Assignment-2

Amazon Fine Food Reviews Analysis

Data Source: https://www.kaggle.com/snap/amazon-fine-food-reviews (https://www.kaggle.com/snap/amazon-fine-food-reviews)

EDA: https://nycdatascience.com/blog/student-works/amazon-fine-foods-visualization/ (https://nycdatascience.com/blog/student-works/amazon-fine-foods-visualization/)

The Amazon Fine Food Reviews dataset consists of reviews of fine foods from Amazon.

Number of reviews: 568,454

Number of users: 256,059

Number of products: 74,258

Timespan: Oct 1999 - Oct 2012

Number of Attributes/Columns in data: 10

Attribute Information:

```
1)Id

2)ProductId - unique identifier for the product

3)UserId - unqiue identifier for the user

4)ProfileName

5)HelpfulnessNumerator - number of users who found the review helpful

6)HelpfulnessDenominator - number of users who indicated whether they found the rev
iew helpful or not

7)Score - rating between 1 and 5

8)Time - timestamp for the review

9)Summary - brief summary of the review

10)Text - text of the review
```

Objective:

Given a review, determine whether the review is positive (Rating of 4 or 5) or negative (rating of 1 or 2).

[Q] How to determine if a review is positive or negative?

[Ans] We could use the Score/Rating. A rating of 4 or 5 could be cosnidered a positive review. A review of 1 or 2 could be considered negative. A review of 3 is nuetral and ignored. This is an approximate and proxy way of determining the polarity (positivity/negativity) of a review.

In [1]:
```
# importing library
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer

import re
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer

from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle
```

In [3]:
```
con = sqlite3.connect('C:\\Users\\Dell\\Downloads\\amazon-fine-food-reviews\\d
atabase.sqlite')

#filtering only positive and negative reviews i.e.
# not taking into consideration those reviews with Score=3
#creating new datasets after applying filter on reviews dataset

filtered_data = pd.read_sql_query(""" SELECT * FROM Reviews WHERE Score != 3
 """, con)

# Give reviews with Score>3 a positive rating, and reviews with a score<3 a ne
gative rating.
def partition(x):
    if x < 3:
        return 'negative'
    return 'positive'

#changing reviews with score less than 3 to be positive and vice-versa
# with the help of this method returning positive and negative based on the sc
ore

actualScore = filtered_data['Score']
positiveNegative = actualScore.map(partition)
filtered_data['Score'] = positiveNegative
```

In [4]:
```
print(filtered_data.shape) #looking at the size of the data
filtered_data.head() # top five reviews, just for understanding
```

```
(525814, 10)
```

Out[4]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenomi
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	

In [5]:
```
# delete the reviews which is same on the basis of few features
final=filtered_data.drop_duplicates(subset={"UserId","ProfileName","Time","Tex
t"}, keep='first', inplace=False)
final.shape # after deleting, look at shape again
```

Out[5]: (364173, 10)

In [6]:
```
final.head() # Look at top five reviews
```

Out[6]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenomi
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	

In [7]:
```
# As we know that helfulnessnumerator will not  be greater than helpfullness d
enominator
# So we will remove that reviews because that reviews no make sense

final=final[final.HelpfulnessNumerator<=final.HelpfulnessDenominator]
```

In [8]:
```
# after removing reviews in above cell which were useless, no make sense
# so look at the reviewsprint(final.shape)
print(final.shape)

#How many positive and negative reviews are present in our dataset?
final['Score'].value_counts()
```

```
(364171, 10)
```

Out[8]:
```
positive    307061
negative     57110
Name: Score, dtype: int64
```

Text Preprocessing

In [13]:

```
# Removing Stop-words
import nltk
nltk.download('stopwords')

stop = set(stopwords.words('english')) #set of stopwords
sno = nltk.stem.SnowballStemmer('english') #initialising the snowball stemmer

def cleanhtml(sentence):  #function to clean the word of any html-tags
    cleanr = re.compile('<.*?>')
    cleantext = re.sub(cleanr, ' ', sentence)
    return cleantext
def cleanpunc(sentence):  #function to clean the word of any punctuation or sp
ecial characters
    cleaned = re.sub(r'[?|!|\'|"|#]',r'',sentence)
    cleaned = re.sub(r'[.|,|)|(|\|/]',r' ',cleaned)
    return  cleaned
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Dell\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

In [14]:
```
#Code for implementing step-by-step the checks mentioned in the pre-processing phase
# this code takes a while to run as it needs to run on 500k sentences.
i=0
str1=' '
final_string=[]
all_positive_words=[] # store words from +ve reviews here
all_negative_words=[] # store words from -ve reviews here.
s=''
for sent in final['Text'].values:
    filtered_sentence=[]

    sent=cleanhtml(sent) # remove HTML tags
    for w in sent.split():
        for cleaned_words in cleanpunc(w).split():
            if((cleaned_words.isalpha()) & (len(cleaned_words)>2)): # assure that cleaned words are alphabetical and length is greater than 2
                if(cleaned_words.lower() not in stop):  # thos words who were not in stop words
                    s=(sno.stem(cleaned_words.lower())).encode('utf8') # changing cleaned words into lower case
                    filtered_sentence.append(s)
                    if (final['Score'].values)[i] == 'positive':  #IF words are positive
                        all_positive_words.append(s) #list of all words used to describe positive reviews
                    if(final['Score'].values)[i] == 'negative': # if words are negative
                        all_negative_words.append(s) #list of all words used to describe negative reviews reviews
                else:
                    continue
            else:
                continue

    str1 = b" ".join(filtered_sentence) #final string of cleaned words


    final_string.append(str1) #final_string dataset appending string after cleaning words
    i+=1
```

In [15]:
```
final['CleanedText']=final_string #adding a column of CleanedText which displays the data after pre-processing of the review
final['CleanedText']=final['CleanedText'].str.decode("utf-8")
```

In [16]:
```
final.head()
```

Out[16]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenomi
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	

In [17]:
```
final.shape # look at the shape of final dataset
```

Out[17]: (364171, 11)

In [18]:
```
# sorting data on the basis of time stamp for time based splitting
sorted_data=final.sort_values('Time', axis=0, ascending=True, inplace=False, k
ind='quicksort', na_position='last')
final=sorted_data
```

In [20]:
```
# importing library
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
from sklearn import model_selection
```

train test split

In [22]:
```
# split the data set into train and test
X_tr, X_tes, y_train, y_test = model_selection.train_test_split(final['Cleaned Text'].values, final['Score'], test_size=0.3, random_state=0)
```

bag of words

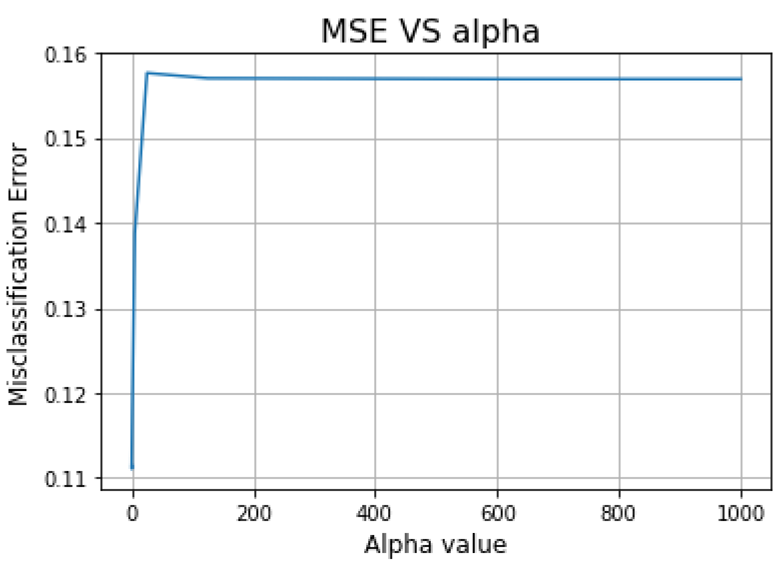
In [23]:
```
# we are collecting all the split words in the form of tokens matrix
count_vect = CountVectorizer()  #in scikit-learn
X_train = count_vect.fit_transform(X_tr)#giving training data set to vectorize the training data
X_test = count_vect.transform(X_tes) #giving testing data set to vectorize the testing data
print("the type of count vectorizer ",type(X_train))
print("the shape of out text BOW vectorizer ",X_train.get_shape())
print("the number of unique words ", X_train.get_shape()[1])
```

```
the type of count vectorizer  <class 'scipy.sparse.csr.csr_matrix'>
the shape of out text BOW vectorizer  (254919, 60080)
the number of unique words  60080
```

To find optimal alpha with BernoulliNB

In [24]:
```
from sklearn.naive_bayes import BernoulliNB # import library for NB

# creating list of alpha values
alpha_values = [0.001 , 0.01 , 0.1, 1 , 5 , 25 , 125 , 625 ,1000]

# empty list that will hold cv scores
cv_scores = []

# perform 10-fold cross validation
for my_alpha in alpha_values: #loop for all the alpha values

    scores = cross_val_score(BernoulliNB(alpha = my_alpha), X_train,y_train, c
v=5,scoring='accuracy',n_jobs=-1) #Evaluate a score by cross-validation
    cv_scores.append(scores.mean())

# changing to misclassification error
MSE = [1 - x for x in cv_scores]

# determining best alpha
optimal_alpha = alpha_values[MSE.index(min(MSE))]
print('\nThe optimal value of alpha is %f.' % optimal_alpha)

# plot misclassification error vs alpha
plt.plot(alpha_values, MSE)


plt.xlabel('Alpha value',size=12)
plt.ylabel('Misclassification Error',size=12)
plt.title('MSE VS alpha ',size=16)
plt.grid()
plt.show()

print("the misclassification error for each alpha value is : ", np.round(MSE,3
))
```

```
The optimal value of alpha is 0.010000.
```




```
the misclassification error for each alpha value is :  [0.111 0.111 0.112 0.1
18 0.139 0.158 0.157 0.157 0.157]
```

In [25]:
```
# training the model with the optimal alpha value
clf = BernoulliNB(optimal_alpha)
nb=clf.fit(X_train,y_train)
```

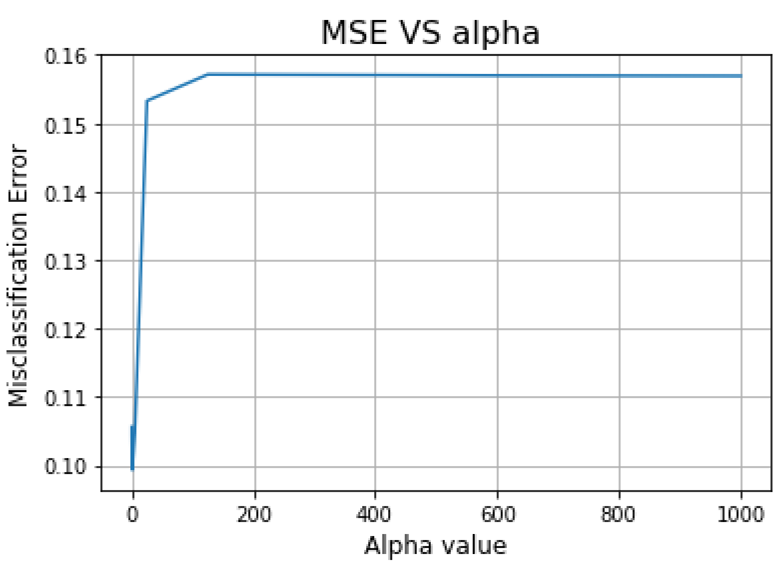
In [26]:
```
# calculating the accuracy score of models

y_pred = clf.predict(X_test)
acc=accuracy_score(y_test, y_pred)
print("accuracy is ", acc)
```

```
accuracy is  0.8913337970929593
```

To find optimal alpha with MultiinomialNB

In [27]:
```
from sklearn.naive_bayes import MultinomialNB # import library for NB

# creating list of alpha values
alpha_values = [0.001 , 0.01 , 0.1, 1 , 5 , 25 , 125 , 625 ,1000]

# empty list that will hold cv scores
cv_scores = []

# perform 10-fold cross validation
for my_alpha in alpha_values: #loop for all the alpha values

    scores = cross_val_score(MultinomialNB(alpha = my_alpha), X_train,y_train,
cv=5,scoring='accuracy',n_jobs=-1) #Evaluate a score by cross-validation
    cv_scores.append(scores.mean())

# changing to misclassification error
MSE = [1 - x for x in cv_scores]

# determining best alpha
optimal_alpha = alpha_values[MSE.index(min(MSE))]
print('\nThe optimal value of alpha is %f.' % optimal_alpha)

# plot misclassification error vs alpha
plt.plot(alpha_values, MSE)


plt.xlabel('Alpha value',size=12)
plt.ylabel('Misclassification Error',size=12)
plt.title('MSE VS alpha ',size=16)
plt.grid()
plt.show()

print("the misclassification error for each alpha value is : ", np.round(MSE,3
))
```

```
The optimal value of alpha is 1.000000.
```




```
the misclassification error for each alpha value is :  [0.106 0.104 0.103 0.0
99 0.105 0.153 0.157 0.157 0.157]
```

In [28]:
```
# training the model with the optimal alpha value
clf = MultinomialNB(optimal_alpha)
nb=clf.fit(X_train,y_train)
```

In [30]:
```
# calculating the accuracy score of models
y_pred = clf.predict(X_test)
acc=accuracy_score(y_test, y_pred)
print("accuracy is ", acc)
```

```
accuracy is  0.9044319554790758
```

In [ ]: