FLIP ROBO

Product Rating Prediction Report

Submitted by:

Rakesh Chaudhary

ACKNOWLEDGMENT

This project would not have seen the light of the day without the following people and their priceless support and cooperation. Hence I extend my gratitude to all of them.

As a student of Data Trained Education, I would first of all like to express my gratitude to FlipRobo Team and seniors for granting me permission to undertake the project report in their esteemed organization. I would also like to express my sincere thanks to Miss Khushboo Mam for supporting me and being always there for me whenever I needed.

During the actual research work with FlipRobo team and other IOT that set the ball rolling for my project. They had been a source of inspiration through their constant guidance; personal interest; encouragement and help.

I convey my **sincere thanks** to them. In spite of their busy schedule they always found time to guide me throughout the project. I am also grateful to them for reposing confidence in my abilities and giving me the freedom to work on my project. Without their invaluable help I would not have been able to do justice to the project.

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.

Model Building Phase

After collecting the data, you need to build a machine learning model.
Before model building do all data preprocessing 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 Preprocessing
4. Model Building
5. Model Evaluation
6. Selecting the best model

Conceptual Background of the Domain Problem:-

As online marketplaces have been popular during the past decades, the online sellers and merchants ask their purchasers to share their opinions about the products they have bought. As a result, millions of reviews are being generated daily which makes it difficult for a potential consumer to make a good decision on whether to buy the product. Analysing this enormous amount of opinions is also hard and time consuming for product manufacturers. This thesis considers the problem of classifying reviews by their overall semantic (positive or negative).

Sentiment analysis and classification is a computational study which attempts to address this problem by extracting subjective information from the

given texts in natural language, such as opinions and sentiments. Different approaches have used to tackle this problem from natural language processing, text analysis, computational linguistics, and biometrics. In recent years, Machine learning methods have got popular in the semantic and review analysis for their simplicity and accuracy.

Amazon is one of the e-commerce giants that people are using every day for online purchases where they can read thousands of reviews dropped by other customers about their desired products. These reviews provide valuable opinions about a product such as its property, quality and recommendations which helps the purchasers to understand almost every detail of a product. This is not only beneficial for consumers but also helps sellers who are manufacturing their own products to understand the consumers and their needs better.

Review of Literature

This project is considering the sentiment classification problem for online reviews using supervised approaches to determine the overall semantic of customer reviews by classifying them into positive and negative sentiment. The data used in this study is a set of beauty product reviews from Amazon & Flipkart that is collected from Web scrapping tool.

We analyse the information available in the literature. Aside from figuring out how consumers are effectively using product reviews and their associated star-ratings to make purchase decisions as well as emphasizing on the imbalanced distribution of the online reviews, this chapter outlines the different approaches undertook by researchers to predict ratings from the text content of reviews.

Rating is a classification or ranking of someone or something based on a comparative assessment of their quality, standard or overall performance. 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.

Customer Reviews help customers to learn more about the product and decide whether it is the right product for them.
Customer Reviews should give customers genuine product feedback from fellow shoppers. We have a zero tolerance policy for any review designed to mislead or manipulate customers.

The following are types of reviews that we do not allow and will remove:

- A review by someone who has a direct or indirect financial interest in the product.
- A review by someone perceived to have a close personal relationship with the product's owner, author or artist.
- A review by the product manufacturer, posing as an unbiased shopper.

- Multiple negative reviews for the same product from one customer.
- A review in exchange for monetary reward.
- A review of a game in exchange for bonus in-game credits.
- A negative review from a seller on a competitor's product.
- A positive review from an artist on a peer's album in exchange for receiving a positive review from them.

Motivation for 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.

There is a large number of papers that have been published in the field of machine learning. One of the most used approaches for sentiment classification is machine learning algorithms. This section attempts to cover some of them.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

Data contains attributes such as Ratings (which is the overall rating of the reviewer), Review Text (which contains all the thoughts of customer for that particular product), Summary (this attributes contains brief review by the customer), New review (which contains all the word count in the review text and summary), Review character count (this attribute contains all the character count of review text and summary attributes).

In this project we are going to predict the ratings. We will understand the sentiments of the customers and based on that we will predict the ratings.

	Ratings	Review_Text	Summary	Full_review	new_review	Review_character_count
0	5	Mannn this is just incredible 😍 i was scared o...	Best in the market!	best market mannn incredible scared online pay...	47	286
1	5	This laptop is soo good and I bought it after ...	Brilliant	brilliant laptop soo good buy lot research exc...	56	322
2	5	Amazing laptop just great i bought this produc...	Brilliant	brilliant amazing laptop great buy product diw...	29	187
3	5	Brought this laptop after fair bit of research...	Terrific	terrific bring laptop fair bit research arguab...	22	164
4	5	11400 Has Much Better Single Core Performance ...	Mind-blowing purchase	mindblowing purchase much well single core per...	24	154
5	5	Great laptop with all the features exclusive!\...	Just wow!	wow great laptop feature exclusive compare asu...	22	131
6	5	The best performance.\n\nI searched a lot befo...	Excellent	excellent best performance search lot buy lapt...	52	320
7	5	Value for money in this price segment...laptop...	Just wow!	wow value money price segmentlaptop similar sp...	46	309
8	5	I am reviewing this laptop after 16 days of us...	Terrific	terrific review laptop day use performance tre...	50	303
9	5	Awesome laptop\nGood perfomance\nBattery life ...	Brilliant	brilliant awesome laptop good perfomance batte...	22	148

- **<u>Data Sources and their formats</u>**

This project was done in Three phases:-

- Data Collection Phase
  In this phase we have scrapped more than 50000 reviews from amazon and flip kart website by selenium web driver. For this project we have scrapped reviews for different laptops, phones, Headphones, smart watches, professional cameras, printer, and monitors.
- Data Pre-processing Phase
  In this phase we have done all the pre-processing steps like Natural Language Process (NLP), outlier's detections and overfitting as well.
  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

All the pre-processing done after that, we built a machine learning model.

- **Data Pre-processing Done**
  - Checking Missing Values

```
df.isna().sum()
```

```
Ratings             0
Review_Text         2
Summary           116
dtype: int64
```

- There are some features having missing values. As we see that the number of missing values are too low.
- So it is good to fill them we will drop them, as it will not affect our dataset.

```
## Droppping Nans
df.dropna(inplace=True)
df.shape
```

```
(56124, 3)
```

  - Checking Unique value in Target variable (Ratings)

```
df['Ratings'].unique()
```

```
array(['5', '4', '3', '1', '2', '2.0 out of 5 stars',
       '1.0 out of 5 stars', '3.0 out of 5 stars', '4.0 out of 5 stars'],
      dtype=object)
```

As we can see at these unique values in our target column we can see that the string entries need to be replaced with the respective numeric values (number of stars)

```
df['Ratings'] = df['Ratings'].replace('1.0 out of 5 stars',1)
df['Ratings'] = df['Ratings'].replace('2.0 out of 5 stars',2)
df['Ratings'] = df['Ratings'].replace('3.0 out of 5 stars',3)
df['Ratings'] = df['Ratings'].replace('4.0 out of 5 stars',4)
df['Ratings'] = df['Ratings'].replace('5.0 out of 5 stars',5)
df['Ratings'] = df['Ratings'].astype('int')
df['Ratings'].value_counts()
```

```
5    19941
1    11596
3     9117
4     7931
2     7539
Name: Ratings, dtype: int64
```

- Removing Unwanted characters from the review

Best in the market! Mannn this is just incredible 😍 i was scared of online payment this is my first time with my dream laptop 😍 very happy to see t he awesome product quality 😃 i5 11400H + RTX 3050 is insaaanneee and you ill get 1tb SSD / great build quality / GAMEPLAY ARE JUST MIND-BLOWING 😲 g ives more performance than ryzen 7 4800h & ryzen 5 5600h trust me I've buy this one by comparing all those cpu's / 90whr battery in this segment are t otaly nice 😍 just go for this product 😃😊

Observations:

- We can see that our reviews holds many emoji's, our model will not understand to these emoji's so we have to handle them.
- There are many special character used like -,:,:,...,@,(,) etc.
- Some words are in Capital letters and some are small letters.

Removing Emojies

```
Full_review=[]
for i in df.Full_Review:
    #appending text after removing the emojis from it
    Full_review.append(clean(i, no_emoji=True))
```

```
## Dropping Full review from dataframe
df.drop('Full_Review',axis=1,inplace=True)

### Adding without emojies column in our dataframe
df['Full_review']=Full_review
```

Lower Casing:-

```
## # Lowercasing the words in review
df['Full_review'] = df['Full_review'].apply(lambda x : x.lower())
```

Removing Unwanted Text and contracted words

```
def unwanted_text(string):
    string = re.sub(r"won't", "will not",string)
    string = re.sub(r"don't", "do not",string)
    string = re.sub(r"doesn't", "does not",string)
    string= re.sub(r"haven't", "have not", string)
    string = re.sub(r"can't", "can not", string)
    string = re.sub(r"im ", "i am", string)
    string = re.sub(r"yo ", "you ",string)
    string = re.sub(r"n\'t", " not", string)
    string = re.sub(r"\'re", " are", string)
    string = re.sub(r"\'s", " is", string)
    string = re.sub(r"\'d", " would", string)
    string = re.sub(r"\'ll", " will", string)
    string = re.sub(r"\'t", " not", string)
    string = re.sub(r"\'ve", " have", string)
    string = re.sub(r"\'m", " am", string)
    string = re.sub(r"<br>", " ", string)
    ##removing all the urls:
    string = re.sub(r'http\S+', '', string)
    return string
```

```
## Decontracted all the reviews
df['Full_review'] = df['Full_review'].apply(lambda x : unwanted_text(x))
```

Removing Special Characters

```
### Removing Special charactors
charactors=['.',':',';','(',')','!','@','$','^','[^\w\s]','/','?','<','>']
for i in charactors:
    df['Full_review'] = df['Full_review'].str.replace(i,'')
```

Replacing Space Keys

```
space_key=['\n','_','-']
for j in space_key:
    df['Full_review'] = df['Full_review'].str.replace(j,' ')
```

Removing Stop words

```
# Removing all the stopwords
stop_charactors = stopwords.words('english')
df['Full_review'] = df['Full_review'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_charactors)]))
```

Lemmatization:

Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma.

Example: - Lemmatize minimizes text ambiguity. Example words like bicycle or bicycles are converted to base word bicycle. Basically, it will convert all words having the same meaning but different representation to their base form.

```
# Defining functiom to convert nltk tag to wordnet tags
def nltk_tag_to_wordnet_tag(nltk_tag):
    if nltk_tag.startswith('J'):
        return wordnet.ADJ
    elif nltk_tag.startswith('V'):
        return wordnet.VERB
    elif nltk_tag.startswith('N'):
        return wordnet.NOUN
    elif nltk_tag.startswith('R'):
        return wordnet.ADV
    else:
        return None

# Defining function to lemmatize our text
def lemmatize_sentence(sentence):
    # tokenize the sentence and find the pos_tag
    nltk_tagged = nltk.pos_tag(nltk.word_tokenize(sentence))
    # tuple of (token, wordnet_tag)
    wordnet_tagged = map(lambda x : (x[0], nltk_tag_to_wordnet_tag(x[1])), nltk_tagged)
    lemmatize_sentence = []
    for word, tag in wordnet_tagged:
        if tag is None:
            lemmatize_sentence.append(word)
        else:
            lemmatize_sentence.append(lemination.lemmatize(word,tag))
    return " ".join(lemmatize_sentence)

df['Full_review'] = df['Full_review'].apply(lambda x : lemmatize_sentence(x))
```

Normalization:

Normalization is **the process of converting a token into its base form.** In the normalization process, the inflectional form of a word is removed so that the base form can be obtained.

```
## scraping noise text:
def scrap(text):
    # remove HTML markup
    text = re.sub("(<.*?>)", "", text)
    # remove non-ascii and digits
    text = re.sub("(\\W)", " ", text)
    text = re.sub("(\\d)", "", text)
    # remove white space
    text = text.strip()
    return text

df['Full_review'] = df['Full_review'].apply(lambda x : scrap(x))
```

Checking Sample Now:-

```
## printing review again:
for i in range(10):
    print(df.Full_review[i])
    print("\n**********Next Review*********\n")
```

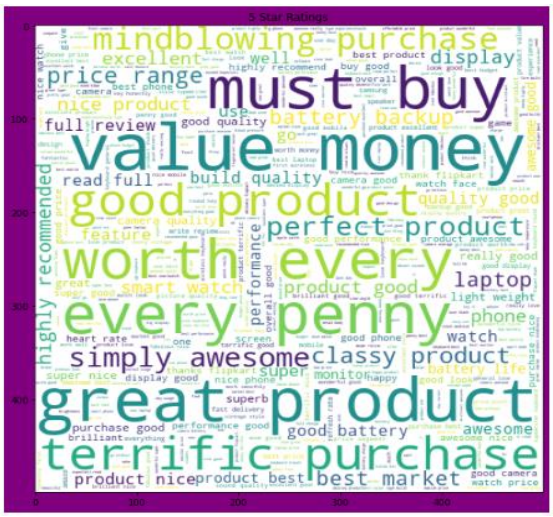
```
best market mannn incredible scared online payment first time dream laptop happy see awesome product quality i h rtx insaaanneee ill get tb ssd great build quality gameplay mindblowing give performance ryzen h ryzen h trust buy one compare cpu whr battery segment totaly nice go product

**********Next Review*********

brilliant laptop soo good buy lot research excellent build quality plastic super tuf buy video edit casual gaming boy performs well expectation pro cp u h good cpu almost fast last generation i rtx perform good gaming k video edit tb ssd worry storage rgb keyboard look good best part trackpad soo smo oth work laptop h read

**********Next Review*********
```

Now it looks pretty good now.

- **<u>Data Inputs- Logic- Output Relationships</u>**

  We have use word cloud method for finding the trends between input and output variable.

  - Five Star Ratings



- Four Star Ratings

- Three Star Ratings

- Two Star Ratings

One Star Ratings

Observation of Word Cloud Visualization:-

`For Rating: 1` It mostly consists of words like watch, use, bad product, waste, time, money, bad experience, issue etc

`For Rating: 2` It mostly consists of words like good, phone, use, watch, poor, issue, waste money, quality good, bad, problem etc

`For Rating: 3` It mostly consists of words like sound quality, good, use, time, camera quality, display, buy, build quality etc

`For Rating: 4` It mostly consists of words like use, buy, phone, watch, good product, good quality, good choice, nice product etc

`For Rating: 5` It mostly consists of words like price range, value money, good product, well, go, simply awesome, perfect product etc

State the set of assumptions (if any) related to the problem under consideration

We assume that exploratory data analysis shows highest word frequency of stop words, which were removed in the subsequent pre-processing steps. No outliers was found in the raw dataset, but some blank rows was found in row dataset which we drop as they are in less quantity.

The features of interest were found to be 'review Text', 'summary' and 'rating' as these were most useful and relevant for model building, prediction as well as abstraction. In the 'review Text' field, mixed bag of short and long reviews were found, some of which produced no or little information about the product. In the exploratory analysis, review Text attribute was explored using the word-cloud visualization for each category.

By looking into the target variable/label, we assumed that it was a multiclass classification type of problem. Also, we observed that our dataset was imbalance so we will have to balance the dataset for better prediction accuracy outcome.

Hardware and Software Requirements and Tools Used

The Ratings Prediction Project is about built a machine learning model that could predict the default case so that the company could to which products are liked by the customers and which are not. So for that we require lots of libraries and packages to work upon this project. Hardware Required:-

- Processor:- Core i5 or above
- Ram:- 8GB or above
- SSD:- 256GB or Above

Software Required:- • Anaconda Promt/Nevigator

Libraries Required:-

- Pandas
- Numpy
- Matplotlib
- Seaborn

Importing NLP libraries:

- **import** re
- **import** nltk
- **from** nltk.corpus **import** stopwords
- nltk.download('stopwords', quiet=**True**)
- nltk.download('punkt', quiet=**True**)
- nltk.download('wordnet', quiet=**True**)
- nltk.download('averaged_perceptron_tagger', quiet=**True**)

importing warnings:

- **import** warnings
- warnings.filterwarnings('ignore')

- Sklearn.preprocessing.StandardScaler
- from sklearn.model_selection import train_test_split,cross_val_score
- sklearn.linear_model.LogisticRegression
- sklearn.trees.DecisionTreeClassifier

•Sklearn.ensemble.RandomForestClassifier,AdaBoostClassifier,GradientBoostigClassifier

• XGBClassifier

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

Handling Outliers Using Zscore:-

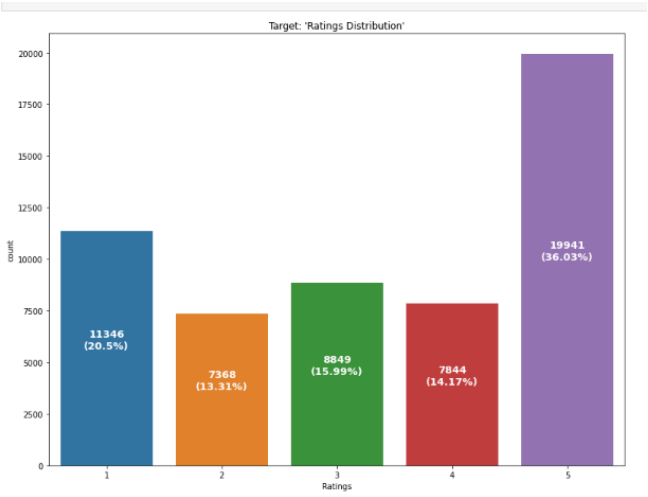
```
from scipy import stats
from scipy.stats import zscore
```

```
z_score=zscore(df[['new_review']])
abs_zscore=np.abs(z_score)
```

```
threshold=3
new_entry=(abs_zscore<threshold).all(axis=1)
df_new=df[new_entry]
print("The shape before: ", df.shape)
print("The shape after: ",df_new.shape)
```

```
The shape before:  (56124, 6)
The shape after:  (55348, 6)
```

- Handling Imbalanced Dataset:



- If we failed to handle this problem then the model will become a disaster because modeling using class-imbalanced data is biased in favor of the majority class.
- Looking at the above count plot for our target variable (Ratings) we can say that the data set is having most number of reviews rated as 5 star and very less number of reviews rated as 2 star and 3 star.
- Which will cause the Imbalance problem for our Machine Learning model and make it bias. So I am selecting equal number of reviews of each rating as a input for our model to avoid any kind of biasness
- For that first I will shuffle the dataset so that we can select data from both web-sites (Amazon and Flipkart).Then I will select equal number of data of every category and ensure that the rating values are balanced.

A typical example of imbalanced data is encountered in e-mail classification problem where emails are classified into ham or spam. The number of spam emails is usually lower than the number of relevant (ham) emails. So, using the original distribution of two classes leads to imbalanced dataset.

If we take 8000 records for every star then our model will not look as imbalanced dataset.

```
# Select data from every Ratings category
df1 = df[df['Ratings']==1][0:8000]
df2 = df[df['Ratings']==2][0:8000]
df3 = df[df['Ratings']==3][0:8000]
df4 = df[df['Ratings']==4][0:8000]
df5 = df[df['Ratings']==5][0:8000]
```

```
### Adding all the dataframes:
df = pd.concat([df1,df2,df3,df4,df5], ignore_index=True)
```

```
df.Ratings.value_counts()
```

```
1    8000
3    8000
5    8000
4    7931
2    7539
```

Testing of Identified Approaches (Algorithms)

Below listed algorithms are used for this project.

- Logistic Regression
- Decision Tree Classifier
- Random forest Classifier
- SVC
- Ada Boost Classifier
- Gradient Boosting Classifier
- Naïve Bayes Multinomial NB
- LGBM Classifier

- XGBoost Classifier

Run and Evaluate selected models

The idea of distributional semantics that is implemented through 'word vectors' has been used heavily in semantic processing for a wide variety of Applications. This simple idea has probably been the most powerful and useful insight in creating semantic processing systems. Supervised techniques for word sense disambiguation require the input words to be tagged with their senses. The sense is the label assigned to the word. We defined the various classification models mentioned above and assigned them to user created variables. Then we defined the function that would train and predict the multiclass labels for us along with the evaluation metrics. I did not include the cross-validation part in the function and rather created a separate function for that metrics to evaluate only the best scored classification models amongst the original list.
We have defied few functions that will help us for finding the best random state, printing the model's score and checking the cross validation. The code is mentioned below screenshots respectively.

```
### Best Random STATE:
def Random_state(Model,Feature,Target):
    maximum_accu=0
    for i in range(11,36):
        x_train,x_test,y_train,y_test=train_test_split(Feature,Target,test_size=0.25,random_state=i)
        Model.fit(Feature,Target)
        train_pred=Model.predict(x_train)
        test_pred=Model.predict(x_test)
        accu_score=accuracy_score(y_test,test_pred)
        print("For Random State {}, the Accuracy Score is: {}".format(i,accu_score))
        if accu_score>maximum_accu:

            maximum_accu=accu_score
            j=i
    print("\n")
    print(" The Highest Accuracy SCORE is: {}".format(maximum_accu))
    print("\n The Best Random State is:")
    return j
```

Defining a Function for Priting Accuracy:

```
def print_score(clf,x,y,randomstate,train=True):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=randomstate)
    clf.fit(x_train,y_train)
    if train:
        y_pred= clf.predict(x_train)
        print("\n**********************************Training Score**************************************")

        print(f"Accuracy Score : {accuracy_score(y_train,y_pred)*100:.2f}%")
        print("\n **********************************Confusion Matrix**************************************")
        print(confusion_matrix(y_train,y_pred))

        print("\n \n Training Classification Report \n" ,classification_report(y_train,y_pred,digits=2))

    elif train==False:
        pred=clf.predict(x_test)
        print('\n \n')
        print("\n**********************************Test Result**************************************************")
        print(f"Accuracy_Score : {accuracy_score(y_test,pred)*100:.2f}%")
        print("\n**********************************Confusion Matrix**************************************")
        print(confusion_matrix(y_test,pred))

        print("\n \n Test Classification Report \n", classification_report(y_test,pred,digits=2))
```

Defining A funtion for cross Validation:

```
def cross_val(Model,independent,dependent,randomstate):
    x_train,x_test,y_train,y_test=train_test_split(independent,dependent,test_size=0.2,random_state=randomstate)
    Model.fit(x_train,y_train)
    pred=Model.predict(x_test)
    for i in range(3,4):
        cv_score=cross_val_score(Model,x,y,cv=i)
        cv_mean=cv_score.mean()
        print('At cv :- ', i)
        print('Cross Validation score is :- ', cv_mean)
        print('Accuracy score is :- ',accuracy_score(y_test,pred))
        print('\n')
```

Key Metrics for success in solving problem under consideration

- Accuracy Score
- F1 Score
- Cross Validation
- Hyper parameter tuning
- Grid Search CV

- Logistic Regression

```
**********************************Test Result**************************************************
Accuracy_Score : 79.24%

**********************************Confusion Matrix**************************************
[[1949  237  125   18   23]
 [ 440  654  274   32   19]
 [ 224  269 1118  116   40]
 [  23   29  143 1239  186]
 [   9    9   10   72 3812]]


 Test Classification Report
              precision    recall  f1-score   support

           1       0.74      0.83      0.78      2352
           2       0.55      0.46      0.50      1419
           3       0.67      0.63      0.65      1767
           4       0.84      0.76      0.80      1620
           5       0.93      0.97      0.95      3912

    accuracy                           0.79     11070
   macro avg       0.75      0.73      0.74     11070
weighted avg       0.79      0.79      0.79     11070
```

Cross Validation:

```
At cv :-  3
Cross Validation score is :-  0.6546031274575488
Accuracy score is :-  0.7924119241192412
```

- Decision Tree Classifier

```
**********Test Result**********
Accuracy_Score : 82.00%

**********Confusion Matrix**********
[[1800  263  173   35   26]
 [ 318  865  255   38   35]
 [ 182  212 1252   93   58]
 [  19   31   68 1349  109]
 [   4    6   21   47 3811]]

 Test Classification Report
              precision    recall  f1-score   support

           1       0.77      0.78      0.78      2297
           2       0.63      0.57      0.60      1511
           3       0.71      0.70      0.70      1797
           4       0.86      0.86      0.86      1576
           5       0.94      0.98      0.96      3889

    accuracy                           0.82     11070
   macro avg       0.78      0.78      0.78     11070
weighted avg       0.82      0.82      0.82     11070
```

Cross Validation:

```
At cv :-  3
Cross Validation score is :-  0.618016921376506
Accuracy score is :-  0.8176151761517615
```

- Random Forest Classifier

```
**********Test Result**********
Accuracy_Score : 85.87%

**********Confusion Matrix**********
[[2087   90   79    3    9]
 [ 412  850  190   11   21]
 [ 247  141 1336   27   53]
 [  39   11   85 1330  129]
 [   1    0    5   11 3903]]

 Test Classification Report
              precision    recall  f1-score   support

           1       0.75      0.92      0.83      2268
           2       0.78      0.57      0.66      1484
           3       0.79      0.74      0.76      1804
           4       0.96      0.83      0.89      1594
           5       0.95      1.00      0.97      3920

    accuracy                           0.86     11070
   macro avg       0.85      0.81      0.82     11070
weighted avg       0.86      0.86      0.85     11070
```

Cross Validation:-

```
cross_val(RandomForestClassifier(),x,y,15)
```

```
At cv :-  5
Cross Validation score is :-  0.74130886466211
Accuracy score is :-  0.8586269196025293
```

- Ada Boost Classifier

```
**********************************Test Result**************************************************
Accuracy_Score : 66.03%

**********************************Confusion Matrix*********************************************
[[1934   26  257   21   30]
 [ 969   30  423   34   28]
 [ 732   38  853  112   69]
 [  68    4  341  963  218]
 [  64   27   70  229 3530]]

 Test Classification Report
              precision    recall  f1-score   support

           1       0.51      0.85      0.64      2268
           2       0.24      0.02      0.04      1484
           3       0.44      0.47      0.46      1804
           4       0.71      0.60      0.65      1594
           5       0.91      0.90      0.91      3920

    accuracy                           0.66     11070
   macro avg       0.56      0.57      0.54     11070
weighted avg       0.63      0.66      0.63     11070
```

Cross Validation:-

```
cross_val(AdaBoostClassifier(),x,y,15)
```

```
At cv :-  3
Cross Validation score is :-  0.6179265802848283
Accuracy score is :-  0.6603432700993677
```

- Gradient Boosting Classifier

```
**********************************Test Result**************************************************
Accuracy_Score : 75.35%

**********************************Confusion Matrix*********************************************
[[1877  211  138   21   21]
 [ 587  530  300   32   35]
 [ 312  260 1031   91  110]
 [  46   20  177 1115  236]
 [  17   10   36   69 3788]]

 Test Classification Report
              precision    recall  f1-score   support

           1       0.66      0.83      0.74      2268
           2       0.51      0.36      0.42      1484
           3       0.61      0.57      0.59      1804
           4       0.84      0.70      0.76      1594
           5       0.90      0.97      0.93      3920

    accuracy                           0.75     11070
   macro avg       0.71      0.68      0.69     11070
weighted avg       0.75      0.75      0.74     11070
```

Cross Validation:-

```
cross_val(GradientBoostingClassifier(),x,y,15)
```

```
At cv :-  3
Cross Validation score is :-  0.6569520232611761
Accuracy score is :-  0.7525745257452574
```

- Extra Tree Classifier

```
**********************************Test Result**************************************************
Accuracy_Score : 85.89%

**********************************Confusion Matrix*********************************************
[[2086   85   83    4   10]
 [ 403  864  191    9   17]
 [ 242  155 1338   25   44]
 [  32   13  102 1322  125]
 [   5    1    7    9 3898]]

 Test Classification Report
              precision    recall  f1-score   support

           1       0.75      0.92      0.83      2268
           2       0.77      0.58      0.66      1484
           3       0.78      0.74      0.76      1804
           4       0.97      0.83      0.89      1594
           5       0.95      0.99      0.97      3920

    accuracy                           0.86     11070
   macro avg       0.84      0.81      0.82     11070
weighted avg       0.86      0.86      0.86     11070
```

Cross Validation:-

```
cross_val(ExtraTreesClassifier(),x,y,15)
```

```
At cv :-  3
Cross Validation score is :-  0.6689849911200794
Accuracy score is :-  0.8579042457091237
```

- Gaussian NB Classifier

```
**********************************Test Result**************************************************
Accuracy_Score : 68.70%

**********************************Confusion Matrix*********************************************
[[2039   20  133    2   74]
 [ 749  208  344    2  181]
 [ 400   43  998    9  354]
 [  62    6  105  465  956]
 [   8    1   14    2 3895]]

 Test Classification Report
              precision    recall  f1-score   support

           1       0.63      0.90      0.74      2268
           2       0.75      0.14      0.24      1484
           3       0.63      0.55      0.59      1804
           4       0.97      0.29      0.45      1594
           5       0.71      0.99      0.83      3920

    accuracy                           0.69     11070
   macro avg       0.74      0.58      0.57     11070
weighted avg       0.72      0.69      0.64     11070
```

Cross Validation:-

```
cross_val(MultinomialNB(),x,y,15)
```

```
At cv :-  3
Cross Validation score is :-  0.606977468538846
Accuracy score is :-  0.6869918699186992
```

- XGB Classifier

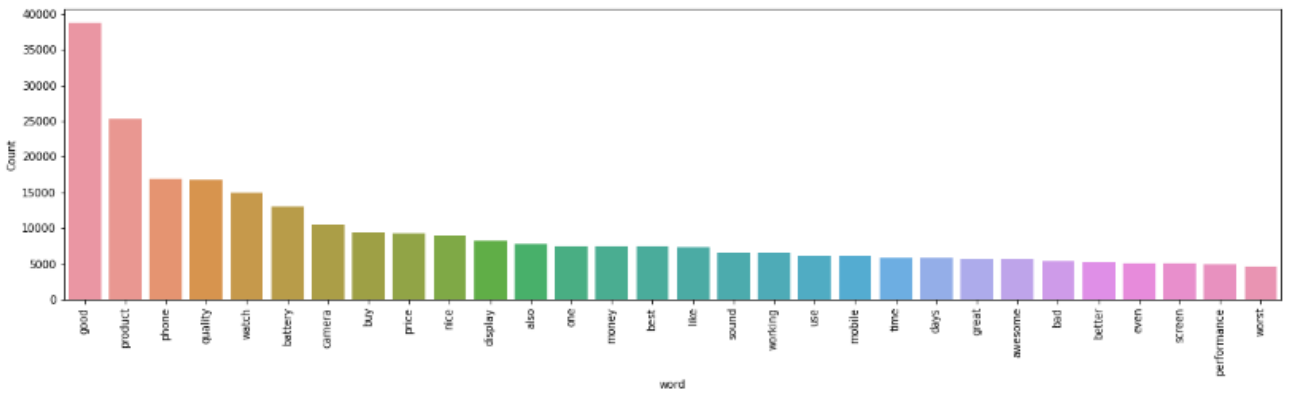
```
**********************************Test Result**************************************************
Accuracy_Score : 81.55%

**********************************Confusion Matrix*********************************************
[[1907  209  121   20   11]
 [ 399  775  268   26   16]
 [ 223  269 1197   73   42]
 [  25   26  121 1307  115]
 [   5    4   18   51 3842]]

 Test Classification Report
              precision    recall  f1-score   support

           1       0.75      0.84      0.79      2268
           2       0.60      0.52      0.56      1484
           3       0.69      0.66      0.68      1804
           4       0.88      0.82      0.85      1594
           5       0.95      0.98      0.97      3920

    accuracy                           0.82     11070
   macro avg       0.78      0.77      0.77     11070
weighted avg       0.81      0.82      0.81     11070
```

Cross Validation:-

```
At cv :-  3
Cross Validation score is :-  0.673772819261067
Accuracy score is :-  0.8155374887082204
```

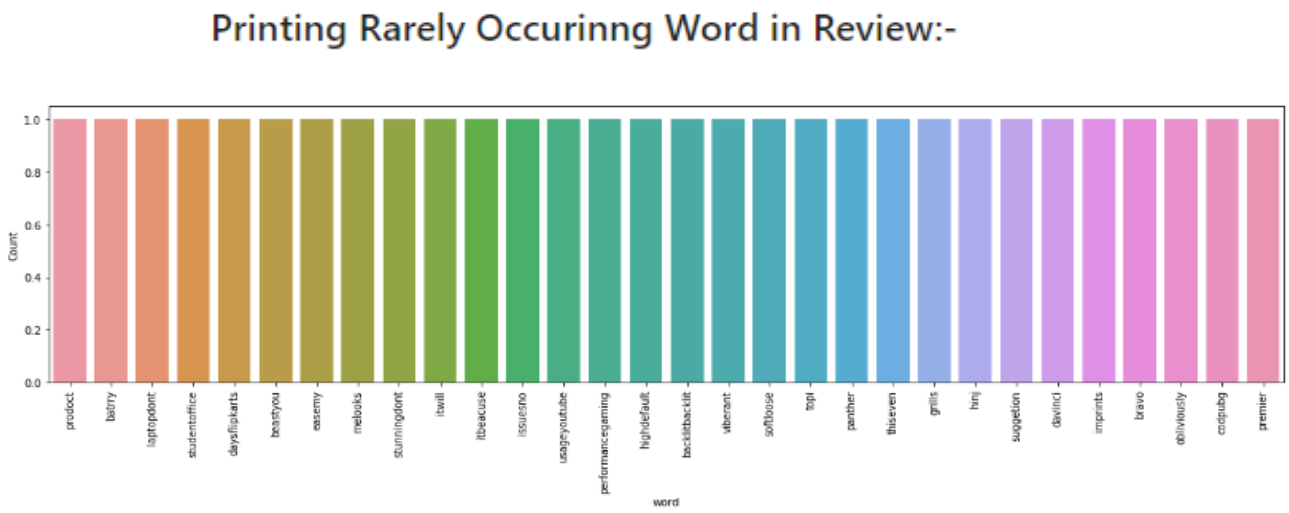
Visualizations

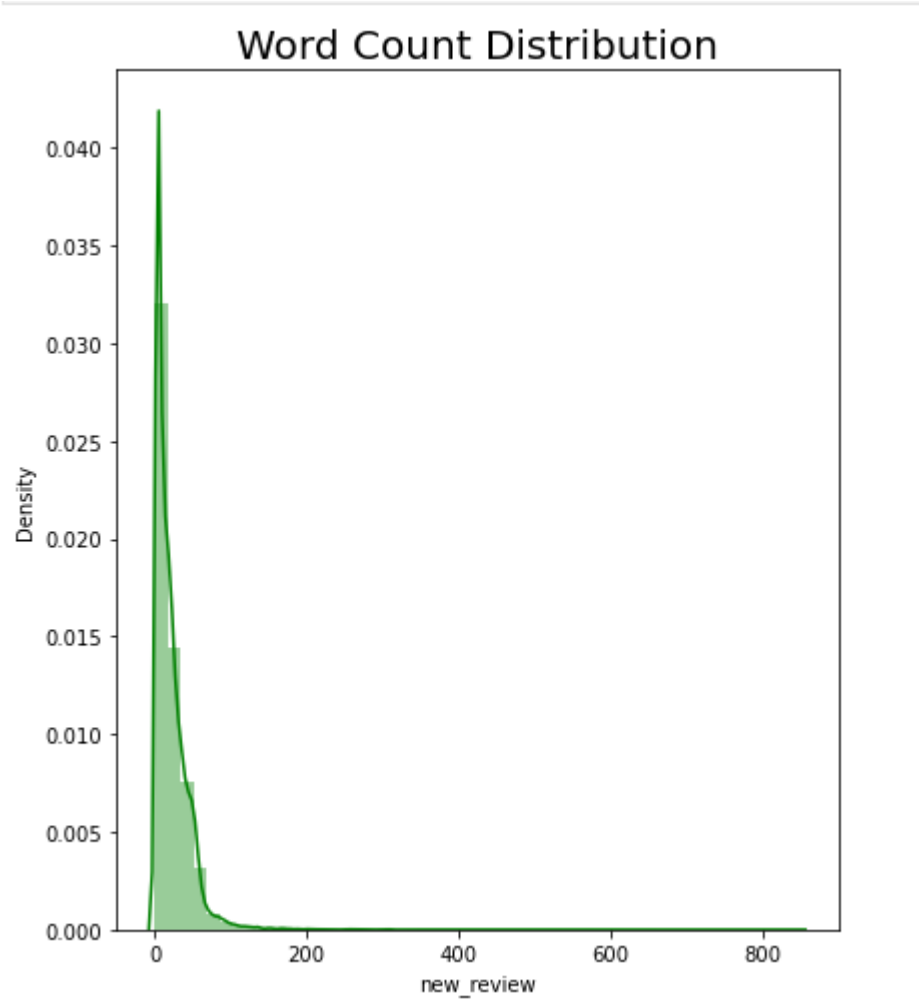
Printing Top Occuring Word in Review:



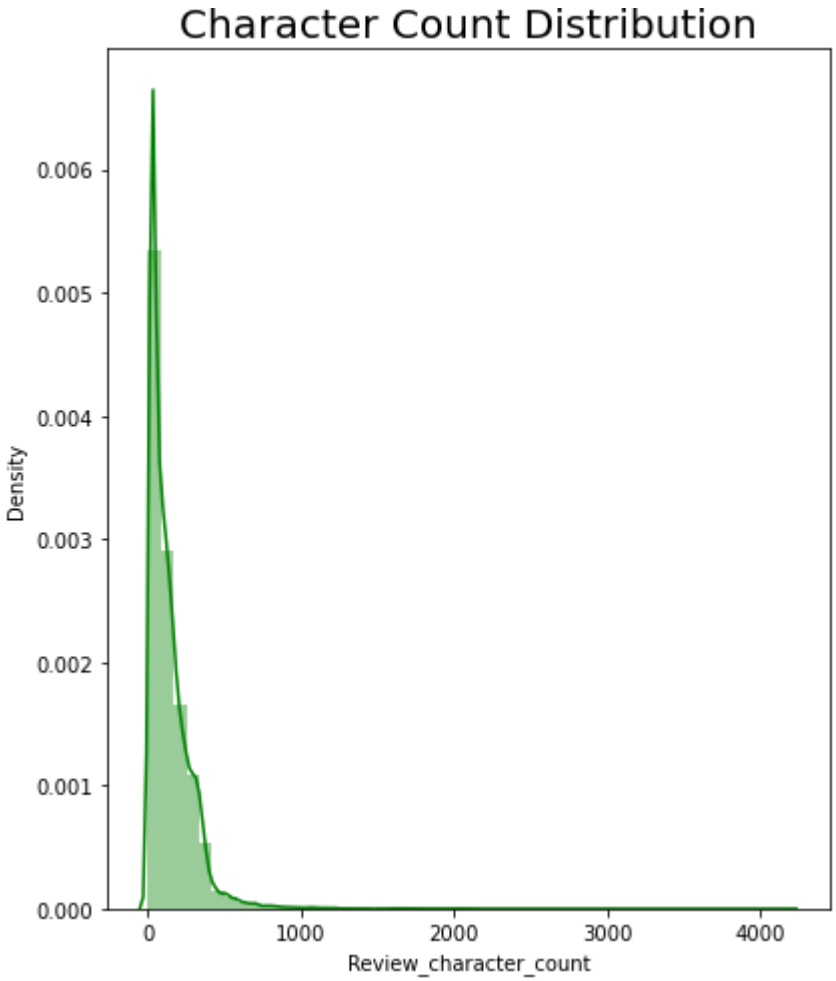
- We have printed the most accruing 20 words in our dataset.

- As we can see that the word 'good' is accruing most of the time.



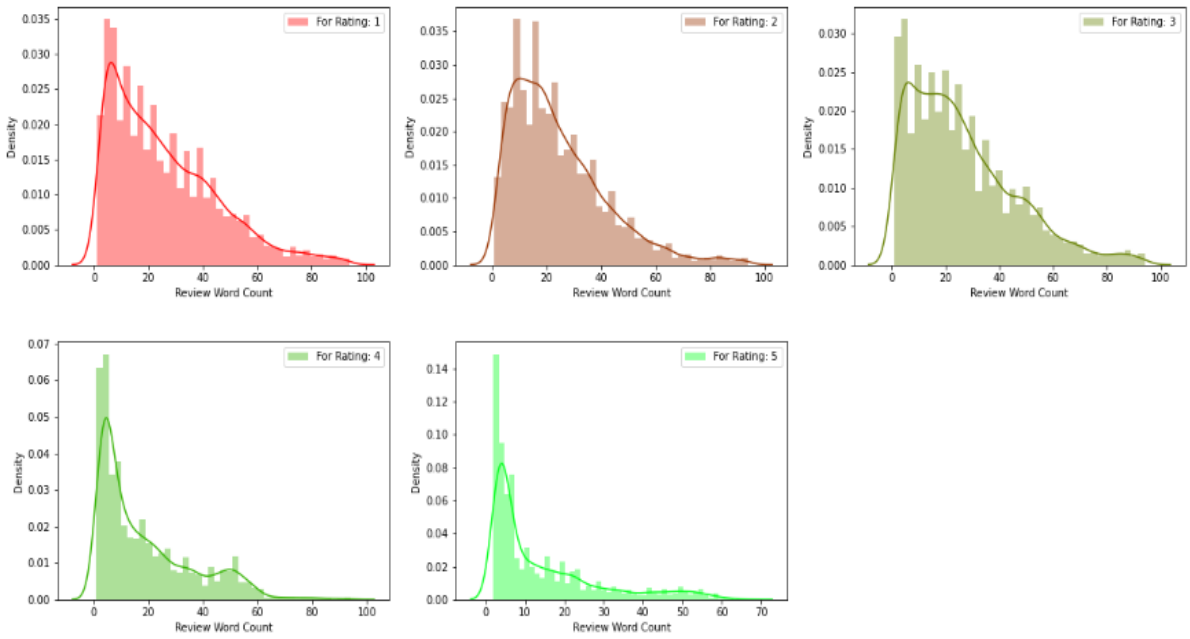


- As we can see that most of the words are accruing 0 to 10 times range.
- Word counts is highly right skewed.



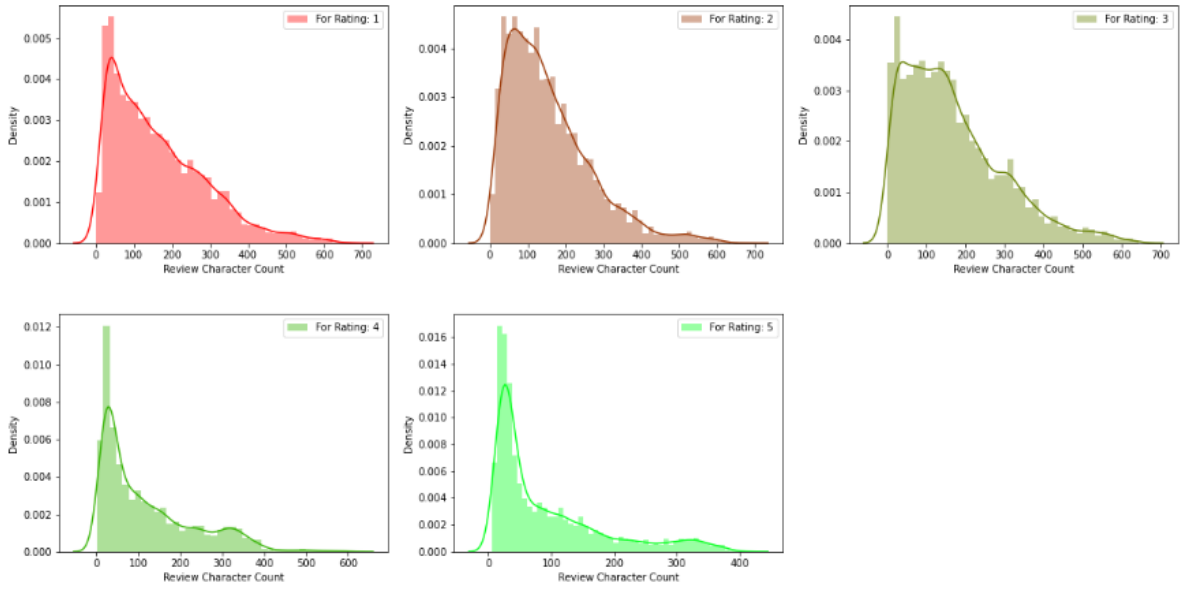
Above plot represents histogram for character count of review text, which is quite similar to the histogram of word count.

Checking review word count distribution for each rating:-



- We noticed that every type of ratings word count distribution is highly right skewed.
- Also we see that most of the word count falling in the range of 0 to 10.

Similarly Checking review character count distribution for each rating:-



- We noticed that for every type rating, the character count distribution is highly right skewed.
- Similarly most of character's count is falling in the range of 0 to 10.

Interpretation of the Results

Starting with univariate analysis, with the help of count plot it was found that the data consists of in equal amount for each rating (i.e., from 1 to 5). Moving further with the removal and replacement of certain terms (like punctuations, extra spaces, numbers, money symbols, smiley etc) as well as removal of stop words.

It was evident that the length of review text decreases by a large amount. This was also depicted by using distribution plots and histograms. With the help of word cloud, it was found that rating 1 consists of words like waste, money, slow, worst, issue, horrible etc, rating 2 consists of words like problem, issue, bad, poor, slow etc, rating 3 consists of word like problem, bad, issue, slow, life, average, nice etc, rating 4 consists of word like good, value, money, nice, performance, great, better, wonderful etc and rating 5 consists of words like excellent, must buy, great, perfect, super, awesome, mind blowing etc.

As we noticed that every model's training score showing greate as well as testing score. But after checking the Cross validation we face the reality of our models. Every model is extremely over fitted.

After Checking the Cross Validation we observe that `RandomForestClassifier` models showing highest accuracy among all of them.

Now, we will do Some tunning for this model. Let's Check if we increase their accuracy or not.

`HyperParameter Tunning:`

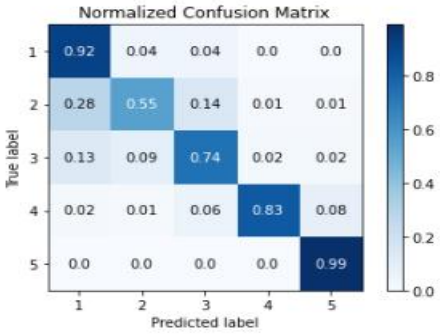
Random Forest Classifier:-

```
 ##Lets select the different parameters for tuning our best model (RandomForestClassifier)
grid_params = {'n_estimators':[100,200],
               'criterion':['gini','entropy'],
               'max_depth': [500,800],
               'bootstrap':[True,False]}

# Train the model with given parameters using GridSearchCV
GSCV =  GridSearchCV(RandomForestClassifier(), grid_params, cv=3, verbose=3,n_jobs=-1)
GSCV.fit(x_train, y_train)
```

```
print(GSCV.best_params_)
```

```
### Model Evaluation

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=15)

rf=RandomForestClassifier(n_estimators=200,criterion="gini",max_depth=800,bootstrap=False)

rf.fit(x_train,y_train)
y_pred=rf.predict(x_test)
print("\n\n")
print("**********************Testing Scores*****************************\n")
print("Accuracy score for testing is : ", accuracy_score(y_test,y_pred))
# Printing the classification report
print(f"\nCLASSIFICATION REPORT: \n {classification_report(y_test, y_pred)}")
# Printing the Confusion matrix
print(f"\nCONFUSION MATRIX: \n {confusion_matrix(y_test, y_pred)}")
```

```
**********Testing Scores**********

Accuracy score for testing is :  0.8548818385488184

CLASSIFICATION REPORT:
              precision    recall  f1-score   support

           1       0.75      0.92      0.83      2836
           2       0.76      0.55      0.64      1842
           3       0.77      0.74      0.75      2235
           4       0.96      0.83      0.89      1982
           5       0.95      0.99      0.97      4942

    accuracy                           0.85     13837
   macro avg       0.84      0.81      0.82     13837
weighted avg       0.86      0.85      0.85     13837


CONFUSION MATRIX:
 [[2609  103  104    9   11]
 [ 524 1018  265   10   25]
 [ 295  201 1649   36   54]
 [  39   18  117 1652  156]
 [   8    3   14   16 4901]]
```



Random Forest Accuracy is 85%

*CONCLUSION *

Key Findings and Conclusions of the Study

We observe that all the models are performing quite well in training model and testing as well. But after checking the cross validation we see that all the models are getting over fitted. In summary, we have tried two types of features. For this two type of features, we tried all the algorithms we mentioned in the model part including Naive Bayes, SVM, RF,LR, DT, LSTM. From the results, we can see that our accuracy on the test set is the best when we use RF on the first type of feature.

Though the random forest classifier model have shown best accuracy. So that is the reason we have selected it our best fit model as well.

Learning Outcomes of the Study in respect of Data Science

In this project we were able to learn various Natural Language Processing techniques like lemmatization, stemming, removal of Stop Words, etc. This project has demonstrated the importance of sampling effectively, modelling and

predicting data. Through different powerful tools of visualization, we were able to analyses and interpret different hidden insights about the data. The few challenges while working on this project are:

1. Imbalanced Dataset.
2. Lots of Text data.

The dataset was highly imbalanced so we balanced the dataset using smote technique. We converted text data into vectors with the help of TfidfVectorizer.

Limitations of this work and Scope for Future Work

One of the main reason our accuracy is not high enough is because of the data imbalance. We have given equal amount of class to our model for that we have to drop most of the data. This could be the reason of low accuracy. We will be looking into datasets obtained from E-commerce websites. This will provide better understanding of our sentiments based on demographics. And lastly we will try to work on this product to achieve a generalize form of this model.

For future we have to look into below mentioned points.

- Try to fetch data from different websites. If data is from different websites, it will help our model to remove the effect of over fitting.
- Try to 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.