Ratings Prediction Project

Submitted by:

SONALI WARKHADE

ACKNOWLEDGMENT

I would like to express my thanks of gratitude to mentors Shubham Yadav as well as Flip Robo who gave me the golden opportunity to do this wonderful project on the topic Micro Credit Defaulter, which also helped me in doing a lot of research and I came to know about so many new things. I am really very thankful to them.

I am making this project to increase my knowledge.

INTRODUCTION

Problem Statement:

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.

This project consists on of 2 phases:

Data Collection Phase

We 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.

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.

Analytical Problem Framing

With the help of Selenium I have scrapped data from different websites like, Snapdeal, Flipkart and Amazon and used Pandas library to save the data in excel file. Just taking a glace on basic code for scrapping from different websites.

```
# Opening the homepage of Nykaa
url = "https://www.nykaa.com/"
driver.get(url)

time.sleep(2)

search_bar = driver.find_element_by_class_name("css-1p0hsio")    # Locating searc_bar by id
search_bar.clear()                                               # clearing search_bar
search_bar.send_keys('Headphones')                                   # sending user input to search bar
search_bar.submit()
```

```
links1=[]
driver.execute_script("window.scrollTo(0,document.body.scrollHeight)")
time.sleep(2)
try:
    for l in driver.find_elements_by_xpath('//div[@class="product-list-box card desktop-cart"]/a'):
        links1.append(l.get_attribute('href'))
        time.sleep(1)
except NoSuchElementException:
        links1.append("-")
        time.sleep(1)
```

```
len(links1)
```

```
Rating1=[]
Review1=[]
Review_title1=[]
for url in links1:
    driver.get(url)
    time.sleep(2)

    try:
        all_reviews = driver.find_element_by_xpath('//a[@class="all-review-btn"]')
        a =all_reviews.get_attribute('href')
        driver.get(a)

        dropdown = driver.find_element_by_xpath('//*[@id="js-pd-scroll-start"]/div[1]/div[2]/section/div/div')
        negative = driver.find_element_by_xpath('//*[@id="js-pd-scroll-start"]/div[1]/div[2]/section/div/div/span')
        action= ActionChains(driver)
        action.move_to_element(dropdown).move_to_element(negative).click().perform()

            # Extracting stars
        try:
            rating = driver.find_elements_by_xpath('//div[@class="desc-section"]/div[1]/div[1]')
            for r in rating:
                Rating1.append(r.text)
        except NoSuchElementException:
            Rating1.append('-')

        time.sleep(1)
        # Extracting review
        try:
            review = driver.find_elements_by_xpath('//div[@class="review-details"]/section/p')
            for r in review:
                Review1.append(r.text)
        except NoSuchElementException:
            Review1.append('-')

        time.sleep(1)
        # Extracting review
        try:
            review_title = driver.find_elements_by_xpath('//div[@class="review-details"]/section/h4')
            for r in review_title:
                Review_title1.append(r.text)
        except NoSuchElementException:
            Review_title1.append('-')
    except TimeoutException:
        pass
    except NoSuchElementException:
        pass
```

Saving the scrapped data into a csv file:

```
# dictionary of lists
dict = {'Rating': Rating1, 'Review': Review1, 'Review_title': Review_title1}

df = pd.DataFrame(dict)

# saving the dataframe
df.to_csv('Nykaa.csv')
```

Snapdeal

```
items=['Headphones', 'Phones', 'smart watches', 'Monitors', 'Home theater', 'Router']
links2=[]
for i in items:

      # Opening the homepage of snapdeal
    url = "https://www.snapdeal.com/"
    driver.get(url)

    time.sleep(2)

    search_bar = driver.find_element_by_xpath('//*[@id="inputValEnter"]')    # Finding the search bar using it's xpath
    search_bar.clear()                                                       # clearing search_bar
    search_bar.send_keys(i)                                                  # sending user input to search bar
    search_button = driver.find_element_by_xpath('//*[@id="sdHeader"]/div[4]/div[2]/div/div[2]/button')       # Locating sea
    search_button.click()

    try:
        for l in driver.find_elements_by_xpath('//a[@class="dp-widget-link noUdLine hashAdded"]'):
            links2.append(l.get_attribute('href'))
            time.sleep(1)
    except NoSuchElementException:
            links2.append("-")
            time.sleep(1)
```

```
len(links2)
```

```
240
```

```
Rating2=[]
Review2=[]
Review_title2=[]
for url in links2:
    driver.get(url)
    time.sleep(2)

        # Extracting stars
    try:
        rating = driver.find_elements_by_xpath('//div[@class="user-review"]//div[@class="rating"]')
        for r in rating:
            child= r.find_elements_by_tag_name("i")
            Rating2.append(len(child))
    except NoSuchElementException:
        Rating2.append('-')

    time.sleep(1)
    # Extracting review
    try:
        review = driver.find_elements_by_xpath('//div[@class="commentreview"]/div/div[2]/div[2]')
        for r in review:
            Review2.append(r.text)
    except NoSuchElementException:
        Review2.append('-')

    time.sleep(1)
    # Extracting review
    try:
        review_title = driver.find_elements_by_xpath('//div[@class="commentreview"]/div/div[2]/div[1]/div[2]')
        for r in review_title:
            Review_title2.append(r.text)
    except NoSuchElementException:
        Review_title2.append('-')
```

```
# dictionary of lists
dict2 = {'Rating': Rating2, 'Review': Review2, 'Review_title': Review_title2}

df = pd.DataFrame(dict2)

# saving the dataframe
df.to_csv('Snapdeal.csv')
```

```
# Opening the homepage of flipkart
url = "https://www.flipkart.com/"
driver.get(url)

try:
    login_X_button = driver.find_element_by_xpath('//button[@class="_2KpZ6l _2doB4z"]') # Button to close login popup
    login_X_button.click()
except NoSuchElementException :
    print("No Login page")

time.sleep(2)
try:
    search_bar = driver.find_element_by_xpath('//*[@id="container"]/div/div[1]/div[1]/div[2]/div[2]/form/div/div/input')
    search_bar.clear()                                          # clearing search_bar
    search_bar.send_keys('laptops')                               # sending user input to search bar
    search_button = driver.find_element_by_xpath('//*[@id="container"]/div/div[1]/div[1]/div[2]/div[2]/form/div/button')
    search_button.click()
except NoSuchElementException:
    driver.get(url)
```

```
links=[]
try:
    for l in driver.find_elements_by_xpath('//a[@class="_1fQZEK"]'):
        links.append(l.get_attribute('href'))
        time.sleep(1)
except NoSuchElementException:
        links.append("-")
        time.sleep(1)

for page in range(0,10):
        nxt_button=driver.find_elements_by_xpath("//a[@class='_1LKTO3']")#scraping the list of buttons from the page
        time.sleep(2)
        try:
            driver.get(nxt_button[1].get_attribute('href'))#getting the link from the list for next page
            time.sleep(2)
        except:
            driver.get(nxt_button[0].get_attribute('href'))
            time.sleep(2)
        try:
            for l in driver.find_elements_by_xpath('//a[@class="_1fQZEK"]'):
                links.append(l.get_attribute('href'))
                time.sleep(1)
        except NoSuchElementException:
                links.append("-")
                time.sleep(1)
```

```
len(links)
```

```
264
```

```
Rating=[]
Review=[]
Review_title=[]
for url in links:
    driver.get(url)
    time.sleep(2)
    try:

        all_reviews = driver.find_element_by_xpath('//div[@class="col JOpGWq"]/a')
        a =all_reviews.get_attribute('href')
        driver.get(a)
        driver.find_element_by_xpath('//*[@id="container"]/div/div[3]/div/div[1]/div[2]/div[1]/div/div[2]/div/div/div/div/se
        driver.find_element_by_xpath('//select[@class="_1EDlbo tVKh2S"]/option[@value="NEGATIVE_FIRST"]').click()
        driver.get(driver.current_url)


        # Extracting stars
        try:
            rating = driver.find_elements_by_xpath('//div[@class="col _2wzgFH K0kLPL"]/div[1]/div[1]')
            for r in rating:
                Rating.append(r.text)
        except NoSuchElementException:
            Rating.append('-')
        except StaleElementReferenceException:
            Rating.append('-')

        time.sleep(1)
        # Extracting review
        try:
            review = driver.find_elements_by_xpath('//div[@class="t-ZTKy"]/div/div')
            for r in review:
                Review.append(r.text)
        except NoSuchElementException:
            Review.append('-')

        time.sleep(1)
        # Extracting review
        try:
            review_title = driver.find_elements_by_xpath('//p[@class="_2-N8zT"]')
            for r in review_title:
                Review_title.append(r.text)
        except NoSuchElementException:
            Review_title.append('-')
    except TimeoutException:
        pass
    except NoSuchElementException:
        pass
```

Amazon

```
#webdriver
driver=webdriver.Chrome(r"C:\Users\SAGAR KADAM\Downloads\chromedriver_win32 (2)\chromedriver.exe")
time.sleep(3)
```

- TV

```
# Opening the homepage of Amazon.in
url = "https://www.amazon.in/"
driver.get(url)

time.sleep(2)
search_bar = driver.find_element_by_id("twotabsearchtextbox")    # Locating searc_bar by id
search_bar.clear()                                               # clearing search_bar
search_bar.send_keys('TV')                                     # sending user input to search bar
search_button = driver.find_element_by_xpath('//div[@class="nav-search-submit nav-sprite"]/span/input')       # Locating sea
search_button.click()
links=[]
try:
    for l in driver.find_elements_by_xpath('//a[@class="a-link-normal a-text-normal"]'):
        links.append(l.get_attribute('href'))
        time.sleep(1)
except NoSuchElementException:
        links.append("-")

```

```
len(links)
```

22

```
Rating=[]
Review=[]
Review_title=[]
for url in links:
    driver.get(url)
    time.sleep(2)
    try:
        all_reviews = driver.find_element_by_xpath('//*[@id="reviews-medley-footer"]/div[2]/a')
        a =all_reviews.get_attribute('href')
        driver.get(a)
        driver.find_element_by_xpath('//*[@id="a-autoid-6-announce"]').click()
        driver.find_element_by_xpath('//*[@id="star-count-dropdown_4"]').click()
        driver.get(driver.current_url)
             # Extracting stars
        try:
            rating = driver.find_elements_by_xpath('//i[@data-hook="review-star-rating"]/span')
            for r in rating:
                Rating.append(2)
        except NoSuchElementException:
            Rating.append('-')

        time.sleep(1)
        # Extracting review
        try:
            review = driver.find_elements_by_xpath('//div[@class="a-section celwidget"]/div[4]')
            for r in review:
                Review.append(r.text)
        except NoSuchElementException:
            Review.append('-')

        time.sleep(1)
        # Extracting review
        try:
            review_title = driver.find_elements_by_xpath('//a[@data-hook="review-title"]/span')
            for r in review_title:
                Review_title.append(r.text)
        except NoSuchElementException:
            Review_title.append('-')

    except NoSuchElementException:
        pass
```

```
# dictionary of lists
dict5 = {'Rating': total_R, 'Review': total_Re, 'Review_title': total_Rt}

df = pd.DataFrame(dict5)

# saving the dataframe
df.to_csv('Amazon.csv')
```

```
filename=['Nykaa.csv', 'Snapdeal.csv','Flipkart1.csv','Flipkart2.csv','Amazon.csv']
```

```
combined_csv = pd.concat([pd.read_csv(f) for f in filename])
```

```
combined_csv.to_excel( "Rating.xlsx", index=False, encoding='utf-8')
```

```
combined_csv.to_excel( "Rating_combined.xlsx", index=False, encoding='utf-8')
```

Firstly, we will start by importing required libraries and databases.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from wordcloud import WordCloud

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier, RandomForestClassifier

from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score

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

```
pd.set_option('display.max_colwidth',100 )
cd=pd.read_excel("Rating.xlsx")
data=pd.DataFrame(data=cd)
data
```

	Unnamed: 0	Rating	Review	Review_title
0	0	5star	Earlier i was using apple ipod but sound was not that much good and today i got this. Seriously ...	"Amazing sound "
1	1	5star	I mean for this price.... they're really nice one Good quality... Aesthetically pleasing and sou...	"Nice "
2	2	5star	perfect fitting nice sound quality awesome	"Amazing Product"
3	3	5star	Decent product. Worth giving a try!	"Good"
4	4	5star	Very good colour for girls . Nice voice i liked it. Price is good	"Very good product"
...	...	...	...	...
29016	1089	2	When I press the input mode change buttons on speaker or the remote there are only three options...	Sounds Quality is good but USB/SD mode not working
29017	1090	2	It's an amazing product but the one thing lack is bass control its poor. At this price range it ...	Ok ok it's sound is good but base quality very poor.
29018	1091	2	sound quality is ok bass is not good and if you are buying it for Bluetooth dont buy the worst B...	For wireless worst product
29019	1092	2	Worst ever music system just in a month it's 2 satellite audio Jack's got bad. Now i have to car...	Got bad in a month 😔. They are hundreds of complaints don't buy. And don't ignore complaints.
29020	1093	2	Firstly when I ordered this product I wanted to give my friend this as gift for his new shop. Th...	Low sound quality

29021 rows × 4 columns

```
data.columns
```

```
Index(['Unnamed: 0', 'Rating', 'Review', 'Review_title'], dtype='object')
```

Above is the list of columns in our database.

```
data.shape
```

```
(29021, 4)
```

Our database consists of 4 columns and 29021 rows.

```
data.dtypes
```

```
Unnamed: 0       int64
Rating          object
Review          object
Review_title    object
dtype: object
```

Our dataset consists of integer and object type of data.

```
#Unnamed: 0 is not required for further analysis hence we will drop the column
data.drop('Unnamed: 0',axis=1, inplace=True)
```

```
data.dropna(inplace=True)#dropped rows which consists of null values
```

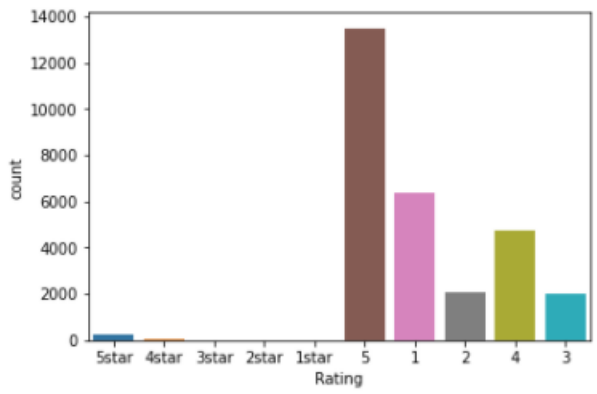
```
#lets check number of unique values in all columns
data.nunique()
```

```
Rating             10
Review          13350
Review_title     1672
dtype: int64
```

Our rating should only consist of 5 unique values i.e 1, 2, 3, 4, 5. let's have a look at rating column data with the help of countplot.

```
sns.countplot(data["Rating"])
```

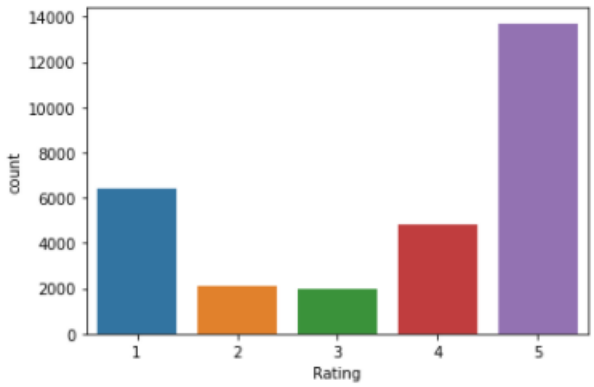
```
<matplotlib.axes._subplots.AxesSubplot at 0x23a33b1bf70>
```




```
data['Rating'] = data['Rating'].replace(['1star'],1, regex=True)#replacing 1star with 1
data['Rating'] = data['Rating'].replace(['2star'],2, regex=True)#replacing 2star with 2
data['Rating'] = data['Rating'].replace(['3star'],3, regex=True)#replacing 3star with 3
data['Rating'] = data['Rating'].replace(['4star'],4, regex=True)#replacing 4star with 4
data['Rating'] = data['Rating'].replace(['5star'],5, regex=True)#replacing 5star with 5
```

```
# reviewing rating column again
sns.countplot(data["Rating"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x23a33d55580>
```




```
#Lets look at counts of each values
data['Rating'].value_counts()
```

```
5    13709
1     6403
4     4802
2     2103
3     2003
Name: Rating, dtype: int64
```

```
data.shape
```

```
(29020, 3)
```

Let's do preprocessing of data with the help of NLP:

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import string
print("Stopwords :\n", stopwords.words("english"))
print("\n \n Punctuations :\n",string.punctuation)
```

```
Stopwords :
 ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours',
'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itsel
f', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'thos
e', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a',
'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against',
'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'of
f', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both',
'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very',
's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'ar
en', "aren't", couldn', "couldn't", didn', "didn't", "doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "have
n't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "should
n't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]


 Punctuations :
 !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

```
punct=string.punctuation
```

```
#function to remove punctuation
def remove_punctuation(txt):
    txt_nopunct="".join([c for c in txt if c not in punct])
    return txt_nopunct
```

```
#function to remove stopwords
def rem_stopworrds(text):
    stop_words=set(stopwords.words("english"))
    word_tokens=word_tokenize(text)
    filtered_text=" ".join([word for word in word_tokens if word not in stop_words])
    return filtered_text
```

```
#let's preprocss all 3 columns
data['Review'] = data['Review'].apply(lambda x: x.lower())# lets change the review into lowercase
data['Review_title'] = data['Review_title'].apply(lambda x: x.lower())# lets change the review_title into lowercase

data['Review'] = data['Review'].apply(lambda x: remove_punctuation(x))
data['Review'] = data['Review'].apply(lambda x: rem_stopworrds(x))

data['Review_title'] = data['Review_title'].apply(lambda x: remove_punctuation(x))
data['Review_title'] = data['Review_title'].apply(lambda x: remove_punctuation(x))
```

```
data['Review']
```

```
0        earlier using apple ipod sound much good today got seriously ipod amazing sound quality bass eve...
1                  mean price theyre really nice one good quality aesthetically pleasing sound quality good
2                                                          perfect fitting nice sound quality awesome
3                                                                  decent product worth giving try
4                                                        good colour girls nice voice liked price good
                                                  ...
29016    press input mode change buttons speaker remote three options aux bluetooth fm option usb sd card...
29017    amazing product one thing lack bass control poor price range could given better features atleast...
29018    sound quality ok bass good buying bluetooth dont buy worst bluetooth quality range buy portable ...
29019    worst ever music system month 2 satellite audio jacks got bad carry service centres dont know ki...
29020    firstly ordered product wanted give friend gift new shop product delivered time woofer working r...
Name: Review, Length: 29020, dtype: object
```

```
import nltk
wn=nltk.WordNetLemmatizer()
```

```
#lemmatization function
def lemmatization(txt):
    text="".join(wn.lemmatize(c) for c in txt)
    return text
```

```
#lemmatized the reviews
data['Review'] = data['Review'].apply(lambda x: lemmatization(x))
```

```
data['Review']
```

```
0        earlier using apple ipod sound much good today got seriously ipod amazing sound quality bass eve...
1                  mean price theyre really nice one good quality aesthetically pleasing sound quality good
2                                                          perfect fitting nice sound quality awesome
3                                                                  decent product worth giving try
4                                                        good colour girls nice voice liked price good
                                                  ...
29016    press input mode change buttons speaker remote three options aux bluetooth fm option usb sd card...
29017    amazing product one thing lack bass control poor price range could given better features atleast...
29018    sound quality ok bass good buying bluetooth dont buy worst bluetooth quality range buy portable ...
29019    worst ever music system month 2 satellite audio jacks got bad carry service centres dont know ki...
29020    firstly ordered product wanted give friend gift new shop product delivered time woofer working r...
Name: Review, Length: 29020, dtype: object
```

Let's join all words in review column in a variable text:

```
text = " ".join(review for review in data['Review'])
print ("There are {} words in the combination of all review.".format(len(text)))
```

```
There are 1862925 words in the combination of all review.
```

```
stopwords_en = set(stopwords.words('english'))
```

Wordcloud

Word cloud of word in text variable:

```
# Create stopword list:
stopwords = set(stopwords_en)

# Generate a word cloud image
wordcloud = WordCloud(stopwords=stopwords, background_color="white").generate(text)

# Display the generated image:
# the matplotlib way:
plt.figure(figsize=(10,8),facecolor='r')
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

Above cloud gives us the most common words in entire reviews in bigfont size, word with smaller size is less used or less common.

```python
#Getting sense of loud words in spam
from wordcloud import WordCloud


view = data['Review'][data['Rating']==1]

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

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

```
#Getting sense of loud words
from wordcloud import WordCloud


view = data['Review_title'][data['Rating']==1]

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

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

Let's use TFidf method to transform data into vectors:

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
tfidf=TfidfVectorizer(max_features=20000, ngram_range=(1,3),analyzer="char")
```

```
x=tfidf.fit_transform(data['Review'])
y=data['Rating']
```

```
print(x.shape,y.shape)
```

```
(29020, 13343) (29020,)
```

We will use under sampling method to balance the target column value counts:

```
from imblearn import under_sampling
from collections import Counter
```

```
from imblearn.under_sampling import RandomUnderSampler

rus=RandomUnderSampler(random_state=0)
x_resample,y_resampled=rus.fit_resample(x,y)
print(sorted(Counter(y_resampled)),y_resampled.shape)
```

```
[1, 2, 3, 4, 5] (10015,)
```

```
y_resampled.value_counts()
```

```
5    2003
4    2003
3    2003
2    2003
1    2003
Name: Rating, dtype: int64
```

```
x_resample.shape
```

```
(10015, 13343)
```

```
y_resampled.shape
```

```
(10015,)
```

```
x=x_resample
y=y_resampled
```

```
x_train,x_test, y_train,y_test=train_test_split(x,y,test_size=0.2, random_state=0)
```

```
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

```
((8012, 13343), (8012,), (2003, 13343), (2003,))
```

Let's find best random state:

```
maxAccu=0
maxRS=0
for i in range(1,200):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=i)
    LR=LogisticRegression()
    LR.fit(x_train,y_train)
    pred=LR.predict(x_test)
    acc=accuracy_score(y_test,pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print("Best accuracy is ",maxAccu, " on Random State ",maxRS)
```

```
Best accuracy is  0.5831253120319521  on Random State  21
```

Model/s Development and Evaluation

```
x_train,x_test, y_train,y_test=train_test_split(x,y,test_size=0.2, random_state=maxRS)
```

```
model=[LogisticRegression(),DecisionTreeClassifier(),SVC(),AdaBoostClassifier(),RandomForestClassifier()]

for m in model:
    m.fit(x_train,y_train)
    #m.score(x_train,y_train)
    pred=m.predict(x_test)
    acc=accuracy_score(y_test,pred)
    print('Accuracy Score of',m,'is:', acc)
    print(confusion_matrix(y_test,pred))
    print(classification_report(y_test,pred))
    print('\n')
```

```
Accuracy Score of LogisticRegression() is: 0.5831253120319521
[[273  61  31  15   8]
 [ 67 253  46  13   2]
 [ 62  53 186  72  43]
 [ 21  23  47 231  88]
 [ 14  20  31 118 225]]
              precision    recall  f1-score   support

           1       0.62      0.70      0.66       388
           2       0.62      0.66      0.64       381
           3       0.55      0.45      0.49       416
           4       0.51      0.56      0.54       410
           5       0.61      0.55      0.58       408

    accuracy                           0.58      2003
   macro avg       0.58      0.59      0.58      2003
weighted avg       0.58      0.58      0.58      2003
```

```
Accuracy Score of DecisionTreeClassifier() is: 0.5621567648527209
[[243  60  39  24  22]
 [ 72 201  50  36  22]
 [ 44  32 236  71  33]
 [ 23  21  43 258  65]
 [ 33  26  38 123 188]]
              precision    recall  f1-score   support

           1       0.59      0.63      0.61       388
           2       0.59      0.53      0.56       381
           3       0.58      0.57      0.57       416
           4       0.50      0.63      0.56       410
           5       0.57      0.46      0.51       408

    accuracy                           0.56      2003
   macro avg       0.57      0.56      0.56      2003
weighted avg       0.57      0.56      0.56      2003


Accuracy Score of SVC() is: 0.6380429355966051
[[302  44  17  18   7]
 [ 66 260  39  14   2]
 [ 53  39 217  72  35]
 [ 15  15  38 262  80]
 [ 12  12  27 120 237]]
              precision    recall  f1-score   support

           1       0.67      0.78      0.72       388
           2       0.70      0.68      0.69       381
           3       0.64      0.52      0.58       416
           4       0.54      0.64      0.58       410
           5       0.66      0.58      0.62       408

    accuracy                           0.64      2003
   macro avg       0.64      0.64      0.64      2003
weighted avg       0.64      0.64      0.64      2003
```

```
Accuracy Score of AdaBoostClassifier() is: 0.45481777333999
[[228  83  51  12  14]
 [ 83 192  76  19  11]
 [ 69  94 118  73  62]
 [ 13  29  68 132 168]
 [  9  24  47  87 241]]
              precision    recall  f1-score   support

           1       0.57      0.59      0.58       388
           2       0.45      0.50      0.48       381
           3       0.33      0.28      0.30       416
           4       0.41      0.32      0.36       410
           5       0.49      0.59      0.53       408

    accuracy                           0.45      2003
   macro avg       0.45      0.46      0.45      2003
weighted avg       0.45      0.45      0.45      2003


Accuracy Score of RandomForestClassifier() is: 0.6255616575137294
[[293  55  15  17   8]
 [ 73 259  30  11   8]
 [ 47  51 221  66  31]
 [ 28  18  31 255  78]
 [ 23  26  26 108 225]]
              precision    recall  f1-score   support

           1       0.63      0.76      0.69       388
           2       0.63      0.68      0.66       381
           3       0.68      0.53      0.60       416
           4       0.56      0.62      0.59       410
           5       0.64      0.55      0.59       408

    accuracy                           0.63      2003
   macro avg       0.63      0.63      0.62      2003
weighted avg       0.63      0.63      0.62      2003
```

Hyper parameter tunning:

```
from sklearn.model_selection import RandomizedSearchCV
#creating  parameter list to pass in RandomizedSearchCV
```

- LogisticRegression

```
#DecisionTreeClassifier parameters
parameters1={'penalty':['l2','l1','elasticnet', 'none'], 'dual':[True,False],'tol':[0.0001],'C':[1.0],
             'fit_intercept':[True,False],'intercept_scaling':[1], 'class_weight':[None], 'random_state':range(0,20),
             'solver':['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'], 'max_iter':[100],
             'multi_class':['auto', 'ovr', 'multinomial'], 'verbose':[0],'warm_start':[True,False],
             'n_jobs':[None], 'l1_ratio':[None]}
```

```
RSV1=RandomizedSearchCV(LogisticRegression(),parameters1,cv=5)
```

```
RSV1.fit(x_train,y_train)
```

```
RandomizedSearchCV(cv=5, estimator=LogisticRegression(),
                   param_distributions={'C': [1.0], 'class_weight': [None],
                                        'dual': [True, False],
                                        'fit_intercept': [True, False],
                                        'intercept_scaling': [1],
                                        'l1_ratio': [None], 'max_iter': [100],
                                        'multi_class': ['auto', 'ovr',
                                                        'multinomial'],
                                        'n_jobs': [None],
                                        'penalty': ['l2', 'l1', 'elasticnet',
                                                    'none'],
                                        'random_state': range(0, 20),
                                        'solver': ['newton-cg', 'lbfgs',
                                                   'liblinear', 'sag', 'saga'],
                                        'tol': [0.0001], 'verbose': [0],
                                        'warm_start': [True, False]})
```

```
RSV1.best_params_
```

```
{'warm_start': True,
 'verbose': 0,
 'tol': 0.0001,
 'solver': 'sag',
 'random_state': 19,
 'penalty': 'none',
 'n_jobs': None,
 'multi_class': 'ovr',
 'max_iter': 100,
 'l1_ratio': None,
 'intercept_scaling': 1,
 'fit_intercept': True,
 'dual': False,
 'class_weight': None,
 'C': 1.0}
```

```
1 RSV_pred1=RSV1.best_estimator_.predict(x_test)
```

```
1 RSV_pred1
```

```
array([5, 2, 5, ..., 1, 3, 4], dtype=int64)
```

```
1 RSV1.score(x_train,y_train)
```

```
0.8578382426360459
```

Hyper parameter tunning with LogisticRegression gives us 85.78 % accuracy.

- DecisionTreeClassifier

```
1 #DecisionTreeClassifier parameters
2 parameters2={'criterion':["gini", "entropy"],'splitter':["best", "random"],
3             'max_depth':[None],'random_state':range(0,20),
4             'max_leaf_nodes':[None],'class_weight':[None], 'min_samples_split':[1,2,3],
5             'min_samples_leaf':[1,2,3],'max_features':[None],'min_impurity_decrease':[0.0,1.0,2.0],
6             'min_weight_fraction_leaf':[0.0,1.0,2.0]}
```

```
1 RSV2=RandomizedSearchCV(DecisionTreeClassifier(),parameters2,cv=5)
```

```
1 RSV2.fit(x_train,y_train)
```

```
RandomizedSearchCV(cv=5, estimator=DecisionTreeClassifier(),
                   param_distributions={'class_weight': [None],
                                        'criterion': ['gini', 'entropy'],
                                        'max_depth': [None],
                                        'max_features': [None],
                                        'max_leaf_nodes': [None],
                                        'min_impurity_decrease': [0.0, 1.0,
                                                                  2.0],
                                        'min_samples_leaf': [1, 2, 3],
                                        'min_samples_split': [1, 2, 3],
                                        'min_weight_fraction_leaf': [0.0, 1.0,
                                                                     2.0],
                                        'random_state': range(0, 20),
                                        'splitter': ['best', 'random']})
```

```
1 RSV2.best_params_
```

```
{'splitter': 'best',
 'random_state': 14,
 'min_weight_fraction_leaf': 0.0,
 'min_samples_split': 2,
 'min_samples_leaf': 2,
 'min_impurity_decrease': 0.0,
 'max_leaf_nodes': None,
 'max_features': None,
 'max_depth': None,
 'criterion': 'entropy',
 'class_weight': None}
```

```
1 RSV_pred2=RSV2.best_estimator_.predict(x_test)
```

```
1  RSV_pred2
```

```
array([5, 1, 5, ..., 1, 5, 3], dtype=int64)
```

```
1  RSV2.score(x_train,y_train)
```

```
0.8358711932101848
```

Hyper parameter tunning with DecisionTreeClassifier gives us 83.59 % accuracy.

- SCV

```
1  #SVC parameters
2  parameters3={'C':[1, 10, 100, 1000],'kernel':['linear', 'poly', 'rbf', 'sigmoid'],
3      'degree':[3],'gamma':['scale', 'auto'],'coef0':[0.0],'shrinking':[True,False],
4      'probability':[True,False],'tol':[0.001],'cache_size':[200],
5      'class_weight':[None],'decision_function_shape':['ovr'],
6      'break_ties':[True,False],'random_state':range(0,20)}
```

```
1  RSV3=RandomizedSearchCV(SVC(),parameters3,cv=5)
```

```
1  RSV3.fit(x_train,y_train)
```

```
RandomizedSearchCV(cv=5, estimator=SVC(),
                   param_distributions={'C': [1, 10, 100, 1000],
                                        'break_ties': [True, False],
                                        'cache_size': [200],
                                        'class_weight': [None], 'coef0': [0.0],
                                        'decision_function_shape': ['ovr'],
                                        'degree': [3],
                                        'gamma': ['scale', 'auto'],
                                        'kernel': ['linear', 'poly', 'rbf',
                                                   'sigmoid'],
                                        'probability': [True, False],
                                        'random_state': range(0, 20),
                                        'shrinking': [True, False],
                                        'tol': [0.001]})
```

```
1  RSV3.best_params_
```

```
{'tol': 0.001,
 'shrinking': True,
 'random_state': 6,
 'probability': False,
 'kernel': 'poly',
 'gamma': 'scale',
 'degree': 3,
 'decision_function_shape': 'ovr',
 'coef0': 0.0,
 'class_weight': None,
 'cache_size': 200,
 'break_ties': False,
 'C': 1}
```

```
1  RSV_pred3=RSV3.best_estimator_.predict(x_test)
```

```
1  RSV_pred3
```

```
array([5, 2, 5, ..., 1, 4, 4], dtype=int64)
```

```
1  RSV3.score(x_train,y_train)
```

```
0.858337493759361
```

Hyper parameter tunning with SVC gives us 85.83 % accuracy.

- AdaBoostClassifier

```
parameters4={'base_estimator':[None],'n_estimators':[20,40, 50,75,100],
             'learning_rate':[0.1,0.01,1.0,2.0],'algorithm':['SAMME.R','SAMME'],
             'random_state':range(0,20)}

```

```
RSV4=RandomizedSearchCV(AdaBoostClassifier(),parameters4,cv=5)
```

```
RSV4.fit(x_train,y_train)
```

```
RandomizedSearchCV(cv=5, estimator=AdaBoostClassifier(),
                   param_distributions={'algorithm': ['SAMME.R', 'SAMME'],
                                        'base_estimator': [None],
                                        'learning_rate': [0.1, 0.01, 1.0, 2.0],
                                        'n_estimators': [20, 40, 50, 75, 100],
                                        'random_state': range(0, 20)})
```

```
RSV4.best_params_
```

```
{'random_state': 0,
 'n_estimators': 100,
 'learning_rate': 2.0,
 'base_estimator': None,
 'algorithm': 'SAMME'}
```

```
RSV_pred4=RSV4.best_estimator_.predict(x_test)
```

```
RSV_pred4
```

```
array([4, 2, 5, ..., 2, 3, 4], dtype=int64)
```

```
RSV4.score(x_train,y_train)
```

```
0.4162506240639041
```

Hyper parameter tunning with adaboost classifier gives us 41.63 % accuracy.

- RandomForestClassifier

```
parameters5={'n_estimators':[100],'criterion':['gini','entropy'],'max_depth':[None],
             'min_samples_split':[2],'min_samples_leaf':[1],'min_weight_fraction_leaf':[0.0],
             'max_features':['auto'],'max_leaf_nodes':[None],'min_impurity_decrease':[0.0],
             'min_impurity_split':[None],'bootstrap':[True,False],'oob_score':[True,False],'n_jobs':[None],
             'random_state':range(0,20),'verbose':[0],'warm_start':[True,False],'class_weight':[None],
             'ccp_alpha':[0.0],'max_samples':[None]}
```

```
RSV5=RandomizedSearchCV(RandomForestClassifier(),parameters5,cv=5)
```

```
RSV5.fit(x_train,y_train)
```

```
RandomizedSearchCV(cv=5, estimator=RandomForestClassifier(),
                   param_distributions={'bootstrap': [True, False],
                                        'ccp_alpha': [0.0],
                                        'class_weight': [None],
                                        'criterion': ['gini', 'entropy'],
                                        'max_depth': [None],
                                        'max_features': ['auto'],
                                        'max_leaf_nodes': [None],
                                        'max_samples': [None],
                                        'min_impurity_decrease': [0.0],
                                        'min_impurity_split': [None],
                                        'min_samples_leaf': [1],
                                        'min_samples_split': [2],
                                        'min_weight_fraction_leaf': [0.0],
                                        'n_estimators': [100], 'n_jobs': [None],
                                        'oob_score': [True, False],
                                        'random_state': range(0, 20),
                                        'verbose': [0],
                                        'warm_start': [True, False]})
```

```
RSV5.best_params_
```

```
{'warm_start': False,
 'verbose': 0,
 'random_state': 12,
 'oob_score': False,
 'n_jobs': None,
 'n_estimators': 100,
 'min_weight_fraction_leaf': 0.0,
 'min_samples_split': 2,
 'min_samples_leaf': 1,
 'min_impurity_split': None,
 'min_impurity_decrease': 0.0,
 'max_samples': None,
 'max_leaf_nodes': None,
 'max_features': 'auto',
 'max_depth': None,
 'criterion': 'gini',
 'class_weight': None,
 'ccp_alpha': 0.0,
 'bootstrap': True}
```

```
1 RSV_pred5=RSV5.best_estimator_.predict(x_test)
```

```
1 RSV_pred5
```

```
array([5, 2, 5, ..., 1, 1, 4], dtype=int64)
```

```
1 RSV5.score(x_train,y_train)
```

```
0.8775586620069895
```

Hyper parameter tunning with RandomForestClassifier gives us 87.76% accuracy.

Considering RandomForestClassifier with RandomizedSearchCV as final model for saving model.

Saving model:

```
1 import joblib
2 joblib.dump(RSV5,"RSVRPP.obj")
```

```
['RSVRPP.obj']
```

```
1 RSVfile=joblib.load("RSVRPP.obj")
2 RSVfile.predict(x_test)
```

```
array([5, 2, 5, ..., 1, 1, 4], dtype=int64)
```

CONCLUSION

In the Ratings Prediction Project, I have extracted ratings and reviews from different websites then saved the extracted csv file into one excel sheet. Our dataset mainly consists of 3 columns and 29021 rows. Columns are Rating, Review and Review title though we have one more column named Unnamed: 0 however that is not required for further analysis.

Then I did some preprocessing like dropping null values and Our rating should only consist of 5 unique values i.e 1, 2, 3, 4, 5 and the rating column actually consists of 10 unique values hence I used the replace method. Then with the help of NLP I have removed punctuations, stopwords from Review and review title column. Then used a lemmatization method to change words with similar meaning to one common word. With the help of wordcloud I performed some visualization.

I have then used the TFidf method to vectorise and transform data. Values in our tar get column have different value counts hence with the use of random under sampler data of target column came down to equal value count. Used 5 methods for model b uilding then with the help of RandomizedSearchCV I have tried to improve accuracy. Finally, I decided to go ahead with the Random Forest Classifier and saved the mod el.