Assignment 4: SMS SPAM Classification

Team Lead: Adarsh Srinivas (Roll No: 310619104003)

2. Import required library

```
In [1]: 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. Read dataset and do pre-processing

Read Dataset

```
In [2]: df = pd.read_csv(r".\spam.csv", encoding='latin-1')
```

```
In [3]: df.head()
```

Out[3]:

	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 [4]: df.shape
```

```
Out[4]: (5572, 5)
```

Drop Unwanted Column

```
In [5]: df = df.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1)
        df = df.rename(columns={"v2" : "Text", "v1":"Label"})
```

```
In [6]:   df.head()
```

Out[6]:

	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

```
In [7]:   df.isnull().sum()
```

```
Out[7]:   Label    0
          Text     0
          dtype: int64
```

```
In [8]:   df.duplicated().sum()
```

```
Out[8]:   403
```

```
In [9]:   df = df.drop_duplicates(keep='first')
          df.duplicated().sum()
```

```
Out[9]:   0
```

```
In [10]:  df.shape
```

```
Out[10]:  (5169, 2)
```

Normalizing the case, Removing the unwanted punctuations, Remove Stopwords

```
In [11]:  ps = PorterStemmer()
```

```
In [12]:  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)
```

In [13]:
```
df['Transformed_Text'] = df['Text'].apply(transform_text)
```

In [15]:
```
df.head()
```

Out[15]:

	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

In [16]:
```
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

In [17]:
```
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

In [18]:
```
x = df.Transformed_Text
y = df.Label
le = LabelEncoder()
y = le.fit_transform(y)
y = y.reshape(-1,1)
```

In [20]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.18, random_s
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

Out[20]:
```
((4238,), (4238, 1), (931,), (931, 1))
```

In [21]:
```
model = Sequential()
```

5. Add Layers

In [22]:
```
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

```
In [24]: model.add(Embedding(total_words_length, 50, input_length = avg_words_len))
```

LSTM Layer

```
In [25]: model.add(LSTM(64))
```

Hidden Layer

```
In [26]: model.add(Dense(64, activation = "relu"))
```

```
In [27]: model.add(Flatten())
```

```
In [28]: model.add(Dropout(0.2))
```

```
In [29]: model.add(Dense(32, activation = "relu"))
```

Output Layer

```
In [30]: model.add(Dense(1, activation = 'sigmoid'))
```

Model Summary

```
In [31]: model.summary()
```

```
Model: "sequential"

_____
 Layer (type)                Output Shape              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

=================================================================
Total params: 372,513
Trainable params: 372,513
Non-trainable params: 0
_____
```

6. Compile the Model

```
In [33]: 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. Fit the Model

```
In [34]: epochs=5
         history = model.fit(x_train, y_train, epochs = epochs, validation_steps=0.18, batc
```

```
Epoch 1/5
424/424 [==============================] - 16s 14ms/step - loss: 0.1346 - accurac
y: 0.9552
Epoch 2/5
424/424 [==============================] - 6s 15ms/step - loss: 0.0356 - accuracy:
0.9887
Epoch 3/5
424/424 [==============================] - 6s 15ms/step - loss: 0.0203 - accuracy:
0.9941
Epoch 4/5
424/424 [==============================] - 6s 14ms/step - loss: 0.0096 - accuracy:
0.9969
Epoch 5/5
424/424 [==============================] - 6s 15ms/step - loss: 0.0043 - accuracy:
0.9988
```

8. Save the Model

```
In [35]: model.save("spam_analysis.h5")
```

9. Test the Model

```
In [36]: test_sequences = tokenizer.texts_to_sequences(x_test)
         x_test = sequence.pad_sequences(test_sequences, maxlen=avg_words_len)
```

```
In [37]: accuracy = model.evaluate(x_test, y_test)
```

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

```
In [38]: 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")
```

```
In [39]: review1 = ["think he goes"]
         predict(review1)
```

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

```
In [40]: review2 = ["Go until jurong point"]
         predict(review2)
```

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

```
In [41]: review3 = ["WINNER!! As a valued network"]
         predict(review3)
```

```
1/1 [==============================] - 0s 47ms/step
spam
```

```
In [42]: review4 = ["URGENT! You have won a 1 week FREE membership"]
         predict(review4)
```

```
1/1 [==============================] - 0s 44ms/step
spam
```

```
1/1 [==============================] - 0s 44ms/step
spam
```