```
from google.colab import drive
drive.mount('/content/drive')
```

```
    Mounted at /content/drive
```

```
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,Dropout
```

```
import pandas as pd
path="/content/drive/MyDrive/datasets_LP5/Google_train_data.csv"
data = pd.read_csv(path)
data.head()
```

	Date	Open	High	Low	Close	Volume
0	1/3/2012	325.25	332.83	324.97	663.59	7,380,500
1	1/4/2012	331.27	333.87	329.08	666.45	5,749,400
2	1/5/2012	329.83	330.75	326.89	657.21	6,590,300
3	1/6/2012	328.34	328.77	323.68	648.24	5,405,900
4	1/9/2012	322.04	322.29	309.46	620.76	11,688,800

```
data.info()
```

```
    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 1258 entries, 0 to 1257
    Data columns (total 6 columns):
     #   Column  Non-Null Count  Dtype
    ---  ------  --------------  -----
     0   Date    1258 non-null   object
     1   Open    1258 non-null   float64
     2   High    1258 non-null   float64
     3   Low     1258 non-null   float64
     4   Close   1258 non-null   object
     5   Volume  1258 non-null   object
    dtypes: float64(3), object(3)
    memory usage: 59.1+ KB
```

```
data["Close"]=pd.to_numeric(data.Close,errors='coerce')
data = data.dropna()
trainData = data.iloc[:,4:5].values
```

```
data.info()
```

```
    <class 'pandas.core.frame.DataFrame'>
    Int64Index: 1149 entries, 0 to 1257
    Data columns (total 6 columns):
     #   Column  Non-Null Count  Dtype
    ---  ------  --------------  -----
     0   Date    1149 non-null   object
     1   Open    1149 non-null   float64
     2   High    1149 non-null   float64
     3   Low     1149 non-null   float64
     4   Close   1149 non-null   float64
     5   Volume  1149 non-null   object
    dtypes: float64(4), object(2)
    memory usage: 62.8+ KB
```

```
sc = MinMaxScaler(feature_range=(0,1))
trainData = sc.fit_transform(trainData)
trainData.shape
```

```
    (1149, 1)
```

```
X_train = []
y_train = []

for i in range (60,1149): #60 : timestep // 1149 : length of the data
    X_train.append(trainData[i-60:i,0])
    y_train.append(trainData[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)) #adding the batch_size axis
X_train.shape
```

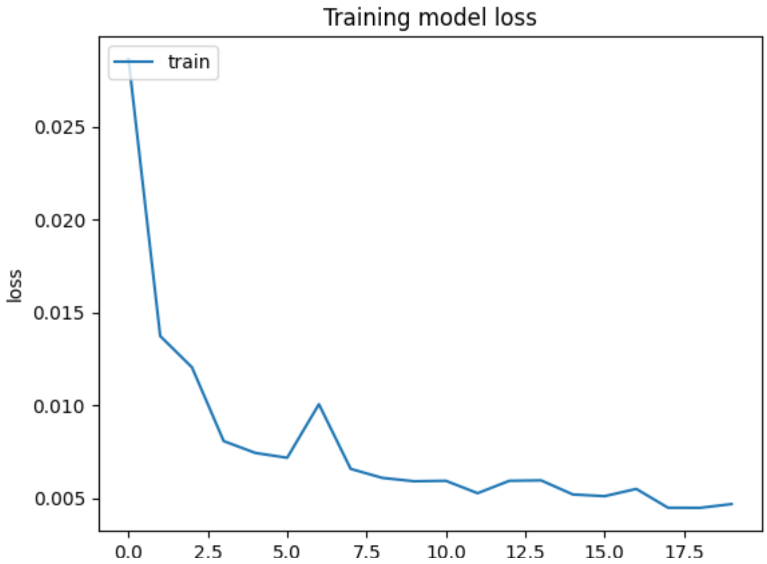
```
(1089, 60, 1)
```

```
model = Sequential()

model.add(LSTM(units=100, return_sequences = True, input_shape =(X_train.shape[1],1)))
model.add(Dropout(0.2))

model.add(LSTM(units=100, return_sequences = True))
model.add(Dropout(0.2))

model.add(LSTM(units=100, return_sequences = True))
model.add(Dropout(0.2))

model.add(LSTM(units=100, return_sequences = False))
model.add(Dropout(0.2))

model.add(Dense(units =1))
model.compile(optimizer='adam',loss="mean_squared_error")
```

```
hist = model.fit(X_train, y_train, epochs = 20, batch_size = 32, verbose=2)
```

```
Epoch 1/20
35/35 - 18s - loss: 0.0286 - 18s/epoch - 516ms/step
Epoch 2/20
35/35 - 7s - loss: 0.0137 - 7s/epoch - 207ms/step
Epoch 3/20
35/35 - 11s - loss: 0.0120 - 11s/epoch - 314ms/step
Epoch 4/20
35/35 - 7s - loss: 0.0081 - 7s/epoch - 202ms/step
Epoch 5/20
35/35 - 9s - loss: 0.0074 - 9s/epoch - 256ms/step
Epoch 6/20
35/35 - 8s - loss: 0.0072 - 8s/epoch - 220ms/step
Epoch 7/20
35/35 - 8s - loss: 0.0100 - 8s/epoch - 237ms/step
Epoch 8/20
35/35 - 9s - loss: 0.0066 - 9s/epoch - 254ms/step
Epoch 9/20
35/35 - 7s - loss: 0.0061 - 7s/epoch - 204ms/step
Epoch 10/20
35/35 - 10s - loss: 0.0059 - 10s/epoch - 296ms/step
Epoch 11/20
35/35 - 8s - loss: 0.0059 - 8s/epoch - 234ms/step
Epoch 12/20
35/35 - 9s - loss: 0.0053 - 9s/epoch - 263ms/step
Epoch 13/20
35/35 - 9s - loss: 0.0059 - 9s/epoch - 252ms/step
Epoch 14/20
35/35 - 7s - loss: 0.0059 - 7s/epoch - 210ms/step
Epoch 15/20
35/35 - 9s - loss: 0.0052 - 9s/epoch - 259ms/step
Epoch 16/20
35/35 - 7s - loss: 0.0051 - 7s/epoch - 205ms/step
Epoch 17/20
35/35 - 9s - loss: 0.0055 - 9s/epoch - 260ms/step
Epoch 18/20
35/35 - 7s - loss: 0.0045 - 7s/epoch - 214ms/step
Epoch 19/20
35/35 - 9s - loss: 0.0045 - 9s/epoch - 248ms/step
Epoch 20/20
35/35 - 9s - loss: 0.0047 - 9s/epoch - 247ms/step
```

```
plt.plot(hist.history['loss'])
plt.title('Training model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train'], loc='upper left')
plt.show()
```



```
testData = pd.read_csv('/content/drive/MyDrive/datasets_LP5/Google_test_data.csv')
testData["Close"]=pd.to_numeric(testData.Close,errors='coerce')
testData = testData.dropna()
testData = testData.iloc[:,4:5]
y_test = testData.iloc[60:,0:].values
#input array for the model
inputClosing = testData.iloc[:,0:].values
inputClosing_scaled = sc.transform(inputClosing)
inputClosing_scaled.shape
X_test = []
length = len(testData)
timestep = 60
for i in range(timestep,length):
    X_test.append(inputClosing_scaled[i-timestep:i,0])
X_test = np.array(X_test)
X_test = np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))
X_test.shape
```

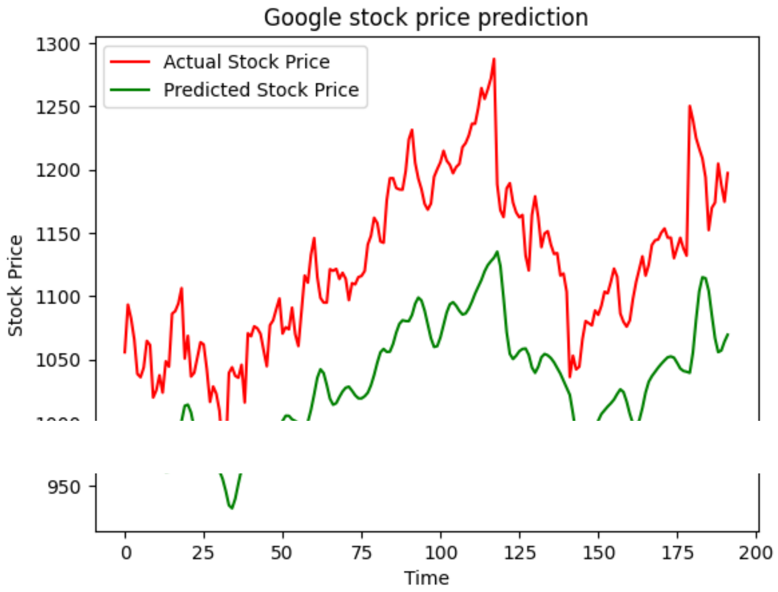
```
(192, 60, 1)
```

```
y_pred = model.predict(X_test)
import pandas as pd
df = pd.DataFrame(y_pred, columns = ['Predicted stock Value'])
print(df)
```

```
6/6 [==============================] - 2s 68ms/step
     Predicted stock Value
0                 1.135320
1                 1.136334
2                 1.147665
3                 1.164555
4                 1.175230
..                     ...
187               1.335884
188               1.310251
189               1.313293
190               1.329612
191               1.342119

[192 rows x 1 columns]
```

```
predicted_price = sc.inverse_transform(y_pred)
```

```
plt.plot(y_test, color = 'red', label = 'Actual Stock Price')
plt.plot(predicted_price, color = 'green', label = 'Predicted Stock Price')
plt.title('Google stock price prediction')
plt.xlabel('Time')
plt.ylabel('Stock Price')
plt.legend()
plt.show()
```



✓ 0s completed at 3:11 PM