RATING PREDICTION

SUBMITTED BY

TUSHAR KUMAR PATEL

2021

ACKNOWLEDGEMENT

I would like to express my special thanks to all my mentors of DataTrained and to SME (Keshav Bansal) under whom I am an intern at Fliprobo. They all have taught me Machine learning because of that knowledge they have provided that has helped me to be able to complete this project. In this whole process some popular websites like google, geeksforgeeks was also very helpful for completing the project. For creating the dataset flipkart was the website from where all the data has been scraped using the automation tool selenium. At last github was the location where the final project was uploaded for the submission.

INTRODUCTION

• Business problem framing

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.

• Conceptual Background of the Domain Problem

E-commerce companies offers various products the list is very long so we have decided to do rating prediction from the available reviews present in the e-commerce websites. To do that first we have to scrap the data from the websites. We need to convert the reviews into lower case, then we have to clean the data in which there are many unnecessary things which are no needed while analysis like unwanted spaces, unnecessary use of words, use of short words that they used to communicate in their daily life that is not required for the analysis prediction.

• Motivation for the Problem Undertaken

- The project was the first provided to me by FlipRobo as a part of the internship program. The exposure to real world data and the opportunity to deploy my skillset in solving a real time problem has been the primary objective. However, the motivation for taking this project was that it is relatively a new field of research. Here we have many options but less concrete solutions. The main motivation is to build models which will be used to predict the rating from the reviews provided by the customers in the e-commerce websites. So, that it can be analyzed when the customer provides reviews then those reviews will be converted into overall ratings.

Analytical Problem Framing

• Data Sources and their formats

- The data was scraped using the automation tool named Selenium and beautifulSoup with the help of programing language Python. Which was converted into CSV format for the further analysis for the rating prediction from

the reivews. After loading the training dataset into Jupyter Notebook using Pandas and it can be seen that there are only two columns named as:

```
In [3]: rr = pd.read_csv(r"C:\Users\tusha\Desktop\flip robo\project 11\Reviews.csv")
        rr
```

Out[3]:

	Unnamed: 0	Ratings	Full_review
0	0	5.0	My first impression and One day full usage. 1....
1	1	2.0	I PURCHASED LAPTOP 1 MONTH BACK AND NOW I AM F...
2	2	4.0	Laptop is good at 8gb ram. After I've upgraded...
3	3	5.0	Everything is best. In this price range it is ...
4	4	5.0	It's almost a year I have this laptop It works...
...	...	...	...
103975	103975	5.0	Bought this few days ago and working very good...
103976	103976	3.0	Maximum speed over WiFi is 100Mbps even though...
103977	103977	5.0	It is working fine. Installation was easy. Onl...
103978	103978	4.0	WORST THING I EVER GOT. NOT AT ALL WORKING. MY...
103979	103979	5.0	Best router

```
In [4]: # checking the shape of the datast.
        print("Shape :", rr.shape)
```

```
Shape : (103980, 3)
```

```
In [5]: # printing the name of the columns.
        print("Columns :", rr.columns)
```

```
Columns : Index(['Unnamed: 0', 'Ratings', 'Full_review'], dtype='object')
```

- ## Data Pre-processing Done

For Data pre-processing we did some data cleaning, where we used wordNet lemmatizer to clean the words and removed special characters using Regexp Tokenizer and filter the words by removing stop words and then used lemmatizers and joined and return the filtered words.
Used TFIDF vectorizer to convert those text into vectors, and split the data and into test and train and trained various Machine learning algorithms.

```
In [6]: # Now checking the datatype of the columns
        print("Datatypes :\n", rr.dtypes)
```

```
Datatypes :
 Unnamed: 0       int64
Ratings         float64
Full_review      object
dtype: object
```

```
In [7]: # This method prints information about a rrFrame including the index dtype and column dtypes, non-null values and memory usage.
        rr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103980 entries, 0 to 103979
Data columns (total 3 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   Unnamed: 0   103980 non-null  int64
 1   Ratings      103862 non-null  float64
 2   Full_review  103868 non-null  object
dtypes: float64(1), int64(1), object(1)
memory usage: 2.4+ MB
```

In [8]:
```
# Checking the missing values present in the dataset.
rr.isnull().sum()
```

Out[8]:
```
Unnamed: 0       0
Ratings        118
Full_review    112
dtype: int64
```

In [9]:
```
# checking for the missig values.
print("Missing Value Count :")
print(rr.isnull().sum())
```

```
Missing Value Count :
Unnamed: 0       0
Ratings        118
Full_review    112
dtype: int64
```

Printing the percentage of the missing values present in the dataset.

In [10]:
```
# Printing the Percentage of the missing values in the dataset.
print("Percentage of the missing values :")
print(rr.isna().mean().round(4) * 100)
```

```
Percentage of the missing values :
Unnamed: 0     0.00
Ratings        0.11
Full_review    0.11
dtype: float64
```

In [12]:
```
# Total missing value count
print("Total Missing Value Count :", rr.isnull().sum().values.sum())
```

```
Total Missing Value Count : 230
```

In [13]:
```
# checking if there is any duplicate values present in the dataset.
print(rr.duplicated().value_counts())
```

```
False    103980
dtype: int64
```

In [14]:
```
# printing the duplicate present in the dataset.
print(rr[rr.duplicated()])
```

```
Empty DataFrame
Columns: [Unnamed: 0, Ratings, Full_review]
Index: []
```

In [15]:
```
# removing the column named as Unnamed: 0
rr = rr.drop("Unnamed: 0", axis=1)
rr
```

Out[15]:

	Ratings	Full_review
0	5.0	My first impression and One day full usage. 1....
1	2.0	I PURCHASED LAPTOP 1 MONTH BACK AND NOW I AM F...
2	4.0	Laptop is good at 8gb ram. After I've upgraded...
3	5.0	Everything is best. In this price range it is ...
4	5.0	It's almost a year I have this laptop It works...
...	...	...

In [20]:
```
# count of reviews w.r.t ratings.
rr.Ratings.value_counts()
```

Out[20]:
```
5.0    64792
4.0    20455
1.0     9684
3.0     6033
2.0     2898
Name: Ratings, dtype: int64
```

As we have seen that the data is imbalanced.

Word Count

In [21]:
```
rr["length"] = rr.Full_review.str.len()
rr.head()
```

Out[21]:

	Ratings	Full_review	length
0	5.0	My first impression and One day full usage. 1....	496
1	2.0	I PURCHASED LAPTOP 1 MONTH BACK AND NOW I AM F...	494
2	4.0	Laptop is good at 8gb ram. After I've upgraded...	305
3	5.0	Everything is best. In this price range it is ...	499
4	5.0	It's almost a year I have this laptop It works...	141

In [24]:
```
# convert text to lowercase

rr["Full_review"] = rr["Full_review"].apply(lambda x: str(x).lower())

# rr["Full_review"] = rr["Full_review"].str.lower()
```

In [25]:
```
rr["Full_review"]

# It can be seen that all the review has been converted into the lowercase.
```

Out[25]:
```
0         my first impression and one day full usage. 1....
1         i purchased laptop 1 month back and now i am f...
2         laptop is good at 8gb ram. after i've upgraded...
3         everything is best. in this price range it is ...
4         it's almost a year i have this laptop it works...
                               ...
103975    bought this few days ago and working very good...
103976    maximum speed over wifi is 100mbps even though...
103977    it is working fine. installation was easy. onl...
103978    worst thing i ever got. not at all working. my...
103979                                          best router
Name: Full_review, Length: 103862, dtype: object
```

In [26]:
```
# removing the punctuation, unnecessary spaces, noise removal
rr["Full_review"] = rr["Full_review"].str.replace(r"[^\w\d\s]", " ")


rr["Full_review"] = rr["Full_review"].str.replace(r"\s+", " ")


rr["Full_review"] = rr["Full_review"].str.replace(r"^\s+|\s+?$", "")
```

In [28]:
```
# Removing the stopwords
import string
import nltk
from nltk.corpus import stopwords

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

rr["Full_review"] = rr["Full_review"].apply(
    lambda x: " ".join(term for term in str(x).split() if term not in stop_words)
)"""

stop = stopwords.words("english")
rr["Full_review"] = rr["Full_review"].apply(
    lambda x: " ".join([word for word in x.split() if word not in (stop)])
)
```

In [29]:
```
rr["Full_review"][2:10]
```

Out[29]:
```
2    laptop good numbrgb ram upgraded numbrgb worki...
3    everything best price range beast boot speed u...
4    almost year laptop works well sometimes lag mo...
5    good performance laptops range fast charging d...
6    best laptop price range good programming gamin...
7    want lowest budget gaming laptop would recomme...
8    pros decent battery good display fast bootup r...
9    fab best budget friendly gaming laptop works s...
Name: Full_review, dtype: object
```

In [30]:
```
rr["clean_length"] = rr.Full_review.str.len()
```

In [31]:
```
rr.head()
```

Out[31]:

	Ratings	Full_review	length	clean_length
0	5.0	first impression one day full usage numbr fist...	496	343
1	2.0	purchased laptop numbr month back facing batte...	494	368
2	4.0	laptop good numbrgb ram upgraded numbrgb worki...	305	216
3	5.0	everything best price range beast boot speed u...	499	332
4	5.0	almost year laptop works well sometimes lag mo...	141	90

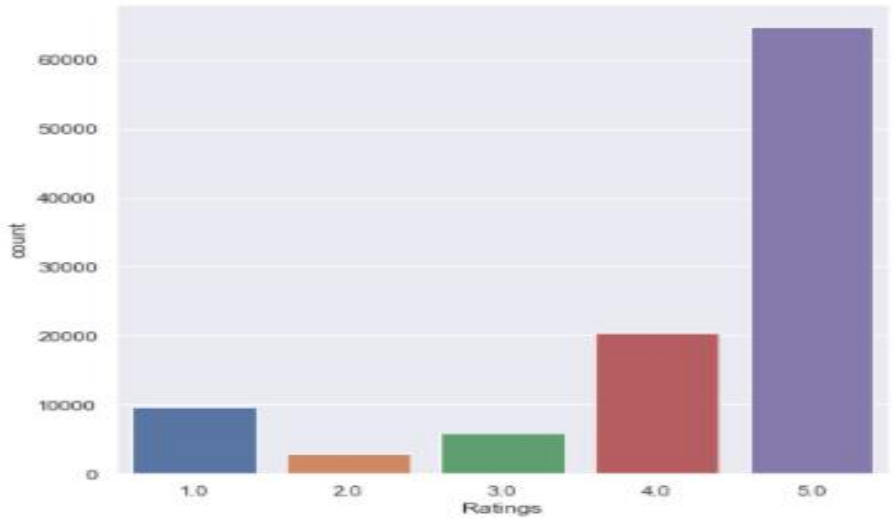
In [32]:
```
# Checking the length of the Original and Cleaned Review.
print("original Review length", rr.length.sum())
print("clean Review length", rr.clean_length.sum())
```

```
original Review length 7441313
clean Review length 5301744
```

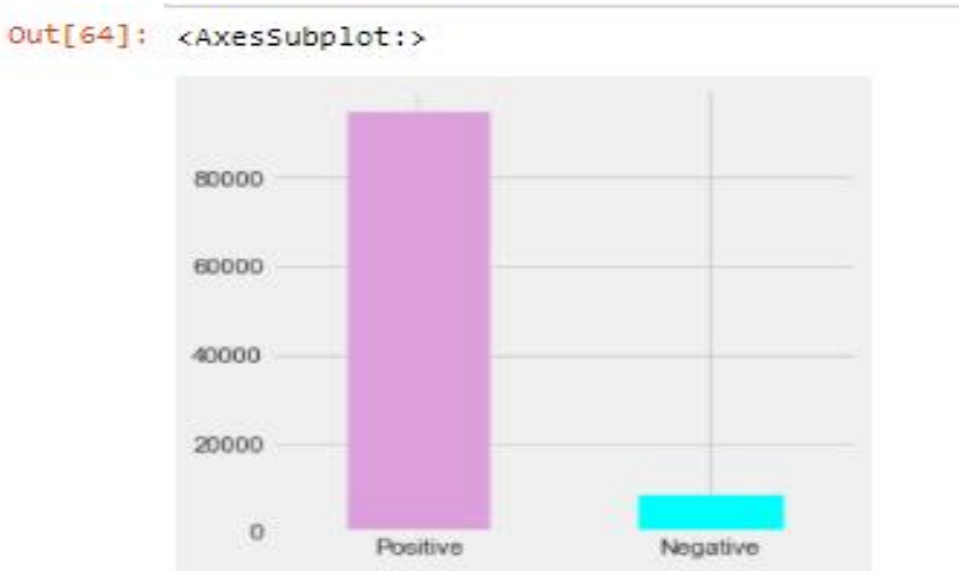
- # Visualization

In this analysis there is only two visualization to be plotted.

1. First one is number of ratings for different rating



2. Second one is about the what is the sentiment of the customer regarding the purchase.



- # Hardware and Software Requirements and Tools Used
  - Hardware: 8GB RAM, 64-bit, 9th gen i7 processor.
  - Software: MS-Excel, Jupyter Notebook, python 3.6.

- **Libraries used:-**

```
In [1]: # importing the required libraries.
        import pandas as pd
        import numpy as np
        import seaborn as sns
        import random
        import plotly.express as px
        import plotly.offline as pyo
        import plotly.graph_objects as go
        import plotly.figure_factory as ff
        import matplotlib.pyplot as plt
        %matplotlib inline

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

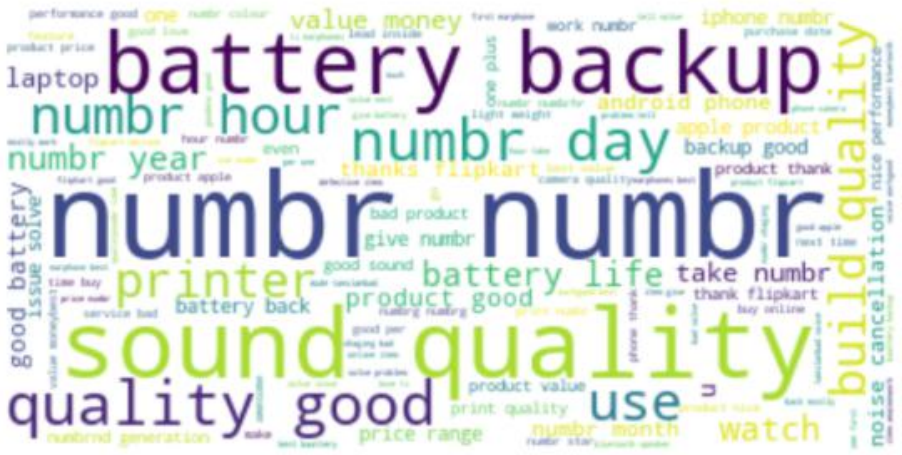
Lemmatization

```
n [40]: import nltk

        from nltk.corpus import stopwords
        from nltk import FreqDist
        from nltk.tokenize import word_tokenize
        from nltk.stem.wordnet import WordNetLemmatizer
        from nltk.corpus import wordnet

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

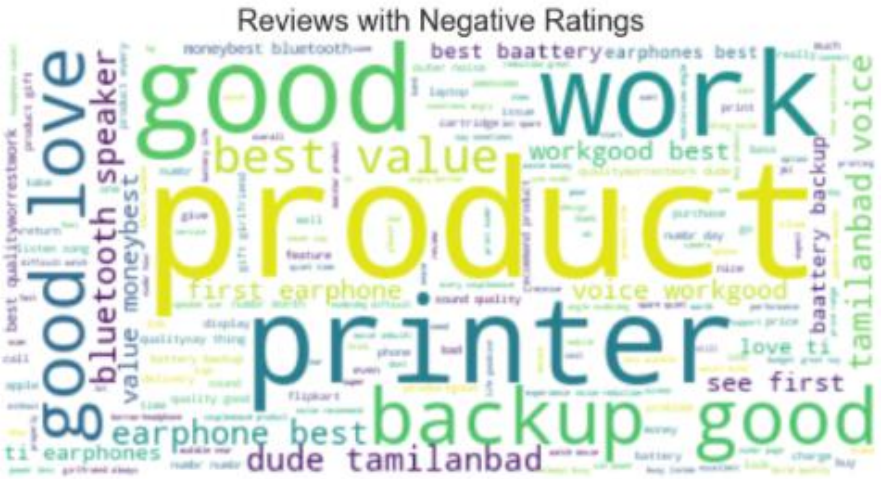
In [55]:
```
# Wordcloud function's input needs to be a single string of text.
# Concatenating all Summaries into a single string.
# Similarly you can build for Text column
reviews_str = reviews_sample.Full_review.str.cat()
wordcloud = WordCloud(background_color="white").generate(reviews_str)
plt.figure(figsize=(12, 12))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```



In [57]:
```
%%time
wordcloud_negative = WordCloud(background_color='white').generate(negative_reviews_str)
wordcloud_positive = WordCloud(background_color='black').generate(positive_reviews_str)
# Plot
fig = plt.figure(figsize=(12,12))
ax1 = fig.add_subplot(211)
ax1.imshow(wordcloud_negative,interpolation='bilinear')
ax1.axis("off")
ax1.set_title('Reviews with Negative Ratings',fontsize=20)
```

```
Wall time: 5.83 s
```

Out[57]:
```
Text(0.5, 1.0, 'Reviews with Negative Ratings')
```



In [58]:
```
fig = plt.figure(figsize=(15, 15))
ax2 = fig.add_subplot(212)
ax2.imshow(wordcloud_positive, interpolation="bilinear")
ax2.axis("off")
ax2.set_title("Reviews with Positive Ratings", fontsize=20)
plt.show()
```

Model/s Development and Evaluation

- **Identification of possible problem-solving approaches (methods).**
  I have did Analysis on this data to understand the value of each feature and the contribution of each feature for model creating and effect of all the feature on the prices. And considering all the point I have built a model that can predict the prices.

- **Testing of Identified Approaches (Algorithms)**
  - Logistic Regression
  - DecisionTree
  - RandomForest
  - PassiveAggressiveClassifier

- **Run and Evaluated selected models**

In [76]:
```
# Logistic Regression

from sklearn.linear_model import LogisticRegression

lor = LogisticRegression()
lor.fit(x_train, y_train)
y_pred = lor.predict(x_test)
scr_lor = cross_val_score(lor, x_over, y_over, cv=5)


print("F1 score \n", f1_score(y_test, y_pred, average="micro"))
print("CV Score :", scr_lor.mean())
print("-----------------------------------------------------------\n")
print("Classification Report \n", classification_report(y_test, y_pred))
print("-----------------------------------------------------------\n")
print("Confusion Matrix \n", confusion_matrix(y_test, y_pred))
```

In [77]:
```
# passive Aggressive Classifier

from sklearn.linear_model import PassiveAggressiveClassifier

pac = PassiveAggressiveClassifier()
pac.fit(x_train, y_train)
y_pred = pac.predict(x_test)
scr_pac = cross_val_score(pac, x_over, y_over, cv=5)


print("F1 score \n", f1_score(y_test, y_pred, average="micro"))
print("CV Score :", scr_pac.mean())
print("-----------------------------------------------------------\n")
print("Classification Report \n", classification_report(y_test, y_pred))
print("-----------------------------------------------------------\n")
print("Confusion Matrix \n", confusion_matrix(y_test, y_pred))
```

In [78]:
```
# Decision tree

from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier()
dt.fit(x_train, y_train)
y_pred = dt.predict(x_test)
scr_dt = cross_val_score(dt, x_over, y_over, cv=5)


print("F1 score \n", f1_score(y_test, y_pred, average="micro"))
print("CV Score :", scr_dt.mean())
print("-----------------------------------------------------------\n")
print("Classification Report \n", classification_report(y_test, y_pred))
print("-----------------------------------------------------------\n")
print("Confusion Matrix \n", confusion_matrix(y_test, y_pred))
```

In [80]:
```
%%time
# Random forest

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier()
rfc.fit(x_train, y_train)
y_pred = rfc.predict(x_test)
scr_rfc = cross_val_score(rfc, x_over, y_over, cv=5)

print("F1 score \n", f1_score(y_test, y_pred, average="micro"))
print("CV Score :", scr_rfc.mean())
print("-----------------------------------------------------------\n")
print("Classification Report \n", classification_report(y_test, y_pred))
print("-----------------------------------------------------------\n")
print("Confusion Matrix \n", confusion_matrix(y_test, y_pred))
```

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

```
F1 score
 0.4746419311026053
CV Score : 0.43183109025805655
-----------------------------------------------------------

Classification Report
              precision    recall  f1-score   support

         1.0       0.53      0.45      0.49     12955
         2.0       0.56      0.52      0.54     12772
         3.0       0.41      0.49      0.45     13054
         4.0       0.40      0.45      0.42     12909
         5.0       0.51      0.46      0.48     13102

    accuracy                           0.47     64792
   macro avg       0.48      0.47      0.48     64792
weighted avg       0.48      0.47      0.48     64792

-----------------------------------------------------------

Confusion Matrix
 [[5874 1620 2079 2001 1381]
 [1289 6586 1975 1824 1098]
 [1401 1488 6408 2422 1335]
 [1209 1191 2729 5872 1908]
 [1300  792 2348 2649 6013]]
```

```
F1 score
 0.4762007655266082
CV Score : 0.45504691937276204
-----------------------------------------------------

Classification Report
              precision    recall  f1-score   support

         1.0       0.38      0.58      0.46     12955
         2.0       0.64      0.55      0.59     12772
         3.0       0.44      0.44      0.44     13054
         4.0       0.54      0.40      0.46     12909
         5.0       0.47      0.42      0.44     13102

    accuracy                           0.48     64792
   macro avg       0.50      0.48      0.48     64792
weighted avg       0.50      0.48      0.48     64792

-----------------------------------------------------

Confusion Matrix
 [[7531  989 1638  990 1807]
 [2587 7029 1469  792  895]
 [3677 1190 5697 1000 1490]
 [3133  899 1805 5137 1935]
 [3103  816 2210 1513 5460]]
```

```
F1 score
 0.6409124583281887
CV Score : 0.595514878380047
-----------------------------------------------------

Classification Report
              precision    recall  f1-score   support

         1.0       0.71      0.64      0.68     12955
         2.0       0.85      0.71      0.77     12772
         3.0       0.54      0.74      0.62     13054
         4.0       0.61      0.54      0.57     12909
         5.0       0.58      0.57      0.57     13102

    accuracy                           0.64     64792
   macro avg       0.66      0.64      0.64     64792
weighted avg       0.66      0.64      0.64     64792

-----------------------------------------------------

Confusion Matrix
 [[8293  428 1895 1075 1264]
 [ 502 9110 1611  654  895]
 [ 759  370 9671 1007 1247]
 [ 967  387 2485 6994 2076]
 [1080  462 2312 1790 7458]]
```

```
F1 score
 0.7061365600691444
CV Score : 0.6626990986541549
-----------------------------------------------------

Classification Report
              precision    recall  f1-score   support

         1.0       0.79      0.72      0.75     12955
         2.0       0.89      0.74      0.81     12772
         3.0       0.58      0.81      0.68     13054
         4.0       0.70      0.62      0.66     12909
         5.0       0.65      0.64      0.64     13102

    accuracy                           0.71     64792
   macro avg       0.72      0.71      0.71     64792
weighted avg       0.72      0.71      0.71     64792

-----------------------------------------------------

Confusion Matrix
 [[ 9332   296  1665   717   945]
 [  398  9506  1545   546   777]
 [  483   296 10570   721   984]
 [  634   252  2249  8022  1752]
 [  930   309  2073  1468  8322]]
Wall time: 1h 57min 16s
```

- Key Metrices used were the Accuracy Score, Cross-validation Score and as this prediction of rating. From the above we can see that there are various models out of which we few gave good accuracy score as more than 71%,

• Interpretation of the Results

➢ From the above visualization and matrices found that Random forest classifier performed the best **i.e. more than 71%.**

CONCLUSION

• Learning Outcomes of the Study in respect of Data Science

The above research will help our client to study that the reviews can be converted into rating

• Limitations of this work and Scope for Future Work

The limitation of the study is that sometime the rating can't be same as the provided one. So based on that again the deciding factors of the might change and we have shortlisted and taken these data for various different kind of products, if the customer has written different word that are not used regularly then our model might fail to predict the accuracy of the rating.