Ratings Prediction

Submitted by:

Amruta Shah

ACKNOWLEDGMENT

A unique opportunity like this comes very rarely. It is indeed a pleasure for me to have worked on this project.

The satisfaction that accompanies the successful completion of this project is incomplete without the mention of the people and references whose guidance has made it possible for me to complete this project.

I am grateful to my internship company **Flip Robo Technologies** with its ideals and inspiration for providing me with facilities that has made this project a success.

Also, I am grateful to my institute “Data Trained” for providing the opportunity to work as an intern in “Flip Robo Technologies” and giving me the great knowledge to complete this project.

I like to acknowledge the effort put in by our SME Khushboo Garg helping me understand the relevant concepts related to Data pre-processing and providing guidance when necessary.

Also based on below references I am able to encourage my knowledge time to time to improve my knowledge.

References:

- DecisionTreeClassifier
- Hyper-parameter Tuning
- SMOTE
- Getting started with XGBoost
- scikit learn

INTRODUCTION

- ## Business Problem Framing

  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.

- ## Conceptual Background of the Domain Problem

  This project can very useful for all eCommerce customers to grow their online business and also to improve their product quality as well as other performance by pre prediction.
  The ability to successfully decide whether a review will be helpful to other customers and thus give the product more exposure is vital to companies that support these reviews, companies like Flipkart, Amazon etc.

- ## Review of Literature

  There are two main methods to approach this problem. Frist one is Data Collection that is you have to scrape the reviews of different laptops, Phones, Headphones, smart watches, Professional Cameras, Printers, Monitors, Home theatre, Router from different eCommerce websites. Basically, we need these columns 1) reviews of the product and 2) rating of the product. And 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.
  Second one is Model Building phase that means after collecting the data, we need to build a machine learning model. Before model building we have to do all data pre-processing steps. With different

models with different hyper parameters and select the best model. By following 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 mode

- Motivation for the Problem Undertaken

  Motivation behind this project is that mainly focuses to prove that combining formerly known data about each user's similarity to other users, with the sentiment analysis of the review text itself, will help us improve the model prediction of what rating the user's review will get.

Analytical Problem Framing

- Mathematical/ Analytical Modelling of the Problem

  Mathematical Summary:
  Dimensions of Dataset: There are 3 columns and 22048 rows in this dataset.
  Null Values: There are null values in this dataset.
  Skewness: Skewness is present in column.
  Statistical Summary: Standard deviation is very high in most of the columns.
  Special Character are present in both dataset (train and test).

- Data Sources and their formats

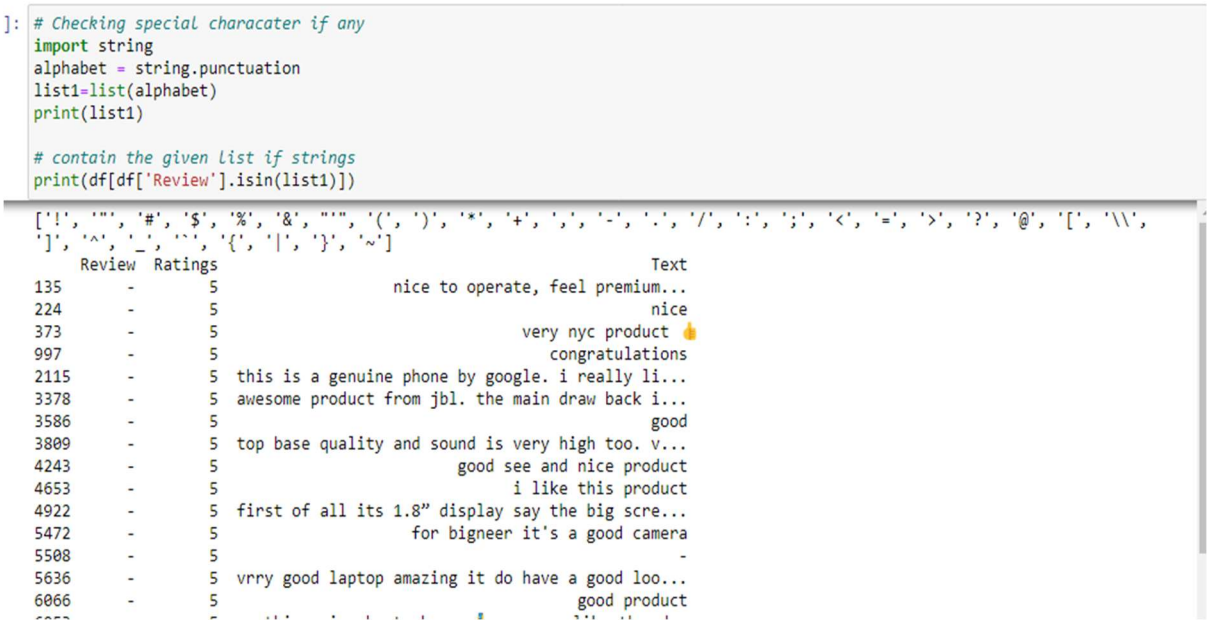
  Data Sources: I have scrapped the train & test data from the different data sources like Amazon, Flipkart etc. In order to improve the ecommerce services , the client wants some predictions that

could help them in further investment and improvement in services of ecommerce sites . This project is multiclass classification type as it has 5 type of class which we have to predict like Rating 1,2,3,4,5 that indicates Best, Good, Average, worst & very worst rating.

Data Formats: Text & Review columns are object datatype which we need to encode & pre-process.

- Data Pre-processing Done

  First of all, I have dropped irrelevant columns 'unknown:0'.
  Then check for the duplicates some duplicates are found which is delated. After dropping the duplicated the shape of dataset is (rows=18174, columns=3) from both train and test dataset.
  Moving further as we have special characters in columns so I have removed from both the columns as shown below.



```
]: # Checking special characater if any
import string
alphabet = string.punctuation
list1=list(alphabet)
print(list1)

# contain the given list if strings
print(df[df['Review'].isin(list1)])
```

```
['!', '"', '#', '$', '%', '&', "'", '(', ')', '*', '+', ',', '-', '.', '/', ':', ';', '<', '=', '>', '?', '@', '[', '\\',
']', '^', '_', '`', '{', '|', '}', '~']
     Review  Ratings                                               Text
135       -        5                  nice to operate, feel premium...
224       -        5                                               nice
373       -        5                                  very nyc product 👍
997       -        5                                     congratulations
2115      -        5  this is a genuine phone by google. i really li...
3378      -        5  awesome product from jbl. the main draw back i...
3586      -        5                                               good
3809      -        5  top base quality and sound is very high too. v...
4243      -        5                          good see and nice product
4653      -        5                                i like this product
4922      -        5  first of all its 1.8” display say the big scre...
5472      -        5                     for bigneer it's a good camera
5508      -        5                                                  -
5636      -        5  vrry good laptop amazing it do have a good loo...
6066      -        5                                       good product
```

```
# remove special character
df['Review'] = df['Review'].str.replace(".", "")
df['Review'] = df['Review'].str.replace("!", "")
df['Review'] = df['Review'].str.replace("&", "")
df['Review'] = df['Review'].str.replace("#", "")
df['Review'] = df['Review'].str.replace("@", "")
df['Review'] = df['Review'].str.replace("4", "")
df['Review'] = df['Review'].str.replace("\n", "")
df['Review'] = df['Review'].str.replace("=", "")
df['Review'] = df['Review'].str.replace("-", "")
df['Review'] = df['Review'].str.replace("_", "")
df['Review'] = df['Review'].str.replace("/", "")
df['Review'] = df['Review'].str.replace("^", "")
df['Review'] = df['Review'].str.replace(",", "")
```

```
df['Text'] = df['Text'].str.replace(".", "")
df['Text'] = df['Text'].str.replace("!", "")
df['Text'] = df['Text'].str.replace("&", "")
df['Text'] = df['Text'].str.replace("#", "")
df['Text'] = df['Text'].str.replace("@", "")
df['Text'] = df['Text'].str.replace("4", "")
df['Text'] = df['Text'].str.replace("\n", "")
df['Text'] = df['Text'].str.replace("=", "")
df['Text'] = df['Text'].str.replace("-", "")
df['Text'] = df['Text'].str.replace("_", "")
df['Text'] = df['Text'].str.replace("/", "")
df['Text'] = df['Text'].str.replace("^", "")
df['Text'] = df['Text'].str.replace(",", "")
df['Text'] = df['Text'].str.replace("2", "")
df['Text'] = df['Text'].str.replace(":", "")
```

As our dataset has objective type data so covert them into int by using Encoding method as shown.

```
# Lets seaprate the catogorical data & numirical data
numerics=['int8','int16','int32','int64','float16','float32','float64']
cat_col=[] #empty list
features=df.columns.values.tolist()
for i in features:
    if df[i].dtype in numerics:
        continue
    cat_col.append(i)
cat_col
```

```
['Review', 'Text']
```

```
# Lets frist covert categorical data(type & column) into int
label = LabelEncoder()
for i in cat_col:
    data=label.fit_transform(df[i])
    pd.Series(data)
    df[i]=data
```

```
df.head(3)
```

	Review	Ratings	Text
0	1663	5	1357
1	729	5	3034
2	1677	5	2088

Also, there are some Null values are present in the all columns so I have used mode method and iterative imputer to deal with it as shown in below fig.

Iterative Imputer

Iterative imputer is a hidden gem of the sklearn library in python. The iterative imputer library provides us with tools to tackle the problem mentioned above. Instead of just replacing values with mean/median, we can have a regressor (Linear/Decision Tree/Random Forest/KNN) to impute missing values. By using the iterative imputer we can intelligently impute the missing values, avoid bias, maintain the relationship between variables, and can get better results.

```
# Creating null value list which we want to impute by using iterative inputer
data=df[['Review', 'Ratings', 'Text']]
```

```
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
imputer = IterativeImputer()

for i in data:
    df1=imputer.fit_transform(df[[i]])
    pd.Series([df1])
    df[i]=df1
```

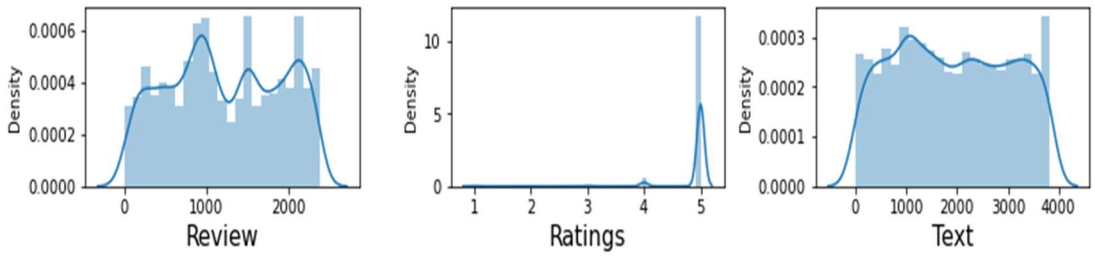
```
df['Ratings'].mode()
```

```
0    5
Name: Ratings, dtype: object
```

```
df['Ratings']=df['Ratings'].replace(np.nan, '5').astype(int)
```

Removing skewness based on the distribution graph as below by using Power Transformation.

```
#Let's see the how data is distributed or Graphical analysis of all features
plt.figure(figsize=(15,20))
plotnumber=1
for column in df:
    if plotnumber<=40:
        ax=plt.subplot(10,4,plotnumber)
        sns.distplot(df[column])
        plt.xlabel(column,fontsize=15)
    plotnumber+=1
plt.tight_layout()
```

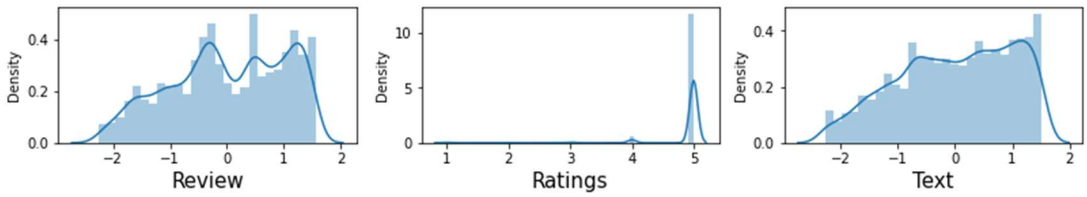


From graph we can see that the almost every columns have skewness, now lets check with skew values also.

As we see there are skewed data. As we see the skewed data is present as per thumb rule +0.5/-0.5 weight let's work on right skewed data by using power transform. As label is our target variable so we are not using the same.

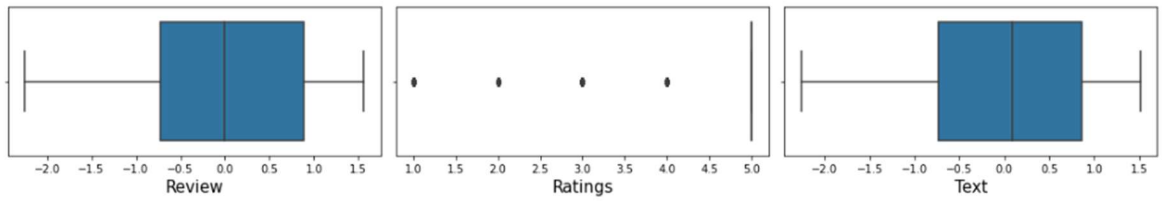
```
# Separate the skewed columns
df1=['Text','Review']

# Using power transformation to remove skewed data
from sklearn.preprocessing import PowerTransformer
pt=PowerTransformer()
df[df1]=pt.fit_transform(df[df1].values)
```

```
# Distribution after Power Transformation
plt.figure(figsize=(15,20))
plotnumber=1
for column in df:
    if plotnumber<=40:
        ax=plt.subplot(10,4,plotnumber)
        sns.distplot(df[column])
        plt.xlabel(column,fontsize=15)
    plotnumber+=1
plt.tight_layout()
```



Removing the Outliers: We have no outliers in dataset but we have data imbalance in target variable as shown in below graph.

```
#Let's see the outliers in the dataset by using box plot or Graphical analysis of all features
myFig=plt.figure(figsize=(20,20))
plotnumber=1
for column in df:
    if plotnumber<=32:
        ax=plt.subplot(8,4,plotnumber)
        sns.boxplot(df[column])
        plt.xlabel(column,fontsize=15)
    plotnumber+=1
plt.tight_layout()
```



Scaling the data:
Scaled the data using StandardScaler method.

Data stadardization

```
# data stadardization
scale=StandardScaler()
x_scaled=scale.fit_transform(x)
```

Handing Class imbalance problem:

I used Oversampling method, at last I used SMOTE method to handle the class imbalance problem.

```
# We have imbalance dataset lets use SMOTE
# Lets use of Resampling Techniques to handle Imbalanced Data
from sklearn.datasets import make_classification
from imblearn.over_sampling import SMOTE
from collections import Counter

ove_smp=SMOTE(sampling_strategy = {1: 6393, 2: 6393, 3: 6393, 4: 6393, 5: 6393}) # selecting data for resampling
x_train_ns,y_train_ns=ove_smp.fit_resample(x_train,y_train)
print(Counter(y_train))
print(Counter(y_train_ns))
```

```
Counter({5.0: 6393, 4.0: 296, 3.0: 60, 1.0: 43, 2.0: 16})
Counter({5.0: 6393, 4.0: 6393, 1.0: 6393, 3.0: 6393, 2.0: 6393})
```

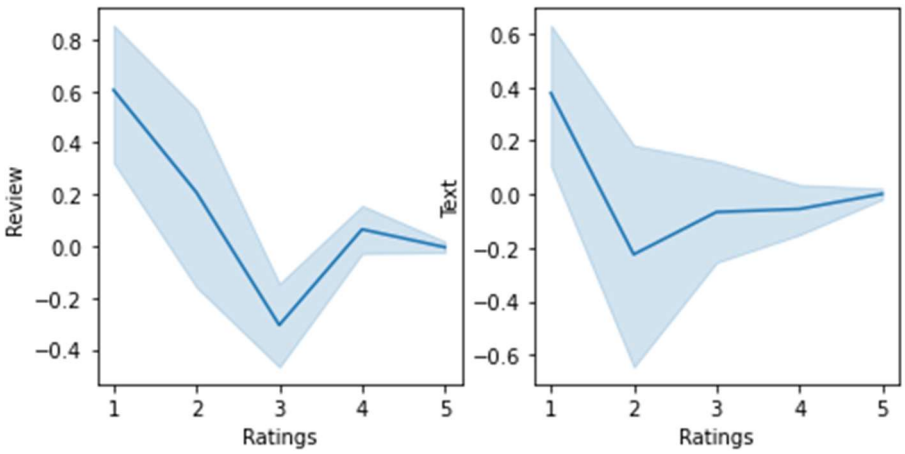
- Data Inputs- Logic- Output Relationships

  Multicollinearity:

  To check the Input output relation, I have plotted the distribution plot and also use VIF technique to check the multicollinearity. As shown in below screenshot. And detected no multicollinearity.

```
#check multicolinearity
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif=pd.DataFrame()
vif["vif"]=[variance_inflation_factor(x_scaled,i) for i in range(x_scaled.shape[1])]
vif["featurs"]=x.columns
print(vif)
```

```
        vif featurs
0  1.000269  Review
1  1.000269    Text
```



- State the set of assumptions (if any) related to the problem under consideration

  This Dataset is class imbalanced.

- Hardware and Software Requirements and Tools Used

Machine: Can use a laptop/desktop.

Operating system: Windows 11, Mac OS X 10.9 Mavericks or Higher

RAM & Processor: 4 GB+ RAM, i3 5th Generation 2.2 Ghz or equivalent/higher .

Tools Used: Jupyter Notebook, Microsoft Excel.

Libraries Used :

```
import pandas as pd        # for data manipulation

import numpy as np         # for mathematical calculations

import seaborn as sns      # for data visualization

import matplotlib.pyplot as plt  #for graphical analysis

%matplotlib inline

from scipy.stats import zscore # to remove outliers

from sklearn.preprocessing import StandardScaler  # for normalize the model

from sklearn.preprocessing import LabelEncoder  # to convert object into int

from sklearn.model_selection import train_test_split  # for train and test model

import warnings                    # to ignore any warnings

warnings.filterwarnings("ignore")

from sklearn import metrics  # for model evaluation

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report.
```

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  I have dropped the duplicated to avoid impact of the same on result.

  Reduced the skewness using power transform method.

  Used SMOTE method to deal with class imbalance problem.

- Testing of Identified Approaches (Algorithms)

  I have selected 5 models for the prediction which is used for the training and testing.

  1) Logistic Regression from sklearn.linear_model
  2) DecisionTreeClassifier from sklearn.tree
  3) RandomForestClassifier from sklearn.ensemble
  4) ADABoostClassifier from sklearn.ensemble
  5) GradientBoostClassifier from sklearn.ensemble

- Run and Evaluate selected models

1) **Logistic Regression from sklearn.linear_model:**
   Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is binary, which means there would be only two possible classes 1 (stands for success/yes) or 0 (stands for failure/no). Mathematically, a logistic regression model predicts P(Y=1) as a function of X. It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc.

2) **DecisionTreeClassifier from sklearn.tree:**

Decision trees can be constructed by an algorithmic approach that can split the dataset in different ways based on different conditions. The two main entities of a tree are decision nodes, where the data is split and leaves, where we get the outcome.

3) **RandomForestClassifier from sklearn.ensemble:**

As we know that a forest is made up of trees and more trees means more robust forest. Similarly, a random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

4) **ADABoostClassifier:**

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

5) **GradientBoostClassifier from sklearn.ensemble:**

The power of gradient boosting machines comes from the fact that they can be used on more than binary classification problems, they can be used on multi-class classification problems and even regression problems.

The Gradient Boosting Classifier depends on a loss function. A custom loss function can be used, and many standardized loss functions are supported by gradient boosting classifiers, but the loss function has to be differentiable. Gradient boosting systems have two other necessary parts: a weak learner and an additive component. Gradient boosting systems use decision trees as their weak learners.

```python
# Imporing ML algorithms and initialisation the same

from sklearn.linear_model import LogisticRegression
LR= LogisticRegression()

from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier()

from sklearn.ensemble import AdaBoostClassifier, GradientBoostingClassifier
ada=AdaBoostClassifier()
gb=GradientBoostingClassifier()

from sklearn.ensemble import RandomForestClassifier
rfc= RandomForestClassifier()

from sklearn.model_selection import cross_val_score
from sklearn.metrics import roc_curve,roc_auc_score, plot_roc_curve, auc

models=[]
models.append(('LogisticRegression', LR))
models.append(('DecisionTreeClassifier', dt))
models.append(('AdaBoostClassifier', ada))
models.append(('GradientBoostingClassifier', gb))
models.append(('RandomForestClassifier', rfc))
```

```
********** LogisticRegression **********
LogisticRegression()
Train Report: 0.31046457062412014
Test Report: 0.06680369989722508
Classification Report:               precision    recall  f1-score   su
pport

         1.0       0.01      0.67      0.01         9
         2.0       0.00      0.67      0.01         3
         3.0       0.01      0.80      0.03        20
         4.0       0.00      0.00      0.00       129
         5.0       0.95      0.06      0.12      2758

    accuracy                           0.07      2919
   macro avg       0.19      0.44      0.03      2919
weighted avg       0.90      0.07      0.11      2919

Confusion Matrix: [[   6    0    2    0    1]
 [   1    2    0    0    0]
 [   1    2   16    0    1]
 [  43   28   51    0    7]
 [ 875  613 1077   22  171]]
Accuracy: 31.02 %
Standard Deviation: 0.16 %

********** DecisionTreeClassifier **********
DecisionTreeClassifier()
Train Report: 0.9993743156577507
Test Report: 0.6135662898252826
Classification Report:               precision    recall  f1-score   su
pport

         1.0       0.01      0.22      0.02         9
         2.0       0.00      0.33      0.01         3
         3.0       0.02      0.30      0.04        20
```

```
         4.0       0.11      0.36      0.17       129
         5.0       0.96      0.63      0.76      2758

    accuracy                           0.61      2919
   macro avg       0.22      0.37      0.20      2919
weighted avg       0.91      0.61      0.73      2919

Confusion Matrix: [[   2    1    0    1    5]
 [   0    1    2    0    0]
 [   2    2    6    2    8]
 [   5    8    4   47   65]
 [ 227  195  241  360 1735]]
Accuracy: 78.36 %
Standard Deviation: 0.51 %

********************** AdaBoostClassifier **********************
AdaBoostClassifier()
Train Report: 0.5311434381354606
Test Report: 0.2552243919150394
Classification Report:               precision    recall  f1-score   su
pport

         1.0       0.01      0.33      0.01         9
         2.0       0.00      0.33      0.00         3
         3.0       0.02      0.65      0.04        20
         4.0       0.08      0.40      0.14       129
         5.0       0.98      0.25      0.39      2758

    accuracy                           0.26      2919
   macro avg       0.22      0.39      0.12      2919
weighted avg       0.93      0.26      0.38      2919

Confusion Matrix: [[  3   0   1   3   2]
 [  0   1   2   0   0]
 [  1   2  13   2   2]
 [ 16  23  28  52  10]
 [469 481 574 558 676]]
Accuracy: 51.06 %
Standard Deviation: 1.59 %

********************** GradientBoostingClassifier ********************
***
GradientBoostingClassifier()
Train Report: 0.6941029250743
Test Report: 0.45940390544707094
Classification Report:               precision    recall  f1-score   su
pport

         1.0       0.01      0.33      0.02         9
         2.0       0.01      0.67      0.01         3
         3.0       0.03      0.70      0.06        20
         4.0       0.14      0.60      0.22       129
         5.0       0.98      0.45      0.62      2758

    accuracy                           0.46      2919
   macro avg       0.23      0.55      0.19      2919
weighted avg       0.93      0.46      0.59      2919
```

```
Confusion Matrix: [[   3    1    0    2    3]
 [   0    2    1    0    0]
 [   1    1   14    2    2]
 [   3   10   15   77   24]
 [ 326  280  424  483 1245]]
Accuracy: 68.07 %
Standard Deviation: 0.72 %

********************** RandomForestClassifier **********************
RandomForestClassifier()
Train Report: 0.9993743156577507
Test Report: 0.6039739636861939
Classification Report:               precision    recall  f1-score   su
pport

         1.0       0.01      0.33      0.02         9
         2.0       0.00      0.33      0.01         3
         3.0       0.02      0.30      0.04        20
         4.0       0.12      0.37      0.19       129
         5.0       0.96      0.62      0.75      2758

    accuracy                           0.60      2919
   macro avg       0.22      0.39      0.20      2919
weighted avg       0.91      0.60      0.72      2919

Confusion Matrix: [[   3    1    0    1    4]
 [   0    1    2    0    0]
 [   2    1    6    3    8]
 [   7    6    5   48   63]
 [ 260  210  247  336 1705]]
Accuracy: 81.26 %
Standard Deviation: 0.63 %
```

	Model	Accuracy_train_score	Accuracy_test_score	Cross_val_score
0	LogisticRegression	31.046457	6.680370	31.015173
1	DecisionTreeClassifier	99.937432	61.356629	78.363835
2	AdaBoostClassifier	53.114344	25.522439	51.055842
3	GradientBoostingClassifier	69.410293	45.940391	68.074456
4	RandomForestClassifier	99.937432	60.397396	81.260754

From above accuraces we can see there are major diffrance in all models test accuracy and cross validation accuracy. Now lets use hyper Tunning Parameter method to get better accuracy. From above table i m selecting Decision Tree Classifier is best model as it has least diffrance in accuracy.

- Key Metrics for success in solving problem under consideration

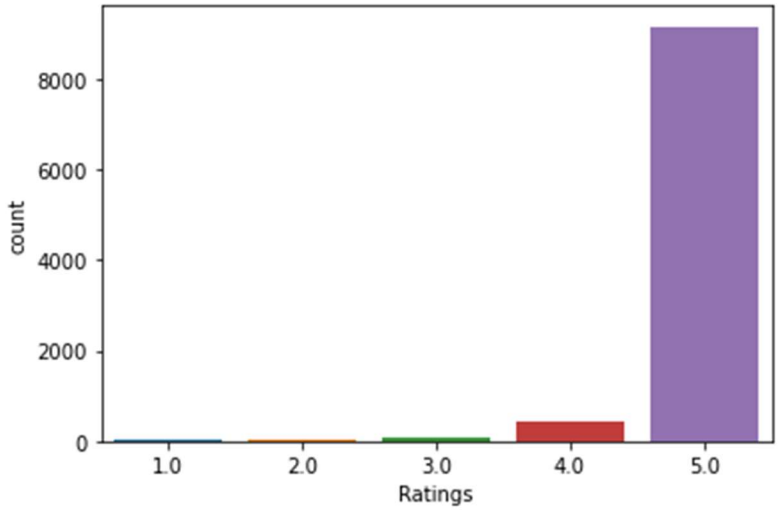
  I have used accuracy score, classification report & confusion matrix as metrices.

We got **our** best model looking at accuracy & confusion matrix is **DecisionTreeClassifier** with Kfold cross validation method with the accuracy score of 78%.

- Visualizations:

Univariate Plots

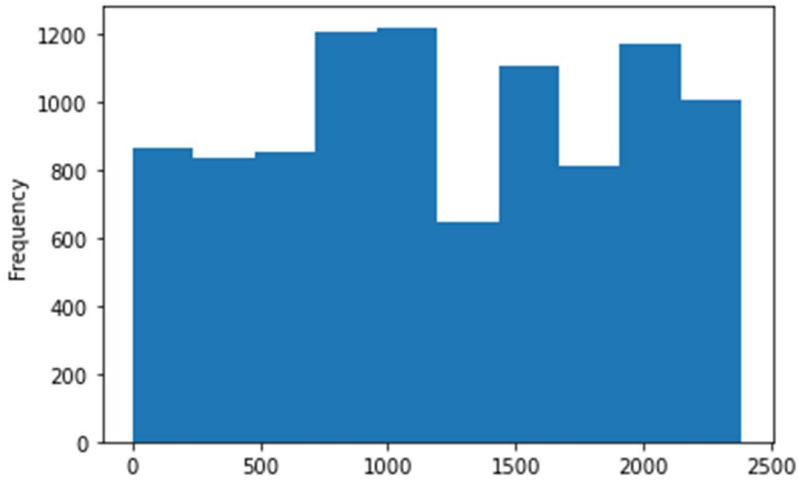
```
[30]: #plot each class frequency
      sns.countplot(x='Ratings',data=df)
      plt.show()
      print(df['Ratings'].value_counts())
```




```
5.0    9151
4.0     425
3.0      80
1.0      52
2.0      19
Name: Ratings, dtype: int64
```

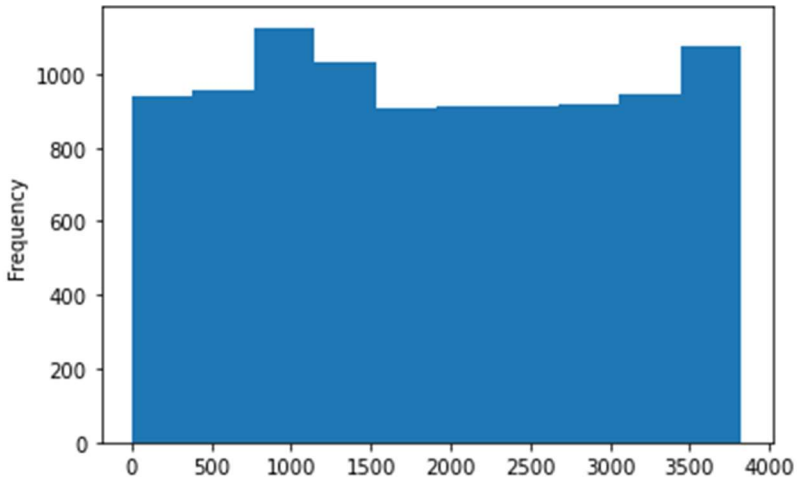
From graph we can see that their are maximum rating count is for 4 & 5 category.

```
#plot each class frequency
df['Review'].plot.hist()
plt.show()
print(df['Review'].value_counts())
```




```
885.0     260
2118.0    192
984.0     161
1523.0    135
743.0     131
          ...
1934.0      1
1111.0      1
2025.0      1
1891.0      1
1199.0      1
Name: Review, Length: 2388, dtype: int64
```

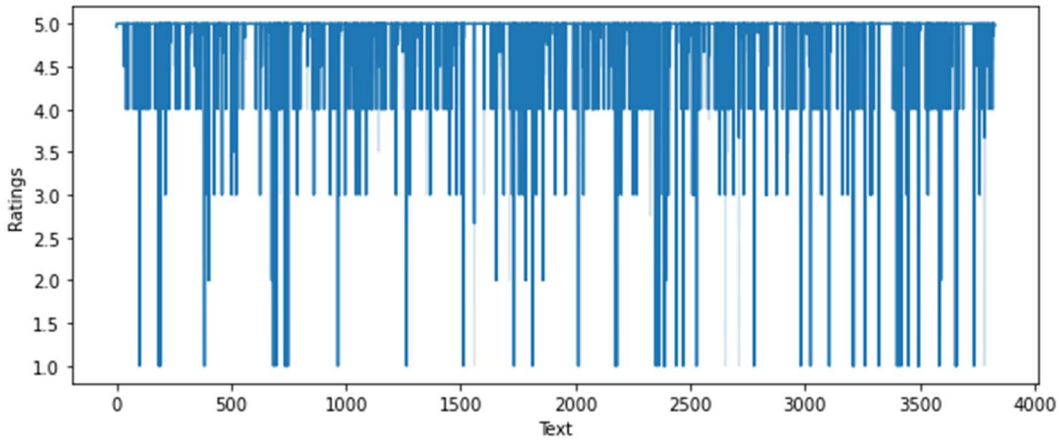
```
#plot each class frequency
df['Text'].plot.hist()
plt.show()
print(df['Text'].value_counts())
```



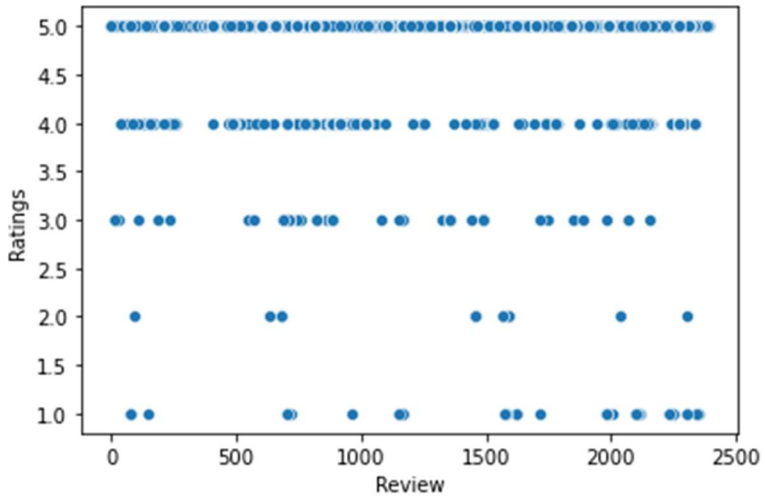

```
3826.0    179
980.0     165
0.0       109
1076.0     60
2236.0     58
          ...
3622.0      1
694.0       1
3706.0      1
141.0       1
3729.0      1
Name: Text, Length: 3827, dtype: int64
```

Bivariant Plot

```
]: #Bivariant graph
   plt.figure(figsize =(10, 4))
   sns.lineplot(x ='Text', y ='Ratings', data = df)
   plt.show()
```



```
#Bivariant graph
plt.figure(figsize =(6, 4))
sns.scatterplot(x ='Review', y ='Ratings', data = df)
plt.show()
```



- Interpretation of the Results

  Looking at the accuracy i m selecting DecisionTreeClassifier further to get better accuracy I have used **RandomizedSearchCV**

  hyper tuning parameter.

```
# Hyper tuning by using RandomizedSearchCV With dt
from sklearn.model_selection import RandomizedSearchCV

para={'min_samples_split':[7,10,8],'ccp_alpha':[0.25,0.1,0.3,0.092],'max_depth':[2,4,11,10,8,9],'random_state':[20,10,30]}
rand=RandomizedSearchCV(estimator=dt, cv=5,param_distributions=para, scoring = 'accuracy')
rand.fit(x_train_ns,y_train_ns)

print(rand.best_params_)
print(rand.best_score_)
```

```
{'random_state': 30, 'min_samples_split': 8, 'max_depth': 11, 'ccp_alpha': 0.25}
0.1999061473486626
```

```
dt=DecisionTreeClassifier(random_state=1, min_samples_split=8, max_depth= 4, ccp_alpha= 0.3)

dt.fit(x_train_ns,y_train_ns)
y_pred=model.predict(x_train_ns)
AS=accuracy_score(y_train_ns,y_pred)
print("Train Report:",AS*100)
pred=model.predict(x_test)
AS2=accuracy_score(y_test,pred)
print("Test Report:",AS2*100)
CR=classification_report(y_test,pred)
print("Classification Report:",CR)
CM=confusion_matrix(y_test,pred)
print("Confusion Matrix:", CM)
accuracies= cross_val_score(model, x_train_ns, y_train_ns, cv=2, scoring='accuracy')
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
cv_score.append(accuracies.mean()*100)
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Train Report: 99.93743156577507
Test Report: 60.3973963686194
Classification Report:               precision    recall  f1-score   support

         1.0       0.01      0.33      0.02         9
         2.0       0.00      0.33      0.01         3
         3.0       0.02      0.30      0.04        20
         4.0       0.12      0.37      0.19       129
         5.0       0.96      0.62      0.75      2758

    accuracy                           0.60      2919
   macro avg       0.22      0.39      0.20      2919
```

Saving the model:

```
#save model
import pickle
Filename='Finalized_model_Rating_Prediction1.pickle'
pickle.dump(dt,open(Filename,'wb'))
```

After finalising the model, I have load Test data set and do all the data pre-processing steps as mentioned above and do the prediction by using saved model.

Lets Do the prediction by using selected saved model

```
df1["predictions"] = df1[['Text','Review']].apply(lambda s: dt.predict(s.values[None])[0], axis=1)
```

```
df1["predictions"]
```

```
0       1.0
1       1.0
2       1.0
3       1.0
4       1.0
       ...
8320    1.0
```

CONCLUSION

- ## Key Findings and Conclusions of the Study

  The key findings, inferences, observations from the whole problem are that I found out how to handle the dataset which is class imbalanced and use for metrics like f1 score. I observed that data with least correlation with target variable do not impact the result.

- ## Learning Outcomes of the Study in respect of Data Science

  Visualization is very useful for detection of outliers in particular

  columns, distribution of data, Heat map for correlation and null values. I choose DecisionTreeClassifier algorithm as my final model since it was having least difference between its cross-validation score and testing accuracy.

  I faced the challenges in data pre-processing & visualizing all the columns at once when there are some object datatypes were in the columns & special character are present in the columns.

- ## Limitations of this work and Scope for Future Work

  Limitations of this solutions is that I could have used hyperparameter tuning for selected model or algorithm, but no change in score.

  Future scope of this project is that we can use to predict the ratings for any ecommerce sites & improve the sales with better profit.