**DL Practical No. 3**

**Code :**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import SimpleRNN, Dense

**# Step 1: Load and clean training data**

df\_train = pd.read\_csv(r'C:\Users\ASUS\Downloads\Google\_stock\_price\Google\_Stock\_Price\_Train.csv')

# Convert 'Open' to float if it's a string with commas

if df\_train['Open'].dtype == 'object':

df\_train['Open'] = df\_train['Open'].str.replace(',', '').astype(float)

**# Step 2: Scale training data**

training\_set = df\_train['Open'].values.reshape(-1, 1)

scaler = MinMaxScaler(feature\_range=(0, 1))

training\_set\_scaled = scaler.fit\_transform(training\_set)

**# Step 3: Create sequences for RNN**

X\_train, y\_train = [], []

time\_step = 60

for i in range(time\_step, len(training\_set\_scaled)):

X\_train.append(training\_set\_scaled[i-time\_step:i, 0])

y\_train.append(training\_set\_scaled[i, 0])

X\_train, y\_train = np.array(X\_train), np.array(y\_train)

X\_train = X\_train.reshape((X\_train.shape[0], X\_train.shape[1], 1))

**# Step 4: Build and train the RNN model**

model = Sequential()

model.add(SimpleRNN(units=50, activation='tanh', return\_sequences=False, input\_shape=(X\_train.shape[1], 1)))

model.add(Dense(1))

model.compile(optimizer='adam', loss='mean\_squared\_error')

model.fit(X\_train, y\_train, epochs=20, batch\_size=32)

**# Step 5: Load and clean test data**

df\_test = pd.read\_csv(r'C:\Users\ASUS\Downloads\Google\_stock\_price\Google\_Stock\_Price\_Test.csv')

if df\_test['Open'].dtype == 'object':

df\_test['Open'] = df\_test['Open'].str.replace(',', '').astype(float)

real\_stock\_price = df\_test['Open'].values.reshape(-1, 1)

**# Step 6: Prepare test input**

total\_data = pd.concat((df\_train['Open'], df\_test['Open']), axis=0)

inputs = total\_data[len(total\_data) - len(df\_test) - time\_step:].values

inputs = inputs.reshape(-1, 1)

inputs\_scaled = scaler.transform(inputs)

X\_test = []

for i in range(time\_step, len(inputs\_scaled)):

X\_test.append(inputs\_scaled[i-time\_step:i, 0])

X\_test = np.array(X\_test)

X\_test = X\_test.reshape((X\_test.shape[0], X\_test.shape[1], 1))

**# Step 7: Predict**

predicted\_stock\_price = model.predict(X\_test)

predicted\_stock\_price = scaler.inverse\_transform(predicted\_stock\_price)

**# Step 8: Plot the results**

plt.figure(figsize=(12, 6))

plt.plot(real\_stock\_price, color='blue', label='Real Google Stock Price')

plt.plot(predicted\_stock\_price, color='red', label='Predicted Google Stock Price')

plt.title('Google Stock Price Prediction using RNN')

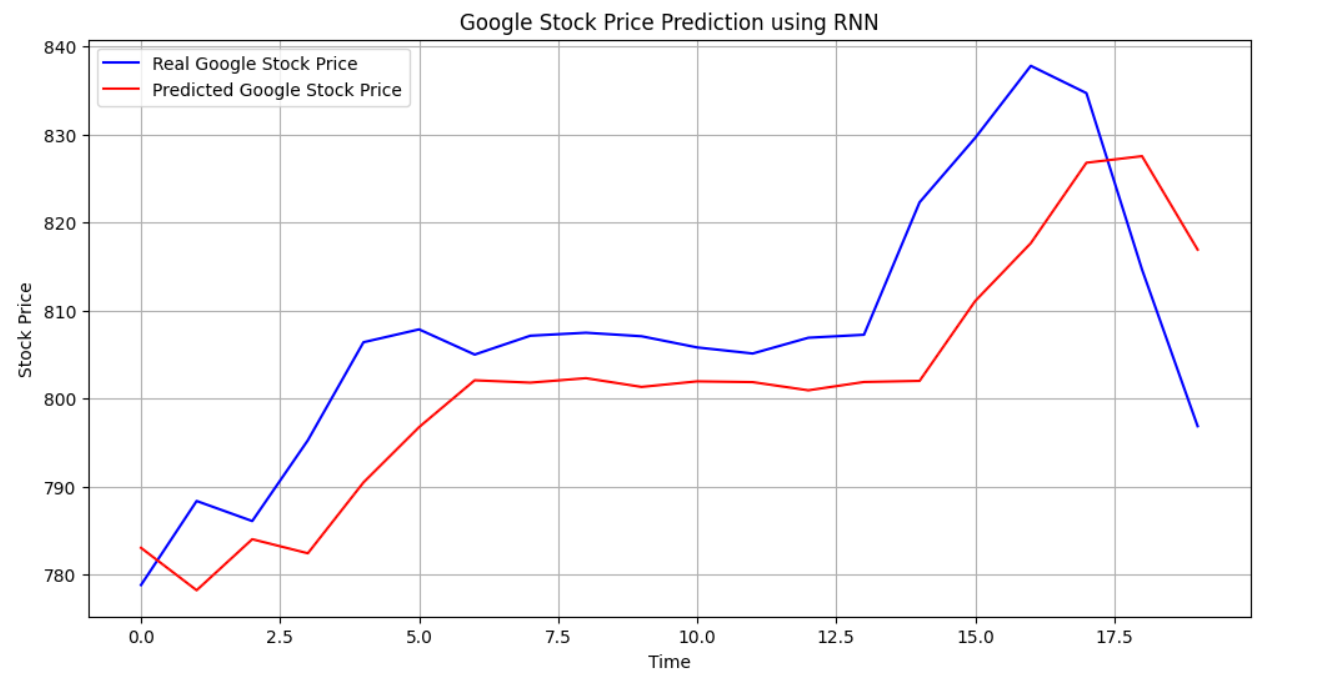
plt.xlabel('Time')

plt.ylabel('Stock Price')

plt.legend()

plt.grid(True)

plt.show()

**Output -**