FLIP ROBO

NAME OF THE PROJECT

Rating Prediction Project

Submitted by:

Shilpi Mohanty

ACKNOWLEDGMENT

I am highly indebted to Flip Robo Technologies Bangalore for their guidance and constant supervision as well as for providing necessary information regarding the Project and also for their support in completing the project.

It is my radiant sentiment to place on record my best regards, deepest sense of gratitude to Mr. Mohd. Kashif, SME for giving the dataset and clear instructions to perform the complete case study process and guided and advised me from time to time.

I perceive as this opportunity as a learning experience in my career development. I will strive to use gained skills and knowledge in the best possible way, and try to work on the improvement, in order to attain desired career objectives.

INTRODUCTION

- ## Business Problem Framing

The rapid development of Web 2.0 and e-commerce has led to a proliferation in the number of online user reviews. Online reviews contain a wealth of sentiment information that is important for many decision-making processes, such as personal consumption decisions, commodity quality monitoring, and social opinion mining. Mining the sentiment and opinions that are contained in online reviews has become an important topic in natural language processing, machine learning, and Web mining.

Customer reviews and eCommerce website profits go hand in hand. Good reviews are a sign of customer loyalty, and businesses that have a lot of positive reviews tend to grow their revenues twice as fast as their competitors. By understanding exactly how your business will benefit from customer reviews, you can work on a strategy that will help you to get more of them.

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 (ratings) 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 which can predict the rating by seeing the review.

- ## Conceptual Background of the Domain Problem

Sentiment is an attitude, thought, or judgment prompted by feeling. Sentiment analysis, which is also known as opinion mining, studies people's sentiments towards certain entities. From a user's perspective, people are able to post their own content through various social media, such as forums, micro-blogs, or online social networking sites. From a researcher's perspective, many social media sites release their application programming interfaces (APIs), prompting data collection and analysis by researchers and developers. However, those types of online data have several flaws that potentially hinder the process of sentiment analysis. The first flaw is that since people can freely post their own content, the quality of their opinions cannot be guaranteed. he second flaw is that ground truth of such online data is not always available. A ground truth is more like a tag of a certain opinion, indicating whether the opinion is positive, negative, or neutral.

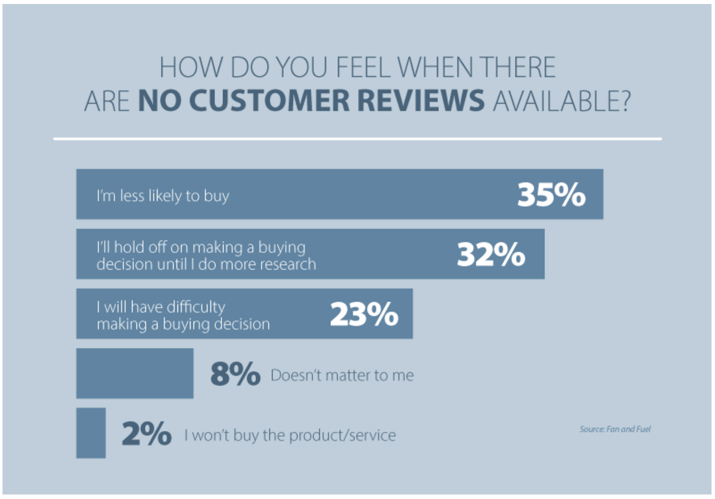
Rating prediction is a well-known recommendation task aiming to predict a user's rating for those items which were not rated yet by customers. Predictions are computed from users' explicit feedback i.e., their ratings provided on some items in the past. Another type of feedback are user reviews provided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

• Review of Literature

When you think about buying something online, how do you decide whether to follow through with your purchase? If you're like most shoppers, you might turn to on online reviews for guidance.

According to research from Nielsen, 73% of online respondents use reviews to make purchase decisions. "Perhaps nothing is more important than word-of-mouth testimonials from satisfied customers, whether in person or online via reviews and social media," added Patrick Dodd, chief commercial offer at Nielsen.

If online reviews aren't available, 92% of customers are impacted, according to a survey from marketing agency Fan & Fuel. That group expressed a lot of hesitation in what would happen next; Thirty-five percent said they were less likely to buy, 32% said they'd wait until they could do more research, 23% said their buying decision would be difficult, and 2% said that they simply wouldn't buy the product/service.



The goal of the project is to build an application which can predict the rating by seeing the review. In the long term, this would allow people to better explain and review their purchase with each other in this increasingly digital world.

• Motivation for the Problem Undertaken

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 (ratings) 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 which can predict the rating by seeing the review.

Many projects review are not accompanied by scale rating system and it is herculean task to compare between different products and make a good choice between them. Therefore, building a model which can predict the user rating from text review have gain importance .This in turn improves the customer buying and long lasting relationship with the brand.

Analytical Problem Framing

• Mathematical/ Analytical Modelling of the Problem

I have scraped reviews and ratings from well-known e-commerce sites like amazon and flipkart and this was saved into CSV format file.

After that I, 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.

In our scrapped dataset our target variable column "Ratings" is a categorical variable i.e., it can be classified as 1, 2, 3, 4 and 5 stars. Therefore, we will be handling this modelling problem as a multi-class classification project.Thus it is a supervised machine learning problem where we can use the techniques of NLP and classification-based algorithms of Machine Learning. Here we will use NLP techniques like word tokenization, lemmatization, stemming and tfidf vectorizer then those processed data will be used to create best model using various classification based supervised ML algorithms like Logistic Regression, Multinomial NB, Decision Tree classifier,SDGClassifier.

• Data Sources and their formats

This project is done in two parts:
1)Data Collection Phase
2)Model Building Phase

Data Collection Phase:
You have to scrape at least 20000 rows of data. You can scrape more data as well, it's up to you. More the data better the model. In this section you need to scrape the reviews of different laptops, Phones, Headphones, smart watches, Professional Cameras, Printers, monitors, home theatre, router from different e-commerce websites.

Basically, we need these columns:
1) reviews of the product.
2) rating of the product.

Fetch an equal number of reviews for each rating, for example if you are fetching 10000 reviews then all ratings 1,2,3,4,5 should be 2000. It will balance our data set. Convert all the ratings to their round number as there are only 5 options for rating i.e., 1,2,3,4,5. If a rating is 4.5 convert it 5.

Model Building Phase:

After collecting the data, you need to build a machine learning model. Before model building do all data pre-processing steps involving NLP. Try different models with different hyper parameters and select the best model. Follow the complete life cycle of data science. Include all the steps mentioned below:

1. Data Cleaning
2. Exploratory Data Analysis and Visualization
3. Data Pre-processing
4. Model Building
5. Model Evaluation
6. Selecting the Best classification model

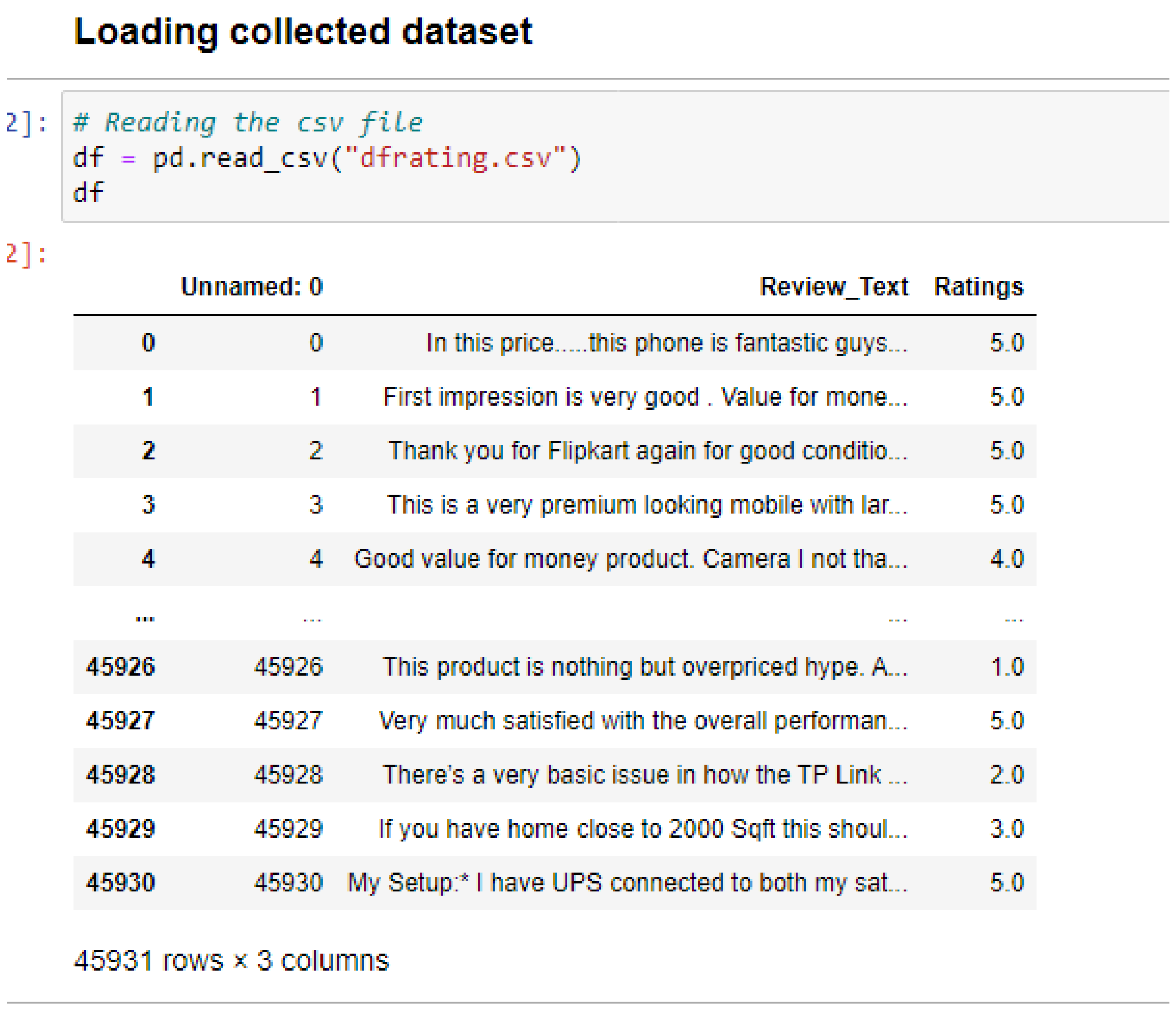
We collected the data from difference e-commerce websites like Amazon and Flipkart. The data is scrapped using Web scraping technique and the framework used is Selenium. I scrapped nearly 45931 records of the reviews data and saved it in a CSV format file. Thus, data consists of one input feature, the string data for the comments, and Ratings as label.

The data set includes:

Columns	**Description**
Review_text :	Content of the review text
Ratings	Ratings out of 5 stars

There is one unnamed column which was irrelevant for model building so it has been deleted.

Let’s check the data now. Below I have attached the snapshot below to give an overview.



Loading collected dataset

```
2]: # Reading the csv file
    df = pd.read_csv("dfrating.csv")
    df
```

2]:

	Unnamed: 0	Review_Text	Ratings
0	0	In this price.....this phone is fantastic guys...	5.0
1	1	First impression is very good . Value for mone...	5.0
2	2	Thank you for Flipkart again for good conditio...	5.0
3	3	This is a very premium looking mobile with lar...	5.0
4	4	Good value for money product. Camera I not tha...	4.0
...	...	...	...
45926	45926	This product is nothing but overpriced hype. A...	1.0
45927	45927	Very much satisfied with the overall performan...	5.0
45928	45928	There's a very basic issue in how the TP Link ...	2.0
45929	45929	If you have home close to 2000 Sqft this shoul...	3.0
45930	45930	My Setup:* I have UPS connected to both my sat...	5.0

45931 rows × 3 columns

Thus, we see there are 45931 rows and 3 columns in the dataset

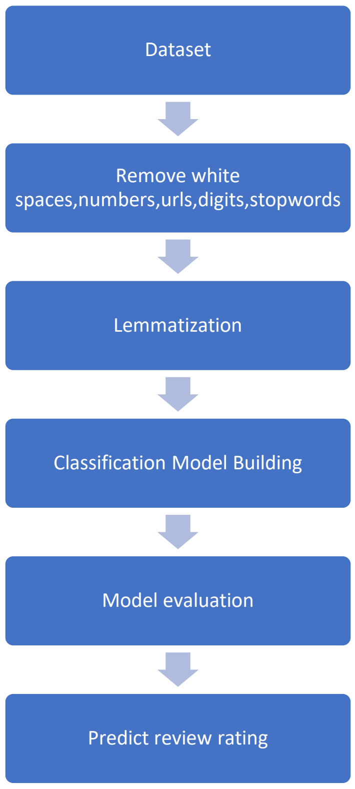
- ## Data Preprocessing Done

Data pre-processing is a step in the data mining and data analysis process that takes raw data and transforms it into a format that can be understood and analysed by computers and machine learning**.**

Raw, real-world data in the form of text, images, video, etc., is messy. It contain errors and inconsistencies, often incomplete, and doesn't have a regular, uniform design.

Machines understands and read data as 1s and 0s. So calculating structured data, like whole numbers and percentages is easy. However, unstructured data, in the form of text and images must first be cleaned filtered and formatted before analysis.

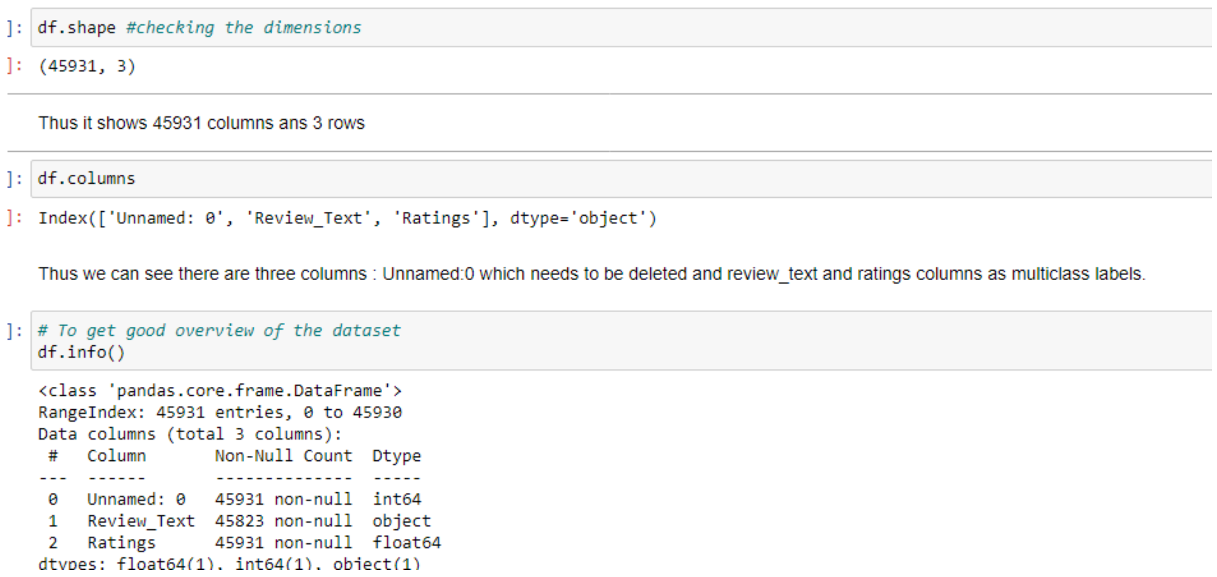
- ## Steps in Data Pre-Processing


Dataset
Remove white spaces,numbers,urls,digits,stopwords
Lemmatization
Classification Model Building
Model evaluation
Predict review rating

In pre-processing we started off with removing of punctuation and special character from the comments and removing column which are not important like column id. We at that point recognize that we had to also remove the stop words which are useless as they do not add any value to the dataset i.e. those comments are meaningless. Further we performed stemming and lemmatizing for the words. Lastly, we applied count vectorizer and afterward split the information into preparing and testing for the further use.

We have undergone various ways of visualizing the data to get meaningful outputs, which would help us in knowing the dataset and our final goals effectively. We tried to segregate the analysis of the dataset into various categories such as based on the word lengths, the toxic words utilized and the degree of toxicity present in them. We have also balanced the class by smote oversampling method.

All this visualization helped us in gaining a wholistic picture which led us to finally sort down to two algorithms which was making our end goal in classifying review rating efficiently.



```
]: df.shape #checking the dimensions
]: (45931, 3)
```

Thus it shows 45931 columns ans 3 rows

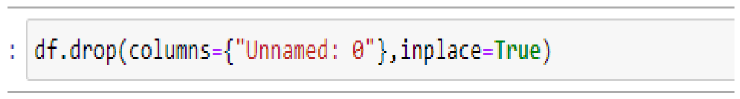
```
]: df.columns
]: Index(['Unnamed: 0', 'Review_Text', 'Ratings'], dtype='object')
```

Thus we can see there are three columns : Unnamed:0 which needs to be deleted and review_text and ratings columns as multiclass labels.

```
]: # To get good overview of the dataset
   df.info()

   <class 'pandas.core.frame.DataFrame'>
   RangeIndex: 45931 entries, 0 to 45930
   Data columns (total 3 columns):
    #   Column       Non-Null Count  Dtype  
   ---  ------       --------------  -----  
    0   Unnamed: 0   45931 non-null  int64  
    1   Review_Text  45823 non-null  object 
    2   Ratings      45931 non-null  float64
   dtypes: float64(1), int64(1), object(1)
```

Observation: Thus we see that there are 3 columns having int,object and float datatype and some missing value are present in review_text .Unammed 0 column was irrelevant so needs to be deleted.

Dropping the column unnamed:0



```
: df.drop(columns={"Unnamed: 0"},inplace=True)
: df
```

	Review_Text	Ratings
0	In this price.....this phone is fantastic guys...	5.0
1	First impression is very good . Value for mone...	5.0
2	Thank you for Flipkart again for good conditio...	5.0
3	This is a very premium looking mobile with lar...	5.0
4	Good value for money product. Camera I not tha...	4.0
...	...	...
45926	This product is nothing but overpriced hype. A...	1.0
45927	Very much satisfied with the overall performan...	5.0
45928	There's a very basic issue in how the TP Link ...	2.0
45929	If you have home close to 2000 Sqft this shoul...	3.0
45930	My Setup:* I have UPS connected to both my sat...	5.0

45931 rows × 2 columns

```
df.shape
```

```
(45931, 2)
```

Thus we see there are 45823 columns and two rows in the dataset.

Null values checking

```
: df.isnull().sum()
```

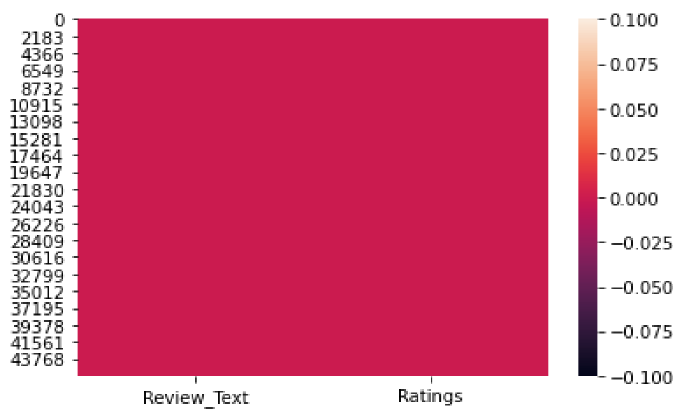
```
: Review_Text    108
  Ratings          0
  dtype: int64
```

Thus we see there are some null values present in the dataset.

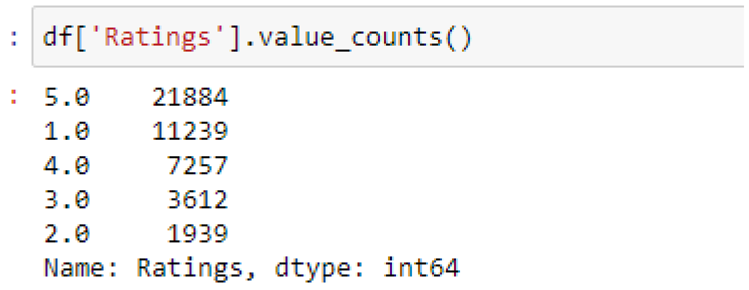
```
: # removing null values
  df.dropna(inplace=True)
```

As there were less null values so decided to drop it.

Null values checking using heatmap:



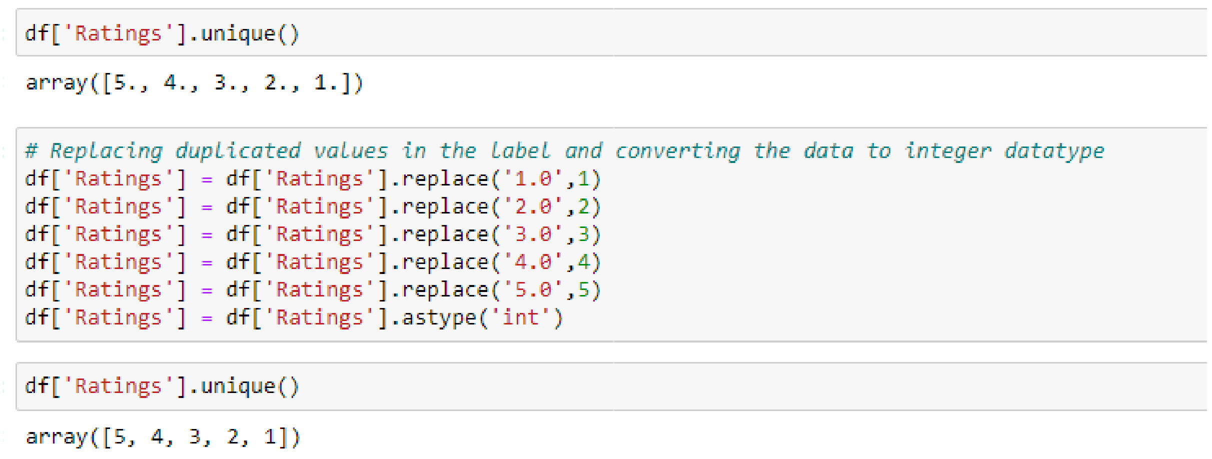
Observation: We can see that there are no null values present in the given after dropping some null values.



```
: df['Ratings'].value_counts()
```

```
: 5.0    21884
  1.0    11239
  4.0     7257
  3.0     3612
  2.0     1939
  Name: Ratings, dtype: int64
```

We see that higher number of ratings given to 5 and 1 and fewer given to 4,3,2. Thus dataset is not balanced.



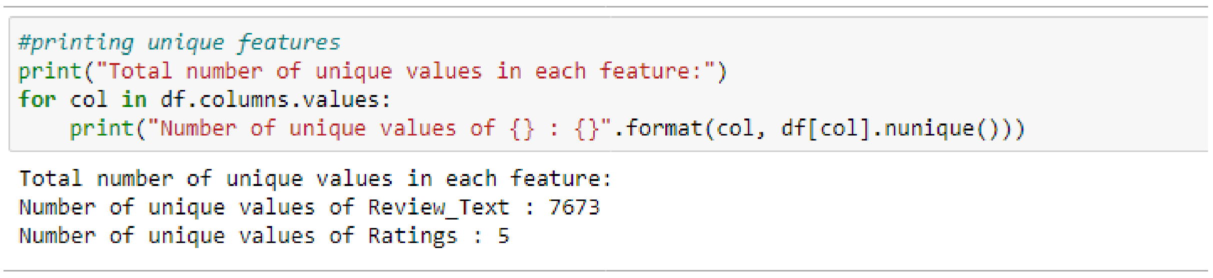
```
df['Ratings'].unique()
```

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

```
# Replacing duplicated values in the label and converting the data to integer datatype
df['Ratings'] = df['Ratings'].replace('1.0',1)
df['Ratings'] = df['Ratings'].replace('2.0',2)
df['Ratings'] = df['Ratings'].replace('3.0',3)
df['Ratings'] = df['Ratings'].replace('4.0',4)
df['Ratings'] = df['Ratings'].replace('5.0',5)
df['Ratings'] = df['Ratings'].astype('int')
```

```
df['Ratings'].unique()
```

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



```
#printing unique features
print("Total number of unique values in each feature:")
for col in df.columns.values:
    print("Number of unique values of {} : {}".format(col, df[col].nunique()))
```

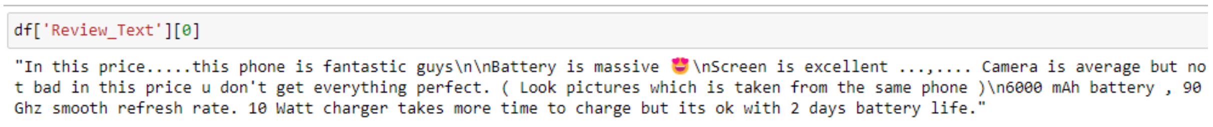
```
Total number of unique values in each feature:
Number of unique values of Review_Text : 7673
Number of unique values of Ratings : 5
```

Thus we there are 5 ratings

Thus we there are 5 ratings and unique review_text is around 7673.

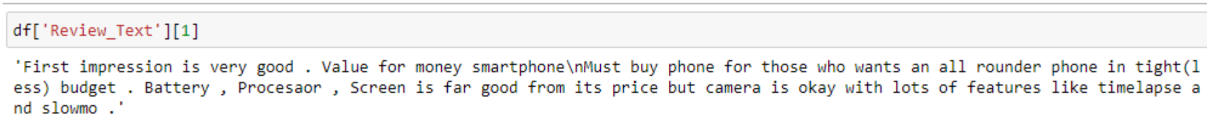
Text Processing

Lets us see the text



```
df['Review_Text'][0]
```

```
"In this price.....this phone is fantastic guys\n\nBattery is massive 😍\nScreen is excellent ...,.... Camera is average but no
t bad in this price u don't get everything perfect. ( Look pictures which is taken from the same phone )\n6000 mAh battery , 90
Ghz smooth refresh rate. 10 Watt charger takes more time to charge but its ok with 2 days battery life."
```



```
df['Review_Text'][1]
```

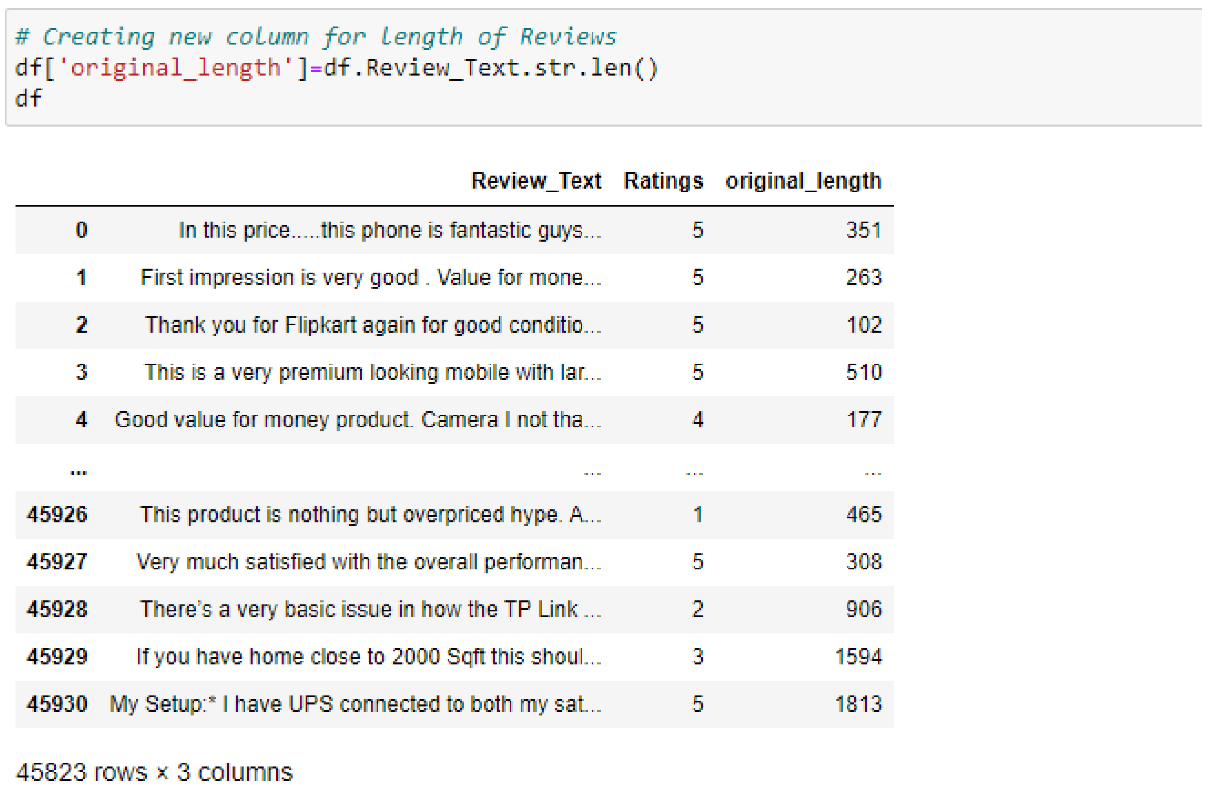
```
'First impression is very good . Value for money smartphone\nMust buy phone for those who wants an all rounder phone in tight(l
ess) budget . Battery , Procesaor , Screen is far good from its price but camera is okay with lots of features like timelapse a
nd slowmo .'
```

Observation: Thus we see that there are many words and numbers which are not important for prediction which need to be removed before model buidling.

Text Preprocessing

Using the text pre-processing techniques, we can remove noise from raw data and makes raw data more valuable for building models. Pre-processing involves the following steps, but these will be performed in a slightly different manner:

1)Text case conversion
2)Removing Punctuations and other special characters
3)Splitting the comments into individual words
4)Removing Stop Words
5)Stemming and Lemmatization
6)Text Standardization-Normalization
7)Splitting dataset into Training and Testing



```
# Creating new column for length of Reviews
df['original_length']=df.Review_Text.str.len()
df
```

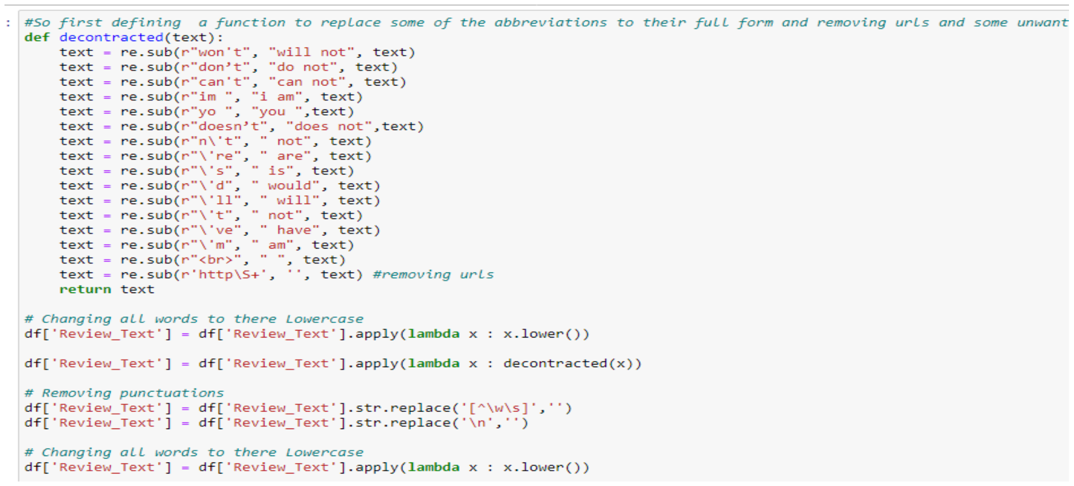
	Review_Text	Ratings	original_length
0	In this price.....this phone is fantastic guys...	5	351
1	First impression is very good . Value for mone...	5	263
2	Thank you for Flipkart again for good conditio...	5	102
3	This is a very premium looking mobile with lar...	5	510
4	Good value for money product. Camera I not tha...	4	177
...	...	...	...
45926	This product is nothing but overpriced hype. A...	1	465
45927	Very much satisfied with the overall performan...	5	308
45928	There's a very basic issue in how the TP Link ...	2	906
45929	If you have home close to 2000 Sqft this shoul...	3	1594
45930	My Setup:* I have UPS connected to both my sat...	5	1813

45823 rows × 3 columns

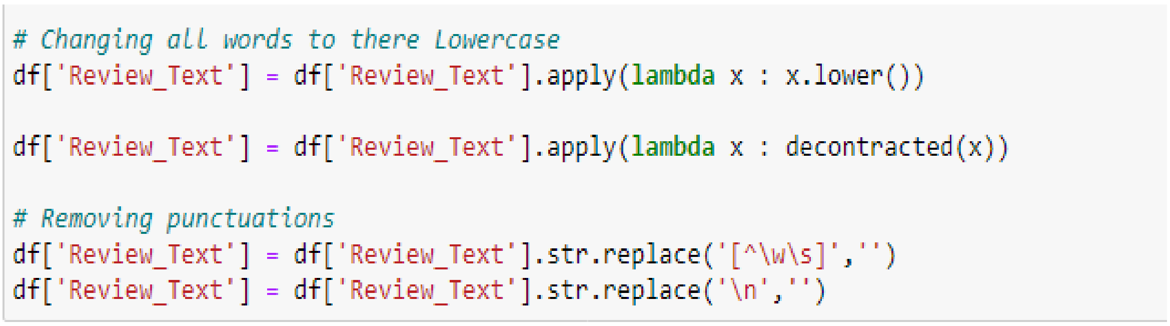
Observation: first we noted down the length of the original text before text processing.

Feature Engineering:

Handling unwanted characters/Conversion to lowercase/Removing punctuation:

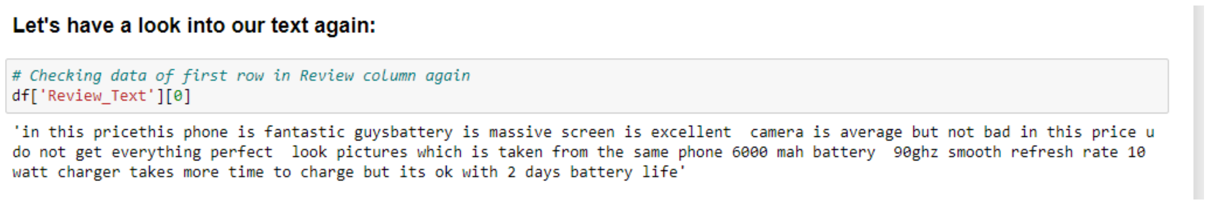


```
#So first defining  a function to replace some of the abbreviations to their full form and removing urls and some unwant
def decontracted(text):
    text = re.sub(r"won't", "will not", text)
    text = re.sub(r"don’t", "do not", text)
    text = re.sub(r"can't", "can not", text)
    text = re.sub(r"im ", "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['Review_Text'] = df['Review_Text'].apply(lambda x : x.lower())

df['Review_Text'] = df['Review_Text'].apply(lambda x : decontracted(x))

# Removing punctuations
df['Review_Text'] = df['Review_Text'].str.replace('[^\w\s]','')
df['Review_Text'] = df['Review_Text'].str.replace('\n','')

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



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

df['Review_Text'] = df['Review_Text'].apply(lambda x : decontracted(x))

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

Observation : Thus we have removed characters like ‘\n’,converted to lowercase,digits,punctuation.



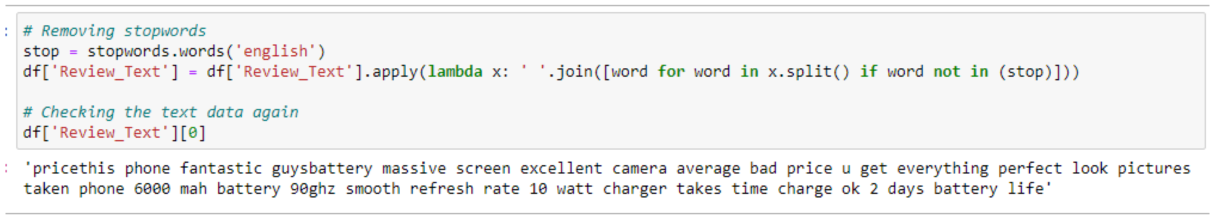
Let's have a look into our text again:

```
# Checking data of first row in Review column again
df['Review_Text'][0]
```

```
'in this pricethis phone is fantastic guysbattery is massive screen is excellent  camera is average but not bad in this price u
do not get everything perfect  look pictures which is taken from the same phone 6000 mah battery  90ghz smooth refresh rate 10
watt charger takes more time to charge but its ok with 2 days battery life'
```

Removing Stopwords

Stop words are a set of commonly used words in a language. Examples of stop words in English are “a”, “the”, “is”, “are” and etc. Stop words are commonly used in Text Mining and Natural Language Processing (NLP) to eliminate words that are so commonly used that they carry very little useful information.



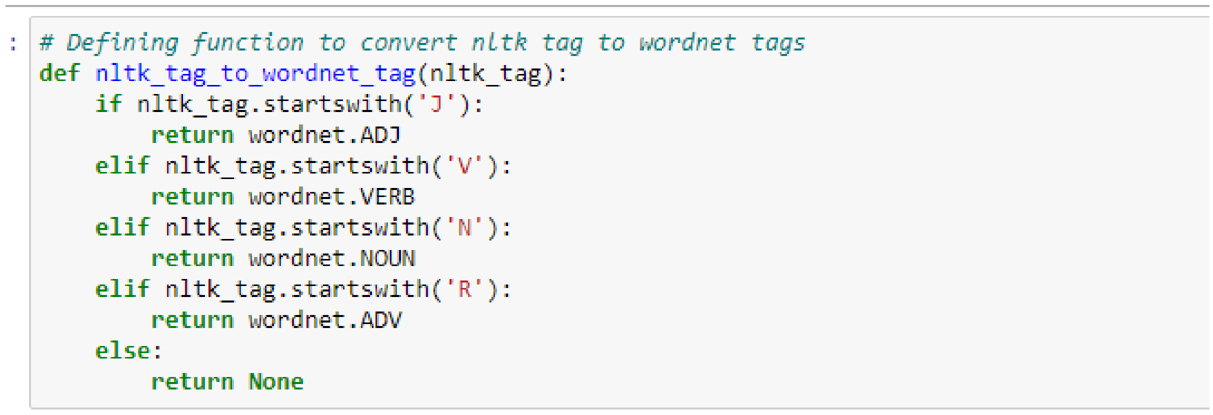
```
# Removing stopwords
stop = stopwords.words('english')
df['Review_Text'] = df['Review_Text'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop)]))

# Checking the text data again
df['Review_Text'][0]
```

```
'pricethis phone fantastic guysbattery massive screen excellent camera average bad price u get everything perfect look pictures
taken phone 6000 mah battery 90ghz smooth refresh rate 10 watt charger takes time charge ok 2 days battery life'
```

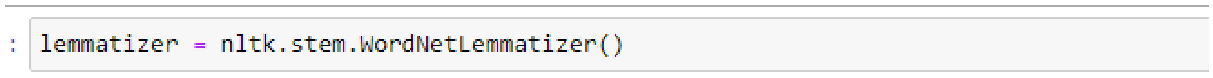
Lemmatization and stemming with Snowball

Stemming is the process of converting inflected/derived words to their word stem or the root form. Basically, a large number of similar origin words are converted to the same word.E.g. words like "stems", "stemmer", "stemming", "stemmed" as based on "stem". This helps in achieving the training process with a better accuracy.

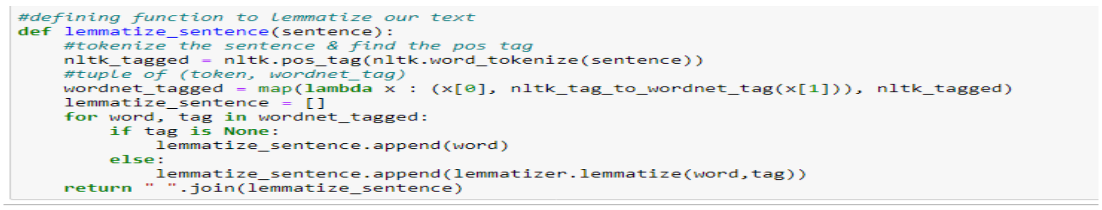
Lemmatizing is the process of grouping together the inflected forms of a word so they can be analysed as a single item. This is quite similar to stemming in its working but differs since it depends on correctly identifying the intended part of speech and meaning of a word in a sentence, as well as within the larger context surrounding that sentence, such as neighboring sentences or even an entire document.



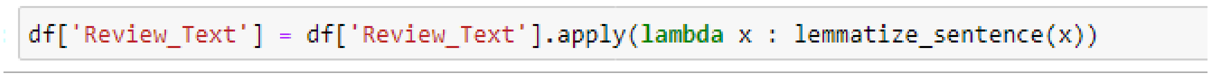
```
# Defining function to convert nltk tag to wordnet tags
def nltk_tag_to_wordnet_tag(nltk_tag):
    if nltk_tag.startswith('J'):
        return wordnet.ADJ
    elif nltk_tag.startswith('V'):
        return wordnet.VERB
    elif nltk_tag.startswith('N'):
        return wordnet.NOUN
    elif nltk_tag.startswith('R'):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



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



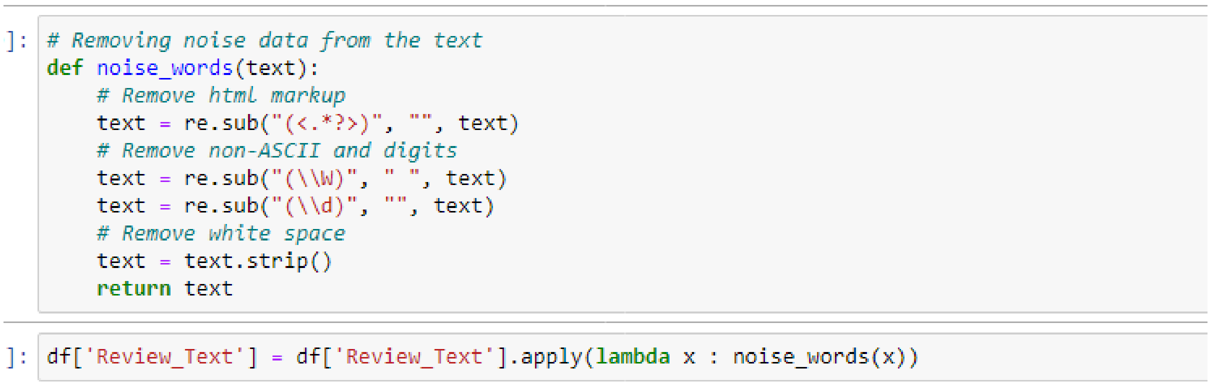
```
#defining function to lemmatize our text
def lemmatize_sentence(sentence):
    #tokenize the sentence & find the pos tag
    nltk_tagged = nltk.pos_tag(nltk.word_tokenize(sentence))
    #tuple of (token, wordnet_tag)
    wordnet_tagged = map(lambda x : (x[0], nltk_tag_to_wordnet_tag(x[1])), nltk_tagged)
    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)
```



```
df['Review_Text'] = df['Review_Text'].apply(lambda x : lemmatize_sentence(x))
```

Text Normalization-Standardization

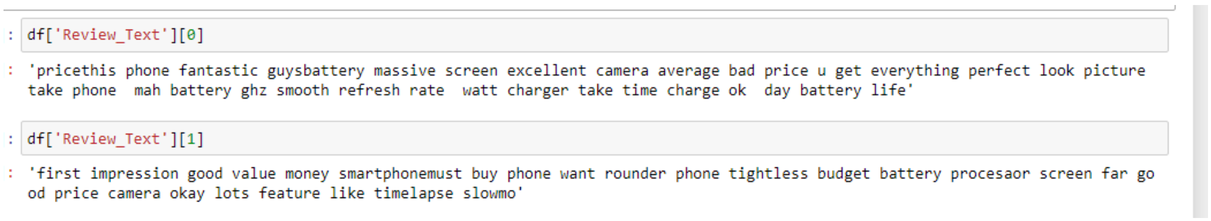
Let's remove all the noise data which is present in the text



```
]: # Removing noise data from the text
   def noise_words(text):
       # Remove html markup
       text = re.sub("(<.*?>)", "", text)
       # Remove non-ASCII and digits
       text = re.sub("(\\W)", " ", text)
       text = re.sub("(\\d)", "", text)
       # Remove white space
       text = text.strip()
       return text
```

```
]: df['Review_Text'] = df['Review_Text'].apply(lambda x : noise_words(x))
```

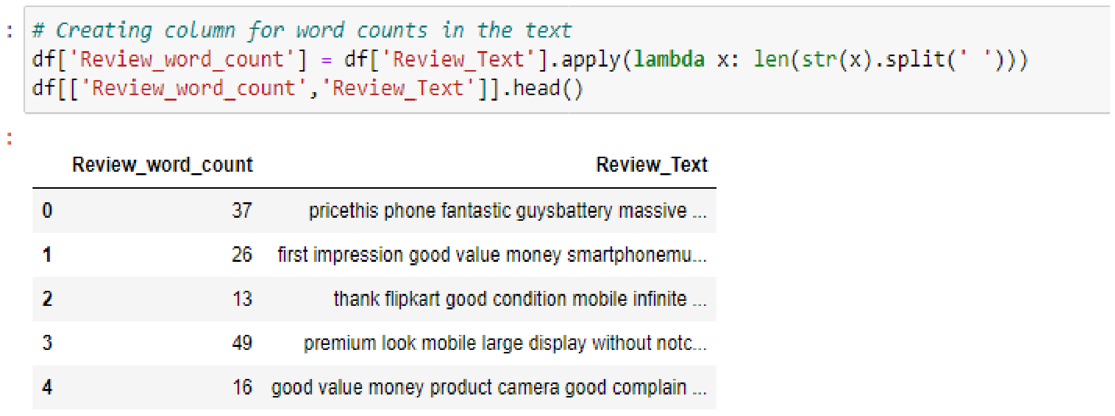
Observation: By defining function noise_words ,it will remove the noise from the text. It will remove any html markups, digits,urls and white spaces from the text. Let's have a look at first two rows of the text .



```
: df['Review_Text'][0]

: 'pricethis phone fantastic guysbattery massive screen excellent camera average bad price u get everything perfect look picture
  take phone  mah battery ghz smooth refresh rate  watt charger take time charge ok  day battery life'

: df['Review_Text'][1]

: 'first impression good value money smartphonemust buy phone want rounder phone tightless budget battery procesaor screen far go
  od price camera okay lots feature like timelapse slowmo'
```

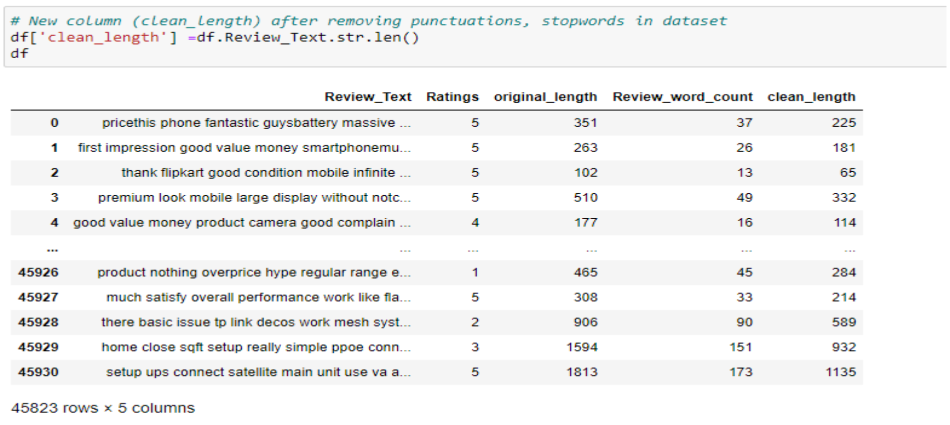
Observation: As we can observe all the text has been converted to lower case and removed irrelevant words, punctuations, stop words etc.

Count of words in the text



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

	Review_word_count	Review_Text
0	37	pricethis phone fantastic guysbattery massive ...
1	26	first impression good value money smartphonemu...
2	13	thank flipkart good condition mobile infinite ...
3	49	premium look mobile large display without notc...
4	16	good value money product camera good complain ...

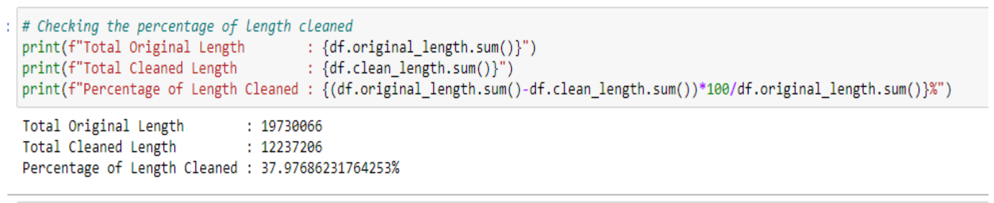


```
# New column (clean_length) after removing punctuations, stopwords in dataset
df['clean_length'] =df.Review_Text.str.len()
df
```

	Review_Text	Ratings	original_length	Review_word_count	clean_length
0	pricethis phone fantastic guysbattery massive ...	5	351	37	225
1	first impression good value money smartphonemu...	5	263	26	181
2	thank flipkart good condition mobile infinite ...	5	102	13	65
3	premium look mobile large display without notc...	5	510	49	332
4	good value money product camera good complain ...	4	177	16	114
...	...	...	...	...	...
45926	product nothing overprice hype regular range e...	1	465	45	284
45927	much satisfy overall performance work like fla...	5	308	33	214
45928	there basic issue tp link decos work mesh syst...	2	906	90	589
45929	home close sqft setup really simple ppoe conn...	3	1594	151	932
45930	setup ups connect satellite main unit use va a...	5	1813	173	1135

45823 rows × 5 columns

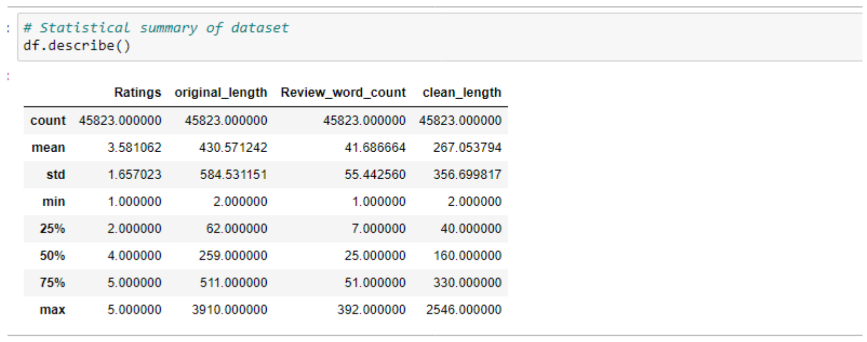
Observation: Thus we see cleaned_length after removing the unwanted charters from the comment text. We can clearly compare the original_length with cleaned_length thus see the characters removed from dataset.



```
# Checking the percentage of length cleaned
print(f"Total Original Length        : {df.original_length.sum()}")
print(f"Total Cleaned Length         : {df.clean_length.sum()}")
print(f"Percentage of Length Cleaned : {(df.original_length.sum()-df.clean_length.sum())*100/df.original_length.sum()}%")
```

```
Total Original Length        : 19730066
Total Cleaned Length         : 12237206
Percentage of Length Cleaned : 37.97686231764253%
```

Thus we see that 37.97% of the text length has been cleaned.



```
# Statistical summary of dataset
df.describe()
```

	Ratings	original_length	Review_word_count	clean_length
count	45823.000000	45823.000000	45823.000000	45823.000000
mean	3.581062	430.571242	41.686664	267.053794
std	1.657023	584.531151	55.442560	356.699817
min	1.000000	2.000000	1.000000	2.000000
25%	2.000000	62.000000	7.000000	40.000000
50%	4.000000	259.000000	25.000000	160.000000
75%	5.000000	511.000000	51.000000	330.000000
max	5.000000	3910.000000	392.000000	2546.000000

Observation: 1)The counts of every column is same which means there are no missing values present in the dataset.

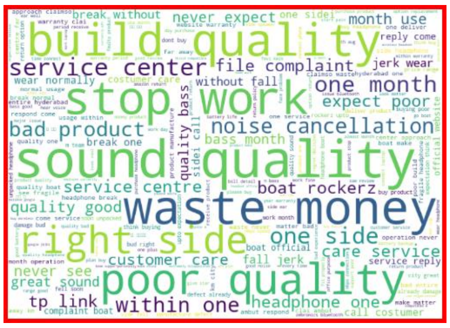
2)The mean value is greater than the median in all the columns except the column Ratings. So, the data in these columns are skewed to right and in target it is skewed to left.

3)There is a huge difference between 75% percentile and max values in original_length and clean_length ,so, we can say there are some outliers present in the data.

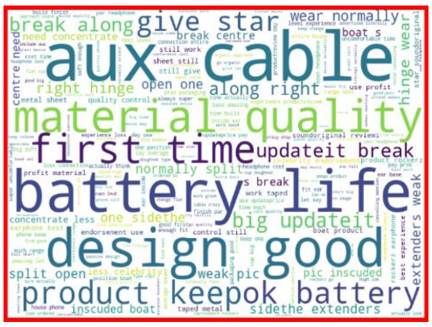
- ## Data Inputs- Logic- Output Relationships

Visualization of the different output labels using wordcloud:So for explaining the input-output logic relationship, I have used wordcloud with customer reviews as per their ratings classification. A word cloud(tag cloud) is a visual representation of words of a text corpus, where word's visual size is proportional to the frequency or importance of the word.

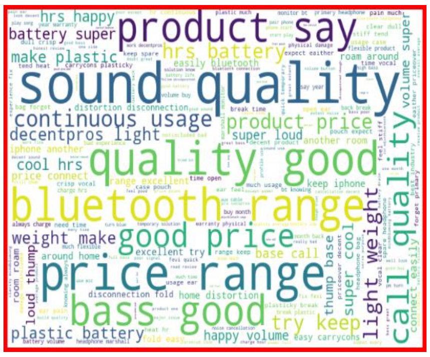
Rating 1



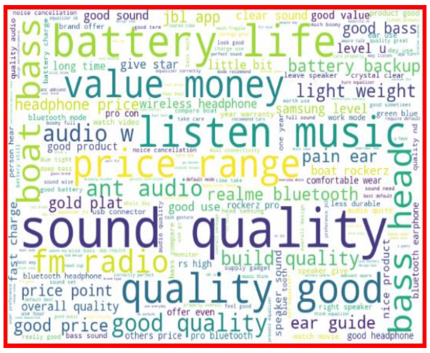
Rating 2



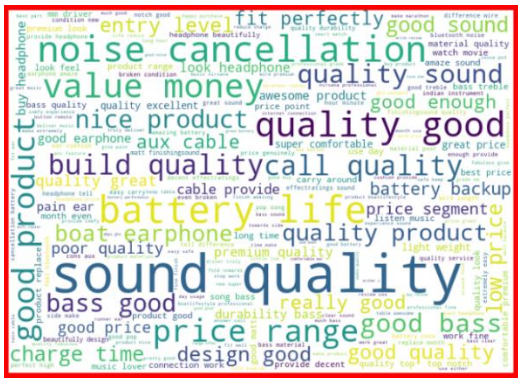
Rating 3



Rating 4



Rating 5



Observation: Thus, we observe that these comments belong to different rating types so using word clouds we can see all the frequently used words in each and every ratings class. It is observed that 5-star rating comments have mostly positive words while the 1-star rating comments is having with negative descriptions.

- State the set of assumptions (if any) related to the problem under consideration

By looking into the target variable/label, we assumed that it was a multiclass classification type of problem. Also, we observed that our dataset was imbalance so we will have to balance the dataset for better prediction accuracy outcome.

Hardware and Software Requirements and Tools Used

Hardware Used:

i. RAM: 8 GB
ii. CPU: Intel® Core™ i3-1005G1 CPU @1.20GHz
iii. GPU: Intel® UHD Graphics

Software Used:

i. Programming language: Python
ii. Distribution: Anaconda Navigator
iii. Browser based language shell: Jupyter Notebook

Libraries/Packages Used:

1. Pandas- a library which is used to read the data, visualisation and analysis of data.

2. NumPy- used for working with array and various mathematical techniques.

3. Seaborn- visualization tool for plotting different types of plot.

4. Matplotlib- It provides an object-oriented API for embedding plots into applications.

5. Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib

6.joblib: It is part of the SciPy ecosystem and provides utilities for pipelining Python jobs.It provides utilities for saving and loading Python objects that make use of NumPy data structures, efficiently.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  The objective of this project is to build an application which can predict the rating by seeing the review. In the long term, this would allow people to better explain and review their purchase with each other in this increasingly digital world.

  Thus this is clearly a multiclass classification problem considering the rating as our label.So we will be loading all the libraries related to it .Scikit-learn library is focused on modeling the data. Some of the most popular groups of models provided by Sklearn are as follows − Before applying algorithms we will have to clean and preprocess the data. Then we will have to split the dataset into training model and testing model. Only then we can apply algorithms on it.

  So we have processes the text comments of the customers with the help of NLP techniques using machine learning. We need to convert text into feature vectors using TF-IDF vectorizer and separate out feature and labels. Also, before building the model, it is made sure that the input data is cleaned and scaled before it was fed into the machine learning models.

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

  Since it's a multiclass classification problem, the following algorithms have been used for model building:

  The algorithms used on training the data are as follows:

  1. MultinomialNB

  2. LinearSVC

  3.Desicion Tree Classifier

  6. Stochastic Gradient Descent Classifier (SDGClassifier)

From all of these above models Stochastic Gradient Descent Classifier (SGD Classifier) gave me good performance so I have used SGD Classifier for further hyper parameter tuning process to improve the model confidence.

Brief description of the selected models:
Model used:

MultinomialNB: The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work. The Multinomial Naive Bayes algorithm is a Bayesian learning approach popular in Natural Language Processing (NLP). The program guesses the tag of a text, such as an email or a newspaper story, using the Bayes theorem. It calculates each tag's likelihood for a given sample and outputs the tag with the greatest chance

LinearSVC: The Linear Support Vector Classifier (SVC) method applies a linear kernel function to perform classification and it performs well with a large number of samples. The objective of a Linear SVC (Support Vector Classifier) is to fit to the data you provide, returning a "best fit" hyperplane that divides, or categorizes, your data. From there, after getting the hyperplane, you can then feed some features to your classifier to see what the "predicted" class is.

Decision Tree Classifier: Decision Tree Learning is supervised learning approach used in statistics, data mining and machine learning. In this formalism, a classification or regression decision tree is used as a predictive model to draw conclusions about a set of observations.Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels

SDGClassifier : Stochastic Gradient Descent (SGD) is a simple yet very efficient approach to fitting linear classifiers and regressors under convex loss functions such as (linear) Support Vector Machines and Logistic Regression. Even though SGD has been around in the machine learning community for a long time, it has received a considerable amount of attention just recently in the context of large-scale learning.

SGD has been successfully applied to large-scale and sparse machine learning problems often encountered in text classification and natural language processing. Given that the data is sparse, the classifiers in this module easily scale to problems with more than 10^5 training examples and more than 10^5 features.

- ## Run and Evaluate selected models

The complete list of algorithms that were used in training and testing the classification model are listed below:

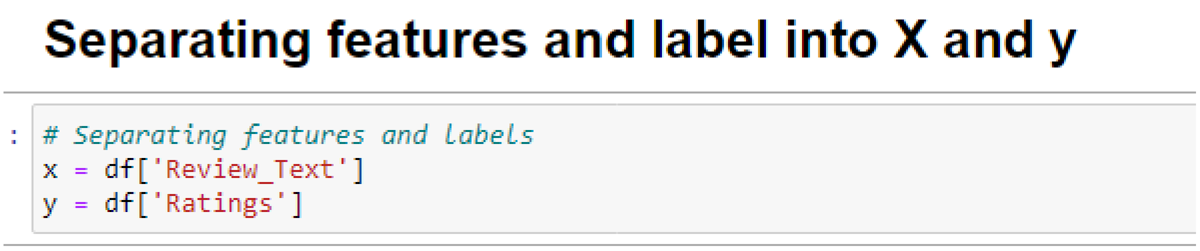
1)Decision Tree Classifier
2)Linear Support Vector Classifier
3)Multinomial Naïve Bayes
4)Stochastic Gradient Descent Classifier

Building Machine Learning Models and evaluation metrics
Models:
We are going to use Decision Tree Classifier, MultinomialNB, Linear Support Vector Classifier and SDGClassifier for finding out the best model among those.

Evaluation metric:
classification_report, confusion_matrix, accuracy_score, F1_score,Precision,Recall

I created a classification function that included the evaluation metrics details for the generation of our Classification Machine Learning models.

First I have separated the dependent and independent variable into X and y,X as Review_Text and y as Ratings.

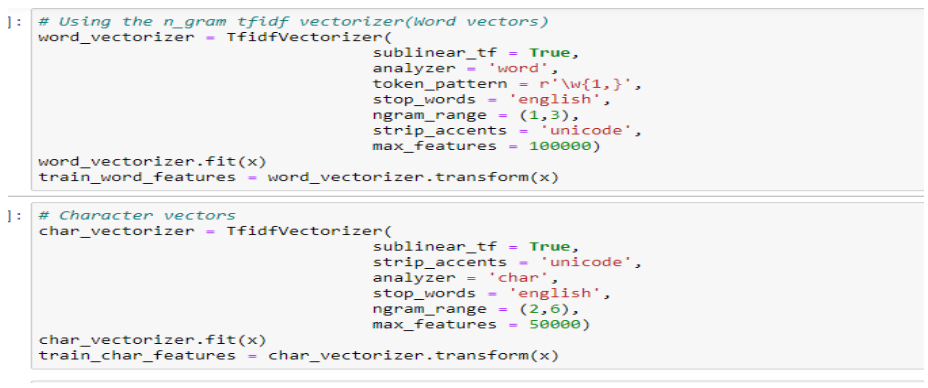


Separating features and label into X and y

```
# Separating features and labels
x = df['Review_Text']
y = df['Ratings']
```

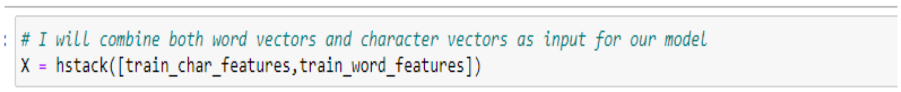
Converting the tokens into vectors

I have used TF_IDF Vectoriser for converting tokens into vectors:The tfidf_vectorizer class implements a vectorizer for calculating term frequency/inverse document frequency (TF/IDF), which can be used as a measure for modeling the importance of terms within documents. It takes in raw texts and returns an array of feature vectors for each input text.In simple words, TFIDF is a numerical statistic that shows the importance of a word in a text document.



```
# Using the n_gram tfidf vectorizer(Word vectors)
word_vectorizer = TfidfVectorizer(
                                  sublinear_tf = True,
                                  analyzer = 'word',
                                  token_pattern = r'\w{1,}',
                                  stop_words = 'english',
                                  ngram_range = (1,3),
                                  strip_accents = 'unicode',
                                  max_features = 100000)
word_vectorizer.fit(x)
train_word_features = word_vectorizer.transform(x)
```

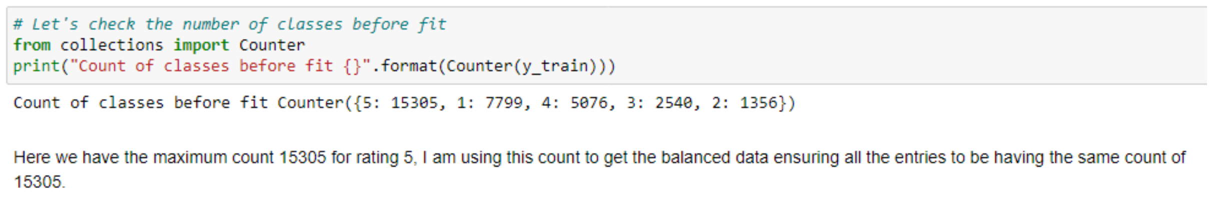
```
# Character vectors
char_vectorizer = TfidfVectorizer(
                                  sublinear_tf = True,
                                  strip_accents = 'unicode',
                                  analyzer = 'char',
                                  stop_words = 'english',
                                  ngram_range = (2,6),
                                  max_features = 50000)
char_vectorizer.fit(x)
train_char_features = char_vectorizer.transform(x)
```



```
# I will combine both word vectors and character vectors as input for our model
X = hstack([train_char_features,train_word_features])
```

Observation: Thus with the help of TfIdf vectorizer,I have converted words and characters into vectors so as to use them as an input to our model. I have converted features into number tokens in the dataset and thus separating out input and output variables.

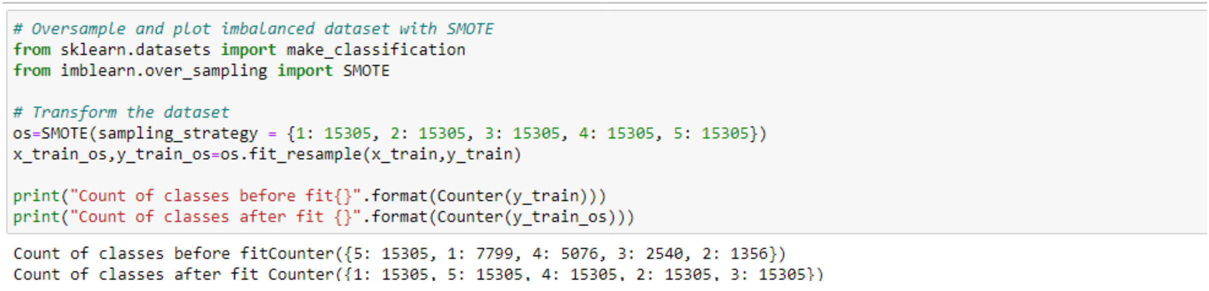
First we need to check whether the data is balanced or not,if not then we need to balance the data.So here we have used Smote oversampling method to balance the data.



```
# Let's check the number of classes before fit
from collections import Counter
print("Count of classes before fit {}".format(Counter(y_train)))
```

```
Count of classes before fit Counter({5: 15305, 1: 7799, 4: 5076, 3: 2540, 2: 1356})
```

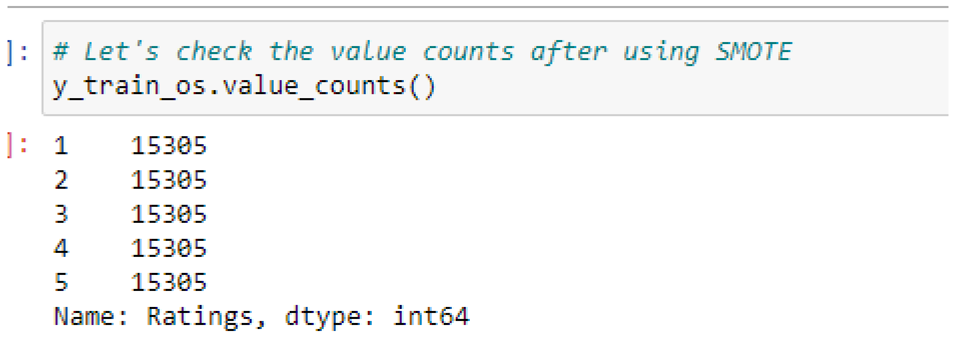
Here we have the maximum count 15305 for rating 5, I am using this count to get the balanced data ensuring all the entries to be having the same count of 15305.



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

# Transform the dataset
os=SMOTE(sampling_strategy = {1: 15305, 2: 15305, 3: 15305, 4: 15305, 5: 15305})
x_train_os,y_train_os=os.fit_resample(x_train,y_train)

print("Count of classes before fit{}".format(Counter(y_train)))
print("Count of classes after fit {}".format(Counter(y_train_os)))
```

```
Count of classes before fitCounter({5: 15305, 1: 7799, 4: 5076, 3: 2540, 2: 1356})
Count of classes after fit Counter({1: 15305, 5: 15305, 4: 15305, 2: 15305, 3: 15305})
```

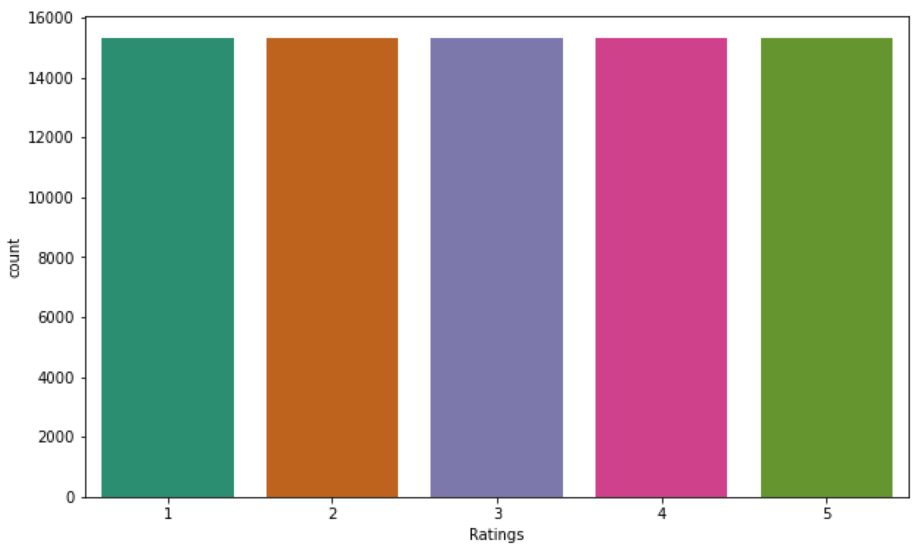
Observation: "SMOTE" is the oversampling mechanism that we are using to ensure that all the categories present in our target label have the same value. I have set the count of all the categories to be 15305 which is the highest count in the target column. After applying oversampling we are once again listing the values of our label column to cross verify the updated information. Here we see that we have successfully resolved the class imbalance problem and now all the categories have same data ensuring that the machine learning model does not get biased towards one category.



```
]: # Let's check the value counts after using SMOTE
   y_train_os.value_counts()
```

```
]: 1    15305
   2    15305
   3    15305
   4    15305
   5    15305
   Name: Ratings, dtype: int64
```

```
# Visualizing the data after oversampling
plt.figure(figsize=(10,6))
sns.countplot(y_train_os,palette="Dark2")
plt.show()
```

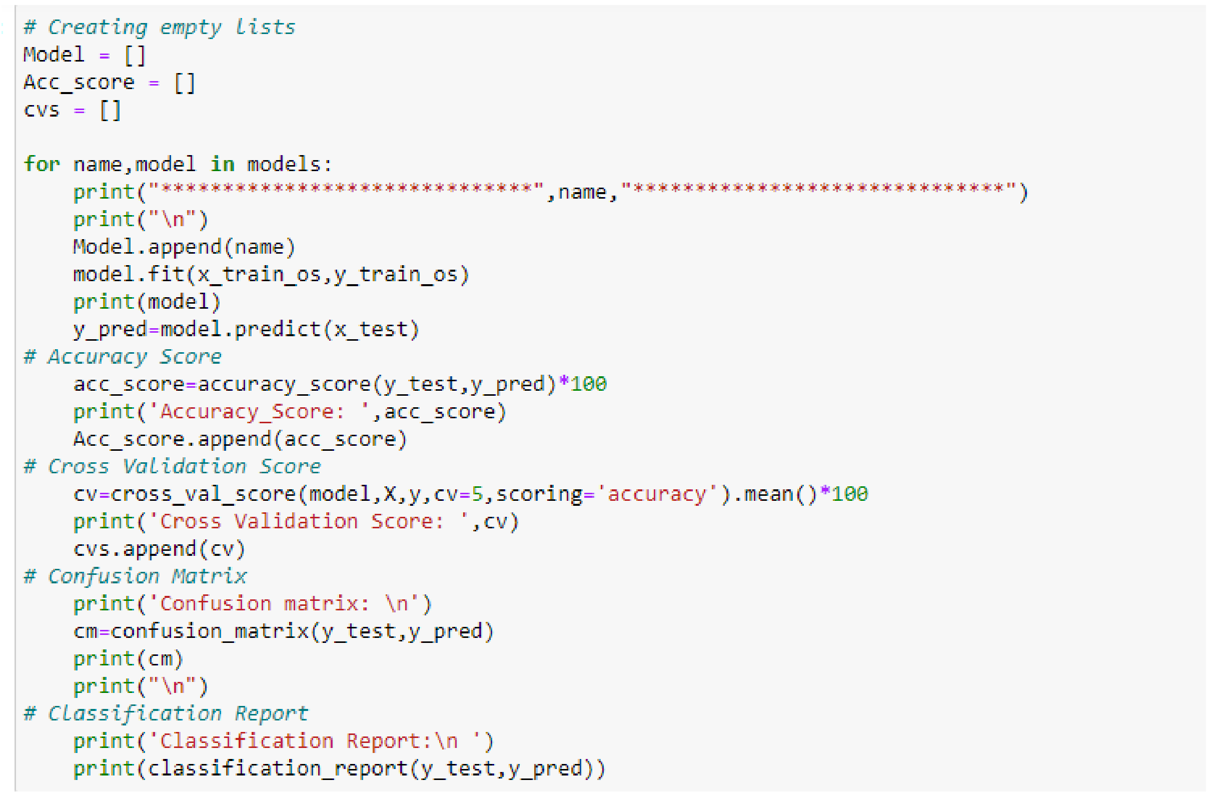


Observation: As we can observe all the categories in the target variable "Ratings" have equal values. The class imbalancing issue has been solved. Now we can build our machine learning models.

Building Models

```
# Creating instances for different Classifiers

mnb = MultinomialNB()
SVC = LinearSVC()
SGD = SGDClassifier()
DTC = DecisionTreeClassifier()


# Creating a list model where all the models will be appended for further evaluation in loop.
models=[]
models.append(('MultinomialNB',mnb))
models.append(('LinearSVC',SVC))
models.append(('SGDClassifier',SGD))
models.append(('DecisionTreeClassifier',DTC))
```

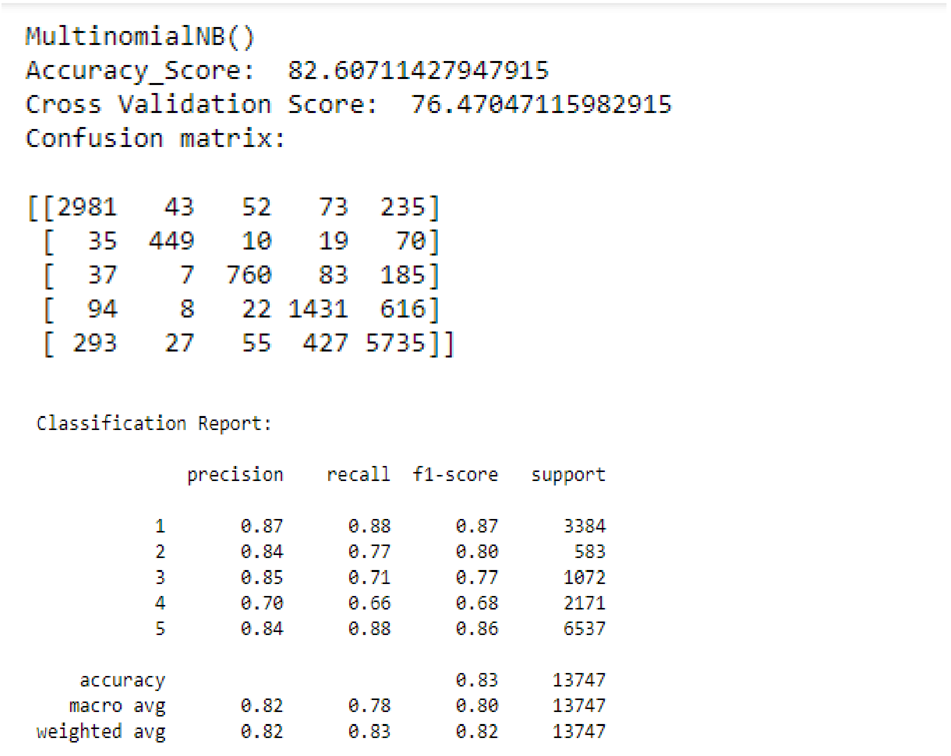
I have created 4 different classification algorithms and are appended in the variable models. Now, let's run a for loop which contains the accuracy of the models along with different evaluation metrics.



```
# Creating empty lists
Model = []
Acc_score = []
cvs = []

for name,model in models:
    print("******************************",name,"******************************")
    print("\n")
    Model.append(name)
    model.fit(x_train_os,y_train_os)
    print(model)
    y_pred=model.predict(x_test)
# Accuracy Score
    acc_score=accuracy_score(y_test,y_pred)*100
    print('Accuracy_Score: ',acc_score)
    Acc_score.append(acc_score)
# Cross Validation Score
    cv=cross_val_score(model,X,y,cv=5,scoring='accuracy').mean()*100
    print('Cross Validation Score: ',cv)
    cvs.append(cv)
# Confusion Matrix
    print('Confusion matrix: \n')
    cm=confusion_matrix(y_test,y_pred)
    print(cm)
    print("\n")
# Classification Report
    print('Classification Report:\n ')
    print(classification_report(y_test,y_pred))
```

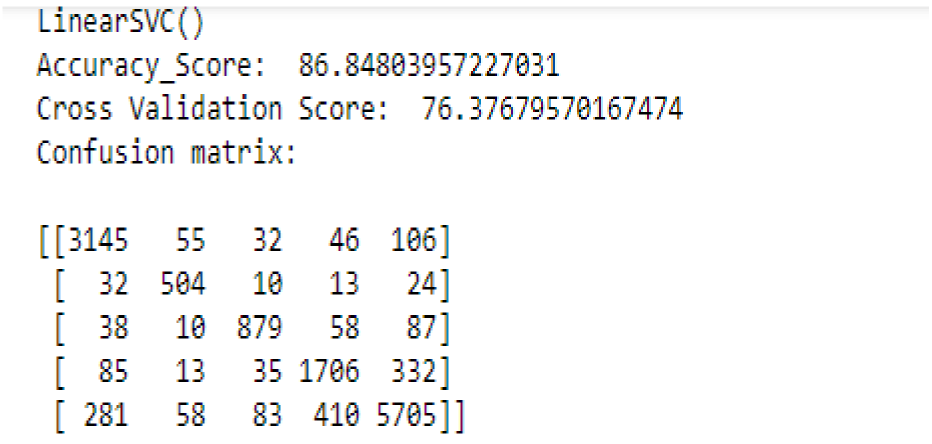
Output:

MultinomialNB

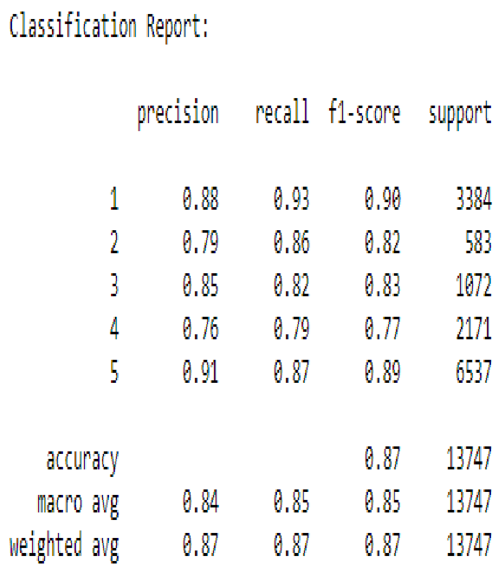


```
MultinomialNB()
Accuracy_Score:  82.60711427947915
Cross Validation Score:  76.47047115982915
Confusion matrix:

[[2981   43   52   73  235]
 [  35  449   10   19   70]
 [  37    7  760   83  185]
 [  94    8   22 1431  616]
 [ 293   27   55  427 5735]]


Classification Report:

              precision    recall  f1-score   support

           1       0.87      0.88      0.87      3384
           2       0.84      0.77      0.80       583
           3       0.85      0.71      0.77      1072
           4       0.70      0.66      0.68      2171
           5       0.84      0.88      0.86      6537

    accuracy                           0.83     13747
   macro avg       0.82      0.78      0.80     13747
weighted avg       0.82      0.83      0.82     13747
```

LinearSVC:

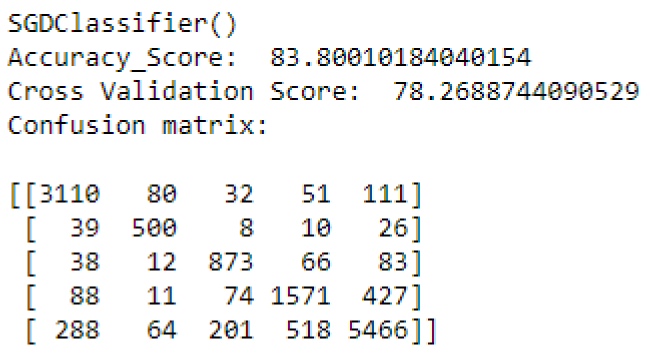


```
LinearSVC()
Accuracy_Score:  86.84803957227031
Cross Validation Score:  76.37679570167474
Confusion matrix:

[[3145   55   32   46  106]
 [  32  504   10   13   24]
 [  38   10  879   58   87]
 [  85   13   35 1706  332]
 [ 281   58   83  410 5705]]
```

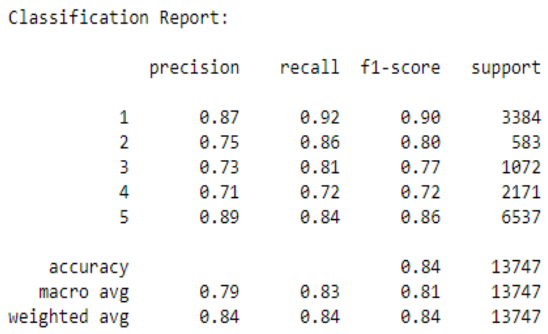


```
Classification Report:

              precision    recall  f1-score   support

           1       0.88      0.93      0.90      3384
           2       0.79      0.86      0.82       583
           3       0.85      0.82      0.83      1072
           4       0.76      0.79      0.77      2171
           5       0.91      0.87      0.89      6537

    accuracy                           0.87     13747
   macro avg       0.84      0.85      0.85     13747
weighted avg       0.87      0.87      0.87     13747
```

SGDClassifier

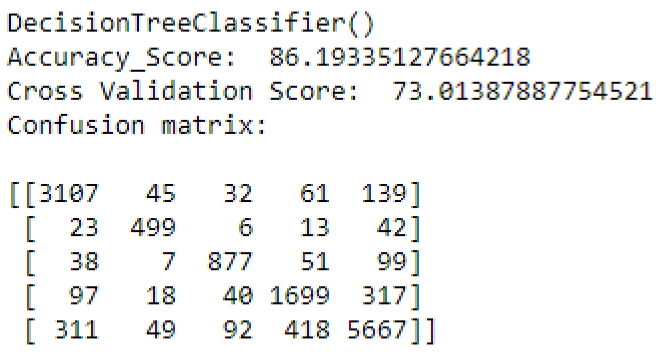


```
SGDClassifier()
Accuracy_Score:  83.80010184040154
Cross Validation Score:  78.2688744090529
Confusion matrix:

[[3110   80   32   51  111]
 [  39  500    8   10   26]
 [  38   12  873   66   83]
 [  88   11   74 1571  427]
 [ 288   64  201  518 5466]]
```

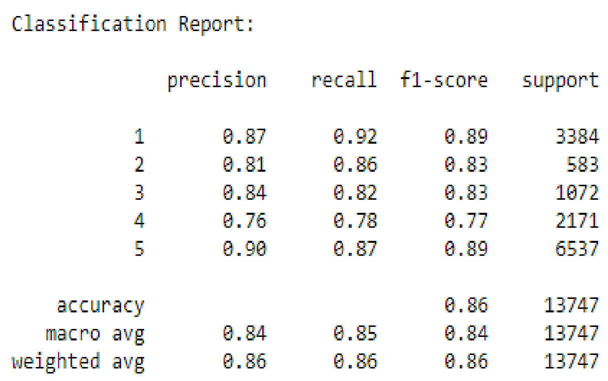


```
Classification Report:

              precision    recall  f1-score   support

           1       0.87      0.92      0.90      3384
           2       0.75      0.86      0.80       583
           3       0.73      0.81      0.77      1072
           4       0.71      0.72      0.72      2171
           5       0.89      0.84      0.86      6537

    accuracy                           0.84     13747
   macro avg       0.79      0.83      0.81     13747
weighted avg       0.84      0.84      0.84     13747
```

Decision Tree Classifier:

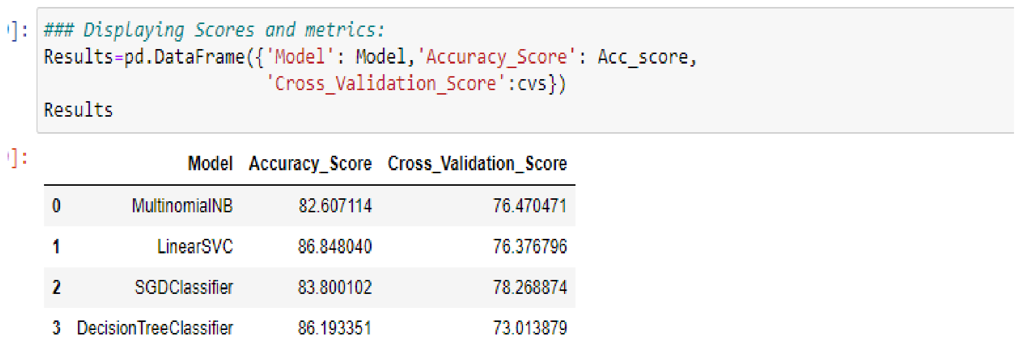


```
DecisionTreeClassifier()
Accuracy_Score:  86.19335127664218
Cross Validation Score:  73.01387887754521
Confusion matrix:

[[3107   45   32   61  139]
 [  23  499    6   13   42]
 [  38    7  877   51   99]
 [  97   18   40 1699  317]
 [ 311   49   92  418 5667]]
```



```
Classification Report:

              precision    recall  f1-score   support

           1       0.87      0.92      0.89      3384
           2       0.81      0.86      0.83       583
           3       0.84      0.82      0.83      1072
           4       0.76      0.78      0.77      2171
           5       0.90      0.87      0.89      6537

    accuracy                           0.86     13747
   macro avg       0.84      0.85      0.84     13747
weighted avg       0.86      0.86      0.86     13747
```

MODEL SELECTION:



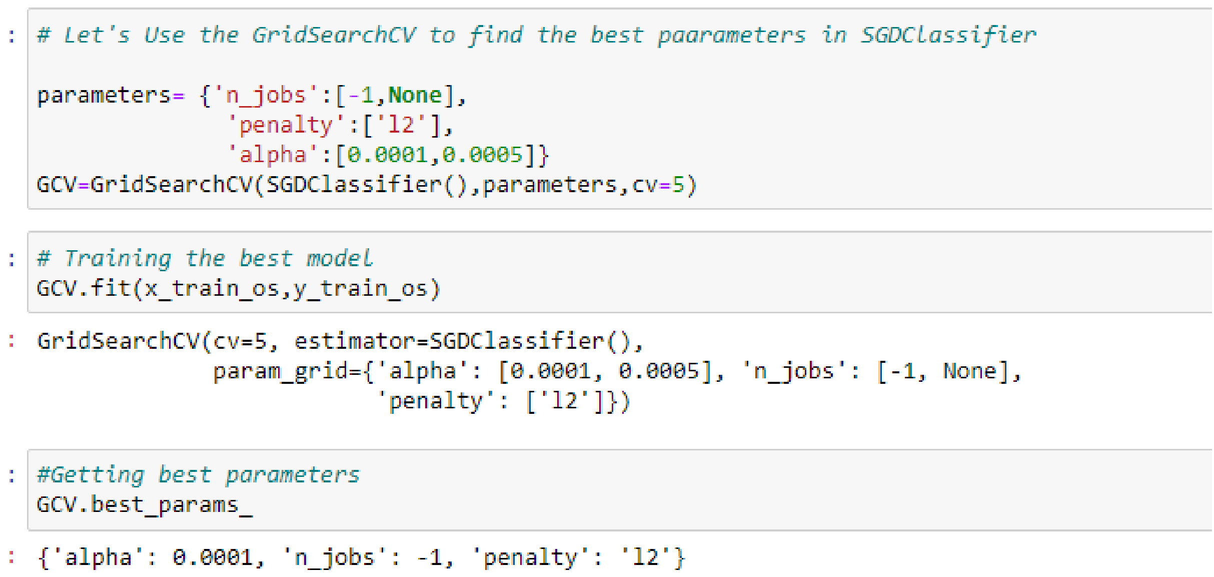
```
]: ### Displaying Scores and metrics:
   Results=pd.DataFrame({'Model': Model,'Accuracy_Score': Acc_score,
                         'Cross_Validation_Score':cvs})
   Results
```

]:

	Model	Accuracy_Score	Cross_Validation_Score
0	MultinomialNB	82.607114	76.470471
1	LinearSVC	86.848040	76.376796
2	SGDClassifier	83.800102	78.268874
3	DecisionTreeClassifier	86.193351	73.013879

Observation: Thus we see that after modelling different classification algorithms,We choose SGDClassifier as the best fitting model as the difference between accuracy and cross validation score is less among other models.Now ,we will try Hyperparameter Tuning to find out the best parameters and using them to improve the scores and metrics values.

Hyperparameter Tuning



```
# Let's Use the GridSearchCV to find the best paarameters in SGDClassifier

parameters= {'n_jobs':[-1,None],
             'penalty':['l2'],
             'alpha':[0.0001,0.0005]}
GCV=GridSearchCV(SGDClassifier(),parameters,cv=5)
```

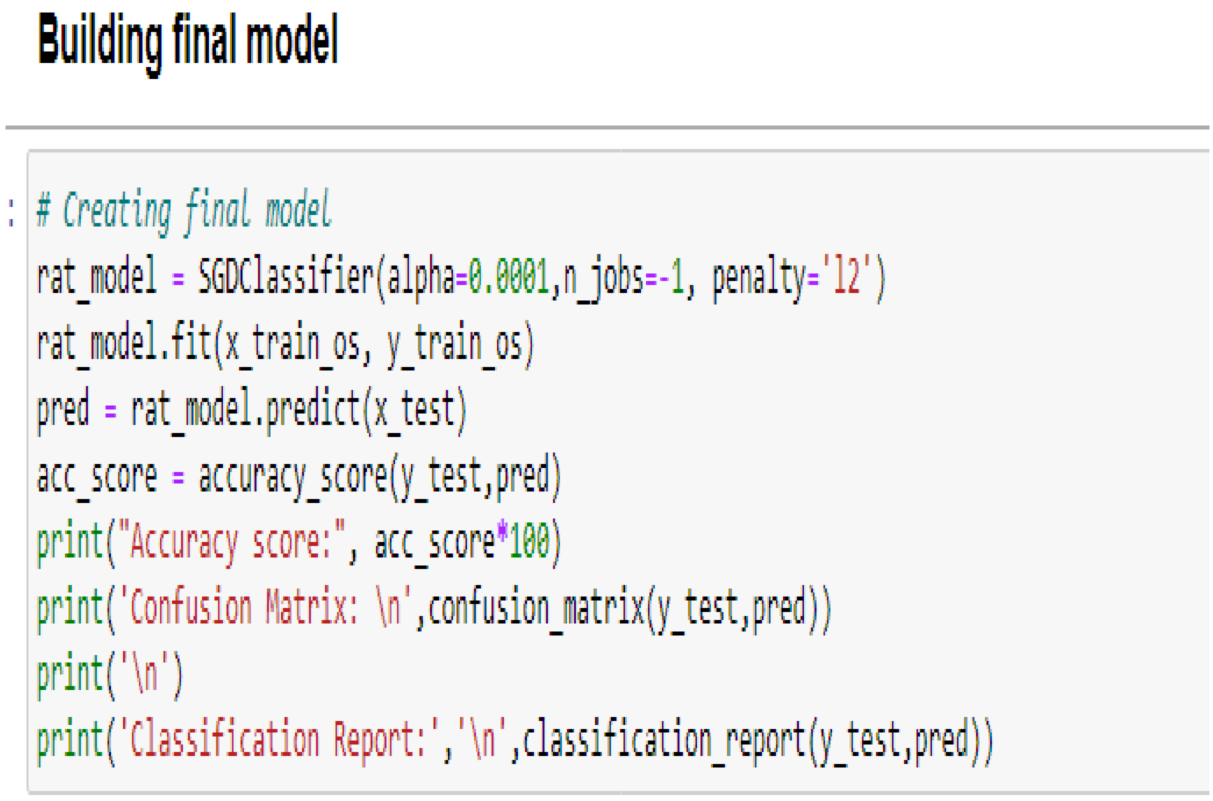
```
# Training the best model
GCV.fit(x_train_os,y_train_os)
```

```
GridSearchCV(cv=5, estimator=SGDClassifier(),
             param_grid={'alpha': [0.0001, 0.0005], 'n_jobs': [-1, None],
                         'penalty': ['l2']})
```

```
#Getting best parameters
GCV.best_params_
```

```
{'alpha': 0.0001, 'n_jobs': -1, 'penalty': 'l2'}
```

Observation: Thus we find the best parameters of SGD Classifier as above.



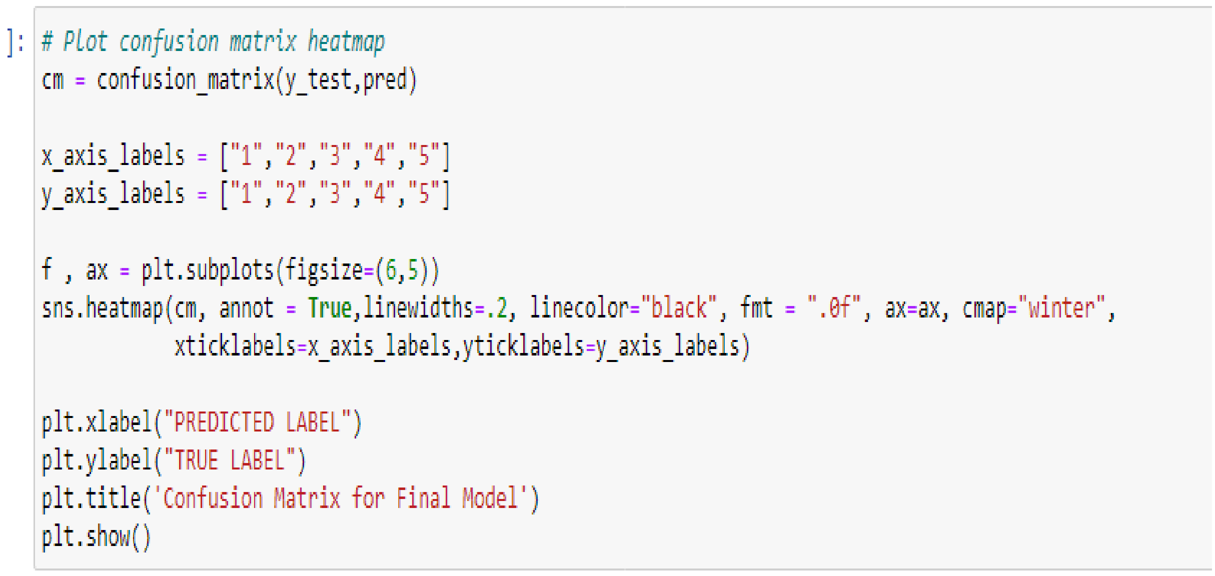
Building final model

```
# Creating final model
rat_model = SGDClassifier(alpha=0.0001,n_jobs=-1, penalty='l2')
rat_model.fit(x_train_os, y_train_os)
pred = rat_model.predict(x_test)
acc_score = accuracy_score(y_test,pred)
print("Accuracy score:", acc_score*100)
print('Confusion Matrix: \n',confusion_matrix(y_test,pred))
print('\n')
print('Classification Report:','\n',classification_report(y_test,pred))
```

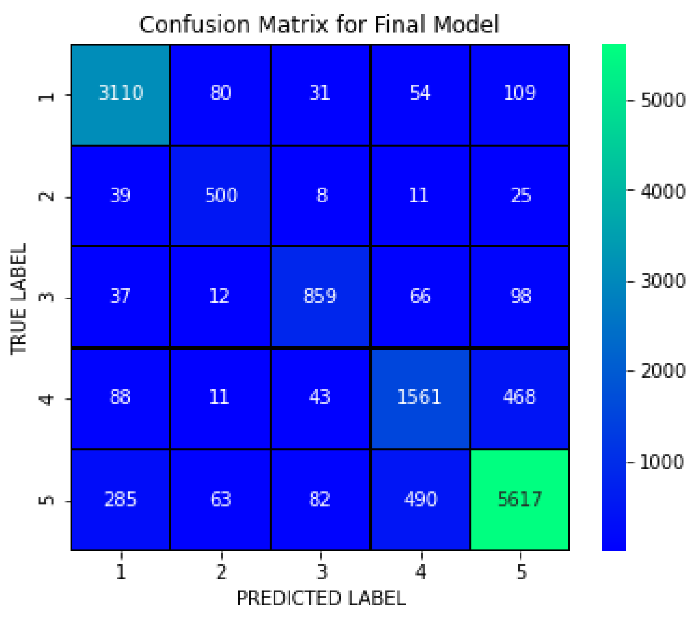
```
Accuracy score: 84.7239397686768
Confusion Matrix:
 [[3110   80   31   54  109]
 [  39  500    8   11   25]
 [  37   12  859   66   98]
 [  88   11   43 1561  468]
 [ 285   63   82  490 5617]]


Classification Report:
               precision    recall  f1-score   support

           1       0.87      0.92      0.90      3384
           2       0.75      0.86      0.80       583
           3       0.84      0.80      0.82      1072
           4       0.72      0.72      0.72      2171
           5       0.89      0.86      0.87      6537

    accuracy                           0.85     13747
   macro avg       0.81      0.83      0.82     13747
weighted avg       0.85      0.85      0.85     13747
```

Observation: Thus we have successfully applied the hyperparameter tuning using GridsearchCV and found out the best parameters of the SGDClassifier and the accuracy of the model has been increased after hyperparameter tuning and received the accuracy score as 85% which is very good and improved from 84%.

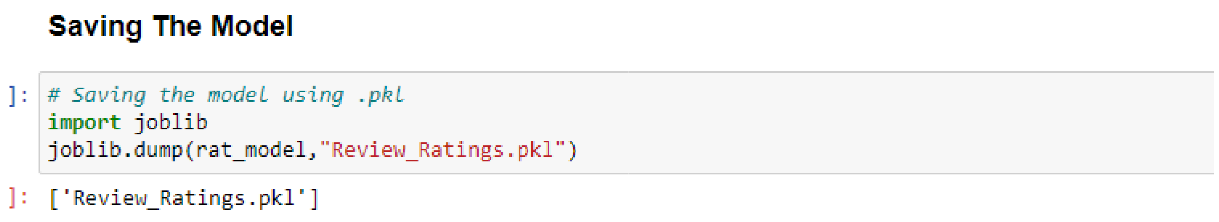
Confusion Matrix



```
]: # Plot confusion matrix heatmap
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=(6,5))
sns.heatmap(cm, annot = True,linewidths=.2, linecolor="black", fmt = ".0f", ax=ax, cmap="winter",
            xticklabels=x_axis_labels,yticklabels=y_axis_labels)

plt.xlabel("PREDICTED LABEL")
plt.ylabel("TRUE LABEL")
plt.title('Confusion Matrix for Final Model')
plt.show()
```



Observation: With the help of confusion matrix we can able to see actual and predicted values for the final model. And also we can understand the number of times we got the correct outputs and the number of times my model missed to provide the correct prediction.



Saving The Model

```
]: # Saving the model using .pkl
   import joblib
   joblib.dump(rat_model,"Review_Ratings.pkl")

]: ['Review_Ratings.pkl']
```

I am using joblib to save the final classification model.

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

The key metrics used here were Accuracy Score, Precision, Recall, F1 score, Confusion Matrix. We tried to find out the best parameters and also to increase our scores by using Hyperparameter Tuning and used GridSearchCV method

1.Accuracy Score Model accuracy is a machine learning model performance metric that is defined as the ratio of true positives and true negatives to all positive and negative observations. In other words,

accuracy tells us how often we can expect our machine learning model will correctly predict an outcome out of the total number of times it made predictions.
Mathematically, it represents the ratio of the sum of true positive and true negatives out of all the predictions.

Accuracy Score = (TP + TN)/ (TP + FN + TN + FP)

2.Precision: The model precision score measures the proportion of positively predicted labels that are actually correct. Precision is also known as the positive predictive value. Precision is used in conjunction with the recall to trade-off false positives and false negatives. Precision is affected by the class distribution. If there are more samples in the minority class, then precision will be lower. Precision can be thought of as a measure of exactness or quality
The precision score is a useful measure of the success of prediction when the classes are very imbalanced. **Mathematically, it represents the ratio of true positive to the sum of true positive and false positive.**
Precision Score = TP / (FP + TP)

3.Recall: Model recall score represents the model's ability to correctly predict the positives out of actual positives. This is unlike precision which measures how many predictions made by models are actually positive out of all positive predictions made. For example: If your machine learning model is trying to identify positive reviews, the recall score would be what percent of those positive reviews did your machine learning model correctly predict as a positive. In other words, it measures how good our machine learning model is at identifying all actual positives out of all positives that exist within a dataset. The higher the recall score, the better the machine learning model is at identifying both positive and negative examples. Recall is also known as sensitivity or the true positive rate. A high recall score indicates that the model is good at identifying positive examples.
Mathematically, it represents the ratio of true positive to the sum of true positive and false negative.

Recall Score = TP / (FN + TP)

4.F1: Model F1 score represents the model score as a function of precision and recall score. F-score is a machine learning model performance metric that gives equal weight to both the Precision and Recall for measuring its performance in terms of accuracy, making it an alternative to Accuracy metrics (it doesn't require us to know the total number of observations). It's often used as a single value that provides high-level information about the model's output quality. This is a useful measure of the model in the scenarios where one tries to optimize either of precision or recall score and as a result, the model performance suffers. The F1 score becomes especially valuable when working on classification models in which your data set is imbalanced.

Mathematically, it can be represented as a harmonic mean of precision and recall score.

F1 Score = 2* Precision Score * Recall Score/ (Precision Score + Recall Score/)

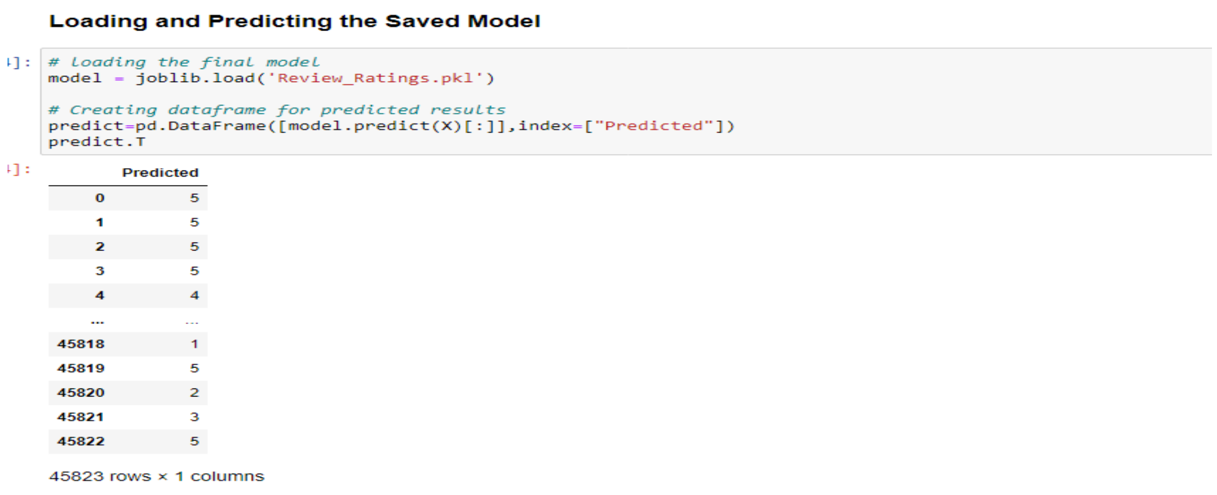
5.Confusion Matrix is one of the evaluation metrics for machine learning classification problems, where a trained model is being evaluated for accuracy and other performance measures. And this matrix is called the confusion matrix since it results in an output that shows how the system is confused between the two classes.

6.Cross validation: Cross-validation is a technique used to protect against overfitting in a predictive model, particularly in a case where the amount of data may be limited. In cross-validation, you make a fixed number of folds (or partitions) of the data, run the analysis on each fold, and then average the overall error estimate.

7.Hyperparameter tuning is choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a model argument whose value is set before the learning process begins. The key to machine learning algorithms is hyperparameter tuning. We must select from a specific list of hyperparameters for a given model as it varies from model to model.

Hyper parameter optimisation in machine learning intends to find the hyper parameters of a given machine learning algorithm that deliver the best performance as measured on a validation set.

Using Scikit-Learn’s GridSearchCV method: GridSearchCV is a library function that is a member of sklearn's model_selection package. It helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, you can select the best parameters from the listed hyperparameters.

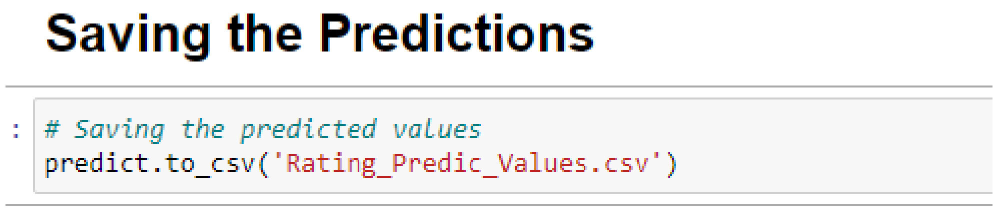


Loading and Predicting the Saved Model

```
# loading the final model
model = joblib.load('Review_Ratings.pkl')

# Creating dataframe for predicted results
predict=pd.DataFrame([model.predict(X)[:]],index=["Predicted"])
predict.T
```

```
       Predicted
    0          5
    1          5
    2          5
    3          5
    4          4
  ...        ...
45818          1
45819          5
45820          2
45821          3
45822          5

45823 rows × 1 columns
```

Observation: Thus after saving the loaded model,i used it for prediction test for reviews ratings. Using classification model, we have got the predicted values for review ratings



Saving the Predictions

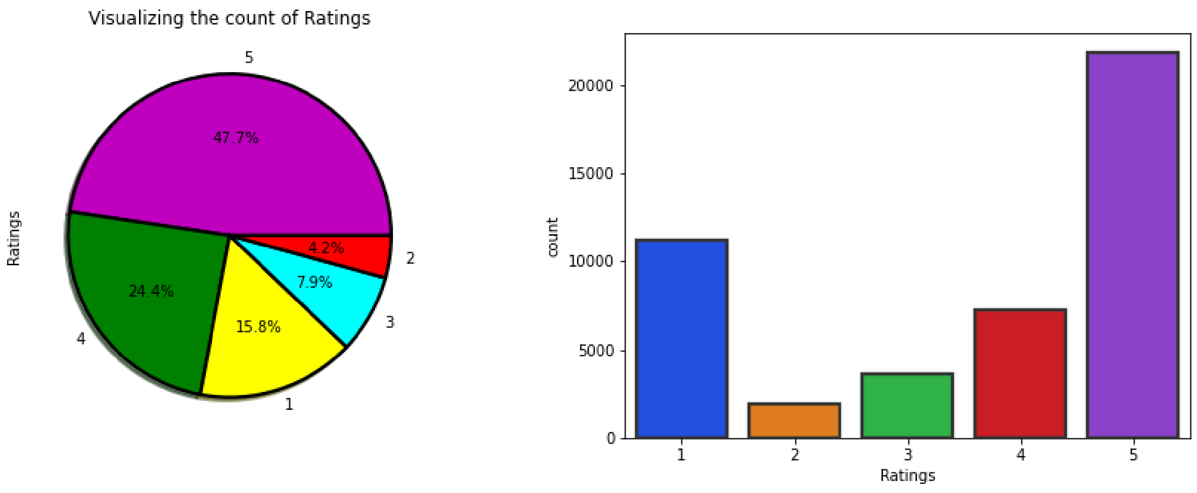
```
# Saving the predicted values
predict.to_csv('Rating_Predic_Values.csv')
```

- ## Visualizations

Now, we will see the different plots done with this dataset in order to know the insight of the data present. Below are the codes given for the plots and the output obtained:

```
# Visualizing the target variable
print(df['Ratings'].value_counts())
f,ax=plt.subplots(1,2,figsize=(15,5))
labels = ['5', '4', '1', '3', '2']
colors = ["m", "green", "yellow", "cyan", "red"]
df['Ratings'].value_counts().plot.pie(autopct='%1.1f%%',ax=ax[0],shadow=True,labels=labels,fontsize=10,colors=colors,
                                       wedgeprops = {'linewidth':2.3, 'edgecolor':'k'},
                                       title = 'Visualizing the count of Ratings')
ax = sns.countplot('Ratings', data=df, ax=ax[1],palette="bright",linewidth=2.3, edgecolor=".2")
plt.show()
```

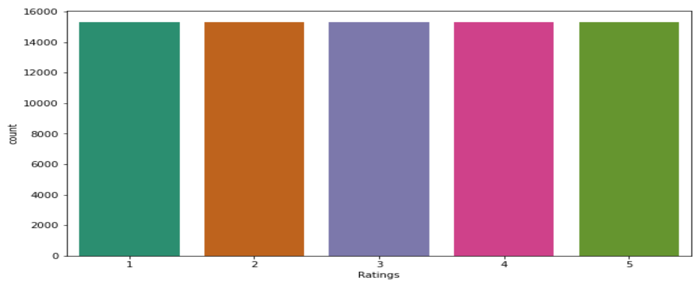
```
5    21842
1    11183
4     7247
3     3612
2     1939
Name: Ratings, dtype: int64
```



Observation: Thus ,we have an idea that more of the customers review pointed towrds 5 star followed by 1 star and least is 2 star .Around 47.7% of the texts are rated as 5 and only 4.2% of the texts rated as 2 stars.So, we can say that this is an "imbalance problem" which we need to make it balanced by manually choosing the same number of records for each and every class and ensuring that the dataset get balanced multiclass label variable.

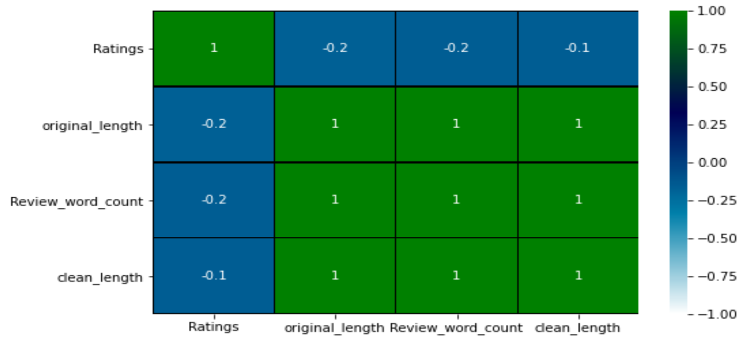
Balance Data:

```
# Visualizing the data after oversampling
plt.figure(figsize=(10,6))
sns.countplot(y_train_os,palette="Dark2")
plt.show()
```



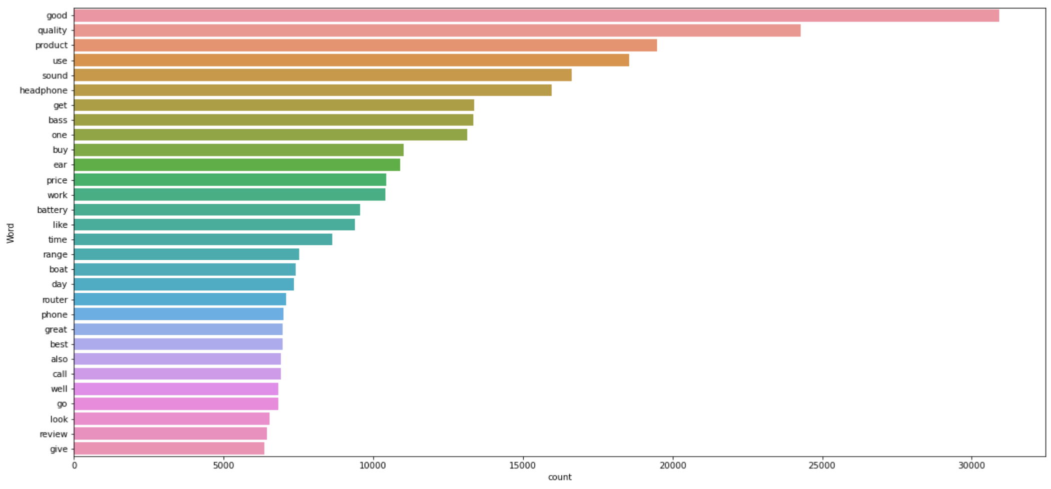
Visualizing the correlation matrix by plotting heat map.

```
]: # Visualizing the correlation matrix by plotting heat map.
   plt.figure(figsize=(8,5))
   sns.heatmap(df_new.corr(),linewidths=.1,vmin=-1, vmax=1,fmt='.1g',linecolor="black",annot=True,cmap="ocean_r",
               annot_kws={'size':10})
   plt.yticks(rotation=0);
```



Observation: The heat map gives the correlation between features and label. We can also observe the correlation between one feature to another.

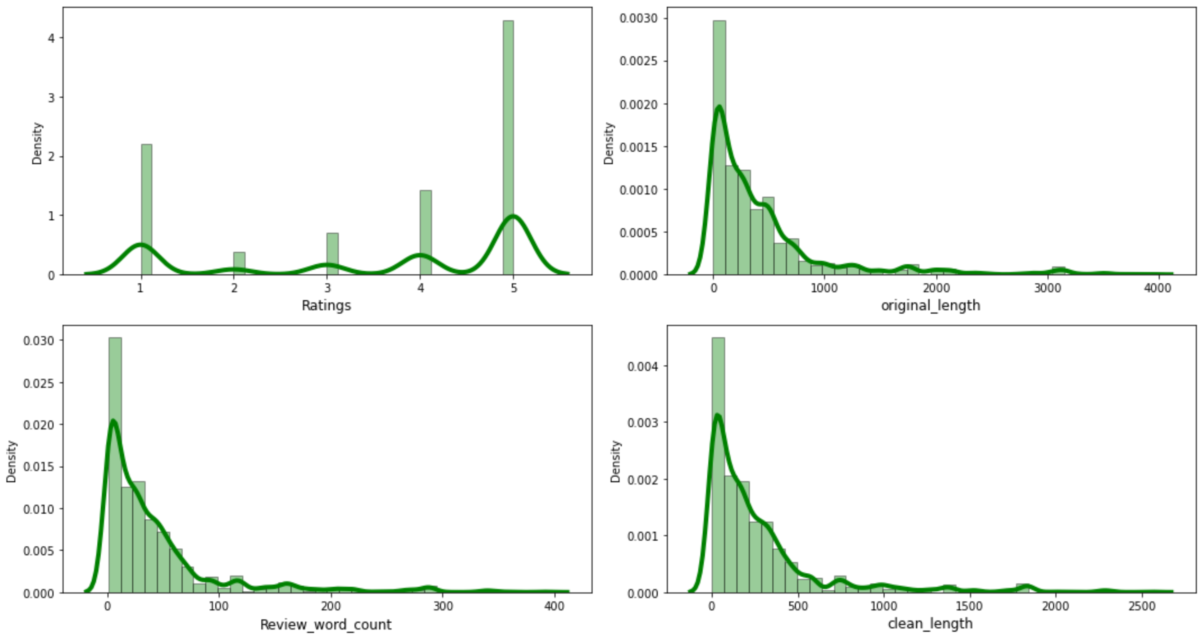
Top 30 most frequently occurring words:



Observation:Thus we observe that the topmost accuring word is “Good” followed by “quality” and least is “give” .

Histogram and density plots for checking how the data has been distributed

```python
# Checking how the data has been distriubted in each column using histogram and density plots
col = ["Ratings","original_length","Review_word_count","clean_length"]
plt.figure(figsize=(15,8),facecolor='white')
plotnumber=1
for column in col:
    if plotnumber<=4:
        ax=plt.subplot(2,2,plotnumber)
        sns.distplot(df[column],color="green",hist=True, kde = True,bins = int(180/5),hist_kws = {'edgecolor':'black'},
            kde_kws = {'linewidth':4})
        plt.xlabel(column,fontsize=12)
    plotnumber+=1
plt.tight_layout()
```



Observation:From the dist plot we can notice that data in all the columns are skewed to right except the target column. Which means the mean value is greater than the median in these columns.

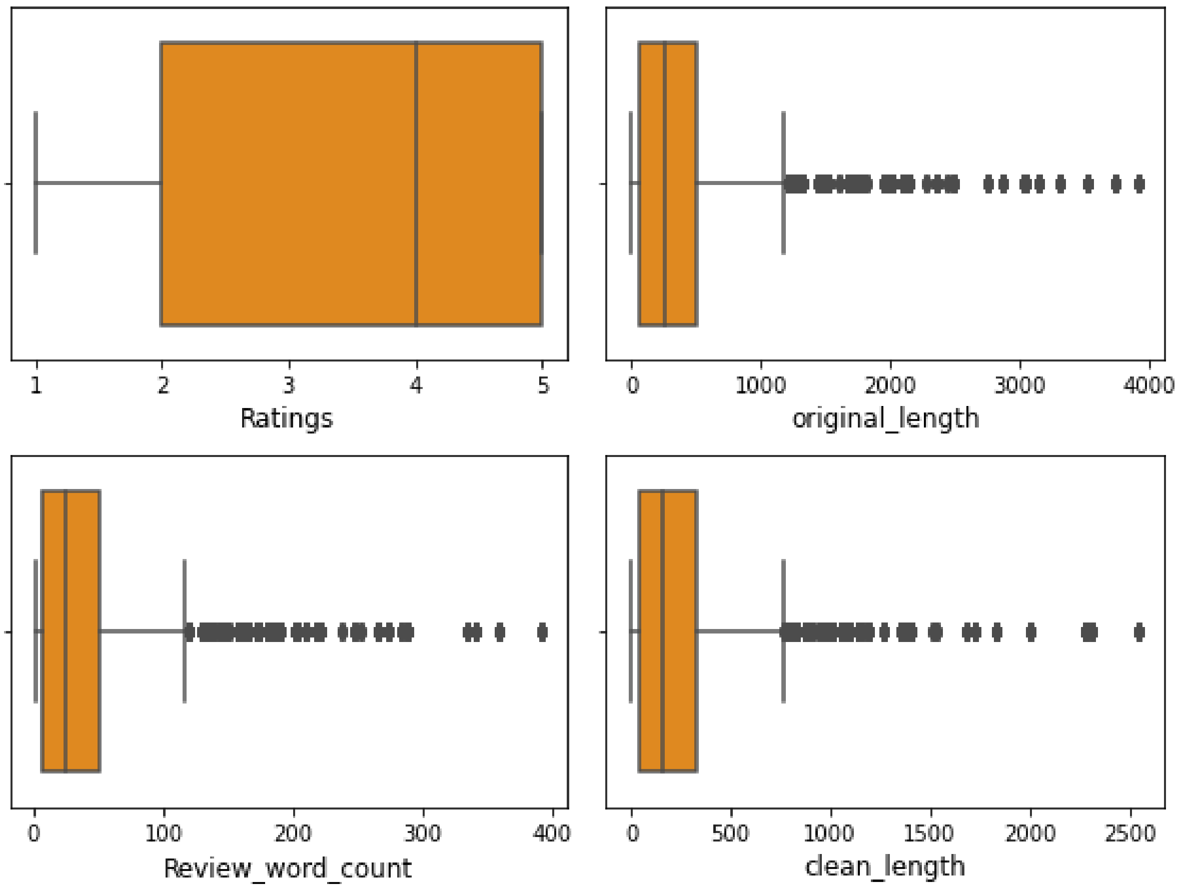
The plot of Review_word_count shows that most of the text is having the no. of words in the range of 0 to 100,some of the text beyond acts as outlier.

Also, the plot of clean_length shows that most of our text is having the number of words in the range of 0 to 500, and some of the text are too lengthy which are out of range which acts as outliers in our data.

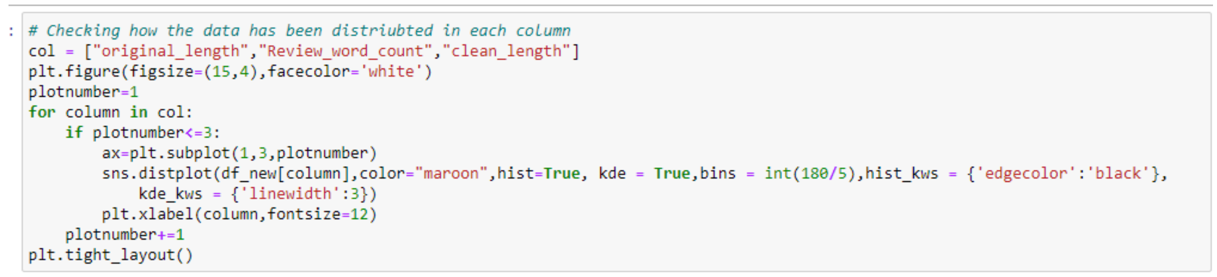
Since there is skewness in the data we need to remove it before building the machine learning models.

Identifying Outiers

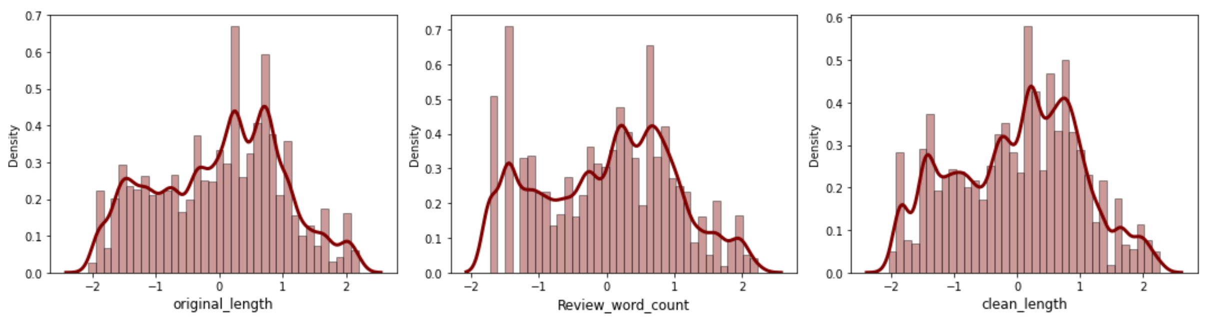
```
]: # Checking how the data has been distriubted in each column using histogram and density plots
col = ["Ratings","original_length","Review_word_count","clean_length"]
plt.figure(figsize=(8,6),facecolor='white')
plotnumber=1
for column in col:
    if plotnumber<=4:
        ax=plt.subplot(2,2,plotnumber)
        sns.boxplot(df[column],color="darkorange")
        plt.xlabel(column,fontsize=12)
    plotnumber+=1
plt.tight_layout()
```



Observation:As we can observe all the columns contains outliers except the target column that is "Ratings" so we have handled the outliers using appropriate techniques.



```
# Checking how the data has been distriubted in each column
col = ["original_length","Review_word_count","clean_length"]
plt.figure(figsize=(15,4),facecolor='white')
plotnumber=1
for column in col:
    if plotnumber<=3:
        ax=plt.subplot(1,3,plotnumber)
        sns.distplot(df_new[column],color="maroon",hist=True, kde = True,bins = int(180/5),hist_kws = {'edgecolor':'black'},
            kde_kws = {'linewidth':3})
        plt.xlabel(column,fontsize=12)
    plotnumber+=1
plt.tight_layout()
```



Observation: The data looks almost normal after removing the skewness compared to the previous data.

CONCLUSIONS

Key Findings and Conclusions of the Study

The objective of this project is to build an application which can predict the rating by seeing the review. In the long term, this would allow people to better explain and review their purchase with each other in this increasingly digital world. This model helps us to understand the ratings of the products. On the basis of product review ratings one can be aware of the positive and negative review of the products and thereby be extra precautioned in buying that product.

Thus this is clearly a multiclass classification problem considering the rating as our label.So we will be loading all the libraries related to it
So first I scrapped the data using different websites like amazon and flipkart. and saved them in csv format.I started with loading the dataset and carry out data analysis and then did the EDA process by removing all the unwanted words,stopwords etc from the review comments with visualization patterns using pie-plot, count plot, bar plot and visualize through wordcloud and learnt about different relationship between the features and target variable.

Then I did the model training first using TF_IDFvectoriser, I converted comments into vectors first , then check for imbalanced class then balanced it using smote oversampling method building the model and finding out the best model out of several classification models on the basis of different evaluation metrices scores like accuracy score, cross validation,f1,confusion matrix.
We find that SDGClassifier model was the best fitted model as the difference between accuracy score and cross validation score was less. Then I did hyperparameter tunning through GridSearchCV on best model taking the parameters to improve the score. Thus, concluded that SDGClassifier was performing well among other models.
Thus, finally concluding saying that SDGClassifier was having the highest precision accuracy for prediction of rating of review comments with machine learning data.

I saved the best model using joblib method and loaded the model for prediction test. Thus, this model can be used in further deployment process.

Learning Outcomes of the Study in respect of Data Science

1.We knew about different Natural Language Processing techniques to be applied for text processing for data cleaning like conversion to lowercase, removal of stopwords,removal of unwated words like question mark,punctuation etc,stemming and finally converting the strings into vectors using TFIDF vectorizer for building of model.

2.In this project, we present various algorithms while predicting classification models with good accuracy. We tested various models such as LinearSVC,SDGClassifier,Decision Tree Classifier,MultinomialNB, and selected the best fit among models and also done hyperparameter tuning.Also we applied different evaluation metrics like hamming loss and log loss apart from accuracy.

Limitations of this work and Scope for Future Work

1.The dataset was imbalance and lot of comments texts.

2.Time consumption was more for model building, so I limited to only 4 models instead of taking more models because the processing time was more than beyond 1-2 hours .

3.More models like Random Forest, Bernoulli Classifier can be taken can be used to solve the same problem.

4. While scrapping also, there were some technical glitch from the websites

5.We can still improve error, model accuracy with some feature engineering and by doing some extensive hyperparameter tuning on it.