s 102ms/step - loss: 0.0034
Epoch 35/40
45/45 [==============================] - 4s 96ms/step - loss: 0.0030
Epoch 36/40
45/45 [==============================] - 4s 97ms/step - loss: 0.0025
Epoch 37/40
45/45 [==============================] - 4s 96ms/step - loss: 0.0025
Epoch 38/40
45/45 [==============================] - 4s 96ms/step - loss: 0.0025
Epoch 39/40
45/45 [==============================] - 4s 96ms/step - loss: 0.0024
Epoch 40/40
45/45 [==============================] - 4s 96ms/step - loss: 0.0024
```

[16]:

```python
import matplotlib.pyplot as plt
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.plot(history.history['loss'])
plt.legend(['Training'])
plt.show()
```



[17]:
```
regressor.save('model')
```

```
WARNING:absl:Found untraced functions such as lstm_cell_layer_call_fn,
lstm_cell_layer_call_and_return_conditional_losses, lstm_cell_1_layer_call_fn,
lstm_cell_1_layer_call_and_return_conditional_losses, lstm_cell_2_layer_call_fn
while saving (showing 5 of 20). These functions will not be directly callable
after loading.

INFO:tensorflow:Assets written to: model/assets

INFO:tensorflow:Assets written to: model/assets
```

[18]:
```
!zip -r model.zip model
```

```
updating: model/ (stored 0%)
updating: model/saved_model.pb (deflated 90%)
updating: model/variables/ (stored 0%)
updating: model/variables/variables.data-00000-of-00001 (deflated 7%)
updating: model/variables/variables.index (deflated 69%)
updating: model/keras_metadata.pb (deflated 93%)
updating: model/assets/ (stored 0%)
```

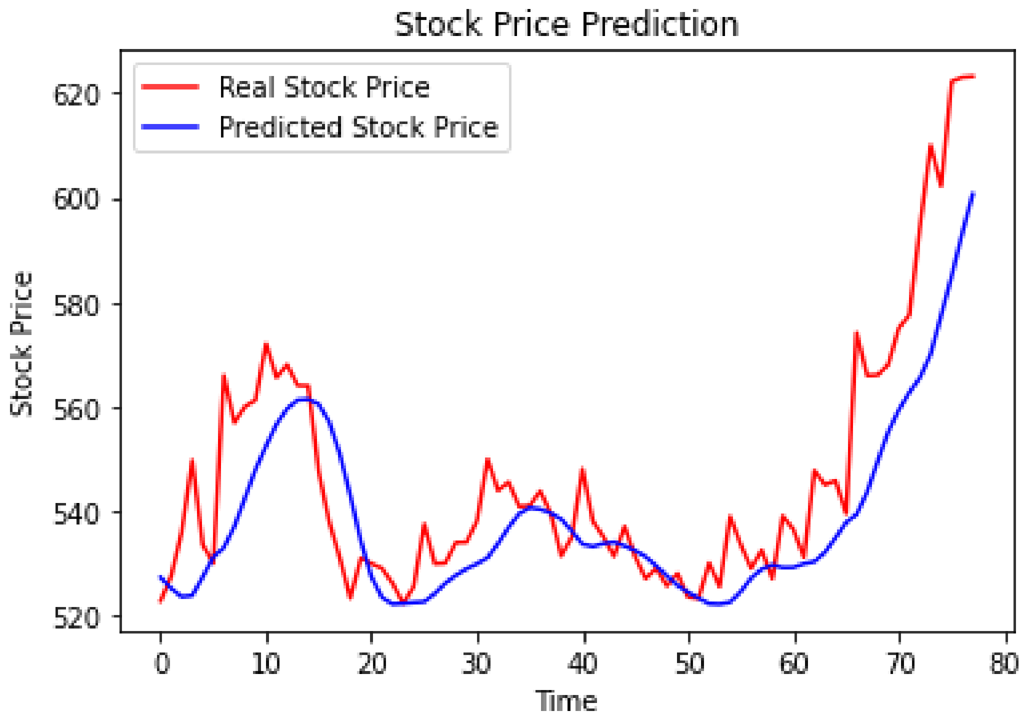
3 Part 3 - Making the predictions and visualising the results

3.0.1 Getting the predicted stock price

[19]:
```
import tensorflow
regressor = tensorflow.keras.models.load_model('model')
dataset_total = dataset
dataset_test = dataset[trainingDataLength:, :]
inputs = dataset_total[len(dataset_total) - len(dataset_test) - 60:]
inputs = inputs.reshape(-1,1)
inputs = sc.transform(inputs)
X_test = []
for i in range(60, len(inputs)):
    X_test.append(inputs[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))
predicted_stock_prices = regressor.predict(X_test)
predicted_stock_prices = sc.inverse_transform(predicted_stock_prices)
```

3.0.2 Visualising the results

[20]:
```
plt.plot(dataset_test, color = 'red', label = 'Real Stock Price')
plt.plot(predicted_stock_prices, color = 'blue', label = 'Predicted Stock␣
↪Price')
plt.title('Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('Stock Price')
plt.legend()
plt.show()
```



[21]:
```
import math
MSE = np.square(np.subtract(dataset_test,predicted_stock_prices)).mean()
RMSE = math.sqrt(MSE)
print(RMSE)
```

14.202359230776263