Name of the Project

RATING PREDICTION PROJECT

Submitted by:

KHUSHBOO GUPTA

ACKNOWLEDGMENT

First and foremost, I would like to thank Flip Robo Technologies to provide me a chance to work on this project. It was a great experience to work on this project under your guidance.

I would like to present my gratitude to the following websites:

- Zendesk
- Kaggle
- Datatrained Notes
- Sklearn.org
- Crazyegg
- Towards data science
- Stackoverflow.com
- Medium.com
- Amazon.com

These websites were of great help and due to this, I was able to complete my project effectively and efficiently.

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

Basic EDA concepts and classification algorithms must be known to work on this project. As the independent variable is in text format, we need to used NLP to execute our model learning properly. One must know about word embedding and tokenizing. How can we decide whether a review is 5 star or 1 star.

• Review of Literature

The rise in E — commerce, has brought a significant rise in the importance of customer reviews. There are hundreds of review sites online and massive amounts of reviews for every product. Customers have changed their way of shopping and according to a recent survey, 70 percent of customers say that they use rating filters to filter out low rated items in their searches.

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 Google, Amazon and Yelp!

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 theatre, 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.

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

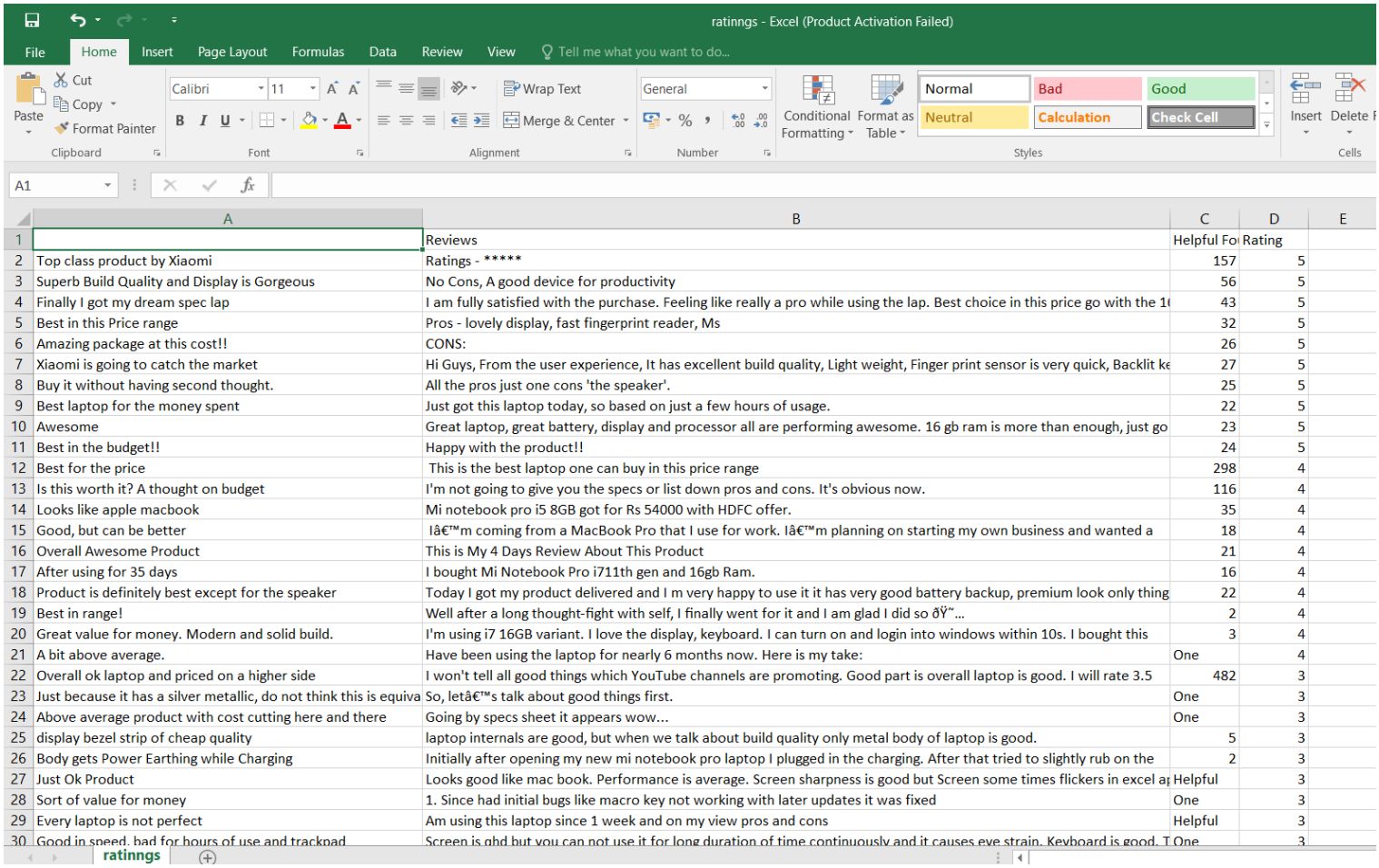
Analytical Problem Framing

- Data Sources and their formats

  In this project our first task is to scrap the data from an e-commerce website. My source of data is Amazon website from where I scrap the reviews and ratings along with review title of different products in different categories like laptop, smartwatch, headphones etc,

  To scrap the data, we will use the selenium. With the help of it scrap the data and save it in a csv file format for future use.

  It contains 4 columns and 3328 rows. Our dataset format is textual which need to be convert and visualize it. Choice of words make a review helpful and good or bad.



	A	B	C	D	E
1		Reviews	Helpful Fo	Rating	
2	Top class product by Xiaomi	Ratings - *****	157	5	
3	Superb Build Quality and Display is Gorgeous	No Cons, A good device for productivity	56	5	
4	Finally I got my dream spec lap	I am fully satisfied with the purchase. Feeling like really a pro while using the lap. Best choice in this price go with the 1(	43	5	
5	Best in this Price range	Pros - lovely display, fast fingerprint reader, Ms	32	5	
6	Amazing package at this cost!!	CONS:	26	5	
7	Xiaomi is going to catch the market	Hi Guys, From the user experience, It has excellent build quality, Light weight, Finger print sensor is very quick, Backlit ke	27	5	
8	Buy it without having second thought.	All the pros just one cons 'the speaker'.	25	5	
9	Best laptop for the money spent	Just got this laptop today, so based on just a few hours of usage.	22	5	
10	Awesome	Great laptop, great battery, display and processor all are performing awesome. 16 gb ram is more than enough, just go	23	5	
11	Best in the budget!!	Happy with the product!!	24	5	
12	Best for the price	This is the best laptop one can buy in this price range	298	4	
13	Is this worth it? A thought on budget	I'm not going to give you the specs or list down pros and cons. It's obvious now.	116	4	
14	Looks like apple macbook	Mi notebook pro i5 8GB got for Rs 54000 with HDFC offer.	35	4	
15	Good, but can be better	Iâ€™m coming from a MacBook Pro that I use for work. Iâ€™m planning on starting my own business and wanted a	18	4	
16	Overall Awesome Product	This is My 4 Days Review About This Product	21	4	
17	After using for 35 days	I bought Mi Notebook Pro i711th gen and 16gb Ram.	16	4	
18	Product is definitely best except for the speaker	Today I got my product delivered and I m very happy to use it it has very good battery backup, premium look only thing	22	4	
19	Best in range!	Well after a long thought-fight with self, I finally went for it and I am glad I did so ðŸ˜...	2	4	
20	Great value for money. Modern and solid build.	I'm using i7 16GB variant. I love the display, keyboard. I can turn on and login into windows within 10s. I bought this	3	4	
21	A bit above average.	Have been using the laptop for nearly 6 months now. Here is my take:	One	4	
22	Overall ok laptop and priced on a higher side	I won't tell all good things which YouTube channels are promoting. Good part is overall laptop is good. I will rate 3.5	482	3	
23	Just because it has a silver metallic, do not think this is equiva	So, letâ€™s talk about good things first.	One	3	
24	Above average product with cost cutting here and there	Going by specs sheet it appears wow...	One	3	
25	display bezel strip of cheap quality	laptop internals are good, but when we talk about build quality only metal body of laptop is good.	5	3	
26	Body gets Power Earthing while Charging	Initially after opening my new mi notebook pro laptop I plugged in the charging. After that tried to slightly rub on the	2	3	
27	Just Ok Product	Looks good like mac book. Performance is average. Screen sharpness is good but Screen some times flickers in excel ar	Helpful	3	
28	Sort of value for money	1. Since had initial bugs like macro key not working with later updates it was fixed	One	3	
29	Every laptop is not perfect	Am using this laptop since 1 week and on my view pros and cons	Helpful	3	
30	Good in speed, bad for hours of use and trackpad	Screen is ghd but you can not use it for long duration of time continuously and it causes eye strain. Keyboard is good. T	One	3	

- ## Dataset Description

The data set contains 3328 samples of different reviews. The data sample contain 4 fields which includes ‘Title’, Review’, ‘Helpful Found’, ‘Rating’.

The rating divided into 5 categories 1 to 5, 5 means an outstanding product and with the decreasing rating product quality descends.

The data set includes:

- Title: It contains the title of any review, or whole review in one line
- Reviews: It contains the full review of a product by a customer.
- Helpful Found: How many helpful a review is getting from different peoples.

- ## Libraries Used

I am using different libraries to explore the datatset.

1. Pandas – It is used to load and store the dataset. We can discuss the dataset with the pandas different attributes like .info, .columns, .shape
2. Seaborn – It is used to plot the different types of plots like catplot, lineplot, countplot and more to have a better visualization of the dataset.
3. Matplotlib.pyplot – It helps to give a proper description to the plotted graph by seaborn and make our graph more informative.
4. Numpy – It is the library to perform the numerical analysis to the dataset
5. Nltk – It uses for natural learning processing algorithms.

Load the Dataset

Importing the libraries

In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

Loading the dataset

In [2]:
```
df=pd.read_csv(r'F:\Internship - Data Science\Rating-Prediction\ratinngs.csv')
df.head()
```

Out[2]:

	Title	Reviews	Helpful Found	Rating
0	Top class product by Xiaomi	Ratings - *****\nDesign - 5\nBuild Quality - 5...	157	5
1	Superb Build Quality and Display is Gorgeous	No Cons, A good device for productivity	56	5
2	Finally I got my dream spec lap	I am fully satisfied with the purchase. Feelin...	43	5
3	Best in this Price range	Pros - lovely display, fast fingerprint reader...	32	5
4	Amazing package at this cost!!	CONS:\nLow sound from speakers, but i mostly u...	26	5

We have successfully load our dataset for our further processes.

Checking the Attributes

- First & last five rows the dataset
- Shape of the dataset
- Columns present in the dataset
- Brief info about the dataset
- Datatype of each column
- Null values present in the dataset
- Number of unique values present in each column

In [4]:
```
df.shape  #total rows & columns present in the dataset
```

Out[4]:
```
(3328, 4)
```

3328 rows and 4 columns

In [5]:
```
df.columns  #columns in the dataset
```

Out[5]:
```
Index(['Title', 'Reviews', 'Helpful Found', 'Rating'], dtype='object')
```

In [6]:
```
df.dtypes  #datatypes of each column
```

Out[6]:
```
Title            object
Reviews          object
Helpful Found    object
Rating            int64
dtype: object
```

In [7]:
```
df.nunique()  #total unique values in each column
```

Out[7]:
```
Title            2695
Reviews          2777
Helpful Found     228
Rating              5
dtype: int64
```

In [8]:
```
df.info()  # a brief info about the dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3328 entries, 0 to 3327
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Title          3319 non-null   object
 1   Reviews        3218 non-null   object
 2   Helpful Found  3328 non-null   object
 3   Rating         3328 non-null   int64
dtypes: int64(1), object(3)
memory usage: 104.1+ KB
```

Null Values

```
In [9]: df.isnull().sum()
```

```
Out[9]: Title             9
        Reviews         110
        Helpful Found     0
        Rating            0
        dtype: int64
```

Dataset contains the null values which has to be handled

```
In [10]: sns.heatmap(df.isnull())   #plotting the null values using heatmap
```

```
Out[10]: <AxesSubplot:>
```

Dropping the null values

```
In [11]: df.dropna(axis=0,inplace=True)
```

```
In [12]: df.isnull().sum()
```

```
Out[12]: Title             0
         Reviews           0
         Helpful Found     0
         Rating            0
         dtype: int64
```

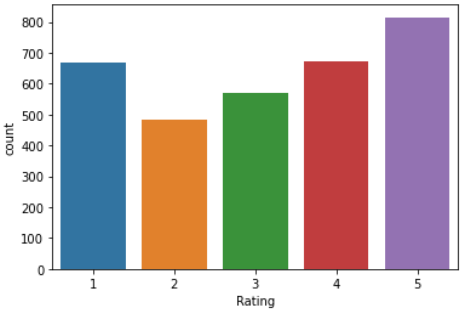
Now we have checked the attributes for the dataset and get a rough idea about the dataset like the no of rows & columns, datatype & null values in the dataset. There are null values present in the title and reviews columns, as they are in textual format and not in much number we can drop the rows containing null values. Dropping the null values make our dataset of 3210 rows and now we can move further.

EXPLORATORY DATA ANALYSIS

Using the countplot we are trying to understand the balancing between each class in target variable. As there is not too difference between the probability we can use this data for our model learning, no need to balance the data.

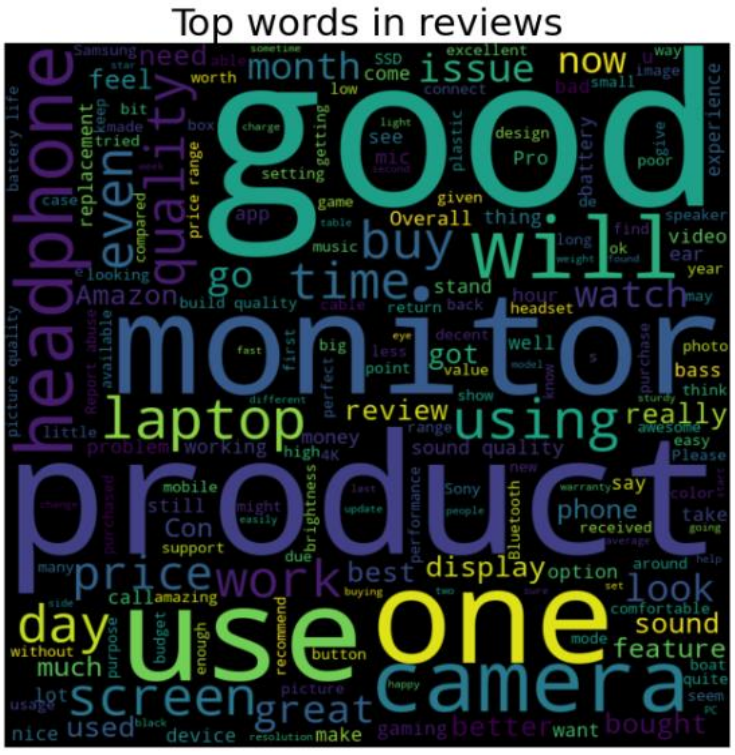
```
In [14]: sns.countplot(df['Rating'])
```

```
Out[14]: <AxesSubplot:xlabel='Rating', ylabel='count'>
```



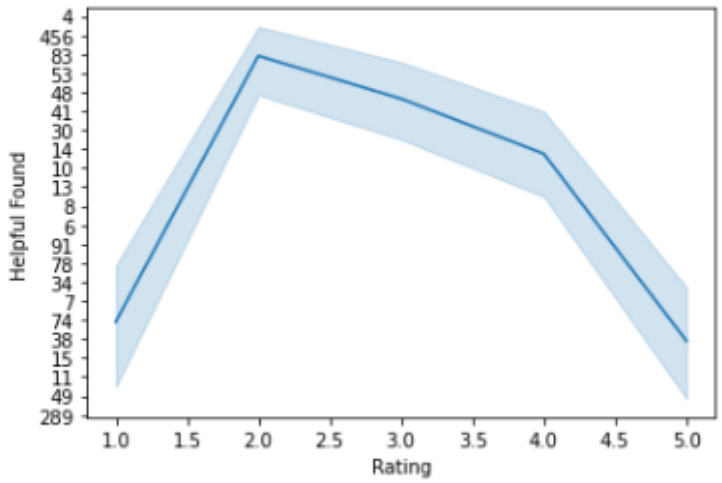
We have most of reviews having rating 5 followed 4 rating, there is no much difference between the ratings so we can go ahead with the data

```
def wordCloud_generator(data, title=None):
    wordcloud = WordCloud(width = 800, height = 800,
                          background_color ='black',
                          min_font_size = 10
                         ).generate(" ".join(data.values))
    plt.figure(figsize = (8, 8), facecolor = None)
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis("off")
    plt.tight_layout(pad = 0)
    plt.title(title,fontsize=30)
    plt.show()
wordCloud_generator(df['Reviews'], title="Top words in reviews")
```



Most appearing words in the review section

```
In [17]: sns.lineplot(df['Rating'],df['Helpful Found'])
Out[17]: <AxesSubplot:xlabel='Rating', ylabel='Helpful Found'>
```



Most of the helfuls either goes to lower ratings or to 5 star rating

Cleaning the comment column

In the reviews and title column we have different types of comments but it contains some special characters and other things which needs to be removed to have a better perception. We are going to remove the extra spacing, symbols etc and keep only the alphabets using regexp_tokenize module.

Data Preprocessing

In [18]:
```
#data preprocessing for review column

# Convert all messages to lower case
df['Reviews'] = df['Reviews'].str.lower()

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


df['Reviews'] = df['Reviews'].apply(lambda x: ' '.join(
    term for term in x.split() if term not in string.punctuation))

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

lem=WordNetLemmatizer()
df['Reviews'] = df['Reviews'].apply(lambda x: ' '.join(
 lem.lemmatize(t) for t in x.split()))
```

In [19]:
```
df.head()
```

Out[19]:

	Title	Reviews	Helpful Found	Rating
0	Top class product by Xiaomi	rating ***** design numbr build quality numbr ...	157	5
1	Superb Build Quality and Display is Gorgeous	cons, good device productivity	56	5
2	Finally I got my dream spec lap	fully satisfied purchase. feeling like really ...	43	5
3	Best in this Price range	pro lovely display, fast fingerprint reader, m...	32	5
4	Amazing package at this cost!!	cons: low sound speakers, mostly use earphone ...	26	5

In [20]:
```
# Keeping only text with letters a to z, 0 to 9 and words like can't, don't, couldn't etc

from nltk.tokenize import regexp_tokenize
df.Reviews = df.Reviews.apply(lambda x: ' '.join(regexp_tokenize(x,"[a-z']+")))
df.head()
```

Out[20]:

	Title	Reviews	Helpful Found	Rating
0	Top class product by Xiaomi	rating design numbr build quality numbr displa...	157	5
1	Superb Build Quality and Display is Gorgeous	cons good device productivity	56	5
2	Finally I got my dream spec lap	fully satisfied purchase feeling like really p...	43	5
3	Best in this Price range	pro lovely display fast fingerprint reader m o...	32	5
4	Amazing package at this cost!!	cons low sound speakers mostly use earphone th	26	5

In [21]:
```
#data preproccessing for title column

# Convert all messages to lower case
df['Title'] = df['Title'].str.lower()

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


df['Title'] = df['Title'].apply(lambda x: ' '.join(
    term for term in x.split() if term not in string.punctuation))

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

lem=WordNetLemmatizer()
df['Title'] = df['Title'].apply(lambda x: ' '.join(
 lem.lemmatize(t) for t in x.split()))

df.Title = df.Title.apply(lambda x: ' '.join(regexp_tokenize(x,"[a-z']+")))
df.head()
```

Out[21]:

	Title	Reviews	Helpful Found	Rating
0	top class product xiaomi	rating design numbr build quality numbr displa...	157	5
1	superb build quality display gorgeous	cons good device productivity	56	5
2	finally got dream spec lap	fully satisfied purchase feeling like really p...	43	5
3	best price range	pro lovely display fast fingerprint reader m o...	32	5
4	amazing package cost	cons low sound speakers mostly use earphone th...	26	5

Statistical Summary & Correlation

We will describe the statistical summary of the dataset and find the correlation of each column.

Statistical Summary

In [26]:
```
df.describe()
```

Out[26]:

	Helpful Found	Rating
count	3210.000000	3210.00000
mean	27.456386	3.15109
std	133.031273	1.47818
min	1.000000	1.00000
25%	1.000000	2.00000
50%	2.000000	3.00000
75%	10.000000	5.00000
max	4303.000000	5.00000

there is a large difference between 75% and max in helpful columnm i.e. a larger number of outliers are present and also high skewness present in this column

Correlation

In [27]:
```
corr=df.corr()
print(corr)
print(sns.heatmap(corr, cmap='twilight',annot=True))
```

```
                Helpful Found    Rating
Helpful Found       1.000000 -0.007083
Rating             -0.007083  1.000000
AxesSubplot(0.125,0.125;0.62x0.755)
```

Skewness

```
In [28]: df.skew()
```

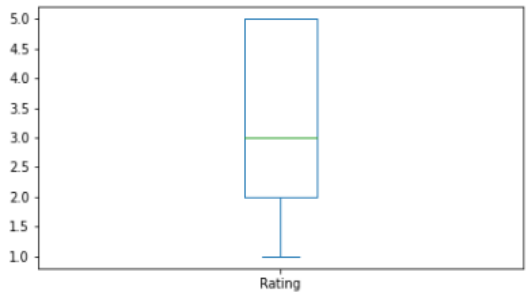
```
Out[28]: Helpful Found    16.083277
         Rating           -0.175348
         dtype: float64
```

```
In [29]: df.drop('Helpful Found',axis=1,inplace=True)  #dropping the column as it is non-relevant to target column
```

Outliers

```
In [30]: df.plot(kind='box',subplots=True,layout=(2,2),figsize=(15,8))
```

```
Out[30]: Rating    AxesSubplot(0.125,0.536818;0.352273x0.343182)
         dtype: object
```



No outliers present in the dataset

- We have 3210 rows in the dataset
- Being only categorical variables in the columns there will be no outliers.
- Also skewness present in the helpful found column, so we drop this..

MODEL BUILDING

Before moving forward towards the model training we have a challenge that our independent variables are in textual form and model can't understand texts so we have to convert them into vectors using TF-IDF technique. It will convert the text into vectors which is easily understandable by model for training.

Converting the text into vectors using TF-IDF

```
In [32]: from sklearn.feature_extraction.text import TfidfVectorizer
         tf_vec = TfidfVectorizer(max_features = 10000, stop_words='english')
         features = tf_vec.fit_transform(df['Reviews'],df['Title'])
         x = features
```

```
In [33]: y=df['Rating']
```

We will import important libraries for the building the ML model and defining the different models for our easiness. Finding the best random state for the train test split.

In [34]:
```
#defining the models

lg=LogisticRegression()
rdc=RandomForestClassifier()
dtc=DecisionTreeClassifier()
knc=KNeighborsClassifier()
ad=AdaBoostClassifier()
gb=GradientBoostingClassifier()
```

Finding the best random state

In [35]:
```
model=[lg,rdc,dtc,knc,ad,gb]
maxAccu=0
bestRS=0
for i in range(40,60):
    x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=i,test_size=.30)
    lg.fit(x_train,y_train)
    pred=lg.predict(x_test)
    acc=accuracy_score(y_test,pred)
    if acc>maxAccu:
        maxAccu=acc
        bestRS=i
print('Best Accuracy score is', maxAccu , 'on random state', bestRS)
```

```
Best Accuracy score is 0.3541017653167186 on random state 58
```

In [36]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=58,test_size=.30)
```

Classification Algorithms

We have use seven different regression algorithms to find the best model for our problem.

- **Logistic Regression**
  - from sklearn.linear_model import LogisticRegression
- **Decision Tree Classifier**
  - from sklearn.tree import DecisionTreeClassifier
- **KNN Classifier**
  - from sklearn.neighbors import KNeighborsClassifier
- **Random Forest Classifier**
  - from sklearn.ensemble import RandomForestClassifier
- **Multinomial NB**
  - from sklearn.naive_bayes import MultinomialNB
- **AdaBoost Classifier**

- from sklearn.ensemble import AdaBoostClassifier

- **GradientBoosting Classifier**

- from sklearn.ensemble import GradientBoostingClassifier

Let's see the different models accuracy at once.

MODEL	ACCURACY
Logistic Regression	0.3541017653167186
Decision Tree Classifier	0.3250259605399723
Random Forest Classifier	0.3509865005192108
KNN Classifier	0.3167185877466251 5
Multinomial NB	0.2824506749740395
Adaboost Classifier	0.3333333333333333
GradientBoost Classifer	0.3426791277258567

Logistic Regression

In [37]:
```
lg.fit(x_train,y_train)
pred1=lg.predict(x_test)
acc=accuracy_score(y_test,pred1)
print('Accuracy Score: ',acc)
print('Confusion Matrix: ' ,'\n',confusion_matrix(y_test,pred1))
print('Classification Report: ','\n',classification_report(y_test,pred1))
```

```
Accuracy Score:  0.3541017653167186
Confusion Matrix:
 [[ 58  11  15  35  65]
 [ 25  27  26  27  50]
 [ 24   9  36  34  64]
 [ 22  10  15  69  81]
 [ 36   3  20  50 151]]
Classification Report:
              precision    recall  f1-score   support

           1       0.35      0.32      0.33       184
           2       0.45      0.17      0.25       155
           3       0.32      0.22      0.26       167
           4       0.32      0.35      0.33       197
           5       0.37      0.58      0.45       260

    accuracy                           0.35       963
   macro avg       0.36      0.33      0.33       963
weighted avg       0.36      0.35      0.34       963
```

Decision Tree Classifier

In [38]:
```
dtc.fit(x_train,y_train)
pred2=dtc.predict(x_test)
acc=accuracy_score(y_test,pred2)
print('Accuracy Score: ',acc)
print('Confusion Matrix: ' ,'\n',confusion_matrix(y_test,pred2))
print('Classification Report: ','\n',classification_report(y_test,pred2))
```

```
Accuracy Score:  0.32502596053997923
Confusion Matrix:
 [[ 52  17  15  33  67]
 [ 22  35  22  25  51]
 [ 31  17  53  24  42]
 [ 42  21  22  60  52]
 [ 43  17  26  61 113]]
Classification Report:
              precision    recall  f1-score   support

           1       0.27      0.28      0.28       184
           2       0.33      0.23      0.27       155
           3       0.38      0.32      0.35       167
           4       0.30      0.30      0.30       197
           5       0.35      0.43      0.39       260

    accuracy                           0.33       963
   macro avg       0.33      0.31      0.32       963
weighted avg       0.33      0.33      0.32       963
```

Rabdom Forest Classifier

In [39]:
```
rdc.fit(x_train,y_train)
pred4=rdc.predict(x_test)
acc=accuracy_score(y_test,pred4)
print('Accuracy Score: ',acc)
print('Confusion Matrix: ' ,'\n',confusion_matrix(y_test,pred4))
print('Classification Report: ','\n',classification_report(y_test,pred4))
```

```
Accuracy Score:  0.3509865005192108
Confusion Matrix:
 [[ 44   9  14  30  87]
 [ 25  28  11  25  66]
 [ 20   5  40  31  71]
 [ 26   6  24  64  77]
 [ 39   7  10  42 162]]
Classification Report:
               precision    recall  f1-score   support

           1       0.29      0.24      0.26       184
           2       0.51      0.18      0.27       155
           3       0.40      0.24      0.30       167
           4       0.33      0.32      0.33       197
           5       0.35      0.62      0.45       260

    accuracy                           0.35       963
   macro avg       0.38      0.32      0.32       963
weighted avg       0.37      0.35      0.33       963
```

KNN Classifier

In [40]:
```
knc.fit(x_train,y_train)
pred5=knc.predict(x_test)
acc=accuracy_score(y_test,pred5)
print('Accuracy Score: ',acc)
print('Confusion Matrix: ' ,'\n',confusion_matrix(y_test,pred5))
print('Classification Report: ','\n',classification_report(y_test,pred5))
```

```
Accuracy Score:  0.31671858774662515
Confusion Matrix:
 [[ 20   6   1  66  91]
 [  3  17   0  62  73]
 [  4   0  20  75  68]
 [  3   1   1 103  89]
 [  4   2   3 106 145]]
Classification Report:
               precision    recall  f1-score   support

           1       0.59      0.11      0.18       184
           2       0.65      0.11      0.19       155
           3       0.80      0.12      0.21       167
           4       0.25      0.52      0.34       197
           5       0.31      0.56      0.40       260

    accuracy                           0.32       963
   macro avg       0.52      0.28      0.26       963
weighted avg       0.49      0.32      0.28       963
```

AdaBoost Classifier

In [41]:
```
ad.fit(x_train,y_train)
pred3=ad.predict(x_test)
acc=accuracy_score(y_test,pred3)
print('Accuracy Score: ',acc)
print('Confusion Matrix: ' ,'\n',confusion_matrix(y_test,pred3))
print('Classification Report: ','\n',classification_report(y_test,pred3))
```

```
Accuracy Score:  0.2824506749740395
Confusion Matrix:
 [[ 18  14  12  40 100]
 [ 11  24  15  30  75]
 [  8  12  24  34  89]
 [ 10  24  19  44 100]
 [ 13  15  12  58 162]]
Classification Report:
              precision    recall  f1-score   support

           1       0.30      0.10      0.15       184
           2       0.27      0.15      0.20       155
           3       0.29      0.14      0.19       167
           4       0.21      0.22      0.22       197
           5       0.31      0.62      0.41       260

    accuracy                           0.28       963
   macro avg       0.28      0.25      0.23       963
weighted avg       0.28      0.28      0.25       963
```

GradientBoost Classifer

In [42]:
```
gb.fit(x_train,y_train)
pred6=gb.predict(x_test)
acc=accuracy_score(y_test,pred6)
print('Accuracy Score: ',acc)
print('Confusion Matrix: ' ,'\n',confusion_matrix(y_test,pred6))
print('Classification Report: ','\n',classification_report(y_test,pred6))
```

```
Accuracy Score:  0.3333333333333333
Confusion Matrix:
 [[ 41   9  18  29  87]
 [ 16  29  18  24  68]
 [ 20  13  40  27  67]
 [ 27  17  17  60  76]
 [ 40   9  22  38 151]]
Classification Report:
              precision    recall  f1-score   support

           1       0.28      0.22      0.25       184
           2       0.38      0.19      0.25       155
           3       0.35      0.24      0.28       167
           4       0.34      0.30      0.32       197
           5       0.34      0.58      0.43       260

    accuracy                           0.33       963
   macro avg       0.34      0.31      0.31       963
weighted avg       0.34      0.33      0.32       963
```

Hence, we are getting the best accuracy score through the Logistic Regression Model. We will go ahead with this to find the cross val score and hypermeter tuning.

Cross Val Score & Hypermeter Tuning

Cross-validation provides information about how well a classifier generalizes, specifically the range of expected errors of the classifier. Cross Val Score tells how the model is generalized at a particular cross validation.

At CV=4 we get the best results i.e. the Random Forest Classifier more generalized at cv=4, so we calculate the hyper parameters at this value.

We will find which parameters of random forest classifier are the best foe our model. We will do this using Grid Search CV method & also calculate the accuracy score at those best parameters.

Cross Val score

In [44]:
```
from sklearn.model_selection import cross_val_score
for i in range(3,7):
    cr=cross_val_score(lg,x,y,cv=i)
    cr_mean=cr.mean()
    print("at cv= ", i)
    print('cross val score = ',cr_mean*100)
```

```
at cv=  3
cross val score =  23.426791277258566
at cv=  4
cross val score =  29.46568044397102
at cv=  5
cross val score =  29.03426791277258
at cv=  6
cross val score =  28.598130841121495
```

Hypermeter Tuning

In [45]:
```
from sklearn.model_selection import GridSearchCV
# creating parameters
param={'penalty':['l1', 'l2', 'elasticnet', 'none'],
       'solver':['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']}

GCV=GridSearchCV(lg,param,cv=4,scoring='accuracy')
GCV.fit(x_train,y_train)
GCV.best_params_
```

Out[45]:
```
{'penalty': 'l2', 'solver': 'liblinear'}
```

In [46]:
```
GCV_pred=GCV.best_estimator_.predict(x_test)
accuracy_score(y_test,GCV_pred)
```

Out[46]:
```
0.3561786085150571
```

Saving the Model

Saving the best model – Logistic Regression in this case for future predictions. Let's see what are the actual test data and what our model predicts.

Saving the model

In [48]:
```
import pickle
filename='rating.pkl'
pickle.dump(lg,open(filename,'wb'))
```

Conclusion

In [49]:
```
a=np.array(y_test)
pred=np.array(GCV_pred)
malignant=pd.DataFrame({'Actual':a,'Predicted':pred})
malignant
```

Out[49]:

	Actual	Predicted
0	5	1
1	5	4
2	2	3
3	5	2
4	5	5
...	...	...
958	1	1
959	1	4
960	4	5
961	5	5
962	3	2

963 rows × 2 columns

Hence up to some good extensions our model predicted so well.

CONCLUSION

Conclusion of the Study

The results of this study suggest following outputs which might be useful to predict the rating of reviews written by customers:

- Word are the one make comment a 5 star or 1 star. Our model has to be that précised to understand the difference of words and predict whether it is a good review or bad about the product.

- Using such models, we can improve the services and satisfy the customer in a better way.

- Learning Outcomes of the Study in respect of Data Science
  - This projects teaches so many new things to me. I get to know new modules, new techniques to handle the dataset.
  - Data cleaning with new method.
  - New modules like wordcloud through which we get a better understanding of words.
  - Converting the comment into vector for proper training and machine understanding.