```
S.Mohanraj
212221230065
```

S.Mohanraj 212221230065

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

```
dataset_train = pd.read_csv('trainset.csv')
```

```
dataset_train.columns
```

```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype='object')
```

```
dataset_train.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2013-01-02	357.385559	361.151062	355.959839	359.288177	359.288177	5115500
1	2013-01-03	360.122742	363.600128	358.031342	359.496826	359.496826	4666500
2	2013-01-04	362.313507	368.339294	361.488861	366.600616	366.600616	5562800
3	2013-01-07	365.348755	367.301056	362.929504	365.001007	365.001007	3332900
4	2013-01-08	365.393463	365.771027	359.874359	364.280701	364.280701	3373900

Next steps: Generate code with `dataset_train` View recommended plots

```
train_set = dataset_train.iloc[:,1:2].values
```

```
type(train_set)
```

```
numpy.ndarray
```

```
train_set.shape
```

```
(1259, 1)
```

```
sc = MinMaxScaler(feature_range=(0,1))
training_set_scaled = sc.fit_transform(train_set)
```

```
training_set_scaled.shape
```

```
(1259, 1)
```

```
X_train_array = []
y_train_array = []
for i in range(60, 1259):
  X_train_array.append(training_set_scaled[i-60:i,0])
  y_train_array.append(training_set_scaled[i,0])
X_train, y_train = np.array(X_train_array), np.array(y_train_array)
X_train1 = X_train.reshape((X_train.shape[0], X_train.shape[1],1))
```

```
X_train.shape
```

```
(1199, 60)
```

```
length = 60
n_features = 1
```

```
model = Sequential()
model.add(layers.SimpleRNN(45,input_shape=(length,n_features)))
model.add(layers.Dense(1))

model.compile(optimizer='adam', loss='mse')

print("Name:S.Mohanraj")
print("Register Number: 212221230065")
model.summary()
```

```
Name:S.Mohanraj
Register Number: 212221230065
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 simple_rnn (SimpleRNN)      (None, 45)                2115      
                                                                 
 dense (Dense)               (None, 1)                 46        
                                                                 
=================================================================
Total params: 2161 (8.44 KB)
Trainable params: 2161 (8.44 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
model.fit(X_train1,y_train,epochs=100, batch_size=32)
```

```
Epoch 1/100
38/38 [==============================] - 2s 11ms/step - loss: 0.0510
Epoch 2/100
38/38 [==============================] - 0s 11ms/step - loss: 0.0026
Epoch 3/100
38/38 [==============================] - 0s 11ms/step - loss: 0.0019
Epoch 4/100
38/38 [==============================] - 0s 11ms/step - loss: 0.0016
Epoch 5/100
38/38 [==============================] - 0s 10ms/step - loss: 0.0015
Epoch 6/100
38/38 [==============================] - 0s 10ms/step - loss: 0.0013
Epoch 7/100
38/38 [==============================] - 0s 10ms/step - loss: 0.0012
Epoch 8/100
38/38 [==============================] - 0s 10ms/step - loss: 0.0011
Epoch 9/100
38/38 [==============================] - 0s 11ms/step - loss: 0.0010
Epoch 10/100
38/38 [==============================] - 0s 10ms/step - loss: 9.7325e-04
Epoch 11/100
38/38 [==============================] - 1s 18ms/step - loss: 9.1160e-04
Epoch 12/100
38/38 [==============================] - 1s 17ms/step - loss: 8.7110e-04
Epoch 13/100
38/38 [==============================] - 1s 18ms/step - loss: 8.6447e-04
Epoch 14/100
38/38 [==============================] - 1s 22ms/step - loss: 8.0017e-04
Epoch 15/100
38/38 [==============================] - 0s 11ms/step - loss: 7.5042e-04
Epoch 16/100
38/38 [==============================] - 0s 11ms/step - loss: 7.2336e-04
Epoch 17/100
38/38 [==============================] - 0s 11ms/step - loss: 7.2214e-04
Epoch 18/100
38/38 [==============================] - 0s 11ms/step - loss: 6.9184e-04
Epoch 19/100
38/38 [==============================] - 0s 11ms/step - loss: 6.8908e-04
Epoch 20/100
38/38 [==============================] - 0s 10ms/step - loss: 7.1505e-04
Epoch 21/100
38/38 [==============================] - 0s 11ms/step - loss: 6.2789e-04
```

```
Epoch 22/100
38/38 [==============================] - 0s 10ms/step - loss: 5.9943e-04
Epoch 23/100
38/38 [==============================] - 0s 10ms/step - loss: 6.6371e-04
Epoch 24/100
38/38 [==============================] - 0s 11ms/step - loss: 5.9087e-04
Epoch 25/100
38/38 [==============================] - 0s 10ms/step - loss: 6.3415e-04
Epoch 26/100
38/38 [==============================] - 0s 11ms/step - loss: 5.4748e-04
Epoch 27/100
38/38 [==============================] - 0s 11ms/step - loss: 6.4649e-04
Epoch 28/100
38/38 [==============================] - 0s 11ms/step - loss: 5.3382e-04
Epoch 29/100
38/38 [==============================] - 0s 11ms/step - loss: 5.2838e-04
```

```
dataset_test = pd.read_csv('testset.csv')
```

```
test_set = dataset_test.iloc[:,1:2].values
```

```
test_set.shape
```

```
(125, 1)
```

```
dataset_total = pd.concat((dataset_train['Open'],dataset_test['Open']),axis=0)
```

```
inputs = dataset_total.values
inputs = inputs.reshape(-1,1)
inputs_scaled=sc.transform(inputs)
X_test = []
for i in range(60,1384):
  X_test.append(inputs_scaled[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))
```

```
X_test.shape
```

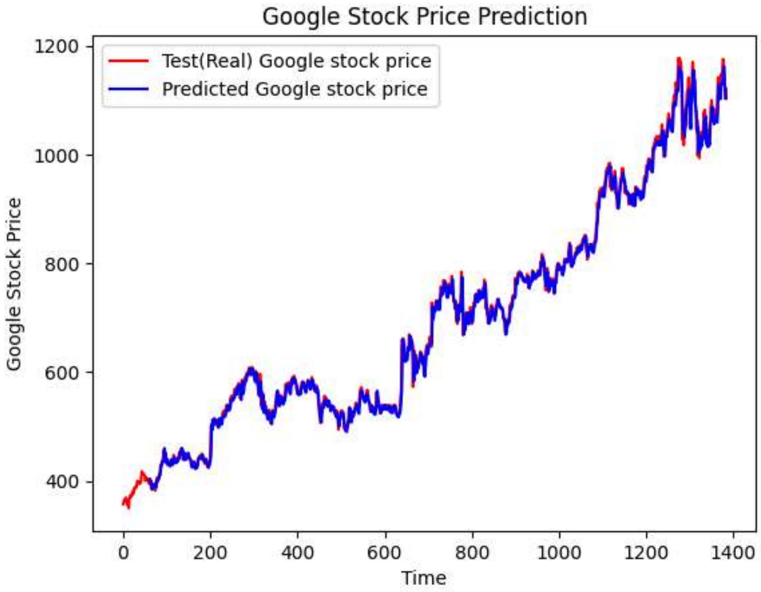
```
(1324, 60, 1)
```

```
predicted_stock_price_scaled = model.predict(X_test)
predicted_stock_price = sc.inverse_transform(predicted_stock_price_scaled)
```

```
42/42 [==============================] - 0s 4ms/step
```

```
print("Name:S.Mohanraj         Register Number:212221230065     ")
plt.plot(np.arange(0,1384),inputs, color='red', label = 'Test(Real) Google stock price')
plt.plot(np.arange(60,1384),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()
```

Name:S.Mohanraj Register Number:212221230065


Google Stock Price Prediction
Test(Real) Google stock price
Predicted Google stock price
Google Stock Price
Time