# Opinion Mining on e-Commerce website

Group Members: Venkata Naga Ravi Teja Lanka | Ravali Kotha | Shruthi Reddy Gujjula | Sai Srikanth Kotamraju

# Contents

# Introduction:

In today's business world, it is essential to put the customers emotional needs ahead of the company's and employee's agendas. Long-term relationship between a business and its customers can change significantly—for better or for worse. To ensure that more of those moments have a positive outcome, companies are investing their money in traditional royalty programs such as Customer-relationship-Management (CRM) technology. But most of these initiatives end with disappointment.

What's lacking here is the spark between the customer and the company which brings the customers trust and loyalty. According to Mckinsey study, after a positive customer experience, more than 85 percent of customers made more purchases. After a negative experience, more than 70 percent made relatively less purchases.

So now, how can a business know exactly what makes the customer feel like they are receiving superior service?

## Sentimental Analysis

The answer lies in deep analysis of Customer sentiments. Opinion mining/sentiment analysis is not just assigning a positive or negative label. By analyzing the sentiment more accurately, and finding the things people are *unhappy* about, we can focus more on what will make a difference.

## The impact of Sentimental Analysis

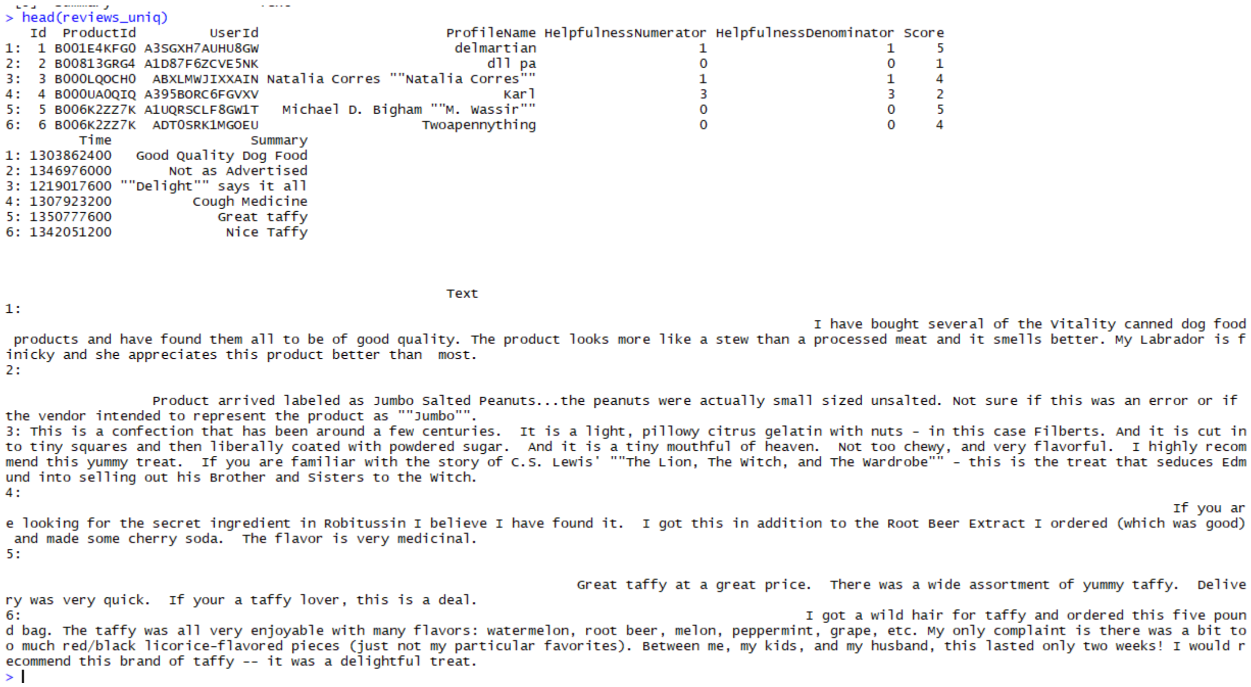
- ❑ Target individuals to improve their service
- ❑ Track customer sentiment over time
- ❑ Determine if customer segments feel more strongly about your company
- ❑ Track how a change in product or service affects how customers feel
- ❑ Determine your key promoters and detractors

# Data Preparation:

## Source Data:

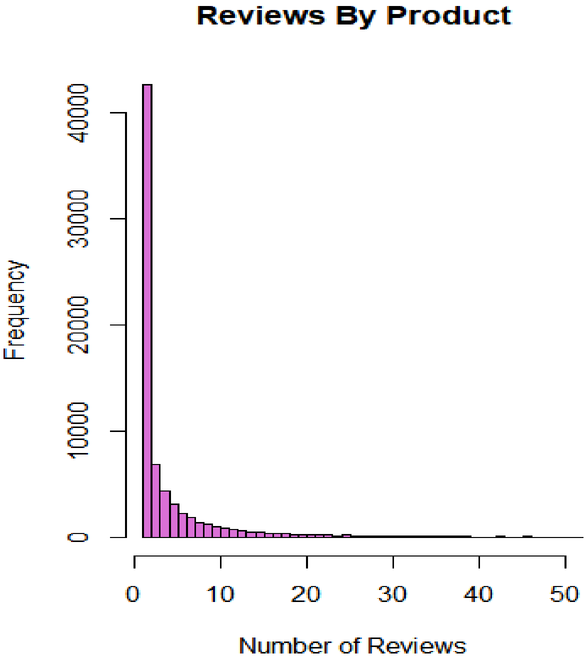
The data source for this project is Amazon Food Reviews dataset available on Kaggle. The dataset contains 500k reviews of 74k products from 250k users. The dataset has 10 columns namely: Id, ProductId, UserId, ProfileName, HelpfulnessNumerator, HelpfulnessDenominator, Score, Time, Summary, Text.

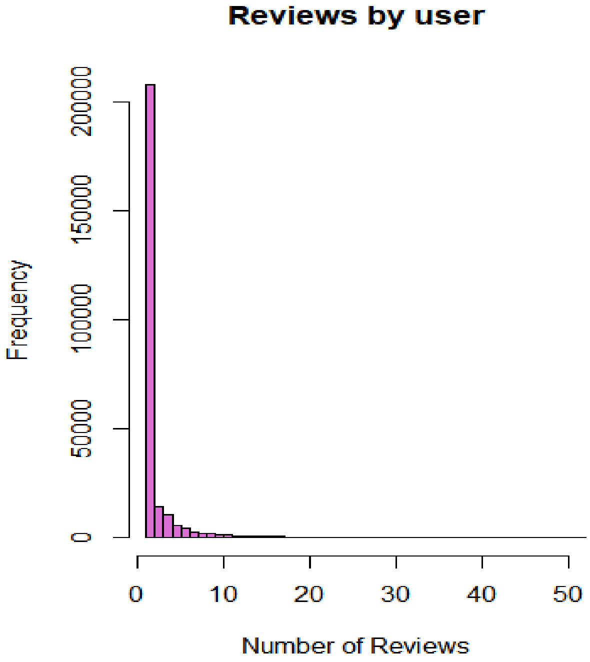
The first 6 observations for the dataset are below:



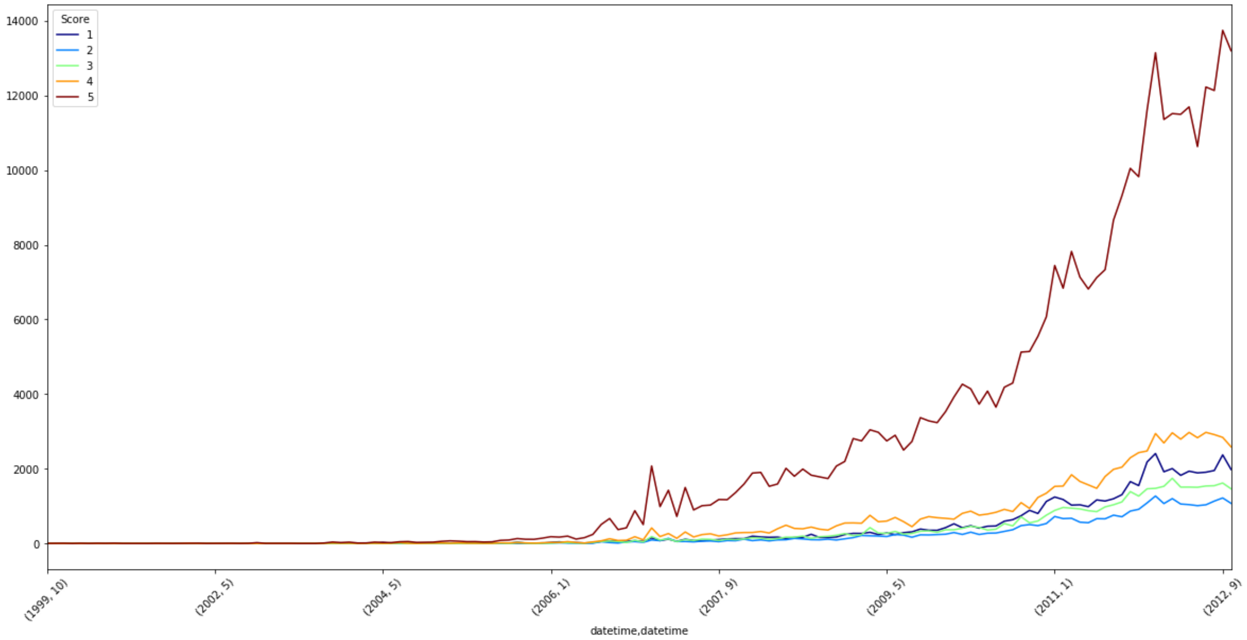
```
> head(reviews_uniq)
   Id  ProductId         UserId                      ProfileName HelpfulnessNumerator HelpfulnessDenominator Score
1:  1 B001E4KFG0 A3SGXH7AUHU8GW                        delmartian                    1                      1     5
2:  2 B00813GRG4 A1D87F6ZCVE5NK                            dll pa                    0                      0     1
3:  3 B000LQOCH0  ABXLMWJIXXAIN Natalia Corres ""Natalia Corres""                    1                      1     4
4:  4 B000UA0QIQ A395BORC6FGVXV                              Karl                    3                      3     2
5:  5 B006K2ZZ7K A1UQRSCLF8GW1T   Michael D. Bigham ""M. Wassir""                    0                      0     5
6:  6 B006K2ZZ7K  ADT0SRK1MGOEU                    Twoapennything                    0                      0     4
         Time                 Summary
1: 1303862400   Good Quality Dog Food
2: 1346976000       Not as Advertised
3: 1219017600 ""Delight"" says it all
4: 1307923200          Cough Medicine
5: 1350777600             Great taffy
6: 1342051200              Nice Taffy



                                                      Text
1:
                                                                                                  I have bought several of the Vitality canned dog food
 products and have found them all to be of good quality. The product looks more like a stew than a processed meat and it smells better. My Labrador is f
inicky and she appreciates this product better than  most.
2:

                  Product arrived labeled as Jumbo Salted Peanuts...the peanuts were actually small sized unsalted. Not sure if this was an error or if
the vendor intended to represent the product as ""Jumbo"".
3: This is a confection that has been around a few centuries.  It is a light, pillowy citrus gelatin with nuts - in this case Filberts. And it is cut in
to tiny squares and then liberally coated with powdered sugar.  And it is a tiny mouthful of heaven.  Not too chewy, and very flavorful.  I highly recom
mend this yummy treat.  If you are familiar with the story of C.S. Lewis' ""The Lion, The Witch, and The Wardrobe"" - this is the treat that seduces Edm
und into selling out his Brother and Sisters to the Witch.
4:
                                                                                                                                              If you ar
e looking for the secret ingredient in Robitussin I believe I have found it.  I got this in addition to the Root Beer Extract I ordered (which was good)
 and made some cherry soda.  The flavor is very medicinal.
5:

                                                                     Great taffy at a great price.  There was a wide assortment of yummy taffy.  Delive
ry was very quick.  If your a taffy lover, this is a deal.
6:                                                                                               I got a wild hair for taffy and ordered this five poun
d bag. The taffy was all very enjoyable with many flavors: watermelon, root beer, melon, peppermint, grape, etc. My only complaint is there was a bit to
o much red/black licorice-flavored pieces (just not my particular favorites). Between me, my kids, and my husband, this lasted only two weeks! I would r
ecommend this brand of taffy -- it was a delightful treat.
> 
```

## Data Exploration:





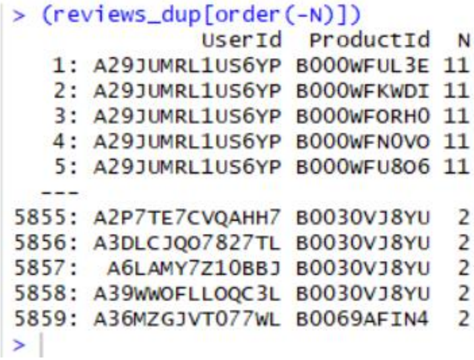
The first histogram gives us the frequency of the total reviews received for all the products. The second histogram gives us the frequency of the total reviews given by each user.



The above graph shows the Rating distribution over time. We observe that users have increased providing a rating score of 5 from 2006.

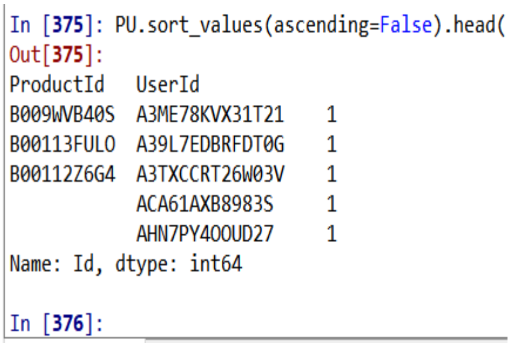
## Remove Duplicates:

We’ve observed that there were multiple entries of the reviews given by each user to each product. We have removed such entries in our data.



```
> (reviews_dup[order(-N)])
                 UserId  ProductId  N
   1: A29JUMRL1US6YP B000WFUL3E 11
   2: A29JUMRL1US6YP B000WFKWDI 11
   3: A29JUMRL1US6YP B000WFORH0 11
   4: A29JUMRL1US6YP B000WFN0VO 11
   5: A29JUMRL1US6YP B000WFU806 11
  ---
5855: A2P7TE7CVQAHH7 B0030VJ8YU  2
5856: A3DLCJQO7827TL B0030VJ8YU  2
5857:  A6LAMY7Z10BBJ B0030VJ8YU  2
5858: A39WWOFLLOQC3L B0030VJ8YU  2
5859: A36MZGJVT077WL B0069AFIN4  2
> |
```

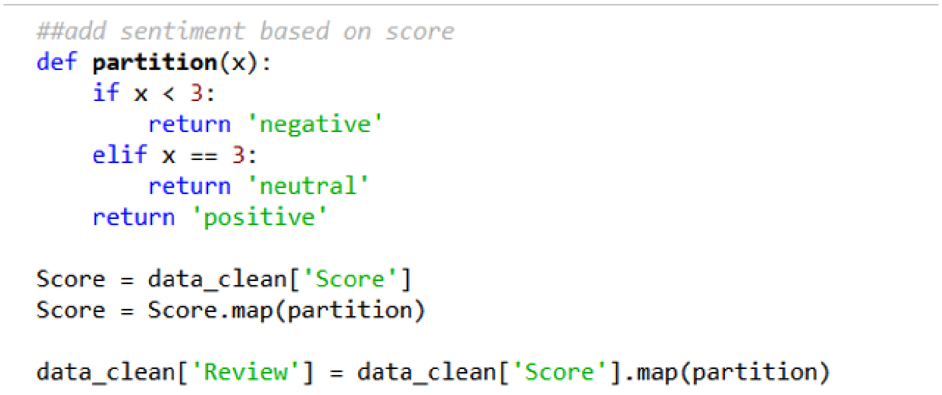
Below is the evidence of cleaned data. We observed that 1% of the total data has been removed as duplicate values.



```
In [375]: PU.sort_values(ascending=False).head(
Out[375]:
ProductId   UserId
B009WVB40S  A3ME78KVX31T21    1
B00113FULO  A39L7EDBRFDT0G    1
B00112Z6G4  A3TXCCRT26W03V    1
            ACA61AXB8983S     1
            AHN7PY4OOUD27     1

Name: Id, dtype: int64

In [376]:
```

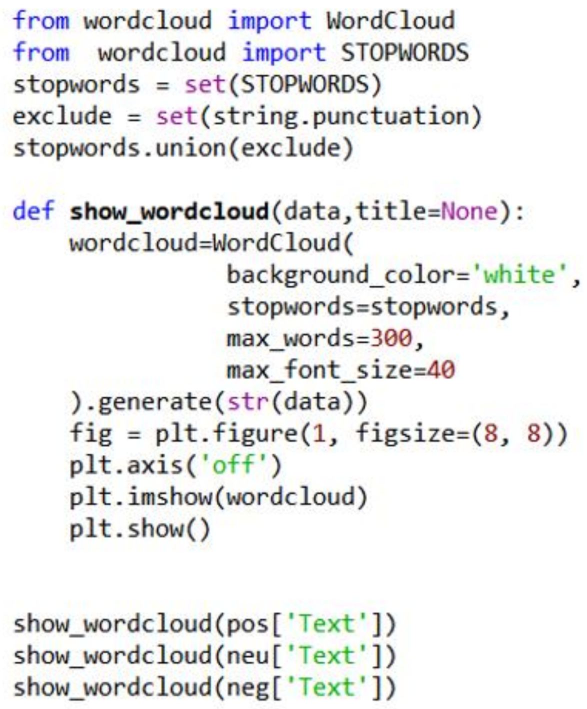
# Word Cloud:

Next let us proceed to splitting the reviews into Positive, Negative, Neutral based on Review score.



```
##add sentiment based on score
def partition(x):
    if x < 3:
        return 'negative'
    elif x == 3:
        return 'neutral'
    return 'positive'

Score = data_clean['Score']
Score = Score.map(partition)

data_clean['Review'] = data_clean['Score'].map(partition)
```

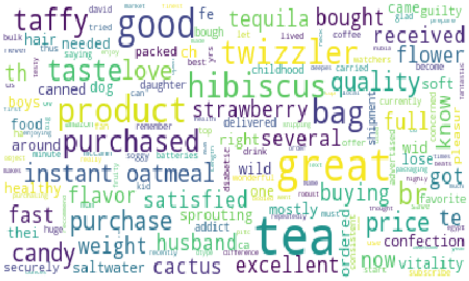
Next, we proceed to development of word cloud which creates group of words to know the products which attracted positive or negative or neutral reviews from customers. Based on the comments we categorize words into three categories. To develop this word cloud first we must clean data to remove stop words and punctuations. The following code explains in detail how to develop a word cloud.



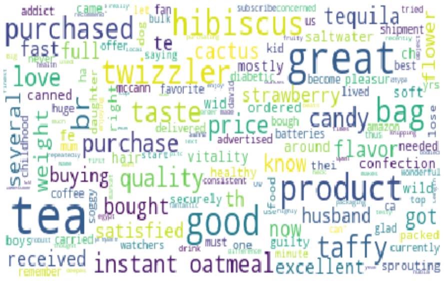
```
from wordcloud import WordCloud
from  wordcloud import STOPWORDS
stopwords = set(STOPWORDS)
exclude = set(string.punctuation)
stopwords.union(exclude)

def show_wordcloud(data,title=None):
    wordcloud=WordCloud(
                background_color='white',
                stopwords=stopwords,
                max_words=300,
                max_font_size=40
    ).generate(str(data))
    fig = plt.figure(1, figsize=(8, 8))
    plt.axis('off')
    plt.imshow(wordcloud)
    plt.show()


show_wordcloud(pos['Text'])
show_wordcloud(neu['Text'])
show_wordcloud(neg['Text'])
```

The output is as follows:

**Positive**



**Neutral**

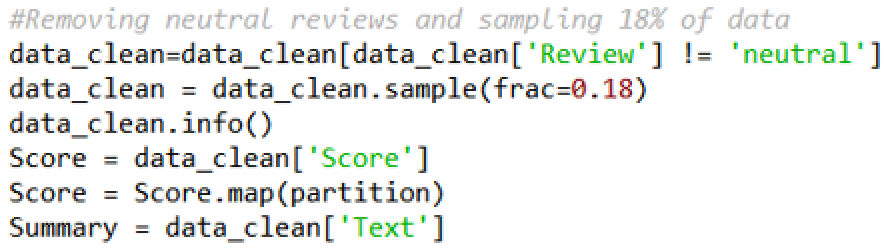


**Negative**

old
product
eating
felidae
pretzels
month
th
retriever
good
flavor
tea
arrived
puppy
purchased
bought
golden
candy
dog
regular
even
days
taste
treat
loved
love
protein
thing
ball
excited
labrador
everlasting
sweetener
high
say

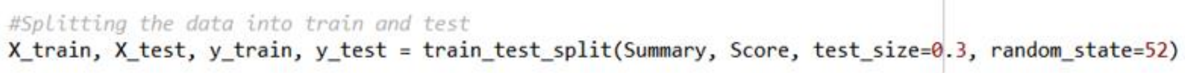
## Classification of Reviews:

Now we developed code to classify comments as positive and negative. First step to accomplish this is to remove neutral comments from the dataset.



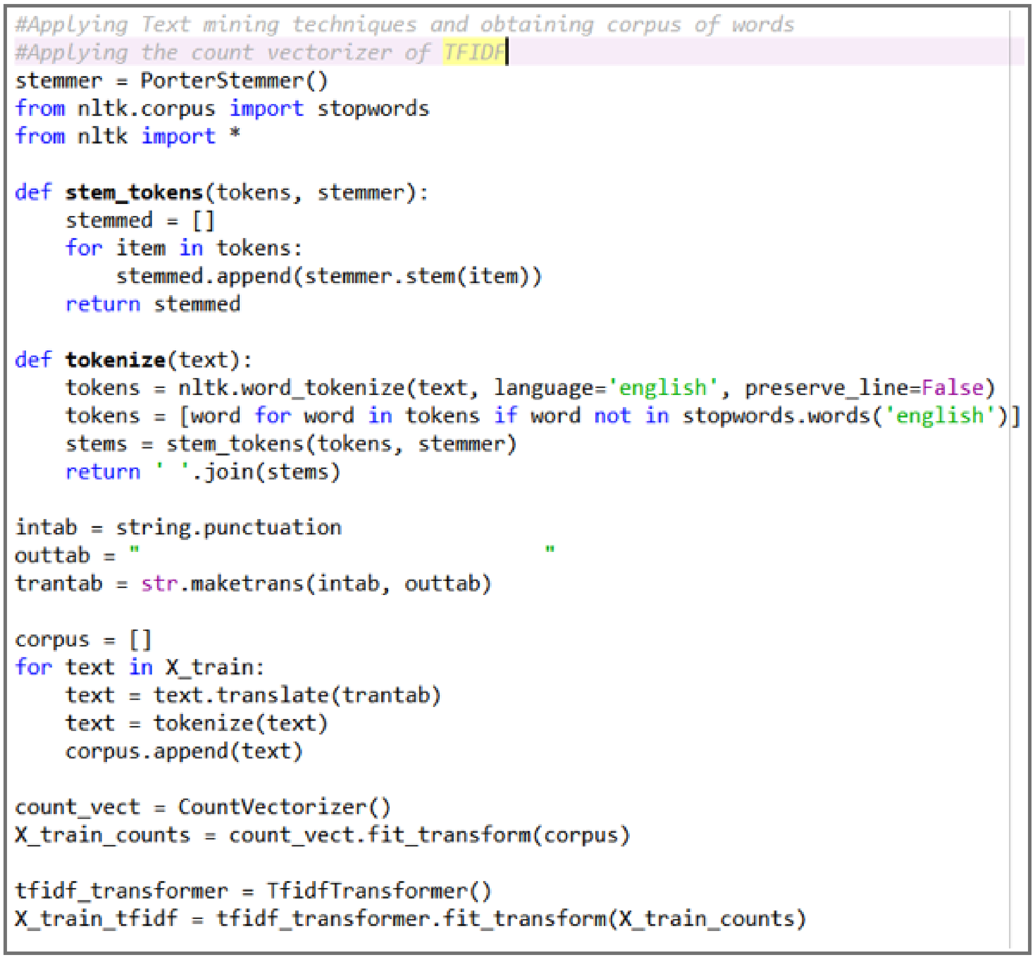
```
#Removing neutral reviews and sampling 18% of data
data_clean=data_clean[data_clean['Review'] != 'neutral']
data_clean = data_clean.sample(frac=0.18)
data_clean.info()
Score = data_clean['Score']
Score = Score.map(partition)
Summary = data_clean['Text']
```

Now our data is ready for splitting. Using train_test_split function in Python we divided the data to train and test in 70 :30 ratio. Following code snippet includes this functionality.

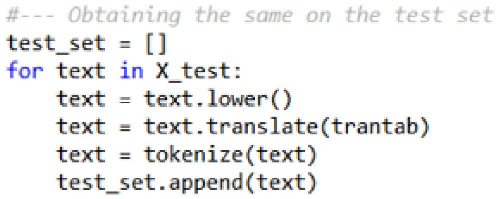


```
#Splitting the data into train and test
X_train, X_test, y_train, y_test = train_test_split(Summary, Score, test_size=0.3, random_state=52)
```

Next, we apply text mining techniques like tokenization, PorterStemmer to clear punctuations and stop words in the data and convert all the words into root words.

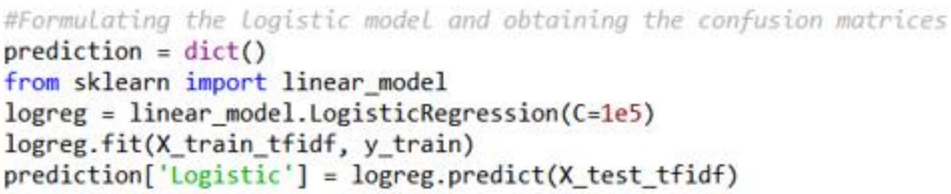


```
#Applying Text mining techniques and obtaining corpus of words
#Applying the count vectorizer of TFIDF
stemmer = PorterStemmer()
from nltk.corpus import stopwords
from nltk import *

def stem_tokens(tokens, stemmer):
    stemmed = []
    for item in tokens:
        stemmed.append(stemmer.stem(item))
    return stemmed

def tokenize(text):
    tokens = nltk.word_tokenize(text, language='english', preserve_line=False)
    tokens = [word for word in tokens if word not in stopwords.words('english')]
    stems = stem_tokens(tokens, stemmer)
    return ' '.join(stems)

intab = string.punctuation
outtab = "                                "
trantab = str.maketrans(intab, outtab)

corpus = []
for text in X_train:
    text = text.translate(trantab)
    text = tokenize(text)
    corpus.append(text)

count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(corpus)

tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
```



```
#--- Obtaining the same on the test set
test_set = []
for text in X_test:
    text = text.lower()
    text = text.translate(trantab)
    text = tokenize(text)
    test_set.append(text)
```

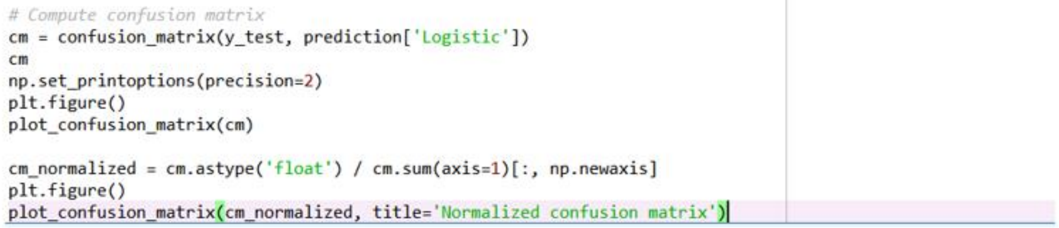
Since our dependent variable is a binary variable, we applied logistic regression to classify comments.



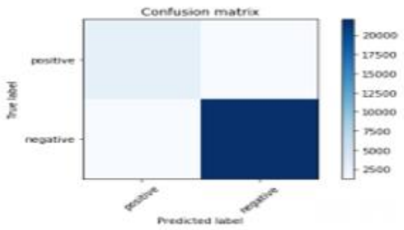
```
#Formulating the logistic model and obtaining the confusion matrices
prediction = dict()
from sklearn import linear_model
logreg = linear_model.LogisticRegression(C=1e5)
logreg.fit(X_train_tfidf, y_train)
prediction['Logistic'] = logreg.predict(X_test_tfidf)
```

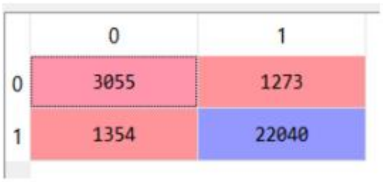
# Observations and Results:

Thus, we reached final output stage. We calculated Confusion matrix and determined True Positive Rate and True Negative Rate.

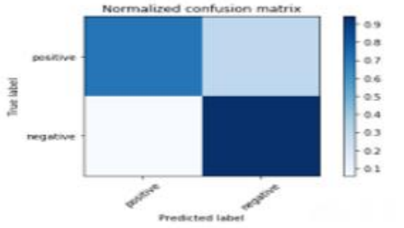


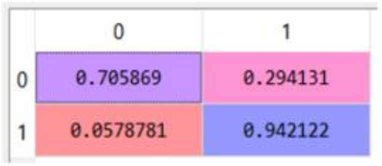
```
# Compute confusion matrix
cm = confusion_matrix(y_test, prediction['Logistic'])
cm
np.set_printoptions(precision=2)
plt.figure()
plot_confusion_matrix(cm)

cm_normalized = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
plt.figure()
plot_confusion_matrix(cm_normalized, title='Normalized confusion matrix')
```





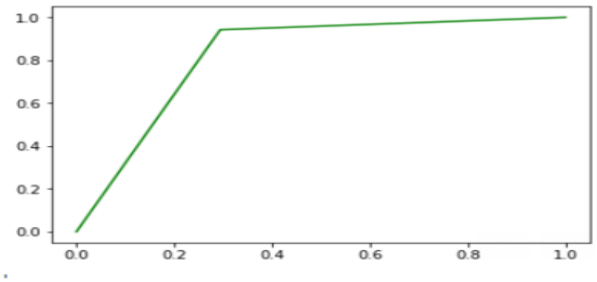
| | 0 | 1 |
|---|---|---|
| 0 | 3055 | 1273 |
| 1 | 1354 | 22040 |

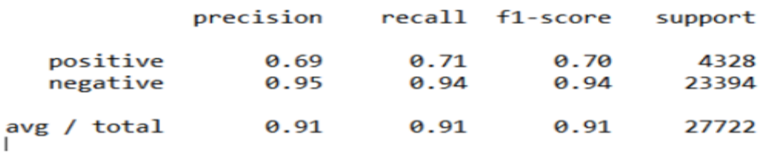




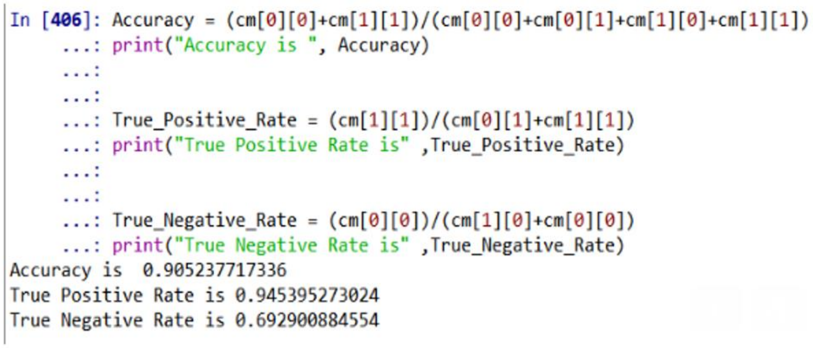
| | 0 | 1 |
|---|---|---|
| 0 | 0.705869 | 0.294131 |
| 1 | 0.0578781 | 0.942122 |

Our model worked well with an accuracy of more than 90%. The results are below.





```
             precision    recall  f1-score   support

   positive       0.69      0.71      0.70      4328
   negative       0.95      0.94      0.94     23394

avg / total       0.91      0.91      0.91     27722
```



```
In [406]: Accuracy = (cm[0][0]+cm[1][1])/(cm[0][0]+cm[0][1]+cm[1][0]+cm[1][1])
     ...: print("Accuracy is ", Accuracy)
     ...:
     ...:
     ...: True_Positive_Rate = (cm[1][1])/(cm[0][1]+cm[1][1])
     ...: print("True Positive Rate is" ,True_Positive_Rate)
     ...:
     ...:
     ...: True_Negative_Rate = (cm[0][0])/(cm[1][0]+cm[0][0])
     ...: print("True Negative Rate is" ,True_Negative_Rate)
Accuracy is  0.905237717336
True Positive Rate is 0.945395273024
True Negative Rate is 0.692900884554
```

We can see that the lift in the ROC curve is pretty good and this indicates that model classified comments with good accuracy.

# Conclusion:

- We classified the reviews into positive and negative with 90.40% accuracy.
- Using this model, you can build a system to reach out to the vendors with a strategic method to increase the business.
- Cluster of products having similar reviews would help in recommending other similar products to the users who are impressed before and will help detect products having negative reviews.
- Online Retail Industry can use this method to filter out comments and reviews with negative remarks to help use these feedbacks in improving services.
- Benefits of using sentiment analysis is that the prediction of sentiments can be done in a short time with the help of text mining on the data set.
- Other areas- politics, law, sociology, psychology, medicine and IT.

# References:

- https://www.mckinsey.com/business-functions/organization/our-insights/the-moment-of-truth-in-customer-service
- https://towardsdatascience.com/five-practical-use-cases-of-customer-sentiment-analysis-for-nps-a3167ac2caaa