FLIP ROBO

Ratings Prediction

Submitted by:

Sonu Rajput

ACKNOWLEDGMENT

I would like to thank my mentors at Data Trained, who taught me the concepts of Data Analysis, building a machine learning model, and tuning the parameters for best outcomes.

For this particular task, I referred the following websites and articles when stuck:

- https://towardsdatascience.com/a-common-mistake-to-avoidwhen-encoding-ordinal-features-79e402796ab4
- https://stackoverflow.com/questions/43590489/gridsearchcvrandom-forest-regressor-tuning-best-params
- https://www.codegrepper.com/codeexamples/delphi/scikit+pca+preserve+column+names+pca+pipeline
- https://stackoverflow.com/questions/22984335/recoveringfeatures-names-of-explained-variance-ratio-in-pca-with-sklearn

I would also like to thank my mentor in Fliprobo, Mr. Shubham Yadav, for providing me with the dataset and problem statement for performing this wonderful task.

INTRODUCTION

Business Problem Framing

Need to predict the ratings (1-5) of various products based on the reviews written by customers based on data scrapped from e-commerce sites.

Conceptual Background of the Domain Problem

Ratings Prediction

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

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 theater, router from different e-commerce websites. Basically, we need these columns-

1) reviews of the product. 2) rating of the product. You can fetch other data as well, if you think data can be useful or can help in the project. It completely depends on your imagination or assumption. Hint: – Try fetching data from different websites. If data is from different websites, it will help our model to remove the effect of over fitting.

- Try to 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 preprocessing 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 Preprocessing
4. Model Building
5. Model Evaluation
6. Selecting the best model

Part 1 - Extraction

I have used Flipkart website to extract reviews from ['laptops', 'Phones', 'Headphones', 'smart watches', 'Professional Cameras', 'Printers', 'monitors', 'Home theater', 'router']

I used the following script for data scraping. First I have imported all the required libraries.

```
# Importing Libraries
import selenium
import pandas as pd
import time
from bs4 import BeautifulSoup

# Importing selenium webdriver
from selenium import webdriver

# Importing required Exceptions which needs to handled
from selenium.common.exceptions import StaleElementReferenceException, NoSuchElementException

#Importing requests
import requests

# importing regex
import re

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

```
Review1=[]
Rating1= []

review1=driver.find_elements_by_xpath('//div[@class="t-ZTKy"]/div/div')
for i in review1:

    Review1.append(i.text)

rating1=driver.find_elements_by_xpath('//div[@class="_3LWZlK _1BLPMq"]')
for j in rating1:

    Rating1.append(j.text)
```

```
nxt_button=driver.find_element_by_xpath('//a[@class="_1LKTO3"]')      #scraping the
time.sleep(2)
nxt_button.click()
```

```
start=0
end=49

for page in range(start,end):
    review1=driver.find_elements_by_xpath('//div[@class="t-ZTKy"]/div/div')
    for j in review1:
        Review1.append(j.text)
    rating1=driver.find_elements_by_xpath('//div[(@class="_3LWZlK _1BLPMq" or @class="_3LWZlK _32lA32 _1BLPMq")or @class="_3LWZlK
    for j in rating1:
        Rating1.append(j.text)
    time.sleep(1)

    nxt_button=driver.find_element_by_xpath('//a[@class="_1LKTO3"][2]')     #scraping the list of buttons from the page
    nxt_button.click()
    time.sleep(1)
```

```
Laptop1=pd.DataFrame({})
Laptop1["Review"]= Review1
Laptop1["Rating"]= Rating1

Laptop1
```

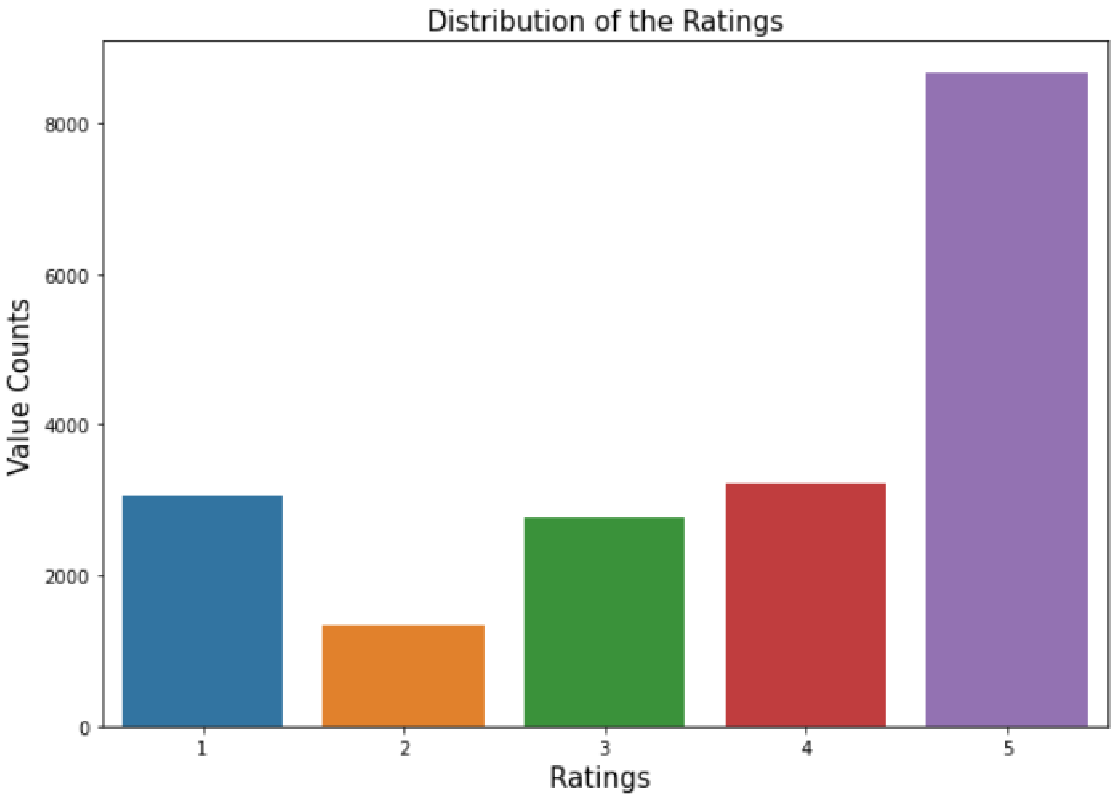
Sample Data Collected

	Review	Rating
0	A bit expensive when we compare with today's i...	4
1	Fantastic value for money machine!! Absolute b...	5
2	The best you can get, looks and performance bo...	5
3	Ultimate machine, best laptop I have ever used...	5
4	For everyone, who is planning to buy MBA M1-\n...	5
...	...	...
407	Awesome!!!!!	5
408	Best tech from Apple. Thank you Flipkart for o...	5
409	Nothing much to say as it is a macbook. M1 pro...	5
410	Not like other company apple give best results...	5
411	Beautiful and gorgeous Device	5

Part 2 - Modelling Pre-processing

```
# Let's see how our Target column is distributed
plt.figure(figsize=(10,7))
sns.countplot(df['Rating'])
plt.title('Distribution of the Ratings', fontsize=15)
plt.xlabel('Ratings ', fontsize=15)
plt.ylabel('Value Counts', fontsize=15)
```

```
Text(0, 0.5, 'Value Counts')
```



Pre-Processing Steps:

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

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

# Replace money symbols with 'moneysymb' (£ can by typed with ALT key + 156)
df['Review'] = df['Review'].str.replace(r'Z|\$', ' ')

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


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

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

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

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

Sample Data after Pre-Processing:

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

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

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

```
# New column (clean_length) after puncuation, stopword removal
df['clean_length']=df.Review.str.len()
df.head()
```

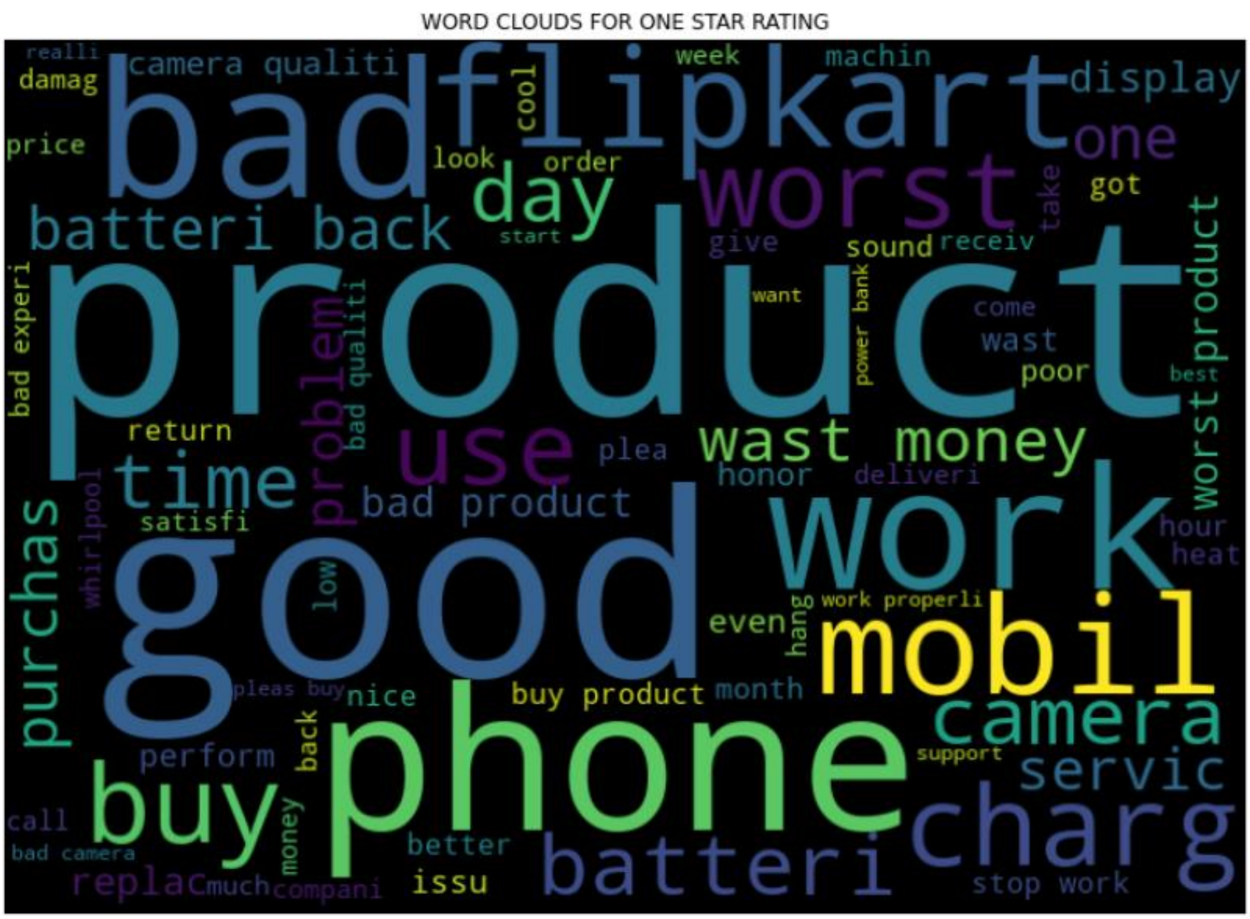
	Review	Rating	length	clean_length
0	bit expensive compare today intel th gen amd r...	4	498	296
1	fantastic value money machine absolute beast f...	5	499	342
2	best get looks performance notch supreme batte...	5	334	200
3	ultimate machine best laptop ever used hands m...	5	183	136
4	everyone planning buy mba pros blazing fast ah...	5	500	330

```python
# writing function for the entire dataset
# Lemmatizing and then Stemming with Snowball to get root words and further reducing characters

from nltk.stem import SnowballStemmer, WordNetLemmatizer
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 token in text:
        if len(token)>=3:
            result.append(lemmatize_stemming(token))

    return result
```

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

for doc in df.review:
    processed_review.append(preprocess(doc))
```

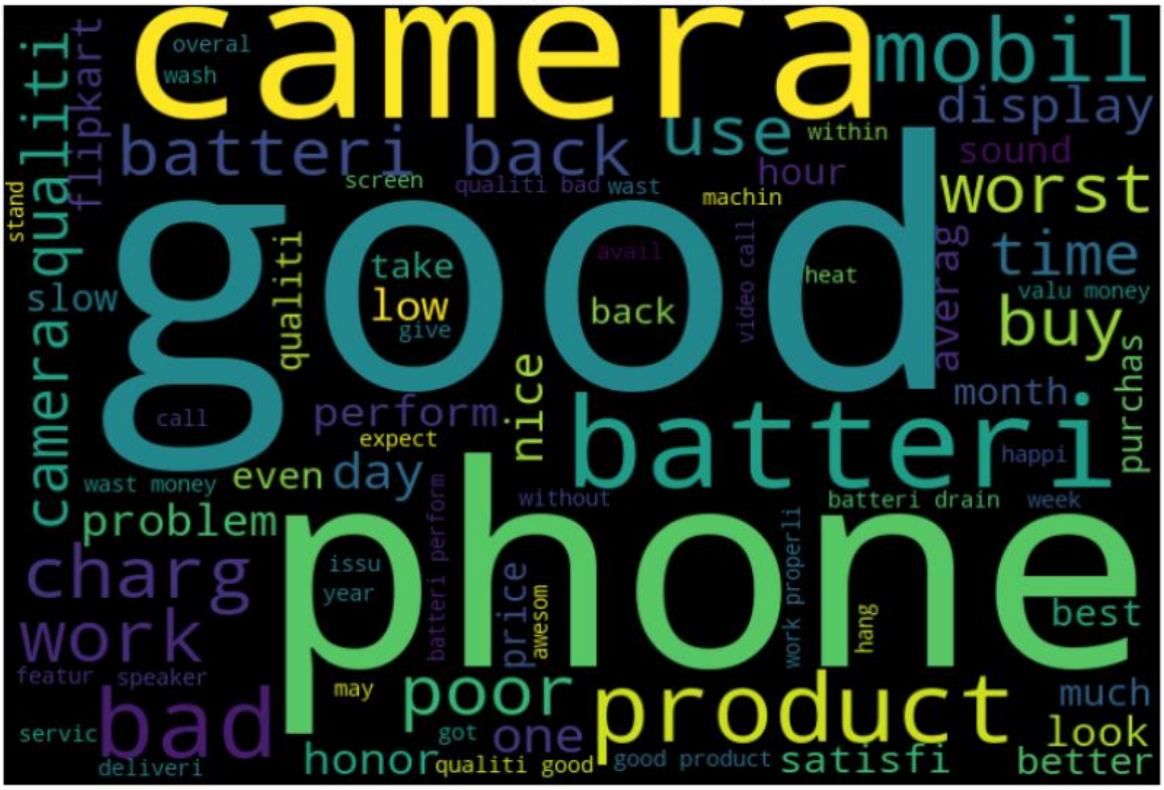
Word Cloud for various Ratings:

Rating=1:



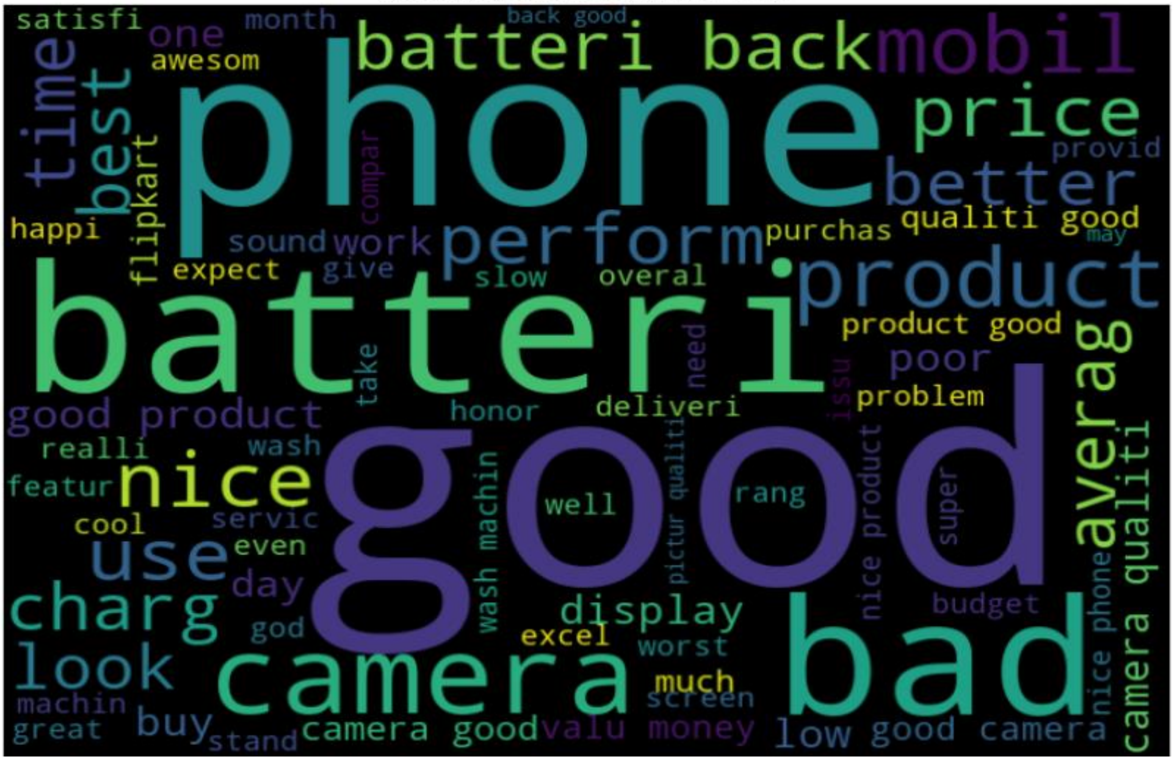
Rating=2:

WORD CLOUDS FOR TWO STAR RATING



Rating=3:

WORD CLOUDS FOR THREE STAR RATING



Rating=4:

WORD CLOUDS FOR FOUR STAR RATING

Rating=5:

WORD CLOUDS FOR FIVE STAR RATING

Before Model Building, Vectoring the dataset

```
tfidf=tf(input='content', encoding='utf-8', lowercase=True,stop_words='english',max_features=10000,ngram_range=(1,3))
```

```
x=tfidf.fit_transform(X).toarray()
```

```
#CHECKING THE SELECTED FEATURE NAMES
tfidf.get_feature_names()[1:9]
```

```
['abl',
 'abl charg',
 'abl fix',
 'abl load',
 'abl use',
 'absent',
 'absolut',
 'absolut beast']
```

List of Models used:

```
# Creating instances

RF=RandomForestClassifier()
MNB=MultinomialNB()
DT=DecisionTreeClassifier()
AD=AdaBoostClassifier()
```

Model Performances:

```
MultinomialNB()


Max Accuracy Score corresponding to Random State  81 is: 0.5753856942496494


Learning Score :  0.6086107145540611
Accuracy Score :  0.5753856942496494
Classification Report:
              precision    recall  f1-score   support

           1       0.59      0.58      0.58       913
           2       0.00      0.00      0.00       403
           3       0.56      0.28      0.37       827
           4       0.21      0.01      0.01       964
           5       0.58      0.97      0.72      2597

    accuracy                           0.58      5704
   macro avg       0.39      0.37      0.34      5704
weighted avg       0.47      0.58      0.48      5704



Confusion Matrix:
 [[ 528    0   69    4  312]
 [ 156    0   75    2  170]
 [ 133    0  229    6  459]
 [  30    0   19    7  908]
 [  47    0   18   14 2518]]
```

```
**************************** DecisionTreeClassifier ******************************


DecisionTreeClassifier()


Max Accuracy Score corresponding to Random State  61 is: 0.4950911640953717


Learning Score :  0.8943572018934556
Accuracy Score :  0.4894810659186536
Classification Report:
               precision    recall  f1-score   support

           1       0.47      0.51      0.49       913
           2       0.15      0.10      0.12       403
           3       0.35      0.28      0.31       827
           4       0.23      0.15      0.18       964
           5       0.61      0.74      0.67      2597

    accuracy                           0.49      5704
   macro avg       0.36      0.35      0.35      5704
weighted avg       0.45      0.49      0.47      5704



Confusion Matrix:
 [[ 462   92  136   36  187]
 [ 158   39   84   22  100]
 [ 147   71  234   76  299]
 [  72   21   74  146  651]
 [ 139   36  144  367 1911]]
```

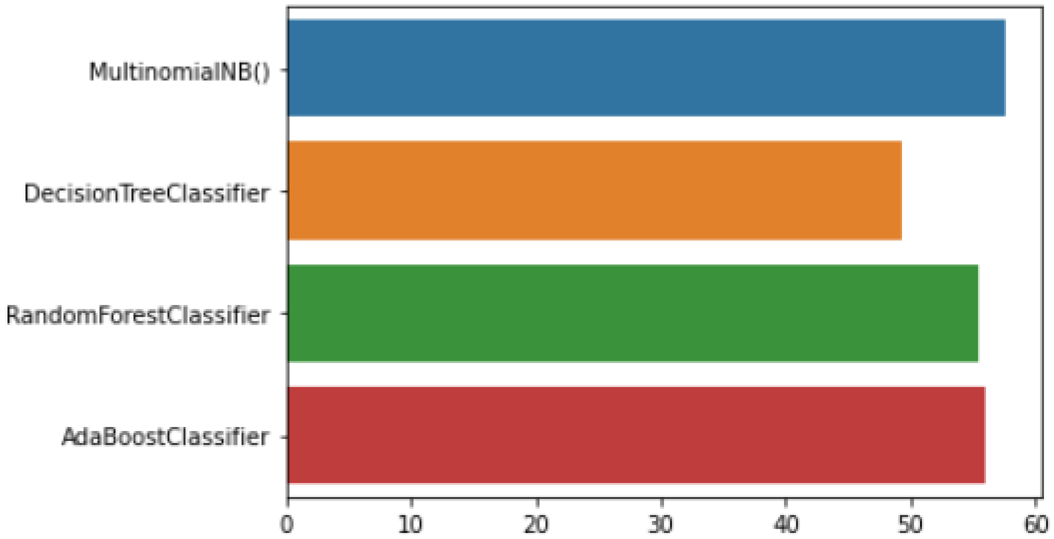
```
*************************** RandomForestClassifier *****************************


RandomForestClassifier()


Max Accuracy Score corresponding to Random State  62 is: 0.5638148667601683


Learning Score :  0.8948080246449771
Accuracy Score :  0.5618863955119214
Classification Report:
               precision    recall  f1-score   support

           1       0.57      0.58      0.58       913
           2       0.17      0.04      0.06       403
           3       0.44      0.28      0.34       827
           4       0.23      0.03      0.06       964
           5       0.60      0.92      0.72      2597

    accuracy                           0.56      5704
   macro avg       0.40      0.37      0.35      5704
weighted avg       0.48      0.56      0.49      5704



Confusion Matrix:
 [[ 531   34  102   12  234]
 [ 145   15  116    3  124]
 [ 149   26  231   23  398]
 [  36    4   29   32  863]
 [  67   11   52   71 2396]]
```

Model Comparisons:

Learning Scores

	Model	Learning Score
0	MultinomialNB()	60.861071
1	DecisionTreeClassifier	89.435720
2	RandomForestClassifier	89.285446
3	AdaBoostClassifier	55.871966

Accuracy



Random Forest Classifier gives best results.

```
Accuracy Score: 0.511921458625526
Confusion Matrix:
 [[ 260    0   21    0  632]
 [  74    0   27    0  302]
 [  39    0   74    0  714]
 [   5    0    0    0  959]
 [  10    0    1    0 2586]]
Classification Report:
              precision    recall  f1-score   support

           1       0.67      0.28      0.40       913
           2       0.00      0.00      0.00       403
           3       0.60      0.09      0.16       827
           4       0.00      0.00      0.00       964
           5       0.50      1.00      0.66      2597

    accuracy                           0.51      5704
   macro avg       0.35      0.27      0.24      5704
weighted avg       0.42      0.51      0.39      5704
```

Sample Predictions:

	Rating	Predicted values
7676	5	5
16596	2	1
9041	4	5
1558	4	5
16066	1	1
...	...	...
7819	3	5
3897	5	5
12726	5	5
2734	4	5
13920	4	5

5704 rows × 2 columns

Saving Model as pkl file

```
# Creating Pickle File
import joblib
joblib.dump(clf_rf,'Ratings_Predict.pkl')
```

```
['Ratings_Predict.pkl']
```

Conclusions and Scope of Improvements:

- We have got a good accuracy score of more than 55 using Random Forest Classifier.

- I have only been able to use 1 website – Flipkart, due to deadline and health issues, but using data from Various other sites like Amazon and Myntra etc can further enhance the model.