Ratings Prediction Project

Submitted by:

NARA LOHITH

ACKNOWLEDGMENT

The project entitled "RATINGS PREDICTION PROJECT" is done by me during my internship with Flip Robo Technologies. I am grateful to Data Trained and Flip Robo Technologies for their guidance during this project. Other reference websites used to complete this project are:

1. Stackoverflow.com
2. Towardsdatascience.com
3. Medium.com

INTRODUCTION

Conceptual Background of the Domain Problem

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.

Problem Statement

Ratings Prediction: We have to predict ratings of all the products based on the reviews given by customers which will be pre-processed and trained.

Analytical Problem Framing

Data Collection Phase

- We 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 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 model

- Firstly we have collected data i.e, we did webscraping of a website Flipcart and passed collected all the reviews and ratings of products that we wanted to scrape such as : ['laptops', 'Phones', 'Headphones', 'smart watches', 'Professional Cameras', 'Printers', 'monitors', 'Home theater', 'router'].

Finally we created a dataframe and stored the ratings and reviews in it and also saved it in csv format. Sample data is as shown below:

```
In [36]: url1='https://www.flipkart.com/'

In [37]: #creating two empty lists

In [39]: #creating a loop where elements from search_list will be taken
         for item in search_list:
             driver_1 =  webdriver.Chrome('/Users/nlohith/Desktop/chromedriver')
             driver_1.get(url1)

             #Closing the Pop-up button appearing on the page
             close_button=driver_1.find_element_by_xpath("//button[@class='_2KpZ6l _2doB4z']")
             close_button.click()

             # searching required fields i.e., giving the path of search bar and passing item as "keys"
             search=driver_1.find_element_by_xpath("//div[@class='_3005Xc']/input")
             search.send_keys(item)


             search_button=driver_1.find_element_by_xpath("//button[@class='L0Z3Pu']")
             search_button.click() #clicking on the search button
             time.sleep(5)

             # Storing href links of each listing on first page
             links=driver_1.find_elements_by_xpath("//a[@class='_1fQZEK']")
             for i in links:
                 driver_2  = webdriver.Chrome('/Users/nlohith/Desktop/chromedriver')
                 url=i.get_attribute('href')
                 driver_2.get(url)
                 # Opening full reviews of the paticular product in that item
                 try:
                     full=driver_2.find_element_by_xpath("//div[@class='col JOpGWq']/a")
                 except:
                     continue
                 driver_3=  webdriver.Chrome('/Users/nlohith/Desktop/chromedriver')
                 url=full.get_attribute('href')
                 driver_3.get(url)
                 driver_2.close()


                 a=driver_3.find_elements_by_xpath("//div[@class='col']/div/div[1]/div") #searching for rating path
                 for j in a[2:]:
                     Rating.append(j.text) #appending the elements in Rating list

                 a=driver_3.find_elements_by_xpath("//div[@class='t-ZTKy']/div/div") #searching for review path
                 a=driver_3.find_elements_by_xpath("//div[@class='col']/div/div[1]/div") #searching for rating path
                 for j in a[2:]:
                     Rating.append(j.text) #appending the elements in Rating list

                 a=driver_3.find_elements_by_xpath("//div[@class='t-ZTKy']/div/div") #searching for review path
                 for k in a:
                     Review.append(k.text) #appending the elements in Review list

                 driver_3.close()

             driver_1.close()
```

```
In [45]: df=pd.DataFrame({})
         df['Rating']=Rating
         df['Review']=Review
         df
```

Out[45]:

	Rating	Review
0	5	Outstanding
1	5	Excellent slim laptop with lightweight. Space ...
2	5	Nice product and value for money vate budget l...
3	1	System is very slow please improve it.....
4	5	Nice thank
...	...	...
719	1	It doesn't produce any sound output... Dont bu...
720	5	Fantastic product & have bought it for 14,499 ...
721	5	I LOVES THIS PRODUCT..ITS AMAZING...REALLY TRU...
722	4	Ok good performance
723	5	Great monitor with terrific colors..

724 rows × 2 columns

```
In [46]: df.to_csv('Ratings_Prediction_Data')
```

- The data is that we saved in csv format we imported it using pandas and to further proceed with the pre-processing steps.

```
In [31]: # Importing all the libraries

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import string
import re
from collections import Counter
# packages from gensim
from gensim import corpora
from gensim.utils import simple_preprocess
from gensim.parsing.preprocessing import STOPWORDS

# packages from sklearn
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer

#packages from nltk
import nltk
from nltk.corpus import wordnet, stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer, SnowballStemmer

from nltk import pos_tag

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

```
In [32]: data = pd.read_csv('Ratings_Prediction_Data')
data.head()
```

Out[32]:

	Unnamed: 0	Rating	Review
0	0	5	Outstanding
1	1	5	Excellent slim laptop with lightweight. Space ...
2	2	5	Nice product and value for money vate budget l...
3	3	1	System is very slow please improve it.....
4	4	5	Nice thank

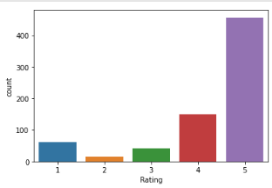
- Then we further checked more about data using info(), shapes using .shape, value counts using value_counts(), null values using .isnull. .sum().sum(), and further visualize it through heatmap as follows:

```
In [33]: data.drop('Unnamed: 0',axis=1,inplace=True)

In [34]: print('Shape of the Dataset is:', data.shape)

Shape of the Dataset is: (724, 2)

In [35]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 724 entries, 0 to 723
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Rating  724 non-null    int64
 1   Review  724 non-null    object
dtypes: int64(1), object(1)
memory usage: 11.4+ KB

In [36]: data.isnull().sum()

Out[36]: Rating    0
Review    0
dtype: int64
```

```
In [37]: #checking null values using heatmap
sns.heatmap(data.isnull())

Out[37]: <matplotlib.axes._subplots.AxesSubplot at 0x19f50b3c310>
```


```
In [38]: print('Number of unique values in Rating :',data['Rating'].nunique())

Number of unique values in Rating : 5
```

```
In [39]: sns.countplot(data['Rating'])
plt.show()
```




```
In [40]: print('Number of unique values in Rating :',data['Rating'].value_counts())

Number of unique values in Rating : 5    457
4    150
1     61
3     41
2     15
Name: Rating, dtype: int64
```

From this we can observe that 63% data has rating 5.

Data Pre-Processing

- In this we will be performing data cleaning such as removing html tags, special characters, converting everything to lowercase, replace email addresses with 'email', URLs with 'webaddress', money symbols with 'moneysymb', 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phonenumber', numbers with 'numbr', whitespace between terms with a single space, removing leading and trailing whitespace and punctuations. We also removed Stopwords.

```
In [41]: # 1. Remove HTML tags
         #Regex rule : '<.*?>'

         def clean(text):
             cleaned = re.compile(r'<.*?>')
             return re.sub(cleaned,'',text) # substring replace with ''(space)

         data.Review = data.Review.apply(clean)
         data.Review

Out[41]: 0                                             Outstanding
         1      Excellent slim laptop with lightweight. Space ...
         2      Nice product and value for money vate budget l...
         3               System is very slow please improve it.....
         4                                              Nice thank
                                      ...
         719    It doesn't produce any sound output... Dont bu...
         720    Fantastic product & have bought it for 14,499 ...
         721    I LOVES THIS PRODUCT..ITS AMAZING...REALLY TRU...
         722                                  Ok good performance
         723                    Great monitor with terrific colors..
         Name: Review, Length: 724, dtype: object

In [42]: # 2. Remove special characters
         def is_special(text):
             rem = ''
             for i in text:
                 if i.isalnum():
                     rem = rem + i
                 else:
                     rem = rem + ' '
             return rem

         data.Review = data.Review.apply(is_special)
         data.Review

Out[42]: 0                                             Outstanding
         1      Excellent slim laptop with lightweight  Space ...
         2      Nice product and value for money vate budget l...
         3               System is very slow please improve it
         4                                              Nice thank
                                      ...
         719    It doesn t produce any sound output    Dont bu...
         720    Fantastic product   have bought it for 14 499 ...
         721    I LOVES THIS PRODUCT  ITS AMAZING   REALLY TRU...
         722                                  Ok good performance
         723                    Great monitor with terrific colors
         Name: Review, Length: 724, dtype: object

In [43]: # 3. Convert everything to lowercase
         def to_lower(text):
             return text.lower()

         data.Review = data.Review.apply(to_lower)
         data.Review

Out[43]: 0                                             outstanding
         1      excellent slim laptop with lightweight  space ...
         2      nice product and value for money vate budget l...
         3               system is very slow please improve it
         4                                              nice thank
                                      ...
         719    it doesn t produce any sound output    dont bu...
         720    fantastic product   have bought it for 14 499 ...
         721    i loves this product  its amazing   really tru...
         722                                  ok good performance
         723                    great monitor with terrific colors
         Name: Review, Length: 724, dtype: object
```

```
In [44]: # Replace email addresses with 'email'
         data['Review'] = data['Review'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$',
                                                  'emailaddress')

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

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

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


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

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

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

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

In [45]: # 4. Remove stopwords
         def rem_stopwords(text):
             stop_words = set(stopwords.words('english'))
             words = word_tokenize(text)
             return [w for w in words if w not in stop_words]

         data.Review = data.Review.apply(rem_stopwords)
         data.Review

Out[45]: 0                                      [outstanding]
         1      [excellent, slim, laptop, lightweight, space, ...
         2      [nice, product, value, money, vate, budget, la...
         3                      [system, slow, please, improve]
```

- Then we performed stemming using Snowball Stemmer and WordNetLemmatizer. Then we further pre-process the text and save the final processed data in processed_review.

```
In [46]: from nltk.stem import SnowballStemmer, WordNetLemmatizer
         stemmer = SnowballStemmer("english")
         import gensim
         def lemmatize_stemming(text):
             return stemmer.stem(WordNetLemmatizer().lemmatize(text,pos='v'))

In [47]: #Tokenize and Lemmatize
         def preprocess(text):
             result=[]
             for token in text:
                 if len(token)>=3:
                     result.append(lemmatize_stemming(token))

             return result

In [48]: # Processing review with above Function
         processed_review = []

         for doc in data.Review:
             processed_review.append(preprocess(doc))

         print(len(processed_review))
         processed_review[:3]

         724

Out[48]: [['outstand'],
          ['excel', 'slim', 'laptop', 'lightweight', 'space', 'save'],
          ['nice', 'product', 'valu', 'money', 'vate', 'budget', 'laptop']]

In [52]: data['Processed_review']=processed_review

In [53]: data
Out[53]:
```

	Rating	Review	Processed_review
0	5	[outstanding]	[outstand]
1	5	[excellent, slim, laptop, lightweight, space, ...	[excel, slim, laptop, lightweight, space, save]
2	5	[nice, product, value, money, vate, budget, la...	[nice, product, valu, money, vate, budget, lap...
3	1	[system, slow, please, improve]	[system, slow, pleas, improv]
4	5	[nice, thank]	[nice, thank]
...	...	...	...
719	1	[produce, sound, output, dont, buy, going, con...	[produc, sound, output, dont, buy, go, connect...
720	5	[fantastic, product, bought, numbr, numbr, rs,...	[fantast, product, buy, numbr, numbr, sale, si...
721	5	[loves, product, amazing, really, true, gaming...	[love, product, amaz, realli, true, game, expe...
722	4	[ok, good, performance]	[good, perform]
723	5	[great, monitor, terrific, colors]	[great, monitor, terrif, color]

724 rows × 3 columns

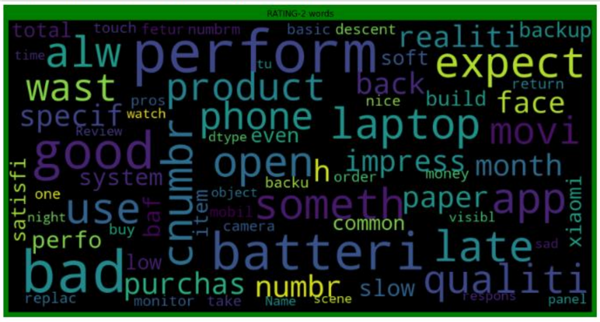
```
In [55]: data['Review'] = data['Processed_review'].apply(lambda x:' '.join(y for y in x))

In [56]: data
Out[56]:
```

	Rating	Review	Processed_review
0	5	outstand	[outstand]
1	5	excel slim laptop lightweight space save	[excel, slim, laptop, lightweight, space, save]
2	5	nice product valu money vate budget laptop	[nice, product, valu, money, vate, budget, lap...
3	1	system slow pleas improv	[system, slow, pleas, improv]
4	5	nice thank	[nice, thank]
...	...	...	...
719	1	produc sound output dont buy go connect audio	[produc, sound, output, dont, buy, go, connect...

- To get sense of loud words in Review corresponding to each rating i., 1, 2, 3, 4 and 5 we have used wordcloud and visualized it as follows:

```
In [60]: Display_wordcloud(data['Review'][data['Rating']==1],"RATING-1")
```

In [61]:
```
Display_wordcloud(data['Review'][data['Rating']==2],"RATING-2")
```



In [62]:
```
Display_wordcloud(data['Review'][data['Rating']==3],"RATING-3")
```

In [63]:
```
Display_wordcloud(data['Review'][data['Rating']==4],"RATING-4")
```

In [64]:
```
Display_wordcloud(data['Review'][data['Rating']==5],"RATING-5")
```

Model/s Development and Evaluation

- Review column have been converted to tokens using TFidfVectorizer. Then using train_test_split we split the data into training and testing dataset.

```
In [65]: from sklearn.feature_extraction.text import TfidfVectorizer

         tf_vec = TfidfVectorizer()
         features = tf_vec.fit_transform(data['Review'])

         X = features
         y = data['Rating']

         print("X.shape = ",X.shape)
         print("y.shape = ",y.shape)

         X.shape =  (724, 1726)
         y.shape =  (724,)

In [66]: # Importing libraries for model training

         from sklearn.linear_model import LogisticRegression
         from sklearn.naive_bayes import MultinomialNB
         from sklearn.tree import DecisionTreeClassifier
         from sklearn.linear_model import LogisticRegression
         from sklearn.neighbors import KNeighborsClassifier

         from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier



         from sklearn.model_selection import cross_val_score, cross_val_predict, train_test_split
         from sklearn.model_selection import GridSearchCV


         # Importing evaluation metrics for model performance....
         from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
         from sklearn.metrics import roc_auc_score, roc_curve, auc
         from sklearn.metrics import precision_score, recall_score, f1_score

In [67]: #splitting the data into training and testing
         x_train,x_test,y_train,y_test=train_test_split(X,y,random_state=42,test_size=0.30,stratify=y)
```

- We have used following algorithms such as: RandomForestClassifier, AdaBoostClassifier, MultinomialNB, DecisionTreeClassifier and KNeighborsClassifier.

```
In [68]: # Creating instances for different Classifiers

         RFC=RandomForestClassifier()
         ADA=AdaBoostClassifier()
         MNB=MultinomialNB()
         DT=DecisionTreeClassifier()
         KNN=KNeighborsClassifier()

In [69]: # List of Models
         models=[]
         models.append(('RandomForestClassifier',RFC))
         models.append(('AdaBoostClassifier',ADA))
         models.append(('MultinomialNB()',MNB))
         models.append(('DecisionTreeClassifier',DT))
         models.append(('KNeighborsClassifier',KNN))
```

- We have formed a loop where all the algorithms will be used one by one and their corresponding accuracy_score, cross_val_score and classification report will be evaluated.

```
In [72]: Model=[]
         score=[]
         cvs=[]

         for name,model in models:
             print('**************',name,'***************')
             print('\n')
             Model.append(name)
             model.fit(x_train,y_train)
             print(model)
             pre=model.predict(x_test)
             print('\n')
             AS=accuracy_score(y_test,pre)
             print('Accuracy_score=',AS)
             score.append(AS*100)
             print('\n')
             sc=cross_val_score(model,X,y,cv=5,scoring='accuracy').mean()
             print('Cross_Val_Score=',sc)
             cvs.append(sc*100)
             print('\n')
             print('classification report\n',classification_report(y_test,pre))
             print('\n')
             cm=confusion_matrix(y_test,pre)
             print(cm)
             print('\n')
             plt.figure(figsize=(10,40))
             plt.subplot(911)
             plt.title(name)
             print(sns.heatmap(cm,annot=True))
             #plt.subplot(912)
             #plt.title(name)
             print('\n\n')
```

- Key metrics used for finalising the model was accuracy_score, cross_val_score. Since in case of RandomForestClassifier it was giving us maximum score among all other models and it's performing well. It's cross_val_score is also satisfactory and it shows that our model is neither underfitting/overfitting.

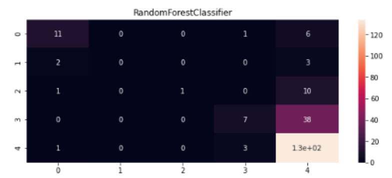
```
************** RandomForestClassifier ***************

RandomForestClassifier()

Accuracy_score= 0.7018348623853211

Cross_Val_Score= 0.6656513409961686

classification report
              precision    recall  f1-score   support

           1       0.73      0.61      0.67        18
           2       0.00      0.00      0.00         5
           3       1.00      0.08      0.15        12
           4       0.64      0.16      0.25        45
           5       0.70      0.97      0.81       138

    accuracy                           0.70       218
   macro avg       0.61      0.36      0.38       218
weighted avg       0.69      0.70      0.63       218


[[ 11   0   0   1   6]
 [  2   0   0   0   3]
 [  1   0   1   0  10]
 [  0   0   0   7  38]
 [  1   0   0   3 134]]
```

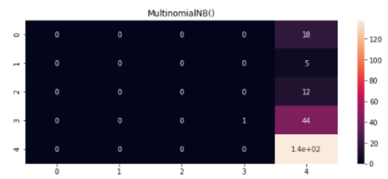
```
************** AdaBoostClassifier ***************

AdaBoostClassifier()

Accuracy_score= 0.6422018348623854

Cross_Val_Score= 0.6395019157088122

classification report
              precision    recall  f1-score   support

           1       0.56      0.28      0.37        18
           2       0.00      0.00      0.00         5
           3       0.00      0.00      0.00        12
           4       0.00      0.00      0.00        45
           5       0.66      0.98      0.79       138

    accuracy                           0.64       218
   macro avg       0.24      0.25      0.23       218
weighted avg       0.46      0.64      0.53       218


[[  5   0   0   0  13]
 [  2   0   0   0   3]
 [  1   0   0   0  11]
 [  1   1   0   0  43]
 [  0   1   0   2 135]]
```

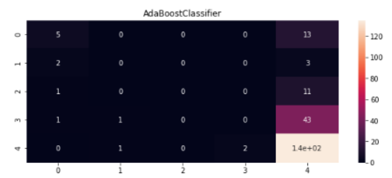
```
************** MultinomialNB() ***************

MultinomialNB()

Accuracy_score= 0.6376146788990825

Cross_Val_Score= 0.6339750957854406

classification report
              precision    recall  f1-score   support

           1       0.00      0.00      0.00        18
           2       0.00      0.00      0.00         5
           3       0.00      0.00      0.00        12
           4       1.00      0.02      0.04        45
           5       0.64      1.00      0.78       138

    accuracy                           0.64       218
   macro avg       0.33      0.20      0.16       218
weighted avg       0.61      0.64      0.50       218


[[  0   0   0   0  18]
 [  0   0   0   0   5]
 [  0   0   0   0  12]
 [  0   0   0   1  44]
 [  0   0   0   0 138]]
```

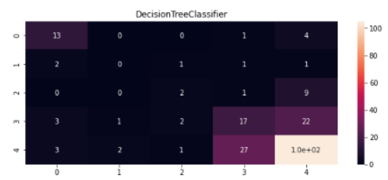
```
*************** DecisionTreeClassifier ***************

DecisionTreeClassifier()

Accuracy_score= 0.6284403669724771

Cross_Val_Score= 0.5868869731800765

classification report
              precision    recall  f1-score   support

           1       0.62      0.72      0.67        18
           2       0.00      0.00      0.00         5
           3       0.33      0.17      0.22        12
           4       0.36      0.38      0.37        45
           5       0.74      0.76      0.75       138

    accuracy                           0.63       218
   macro avg       0.41      0.41      0.40       218
weighted avg       0.62      0.63      0.62       218


[[ 13   0   0   1   4]
 [  2   0   1   1   1]
 [  0   0   2   1   9]
 [  3   1   2  17  22]
 [  3   2   1  27 105]]
```

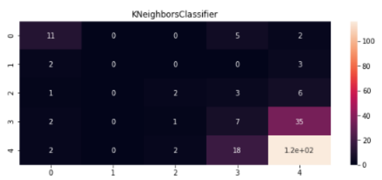
```
*************** KNeighborsClassifier ***************

KNeighborsClassifier()

Accuracy_score= 0.6238532110091743

Cross_Val_Score= 0.5551819923371648

classification report
              precision    recall  f1-score   support

           1       0.61      0.61      0.61        18
           2       0.00      0.00      0.00         5
           3       0.40      0.17      0.24        12
           4       0.21      0.16      0.18        45
           5       0.72      0.84      0.77       138

    accuracy                           0.62       218
   macro avg       0.39      0.35      0.36       218
weighted avg       0.57      0.62      0.59       218


[[ 11   0   0   5   2]
 [  2   0   0   0   3]
 [  1   0   2   3   6]
 [  2   0   1   7  35]
 [  2   0   2  18 116]]
```











Model Summary:

In [74]:
```
result=pd.DataFrame({'Model': Model,'Score': score,'Cross_Val_Score':cvs})
result
```

Out[74]:

	Model	Score	Cross_Val_Score
0	RandomForestClassifier	70.183486	66.565134
1	AdaBoostClassifier	64.220183	63.950192
2	MultinomialNB()	63.761468	63.397510
3	DecisionTreeClassifier	62.844037	58.688697
4	KNeighborsClassifier	62.385321	55.518199

- Since RandomForestClassifier was giving us maximum score we applied hyper parameter tuning on it to further improve the result using GridSearchCV.

```
In [76]: from sklearn.ensemble import RandomForestClassifier
         from sklearn.model_selection import GridSearchCV
         rf=RandomForestClassifier()
         parameters={'n_estimators':[50,100,150,200,250,300,350,400,450,500],'max_depth':[5, 8, 15, 25, 30]}
         clf=GridSearchCV(rf,parameters,cv=5)
         clf.fit(X,y)
         clf.best_params_

Out[76]: {'max_depth': 30, 'n_estimators': 200}
```

```
In [77]: rf=RandomForestClassifier(n_estimators=200,max_depth=30)
         rf.fit(x_train,y_train)
         predrf=rf.predict(x_test)
         print(accuracy_score(y_test,predrf))
         print(confusion_matrix(y_test,predrf))
         print(classification_report(y_test,predrf))

0.6834862385321101
[[  8   0   0   0  10]
 [  2   0   0   0   3]
 [  0   0   0   0  12]
 [  0   0   0   4  41]
 [  0   0   0   1 137]]
              precision    recall  f1-score   support

           1       0.80      0.44      0.57        18
           2       0.00      0.00      0.00         5
           3       0.00      0.00      0.00        12
           4       0.80      0.09      0.16        45
           5       0.67      0.99      0.80       138

    accuracy                           0.68       218
   macro avg       0.45      0.31      0.31       218
weighted avg       0.66      0.68      0.59       218
```

- Since it didn't show any improvement in result even after hyperparameter tuning. Hence, we will be choosing the RandomForestClassifier Model with default values that we got earlier.

- We will further use that model to predict our test data set as follows:

```
In [79]: test_data=pd.DataFrame(data=y_test,)
         test_data['Predicted values']=Predrf

         test_data.to_csv('Ratings_Predict.csv')

         test_data

Out[79]:
```

	Rating	Predicted values
266	2	1
686	5	5
272	1	1
448	5	5
692	5	5
...	...	...
720	5	5
442	5	5
618	5	5
97	5	5
446	5	5

218 rows × 2 columns

```
In [80]: # Creating Pickle File
         import joblib
         joblib.dump(RF,'Ratings_Prediction_dataset.pkl')

Out[80]: ['Ratings_Prediction_dataset.pkl']
```

<u>CONCLUSION</u>

- We have got the accuracy score of 70% using RandomForestClassifier. Hence, our model is working well.
- Since the data was quite imbalanced as it contained more than 60% ratings as 5.
- We had less data.
- I have included one more additional work folder in the zip file where I have scraped data from amazon as well. Hence it contains data from two websites i.e., amazon and flipcart.