DL_A4

March 5, 2024

Name -Ashish Ramesh Walke || RollNo.- 4272 || Batch-B8

Assignment No 4: : Recurrent neural network (RNN) - Use the Google stock prices dataset and design a time series analysis and prediction system using RNN.

[1]:
```
import pandas as pd
import numpy as np
```

[2]:
```
train_df = pd.read_csv(r'Google_Stock_Price_Train.csv') #Path where the CSV␣
↪file is stored.
```

[3]:
```
train_df
```

[3]:

	Date	Open	High	Low	Close	Volume
0	1/3/2012	325.25	332.83	324.97	663.59	7,380,500
1	1/4/2012	331.27	333.87	329.08	666.45	5,749,400
2	1/5/2012	329.83	330.75	326.89	657.21	6,590,300
3	1/6/2012	328.34	328.77	323.68	648.24	5,405,900
4	1/9/2012	322.04	322.29	309.46	620.76	11,688,800
…	…	…	…	…	…	…
1253	12/23/2016	790.90	792.74	787.28	789.91	623,400
1254	12/27/2016	790.68	797.86	787.66	791.55	789,100
1255	12/28/2016	793.70	794.23	783.20	785.05	1,153,800
1256	12/29/2016	783.33	785.93	778.92	782.79	744,300
1257	12/30/2016	782.75	782.78	770.41	771.82	1,770,000

[1258 rows x 6 columns]

[4]:
```
test_df = pd.read_csv(r'Google_Stock_Price_Test.csv') #Path where the CSV file␣
↪is stored.
```

[5]:
```
 test_df
```

[5]:

	Date	Open	High	Low	Close	Volume
0	1/3/2017	778.81	789.63	775.80	786.14	1,657,300
1	1/4/2017	788.36	791.34	783.16	786.90	1,073,000
2	1/5/2017	786.08	794.48	785.02	794.02	1,335,200
3	1/6/2017	795.26	807.90	792.20	806.15	1,640,200
4	1/9/2017	806.40	809.97	802.83	806.65	1,272,400

```
5   1/10/2017  807.86  809.13  803.51  804.79  1,176,800
6   1/11/2017  805.00  808.15  801.37  807.91  1,065,900
7   1/12/2017  807.14  807.39  799.17  806.36  1,353,100
8   1/13/2017  807.48  811.22  806.69  807.88  1,099,200
9   1/17/2017  807.08  807.14  800.37  804.61  1,362,100
10  1/18/2017  805.81  806.21  800.99  806.07  1,294,400
11  1/19/2017  805.12  809.48  801.80  802.17    919,300
12  1/20/2017  806.91  806.91  801.69  805.02  1,670,000
13  1/23/2017  807.25  820.87  803.74  819.31  1,963,600
14  1/24/2017  822.30  825.90  817.82  823.87  1,474,000
15  1/25/2017  829.62  835.77  825.06  835.67  1,494,500
16  1/26/2017  837.81  838.00  827.01  832.15  2,973,900
17  1/27/2017  834.71  841.95  820.44  823.31  2,965,800
18  1/30/2017  814.66  815.84  799.80  802.32  3,246,600
19  1/31/2017  796.86  801.25  790.52  796.79  2,160,600
```

[6]:
```
test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    20 non-null     object
 1   Open    20 non-null     float64
 2   High    20 non-null     float64
 3   Low     20 non-null     float64
 4   Close   20 non-null     float64
 5   Volume  20 non-null     object
dtypes: float64(4), object(2)
memory usage: 1.1+ KB
```

Data Preprocessing

[7]:
```
from sklearn.preprocessing import MinMaxScaler
```

[8]:
```
# Convert 'Close' column to string type and remove commas
train_df['Close'] = train_df['Close'].astype(str).str.replace(',', '').
 ↪astype(float)
test_df['Close'] = test_df['Close'].astype(str).str.replace(',', '').
 ↪astype(float)
```

[9]:
```
# Normalize the training and testing data separately
train_scaler = MinMaxScaler()
train_df['Normalized Close'] = train_scaler.fit_transform(train_df['Close'].
 ↪values.reshape(-1, 1))
test_scaler = MinMaxScaler()
```

```
test_df['Normalized Close'] = test_scaler.fit_transform(test_df['Close'].values.
↪reshape(-1, 1))
```

[10]:
```
# Convert the data to the appropriate format for RNN
x_train = train_df['Normalized Close'].values[:-1].reshape(-1, 1, 1)
y_train = train_df['Normalized Close'].values[1:].reshape(-1, 1, 1)
x_test = test_df['Normalized Close'].values[:-1].reshape(-1, 1, 1)
y_test = test_df['Normalized Close'].values[1:].reshape(-1, 1, 1)
```

[11]:
```
print("x_train shape: ",x_train.shape)
print("y_train shape: ",y_train.shape)
print("x_test shape: ",x_test.shape)
print("y_test shape: ",y_test.shape)
```

```
x_train shape:  (1257, 1, 1)
y_train shape:  (1257, 1, 1)
x_test shape:  (19, 1, 1)
y_test shape:  (19, 1, 1)
```

[12]:
```
test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 7 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Date              20 non-null     object
 1   Open              20 non-null     float64
 2   High              20 non-null     float64
 3   Low               20 non-null     float64
 4   Close             20 non-null     float64
 5   Volume            20 non-null     object
 6   Normalized Close  20 non-null     float64
dtypes: float64(5), object(2)
memory usage: 1.2+ KB
```

Building our Model

[13]:
```
 from keras.models import Sequential
from keras.layers import LSTM, Dense
```

```
WARNING:tensorflow:From C:\Users\Ashish\anaconda3\Lib\site-
packages\keras\src\losses.py:2976: The name
tf.losses.sparse_softmax_cross_entropy is deprecated. Please use
tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

[14]:
```
model = Sequential()
model.add(LSTM(4, input_shape=(1, 1)))
```

```
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.summary()
```

```
WARNING:tensorflow:From C:\Users\Ashish\anaconda3\Lib\site-
packages\keras\src\backend.py:873: The name tf.get_default_graph is deprecated.
Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From C:\Users\Ashish\anaconda3\Lib\site-
packages\keras\src\optimizers\__init__.py:309: The name tf.train.Optimizer is
deprecated. Please use tf.compat.v1.train.Optimizer instead.

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 4)                 96

 dense (Dense)               (None, 1)                 5

=================================================================
Total params: 101 (404.00 Byte)
Trainable params: 101 (404.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Building our Modely()

[15]:
```
model.fit(x_train, y_train, epochs=50, batch_size=1, verbose=1)
```

```
Epoch 1/50
WARNING:tensorflow:From C:\Users\Ashish\anaconda3\Lib\site-
packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTensorValue
is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

1257/1257 [==============================] - 8s 3ms/step - loss: 0.0492
Epoch 2/50
1257/1257 [==============================] - 4s 3ms/step - loss: 0.0061
Epoch 3/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.9633e-04
Epoch 4/50
1257/1257 [==============================] - 3s 3ms/step - loss: 7.6611e-04
Epoch 5/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.6682e-04
Epoch 6/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.6955e-04
Epoch 7/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.6812e-04
Epoch 8/50
```

```
1257/1257 [==============================] - 4s 3ms/step - loss: 7.5853e-04
Epoch 9/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.6718e-04
Epoch 10/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.6423e-04
Epoch 11/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.4985e-04
Epoch 12/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.5008e-04
Epoch 13/50
1257/1257 [==============================] - 3s 3ms/step - loss: 7.6696e-04
Epoch 14/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.6529e-04
Epoch 15/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.4453e-04
Epoch 16/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.5847e-04
Epoch 17/50
1257/1257 [==============================] - 3s 3ms/step - loss: 7.6124e-04
Epoch 18/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.6739e-04
Epoch 19/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.6197e-04
Epoch 20/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.5157e-04
Epoch 21/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.6060e-04
Epoch 22/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.5561e-04
Epoch 23/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.5242e-04
Epoch 24/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.6012e-04
Epoch 25/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.5640e-04
Epoch 26/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.5184e-04
Epoch 27/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.5468e-04
Epoch 28/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.5712e-04
Epoch 29/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.5842e-04
Epoch 30/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.5292e-04
Epoch 31/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.6508e-04
Epoch 32/50
```

```
1257/1257 [==============================] - 4s 3ms/step - loss: 7.5352e-04
Epoch 33/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.5178e-04
Epoch 34/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.6670e-04
Epoch 35/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.4873e-04
Epoch 36/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.7639e-04
Epoch 37/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.4424e-04
Epoch 38/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.4913e-04
Epoch 39/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.5499e-04
Epoch 40/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.5923e-04
Epoch 41/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.5799e-04
Epoch 42/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.5670e-04
Epoch 43/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.5790e-04
Epoch 44/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.6846e-04
Epoch 45/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.4961e-04
Epoch 46/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.6863e-04
Epoch 47/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.4557e-04
Epoch 48/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.5276e-04
Epoch 49/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.4168e-04
Epoch 50/50
1257/1257 [==============================] - 4s 3ms/step - loss: 7.4874e-04
```

```
[15]: <keras.src.callbacks.History at 0x183a2c4de10>
```

Evaluating our Model

```
[16]: test_loss = model.evaluate(x_test, y_test)
      print('Testing loss: ', test_loss)
```

```
1/1 [==============================] - 1s 1s/step - loss: 0.0250
Testing loss:  0.025006726384162903
```

Testing our Model

[17]:

```
y_pred = model.predict(x_test)
```

```
1/1 [==============================] - 1s 850ms/step
```

[18]:

```
# Inverse transform the normalized values to get the actual values
y_test_actual = test_scaler.inverse_transform(y_test.reshape(-1, 1))
y_pred_actual = test_scaler.inverse_transform(y_pred.reshape(-1, 1))
```

[19]:

```
i=1
```

[20]:

```
print("Actual value: {:.2f}".format(y_test_actual[i][0]))
print("Predicted value: {:.2f}".format(y_pred_actual[i][0]))
```

```
Actual value: 794.02
Predicted value: 787.29
```