RATINGS PROJECT

SUBMITTED BY

Prerna Sharma

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.

Some of the reference sources are as follows:

- Internet
- Coding Ninjas
- Medium.com

- Analytics Vidhya
- StackOverflow

INTRODUCTION BUSINESS PROBLEM FRAMING

- This is a Machine Learning Project performed on customer reviews. Reviews are processed using common NLP techniques.

- Millions of people use **Amazon and Flipkart** to buy products. For every product, people can rate and write a review. If a product is good, it gets a positive review and gets a higher star rating, similarly, if a product is bad, it gets a negative review and lower star rating. My aim in this project is to predict star rating automatically based on the product review.

- The range of star rating is 1 to 5. That means if the product review is negative, then it will get low star rating (possibly 1 or 2), if the product is average then it will get medium star rating (possibly 3), and if the product is good, then it will get higher star rating (possibly 4 or 5).

- This task is similar to Sentiment Analysis, but instead of predicting the positive and negative sentiment (sometimes neutral also), here I need to predict the star rating.

AIM OF THIS PROJECT

Our goal is to make a system that automatically detects the star rating based on 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.

- There is a client who has a website where people write different reviews for technical products. Now they want to add 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.

REVIEW OF LITERATURE

- This project is more about exploration, feature engineering and classification that can be done on this data. Since we scrape huge amount of data that includes five stars rating, we can do better data exploration and derive some interesting features using the available columns.

- We can categorize the ratings as:

1.0, 2.0, 3.0, 4.0 and 5.0 stars

- The goal of this project is to build an application which can predict the rating by seeing the review. In the long term, this would allow people to better explain and reviewing their purchase with each other in this increasingly digital world.

MOTIVATION OF THE PROBLEM UNDERTAKEN

- 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) technology. 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

MATHEMATICAL/ANALYTICAL MODELLING OF THE PROBLEM

- In our scrapped dataset, our target variable "**Rating** " is a **categorical** variable i.e., it can be classified as ‘1.0’, ‘2.0’,’3.0’,’4.0’,’5.0’. Therefore, we will be handling this modelling problem as classification.

- This project is done in two parts:

Data Collection Phase:

- You have to scrape at least 20000 rows of data. You can scrape more data as well, it’s up to you. More the data better the model.

- In this section you need to scrape the reviews of different laptops, Phones, Headphones, smart watches, Professional Cameras, Printers, monitors, home theatre, router from different e-commerce websites.

- Basically, we need these columns-

1) reviews of the product.
2) rating of the product.

- Fetch an equal number of reviews for each rating, for example if you are fetching 10000 reviews then all ratings 1,2,3,4,5 should be 2000. It will balance our data set.

- Convert all the ratings to their round number, as there are only 5 options for rating i.e., 1,2,3,4,5. If a rating is 4.5 convert it 5.

Model Building Phase:

After collecting the data, you need to build a machine learning model. Before model building do all data pre-processing steps involving NLP. Try different models with different hyper parameters and select the best model. Follow the complete life cycle of data science. Include all the steps like-

1. Data Cleaning
2. Exploratory Data Analysis
3. Data Pre-processing
4. Model Building
5. Model Evaluation
6. Selecting the best model

DATA SOURCES AND THEIR FORMATS

In this phase, we scraped nearly 36000 of reviews data from Amazon of different products like laptop,phone and camera etc. and it is collected by using Webscraping and Selenium.

Loading Dataset

In [3]:
```
import pandas as pd
df=pd.read_csv('Ratings.csv')
df  #Checking the dataset
```

Out[3]:

	Unnamed: 0	Product_Review	Ratings
0	0	Worth to buy if your are comfortable with one ...	5.0
1	1	Cons-Worst call quality voice sounds robotic a...	2.0
2	2	Nice pair of True wireless stereo earphones fr...	5.0
3	3	These are my first TWS earbuds. Before this I ...	4.0
4	4	Honestly to say this was my first buds and it ...	1.0
...	...	...	...
36395	36395	I purchased it for my Mother, Decent product i...	4.0
36396	36396	Battery is getting drained out quite fast. 7% ...	1.0
36397	36397	Not as good as redmi 8a, no type C , no fast c...	3.0
36398	36398	Worst phone.. overall performance is just bakw...	5.0
36399	36399	COVID 19 drastically changed everything. Looks...	3.0

36400 rows × 3 columns

- In the end, we combined all the data frames into a single data frame and it looks like as follows:

- Then, we will save this data in a csv file, so that we can do the pre-processing and model building.

DATA PRE-PROCESSING

- Handling missing data using fillna and checking the datatypes

Data pre-processing

In [7]:
```
#Checking for null values
df.isnull().sum()
```

Out[7]:
```
Product_Review    80
Ratings            0
dtype: int64
```

In [8]:
```
#We can handle missing data by filling them with 'No Review' using fillna()
df['Product_Review'].fillna('No review',inplace=True)
```

In [9]:
```
df.isnull().sum()  #Checking after filling them
```

Out[9]:
```
Product_Review    0
Ratings           0
dtype: int64
```

In [10]:
```
df.info()   #Checking the datatype of all the columns present
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36400 entries, 0 to 36399
Data columns (total 2 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Product_Review  36400 non-null  object 
 1   Ratings         36400 non-null  float64
dtypes: float64(1), object(1)
memory usage: 568.9+ KB
```

- Checking average rating and value counts of each rating present

In [11]:
```
#Checking the average rating given by the users
avg = df['Ratings'].mean()
Avg = round(avg,1)
print("Average rating given by users is " + str(Avg))
```

```
Average rating given by users is 3.3
```

In [12]:
```
#Checking the value counts of the rating
df['Ratings'].value_counts()
```

Out[12]:
```
5.0    13865
1.0    11115
4.0     6200
3.0     3040
2.0     2180
Name: Ratings, dtype: int64
```

Pre-processing using Natural Language Processing (NLP):

- We cleaned the data using regex, matching patterns in the comments and replacing them with more organized counterparts. Cleaner data leads to a more efficient model and higher accuracy. Following steps are involved:

1. Removing Punctuations and other special characters
2. Splitting the comments into individual words
3. Removing Stop Words

- There is a corpus of stop-words, that are high-frequency words such as "the", "to" and "also", and that we sometimes want to litter out of a document before further processing. Stop-words usually have little lexical content, don’t alter the general meaning of a sentence and their presence in a text fails to distinguish it from other texts. We used the one from Natural Language Toolkit a leading platform for building Python programs to work with human language.

- The code is attached below:

```
n [21]: def clean_text(df, df_column_name):

            #Converting all messages to lowercase
            df[df_column_name] = df[df_column_name].str.lower()

            #Replace email addresses with 'email'
            df[df_column_name] = df[df_column_name].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$','emailaddress')

            #Replace URLs with 'webaddress'
            df[df_column_name] = df[df_column_name].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$','webaddress')

            #Replace money symbols with 'dollars' (£ can by typed with ALT key + 156)
            df[df_column_name] = df[df_column_name].str.replace(r'£|\$', 'dollars')

            #Replace 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phonenumber'
            df[df_column_name] = df[df_column_name].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$','phonenumber')

            #Replace numbers with 'numbr'
            df[df_column_name] = df[df_column_name].str.replace(r'\d+(\.\d+)?', 'numbr')

            #Remove punctuation
            df[df_column_name] = df[df_column_name].str.replace(r'[^\w\d\s]', ' ')

            #Replace whitespace between terms with a single space
            df[df_column_name] = df[df_column_name].str.replace(r'\s+', ' ')

            #Remove leading and trailing whitespace
            df[df_column_name] = df[df_column_name].str.replace(r'^\s+|\s+?$', '')

            #Remove stopwords
            stop_words = set(stopwords.words('english') + ['u', 'ü', 'â', 'ur', '4', '2', 'im', 'dont', 'doin', 'ure'])
            df[df_column_name] = df[df_column_name].apply(lambda x: ' '.join(term for term in x.split() if term not in stop_words))

n [22]: #Calling the class
        clean_text(df, 'Product_Review')
        df['Product Review'].tail(3)
```

```
In [25]: # Lemmatizing and then Stemming with Snowball to get root words and further reducing characters
stemmer = SnowballStemmer("english")
import gensim
def lemmatize_stemming(text):
    return stemmer.stem(WordNetLemmatizer().lemmatize(text,pos='v'))

#Tokenize and Lemmatize
def preprocess(text):
    result=[]
    for token in text:
        if len(token)>=3:
            result.append(lemmatize_stemming(token))

    return result
```

```
In [27]: import nltk
nltk.download('wordnet')

[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\stead\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\wordnet.zip.

Out[27]: True
```

```
In [28]: #Processing review with above Function
processed_review = []

for doc in df.Product_Review:
    processed_review.append(preprocess(doc))

print(len(processed_review))
processed_review[:3]

36400
```

- Tokenizing the data using RegexpTokenizer

```
In [22]: #Calling the class
clean_text(df, 'Product_Review')
df['Product_Review'].tail(3)

Out[22]: 36397    good redmi numbra type c fast charge sound low...
36398    worst phone overall performance bakwaz buy alw...
36399    covid numbr drastically changed everything loo...
Name: Product_Review, dtype: object
```

```
In [23]: #Tokenizing the data using RegexpTokenizer
from nltk.tokenize import RegexpTokenizer
tokenizer=RegexpTokenizer(r'\w+')
df['Product_Review'] = df['Product_Review'].apply(lambda x: tokenizer.tokenize(x.lower()))
df.head()

Out[23]:
```

	Product_Review	Ratings
0	[worth, buy, comfortable, one, kidney]	5.0
1	[cons, worst, call, quality, voice, sounds, ro...	2.0
2	[nice, pair, true, wireless, stereo, earphones...	5.0
3	[first, tws, earbuds, using, oneplus, bullets,...	4.0
4	[honestly, say, first, buds, comes, surprise, ...	1.0

Stemming and Lemmatizing:

• **Stemming** is the process of converting inflected/derived words to their word stem or the root form. Basically, a large number of similar origin words are converted to the same word. E.g., words like "stems", "stemmer", "stemming", "stemmed" as based on "stem". This helps in achieving the training process with a better accuracy.

```
In [25]: # Lemmatizing and then Stemming with Snowball to get root words and further reducing characters
stemmer = SnowballStemmer("english")
import gensim
def lemmatize_stemming(text):
    return stemmer.stem(WordNetLemmatizer().lemmatize(text,pos='v'))

#Tokenize and Lemmatize
def preprocess(text):
    result=[]
    for token in text:
        if len(token)>=3:
            result.append(lemmatize_stemming(token))

    return result
```

```
In [27]: import nltk
nltk.download('wordnet')

[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\stead\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\wordnet.zip.

Out[27]: True
```

```
In [28]: #Processing review with above Function
processed_review = []

for doc in df.Product_Review:
    processed_review.append(preprocess(doc))

print(len(processed_review))
processed_review[:3]

36400
```

• **Lemmatizing** is the process of grouping together the inflected forms of a word so they can be analysed as a single item. This is quite similar to stemming in its working but differs since it depends on correctly identifying the intended part of speech and meaning of a word in a sentence, as well as within the larger context surrounding that sentence, such as neighbouring sentences or even an entire document.

• The **wordnet library in nltk** will be used for this purpose. Stemmer and Lemmatizer are also imported from nltk.

• Processing the review and assigning the updated review in the data frame

```
In [27]: import nltk
         nltk.download('wordnet')

[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\stead\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\wordnet.zip.

Out[27]: True
```

```
In [28]: #Processing review with above Function
         processed_review = []

         for doc in df.Product_Review:
             processed_review.append(preprocess(doc))

         print(len(processed_review))
         processed_review[:3]
```

df['Product_Review'] = df['clean_review'].apply(lambda x:' '.join(y for y in x))

```
In [29]: df['clean_review']=processed_review  #Assigning this to the dataframe
         df.head()
```

Out[29]:

	Product_Review	Ratings	clean_review
0	[worth, buy, comfortable, one, kidney]	5.0	[worth, buy, comfort, one, kidney]
1	[cons, worst, call, quality, voice, sounds, ro...	2.0	[con, worst, call, qualiti, voic, sound, robot...
2	[nice, pair, true, wireless, stereo, earphones...	5.0	[nice, pair, true, wireless, stereo, earphon, ...
3	[first, tws, earbuds, using, oneplus, bullets,...	4.0	[first, tws, earbud, use, oneplus, bullet, wir...
4	[honestly, say, first, buds, comes, surprise, ...	1.0	[honest, say, first, bud, come, surpris, numbr...

```
In [30]: df['Product_Review'] = df['clean_review'].apply(lambda x:' '.join(y for y in x))
         df.head()
```

Out[30]:

	Product_Review	Ratings	clean_review
0	worth buy comfort one kidney	5.0	[worth, buy, comfort, one, kidney]
1	con worst call qualiti voic sound robot end lo...	2.0	[con, worst, call, qualiti, voic, sound, robot...
2	nice pair true wireless stereo earphon oneplus...	5.0	[nice, pair, true, wireless, stereo, earphon, ...
3	first tws earbud use oneplus bullet wireless e...	4.0	[first, tws, earbud, use, oneplus, bullet, wir...
4	honest say first bud come surpris numbr fit pr...	1.0	[honest, say, first, bud, come, surpris, numbr...

- Getting sense of words for all ratings using WordCloud

Word Cloud is a data visualization technique used for representing text data in which the size of each **word** indicates its frequency or importance.
Similarly, we found the sense of words for ratings 2.0 – 5.0 and the output will be as follows:

```
buy ]]
```

In [29]:
```python
df['clean_review']=processed_review  #Assigning this to the dataframe
df.head()
```

Out[29]:

	Product_Review	Ratings	clean_review
0	[worth, buy, comfortable, one, kidney]	5.0	[worth, buy, comfort, one, kidney]
1	[cons, worst, call, quality, voice, sounds, ro...	2.0	[con, worst, call, qualiti, voic, sound, robot...
2	[nice, pair, true, wireless, stereo, earphones...	5.0	[nice, pair, true, wireless, stereo, earphon, ...
3	[first, tws, earbuds, using, oneplus, bullets,...	4.0	[first, tws, earbud, use, oneplus, bullet, wir...
4	[honestly, say, first, buds, comes, surprise, ...	1.0	[honest, say, first, bud, come, surpris, numbr...

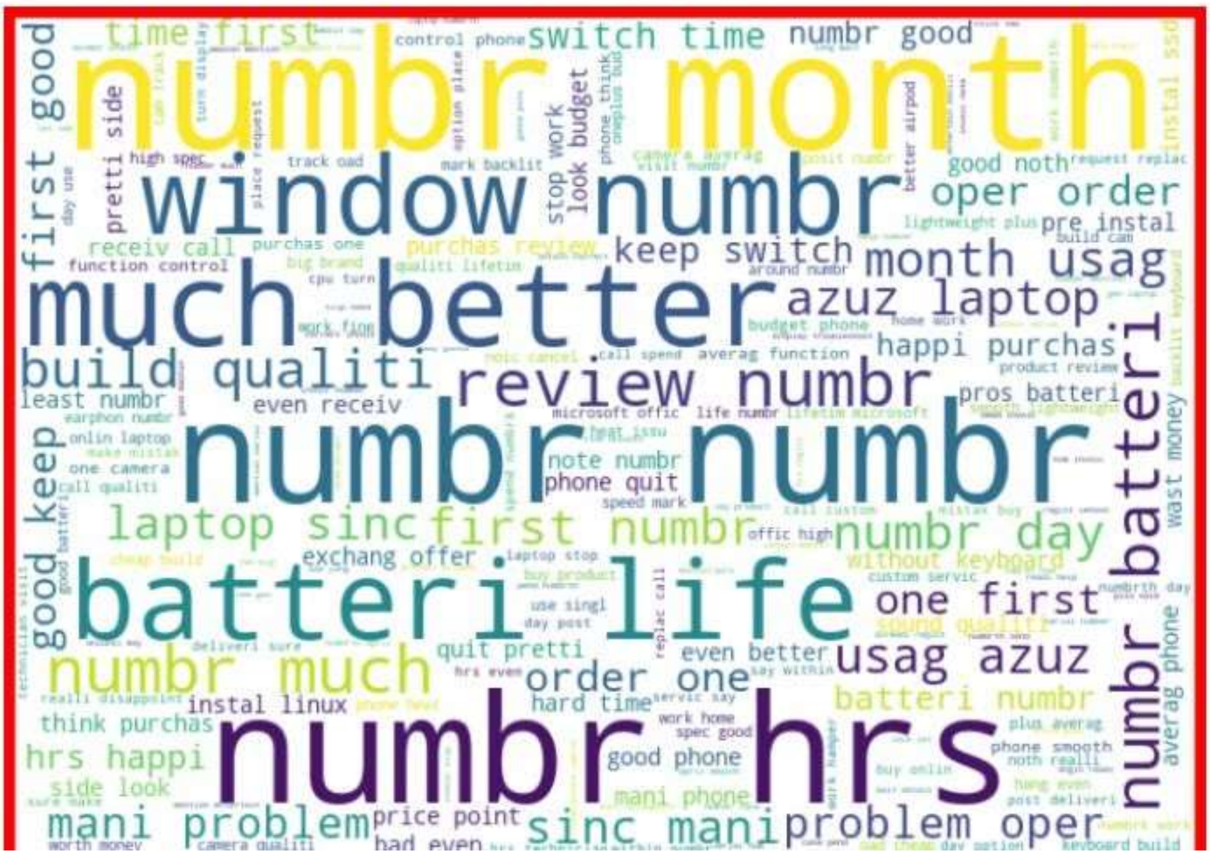
In [30]:
```python
df['Product_Review'] = df['clean_review'].apply(lambda x:' '.join(y for y in x))
df.head()
```

Out[30]:

	Product_Review	Ratings	clean_review
0	worth buy comfort one kidney	5.0	[worth, buy, comfort, one, kidney]
1	con worst call qualiti voic sound robot end lo...	2.0	[con, worst, call, qualiti, voic, sound, robot...
2	nice pair true wireless stereo earphon oneplus...	5.0	[nice, pair, true, wireless, stereo, earphon, ...
3	first tws earbud use oneplus bullet wireless e...	4.0	[first, tws, earbud, use, oneplus, bullet, wir...
4	honest say first bud come surpris numbr fit pr...	1.0	[honest, say, first, bud, come, surpris, numbr...

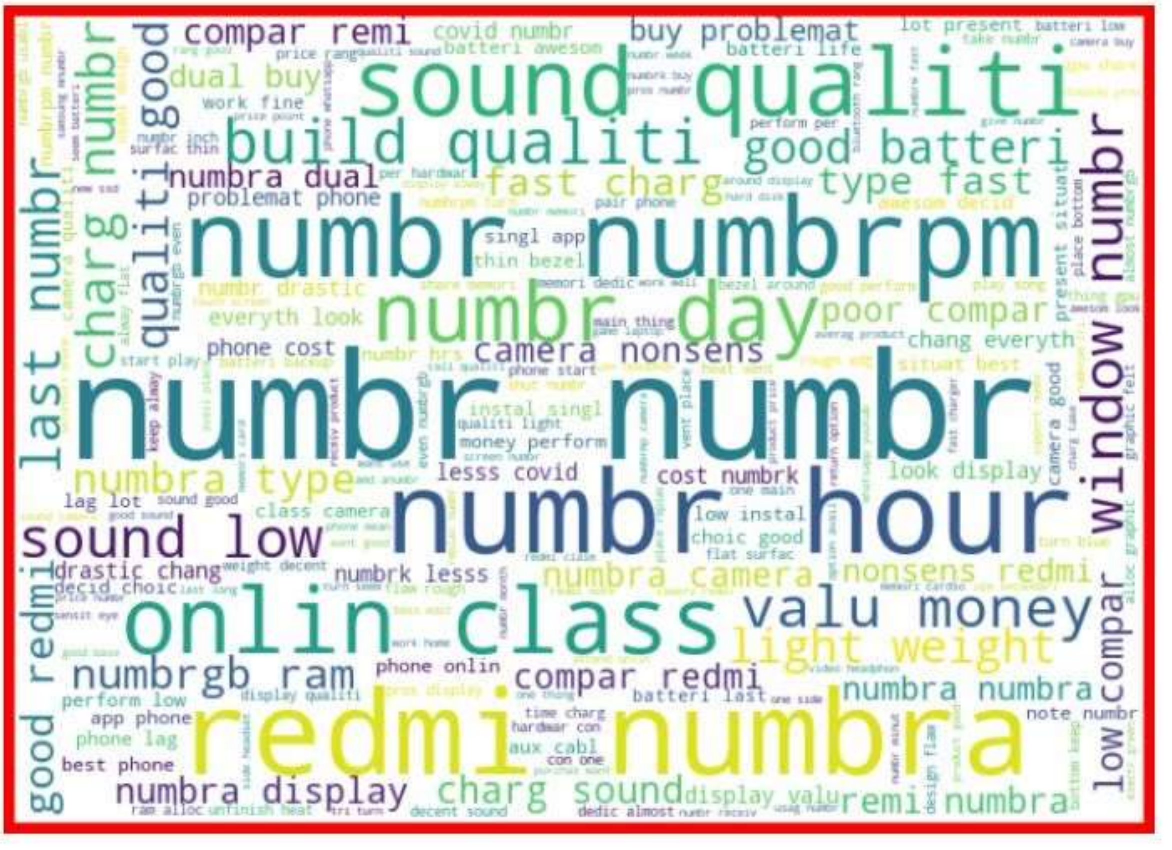
For rating 2.0:

In [35]:
```python
#Getting sense of words in Rating 2
one = df['Product_Review'][df['Ratings']==2.0]

one_cloud = WordCloud(width=700,height=500,background_color='white',max_words=200).generate(' '.join(one))

plt.figure(figsize=(10,8),facecolor='r')
plt.imshow(one_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```



For rating 3.0:

In [34]:
```
#Getting sense of words in Rating 3
one = df['Product_Review'][df['Ratings']==3.0]

one_cloud = WordCloud(width=700,height=500,background_color='white',max_words=200).generate(' '.join(one))

plt.figure(figsize=(10,8),facecolor='r')
plt.imshow(one_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```

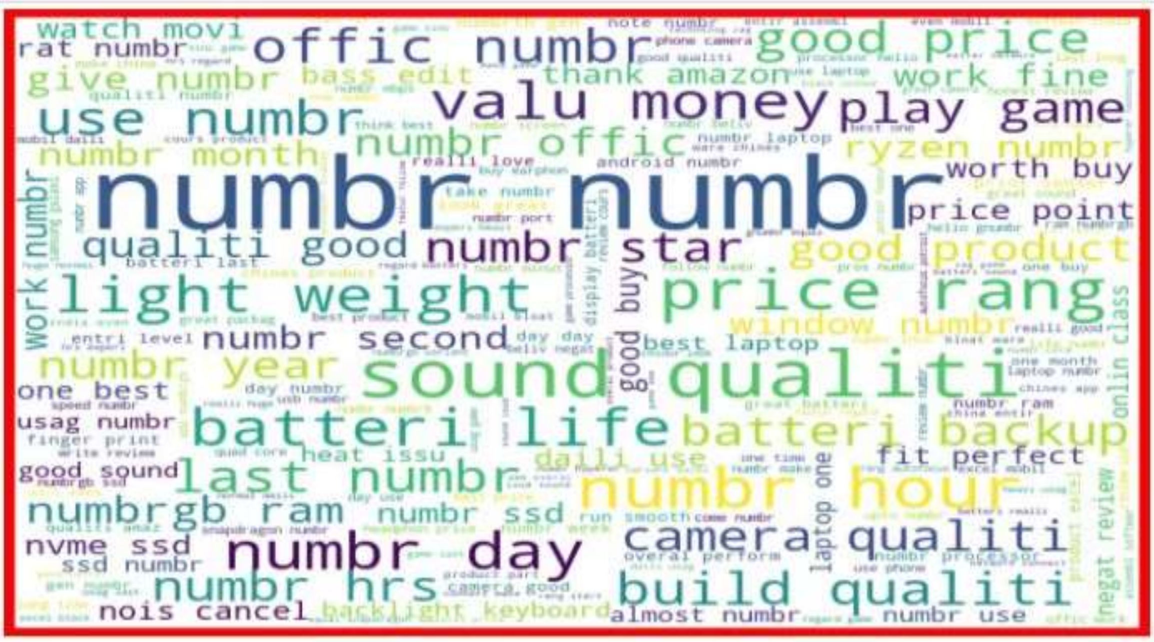


For rating 4.0:

```
In [33]: #Getting sense of words in Rating 4
         one = df['Product_Review'][df['Ratings']==4.0]

         one_cloud = WordCloud(width=700,height=500,background_color='white',max_words=200).generate(' '.join(one))

         plt.figure(figsize=(10,8),facecolor='r')
         plt.imshow(one_cloud)
         plt.axis('off')
         plt.tight_layout(pad=0)
         plt.show()
```

For rating 5.0:

```
In [32]: #Getting sense of words in Rating 5
         one = df['Product_Review'][df['Ratings']==5.0]

         one_cloud = WordCloud(width=700,height=500,background_color='white',max_words=200).generate(' '.join(one))

         plt.figure(figsize=(10,8),facecolor='r')
         plt.imshow(one_cloud)
         plt.axis('off')
         plt.tight_layout(pad=0)
         plt.show()
```



Observations:

The enlarged texts are the most number of words used there and small texts are the less number of words used.

It varies according to the ratings.

Feature Extraction:

Here we can finally convert our text to numeric using Tf-idf Vectorizer.

Term Frequency Inverse Document Frequency (TF-IDF):

This is very common algorithm to transform text into a meaningful representation of numbers which is used to fit machine algorithm for prediction.

Feature Extraction

```
n [38]: #Converting text into numeric using TfidfVectorizer
        #create object
        tf = TfidfVectorizer()

        #fitting
        features = tf.fit_transform(df['Product_Review'])
        x=features
        y=df[['Ratings']]

        x.shape

ut[38]: (36400, 2927)

n [39]: y.shape

ut[39]: (36400, 1)
```

HARDWARE AND SOFTWARE REQUIREMENTS AND TOOLS USED

HARDWARE: HP ENVI X360AQ105X ***SOFTWARE:***

Jupyter Notebook (Anaconda 3) – Python 3.7.6

Libraries Used:

```
In [15]: #Importing required libraries
         import re # for regex
         import string
         import nltk
         from nltk.corpus import stopwords
         from nltk.tokenize import word_tokenize
         from nltk.stem import SnowballStemmer, WordNetLemmatizer
         from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
```

```
In [16]: !pip install wordcloud

         Collecting wordcloud
           Downloading wordcloud-1.8.1-cp38-cp38-win_amd64.whl (155 kB)
         Requirement already satisfied: numpy>=1.6.1 in c:\users\stead\anaconda3\lib\site-packages (from wordcloud) (1.18.5)
         Requirement already satisfied: matplotlib in c:\users\stead\anaconda3\lib\site-packages (from wordcloud) (3.2.2)
         Requirement already satisfied: pillow in c:\users\stead\anaconda3\lib\site-packages (from wordcloud) (7.2.0)
         Requirement already satisfied: python-dateutil>=2.1 in c:\users\stead\anaconda3\lib\site-packages (from matplotlib->wordcloud)
         (2.8.1)
         Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\stead\anaconda3\lib\site-packages (from matplotlib->wordcloud) (1.
         2.0)
         Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in c:\users\stead\anaconda3\lib\site-packages (from mat
         plotlib->wordcloud) (2.4.7)
         Requirement already satisfied: cycler>=0.10 in c:\users\stead\anaconda3\lib\site-packages (from matplotlib->wordcloud) (0.10.0)
         Requirement already satisfied: six>=1.5 in c:\users\stead\anaconda3\lib\site-packages (from python-dateutil>=2.1->matplotlib->w
         ordcloud) (1.15.0)
         Installing collected packages: wordcloud
         Successfully installed wordcloud-1.8.1
```

```
In [17]: from wordcloud import WordCloud
```

```
In [20]: import nltk
         nltk.download('stopwords')
```

MODEL/S DEVELOPMENT AND EVALUATION

- Listing down all the algorithms used for the training and testing.

Model building

```
In [40]: #Importing train_test_split, Logistic Regression and accuracy_score
         from sklearn.model_selection import train_test_split
         from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import accuracy_score
```

```
In [41]: def max_acc_score(reg,x,y):
             max_score=0
             for r_state in range (42,101):
                 x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=r_state,test_size=0.20)
                 reg.fit(x_train,y_train)
                 pred=reg.predict(x_test)
                 acc_score=accuracy_score(y_test,pred)
                 print("The accuracy score at r_state", r_state, "is", acc_score)
                 if acc_score>max_score:
                     max_score=acc_score
                     final_r_state=r_state
             print("The maximum accuracy score", max_score, "is achieved at", final_r_state)
             return max_score
```

```
In [42]: LR=LogisticRegression()
         max_acc_score(LR,x,y)
```

```
In [43]: #Creating train_test_split using best random_state
         x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=56,test_size=.20)
```

- Running and evaluating the models

Finding best model

In [44]:
```
#Importing various classification models for testing
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
```

In [45]:
```
#Initializing the instance of the model
LR=LogisticRegression()
mnb=MultinomialNB()
dtc=DecisionTreeClassifier()
knc=KNeighborsClassifier()
rfc=RandomForestClassifier()
abc=AdaBoostClassifier()
gbc=GradientBoostingClassifier()
```

In [46]:
```
models= []
models.append(('Logistic Regression',LR))
models.append(('MultinomialNB',mnb))
models.append(('DecisionTreeClassifier',dtc))
models.append(('KNeighborsClassifier',knc))
models.append(('RandomForestClassifier',rfc))
models.append(('AdaBoostClassifier',abc))
models.append(('GradientBoostingClassifier',gbc))
```

In [47]:
```
#Importing required modules and metrices
from sklearn.metrics import confusion_matrix,classification_report
from sklearn.model_selection import cross_val_score
```

In [48]:
```
#Making a for loop and calling the algorithm one by one and save data to respective model using append function
Model=[]
score=[]
cvs=[]
rocscore=[]
for name,model in models:
    print('******************************',name,'****************************')
    print('\n')
    Model.append(name)
    model.fit(x_train,y_train)
    print(model)
    pre=model.predict(x_test)
    print('\n')
    AS=accuracy_score(y_test,pre)
    print('accuracy_score: ',AS)
    score.append(AS*100)
    print('\n')
    sc=cross_val_score(model,x,y,cv=5,scoring='accuracy').mean()
    print('cross_val_score: ',sc)
    cvs.append(sc*100)
    print('\n')
    print('Classification report:\n ')
    print(classification_report(y_test,pre))
    print('\n')
    print('Confusion matrix: \n')
    cm=confusion_matrix(y_test,pre)
    print(cm)
    print('\n\n\n')
```

After running the above code, the output will be as follows:

```
In [49]: #Finalizing the result
result=pd.DataFrame({'Model':Model, 'Accuracy_score': score,'Cross_val_score':cvs})
result
```

Out[49]:

	Model	Accuracy_score	Cross_val_score
0	Logistic Regression	89.711538	59.967033
1	MultinomialNB	87.239011	57.736264
2	DecisionTreeClassifier	90.439560	52.329670
3	KNeighborsClassifier	88.337912	50.505495
4	RandomForestClassifier	90.439560	64.656593
5	AdaBoostClassifier	54.258242	44.609890
6	GradientBoostingClassifier	88.997253	59.222527

We can see that Random Forest and Gradient Boosting and Logistic Regression algorithms are performing well. Now we will try Hyperparameter Tuning to find out the best parameters and try to increase the scores.

Key Metrics for success in solving problem under consideration

The key metrics used here were accuracy_score, cross_val_score, classification report, and confusion matrix. We tried to find out the best parameters and also to increase our scores by using Hyperparameter Tuning and we will be using GridSearchCV method.

1. Cross Validation:

Cross-validation helps to find out the over fitting and under fitting of the model. In the cross validation the model is made to run on different subsets of the dataset which will get multiple measures of the model. If we take 5 folds, the data will be divided into 5 pieces where each part being 20% of full dataset. While running the Cross-validation the 1st part (20%) of the 5 parts will be kept out as a holdout set for validation and everything else is used for training data. This way we will get the first estimate of the model quality of the dataset.

In the similar way further iterations are made for the second 20% of the dataset is held as a holdout set and remaining 4 parts are used for training data during process. This way we will get the second estimate of the model quality of the dataset. These steps are repeated during the cross-validation process to get the remaining estimate of the model quality.

2. Confusion Matrix:

A **confusion matrix**, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a **matching matrix**). Each row of the matrix represents the instances in a predicted class, while each column represents the instances in an actual class (or vice versa). The name stems from the fact that it makes it easy to see whether the system is confusing two classes (i.e., commonly mislabelling one as another).

It is a special kind of contingency table, with two dimensions ("actual" and "predicted"), and identical sets of "classes" in both dimensions (each combination of dimension and class is a variable in the contingency table).

3. Classification Report:

The classification report visualizer displays the precision, recall, F1, and support scores for the model. There are four ways to check if the predictions are right or wrong:

1. **TN / True Negative**: the case was negative and predicted negative
2. **TP / True Positive**: the case was positive and predicted positive
3. **FN / False Negative**: the case was positive but predicted negative
4. **FP / False Positive**: the case was negative but predicted positive

Precision: Precision is the ability of a classifier not to label an instance positive that is actually negative. For each class, it is defined as the ratio of true positives to the sum of a true positive and false positive. It is the accuracy of positive predictions. The formula of precision is given below:
Precision = TP/ (TP + FP)
Recall: Recall is the ability of a classifier to find all positive instances. For each class it is defined as the ratio of true positives to the sum of true positives and false negatives. It is also the fraction of positives that were correctly identified. The formula of recall is given below:
Recall = TP/(TP+FN)
F1 score: The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. F1 scores are lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy. The formula is:

```
Accuracy score:  90.4532967032967
Cross validation score:  57.6978721718759
Classification report:

              precision    recall  f1-score   support

         1.0       0.90      0.94      0.92      2286
         2.0       0.96      0.86      0.91       425
         3.0       0.88      0.86      0.87       606
         4.0       0.88      0.86      0.87      1186
         5.0       0.92      0.91      0.92      2777

    accuracy                           0.90      7280
   macro avg       0.91      0.89      0.90      7280
weighted avg       0.90      0.90      0.90      7280

Confusion matrix:

[[2140    1   17   36   92]
 [  14  366   15    6   24]
 [  54    5  520   17   10]
 [  28   10   26 1024   98]
 [ 148    0   12   82 2535]]
```

F1 Score = 2*(Recall * Precision) / (Recall + Precision)
Support: Support is the number of actual occurrences of the class in the specified dataset. Imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or rebalancing. Support doesn't change between models but instead diagnoses the evaluation process.
4. Hyperparameter Tuning:
There is a list of different machine learning models. They all are different in some way or the other, but what makes them different is nothing but input parameters for the model. These input parameters are named as **Hyperparameters.** These hyperparameters will define the architecture of the model, and the best part about these is that you get a choice to select these

for your model. You must select from a specific list of hyperparameters for a given model as it varies from model to model.
We are not aware of optimal values for hyperparameters which would generate the best model output. So, what we tell the model is to explore and select the optimal model architecture automatically. This selection procedure for hyperparameter is known as **Hyperparameter Tuning. We can do tuning by using GridSearchCV.**
GridSearchCV is a function that comes in Scikit-learn (or SK-learn) model selection package. An important point here to note is that we need to have Scikit-learn library installed on the computer. This function helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyperparameters.

Hyperparameter Tuning

In [50]:
```
#RandomForestClassifier
parameters={'n_estimators':[1,10,100]}
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
rfc=RandomForestClassifier(random_state=76)   #Using the best random state we obtained
rfc=GridSearchCV(rfc,parameters,cv=3,scoring='accuracy')
rfc.fit(x_train,y_train)
print(rfc.best_params_)      #Printing the best parameters obtained
print(rfc.best_score_)       #Mean cross-validated score of best_estimator
```

```
{'n_estimators': 100}
0.8946085551525415
```

In [51]:
```
#Using the best parameters obtained
rfc=RandomForestClassifier(random_state=56,n_estimators=100)
rfc.fit(x_train,y_train)
pred=rfc.predict(x_test)
print("Accuracy score: ",accuracy_score(y_test,pred)*100)
print('Cross validation score: ',cross_val_score(rfc,x,y,cv=3,scoring='accuracy').mean()*100)
print('Classification report: \n')
print(classification_report(y_test,pred))
print('Confusion matrix: \n')
print(confusion_matrix(y_test,pred))
```

```
Accuracy score:  90.4532967032967
Cross validation score:  57.6978721718759
Classification report:

              precision    recall  f1-score   support

         1.0       0.90      0.94      0.92      2286
         2.0       0.96      0.86      0.91       425
         3.0       0.88      0.86      0.87       606
         4.0       0.88      0.86      0.87      1186
         5.0       0.92      0.91      0.92      2777

    accuracy                           0.90      7280
   macro avg       0.91      0.89      0.90      7280
weighted avg       0.90      0.90      0.90      7280
```

```
In [52]: #GradientBoostingClassifier
         parameters={'n_estimators':[1,10,100]}

         from sklearn.ensemble import GradientBoostingClassifier
         from sklearn.model_selection import GridSearchCV
         gbc=GradientBoostingClassifier(random_state=76)   #Using the best random state we obtained
         gbc=GridSearchCV(gbc,parameters,cv=3,scoring='accuracy')
         gbc.fit(x_train,y_train)
         print(gbc.best_params_)     #Printing the best parameters obtained
         print(gbc.best_score_)      #Mean cross-validated score of best_estimator

         {'n_estimators': 100}
         0.8836195252684691
```

```
In [53]: #Using the best parameters obtained
         gbc=GradientBoostingClassifier(random_state=56,n_estimators=100)
         gbc.fit(x_train,y_train)
         pred=gbc.predict(x_test)
         print("Accuracy score: ",accuracy_score(y_test,pred)*100)
         print('Cross validation score: ',cross_val_score(gbc,x,y,cv=3,scoring='accuracy').mean()*100)
         print('Classification report: \n')
         print(classification_report(y_test,pred))
         print('Confusion matrix: \n')
         print(confusion_matrix(y_test,pred))

         Accuracy score:  88.99725274725274
         Cross validation score:  49.65669627427923
         Classification report:
```

After applying Hyperparameter Tuning, we can see that RandomForestClassifier Algorithm is performing well as the scores are improved, i.e., accuracy score from 90.4 to 90.5 and cross_val_score from 57.344 to 64.346. Now, we will finalize Random ForestClassifier algorithm model as the final model.

FINAL THE MODE

In [54]:
```
rfc_prediction=rfc.predict(x)

#Making a dataframe of predictions
rating_prediction=pd.DataFrame({'Predictions':rfc_prediction})
rating_prediction
```

Out[54]:

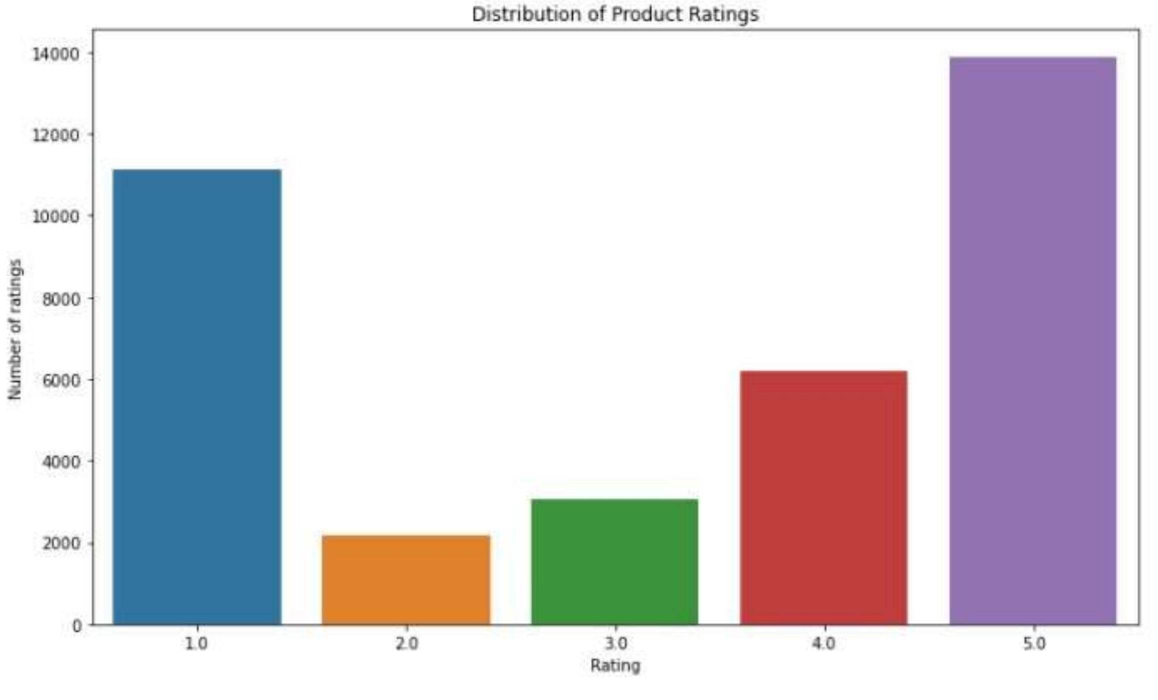
	Predictions
0	5.0
1	2.0
2	5.0
3	4.0
4	1.0
...	...
36395	4.0
36396	1.0
36397	3.0
36398	5.0
36399	3.0

36400 rows × 1 columns

DATA VISUALIZATION

Data Visualization

[13]:
```
f, axes = plt.subplots(figsize=(12,7))
ax = sns.countplot(x=df['Ratings'])
ax.set(title="Distribution of Product Ratings", xlabel="Rating", ylabel="Number of ratings")
plt.show()
```



Observations-: 1.5 has been given the maximum ratings by the users, followed by 1, 4, 3 and 2

Summary

After the completion of this project, we got an insight of how to collect data, preprocessing the data, analyzing the data and building a model.

1.we collected the reviews and ratings data from e-commerce website Amazon it was done by using Webscraping. The framework used for webscraping was Selenium, which has an advantage of automating our process of collecting data.

2.We collected almost 36000+ of data which contained the ratings from 1.0 to 5.0 and their reviews.

3.en, the scrapped data was combined in a single dataframe and saved in a csv file so that we can open it and analyze the data.

4.We did the preprocessing using NLP and the steps are as follows:

a.Removing Punctuations and other special characters

b.Splitting the comments into individual words

c.Removing Stop Words

d.Stemming and Lemmatising

e.Applying Count Vectoriser

f.Splitting dataset into Training and Testing

5.After separating our train and test data, we started running different machine learning classification algorithms to find out the best performing model.

6.We found that RandomForest and GradienBoosting Algorithms and Logistic Regression were performing well, according to their accuracy and cross val scores.

7.Then, we performed Hyperparameter Tuning techniques using GridSearchCV for getting the best parameters and improving the scores. In that, RandomForestClassifier performed well and we finalised that model.

8.We saved the model in pkl format and then saved the predicted values in a csv format.

9.The problems we faced during this project were:

a.More time consumption during hyperparameter tuning for both models, as the data was large.

b.Less number of parameters were used during tuning.

c.Scrapping of data from different websites were of different process and the length of data were differing in most cases so I sticked to Amazon and Scrapped data which are famousin the site.

d.Some of the reviews were bad and the text had more wrong information about the product.

e.WordCloud was not showing proper text which had more positive and negative weightage.

10.Areas of improvement:

a.Less time complexity

b.More accurate reviews can be given

c.Less errors can be avoided.