Google stock price prediction using RNN

```
# import the library
import numpy as np
import pandas as pd
```

```
# load the train data
df = pd.read_csv('Google_Stock_Price_Train.csv')
train_data = df.iloc[:,1:2].values # to get open column in array form
train_data[:5]
```

```
array([[325.25],
       [331.27],
       [329.83],
       [328.34],
       [322.04]])
```

```
# Feature scaling
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range=(0,1))
train_data_scaled = sc.fit_transform(train_data)
train_data_scaled[:5]
```

```
array([[0.08581368],
       [0.09701243],
       [0.09433366],
       [0.09156187],
       [0.07984225]])
```

```
#shape of dataset
df.shape
```

```
(1258, 6)
```

```
# X(Feature) and y(label)
X_train = [] # create an empty list
y_train = [] # create an empty list
for i in range(60,1258): # 1258 is the number of rows
  X_train.append(train_data_scaled[i-60:i,0]) # first 60 timesteps as feature
  y_train.append(train_data_scaled[i,0]) # 61st timestep as target
X_train,y_train = np.array(X_train), np.array(X_train) # to convert to array for feeding in our LSTM model
```

```
# reshape the data
X_train = np.reshape(X_train,(X_train.shape[0],X_train.shape[1],1)) # for feeding into LSTm model
X_train.shape
```

```
(1198, 60, 1)
```

```
# Build the model
# Import the libraries
from keras.models import Sequential
from keras.layers import Dense , LSTM , Dropout
```

```
# intialise model
model = Sequential()
# first layer
model.add(LSTM(units=50,return_sequences=True , input_shape=(X_train.shape[1],1)))
model.add(Dropout(0.2))
# second layer
model.add(LSTM(units=50,return_sequences=True))
model.add(Dropout(0.2))
# third layer
model.add(LSTM(units=50,return_sequences=True))
model.add(Dropout(0.2))
# fourth layer
model.add(LSTM(units=50))
model.add(Dropout(0.2))
# output layer
model.add(Dense(units=1))
```

```
# Compile the model
model.compile(optimizer='adam',loss='mean_squared_error')
```

```
# fit or train the model
model.fit(X_train,y_train,epochs=100,batch_size=32)
```

```
38/38 [==============================] - 5s 133ms/step - loss: 0.0028
Epoch 73/100
38/38 [==============================] - 4s 115ms/step - loss: 0.0031
Epoch 74/100
38/38 [==============================] - 6s 149ms/step - loss: 0.0029
Epoch 75/100
38/38 [==============================] - 5s 124ms/step - loss: 0.0034
Epoch 76/100
38/38 [==============================] - 4s 113ms/step - loss: 0.0029
Epoch 77/100
38/38 [==============================] - 6s 152ms/step - loss: 0.0028
Epoch 78/100
38/38 [==============================] - 4s 114ms/step - loss: 0.0030
Epoch 79/100
38/38 [==============================] - 4s 115ms/step - loss: 0.0030
Epoch 80/100
38/38 [==============================] - 6s 159ms/step - loss: 0.0028
Epoch 81/100
38/38 [==============================] - 4s 111ms/step - loss: 0.0029
Epoch 82/100
38/38 [==============================] - 4s 111ms/step - loss: 0.0029
Epoch 83/100
38/38 [==============================] - 6s 154ms/step - loss: 0.0028
Epoch 84/100
38/38 [==============================] - 4s 114ms/step - loss: 0.0027
Epoch 85/100
38/38 [==============================] - 4s 113ms/step - loss: 0.0029
Epoch 86/100
38/38 [==============================] - 6s 161ms/step - loss: 0.0029
Epoch 87/100
38/38 [==============================] - 4s 112ms/step - loss: 0.0029
Epoch 88/100
38/38 [==============================] - 4s 112ms/step - loss: 0.0028
Epoch 89/100
38/38 [==============================] - 6s 155ms/step - loss: 0.0030
Epoch 90/100
38/38 [==============================] - 4s 114ms/step - loss: 0.0029
Epoch 91/100
38/38 [==============================] - 4s 111ms/step - loss: 0.0027
Epoch 92/100
38/38 [==============================] - 6s 157ms/step - loss: 0.0031
Epoch 93/100
38/38 [==============================] - 4s 111ms/step - loss: 0.0027
Epoch 94/100
38/38 [==============================] - 4s 113ms/step - loss: 0.0026
Epoch 95/100
38/38 [==============================] - 6s 157ms/step - loss: 0.0027
Epoch 96/100
38/38 [==============================] - 4s 109ms/step - loss: 0.0027
Epoch 97/100
38/38 [==============================] - 4s 114ms/step - loss: 0.0027
Epoch 98/100
38/38 [==============================] - 6s 159ms/step - loss: 0.0027
Epoch 99/100
38/38 [==============================] - 4s 108ms/step - loss: 0.0026
Epoch 100/100
38/38 [==============================] - 5s 119ms/step - loss: 0.0026
<keras.src.callbacks.History at 0x79f25ed47d00>
```

```
# Predict the values
# Load the test data
df_test= pd.read_csv('Google_Stock_Price_Test.csv')
actual_values = df_test.iloc[:,1:2].values # to get open column in array form


# Create data for January prediction
df_total = pd.concat((df['Open'],df_test['Open']),axis=0)
test_input = df_total[len(df_total)-len(df_test)-60:].values
test_input = test_input.reshape(-1,1)
test_input_scaled = sc.transform(test_input)
X_test= []
for i in range(60,80):
  X_test.append(test_input_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))



pred = model.predict(X_test)
```

```
1/1 [==============================] - 2s 2s/step
```

```
pred
```

```
array([[0.8992993 ],
       [0.89937186],
       [0.89941514],
       [0.89947283],
       [0.89939964],
       [0.8993942 ],
       [0.8994673 ],
       [0.8995819 ],
       [0.89977187],
       [0.89989257],
       [0.8999163 ],
       [0.8999249 ],
       [0.9001052 ],
       [0.9001887 ],
       [0.90016216],
       [0.90035486],
       [0.90070283],
       [0.9010489 ],
       [0.9016806 ],
       [0.90255934]], dtype=float32)
```

```
pred_test = sc.inverse_transform(pred)
pred_test
```

```
array([[762.54736],
       [762.5863 ],
       [762.60956],
       [762.64056],
       [762.60126],
       [762.5984 ],
       [762.63763],
       [762.6992 ],
       [762.8014 ],
       [762.8662 ],
       [762.87897],
       [762.8836 ],
       [762.9805 ],
       [763.0254 ],
       [763.01117],
       [763.11475],
       [763.3018 ],
       [763.4878 ],
       [763.8274 ],
       [764.2998 ]], dtype=float32)
```

```
# Visualising the result
import matplotlib.pyplot as plt
plt.plot(actual_values , color = 'red' , label = 'Actual Value')
plt.plot(pred_test, color='blue' , label = 'Predicted Values')
plt.title (' Actual Vs Predcited Stock Price')
plt.xlabel('Time')
plt.ylabel('Google Stock Price')
plt.legend(loc='best')
plt.show()
```