FLIP ROBO

Rating Prediction Project

Submitted by:

HINAL SETH

ACKNOWLEDGMENT

I'd want to express my heartfelt thanks to the "Flip Robo" team for providing me with the opportunity to work on such a beautiful idea and for helping me develop my data analysis abilities. And I'd want to express my heartfelt appreciation to Ms. Swati Mahaseth (SME Flip Robo), who has helped me overcome all of the challenges I've encountered while working on this project, as well as encouraged me much with his wise words and unwavering support, resulting in a lovely project.

Lastly, I thank almighty, my parents, brother, sister and friends for their constant encouragement without which this assignment would not be possible.

References:

1. https://www.amazon.in/
2. https://www.flipkart.com/
3. https://www.wiki.org/
4. https://www.medium.com/

INTRODUCTION

- Business Problem Framing

  Rating prediction is a well-known recommendation job that attempts to forecast a user's rating for goods that she has not yet evaluated. Users' explicit input, i.e. their prior evaluations on particular goods, is used to calculate predictions. User reviews, which indirectly indicate consumers' thoughts on things, are another sort of feedback. Recent research suggests that inferred opinions from user evaluations on things are significant predictors of implicit feedback or even ratings from users, and hence should be included in computation.

  Customer reviews have become increasingly important as E-commerce has grown in popularity. There are hundreds of review sites online, and each product has a large number of reviews. Buyers' buying habits have evolved, and 70 percent of customers claim they use rating filters to filter out low-rated goods in their searches, according to a recent poll. Companies that encourage reviews, such as Google, Amazon, and Yelp!, rely on their ability to determine if a review will be beneficial to other consumers and so increase product visibility. There are two major approaches to this problem. The first is based on review text content analysis and employs natural language processing methods (the NLP).

  This strategy lacks the information that may be gained from the customer-item connection. The second is based on recommender systems, notably collaborative filtering, and is concerned with the reviewer's perspective.

  We have a customer that runs a website where users may leave evaluations for various technological items. They are now adding a new element to their website, requiring reviewers to provide stars (ratings) with their reviews. The rating scale ranges from one to five stars, with just five alternatives available: one star, two stars, three stars, four stars, and five stars. They now seek to forecast ratings for reviews that were written in the past but did not receive a rating. As a result, we must create an application that can anticipate the rating based on the review.

• Conceptual Background of the Domain Problem

To our knowledge, all current efforts on recommendation approaches based on inferred opinions from user reviews are either focused on the item recommendation job or employ only the opinion information, fully disregarding user ratings. The technique suggested in this study fills this need by offering a simple, tailored, and scalable rating prediction platform that takes into account both user ratings and inferred opinions from their reviews. The suggested framework's usefulness is demonstrated by experimental results on a dataset including user ratings and reviews from real-world Amazon and Flipkart Product Review Data.

In today's e-commerce applications, such as targeted advertising, tailored marketing, and information retrieval, recommendation systems are critical components. In recent years, the value of contextual data has prompted the creation of tailored recommendations based on the contextual data available to consumers. Review-based recommendation, as opposed to traditional algorithms that rely only on a user's rating history, should present consumers with more relevant results. We present a review-based recommendation system that mines user evaluations for contextual information. The characteristics gained by analysing textual reviews utilising methods established in the Natural Language Processing (NLP) and information retrieval disciplines to compute a utility function over a given item are the focus of the proposed methodology. An item's utility is a metric that indicates how much it is preferred in the context of the user.

- ## Review of Literature

  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 mining. Mining the sentiments and opinions that are contained in online reviews has become an important topic in NLP, machine learning, and Web mining.

  In real life, people's decisions are frequently influenced by the actions or recommendations of their friends. The use of social data has been extensively researched. Based on probabilistic matrix factorization, Yang et al. introduce the notion of "Trust Circles" in social networks. Individual choice, according to Jiang et al., is another crucial component. Some websites do not always give structured information, and all of these approaches do not take use of the user's unstructured data, such as reviews and explicit social network data, making it difficult to make a reasonable forecast for each user. The sentiment factor phrase is utilised to improve social recommendation in this problem.

- ## Motivation for the Problem Undertaken

  The major goal has been to gain exposure to real-world data and the ability to apply my talents to a real-time situation. Many product reviews lack a scale grading mechanism, instead relying solely on a textual evaluation. In this circumstance, comparing several items in order to make a decision between them becomes difficult and time-consuming. As a result, algorithms that can predict the user rating based on the text review are vital. Getting a feel of a textual evaluation as a whole might improve the consumer experience. However, I was drawn to this project since it is in a relatively new field of study. We have a lot of alternatives, but fewer solid answers.

The main aim is to create a prototype of an online hate and abuse review classifier that can be used to identify hate and positive comments so that they may be regulated and rectified based on the reviewer's preferences.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  The Ratings in this issue can be 1, 2, 3, 4, or 5, and indicate the product's probable ness to the client. As a result, it's evident that this is a multi-classification problem, and I'll have to employ all of the classification methods to create the model. One sort of supervised learning method that we might use is classification. Only categorisation will be done here. Because the dataset only has one feature, filtering the words is required to avoid overfitting. To find the regularisation parameter, we would first eliminate email, phone numbers, web addresses, spaces, and stop words from the categorization section of the project. While building the model, I utilised all of the classification methods, then tuned and saved the best model.

- Data Sources and their formats

  The data set contains nearly 23402 samples with 2 features. Since Ratings is 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 likelyness of the product to the customer. The data set includes:

  • Review: Text Content of the Review.

  • Rating : Ratings out of 5 stars.

  This project is more about exploration, feature engineering and classification that can be done on this data. Since the data set is huge and includes multi classification of ratings, we can do good amount of data exploration and derive some interesting features using the Review column available. We need to build a model that can predict Ratings of the reviewer.

- Data Preprocessing Done

  The process of turning raw data into a legible format for use by Machine Learning models is known as data pre-processing. Data pre-processing is a crucial stage in Machine Learning since the quality of data and the relevant information that can be gleaned from it has a direct impact on our model's capacity to learn; consequently, we must pre-process our data before feeding it into our model. I utilised the pre-processing techniques mentioned below:

1. Importing required libraries and saving the dataset as a data frame
2. Checked some statistical data such as shape, number of unique values present, information, null values, and value counts, among other things.
3. I looked for null values and used the imputation method to replace them.
4. By plotting a distribution plot and a wordcloud for each rating, I was able to visualise each characteristic using the seaborn and matplotlib packages.
5. Removing punctuation and other special characters, splitting comments into distinct words, removing stop words, stemming, and lemmatization are some of the text pre-processing techniques that have been completed.
6. TF-IDF vectorizer was employed once the data was cleansed. It will assist in the conversion of text data to feature vectors, which can then be utilised as input in our 6 modelling. It's a standard algorithm for converting text to numbers. It assesses a word's uniqueness by comparing the frequency with which it appears in a document with the number of documents in which it appears.
7. Using SMOTE method to balance the target variable.

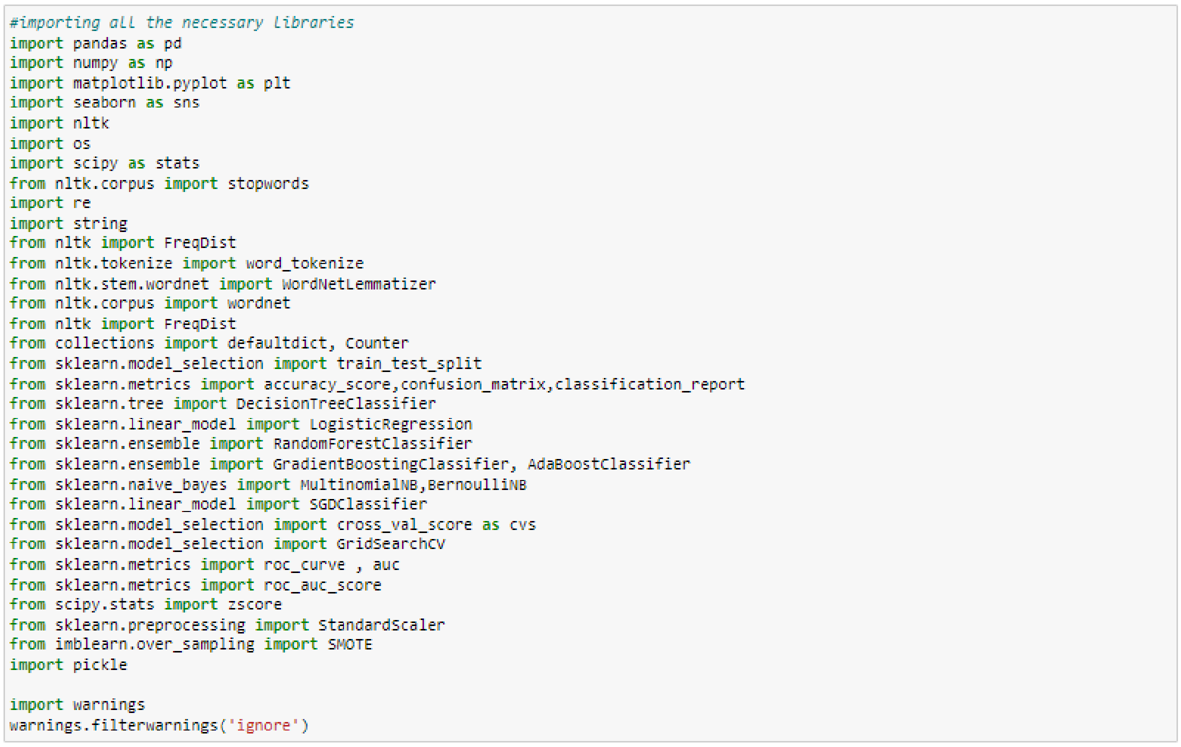
- Data Inputs- Logic- Output Relationships

  To study the relationship between Input and Output of this dataset we analyse words' frequency for each label so that we can get the most frequent words that were used in the different features.

- Hardware and Software Requirements and Tools Used

  We should be familiar with the hardware and software essential for the project's effective completion before beginning it. The following hardware and software are required:

- Minimum Hardware Requirement: i3 core processor, 8GB RAM and 256 ROM/SSD
- Minimum Software Requirement: Anaconda Navigator and browser based Python programming tool like Jupyter Notebook.



```
#importing all the necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
import os
import scipy as stats
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 collections import defaultdict, Counter
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier, AdaBoostClassifier
from sklearn.naive_bayes import MultinomialNB,BernoulliNB
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import cross_val_score as cvs
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import roc_curve , auc
from sklearn.metrics import roc_auc_score
from scipy.stats import zscore
from sklearn.preprocessing import StandardScaler
from imblearn.over_sampling import SMOTE
import pickle

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

Model/s Development and Evaluation

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

  Using the TF-IDF vectorizer, I transformed text into feature vectors and separated our feature and labels. In addition, before feeding the input data into the machine learning models, I made sure that it was cleaned and scaled. Simply making the Reviews more relevant so that we can digest fewer words with more accuracy. Extra spaces were removed, the email address was turned to an email keyword, and the phone number was changed, among other things. As far as possible, I tried to keep reviews short and relevant.

- ### Testing of Identified Approaches (Algorithms)

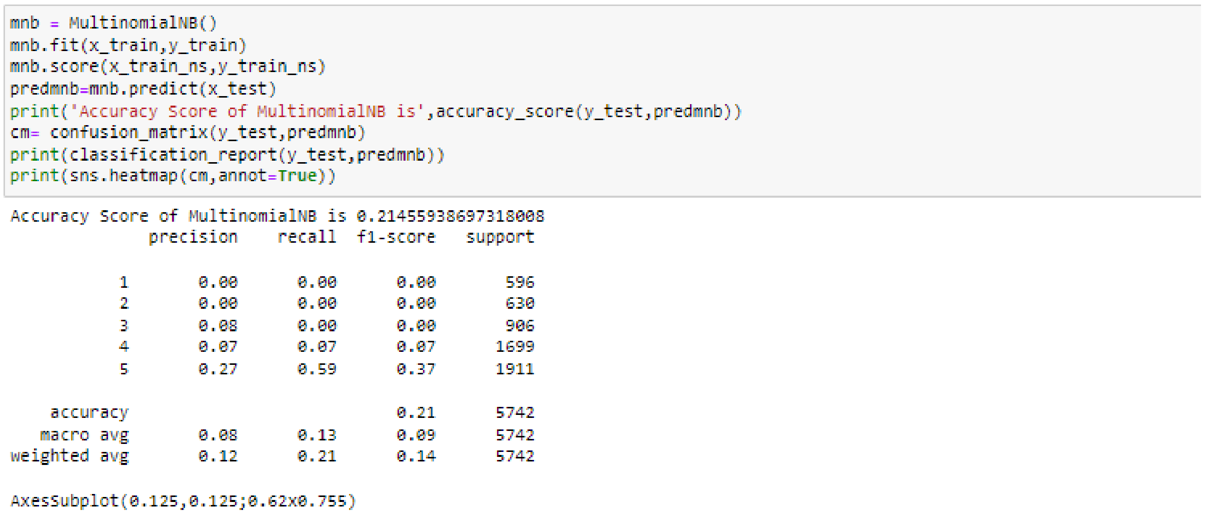
  We must anticipate Ratings in this nlp-based project, which is a multiclassification task. I used TFIDF vectorizer to turn the text into vectors, separated our feature and labels, and then built the model using One Vs Rest Classifier. I choose RandomForestClassifier as the best suited algorithm for our final model among all the algorithms. I have chosen it since it performs well when compared to other algorithms when assessing with different metrics. I have used and tested the following methods:

  1. MultinomialNB
  2. Logistic Regression
  3. BernoulliNB
  4. SGDClassifier
  5. GradientBoostingClassifier
  6. RandomForestClassifier
  7. DecisionTreeClassifier

- ## Run and Evaluate selected models

  Describing all the algorithms used along with the snapshot of their code:

  1. MultinomialNB



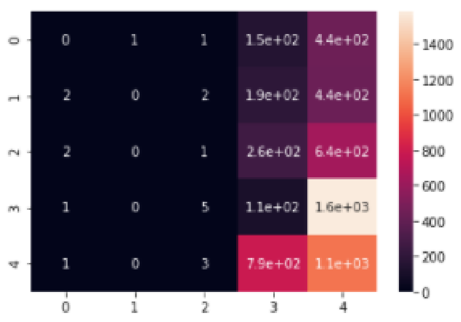
```
mnb = MultinomialNB()
mnb.fit(x_train,y_train)
mnb.score(x_train_ns,y_train_ns)
predmnb=mnb.predict(x_test)
print('Accuracy Score of MultinomialNB is',accuracy_score(y_test,predmnb))
cm= confusion_matrix(y_test,predmnb)
print(classification_report(y_test,predmnb))
print(sns.heatmap(cm,annot=True))
```

```
Accuracy Score of MultinomialNB is 0.21455938697318008
              precision    recall  f1-score   support

           1       0.00      0.00      0.00       596
           2       0.00      0.00      0.00       630
           3       0.08      0.00      0.00       906
           4       0.07      0.07      0.07      1699
           5       0.27      0.59      0.37      1911

    accuracy                           0.21      5742
   macro avg       0.08      0.13      0.09      5742
weighted avg       0.12      0.21      0.14      5742

AxesSubplot(0.125,0.125;0.62x0.755)
```



2. Logistic Regression

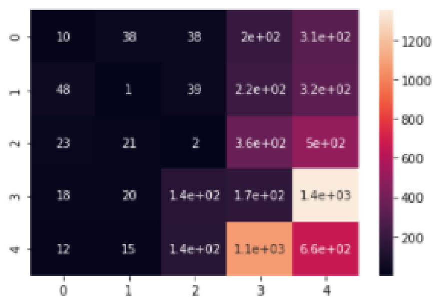
```
lr = LogisticRegression(solver='lbfgs')
lr.fit(x_train,y_train)
lr.score(x_train_ns,y_train_ns)
predlr=lr.predict(x_test)
print('Accuracy Score of Logistic Regression is',accuracy_score(y_test,predlr))
cm= confusion_matrix(y_test,predlr)
print(classification_report(y_test,predlr))
print(sns.heatmap(cm,annot=True))
```

```
Accuracy Score of Logistic Regression is 0.1469871125043539
              precision    recall  f1-score   support

           1       0.09      0.02      0.03       596
           2       0.01      0.00      0.00       630
           3       0.01      0.00      0.00       906
           4       0.08      0.10      0.09      1699
           5       0.21      0.35      0.26      1911

    accuracy                           0.15      5742
   macro avg       0.08      0.09      0.08      5742
weighted avg       0.11      0.15      0.12      5742

AxesSubplot(0.125,0.125;0.62x0.755)
```



3. BernoulliNB

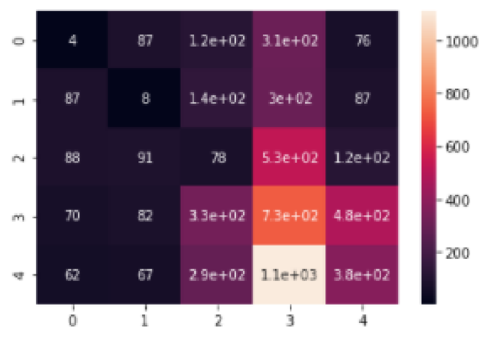
```
bnb = BernoulliNB()
bnb.fit(x_train,y_train)
bnb.score(x_train_ns,y_train_ns)
predbnb=bnb.predict(x_test)
print('Accuracy Score of BernoulliNB is',accuracy_score(y_test,predbnb))
cm= confusion_matrix(y_test,predbnb)
print(classification_report(y_test,predbnb))
print(sns.heatmap(cm,annot=True))
```

```
Accuracy Score of BernoulliNB is 0.20916057122953674
              precision    recall  f1-score   support

           1       0.01      0.01      0.01       596
           2       0.02      0.01      0.02       630
           3       0.08      0.09      0.08       906
           4       0.25      0.43      0.31      1699
           5       0.33      0.20      0.25      1911

    accuracy                           0.21      5742
   macro avg       0.14      0.15      0.13      5742
weighted avg       0.20      0.21      0.19      5742

AxesSubplot(0.125,0.125;0.62x0.755)
```



4. SGDClassifier

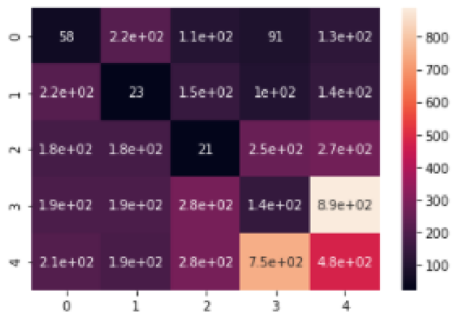
```
sgd=SGDClassifier()
sgd.fit(x_train_ns,y_train_ns)
sgd.score(x_train_ns,y_train_ns)
predsgd=sgd.predict(x_test)
print('Accuracy score of SGDClassifier is',accuracy_score(y_test,predsgd))
cm=confusion_matrix(y_test,predsgd)
print(classification_report(y_test,predsgd))
print(sns.heatmap(cm,annot=True))
```

```
Accuracy score of SGDClassifier is 0.12608847091605713
              precision    recall  f1-score   support

           1       0.07      0.10      0.08       596
           2       0.03      0.04      0.03       630
           3       0.02      0.02      0.02       906
           4       0.11      0.08      0.09      1699
           5       0.25      0.25      0.25      1911

    accuracy                           0.13      5742
   macro avg       0.10      0.10      0.10      5742
weighted avg       0.13      0.13      0.13      5742

AxesSubplot(0.125,0.125;0.62x0.755)
```



5. GradientBoostingClassifier

```
gc = GradientBoostingClassifier(random_state=9)
gc.fit(x_train_ns,y_train_ns)
gc.score(x_train_ns,y_train_ns)
predgc=gc.predict(x_test)
print('Accuracy Score of GradientBoostingClassifier is',accuracy_score(y_test,predgc))
cm=confusion_matrix(y_test,predgc)
print(classification_report(y_test,predgc))
```

```
Accuracy score of GradientBoostingClassifier is 0.5260884709160571З
              precision    recall  f1-score   support


      1       0.17      0.20      0.18       596
      2       0.13      0.14      0.13       630
      3       0.12      0.12      0.12       906
      4       0.21      0.18      0.19      1699
      5       0.35      0.35      0.35      1911


    accuracy                           0.53      5742
   macro avg       0.20      0.20      0.20      5742
weighted avg       0.23      0.23      0.23      5742
```

6. RandomForestClassifier

```
rf = RandomForestClassifier()
rf.fit(x_train_ns,y_train_ns)
rf.score(x_train_ns,y_train_ns)
predrf=rf.predict(x_test)
print('Accuracy score of RandomForestClassifier is',accuracy_score(y_test,predrf))
cm=confusion_matrix(y_test,predrf)
print(classification_report(y_test,predrf))
```

```
Accuracy score of RandomForestClassifier is 0.58608847091605713
              precision    recall  f1-score   support


           1       0.67      0.70      0.68       596
           2       0.16      0.14      0.13       630
           3       0.26      0.19      0.22       906
           4       0.29      0.48      0.39      1699
           5       0.75      0.65      0.75      1911


    accuracy                           0.59      5742
   macro avg       0.43      0.42      0.43      5742
weighted avg       0.58      0.56      0.56      5742
```

7. DecisionTreeClassifier

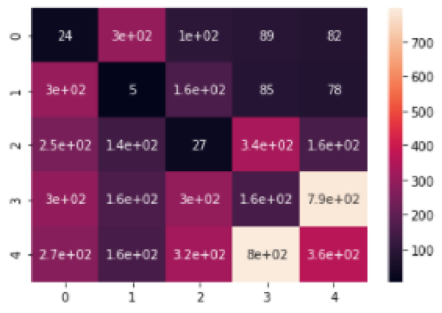
```
dt = DecisionTreeClassifier()
dt.fit(x_train_ns,y_train_ns)
dt.score(x_train_ns,y_train_ns)
preddt=dt.predict(x_test)
print('Accuracy score of DecisionTree Classifier is',accuracy_score(y_test,preddt))
cm=confusion_matrix(y_test,preddt)
print(classification_report(y_test,preddt))
print(sns.heatmap(cm,annot=True))
```

```
Accuracy score of DecisionTree Classifier is 0.10031347962382445
              precision    recall  f1-score   support

           1       0.02      0.04      0.03       596
           2       0.01      0.01      0.01       630
           3       0.03      0.03      0.03       906
           4       0.11      0.09      0.10      1699
           5       0.25      0.19      0.22      1911

    accuracy                           0.10      5742
   macro avg       0.08      0.07      0.08      5742
weighted avg       0.12      0.10      0.11      5742

AxesSubplot(0.125,0.125;0.62x0.755)
```

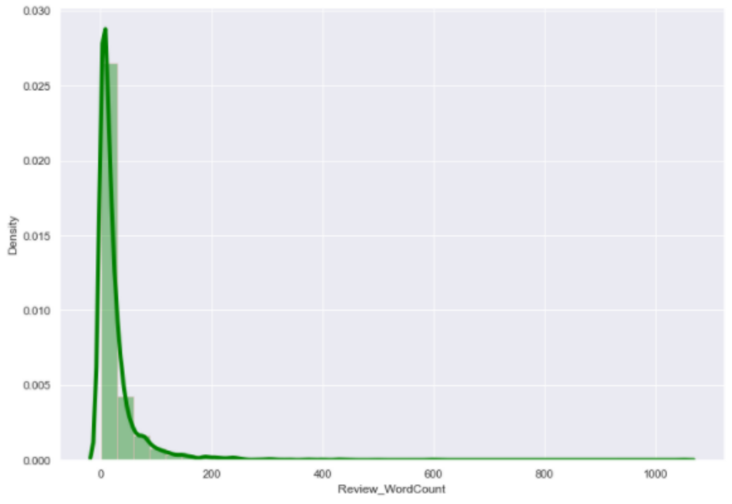


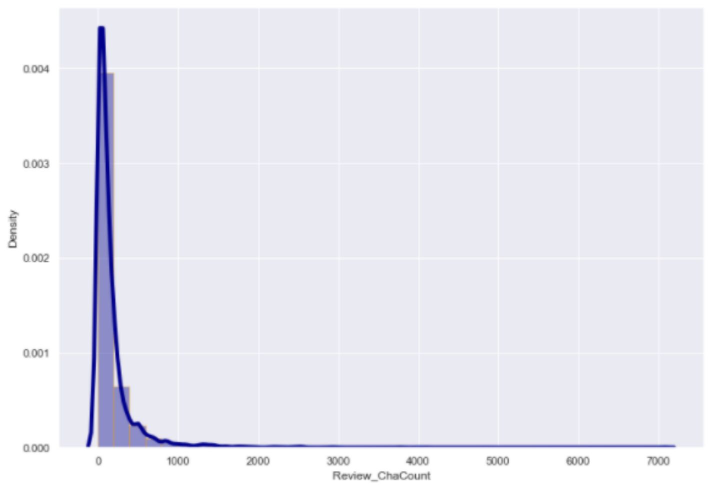
- Key Metrics for success in solving problem under consideration

  I have used f1_score, precision_score, recall_score, multilabel_confusion_matrix and hamming loss all these evaluation metrics to select best suitable algorithm for our final model.

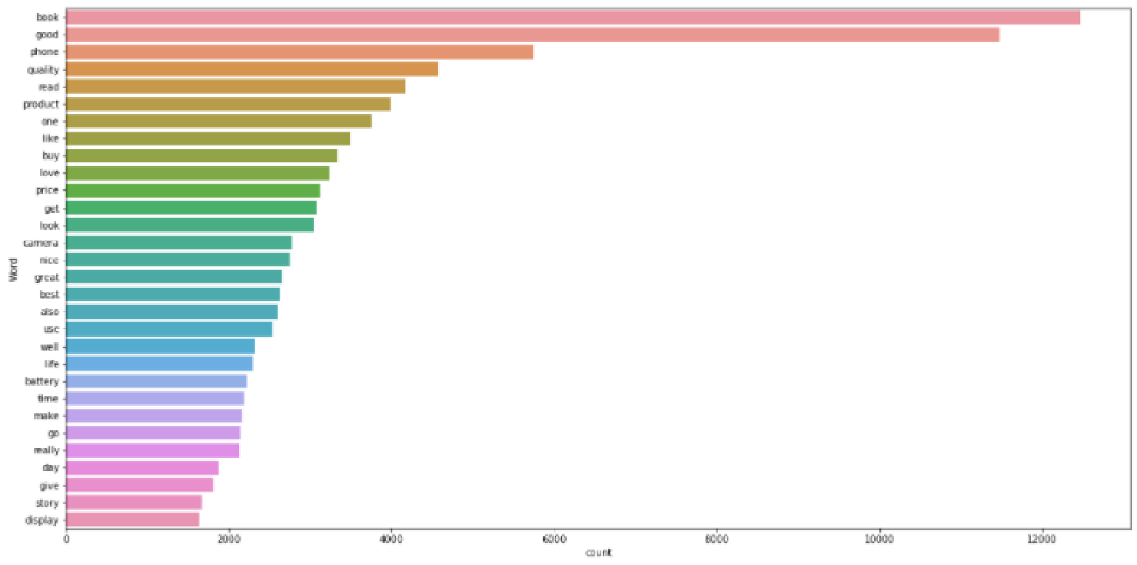
- Visualizations

  1. Histogram:

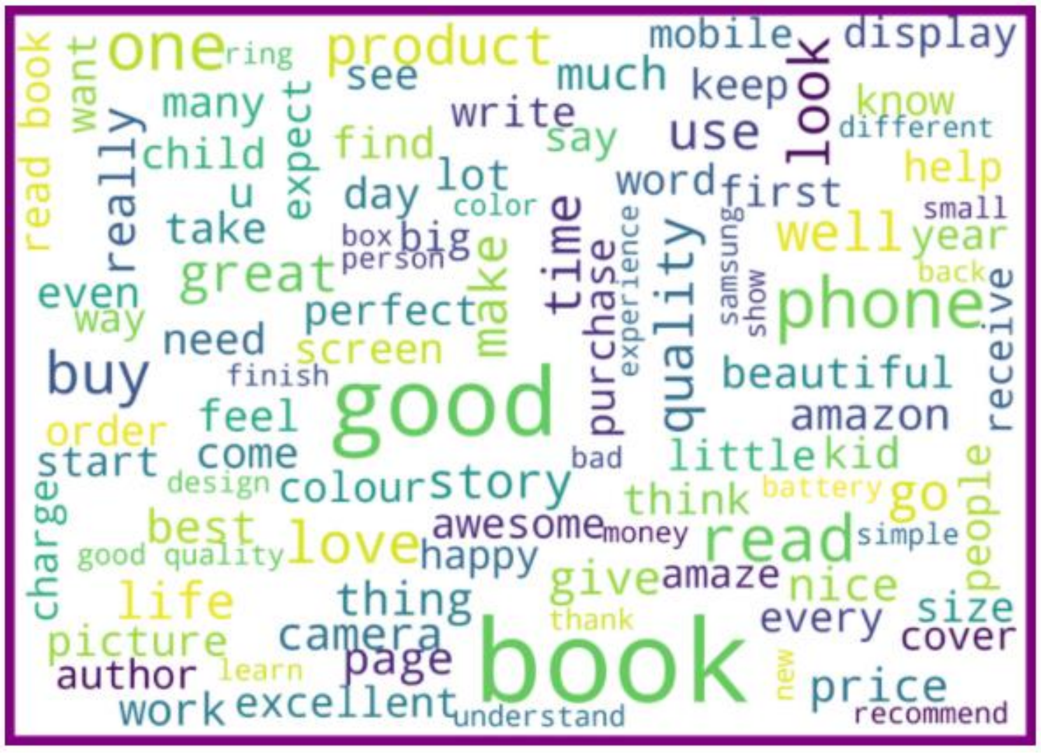


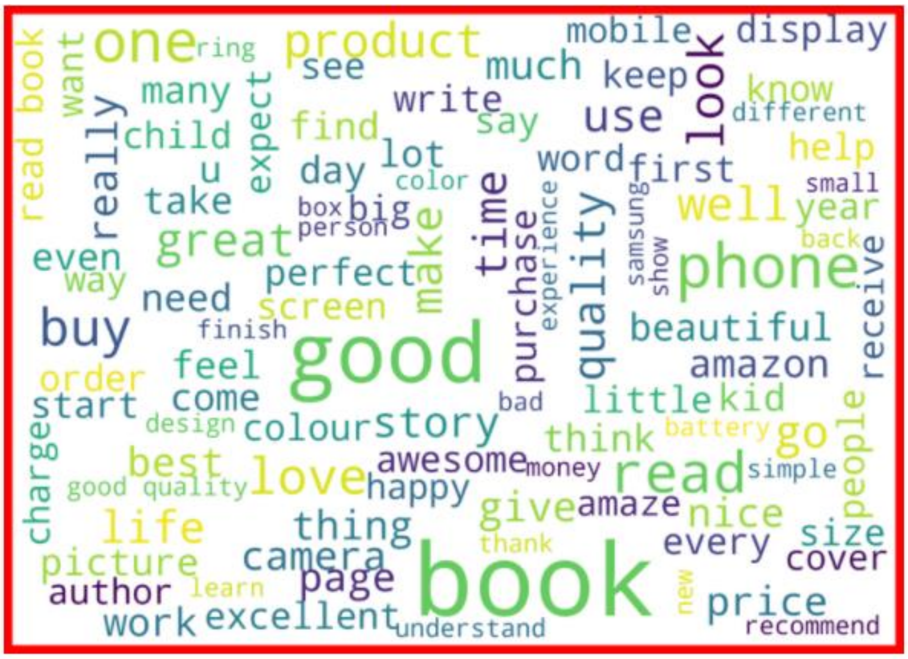


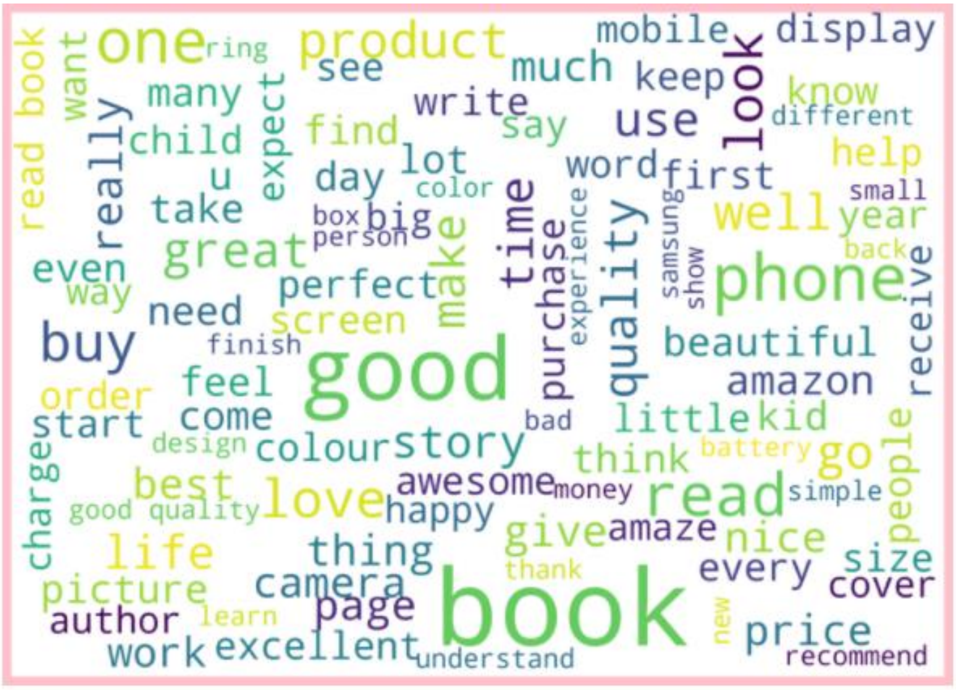
2. Bar Plot



3. Word Cloud

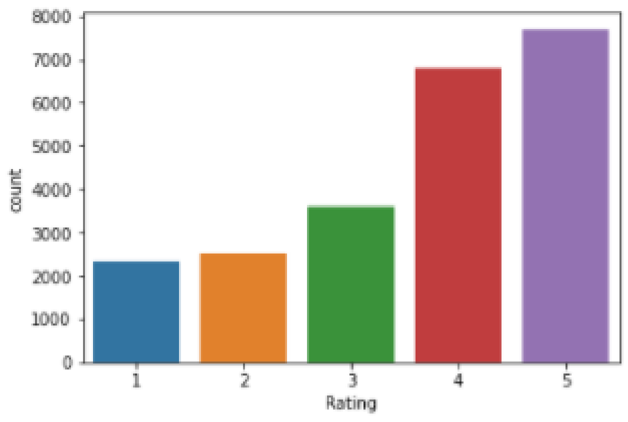



read book
want
one
ring
product
see
mobile
display
look
much
keep
many
expect
write
know
different
child
find
say
use
really
help
u
lot
day
word
first
small
take
color
box
big
person
well
year
great
time
back
make
purchase
experience
quality
samsung
show
even
way
perfect
phone
receive
need
screen
finish
buy
beautiful
good
amazon
feel
order
little
kid
start
come
bad
design
colour
story
think
battery
go
charge
best
awesome
money
people
read
simple
good quality
love
happy
give
amaze
nice
life
thing
every
size
picture
camera
cover
author
learn
page
book
new
price
work
excellent
understand
recommend


read book
want
one
ring
product
see
mobile
display
look
much
keep
many
expect
write
know
different
child
find
say
use
really
help
u
lot
day
word
first
small
take
color
box
big
person
well
year
great
time
back
make
purchase
experience
quality
samsung
show
even
way
perfect
phone
receive
need
screen
finish
buy
beautiful
good
amazon
feel
order
little
kid
start
come
bad
design
colour
story
think
battery
go
charge
best
awesome
money
people
read
simple
good quality
love
happy
give
amaze
nice
life
thing
every
size
picture
camera
cover
author
learn
page
book
new
price
work
excellent
understand
recommend

book
good
phone
read
one
product
quality
time
buy
great
love
look
use
mobile
display
much
keep
write
really
many
child
expect
find
say
know
different
help
small
year
well
first
word
lot
day
color
u
take
box
big
person
make
purchase
experience
samsung
show
back
receive
even
way
perfect
need
screen
finish
beautiful
amazon
feel
order
start
come
design
colour
story
bad
little
kid
think
battery
go
people
charge
best
awesome
money
simple
good quality
happy
give
amaze
nice
life
thing
thank
every
size
picture
camera
cover
author
learn
page
new
price
work
excellent
understand
recommend
read book
want
ring
see

4. Count Plot



- Interpretation of the Results

  The words that indicate the Reviewer's opinion on items are plainly visible in the plots above.

  In this word cloud, the most frequently used terms for each Rating are presented.

CONCLUSION

- Key Findings and Conclusions of the Study

  In this project we have tried to detect the ratings in commercial websites on a scale of 1 to 5 on the basis of reviews given by users. We made use of NLP and Machine Learning to do so. We interpreted that Random Forest Classifier gives us the best result as compared to the all the other algorithms we have used.

- Learning Outcomes of the Study in respect of Data Science

  In this project we were able to learn various NLP techniques like lemmatization, stemming, removal of stop words, etc. This project has demonstrated the importance of sampling effectively, modelling and predicting data.

  The power of visualisation has aided us in comprehending data through graphical representation, allowing me to comprehend what the data is attempting to convey. To eliminate unrealistic values, punctuations, urls, email addresses, and stop words, data cleansing is one of the most crucial procedures. This research is an experimental attempt to compare the outcomes of seven machine learning methods in calculating Rating.

- Limitations of this work and Scope for Future Work

  While we were unable to achieve our aim of maximum accuracy in the Ratings prediction project, we were able to develop a system that, with little tweaking and deep learning algorithms, can get quite close. There is always space for improvement in any endeavour. Because of the nature of this project, various algorithms can be merged as modules and their findings mixed to improve the accuracy of the final output. More algorithms may be added to this model to improve it even further. These algorithms' output, however, must be in the same format as the others. The modules are simple to add once that criterion is met, as seen in the code. This provides a great degree of modularity and versatility to the project.

As we all know, the content of words in reviews is entirely up to the reviewer, and they may evaluate things differently depending on who they are. As a result, predicting ratings based on more accurate evaluations is challenging. Even with additional data and intensive hyperparameter adjustment, we can still enhance our accuracy.