DEEP LEARNING ASSIGNMENT -7

Submitted By: Dev Bansal

Roll No: 102117160

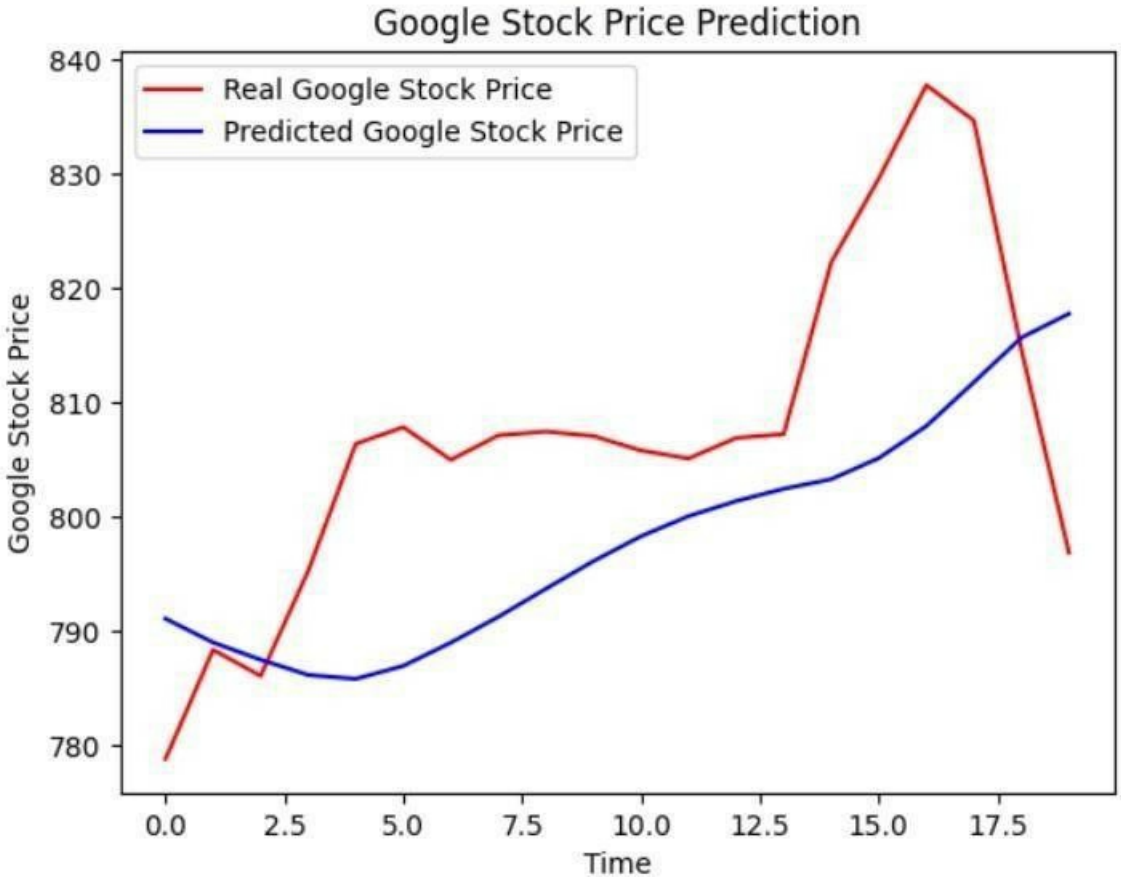
Batch: CS6

The branch of Deep Learning which facilitates this is Recurrent Neural Networks.

Classic RNNs have a short memory and were neither popular nor powerful for this exact reason. But a recent major improvement in Recurrent Neural Networks gave rise to the popularity of LSTMs (Long Short Term Memory RNNs) which has completely changed the playing field. In this part, we will learn how to implement this ultra-powerful model, and we will take the challenge to use it to predict the real Google stock price. A similar challenge has already been faced by researchers at Stanford University and we will aim to do at least as good as them.

```
import numpy as np import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler from keras.models import Sequential from keras.layers import Dense, LSTM

dataset_train = pd.read_csv('Google_Stock_Price_Train.csv')
training_set = dataset_train.iloc[:, 1:2].values
scaler = MinMaxScaler(feature_range=(0, 1)) training_set_scaled = scaler.fit_transform(training_set)
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)

X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
model = Sequential()

model.add(LSTM(units=50, return_sequences=True, input_shape=(X_train.shape[1], 1)))
model.add(LSTM(units=50, return_sequences=False)) model.add(Dense(units=1))
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(X_train, y_train, epochs=20, batch_size=32)
dataset_test = pd.read_csv('Google_Stock_Price_Test.csv')
real_stock_price = dataset_test.iloc[:, 1:2].values # "Open" prices
dataset_total = pd.concat((dataset_train['Open'], dataset_test['Open']), axis=0)

inputs = dataset_total[len(dataset_total) - len(dataset_test) - 60:].values
inputs = inputs.reshape(-1, 1)
inputs = scaler.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_price = model.predict(X_test)
predicted_stock_price = scaler.inverse_transform(predicted_stock_price)

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()
```

OUTPUT-


Google Stock Price Prediction
Real Google Stock Price
Predicted Google Stock Price
Google Stock Price
840
830
820
810
800
790
780
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Time