Ex05 : Stock Price Prediction

```
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

```
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(50,input_shape=(length,n_features)))
model.add(layers.Dense(1))

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

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (None, 50)                2600

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 2,651
Trainable params: 2,651
Non-trainable params: 0
_________________________________________________________________
```

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

```
Epoch 72/100
38/38 [==============================] - 0s 13ms/step - loss: 1.8681e-04
Epoch 73/100
38/38 [==============================] - 0s 12ms/step - loss: 1.7412e-04
Epoch 74/100
38/38 [==============================] - 0s 12ms/step - loss: 1.9892e-04
Epoch 75/100
38/38 [==============================] - 0s 13ms/step - loss: 2.0079e-04
Epoch 76/100
38/38 [==============================] - 0s 12ms/step - loss: 1.9732e-04
Epoch 77/100
38/38 [==============================] - 0s 12ms/step - loss: 1.8932e-04
Epoch 78/100
38/38 [==============================] - 0s 12ms/step - loss: 1.8993e-04
Epoch 79/100
38/38 [==============================] - 0s 12ms/step - loss: 1.7539e-04
Epoch 80/100
```

```
38/38 [==============================] - 0s 13ms/step - loss: 1.7513e-04
Epoch 81/100
38/38 [==============================] - 1s 14ms/step - loss: 1.9510e-04
Epoch 82/100
38/38 [==============================] - 1s 14ms/step - loss: 1.9118e-04
Epoch 83/100
38/38 [==============================] - 1s 13ms/step - loss: 1.8032e-04
Epoch 84/100
38/38 [==============================] - 0s 13ms/step - loss: 1.7188e-04
Epoch 85/100
38/38 [==============================] - 1s 13ms/step - loss: 1.6829e-04
Epoch 86/100
38/38 [==============================] - 1s 13ms/step - loss: 1.7517e-04
Epoch 87/100
38/38 [==============================] - 0s 13ms/step - loss: 2.1892e-04
Epoch 88/100
38/38 [==============================] - 1s 13ms/step - loss: 1.9083e-04
Epoch 89/100
38/38 [==============================] - 1s 13ms/step - loss: 1.7601e-04
Epoch 90/100
38/38 [==============================] - 0s 13ms/step - loss: 1.7040e-04
Epoch 91/100
38/38 [==============================] - 0s 12ms/step - loss: 1.7095e-04
Epoch 92/100
38/38 [==============================] - 0s 12ms/step - loss: 1.6537e-04
Epoch 93/100
38/38 [==============================] - 0s 12ms/step - loss: 1.6988e-04
Epoch 94/100
38/38 [==============================] - 0s 12ms/step - loss: 1.6980e-04
Epoch 95/100
38/38 [==============================] - 1s 13ms/step - loss: 1.6139e-04
Epoch 96/100
38/38 [==============================] - 0s 12ms/step - loss: 1.7332e-04
Epoch 97/100
38/38 [==============================] - 0s 13ms/step - loss: 1.6797e-04
Epoch 98/100
38/38 [==============================] - 0s 13ms/step - loss: 1.6506e-04
Epoch 99/100
38/38 [==============================] - 0s 12ms/step - loss: 1.6302e-04
Epoch 100/100
38/38 [==============================] - 0s 12ms/step - loss: 1.7935e-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))
```

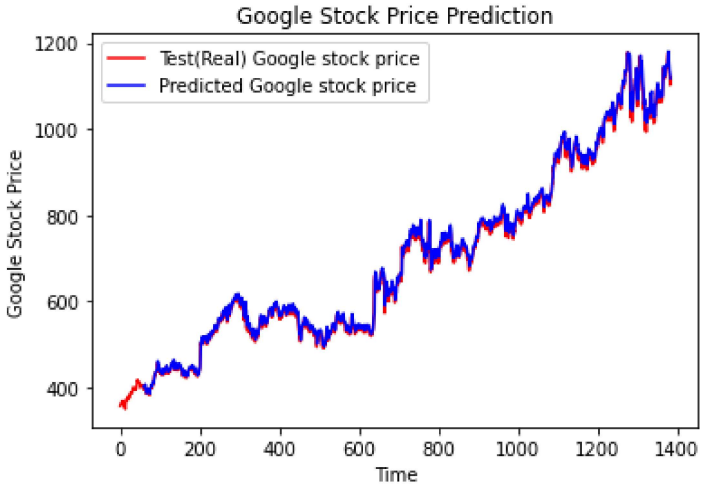
```
X_test.shape
```

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

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

```
42/42 [==============================] - 0s 5ms/step
```

```
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()
```




```
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((-1,60,1))
```

```
X_train.shape
```

```
(1199, 60)
```

```
X_train1.shape
```

```
(1199, 60, 1)
```

```
length = 60
n_features = 1


model = Sequential()
model.add(layers.SimpleRNN(50,input_shape=(length,n_features)))
model.add(layers.Dense(1))

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


model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn_1 (SimpleRNN)    (None, 50)                2600

 dense_1 (Dense)             (None, 1)                 51

=================================================================
Total params: 2,651
Trainable params: 2,651
Non-trainable params: 0
_________________________________________________________________
```

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

```
Epoch 1/100
38/38 [==============================] - 1s 13ms/step - loss: 0.0453
Epoch 2/100
38/38 [==============================] - 1s 13ms/step - loss: 0.0019
Epoch 3/100
38/38 [==============================] - 1s 13ms/step - loss: 0.0012
Epoch 4/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0010
Epoch 5/100
38/38 [==============================] - 0s 13ms/step - loss: 8.9347e-04
Epoch 6/100
38/38 [==============================] - 0s 13ms/step - loss: 9.0204e-04
Epoch 7/100
38/38 [==============================] - 0s 12ms/step - loss: 7.8451e-04
Epoch 8/100
38/38 [==============================] - 0s 13ms/step - loss: 7.0413e-04
Epoch 9/100
38/38 [==============================] - 0s 13ms/step - loss: 6.6012e-04
Epoch 10/100
38/38 [==============================] - 0s 13ms/step - loss: 6.0176e-04
Epoch 11/100
38/38 [==============================] - 0s 13ms/step - loss: 6.6600e-04
Epoch 12/100
38/38 [==============================] - 1s 13ms/step - loss: 5.6487e-04
Epoch 13/100
38/38 [==============================] - 0s 12ms/step - loss: 5.4198e-04
Epoch 14/100
38/38 [==============================] - 0s 12ms/step - loss: 5.1287e-04
Epoch 15/100
38/38 [==============================] - 0s 12ms/step - loss: 4.7806e-04
Epoch 16/100
38/38 [==============================] - 0s 13ms/step - loss: 4.7933e-04
```

```
Epoch 17/100
38/38 [==============================] - 0s 13ms/step - loss: 4.6309e-04
Epoch 18/100
38/38 [==============================] - 1s 14ms/step - loss: 4.4885e-04
Epoch 19/100
38/38 [==============================] - 0s 12ms/step - loss: 4.2817e-04
Epoch 20/100
38/38 [==============================] - 0s 12ms/step - loss: 4.7541e-04
Epoch 21/100
38/38 [==============================] - 0s 13ms/step - loss: 5.5473e-04
Epoch 22/100
38/38 [==============================] - 0s 13ms/step - loss: 4.1189e-04
Epoch 23/100
38/38 [==============================] - 1s 14ms/step - loss: 3.7589e-04
Epoch 24/100
38/38 [==============================] - 0s 12ms/step - loss: 4.1870e-04
Epoch 25/100
38/38 [==============================] - 0s 13ms/step - loss: 3.7740e-04
Epoch 26/100
38/38 [==============================] - 0s 12ms/step - loss: 3.7098e-04
Epoch 27/100
38/38 [==============================] - 0s 13ms/step - loss: 3.5779e-04
Epoch 28/100
38/38 [==============================] - 0s 12ms/step - loss: 3.6157e-04
Epoch 29/100
38/38 [==============================] - 0s 13ms/step - loss: 3.5012e-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 5ms/step
```

```
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()
```