In [1]:
```
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
```

In [2]:
```
data = pd.read_csv('Google_train_data.csv')
data.head()
```

Out[2]:

	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

In [3]:
```
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
```

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

In [5]:
```
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
```

In [6]:
```
sc = MinMaxScaler(feature_range=(0,1))
trainData = sc.fit_transform(trainData)
trainData.shape
```

Out[6]:
```
(1149, 1)
```

In [7]:
```
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)
```

In [8]:
```
X_train = np.reshape(X_train,(X_train.shape[0],X_train.shape[1],1)) #adding the batch_size axis
X_train.shape
```

Out[8]:
```
(1089, 60, 1)
```

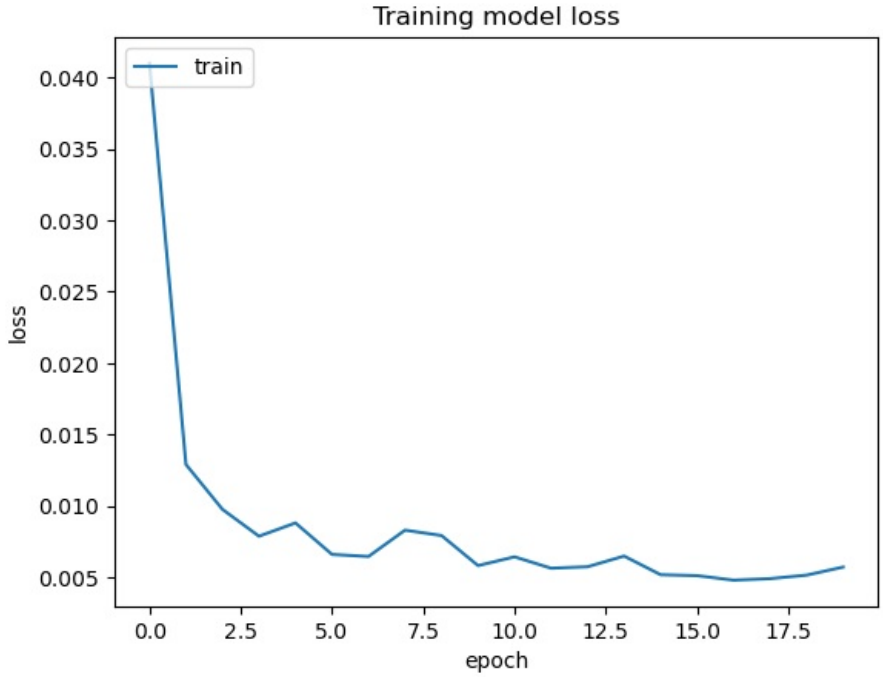
In [9]:
```
model = Sequential()

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")
```

In [10]:
```
hist = model.fit(X_train, y_train, epochs = 20, batch_size = 32, verbose=2)
```

```
Epoch 1/20
35/35 - 18s - loss: 0.0410 - 18s/epoch - 522ms/step
Epoch 2/20
35/35 - 4s - loss: 0.0129 - 4s/epoch - 124ms/step
Epoch 3/20
35/35 - 4s - loss: 0.0098 - 4s/epoch - 117ms/step
Epoch 4/20
35/35 - 4s - loss: 0.0079 - 4s/epoch - 113ms/step
Epoch 5/20
35/35 - 4s - loss: 0.0088 - 4s/epoch - 117ms/step
Epoch 6/20
35/35 - 4s - loss: 0.0066 - 4s/epoch - 125ms/step
Epoch 7/20
35/35 - 4s - loss: 0.0065 - 4s/epoch - 123ms/step
Epoch 8/20
35/35 - 4s - loss: 0.0083 - 4s/epoch - 122ms/step
Epoch 9/20
35/35 - 4s - loss: 0.0079 - 4s/epoch - 128ms/step
Epoch 10/20
35/35 - 4s - loss: 0.0058 - 4s/epoch - 127ms/step
Epoch 11/20
35/35 - 4s - loss: 0.0064 - 4s/epoch - 123ms/step
Epoch 12/20
35/35 - 4s - loss: 0.0057 - 4s/epoch - 128ms/step
Epoch 13/20
35/35 - 5s - loss: 0.0058 - 5s/epoch - 132ms/step
Epoch 14/20
35/35 - 4s - loss: 0.0065 - 4s/epoch - 125ms/step
Epoch 15/20
35/35 - 4s - loss: 0.0052 - 4s/epoch - 123ms/step
Epoch 16/20
35/35 - 5s - loss: 0.0051 - 5s/epoch - 129ms/step
Epoch 17/20
35/35 - 5s - loss: 0.0048 - 5s/epoch - 134ms/step
Epoch 18/20
35/35 - 4s - loss: 0.0049 - 4s/epoch - 126ms/step
Epoch 19/20
35/35 - 5s - loss: 0.0052 - 5s/epoch - 130ms/step
Epoch 20/20
35/35 - 5s - loss: 0.0057 - 5s/epoch - 130ms/step
```

In [11]:
```
plt.plot(hist.history['loss'])
plt.title('Training model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train'], loc='upper left')
plt.show()
```



In [12]:
```
testData = pd.read_csv('Google_test_data.csv')
```

In [12]:
```
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
```

Out[12]:
```
(192, 60, 1)
```

In [13]:
```
y_pred = model.predict(X_test)
y_pred
```

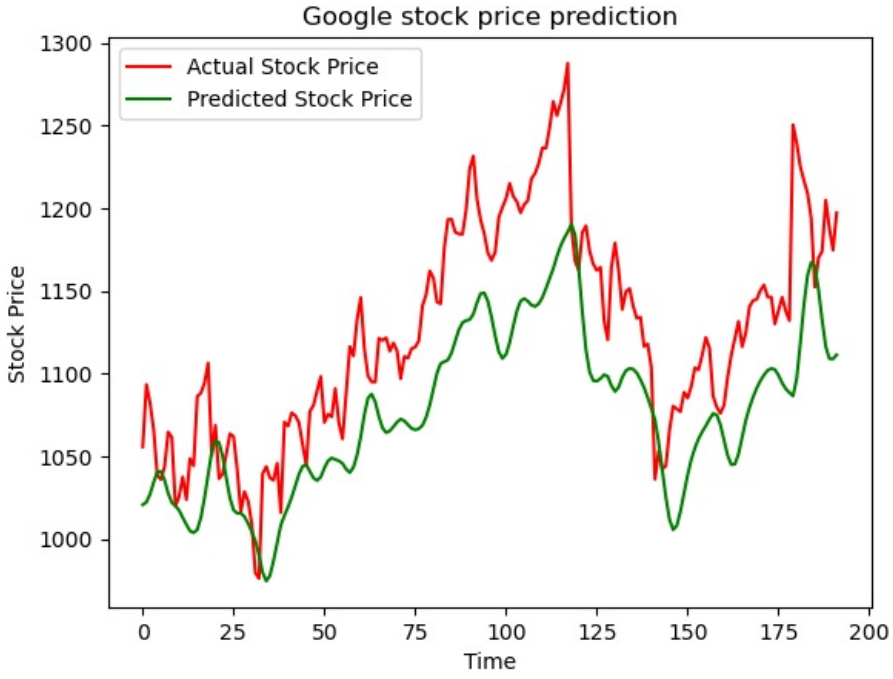
Out[13]:
```
6/6 [==============================] - 3s 49ms/step
array([[1.228829 ],
       [1.2324057],
       [1.2435347],
       [1.2603383],
       [1.2745281],
       [1.2757915],
       [1.2634943],
       [1.2450262],
       [1.2318469],
       [1.22733  ],
       [1.2210493],
       [1.2108315],
       [1.201168 ],
       [1.1921239],
       [1.1896354],
       [1.1937176],
       [1.2107441],
       [1.238451 ],
       [1.2708741],
       [1.3033404],
       [1.3182075],
       [1.3160553],
       [1.2960873],
       [1.2661567],
       [1.2381543],
       [1.2213371],
       [1.2168546],
       [1.2171685],
       [1.2120867],
       [1.2027179],
       [1.1913445],
       [1.1779636],
       [1.1582092],
       [1.1335381],
       [1.1218702],
       [1.1293883],
       [1.1503929],
       [1.175755 ],
       [1.2002282],
       [1.213703 ],
       [1.2256863],
       [1.2395592],
       [1.2558742],
       [1.2716153],
       [1.2827722],
       [1.2848774],
       [1.2758338],
       [1.2663727],
       [1.2628279],
       [1.2677188],
       [1.2803869],
       [1.2904111],
       [1.2943019],
       [1.2922245],
       [1.2903619],
       [1.2866232],
       [1.2786279],
       [1.2739633],
       [1.2814553],
       [1.2986537],
       [1.3242108],
       [1.3552634],
       [1.3783724],
       [1.384081 ],
       [1.3730997],
```

```
[1.3525435],
[1.3364729],
[1.3299943],
[1.3324331],
[1.3382453],
[1.3446915],
[1.348985 ],
[1.3464996],
[1.341139 ],
[1.3359693],
[1.3339168],
[1.3354504],
[1.3402269],
[1.3514566],
[1.3687694],
[1.3913292],
[1.4134883],
[1.426968 ],
[1.4295595],
[1.4318852],
[1.4419456],
[1.458434 ],
[1.4741606],
[1.4841614],
[1.4872159],
[1.4884336],
[1.4962959],
[1.5122342],
[1.5248644],
[1.5259094],
[1.5141618],
[1.492205 ],
[1.4659718],
[1.4433911],
[1.4340752],
[1.4397178],
[1.4565485],
[1.4798015],
[1.501039 ],
[1.5145904],
[1.5178795],
[1.5143944],
[1.5086881],
[1.5069504],
[1.51078  ],
[1.5195614],
[1.5324538],
[1.546037 ],
[1.5602311],
[1.5770125],
[1.5915608],
[1.6025995],
[1.6115571],
[1.6222723],
[1.6069208],
[1.5610399],
[1.4991636],
[1.446122 ],
[1.4150845],
[1.4030253],
[1.4021484],
[1.4057997],
[1.4111038],
[1.4087529],
[1.3958328],
[1.3877419],
[1.3937417],
[1.4077022],
[1.4169526],
[1.4199702],
[1.4189324],
[1.4136696],
[1.4042083],
[1.3931432],
[1.3789831],
[1.3640887],
[1.3480294],
[1.3169231],
[1.2779368],
[1.2390182],
[1.2080488],
[1.1937903],
[1.1994509],
[1.2194681],
[1.2449197],
[1.270917 ],
[1.2924753],
[1.308565 ],
[1.3214374],
```

```
[1.3312373],
[1.3396143],
[1.3488489],
[1.3567415],
[1.35477  ],
[1.3411698],
[1.3197696],
[1.2980295],
[1.2850486],
[1.2858505],
[1.300122 ],
[1.324147 ],
[1.3473235],
[1.3652092],
[1.3798498],
[1.3924788],
[1.4026672],
[1.4107989],
[1.4172354],
[1.4198717],
[1.4185153],
[1.4104654],
[1.3995644],
[1.3914303],
[1.3860489],
[1.3815197],
[1.4057149],
[1.4550912],
[1.5095196],
[1.5504295],
[1.5687686],
[1.5630621],
[1.530771 ],
[1.4882355],
[1.4507228],
[1.4336972],
[1.4333547],
[1.4390545]], dtype=float32)
```

In [14]:
```
predicted_price = sc.inverse_transform(y_pred)
```

In [15]:
```
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()
```



In [ ]: