Rating Prediction Analysis

Submitted by:
Anchal Awasthi

ACKNOWLEDGMENT

I would like to thank FlipRobo Technologies to provide me this valuable data in order to perform this analysis. I would also like to thank my mentor Khushboo Garg for helping me throughout this project.

INTRODUCTION

• Business Problem Framing

The prediction for rating for a particular review on any e-commerce site can help their listers/makers of those products to find the problems or positives that a customer is facing after buying this product. This can help these e-commerce sites to grow or retain their businesses.
Here, we have built a ML model that processes Review posted on Amazon using common NLP techniques.
This task is similar to Sentiment Analysis, but instead of predicting the positive and negative sentiment (sometimes neutral also), here I need to predict the star rating

• Conceptual Background of the Domain Problem

Sentiment Analysis is the most commonly used approach to analyze data which is in the form of text and to identify sentiment content from the text. Opinion Mining is another name for sentiment analysis. A wide range of text data is getting generated in the form of suggestions, feedback's, tweets and comments. E-Commerce portals are generating a lot of data every day in the form of customer reviews. Analyzing E-Commerce data will help online retailers to understand customer expectations, provide better shopping experience and to increase the sales. Sentiment Analysis can be used to identify positive, negative and neutral information from the customer reviews. Researchers have developed a lot of techniques in Sentiment Analysis. Mostly sentiment Analysis is done using a single machine learning algorithm. This work uses Amazon customer review data and focuses on finding aspect terms from each review, identifying the Parts-of-Speech, applying classification algorithms to find the score of positivity, negativity and neutrality of each review.

• Aim of the Project

Our aim is to build a ML model that can easily identify the rating or star by taking input the text review.

- # Review of Literature

The advent of electronic commerce with growth in internet and network technologies has led customers to move to online retail platforms such as Amazon, Walmart, Flip Kart, etc. People often rely on customer reviews of products before they buy online. These reviews are often rich in information describing the product. Customers often choose to compare between various products and brands based on whether an item has a positive or negative review. More often, these reviews act as a feedback mechanism for the seller. Through this medium, sellers strategize their future sales and product improvement.

There is a client who has a website where people write different reviews for technical products. Now they want to add 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.

This project is more about exploration, feature engineering and classification that can be done on this data. Since we scrape huge amount of data that includes five stars rating, we can do better data exploration and derive some interesting features using the available columns.

We can categorize the ratings as: 1.0, 2.0, 3.0, 4.0 and 5.0 stars

The goal of this project is to build an application which can predict the rating by seeing the review. In the long term, this would allow people to better explain and reviewing their purchase with each other in this increasingly digital world.

- # Motivation for the Problem Undertaken

The motivation that driven me towards the timeline of this project was the sheer size and complexity of this dataset. I have tried my best to follow best practices of machine learning throughout the project and follow the steps that doesn't only predict the best results but also tried to keep the process simple to understand.

Analytical Problem Framing

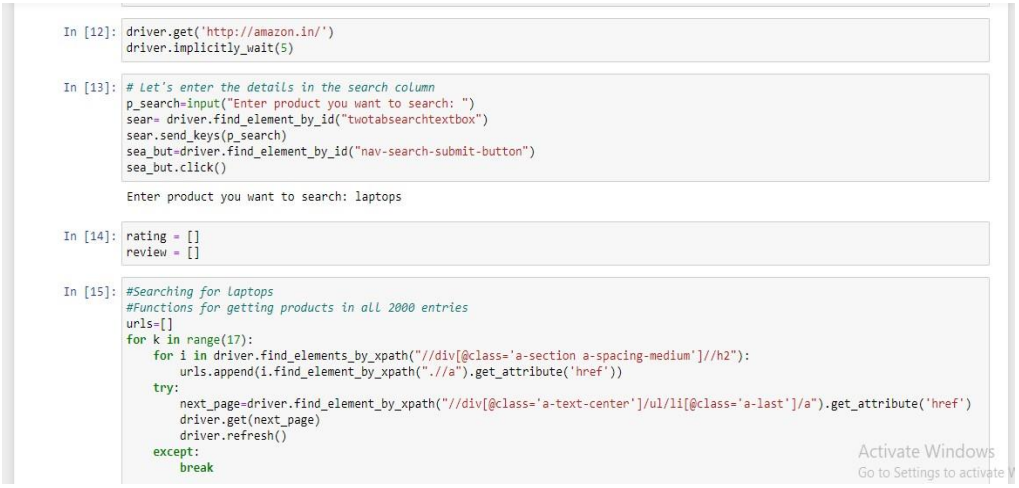
- Mathematical/ Analytical Modeling of the Problem

Mathematical Tools Used: Mean, Average, IQR, Standard deviation and Median for gaining insights of the dataset. Analytical Tools Used: Correlation and Skewness for finding the relationships of dependent and independent variable and checking the distribution of data.

Packages Used: scikit-learn, pandas, seaborn etc.

- Data Sources and their formats

Data was been scraped by me using Selenium and then making csv file of scraped data and using it in analysis. Whereas originally data belongs Amazon.in. The data contained 2 columns and 22000 entries which are shown below:



```
In [12]: driver.get('http://amazon.in/')
         driver.implicitly_wait(5)

In [13]: # Let's enter the details in the search column
         p_search=input("Enter product you want to search: ")
         sear= driver.find_element_by_id("twotabsearchtextbox")
         sear.send_keys(p_search)
         sea_but=driver.find_element_by_id("nav-search-submit-button")
         sea_but.click()

         Enter product you want to search: laptops

In [14]: rating = []
         review = []

In [15]: #Searching for laptops
         #Functions for getting products in all 2000 entries
         urls=[]
         for k in range(17):
             for i in driver.find_elements_by_xpath("//div[@class='a-section a-spacing-medium']//h2"):
                 urls.append(i.find_element_by_xpath(".//a").get_attribute('href'))
             try:
                 next_page=driver.find_element_by_xpath("//div[@class='a-text-center']/ul/li[@class='a-last']/a").get_attribute('href')
                 driver.get(next_page)
                 driver.refresh()
             except:
                 break
```

In [15]:
```
#Searching for laptops
```

```
    for i in driver.find_elements_by_xpath("//div[@class='a-section a-spacing-medium']//h2"):
        urls.append(i.find_element_by_xpath(".//a").get_attribute('href'))

        next_page=driver.find_element_by_xpath("//div[@class='a-text-center']/ul/li[@class='a-last']/a").get_attribute('href')
        driver.get(next_page)
```

```
    mls.apped(i)
    driver.implicitly wait(5)
```

```
        driver.find_element_by_xpath("//a[1][@id='acrCustomerReviewLink']").click()
    except NoSuchElementException:
        print("No rating")
```

```
        driver.find_element_by_xpath("//a[@class='a-link-emphasis a-text-bold']").click()
    except NoSuchElementException:
```

```
#Scrapping the detail
    S_page=1
    E_page=60
```

```
        try:
                lng=driver.find_elements_by_xpath("//div[@class='a-section celwidget']/div[2]/a[1]"
                 i in Rating:
                 ratings=i.get_attribute('title')
```

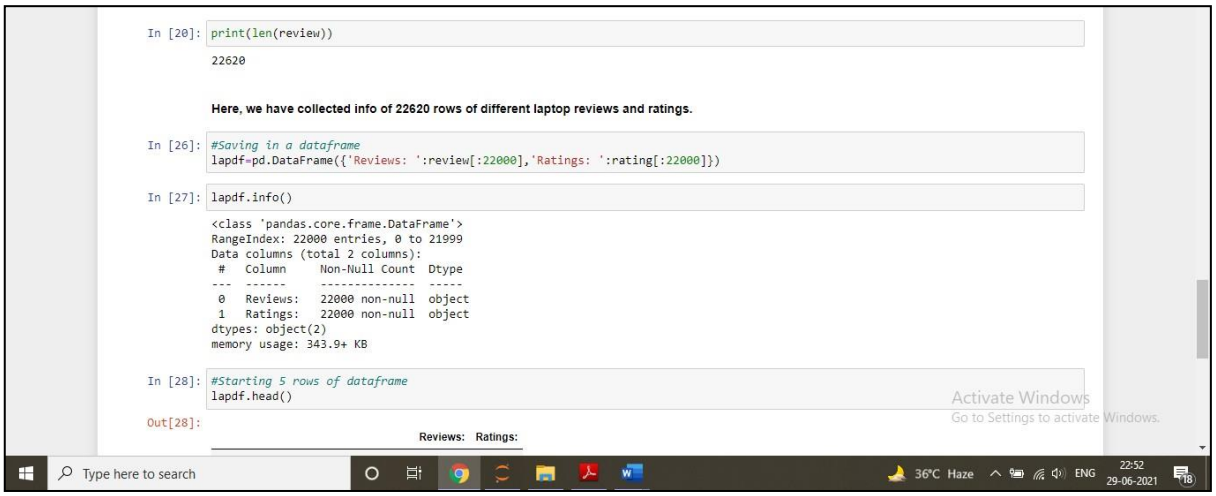
```
        #Printing the page scrapped
        print("Product review and rating  of page {} scraped ".format(page+1))

        #Looping for going to next page automatically
        try:
            next_page=driver.find_element_by_xpath("//div[@id='cm_cr-pagination_bar']/ul/li[2]/a")
            if next page.text--'  *x- °<g*':
                 nezt page . click(
```

Product re'.'iei' a nd rating a* page 10 scraped
Product re'.'iei- a nd rating a* page 11sdraped

Product re'.'iei' a nd rating a* page 16 scraped
Product re'.'iei- a nd rating a* page 17 sdraped

Product ce.ver.1 and rating o' page 19 scraped



```
In [20]: print(len(review))
         22620

         Here, we have collected info of 22620 rows of different laptop reviews and ratings.

In [26]: #Saving in a dataframe
         lapdf=pd.DataFrame({'Reviews: ':review[:22000],'Ratings: ':rating[:22000]})

In [27]: lapdf.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 22000 entries, 0 to 21999
         Data columns (total 2 columns):
          #   Column      Non-Null Count  Dtype
         ---  ------      --------------  -----
          0   Reviews:    22000 non-null  object
          1   Ratings:    22000 non-null  object
         dtypes: object(2)
         memory usage: 343.9+ KB

In [28]: #Starting 5 rows of dataframe
         lapdf.head()

Out[28]:
                                                Reviews:  Ratings:
```

Figure 1

In [39]:
```
df=pd.read_csv('RatRew.csv')
df  #Checking the dataset
```

Out[39]:

	Unnamed: 0	Reviews:	Ratings:
0	0	Nice Product - The large screen and decent pro...	5.0
1	1	Description said 15.6". Laptop arrived. Laptop...	1.0
2	2	Bulid of laptop is sturdyBettery is also goodD...	4.0
3	3	Got this delivered today, this laptop is a bea...	5.0
4	4	Nice Product - The large screen and decent pro...	5.0
...	...	...	...
21995	21995	This laptop comes with ryzen 5000 series proce...	4.0
21996	21996	the features this laptop serves for the price ...	5.0
21997	21997	The invoice doesn't contain the serial number ...	1.0
21998	21998	Check full review in U tube from KM's frank view	3.0
21999	21999	With Intelligent Fan cooling setup, Cinebench ...	4.0

22000 rows × 3 columns

In [40]:
```
#Structure of data
df.shape#We are having 22000 rows and 2 columns.
```

Out[40]: (22000, 3)

- For keeping only, the useful data we used feature engineering. We removed the data if unnecessary by qualifying them on certain conditions like their uniqueness, their correlation with target variable and the no of outliers present in that particular variable.

- We then tried to treat the missing values.

- After handling the missing values pre-processing was done using NLP techniques like Regex, Tokenization, Lemmatization and removing all the unnecessary stop words.

Data Pre-Processing

In [42]:
```
#We will be removing the first column.
df.drop('Unnamed: 0',axis=1,inplace=True)
```

In [43]:
```
#Data Info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22000 entries, 0 to 21999
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Reviews:   21940 non-null  object
 1   Ratings:   22000 non-null  float64
dtypes: float64(1), object(1)
memory usage: 343.9+ KB
```

In [44]:
```
#We will find the missing values first
df.isnull().sum()
```

Out[44]:
```
Reviews:     60
Ratings:      0
dtype: int64
```


In [45]:
```
#Treating the missing values
df['Reviews: '].fillna('No review present',inplace=True)
```

In [46]:
```
#Checking the missing values again
df.isnull().sum()
```

Out[46]:
```
Reviews:     0
Ratings:     0
dtype: int64
```

In [47]:
```
#Unique values present in Ratings: column
df['Ratings: '].value_counts()
```

Out[47]:
```
5.0    8332
1.0    6118
4.0    4434
3.0    2038
2.0    1078
Name: Ratings: , dtype: int64
```

In [48]:
```
#Describing data
df.describe() #Mean rating is 3.3
```

Out[48]:

	Ratings:
count	22000.000000
mean	3.353818
std	1.659143
min	1.000000

In [54]:
```
def clean_text(df, df_column_name):

    #Converting all messages to lowercase
    df[df_column_name] = df[df_column_name].str.lower()

    #Replace email addresses with 'e-mail'
    df[df_column_name] = df[df_column_name].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$','e-mail')

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

    #Replace money symbols with 'dollars' (£ can by typed with ALT key + 156)
    df[df_column_name] = df[df_column_name].str.replace(r'£|\$', 'dollars')

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

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

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

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

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

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

```
    #Remove stopwords
    stop_words = set(stopwords.words('english') + ['u', 'ü', 'â', 'ur', '4', '2', 'im', 'dont', 'doin', 'ure'])
    df[df_column_name] = df[df_column_name].apply(lambda x: ' '.join(term for term in x.split() if term not in stop_words))
```

In [55]:
```
## Calling the class
clean_text(df, 'Reviews: ')
df['Reviews: '].head(3)
```

Out[55]:
```
0    nice product large screen decent processor rea...
1    description said numbr laptop arrived laptop b...
2    bulid laptop sturdybettery also gooddisplay co...
Name: Reviews: , dtype: object
```

In [56]:
```
#Using RegexpTokenizer for tokenizing the data
from nltk.tokenize import RegexpTokenizer
tokenizer=RegexpTokenizer(r'\w+')
df['Reviews: '] = df['Reviews: '].apply(lambda x: tokenizer.tokenize(x.lower()))
df.head()
```

In [59]:
```
#First Lemmatizing and then Stemming(Snowball Method) for reducing characters
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
```

In [60]:
```
import nltk
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\DELL\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```


Out[60]:
```
True
```

In [62]:
```
#Processing review with above Function
processed_review = []

for doc in df['Reviews: ']:
    processed_review.append(preprocess(doc))

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

```
22000
```

- ## Data Inputs- Logic- Output Relationships

  After data pre-processing was completed, we were left with 2 columns excluding unnamed column.
  Other than Ratings column, the Reviews column were taken as input variables.

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

  It was pretty clear that machine learning will be used to predict the results for the dataset.

- ## Hardware and Software Requirements and Tools Used

Hardware Requirements: Laptop or PC with 4GB RAM or plus having i3 or above processor.

Software Requirements: Python and Jupyter Notebook.
Jupyter Tools/ Packages: pandas, numpy, seaborn, matplotlib.pyplot, import warnings, sklearn.preprocessing, sklearn.model_selection, sklearn.linear_model, nltk, re etc.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  - For that, we used various statistical tools like median, correlation, data skewness, standard deviation for data analysis.
  - We also used scikit-learn and other numerous analytical tools for achieving the target.

- Testing of Identified Approaches (Algorithms)

In this analysis, we have used three algorithms and three ensemble techniques to boost the predictive scores for better results. With the help of this approach, we will be able to successfully micro credit loan repayment predictions.

ALGORITHM USED:

- Logistic Regression
- Decision Tree Classifier
- MultinomialNB
- KNeighborsClassifier

ENSEMBLE TECHNIQUE USED:

- Gradient Boosting Classifier
- AdaBoost Classifier
- RandomForestClassifier

We ran these algorithms in order to find best fitting model and best accuracy and other

different parameters. The working of these algorithms are defined below:

- **Logistic Regression**

  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 dichotomous, which means there would be only two possible classes.

  In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no).

- **Decision Tree Classifier**

  Decision tree analysis is a predictive modelling tool that can be applied across many areas. Decision trees can be constructed by an algorithmic approach that can split the dataset in different ways based on different conditions. Decisions trees are the most powerful algorithms that falls under the category of supervised algorithms.

  Classification decision trees − In this kind of decision trees, the decision variable is categorical. The above decision tree is an example of classification decision tree.

- **Multinomial NB**

  Naïve Bayes algorithms is a classification technique based on applying Bayes' theorem with a strong assumption that all the predictors are independent to each other. In simple words, the assumption is that the presence of a feature in a class is independent to the presence of any other feature in the same class.

  The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work.

- **Gradient Boosting Classifier**

  Gradient boosting classifiers are a group of machine learning algorithms that combine many weak learning models together to create a strong predictive model. Decision trees are usually used when doing gradient boosting.

- **AdaBoost Classifier**

  It combines multiple classifiers to increase the accuracy of classifiers. AdaBoost is an iterative ensemble method. AdaBoost classifier builds a strong classifier by combining multiple poorly performing classifiers so that you will get high accuracy strong classifier. The basic concept behind Adaboost is to set the weights of classifiers and training the data sample in each iteration such that it ensures the accurate predictions of unusual observations.

- **Random Forest Classifier**

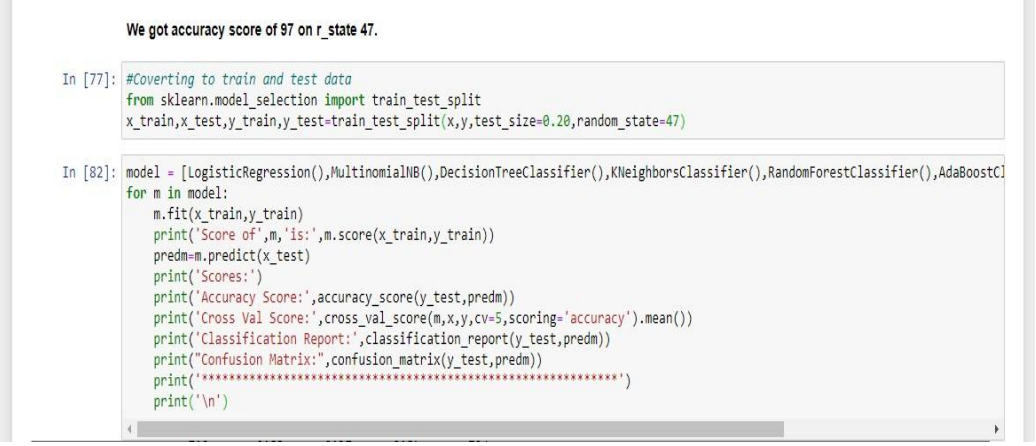
  It technically is an ensemble method (based on the divide-and- conquer approach) of decision trees generated on a randomly split dataset. This collection of decision tree classifiers is also known as the forest. The individual decision trees are generated using an attribute selection indicator such as information gain, gain ratio, and Gini index for each attribute. Each tree depends on an independent random sample. In a classification problem, each tree votes and the most popular class is chosen as the final result. In the case of regression, the average of all the tree

outputs is considered as the final result. It is simpler and more powerful compared to the other non-linear classification algorithms.

- # Run and Evaluate selected models

  We have tried to fit different algorithms on our dataset and their description and snapshots are pasted below:

  - Applying different algorithms (Code)



We got accuracy score of 97 on r_state 47.

```
In [77]: #Coverting to train and test data
         from sklearn.model_selection import train_test_split
         x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=47)

In [82]: model = [LogisticRegression(),MultinomialNB(),DecisionTreeClassifier(),KNeighborsClassifier(),RandomForestClassifier(),AdaBoostCl
         for m in model:
             m.fit(x_train,y_train)
             print('Score of',m,'is:',m.score(x_train,y_train))
             predm=m.predict(x_test)
             print('Scores:')
             print('Accuracy Score:',accuracy_score(y_test,predm))
             print('Cross Val Score:',cross_val_score(m,x,y,cv=5,scoring='accuracy').mean())
             print('Classification Report:',classification_report(y_test,predm))
             print("Confusion Matrix:",confusion_matrix(y_test,predm))
             print('**********************************************************')
             print('\n')
```

Figure 3: Using Different Models and Ensemble Techniques

  - Output (Scores)

```
Score of LogisticRegression() is: 0.9616477272727273
Scores:
Accuracy Score: 0.9706818181818182
Cross Val Score: 0.6017727272727273
Classification Report:               precision    recall  f1-score   support

         1.0       0.95      0.99      0.97      1179
         2.0       0.85      0.88      0.87       204
         3.0       0.99      0.93      0.96       395
         4.0       0.98      0.97      0.97       872
         5.0       0.99      0.98      0.98      1750

    accuracy                           0.97      4400
   macro avg       0.95      0.95      0.95      4400
weighted avg       0.97      0.97      0.97      4400

Confusion Matrix: [[1167    9    1    2    0]
 [  11  180    2    9    2]
 [   9   12  369    5    0]
```

```
Score of MultinomialNB() is: 0.941875
Scores:
Accuracy Score: 0.9513636363636364
Cross Val Score: 0.5637272727272726
Classification Report:               precision    recall  f1-score   support

         1.0       0.95      0.96      0.96      1179
         2.0       0.96      0.77      0.86       204
         3.0       1.00      0.86      0.93       395
         4.0       0.93      0.94      0.94       872
         5.0       0.95      0.99      0.97      1750

    accuracy                           0.95      4400
   macro avg       0.96      0.91      0.93      4400
weighted avg       0.95      0.95      0.95      4400

Confusion Matrix: [[1137    1    0   24   17]
 [  17  158    1   24    4]
 [  19    3  341   11   21]
```

```
Score of DecisionTreeClassifier() is: 0.9646022727272727
Scores:
Accuracy Score: 0.9740909090909091
Cross Val Score: 0.502909090909091
Classification Report:               precision    recall  f1-score   support

         1.0       0.96      0.99      0.98      1179
         2.0       0.83      0.92      0.87       204
         3.0       0.99      0.93      0.96       395
         4.0       0.99      0.96      0.98       872
         5.0       0.99      0.98      0.99      1750

    accuracy                           0.97      4400
   macro avg       0.95      0.96      0.95      4400
weighted avg       0.98      0.97      0.97      4400

Confusion Matrix: [[1167   10    1    1    0]
 [  11  187    2    2    2]
```

```
Score of KNeighborsClassifier() is: 0.9598295454545455
Scores:
Accuracy Score: 0.9677272727272728
Cross Val Score: 0.48395454545454547
Classification Report:               precision    recall  f1-score   support

         1.0       0.96      0.99      0.98      1179
         2.0       0.91      0.79      0.85       204
         3.0       0.93      0.95      0.94       395
         4.0       0.95      0.99      0.97       872
         5.0       1.00      0.97      0.98      1750

    accuracy                           0.97      4400
   macro avg       0.95      0.94      0.94      4400
weighted avg       0.97      0.97      0.97      4400

Confusion Matrix: [[1163    3    9    4    0]
 [  11  162   14   15    2]
```

```
Score of RandomForestClassifier() is: 0.9646022727272727
Scores:
Accuracy Score: 0.9740909090909091
Cross Val Score: 0.6080454545454546
Classification Report:               precision    recall  f1-score   support

         1.0       0.96      0.99      0.98      1179
         2.0       0.83      0.92      0.87       204
         3.0       0.99      0.93      0.96       395
         4.0       0.99      0.96      0.98       872
         5.0       0.99      0.98      0.99      1750

    accuracy                           0.97      4400
   macro avg       0.95      0.96      0.95      4400
weighted avg       0.98      0.97      0.97      4400

Confusion Matrix: [[1167   10    1    1    0]
```

```
Score of AdaBoostClassifier() is: 0.5553977272727273
Scores:
Accuracy Score: 0.5493181818181818
Cross Val Score: 0.46154545454545454
Classification Report:              precision    recall  f1-score   support

         1.0       0.55      0.72      0.63      1179
         2.0       1.00      0.12      0.22       204
         3.0       0.41      0.36      0.38       395
         4.0       0.39      0.29      0.34       872
         5.0       0.62      0.65      0.64      1750

    accuracy                           0.55      4400
   macro avg       0.59      0.43      0.44      4400
weighted avg       0.56      0.55      0.53      4400

Confusion Matrix: [[ 850    0   75  142  112]
```

```
Score of GradientBoostingClassifier() is: 0.9574431818181818
Scores:
Accuracy Score: 0.9668181818181818
Cross Val Score: 0.5582272727272728
Classification Report:              precision    recall  f1-score   support

         1.0       0.93      0.99      0.96      1179
         2.0       0.96      0.76      0.85       204
         3.0       0.99      0.93      0.96       395
         4.0       0.98      0.97      0.97       872
         5.0       0.99      0.98      0.98      1750

    accuracy                           0.97      4400
   macro avg       0.97      0.93      0.94      4400
weighted avg       0.97      0.97      0.97      4400
```

Figure 4:Scores of different models

- # Key Metrics for success in solving problem under consideration

  When building and optimizing your classification model, measuring how accurately it predicts your expected outcome is crucial.
  However, this metric alone is never the entire story, as it can still offer misleading results. That's where these additional performance evaluations come into play to help tease out more meaning from your model.
  The various metrics used to evaluate our model used in this analysis are:

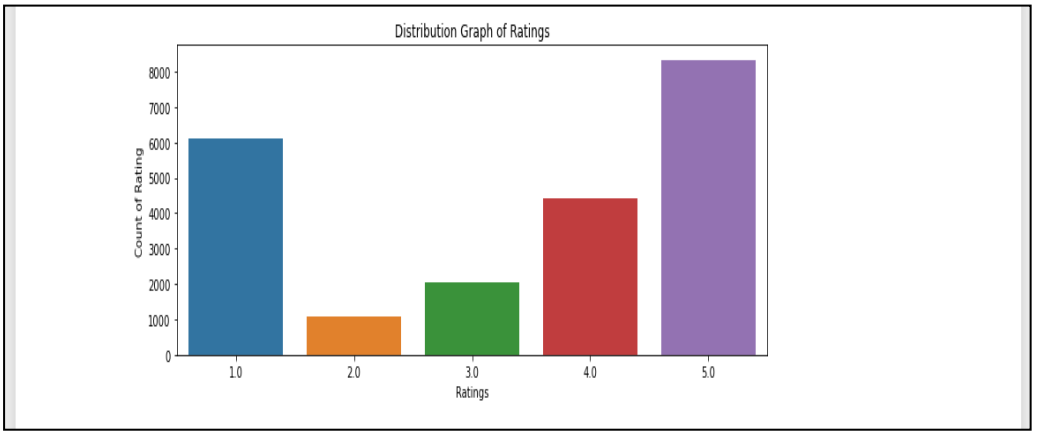
- **Confusion Matrix**
  Evaluation of the performance of a classification model is based on the counts of test records correctly and incorrectly predicted by the model. The confusion matrix provides a more insightful picture which is not only the performance of a predictive model, but also which classes are being predicted correctly and incorrectly, and what type of errors are being made.
- **Accuracy Score**
  Accuracy is one metric for evaluating classification models. Informally, **accuracy** is the fraction of predictions our model got right.
- **Precision**
  It is the ratio of *True Positives* to all the positives predicted by the model.
- **Recall**
  It is the ratio of *True Positives* to all the positives in your Dataset.
- **F1 Score**
  F-Measure provides a single score that balances both the concerns of precision and recall in one number. A good F1 score means that you have low false positives and low false negatives, so you're correctly identifying real threats, and you are not disturbed by false alarms.

- Visualizations

  In this analysis, we have used a single graphs for getting insights of the data that was given to us so that we can interpret the data to get desired results.

  - **Distribution Plots**

    For finding distribution of different variables.



- Interpretation of the Results

```
In [83]: #Hyper-Paramter Tuning(using GridCVSearch)
         parameters={'n_estimators':[1,10,100]}

         from sklearn.ensemble import GradientBoostingClassifier
         from sklearn.model_selection import GridSearchCV
         gb=GradientBoostingClassifier(random_state=47)
         gb=GridSearchCV(gb,parameters,cv=3,scoring='accuracy')
         gb.fit(x_train,y_train)
         print(gb.best_params_)
         print(gb.best_score_)

         {'n_estimators': 100}
         0.9563068695555201

In [84]: gb=GradientBoostingClassifier(random_state=47,n_estimators=100)
         gb.fit(x_train,y_train)
         pred=gb.predict(x_test)
         print("Accuracy score: ",accuracy_score(y_test,pred)*100)
         print('Cross validation score: ',cross_val_score(gb,x,y,cv=3,scoring='accuracy').mean()*100)
         print('Classification report: \n')
         print(classification_report(y_test,pred))
         print('Confusion matrix: \n')
         print(confusion_matrix(y_test,pred))
```

```
Accuracy score:  96.77272727272728
Cross validation score:  53.786441541030946
Classification report:

              precision    recall  f1-score   support

         1.0       0.93      0.99      0.96      1179
         2.0       0.93      0.79      0.85       204
         3.0       0.99      0.93      0.96       395
         4.0       0.98      0.97      0.97       872
         5.0       0.99      0.98      0.98      1750

    accuracy                           0.97      4400
   macro avg       0.96      0.93      0.95      4400
weighted avg       0.97      0.97      0.97      4400

Confusion matrix:

[[1168    3    1    2    5]
 [  30  161    2    9    2]
 [  17    3  369    5    1]
 [   5    6    1  842   18]
 [  32    0    0    0 1718]]
```

Figure 5: Scores for best fitted model using Random Forest Classifier

We got our best scores using Random Forest Classifier. By comparing it with results and metrics of other algorithm we came to a conclusion that it will be the best fit to our dataset.

CONCLUSION

- Key Findings and Conclusions of the Study

- After using different models for classification, we concluded that Random Forest Classifier was best suited to train the dataset for Rating Prediction Analysis.
- The dataset was imbalanced in nature.
- Some data had to be omitted from the datasets in order to remove data irrelevant to the dataset.

- Learning Outcomes of the Study in respect of Data Science

- As the datasets was huge it created problems when we tried to manipulate it or cleanse it. As when we tried to cleanse it we had to take care that data doesn't loose its relevancyand structure. So, we tried to remove much of the unnecessary data from the datasets that was in our reach during the given time.
- And finally, we came to know that the best algorithm used to train the machine for this the dataset is Random Forest Classifier as all the values along the metrics were highest.

- Limitations of this work and Scope for Future Work

- More time consumption during hyperparameter tuning for both models, as the data was large.
- Less number of parameters were used during tuning.
- Scrapping of data from different websites were of different process and the length of data were differing in most cases so I sticked to Amazon and Scrapped data which are famous in the site.
- Some of the reviews were bad and the text had more wrong information about the product.