Project Report On

“Rating Prediction Project”

Submitted By

Aamina Ruvaida

ACKNOWLEDGMENT

I sincerely thank to “Flip Robo Technologies” who let me to work on this wonderful project. Also thanked to Data Trained team who gives the guidance and right direction in the completion of project. This project has helped me to understand a lot and learned various thing while completing the project.

This project has helped me to understand how to perform text pre-processing using the NLP. Sincerely thanks to my parents who is the backbone for me and encourage me in all aspects of life.

References:

Text Preprocessing — NLP Basics. Text Preprocessing is the first step in… | by Nupur Kapur | Analytics Vidhya | Medium

Hyperparameter Tuning the Random Forest in Python | by Will Koehrsen | Towards Data Science

Opinion-Driven Matrix Factorization for Rating Prediction | SpringerLink

TABLE OF CONTENTS:

INTRODUCTION

• Business Problem Framing

- ❖ Rating prediction is a well-known recommendation task aiming to predict a user's rating for those items which were not rated yet by her. Predictions are computed from users' explicit feedback, their ratings provided on some items in the past.
- ❖ Another type of feedback are user reviews provided on items which implicitly express users' opinions on items. Recent studies indicate that opinions inferred from users' reviews on items are strong predictors of user's implicit feedback or even ratings and thus, should be utilized in computation.
- ❖ As far as we know, all the recent works on recommendation techniques utilizing opinions inferred from users' reviews are either focused on the item recommendation task or use only the opinion information, completely leaving users' ratings out of consideration.
- ❖ The approach proposed in this paper is filling this gap, providing a simple, personalized and scalable rating prediction framework utilizing both ratings provided by users and opinions inferred from their reviews.
- ❖ Experimental results provided on a dataset containing user ratings and reviews from the real-world Amazon Product Review Data show the effectiveness of the proposed framework.
- ❖ 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

- ❖ Compared to the traditional systems which mainly utilize user's rating history, review-based recommendation hopefully provides more relevant results to users. We introduce a review-based recommendation approach that obtains contextual information by mining user reviews.
- ❖ The proposed approach relates to features obtained by analysing textual reviews using methods developed in Natural Language Processing (NLP) and information retrieval discipline to compute a utility function over a

given item. An item utility is a measure that shows how much it is preferred according to user's current context.

- ❖ all the recent works on recommendation techniques utilizing opinions inferred from user's reviews are either focused on the item recommendation task or use only the opinion information, completely leaving user's ratings out of consideration.
- ❖ The approach proposed in this report is filling this gap, providing a simple, personalized and scalable rating prediction framework utilizing both ratings provided by users and opinions inferred from their reviews.
- ❖ Recommendation systems are an important unit in today's e-commerce applications, such as targeted advertising, personalized marketing and information retrieval. In recent years, the importance of contextual information has motivated generation of personalized recommendations according to the available contextual information of users.
- ❖ Before customer buys any product, they look into the ratings of the product. How many people has given the review, ratings? Whether the product is good or not. Then they will decide to buy the product or not. So, rating the product is very important in business. Most of business firm focused on customer service. Better the service more the customer. It also helps the business to understand what customer sees in the product, whether the customer likes the quality of product, cheaper price, long durable, offer on product etc. Based on this business can be focus in those area to give the better customer service.

• Review of Literature

- ❖ In real life, people's decision is often affected by friends' action or recommendation. How to utilize social information has been extensively studied. Yang et al.
- ❖ Propose the concept of "Trust Circles" in social network based on probabilistic matrix factorization. Jiang et al. propose another important factor, the individual preference. some websites do not always offer structured information, and all of these methods do not leverage user's unstructured information, reviews, explicit social networks information is not always available and it is difficult to provide a good prediction for each user. For this problem the sentiment factor term is used to improve social recommendation.
- ❖ The rapid development of Web 2.0 and e-commerce has led to a proliferation in the number of online user reviews. Online reviews contain a wealth of sentiment information that is important for many

decision-making processes, such as personal consumption decisions, commodity quality monitoring, and social opinion mining.

- ❖ Mining the sentiment and opinions that are contained in online reviews has become an important topic in natural language processing, machine learning, and Web mining.

• Motivation for the Problem Undertaken

- ❖ The dataset was provided by the Flip Robo Technologies. Models able to predict the user rating from the text review are critically important. Getting an overall sense of a textual review could in turn improve consumer experience. My motivation on this as this new data set that involves using the Natural Language Pre-processing technique used on the project. It helps me to work on removal of irrelevant words from the review.
- ❖ It is interesting to work on the project how customer gives reviews and ratings for the product. It is excited to build the model that predicts the ratings based on the review.
- ❖ Customer also shows their anger, frustration in the review if they didn't like the product of, they have expected. This kind of review will decrease the sell in business.
- ❖ The exposure to real world data and the opportunity to deploy my skillset in solving a real time problem has been the primary objective. Many product reviews are not accompanied by a scale rating system, consisting only of a textual evaluation.

Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the Problem

  - ❖ In this particular problem the Ratings can be 1, 2, 3, 4 or 5, which represents the likely ness of the product to the customer. So clearly it is a multi-classification problem and I have to use all classification algorithms while building the model.
  - ❖ We would perform one type of supervised learning algorithms: Classification. Here, we will only perform classification. Since there only 1 feature in the dataset, filtering the words is needed to prevent overfit. In order to determine the regularization parameter, throughout the project in classification part.
  - ❖ We would first remove email, phone number, web address, spaces and stops words etc. In order to further improve our models, we also performed TFID in order to convert the tokens from the train documents into vectors so that machine can do further processing.
  - ❖ I have used all the classification algorithms while building model then turned the best model and saved the best model.
  - ❖ Model Building Phase: After collecting the data, I built a machine learning model. Before model building, have done all data pre-processing steps. The complete life cycle of data science that I have used in this project are as follows:

- · Data Cleaning
- · Exploratory Data Analysis
- · Data Pre-processing
- · Model Building
- · Model Evaluation
- · Selecting the best model

- ## Data Sources and their formats

❖ The data set contains nearly 35,174 samples with 3 features. Since Ratings is my target column and it is a categorical column with 5 categories so this problem is a Multi Classification Problem. The Ratings can be 1, 2, 3, 4 or 5, The data set includes:

- Review_Title : Title of the Review.
- Review_Text : Text Content of the Review.
- Ratings : Ratings out of 5 stars.

```
#Loading the Scraped dataset
df=pd.read_csv('Amazon_Flipkart_Reviews.csv')
df
```

	Unnamed: 0	Review Title	Review Text	Ratings
0	0	Suitable for School kids	If you are a College student or a professional...	2.0 out of 5 stars
1	1	Misrepresentation on MS Office 2019 license - ...	Update after one month usage - MS Office 2019 ...	2.0 out of 5 stars
2	2	The sold me renewed laptop	It's look like renewed laptop because laptop c...	2.0 out of 5 stars
3	3	Amazon dupes with specification/ battery sucks	I had seen the specifications and bought...	2.0 out of 5 stars
4	4	Low Quality and Low Battery performance	Build Quality was Low, No match for the price....	2.0 out of 5 stars
...	...	...	...	...
35169	19392	Terrific purchase	Excellent monitor got it at great price of 720...	5
35170	19393	Worth the money	Good, but rate high when compared	4
35171	19394	Excellent	Super Q	5
35172	19395	Simply awesome	Super se uper	5
35173	19396	Awesome	Worthy Offer, monitor is good	5

35174 rows × 4 columns

❖ This project is more about exploration, feature engineering and classification, data pre-processing that can be done on this data. It includes multi classification of ratings.

• Data Pre-processing Done

❖ Data pre-processing is the process of converting raw data into a well readable format to be used by Machine Learning model. Data pre-processing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn; therefore, it is extremely important that we pre-process our data before feeding it into our model.

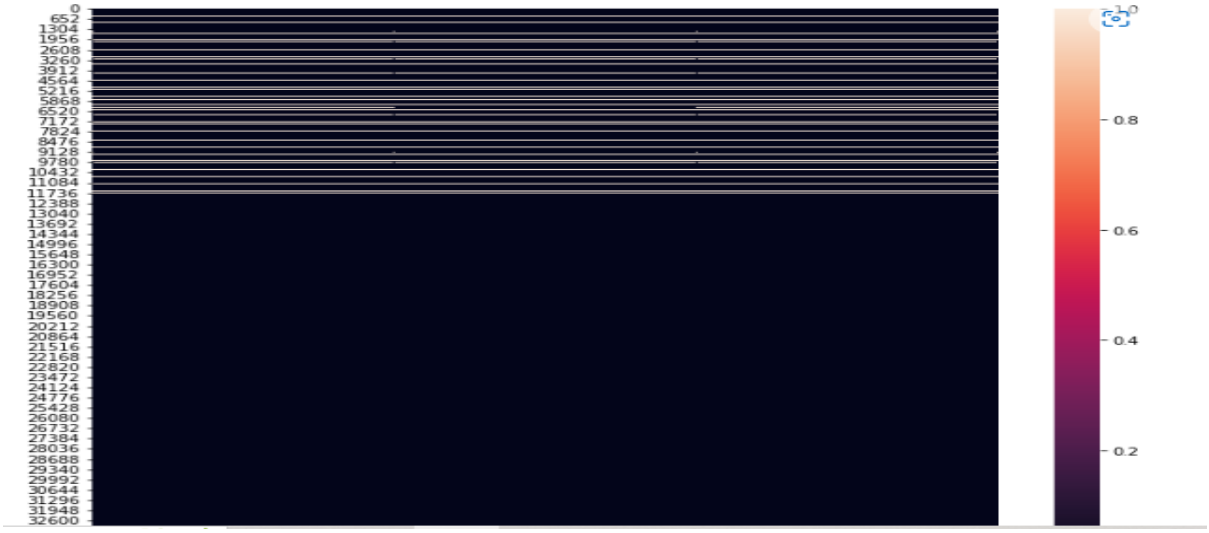
I have used following pre-processing steps:

✓ Importing necessary libraries and loading dataset as a data frame.
✓ Checked some statistical information like shape, number of unique values present, info, null values, value counts etc.
✓ Checked for null values and I replaced those null values using imputation method. And removed Unnamed: 0.
✓ Visualized each feature using seaborn and matplotlib libraries by plotting distribution plot and word cloud for each rating.

- ✓ Done text pre-processing techniques like Removing Punctuations and other special characters, Splitting the comments into individual words, Removing Stop Words, Stemming and Lemmatization.
- ✓ After getting a cleaned data used TF-IDF vectorizer. It'll help to transform the text data to feature vector which can be used as input in our 6 modelling. It is a common algorithm to transform text into numbers. It measures the originality of a word by comparing the frequency of appearance of a word in a document with the number of documents the words appear in.
- ✓ ü Balanced the data using SMOTE method

Before removing the null value

```
plt.figure(figsize=(12,10))
sns.heatmap(df.isnull())
plt.show()
```



Replace the null value with mode.

```
#Lets replace the nan with mode of each columns
df['Review Title']=df['Review Title'].fillna(df['Review Title'].mode()[0])
df['Review Text']=df['Review Text'].fillna(df['Review Text'].mode()[0])
df['Ratings']=df['Ratings'].fillna(df['Ratings'].mode()[0])
```

```
#Checking the null value after replaced with mode
df.isnull().sum()
```

```
Review Title    0
Review Text     0
Ratings         0
dtype: int64
```

```
plt.figure(figsize=(12,10))
sns.heatmap(df.isnull())
plt.show()
```

- **Data Inputs- Logic- Output Relationships**

❖ The dataset consists of 2 features with a label. The features are independent and label is dependent as our label varies the values(text) of our independent variable's changes.
❖ · I checked the distribution of skewness using distribution plots and used count plots to check the counts available in each column as a part of univariate analysis.
❖ · Got to know the frequently occurring and rare occurring word with the help of count plot. And was able to see the words in the Review text with reference to their ratings using word cloud.

Hardware and Software Requirements and Tools Used

- Tool: Jupyter Notebook

❖ Web-based interactive computing notebook Environment. Software Requirement
- The client environment used is Windows.
- Hardware Requirement: • CPU: 2 x 64-bit, 2.8 GHz, 8.00 GT/s CPUs or better.

❖ Memory: RAM size of 12
- Libraries: • Pandas: For reading CSV file, Converting dataset into a data frame, handling date datatype, and more. • Seaborn and matplotlib: For EDA and Visualization.

❖ import numpy as np: It is defined as a Python package used for performing the various numerical computations and processing of the multidimensional and single dimensional array elements. The calculations using Numpy arrays are faster than the normal Python array.
❖ import pandas as pd: Pandas is a Python library that is used for faster data analysis, data cleaning and data pre-processing. The data-frame term is coming from Pandas only.
❖ import matplotlib.pyplot as plt: Matplotlib and Seaborn acts as the backbone of data visualization through Python.
❖ Matplotlib: It is a Python library used for plotting graphs with the help of other libraries like Numpy and Pandas. It is a

powerful tool for visualizing data in Python. It is used for creating statical interferences and plotting 2D graphs of arrays.

- ❖ import seaborn as sns: Seaborn is also a Python library used for plotting graphs with the help of Matplotlib considered as a superset of the Matplotlib library. It helps in visualizing univariate and bivariate data.

```
#Importing the Libraries
import numpy as np
import pandas as pd

#Importing the Libraries for visualization
import matplotlib.pyplot as plt
import seaborn as sns

#Importing nltk Libraries
import nltk
from nltk.corpus import stopwords
import re
import string
from nltk.tokenize import word_tokenize
from nltk.corpus import wordnet
from nltk.stem.wordnet import WordNetLemmatizer

#Importing Libraries for Evaluation Metrics
from sklearn import metrics
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.metrics import roc_curve, roc_auc_score, hamming_loss, log_loss

#Importing Library to remove Outliers
from scipy.stats import zscore

#Importing Library for model building
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.naive_bayes import MultinomialNB

import warnings
warnings.filterwarnings('ignore')
```

Model/s Development and Evaluation

- **Identification of possible problem-solving approaches (methods)**

I have converted text into feature vectors using TF-IDF vectorizer and separated our feature and labels. Also, before building the model, I made sure that the input data is cleaned and scaled before it was fed into the machine learning models. Just making the Reviews more appropriate so that we'll get less word to process and get more accuracy. Removed extra spaces, converted email address into email keyword, and phone number etc. Tried to make Reviews small and more appropriate as much as possible.

```
#Let's apply the regular expression to make the contracted words to full form words, removing url and unwanted text
def decontracted(text):
    text = re.sub(r"won't", "will not", text)
    text = re.sub(r"don't", "do not", text)
    text = re.sub(r"can't", "can not", text)
    text = re.sub(r"im ", "i am", text)
    text = re.sub(r"yo ", "you ",text)
    text = re.sub(r"doesn't", "does not",text)
    text = re.sub(r"n\'t", " not", text)
    text = re.sub(r"\'re", " are", text)
    text = re.sub(r"\'s", " is", text)
    text = re.sub(r"\'d", " would", text)
    text = re.sub(r"\'ll", " will", text)
    text = re.sub(r"\'t", " not", text)
    text = re.sub(r"\'ve", " have", text)
    text = re.sub(r"\'m", " am", text)
    text = re.sub(r"<br>", " ", text)
    text = re.sub(r'http\S+', '', text) #removing urls
    return text
```

```
#Converting the Review to lower case
df['Review']=df['Review'].apply(lambda x: x.lower())
```

```
df['Review'].head() #We have converted to lower case
```

```
0    suitable for school kids if you are a college ...
1    misrepresentation on ms office 2019 license - ...
2    the sold me renewed laptop it's look like rene...
3    amazon dupes with specification/ battery sucks...
4    low quality and low battery performance build ...
Name: Review, dtype: object
```

```
#Noice removal
def scrub_words(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['Review']=df['Review'].apply(lambda x:scrub_words(x))
```

```
df['Review'][0]
```

```
'suitable school kids college student professional depends heavily laptop pretty much everyday laptop hangs often runs cannot i
nstall essential computer science software eclipse android studio laptop diesbrbri took laptop thinking good performance based
configurationbrbrbut someone wants attend online classes browse may go laptop'
```

- **Testing of Identified Approaches (Algorithms)**

❖ In this NLP based project we need to predict Ratings which is a multiclassification problem. I have converted the text into vectors using TFIDF vectorizer and separated our feature and labels then build the model using One Vs Rest Classifier. Among all the algorithms which I have used for this purpose I have chosen Random Forest Classifier as best suitable algorithm for our final model as it is performing well compared to other algorithms while evaluating with different metrics, I have used following algorithms and evaluated them

✓ Decision Tree Classifier
✓ Random Forest Classifier
✓ Ada Boost Classifier
✓ Gradient Boosting Classifier
✓ Logistic Regression
✓ Multinomial NB
✓ XGB Classifier
✓ SGD Classifier

❖ From all of these above models Random Forest Classifier was giving me good performance with less difference in accuracy score and cv score.

- **Key Metrics for success in solving problem under consideration**

  In this project I have used the Mean Squared Error, Mean Absolute Error, Hamming Loss, Classification Report, Confusion Matrix, Accuracy Score.

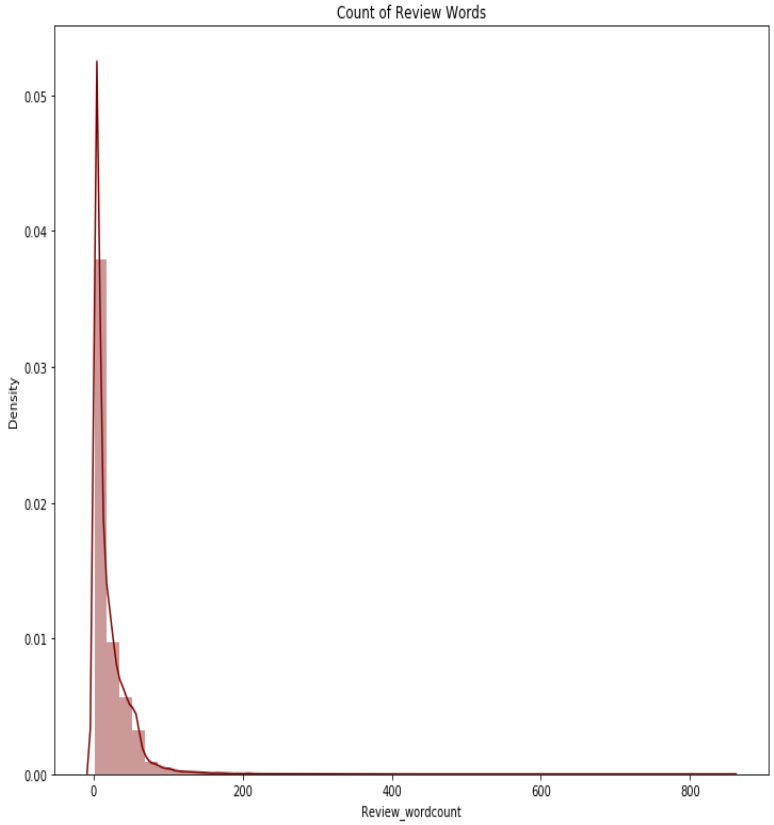
❖ **Precision** can be seen as a measure of quality; higher precision means that an algorithm returns more relevant results than irrelevant ones.
❖ **Recall** is used as a measure of quantity and high recall means that an algorithm returns most of the relevant results.

- **Accuracy score** is used when the True Positives and True negatives are more important. Accuracy can be used when the class distribution is similar.
- **F1-score** is used when the False Negatives and False Positives are crucial. While F1-score is a better metric when there are imbalanced classes.
- The **Hamming loss** is the **fraction of labels that are incorrectly predicted**. Read more in the User Guide. Ground truth (correct) labels. Predicted labels, as returned by a classifier. Sample weights. New in version 0.18. Return the average Hamming loss between element of y_true and y_pred.
- **Mean squared error (MSE) measures the amount of error in statistical models**. It assesses the average squared difference between the observed and predicted values. When a model has no error, the MSE equals zero. As model error increases, its value increases. The mean squared error is also known as the mean squared deviation (MSD).
- **Mean Absolute Error or MAE.** We know that an error basically is the absolute difference between the actual or true values and the values that are predicted.

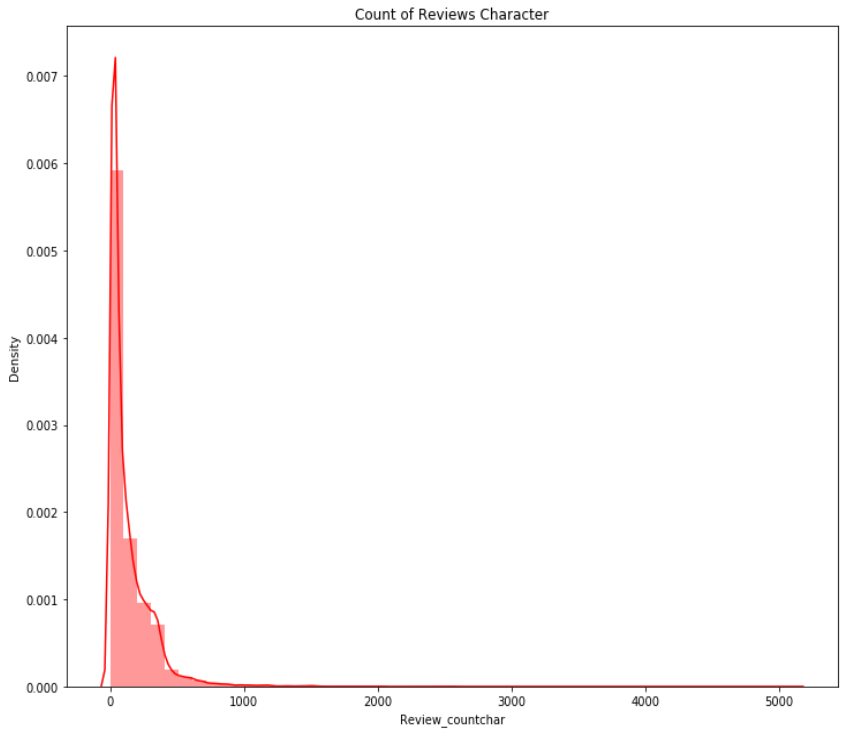
```
#Importing libraries for Evaluation Metrics
from sklearn import metrics
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.metrics import roc_curve, roc_auc_score, hamming_loss, log_loss
```

- **Visualizations**

➢ Distribution plot for Word count:



➢ Distribution plot for Character count:

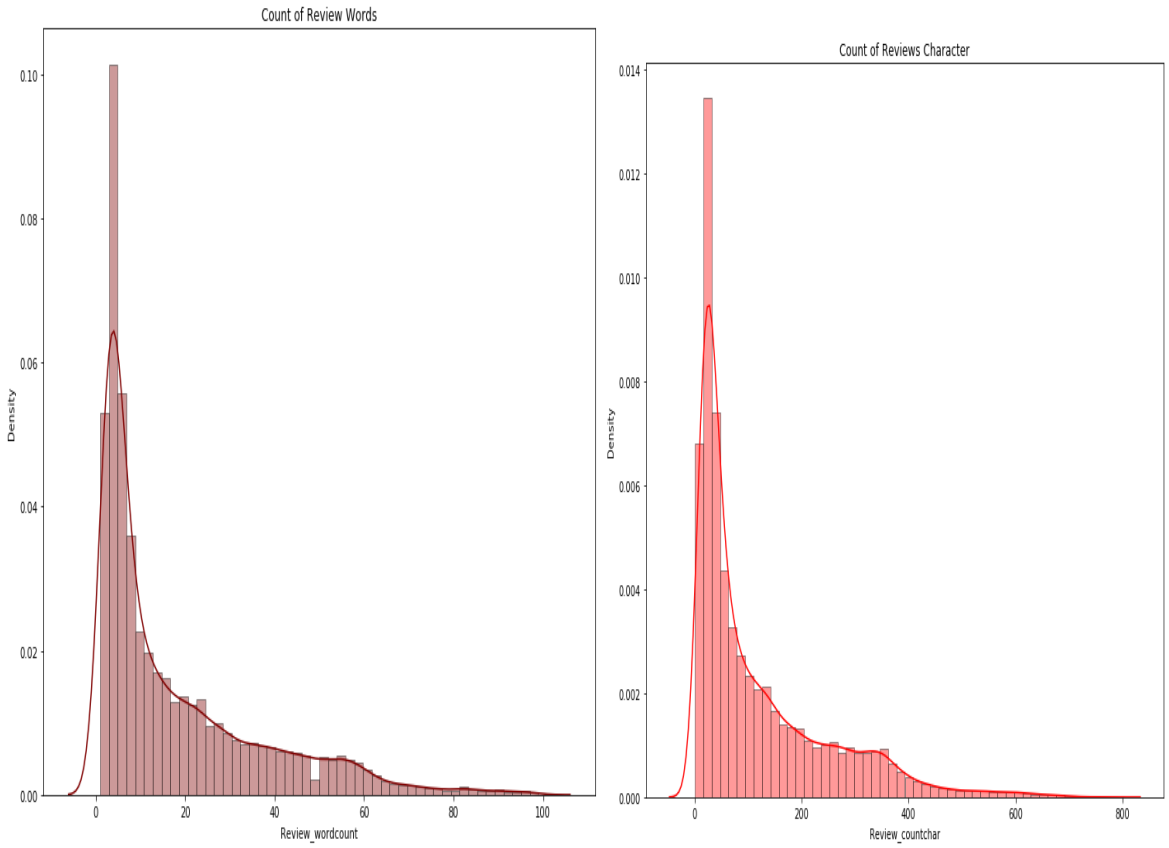


❖ By observing the histogram, we can clearly see that most of our text is having the number of words in the range of 0 to 200, But

some of the reviews are too lengthy which may act like outliers in our data.

- ❖ Above plot represents histogram for character count of Review text, which is quite similar to the histogram of word count.
- ❖ As we know that some of the reviews are too lengthy, so i have to treat them as outliers and remove them using zscore method. After removing the outliers, the word count and character count looks as below.

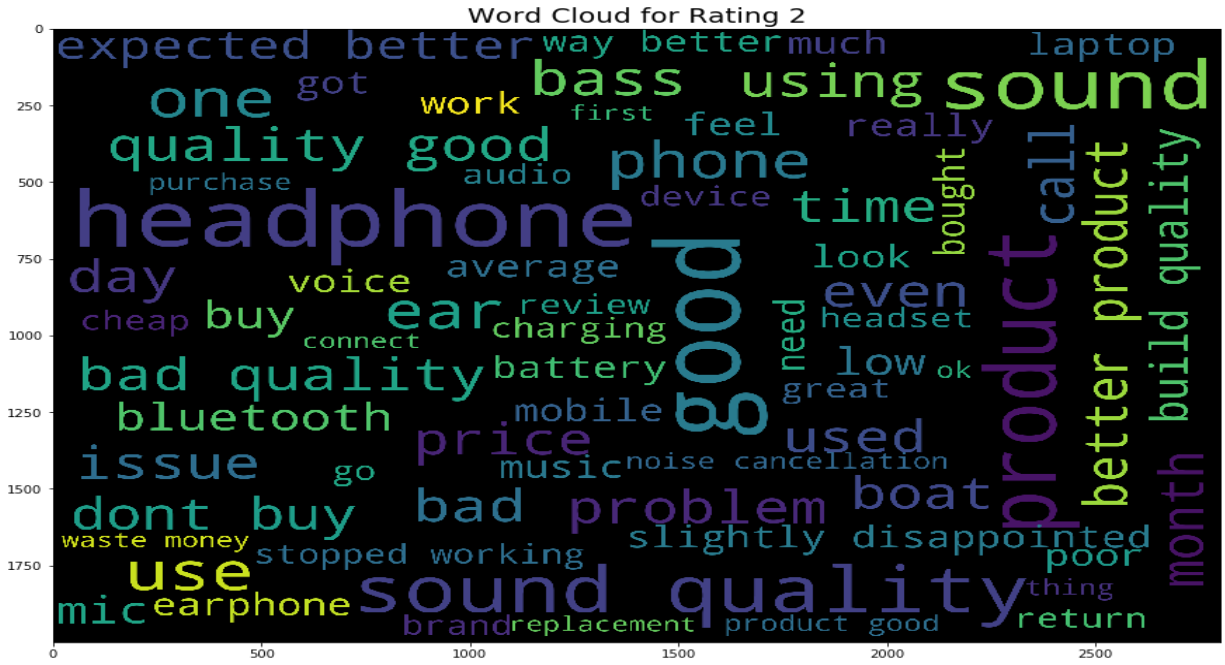
<u>Distribution plot after removing the outliers:</u>



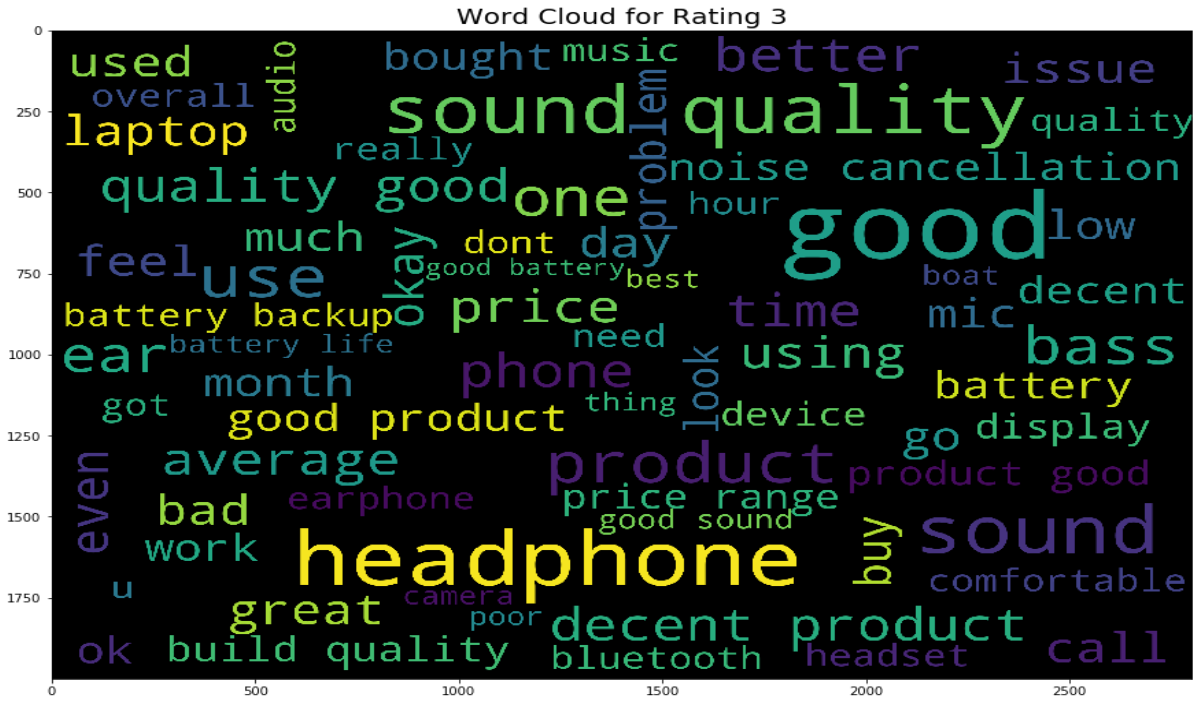
Word Cloud for Ratings:

Rating 1:

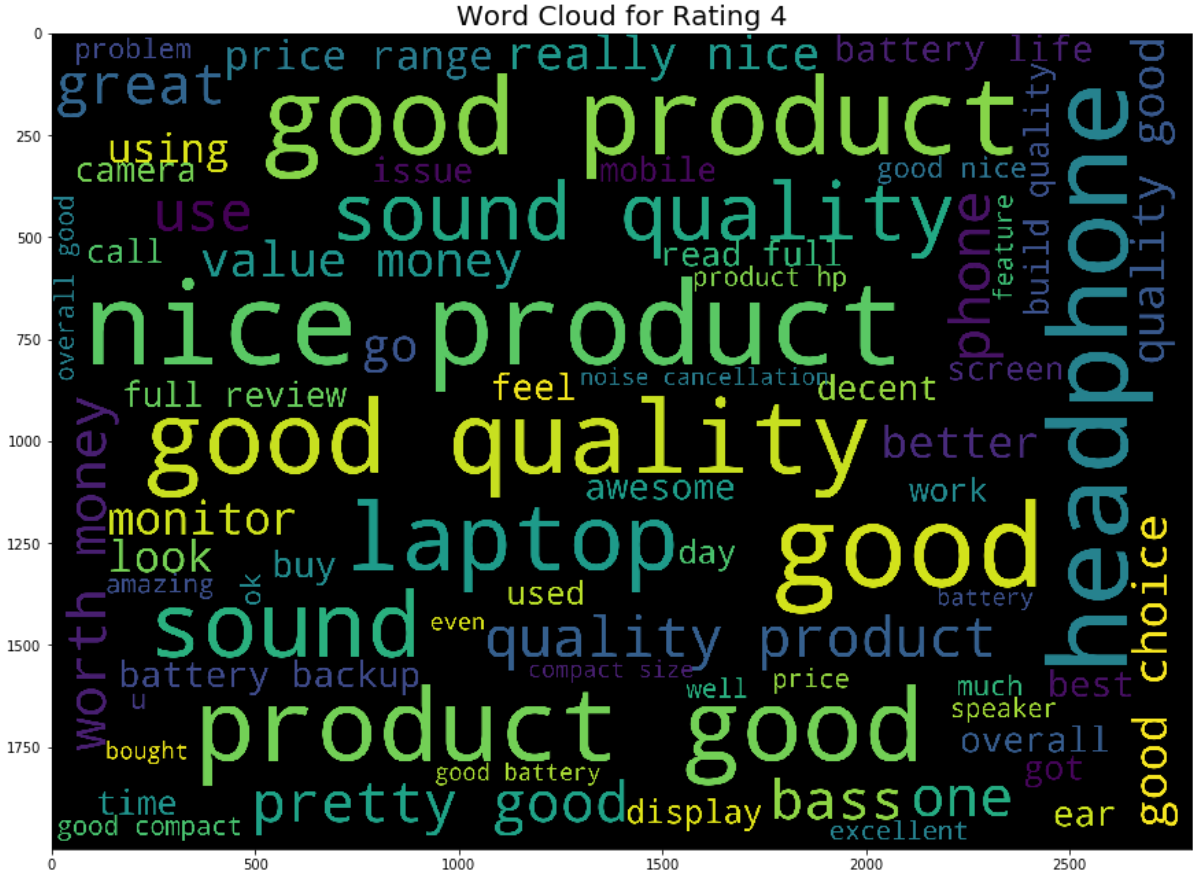
Rating 2:



Rating 3:



Rating 4:



Rating 5:

- **Run and evaluate selected models**

  I have used 8 classification algorithms. First, I have created 8 different classification algorithms and are appended in the variable models. Followed by TFIDF vectorization and data balancing. Then, ran a for loop which contained the accuracy of the models along with different evaluation metrics.

```
#Importing library for model building
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.naive_bayes import MultinomialNB
```

```
#Initiating the model
dr=DecisionTreeClassifier()
rf=RandomForestClassifier()
ada=AdaBoostClassifier()
gbc=GradientBoostingClassifier()
lg=LogisticRegression()
mnb=MultinomialNB()
xgb=XGBClassifier()
sgd=SGDClassifier()
```

```
#Defining the model
def algorithm(model):
    print("*"*30+model.__class__.__name__+"*"*30)
    model.fit(x_train,y_train)
    y_pred=model.predict(x_test)
    print("Accuracy Score:",accuracy_score(y_test,y_pred)*100,'\n')
    print("Classification report:\n",classification_report(y_test,y_pred))
    print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred),'\n')
    print("Mean Squared Error:",mean_squared_error(y_test,y_pred))
    print("Mean Absolute Error:",mean_absolute_error(y_test,y_pred))
    print("Hamming Loss:",hamming_loss(y_test,y_pred))
    print("*"*100)
    print("\n\n")
```

```
for model in [dr,rf,ada,gbc,lg,mnb,xgb,sgd]:
    algorithm(model)
```

Decision Tree Classifier:

```
******************************DecisionTreeClassifier******************************
Accuracy Score: 83.02195685670262

Classification report:
               precision    recall  f1-score   support

           1       0.83      0.85      0.84      1641
           2       0.64      0.57      0.60       768
           3       0.62      0.65      0.64       955
           4       0.76      0.74      0.75      1839
           5       0.92      0.93      0.92      5181

    accuracy                           0.83     10384
   macro avg       0.75      0.75      0.75     10384
weighted avg       0.83      0.83      0.83     10384

Confusion Matrix:
 [[1396   91   81   32   41]
 [ 143  439   85   61   40]
 [  62   80  621  105   87]
 [  53   45  120 1367  254]
 [  30   36   92  225 4798]]

Mean Squared Error: 0.5200308166409862
Mean Absolute Error: 0.26251926040061635
Hamming Loss: 0.1697804314329738
****************************************************************************************************
```

In Decision Tree Classifier we got the accuracy score 83% and we can see above the classification report, confusion matrix, mean squared error, mean absolute error, hamming loss values.

Random Forest Classifier:

```
*****************************RandomForestClassifier*****************************
Accuracy Score: 85.67026194144837

Classification report:
              precision    recall  f1-score   support

           1       0.79      0.93      0.86      1641
           2       0.83      0.51      0.63       768
           3       0.75      0.62      0.68       955
           4       0.86      0.73      0.79      1839
           5       0.90      0.97      0.93      5181

    accuracy                           0.86     10384
   macro avg       0.83      0.75      0.78     10384
weighted avg       0.85      0.86      0.85     10384

Confusion Matrix:
 [[1532   23   30   25   31]
 [ 211  391   70   33   63]
 [ 114   34  589   76  142]
 [  53   15   80 1335  356]
 [  20    6   21   85 5049]]

Mean Squared Error: 0.4327812018489985
Mean Absolute Error: 0.2205315870570108
Hamming Loss: 0.14329738058551617
*********************************************************************************************
```

In Random Forest Classifier we got the accuracy score 86% and we can see above the classification report, confusion matrix, mean squared error, mean absolute error, hamming loss values.

Ada Boost Classifier:

```
*****************************AdaBoostClassifier*****************************
Accuracy Score: 76.9453004622496

Classification report:
              precision    recall  f1-score   support

           1       0.64      0.91      0.75      1641
           2       0.65      0.28      0.39       768
           3       0.76      0.31      0.44       955
           4       0.60      0.73      0.66      1839
           5       0.92      0.90      0.91      5181

    accuracy                           0.77     10384
   macro avg       0.71      0.62      0.63     10384
weighted avg       0.78      0.77      0.76     10384

Confusion Matrix:
 [[1493   43    8   77   20]
 [ 366  217   17  140   28]
 [ 208   51  293  346   57]
 [ 125   16   43 1337  318]
 [ 157    8   25  341 4650]]

Mean Squared Error: 0.800751155624037
Mean Absolute Error: 0.3712442218798151
Hamming Loss: 0.23054699537750384
*********************************************************************************************
```

In Ada Boost Classifier we got the accuracy score 77% and we can see above the classification report, confusion matrix, mean squared error, mean absolute error, hamming loss values.

Gradient Boosting Classifier:

```
*****************************GradientBoostingClassifier*****************************
Accuracy Score: 79.25654853620955

Classification report:
              precision    recall  f1-score   support

           1       0.84      0.83      0.84      1641
           2       0.73      0.35      0.47       768
           3       0.67      0.46      0.55       955
           4       0.82      0.64      0.71      1839
           5       0.79      0.96      0.87      5181

    accuracy                           0.79     10384
   macro avg       0.77      0.65      0.69     10384
weighted avg       0.79      0.79      0.78     10384

Confusion Matrix:
 [[1360   38   32   15  196]
 [ 161  266   65   37  239]
 [  63   44  439   78  331]
 [  21   12   84 1169  553]
 [  10    6   34  135 4996]]

Mean Squared Error: 0.8685477657935285
Mean Absolute Error: 0.37008859784283515
Hamming Loss: 0.20743451463790447
*********************************************************************************************
```

In Gradient Boosting Classifier we got the accuracy score 79% and we can see above the classification report, confusion matrix, mean squared error, mean absolute error, hamming loss values.

Logistic Regression:

```
*****************************LogisticRegression*****************************
Accuracy Score: 85.59322033898306

Classification report:
              precision    recall  f1-score   support

           1       0.85      0.90      0.87      1641
           2       0.75      0.54      0.63       768
           3       0.70      0.63      0.66       955
           4       0.81      0.77      0.79      1839
           5       0.91      0.96      0.93      5181

    accuracy                           0.86     10384
   macro avg       0.80      0.76      0.78     10384
weighted avg       0.85      0.86      0.85     10384

Confusion Matrix:
 [[1480   54   44   27   36]
 [ 166  412   90   38   62]
 [  63   60  600  129  103]
 [  28   18   85 1407  301]
 [  12    2   34  144 4989]]

Mean Squared Error: 0.39175654853620956
Mean Absolute Error: 0.2097457627118644
Hamming Loss: 0.1440677966101695
*********************************************************************************************
```

In Logistic Regression we got the accuracy score 86% and we can see above the classification report, confusion matrix, mean squared error, mean absolute error, hamming loss values.

Multinomial NB:

```
*****************************MultinomialNB*****************************
Accuracy Score: 80.44106317411402

Classification report:
              precision    recall  f1-score   support

           1       0.79      0.90      0.84      1641
           2       0.94      0.33      0.49       768
           3       0.80      0.41      0.54       955
           4       0.76      0.64      0.69      1839
           5       0.81      0.98      0.89      5181

    accuracy                           0.80     10384
   macro avg       0.82      0.65      0.69     10384
weighted avg       0.81      0.80      0.79     10384

Confusion Matrix:
 [[1474    7   15   30  115]
 [ 234  256   45   86  147]
 [ 101    7  392  164  291]
 [  36    1   25 1175  602]
 [  14    1   14   96 5056]]

Mean Squared Error: 0.6935670261941448
Mean Absolute Error: 0.3229969183359014
Hamming Loss: 0.19558936825885978
*************************************************************************************************
```

In Multinomial NB we got the accuracy score 80% and we can see above the classification report, confusion matrix, mean squared error, mean absolute error, hamming loss values.

XGB Classifier:

```
*****************************XGBClassifier*****************************
[12:40:07] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly s
et eval_metric if you'd like to restore the old behavior.
Accuracy Score: 84.66872110939909

Classification report:
              precision    recall  f1-score   support

           1       0.82      0.90      0.86      1641
           2       0.67      0.52      0.59       768
           3       0.68      0.65      0.66       955
           4       0.78      0.76      0.77      1839
           5       0.93      0.95      0.94      5181

    accuracy                           0.85     10384
   macro avg       0.78      0.75      0.76     10384
weighted avg       0.84      0.85      0.84     10384

Confusion Matrix:
 [[1472   69   50   26   24]
 [ 177  400  103   50   38]
 [  75   81  619  114   66]
 [  42   29  105 1395  268]
 [  25   14   39  197 4906]]

Mean Squared Error: 0.4058166409861325
Mean Absolute Error: 0.22033898305084745
Hamming Loss: 0.15331278890600925
*************************************************************************************************
```

In XGB Classifier we got the accuracy score 85% and we can see above the classification report, confusion matrix, mean squared error, mean absolute error, hamming loss values.

SGD Classifier:

```
*****************************SGDClassifier*****************************
Accuracy Score: 85.76656394453005

Classification report:
              precision    recall  f1-score   support

           1       0.82      0.92      0.87      1641
           2       0.75      0.53      0.62       768
           3       0.73      0.62      0.67       955
           4       0.82      0.76      0.79      1839
           5       0.91      0.97      0.94      5181

    accuracy                           0.86     10384
   macro avg       0.81      0.76      0.78     10384
weighted avg       0.85      0.86      0.85     10384

Confusion Matrix:
 [[1504   46   33   27   31]
 [ 192  406   75   39   56]
 [  84   64  589  113  105]
 [  31   22   82 1398  306]
 [  20    2   29  121 5009]]

Mean Squared Error: 0.39551232665639446
Mean Absolute Error: 0.2094568567026194
Hamming Loss: 0.14233436055469953
************************************************************************************************************
```

In SGD Classifier we got the accuracy score 86% and we can see above the classification report, confusion matrix, mean squared error, mean absolute error, hamming loss values.

Cross Validation Score:

```
def cross_val(CV):
    print("*"*30+CV.__class__.__name__+"*"*30)
    print("Cross Validation Score :",cross_val_score(CV, x,y,cv=3).mean()*100)
    print("\n")
```

```
for CV in [dr,rf,ada,gbc,lg,mnb,xgb,sgd]:
    cross_val(CV)
```

```
*****************************DecisionTreeClassifier*****************************
Cross Validation Score : 88.3329146780541


*****************************RandomForestClassifier*****************************
Cross Validation Score : 95.86368584107846


*****************************AdaBoostClassifier*****************************
Cross Validation Score : 68.70635518713891


*****************************GradientBoostingClassifier*****************************
Cross Validation Score : 83.16335928996065


*****************************LogisticRegression*****************************
Cross Validation Score : 93.5828518797622


*****************************MultinomialNB*****************************
Cross Validation Score : 90.69747969521896
```

All the model gives the good cross validation score. I will choose Random Forest Classifier as it performed well and gives good accuracy score and cross validation score as compare with other models.

Hyper Parameter Tuning:

```
grid_params={
    'n_estimators':[100,150],
    'criterion':['gini','entropy'],
    'max_depth':range(2,10,3),
    'min_samples_leaf':range(1,10,3),
    'min_samples_split':range(2,10,3),
    'max_features':['auto','log2']

}
```

```
grid_search=GridSearchCV(estimator=rf,param_grid=grid_params,cv=3,n_jobs=-1,verbose=3)
```

```
grid_search.fit(x_train,y_train)
```

```
Fitting 3 folds for each of 216 candidates, totalling 648 fits

GridSearchCV(cv=3, estimator=RandomForestClassifier(), n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': range(2, 10, 3),
                         'max_features': ['auto', 'log2'],
                         'min_samples_leaf': range(1, 10, 3),
                         'min_samples_split': range(2, 10, 3),
                         'n_estimators': [100, 150]},
             verbose=3)
```

```
grid_search.best_params_
```

```
{'criterion': 'gini',
 'max_depth': 8,
 'max_features': 'auto',
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'n_estimators': 150}
```

After tuning the parameter for our model, it gives an 86% accuracy in Random Forest Classifier model. It is good score.

```
print("Accuracy Score:",accuracy_score(y_test,pred)*100,'\n')
print("Classification report:\n",classification_report(y_test,pred))
print("Confusion Matrix:\n",confusion_matrix(y_test,pred),'\n')
print("Mean Squared Error:",mean_squared_error(y_test,pred))
print("Mean Absolute Error:",mean_absolute_error(y_test,pred))
print("Hamming Loss:",hamming_loss(y_test,pred))
```

```
Accuracy Score: 85.50654853620955

Classification report:
              precision    recall  f1-score   support

           1       0.80      0.93      0.86      1641
           2       0.87      0.49      0.63       768
           3       0.74      0.63      0.68       955
           4       0.86      0.72      0.79      1839
           5       0.89      0.97      0.93      5181

    accuracy                           0.86     10384
   macro avg       0.83      0.75      0.78     10384
weighted avg       0.85      0.86      0.85     10384

Confusion Matrix:
 [[1525   25   33   24   34]
 [ 208  380   69   34   77]
 [ 120   18  602   61  154]
 [  42   11   88 1323  375]
 [  19    2   22   89 5049]]

Mean Squared Error: 0.4412557781201849
Mean Absolute Error: 0.22419106317411402
Hamming Loss: 0.14493451463790447
```

- **Interpretation of the Results**

Saving the Final Model:

```
import pickle
```

```
filename='RatingPrediction.pickle'
```

```
pickle.dump(model,open(filename,'wb'))
```

```
loaded_model=pickle.load(open(filename,'rb'))
loaded_model
```

```
RandomForestClassifier(n_estimators=150)
```

```
prediction=loaded_model.predict(x_test)
prediction
```

```
array([5, 4, 5, ..., 1, 4, 5])
```

```
df=pd.DataFrame([loaded_model.predict(x_test)[:],y_test[:]],index=['Predicted','Actual'])
df
```

	0	1	2	3	4	5	6	7	8	9	...	10374	10375	10376	10377	10378	10379	10380	10381	10382	10383
Predicted	5	4	5	4	3	5	5	4	5	3	...	1	2	5	4	3	5	5	1	4	5
Actual	5	4	5	4	3	5	5	4	5	3	...	1	2	4	4	2	5	4	1	4	5

2 rows × 10384 columns

Predicted the value for Ratings. The Predicted and Actual Value is almost similar. As our model has performed well.

CONCLUSION

- **Key Findings and Conclusions of the Study**

- I have collected the data using the web scraping tool selenium from Amazon and Flipkart website.
- Loaded the dataset and analyse the data.
- Data set contains the null value, replace the null value with mode. Done the data cleaning, removed the column Unnamed: 0 from the dataset.
- Using the nltk tool done the text pre-processing, removed the unnecessary word, punctuation, stop words, phone number, url from the reviews.
- Checked the count for word and character from review.

- Using the distribution list, we found that there is a outliers present. Removed the outliers using the zscore.
- Used various visualization tool to analyse the data on the scale of rating 1 to 5.
- Converted the text data into vector using the TDIDF Vectorizer.
- Balanced the data using the over sampling technique SMOTE.
- Splitted the data into training and testing.
- Used various machine learning model to analyse and check the accuracy score and its metrics. Chosen the best model that performed well and gives a good accuracy score. In our project Random Forest Classifier gives a good accuracy score 86%.
- Then done the hyper parameter tuning for our final model. Finally saved the model using the pickle and predicted the ratings value.
- The predicted and actual value is almost similar. As our model has performed well.

• Learning Outcomes of the Study in respect of Data Science

- I have scrapped the reviews and ratings of different technical products from flipkart.com and amazon.in websites. Improvement in computing technology has made it possible to examine social information that cannot previously be captured, processed and analysed.
- New analytical techniques (NLP) of machine learning can be used in property research. The power of visualization has helped us in understanding the data by graphical representation it has made me to understand what data is trying to say. Data cleaning is one of the most important steps remove unrealistic values, punctuations, URLs, email address and stop words.
- This study is an exploratory attempt to use 6 machine learning algorithms in estimating Rating, and then compare their results.
- To conclude, the application of NLP in Rating classification is still at an early stage.

- We hope this study has moved a small step ahead in providing some methodological and empirical contributions to crediting institutes, and presenting an alternative approach to the valuation of Ratings.

• Limitations of this work and Scope for Future Work

- As we know the content of text in reviews is totally depends on the reviewer and they may rate differently which is totally depends on that particular person.
- So, it is difficult to predict ratings based on the reviews with higher accuracies. Still, we can improve our accuracy by fetching more data and by doing extensive hyperparameter tuning.
- While we couldn't reach out goal of maximum accuracy in Ratings prediction project, we did end up creating a system that can with some improvement and deep learning algorithms get very close to that goal.
- As with any project there is room for improvement here. The very nature of this project allows for multiple algorithms to be integrated together as modules and their results can be combined to increase the accuracy of the final result.
- This model can further be improved with the addition of more algorithms into it. However, the output of these algorithms needs to be in the same format as the others.
- Once that condition is satisfied, the modules are easy to add as done in the code. This provides a great degree of modularity and versatility to the project.