RATINGS PREDICTION PROJECT

Submitted By

Girija Chandra Mohan

ACKNOWLEDGMENT

I would like to express my deepest gratitude to my SME (Subject Matter Expert) Khushboo Garg as well as Flip Robo Technologies who gave me the opportunity to do this project on Customer Ratings Prediction, which also helped me in doing lots of research wherein I came to know about so many new things especially the Natural Language Processing and Natural Language Toolkit parts. Also, I have utilized a few external resources that helped me to complete this project. I ensured that I learn from the samples and modify things according to my project requirement.

All the external resources that were used in creating this project are listed below:

1) https://www.google.com/

2)https://www.youtube.com/

3)https://scikit-learn.org/stable/user_guide.html

4)https://github.com/

5)https://www.kaggle.com/

6) https://medium.com/

7)https://towardsdatascience.com/

8)https://www.analyticsvidhya.com/

9)https://www.nltk.org/api/nltk.html

10) https://www.nltk.org/data.html

INTRODUCTION

BUSINESS PROBLEM FRAMING

This is a Machine Learning Project performed on customer reviews. Reviews are processed using common NLP techniques.

Millions of people use Amazon and Flipkart to buy products. For every product, people can rate and write a review. If a product is good, it gets a positive review and gets a higher star rating, similarly, if a product is bad, it gets a negative review and lower star rating. My aim in this project is to predict star rating automatically based on the product review.

The range of star rating is 1 to 5. That means if the product review is negative, then it will get low star rating (possibly 1 or 2), if the product is average then it will get medium star rating (possibly 3), and if the product is good, then it will get higher star rating (possibly 4 or 5).

This task is similar to Sentiment Analysis, but instead of predicting the positive and negative sentiment (sometimes neutral also), here I need to predict the star rating.

AIM OF THIS PROJECT

Our goal is to make a system that automatically detects the star rating based on the review.

CONCEPTUAL BACKGROUND OF THE DOMAIN PROBLEM

- The advent of electronic commerce with growth in internet and network technologies has led customers to move to online retail platforms such as Amazon, Walmart, Flip Kart, etc. People often rely on customer reviews of products before they buy online. These reviews are often rich in information describing the product. Customers often choose to compare between various products and brands based on whether an item has a positive or negative review. More often, these reviews act as a feedback mechanism for the seller. Through this medium, sellers strategize their future sales and product improvement.

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

REVIEW OF LITERATURE

- This project is more about exploration, feature engineering and classification that can be done on this data. Since we scrape huge amount of data that includes five stars rating, we can do better data exploration and derive some interesting features using the available columns.

- We can categorize the ratings as:

    1.0, 2.0, 3.0, 4.0 and 5.0 stars

•The goal 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 reviewing their purchase with each other in this increasingly digital world.

MOTIVATION OF THE PROBLEM UNDERTAKEN

- Every day we come across various products in our lives, on the digital medium we swipe across hundreds of product choices under one category. It will be tedious for the customer to make selection. Here comes 'reviews' where customers who have already got that product leave a rating after using them and brief their experience by giving reviews.
- As we know ratings can be easily sorted and judged whether a product is good or bad. But when it comes to sentence reviews, we need to read through every line to make sure the review conveys a positive or negative sense. In the era of artificial intelligence, things like that have got easy with the Natural Language Processing (NLP) technology. Therefore, it is important to minimize the number of false positives our model produces, to encourage all constructive conversation.
- Our model also provides benefit for the platform hosts as it replaces the need to manually moderate discussions, saving time and resources. Employing a machine learning model to predict ratings promotes easier way to distinguish between products qualities, costs and many other features.

ANALYTICAL PROBLEM FRAMING

MATHEMATICAL/ANALYTICAL MODELLING OF THE PROBLEM

In our scrapped dataset, our target variable "Rating " is a categorical variable i.e., it can be classified as ‘1.0’, ‘2.0’,’3.0’,’4.0’,’5.0’. Therefore, we will be handling this modelling problem as classification.

This project is done in two parts:

- Data Collection Phase
- 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 like

1. Data Cleaning
2. Exploratory Data Analysis
3. Data Pre-processing
4. Model Building
5. Model Evaluation
6. Selecting the best mode

DATA SOURCES AND THEIR FORMATS

- 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.

- We scrapped nearly 25000 of the reviews data and saved each data in a separate data frame

- In the end, we combined all the data frames into a single data frame and it looks like as follows:

```
# to combine flipkart and amazon reviews files together in a single csv file

flipkart = pd.read_csv(r"C:\python\Ratings_Flipkart.csv")
amazon = pd.read_csv(r"C:\python\Ratings_amazon.csv")
reviews = pd.concat([amazon, flipkart], ignore_index=True)
reviews
```

	Unnamed: 0	Title	Review_text	Ratings
0	0	Open box product delivered	Product seal was open / broken and original ad...	3
1	1	Awesome features and it's "made in India" budg...	It's a "made in India" laptop with great featu...	5
2	2	Open box product delivered	Product seal was open / broken and original ad...	3
3	3	Awesome features and it's "made in India" budg...	It's a "made in India" laptop with great featu...	5
4	4	Open box product delivered	Product seal was open / broken and original ad...	3
...	...	...	...	...

- Then, we will save this data in a csv file, so that we can do the pre-processing and model building.

- Handling the missing values with Na and removing it.

```
df.isna().sum()
```

```
Title           0
Review_text    184
Ratings          0
dtype: int64
```

Using the isna() and sum() options together we can confirm that there some missing values in Review_text column that needs to be treated.

```
df.dropna(inplace=True)
print("We have {} Rows and {} Columns in our dataframe after removing NaN".format(df.shape[0], df.shape
```

```
We have 24476 Rows and 3 Columns in our dataframe after removing NaN
```

- Checking the info of the dataset and combining review text and Title into single column to perform NLP processing.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 24476 entries, 0 to 24659
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Title        24476 non-null  object
 1   Review_text  24476 non-null  object
 2   Ratings      24476 non-null  int64 
dtypes: int64(1), object(2)
memory usage: 764.9+ KB
```

```
# Now combining the "Review_title" and "Review_text" columns into one single column called "Review"
df['Review'] = df['Title'].map(str)+' '+df['Review_text']
df
```

	Title	Review_text	Ratings	Review
0	Open box product delivered	Product seal was open / broken and original ad...	3	Open box product delivered Product seal was op...
1	Awesome features and it's "made in India" budg...	It's a "made in India" laptop with great featu...	5	Awesome features and it's "made in India" budg...
2	Open box product delivered	Product seal was open / broken and original ad...	3	Open box product delivered Product seal was op...
3	Awesome features and it's "made in India" budg...	It's a "made in India" laptop with great featu...	5	Awesome features and it's "made in India" budg...
4	Open box product delivered	Product seal was open / broken and original ad...	3	Open box product delivered Product seal was op...
...	...	...	...	...

Pre-processing using Natural Language Processing (NLP):

- We cleaned the data using regex, matching patterns in the comments and replacing them with more organized counterparts. Cleaner data leads to a more efficient model and higher accuracy.
- Following steps are involved:

    1. Removing Punctuations and other special characters
    2. Splitting the comments into individual words
    3. Removing Stop Words

- There is a corpus of stop-words, that are high-frequency words such as "the", "to" and "also", and that we sometimes want to litter out of a document before further processing. Stop-words usually have little lexical content, don't alter the general meaning of a sentence and their presence in a text fails to distinguish it from other texts. We used the one from Natural Language Toolkit a leading platform for building Python programs to work with human language.
- The code is attached below:

```
def cleaning(df, df_column_name):

    #Converting  messages to lowercase
    df[df_column_name] = df[df_column_name].str.lower()

    #Replace email addresses with 'email'
    df[df_column_name] = df[df_column_name].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$','emailaddress')

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

    #Replace money symbols with 'dollars' (£ ---> ALT key + 156)
    df[df_column_name] = df[df_column_name].str.replace(r'£|\$', 'dollars')

    #Replace 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phonenumber'
    df[df_column_name] = df[df_column_name].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$','phonenumber')

    #Replace numbers with 'numbr'
    df[df_column_name] = df[df_column_name].str.replace(r'\d+(\.\d+)?', 'numbr')

    #Remove punctuation
    df[df_column_name] = df[df_column_name].str.replace(r'[^\w\d\s]', ' ')

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

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

    #Remove stopwords
    stop_words = set(stopwords.words('english') + ['u', 'ü', 'â', 'ur', '4', '2', 'im', 'dont', 'doin', 'ure'])
    df[df_column_name] = df[df_column_name].apply(lambda x: ' '.join(term for term in x.split() if term not in stop_words))
```

- Tokenizing the data using RegexpTokenizer

```
cleaning(df, 'Review')
df['Review'].tail(3)
```

```
24657                          perfect product nice product
24658     super mondol nice also sumthnice like hd monitor
24659    brilliant perfect picture quality well working...
Name: Review, dtype: object
```

```
df['Review'].head(3)
```

```
0    open box product delivered product seal open b...
1    awesome features made india budget laptop made...
2    open box product delivered product seal open b...
Name: Review, dtype: object
```

Tokenizing the data using RegexpTokenizer

```
from nltk.tokenize import RegexpTokenizer

tokenizer=RegexpTokenizer(r'\w+')
df['Review'] = df['Review'].apply(lambda x: tokenizer.tokenize(x.lower()))
df.head()
```

	Ratings	Review
0	3	[open, box, product, delivered, product, seal,...
1	5	[awesome, features, made, india, budget, lapto...
2	3	[open, box, product, delivered, product, seal,...
3	5	[awesome, features, made, india, budget, lapto...

Stemming and Lemmatizing:

• 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 neighbouring sentences or even an entire document.

• The wordnet library in nltk will be used for this purpose. Stemmer and Lemmatizer are also imported from nltk.

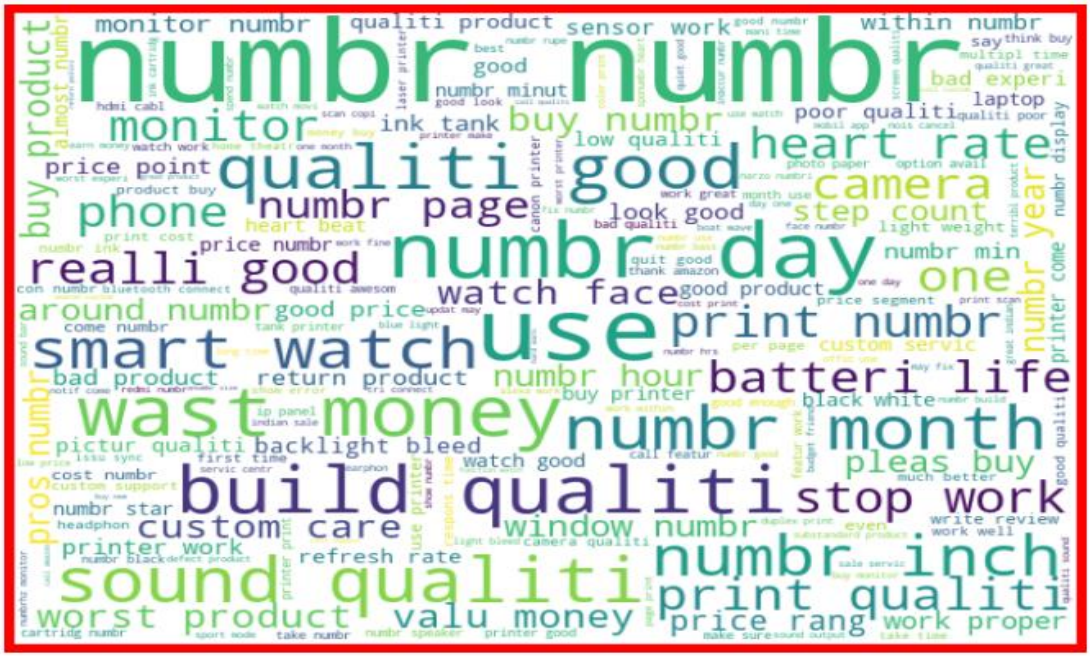
```
stemmer = SnowballStemmer("english")
import gensim
def lemmatize_stemming(text):
    return stemmer.stem(WordNetLemmatizer().lemmatize(text,pos='v'))

#Tokenize and Lemmatize
def preprocess(text):
    result=[]
    for i in text:
        if len(i)>=3:
            result.append(lemmatize_stemming(i))

    return result
```

```
#Processing review with above Function
processed_review = []

for i in df.Review:
    processed_review.append(preprocess(i))

print(len(processed_review))
processed_review[:5]
```

```
24476

[['open',
  'box',
  'product',
```

Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance.

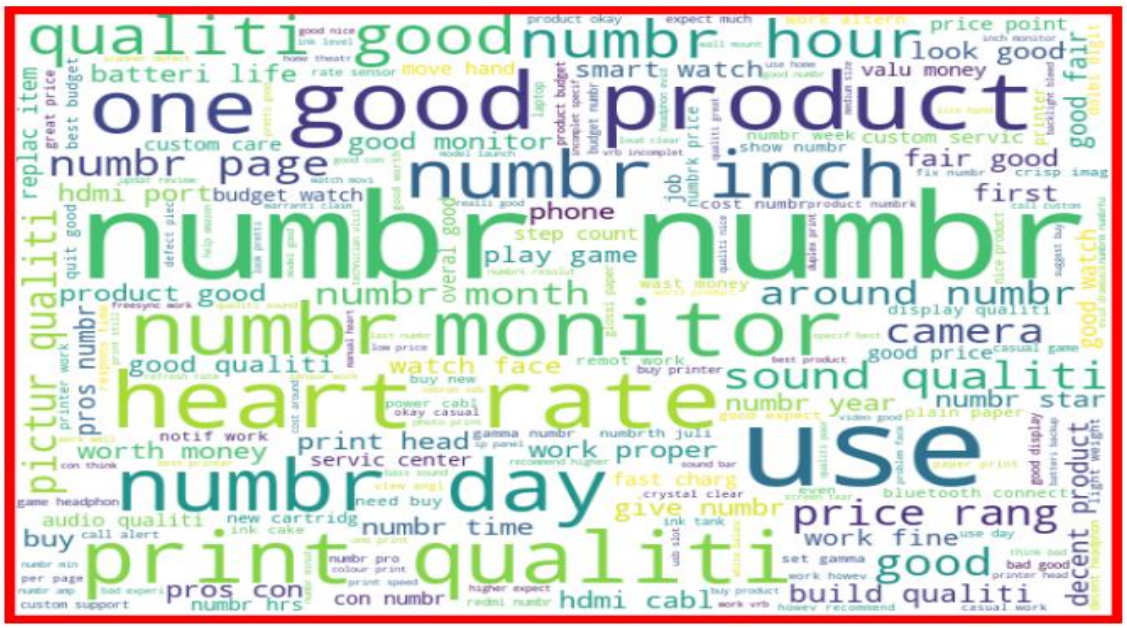
```
one = df['Review'][df['Ratings']==1]

one_cloud = WordCloud(width=700,height=500,background_color='white',max_words=200).generate(' '.join(one))

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



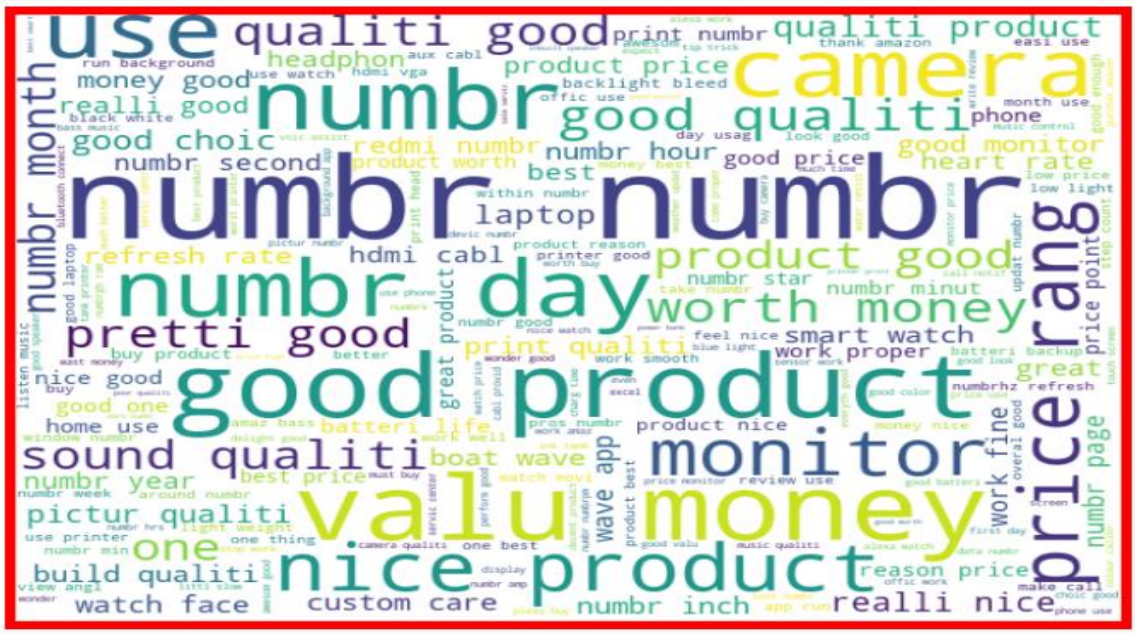
Similarly, we found the sense of words for ratings 2 – 5.

For rating 2:

For Rating 3:



For Rating 4:



For Rating 5:

Observations:

-> The enlarged texts are the most number of words used there and small texts are the less number of words used.

-> It varies according to the ratings. Feature Extraction: Here we can finally convert our text to numeric using Tf-idf Vectorizer.

Term Frequency Inverse Document Frequency (TF-IDF):

This is very common algorithm to transform text into a meaningful representation of numbers which is used to fit machine algorithm for prediction.

```
tf = TfidfVectorizer()
```

```
features = tf.fit_transform(df['Review'])
x=features
```

```
y=df['Ratings']
```

```
x.shape
```

```
(24476, 6891)
```

```
y.shape
```

```
(24476,)
```

HARDWARE AND SOFTWARE REQUIREMENTS AND TOOLS USED

- For doing this project, the hardware used is a laptop with high end specification and a stable internet connection. While coming to software part, I had used anaconda navigator and in that I have used Jupyter notebook to do my python programming and analysis.
- For using an CSV file, Microsoft excel is needed. In Jupyter notebook, I had used lots of python libraries to carry out this project and I have mentioned below with proper justification:

Libraries Used:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

# Importing nltk libraries
import re
import string
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords', quiet=True)
nltk.download('punkt', quiet=True)
nltk.download('wordnet', quiet=True)
nltk.download('averaged_perceptron_tagger', quiet=True)
from nltk import FreqDist
from nltk.tokenize import word_tokenize
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import wordnet
from nltk import FreqDist

from scipy import stats
from scipy.stats import zscore
from scipy.sparse import hstack
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.metrics import f1_score, precision_score, confusion_matrix, accuracy_score, classification_report

from sklearn.svm import LinearSVC
from sklearn.naive_bayes import MultinomialNB, GaussianNB, BernoulliNB
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDClassifier
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
from sklearn.tree import DecisionTreeClassifier

import warnings
warnings.simplefilter("ignore")
warnings.filterwarnings("ignore")
import joblib

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

MODEL/S DEVELOPMENT AND EVALUATION

Various models used for training are:

- MultinomialNB()
- DecisionTreeclassifier()
- RandomForestClassifier()
- KNN()
- SVC()

Run and Evaluate selected models

- We have used a total of 5 Regression Models after choosing the random state amongst 1-100 number. Then it is checked for best fold for CV using the following codes.

```
maxAcc=0
maxRS=0

for i in range(1,100):
    xtrain,xtest,ytrain,ytest= train_test_split(x,y,test_size=20,random_state=i)
    mnb=MultinomialNB()
    mnb.fit(xtrain,ytrain)
    predmnb= mnb.predict(xtest)
    acc= accuracy_score(ytest,predmnb)
    if acc>maxAcc:
        maxAcc=acc
        maxRS=i
print("Best Accuracy_score is", maxAcc, 'on Random_state',maxRS)
```

```
Best Accuracy_score is 0.9 on Random_state 29
```

```
pred_tr= mnb.predict(xtrain)
pred_ts= mnb.predict(xtest)
```

```
train_accuracy= accuracy_score(ytrain,pred_tr)
test_accuracy= accuracy_score(ytest,pred_ts)
```

```
from sklearn.model_selection import cross_val_score
for j in range(2,10):
    cv_score= cross_val_score(lr,x,y,cv=j)
    cv_mean=cv_score.mean()
    print(f"At cross fold {j} the cv score is {cv_mean} and accuracy score for training is {train_accuracy} and the accuracy for
    print("\n")
```

```
At cross fold 2 the cv score is 0.41085144631475734 and accuracy score for training is 0.6593065096499836 and the accuracy for
testing is 0.75


At cross fold 3 the cv score is 0.42642456257473144 and accuracy score for training is 0.6593065096499836 and the accuracy for
testing is 0.75


At cross fold 4 the cv score is 0.4147327994770387 and accuracy score for training is 0.6593065096499836 and the accuracy for t
esting is 0.75
```

```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=20, random_state=29)
```

MultinomialNB

```
#MultinomialNB
mnb=MultinomialNB()
mnb.fit(xtrain,ytrain)
predmnb= mnb.predict(xtest)
print('acc=', accuracy_score(ytest,predmnb))
print(confusion_matrix(ytest,predmnb))
print(classification_report(ytest,predmnb))
```

```
scr1=cross_val_score(mnb, x,y, cv=9)
print('Cross Validation Score of MultinomialNB is:', scr1.mean())
```

```
Cross Validation Score of MultinomialNB is: 0.3971513249777047
```

```
diff= 0.9- 0.422
diff
```

```
0.47800000000000004
```

DecisionTreeClassifier

```
#DecisionTreeClassifier
dtc=DecisionTreeClassifier()
dtc.fit(xtrain,ytrain)
preddtc= dtc.predict(xtest)
print('acc=', accuracy_score(ytest,preddtc))
print(confusion_matrix(ytest,preddtc))
print(classification_report(ytest,preddtc))
```

```
scr3=cross_val_score(dtc, x,y, cv=9)
print('Cross Validation Score of DecisionTreeClassifier is:', scr3.mean())
```

```
Cross Validation Score of DecisionTreeClassifier is: 0.464156687868233
```

```
diff=0.95-0.4614
diff
```

```
0.4886
```

RandomForestClassifier

```
#RandomForestClassifier
rf=RandomForestClassifier()
rf.fit(xtrain,ytrain)
predrf= rf.predict(xtest)
print('acc=', accuracy_score(ytest,predrf))
print(confusion_matrix(ytest,predrf))
print(classification_report(ytest,predrf))
```

```
scr=cross_val_score(rf, x,y, cv=9)
print('Cross Validation Score of RandomForestClassifier is:', scr.mean())
```

```
Cross Validation Score of RandomForestClassifier is: 0.48311376070594964
```

```
diff= 1.0-0.4831
diff
```

```
0.5169
```

KNN

```
#KNN
knn=KNeighborsClassifier()
knn.fit(xtrain,ytrain)
predknn= knn.predict(xtest)
print('acc=', accuracy_score(ytest,predknn))
print(confusion_matrix(ytest,predknn))
print(classification_report(ytest,predknn))
```

```
acc= 0.95
[[9 0 0 0]
 [0 1 0 0]
 [0 0 2 0]
 [0 0 1 7]]
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         9
           3       1.00      1.00      1.00         1
           4       0.67      1.00      0.80         2
           5       1.00      0.88      0.93         8

    accuracy                           0.95        20
   macro avg       0.92      0.97      0.93        20
weighted avg       0.97      0.95      0.95        20
```

```
scr2=cross_val_score(knn, x,y, cv=9)
print('Cross Validation Score of KNeighborsClassifier is:', scr2.mean())
```

```
Cross Validation Score of KNeighborsClassifier is: 0.3215309027251945
```

```
difff=0.95-0.321
diff
```

```
0.5169
```

SVC

```
#SVC
svc=SVC()
svc.fit(xtrain,ytrain)
predsvc= svc.predict(xtest)
print('acc=', accuracy_score(ytest,predsvc))
print(confusion_matrix(ytest,predsvc))
print(classification_report(ytest,predsvc))
```

```
acc= 1.0
[[9 0 0 0]
 [0 1 0 0]
 [0 0 2 0]
 [0 0 0 8]]
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         9
           3       1.00      1.00      1.00         1
           4       1.00      1.00      1.00         2
           5       1.00      1.00      1.00         8

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20
```

```
scr5=cross_val_score(svc, x,y, cv=9)
print('Cross Validation Score of KNeighborsClassifier is:', scr5.mean())
```

```
Cross Validation Score of KNeighborsClassifier is: 0.4570515640361821
```

```
diff=1.0-0.457
diff
```

```
0.5429999999999999
```

Key Metrics for success in solving problem under consideration

The key metrics used here were accuracy_score, cross_val_score, classification report, and confusion matrix. We tried to find out the best parameters and also to increase our scores by using Hyperparameter Tuning and we will be using GridSearchCV method.

1. **Cross Validation**:

   Cross-validation helps to find out the over fitting and under fitting of the model. In the cross validation the model is made to run on

different subsets of the dataset which will get multiple measures of the model. If we take 5 folds, the data will be divided into 5 pieces where each part being 20% of full dataset. While running the Cross-validation the 1st part (20%) of the 5 parts will be kept out as a holdout set for validation and everything else is used for training data. This way we will get the first estimate of the model quality of the dataset. In the similar way further iterations are made for the second 20% of the dataset is held as a holdout set and remaining 4 parts are used for training data during process. This way we will get the second estimate of the model quality of the dataset. These steps are repeated during the cross-validation process to get the remaining estimate of the model quality.

2. **Confusion Matrix**:

   A confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each row of the matrix represents the instances in a predicted class, while each column represents the instances in an actual class (or vice versa). The name stems from the fact that it makes it easy to see whether the system is confusing two classes (i.e., commonly mislabelling one as another). It is a special kind of contingency table, with two dimensions ("actual" and "predicted"), and identical sets of "classes" in both dimensions (each combination of dimension and class is a variable in the contingency table).

3. **Classification Report**:

The classification report visualizer displays the precision, recall, F1, and support scores for the model. There are four ways to check if the predictions are right or wrong:

1. TN / True Negative: the case was negative and predicted negative

2. TP / True Positive: the case was positive and predicted positive
3. FN / False Negative: the case was positive but predicted negative
4. FP / False Positive: the case was negative but predicted positive

Precision:

Precision is the ability of a classifier not to label an instance positive that is actually negative. For each class, it is defined as the ratio of true positives to the sum of a true positive and false positive. It is the accuracy of positive predictions. The formula of precision is given below:

$$\text{Precision} = TP/ (TP + FP)$$

Recall:

Recall is the ability of a classifier to find all positive instances. For each class it is defined as the ratio of true positives to the sum of true positives and false negatives. It is also the fraction of positives that were correctly identified. The formula of recall is given below:

$$\text{Recall} = TP/(TP+FN)$$

F1 score:

The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. F1 scores are

lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy. The formula is:

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

Support:

Support is the number of actual occurrences of the class in the specified dataset. Imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or rebalancing. Support doesn't change between models but instead diagnoses the evaluation process.

4. **Hyperparameter Tuning:**

There is a list of different machine learning models. They all are different in some way or the other, but what makes them different is nothing but input parameters for the model. These input parameters are named as Hyperparameters. These hyperparameters will define the architecture of the model, and the best part about these is that you get a choice to select these for your model. You must select from a specific list of hyperparameters for a given model as it varies from model to model.

We are not aware of optimal values for hyperparameters which would generate the best model output. So, what we tell the model is to explore and select the optimal model architecture automatically. This selection procedure for hyperparameter is known as Hyperparameter Tuning. We can do tuning by using GridSearchCV.

GridSearchCV is a function that comes in Scikit-learn (or SK-learn) model selection package. An important point here to note is that we need to have Scikit-learn library installed on the computer. This function helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyperparameters.

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

```
DecisionTreeClassifier()
```

```
DecisionTreeClassifier()
```

```
parameters={'criterion': ["gini", "entropy"],
            'splitter': ["best", "random"],
            'max_depth':[1,2,3],
            'min_samples_split': [2,3,4,5],
            'max_features':["auto", "sqrt", "log2"]}
```

```
GCV=GridSearchCV(DecisionTreeClassifier(), parameters, cv=9, scoring='accuracy')
GCV.fit(xtrain,ytrain)
GCV.best_params_
```

```
{'criterion': 'gini',
 'max_depth': 3,
 'max_features': 'auto',
 'min_samples_split': 4,
 'splitter': 'best'}
```

```
GCV.best_estimator_
```

```
DecisionTreeClassifier(max_depth=3, max_features='auto', min_samples_split=4)
```

```
GCV_pred=GCV.best_estimator_.predict(xtest)
accuracy_score(ytest,GCV_pred)
```

```
0.35
```

```
dtc=DecisionTreeClassifier(criterion= 'entropy',
                           max_features= 'sqrt',
                           splitter= 'best')
dtc.fit(xtrain,ytrain)
preddtc= dtc.predict(xtest)
print('acc=', accuracy_score(ytest,preddtc))
print(confusion_matrix(ytest,preddtc))
print(classification_report(ytest,preddtc))
```

```
acc= 0.95
[[9 0 0 0]
 [0 1 0 0]
 [0 0 2 0]
 [0 0 1 7]]
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         9
           3       1.00      1.00      1.00         1
           4       0.67      1.00      0.80         2
           5       1.00      0.88      0.93         8
```

After applying Hyperparameter Tuning, we can see that DecisionTreeClassifier Algorithm is performing well as the scores are improved, i.e., accuracy score from 90 to 95 Now, we will finalize DecisionTreeClassifier algorithm model as the final model.

FINAL MODEL

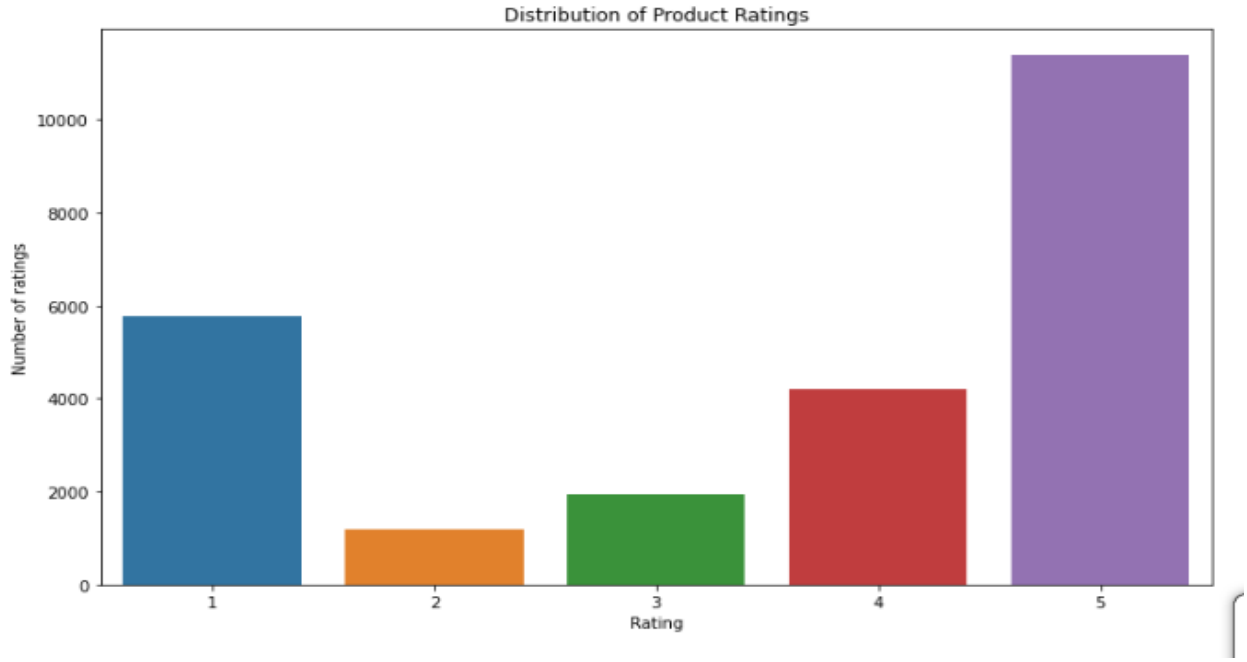
```
import pickle
filename= 'Ratings_Review.pkl'
pickle.dump(dtc,open(filename, 'wb'))
```

```
x=np.array(ytest)
predicted= np.array(dtc.predict(xtest))
df_con= pd.DataFrame({'original': x, 'Predicted': predicted}, index= range(len(x)))
df_con
```

	original	Predicted
0	1	1
1	5	5
2	1	1
3	3	3
4	5	5
5	1	1
6	1	1
7	1	1
8	5	4
9	1	1
10	1	1
11	5	5
12	5	5
13	5	5
14	1	1

DATA VISUALIZATION

```
f, axes = plt.subplots(figsize=(12,7))
ax = sns.countplot(x=df['Ratings'])
ax.set(title="Distribution of Product Ratings", xlabel="Rating", ylabel="Number of ratings")
plt.show()
```



CONCLUSION

Key Findings and Conclusions of the Study

-> After the completion of this project, we got an insight of how to collect data, pre-processing the data, analysing the data and building a model.

-> First, we collected the reviews and ratings data from different ecommerce websites like Amazon and Flipkart and it was done by using Webscraping. The framework used for webscraping was Selenium, which has an advantage of automating our process of collecting data.

-> We collected almost 25000 of data which contained the ratings from 1.0 to 5.0 and their reviews.

-> Then, the scrapped data was combined in a single dataframe and saved in a csv file so that we can open it and analyze the data.

-> We did the pre-processing using NLP and the steps are as follows:

Removing Punctuations and other special characters

- Splitting the comments into individual words
- Removing Stop Words
- Stemming and Lemmatising
- Applying Count Vectorizer
- Splitting dataset into Training and Testing

-> After separating our train and test data, we started running different machine learning classification algorithms to find out the best performing model.

-> We found that RandomForest and DecisionTree Algorithms were performing well, according to their accuracy and cross val scores.

-> Then, we performed Hyperparameter Tuning techniques using GridSearchCV for getting the best parameters and improving the scores. In that, DecisionTreeClassifier performed well and we finalised that model.

-> We saved the model in pkl format and then saved the predicted values in a csv format.

-> The problems we faced during this project were:

- More time consumption during hyperparameter tuning for both models, as the data was large.
- Less number of parameters were used during tuning.

- Scrapping of data from different websites were of different process and the length of data were differing in most cases.
- Some of the reviews were bad and the text had more wrong information about the product.
- WordCloud was not showing proper text which had more positive and negative weightage.

->Areas of improvement:

- Less time complexity
- More computational power can be given
- Less errors can be avoided.