

Assignment-4

| | |
|---------------------|-----------------|
| Assignment Date | 17 October 2022 |
| Student Name | T.Sivamartin |
| Student Roll Number | 962719106033 |
| Maximum Marks | 2Marks |

Question 1:

Download the dataset

Link:

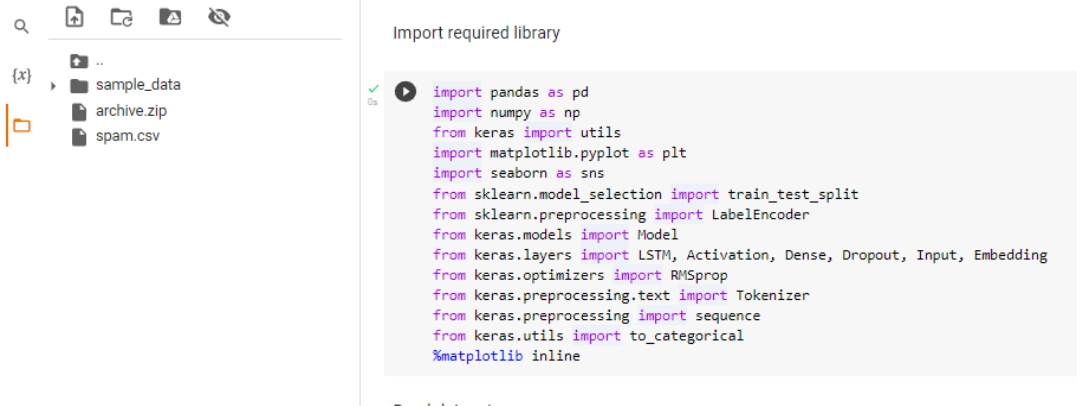
https://drive.google.com/file/d/1Sjqx5H5R86tRp2YZKzzd4_iEfjChZ3ob/view?usp=sharing

Question 2:

Import required library

Solution:

```
import pandas as pd
import numpy as np
from keras import utils
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from keras.models import Model
from keras.layers import LSTM, Activation, Dense, Dropout, Input, Embedding
from keras.optimizers import RMSprop
from keras.preprocessing.text import Tokenizer
from keras.preprocessing import sequence
from keras.utils import to_categorical
```



Question 3:

Read dataset and do pre-processing

Solution:

Read dataset

```
!unzip "/content/archive.zip"
```

```
df = pd.read_csv('spam.csv', delimiter=',', encoding='latin-1')
df
```

Pre processing

```
df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], axis=1, inplace=True)
df
```

```
sns.countplot(df.v1, palette='Set3')
plt.xlabel('Label')
plt.title('Number of ham and spam messages')
```

```
X = df.v2
Y = df.v1
le = LabelEncoder()
Y = le.fit_transform(Y)
Y = Y.reshape(-1,1)
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.15)
```

```
max_words = 1000
max_len = 150
tok = Tokenizer(num_words=max_words)
tok.fit_on_texts(X_train)
sequences = tok.texts_to_sequences(X_train)
sequences_matrix = utils.pad_sequences(sequences, maxlen=max_len)
```

sequences_matrix.shape

sequences_matrix.ndim

sequences_matrix = np.reshape(sequences_matrix,(4736,150,1))

sequences_matrix.ndim

🔍

📁 📄 📄 📄 📄

{x}

sample_data

archive.zip

spam.csv

<>

☰

Read dataset

✓ 2s

[2] !unzip "/content/archive.zip"

Archive: /content/archive.zip
inflating: spam.csv

✓ 0s

[3] df = pd.read_csv('spam.csv',delimiter=',',encoding='latin-1')

df

| | v1 | v2 | Unnamed: 2 | Unnamed: 3 | Unnamed: 4 |
|------|------|---|------------|------------|------------|
| 0 | ham | Go until jurong point, crazy.. Available only ... | NaN | NaN | NaN |
| 1 | ham | Ok lar... Joking wif u oni... | NaN | NaN | NaN |
| 2 | spam | Free entry in 2 a wkly comp to win FA Cup fina... | NaN | NaN | NaN |
| 3 | ham | U dun say so early hor... U c already then say... | NaN | NaN | NaN |
| 4 | ham | Nah I don't think he goes to usf, he lives aro... | NaN | NaN | NaN |
| ... | ... | ... | ... | ... | ... |
| 5567 | spam | This is the 2nd time we have tried 2 contact u... | NaN | NaN | NaN |
| 5568 | ham | Will i_b going to esplanade fr home? | NaN | NaN | NaN |
| 5569 | ham | Pity, * was in mood for that. So...any other s... | NaN | NaN | NaN |
| 5570 | ham | The guy did some bitching but I acted like i'd... | NaN | NaN | NaN |

🔍

📁 📄 📄 📄 📄

{x}

sample_data

archive.zip

spam.csv

<>

☰

Pre processing

✓ 2s

[4] df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],axis=1,inplace=True)

df

| | v1 | v2 |
|------|------|---|
| 0 | ham | Go until jurong point, crazy.. Available only ... |
| 1 | ham | Ok lar... Joking wif u oni... |
| 2 | spam | Free entry in 2 a wkly comp to win FA Cup fina... |
| 3 | ham | U dun say so early hor... U c already then say... |
| 4 | ham | Nah I don't think he goes to usf, he lives aro... |
| ... | ... | ... |
| 5567 | spam | This is the 2nd time we have tried 2 contact u... |
| 5568 | ham | Will i_b going to esplanade fr home? |
| 5569 | ham | Pity, * was in mood for that. So...any other s... |
| 5570 | ham | The guy did some bitching but I acted like i'd... |
| 5571 | ham | Rofl. Its true to its name |

5572 rows × 2 columns

{X}
sample_data
archive.zip
spam.csv

```
sns.countplot(df.v1,palette='Set3')
plt.xlabel('Label')
plt.title('Number of ham and spam messages')
```

FutureWarning: Pass the following variable as a keyword arg: x. From version

Text(0.5, 1.0, 'Number of ham and spam messages')

| Label | count |
|-------|-------|
| ham | ~4800 |
| spam | ~800 |

{X}
sample_data
archive.zip
spam.csv

```
[7] X = df.v2
Y = df.v1
le = LabelEncoder()
Y = le.fit_transform(Y)
Y = Y.reshape(-1,1)

[8] X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.15)

[9] max_words = 10000
max_len = 150
tok = Tokenizer(num_words=max_words)
tok.fit_on_texts(X_train)
sequences = tok.texts_to_sequences(X_train)
sequences_matrix = utils.pad_sequences(sequences,maxlen=max_len)

[10] sequences_matrix.shape
(4736, 150)

[11] sequences_matrix.ndim
2

[12] sequences_matrix = np.reshape(sequences_matrix,(4736,150,1))
sequences_matrix.ndim
3
```

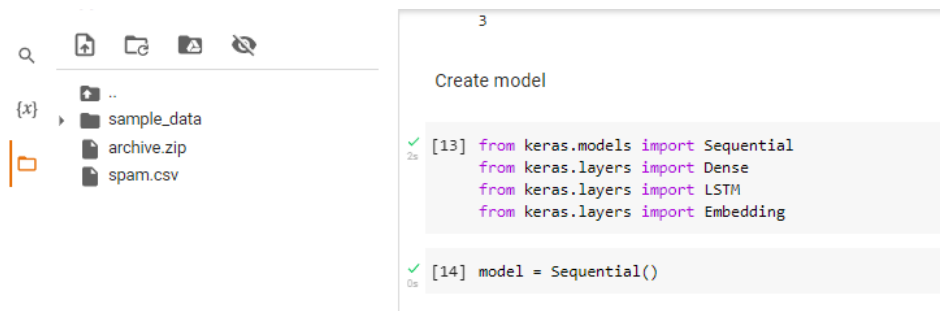
Question 4:

Create model

Solution:

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Embedding
```

```
model = Sequential()
```

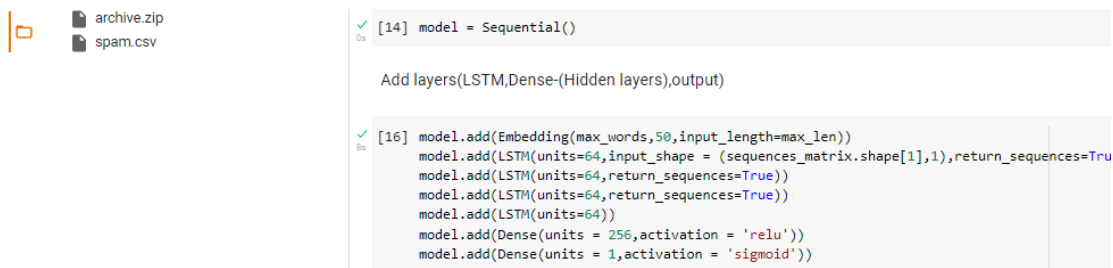


Question 5:

Add layers(LSTM,Dense-(Hidden layers),output)

Solution:

```
model.add(Embedding(max_words,50,input_length=max_len))
model.add(LSTM(units=64,input_shape = (sequences_matrix.shape[1],1),return_sequences=True))
model.add(LSTM(units=64,return_sequences=True))
model.add(LSTM(units=64,return_sequences=True))
model.add(LSTM(units=64))
model.add(Dense(units = 256,activation = 'relu'))
model.add(Dense(units = 1,activation = 'sigmoid'))
```

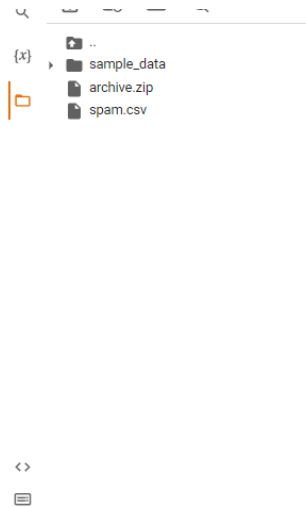


Question 6:

Compile the model

Solution:

```
model.summary()
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
```



```
Compile the model

[17] model.summary()
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])

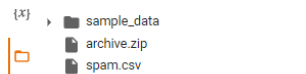
Model: "sequential"
-----
Layer (type)                Output Shape              Param #
-----
embedding (Embedding)       (None, 150, 50)           50000
lstm (LSTM)                  (None, 150, 64)           29440
lstm_1 (LSTM)                (None, 150, 64)           33024
lstm_2 (LSTM)                (None, 150, 64)           33024
lstm_3 (LSTM)                (None, 64)                33024
dense (Dense)                (None, 256)               16640
dense_1 (Dense)              (None, 1)                 257
-----
Total params: 195,409
Trainable params: 195,409
Non-trainable params: 0
```

Question 7:

Fit the model

Solution:

```
X = model.fit(sequences_matrix,Y_train,batch_size=128,epochs=5,validation_split=0.2)
X
```



```
Fit the model

[18] X = model.fit(sequences_matrix,Y_train,batch_size=128,epochs=5,validation_split=0.2)
X

Epoch 1/5
30/30 [=====] - 43s 1s/step - loss: 0.4490 - accuracy: 0.8688 - val_loss: 0.4257 - val_accuracy: 0.8513
Epoch 2/5
30/30 [=====] - 33s 1s/step - loss: 0.2615 - accuracy: 0.9092 - val_loss: 0.1283 - val_accuracy: 0.9610
Epoch 3/5
30/30 [=====] - 35s 1s/step - loss: 0.0724 - accuracy: 0.9794 - val_loss: 0.0896 - val_accuracy: 0.9705
Epoch 4/5
30/30 [=====] - 33s 1s/step - loss: 0.0529 - accuracy: 0.9857 - val_loss: 0.0853 - val_accuracy: 0.9778
Epoch 5/5
30/30 [=====] - 33s 1s/step - loss: 0.0392 - accuracy: 0.9900 - val_loss: 0.0836 - val_accuracy: 0.9768
<keras.callbacks.History at 0x7f263f739bd0>

Save the model
```

Question 8:

Save the model

Solution:

```
model.save
```

{x} ▶ sample_data
archive.zip
spam.csv

Save the model

✓ [19] model.save

<bound method Model.save of <keras.engine.sequential.Sequential object at 0x7f2643a9c750>>

Question 9:

Test the model

Solution:

```
test_sequences = tok.texts_to_sequences(X_test)
test_sequences_matrix = utils.pad_sequences(test_sequences,maxlen=max_len)
```

```
accr = model.evaluate(test_sequences_matrix,Y_test)
```

```
l = accr[0]
a =accr[1]
print('Test set\n Loss: {:.3f}\n Accuracy: {:.3f}'.format(l,a))
```

Files

🔍 📁 📄 📁 📄

{x} ▶ ..
sample_data
archive.zip
spam.csv

+ Code + Text

✓ [19] model.save

<bound method Model.save of <keras.engine.sequential.Sequential object at 0x7f2643a9c750>>

Test the model

✓ [20] test_sequences = tok.texts_to_sequences(X_test)

test_sequences_matrix = utils.pad_sequences(test_sequences,maxlen=max_len)

✓ [21] accr = model.evaluate(test_sequences_matrix,Y_test)

27/27 [=====] - 4s 88ms/step - loss: 0.0814 - accuracy: 0.9797

✓ [22] l = accr[0]

a =accr[1]

print('Test set\n Loss: {:.3f}\n Accuracy: {:.3f}'.format(l,a))

Test set

Loss: 0.081

Accuracy: 0.980

<>