ASSIGNMENT-4

TEAM ID	**PNT2022TMID38460**
PROJECT NAME	**ANALYTICS FOR HOSPITAL HEALTH-CARE DATA**

• Download dataset

• Import required library

In [96]:

```
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
from keras.utils import np_utils
import keras
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
```

In [97]:

```
os.listdir("D:\IBM\IBM")
print("Change successful.")
```

```
Change successful.
```

• Read dataset and do pre-processing

In [99]:

```
spam_df = pd.read_csv(filepath_or_buffer='Spam.csv',
delimiter=',',encoding='latin-1')
```

```
spam_df.head()
```

Out[99]:

	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

In [100]:

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

Out[100]:

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

In [101]:

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

Out[101]:

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

In [102]:

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

Out[102]:

```
(5572, 2)
```

In [103]:

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

Out[103]:

	v1	v2
count	5572	5572
unique	2	5169

	v1	v2
top	ham	Sorry, I'll call later
freq	4825	30

In [104]:

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

Out[104]:

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

In [105]:

```
#Check for duplicated rows
spam_df.duplicated().sum()
```

Out[105]:

```
403
```

In [106]:

```
#Remove the duplicated rows
spam_df = spam_df.drop_duplicates()
spam_df.duplicated().sum()
```

Out[106]:

```
0
```

In [107]:

```
#Display the count of spam and ham labels
#Stratified-split is required
spam_df['v1'].hist(bins=3)
```

Out[107]:

In [108]:

```
def wordcloud_vis(column):
 mostcommon = nltk.FreqDist(spam_df[column]).most_common(100)
 wordcloud = WordCloud(width=1600, height=800,
background_color='white').generate(str(mostcommon))
 fig = plt.figure(figsize=(30,10), facecolor='white')
 plt.imshow(wordcloud) #, interpolation="bilinear")
 plt.axis('off')
 plt.show()
```

In [109]:

```
#Plot the word-cloud before removing stopwords, performing lemmatization
wordcloud_vis('v2')
```

In [110]:

```
#Retain only the letters and spaces
```

```
spam_df['alpha_text'] = spam_df['v2'].apply(lambda x: re.sub(r'[^a-zA-Z ]+',
'', x.lower()))
spam_df.head()
```

Out[110]:

	v1	v2	alpha_text
0	ham	Go until jurong point, crazy.. Available only ...	go until jurong point crazy available only in ...
1	ham	Ok lar... Joking wif u oni...	ok lar joking wif u oni
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	free entry in a wkly comp to win fa cup final...
3	ham	U dun say so early hor... U c already then say...	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...	nah i dont think he goes to usf he lives aroun...

In [111]:

```
#Remove stop-words
nltk.download('stopwords')
spam_df['imp_text'] = spam_df['alpha_text'].apply(lambda x : ' '.join([word
for word in x.split() if not word in set(stopwords.words('english'))]))
spam_df.head()
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ELCOT\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[111]:

	v1	v2	alpha_text	imp_text
0	ham	Go until jurong point, crazy.. Available only ...	go until jurong point crazy available only in ...	go jurong point crazy available bugis n great ...
1	ham	Ok lar... Joking wif u oni...	ok lar joking wif u oni	ok lar joking wif u oni
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	free entry in a wkly comp to win fa cup final...	free entry wkly comp win fa cup final tkts st ...
3	ham	U dun say so early hor... U c already then say...	u dun say so early hor u c already then say	u dun say early hor u c already say
4	ham	Nah I don't think he goes to usf, he lives aro...	nah i dont think he goes to usf he lives aroun...	nah dont think goes usf lives around though

In [112]:

```
#Tokenize the data
def tokenize(data):
 generated_token = list(data.split())
 return generated_token
spam_df['token_text'] = spam_df['imp_text'].apply(lambda x: tokenize(x))
spam_df.head()
```

Out[112]:

	v1	v2	alpha_text	imp_text	token_text
0	ham	Go until jurong point, crazy.. Available only ...	go until jurong point crazy available only in ...	go jurong point crazy available bugis n great ...	[go, jurong, point, crazy, available, bugis, n...
1	ham	Ok lar... Joking wif u oni...	ok lar joking wif u oni	ok lar joking wif u oni	[ok, lar, joking, wif, u, oni]
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	free entry in a wkly comp to win fa cup final...	free entry wkly comp win fa cup final tkts st ...	[free, entry, wkly, comp, win, fa, cup, final,...
3	ham	U dun say so early hor... U c already then say...	u dun say so early hor u c already then say	u dun say early hor u c already say	[u, dun, say, early, hor, u, c, already, say]
4	ham	Nah I don't think he goes to usf, he lives aro...	nah i dont think he goes to usf he lives aroun...	nah dont think goes usf lives around though	[nah, dont, think, goes, usf, lives, around, t...

In [113]:

```
#Perform lemmatization
nltk.download('wordnet')
nltk.download('omw-1.4')
lemmatizer = WordNetLemmatizer()
def lemmatization(list_of_words):
 lemmatized_list = [lemmatizer.lemmatize(word) for word in list_of_words]
 return lemmatized_list
spam_df['lemmatized_text'] = spam_df['token_text'].apply(lambda x:
lemmatization(x))
spam_df.head()
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\ELCOT\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\ELCOT\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

Out[113]:

	v1	v2	alpha_text	imp_text	token_text	lemmatized_text

	v1	v2	alpha_text	imp_text	token_text	lemmatized_text
0	ham	Go until jurong point, crazy.. Available only ...	go until jurong point crazy available only in ...	go jurong point crazy available bugis n great ...	[go, jurong, point, crazy, available, bugis, n...	[go, jurong, point, crazy, available, bugis, n...
1	ham	Ok lar... Joking wif u oni...	ok lar joking wif u oni	ok lar joking wif u oni	[ok, lar, joking, wif, u, oni]	[ok, lar, joking, wif, u, oni]
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	free entry in a wkly comp to win fa cup final...	free entry wkly comp win fa cup final tkts st ...	[free, entry, wkly, comp, win, fa, cup, final,...	[free, entry, wkly, comp, win, fa, cup, final,...
3	ham	U dun say so early hor... U c already then say...	u dun say so early hor u c already then say	u dun say early hor u c already say	[u, dun, say, early, hor, u, c, already, say]	[u, dun, say, early, hor, u, c, already, say]
4	ham	Nah I don't think he goes to usf, he lives aro...	nah i dont think he goes to usf he lives aroun...	nah dont think goes usf lives around though	[nah, dont, think, goes, usf, lives, around, t...	[nah, dont, think, go, usf, life, around, though]

In [114]:

```
#Combine the tokens (into sentences) to get the final cleansed data
spam_df['clean'] = spam_df['lemmatized_text'].apply(lambda x: ' '.join(x))
spam_df.head()
```

Out[114]:

	v1	v2	alpha_text	imp_text	token_text	lemmatized_text	clean
0	ham	Go until jurong point, crazy.. Available only ...	go until jurong point crazy available only in ...	go jurong point crazy available bugis n great ...	[go, jurong, point, crazy, available, bugis, n...	[go, jurong, point, crazy, available, bugis, n...	go jurong point crazy available bugis n great ...
1	ham	Ok lar... Joking wif u oni...	ok lar joking wif u oni	ok lar joking wif u oni	[ok, lar, joking, wif, u, oni]	[ok, lar, joking, wif, u, oni]	ok lar joking wif u oni
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	free entry in a wkly comp to win fa cup final...	free entry wkly comp win fa cup final tkts st ...	[free, entry, wkly, comp, win, fa, cup, final,...	[free, entry, wkly, comp, win, fa, cup, final,...	free entry wkly comp win fa cup final tkts st ...
3	ham	U dun say so early hor... U c already then	u dun say so early hor u c already then say	u dun say early hor u c already say	[u, dun, say, early, hor, u, c, already, say]	[u, dun, say, early, hor, u, c, already, say]	u dun say early hor u c already say

	v1	v2	alpha_text	imp_text	token_text	lemmatized_text	clean
		say...					
4	ham	Nah I don't think he goes to usf, he lives aro...	nah i dont think he goes to usf he lives aroun...	nah dont think goes usf lives around though	[nah, dont, think, goes, usf, lives, around, t...	[nah, dont, think, go, usf, life, around, though]	nah dont think go usf life around though

In [115]:

```
#Display the wordcloud after preprocessing
wordcloud_vis('clean')
```

In [116]:

```
#Number of unique words in spam and ham
df1 = spam_df.loc[spam_df['v1'] == 'spam']
df2 = spam_df.loc[spam_df['v1'] == 'ham']
spam = set()
df1['clean'].str.lower().str.split().apply(spam.update)
print("Number of unique words in spam", len(spam))
ham = set()
df2['clean'].str.lower().str.split().apply(ham.update)
print("Number of unique words in ham", len(ham))
```

```
Number of unique words in spam 2037
Number of unique words in ham 6738
```

In [117]:

```
#Find the number of overlapping words between spam and ham labels
print("Number of overlapping words between spam and ham: ", len(spam & ham))
```

```
Number of overlapping words between spam and ham:  895
```

In [118]:

```
#Maximum number of words in a sentence
#Useful for applying padding
spam_df['clean'].apply(lambda x:len(str(x).split())).max()
```

Out[118]:

```
80
```

In [119]:

```
#Prepare the data for training
X = spam_df['clean']
y = spam_df['v1']
```

In [120]:

```
#Convert the class labels into integer values
le = LabelEncoder()
y = le.fit_transform(y)
y
```

Out[120]:

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

In [121]:

```
X.shape
```

Out[121]:

```
(5169,)
```

In [122]:

```
y.shape
```

Out[122]:

```
(5169,)
```

In [128]:

```
#Split the data into train, test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15,
random_state=42, stratify=y)
```

In [142]:

```
from keras.preprocessing import sequence
tokenizer = Tokenizer(num_words=1000)
tokenizer.fit_on_texts(X_train)
tokenized_train = tokenizer.texts_to_sequences(X_train)
X_train = sequence.pad_sequences(tokenized_train, maxlen=100)
```

In [143]:

```
tokenized_test = tokenizer.texts_to_sequences(X_test)
X_test = sequence.pad_sequences(tokenized_test, maxlen=100)
```

● Create Model

In [144]:

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

● Add Layers (LSTM, Dense-(Hidden Layers), Output)

In [145]:

```
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'))
```

In [146]:

```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_2 (Embedding)     (None, 100, 50)           50000
 lstm_4 (LSTM)               (None, 100, 64)           29440

 lstm_5 (LSTM)               (None, 32)                12416

 dense_6 (Dense)             (None, 64)                2112

 dense_7 (Dense)             (None, 32)                2080

 dense_8 (Dense)             (None, 1)                 33

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

● Compile the Model

In [147]:

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

● Fit the Model

In [148]:

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

```
Epoch 1/10
28/28 [==============================] - 34s 802ms/step - loss: 0.4799 - accu
racy: 0.8503 - val_loss: 0.3464 - val_accuracy: 0.8760
Epoch 2/10
28/28 [==============================] - 20s 706ms/step - loss: 0.2708 - accu
racy: 0.8876 - val_loss: 0.1725 - val_accuracy: 0.9465
Epoch 3/10
28/28 [==============================] - 18s 638ms/step - loss: 0.1105 - accu
racy: 0.9738 - val_loss: 0.0793 - val_accuracy: 0.9761
Epoch 4/10
28/28 [==============================] - 17s 613ms/step - loss: 0.0544 - accu
racy: 0.9841 - val_loss: 0.0706 - val_accuracy: 0.9772
Epoch 5/10
28/28 [==============================] - 24s 871ms/step - loss: 0.0428 - accu
racy: 0.9878 - val_loss: 0.0798 - val_accuracy: 0.9738
Epoch 6/10
28/28 [==============================] - 19s 672ms/step - loss: 0.0323 - accu
racy: 0.9895 - val_loss: 0.0730 - val_accuracy: 0.9795
```

Out[148]:

● Save The Model

In [149]:

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

● Test The Model

In [150]:

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

```
25/25 [==============================] - 2s 91ms/step - loss: 0.0620 - accura
cy: 0.9845
Accuracy of the model on Testing Data is -  98.45361113548279 %
```