ASSIGNMENT 4

Problem Statement :- SMS SPAM Classification

Assignment Date	**27 OCTOBER 2022**
Student Name	**MONISHA J**
Student Roll Number	**612419104015**
Maximum Marks	**2 MARKS**

Tasks

Perform the Below Tasks to complete the assignment:-

- Download the Dataset:- Dataset
- Import required library
- Read dataset and do pre-processing
- Create Model
- Add Layers (LSTM, Dense-(Hidden Layers), Output)
- Compile the Model
- Fit the Model
- Save The Model
- Test The Model

1. Download The Dataset :

https://www.kaggle.com/code/kredy10/simple-lstm-for-text-classification/data

2. Import The Required Libraries

```
import os
import re
import
pandas as pd
import numpy
as np import
nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from  wordcloud  import
WordCloud        import
matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout, Embedding
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.preprocessing.text import Tokenizer
import keras
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from google.colab import drive
```

```
import os
import re
import pandas as pd
import numpy as np
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout, Embedding
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.preprocessing.text import Tokenizer
import keras
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from google.colab import drive
```

```
#Mount and access drive
drive.mount('/content/drive',force_remount=True)
os.chdir('/content/drive/My Drive')
print("Change successful.")
```

```
Mounted at /content/drive
Change successful.
```

3.Read The Dataset And Do Pre-Processing

spam_df = pd.read_csv(filepath_or_buffer='Dataset-3_Spam.csv', delimiter=',',encoding='latin’)

spam_df.head()

```
spam_df = pd.read_csv(filepath_or_buffer='Dataset-3_Spam.csv', delimiter=',',encoding='latin-1')
spam_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

```
#List the column names
spam_df.columns
```

```
#List the column names
spam_df.columns
```

```
Index(['v1', 'v2', 'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], dtype='object')
```

```
#Drop the unnamed columns
spam_df.drop(columns=['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], axis=1, inplace=True)
spam_df.columns
```

```
#Drop the unnamed columns
spam_df.drop(columns=['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], axis=1, inplace=True)
spam_df.columns
```

```
Index(['v1', 'v2'], dtype='object')
```

```
#Print the number of rows in the dataset
spam_df.shape
```

```
#Print the number of rows in the dataset
spam_df.shape
```

```
(5572, 2)
```

```
#Get the summary statistics of the dataset
spam_df.describe()
```

```
#Get the summary statistics of the dataset
spam_df.describe()
```

	v1	v2
count	5572	5572
unique	2	5169
top	ham	Sorry, I'll call later
freq	4825	30

```
#Check for null values
spam_df.isna().sum()
```

```
#Check for null values
spam_df.isna().sum()
```

```
v1    0
v2    0
dtype: int64
```

```
nltk.download('stopwords',quiet=True)
nltk.download('all',quiet=True)
```

```
nltk.download('stopwords',quiet=True)
nltk.download('all',quiet=True)
```

```
True
```

```
Ps = PorterStemmer()
input = []
```

```
ps = PorterStemmer()
input = []
```

```
for i in range(0,5572):
  v2 = data['v2'][i]

  #removing punctuation
  v2 = re.sub('[^a-zA-Z]',' ',v2)

  #converting to lower case
  v2 = v2.lower()

  #splitting the sentence
  v2 = v2.split()

  #removing the stopwords and stemming
  v2 = [ps.stem(word) for word in v2 if not word in set(stopwords.words('english'))]

  v2 = ' '.join(v2)

  input.append(v2)
```



```
#creating document term matrix
cv = CountVectorizer(max_features=2000)
x = cv.fit_transform(input).toarray()
x.shape
```

```
#creating document term matrix
cv = CountVectorizer(max_features=2000)
x = cv.fit_transform(input).toarray()
x.shape
```

```
(5572, 2000)
```

```
le = preprocessing.LabelEncoder()

data['v1'] = le.fit_transform(data['v1'])
data['v1'].unique()
```

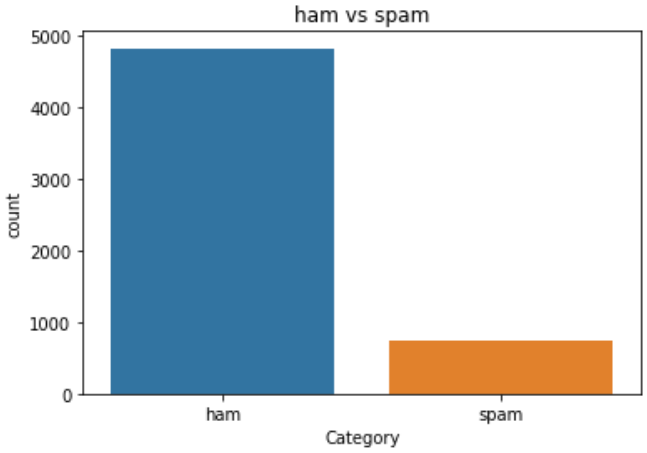
```
le = preprocessing.LabelEncoder()

data['v1'] = le.fit_transform(data['v1'])
data['v1'].unique()
```

```
array([0, 1])
```

```
sns.countplot(data=data,x="Category")
plt.title("ham vs spam")
plt.show()
```


```
sns.countplot(data=data,x="Category")
plt.title("ham vs spam")
plt.show()
```

4. Create The Model

```
#Create a wrapper to add layers to the model
model = Sequential()
```


```
#Create a wrapper to add layers to the model
model = Sequential()
```


5. Add Layers (LSTM, Dense-(Hidden Layers), Output)

```
model.add(Embedding(1000, output_dim=50, input_length=100))
model.add(LSTM(units=64 , return_sequences = True, dropout = 0.2))
model.add(LSTM(units=32 , dropout = 0.1))
model.add(Dense(units = 64 , activation = 'relu'))
model.add(Dense(units = 32 , activation = 'relu'))
model.add(Dense(1, activation='sigmoid'))
model.summary()
```

```
model.add(Embedding(1000, output_dim=50, input_length=100))
model.add(LSTM(units=64 , return_sequences = True, dropout = 0.2))
model.add(LSTM(units=32 , dropout = 0.1))
model.add(Dense(units = 64 , activation = 'relu'))
model.add(Dense(units = 32 , activation = 'relu'))
model.add(Dense(1, activation='sigmoid'))
```

```
model.summary()
```

```
Model: "sequential_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_14 (Embedding)    (None, 100, 50)           50000

 lstm_38 (LSTM)              (None, 100, 64)           29440

 lstm_39 (LSTM)              (None, 32)                12416

 dense_25 (Dense)            (None, 64)                2112

 dense_26 (Dense)            (None, 32)                2080

 dense_27 (Dense)            (None, 1)                 33

=================================================================
Total params: 96,081
Trainable params: 96,081
Non-trainable params: 0
_________________________________________________________________
```

6. Compile The Model

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'

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

7. Fit The Model

model.fit(X_train, y_train,
batch_size=128,epochs=10,validation_split=0.2,callbacks=[EarlyStopping(monitor='val_loss'
,patience=2)])

```
model.fit(X_train, y_train, batch_size=128,epochs=10,validation_split=0.2,callbacks=[EarlyStopping(monitor='val_loss',patience=2)])
```

```
Epoch 1/10
28/28 [==============================] - 13s 308ms/step - loss: 0.4777 - accuracy: 0.8603 - val_loss: 0.3748 - val_accuracy: 0.8760
Epoch 2/10
28/28 [==============================] - 8s 272ms/step - loss: 0.3768 - accuracy: 0.8731 - val_loss: 0.3598 - val_accuracy: 0.8760
Epoch 3/10
28/28 [==============================] - 8s 271ms/step - loss: 0.2600 - accuracy: 0.8990 - val_loss: 0.1339 - val_accuracy: 0.9647
Epoch 4/10
28/28 [==============================] - 8s 272ms/step - loss: 0.0874 - accuracy: 0.9772 - val_loss: 0.0870 - val_accuracy: 0.9738
Epoch 5/10
28/28 [==============================] - 8s 271ms/step - loss: 0.0602 - accuracy: 0.9829 - val_loss: 0.0748 - val_accuracy: 0.9761
Epoch 6/10
28/28 [==============================] - 7s 268ms/step - loss: 0.0520 - accuracy: 0.9843 - val_loss: 0.0687 - val_accuracy: 0.9772
Epoch 7/10
28/28 [==============================] - 8s 270ms/step - loss: 0.0350 - accuracy: 0.9898 - val_loss: 0.0646 - val_accuracy: 0.9795
Epoch 8/10
28/28 [==============================] - 8s 269ms/step - loss: 0.0269 - accuracy: 0.9920 - val_loss: 0.0685 - val_accuracy: 0.9761
Epoch 9/10
28/28 [==============================] - 8s 272ms/step - loss: 0.0215 - accuracy: 0.9943 - val_loss: 0.0711 - val_accuracy: 0.9772
<keras.callbacks.History at 0x7f9280f9aa90>
```

8. Save The Model

model.save('spam-classifier.h5')

```
model.save('spam-classifier.h5')
```

9. Test The Model

print("Accuracy of the model on Testing Data is - " , model.evaluate(X_test,y_test)[1]*100 , "%")

```
print("Accuracy of the model on Testing Data is - " , model.evaluate(X_test,y_test)[1]*100 , "%")
```

```
25/25 [==============================] - 1s 26ms/step - loss: 0.0625 - accuracy: 0.9871
Accuracy of the model on Testing Data is -  98.71134161949158 %
```