Ratings Prediction Project

Build predictive models to automate the process of predicting the ratings by seeing the review.

Table of Contents:

Problem Definition

We have a client who has a website where people write different reviews for technical products. Now they are adding a new feature to their website i.e. the reviewer will have to add stars (rating) as well with the review. The rating is out 5 stars and it only has 5 options available 1 star, 2 stars, 3 stars, 4 stars, 5 stars. Now they want to predict ratings for the reviews which were written in the past and they don't have rating. So we, we have to build an application which can predict the rating by seeing the review.

Conceptual background of the domain problem:

Nowadays, a massive amount of reviews is available online. Besides offering a valuable source of information, these informational contents generated by users, also called User Generated Contents (UGC) strongly impact the purchase decision of customers. As a matter of fact, a recent survey (Hinckley, 2015) revealed that 67.7% of consumers are effectively influenced by online reviews when making their purchase decisions. More precisely, 54.7% recognized that these reviews were either fairly, very or absolutely important in their purchase decision making. Relying on online reviews has thus become a second nature for consumers

Review of the literature:

The rapid development of Web 2.0 and e-commerce has led to a proliferation in the number of online user reviews. Online reviews contain a wealth of sentiment information that is important for many decision-making processes, such as personal consumption decisions, commodity quality monitoring, and social 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.

Motivation for the problem undertaken:

Many product reviews are not accompanied by a scale rating system, consisting only of a textual evaluation. In this 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. Getting an overall sense of a textual review could in turn improve consumer experience.

Data Analysis

Data Set Description:

Data has been scrapped from FLIPKART e-commerce platform.

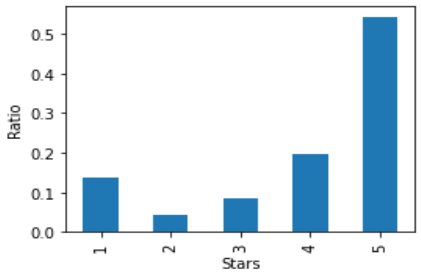
1) There are in total 30807 rows and 3 columns of ratings and reviews are in our dataset.
2) Ratings: It is the Label column, which includes ratings in the form of integers from 1 to 5
3) Full_Review: It contains text data on the basis of which we have to build a model to predict ratings.

Read the excel file and convert into data frame:

	Unnamed: 0	Ratings	Full_review
0	0	5	This is the best laptop in this range.I reciev...
1	1	5	Good product as used of now.... Everything is ...
2	2	5	AWESOME LAPTOP. It supports many high spec gam...
3	3	4	For that price... it's exceptionally good. Pla...
4	4	4	RAM upgrade is must do because the useable RAM...

4) We found the occurrence of ratings ratio as shown below:

```
plot_labels(Rating, "stars")
```



- The dataset is imbalanced.

```
print('Rating counts','\n',Rating.Ratings.value_counts())
```

```
Rating counts
 5    16683
4     6018
1     4185
3     2636
2     1285
Name: Ratings, dtype: int64
```

Observation:

- Maximum 27754 numbers of ratings present are of 5 star and minimum 2204 is of 2 star.
- We then create two more columns length and clean length on the basis of the lengths of the text before and after cleaning for our analysis purpose.

```
#convert text to lowercase
Rating['Full_review']=Rating['Full_review'].str.lower()
```

```
Rating['Full_review']=Rating['Full_review'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$', 'emailaddress')

Rating['Full_review']=Rating['Full_review'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$', 'webaddress')

Rating['Full_review']=Rating['Full_review'].str.replace(r'£|\$', 'dollers')

Rating['Full_review']=Rating['Full_review'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$', 'phonenumber')

Rating['Full_review']=Rating['Full_review'].str.replace(r'\d+(\.\d+)?', 'numbr')
```

```
#remove punctuation
Rating['Full_review']=Rating['Full_review'].str.replace(r'[^\w\d\s]', ' ')

#replace whitespace between terms with a single space
Rating['Full_review']=Rating['Full_review'].str.replace(r'\s+', ' ')

#Remove leading and trailing whitespace
Rating['Full_review']=Rating['Full_review'].str.replace(r'^\s+|\s+?$', '')
```

```
Rating.head()
```

	Ratings	Full_review	length
0	5	this is the best laptop in this range i reciev...	500
1	5	good product as used of now everything is good...	271
2	5	awesome laptop it supports many high spec game...	96
3	4	for that price it s exceptionally good played ...	342
4	4	ram upgrade is must do because the useable ram...	502

```
#Remove stopwords
import string
import nltk
from nltk.corpus import stopwords

stop_words = set(stopwords.words('english') + ['u', 'ur', '4', '2', 'im', 'dont', 'doin', 'ure'])

Rating['Full_review'] = Rating['Full_review'].apply(lambda x: ' '.join(term for term in x.split() if term not in stop_words))
```

```
Rating['clean_length'] = Rating.Full_review.str.len()
```

```
Rating.head()
```

	Ratings	Full_review	length	clean_length
0	5	best laptop range recieved late delivery due b...	500	337
1	5	good product used everything good also ssd slo...	271	150
2	5	awesome laptop supports many high spec games l...	96	84
3	4	price exceptionally good played far cry numbr ...	342	254
4	4	ram upgrade must useable ram numbrgb ryzen num...	502	393

```
print('original Review length', Rating.length.sum())
print('clean Review length', Rating.clean_length.sum())
```

```
original Review length 1916872
clean Review length 1363269
```

EDA Concluding Remark

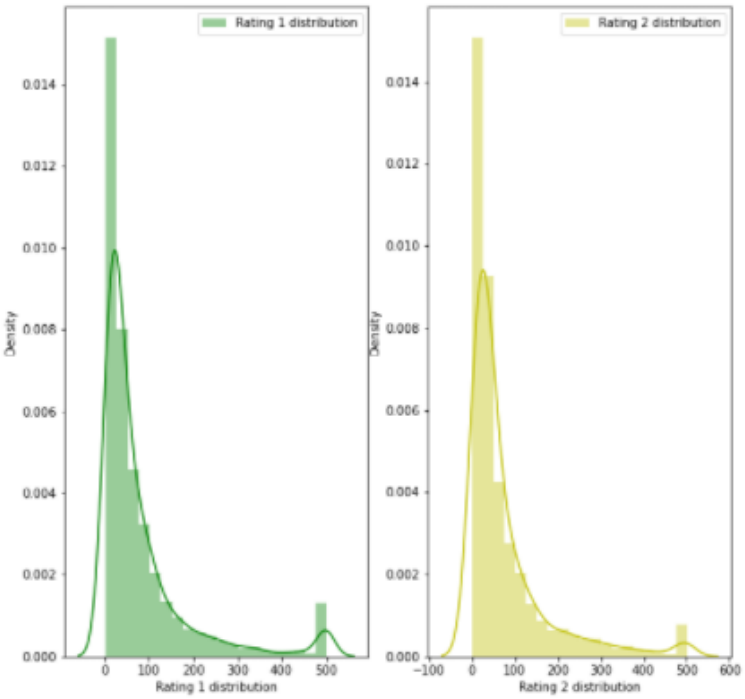
Rating 1 and Rating 2 distribution before cleaning the reviews:

```
f,ax = plt.subplots(1,2,figsize=(10,10))

sns.distplot(Rating[Rating['Ratings']==1]['length'],bins=20,ax=ax[0],label='Rating 1 distribution',color='g')
ax[0].set_xlabel('Rating 1 distribution')
ax[0].legend()

sns.distplot(Rating[Rating['Ratings']==2]['length'],bins=20,ax=ax[1],label='Rating 2 distribution',color='y')
ax[1].set_xlabel('Rating 2 distribution')
ax[1].legend()

plt.show()
```



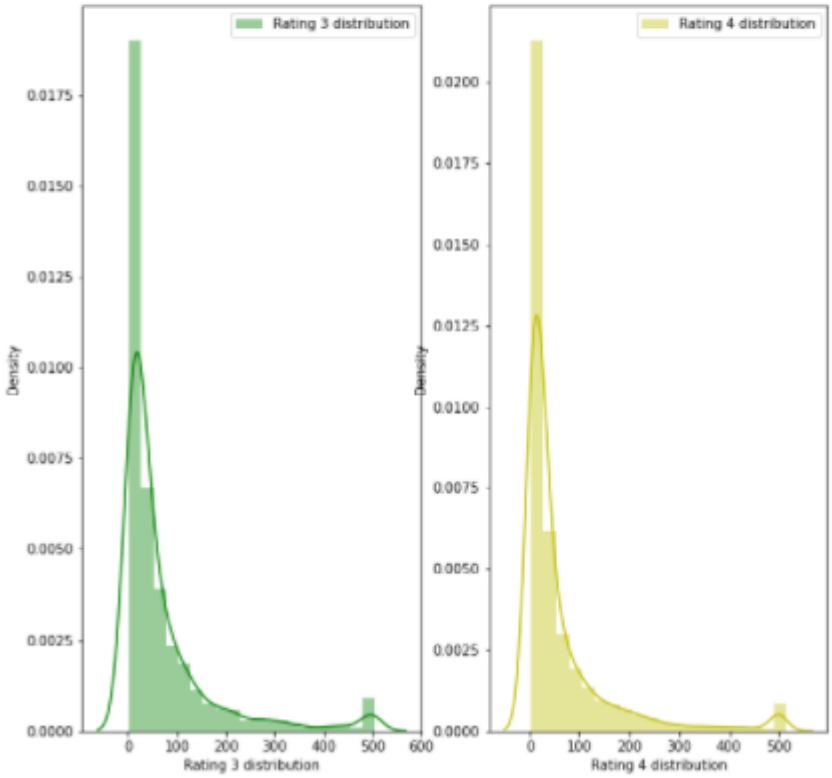
Rating 3 and Rating 4 distribution before cleaning the reviews:

```
f,ax = plt.subplots(1,2,figsize=(10,10))

sns.distplot(Rating[Rating['Ratings']==3]['length'],bins=20,ax=ax[0],label='Rating 3 distribution',color='g')
ax[0].set_xlabel('Rating 3 distribution')
ax[0].legend()

sns.distplot(Rating[Rating['Ratings']==4]['length'],bins=20,ax=ax[1],label='Rating 4 distribution',color='y')
ax[1].set_xlabel('Rating 4 distribution')
ax[1].legend()

plt.show()
```



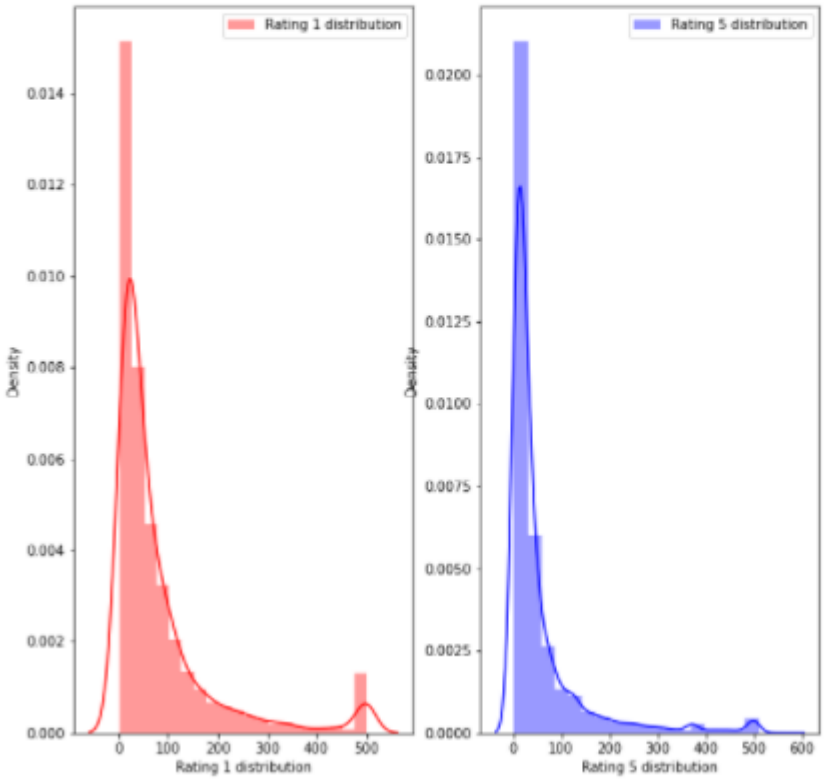
Rating 1 and Rating 5 distribution before cleaning reviews:

```
f,ax = plt.subplots(1,2,figsize=(10,10))

sns.distplot(Rating[Rating['Ratings']==1]['length'],bins=20,ax=ax[0],label='Rating 1 distribution',color='r')
ax[0].set_xlabel('Rating 1 distribution')
ax[0].legend()

sns.distplot(Rating[Rating['Ratings']==5]['length'],bins=20,ax=ax[1],label='Rating 5 distribution',color='b')
ax[1].set_xlabel('Rating 5 distribution')
ax[1].legend()

plt.show()
```



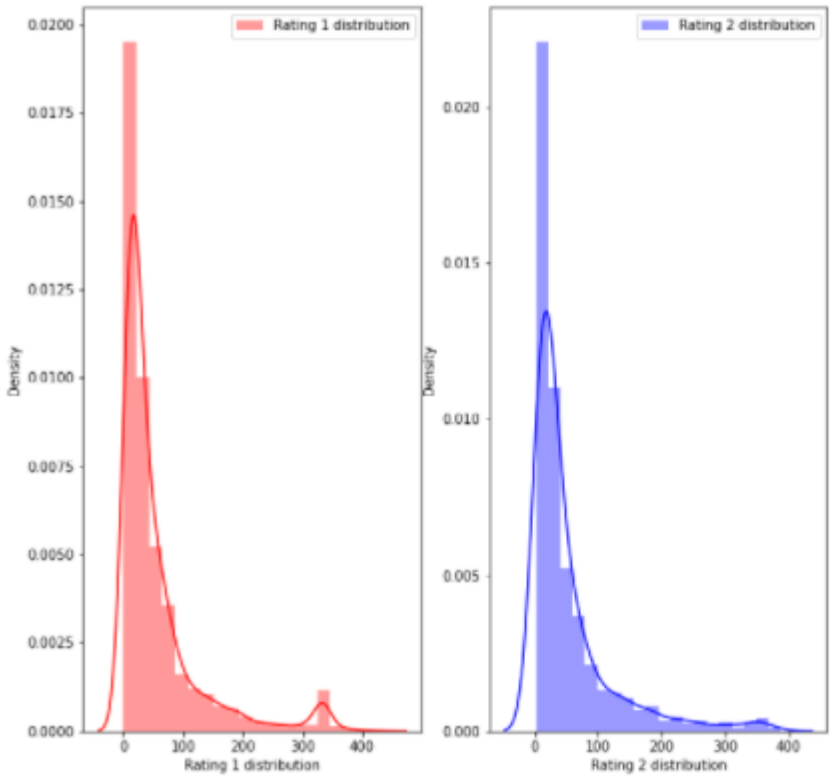
Rating 1 and Rating 2 distribution after cleaning the reviews:

```
f,ax = plt.subplots(1,2,figsize=(10,10))

sns.distplot(Rating[Rating['Ratings']==1]['clean_length'],bins=20,ax=ax[0],label='Rating 1 distribution',color='r')
ax[0].set_xlabel('Rating 1 distribution')
ax[0].legend()

sns.distplot(Rating[Rating['Ratings']==2]['clean_length'],bins=20,ax=ax[1],label='Rating 2 distribution',color='b')
ax[1].set_xlabel('Rating 2 distribution')
ax[1].legend()

plt.show()
```



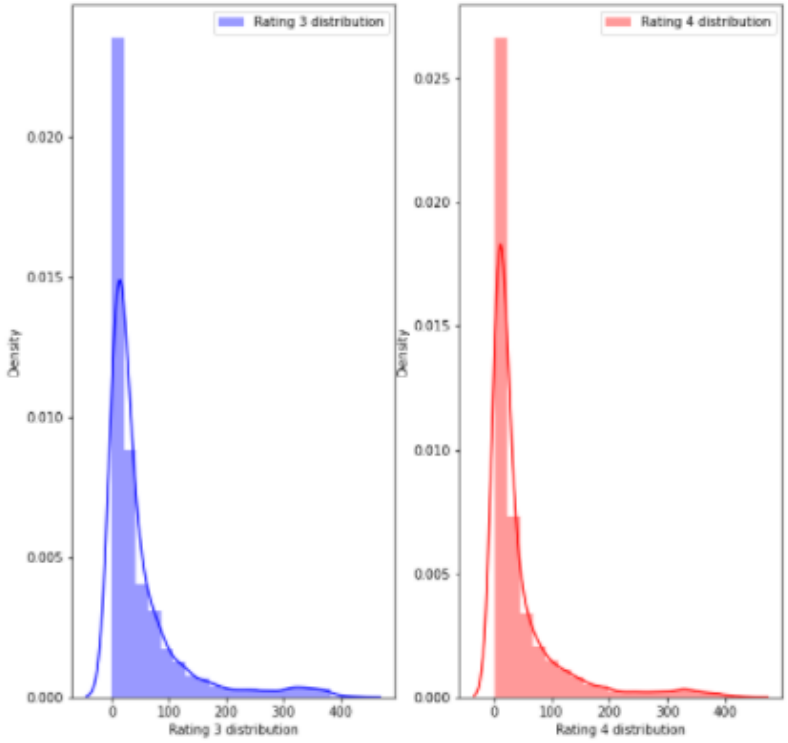
Rating 3 and Rating 4 distribution after cleaning the reviews:

```
f,ax = plt.subplots(1,2,figsize=(10,10))

sns.distplot(Rating[Rating['Ratings']==3]['clean_length'],bins=20,ax=ax[0],label='Rating 3 distribution',color='b')
ax[0].set_xlabel('Rating 3 distribution')
ax[0].legend()

sns.distplot(Rating[Rating['Ratings']==4]['clean_length'],bins=20,ax=ax[1],label='Rating 4 distribution',color='r')
ax[1].set_xlabel('Rating 4 distribution')
ax[1].legend()

plt.show()
```



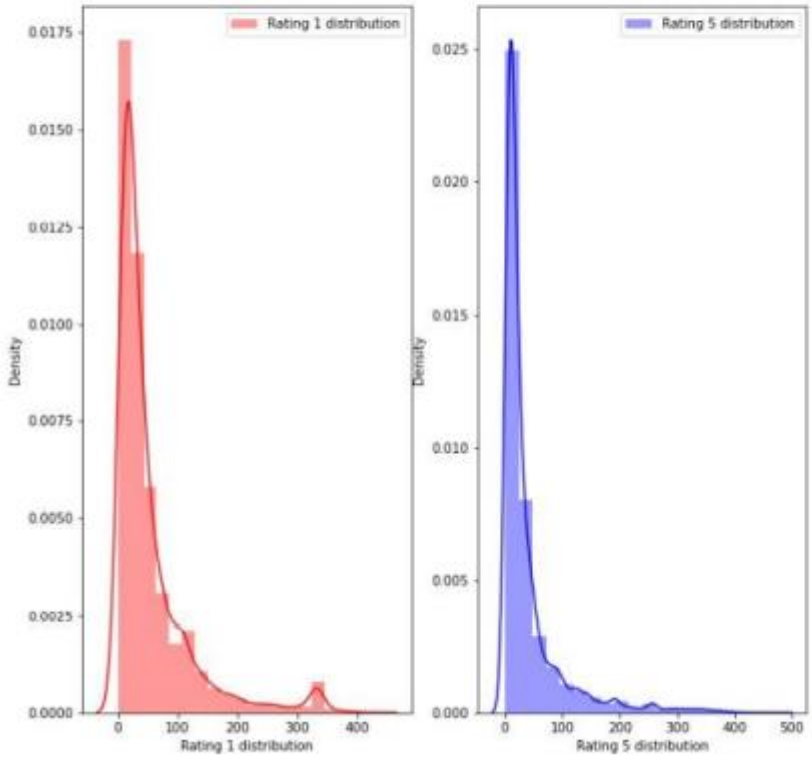
Rating 1 and Rating 5 distribution after cleaning the reviews:

```
f,ax = plt.subplots(1,2,figsize=(10,10))

sns.distplot(Rating[Rating['Ratings']==1]['clean_length'],bins=20,ax=ax[0],label='Rating 1 distribution',color='r')
ax[0].set_xlabel('Rating 1 distribution')
ax[0].legend()

sns.distplot(Rating[Rating['Ratings']==5]['clean_length'],bins=20,ax=ax[1],label='Rating 5 distribution',color='b')
ax[1].set_xlabel('Rating 5 distribution')
ax[1].legend()

plt.show()
```



Getting sense of review Loud words in Rating 1:

```
#getting sense of review loud words in Rating 1
from wordcloud import WordCloud

Rating1=Rating['Full_review'][Rating['Ratings']==1]

spam_cloud = WordCloud(width=700,height=500,background_color='white',max_words=20).generate(' '.join(Rating1))

plt.figure(figsize=(10,8),facecolor='r')
plt.imshow(spam_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```

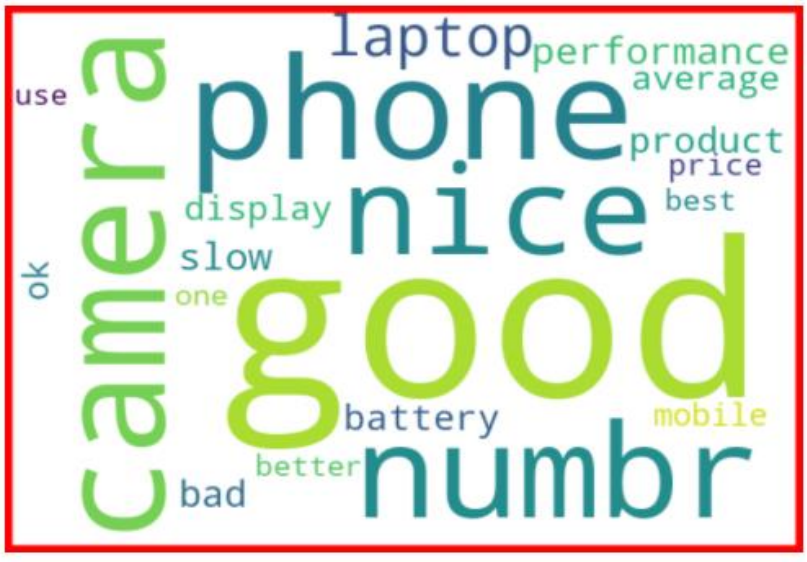
Getting sense of review Loud words in Rating 3:

```
#getting sense of review loud words in Rating 3

Rating3=Rating['Full_review'][Rating['Ratings']==3]

spam_cloud = WordCloud(width=700,height=500,background_color='white',max_words=20).generate(' '.join(Rating3))

plt.figure(figsize=(10,8),facecolor='r')
plt.imshow(spam_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```



Getting sense of review Loud words in Rating 2:

```
#getting sense of review Loud words in Rating 2

Rating2=Rating['Full_review'][Rating['Ratings']==2]

spam_cloud = WordCloud(width=700,height=500,background_color='white',max_words=20).generate(' '.join(Rating2))

plt.figure(figsize=(10,8),facecolor='r')
plt.imshow(spam_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```

Getting sense of review Loud words in Rating 4:

```
Rating4=Rating['Full_review'][Rating['Ratings']==4]

spam_cloud = WordCloud(width=700,height=500,background_color='white',max_words=20).generate(' '.join(Rating4))

plt.figure(figsize=(10,8),facecolor='r')
plt.imshow(spam_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```

Getting sense of review Loud words in Rating 5:

```
Rating5=Rating['full_review'][Rating['Ratings']==5]

spam_cloud = WordCloud(width=700,height=500,background_color='white',max_words=20).generate(' '.join(Rating5))

plt.figure(figsize=(10,8),facecolor='r')
plt.imshow(spam_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```

Identification of possible problem-solving approaches (methods)

After collecting the data, we need to build a machine learning model. Before model buildings we do all data pre-processing steps involving NLP. Try different models with different hyper parameters and select the best model.

a) Data Cleaning b) Exploratory Data Analysis c) Data Pre-processing d) Model Building e) Model Evaluation f) Selecting the best model.

Pre-Processing Pipeline

We first looked for the null values present in the dataset. We noticed that there were no null values present in our dataset. Then we performed text processing. Data usually comes from a variety of sources and often in different formats. For this reason, transforming your raw data is essential. However, this is not a simple process, as text data often contains redundant and repetitive words. This means that processing the text data is the first step in our solution. The fundamental steps involved in text pre-processing are, cleaning the raw data tokenizing the cleaned data.

Some of the steps are as follows: -

This phase involves the deletion of words or characters that do not add value to the meaning of the text. Some of the standard cleaning steps are listed below:

- Lowering case
- Removal of special characters
- Removal of stop words
- Removal of hyperlinks
- Removal of numbers
- Removal of whitespaces

Lowering Case:

Lowering the case of text is essential for the following reasons: The words, 'TEXT', 'Text', 'text' all add the same value to a sentence lowering the case of all the words is very helpful for reducing the dimensions by decreasing the size of the vocabulary.

Removal of special characters:

This is another text processing technique that will help to treat words like 'hurray' and 'hurray!' in the same way.

Removal of stop words:

Stop words are commonly occurring words in a language like 'the', 'a', and so on. Most of the time they can be removed from the text because they don't provide valuable information.

Set of assumptions related to the problem under consideration:

By looking into the target variable label we assumed that it was a Multiclass classification type of problem. We observed that dataset was imbalance so we will have to balance the dataset for better outcome.

DATA INPUTS- LOGIC- OUTPUT RELATIONSHIPS

For this data's input and output logic, we will analyse words frequency for each label, so that we can get the most frequent words that were used in different features.

Building Machine Learning Models

Pre-processing involved the following steps: -

- Removing Punctuations and other special characters
- Removing Stop Words
- Stemming and Lemmatising Applying
- tfidf Vectorizer
- Splitting dataset into Training and Testing

Testing of identified Approaches(Algorithms): -

- Decision tree classifier
- Kneighbors classifier
- MultinomialNB
- Random forest classifier
- Adaboost classifier
- Gradient boosting classifier
- Bagging classifier
- Extra trees classifier

RUN AND EVALUATE SELECTED MODELS

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
from collections import defaultdict, Counter
%matplotlib inline

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

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import MultinomialNB

#Importing Boosting models
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier

#Importing error metrics
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score,roc_curve,auc
from sklearn.model_selection import GridSearchCV,cross_val_score
```

```
result = pd.DataFrame({'Model': Model, 'Accuracy_score': score,'Cross_val_score': cvs})
result
```

	Model	Accuracy_score	Cross_val_score
0	KNeighborsClassifier	41.999351	53.114420
1	DecisionTreeClassifier	50.989938	58.161991
2	RandomForestClassifier	56.231743	63.118645
3	AdaBoostClassifier	49.058747	62.219620
4	MultinomialNB	49.740344	61.995489
5	GradientBoostingClassifier	50.860110	62.307114
6	BaggingClassifier	51.996105	60.463511
7	ExtraTreesClassifier	56.199286	63.076473

KEYMETRICS FOR SUCCESSIN SOLVING PROBLEMUNDE CONSIDERATION:

On the basis of accuracy and confusion matrix we save Random Forest classifier as our final model.

KEYMETRICS FORSUCCESSIN SOLVING PROBLEM UNDER CONSIDERATION:

- When it comes to the evaluation of a data science model's performance, sometimes accuracy may not be the best indicator.
- Some problems that we are solving in real life might have a very imbalanced class and using accuracy might not give us enough confidence to understand the algorithm's performance.
- In the Rating Prediction problem that we are trying to solve, the data is balanced. So accuracy score nearly tells the right predictions. So the problem of overfitting in this problem is nearly not to occur. So here, we are using an accuracy score to find a better model.

CONCLUSION

KEY FINDINGS AND CONCLUSIONS OF THE STUDY

In this project we have tried to detect the Ratings in commercial websites on a scale of 1 to 5 on the basis of the reviews given by the users. We made use of natural language processing and machine learning algorithms in order to do so. We interpreted that Random forest classifier model is giving us best results.

LEARNING OUTCOMES OF THE STUDY IN RESPECT OFDATA SCIENCE

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

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

Through different powerful tools of visualization, we were able to analyses and interpret different hidden insights about the data.

The few challenges while working on this project are: -

- Imbalanced Data set.
- Lots of text data.

The dataset was highly imbalanced so we balanced the dataset using smote technique. We converted text data into vectors with the help of tfidf vectorizer.

LIMITATIONS OF THIS WORK AND SCOPE FOR FUTURE WORK

While we couldn't reach out goal of maximum accuracy in 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. As with any project there is room for improvement here. 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. However, the output of these algorithms needs to be in the same format as the others. Once that condition is satisfied, the modules are easy to add as done in the code. This provides a great degree of modularity and versatility to the project.