Assignment-4

SMS SPAM Classification

Assignment Date	:	26 October 2022
Student Name	:	P.DHINESHKUMAR
Student Roll Number	:	912419104007
Maximum Marks	:	2 Marks

Task 1:

Download the dataset

Download Dataset

Task 2:

Question- 1:

Import the necessary libraries

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
%matplotlib inline
```

Output:

1. Import the necessary libraries

In [1]:

```
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
%matplotlib inline
```

Solution:

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

Output:

```
In [2]: dataframe = pd.read_csv('spam.csv',delimiter=',',encoding='latin-1')
        dataframe
```

Out[2]:

	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 Ì_ 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
5571	ham	Rofl. Its true to its name	NaN	NaN	NaN

5572 rows × 5 columns

Task 3:

Question- 2:

Preprocessing

Solution:

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

Output:

2)Preprocessing

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

Out[3]:

	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 Ì_ 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

Solution:

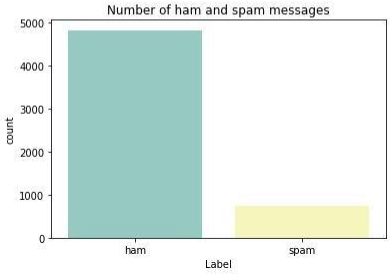
```
sns.countplot(dataframe.v1,palette='Set3')
plt.xlabel('Label')
plt.title('Number of ham and spam messages')
```

Output:

In [4]:
```
sns.countplot(dataframe.v1,palette='Set3')
plt.xlabel('Label')
plt.title('Number of ham and spam messages')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretat
ion.
  FutureWarning
```

Out[4]: Text(0.5, 1.0, 'Number of ham and spam messages')



Solution:

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

Output:

In [5]:

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

Task 4:

Question- 3:

Split into training and test data

Solution:

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

Output:

3)Split into training and test data.

In [6]:

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

Solution:

```
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)
```

Output:

In [7]:
```
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)
```

Solution:

```
sequences_matrix.shape
```

Output:

In [8]:
```
sequences_matrix.shape
```

Out[8]:
```
(4736, 150)
```

Solution:

```
sequences_matrix.ndim
```

Output:

```
In [9]: sequences_matrix.ndim

Out[9]: 2
```

Solution:

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

Output:

```
In [10]: sequences_matrix = np.reshape(sequences_matrix,(4736,150,1))
```

Solution:

```
sequences_matrix.ndim #3d shape verification to
    proceed to RNN LSTM
```

Output:

```
In [11]: sequences_matrix.ndim #3d shape verification to proceed to RNN LSTM

Out[11]: 3
```

Task 5:

Question- 4:

Create model for RNN

Solution:

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

Output:

4)Create model for RNN

In [12]:

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

Solution:

```
model = Sequential()
```

Output:

```
In [13]: model = Sequential()
```

Task 6:

Question- 5:

Add Layers

Solution:

```
model.add(Embedding(max_words,50,input_length=max_len))
```

Output:

5)Add Layers

```
In [14]: model.add(Embedding(max_words,50,input_length=max_len))
```

Solution:

```
model.add(LSTM(units=64,input_shape =
  (sequences_matrix.shape[1],1),return_sequences=True))
```

Output:

```
In [15]: model.add(LSTM(units=64,input_shape = (sequences_matrix.shape[1],1),return_sequences=True))
```

Solution:

model.add(LSTM(units=64,return_sequences=True))

Output:

```
In [16]: model.add(LSTM(units=64,return_sequences=True))
```

Solution:

model.add(LSTM(units=64,return_sequences=True))

Output:

```
In [17]: model.add(LSTM(units=64,return_sequences=True))
```

Solution:

model.add(LSTM(units=64))

Output:

```
In [18]: model.add(LSTM(units=64))
```

Solution:

model.add(Dense(units = 256,activation = 'relu'))

Output:

```
In [19]: model.add(Dense(units = 256,activation = 'relu'))
```

Solution:

```
model.add(Dense(units = 1,activation = 'sigmoid'))
```

Output:

```
In [20]: model.add(Dense(units = 1,activation = 'sigmoid'))
```

Task 7:

Question- 6:

Compile the model

Solution:

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

Output:

6)Compile the model

In [21]:
```
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
_________________________________________________________________
```

Fit the model

Solution:

```
modelf =
    model.fit(sequences_matrix,Y_train,batch_size=128,e
    pochs        =10,validation_split=0.2)
modelf
```

Output:

```
In [22]: modelf = model.fit(sequences_matrix,Y_train,batch_size=128,epochs=10,validation_split=0.2)
         modelf

Epoch 1/10
30/30 [==============================] - 45s 1s/step - loss: 0.4619 - accuracy: 0.8456 - val_loss: 0.4486 - val_accuracy: 0.8460
Epoch 2/10
30/30 [==============================] - 35s 1s/step - loss: 0.3779 - accuracy: 0.8728 - val_loss: 0.4079 - val_accuracy: 0.8460
Epoch 3/10
30/30 [==============================] - 34s 1s/step - loss: 0.2314 - accuracy: 0.9116 - val_loss: 0.1126 - val_accuracy: 0.9662
Epoch 4/10
30/30 [==============================] - 39s 1s/step - loss: 0.0753 - accuracy: 0.9794 - val_loss: 0.0773 - val_accuracy: 0.9778
Epoch 5/10
30/30 [==============================] - 37s 1s/step - loss: 0.0471 - accuracy: 0.9879 - val_loss: 0.0545 - val_accuracy: 0.9842
Epoch 6/10
30/30 [==============================] - 34s 1s/step - loss: 0.0331 - accuracy: 0.9913 - val_loss: 0.0506 - val_accuracy: 0.9863
Epoch 7/10
30/30 [==============================] - 34s 1s/step - loss: 0.0253 - accuracy: 0.9939 - val_loss: 0.0446 - val_accuracy: 0.9916
Epoch 8/10
30/30 [==============================] - 34s 1s/step - loss: 0.0210 - accuracy: 0.9950 - val_loss: 0.0572 - val_accuracy: 0.9852
Epoch 9/10
30/30 [==============================] - 34s 1s/step - loss: 0.0171 - accuracy: 0.9968 - val_loss: 0.0495 - val_accuracy: 0.9895
Epoch 10/10
30/30 [==============================] - 34s 1s/step - loss: 0.0124 - accuracy: 0.9976 - val_loss: 0.0536 - val_accuracy: 0.9916
Out[22]:
```

Task 8:

Question- 7:

Save the model

Solution:

```
model.save
```

Output:

Task 9:

Question- 8:

Testing the model

Solution:

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

Output:

8. Testing the model

```
In [24]: test_sequences = tok.texts_to_sequences(X_test)
         test_sequences_matrix = utils.pad_sequences(test_sequences,maxlen=max_len)
```

Solution:

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

Output:

```
In [25]: accr = model.evaluate(test_sequences_matrix,Y_test)
27/27 [==============================] - 6s 176ms/step - loss: 0.1019 - accuracy: 0.9809
```

Solution:

```
l = accr[0]
a =accr[1]
print('Test set\n  Loss: {:0.3f}\n  Accuracy:
    {:0.3f}'.format(l,a))
```

Output:

```
In [26]: l = accr[0]
         a =accr[1]
         print('Test set\n  Loss: {:0.3f}\n  Accuracy: {:0.3f}'.format(l,a))

         Test set
           Loss: 0.102
           Accuracy: 0.981
```

Accuracy and Loss Graph

Solution:

```
Results = pd.DataFrame({"Train Loss":
        modelf.history['loss'], "Validation Loss":
        modelf.history['val_loss'],
        "Train Accuracy":
           modelf.history['accuracy'], "Validation
         Accuracy": modelf.history['val_accuracy']
           })
fig, ax = plt.subplots(nrows=2, figsize=(16, 9))
results[["Train Loss", "Validation Loss"]].plot(ax=ax[0])
results[["Train Accuracy", "Validation
          Accuracy"]].plot(ax=ax[1])
```

```
ax[0].set_xlabel("Epoch")
ax[1].set_xlabel("Epoch")
plt.show()
```

Output:

In [27]:

```
results = pd.DataFrame({"Train Loss": modelf.history['loss'], "Validation Loss": modelf.history['val_loss'],
            "Train Accuracy": modelf.history['accuracy'], "Validation Accuracy": modelf.history['val_accuracy']
           })
fig, ax = plt.subplots(nrows=2, figsize=(16, 9))
results[["Train Loss", "Validation Loss"]].plot(ax=ax[0])
results[["Train Accuracy", "Validation Accuracy"]].plot(ax=ax[1])
ax[0].set_xlabel("Epoch")
ax[1].set_xlabel("Epoch")
plt.show()
```