Final_Google_Stock Price

April 18, 2023

LSTM that predicts upwards and downwards trend of google stockprice Many layers with dropout regularisation to prevent overfitting

Part 1 - Data Preprocessing

[1]:
```
# Importing the libraries
import numpy as np #allow to make arrays
import matplotlib.pyplot as plt #visualize results on charts
import pandas as pd #import dataset and manage easily
```

Load the Training Dataset and Use the Open Stock Price Column to Train Your Model.

[2]:
```
# Importing the training set - only importing training set, test set later  on
#rnn has no idea of the test set's data, then after training is done, test set␣
↪will eb important
dataset_train = pd.read_csv(r'D:
↪\SRB_Backup\Deep_Learning\LP_V\SRB_Dataset\stock\Google_Stock_Price_Train.
↪csv')


#need to make into numpy arrays because only nump arrays can be input values in␣
↪keras
training_set = dataset_train.iloc[:, 1:2].values

#getting everything from the columns (.values makes the numpy array)
```

[3]:
```
training_set.shape
```

[3]:
```
(1258, 1)
```

[4]:
```
# Feature Scaling
# Normalizing the Dataset
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range = (0, 1))
training_set_scaled = sc.fit_transform(training_set)
#fit (gets min and max on data to apply formula) tranform(compute scale stock␣
↪prices to each formula)
```

Creating X_train and y_train Data Structures.

[5]:
```
# Creating a data structure with 60 timesteps and 1 output
#60 times steps- at each time t and look at 60 previous time steps, then make␣
↪new prediction
# 1 time step leads to overfitting, 20 is still too low
#60 previous financial days- in 3 months
X_train = []
y_train = []
for i in range(60, 1257): # upper bound is number of values
  X_train.append(training_set_scaled[i-60:i, 0]) #takes 60 previous stock␣
↪prices from 60 past stock prices
  y_train.append(training_set_scaled[i, 0]) #contains stock price learned to␣
↪predict
X_train, y_train = np.array(X_train), np.array(y_train) # make into numpy␣
↪arrays
#Need to add dimension to because not only prescition with one stock price but␣
↪other indicators
# (like other columns in dataset  or other stocks that may affect this one )
```

[6]:
```
# Reshaping- add dimension in numpy array
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
#adds dimension in numpy array currently only have one indicator, with new␣
↪dimension will
# have more indicators, be compatible for "input shape" of RNN
# format according to keras documentation
```

[7]:
```
X_train.shape
```

[7]: (1197, 60, 1)

Part 2 - Building the RNN stacked lstm with dropout regularization to prevent overfitting

[8]:
```
# Importing the Keras libraries and packages
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
```

[9]:
```
# Initialising the RNN
regressor = Sequential()
#reps sequence of layers, predicting continous values (so it is a regression)
```

[10]:
```
# Adding the first LSTM layer and some Dropout regularisation
#dropout to prevent overfitting
regressor.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.
↪shape[1], 1)))
#regressor- object of sequential class, can add layers to networ.
#use lstm class and create object of lstm class- 3 args
```

```
#num of units, return sequences- set to true because is stacked lstms, and shape
#units- neurons in first layer. 50 in layers for high dimensionality, can␣
 ↪capture upward and downward
regressor.add(Dropout(0.2))
# takes arg of dropout late- num of neurons want to drop. dropping 20% of␣
 ↪neurons to be ignored
#during trianing for each iteration. 10 neurons will be dropped out
```

[11]:
```
# Adding a second LSTM layer and some Dropout regularisation
# total of 4 layers, simply need to copy, only change is input shape so dont␣
 ↪need to specify that,
#automatically recognised through input shape

regressor.add(LSTM(units = 50, return_sequences = True))

regressor.add(Dropout(0.2))
```

[12]:
```
# Adding a third LSTM layer and some Dropout regularisation
# same as second layer
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))
```

[13]:
```
# Adding a fourth LSTM layer and some Dropout regularisation
# almost same, but return sequence is false because it is the last lstm layer
#(so it is removed becasue default is false)
regressor.add(LSTM(units = 50))
regressor.add(Dropout(0.2))
```

[14]:
```
# Adding the output layer
#add fully connected layer through dense class- dimesion/units/neurons is 1
regressor.add(Dense(units = 1))
```

[15]:
```
# Compiling the RNN
#regressior because predicting continuous value,
regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')
```

Fitting the Model.

[16]:
```
# Fitting the RNN to the Training set
#have not made connection to training set, training will take place
regressor.fit(X_train, y_train, epochs = 200, batch_size = 32)
#100 gives good convergence trained on certain batch sizes,
```

```
Epoch 1/200
38/38 [==============================] - 4s 33ms/step - loss: 0.0334
Epoch 2/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0068
Epoch 3/200
```

```
38/38 [==============================] - 1s 33ms/step - loss: 0.0064
Epoch 4/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0060
Epoch 5/200
38/38 [==============================] - 1s 37ms/step - loss: 0.0058
Epoch 6/200
38/38 [==============================] - 1s 35ms/step - loss: 0.0059
Epoch 7/200
38/38 [==============================] - 1s 36ms/step - loss: 0.0055
Epoch 8/200
38/38 [==============================] - 1s 35ms/step - loss: 0.0050
Epoch 9/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0043
Epoch 10/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0047
Epoch 11/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0037
Epoch 12/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0055
Epoch 13/200
38/38 [==============================] - 1s 34ms/step - loss: 0.0045
Epoch 14/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0040
Epoch 15/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0042
Epoch 16/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0035
Epoch 17/200
38/38 [==============================] - 1s 35ms/step - loss: 0.0043
Epoch 18/200
38/38 [==============================] - 1s 37ms/step - loss: 0.0037
Epoch 19/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0039
Epoch 20/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0038
Epoch 21/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0035
Epoch 22/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0032
Epoch 23/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0035
Epoch 24/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0032
Epoch 25/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0035
Epoch 26/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0032
Epoch 27/200
```

```
38/38 [==============================] - 1s 32ms/step - loss: 0.0033
Epoch 28/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0032
Epoch 29/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0029
Epoch 30/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0031
Epoch 31/200
38/38 [==============================] - 1s 34ms/step - loss: 0.0029
Epoch 32/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0029
Epoch 33/200
38/38 [==============================] - 1s 34ms/step - loss: 0.0030
Epoch 34/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0029
Epoch 35/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0028
Epoch 36/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0028
Epoch 37/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0029
Epoch 38/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0026
Epoch 39/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0027
Epoch 40/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0030
Epoch 41/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0026
Epoch 42/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0029
Epoch 43/200
38/38 [==============================] - 1s 35ms/step - loss: 0.0028
Epoch 44/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0022
Epoch 45/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0024
Epoch 46/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0026
Epoch 47/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0024
Epoch 48/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0023
Epoch 49/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0025
Epoch 50/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0027
Epoch 51/200
```

```
38/38 [==============================] - 1s 33ms/step - loss: 0.0029
Epoch 52/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0024
Epoch 53/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0022
Epoch 54/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0023
Epoch 55/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0021
Epoch 56/200
38/38 [==============================] - 1s 35ms/step - loss: 0.0023
Epoch 57/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0021
Epoch 58/200
38/38 [==============================] - 1s 35ms/step - loss: 0.0024
Epoch 59/200
38/38 [==============================] - 1s 34ms/step - loss: 0.0019
Epoch 60/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0022
Epoch 61/200
38/38 [==============================] - 1s 34ms/step - loss: 0.0020
Epoch 62/200
38/38 [==============================] - 1s 34ms/step - loss: 0.0022
Epoch 63/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0020
Epoch 64/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0019
Epoch 65/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0023
Epoch 66/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0021
Epoch 67/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0019
Epoch 68/200
38/38 [==============================] - 1s 35ms/step - loss: 0.0017
Epoch 69/200
38/38 [==============================] - 1s 35ms/step - loss: 0.0019
Epoch 70/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0020
Epoch 71/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0018
Epoch 72/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0018
Epoch 73/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0019
Epoch 74/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0016
Epoch 75/200
```

```
38/38 [==============================] - 1s 33ms/step - loss: 0.0020
Epoch 76/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0018
Epoch 77/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0017
Epoch 78/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0018
Epoch 79/200
38/38 [==============================] - 1s 34ms/step - loss: 0.0018
Epoch 80/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0017
Epoch 81/200
38/38 [==============================] - 1s 35ms/step - loss: 0.0016
Epoch 82/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0018
Epoch 83/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0014
Epoch 84/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0015
Epoch 85/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0015
Epoch 86/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0016
Epoch 87/200
38/38 [==============================] - 1s 34ms/step - loss: 0.0016
Epoch 88/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0015
Epoch 89/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0015
Epoch 90/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0015
Epoch 91/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0016
Epoch 92/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0014
Epoch 93/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0015
Epoch 94/200
38/38 [==============================] - 1s 35ms/step - loss: 0.0015
Epoch 95/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0014
Epoch 96/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0016
Epoch 97/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0015
Epoch 98/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0016
Epoch 99/200
```

```
38/38 [==============================] - 1s 32ms/step - loss: 0.0015
Epoch 100/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0016
Epoch 101/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0014
Epoch 102/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0013
Epoch 103/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0015
Epoch 104/200
38/38 [==============================] - 1s 34ms/step - loss: 0.0014
Epoch 105/200
38/38 [==============================] - 1s 34ms/step - loss: 0.0014
Epoch 106/200
38/38 [==============================] - 1s 34ms/step - loss: 0.0014
Epoch 107/200
38/38 [==============================] - 1s 37ms/step - loss: 0.0013
Epoch 108/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0013
Epoch 109/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0014
Epoch 110/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0014
Epoch 111/200
38/38 [==============================] - 1s 35ms/step - loss: 0.0016
Epoch 112/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0014
Epoch 113/200
38/38 [==============================] - 1s 34ms/step - loss: 0.0012
Epoch 114/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0012
Epoch 115/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0012
Epoch 116/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0013
Epoch 117/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0012
Epoch 118/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0013
Epoch 119/200
38/38 [==============================] - 1s 34ms/step - loss: 0.0013
Epoch 120/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0012
Epoch 121/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0013
Epoch 122/200
38/38 [==============================] - 1s 34ms/step - loss: 0.0012
Epoch 123/200
```

```
38/38 [==============================] - 1s 33ms/step - loss: 0.0012
Epoch 124/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0013
Epoch 125/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0011
Epoch 126/200
38/38 [==============================] - 1s 34ms/step - loss: 0.0013
Epoch 127/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0014
Epoch 128/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0012
Epoch 129/200
38/38 [==============================] - 1s 35ms/step - loss: 0.0013
Epoch 130/200
38/38 [==============================] - 1s 35ms/step - loss: 0.0012
Epoch 131/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0013
Epoch 132/200
38/38 [==============================] - 1s 37ms/step - loss: 0.0012
Epoch 133/200
38/38 [==============================] - 1s 35ms/step - loss: 0.0013
Epoch 134/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0012
Epoch 135/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0012
Epoch 136/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0012
Epoch 137/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0011
Epoch 138/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0012
Epoch 139/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0011
Epoch 140/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0013
Epoch 141/200
38/38 [==============================] - 1s 32ms/step - loss: 0.0010
Epoch 142/200
38/38 [==============================] - 1s 34ms/step - loss: 0.0012
Epoch 143/200
38/38 [==============================] - 1s 34ms/step - loss: 0.0012
Epoch 144/200
38/38 [==============================] - 1s 35ms/step - loss: 0.0013
Epoch 145/200
38/38 [==============================] - 1s 36ms/step - loss: 0.0013
Epoch 146/200
38/38 [==============================] - 1s 33ms/step - loss: 0.0011
Epoch 147/200
```

```
38/38 [==============================] - 1s 33ms/step - loss: 0.0012
Epoch 148/200
38/38 [==============================] - 2s 43ms/step - loss: 0.0011
Epoch 149/200
38/38 [==============================] - 2s 44ms/step - loss: 0.0010
Epoch 150/200
38/38 [==============================] - 2s 45ms/step - loss: 0.0014
Epoch 151/200
38/38 [==============================] - 2s 45ms/step - loss: 0.0013
Epoch 152/200
38/38 [==============================] - 2s 44ms/step - loss: 0.0012
Epoch 153/200
38/38 [==============================] - 2s 46ms/step - loss: 0.0012
Epoch 154/200
38/38 [==============================] - 2s 43ms/step - loss: 0.0010
Epoch 155/200
38/38 [==============================] - 2s 41ms/step - loss: 0.0011
Epoch 156/200
38/38 [==============================] - 2s 45ms/step - loss: 0.0011
Epoch 157/200
38/38 [==============================] - 2s 46ms/step - loss: 0.0011
Epoch 158/200
38/38 [==============================] - 2s 46ms/step - loss: 0.0010
Epoch 159/200
38/38 [==============================] - 2s 45ms/step - loss: 0.0011
Epoch 160/200
38/38 [==============================] - 2s 47ms/step - loss: 0.0012
Epoch 161/200
38/38 [==============================] - 2s 44ms/step - loss: 0.0010
Epoch 162/200
38/38 [==============================] - 2s 43ms/step - loss: 0.0011
Epoch 163/200
38/38 [==============================] - 2s 45ms/step - loss: 9.7145e-04
Epoch 164/200
38/38 [==============================] - 2s 40ms/step - loss: 0.0011
Epoch 165/200
38/38 [==============================] - 2s 45ms/step - loss: 0.0011
Epoch 166/200
38/38 [==============================] - 2s 44ms/step - loss: 9.9821e-04
Epoch 167/200
38/38 [==============================] - 2s 45ms/step - loss: 0.0011
Epoch 168/200
38/38 [==============================] - 2s 45ms/step - loss: 0.0010
Epoch 169/200
38/38 [==============================] - 2s 44ms/step - loss: 0.0011
Epoch 170/200
38/38 [==============================] - 2s 46ms/step - loss: 0.0013
Epoch 171/200
```

```
38/38 [==============================] - 2s 43ms/step - loss: 0.0011
Epoch 172/200
38/38 [==============================] - 2s 45ms/step - loss: 9.2845e-04
Epoch 173/200
38/38 [==============================] - 2s 43ms/step - loss: 0.0010
Epoch 174/200
38/38 [==============================] - 2s 40ms/step - loss: 9.8681e-04
Epoch 175/200
38/38 [==============================] - 2s 44ms/step - loss: 0.0011
Epoch 176/200
38/38 [==============================] - 2s 44ms/step - loss: 9.7820e-04
Epoch 177/200
38/38 [==============================] - 2s 47ms/step - loss: 9.9271e-04
Epoch 178/200
38/38 [==============================] - 2s 46ms/step - loss: 9.8680e-04
Epoch 179/200
38/38 [==============================] - 2s 47ms/step - loss: 0.0011
Epoch 180/200
38/38 [==============================] - 2s 46ms/step - loss: 0.0010
Epoch 181/200
38/38 [==============================] - 2s 44ms/step - loss: 0.0011
Epoch 182/200
38/38 [==============================] - 2s 45ms/step - loss: 0.0010
Epoch 183/200
38/38 [==============================] - 2s 41ms/step - loss: 0.0011
Epoch 184/200
38/38 [==============================] - 2s 46ms/step - loss: 9.9955e-04
Epoch 185/200
38/38 [==============================] - 2s 46ms/step - loss: 8.6641e-04
Epoch 186/200
38/38 [==============================] - 2s 47ms/step - loss: 0.0011
Epoch 187/200
38/38 [==============================] - 2s 41ms/step - loss: 0.0010
Epoch 188/200
38/38 [==============================] - 2s 43ms/step - loss: 9.4438e-04
Epoch 189/200
38/38 [==============================] - 2s 41ms/step - loss: 0.0011
Epoch 190/200
38/38 [==============================] - 2s 42ms/step - loss: 9.2289e-04
Epoch 191/200
38/38 [==============================] - 2s 42ms/step - loss: 0.0012
Epoch 192/200
38/38 [==============================] - 2s 42ms/step - loss: 9.3167e-04
Epoch 193/200
38/38 [==============================] - 2s 40ms/step - loss: 8.9415e-04
Epoch 194/200
38/38 [==============================] - 2s 42ms/step - loss: 0.0011
Epoch 195/200
```

```
38/38 [==============================] - 2s 43ms/step - loss: 9.0057e-04
Epoch 196/200
38/38 [==============================] - 2s 41ms/step - loss: 9.2382e-04
Epoch 197/200
38/38 [==============================] - 2s 46ms/step - loss: 0.0012
Epoch 198/200
38/38 [==============================] - 2s 45ms/step - loss: 0.0012
Epoch 199/200
38/38 [==============================] - 2s 41ms/step - loss: 9.5528e-04
Epoch 200/200
38/38 [==============================] - 2s 41ms/step - loss: 0.0010
```

[16]:
```
<keras.callbacks.History at 0x2930990c760>
```

0.0.1 Part 3 - Making the predictions and visualising the results

[17]:
```
# Getting the Test Set
dataset_test = pd.read_csv('D:
 ↪\SRB_Backup\Deep_Learning\LP_V\SRB_Dataset\stock\Google_Stock_Price_Test.
 ↪csv')
real_stock_price = dataset_test.iloc[:, 1:2].values
```

[18]:
```
# Getting the predicted stock price
dataset_total = pd.concat((dataset_train['Open'], dataset_test['Open']), axis =␣
 ↪0)
inputs = dataset_total[len(dataset_total) - len(dataset_test) - 60:].values
#getting input of each previous financial days
inputs = inputs.reshape(-1,1)
inputs = sc.transform(inputs)
X_test = []
```

[19]:
```
inputs.shape
```

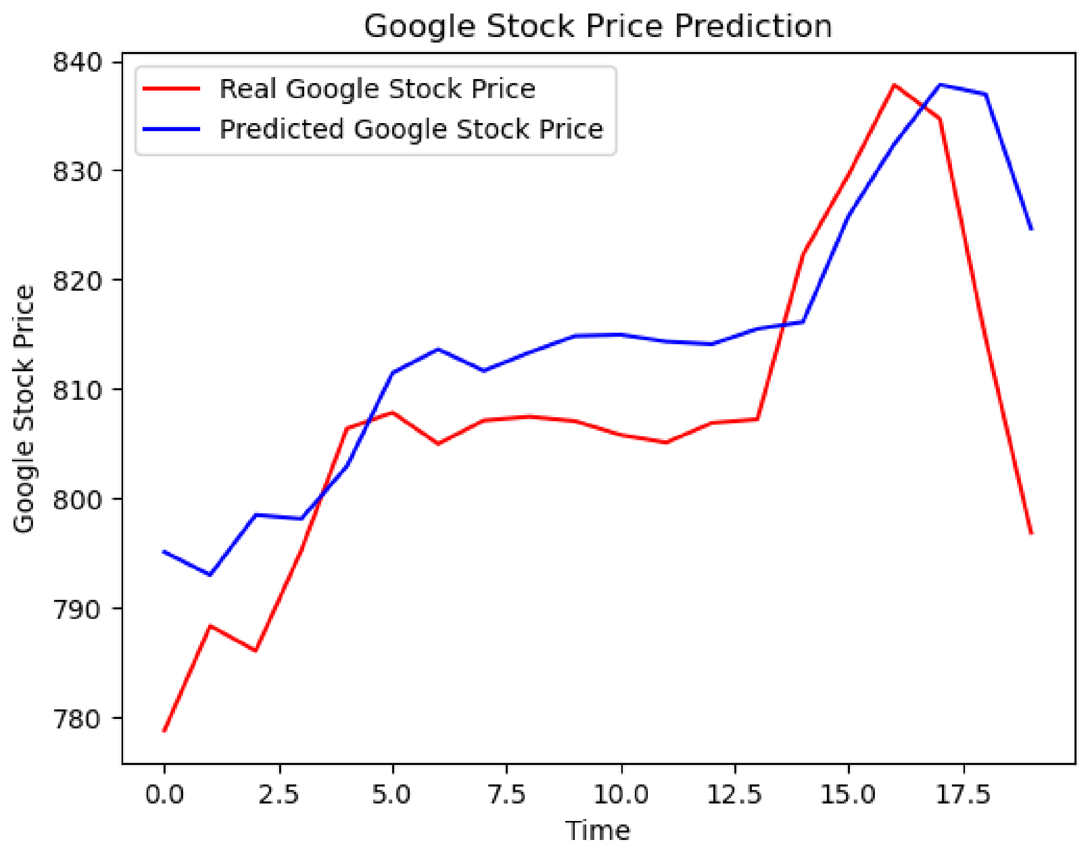
[19]:
```
(80, 1)
```

[20]:
```
for i in range(60, 80):
    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_price = regressor.predict(X_test)
predicted_stock_price = sc.inverse_transform(predicted_stock_price)
```

```
1/1 [==============================] - 1s 793ms/step
```

[21]:
```
# Visualising the results
plt.plot(real_stock_price, color = 'red',label = 'Real Google Stock Price')
plt.plot(predicted_stock_price, color = 'blue',label='Predicted Google Stock␣
 ↪Price')
```

```
plt.title('Google Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('Google Stock Price')
plt.legend()
plt.show()
```



For reference view https://www.youtube.com/watch?v=8XYYaakei4A

https://www.simplilearn.com/tutorials/machine-learning-tutorial/stock-price-prediction-using-machine-learning

```
[ ]:
```