ACKNOWLEDGMENT

The following project 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: The desired output to an input of a text review is a "star" rating on a continuum from 1 to 5.The source of data comes from e-commerce websites like www.amazon.com and www.flipkart.com,

First and foremost, I would like to warmly thank the "Flip Robo" team, who has given me this opportunity to deal with an interesting project on NLP and it has helped me to improve my analysis skills.

Also, I would like to acknowledge the wordnetweb.princeton.edu, www.kaggle.com/ getting-started-with-text-pre-processing, and www.analyticsvidhya.com who shared precious information on text pre-processing and helped me get more familiar with machine learning, Python language, and NLP.

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, and 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 that 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 e-commerce companies like amazon and Flipkart that support these reviews.

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 Modelling 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 47356 rows × 4 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 the Ratings of the reviewer

Data pre-processing done

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

● 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 plots and word clouds for each rating.

● 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 a classification algorithm

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 occurring and rarely occurring words with the help of a count plot.

● And was able to see the words in the Review text with reference to their ratings using a word cloud.

Importing Important libraries:

We need some libraries to be imported to work upon the dataset, we would import the dataset by using pandas' read_csv method.

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

Feature Information:

Review_Title : Title of the Review.

Review_Text : Text Content of the Review.

Ratings : Ratings out of 5 stars

Loading Data Set into a variable:

As per the client's requirement for this rating prediction project I have scraped reviews and ratings from well known e-commerce sites. This is then saved into .csv format. Also I have shared the script for web scraping into the Github repository. Then loaded this data into a data frame and did some of the important natural language processing steps and gone through several EDA steps to analyse the data. After all the necessary steps I have built an NLP ML model to predict the ratings.

Here I am loading the dataset into the variable **df_rp.**

```
# Dataset
df_rp = pd.read_csv("Ratings_Prediction.csv")
df_rp
```

	Unnamed: 0	Review_Title	Review_Text	Ratings
0	0	Really Nice	Actually, I felt like this infinix brand is no...	4
1	1	Wonderful	Laptop is nice but after sometimes heating sta...	4
2	2	Great product	brilliant design, so sleek and beautiful. stil...	5
3	3	Simply awesome	Everything excellent in price point	5
4	4	Awesome	Mind blowing purchase ever.. very nice perform...	5
...	...	...	...	...
47351	47351	Very bad body gaje	Big lens is worst product body gaje poor	2.0 out of 5 stars
47352	47352	For bigginers ok	Average performance..	2.0 out of 5 stars
47353	47353	I hate it's auto focus	I am hate this camera because of it's more blu...	2.0 out of 5 stars
47354	47354	No Have Warranty Details	No Have Warranty Details	2.0 out of 5 stars
47355	47355	NaN	NaN	NaN

47356 rows × 4 columns

Ratings is the target variable.

Exploratory Data Analysis:

E**xploratory Data Analysis** refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations. We performed some bi-variate analysis on the data to get a better overview of the data and to find outliers in our data-set. Outliers can occur due to some kind of errors while collecting the data and need to be removed so that it doesn't affect the performance of our model.

Checking the shape of the dataset:

Since unnamed column is not necessary in the dataset, so I am dropping the unnamed column and checking the shape of the dataset.

```
# Dropping unnecessary column
df_rp.drop(columns = 'Unnamed: 0',inplace = True)
```

```
# Checking shape of my dataset
df_rp.shape
```

```
(47356, 3)
```

We can see that the dataset has 47353 rows and 3 columns.

Getting detailed information about both the datasets:

The dataset has 47353 observations and 3 columns including the target variable. The target variable is Ratings which is of integer data type.

After applying the info() function to the datasets we get to know the data types of all the features and missing values of both the datasets.

```
# Checking all column names
df_rp.columns
```

```
Index(['Review_Title', 'Review_Text', 'Ratings'], dtype='object')
```

```
# Checking the info about the dataset
df_rp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 47356 entries, 0 to 47355
Data columns (total 3 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Review_Title  44071 non-null  object
 1   Review_Text   44520 non-null  object
 2   Ratings       44071 non-null  object
dtypes: object(3)
memory usage: 1.1+ MB
```

We concluded that we have 3 columns in the datasets, all are of object dtype. Also, there are so some missing values in all the columns.

Checking the Null Values:

```
# Checking null values in the datase
df_rp.isnull().sum()
```

```
Review_Title    3285
Review_Text     2836
Ratings         3285
dtype: int64
```

Treating the Missing values in the dataset:

We have null values in all the columns and they are categorical in nature. So we will apply Simple Imputer (most_frequent) strategy

```
# Checking the mode of Review_Title column
df_rp["Review_Title"].mode()
```

```
0    Wonderful
dtype: object
```

```
# Checking the mode of Review_Text column
df_rp["Review_Text"].mode()
```

```
0    Good
dtype: object
```

```
# Checking the mode of Ratings column
df_rp["Ratings"].mode()
```

```
0    5
dtype: object
```

We can see the most frequent values in Review_title is Wonderful, Review_text is Good and Ratings is 5 respectively.Applying mode method.

```
# Replacing nan values with there mode as all the columns are categorical.
df_rp["Review_Title"] = df_rp["Review_Title"].fillna(df_rp["Review_Title"].mode()[0])
df_rp["Review_Text"] = df_rp["Review_Text"].fillna(df_rp["Review_Text"].mode()[0])
df_rp["Ratings"] = df_rp["Ratings"].fillna(df_rp["Ratings"].mode()[0])
```

Checking the null values again

```
# Checking null values in the dataset again
df_rp.isnull().sum()
```

```
Review_Title    0
Review_Text     0
Ratings         0
dtype: int64
```

Now there is no null values in the datasets.

Data Pre-Processing:

I made some changes to our target column to make it easier for model building.

```
# Checking the unique value count of target column
df_rp['Ratings'].unique()
```

```
array(['4', '5', '3', '1', '2', '2.0 out of 5 stars',
       '3.0 out of 5 stars', '1.0 out of 5 stars', '5.0 out of 5 stars',
       '4.0 out of 5 stars'], dtype=object)
```

Looking the above entries in target column, we need to replace the string entries to there respective values. And lastly, change the column type to integer.

```
# Replacing the string entries in target column
df_rp['Ratings'] = df_rp['Ratings'].replace('1.0 out of 5 stars',1)
df_rp['Ratings'] = df_rp['Ratings'].replace('2.0 out of 5 stars',2)
df_rp['Ratings'] = df_rp['Ratings'].replace('3.0 out of 5 stars',3)
df_rp['Ratings'] = df_rp['Ratings'].replace('4.0 out of 5 stars',4)
df_rp['Ratings'] = df_rp['Ratings'].replace('5.0 out of 5 stars',5)
```

```
# Changing the column data type
df_rp['Ratings'] = df_rp['Ratings'].astype('int')
```

```
# Checking the unique value count of target column again
df_rp['Ratings'].unique()
```

```
array([4, 5, 3, 1, 2])
```


I have changed all the ratings from Ratings column into numerical forms and converted its type to an integer.

I have joined both columns Review_title and Review_text into a new column as Review and dropped the columns Review_title and Review_text from the dataset.

```
# Combining Review text and title
df_rp['Review'] = df_rp['Review_Title'].map(str)+' '+df_rp['Review_Text']
```

Since we have obtained Review from Review_Title and Review_Text let's drop Review_Title & Review_Text.

```
# Dropping unnecessary columns
df_rp.drop(columns = 'Review_Title',inplace = True)
df_rp.drop(columns = 'Review_Text',inplace = True)
```

By observing the Reviews we can say that there are many words, numbers, as well as punctuations which are not important for our predictions. So we need to do some text processing.

Text Processing:

The term text processing refers to the automation of analyzing electronic text. This allows machine learning models to get structured information about the text to use for analysis, manipulation of the text, or to generate new text.

```
# Here I am defining a function to replace some of the contracted words to their full form and removing urls
# and unwanted text
def decontracted(text):
    text = re.sub(r"won't", "will not", text)
    text = re.sub(r"don’t", "do not", text)
    text = re.sub(r"can't", "can not", text)
    text = re.sub(r"im ", "i am", text)
    text = re.sub(r"yo ", "you ",text)
    text = re.sub(r"doesn’t", "does not",text)
    text = re.sub(r"n\'t", " not", text)
    text = re.sub(r"\'re", " are", text)
    text = re.sub(r"\'s", " is", text)
    text = re.sub(r"\'d", " would", text)
    text = re.sub(r"\'ll", " will", text)
    text = re.sub(r"\'t", " not", text)
    text = re.sub(r"\'ve", " have", text)
    text = re.sub(r"\'m", " am", text)
    text = re.sub(r"<br>", " ", text)
    text = re.sub(r'http\S+', '', text) #removing urls
    return text
```

```
# Changing all words to there Lowercase
df_rp['Review'] = df_rp['Review'].apply(lambda x : x.lower())

df_rp['Review'] = df_rp['Review'].apply(lambda x : decontracted(x))
```

```
# Removing punctuations
df_rp['Review'] = df_rp['Review'].str.replace('[^\w\s]','')
df_rp['Review'] = df_rp['Review'].str.replace('\n',' ')
```

For text processing I have defined a function to replace some of the words with proper words. All text is converted to lowercase and removed different punctuations from the text of Review column.

Removing Stop Words:

Now i have removed all stop words from the text data.

```
# Removing stop words
stop = stopwords.words('english')
df_rp['Review'] = df_rp['Review'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop)]))
```

```
# Checking the text data again
df_rp['Review'][0]
```

```
'really nice actually felt like infinix brand much popular guys believe satisfies 100 get free trail 1 month ms office con gett
ing heat instantly feel hot using laptop apart wonderful laptop'
```

Lemmatization:

Lemmatization is the process of grouping together the different inflected forms of a word so they can be analysed as a single item. Lemmatization is similar to stemming but it brings context to the words.

```
# Initialising lemmatizer
lemmatizer = nltk.stem.WordNetLemmatizer()
```

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

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

```
import nltk
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
df_rp['Review'] = df_rp['Review'].apply(lambda x : lemmatize_sentence(x))
```

For lemmatizing the text I have defined these two functions first will give the wordnet tag for the nltk_tagged word then with respect to this wordnet tag lemmatization of each word is done.

Text Normalization – Standardization

```
# Noise removal
def scrub_words(text):
    # Remove html markup
    text = re.sub("(<.*?>)", "", text)
    # Remove non-ascii and digits
    text = re.sub("(\\W)", " ", text)
    text = re.sub("(\\d)", "", text)
    # Remove white space
    text = text.strip()
    return text
```

I have defined a function scrub_words for removing the noise from the text. It will remove any html markups, digits and white spaces from the text.

```
df_rp['Review'] = df_rp['Review'].apply(lambda x : scrub_words(x))
```

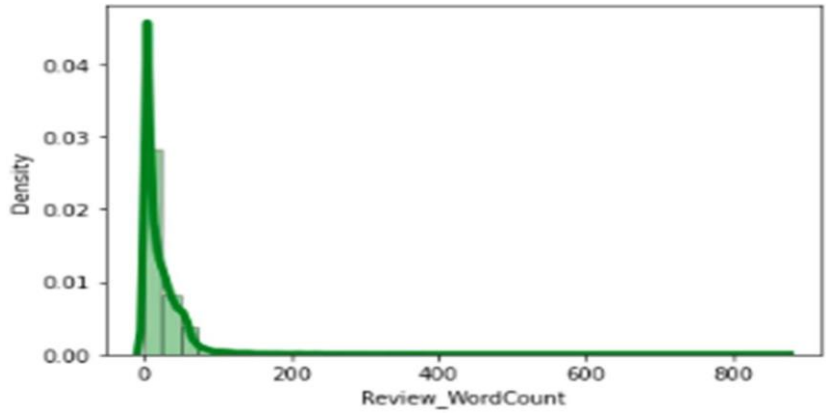
Finally for standardizing our test and removing numbers from it I have defined a function as scrub words as shown in above code and applied to the review column.

Data Visualization

word_count of review

```
# Creating column for word counts in the text
df_rp['Review_WordCount'] = df_rp['Review'].apply(lambda x: len(str(x).split(' ')))
df_rp[['Review_WordCount','Review']].head()
```

	Review_WordCount	Review
0	31	really nice actually felt like infinix brand m...
1	28	wonderful laptop nice sometimes heat startedpe...
2	32	great product brilliant design sleek beautiful...
3	6	simply awesome everything excellent price point
4	8	awesome mind blow purchase ever nice performan...

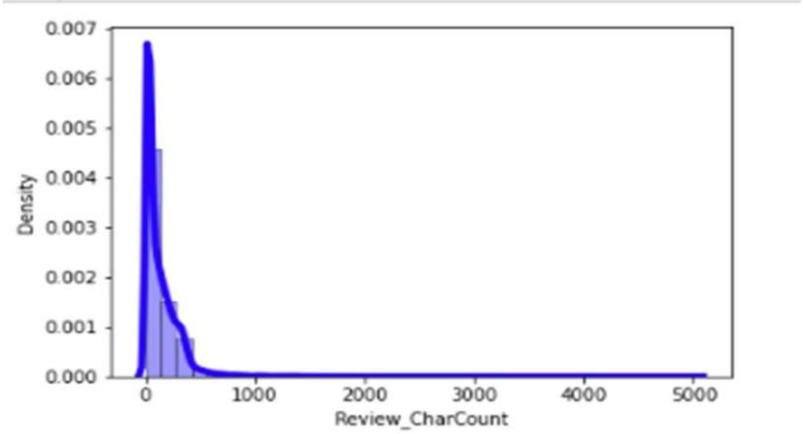


Above figure shows the number of words from each review text. Looking at this histogram we can conclude that most of the review text is in the range of 0 to 200 of words. Rest reviews can be considered as outliers in our data.

Character count of review

```
# Creating column for character counts in the text
df_rp['Review_CharCount'] = df_rp['Review'].str.len()
df_rp[['Review_CharCount','Review']].head()
```

	Review_CharCount	Review
0	175	really nice actually felt like infinix brand m...
1	202	wonderful laptop nice sometimes heat startedpe...
2	194	great product brilliant design sleek beautiful...
3	47	simply awesome everything excellent price point
4	56	awesome mind blow purchase ever nice performan...



The plot for character count is almost similar to the plot of word count. We can see that most of the reviews are in the range of 0 to 1500 numbers of characters. Looking at these plots I have decided to remove the data with too long reviews by considering them as outliers.

Removing Outliers

Some of the reviews are too lengthy, so we have to treat them as outliers and remove them using z_score method.

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

z_score = zscore(df_rp[['Review_WordCount']])
abs_z_score = np.abs(z_score)
filtering_entry = (abs_z_score < 3).all(axis = 1)
df_rp = df_rp[filtering_entry]
df_rp.shape
```

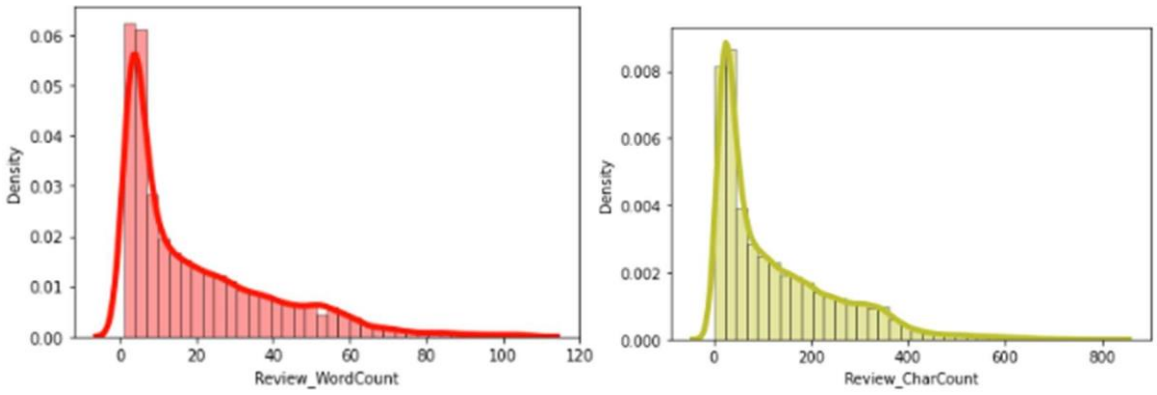
```
(46578, 4)
```

```
Data_loss=((21661-21313)/21661)*100
Data_loss
```

```
1.6065740270532294
```

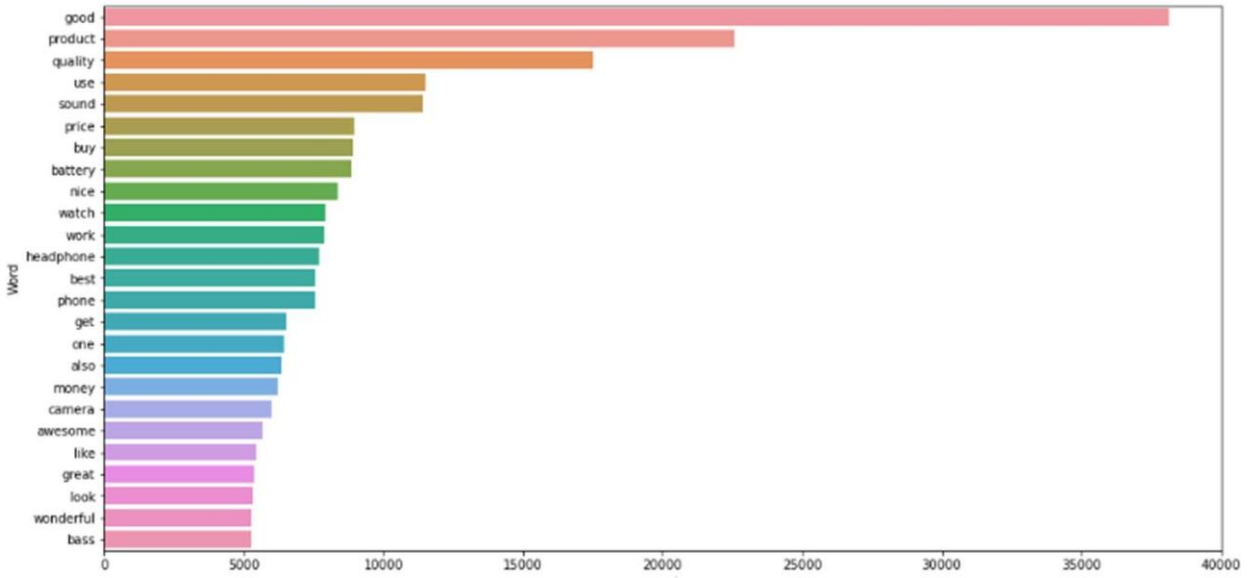
And by removing these outliers I am not losing much of the data so it is good to remove those entries for better results.

Plotting histograms for word counts and character counts again after removing outliers

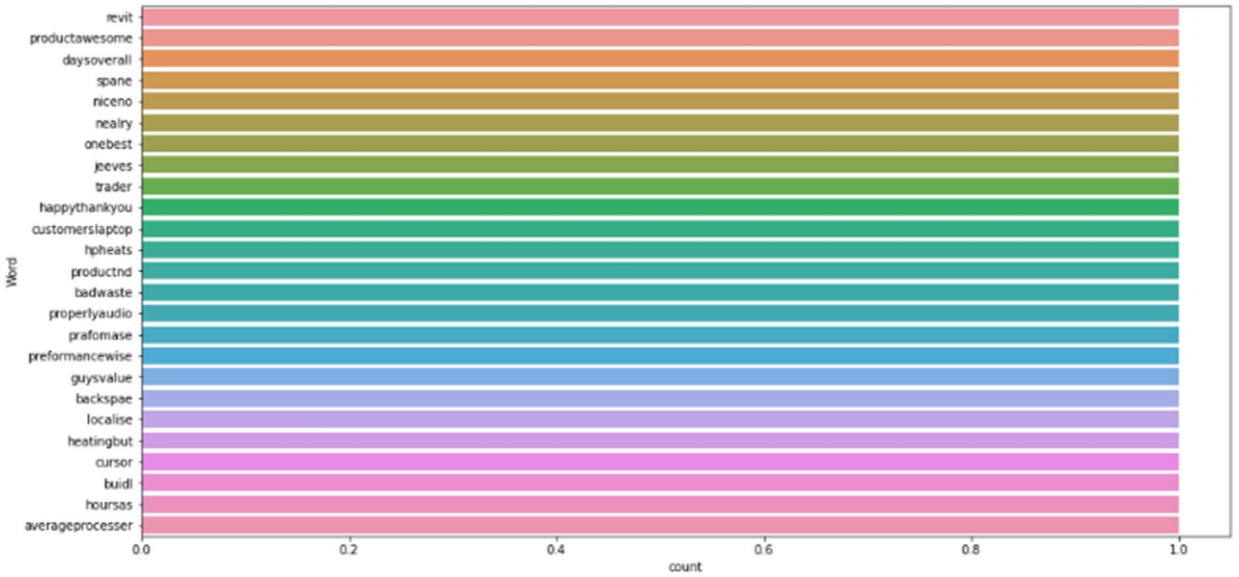


After plotting histograms for word counts and character counts after removing outliers we can see now we are with good range of number of words and characters.

Top 25 most frequently occurring words:




```
1 rare_words(df_rp['Review'])
```



- ☐ By seeing the above plot we can see that Good, product, quality..... are occurring frequently. And the second plot shows rarely occuring words.

Word Cloud for particular ratings:

Rating 1:

Rating 2:

Rating 3:

Rating 4:

Rating 5:

- From the above plots we can clearly see the words which are indication of Reviewer's opinion on products.
- Here most frequent words used for each Rating is displayed in the word cloud.

Oversampling

```
# Separating feature and target
x = df_rp['Review']
y = df_rp['Ratings']
```

```
# Checking the value counts of Ratings column
y.value_counts()
```

```
5    24661
4     8559
1     5740
3     4310
2     3308
Name: Ratings, dtype: int64
```

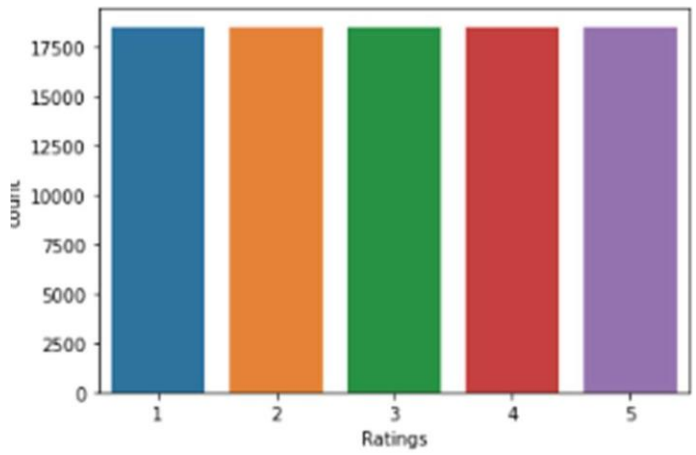
Since there is imbalance in the data of target variable (Ratings), we will apply SMOTE analysis and do oversampling in order to make data balanced .

```
# Oversample and plot imbalanced dataset with SMOTE
from sklearn.datasets import make_classification
from imblearn.over_sampling import SMOTE

# Transforming the dataset
sm = SMOTE(sampling_strategy = {1: 18485, 2: 18485, 3: 18485, 4: 18485, 5: 18485})
x_train1,y_train1 = sm.fit_resample(x_train,y_train)

print("The number of classes after fit {}".format(Counter(y_train1)))
```

```
The number of classes after fit Counter({1: 18485, 5: 18485, 2: 18485, 4: 18485, 3: 18485})
```



Model Building:

Converting text data into vectors using Tfidf Vectorizer

```
# Using the n_gram tfidf vectorizer(Word vectors)
from sklearn.feature_extraction.text import TfidfVectorizer
word_vectorizer = TfidfVectorizer(
                                  sublinear_tf = True,
                                  strip_accents = 'unicode',
                                  analyzer = 'word',
                                  token_pattern = r'\w{1,}',
                                  stop_words = 'english',
                                  ngram_range = (1,3),
                                  max_features = 17000)
word_vectorizer.fit(x)
train_word_features = word_vectorizer.transform(x)
```

```
# Character vectorizer
char_vectorizer = TfidfVectorizer(
                                  sublinear_tf = True,
                                  strip_accents = 'unicode',
                                  analyzer = 'char',
                                  stop_words = 'english',
                                  ngram_range = (2,6),
                                  max_features = 8000)
char_vectorizer.fit(x)
train_char_features = char_vectorizer.transform(x)
```

I have used 6 classification algorithms. First, I have created 3 different classification algorithms and are appended in the variable models. Followed by TFIDF vectorization and data balancing. Then, ran a for loop which contained the accuracy of the models along with different evaluation metrics

Run and Evaluate different models:

1). Logistic Regression:

```
1 lg=LogisticRegression()
2 lg.fit(x_train1, y_train1)
3 lg.score(x_train1, y_train1)
4 pred_lg=lg.predict(x_test)
5
6 print("accuracy score: ",accuracy_score(y_test,pred_lg))
7 print(confusion_matrix(y_test,pred_lg))
8 print(classification_report(y_test,pred_lg))
```

```
accuracy score:  0.8506655216831258
[[1163  142   71   12   12]
 [ 138  538  117   29   14]
 [  45  153  732  137   30]
 [  17   69  156 1755  139]
 [  23   34  102  299 5718]]
              precision    recall  f1-score   support

           1       0.84      0.83      0.83      1400
           2       0.57      0.64      0.61       836
           3       0.62      0.67      0.64      1097
           4       0.79      0.82      0.80      2136
           5       0.97      0.93      0.95      6176

    accuracy                           0.85     11645
   macro avg       0.76      0.78      0.77     11645
weighted avg       0.86      0.85      0.85     11645
```

2. Decision Tree Classifier:

```
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:  0.8087591240875912
[[1060  143   94   46   57]
 [ 154  461   88   56   77]
 [  88  134  647  117  111]
 [  63  102  124 1648  199]
 [  73   80  164  257 5602]]
              precision    recall  f1-score   support

           1       0.74      0.76      0.75      1400
           2       0.50      0.55      0.53       836
           3       0.58      0.59      0.58      1097
           4       0.78      0.77      0.77      2136
           5       0.93      0.91      0.92      6176

    accuracy                           0.81     11645
   macro avg       0.70      0.72      0.71     11645
weighted avg       0.81      0.81      0.81     11645
```

3.Random Forest Classifier:

```
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:  0.8519536281665951
[[1196  131   46   11   16]
 [ 181  516   77   28   34]
 [  75  168  683  111   60]
 [  28   94  136 1732  146]
 [  33   53   81  215 5794]]
              precision    recall  f1-score   support

           1       0.79      0.85      0.82      1400
           2       0.54      0.62      0.57       836
           3       0.67      0.62      0.64      1097
           4       0.83      0.81      0.82      2136
           5       0.96      0.94      0.95      6176

    accuracy                           0.85     11645
   macro avg       0.76      0.77      0.76     11645
weighted avg       0.86      0.85      0.85     11645
```

Cross Validation Score:

```
from sklearn.model_selection import cross_val_score
```

```
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:  0.787083525489136
```

```
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:  0.704791613226035
```

```
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:  0.7745029635207046
```

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

Next the Random Forest Classifier model.

Lastly, the Decision Tree Classifier model.

Hyper Parameter Tuning:

Since the Accuracy Score and the Crosss Validation Score of the Logistic Regression Model have the least different between them, we shalll consider it for hyper parameter tuning.

We shall use GridSearchCV for hyper parameter tuning.

Since the Accuracy Score and the Crosss Validation Score of the Logistic Regression Model have the least different between them, we shalll consider it for hyper parameter tuning.

We shall use GridSearchCV for hyper parameter tuning.

```
from sklearn.model_selection import GridSearchCV
```

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

```
grid_lg.fit(x_train1, y_train1)
```

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

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
grid_lg.best_params_
```

```
{'C': 1.0, 'penalty': 'l2', 'solver': 'newton-cg'}
```

```
z r 1na1_uode1= Log1st1cRegress1on(c=e.1,pena1ty= ' 12 ' , so1ver• ' newton-eg' )

' r1na1_uode1.f1t (x_tra1n1,y_tra1n1)
- pred • F ina1_Hodel . predict(x_test)
s pr1nt ("accuracy score: ", accuracy_score(y_test,pred))
o pr "Int (confus1on_• atr 1x(y_test, pred))
° pr "Int (c1ass1+1cat1on_re port(y_test,pred))
```

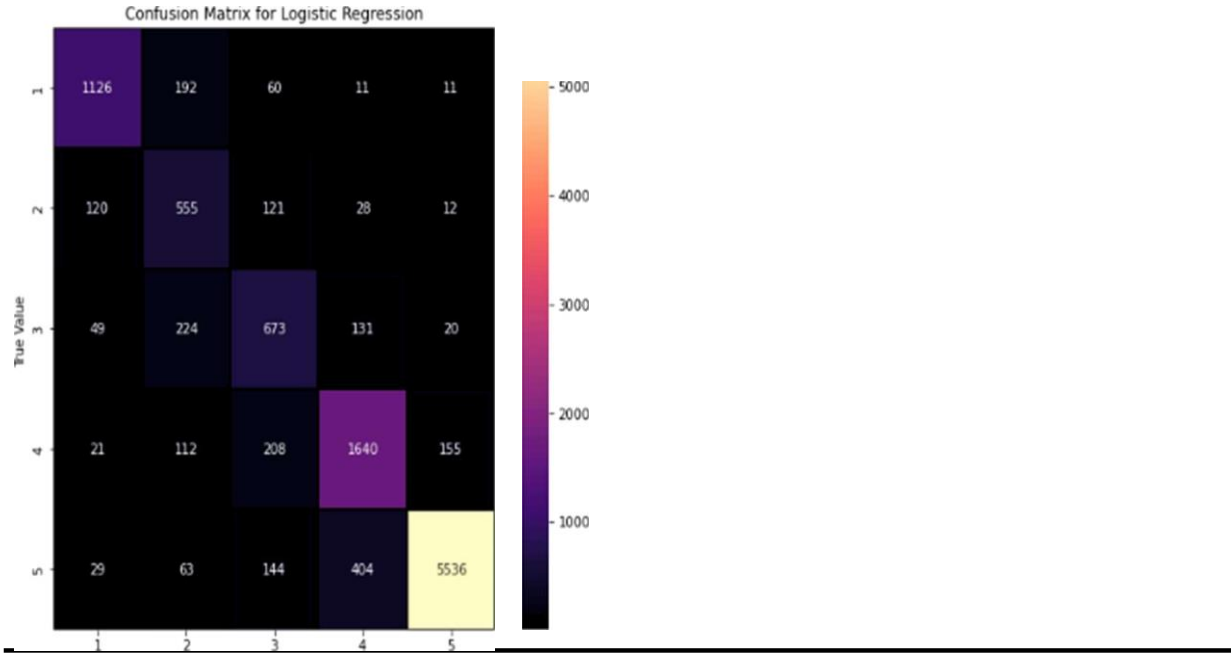
```
accuracy score: e.# 8376sgs#3e8z#
f(1t26  ts2   se   tt    tj
( z2e   sss  121   28    12]
[  49   224  673  13t    2e]
[  21   112   2 8 1646 155]
(  29    63 144 484 SS36] l
```

	precision	recall	f1-score	support
	e.so	e.se	e.82	14ee
2	e.w	e.ss	e.ss	s3s
a	e.ss	e.st	e.ss	ies7
<	e.ya	e.yy	e.ys	2136
s	e.s7	6.se	e.93	6176
accuracy			8.82	11645
aacro avg	e.72	a.ys	e.73	11645
we1ghted avg	e.as	8.82	8.83	11645

```
cm = confusion_matrix(y_test, pred)

x_axis_labels = ["1","2","3","4","5"]
y_axis_labels = ["1","2","3","4","5"]

f, ax = plt.subplots(figsize =(8,8))
sns.heatmap(cm, annot = True, linewidths=0.2, linecolor="black", fmt = ".0f", ax=ax, cmap="magma",xticklabels = x_axis_label
plt.xlabel("Predicted Value")
plt.ylabel("True Value ")
plt.title('Confusion Matrix for Logistic Regression')
plt.show()
```



Saving the Model:

We are saving the model by using python's pickle library. It will be used further for the predictio n.

```
import joblib
joblib.dump(Final_Model,"Ratings_Prediction_Project.pkl")
```

```
['Ratings_Prediction_Project.pkl']
```

The logistic Regression Model is our best model with accuracy of 83.16 %.

Prediction:

```
# Loading the saved model
ratings_prediction_model=joblib.load("Ratings_Prediction_Project.pkl")

# Prediction
prediction = ratings_prediction_model.predict(x_test)
prediction
```

```
array([5, 5, 1, ..., 2, 5, 5])
```

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

	0	1	2	3	4	5	6	7	8	9	...	11635	11636	11637	11638	11639	11640	11641	11642	11643	11644
Predicted Rating	5	5	1	5	2	5	2	5	5	3	...	5	5	4	4	4	1	2	2	5	5
Actual Rating	5	5	1	5	5	5	2	5	5	3	...	5	5	4	4	4	1	4	2	5	5

2 rows × 11645 columns

The actual and predicted values are almost similar.

Key Findings and Conclusions

- In this project I 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 reviews column and chose equal number of text from each rating class to eliminate problem of imbalance. By doing different EDA steps I have analysed the text.
- We have checked frequently occurring words in our data as well as rarely occurring words.
- After all these steps I have built function to train and test different algorithms and using various evaluation metrics I have selected Logistic Regresson model for our final model.
- Finally by doing hyperparameter tuning we got optimum parameters for our final model.

Learning Outcomes of the Study in respect of Data Science

I have scrapped the reviews and ratings of different technical products from flipkart.com and amazon.in websites. Improvement in computing technology has made it possible to examine social information that cannot previously be captured, processed and analysed. New analytical techniques(NLP) of machine learning can be used in property research. The power of visualization has helped us in understanding the data by graphical representation it has made me to understand what data is trying to say. Data cleaning is one of the most important steps to remove unrealistic values, punctuations, urls, email address and stop words.

This study is an exploratory attempt to use 3 machine learning algorithms in estimating Rating, and then compare their results.

To conclude, the application of NLP in Rating classification is still at an early stage. We hope this study has moved a small step ahead in providing some methodological and empirical contributions to crediting institutes, and presenting an alternative approach to the valuation of Ratings.

Limitations of this work and Scope for Future Work

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

Thank you