Rating using NLP techinques (project-2)

Classifying Amazon reviews based on customer ratings using NLP. Reviews provide objective feedback to a product and are therefore inherently useful for consumers. These ratings are often summarized by a numerical rating, or the number of stars. Of course there is more value in the actual text itself than the quantified stars. And at times, the given rating does not truly convey the experience of the product – the heart of the feedback is actually in the text itself. The goal therefore is to build a classifier that would understand the essence of a piece of review and assign it the most appropriate rating based on the meaning of the text.

Question

Need to apply NLP techniques in order to clean the data and train model in order to perform rating prediction

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
df=pd.read_csv('C:/Users/joisp/Downloads/Musical_instruments_reviews.csv')
```

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

	reviewerID	asin	reviewerName	helpful	reviewText	overall	summary	unixReviewTime	reviewTime
0	A2IBPI20UZIR0U	1384719342	cassandra tu "Yeah, well, that's just like, u...	[0, 0]	Not much to write about here, but it does exac...	5.0	good	1393545600	02 28, 2014
1	A14VAT5EAX3D9S	1384719342	Jake	[13, 14]	The product does exactly as it should and is q...	5.0	Jake	1363392000	03 16, 2013
2	A195EZSQDW3E21	1384719342	Rick Bennette "Rick Bennette"	[1, 1]	The primary job of this device is to block the...	5.0	It Does The Job Well	1377648000	08 28, 2013
3	A2C00NNG1ZQQG2	1384719342	RustyBill "Sunday Rocker"	[0, 0]	Nice windscreen protects my MXL mic and preven...	5.0	GOOD WINDSCREEN FOR THE MONEY	1392336000	02 14, 2014
4	A94QU4C90B1AX	1384719342	SEAN MASLANKA	[0, 0]	This pop filter is great. It looks and perform...	5.0	No more pops when I record my vocals.	1392940800	02 21, 2014

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10261 entries, 0 to 10260
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   reviewerID      10261 non-null  object
 1   asin            10261 non-null  object
 2   reviewerName    10234 non-null  object
 3   helpful         10261 non-null  object
 4   reviewText      10254 non-null  object
 5   overall         10261 non-null  float64
 6   summary         10261 non-null  object
 7   unixReviewTime  10261 non-null  int64
 8   reviewTime      10261 non-null  object
dtypes: float64(1), int64(1), object(7)
memory usage: 721.6+ KB
```

In [5]:
```
'''
we are intersted only in two columns i.e
1. 'reviewText'
2. 'overall'
so we will only use those columns in our dataset
'''
print()
```

In [6]:
```
df=pd.read_csv('C:/Users/joisp/Downloads/Musical_instruments_reviews.csv',usecols = ['rev
iewText','overall'])
```

In [7]:
```
df.head()
```

	reviewText	overall
0	Not much to write about here, but it does exac...	5.0
1	The product does exactly as it should and is q...	5.0
2	The primary job of this device is to block the...	5.0
3	Nice windscreen protects my MXL mic and preven...	5.0
4	This pop filter is great. It looks and perform...	5.0

In [8]:
```
df.shape
```

```
(10261, 2)
```

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10261 entries, 0 to 10260
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   reviewText  10254 non-null  object
 1   overall     10261 non-null  float64
dtypes: float64(1), object(1)
memory usage: 160.5+ KB
```

In [10]:
```
# To check the random values or texts in our dataset we use command sample
```

In [11]:
```
df.sample(5)
```

	reviewText	overall
3708	This arrived with one end of the security stra...	1.0
4460	These are condenser mics so you will need phan...	5.0
316	Martin makes great sounding strings. I decide...	4.0
8138	Even though I have a tuner in my pedalboard se...	4.0
3243	Works for my Behringer XM1800S Mics. It came w...	4.0

In [12]:
```
df['overall'].value_counts()
```

```
5.0    6938
4.0    2084
3.0     772
2.0     250
1.0     217
Name: overall, dtype: int64
```

In [13]:
```
'''
In the above output we can see that in our dataset
1 star rated products are --> 217
2 star rated products are --> 250
3 star rated products are --> 772
4 star rated products are --> 2084
5 star rated products are --> 6938
'''
print()
```

Preprocessing the dataset

In [14]:
```
# Here a tool that I found from github which preprocess the data and makes my task easier
```

In [15]:
```
#!pip install git+https://github.com/laxmimerit/preprocess_kgptalkie.git --upgrade --forc
e-reinstall
```

In [16]:
```
#!pip install spacy==2.2.3
```

In [17]:
```
#!pip install textblob==0.15.3
```

In [18]:
```
import preprocess_kgptalkie as ps
import re
```

In [19]:
```
'''
The above package imports everything necessary to pre processing and also imports from sk
learn
like 'CountVectorizer' which are necessary for these kinds problem
some of imports done in this package are:

import re
import os
import sys
import json

import pandas as pd
import numpy as np
import spacy
from spacy.lang.en.stop_words import STOP_WORDS as stopwords
from bs4 import BeautifulSoup
import unicodedata
from textblob import TextBlob
import en_core_web_sm

from sklearn.feature_extraction.text import CountVectorizer
'''
```

```
"\nThe above package imports everything necessary to pre processing and also imports from sklearn \nlike 'CountVectorizer' whi
ch are necessary for these kinds problem\nsome of imports done in this package are:\n\nimport re\nimport os\nimport sys\nimpor
t json\n\nimport pandas as pd\nimport numpy as np\nimport spacy\nfrom spacy.lang.en.stop_words import STOP_WORDS as stopwords
\nfrom bs4 import BeautifulSoup\nimport unicodedata\nfrom textblob import TextBlob\nimport en_core_web_sm\n\nfrom sklearn.feat
ure_extraction.text import CountVectorizer\n"
```

In [20]:
```
def get_clean(x):
    x = str(x).lower().replace('\\', '').replace('_', ' ')
    x = ps.cont_exp(x)
    x = ps.remove_emails(x)
    x = ps.remove_urls(x)
    x = ps.remove_html_tags(x)
    x = ps.remove_accented_chars(x)
    x = ps.remove_special_chars(x)
    x = re.sub("(.)\\1{2,}", "\\1", x)
    return x
```

In [21]:
```
j=get_clean(df)
```

In [22]:
```
j
```

```
'reviewtext overall0 not much to write about here but it does exac 501 the product does exactly as it should and is q 502 the
primary job of this device is to block the 503 nice windscreen protects my mxl mic and preven 504 this pop filter is great it
looks and perform 50 10256 great just as expected thank to all 5010257 i have been thinking about trying the nanoweb st 501025
8 i have tried coated strings in the past incl 4010259 well made by elixir and developed with taylor 4010260 these strings are
really quite good but i wou 4010261 rows x 2 columns'
```

In [23]:
```
# get_clean removes all unnecessary stuff from our data set
# for eg ps.remove_urls(x) removes url's from our data set
```

In [24]:
```
df['reviewText'] = df['reviewText'].apply(lambda x: get_clean(x))
```

In [25]:
```
df.head()
```

	reviewText	overall
0	not much to write about here but it does exact...	5.0
1	the product does exactly as it should and is q...	5.0
2	the primary job of this device is to block the...	5.0
3	nice windscreen protects my mxl mic and preven...	5.0
4	this pop filter is great it looks and performs...	5.0

In [26]:
```
# now the df has been preprossed
```

Implementing algorithms

TFIDF and Linear SVM algos

In [27]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [28]:
```
from sklearn.model_selection import train_test_split
```

In [29]:
```
from sklearn.svm import LinearSVC
```

In [30]:
```
from sklearn.metrics import classification_report
```

In [31]:
```
tfidf=TfidfVectorizer(max_features=20000,ngram_range=(1,5),analyzer='char')
```

In [32]:
```
x= tfidf.fit_transform(df['reviewText'])
```

In [33]:
```
y=df['overall']
```

In [34]:
```
x.shape , y.shape
```

```
((10261, 20000), (10261,))
```

In [35]:
```
x_train , x_test , y_train , y_test = train_test_split(x,y,train_size = 0.8, random_state
=0 )
```

In [36]:
```
x_train.shape
```

```
(8208, 20000)
```

In [37]:
```
clf= LinearSVC()
```

In [38]:
```
clf.fit(x_train,y_train)
```

```
LinearSVC()
```

In [39]:
```
yPred=clf.predict(x_test)
```

In [40]:
```
print(classification_report(y_test,yPred))
```

```
              precision    recall  f1-score   support

         1.0       0.43      0.08      0.13        39
         2.0       0.20      0.02      0.03        55
         3.0       0.28      0.10      0.15       134
         4.0       0.40      0.20      0.27       451
         5.0       0.73      0.93      0.82      1374

    accuracy                           0.68      2053
   macro avg       0.41      0.27      0.28      2053
weighted avg       0.61      0.68      0.62      2053
```

In [41]:
```
'''
The precision of 1.0 and 2.0 is too low so will make some changes in Linear SVC and try a
gain...
'''
print()
```

In [42]:
```
clf_2= LinearSVC(C = 20 , class_weight = 'balanced')
```

In [43]:
```
clf_2.fit(x_train,y_train)
```

```
C:\Users\joisp\anaconda3\lib\site-packages\sklearn\svm\_base.py:976: ConvergenceWarning: Liblinear failed to converge, increas
e the number of iterations.
  warnings.warn("Liblinear failed to converge, increase "


LinearSVC(C=20, class_weight='balanced')
```

In [44]:
```
yPred_2=clf_2.predict(x_test)
```

In [45]:
```
print(classification_report(y_test,yPred_2))
```

```
              precision    recall  f1-score   support

         1.0       0.36      0.23      0.28        39
         2.0       0.19      0.11      0.14        55
         3.0       0.25      0.28      0.26       134
         4.0       0.34      0.34      0.34       451
         5.0       0.77      0.79      0.78      1374

    accuracy                           0.63      2053
   macro avg       0.38      0.35      0.36      2053
weighted avg       0.62      0.63      0.62      2053
```

In [46]:
```
'''
Now the accuraices have got a litle better
'''
print()
```

In [47]:
```
# Now we will be testing our model with external inputs
```

In [48]:
```
x = 'this product is really bad. i do not like it'
```

In [49]:
```
x=get_clean(x)
```

In [50]:
```
vec= tfidf.transform([x])
```

In [51]:
```
clf_2.predict(vec)
```

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

In [52]:
```
''' Array 1. means that the rating is 1 star {Lowest rating} '''
```

In [53]:
```
# Another example
```

In [54]:
```
y = 'this product is really good. thanks a lot for speedy delivery'
```

In [55]:
```
y=get_clean(y)
```

In [56]:
```
vec= tfidf.transform([y])
```

In [57]:
```
clf_2.predict(vec)
```

```
array([5.])
```

In [58]:
```
''' Array 5. means that the rating is 5 star {Highest rating}'''
```

In [ ]:
```
# ----------******----------------*******------------------------******-------------#
```