

```
In [1]: import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
```

```
In [58]: data = pd.read_csv('GOOG.csv', date_parser = True)
data.tail()
```

```
Out[58]:
```

	Date	Open	High	Low	Close	Adj Close	Volume
3804	2019-09-30	1220.969971	1226.000000	1212.300049	1219.000000	1219.000000	1404100
3805	2019-10-01	1219.000000	1231.229980	1203.579956	1205.099976	1205.099976	1273500
3806	2019-10-02	1196.979980	1196.979980	1171.290039	1176.630005	1176.630005	1615100
3807	2019-10-03	1180.000000	1189.060059	1162.430054	1187.829956	1187.829956	1621200
3808	2019-10-04	1191.890015	1211.439941	1189.170044	1209.000000	1209.000000	1021092

```
In [3]: data_training = data[data['Date']<'2019-01-01'].copy()
data_test = data[data['Date']>='2019-01-01'].copy()
```

```
In [4]: data_training = data_training.drop(['Date', 'Adj Close'], axis = 1)
```

```
In [5]: scaler = MinMaxScaler()
data_training = scaler.fit_transform(data_training)
data_training
```

```
Out[5]: array([[3.30294890e-04, 9.44785459e-04, 0.00000000e+00, 1.34908021e-04,
   5.43577404e-01],
 [7.42148227e-04, 2.98909923e-03, 1.88269054e-03, 3.39307537e-03,
  2.77885613e-01],
 [4.71386886e-03, 4.78092896e-03, 5.42828241e-03, 3.83867225e-03,
  2.22150736e-01],
 ...,
 [7.92197108e-01, 8.11970141e-01, 7.90196475e-01, 8.15799920e-01,
  2.54672037e-02],
 [8.18777193e-01, 8.21510648e-01, 8.20249255e-01, 8.10219301e-01,
  1.70463908e-02],
 [8.19874096e-01, 8.19172449e-01, 8.12332341e-01, 8.09012935e-01,
  1.79975186e-02]])
```

```
In [6]: # create RNN with 60 timesteps, i.e. look 60 previous time steps
```

```
In [7]: data_training[0:10]
```

```
Out[7]: array([[3.30294890e-04, 9.44785459e-04, 0.00000000e+00, 1.34908021e-04,
   5.43577404e-01],
   [7.42148227e-04, 2.98909923e-03, 1.88269054e-03, 3.39307537e-03,
   2.77885613e-01],
   [4.71386886e-03, 4.78092896e-03, 5.42828241e-03, 3.83867225e-03,
   2.22150736e-01],
   [4.91367646e-03, 4.01532941e-03, 3.15578542e-03, 1.98678849e-03,
   1.85522018e-01],
   [2.35285614e-03, 2.54928676e-03, 3.28434064e-03, 2.44873974e-03,
   1.11762967e-01],
   [2.34877785e-03, 2.52892558e-03, 3.60779701e-03, 3.22955376e-03,
   8.62763771e-02],
   [3.63326671e-03, 2.80177162e-03, 4.03492722e-03, 2.51005881e-03,
   7.55243925e-02],
   [2.48334262e-03, 1.52712947e-03, 2.50886935e-03, 8.17608079e-04,
   6.31682127e-02],
   [1.26817570e-03, 8.02253103e-04, 2.57107531e-03, 9.64778600e-04,
   5.97732318e-02],
   [1.43128522e-03, 5.00900100e-04, 1.53849690e-03, 9.81131336e-05,
   1.11151095e-01]])
```

```
In [8]: X_train = []
y_train = []
```

```
In [9]: for i in range(60, data_training.shape[0]):
    X_train.append(data_training[i-60:i])
    y_train.append(data_training[i, 0])
```

```
In [10]: X_train, y_train = np.array(X_train), np.array(y_train)
```

```
In [11]: X_train.shape
```

```
Out[11]: (3557, 60, 5)
```

Building LSTM

```
In [14]: from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
```

```
In [42]: regressor = Sequential()

regressor.add(LSTM(units = 60, activation = 'relu', return_sequences = True, input_shape=(60,5)))
regressor.add(Dropout(0.2))

regressor.add(LSTM(units = 60, activation = 'relu', return_sequences = True))
regressor.add(Dropout(0.2))

regressor.add(LSTM(units = 80, activation = 'relu', return_sequences = True))
regressor.add(Dropout(0.2))

regressor.add(LSTM(units = 120, activation = 'relu'))
regressor.add(Dropout(0.2))

regressor.add(Dense(units = 1))
```

```
In [43]: regressor.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
<hr/>		
lstm_5 (LSTM)	(None, 60, 60)	15840
<hr/>		
dropout_4 (Dropout)	(None, 60, 60)	0
<hr/>		
lstm_6 (LSTM)	(None, 60, 60)	29040
<hr/>		
dropout_5 (Dropout)	(None, 60, 60)	0
<hr/>		
lstm_7 (LSTM)	(None, 60, 80)	45120
<hr/>		
dropout_6 (Dropout)	(None, 60, 80)	0
<hr/>		
lstm_8 (LSTM)	(None, 120)	96480
<hr/>		
dropout_7 (Dropout)	(None, 120)	0
<hr/>		
dense_1 (Dense)	(None, 1)	121
<hr/>		
Total params: 186,601		
Trainable params: 186,601		
Non-trainable params: 0		

In [44]: regressor.compile(optimizer='adam', loss = 'mean_squared_error')

In [45]: regressor.fit(X_train, y_train, epochs=50, batch_size=32)

Train on 3557 samples

Epoch 1/50
3557/3557 [=====] - 16s 5ms/sample - loss: 0.0137

Epoch 2/50
3557/3557 [=====] - 12s 3ms/sample - loss: 0.0022

Epoch 3/50
3557/3557 [=====] - 12s 3ms/sample - loss: 0.0018

Epoch 4/50
3557/3557 [=====] - 12s 3ms/sample - loss: 0.0016

Epoch 5/50
3557/3557 [=====] - 12s 3ms/sample - loss: 0.0016

Epoch 6/50
3557/3557 [=====] - 12s 3ms/sample - loss: 0.0016

Epoch 7/50
3557/3557 [=====] - 12s 3ms/sample - loss: 0.0014

Epoch 8/50
3557/3557 [=====] - 12s 3ms/sample - loss: 0.0016

Epoch 9/50
3557/3557 [=====] - 12s 3ms/sample - loss: 0.0013

Epoch 10/50
3557/3557 [=====] - 12s 3ms/sample - loss: 0.0013

Epoch 11/50
3557/3557 [=====] - 12s 3ms/sample - loss: 0.0013

Epoch 12/50
3557/3557 [=====] - 12s 3ms/sample - loss: 0.0013

Epoch 13/50
3557/3557 [=====] - 17s 5ms/sample - loss: 0.0013

Epoch 14/50
3557/3557 [=====] - 18s 5ms/sample - loss: 0.0011

Epoch 15/50
3557/3557 [=====] - 14s 4ms/sample - loss: 0.0012

Epoch 16/50
3557/3557 [=====] - 13s 4ms/sample - loss: 9.7986e-04

Epoch 17/50
3557/3557 [=====] - 13s 4ms/sample - loss: 0.0011

Epoch 18/50
3557/3557 [=====] - 14s 4ms/sample - loss: 0.0010

Epoch 19/50
3557/3557 [=====] - 15s 4ms/sample - loss: 8.0842e-04

Epoch 20/50
3557/3557 [=====] - 14s 4ms/sample - loss: 9.6403e-04

Epoch 21/50
3557/3557 [=====] - 13s 4ms/sample - loss: 9.2826e-04

Epoch 22/50
3557/3557 [=====] - 14s 4ms/sample - loss: 9.3406e-04

Epoch 23/50
3557/3557 [=====] - 14s 4ms/sample - loss: 9.3298e-04

Epoch 24/50
3557/3557 [=====] - 14s 4ms/sample - loss: 8.5449e-04

Epoch 25/50
3557/3557 [=====] - 14s 4ms/sample - loss: 9.3350e-04

Epoch 26/50
3557/3557 [=====] - 13s 4ms/sample - loss: 8.9023e-04

Epoch 27/50
3557/3557 [=====] - 13s 4ms/sample - loss: 9.0078e-04

Epoch 28/50
3557/3557 [=====] - 13s 4ms/sample - loss: 8.7865e-04

Epoch 29/50
3557/3557 [=====] - 13s 4ms/sample - loss: 7.7264e-04

Epoch 30/50

```
3557/3557 [=====] - 13s 4ms/sample - loss: 6.7656e-04
Epoch 31/50
3557/3557 [=====] - 13s 4ms/sample - loss: 8.1103e-04
Epoch 32/50
3557/3557 [=====] - 13s 4ms/sample - loss: 8.3787e-04
Epoch 33/50
3557/3557 [=====] - 13s 4ms/sample - loss: 7.0893e-04
Epoch 34/50
3557/3557 [=====] - 13s 4ms/sample - loss: 7.5235e-04
Epoch 35/50
3557/3557 [=====] - 13s 4ms/sample - loss: 7.4276e-04
Epoch 36/50
3557/3557 [=====] - 13s 4ms/sample - loss: 7.5183e-04
Epoch 37/50
3557/3557 [=====] - 13s 4ms/sample - loss: 7.6802e-04
Epoch 38/50
3557/3557 [=====] - 13s 4ms/sample - loss: 6.9164e-04
Epoch 39/50
3557/3557 [=====] - 13s 4ms/sample - loss: 6.8079e-04
Epoch 40/50
3557/3557 [=====] - 13s 4ms/sample - loss: 6.7066e-04
Epoch 41/50
3557/3557 [=====] - 14s 4ms/sample - loss: 7.2075e-04
Epoch 42/50
3557/3557 [=====] - 14s 4ms/sample - loss: 7.1259e-04
Epoch 43/50
3557/3557 [=====] - 13s 4ms/sample - loss: 7.1577e-04
Epoch 44/50
3557/3557 [=====] - 13s 4ms/sample - loss: 6.5169e-04
Epoch 45/50
3557/3557 [=====] - 13s 4ms/sample - loss: 6.5112e-04
Epoch 46/50
3557/3557 [=====] - 13s 4ms/sample - loss: 6.0908e-04
Epoch 47/50
3557/3557 [=====] - 15s 4ms/sample - loss: 6.6632e-04
Epoch 48/50
3557/3557 [=====] - 15s 4ms/sample - loss: 6.9701e-04
Epoch 49/50
3557/3557 [=====] - 16s 4ms/sample - loss: 6.2277e-04
Epoch 50/50
3557/3557 [=====] - 16s 4ms/sample - loss: 6.4571e-04
Out[45]: <tensorflow.python.keras.callbacks.History at 0x230c796f940>
```

Prepare test dataset

```
In [65]: data_test.head()
```

```
Out[65]:
```

	Date	Open	High	Low	Close	Adj Close	Volume
3617	2019-01-02	1016.570007	1052.319946	1015.710022	1045.849976	1045.849976	1532600
3618	2019-01-03	1041.000000	1056.979980	1014.070007	1016.059998	1016.059998	1841100
3619	2019-01-04	1032.589966	1070.839966	1027.417969	1070.709961	1070.709961	2093900
3620	2019-01-07	1071.500000	1074.000000	1054.760010	1068.390015	1068.390015	1981900
3621	2019-01-08	1076.109985	1084.560059	1060.530029	1076.280029	1076.280029	1764900

```
In [66]: data_training.tail(60)
```

Out[66]:

	Date	Open	High	Low	Close	Adj Close	Volume
3557	2018-10-04	1195.329956	1197.510010	1155.576050	1168.189941	1168.189941	2209500
3558	2018-10-05	1167.500000	1173.500000	1145.119995	1157.349976	1157.349976	1184300
3559	2018-10-08	1150.109985	1168.000000	1127.364014	1148.969971	1148.969971	1932400
3560	2018-10-09	1146.150024	1154.349976	1137.572021	1138.819946	1138.819946	1308700
3561	2018-10-10	1131.079956	1132.170044	1081.130005	1081.219971	1081.219971	2675700
3562	2018-10-11	1072.939941	1106.400024	1068.270020	1079.319946	1079.319946	2949000
3563	2018-10-12	1108.000000	1115.000000	1086.401978	1110.079956	1110.079956	2101300
3564	2018-10-15	1108.910034	1113.446045	1089.000000	1092.250000	1092.250000	1372400
3565	2018-10-16	1104.589966	1124.219971	1102.500000	1121.280029	1121.280029	1928500
3566	2018-10-17	1126.459961	1128.989990	1102.189941	1115.689941	1115.689941	1467200
3567	2018-10-18	1121.839966	1121.839966	1077.089966	1087.969971	1087.969971	2094500
3568	2018-10-19	1093.369995	1110.359985	1087.750000	1096.459961	1096.459961	1267600
3569	2018-10-22	1103.060059	1112.229980	1091.000000	1101.160034	1101.160034	1514200
3570	2018-10-23	1080.890015	1107.890015	1070.000000	1103.689941	1103.689941	1848700
3571	2018-10-24	1104.250000	1106.119995	1048.739990	1050.709961	1050.709961	1982400
3572	2018-10-25	1071.790039	1110.979980	1069.550049	1095.569946	1095.569946	2545800
3573	2018-10-26	1037.030029	1106.530029	1034.089966	1071.469971	1071.469971	4187600
3574	2018-10-29	1082.469971	1097.040039	995.830017	1020.080017	1020.080017	3880700
3575	2018-10-30	1008.460022	1037.489990	1000.750000	1036.209961	1036.209961	3212700
3576	2018-10-31	1059.810059	1091.939941	1057.000000	1076.770020	1076.770020	2529800
3577	2018-11-01	1075.800049	1083.974976	1062.459961	1070.000000	1070.000000	1482000
3578	2018-11-02	1073.729980	1082.974976	1054.609985	1057.790039	1057.790039	1839000
3579	2018-11-05	1055.000000	1058.469971	1021.239990	1040.089966	1040.089966	2441400
3580	2018-11-06	1039.479980	1064.344971	1038.069946	1055.810059	1055.810059	1233300
3581	2018-11-07	1069.000000	1095.459961	1065.900024	1093.390015	1093.390015	2058400
3582	2018-11-08	1091.380005	1093.270020	1072.204956	1082.400024	1082.400024	1488200
3583	2018-11-09	1073.989990	1075.560059	1053.109985	1066.150024	1066.150024	1343200
3584	2018-11-12	1061.390015	1062.119995	1031.000000	1038.630005	1038.630005	1471800
3585	2018-11-13	1043.290039	1056.604980	1031.150024	1036.050049	1036.050049	1513700
3586	2018-11-14	1050.000000	1054.563965	1031.000000	1043.660034	1043.660034	1565900
3587	2018-11-15	1044.709961	1071.849976	1031.780029	1064.709961	1064.709961	1836100
3588	2018-11-16	1059.410034	1067.000000	1048.979980	1061.489990	1061.489990	1658100
3589	2018-11-19	1057.199951	1060.790039	1016.260010	1020.000000	1020.000000	1858600
3590	2018-11-20	1000.000000	1031.739990	996.020020	1025.760010	1025.760010	2449100

	Date	Open	High	Low	Close	Adj Close	Volume
3591	2018-11-21	1036.760010	1048.560059	1033.469971	1037.609985	1037.609985	1534300
3592	2018-11-23	1030.000000	1037.589966	1022.398987	1023.880005	1023.880005	691500
3593	2018-11-26	1038.349976	1049.310059	1033.910034	1048.619995	1048.619995	1942800
3594	2018-11-27	1041.000000	1057.579956	1038.489990	1044.410034	1044.410034	1803200
3595	2018-11-28	1048.760010	1086.839966	1035.760010	1086.229980	1086.229980	2475400
3596	2018-11-29	1076.079956	1094.244995	1076.000000	1088.300049	1088.300049	1468900
3597	2018-11-30	1089.069946	1095.569946	1077.880005	1094.430054	1094.430054	2580200
3598	2018-12-03	1123.140015	1124.650024	1103.665039	1106.430054	1106.430054	1991200
3599	2018-12-04	1103.119995	1104.420044	1049.979980	1050.819946	1050.819946	2345200
3600	2018-12-06	1034.260010	1071.199951	1030.770020	1068.729980	1068.729980	2769200
3601	2018-12-07	1060.010010	1075.260010	1028.500000	1036.579956	1036.579956	2101200
3602	2018-12-10	1035.050049	1048.449951	1023.289978	1039.550049	1039.550049	1807700
3603	2018-12-11	1056.489990	1060.599976	1039.839966	1051.750000	1051.750000	1394700
3604	2018-12-12	1068.000000	1081.650024	1062.790039	1063.680054	1063.680054	1523800
3605	2018-12-13	1068.069946	1079.760010	1053.930054	1061.900024	1061.900024	1329800
3606	2018-12-14	1049.979980	1062.599976	1040.790039	1042.099976	1042.099976	1686600
3607	2018-12-17	1037.510010	1053.150024	1007.900024	1016.530029	1016.530029	2385400
3608	2018-12-18	1026.089966	1049.479980	1021.440002	1028.709961	1028.709961	2192500
3609	2018-12-19	1033.989990	1062.000000	1008.049988	1023.010010	1023.010010	2479300
3610	2018-12-20	1018.130005	1034.219971	996.359985	1009.409973	1009.409973	2673500
3611	2018-12-21	1015.299988	1024.020020	973.690002	979.539978	979.539978	4596000
3612	2018-12-24	973.900024	1003.539978	970.109985	976.219971	976.219971	1590300
3613	2018-12-26	989.010010	1040.000000	983.000000	1039.459961	1039.459961	2373300
3614	2018-12-27	1017.150024	1043.890015	997.000000	1043.880005	1043.880005	2109800
3615	2018-12-28	1049.619995	1055.560059	1033.099976	1037.079956	1037.079956	1414800
3616	2018-12-31	1050.959961	1052.699951	1023.590027	1035.609985	1035.609985	1493300

In [67]: `past_60_days = data_training.tail(60)`

In [68]: `df = past_60_days.append(data_test, ignore_index = True)
df = df.drop(['Date', 'Adj Close'], axis = 1)
df.head()`

```
Out[68]:
```

	Open	High	Low	Close	Volume
0	1195.329956	1197.510010	1155.576050	1168.189941	2209500
1	1167.500000	1173.500000	1145.119995	1157.349976	1184300
2	1150.109985	1168.000000	1127.364014	1148.969971	1932400
3	1146.150024	1154.349976	1137.572021	1138.819946	1308700
4	1131.079956	1132.170044	1081.130005	1081.219971	2675700

```
In [69]: inputs = scaler.transform(df)
inputs
```

```
Out[69]: array([[0.93805611, 0.93755773, 0.92220906, 0.91781776, 0.0266752 ],
 [0.91527437, 0.91792904, 0.91350452, 0.90892169, 0.01425359],
 [0.90103881, 0.91343268, 0.89872289, 0.90204445, 0.02331778],
 ...,
 [0.93940683, 0.93712442, 0.93529076, 0.9247443 , 0.01947328],
 [0.92550693, 0.93064972, 0.92791493, 0.9339358 , 0.01954719],
 [0.93524016, 0.94894575, 0.95017564, 0.95130949, 0.01227612]])
```

```
In [70]: X_test = []
y_test = []

for i in range(60, inputs.shape[0]):
    X_test.append(inputs[i-60:i])
    y_test.append(inputs[i, 0])
```

```
In [71]: X_test, y_test = np.array(X_test), np.array(y_test)
X_test.shape, y_test.shape
```

```
Out[71]: ((192, 60, 5), (192,))
```

```
In [72]: y_pred = regressor.predict(X_test)
```

```
In [73]: scaler.scale_
```

```
Out[73]: array([8.18605127e-04, 8.17521128e-04, 8.32487534e-04, 8.20673293e-04,
 1.21162775e-08])
```

```
In [74]: scale = 1/8.18605127e-04
scale
```

```
Out[74]: 1221.5901990069017
```

```
In [75]: y_pred = y_pred*scale
y_test = y_test*scale
```

Visualization

```
In [76]: # Visualising the results
plt.figure(figsize=(14,5))
plt.plot(y_test, color = 'red', label = 'Real Google Stock Price')
plt.plot(y_pred, 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()
```

