FLIP ROBO

NAME OF THE PROJECT

Product Review Rating Prediction Using NLP

Submitted by:

Prathamesh Nayak

I would like to express my special gratitude to the “Flip Robo” team, who has given me this opportunity to deal with a beautiful dataset and it has helped me to improve my analysis skills. And I want to express my huge gratitude to Ms. Khushboo Garg (SME Flip Robo), she is the person who has helped me to get out of all the difficulties I faced while doing the project.

A huge thanks to “Data trained” who are the reason behind my Internship at Fliprobo. Last but not least my parents have been my backbone in every step of my life.

References use in this project:

1. SCIKIT Learn Library Documentation
2. Blogs from towards data science, Analytics Vidya, Medium
3. Andrew Ng Notes on Machine Learning (GitHub)
4. Data Science Projects with Python Second Edition by Packt
5. Hands on Machine learning with scikit learn and tensor flow by Aurelien Geron
6. Lackermair, G., Kailer, D. & Kanmaz, K. (2013). Importance of online product reviews from a consumer’s perspective. Horizon Research Publishing, 1-5. doi: 10.13189/aeb.2013.010101
7. Baccianella, S., Esuli, A. & Sebastiani, F. (2009). Multi-facet rating of product reviews. Proceedings of the 31st European Conference on Information Retrieval (ECIR), 461- 472
8. Chevalier, J. & Mayzlin, D. (2006). The Effect of Word of Mouth on Sales: Online Book Reviews. Journal of Marketing Research, 43, 345-354. doi: 10.3386/w10148
9. Pang, B., Lee, L. & Vaithyanathan, S. (2002). Thumbs up? Sentiment Classification using Machine Learning Techniques. Proceedings of EMNLP, 271-278. doi: 10.3115/1118693.1118704
10. Dave, K., Lawrence, S. & Pennock, D. M. (2003). Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews. Proceedings of the 12th International Conference on World Wide Web, 519–528. doi: 10.1145/775152.775226
11. McAuley, J., Pandey, R., Leskovec J. (2015). Inferring networks of substitutable and complementary products. Knowledge Discovery and Data Mining.
12. A NithyaKalyani, S Ushasukhanya, TYJ Nagamalleswari, and S Girija. 2018. Rating prediction using textual reviews. In Journal of Physics: Conference Series, Vol. 1000. IOP Publishing, 012044.
13. Lizhen Qu, Georgiana Ifrim, and Gerhard Weikum. 2010. The bag of-opinions method for review rating prediction from sparse text patterns. In Proceedings of

the 23rd international conference on computational linguistics. Association for Computational Linguistics, 913–921
14. F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Recommender Systems Handbook,Springer, 2011.
15. Leon, Vasant, Sheldon, Using Properties of the Amazon Graph to Better Understand Reviews

Introduction

Business Problem Framing

The Internet is the best source nowadays for any organization to know public opinions about their products and services. Many consumers form an opinion about a product just by reading a few reviews. Online product reviews provided by consumers who previously purchased products have become a major information source for consumers and marketers regarding product quality. Research has shown that consumer online product ratings reflect both the customers' experience with the product and the influence of others' ratings. Websites prominently display consumers' product ratings, which influence consumers' buying decisions and willingness to pay.

The opinion information is very useful for users and customers alike, many of whom typically read product or service reviews before buying them. Businesses can also use the opinion information to design better strategies for production and marketing. Hence, in recent years, sentiment analysis and opinion mining have become a popular topic for machine learning and data mining.
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

A recent survey (Hinckley, 2015) revealed that 67.7% of consumers are effectively influenced by online reviews when making their purchase decisions. More precisely, 54.7% recognized that these reviews were either fairly, very or absolutely important in their purchase decision making. Relying on online reviews has thus become a second nature for consumers.

Consumers want to find useful information as quickly as possible. However, searching and comparing text reviews can be frustrating for users as they feel submerged with information. Indeed, the massive amount of text reviews as well as its unstructured text format prevent the user from choosing a product with ease. The star-rating, i.e., stars from 1 to 5 on online platforms, rather than its text content gives a quick overview of the product quality. This numerical information is the number one factor used in an early phase by consumers to compare products before making their purchase decision.

Generally, the ratings and the price of the product are simple heuristics used by the customers to decide over the final purchase of the product. But often, the overall star ratings of the product reviews may not capture the exact polarity of the sentiments. This makes rating prediction a hard problem as customers may assign different ratings for a particular review. For example,
For instance, a user may rate a product as good and assign a 5- star score while another user may write the same comment and give only 3 stars. In addition, reviews may contain anecdotal information, which do not provide any helpful information and complicates the predictive task.

The question that arises is how to successfully predict a user's numerical rating from its review text content. One solution is to rely on supervised machine learning techniques such as text classification which allows to automatically classify a document into a fixed set of classes after being trained over past annotated data.

Different users may have different sentiment expression preferences. For example, some users choose to use "good" to describe a just-so-so product, but others may use "good" to describe an excellent product. Beside the user bias, there is also a product bias. We may use different opinion words to review different products, or even use the same opinion word to express different

sentiment polarities for different products; for example, the opinion word “long” can be express a “positive” feeling for cell phone’s battery life, but may have a “negative” feeling for a camera’s focus time. Therefore, it is important to consider the relationship between the review-authors, as well as that of the target products, for review rating prediction.

Review of Literature

According to Lackermair, Kailer and Kanmaz (2013), product reviews and ratings represent an important source of information for consumers and are helpful tools in order to support their buying decisions [6]. They also found out that consumers are willing to compare both positive and negative reviews when searching for a specific product. The authors argue that customers need compact and concise information about the products. Therefore, consumers first need to pre-select the potential products matching their requirements. With this aim in mind, consumers use the star ratings as an indicator for selecting products. Later, when a limited number of potential products have been chosen, reading the associated text review will reveal more details about the products and therefore help consumers make a final decision.

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 (Baccianella, Esuli & Sebastiani, 2009) [7].

Chevalier and Mayzlin (2006) [8] also analyze the distribution of ratings in online reviews and come to the same conclusion: the resulting data presents an asymmetric bimodal distribution, where reviews are overwhelmingly positive. Pang, Lee and Vaithyanathan (2002) [9] approach this predictive task as an opinion mining problem enabling to automatically distinguish between positive and negative reviews. In order to determine the reviews polarity, the authors use text classification techniques by training and testing binary classifiers on movie reviews containing 36.6% of negative reviews and 63.4% of positive reviews. On top of that, they also try to identify appropriate features to enhance the performance of the classifiers.

Dave, Lawrence, and Pennock (2003) [10] also deal with the issue of class imbalance with a majority of positive reviews and show similar results. SVM outperforms Naïve Bayes with an accuracy greater than 85% and the implementation of part-of-speech as well as stemming is also ineffective. Nevertheless, while the previous research led to better results with unigrams, this

study shows that bigrams turn out to be more effective at capturing context than unigrams in the specific case of their datasets.

In a slightly different perspective, McAuley and Leskovec (2013) [11] intend to understand hidden dimensions of customer's opinions to better predict the numerical ratings. They also find out that the user information is a rich source of information to take into account when predicting ratings with review text.
On the other hand, [12], presented a model which predicts the star rating of a review using sentiment dictionaries. Like most polarity determining approaches, the paper uses the unigram model to represent text. The unigram model often fails to capture phrase patterns properly which leads to polarity incoherence. To overcome this drawback, the authors also employ a n-gram model. But this way of representing text vectors leads to the creation of large sparse matrices that are space inefficient known as n-gram sparsity bottlenecks

To overcome the problem of polarity incoherence, [13] introduces the Bag of opinions model. This model overcomes the limitations of the unigram and n-gram models. It has 3 components: root word, modifiers and negation words. Each opinion from the corpora of reviews is assigned a score using ridge regression method. At the end, a final score is calculated by combining all the independent scores of all the opinions.

Memory-based and model-based (e.g., matrix factorization) [14] are the two most popular methods for collaborative filtering. However, both systems take a sparse user-product rating matrix as an input and may fail to capture the quality of the users and their reviews.

In a similar project 'Using Properties of the Amazon Graph to Better Understand Reviews' [15] proposed by Leon, Vasant, Sheldon (2011), the predicted feedback score was represented with a linear combination of the following features: Original Rating, Review Helpfulness Count, Review Unhelpfulness Count, Review Helpfulness Ratio, Reviewer Helpfulness Ratio, Reviewer Bias, Reviewer Expertise, Reviewer Clustering, Reviewer Betweenness, and Reviewer Degree. However, the goal of their project is to predict the rating of a customer who reviews the same product more than once, while our project is to predict the user's first-time rating score. Also, they use stochastic gradient descent to minimize the mean square error to capture the weights of such features.

Motivation for the Problem Undertaken

The project was the first provided to me by Flip Robo Technologies as a part of the internship programme. The exposure to real world data and the opportunity to deploy my skills in solving a real time problem has been the primary motivation.

Data needed for this project is required to scrap from the E-commerce platform and data cleaning operation over it. Features derived from textual reviews are used to predict its corresponding star ratings. To accomplish it, the prediction problem is transformed into a multi-class classification task to classify reviews to one of the five classes corresponding to its star rating. Getting an overall sense of a textual review could in turn improve consumer experience. However, the motivation for taking this project was that it is relatively a new field of research.

Analytical Problem Framing

Mathematical / Analytical Modeling of the Problem

In order to apply text classification, the unstructured format of text has to be converted into a structured format for the simple reason that it is much easier for computers to deal with numbers than text. This is mainly achieved by projecting the textual contents into the Vector Space Model, where text data is converted into vectors of numbers.

In the field of text classification, documents are commonly treated like a Bag-of-Words (BoW), meaning that each word is independent from the others that are present in the document. They are examined without regard to grammar nor word order. In such a model, the term frequency (occurrence of each word) is used as a feature in order to train the classifier. However, using the term frequency implies that all terms are considered equally important. As its name suggests, the term frequency simply weights each term based on their occurrence frequency and does not take the discriminatory power of terms into account. To address this problem and penalize words that are too frequent, each word is given a term frequency inverse document frequency (tf-idf) score which is defined as follow:

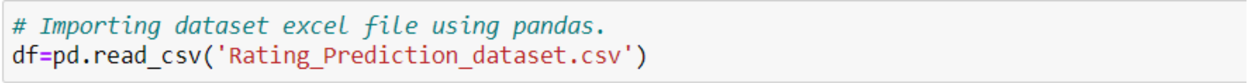
$$\mathrm{tf} \; - \mathrm{id}f_{t,d} = \mathrm{t}f_{\mathrm{t,d}} * \mathrm{id}f_{\mathrm{t}}$$

where:

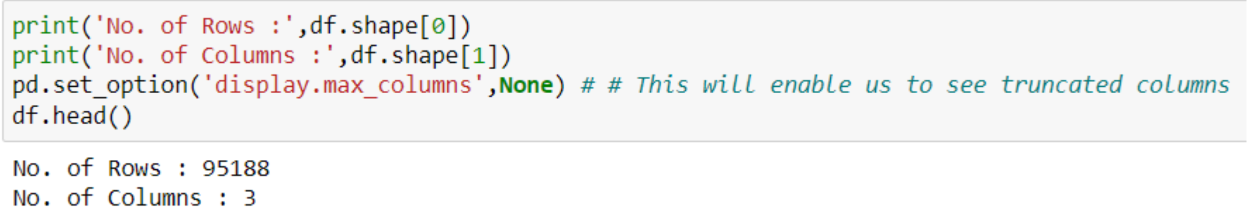
- $\mathrm{t}f_{\mathrm{t,d}} = \frac{\mathrm{n}_{\mathrm{t,d}}}{\sum_k n_{k,d}}$ with $\mathrm{n}_{\mathrm{t,d}}$ the number of term t contained in a document d, and $\sum_k n_{k,d}$ the total number of terms k in the document d
- $\mathrm{id}f_{\mathrm{t}} = \log \frac{N}{df_{\mathrm{t}}}$ with N the total number of documents and $\mathrm{d}f_{\mathrm{t}}$ the number of documents containing the term t

Data Sources and their formats

Data is collected from Amazon.in and flipkart.com using selenium and saved in a CSV file. Around 95000 Reviews are collected for this project.



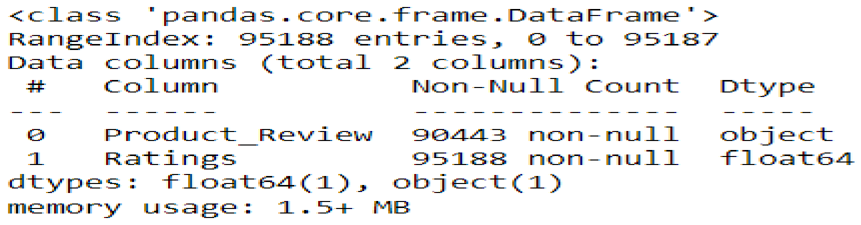
```
# Importing dataset excel file using pandas.
df=pd.read_csv('Rating_Prediction_dataset.csv')
```



```
print('No. of Rows :',df.shape[0])
print('No. of Columns :',df.shape[1])
pd.set_option('display.max_columns',None) # # This will enable us to see truncated columns
df.head()
```

```
No. of Rows : 95188
No. of Columns : 3
```

This is a multi-classification problem and Rating is our target feature class to be predicated in this project. There are five different categories in feature target i.eThe rating is out of 5 stars and it only has 5 options available 1 star, 2 stars, 3 stars, 4 stars, 5 stars.

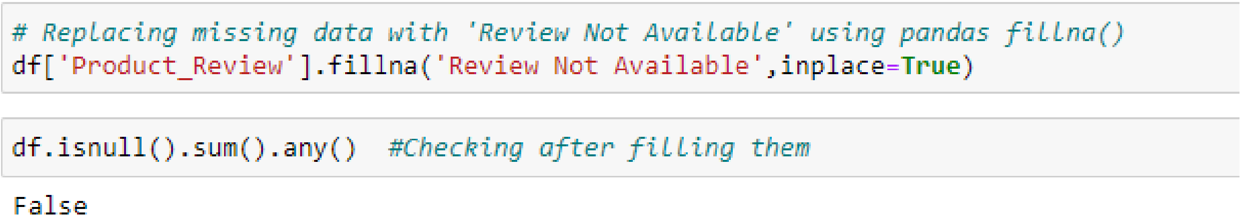


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 95188 entries, 0 to 95187
Data columns (total 2 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Product_Review  90443 non-null  object 
 1   Ratings         95188 non-null  float64
dtypes: float64(1), object(1)
memory usage: 1.5+ MB
```

There are some missing values in product review. The datatype of Product review is object while datatypes of Ratings is float.

Data Pre-processing

The dataset is large and it may contain some data errors. In order to reach clean, error free data some data cleaning & data pre-processing performed data.
• Missing Value Imputation:
Missing value in product reviews are replace with 'Review Not Available'



```
# Replacing missing data with 'Review Not Available' using pandas fillna()
df['Product_Review'].fillna('Review Not Available',inplace=True)
```

```
df.isnull().sum().any()  #Checking after filling them
```

```
False
```

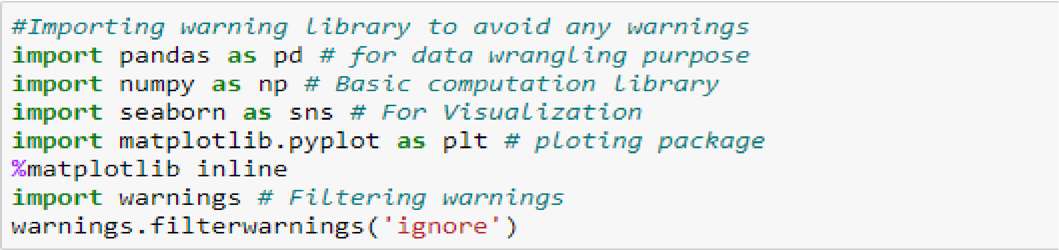
• Data is pre-processed using the following techniques:

1. Convert the text to lowercase
2. Remove the punctuations, digits and special characters
3. Tokenize the text, filter out the adjectives used in the review and create a new column in data frame
4. Remove the stop words
5. Stemming and Lemmatization
6. Applying Text Vectorization to convert text into numeric

Data Inputs- Logic- Output Relationships

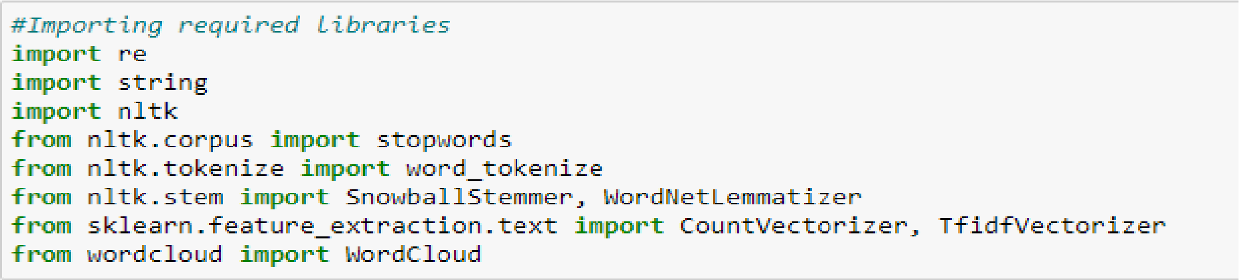
The dataset consists of 2 features with a label. The features are independent and the label is dependent as our label varies the values (text) of our independent variable’s changes. Using word clouds, we can see most occurring words for different categories.

Libraries Used – General library for data wrangling & visualization



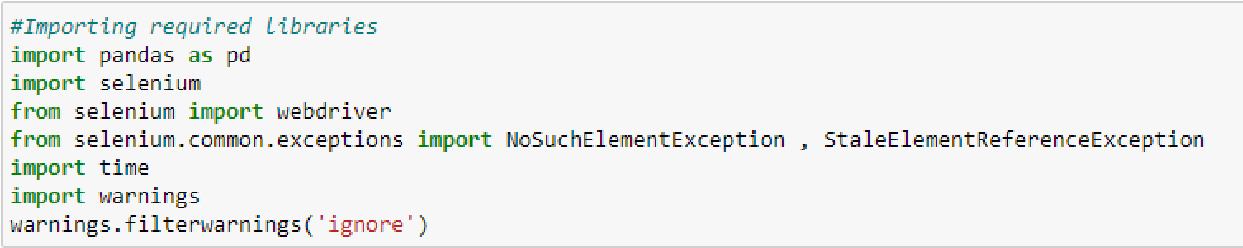
```
#Importing warning library to avoid any warnings
import pandas as pd # for data wrangling purpose
import numpy as np # Basic computation library
import seaborn as sns # For Visualization
import matplotlib.pyplot as plt # ploting package
%matplotlib inline
import warnings # Filtering warnings
warnings.filterwarnings('ignore')
```

Libraries used for Text Mining / Text Analytics are:



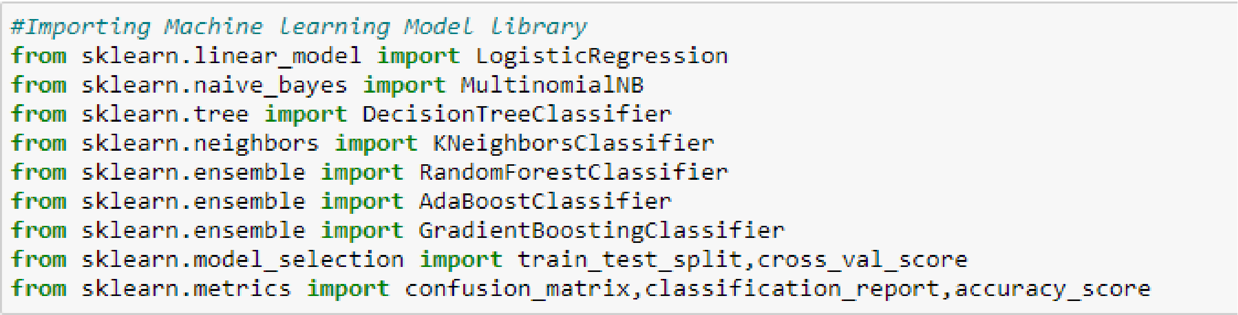
```
#Importing required libraries
import re
import string
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import SnowballStemmer, WordNetLemmatizer
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from wordcloud import WordCloud
```

Libraries used for web scraping data from e-commerce website are



```
#Importing required libraries
import pandas as pd
import selenium
from selenium import webdriver
from selenium.common.exceptions import NoSuchElementException , StaleElementReferenceException
import time
import warnings
warnings.filterwarnings('ignore')
```

Libraries used for machine learning model building



```
#Importing Machine Learning Model Library
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import train_test_split,cross_val_score
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
```

Models Development & Evaluation

Identification Of Possible Problem-Solving Approaches (Methods)

First part of problem solving is to scrape data from amazon.in and flipkart.com websites which we have already done. Second is performing text mining operations to convert textual review in ML algorithm usable form. Third part of problem building machine learning model to predict rating on review. This problem can be solved using classification-based machine learning algorithms like logistics regression. Further Hyperparameter tuning performed to build a more accurate model out of the best model.

Testing of Identified Approaches (Algorithms)

The different classification algorithm used in this project to build ML model are as below:

- ❖ **Random Forest classifier**
- ❖ **Decision Tree classifier**
- ❖ **Logistics Regression**
- ❖ **AdaBoost Classifier**
- ❖ **Gradient Boosting Classifier**

Key Metrics for Success in Solving Problem Under Consideration

▪ 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.
▪ Cross validation Score: To run cross-validation on multiple metrics and also to return train scores, fit times and score times. Get predictions from each split of

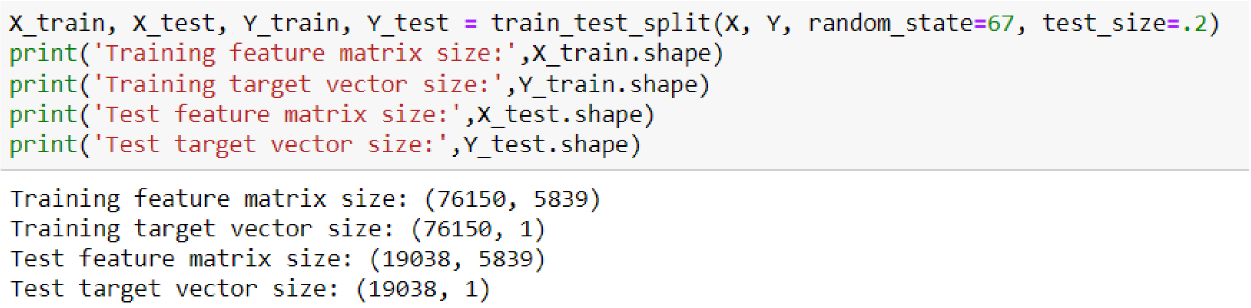
cross-validation for diagnostic purposes. Make a scorer from a performance metric or loss function.

▪ AUC_ROC _score: ROC curve. It is a plot of the false positive rate (xaxis) versus the true positive rate (y-axis) for a number of different candidate threshold values between 0.0 and 1.0

▪ We have used Accuracy Score and Cross validation score as key parameters for model evaluation in this project since balancing of data is performed.

Run And Evaluate Selected Models

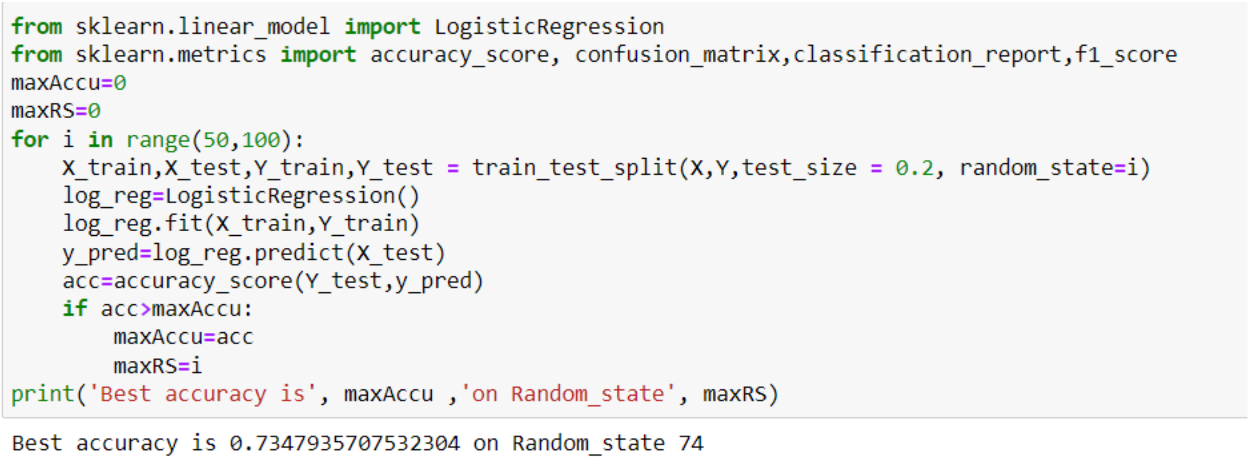
Logistics Regression



```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=67, test_size=.2)
print('Training feature matrix size:',X_train.shape)
print('Training target vector size:',Y_train.shape)
print('Test feature matrix size:',X_test.shape)
print('Test target vector size:',Y_test.shape)
```

```
Training feature matrix size: (76150, 5839)
Training target vector size: (76150, 1)
Test feature matrix size: (19038, 5839)
Test target vector size: (19038, 1)
```

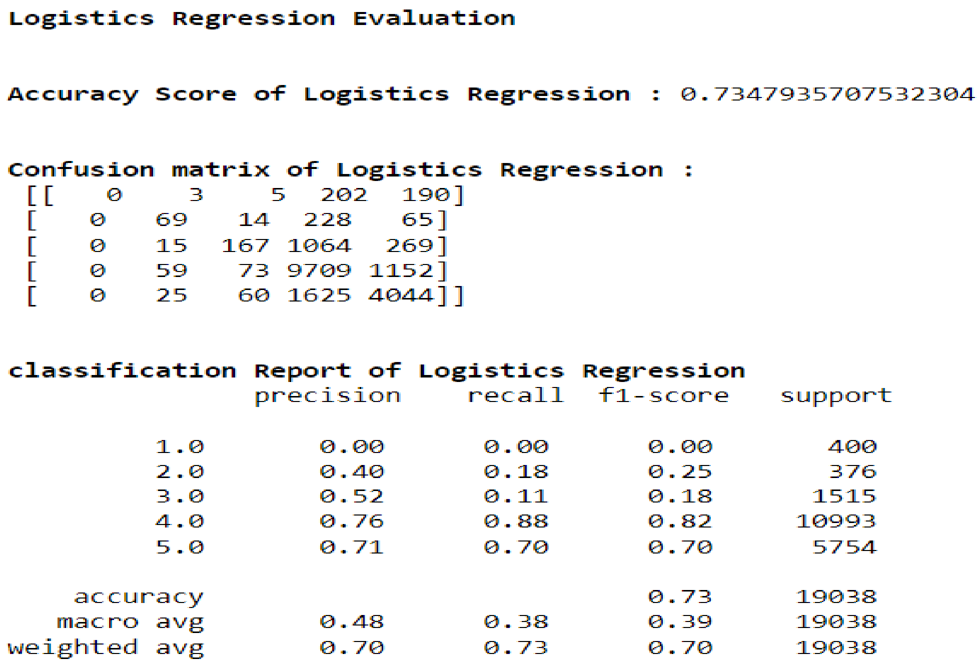
Train-test split is used to split data into training data & testing data. Further best random state is investigated through a loop.



```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix,classification_report,f1_score
maxAccu=0
maxRS=0
for i in range(50,100):
    X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.2, random_state=i)
    log_reg=LogisticRegression()
    log_reg.fit(X_train,Y_train)
    y_pred=log_reg.predict(X_test)
    acc=accuracy_score(Y_test,y_pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print('Best accuracy is', maxAccu ,'on Random_state', maxRS)
```

```
Best accuracy is 0.7347935707532304 on Random_state 74
```

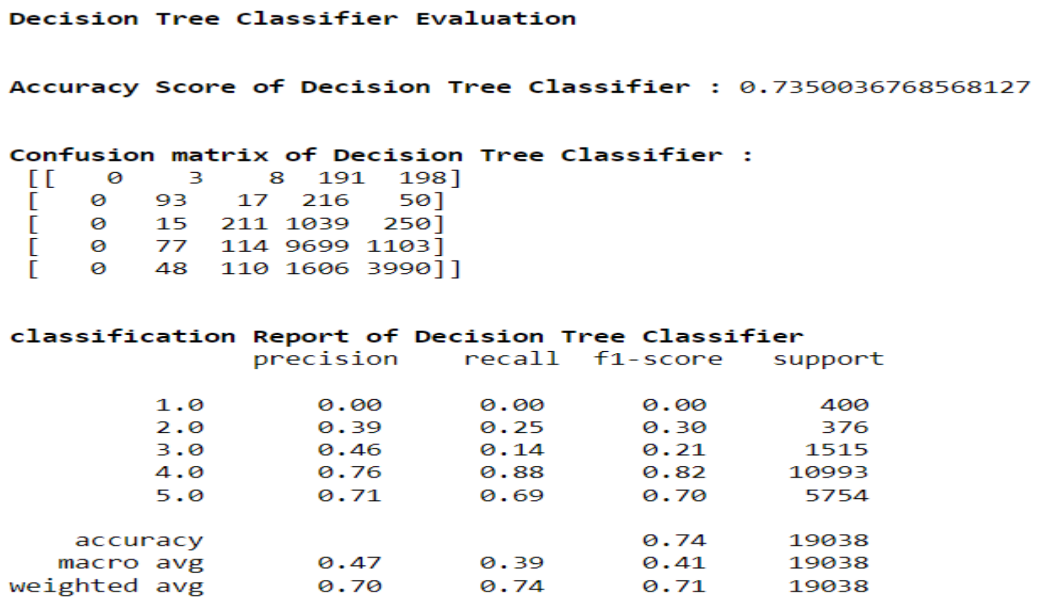
Logistics regression evaluation matrix is shown below:



```
Logistics Regression Evaluation


Accuracy Score of Logistics Regression : 0.7347935707532304


Confusion matrix of Logistics Regression :
 [[   0    3    5  202  190]
 [   0   69   14  228   65]
 [   0   15  167 1064  269]
 [   0   59   73 9709 1152]
 [   0   25   60 1625 4044]]


classification Report of Logistics Regression
               precision    recall  f1-score   support

         1.0       0.00      0.00      0.00       400
         2.0       0.40      0.18      0.25       376
         3.0       0.52      0.11      0.18      1515
         4.0       0.76      0.88      0.82     10993
         5.0       0.71      0.70      0.70      5754

    accuracy                           0.73     19038
   macro avg       0.48      0.38      0.39     19038
weighted avg       0.70      0.73      0.70     19038
```

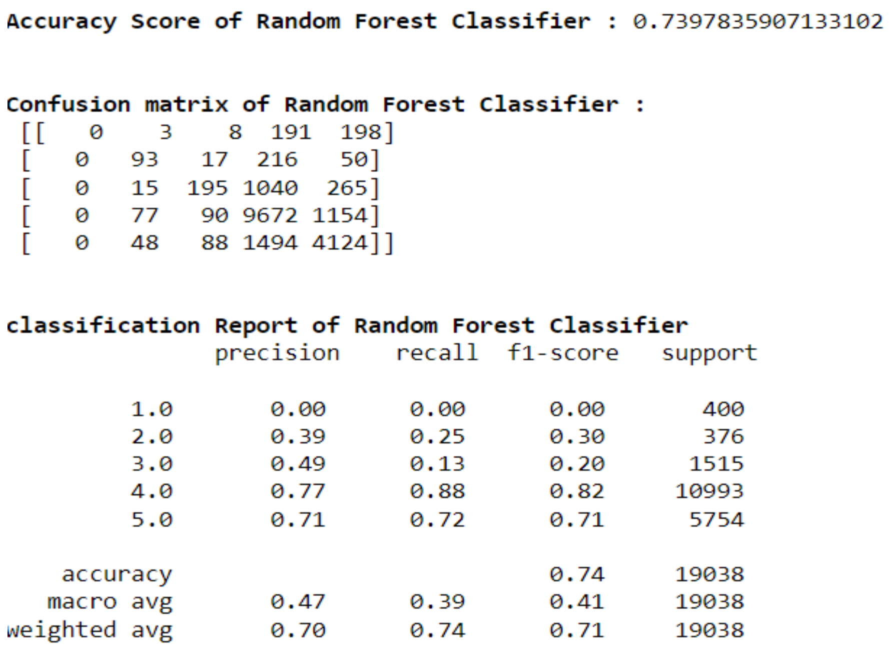
Decision Tree Classifier:

Decision Tree Classifier model is built and evaluation matrix is shown as below:



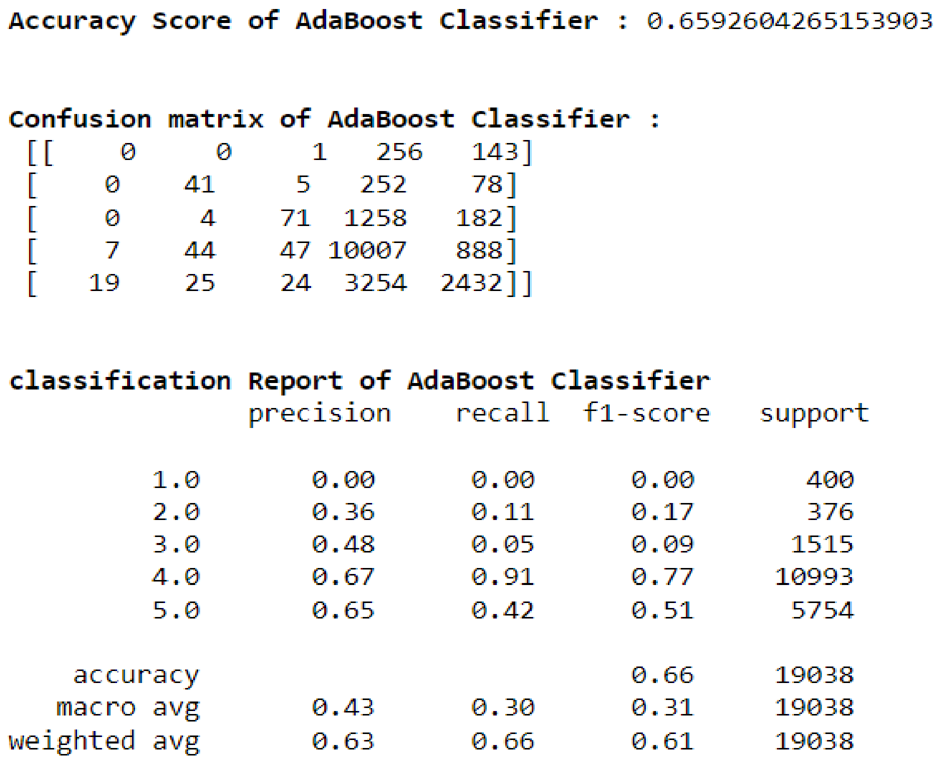
```
Decision Tree Classifier Evaluation


Accuracy Score of Decision Tree Classifier : 0.7350036768568127


Confusion matrix of Decision Tree Classifier :
 [[   0    3    8  191  198]
 [   0   93   17  216   50]
 [   0   15  211 1039  250]
 [   0   77  114 9699 1103]
 [   0   48  110 1606 3990]]


classification Report of Decision Tree Classifier
               precision    recall  f1-score   support

         1.0       0.00      0.00      0.00       400
         2.0       0.39      0.25      0.30       376
         3.0       0.46      0.14      0.21      1515
         4.0       0.76      0.88      0.82     10993
         5.0       0.71      0.69      0.70      5754

    accuracy                           0.74     19038
   macro avg       0.47      0.39      0.41     19038
weighted avg       0.70      0.74      0.71     19038
```

Random Forest Classifier



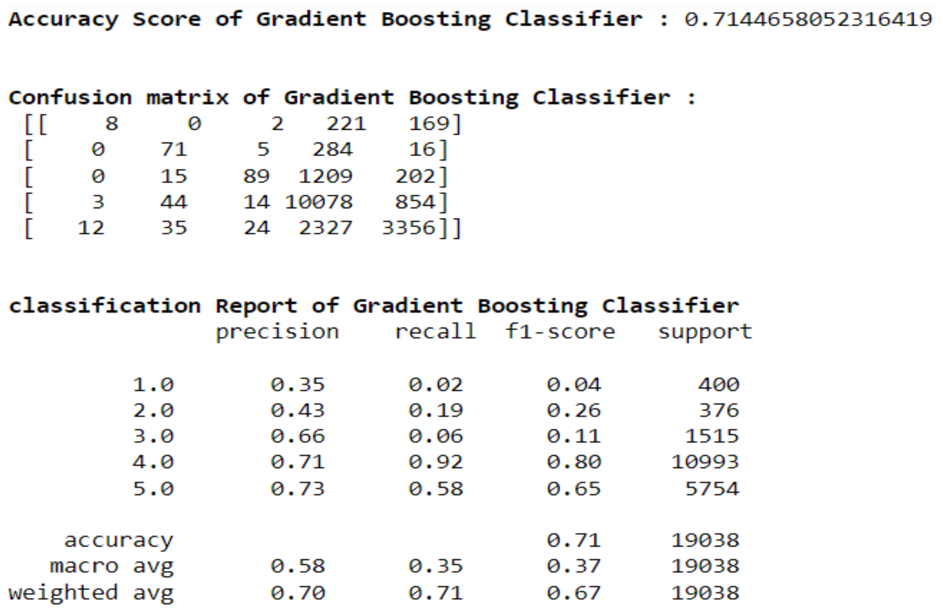
```
Accuracy Score of Random Forest Classifier : 0.7397835907133102


Confusion matrix of Random Forest Classifier :
 [[   0    3    8  191  198]
 [   0   93   17  216   50]
 [   0   15  195 1040  265]
 [   0   77   90 9672 1154]
 [   0   48   88 1494 4124]]


classification Report of Random Forest Classifier
              precision    recall  f1-score   support

         1.0       0.00      0.00      0.00       400
         2.0       0.39      0.25      0.30       376
         3.0       0.49      0.13      0.20      1515
         4.0       0.77      0.88      0.82     10993
         5.0       0.71      0.72      0.71      5754

    accuracy                           0.74     19038
   macro avg       0.47      0.39      0.41     19038
weighted avg       0.70      0.74      0.71     19038
```

AdaBoost Classifier



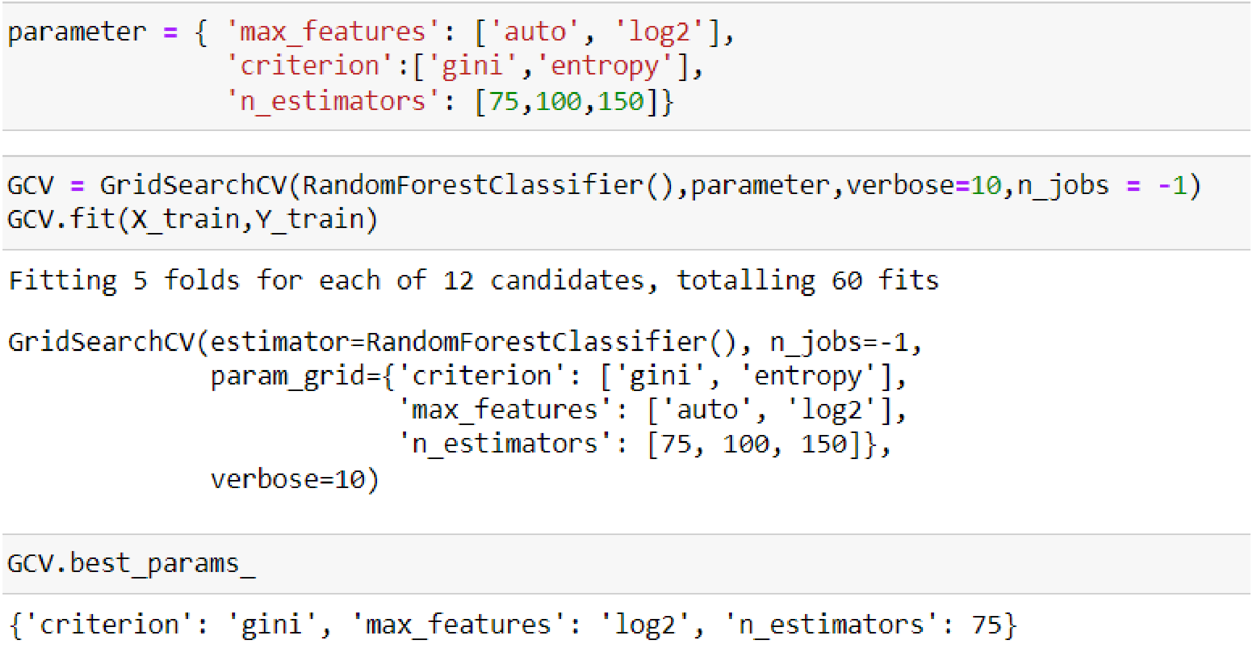
```
Accuracy Score of AdaBoost Classifier : 0.6592604265153903


Confusion matrix of AdaBoost Classifier :
 [[    0     0     1   256   143]
 [    0    41     5   252    78]
 [    0     4    71  1258   182]
 [    7    44    47 10007   888]
 [   19    25    24  3254  2432]]


classification Report of AdaBoost Classifier
              precision    recall  f1-score   support

         1.0       0.00      0.00      0.00       400
         2.0       0.36      0.11      0.17       376
         3.0       0.48      0.05      0.09      1515
         4.0       0.67      0.91      0.77     10993
         5.0       0.65      0.42      0.51      5754

    accuracy                           0.66     19038
   macro avg       0.43      0.30      0.31     19038
weighted avg       0.63      0.66      0.61     19038
```

Gradient Boosting Classifier



```
Accuracy Score of Gradient Boosting Classifier : 0.7144658052316419


Confusion matrix of Gradient Boosting Classifier :
 [[    8     0     2   221   169]
 [    0    71     5   284    16]
 [    0    15    89  1209   202]
 [    3    44    14 10078   854]
 [   12    35    24  2327  3356]]


classification Report of Gradient Boosting Classifier
               precision    recall  f1-score   support

         1.0       0.35      0.02      0.04       400
         2.0       0.43      0.19      0.26       376
         3.0       0.66      0.06      0.11      1515
         4.0       0.71      0.92      0.80     10993
         5.0       0.73      0.58      0.65      5754

    accuracy                           0.71     19038
   macro avg       0.58      0.35      0.37     19038
weighted avg       0.70      0.71      0.67     19038
```

5-fold Cross validation performed over all models. We can see that Random Forest Classifier gives us good Accuracy and maximum f1 score along with best Cross-validation score. Hyperparameter tuning is applied over the Random Forest model and used as the final model.



```
parameter = { 'max_features': ['auto', 'log2'],
              'criterion':['gini','entropy'],
              'n_estimators': [75,100,150]}
```

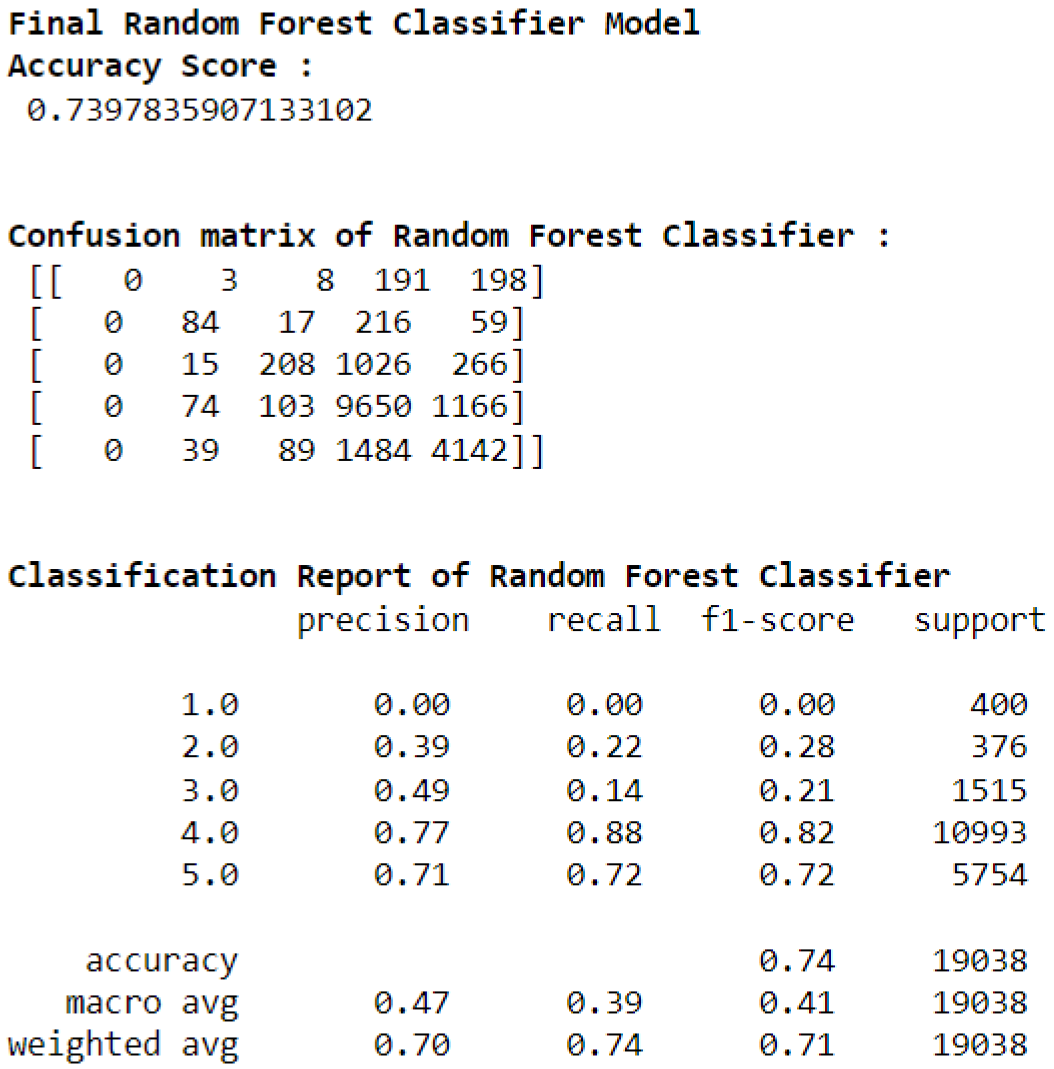
```
GCV = GridSearchCV(RandomForestClassifier(),parameter,verbose=10,n_jobs = -1)
GCV.fit(X_train,Y_train)
```

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits

GridSearchCV(estimator=RandomForestClassifier(), n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_features': ['auto', 'log2'],
                         'n_estimators': [75, 100, 150]},
             verbose=10)
```

```
GCV.best_params_
```

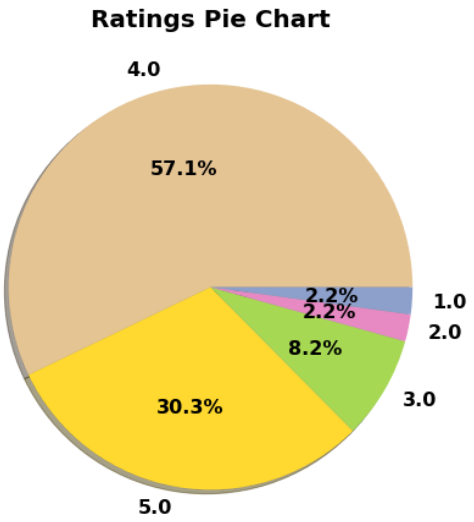
```
{'criterion': 'gini', 'max_features': 'log2', 'n_estimators': 75}
```

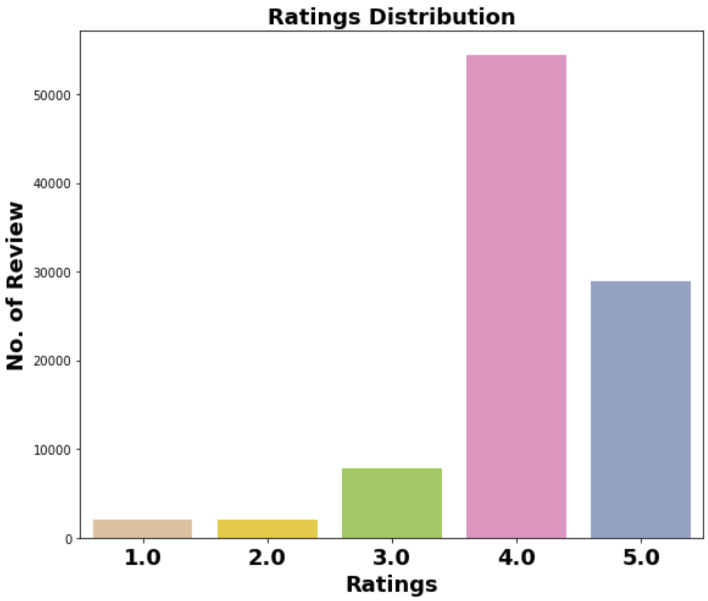
Final model is built using the best parameters in hyper parameters tuning. The corresponding evaluation matrix shown below:



```
Final Random Forest Classifier Model
Accuracy Score :
 0.7397835907133102


Confusion matrix of Random Forest Classifier :
 [[    0     3     8   191   198]
 [    0    84    17   216    59]
 [    0    15   208  1026   266]
 [    0    74   103  9650  1166]
 [    0    39    89  1484  4142]]


Classification Report of Random Forest Classifier
               precision    recall  f1-score   support

         1.0       0.00      0.00      0.00       400
         2.0       0.39      0.22      0.28       376
         3.0       0.49      0.14      0.21      1515
         4.0       0.77      0.88      0.82     10993
         5.0       0.71      0.72      0.72      5754

    accuracy                           0.74     19038
   macro avg       0.47      0.39      0.41     19038
weighted avg       0.70      0.74      0.71     19038
```

Visualizations

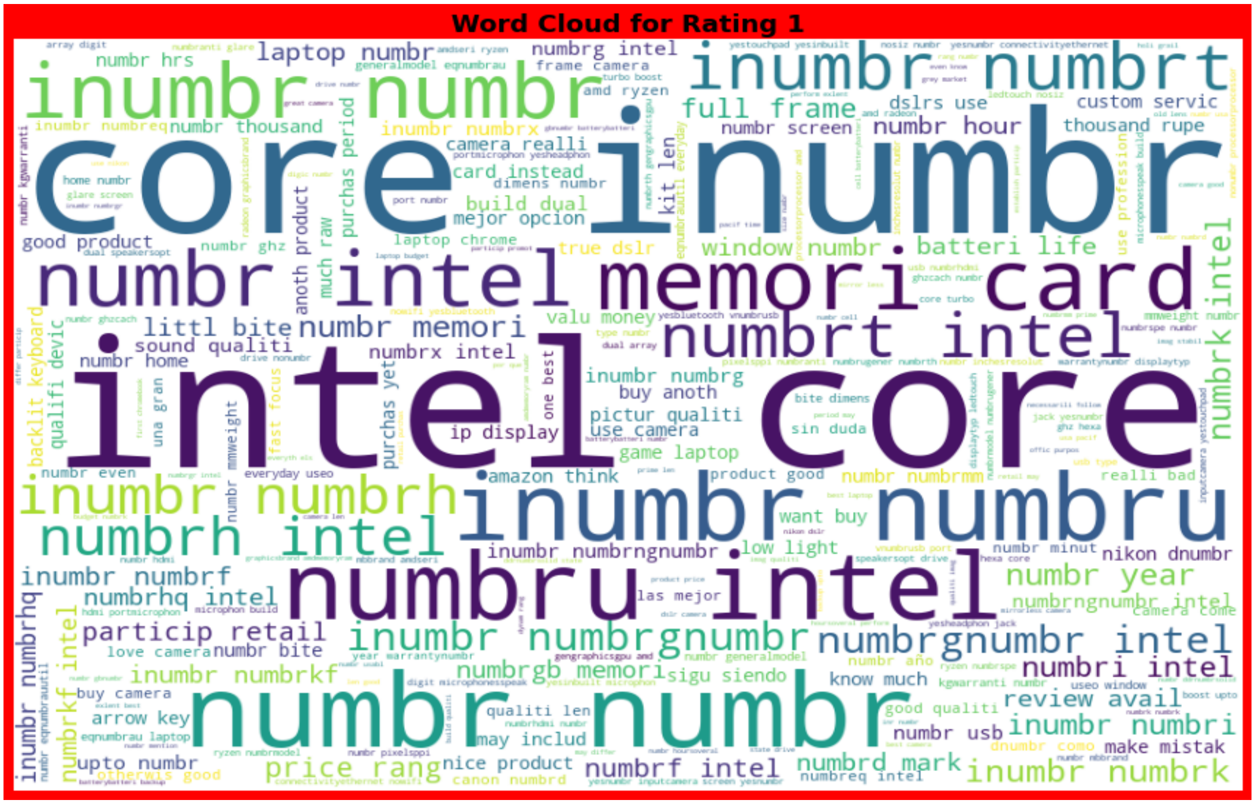




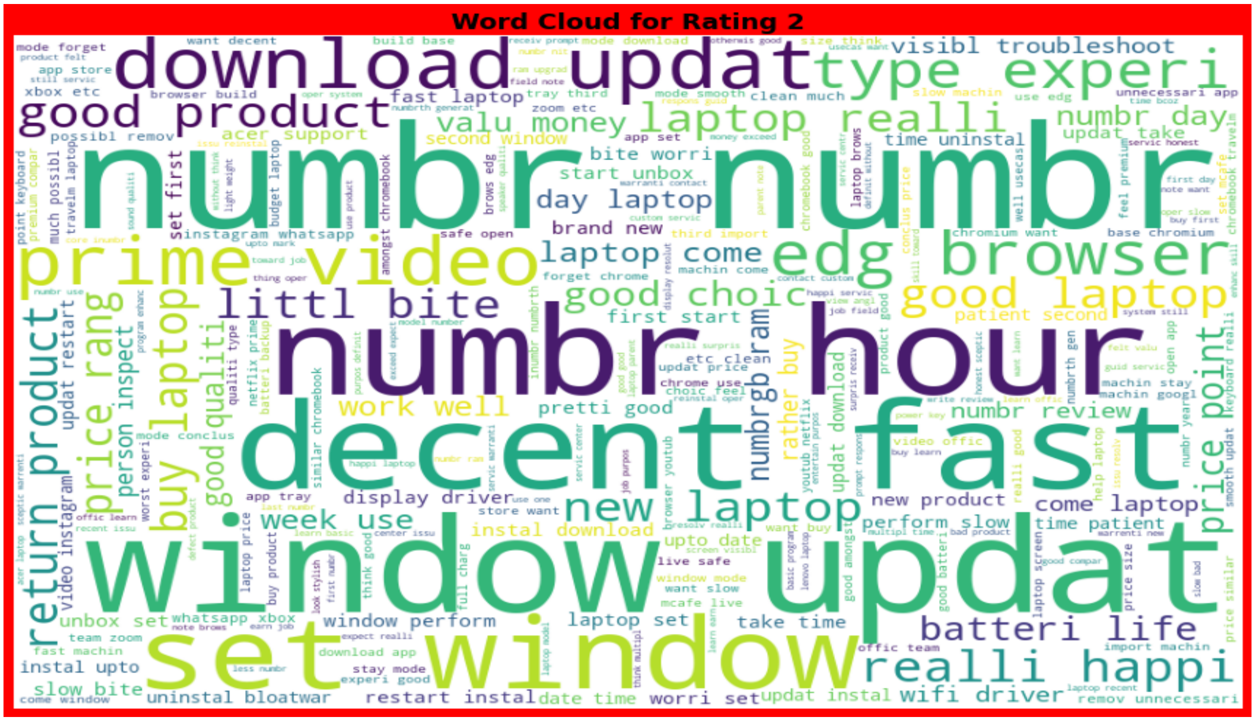
1. Around 57.1% of customers were given a 4- star rating followed by 2.2% of customers given the lowest 1-star rating.

2. Average Rating is 4.11

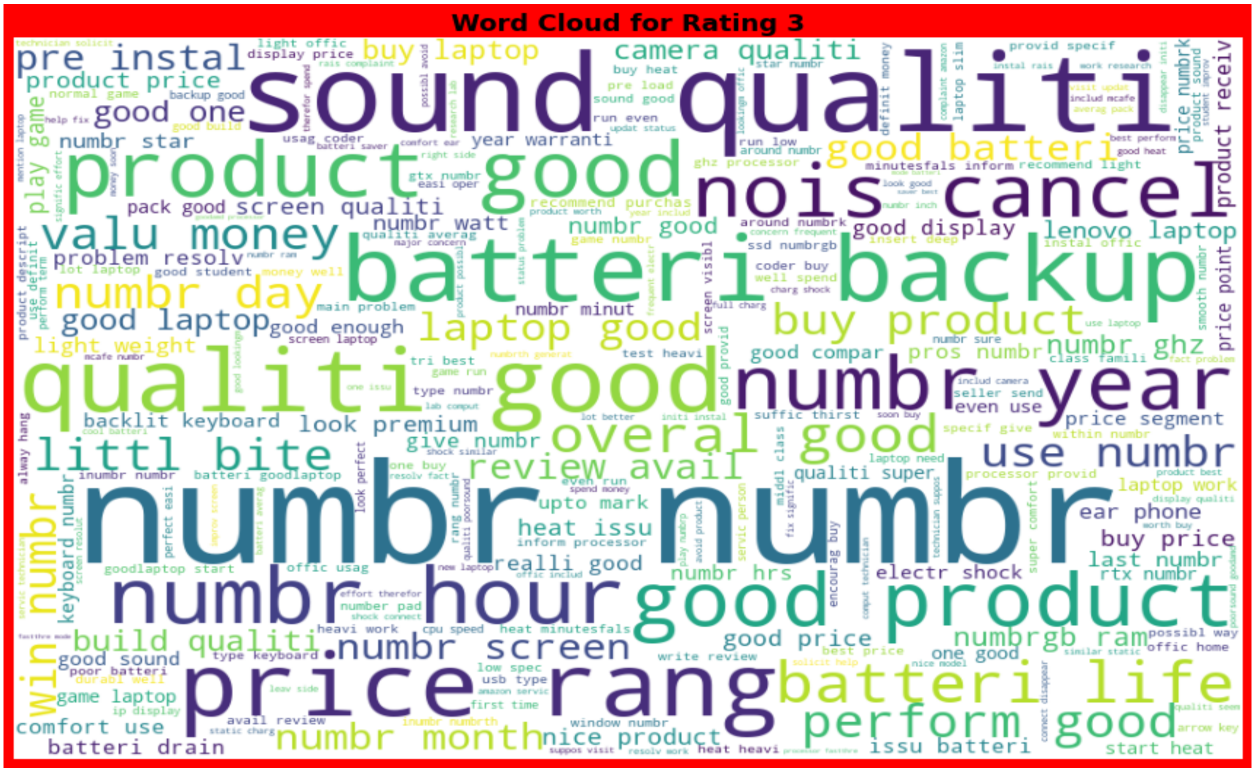
Word Cloud:
• Word Cloud is a visualization technique for text data wherein each word is picturized with its importance in the context or its frequency.
• The more commonly the term appears within the text being analyzed, the larger the word appears in the image generated.
• The enlarged texts are the greatest number of words used there and small texts are the smaller number of words used.



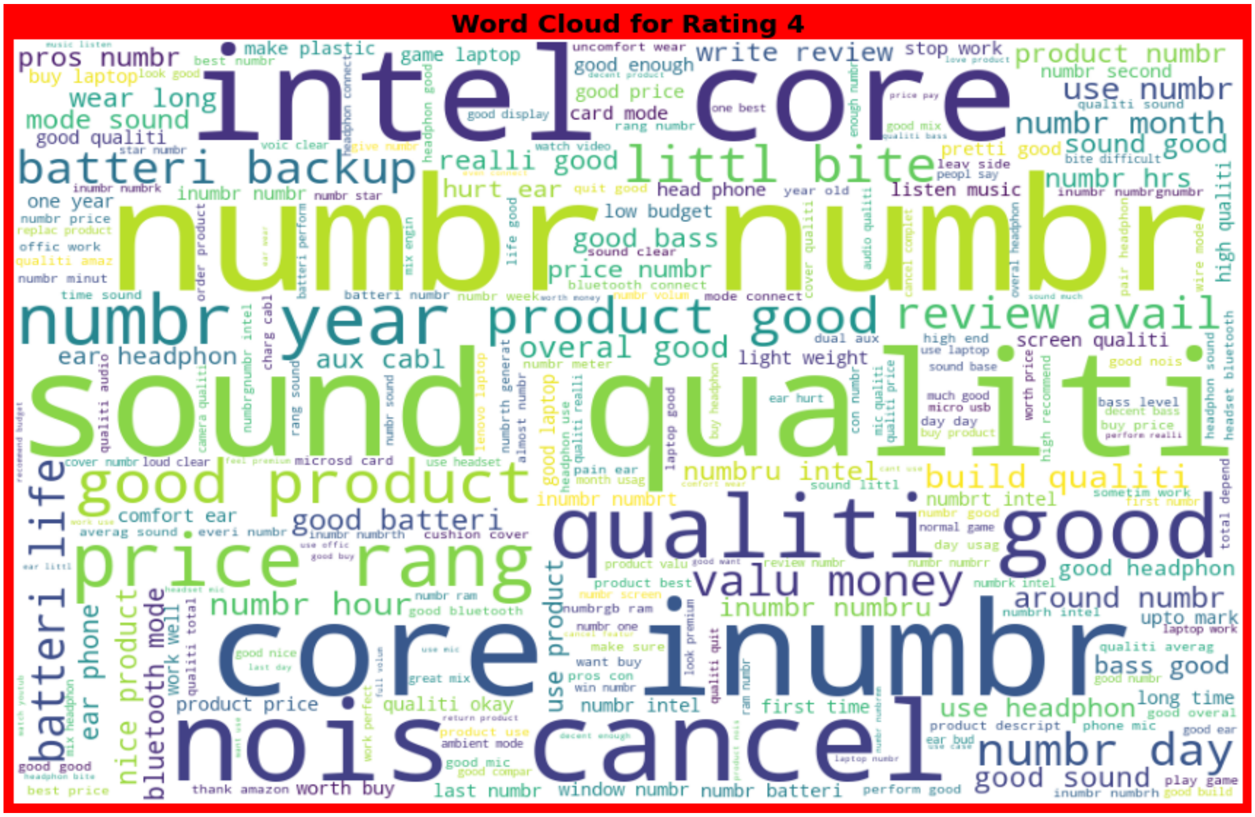
Word Cloud for Rating 1



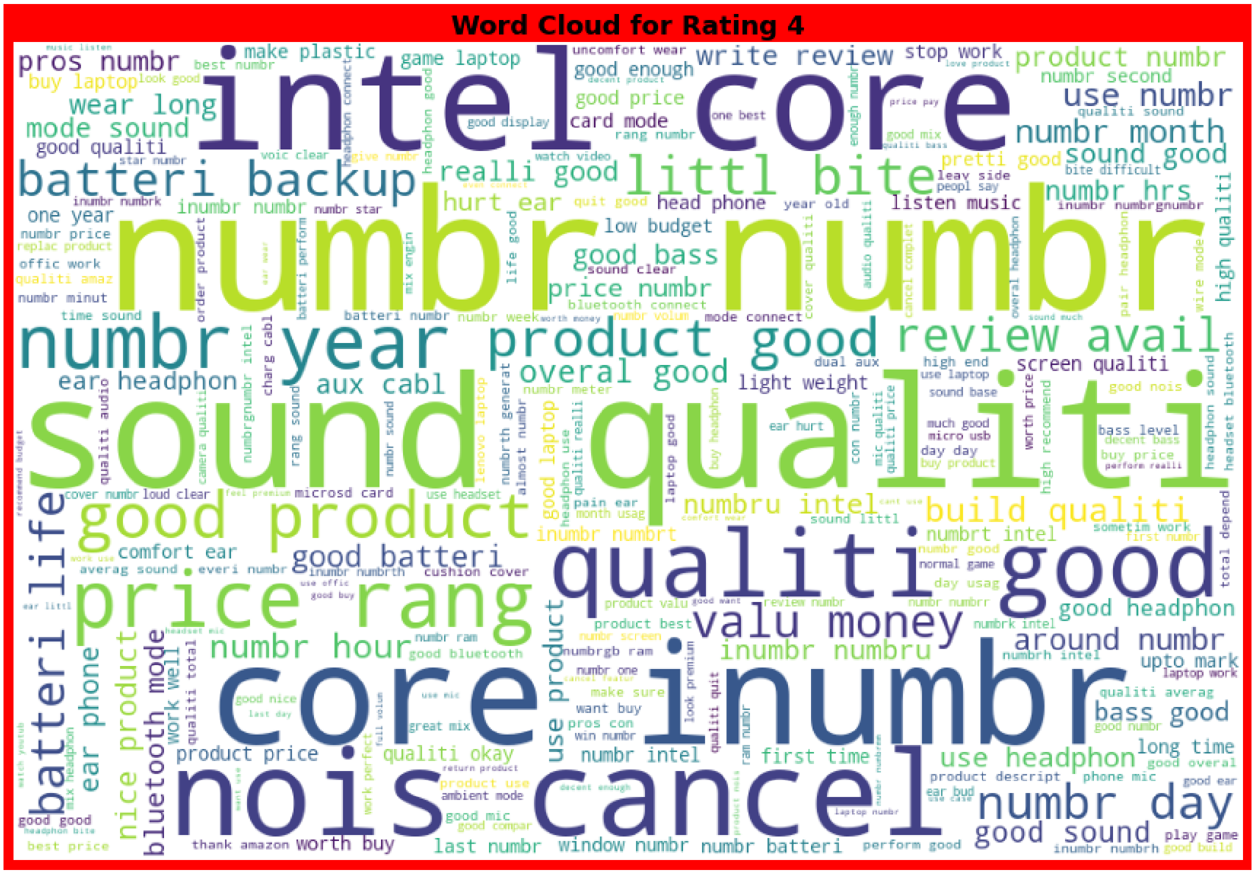
Word Cloud for Rating 2



Word Cloud for Rating 3



Word Cloud for Rating 4



Word Cloud for Rating 4

Conclusion

Key Findings and Conclusions of the Study

Final Model is giving us an Accuracy score of 73.97 % which is slightly improved compared to earlier Accuracy score of 73.50%.

Learning Outcomes of the Study in respect of Data Science

➢ Hands on chance to enhance my web scraping skillset.

➢ In this project we were able to learn various Natural language processing techniques like lemmatization, stemming, removal of Stop words.

➢ This project has demonstrated the importance of sampling effectively, modeling and predicting data.

Limitations of this work and Scope for Future Work

➢ More input features can be scrap to build prediction model.

➢ There is scope for application of advanced deep learning NLP tool to enhanced text mining operation which eventually help in building more accurate model with good cross validation score