FLIP ROBO

Project Report On
Ratings Prediction

Submitted By: OLIVER RAMAN | Internship 26

ACKNOWLEDGEMENT

The following research papers helped me understand the importance of ratings, the various factors affecting the rating of a particular product, different types of ratings and their meaning & finally, helped me in my model building & predictions:

1."Prediction of rating based on review text of Yelp reviews"- Sasank Channapragada, Ruchika Shivaswamy

In this project, we aim to predict the rating of businesses listed in the Yelp dataset based on review text. We also intend to classify the reviews as funny, useful or cool, the metric used by Yelp to evaluate user reviews. Linear regression and different classification techniques such as Na¨ıve Bayes' Classification and Support Vector Machines were used for different features.

2. "Review-Based Rating Prediction" - Tal Hadad

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. Compared to the traditional systems which mainly utilize users' rating history, review-based recommendations hopefully provide 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 analyzing 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 the user's current context.

3. "Rating Prediction Based on Social Sentiment From Textual Reviews" - Xiaojiang Lei, Guoshuai Zhao, Xueming Qian

In recent years, we have witnessed a flourish of review websites. It presents a great opportunity to share our viewpoints for various products we purchase. However, at the same time we face the information overloading problem. How to mine valuable information from reviews to understand a user's preferences and make an accurate recommendation is crucial. Traditional recommender systems (RS) consider some factors, such as user's purchase records, product category, geographic location, and other social network factors. In this work, we propose a sentiment-based rating prediction method (RPS) to improve prediction accuracy in recommender systems. In particular, the proposed method is based on probabilistic matrix factorization. We conduct a performance evaluation of the three sentimental factors on a real-world dataset collected from Yelp. Our experimental results show the sentiment can well characterize user preferences, which help to improve the recommendation performance.

4. "Prediction of Reviews Rating: A Survey of Methods, Techniques and Hybrid Architectures" - Alaa Alqahtani, Abdulmalik Alsalman, Fouzi Harrag

Sentiment Analysis is the process of automatically identifying opin-ions expressed in text on certain subjects. The accuracy of sentiment analysis has a direct effect on decision mak-ing in both business and government. Also, recommender systems are one of the hot topics in the field of big data,especially with the development and growth of the WorldWide Web.This paper presents a survey covering the methods in the field of reviews rating prediction. We also propose a hybrid architecture that combines techniques from sen-timent analysis and recommender systems fields, to develop a solution that has the ability to predict rating of products while taking into consideration the analysis of sentiments included in the products reviews.

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 him/her. Predictions are computed from users' explicit feedback, i.e. 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.
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 ratings. So we have to build an application which can predict the rating by seeing the review.

Conceptual Background of the Domain Problem

The rise in E-commerce has brought a significant rise in the importance of customer reviews. There are hundreds of review sites online and massive amounts of reviews for every product. Customers have changed their way of shopping and according to a recent survey, 70 percent of customers say that they use rating filters to filter out low rated items in their searches. The ability to successfully decide whether a review will be helpful to other customers and thus give the product more exposure is vital to companies that support these reviews, companies like Google, Amazon and Yelp!.

There are two main methods to approach this problem. The first one is based on review text content analysis and uses the principles of natural language process (the NLP method). This method lacks the insights that can be drawn from the relationship between customers and items. The second one is based on recommender systems, specifically on collaborative filtering, and focuses on the reviewer's point of view.

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. Compared to the traditional systems which mainly utilize user's rating history, review-based recommendation hopefully provide more relevant results to users. We introduce a review-based recommendation approach that obtains contextual information by mining user reviews.

Review of Literature

In Alshari, E.Azman, A.Mustapha, N.Doraisamy,S.Alksher, M. (2016), the authors used the combination of sentiment analysis with information retrieval to predict the rating ofAmazon comments. Vector Space Model (VSM) was applied as a supervised classifier. They compared it with the combination of VSM with sentiment analysis. TheLexical dictionary approach was used as sentiment analy-sis with the VSM. The obtained result shows that the usage of sentiment analysis has a positive effect on the performance of the classifier in their rating prediction.

Lei, X., Qian, X. (2015). The authors proposed a rating predictor that depends on both: the sentiment of the product/service review and the service reputation into their recommender system. To solve the cold starting problem in the similar-ity calculation, they used an item called 'virtual friend'.

In Zhang, Y Lai, G. Zhang, M.Zhang, Y.Liu, Y, and Ma, S. (2014), they proposed the Explicit Factor Model (EFM)that helps in generating the recommendation. As a first step, they used the aspect approach to extract the as-pect of the reviews. After that, the recommendation sys-tem will recommend or not recommend the products to the users with a short explanation for that. They did their result under two conditions: the offline experiment andonline experiment. And their evaluation is categorized under two sides: the rating prediction and the top-k rec-ommendation. For the offline experiment, the obtained result outperformed the baseline algorithms. But in the online experiment, the result is varying due to the behav-ior of the purchaser.

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, i.e. 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.

Yang et al. propose the concept of "Trust Circles" in social networks based on probabilistic matrix factorization.

Motivation for the Problem Undertaken

The exposure to real world data and the opportunity to use my skills in solving a real time problem has been the main motivation for the problem undertaken.

Many product reviews are not accompanied by a standard scale rating system, and consist only of a textual evaluation. This is one such case, it becomes daunting and time-consuming to compare different products in order to eventually make a choice between them. Therefore, models able to predict the user rating from the text review are critically important.

This is my 5th internship project and the fourth one in which I have to build a machine learning model. Also, we had to scrape the reviews for the model using various web scraping techniques & data cleaning techniques.

ANALYTICAL PROBLEM FRAMING

Mathematical/Analytical Modeling of the Problem

In this particular problem the Ratings are 1, 2, 3, 4 or 5, which represents the like or the satisfaction of the product derived by the customer. Clearly it is a multi classification problem and we have to use all classification algorithms while building the model.

We would perform one type of supervised learning algorithms: Classification. Since there is 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 the classification part, we would first remove email, phone number, web address, spaces and stop 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 the model can do further processing.

I have used all the classification algorithms while building models, then tuned the best model and saved the best model.

Data Sources & their formats

The data set contains 21661 rows and 3 columns. 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, which represents the like or the satisfaction of the product derived by the customer

The data set includes:

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

We need to build a model that can predict Ratings of the reviewer.

Data preprocessing done

Data preprocessing is the process of converting raw data into a well readable format to be used by Machine Learning models. Data preprocessing 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.

- As a first step I have scrapped the required review & rating data using selenium from flipkart.com & amazon.in websites. And saved the data frame in csv format.
- Next, I have imported all necessary libraries and loaded the 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 methods. And removed the column Unnamed: 0.
- Did some text processing using techniques like Removing Punctuations and other special characters, Splitting the comments into individual words, Removing Stop Words, Stemming and Lemmatization
- Visualized each feature using seaborn and matplotlib libraries by plotting distribution plot and word cloud for each ratings.
- Removed some outliers using the z-score method.
- After getting the data cleaned, I used a TF-IDF vectorizer. It'll help to transform the text data to feature vectors which can be used as input in our model building. 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. Mathematically, TF-IDF = TF(t*d)*IDF(t,d)
- Balanced the data using the SMOTE method.
- Lastly proceeded with model building with classification algorithms.

Data Inputs- Logic- Output Relationships

The dataset consists of 2 features with a target variable. The features are independent and the target is dependent.

- I checked the distribution of skewness using dist plots and used count plots to check the counts available in each column as a part of univariate analysis.
- Checked the frequently occuring and rarely occurring words 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 & Tools used

To build the machine learning projects it is important to have the following hardware and software requirements and tools.

<u>Hardware required:</u>

- Processor: core i5 or above
- RAM: 8 GB or above
- ROM/SSD: 250 GB or above

Software required:
- Anaconda - language used Python 3

Libraries Used:
- import pandas as pd
- import numpy as np
- import seaborn as sns
- import matplotlib.pyplot as plt
- import nltk
- from nltk.corpus import stopwords
- import re
- import string
- from nltk import FreqDist
- from nltk.tokenize import word_tokenize
- from nltk.stem.wordnet import WordNetLemmatizer
- from nltk.corpus import wordnet
- from nltk import FreqDist
- from scipy import stats
- from scipy.stats import z score
- from wordcloud import WordCloud, STOPWORDS
- from sklearn.feature_extraction.text import TfidfVectorizer
- from scipy.sparse import hstack
- from sklearn.model_selection import train_test_split
- from collections import Counter
- from sklearn.datasets import make_classification
- from imblearn.over_sampling import SMOTE
- from sklearn.linear_model import LogisticRegression
- from sklearn.metrics import accuracy_score
- from sklearn.metrics import confusion_matrix,classification_report
- from sklearn.tree import DecisionTreeClassifier
- from sklearn.svm import SVC
- from sklearn.ensemble import RandomForestClassifier
- from sklearn.ensemble import ExtraTreesClassifier
- from sklearn.model_selection import GridSearchCV

MODEL BUILDING & EVALUATION

Identification of possible problem-solving approaches (methods)

- I have converted text into feature vectors using TF-IDF vectorizer and separated our features and target.
- Also, before building the model, I made sure that the input data was cleaned and scaled before it was fed into the machine learning models.

- Making the Reviews more appropriate so that we'll have less words 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.
- Removed outliers present in the data using the z-score method.
- Balanced the data using oversampling methods.
- Then followed by model building with all Classification algorithms.

Testing of Identified Approaches (Algorithms)

In this project we need to predict Ratings from the reviews, which is a multi-classification problem. I have converted the text into vectors using TFIDF vectorizer and separated our features and target & then built the model.

Among all the algorithms which I have used for this purpose I have chosen the Logistic Regression Model as the most suitable algorithm for our final model as it is performing well compared to other algorithms.

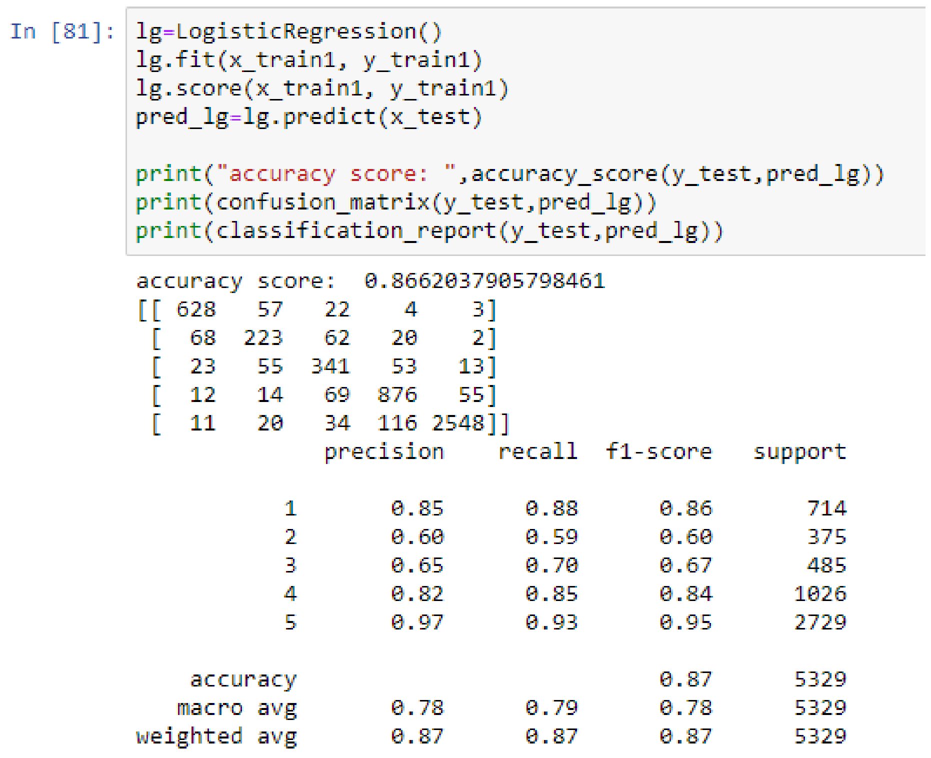
I have used following algorithms and evaluated them:

- Logistic Regression
- Decision Tree Classifier
- Random Forest Classifier
- Extra Trees Classifier
- SVC

Run and Evaluate selected models

<u>Classification Models:</u>

LOGISTIC REGRESSION:

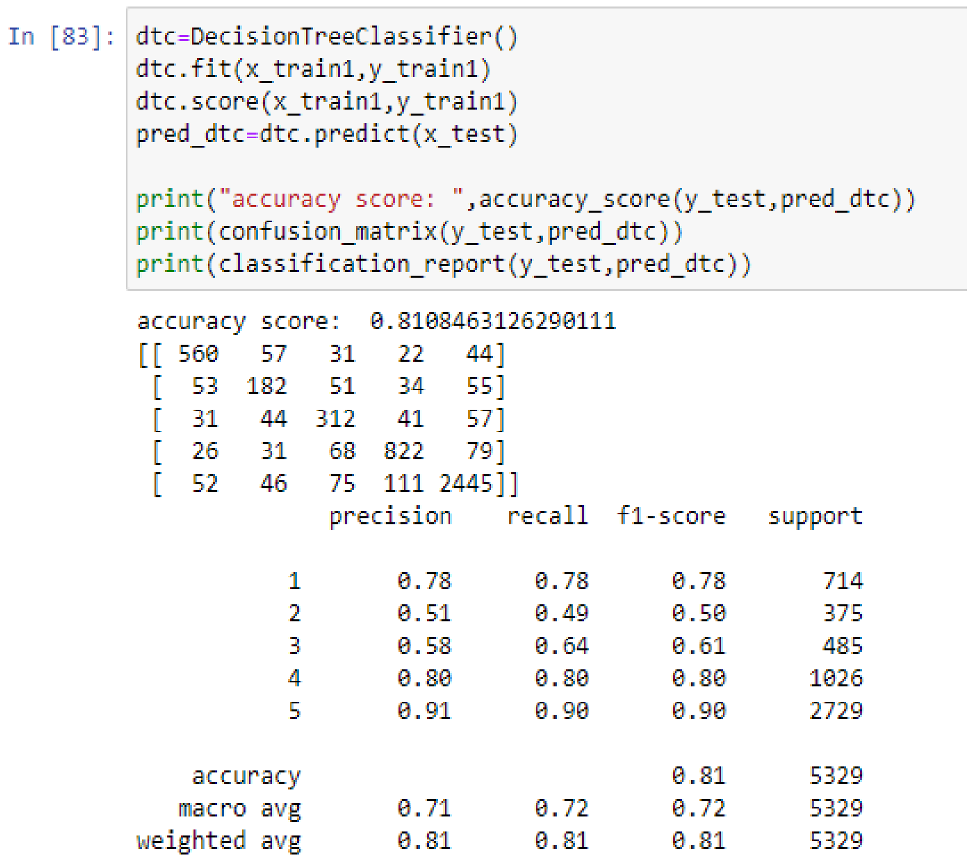


```
In [81]: lg=LogisticRegression()
         lg.fit(x_train1, y_train1)
         lg.score(x_train1, y_train1)
         pred_lg=lg.predict(x_test)

         print("accuracy score: ",accuracy_score(y_test,pred_lg))
         print(confusion_matrix(y_test,pred_lg))
         print(classification_report(y_test,pred_lg))
```

```
accuracy score:  0.8662037905798461
[[ 628   57   22    4    3]
 [  68  223   62   20    2]
 [  23   55  341   53   13]
 [  12   14   69  876   55]
 [  11   20   34  116 2548]]
              precision    recall  f1-score   support

           1       0.85      0.88      0.86       714
           2       0.60      0.59      0.60       375
           3       0.65      0.70      0.67       485
           4       0.82      0.85      0.84      1026
           5       0.97      0.93      0.95      2729

    accuracy                           0.87      5329
   macro avg       0.78      0.79      0.78      5329
weighted avg       0.87      0.87      0.87      5329
```

The Logistic Regression model gave us an Accuracy Score of 86.62 %.

DECISION TREE CLASSIFIER:

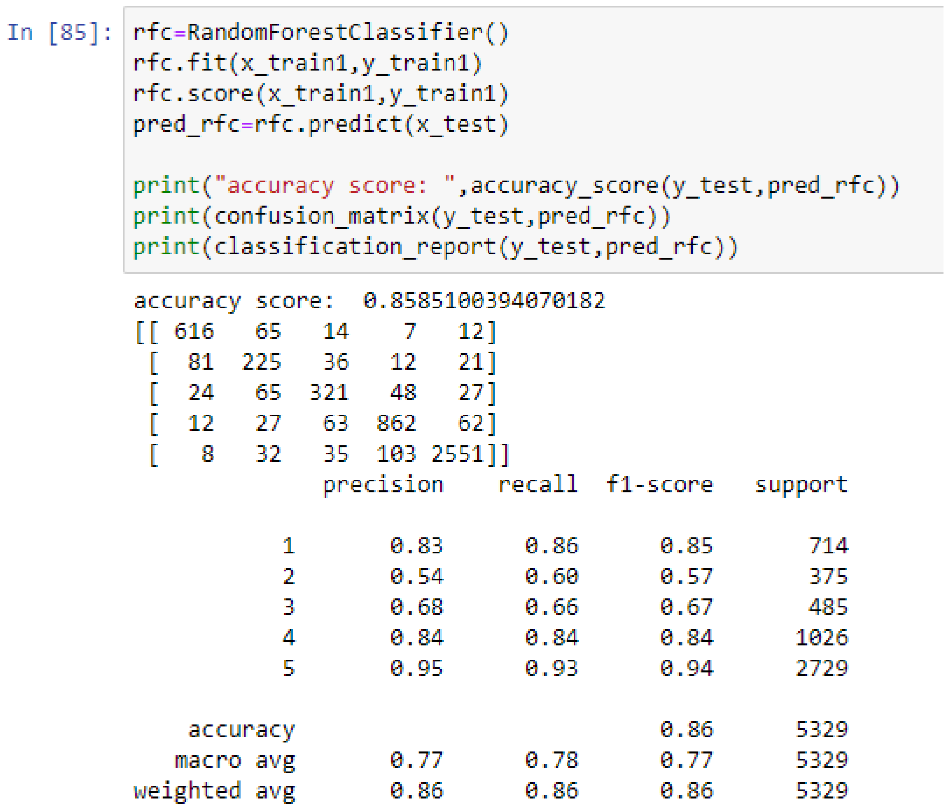


```
In [83]: dtc=DecisionTreeClassifier()
         dtc.fit(x_train1,y_train1)
         dtc.score(x_train1,y_train1)
         pred_dtc=dtc.predict(x_test)

         print("accuracy score: ",accuracy_score(y_test,pred_dtc))
         print(confusion_matrix(y_test,pred_dtc))
         print(classification_report(y_test,pred_dtc))
```

```
accuracy score:  0.8108463126290111
[[ 560   57   31   22   44]
 [  53  182   51   34   55]
 [  31   44  312   41   57]
 [  26   31   68  822   79]
 [  52   46   75  111 2445]]
              precision    recall  f1-score   support

           1       0.78      0.78      0.78       714
           2       0.51      0.49      0.50       375
           3       0.58      0.64      0.61       485
           4       0.80      0.80      0.80      1026
           5       0.91      0.90      0.90      2729

    accuracy                           0.81      5329
   macro avg       0.71      0.72      0.72      5329
weighted avg       0.81      0.81      0.81      5329
```

The Decision Tree Classifier Model gave us an Accuracy Score of 81.08%.

RANDOM FOREST CLASSIFIER:



```
In [85]: rfc=RandomForestClassifier()
         rfc.fit(x_train1,y_train1)
         rfc.score(x_train1,y_train1)
         pred_rfc=rfc.predict(x_test)

         print("accuracy score: ",accuracy_score(y_test,pred_rfc))
         print(confusion_matrix(y_test,pred_rfc))
         print(classification_report(y_test,pred_rfc))
```

```
accuracy score:  0.8585100394070182
[[ 616   65   14    7   12]
 [  81  225   36   12   21]
 [  24   65  321   48   27]
 [  12   27   63  862   62]
 [   8   32   35  103 2551]]
              precision    recall  f1-score   support

           1       0.83      0.86      0.85       714
           2       0.54      0.60      0.57       375
           3       0.68      0.66      0.67       485
           4       0.84      0.84      0.84      1026
           5       0.95      0.93      0.94      2729

    accuracy                           0.86      5329
   macro avg       0.77      0.78      0.77      5329
weighted avg       0.86      0.86      0.86      5329
```

The Random Forest Classifier model gave us an Accuracy Score of 85.85 %.

EXTRA TREES CLASSIFIER:

```
In [87]: etc=ExtraTreesClassifier()
         etc.fit(x_train1,y_train1)
         etc.score(x_train1,y_train1)

         pred_etc=etc.predict(x_test)
         print("accuracy score: ",accuracy_score(y_test,pred_etc))
         print(confusion_matrix(y_test,pred_etc))
         print(classification_report(y_test,pred_etc))
```

```
accuracy score:  0.863013698630137
[[ 619   63   19    5    8]
 [  79  222   39   15   20]
 [  23   62  327   49   24]
 [  10   29   63  856   68]
 [   6   30   35   83 2575]]
              precision    recall  f1-score   support

           1       0.84      0.87      0.85       714
           2       0.55      0.59      0.57       375
           3       0.68      0.67      0.68       485
           4       0.85      0.83      0.84      1026
           5       0.96      0.94      0.95      2729

    accuracy                           0.86      5329
   macro avg       0.77      0.78      0.78      5329
weighted avg       0.87      0.86      0.86      5329
```

The Extra Trees Classifier model gave us an Accuracy Score of 86.30 %.

SUPPORT VECTOR CLASSIFIER (SVC):

```
In [89]: svc = SVC()
         svc.fit(x_train1, y_train1)
         svc.score(x_train1, y_train1)
         svc_pred = svc.predict(x_test)

         print("accuracy score: ",accuracy_score(y_test,svc_pred))
         print(confusion_matrix(y_test,svc_pred))
         print(classification_report(y_test,svc_pred))
```

```
accuracy score:  0.8744604991555639
[[ 644   39   14    7   10]
 [  92  190   52   18   23]
 [  24   36  332   47   46]
 [   9   10   56  869   82]
 [   7   12   17   68 2625]]
              precision    recall  f1-score   support

           1       0.83      0.90      0.86       714
           2       0.66      0.51      0.57       375
           3       0.70      0.68      0.69       485
           4       0.86      0.85      0.85      1026
           5       0.94      0.96      0.95      2729

    accuracy                           0.87      5329
   macro avg       0.80      0.78      0.79      5329
weighted avg       0.87      0.87      0.87      5329
```

The Support Vector Classifier (SVC) model gave us an Accuracy Score of 87.44 %.

From the above classification models, the highest accuracy score belongs to the SVC Model. Followed by the Logistic Regression Model.

Next, the Extra Trees Classifier model & Random Forest Classifier model.

Lastly, the Decision Tree Classifier model has the lowest accuracy score among the other models.

Cross Validation Scores:

```
In [91]: scr_lg=cross_val_score(lg,train_features,y,cv=5)
         print("Cross validation score of this model is: ",scr_lg.mean())
```

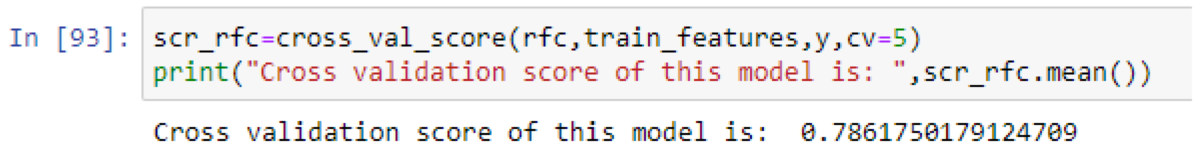
```
Cross validation score of this model is:  0.8026432631662027
```

The cross validation score of the Logistic Regression Model is 80.26 %.

```
In [92]: scr_dtc=cross_val_score(dtc,train_features,y,cv=5)
         print("Cross validation score of this model is: ",scr_dtc.mean())
```

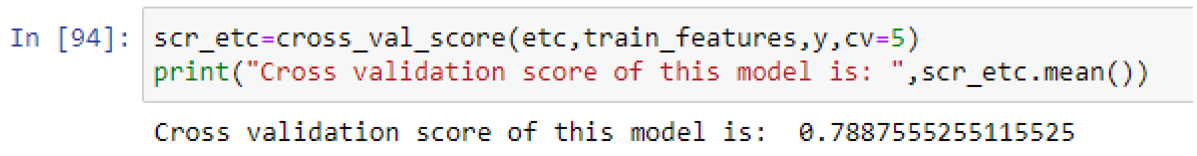
```
Cross validation score of this model is:  0.7240076755309317
```

The cross validation score of the Decision Tree Classifier Model is 72.40 %.



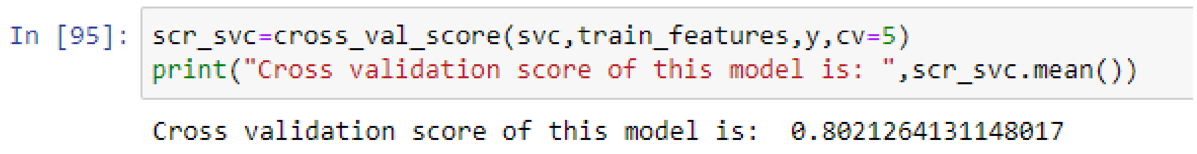
```
In [93]: scr_rfc=cross_val_score(rfc,train_features,y,cv=5)
         print("Cross validation score of this model is: ",scr_rfc.mean())

         Cross validation score of this model is:  0.7861750179124709
```

The cross validation score of the Random Forest Classifier Model is 78.61 %.



```
In [94]: scr_etc=cross_val_score(etc,train_features,y,cv=5)
         print("Cross validation score of this model is: ",scr_etc.mean())

         Cross validation score of this model is:  0.7887555255115525
```

The cross validation score of the Extra Trees Classifier Model is 78.87 %.



```
In [95]: scr_svc=cross_val_score(svc,train_features,y,cv=5)
         print("Cross validation score of this model is: ",scr_svc.mean())

         Cross validation score of this model is:  0.8021264131148017
```

The cross validation score of the Support Vector Classifier (SVC) Model is 80.21 %.

From the above Cross Validation Scores, the highest score belongs to the Logistic Regression Model. Followed by the SVC Model.

Next the Extra Trees Classifier model & Random Forest Classifier model.

Lastly, the Decision Tree Classifier model.

Hyper Parameter Tuning:

Since the Accuracy Score and the Cross Validation Score of the Logistic Regression Model have the least difference between them, we shall consider it for hyper parameter tuning.

We shall use GridSearchCV for hyper parameter tuning.

```
In [96]: from sklearn.model_selection import GridSearchCV
```

```
In [97]: parameters={
             'C': [0.5,1.0],
             'penalty': ['l1', 'l2'],
             'solver':['newton-cg','lbfgs']}
         grid_lg = GridSearchCV(lg, param_grid = parameters, cv = 4)
```

```
In [98]: grid_lg.fit(x_train1, y_train1)

Out[98]: GridSearchCV(cv=4, estimator=LogisticRegression(),
                      param_grid={'C': [0.5, 1.0], 'penalty': ['l1', 'l2'],
                                  'solver': ['newton-cg', 'lbfgs']})
```

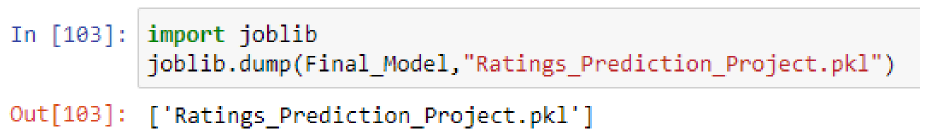
```
In [99]: grid_lg.best_params_

Out[99]: {'C': 1.0, 'penalty': 'l2', 'solver': 'newton-cg'}
```

```
In [100]: Final_Model= LogisticRegression(C=0.1,penalty='l2',solver='newton-cg')

          Final_Model.fit(x_train1,y_train1)
          pred = Final_Model.predict(x_test)
          print("accuracy score: ",accuracy_score(y_test,pred))
          print(confusion_matrix(y_test,pred))
          print(classification_report(y_test,pred))

          accuracy score:  0.8316757365359354
          [[ 596   93   19    3    3]
           [  54  249   51   18    3]
           [  20   78  336   41   10]
           [  16   36  105  812   57]
           [  13   45   59  173 2439]]
                        precision    recall  f1-score   support

                     1       0.85      0.83      0.84       714
                     2       0.50      0.66      0.57       375
                     3       0.59      0.69      0.64       485
                     4       0.78      0.79      0.78      1026
                     5       0.97      0.89      0.93      2729

              accuracy                           0.83      5329
             macro avg       0.74      0.78      0.75      5329
          weighted avg       0.85      0.83      0.84      5329
```

After Hyper Parameter Tuning, we have got an accuracy score of 83.16 %.

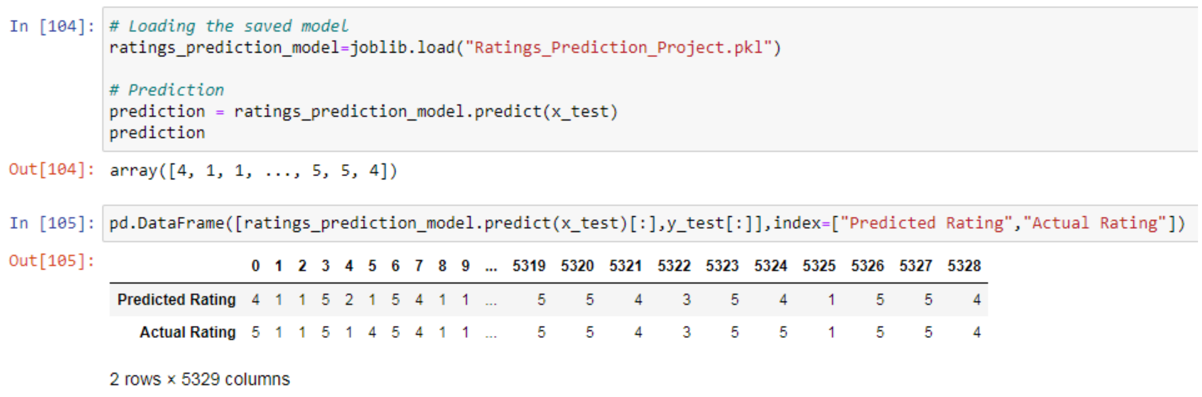
Saving the final model and predicting the saved model

Now we shall save the best model.



```
In [103]: import joblib
          joblib.dump(Final_Model,"Ratings_Prediction_Project.pkl")

Out[103]: ['Ratings_Prediction_Project.pkl']
```

Predictions from the saved model.



```
In [104]: # Loading the saved model
          ratings_prediction_model=joblib.load("Ratings_Prediction_Project.pkl")

          # Prediction
          prediction = ratings_prediction_model.predict(x_test)
          prediction

Out[104]: array([4, 1, 1, ..., 5, 5, 4])

In [105]: pd.DataFrame([ratings_prediction_model.predict(x_test)[:],y_test[:]],index=["Predicted Rating","Actual Rating"])

Out[105]:
```

	0	1	2	3	4	5	6	7	8	9	...	5319	5320	5321	5322	5323	5324	5325	5326	5327	5328
Predicted Rating	4	1	1	5	2	1	5	4	1	1	...	5	5	4	3	5	4	1	5	5	4
Actual Rating	5	1	1	5	1	4	5	4	1	1	...	5	5	4	3	5	5	1	5	5	4

2 rows × 5329 columns

The actual and predicted values are almost similar.

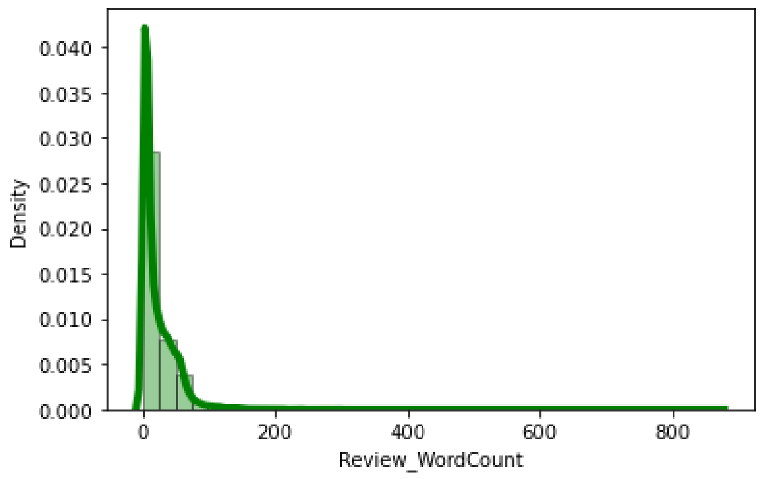
Key Metrics for success in solving problem under consideration

I have used the following metrics for evaluation:

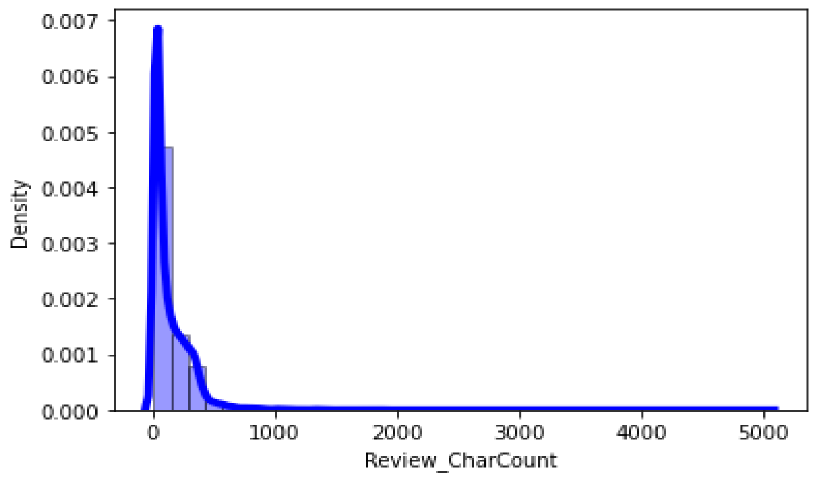
- Accuracy score, which is used when the True Positives and True negatives are more important.
- Confusion Matrix, which is a matrix used to determine the performance of the classification models for a given set of test data.
- Classification report, which is used to measure the quality of predictions from a classification algorithm.
- Cross Validation Score, which is a technique for evaluating ML models by training several ML models on subsets of the available input data and evaluating them on the complementary subset of the data.

Visualizations

i) Plotting word count and character count using hist plot:

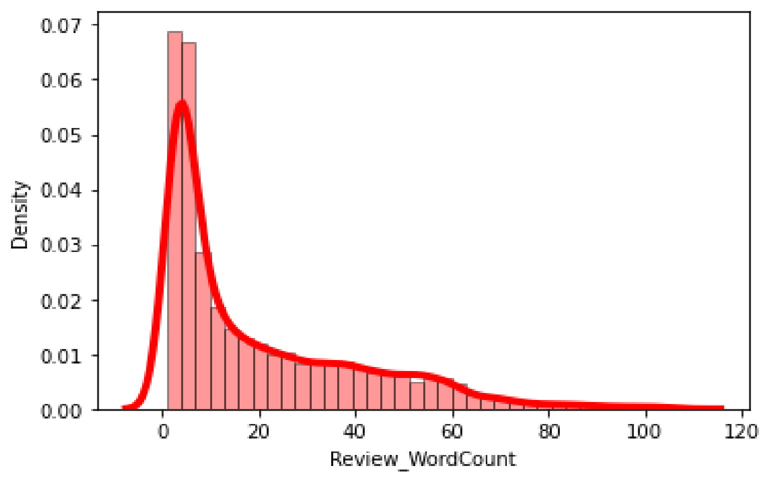


OBSERVATION : By observing the histogram we can clearly see that most of our text is having a word count in the range of 0 to 200. But some of the reviews are too lengthy which may act like outliers in our data.

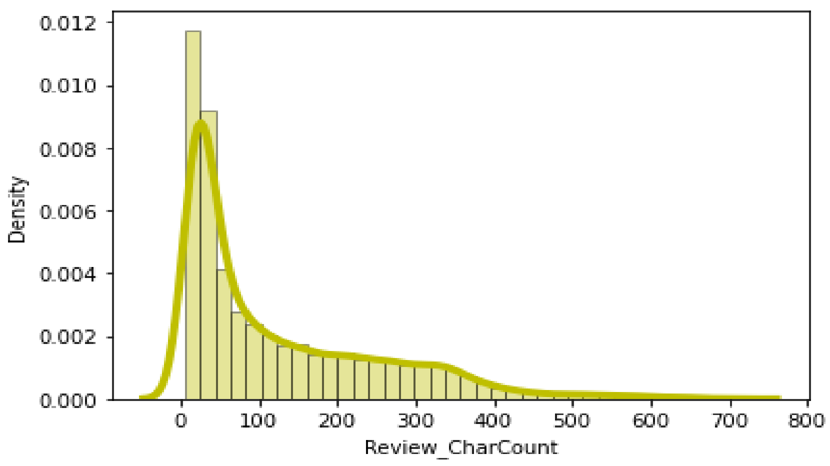


OBSERVATION : The above plot represents histogram for character counts of Review text, which is quite similar to the histogram of word count. Most of the characters are between 0 - 1000.

Some of the reviews are too lengthy, so we have to treat them as outliers and remove them using the z_score method. After removing the outliers the word count and character count looks as below.

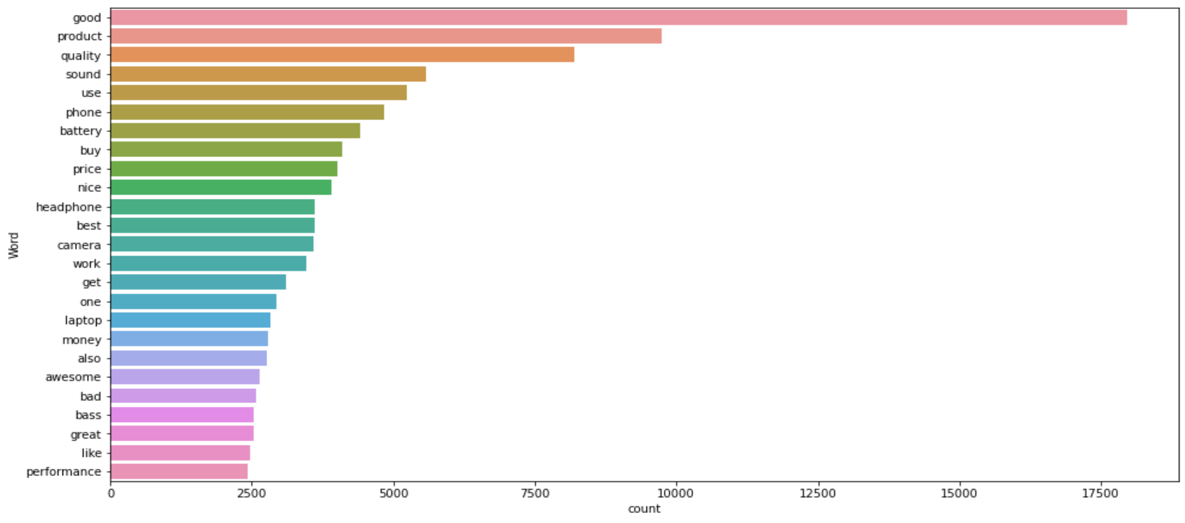


OBSERVATION : After plotting histograms for word counts after removing outliers we can see we are left out with a good range of number of words.

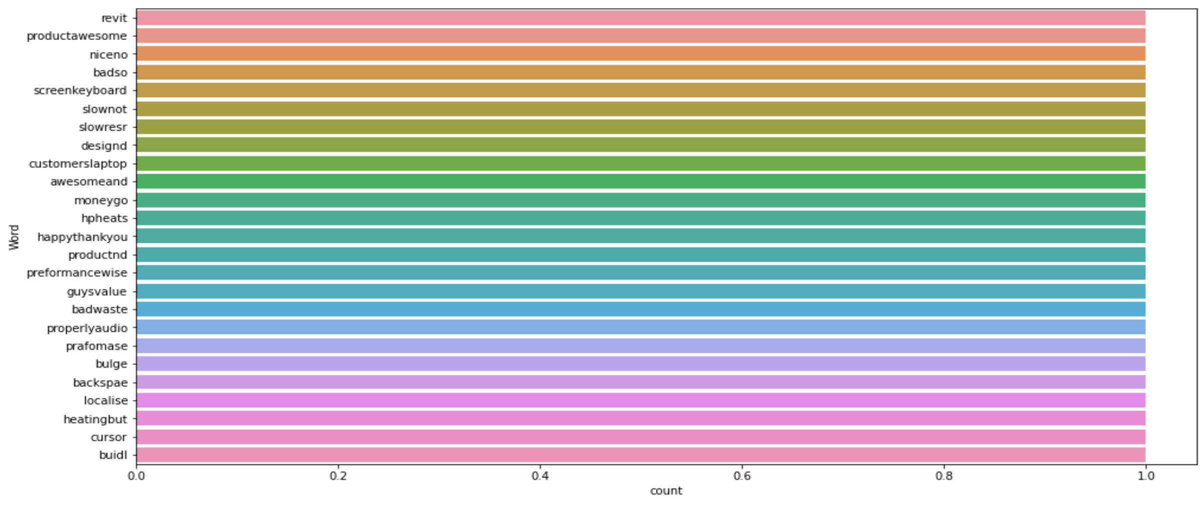


OBSERVATION : After plotting histograms for character counts and after removing outliers we can see we are left out with a good range of number of characters.

ii)Top 25 most frequently occuring and rarely occurring words:

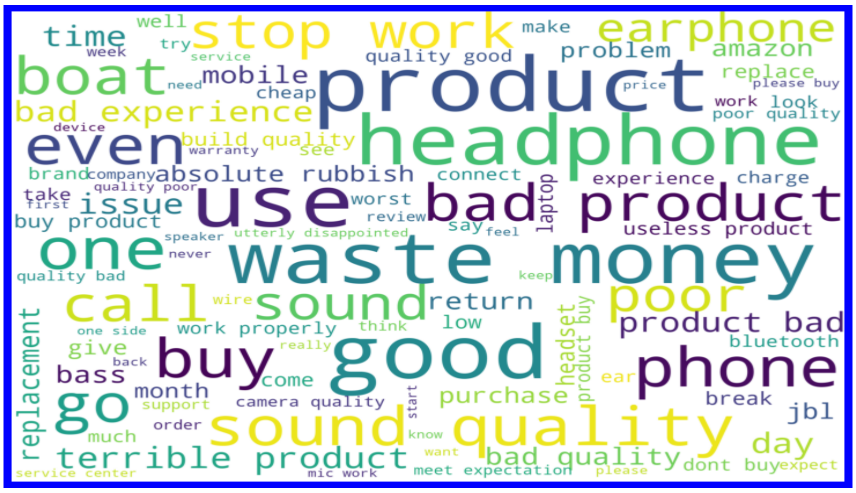


OBSERVATION : The Above list of words have a frequent occurrence. Mainly, 'good',' product' & 'quality'.

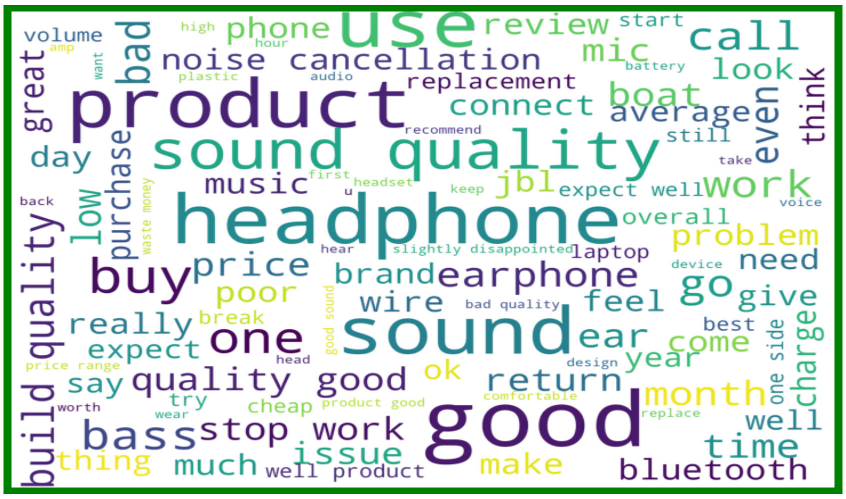


OBSERVATION : Above list of words have rare occurrence in Review. Only 1 time each.

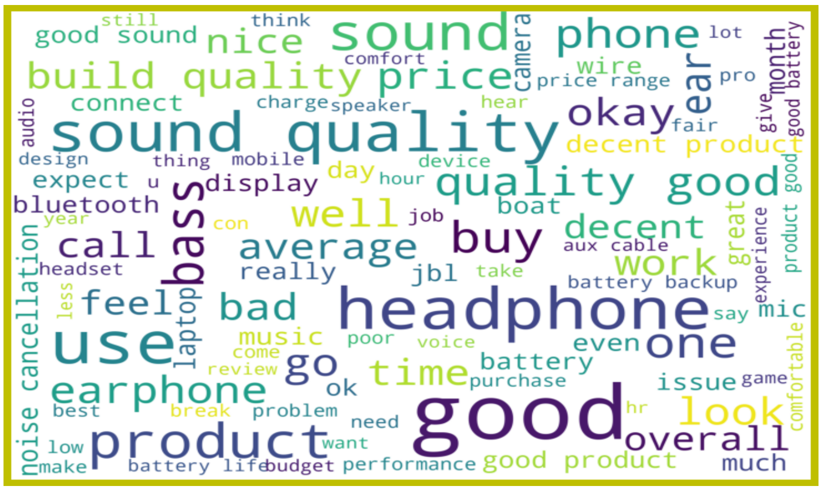
iii) Word Clouds for different ratings:



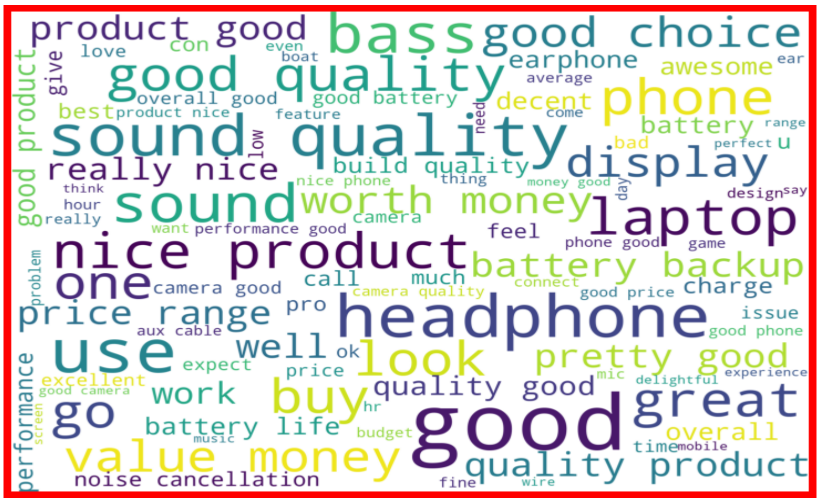
OBSERVATION : Above is the word cloud of words used most frequently to least with the rating of 1 star.



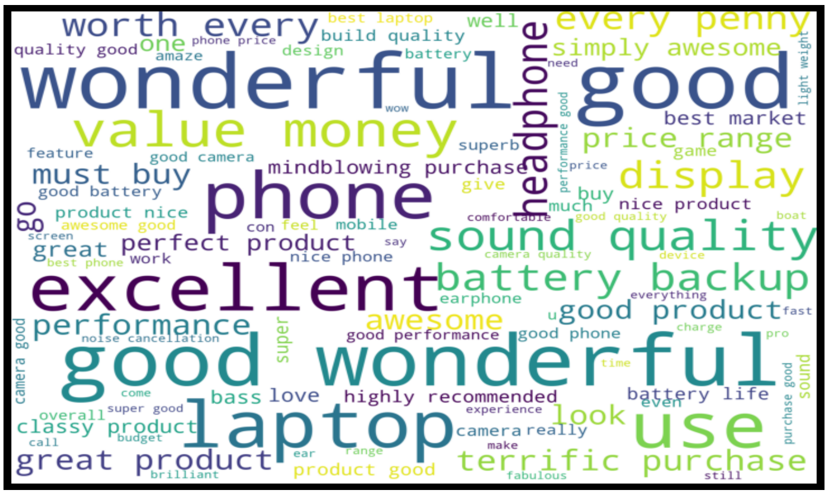
OBSERVATION : Above is the word cloud of words used most frequently to least with the rating of 2 stars.



OBSERVATION : Above is the word cloud of words used most frequently to least with the rating of 3 stars.



OBSERVATION : Above is the word cloud of words used most frequently to least with the rating of 4 stars.



OBSERVATION : Above is the word cloud of words used most frequently to least with the rating of 5 stars.

Interpretation of the Results

The dataset was scrapped from flipkart.com & amazon.in websites. The dataset was very challenging to handle; it had 3 features with 21661 samples. We conducted various text processing, Lemmatization, Removing stop words & Text standardization, in order to clean our dataset and make it suitable for visualization and model building.

- From the above plots we can clearly see the words which are an indication of the Reviewer's opinion on products.
- Distribution plots were used to see the word counts and character counts before and after removing outliers.
- The most frequently and rarely occurring words adre also displayed using bar plots.
- The most frequent words used for each Rating are displayed in the word cloud.
- We used multiple algorithms while building a suitable classification model using the dataset to get the best model out of it.
- And we have to use multiple metrics like accuracy score, confusion matrix, classification report & cross validation score, which helped us to decide the best model.
- I found the Logistic Regression Model to be the best model with an 86.62 % accuracy score.
- Also, I tried to improve the accuracy of the best model by running hyper parameter tuning. After multiple attempts we were unable to increase the accuracy score of our model. After hyper parameter tuning the accuracy score of our best model was 83.16 %.
- At last I have predicted the ratings using the saved model. I was able to get the predictions near to actual values.

CONCLUSION

Key Findings and Conclusions of the Study

- In this project, we have used machine learning algorithms to predict the ratings. We have collected data of reviews and ratings for different products from amazon.in and flipkart.com.
- We have tried to detect the Ratings in commercial websites on a scale of 1 to 5 on the basis of the reviews given by the users. We made use of natural language processing and machine learning algorithms in order to do so.
- Then I have done different text processing for the reviews column and chose an equal number of text from each rating class to eliminate the problem of imbalance.
- By doing different EDA steps I have analyzed the text.
- We have checked frequently occurring words in our data as well as rarely occurring words.
- After all these steps I have built a function to train and test different algorithms using various evaluation metrics.
- We have selected the Logistic Regression Model as our final model.
- Finally by doing hyperparameter tuning we got optimum parameters for our final model.
- It was good that the predicted and actual values were almost the same.

Learning Outcomes of the Study in respect of Data Science

In this project, we have used machine learning algorithms to predict the Ratings. We have mentioned the step by step procedure in analyzing & preparing the dataset and finding the correlation between the features.

Data cleaning is an important step in model building to remove unrealistic values and unnecessary punctuations, urls, email address, stop words, etc. Visualization tools such as distribution plots, histograms & word clouds helped us in understanding the data.

The cleaned data was given as an input to 5 algorithms and a hyper parameter tuning was done to find the best model. We calculated the performance of each model using different performance metrics and compared them based on these metrics. Finally, we saved the best model.

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

LIMITATIONS:

- As we know the content of text in reviews totally depends on the reviewer and they may rate differently which totally depends on that particular person.
- So it is difficult to predict ratings based on the reviews with higher accuracy.
- While we couldn't reach our goal of maximum accuracy in the 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.

FUTURE WORK:

- 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.