Assignment 01: Analysing Spam Collection Data

The comments/sections provided are your cues to perform the assignment. You don't need to limit yourself to the number of rows/cells provided. You can add additional rows in each section to add more lines of code.

If at any point in time you need help on solving this assignment, view our demo video to understand the different steps of the code.

Happy coding!

Analysing Spam Collection Data

DESCRIPTION

Problem:

Analyze the given Spam Collection dataset to:

1. View information on the spam data,
2. View the length of messages,
3. Define a function to eliminate stopwords,
4. Apply Bag of Words,
5. Apply tf-idf transformer, and
6. Detect Spam with Naïve Bayes model.

1. View information on the spam data

In [1]:
```
#import required Libraries
import pandas as pd
import string
from nltk.corpus import stopwords
```

In [2]:
```
#Get the spam data collection
df_spam_collection = pd.read_csv('D:\\NIPUN_SC_REC\\3_Practice_Project\\Course_5_Data Science with Python\\Prac
                                 sep='\t')
```

In [3]:
```
#viewing first five records
df_spam_collection.head()
```

Out[3]:

	ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
0	ham	Ok lar... Joking wif u oni...
1	spam	Free entry in 2 a wkly comp to win FA Cup fina...
2	ham	U dun say so early hor... U c already then say...
3	ham	Nah I don't think he goes to usf, he lives aro...
4	spam	FreeMsg Hey there darling it's been 3 week's n...

In [4]:
```
df_spam_collection.describe()
```

Out[4]:

	ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
count	5571	5571
unique	2	5168
top	ham	Sorry, I'll call later
freq	4824	30

In [5]:
```
#Reading the file once again to include Feature name (response and message)
df_spam_collection = pd.read_csv('D:\\NIPUN_SC_REC\\3_Practice_Project\\Course_5_Data Science with Python\\Prac
                                 sep='\t',names=['response','message'])
```

In [6]:
```
#Using head(), verify the dataset
df_spam_collection.head()
```

Out[6]:

	response	message
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...

In [7]:
```
#view response using group by and describe method
df_spam_collection.groupby('response').describe()
```

Out[7]:

	message				
	count	unique	top	freq
response				
ham	4825	4516	Sorry, I'll call later	30
spam	747	653	Please call our customer service representativ...	4

2. View the length of messages

In [8]:
```
#Verify length of the messages and also add it as a new column
df_spam_collection['length'] = df_spam_collection['message'].apply(len)
```

In [9]:
```
df_spam_collection.head()
```

Out[9]:

	response	message	length
0	ham	Go until jurong point, crazy.. Available only ...	111
1	ham	Ok lar... Joking wif u oni...	29
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	155
3	ham	U dun say so early hor... U c already then say...	49
4	ham	Nah I don't think he goes to usf, he lives aro...	61

3. Define a function to eliminate stopwords

In [10]:
```
#define a function to get rid of stopwords present in the messages
def remove_stopwords(stpwrd):
    #checking characters to check whether punctuations are present
    remove_punctuation = (char for char in stpwrd if char not in string.punctuation)
    #checking for no punctuation in sentence
    remove_punctuation = ''.join(remove_punctuation)
    #removing stopwords
    return [word for word in remove_punctuation.split() if word.lower() not in stopwords.words('english')]
```

In [11]:
```
df_spam_collection['message'].head(5).apply(remove_stopwords)
```

Out[11]:
```
0    [Go, jurong, point, crazy, Available, bugis, n...
1                      [Ok, lar, Joking, wif, u, oni]
2    [Free, entry, 2, wkly, comp, win, FA, Cup, fin...
3          [U, dun, say, early, hor, U, c, already, say]
4    [Nah, dont, think, goes, usf, lives, around, t...
Name: message, dtype: object
```

4. Apply Bag of Words

In [12]:
```
#start text processing with vectorizer
from sklearn.feature_extraction.text import CountVectorizer
```

In [13]:
```
#use bag of words by applying the function and fit the data into it
bag_of_words_transformer = CountVectorizer(analyzer=remove_stopwords).fit(df_spam_collection['message'])
```

In [14]:
```
#print length of bag of words stored in the vocabulary_ attribute
print (len(bag_of_words_transformer.vocabulary_))
```

```
11425
```

In [15]:
```
#store bag_of_words for messages using transform method
message_bag_of_words = bag_of_words_transformer.transform(df_spam_collection['message'])
```

5. Apply tf-idf transformer

In [16]:
```
#apply tfidf transformer and fit the bag of words into it (transformed version)
from sklearn.feature_extraction.text import TfidfTransformer
tfidf_transformer = TfidfTransformer().fit(message_bag_of_words)
```

In [17]:
```
#print shape of the tfidf
message_tfidf = tfidf_transformer.transform(message_bag_of_words)
print(message_tfidf.shape)
```

```
(5572, 11425)
```

6. Detect Spam with Naïve Bayes model

In [18]:
```
#choose naive Bayes model to detect the spam and fit the tfidf data into it
from sklearn.naive_bayes import MultinomialNB
spam_detection_model = MultinomialNB().fit(message_tfidf,df_spam_collection['response'])
```

In [19]:
```
#check model for the predicted and expected value say for message#2
message = df_spam_collection['message'][2]
bag_of_words_for_message = bag_of_words_transformer.transform([message])
tfidf = tfidf_transformer.transform(bag_of_words_for_message)

print('checking model for message#2..')
print ('Predicted:',spam_detection_model.predict(tfidf)[0])
print ('Expected:',df_spam_collection.response[2])
```

```
checking model for message#2..
Predicted: spam
Expected: spam
```

In [20]:
```
#check model for the predicted and expected value say for message#4
message = df_spam_collection['message'][4]
bag_of_words_for_message = bag_of_words_transformer.transform([message])
tfidf = tfidf_transformer.transform(bag_of_words_for_message)
#check model for the predicted and expected value say for message#5
print('checking model for message#4..')
print ('Predicted:',spam_detection_model.predict(tfidf)[0])
print ('Expected:',df_spam_collection.response[4])
```

```
checking model for message#4..
Predicted: ham
Expected: ham
```

In [ ]:
```

```