PROJECT NAME:-

RATINGS PREDICTION PROJECT

SUBMITTED BY:-

ATUL DAHIMA

INTRODUCTION

PEOBLEM STATEMENT:-

- RATINGS PREDICTION
-
- PROJECT
-
- We have a client who has a website where people write different reviews for technical products.
- Now they are adding a new feature to their website i.e. The reviewer will have to add stars(rating)
- as well with the review. The rating is out 5 stars and it only has 5 options available 1 star, 2 stars,
- 3 stars, 4 stars, 5 stars. Now they want to predict ratings for the reviews which were written in the
- past and they don't have a rating. So, we have to build an application which can predict the rating
- by seeing the review.
-
- Data Collection Phase
-
- You have to scrape at least 20000 rows of data. You can scrape more data as well, it's up to you.
- more the data better the model
- In this section you need to scrape the reviews of different laptops, Phones, Headphones, smart
- watches, Professional Cameras, Printers, Monitors, Home theater, Router from different ecommerce websites.
- Basically, we need these columns
-
- 1) reviews of the product.
-
- 2) rating of the product.
- You can fetch other data as well, if you think data can be useful or can help in the project. It
- completely depends on your imagination or assumption.
- Hint:
-
- • Try to fetch data from different websites. If data is from different websites, it will help our
- model to remove the effect of over fitting.
-
- • Try to fetch an equal number of reviews for each rating, for example if you are fetching
- 10000 reviews then all ratings 1,2,3,4,5 should be 2000. It will balance our data set.
-
- • Convert all the ratings to their round number, as there are only 5 options for rating i.e.,
- 1,2,3,4,5. If a rating is 4.5 convert it 5.

-
- Model Building Phase
- After collecting the data, you need to build a machine learning model. Before model building do
- all data preprocessing steps involving NLP. Try different models with different hyper parameters
- and select the best model.
-
- Follow the complete life cycle of data science. Include all the steps like
-
- 1. Data Cleaning
-
- 2. Exploratory Data Analysis
-
- 3. Data Preprocessing
-
- 4. Model Building
-
- 5. Model Evaluation
-
- 6. Selecting the best mode**Comment text:** This column contains the comments extracted from various social media platforms.

START MODEL BUILDING:-

First we will scrap the data for our from flipkart

We will scrap ratings and reviews for a product and then we will predict the ratings on the basis of the review

First import the required libraries for the web scraping and the chrome driver

In [1]:
```
import pandas as pd
import selenium
from selenium import webdriver
import time
from selenium.common.exceptions import NoSuchElementException     #importing exception


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

In [2]:
```
#Let's first connect to web driver
driver=webdriver.Chrome(r"C:\Users\ankus\Downloads\chromedriver_win32 (5)\chromedriver.exe")
```

Now we will get the url of the product of which we have to scrap the review and ratings and then we will scrap the required details using xpath

In [4]:
```
#scrap details from yatra.com
url="https://www.flipkart.com/mi-r4cm-300-mbps-router/product-reviews/itm4ed0ab10f43e5?pid=RTRFZ9XMNKDBSHCY&lid=LSTRTRFZ9XMNKDBSH
driver.get(url)
```

In [5]:
```
for page in range(1,184):
    for i in driver.find_elements_by_xpath('//div[@class="_3LWZlK _1BLPMq"]'):

        try:
            ratings.append(i.text)

        except:
            ratings.append(" ")

    for i in driver.find_elements_by_xpath('//div[@class="t-ZTKy"]'):
        try:
            review.append(i.text)

        except:
            review.append(" ")

    nxt_button =driver.find_elements_by_xpath("//a[@class='_1LKTO3']")



    try:
        driver.get(nxt_button[1].get_attribute('href'))

    except:
        driver.get(nxt_button[0].get_attribute('href'))
```

Now make the dataframe of the scrapped data and start the model building

In [13]:
```
data=pd.DataFrame()
data['Reviews']=review
data['Ratings']=ratings
data
```

Out[13]:

	Reviews	Ratings
0	Main Features\n● Four high gain antenna for st...	5
1	First of all I am not recommending to buy this...	4
2	These Router is just awesome.. I have used Mi ...	5
3	Smart features,looks is also impressive and ap...	5
4	Good value for money product\n\nRange-good\n\n...	3
...	...	...
1332	Good product but shot ranges	5
1333	Good router.☺	4
1334	Super speed.	3
1335	Nice Router	5
1336	Nice product	5

1337 rows × 2 columns

MODEL BUILDING

- *Mathematical/ Analytical Modeling of the Problem*

In [15]:
```
data.head()
```

Out[15]:

	Reviews	Ratings
0	Main Features\n● Four high gain antenna for st...	5
1	First of all I am not recommending to buy this...	4
2	These Router is just awesome.. I have used Mi ...	5
3	Smart features,looks is also impressive and ap...	5
4	Good value for money product\n\nRange-good\n\n...	3

In [16]:
```
#import all the required libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [17]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1337 entries, 0 to 1336
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   Reviews  1337 non-null   object
 1   Ratings  1337 non-null   object
dtypes: object(2)
memory usage: 21.0+ KB
```

We will start the model building with importing some important required libraries like:- Pandas, Numpy, Seaborn ,Matplotlib etc. and import the datset using pandas (train and test dataset separately)

Few libraries are used initially for different purpose like:-

- *Numpy*:- for mathematical operation
- *Pandas*:- for data preprocessing
- *Matplotlib and Seaborn*:- for data visualization

In [18]:
```
#data description
data.describe()
```

Out[18]:

	Reviews	Ratings
count	1337	1337
unique	1039	3
top	Good	5
freq	79	872

In [20]:
```
from nltk.stem import WordNetLemmatizer
import nltk
from nltk.corpus import  stopwords
import string
```

In [22]:
```
data['length'] =data['Reviews'].str.len()
data.head()
```

Out[22]:

	Reviews	Ratings	length
0	Main Features\n• Four high gain antenna for st...	5	376
1	First of all I am not recommending to buy this...	4	510
2	These Router is just awesome.. I have used Mi ...	5	510
3	Smart features,looks is also impressive and ap...	5	113
4	Good value for money product\n\nRange-good\n\n...	3	202

After this check the data description which tells the mathematical information about the dataset i.e. mean , median , mode of the columns , std of the columns and some other values like 25th%, 50th%, 75th% of the columns which helps further in the model building

In above code we again import some libraries like WordLemmatizer for lemmatization and import nltk and download stopwords using nltk and import string

And after the data description import some other important libraries like shown above in the picture for the further processing of the model and check the length of the every row of reviews column

In [23]:
```
# Convert all messages to lower case
data['Reviews'] = data['Reviews'].str.lower()

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

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

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

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


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


data['Reviews'] = data['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'])
data['Reviews'] =data['Reviews'].apply(lambda x: ' '.join(
    term for term in x.split() if term not in stop_words))

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

In above code we change the case of the text and replace some special character , hyperlinks and numbers with their respectives to make the code much easier and shorter than before

```
In [25]: data['clean_length'] =data.Reviews.str.len()
         data.head()
```

Out[25]:

	Reviews	Ratings	length	cReviews	clean_length
0	main feature • four high gain antenna strong s...	5	376	main features\n• four high gain antenna for st...	339
1	first recommending buy item due following reas...	4	510	first of all i am not recommending to buy this...	343
2	router awesome.. used mi router numbrc ordered...	5	510	these router is just awesome.. i have used mi ...	313
3	smart features,looks also impressive app optim...	5	113	smart features,looks is also impressive and ap...	94
4	good value money product range-good speed conn...	3	202	good value for money product\n\nrange-good\n\n...	136

```
In [26]: # Total length removal
         print ('Origian Length',data.length.sum())
         print ('Clean Length',data.clean_length.sum())
```

```
Origian Length 68079
Clean Length 50200
```

Here we check the original length and clean length after doing some cleaning on the comment_text column

```
In [29]: from sklearn.naive_bayes import MultinomialNB
         from sklearn.model_selection import train_test_split
         from sklearn.metrics import accuracy_score, confusion_matrix, classification_report,roc_curve,roc_auc_score,auc
         from sklearn.model_selection import train_test_split
         from sklearn.metrics import accuracy_score,classification_report,confusion_matrix,f1_score
         from sklearn.linear_model import LogisticRegression
         from sklearn.model_selection import cross_val_score,GridSearchCV
         from sklearn.naive_bayes import MultinomialNB
         from sklearn.tree import DecisionTreeClassifier
         from sklearn.ensemble import RandomForestClassifier
         from sklearn.naive_bayes import GaussianNB
         from sklearn.linear_model import LogisticRegression
         from sklearn.svm import SVC
         from sklearn.tree import DecisionTreeClassifier
```

```
In [30]: #  Convert text into vectors using TF-IDF
         from sklearn.feature_extraction.text import TfidfVectorizer
         tf_vec = TfidfVectorizer(max_features = 10000, stop_words='english')
         features = tf_vec.fit_transform(data['Reviews'])
         x = features
```

```
In [32]: y=data['Ratings']
```

```
In [33]: x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=56,test_size=.30)
```

```
In [34]: y_train.shape,y_test.shape
```

```
Out[34]: ((935,), (402,))
```

Now after import the classification model and other required libraries start doing the model building pn the train dataset

In the above code we convert the text into vectors using Tf-idf vectorizer

Now start implementing the classification model for predicting that malignant comment or not

Logistic Regression:-

Logistic Regression

In [35]:
```
# LogisticRegression
LG = LogisticRegression(C=1, max_iter = 3000)

LG.fit(x_train, y_train)

y_pred_train = LG.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = LG.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.6909090909090909
Test accuracy is 0.6840796019900498
[[  0   1  42]
 [  0   2  81]
 [  0   3 273]]
              precision    recall  f1-score   support

           3       0.00      0.00      0.00        43
           4       0.33      0.02      0.04        83
           5       0.69      0.99      0.81       276

    accuracy                           0.68       402
   macro avg       0.34      0.34      0.29       402
weighted avg       0.54      0.68      0.57       402
```

Decision Tree Classifier:-

Decision Tree Classifier

In [36]:
```
# DecisionTreeClassifier
DT = DecisionTreeClassifier()

DT.fit(x_train, y_train)
y_pred_train = DT.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = DT.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.8834224598930481
Test accuracy is 0.5771144278606966
[[  1   8  34]
 [  4  10  69]
 [ 16  39 221]]
              precision    recall  f1-score   support

           3       0.05      0.02      0.03        43
           4       0.18      0.12      0.14        83
           5       0.68      0.80      0.74       276

    accuracy                           0.58       402
   macro avg       0.30      0.31      0.30       402
weighted avg       0.51      0.58      0.54       402
```

Random forest classifier:-

Random Forest Classifier

In [37]:
```
RF = RandomForestClassifier()

RF.fit(x_train, y_train)
y_pred_train = RF.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = RF.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.8834224598930481
Test accuracy is 0.6492537313432836
[[  0   3  40]
 [  0   2  81]
 [  5  12 259]]
              precision    recall  f1-score   support

           3       0.00      0.00      0.00        43
           4       0.12      0.02      0.04        83
           5       0.68      0.94      0.79       276

    accuracy                           0.65       402
   macro avg       0.27      0.32      0.28       402
weighted avg       0.49      0.65      0.55       402
```

Cross validation

In [43]:
```
from sklearn.model_selection import cross_val_score
scr=cross_val_score(LG,x,y,cv=5)
print("Cross Validation score of Logistic Regression model:",scr.mean())
```

```
Cross Validation score of Logistic Regression model: 0.6447258091564649
```

In [44]:
```
scr=cross_val_score(RF,x,y,cv=5)
print("Cross Validation score of Random FOrest:",scr.mean())
```

```
Cross Validation score of Random FOrest: 0.627530884901336
```

In [45]:
```
scr=cross_val_score(svc,x,y,cv=5)
print("Cross Validation score of SVC:",scr.mean())
```

```
Cross Validation score of SVC: 0.6469702051540052
```

In [ ]:

In [ ]:

Save model

In [40]:
```
import joblib
joblib.dump(RF,"Ratings Prediction project.pkl")
```

Out[40]:
```
['Ratings Prediction project.pkl']
```

Finally we will do the cross validation of the model and save the model