NLP_Spam Detection_Completed

July 19, 2020

[2]:
```python
import pandas as pd
import string
from nltk.corpus import stopwords
```

[3]:
```python
#Get the spam data collection
df_spam_collection = pd.read_csv("SpamCollection", sep="        ",␣
 ↪names=['response', 'message'])
```

[4]:
```python
df_spam_collection.head()
```

[4]:
```
  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…
```

[6]:
```python
df_spam_collection.describe()
```

[6]:
```
       response                 message
count      5572                    5572
unique        2                    5169
top         ham  Sorry, I'll call later
freq       4825                      30
```

[7]:
```python
#view response
df_spam_collection.groupby('response').describe()
```

[7]:
```
         message                                                        \
           count unique                                            top
response
ham         4825   4516                           Sorry, I'll call later
spam         747    653  Please call our customer service representativ…


         freq
response
```

```
ham        30
spam        4
```

[9]:
```
#Verify length of the messages and also add it as a new column
df_spam_collection["length"] = df_spam_collection["message"].apply(len)
df_spam_collection.head()
```

[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
```

[ ]:
```

```

[11]:
```
#define a function to get rid of stopwords present in the messages
def message_text_process(mess):
    no_punctuation = [char for char in mess if char not in string.punctuation]
    no_punctuation = "".join(no_punctuation)
    return [word for word in no_punctuation.split() if word.lower() not in␣
 ↪stopwords.words('english')]
```

[12]:
```
#verify that function is working
df_spam_collection["message"].head().apply(message_text_process)
```

[12]:
```
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
```

[15]:
```
#start text processing with vectorizer
from sklearn.feature_extraction.text import CountVectorizer
```

[16]:
```
#use bag of words by applying the function and fit the data into it
bag_of_words_transformer = CountVectorizer(analyzer=message_text_process).
 ↪fit(df_spam_collection["message"])
```

[18]:
```
#print length of bag of words stored in the vocabulary_ attribute
len(bag_of_words_transformer.vocabulary_)
```

[18]:
```
11425
```

[19]:
```
message_bagofwords = bag_of_words_transformer.
 ↪transform(df_spam_collection["message"])
```

```
[21]: #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_bagofwords)
      message_tfidf_transformer = tfidf_transformer.transform(message_bagofwords)
```

```
[22]: #print shape of the tfidf
      message_tfidf_transformer.shape
```

```
[22]: (5572, 11425)
```

```
[38]: #choose naive Bayes model to detect the spam and fit the tfidf data into it
      from sklearn.naive_bayes import MultinomialNB
      spam_detect_model = MultinomialNB().fit(message_tfidf_transformer,␣
       ↪df_spam_collection["response"])
```

```
[41]: #check model for the predicted and expected value say for message#2 and␣
       ↪message#5
      message = df_spam_collection["message"][6]
      bow_for_message = bag_of_words_transformer.transform([message])
      tfidf = tfidf_transformer.transform(bow_for_message)
```

```
[42]: print("Predicted : " , spam_detect_model.predict(tfidf)[0])
      print("Actual : ", df_spam_collection["response"][6])
```

```
Predicted :  ham
Actual :  ham
```