Ratings Project

Submitted by:

ASHEEM SIWACH

ACKNOWLEDGMENT

I would like to thank Flip Robo Technologies for providing me with the opportunity to work on this project from which I have learned a lot. I am also grateful to Mr. Shubham Yadav for his constant guidance and support.

Reference sources are :-

- Google
- Stackoverflow.com
- Analytics vidhya
- Notes and repository from DataTrained

INTRODUCTION

• Business Problem Framing

We have a client who has a website where people write different reviews for technical products. Now they are adding a new feature to their website i.e. The reviewer will have to add stars(rating) as well with the review. The rating is out 5 stars and it only has 5 options available 1 star, 2 stars, 3 stars, 4 stars, 5 stars. Now they want to predict ratings for the reviews which were written in the past and they don't have a rating. So, we have to build an application which can predict the rating by seeing the review.

• Conceptual Background of the Domain Problem

The advent of electronic commerce with growth in internet and network technologies has led customers to move to online retail platforms such as Amazon, Walmart, Flip Kart, etc. People often rely on customer reviews of products before they buy online. These reviews are often rich in information describing the product. Customers often choose to compare between various products and brands based on whether an item has a positive or negative review. More often, these reviews act as a feedback mechanism for

the seller. Through this medium, sellers strategize their future sales and product improvement.

- ## Motivation for the Problem Undertaken

There are many websites/applications who are providing online shopping services. But as there is no direct contact between the seller and consumer it becomes somewhat difficult for the seller what customers want and what they should provide to their customers.

Every day we come across various products in our lives, on the digital medium we swipe across hundreds of product choices under one category. It will be tedious for the customer to make selection. Here comes 'reviews' where customers who have already got that product leave a rating after using them and brief their experience by giving reviews.

As we know ratings can be easily sorted and judged whether a product is good or bad. But when it comes to sentence reviews, we need to read through every line to make sure the review conveys a positive or negative sense. In the era of artificial intelligence, things like that have got easy with the Natural Language Processing (NLP) technique. Therefore, it is important to minimize the number of false positives our model produces, to encourage all constructive conversation.

Our model also provides beneficence for the platform hosts as it replaces the need to manually moderate discussions, saving time and resources. Employing a machine learning model to predict ratings promotes easier way to distinguish between products qualities, costs and many other features.

Analytical Problem Framing

- ## Data Collection

  For data collection we have used Selenium Web Scrapping tool. By using this tool we gathered almost 20000 rows data which included of smartphones, watches, monitors, refrigerators, headphones, microwaves etc.

Together we collected data and then combine all into a single dataframe using pandas dataframe.

[1]:
```
# Calling webdriver and going to the url
driver = webdriver.Chrome('chromedriver.exe')

driver.get('https://www.amazon.in/')
```

[3]:
```
# Using search bar for finding the desired data to scrape for smartphones
driver.find_element_by_xpath("//input[@id='twotabsearchtextbox']").send_keys('smartphones',Keys.ENTER)
```

[4]:
```
# Create empty list to store product urls and then append urls into it
product_url = []
for i in tqdm_notebook(range(0,2)):
    URLS = driver.find_elements_by_xpath("//div[@class='a-section a-spacing-none']/div/h2/a")
    for u in URLS:
        product_url.append(u.get_attribute("href"))

    # Going to next page for more urls
    try:driver.find_element_by_xpath("//*[@id='search']/div[1]/div/div[1]/div/span[3]/div[2]/div[28]/span/div/div/ul/li[7]/a
    except NoSuchElementException:
        pass
```

In [10]:
```
# Using forloop to scrape the data from each product url
for url in tqdm_notebook(product_url):
    driver.get(url)
    time.sleep(1.5)

    # going to ratings page
    try:driver.find_element_by_xpath("//*[@id='acrCustomerReviewLink']").click()
    except NoSuchElementException:
        print(cl("No ratings available",color='blue',attrs=['dark']))
        pass

    # Click to see all reviews of product
    try:driver.find_element_by_xpath('//*[@id="reviews-medley-footer"]/div[2]/a').click()
    except NoSuchElementException :
        print(cl("No more reviews",color='green',attrs=['dark']))
        pass

    time.sleep(1.5)
    for i in tqdm_notebook(range(0,30)):
        time.sleep(1)

        # Fetch ratings of product from each review
        try:
            RATINGS = driver.find_elements_by_xpath('//div[@class="a-section celwidget"]/div[2]/a[1]')
            for r in RATINGS:
                driver.implicitly_wait(2)
```

```
        # fetching reviews of product
        try:
            REVIEWS = driver.find_elements_by_xpath('//div[@class="a-section celwidget"]/div[4]')
            for v in REVIEWS:
                reviews.append(v.text.replace('\n',''))
        except NoSuchElementException:
            reviews.append("Not Available")

        #Printing the page scrapped
        print("Scraping page {} ".format(i+1))

        # Going to next page of reviews
        try:driver.find_element_by_xpath('//*[@id="cm_cr-pagination_bar"]/ul/li[2]/a').click()

        except NoSuchElementException:
            print("No more reviews")
            pass
```

- ## Data Preprocessing Done

Data Preprocessing is the major step before model building as without proper cleaning ,proper formatting and understanding the data we cannot do good analysis. So Data Preprocessing being the major step is done by following ways :

a. Importing the suitable libraires and then load the data
b. Checking null values
c. Dropping duplicated data

Importing the suitable libraires and then load the data:
First step in data preprocessing is importing the libraires which will be helpful in analysis.After loading we will check the shape ,information of dataset.

Importing Libraries

```
In [1]:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import string
import re
from nltk.corpus import stopwords
from nltk.tokenize import regexp_tokenize
from wordcloud import WordCloud

from nltk.stem import WordNetLemmatizer
import warnings
warnings.filterwarnings('ignore')

# packages from gensim
from gensim import corpora
from gensim.parsing.preprocessing import STOPWORDS

```

Loading the data

Loading dataset

```
In [2]:
# Loading smartphones dataset
smartphones = pd.read_csv(r"C:\Users\siwac\machine learning models\smartphones.csv")
smartphones.head(5)
```

Out[2]:

	Reviews	Ratings
0	Please be aware that this is a downgraded vers...	2
1	What a Smartphone by Xiaomi! I am thoroughly i...	5
2	Received my Mi 11x Pro today🥰 . Really excited...	4
3	I love gaming and have been waiting for a SDM ...	5
4	I am a oneplus 5 user for the last 4 years. So...	3

```
In [3]:
# Loading laptops dataset
laptops = pd.read_csv(r"C:\Users\siwac\machine learning models\laptops.csv")
laptops.head(5)
```

Out[3]:

	Reviews	Ratings
0	The laptop came in excellent package - the pac...	5
1	Its superbIts world's lightest laptopSuperb ba...	5

Checking null values :

As now we have loaded the data we will check the null values present in dataset. For checking the null values we will use heatmap.

```
In [8]:
# Concatenating all dataset together
df = pd.concat([smartphones,laptops,iphone,refrigerators,monitors,headphones],ignore_index=True)
df.shape
```

```
Out[8]: (39752, 2)
```

```
In [9]:
# Removing all duplicated data from dataset
df.drop_duplicates(inplace=True,ignore_index=True)
```

```
In [10]:
# Checking null values
df.isnull().sum()
```

```
Out[10]: Reviews    5
         Ratings    0
         dtype: int64
```

```
In [11]:
# Dropping null values
df.dropna(inplace=True)
```

```
In [12]:
# Checking data info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 18516 entries, 0 to 18520
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
```

So there are no null values in our dataset.

- ## Hardware and Software Requirements and Tools Used

  There is no such requirement for hardware ,but I have used intel i5 8th generation processor.

Software: Jupyter Notebook (Anaconda 3)

Language : Python 3.9

Libraries used in project:

a. Pandas
b. Gensim
c. Nltk
d. Seaborn
e. Sklearn
f. Re

- ## Data Cleaning

  - As we are working on text data so NLP technique_is used here to deal with the data. Using this technique we will clean data into simple steps.
  - STEP 1 : Converting textual data into lower case letters.
  - STEP 2:Removal of stopwords and punctuations
  - STEP 3: Converting text into tokens using a method regexp_tokenize
  - STEP 4 Lemmatization(converting words into their simpler forms)

Converting all upper case letters into lower case letters:

- We have used str.lower() method to convert the uppercase letters into lower caser letters.

```
In [19]: 1 # Lowering the upper case letters
         2
         3 reviews['Reviews'] = reviews['Reviews'].str.lower()

In [20]: 1 # Checking sample review
         2 reviews['Reviews'][0]

Out[20]: 'please be aware that this is a downgraded version of globally launched mi 11 and mi 11 pro.reason for low rating -1. ordinary
         display. this is not the 2k display as featured on global versions of mi 11 series2. no gorilla glass victus unlike global vers
         ion.3. band tests shows poor build quality which may result in breakage of phone due to bending4. such a cheap tactic of sellin
         g a phone with the promise of fast charging and not providing charger with a adequate charging capacity.5. no curved display un
         like global version.6. took almost 20 days to get my hands on the phone post order.7. reported heating issues in sd 888 while p
         laying intensive gamesin short, xiaomi has attempted to sell big story by first launching mi 11 globally and then putting "x" a
         nd selling subpar product in india through illusion.i guess, xiaomi should not have cut the corners and compromise the quality
         in indian versions. indians are willing to pay premium and success of oneplus is the testament to that. any cunning move to und
         ercut the value proposition will hurt xiaomi in long run.(note that i currently use redmi note 20 pro and have used so many pho
         nes of xiaomi ever since it launched first ever phone in india and hence hold a adequate credentials to voice my criticism as a
```

Removal of stopwords and then converting sentences into tokens

- After applying the lower case method we defined a function which will remove stopwords and convert the sentence data into tokens so that unnecessary text can be removed easily. Using the regexp_tokenize() method we simply keep the data from a-z case letters and remove all numbers.

In [21]:
```
# Removal of stopwords and unnecessary words like punctuations, signs, numbers
def stop_words(text):
    stp_wrds=set(STOPWORDS).union(set(string.punctuation))
    tokens=regexp_tokenize(text,"[a-z']+")
    return [t for t in tokens if t not in stp_wrds]
```

In [22]:
```
# Apply the above function to the columns reviews
reviews['Reviews']=reviews["Reviews"].apply(stop_words)
```

In [23]:
```
reviews
```

Out[23]:

	Reviews	Ratings
0	[aware, downgraded, version, globally, launche...	2.0
1	[smartphone, xiaomi, thoroughly, impressed, pl...	5.0
2	[received, mi, x, pro, today, excited, came, p...	4.0
3	[love, gaming, waiting, sdm, k, compromise, as...	5.0
4	[oneplus, user, years, compare, oneplus, mi, x...	3.0

Lemmatizating the tokens (convert them into simpler forms)

- Lemmatization technique is used to convert the word into its simpler form like from plural to singular, from 3rd for of verb to 1st form of verb. We applied all the techniques and append the result into list which will be stored into a new column.

In [25]:
```
# After stopwords removal now we are going to lemmatze words

result = [] # empty list to store data
lemma = WordNetLemmatizer()

# function for lemmatization
def lemmatize(text):
    word1=lemma.lemmatize(text,pos='n')
    word2=lemma.lemmatize(word1,pos='v')
    return lemma.lemmatize(word2,pos=("a"))

# function for filtering and apply above function to the column
def preprocess(text):
    result=[]
    for token in text:
        if len(token)>=3:
            result.append(lemmatize(token))

    return result
```

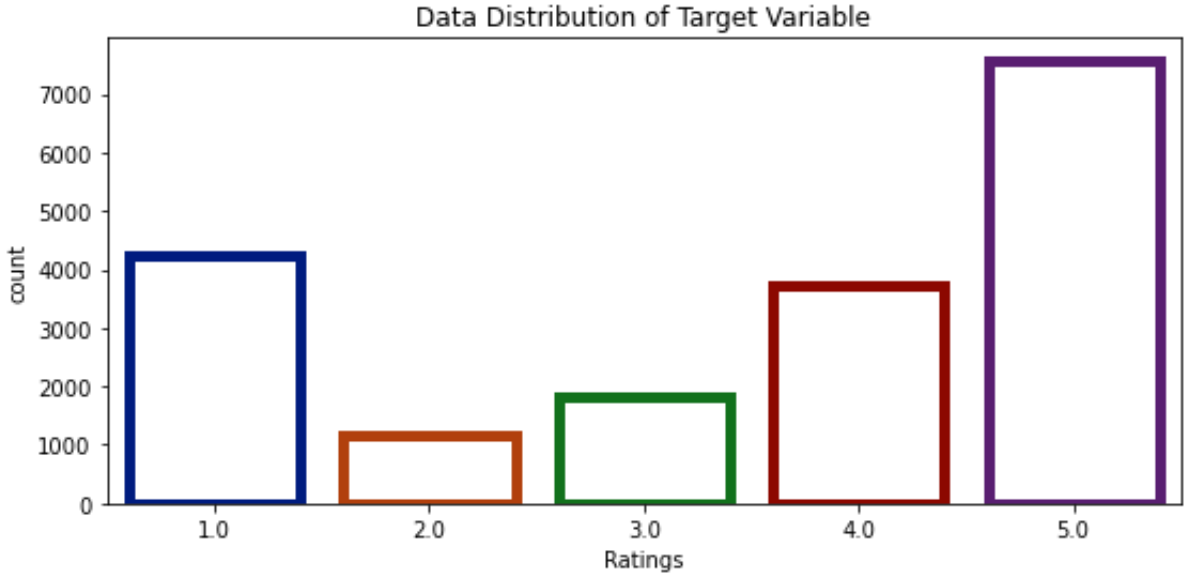
Cleaned text

- After the completion of all the steps we are left with clean data. Now this data can be used for model building after converting textual data into vector form.

```
In [27]:
# Assigning new column to the processed reviews
reviews['cleaned_reviews'] = processed_review

In [28]:
# Changing uncleaned reviews into cleaned reviews
reviews['Reviews'] = reviews['cleaned_reviews'].apply(lambda x:' '.join(y for y in x))

In [29]:
# checking sample of cleaned review
reviews['Reviews'][0]

Out[29]: 'aware downgrade version globally launch pro reason low rat ordinary display display feature global version series gorilla glas
s victus unlike global version band test show poor build quality result breakage phone bend cheap tactic sell phone promise fas
t charge provide charger adequate charge capacity curve display unlike global version take day hand phone post order report hea
t issue play intensive gamesin short xiaomi attempt sell big story launch globally put sell subpar product india illusion guess
xiaomi cut corner compromise quality indian version indian will pay premium success oneplus testament cunning undercut value pr
oposition hurt xiaomi long run note currently use redmi note pro phone xiaomi launch phone india hold adequate credential voice
criticism fan'
```

• Exploratory Data Analysis

```
In [16]:
# Let's check total count of each rating
reviews['Ratings'].value_counts()

Out[16]: 5.0    7582
1.0    4251
4.0    3719
3.0    1816
2.0    1148
Name: Ratings, dtype: int64

In [17]:
# Let's check data distribution of target variable
plt.figure(figsize=(9,4),frameon=False)
sns.countplot('Ratings',data=reviews,color="salmon", facecolor=(0, 0, 0, 0),
                    linewidth=5,
                    edgecolor=sns.color_palette("dark", 5))
plt.title("Data Distribution of Target Variable")
plt.xlabel("Ratings")
plt.show()
```



Wordcloud to see the words which are most repeated in 5star ratings:

```
# Using wordcloud see the texts which are most repeated in 5 star ratings
star_5 = reviews['Reviews'][reviews['Ratings']==5.0]
combined_5star=' '.join(star_5)
plt.figure(figsize=(18,12),facecolor='grey')
wordcloud = WordCloud(max_font_size=50,max_words=250,background_color='white').generate(combined_5star)

# Display the generated image:
plt.imshow(wordcloud, interpolation='lanczos')
plt.axis("off")
plt.show()
```

Wordcloud to see the words which are most repeated in 4star ratings:

```
# Using wordcloud see the texts which are most repeated in 4 star ratings
star_4 = reviews['Reviews'][reviews['Ratings']==4.0]
combined_4star=' '.join(star_4)
plt.figure(figsize=(18,12),facecolor='grey')
wordcloud = WordCloud(max_font_size=50,max_words=250,background_color='white').generate(combined_4star)

# Display the generated image:
plt.imshow(wordcloud, interpolation='lanczos')
plt.axis("off")
plt.show()
```

Wordcloud to see the words which are most repeated in 3star ratings:

```
# Using wordcloud see the texts which are most repeated in 3 star ratings
star_3 = reviews['Reviews'][reviews['Ratings']==3.0]
combined_3star=' '.join(star_3)
plt.figure(figsize=(18,12),facecolor='grey')
wordcloud = WordCloud(max_font_size=50,max_words=250,background_color='white').generate(combined_3star)

# Display the generated image:
plt.imshow(wordcloud, interpolation='lanczos')
plt.axis("off")
plt.show()
```

Wordcloud to see the words which are most repeated in 2star ratings:

```
# Using wordcloud see the texts which are most repeated in 2 star ratings
star_2 = reviews['Reviews'][reviews['Ratings']==2.0]
combined_2star=' '.join(star_2)
plt.figure(figsize=(18,12),facecolor='grey')
wordcloud = WordCloud(max_font_size=50,max_words=250,background_color='white').generate(combined_2star)

# Display the generated image:
plt.imshow(wordcloud, interpolation='lanczos')
plt.axis("off")
plt.show()
```

Wordcloud to see the words which are most repeated in 1star ratings:

```
# Using wordcloud see the texts which are most repeated in 1 star ratings
star_1 = reviews['Reviews'][reviews['Ratings']==1]
combined_1star=' '.join(star_1)
plt.figure(figsize=(18,12),facecolor='grey')
wordcloud = WordCloud(max_font_size=50,max_words=250,background_color='white').generate(combined_1star)

# Display the generated image:
plt.imshow(wordcloud, interpolation='lanczos')
plt.axis("off")
plt.show()
```

Observation :

There is not much distinction in wordcloud of different ratings because

- our dataset is imbalanced.
- Proper reviews are not present.

• Model Building :

- Convert text into vectors using Tf-Idf Vectorizer
- Using feature selection method selecting the features suitable for model building
- Spliting dataset into training and testing

Tf-Idf Vectorizer:

- While using the tf-idf Vectorizer technique in parameters we are giving only 750 features as it will create thousands of non-valuable features.
- After converting into vectors we will using feature selection method to selected the most desired features necessary for best result from the algorithm.

In [38]:
```
# copying dataset
df_model = reviews.copy()
```

In [39]:
```
# Importing library to vectorize the data for model building
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [40]:
```
# Applying function for vectorizing
tf_idf1 = TfidfVectorizer(max_features=750,norm='l2')

```

In [41]:
```
# fit and transform the dataset
features = tf_idf1.fit_transform(df_model['Reviews'])
features
```

Out[41]:
```
<18516x750 sparse matrix of type '<class 'numpy.float64'>'
        with 323487 stored elements in Compressed Sparse Row format>
```

In [42]:
```
# storing in dataframe
vectorized_features= pd.DataFrame(features.todense(),columns=tf_idf1.get_feature_names())
vectorized_features
```

Out[42]:

	able	absolutely	accord	accurate	acer	actually	adapter	add	adjust	adjustable	...	worry	worth	write	wrong	xiaomi	year	yes	you	youth
0	0.0	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.389802	0.000000	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.173649	0.000000	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.21108	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.121831	0.092367	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.000000	0.209272	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
18511	0.0	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	
18512	0.0	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	
18513	0.0	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	
18514	0.0	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	
18515	0.0	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	

18516 rows × 750 columns

Feature selection :

In [47]:
```
# Using feature selection for better model building
selection = SelectFromModel(LogisticRegression())
selection.fit(vectorized_features,y)
```

Out[47]:
```
SelectFromModel(estimator=LogisticRegression())
```

In [48]:
```
# finding valuable features
feature_name = vectorized_features.columns[selection.get_support()]
feature_name
```

Out[48]:
```
Index(['absolutely', 'adapter', 'add', 'amaze', 'amazon', 'amd', 'amoled',
       'anc', 'android', 'angle',
       ...
       'weight', 'wide', 'wifi', 'wire', 'wireless', 'wise', 'worry', 'wrong',
       'xiaomi', 'yes'],
      dtype='object', length=297)
```

In [49]:
```
# Apply transformation on train independent dataset.

selection_New_x = selection.transform(vectorized_features)
New_x = pd.DataFrame(selection_New_x,columns=feature_name)
New_x
```

Out[49]:

	absolutely	adapter	add	amaze	amazon	amd	amoled	anc	android	angle	...	weight	wide	wifi	wire	wireless	wise	worry	wrong	x
0	0.0	0.00000	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	0.0	...	0.000000	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.3
1	0.0	0.00000	0.0	0.120381	0.0	0.0	0.162607	0.0	0.0	0.0	...	0.000000	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.1
2	0.0	0.21108	0.0	0.000000	0.0	0.0	0.195027	0.0	0.0	0.0	...	0.000000	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0
3	0.0	0.00000	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	0.0	...	0.000000	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.1
4	0.0	0.00000	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	0.0	...	0.000000	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
18511	0.0	0.00000	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	0.0	...	0.000000	0.0	0.0	0.000000	0.157517	0.0	0.0	0.0	0.0
18512	0.0	0.00000	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	0.0	...	0.000000	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0
18513	0.0	0.00000	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	0.0	...	0.000000	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0
18514	0.0	0.00000	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	0.0	...	0.099687	0.0	0.0	0.308569	0.115878	0.0	0.0	0.0	0.0
18515	0.0	0.00000	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	0.0	...	0.000000	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0

18516 rows × 297 columns

Spliting the dataset into training and testing

In [50]:
```
# Let's now find the best random state
max_f1_score=0
for i in range(1,100):
    x_train,x_test,y_train,y_test=train_test_split(New_x,y,test_size=.22,random_state=i)
    mnb=MultinomialNB()
    mnb.fit(x_train,y_train)
    y_pred=mnb.predict(x_test)
    f1=f1_score(y_test,y_pred, average='weighted')

    if f1>max_f1_score:
        max_f1_score=f1
        final_random_state=i
print("Max f1 score corresponding to ",final_random_state,"is ",max_f1_score,".")
```

```
Max f1 score corresponding to  89 is  0.4289328029730464 .
```

In [51]:
```
# Let's split the train and test dataset
x_train,x_test,y_train,y_test = train_test_split(New_x,y,test_size=0.2,random_state=final_random_state)
print("Train shapes : X = {}, y = {}".format(x_train.shape,y_train.shape))
print("Test shapes : X = {}, y = {}".format(x_test.shape,y_test.shape))
# Success
print("\nTraining and testing split was successful.")
```

```
Train shapes : X = (14812, 297), y = (14812,)
Test shapes : X = (3704, 297), y = (3704,)

Training and testing split was successful.
```

- Model Evaluation

In [52]:
```
# function for finding model's accuracy and other metrics

def metrics_score(model):
    print(model)
    model.fit(x_train,y_train)
    print("\nTraining ",round(model.score(x_train,y_train)*100,1),"%")
    y_pred=model.predict(x_test)
    print("\nAccuracy score - ",round(accuracy_score(y_test,y_pred)*100,1),"%")
    print("\nClassification Report - \n{}".format(classification_report(y_test,y_pred)))

    print("\n\nCross Validation Score - ")
    for i in range(2,10):
        cross_val=cross_val_score(model,New_x,y,cv=i,n_jobs=1)    # cross validation

        print("Coss validation score at ",i,"is",round(cross_val.mean()*100,1),"%")
        print(round(cross_val.std()*100,1))
```

In [53]:
```
metrics_score(MultinomialNB(alpha=1.0,fit_prior=True))
```

```
MultinomialNB()

Training  52.3 %

Accuracy score -  53.7 %

Classification Report -
              precision    recall  f1-score   support

         1.0       0.64      0.58      0.61       893
         2.0       0.00      0.00      0.00       200
         3.0       1.00      0.00      0.01       356
         4.0       0.27      0.02      0.03       712
         5.0       0.51      0.95      0.66      1543

    accuracy                           0.54      3704
   macro avg       0.48      0.31      0.26      3704
weighted avg       0.52      0.54      0.43      3704
```

```
GaussianNB()

Training  45.7 %

Accuracy score -  43.3 %

Classification Report -
              precision    recall  f1-score   support

         1.0       0.48      0.58      0.52       893
         2.0       0.12      0.26      0.17       200
         3.0       0.18      0.26      0.21       356
         4.0       0.28      0.16      0.21       712
         5.0       0.64      0.54      0.59      1543

    accuracy                           0.43      3704
   macro avg       0.34      0.36      0.34      3704
weighted avg       0.46      0.43      0.44      3704


Cross Validation Score -
Coss validation score at  2 is 22.8 %
0.4
Coss validation score at  3 is 32.7 %
4.9
Coss validation score at  4 is 34.2 %

BernoulliNB()

Training  52.2 %

Accuracy score -  51.5 %

Classification Report -
              precision    recall  f1-score   support

         1.0       0.52      0.71      0.60       893
         2.0       0.22      0.11      0.15       200
         3.0       0.36      0.15      0.21       356
         4.0       0.30      0.20      0.24       712
         5.0       0.60      0.69      0.64      1543

    accuracy                           0.51      3704
   macro avg       0.40      0.37      0.37      3704
weighted avg       0.48      0.51      0.48      3704


Cross Validation Score -
Coss validation score at  2 is 47.1 %
4.5
Coss validation score at  3 is 47.3 %
4.7
Coss validation score at  4 is 48.7 %
6.1

LogisticRegression()

Training  55.3 %

Accuracy score -  55.0 %

Classification Report -
              precision    recall  f1-score   support

         1.0       0.60      0.65      0.62       893
         2.0       0.08      0.01      0.01       200
         3.0       0.37      0.07      0.12       356
         4.0       0.37      0.15      0.21       712
         5.0       0.56      0.86      0.68      1543

    accuracy                           0.55      3704
   macro avg       0.39      0.35      0.33      3704
weighted avg       0.49      0.55      0.49      3704


Cross Validation Score -
Coss validation score at  2 is 51.5 %
3.3
Coss validation score at  3 is 52.8 %
4.0
Coss validation score at  4 is 52.5 %
```

Hyperparameter Tunning :

In [59]:
```
# Defining parameters
solvers = ['newton-cg', 'lbfgs', 'liblinear']
penalty = ['l2']
c_values = [100, 10, 1.0, 0.1, 0.01]
```

In [60]:
```
from sklearn.model_selection import RepeatedStratifiedKFold, GridSearchCV

# define grid search
grid = dict(solver=solvers,penalty=penalty,C=c_values)
# Cross validation technique
cv_method = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)

grid_search = GridSearchCV(estimator=LogisticRegression(), param_grid=grid, n_jobs=-1, cv=cv_method, scoring='accuracy',erro
grid_result = grid_search.fit(New_x, y)
```

In [61]:
```
# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

```
Best: 0.538508 using {'C': 1.0, 'penalty': 'l2', 'solver': 'liblinear'}
0.534187 (0.008788) with: {'C': 100, 'penalty': 'l2', 'solver': 'newton-cg'}
0.533359 (0.009112) with: {'C': 100, 'penalty': 'l2', 'solver': 'lbfgs'}
0.536924 (0.008259) with: {'C': 100, 'penalty': 'l2', 'solver': 'liblinear'}
0.535231 (0.008989) with: {'C': 10, 'penalty': 'l2', 'solver': 'newton-cg'}
0.535249 (0.009186) with: {'C': 10, 'penalty': 'l2', 'solver': 'lbfgs'}
0.537428 (0.008240) with: {'C': 10, 'penalty': 'l2', 'solver': 'liblinear'}
0.537770 (0.008779) with: {'C': 1.0, 'penalty': 'l2', 'solver': 'newton-cg'}
0.537950 (0.008973) with: {'C': 1.0, 'penalty': 'l2', 'solver': 'lbfgs'}
0.538508 (0.007772) with: {'C': 1.0, 'penalty': 'l2', 'solver': 'liblinear'}
0.528174 (0.006082) with: {'C': 0.1, 'penalty': 'l2', 'solver': 'newton-cg'}
0.528210 (0.006023) with: {'C': 0.1, 'penalty': 'l2', 'solver': 'lbfgs'}
0.524016 (0.005959) with: {'C': 0.1, 'penalty': 'l2', 'solver': 'liblinear'}
0.462465 (0.004260) with: {'C': 0.01, 'penalty': 'l2', 'solver': 'newton-cg'}
0.462465 (0.004260) with: {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
0.442176 (0.003898) with: {'C': 0.01, 'penalty': 'l2', 'solver': 'liblinear'}
```

```
# Using the best parameters obtained
model = LogisticRegression(C= 1.0, penalty= 'l2', solver='liblinear')
model.fit(x_train,y_train)
y_pred = model.predict(x_test)
print("Accuracy score = ",round(accuracy_score(y_test,y_pred)*100,1),"%")
print('Cross validation score: ',cross_val_score(model,New_x,y,cv=3,scoring='accuracy').mean()*100)
print("Confusion matrix\n",confusion_matrix(y_test,y_pred))
print("Classification report\n",classification_report(y_test,y_pred))
```

```
Accuracy score =  54.8 %
Cross validation score:  52.91099589544178
Confusion matrix
 [[ 572    4    5   23  289]
 [ 100    1   14   21   64]
 [ 101    0   19   45  191]
 [  73    0    6   86  547]
 [ 107    0    7   79 1350]]
Classification report
              precision    recall  f1-score   support

         1.0       0.60      0.64      0.62       893
         2.0       0.20      0.01      0.01       200
         3.0       0.37      0.05      0.09       356
         4.0       0.34      0.12      0.18       712
         5.0       0.55      0.87      0.68      1543

    accuracy                           0.55      3704
   macro avg       0.41      0.34      0.32      3704
weighted avg       0.49      0.55      0.48      3704
```

CONCLUSION

- ## Key Findings and Conclusions of the Study

Working on this project was a wonderful experience as we have to collect data of reviews and ratings of customers who are using online retailer websites for shopping. After scrapping the data , we cleaned it by removing the stopwords, punctuations ,numbers and other unnecessary words that were not useful for our model building. While cleaning the dataset we find out that there are a lot of noise present in customer reviews.

Major problem we faced was in data cleaning because customers are not using proper language format or using other languages in reviews which

creates a major problem in identifying the real words and this leads to create more features.

* We have collected a lot of data from e-commerce websites. Although for better model building it should be of different websites but due to the size and convenience i have used data from Amazon only.
* After collecting all the data we combine it together into a dataframe and find out that duplicated data is present.
* Then after using Gensim stopwords library we removed the unnecessary words.
* After removal of stopwords we tokenize the sentence into words.
* Then using the Lemmatization technique we converted words into thier simpler form.
* After lemmatization we converted the textual data into vector form for model building.
* Now we splited the dataset into training and testing dataset for model building.
* Tried a lot of algorithms on dataset but `Bernoulli Naive Bayes`, `Multibnomail Naive Bayes` and `Logistic Regression` gives the better result.
* We finalised the `Logistic Regression` model and then do hyperparmater tunning by `GridSearchCV` for findig best result out of it

- ## Suggestions:

a. Less time complexity

b. More accurate reviews can be given

c. Less errors can be avoided.

Thankyou