FLIP ROBO

Ratings Prediction

Submitted by:
Himanshu Soni

ACKNOWLEDGMENT

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

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

- https://towardsdatascience.com/a-common-mistake-to-avoid-when-encoding-ordinal-features-79e402796ab4
- https://stackoverflow.com/questions/43590489/gridsearchcv-random-forest-regressor-tuning-best-params
- https://www.codegrepper.com/code-examples/delphi/scikit+pca+preserve+column+names+pca+pipeline
- https://stackoverflow.com/questions/22984335/recovering-features-names-of-explained-variance-ratio-in-pca-with-sklearn

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

INTRODUCTION

Business Problem Framing

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

Conceptual Background of the Domain Problem

Ratings Prediction

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

Data Collection Phase –

You have to scrape at least 20000 rows of data. You can scrape more data as well, it's up to you. More the data better the model In this section you need to scrape the reviews of different laptops, Phones, Headphones, smart watches, Professional Cameras, Printers, monitors, Home theater, router from different e-commerce websites. Basically, we need these columns-

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

- Try to fetch an equal number of reviews for each rating, for example if you are fetching 10000 reviews then all ratings 1,2,3,4,5 should be 2000. It will balance our data set.
- Convert all the ratings to their round number, as there are only 5 options for rating i.e., 1,2,3,4,5. If a rating is 4.5 convert it 5.

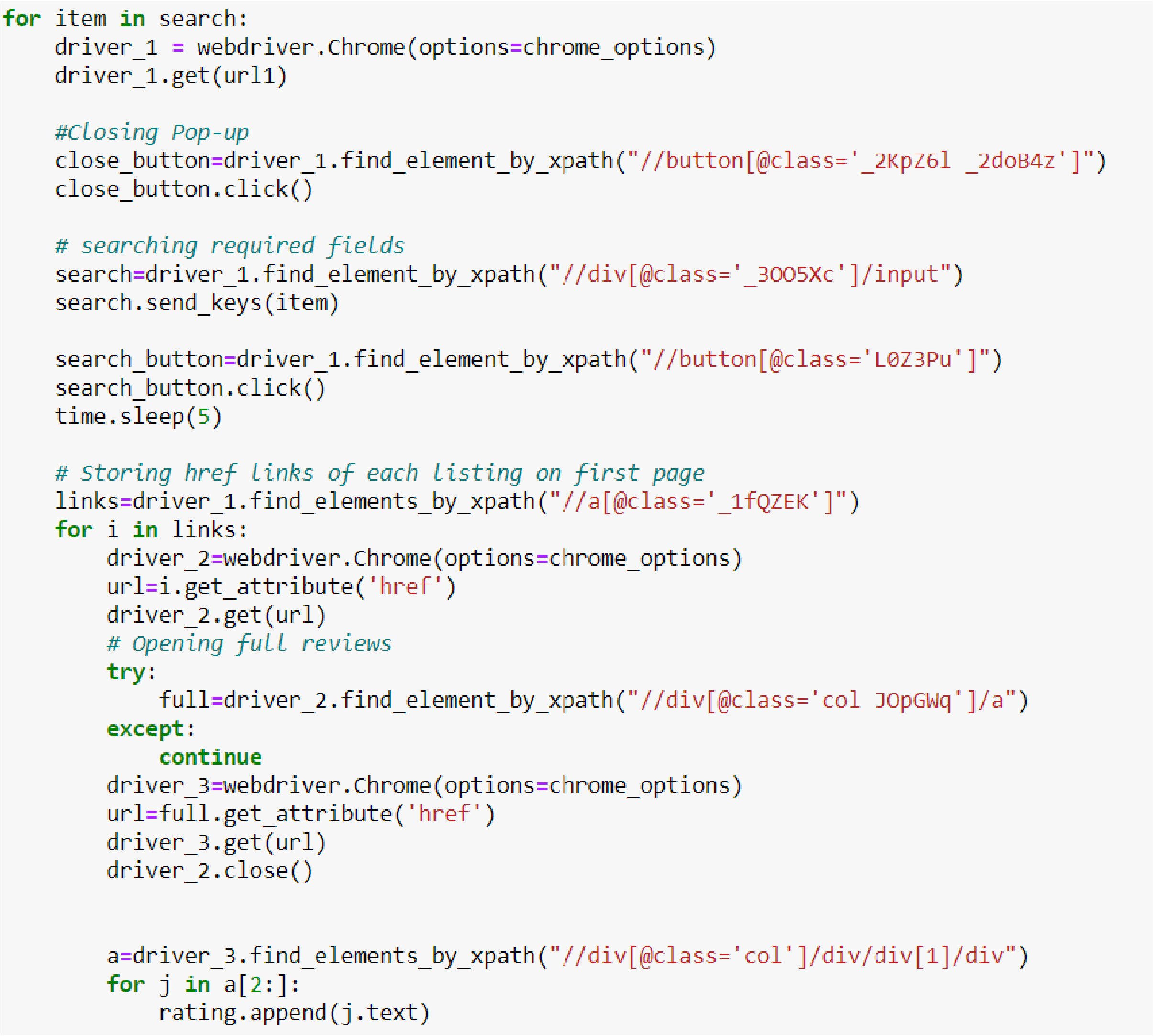
Model Building Phase

After collecting the data, you need to build a machine learning model. Before model building do all data preprocessing steps involving NLP. Try different models with different hyper parameters and select the best model. Follow the complete life cycle of data science. Include all the steps like-

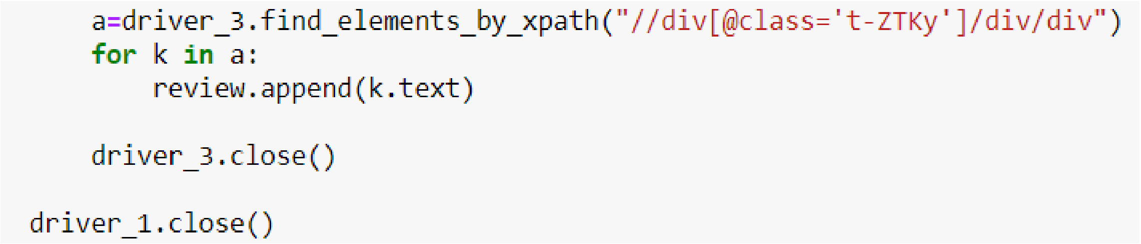
1. Data Cleaning
2. Exploratory Data Analysis
3. Data Preprocessing
4. Model Building
5. Model Evaluation
6. Selecting the best model

Part 1 - Extraction

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



```
for item in search:
    driver_1 = webdriver.Chrome(options=chrome_options)
    driver_1.get(url1)

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

    # searching required fields
    search=driver_1.find_element_by_xpath("//div[@class='_3OO5Xc']/input")
    search.send_keys(item)

    search_button=driver_1.find_element_by_xpath("//button[@class='L0Z3Pu']")
    search_button.click()
    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(options=chrome_options)
        url=i.get_attribute('href')
        driver_2.get(url)
        # Opening full reviews
        try:
            full=driver_2.find_element_by_xpath("//div[@class='col JOpGWq']/a")
        except:
            continue
        driver_3=webdriver.Chrome(options=chrome_options)
        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")
        for j in a[2:]:
            rating.append(j.text)
```



```
   a=driver_3.find_elements_by_xpath("//div[@class='t-ZTKy']/div/div")
   for k in a:
       review.append(k.text)

   driver_3.close()

 driver_1.close()
```

Sample Data Collected

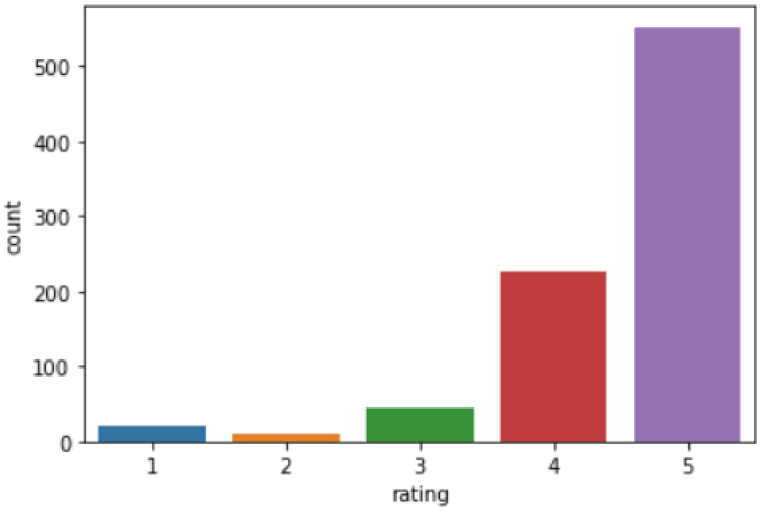
	rating	review
0	5	Everything is nice about this product as menti...
1	5	Its an amazing experience to work on this lapt...
2	5	I did research on laptops for past 2 weeks and...
3	5	First Of All, I am writing this review after u...
4	5	All as described. Finally a speedy experience ...

Part 2 - Modelling

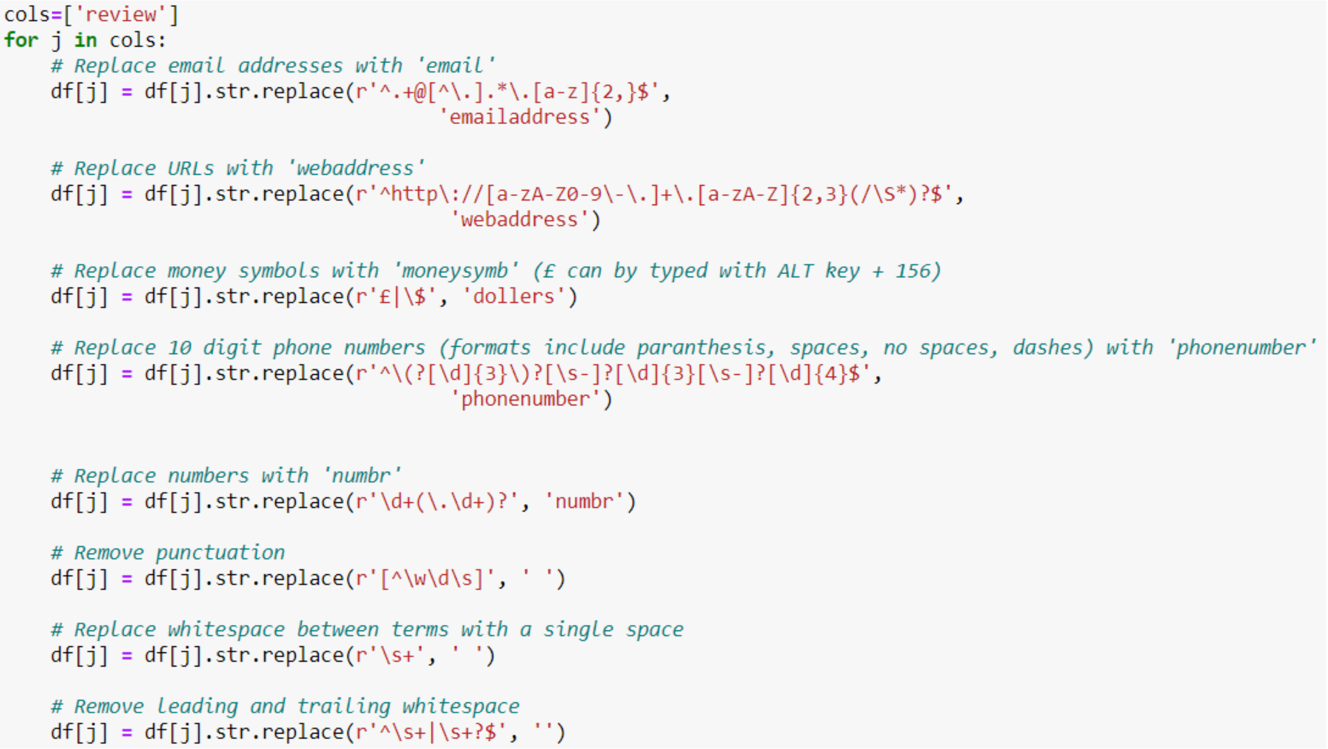
Pre-processing

```
# Let's see how our Target column is distributed
import seaborn as sns
sns.countplot('rating',data=df)
```

```
<AxesSubplot:xlabel='rating', ylabel='count'>
```

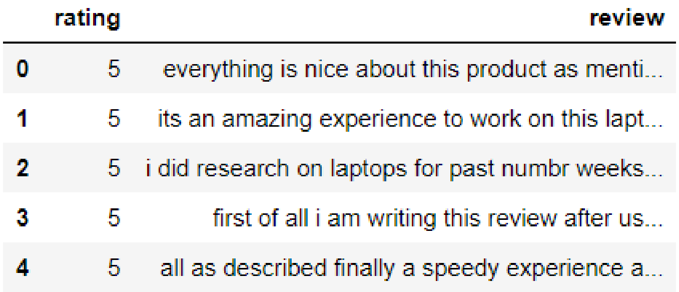


Pre-Processing Steps:

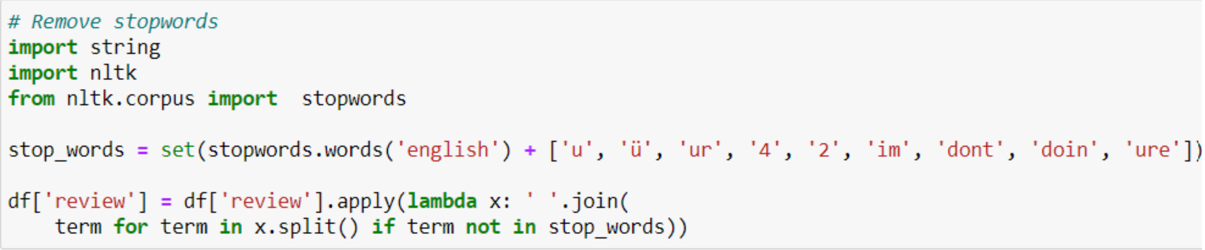


```
cols=['review']
for j in cols:
    # Replace email addresses with 'email'
    df[j] = df[j].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$',
                                     'emailaddress')

    # Replace URLs with 'webaddress'
    df[j] = df[j].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)
    df[j] = df[j].str.replace(r'£|\$', 'dollers')

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


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

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

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

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

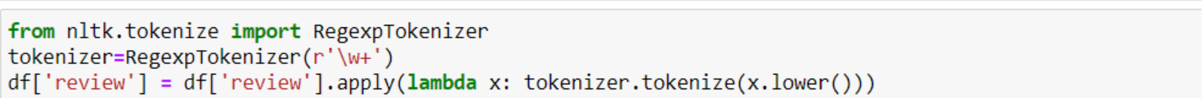
Sample Data after Pre-Processing:



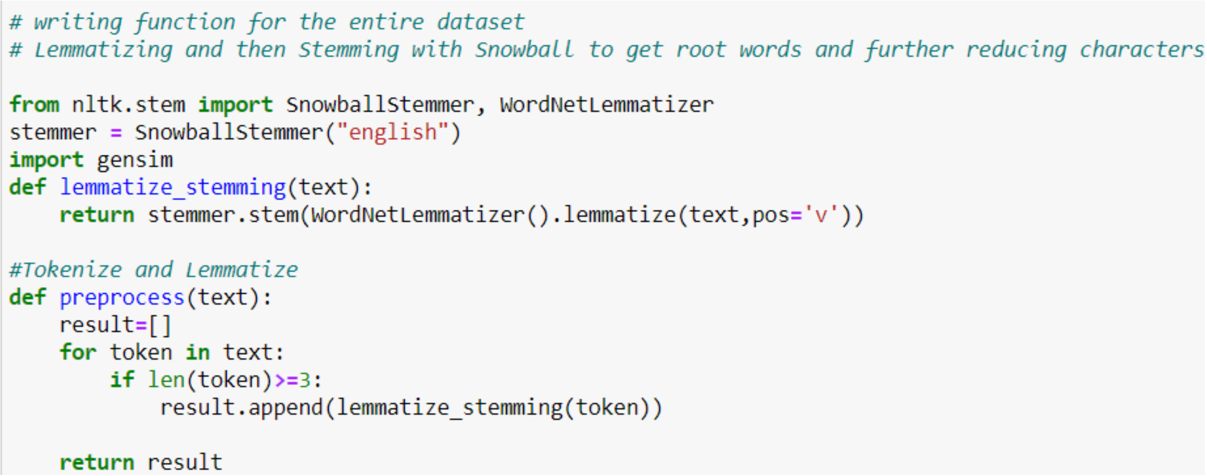
	rating	review
0	5	everything is nice about this product as menti...
1	5	its an amazing experience to work on this lapt...
2	5	i did research on laptops for past numbr weeks...
3	5	first of all i am writing this review after us...
4	5	all as described finally a speedy experience a...



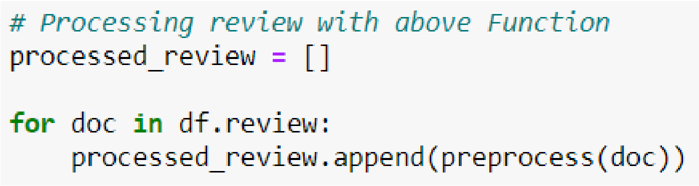
```
# Remove stopwords
import string
import nltk
from nltk.corpus import  stopwords

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

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



```
from nltk.tokenize import RegexpTokenizer
tokenizer=RegexpTokenizer(r'\w+')
df['review'] = df['review'].apply(lambda x: tokenizer.tokenize(x.lower()))
```



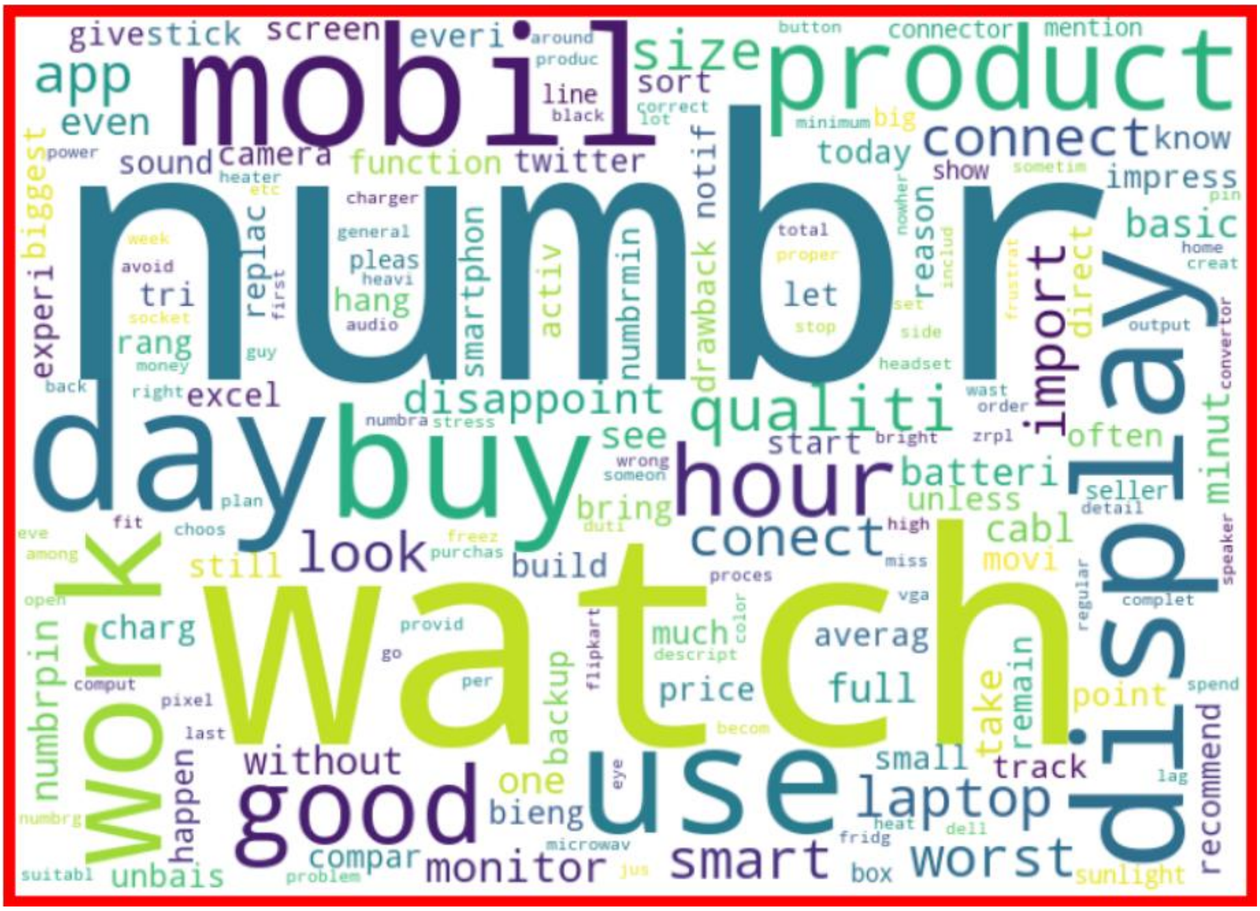
```
# writing function for the entire dataset
# Lemmatizing and then Stemming with Snowball to get root words and further reducing characters

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

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

    return result
```



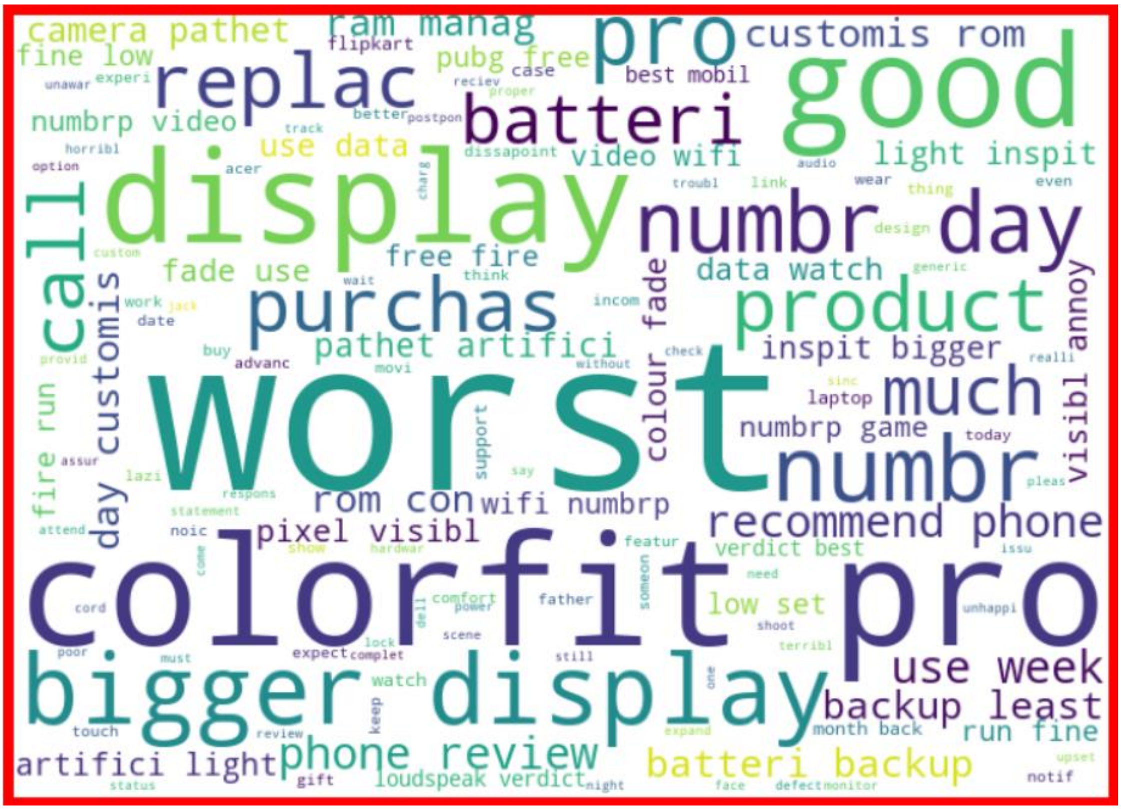
```
# Processing review with above Function
processed_review = []

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

Word Cloud for various Ratings:

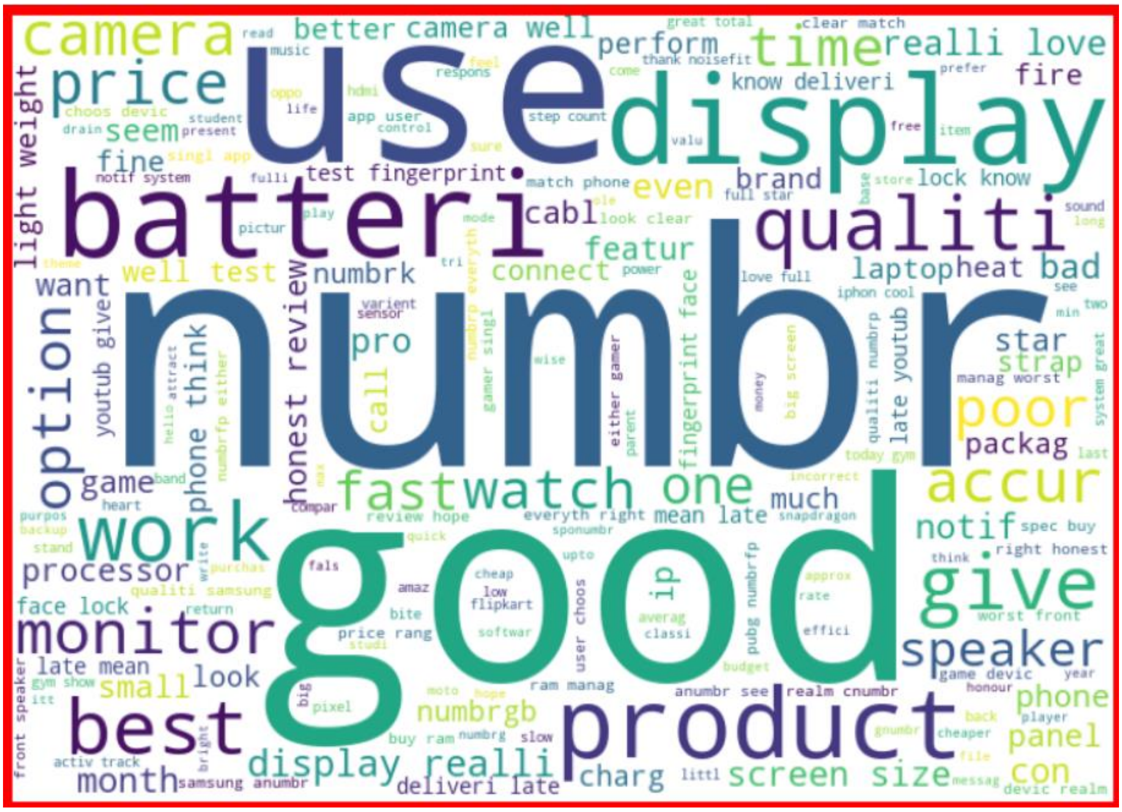
Rating=1:



Rating=2:



Rating=3:



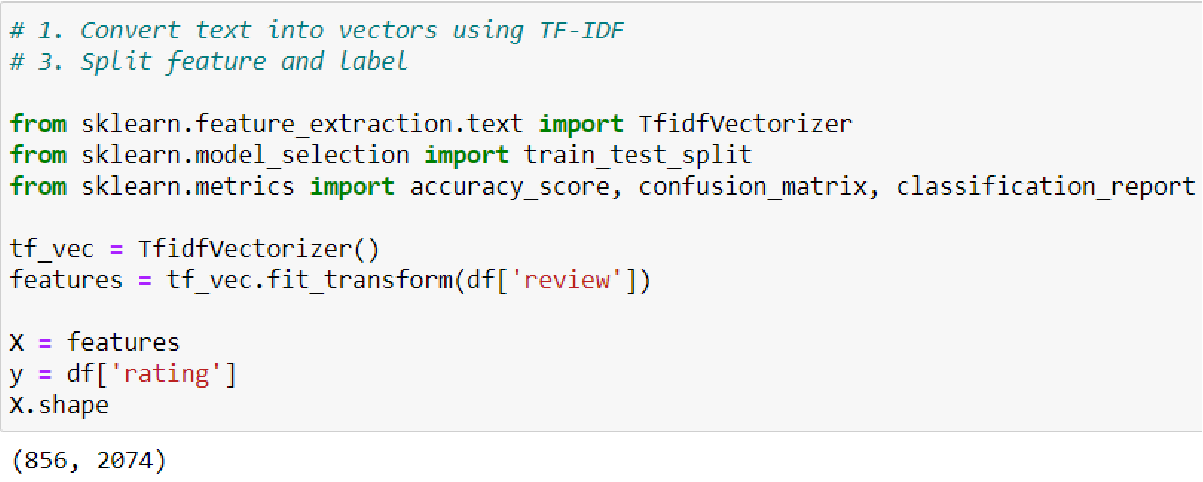
Rating=4:

numbr
numbr numbr
good
laptop
batteri
qualiti
watch
display
phone
monitor
look
use
camera
work
game
product
perform
price
featur
nice
time
batteri life
screen
best
great
give
buy
better
perfect
design
smart watch
valu money
day
decent
perform good
good product
speaker
purchas
littl
sound
expect
smooth
con numbr
pretti
fine
thank flipkart
recommend
build qualiti
much
pro
feel
keyboard
realli good
awesom
thing
review
overal
budget
speed
test
one
charg
cool
bright
chang
happi
sleep
numbrk
realli

Rating=5:

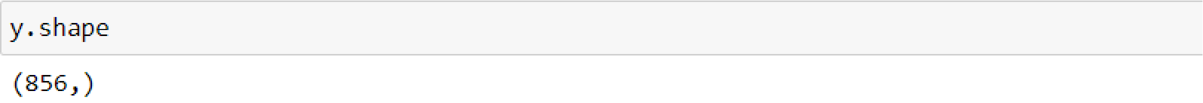
good
numbr
display
laptop
phone
watch
monitor
perform
smart watch
product
numbr numbr
camera
qualiti
best
price
awesom
nice
work
one
realli
better
game
look
give
buy
great
screen
time
smooth
numbr day
mobil
amaz
deliveri
colour
heart rate
price rang
charg
use
design
batteri
batteri life
day
color
video
featur
love
perfect
overal
excel
speaker
fast charg
view angl
pretti good
good batteri
watch price
sponumbr
valu money
batteri backup
thank flipkart
cool
build qualiti
realli good
decent
purchas
bright
averag
thank
sound
super
experi
offer
issu
processor
hdmi
everyth
side
high
need
without
power
offic
mode
redmi
much
realm
boot
numbrk
review
happi
low
recommend
ram
superb
worth
keep
ssd
accur
wonder
numbrhz
best budget
connect
say
enough
updat
rang
make
well
pretti
expect
excel product
price point
lot
nice product
less
think
con
numbr hour
accuraci
first
pleas
satisfi
set
audio
packag
even
light weight

Before Model Building, Vectorizing the dataset



```
# 1. Convert text into vectors using TF-IDF
# 3. Split feature and label

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

tf_vec = TfidfVectorizer()
features = tf_vec.fit_transform(df['review'])

X = features
y = df['rating']
X.shape
```

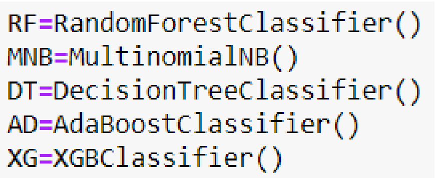
```
(856, 2074)
```



```
y.shape
```

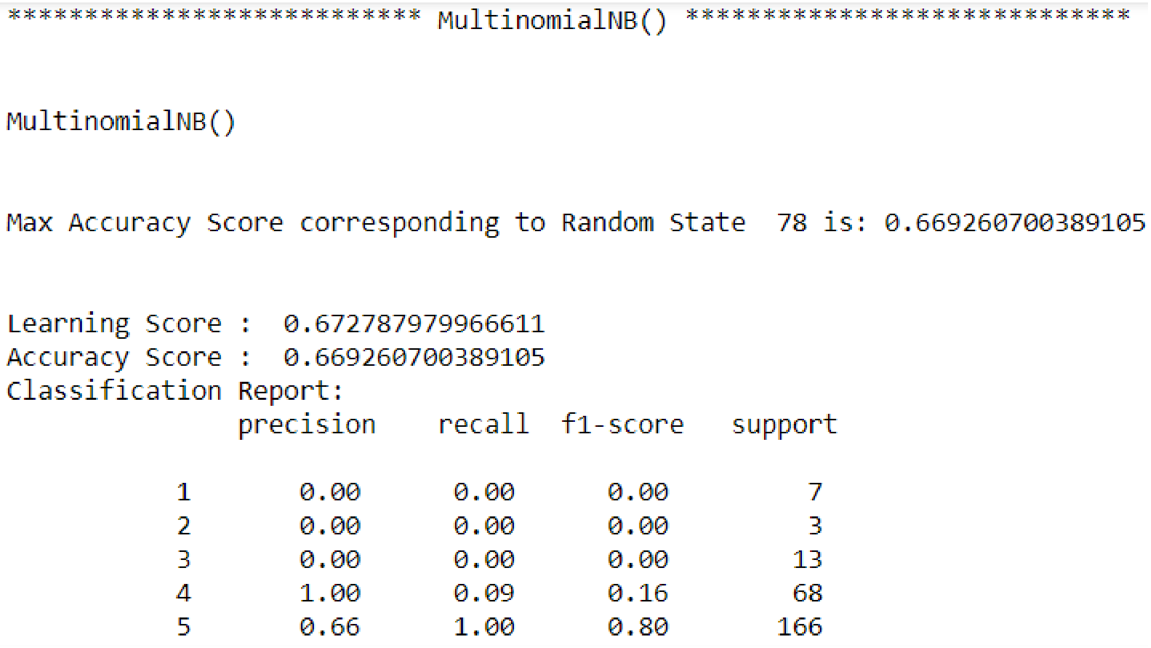
```
(856,)
```

List of Models used:

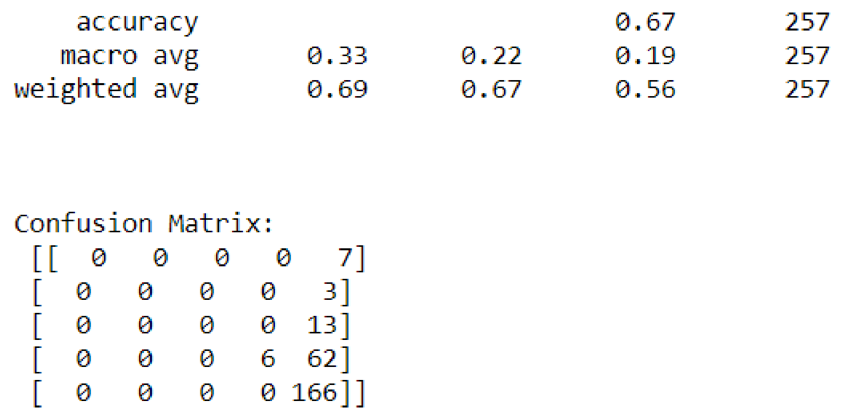


```
RF=RandomForestClassifier()
MNB=MultinomialNB()
DT=DecisionTreeClassifier()
AD=AdaBoostClassifier()
XG=XGBClassifier()
```

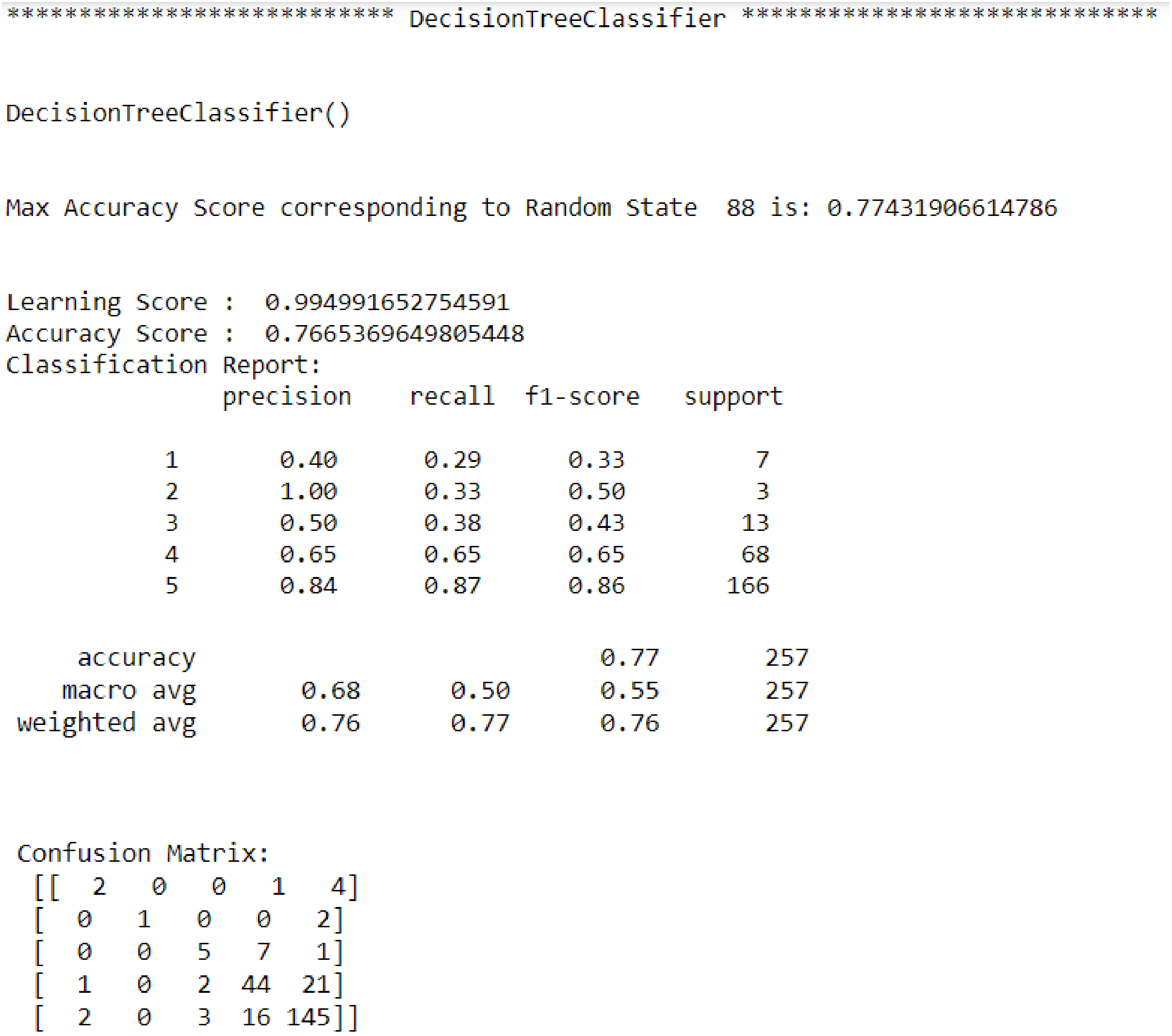
Model Performances:



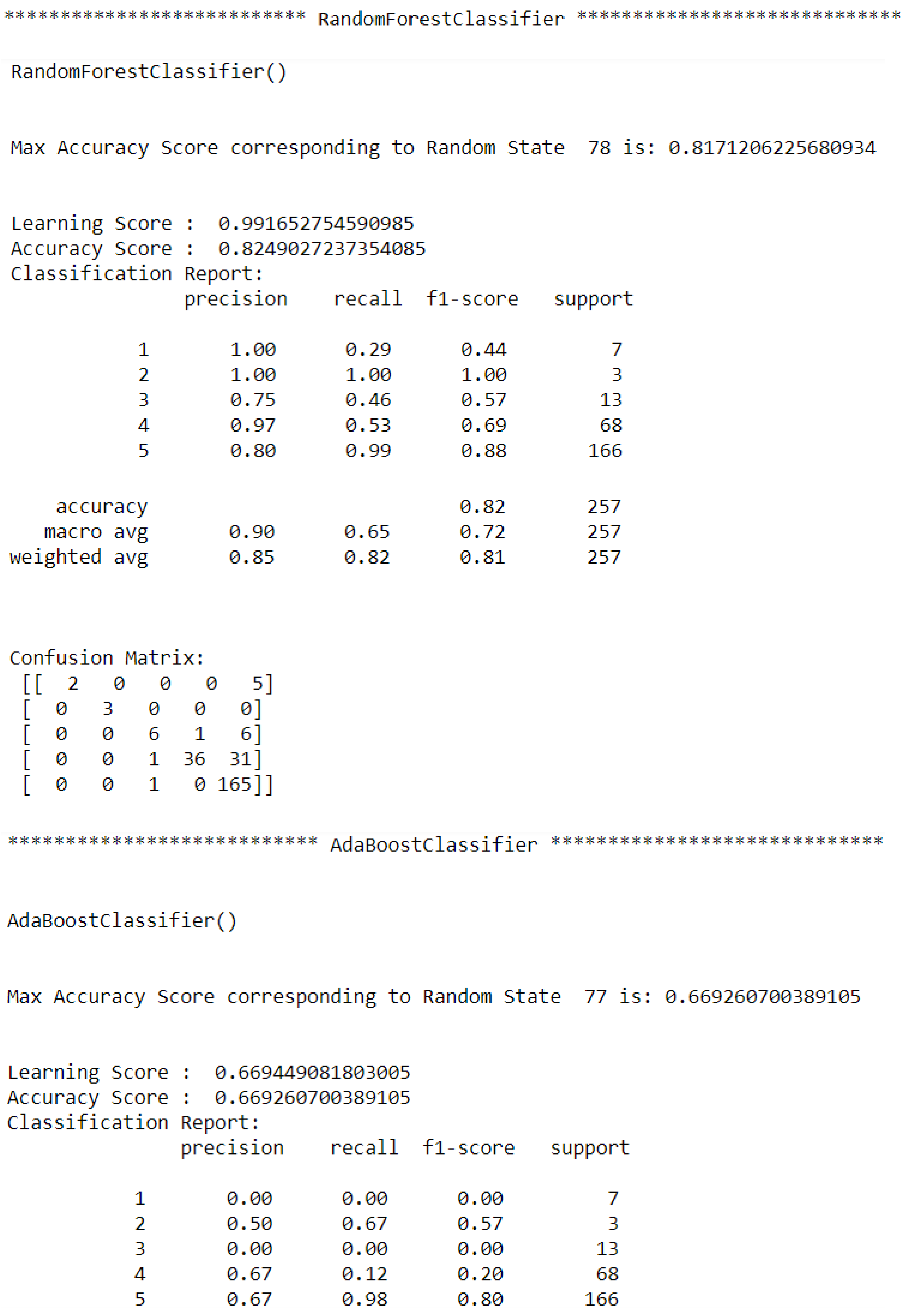
```
*************************** MultinomialNB() *****************************


MultinomialNB()


Max Accuracy Score corresponding to Random State  78 is: 0.669260700389105


Learning Score :  0.672787979966611
Accuracy Score :  0.669260700389105
Classification Report:
               precision    recall  f1-score   support

           1       0.00      0.00      0.00         7
           2       0.00      0.00      0.00         3
           3       0.00      0.00      0.00        13
           4       1.00      0.09      0.16        68
           5       0.66      1.00      0.80       166
```



```
    accuracy                           0.67       257
   macro avg       0.33      0.22      0.19       257
weighted avg       0.69      0.67      0.56       257



Confusion Matrix:
 [[  0   0   0   0   7]
 [  0   0   0   0   3]
 [  0   0   0   0  13]
 [  0   0   0   6  62]
 [  0   0   0   0 166]]
```



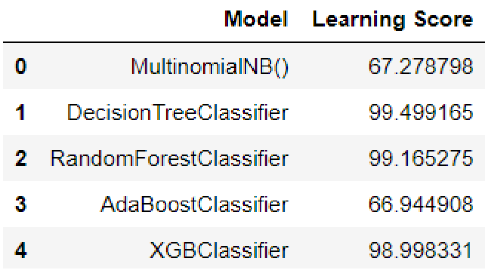
```
*************************** DecisionTreeClassifier *****************************


DecisionTreeClassifier()


Max Accuracy Score corresponding to Random State  88 is: 0.77431906614786


Learning Score :  0.994991652754591
Accuracy Score :  0.7665369649805448
Classification Report:
               precision    recall  f1-score   support

           1       0.40      0.29      0.33         7
           2       1.00      0.33      0.50         3
           3       0.50      0.38      0.43        13
           4       0.65      0.65      0.65        68
           5       0.84      0.87      0.86       166

    accuracy                           0.77       257
   macro avg       0.68      0.50      0.55       257
weighted avg       0.76      0.77      0.76       257



Confusion Matrix:
 [[  2   0   0   1   4]
 [  0   1   0   0   2]
 [  0   0   5   7   1]
 [  1   0   2  44  21]
 [  2   0   3  16 145]]
```



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

RandomForestClassifier()


Max Accuracy Score corresponding to Random State  78 is: 0.8171206225680934


Learning Score :  0.991652754590985
Accuracy Score :  0.8249027237354085
Classification Report:
               precision    recall  f1-score   support

           1       1.00      0.29      0.44         7
           2       1.00      1.00      1.00         3
           3       0.75      0.46      0.57        13
           4       0.97      0.53      0.69        68
           5       0.80      0.99      0.88       166

    accuracy                           0.82       257
   macro avg       0.90      0.65      0.72       257
weighted avg       0.85      0.82      0.81       257



Confusion Matrix:
 [[  2   0   0   0   5]
 [  0   3   0   0   0]
 [  0   0   6   1   6]
 [  0   0   1  36  31]
 [  0   0   1   0 165]]

**************************** AdaBoostClassifier ******************************


AdaBoostClassifier()


Max Accuracy Score corresponding to Random State  77 is: 0.669260700389105


Learning Score :  0.669449081803005
Accuracy Score :  0.669260700389105
Classification Report:
               precision    recall  f1-score   support

           1       0.00      0.00      0.00         7
           2       0.50      0.67      0.57         3
           3       0.00      0.00      0.00        13
           4       0.67      0.12      0.20        68
           5       0.67      0.98      0.80       166
    accuracy                           0.67       257
   macro avg       0.37      0.35      0.31       257
weighted avg       0.62      0.67      0.57       257



Confusion Matrix:
 [[  0   0   0   0   7]
 [  0   2   0   0   1]
 [  0   0   0   1  12]
 [  0   1   0   8  59]
 [  0   1   0   3 162]]

 *************************** XGBClassifier *****************************

Learning Score :  0.989983305509182
Accuracy Score :  0.8210116731517509
Classification Report:
              precision    recall  f1-score   support

           1       1.00      0.43      0.60         7
           2       0.75      1.00      0.86         3
           3       1.00      0.46      0.63        13
           4       0.79      0.60      0.68        68
           5       0.82      0.95      0.88       166

    accuracy                           0.82       257
   macro avg       0.87      0.69      0.73       257
weighted avg       0.83      0.82      0.81       257

     accuracy                           0.82       257
    macro avg       0.87      0.69      0.73       257
 weighted avg       0.83      0.82      0.81       257


 Confusion Matrix:
  [[  3   0   0   0   4]
  [  0   3   0   0   0]
  [  0   0   6   4   3]
  [  0   0   0  41  27]
  [  0   1   0   7 158]]
```

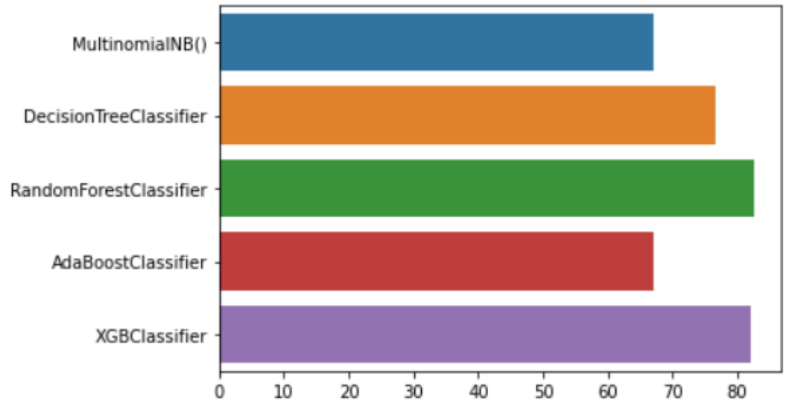
Model Comparisons:

Learning Scores



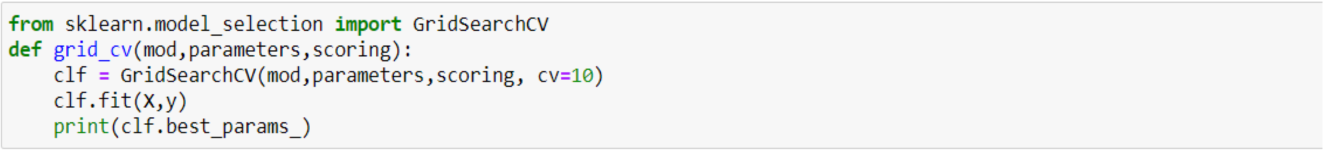
	Model	Learning Score
0	MultinomialNB()	67.278798
1	DecisionTreeClassifier	99.499165
2	RandomForestClassifier	99.165275
3	AdaBoostClassifier	66.944908
4	XGBClassifier	98.998331

Accuracy

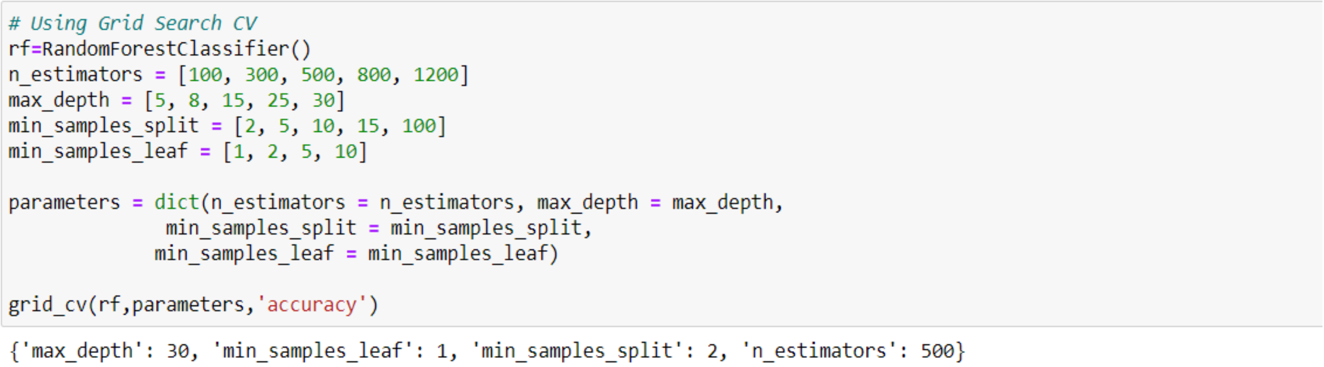


Random Forest Classifier gives best results. Hyperparameter Tuning

I have used Grid Search CV

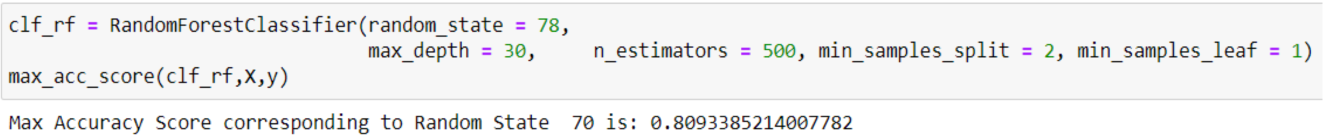


```
from sklearn.model_selection import GridSearchCV
def grid_cv(mod,parameters,scoring):
    clf = GridSearchCV(mod,parameters,scoring, cv=10)
    clf.fit(X,y)
    print(clf.best_params_)
```



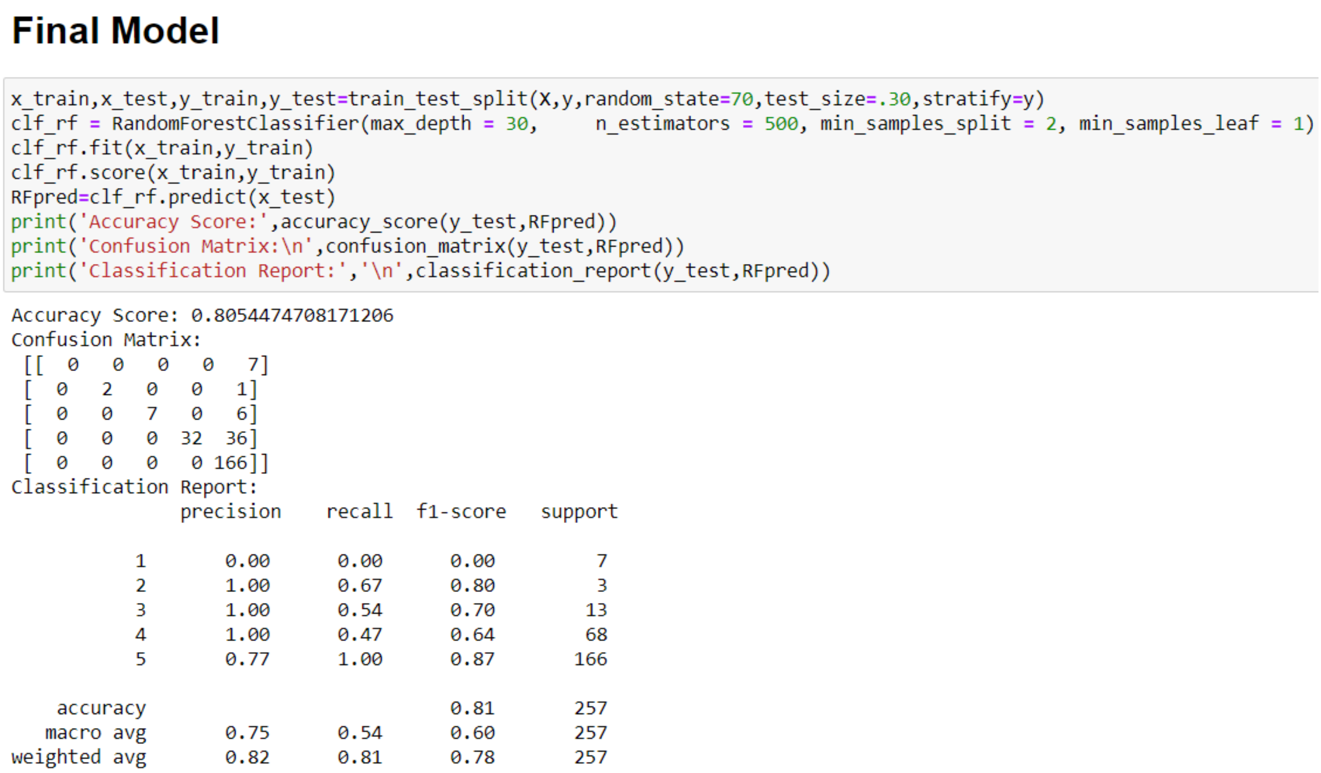
```
# Using Grid Search CV
rf=RandomForestClassifier()
n_estimators = [100, 300, 500, 800, 1200]
max_depth = [5, 8, 15, 25, 30]
min_samples_split = [2, 5, 10, 15, 100]
min_samples_leaf = [1, 2, 5, 10]

parameters = dict(n_estimators = n_estimators, max_depth = max_depth,
              min_samples_split = min_samples_split,
             min_samples_leaf = min_samples_leaf)

grid_cv(rf,parameters,'accuracy')
```

```
{'max_depth': 30, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 500}
```



```
clf_rf = RandomForestClassifier(random_state = 78,
                                max_depth = 30,      n_estimators = 500, min_samples_split = 2, min_samples_leaf = 1)
max_acc_score(clf_rf,X,y)
```

```
Max Accuracy Score corresponding to Random State  70 is: 0.8093385214007782
```

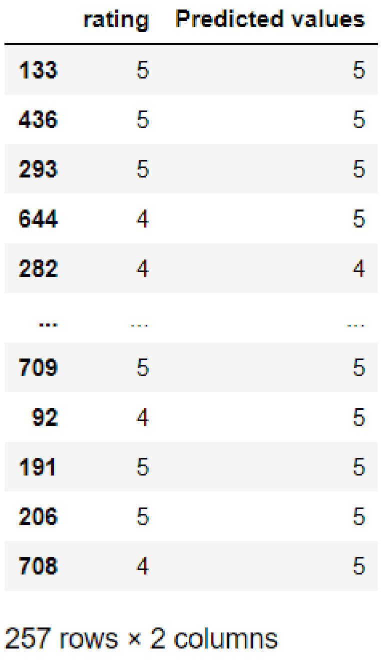


Final Model

```
x_train,x_test,y_train,y_test=train_test_split(X,y,random_state=70,test_size=.30,stratify=y)
clf_rf = RandomForestClassifier(max_depth = 30,     n_estimators = 500, min_samples_split = 2, min_samples_leaf = 1)
clf_rf.fit(x_train,y_train)
clf_rf.score(x_train,y_train)
RFpred=clf_rf.predict(x_test)
print('Accuracy Score:',accuracy_score(y_test,RFpred))
print('Confusion Matrix:\n',confusion_matrix(y_test,RFpred))
print('Classification Report:','\n',classification_report(y_test,RFpred))
```

```
Accuracy Score: 0.8054474708171206
Confusion Matrix:
 [[  0   0   0   0   7]
 [  0   2   0   0   1]
 [  0   0   7   0   6]
 [  0   0   0  32  36]
 [  0   0   0   0 166]]
Classification Report:
               precision    recall  f1-score   support

           1       0.00      0.00      0.00         7
           2       1.00      0.67      0.80         3
           3       1.00      0.54      0.70        13
           4       1.00      0.47      0.64        68
           5       0.77      1.00      0.87       166

    accuracy                           0.81       257
   macro avg       0.75      0.54      0.60       257
weighted avg       0.82      0.81      0.78       257
```

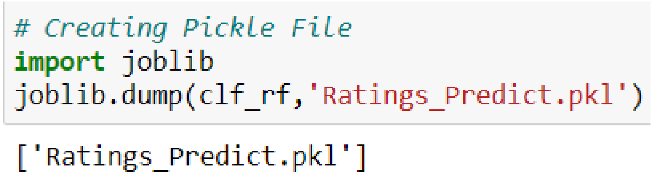
Sample Predictions:



	rating	Predicted values
133	5	5
436	5	5
293	5	5
644	4	5
282	4	4
...	...	...
709	5	5
92	4	5
191	5	5
206	5	5
708	4	5

257 rows × 2 columns

Saving Model as pkl file



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

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

Conclusions and Scope of Improvements:

- We have got a good accuracy score of more than 80 using Random Forest Classifier.

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

- As my data was limited, I did not go for under-sampling, but it was needed as not all classes have equal percentage of data for model training. Ideally, I could have reduced the number of data points having Rating = 5, as that contributes to about 35-40% of the data.

- More data needed for Ratings 1 and 2.