Assignment– 4

SMSSPAM Classification

AssignmentDate	15 November2022
Team id	PNT2022TMID30815
Project name	Real time communication system powered by AI for specially abled.
MaximumMarks	2 Marks

TASKS:

1. Downloadthedataset

2. Importrequired library

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow

import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer

import string

from tensorflow.keras.preprocessing import sequence

from keras.models import Model, Sequential
from keras.preprocessing.text import Tokenizer
from keras.optimizers import Adam, RMSprop
from keras.layers import Input, Embedding, LSTM, Dense, Flatten, Dropout

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
```

3. ReaddatasetanddoPre-processing

Read Dataset

```
df = pd.read_csv(r"C:\Users\manok\Documents\Sem_7\HX5001-HX6001\Assignment\Assignment_4\spam.csv", encoding='latin-1')
```

```
df.head()
```

	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

```
df.shape
```

```
(5572, 5)
```

Drop Unwanted Column

```
df = df.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1)
df = df.rename(columns={"v2" : "Text", "v1":"Label"})
```

```
df.head()
```

	Label	Text
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...

Remove Duplicate and Null Data

```
df.isnull().sum()
```

```
Label    0
Text     0
dtype: int64
```

```
df.duplicated().sum()
```

```
403
```

```
df = df.drop_duplicates(keep='first')
df.duplicated().sum()
```

```
0
```

```
df.shape
```

```
(5169, 2)
```

Normalizing the case, Removing the unwanted punctuations, Remove Stopwords

```
ps = PorterStemmer()
```

```
def transform_text(text):
    text = text.lower()
    text = nltk.word_tokenize(text)

    y = []
    for i in text:
        if i.isalnum():
            y.append(i)

    text = y[:]
    y.clear()

    for i in text:
        if i not in stopwords.words('english') and i not in string.punctuation:
            y.append(i)

    text = y[:]
    y.clear()

    for i in text:
        y.append(ps.stem(i))


    return " ".join(y)
```

```
df['Transformed_Text'] = df['Text'].apply(transform_text)
```

```
df.head()
```

	Label	Text	Transformed_Text
0	ham	Go until jurong point, crazy.. Available only ...	go jurong point crazi avail bugi n great world...
1	ham	Ok lar... Joking wif u oni...	ok lar joke wif u oni
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	free entri 2 wkli comp win fa cup final tkt 21...
3	ham	U dun say so early hor... U c already then say...	u dun say earli hor u c alreadi say
4	ham	Nah I don't think he goes to usf, he lives aro...	nah think goe usf live around though

Counting Words

```
avg_words_len=round(sum([len(i.split()) for i in df['Text']])/len(df['Text']))
print(avg_words_len)
# avg_words_len=200
```

```
15
```

```
s = set()
for sent in df['Transformed_Text']:
  for word in sent.split():
    s.add(word)
total_words_length=len(s)
print(total_words_length)
# total_words_length=2000
```

```
6736
```

4. Create model

```
x = df.Transformed_Text
y = df.Label
le = LabelEncoder()
y = le.fit_transform(y)
y = y.reshape(-1,1)
```

```
# y = df['Label'].values
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.18, random_state=10)
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

```
((4238,), (4238, 1), (931,), (931, 1))
```

```
model = Sequential()
```

5. Addlayers

```
tokenizer = Tokenizer(num_words = total_words_length, lower = True)
tokenizer.fit_on_texts(x_train)
sequences = tokenizer.texts_to_sequences(x_train)
x_train = sequence.pad_sequences(sequences, maxlen = avg_words_len)
```

Input Layer

```
# model.add(Input(shape=(1), dtype=tf.string))
# model.add(Input(name='inputs',shape=[avg_words_len]))
```

```
model.add(Embedding(total_words_length, 50, input_length = avg_words_len))
```

LSTM Layer

```
model.add(LSTM(64))
```

Hidden Layer

```
model.add(Dense(64, activation = "relu"))
```

```
model.add(Flatten())
```

```
model.add(Dropout(0.2))
```

```
model.add(Dense(32, activation = "relu"))
```

Output Layer

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

Model Summary

```
model.summary()
```

```
Model:"sequential"
_________________________________________________________________
 Layer(type)                 OutputShape              Param#
=================================================================
 embedding(Embedding)        (None, 15, 50)           336800

 lstm(LSTM)                  (None, 64)               29440

 dense(Dense)                (None, 64)               4160

 flatten(Flatten)            (None, 64)               0

 dropout(Dropout)            (None, 64)               0

 dense_1(Dense)              (None, 32)               2080

 dense_2(Dense)              (None, 1)                33

=================================================================
Totalparams:372,513
Trainableparams:372,513
Non-trainableparams:0
_________________________________________________________________
```

6. Compilethemodel

```
adam = Adam(learning_rate = 0.001, beta_1 = 0.85, beta_2 = 0.97, epsilon = 1e-07)
model.compile(loss = "binary_crossentropy", optimizer = adam, metrics = ["accuracy"])
```

7. Fitthemodel

```
epochs=5
history = model.fit(x_train, y_train, epochs = epochs, validation_steps=0.18, batch_size=10)
```

```
Epoch1/5
424/424[==============================]- 16s14ms/step - loss:0.1346 -a
ccuracy:
0.9552Epoch2/5
424/424[==============================]- 6s15ms/step- loss:0.0356 - ac
curacy:
0.9887Epoch3/5
424/424[==============================]- 6s15ms/step- loss:0.0203 - ac
curacy:
0.9941Epoch4/5
424/424[==============================]- 6s14ms/step- loss:0.0096 - ac
curacy:
0.9969Epoch5/5
424/424[==============================]- 6s15ms/step- loss:0.0043 - ac
curacy:0.9988
```

8. Savethemodel

```
model.save("spam_analysis.h5")
```

9. Testthemodel

```
test_sequences = tokenizer.texts_to_sequences(x_test)
x_test = sequence.pad_sequences(test_sequences, maxlen=avg_words_len)
```

```
accuracy = model.evaluate(x_test, y_test)
```

```
30/30 [==============================] - 2s 10ms/step - loss: 0.2072 - accuracy: 0.9731
```

```
def predict(message):
    txt = tokenizer.texts_to_sequences(message)
    txt = sequence.pad_sequences(txt, maxlen=avg_words_len)
    pred = model.predict(txt)
    if pred>0.5:
        print("spam")
    else:
        print("Harm")
```

```
review1 = ["think he goes"]
predict(review1)
```

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

```
review2 = ["Go until jurong point"]
predict(review2)
```

```
1/1 [==============================] - 0s 46ms/step
Harm
```