Assignment 02: Sentiment Analysis using NLP

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!

DESCRIPTION

What to:

Analyze the Sentiment dataset using NLP to:

1. View the observations,
2. Verify the length of the messages and add it as a new column,
3. Apply a transformer and fit the data in the bag of words,
4. Print the shape for the transformer, and
5. Check the model for predicted and expected values.

1. View the Observation

In [1]:
```
#import required libraries
import pandas as pd
```

In [2]:
```
#get the sentiment dataset
df_sentiment = pd.read_csv('D:\\NIPUN_SC_REC\\3_Practice_Project\\Course_5_Data Science with Python\\Practice_p
                           sep='\t',names=['comment','label'])
```

In [3]:
```
#view first 10 observations.
# 1 indicates positive sentiment and 0 indicate negative sentiment
df_sentiment.head(10)
```

Out[3]:

	comment	label
0	A very, very, very slow-moving, aimless movie ...	0
1	Not sure who was more lost - the flat characte...	0
2	Attempting artiness with black & white and cle...	0
3	Very little music or anything to speak of.	0
4	The best scene in the movie was when Gerardo i...	1
5	The rest of the movie lacks art, charm, meanin...	0
6	Wasted two hours.	0
7	Saw the movie today and thought it was a good ...	1
8	A bit predictable.	0
9	Loved the casting of Jimmy Buffet as the scien...	1

In [4]:
```
# view more information about the setiment data using describe method
df_sentiment.describe()
```

Out[4]:

	label
count	748.000000
mean	0.516043
std	0.500077
min	0.000000
25%	0.000000
50%	1.000000
75%	1.000000
max	1.000000

In [5]:
```
#view more info on data
df_sentiment.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 748 entries, 0 to 747
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   comment  748 non-null    object
 1   label    748 non-null    int64 
dtypes: int64(1), object(1)
memory usage: 8.8+ KB
```

In [6]:
```
# view data using group by and describe method
df_sentiment.groupby('label').describe()
```

Out[6]:

	comment				
	count	unique	top	freq
label				
0	362	361	Not recommended.	2
1	386	384	Definitely worth checking out.	2

2. Verify the length of the messages and add it as a new column

In [7]:
```
# Verify length of the messages and also add it also as a new column (feature)
df_sentiment['length'] =df_sentiment['comment'].apply(len)
```

In [8]:
```
# view first 5 messages with length
df_sentiment.head()
```

Out[8]:

	comment	label	length
0	A very, very, very slow-moving, aimless movie ...	0	87
1	Not sure who was more lost - the flat characte...	0	99
2	Attempting artiness with black & white and cle...	0	188
3	Very little music or anything to speak of.	0	44
4	The best scene in the movie was when Gerardo i...	1	108

In [9]:
```
#view first
df_sentiment[df_sentiment['length']>50]['comment'].iloc[0]
```

Out[9]:
```
'A very, very, very slow-moving, aimless movie about a distressed, drifting young man.  '
```

3. Apply a transformer and fit the data in the bag of words

In [10]:
```
# start text processing with vectorizer
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
```

In [11]:
```
# define a function to get rid of stopwords present in the messages
def message_text_process(mess):
    # Check characters to see if there are punctuations
    no_punctuation = [char for char in mess if char not in string.punctuation]
    # now form the sentence.
    no_punctuation = ''.join(no_punctuation)
    # Now eliminate any stopwords
    return [word for word in no_punctuation.split() if word.lower() not in stopwords.words('english')]
```

In [12]:
```
# bag of words by applying the function and fit the data (comment) into it
import string
from nltk.corpus import stopwords
bag_of_words = CountVectorizer(analyzer=message_text_process).fit(df_sentiment['comment'])
```

In [13]:
```
# apply transform method for the bag of words
comment_bagofwords = bag_of_words.transform(df_sentiment['comment'])
```

In [14]:
```
# apply tfidf transformer and fit the bag of words into it (transformed version)
from sklearn.feature_extraction.text import TfidfTransformer
tfidf_transformer = TfidfTransformer().fit(comment_bagofwords)
```

4. Print the shape for the transformer

In [15]:
```
# print shape of the tfidf
comment_tfidf = tfidf_transformer.transform(comment_bagofwords)
print (comment_tfidf.shape)
```

```
(748, 3259)
```

5. Check the model for predicted and expected values

In [16]:
```
#choose naive Bayes model to detect the spam and fit the tfidf data into it
from sklearn.naive_bayes import MultinomialNB
sentiment_detection_model = MultinomialNB().fit(comment_tfidf,df_sentiment['label'])
```

In [17]:
```
# check model for the predicted  and expected value say for comment#1
# 1 indicates positive sentiment and 0 indicate negative sentiment
comment = df_sentiment['comment'][1]
bag_of_words_for_comment = bag_of_words.transform([comment])
tfidf = tfidf_transformer.transform(bag_of_words_for_comment)

print ("For Comment #1..")
print ('predicted sentiment label ', sentiment_detection_model.predict(tfidf)[0])
print ('expected sentiment label', df_sentiment.label[1])
```

```
For Comment #1..
predicted sentiment label  0
expected sentiment label 0
```

In [18]:
```
# check model for the predicted  and expected value say for comment#4
# 1 indicates positive sentiment and 0 indicate negative sentiment
comment = df_sentiment['comment'][4]
bag_of_words_for_comment = bag_of_words.transform([comment])
tfidf = tfidf_transformer.transform(bag_of_words_for_comment)

print ("For Comment #4..")
print ('predicted sentiment label ', sentiment_detection_model.predict(tfidf)[0])
print ('expected sentiment label', df_sentiment.label[4])
```

```
For Comment #4..
predicted sentiment label  1
expected sentiment label 1
```