FLIP ROBO

RATING PREDICTION

SUBMITTED BY

Abhijit Sarkar

ACKNOWLEDGMENT

I would like to express my gratitude for the opportunity and would like to thank Flip Robo Technologies for giving me an opportunity to work on this Given Project. While going through this project I could Somewhere find the new modes of facts and a conceptual thinking and behaviour of the user behind the scene and in fact how this review is beneficial for the new consumers and manufacturers. I am very grateful to DATA TRAINED team for providing me the adequate Trainings which actually helped me a lot to complete this project in the given time. I took help from Mr. Mohd Kashif where I faced the problem.

During the completion of the project, I found various issues working with NLP and I overcome those problem with the help of Google, You Tube Videos of Mr. Nair, Kaggle and Medium. I Used Sklearn library for solving this data set.

INTRODUCTION

➢ BUSINESS PROBLEM FRAMING.

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.

➢ CONCEPTUAL BACKGROUND OF THE DOMAIN PROBLEM

Recommendation systems are an important unit in today's e-commerce applications, such as targeted advertising, personalized marketing and information retrieval. In recent years, the importance of contextual information has motivated generation of personalized recommendations according to the available contextual information of users. Compared to the traditional systems which mainly utilize users' rating history, review-based recommendation hopefully provides more relevant results to users. We introduce a review-based recommendation approach that obtains contextual information by mining user reviews.

The proposed approach relates to features obtained by analysing textual reviews using methods developed in Natural Language Processing (NLP) and information retrieval discipline to compute a utility function over a given item. An item utility is a measure that shows how much it is preferred according to user's current context. In our system, the context inference is modelled as similarity between the user reviews history and the item reviews history. As an ex: application, we used our method to mine contextual data from customers' reviews of movies and use it to produce review-based rating prediction. The predicted ratings can generate recommendations that are item-based and should appear at the recommended items list in the product page. Our evaluations (surprisingly) suggest that our system can help produce better prediction rating scores in comparison to the standard prediction methods.

➢ REVIEW OF LITERATURE.

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.

➢ MOTIVATION FOR THE PROBLEM UNDERTAKEN

Our goal is to build a prototype that can predict ratings as per the reviews received from the customer. I have scrapped more than 40,000 data from e-commerce website and used that to built a model that can classify the ratings from reviews:

Basically,

we need these columns

1) reviews of the product.

2) rating of the product.

The data set includes:

- Ratings: It includes unique Ids associated with each comment text given.
- Reviews: This column contains the comments extracted from various E-commerce reviews from different customers.

ANALYTICAL PROBLEM FRAMING

✓ Mathematical/ Analytical Modelling of the Problem

In this project, we will develop and evaluate the performance and classification prediction on the trained dataset based on reviews which is provided. Once we get a best fit model, then we would apply the same data set to check the model working capabilities. We will use various classification algorithm to predict our Target Variable.

Let's have an overview of the Few algorithms we will use in predictions:

Logistic Regression:

Logistic regression analysis is valuable for predicting the likelihood of an event. It helps determine the probabilities between any two classes

Decision Tree Regression:

Decision tree regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output.

KNeighbors Classifier:

By default, the KNeighbors Classifier looks for the 5 nearest neighbors. We must explicitly tell the classifier to use Euclidean distance for determining the proximity between neighbouring points.

Random Forest Classifier:

A random forest classifier. A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

TF-IDF

Term frequency-inverse document frequency and it is a measure, used in the fields of information retrieval (IR) and machine learning, that can quantify the importance or relevance of string representations (words, phrases, lemmas, etc)

Stemming is a process that stems or removes last few characters from a word, often leading to incorrect meanings and spelling.
Lemmatization considers the context and converts the word to its meaningful base form, which is called Lemma. For instance, stemming the word 'Caring' would return 'Car'.

✓ Data Sources and their formats

There are two data-set in csv format: train and test dataset.

These data is collected from amazon which is an e-commerce website and we used selenium and beautiful soup to scrape the following. These had two main columns:

1. Ratings: Shares the ratings received from customer.
2. Reviews: This column contains the comments extracted from various E-commerce platforms.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42824 entries, 0 to 42823
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Rattings  42824 non-null  int64
 1   Review    42820 non-null  object
dtypes: int64(1), object(1)
memory usage: 669.2+ KB
```

✓ Data Pre-processing

- Loading Both Data Set and checking top 5 with Data types
- Finding the unique values in categorical and numerical columns.
- Finding Data Missing percentage
- Finding nan values
- Finding duplicated values in columns rows.

Loading Data set

```
In [8]: # Importing dataset excel file using pandas.
        df=pd.read_csv(r"Ratings_scraped.csv")

In [10]: df.head(10)
```

Out[10]:

	Rattings	Review
0	1	Please don't purchase any Lenovo product.4 mon...
1	1	Please don't purchase any Lenovo product.4 mon...
2	1	Please don't purchase any Lenovo product.4 mon...
3	4	the laptop whole package is awesome. although ...
4	4	Pros:Good specs, others have written enough ab...
5	5	(1 month usage update below)I read a lot of re...
6	5	Bought laptop from Amazon for first time. Got ...
7	4	Ok so this is a great gaming laptop no doubt.....
8	1	This is the best gaming performance I've had f...
9	5	Initially thought that the 16" laptop was a bi...

```
In [12]: df.tail(2)
```

Out[12]:

	Rattings	Review
42822	1	Half of the functionality of the app are paid ...
42823	5	Very good product if you are looking for WIFI ...

```
In [14]: df.columns
```

Columns checks:

```
In [14]: df.columns
Out[14]: Index(['Rattings', 'Review'], dtype='object')
```

About Columns

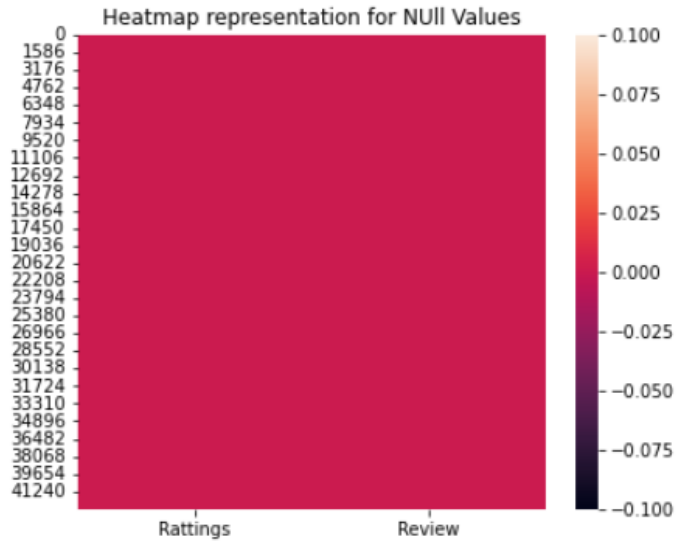
- Rating : Shows rating given by customer
- Review : Comments Shared

Shape Size

```
In [18]: print("About Train Data Set :\n\n","DataType ",type(df),"\nShape ",df.shape,"\nSize ",df.size,)

About Train Data Set :

 DataType  <class 'pandas.core.frame.DataFrame'>
Shape  (42824, 2)
Size  85648
```

Null Values:

```
In [24]: #dropping null values
         df.dropna(inplace=True)

         # after dropping
         df.isnull().sum()

Out[24]: Rattings    0
         Review      0
         dtype: int64

In [35]: plt.figure(figsize=(6,5))
         sns.heatmap(df.isnull())
         plt.title("Heatmap representation for NUll Values")

Out[35]: Text(0.5, 1.0, 'Heatmap representation for NUll Values')
```



Duplicated Rows:

```
In [16]: train_df.duplicated().sum()

Out[16]: 0
```

✓ Data Inputs- Logic- Output Relationships

I used the following to determine the relationship between variable:

- Used univariate, Bivariate, Multi-variate Graph to check for relations.
- Word Clouds was used to check the words in comments provided. Almost all the comments were checked.
- Pie plot along with bar graph is used to check the distribution. (In both the ways)
- Stop Words were added with Extra Words to unfold new relationship.
- Lemmatization was used to standardization
- Finally, Term Frequency-Inverse Document Frequency was used to convert the comments to vectors.

By the Use of these Graph and Tools,

I Uncovered the is a relationship between continuous numerical variable and The Target Variable.

✓ Hardware and Software Requirements and Tools Used

Hardware:

+ i5 9400f
+ 8GB RAM

SOFTWARE USED:

- ***I Used Jupiter Note Book.***
- ***Microsoft Office 2020***
- ***Windows 11 OS***

Library used: To run the program and to build the model we need some basic libraries as follows:

- ***NumPy***
- ***Pandas***
- ***Seaborn***
- ***Matplotlib***
- ***SciPy***
- ***Sklearn***
- ***Pickle***
- ***Imbalance Learn***
- ***NLTK***

1) ***import pandas as pd:***
   Pandas are a popular Python-based data analysis toolkit which can be imported using import pandas as pd. It presents a diverse range of utilities, ranging from parsing multiple file formats to converting an entire data table into a NumPy matrix array. This makes pandas a trusted ally in data science and machine learning.
2) ***import NumPy as np:***
   NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

3) ***import seaborn as sns:***
   Seaborn is a data visualization library built on top of matplotlib and closely integrated with pandas' data structures in Python. Visualization is the central part of Seaborn which helps in exploration and understanding of data.

4) ***Import matplotlib.pyplot as plt:***
   matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

5. scipy:

SciPy is a free and open-source Python library used for scientific computing and technical computing. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

6. Sklearn

Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines.

7. Pickle

"Pickling" is the process whereby a Python object hierarchy is converted into a byte stream, and "unpickling" is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy.

8. IMB learn (SMOTE)

One approach to addressing imbalanced datasets is to oversample the minority class. The simplest approach involves duplicating examples in the minority class, although these examples don't add any new information to the model. Instead, new examples can be synthesized from the existing examples. This is a type of data augmentation for the minority class and is referred to as the Synthetic Minority Oversampling Technique, or SMOTE for short.

9.NLTK

The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing for English written in the Python programming language.

Model/s Development and Evaluation.

1. Identification of possible problem-solving approaches (methods)

Checked Total Numbers of Rows and Column

Checked All Column Name, Type of All Data, Null Values

Checked for special character present in dataset or not

Checked total number of unique values, Information about Data

Checked Description of Data and Dataset, Dropped irrelevant Columns

Replaced special characters and irrelevant data and checked all features through visualization.

Checked Descriptive and correlation of features

Replace Numbers with 'numbr', Punctation, extra space, leadning and trailing white space

Removed \n, removed stop words, Words of Sentence and Calculated length of sentence

Checked the word which are offensive using Word Cloud

Checked the word which are not offensive using Word Cloud

Converted text into vectors using TF-IDF

2. Testing of Identified Approaches (Algorithms)

```
# linear_model,train test and metrics
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix

# Clssifiers
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.linear_model import SGDClassifier
```

```
# cross Validation
from sklearn.model_selection import cross_val_score

# Ensamble
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier

# neural network
from sklearn.neural_network import MLPClassifier

# hyper paramter
from sklearn.model_selection import GridSearchCV
from xgboost import XGBClassifier
```

```
[80]: lg = LogisticRegression()
dtc= DecisionTreeClassifier()
knn= KNeighborsClassifier()
svc= SVC()
xg = XGBClassifier()
sgd= SGDClassifier()
etc= ExtraTreesClassifier()
rfc= RandomForestClassifier()
ada= AdaBoostClassifier()
gbc= GradientBoostingClassifier()
mlp= MLPClassifier()
```

Run and evaluate selected models.

Linear Model

```
******************************
click to scroll output; double click to hide
Training Score 0.9321286781877627
Accuracy Training Score = 0.9321286781877627  Accuracy Test Score = 0.9319243344231667

Training Confusion_Matrix
 [[ 1241     1     3   117    13]
 [    8   327     3   103     5]
 [    4     0  1842   367    25]
 [    5     0    15 22472   132]
 [    0     1     5  1518  6049]]
Testing Confusion_Matrix
 [[ 304    0    2   25    2]
 [   2   70    1   23    0]
 [   0    0  451   75    7]
 [   5    0    4 5690   39]
 [   0    0    1  397 1466]]
Classification Report
              precision    recall  f1-score   support

           1       0.98      0.91      0.94       333
           2       1.00      0.73      0.84        96
           3       0.98      0.85      0.91       533
           4       0.92      0.99      0.95      5738
           5       0.97      0.79      0.87      1864

    accuracy                           0.93      8564
   macro avg       0.97      0.85      0.90      8564
weighted avg       0.94      0.93      0.93      8564
```

KNeighbors Classifier:

```
*****************************
 KNeighborsClassifier()
 *****************************
Training Score 0.953993460999533
Accuracy Training Score = 0.953993460999533  Accuracy Test Score = 0.9472209248014947

Training Confusion_Matrix
 [[ 1298     0     8    55    14]
 [    0   413     2    18    13]
 [    4     0  2006   154    74]
 [    3     2    31 22189   399]
 [    0     1    23   775  6774]]
Testing Confusion_Matrix
 [[ 314    0    3   15    1]
 [   0   86    0    9    1]
 [   0    0  483   39   11]
 [   4    0    9 5584  141]
 [   0    0    8  211 1645]]
Classification Report
              precision    recall  f1-score   support

           1       0.99      0.94      0.96       333
           2       1.00      0.90      0.95        96
           3       0.96      0.91      0.93       533
           4       0.95      0.97      0.96      5738
           5       0.91      0.88      0.90      1864

    accuracy                           0.95      8564
   macro avg       0.96      0.92      0.94      8564
weighted avg       0.95      0.95      0.95      8564
```

Model-Decision Tree Classifier:

```
*****************************
 DecisionTreeClassifier()
 *****************************
Training Score 0.9735228865016348
Accuracy Training Score = 0.9735228865016348  Accuracy Test Score = 0.9638019617001401

Training Confusion_Matrix
 [[ 1355     0     0    20     0]
 [    0   441     0     5     0]
 [    0     0  2116   122     0]
 [    0     0     2 22553    69]
 [    0     0     4   685  6884]]
Testing Confusion_Matrix
 [[ 319    1    4    8    1]
 [   0   90    0    6    0]
 [   1    0  498   30    4]
 [   4    0    8 5684   42]
 [   7    1   10  183 1663]]
Classification Report
              precision    recall  f1-score   support

           1       0.96      0.96      0.96       333
           2       0.98      0.94      0.96        96
           3       0.96      0.93      0.95       533
           4       0.96      0.99      0.98      5738
           5       0.97      0.89      0.93      1864

    accuracy                           0.96      8564
   macro avg       0.97      0.94      0.95      8564
weighted avg       0.96      0.96      0.96      8564

 * * * * *
 MODEL CROSSVALIDATION
A cross-fold  2 CV_mean  0.5435077066791218 Training Score  0.9735228865016348 Testing Score  0.9638019617001401
A cross-fold  3 CV_mean  0.6892346231020766 Training Score  0.9735228865016348 Testing Score  0.9638019617001401
A cross-fold  4 CV_mean  0.6713685193834656 Training Score  0.9735228865016348 Testing Score  0.9638019617001401
A cross-fold  5 CV_mean  0.738183092013078 Training Score  0.9735228865016348 Testing Score  0.9638019617001401
Model Score
 0.9788479034162358
```

Model-Random Forest Classifier:

```
*****************************
 RandomForestClassifier()
 *****************************
Training Score 0.9735228865016348
Accuracy Training Score = 0.9735228865016348  Accuracy Test Score = 0.9648528724894909

Training Confusion_Matrix
 [[ 1355     0     0    20     0]
 [    0   441     0     5     0]
 [    0     0  2116   122     0]
 [    0     0     2 22553    69]
 [    0     0     4   685  6884]]
Testing Confusion_Matrix
 [[ 319    0    0   14    0]
 [   0   90    0    6    0]
 [   0    0  495   35    3]
 [   0    0    0 5711   27]
 [   0    0    2  214 1648]]
Classification Report
               precision    recall  f1-score   support

           1       1.00      0.96      0.98       333
           2       1.00      0.94      0.97        96
           3       1.00      0.93      0.96       533
           4       0.96      1.00      0.97      5738
           5       0.98      0.88      0.93      1864

    accuracy                           0.96      8564
   macro avg       0.99      0.94      0.96      8564
weighted avg       0.97      0.96      0.96      8564
```

```
Model Score
 0.9943892923849014
```

Model- ADA Boost Classifier

```
Training Score 0.6771076599719758
Accuracy Training Score = 0.6771076599719758  Accuracy Test Score = 0.6893974778141055

Training Confusion_Matrix
 [[  559     3     4   711    98]
 [    4   226    19   129    68]
 [   18     4     3  2156    57]
 [   59    50    37 22044   434]
 [   13     0     1  7196   363]]
Testing Confusion_Matrix
 [[ 150    0    0  156   27]
 [   2   52    2   25   15]
 [   3    0    2  512   16]
 [  16   13   12 5605   92]
 [   3    0    0 1766   95]]
Classification Report
               precision    recall  f1-score   support

           1       0.86      0.45      0.59       333
           2       0.80      0.54      0.65        96
           3       0.12      0.00      0.01       533
           4       0.70      0.98      0.81      5738
           5       0.39      0.05      0.09      1864

    accuracy                           0.69      8564
   macro avg       0.57      0.40      0.43      8564
weighted avg       0.60      0.69      0.59      8564

 * * * * *
```

```
Model Score
 0.7187834314891549
```

Model- Extra Trees Classifier

```
*****************************
 ExtraTreesClassifier()
 *****************************
Training Score 0.9735228865016348
Accuracy Training Score = 0.9735228865016348  Accuracy Test Score = 0.9646193367585241

Training Confusion_Matrix
 [[ 1355     0     0    20     0]
 [    0   441     0     5     0]
 [    0     0  2116   122     0]
 [    0     0     2 22553    69]
 [    0     0     4   685  6884]]
Testing Confusion_Matrix
 [[ 319    0    0   12    2]
 [   0   90    0    6    0]
 [   0    0  495   35    3]
 [   0    0    0 5707   31]
 [   0    0    2  212 1650]]
Classification Report
              precision    recall  f1-score   support

           1       1.00      0.96      0.98       333
           2       1.00      0.94      0.97        96
           3       1.00      0.93      0.96       533
           4       0.96      0.99      0.97      5738
           5       0.98      0.89      0.93      1864

    accuracy                           0.96      8564
   macro avg       0.99      0.94      0.96      8564
weighted avg       0.97      0.96      0.96      8564
```

Model- Gradient Classifier

```
*****************************
 GradientBoostingClassifier()
 *****************************
Training Score 0.8871730499766465
Accuracy Training Score = 0.8871730499766465  Accuracy Test Score = 0.8880196170014012

Training Confusion_Matrix
 [[ 1147     0     0   221     7]
 [    0   438     0     8     0]
 [    0     0  1440   746    52]
 [    0     0     0 22549    75]
 [    0     0     0  2756  4817]]
Testing Confusion_Matrix
 [[ 285    0    0   45    3]
 [   2   89    1    4    0]
 [   1    0  356  161   15]
 [   0    0    1 5710   27]
 [   1    0    3  695 1165]]
Classification Report
              precision    recall  f1-score   support

           1       0.99      0.86      0.92       333
           2       1.00      0.93      0.96        96
           3       0.99      0.67      0.80       533
           4       0.86      1.00      0.92      5738
           5       0.96      0.62      0.76      1864

    accuracy                           0.89      8564
   macro avg       0.96      0.81      0.87      8564
weighted avg       0.90      0.89      0.88      8564
```

Hyper-Parameter: Random Forest Classifier

In [97]:
```
#random Forest Training and score
rfc=RandomForestClassifier(n_estimators=200, criterion='gini', max_features='sqrt')
rfc.fit(x_train,y_train)
rfc_score=rfc.score(x_train,y_train)

#predict random Forest
pred_train1=rfc.predict(x_train)
y_pred=rfc.predict(x_test)

#result random Forest
print("Model Score ",rfc_score)
print('\033[1m'+'Final Random Forest Classifier Model'+'\033[0m')
print('\033[1m'+'Accuracy Score :'+'\033[0m\n', accuracy_score(y_test, y_pred))
print('\n')
print('\033[1m'+'Confusion matrix of Random Forest Classifier :'+'\033[0m \n',confusion_matrix(y_test, y_pred))
print('\n')
print('\033[1m'+'Classification Report of Random Forest Classifier'+'\033[0m \n',classification_report(y_test, y_pred))
```

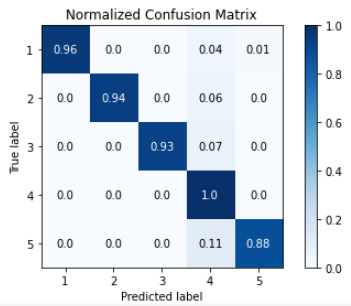
```
Model Score  0.9735228865016348
Final Random Forest Classifier Model
Accuracy Score :
 0.9653199439514246


Confusion matrix of Random Forest Classifier :
 [[ 319    0    0   12    2]
 [   0   90    0    6    0]
 [   0    0  495   36    2]
 [   0    0    0 5714   24]
 [   0    0    2  213 1649]]


Classification Report of Random Forest Classifier
              precision    recall  f1-score   support

           1       1.00      0.96      0.98       333
           2       1.00      0.94      0.97        96
           3       1.00      0.93      0.96       533
           4       0.96      1.00      0.98      5738
           5       0.98      0.88      0.93      1864

    accuracy                           0.97      8564
   macro avg       0.99      0.94      0.96      8564
weighted avg       0.97      0.97      0.96      8564
```

In [99]:
```
#Accuracy_Score
test_accuracy=accuracy_score(y_test,y_pred)

#cross val
for i in range(2,6):
    cv_score=cross_val_score(rfc,x,y,cv=i)
    cv_mean=cv_score.mean()

    #print
    print("A cross-fold ",i,"CV_Score ",cv_mean,"Testing Score ",test_accuracy)
```

```
A cross-fold  2 CV_Score  0.6308267164876227 Testing Score  0.9653199439514246
A cross-fold  3 CV_Score  0.7275116044723454 Testing Score  0.9653199439514246
A cross-fold  4 CV_Score  0.7058617468472677 Testing Score  0.9653199439514246
A cross-fold  5 CV_Score  0.7774871555347967 Testing Score  0.9653199439514246
```

In [104]:
```
import scikitplot as skplt

# Creating a normalized confusion matrix here
skplt.metrics.plot_confusion_matrix(y_test,y_pred, normalize=True)
```

Out[104]: <AxesSubplot:title={'center':'Normalized Confusion Matrix'}, xlabel='Predicted label', ylabel='True label'>



Model SCORE:

```
In [105]: from sklearn.metrics import roc_auc_score
          y_pred_proba=rfc.predict_proba(x_test)
          y_pred_proba

          print("Model Score\n",roc_auc_score(y_test,y_pred_proba,multi_class='ovr'))

          Model Score
           0.9945873297690634
```

SAVING MODEL

Model Saved as "Rating_Predict.pkl"

Saving Model and Loading Model

```
In [107]: # loading pack file
          pickled_model= pickle.load(open(filename,'rb'))
          result=pickled_model.score(x_test,y_test)
          result*100

Out[107]: 96.53199439514246

In [111]: conclude=pd.DataFrame([pickled_model.predict(x_test)[:],pred_test[:]],index=['Predicted','Original'])
          conclude
```

Out[111]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
Predicted	4	4	4	4	4	4	5	4	4	4	5	4	3	4	4	4	4	5	4	4	1	4	5	4	4	4	1	1	5	3	5	4	4	4	1	4	4	3	4
Original	4	4	4	4	4	4	5	4	4	5	5	4	3	4	4	4	4	5	4	4	1	4	5	4	4	4	1	1	5	3	5	4	4	4	1	4	4	3	4

```
In [ ]:

In [ ]:
```

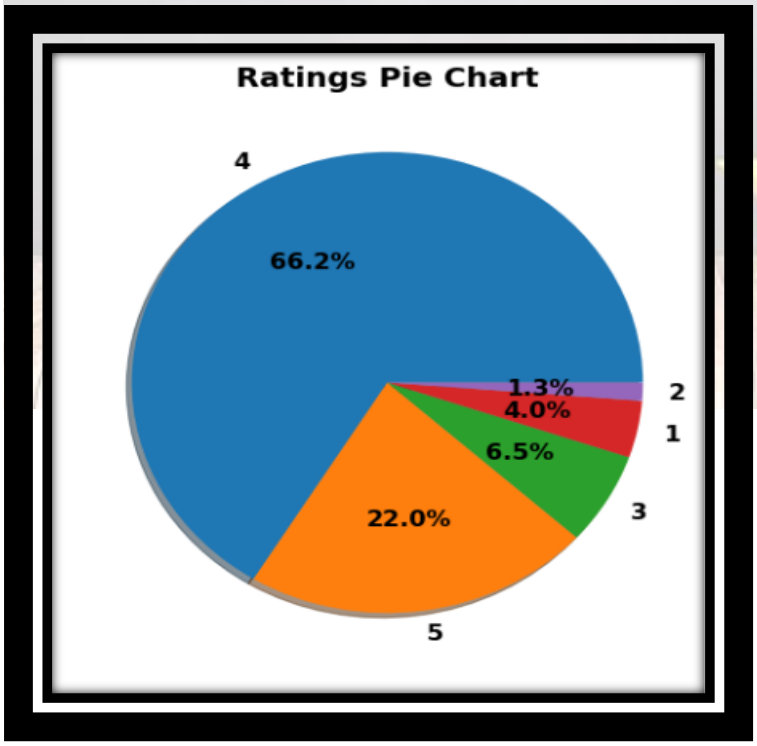
KEY METRICS FOR SUCCESS IN SOLVING PROBLEM UNDER CONSIDERATION.

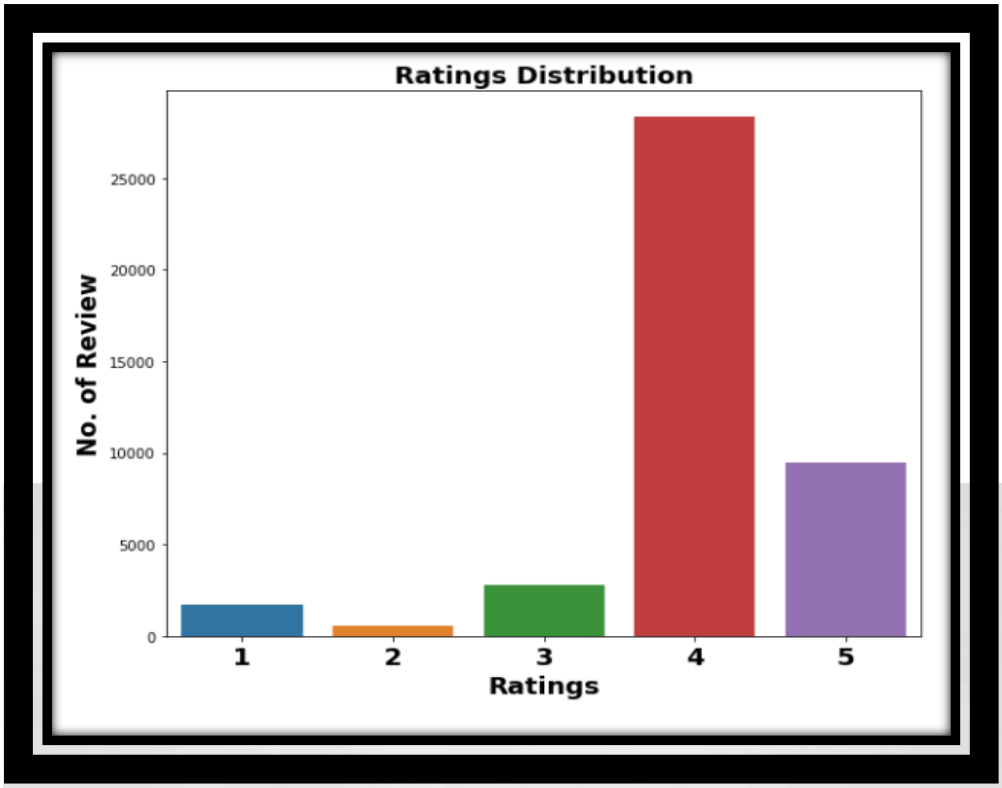
The Following Metrics were Used:

- Accuracy Score
- Classification Report
- Confusion Matrix
- F1 Score

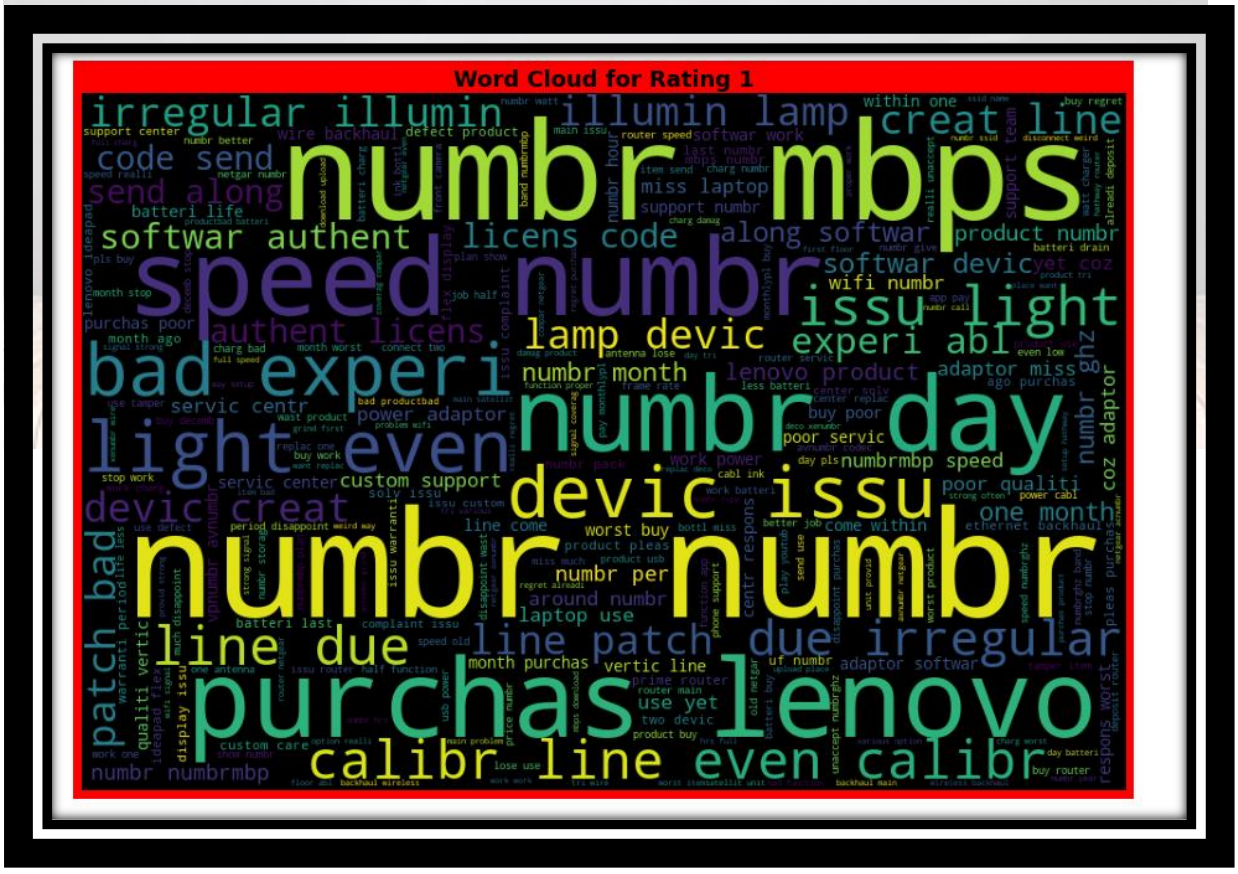
```
#result random Forest
print("Model Score ",rfc_score)
print("Accuracy Training Score ",accuracy_score(y_train,pred_train1)," Accuracy Test Score ",accuracy_score(y_test,pred_test1),"
print("Training Confusion_Matrix \n",confusion_matrix(y_train,pred_train1)," \nTesting Confusion_Matrix",confusion_matrix(y_test,
print("Classification Report \n",classification_report(y_test,pred_test1))
```

Visualizations

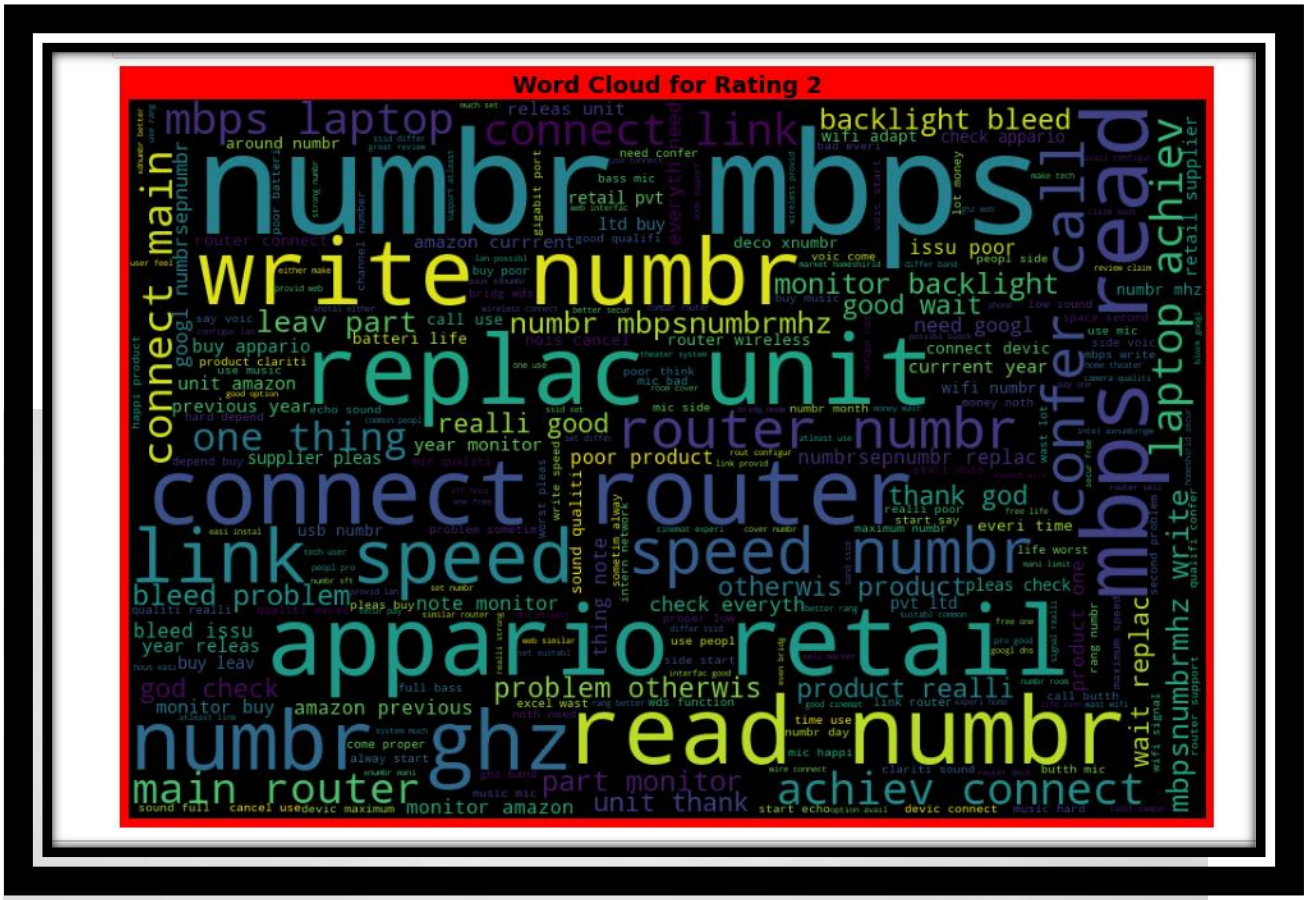




Word clouds 1

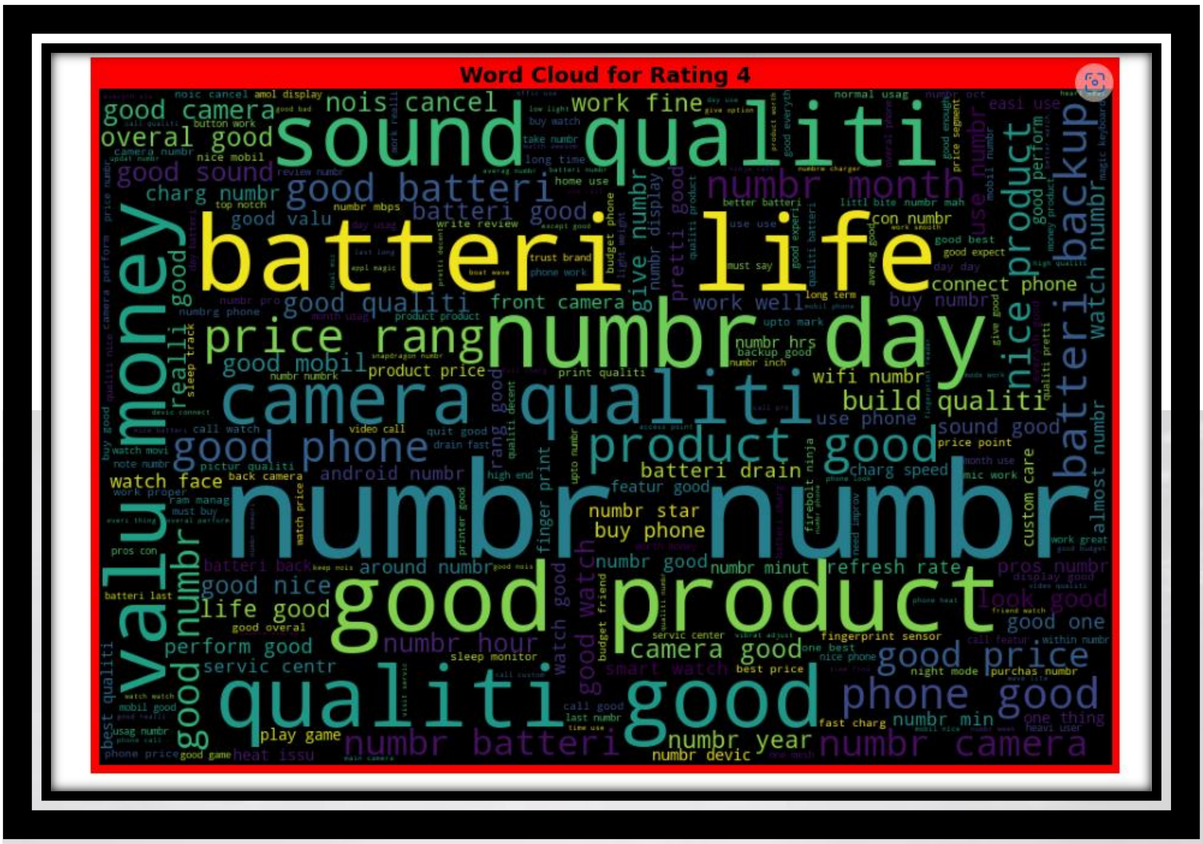


Word clouds 2

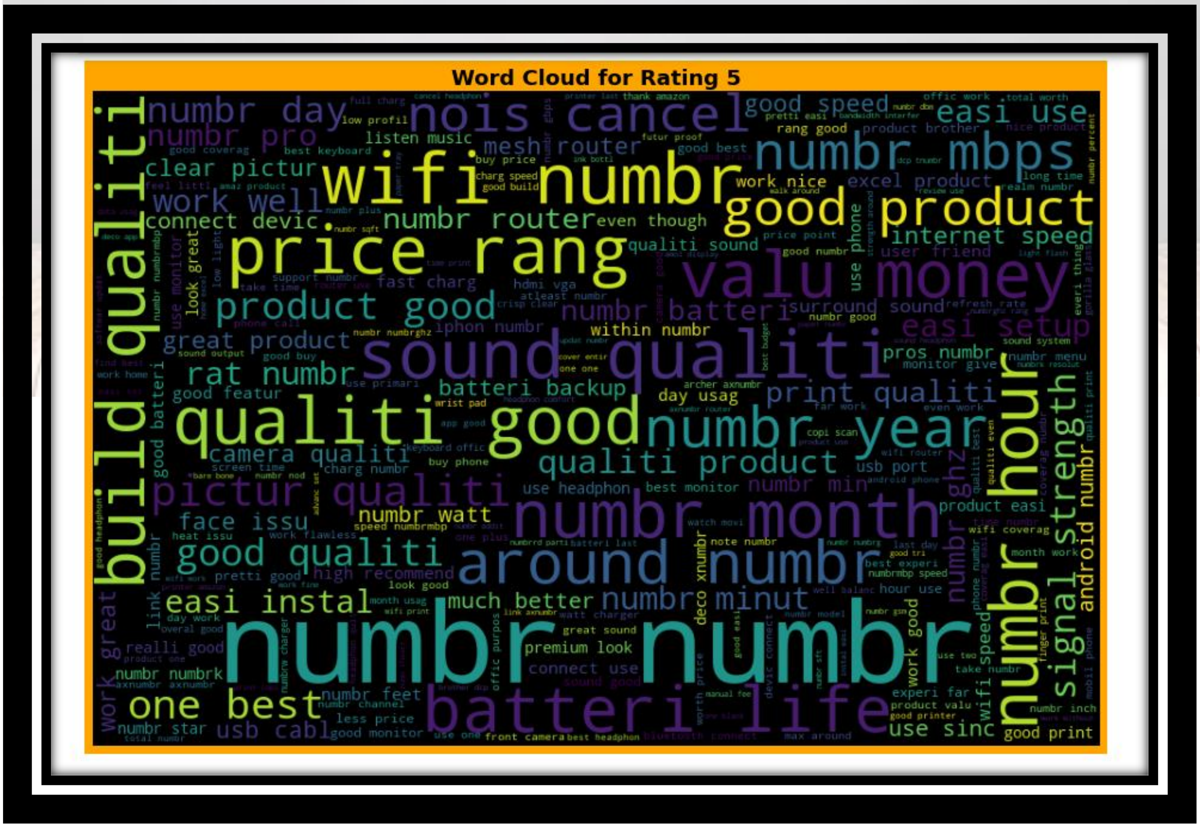


Word clouds 3

Word clouds 4



Word clouds 5



Interpretation of the Results

- We were asked to scrape 20000 data from E-commerce portal whereas we scraped more than 40,000 data.
- This dataset was very special as it had a separate data type dataset having comments as target columns.
- Firstly, the datasets were having few null values. Those were dropped
- The data set had no null duplicated rows.
- I found maximum label data columns.
- Maximum Word Clouds and Count plot was used followed to determine the condition and relation with one and another.
- I notice a huge number of symbols were present into comment texts which was cleaned and letter processed with the help of NLP (nltk) library like (stop words, lemmatization was used on the data set, finally TF-IDF was used to convert them to the vectors.)
- Scaling technique (By lemmatization) was done on both the train dataset and test data set, this has a good impact like it will help the model not to get biased.
- We have to use multiple models while building model using train dataset as to get the best model out of it.
- Extra Trees and Random Forest were the best among all the models. Result received with both model is approx. above 95 percent plus.
- However Random Forest shows good accuracy score and CV_Score close to testing score, hence this Shows our model is performing extremely well.
- Finally selected Extra Tree Regressor and saved the model and finally printed the score and finally compared the data with the test data set.
- Later the Test Data Set is vectorised and Test was applied to check the outcome.

CONCLUSION

➢ Key Findings and Conclusions of the Study

By, the help of This data from Data set I have made a RATING PREDICTION Model. We have done EDA, cleaned data and Visualized Data on the Data Set. While cleaning the data Set it is analysed missing values from data set which is irrelevant so I dropped that column. After that we have done prediction on basis of Data using Data Pre-processing, Checked Correlation, removed Punctation, extra space, leaning and trailing white space, ¥n, stop words, converted text into vectors using TF-IDF and at last train our data by splitting our data through train-test split process.

We worked with various classifiers while using these given models and finally selected best model which was giving best accuracy, F1 score and CV score. Random Forest Classifier model was selected and Hyper Tunning was used with Grid Search CV, And Best was Saved on the basis of Accuracy score, F1 score and CV score and at last I used that model to compare with predicted and Actual test data.

Thus, our project Stands completed.

And saved the model as filename='Rating_Predict.pkl'

➢ Learning Outcomes of the Study in respect of Data Science

I found that the dataset was quite interesting to handle as it contains all types of data in it. Improvement in computing technology has made it possible to examine reviews information that cannot previously be captured, processed and analysed. New analytical techniques of machine learning can be used in Rating Prediction.

The power of visualization has helped us in understanding the data by graphical representation it has made me to understand what data is trying to say. Data cleaning is one of the most important steps to remove missing value and to replace null value and zero values with their respective mean, median or mode.

This study is an exploratory Data Analysis has attempted to use machine learning algorithms in finding the probability of Ratings from the given Reviews in each model. To

conclude the application of Machine Learning in prediction is still at an early stage. I hope this study has moved a small step ahead in providing solution to the companies.

The Challenges I faced was when I was not much aware of NLP and I used various sources like YOU-Tube, Kaggle and Medium and NLTK library where I faced problem. I faced issues even at the time of binning process. Even The Algorithm was working on huge time to complete the test and CV_Score generating was again time taking in each of the cases.

Finally, I had to run the test twice and thrice with different standardization process. I was actually thinking to get 100 percent out of those.

However, I finally achieved a good model out of this.

➢ Limitations of this work and Scope for Future Work

This model doesn't predict future probability. The future will be unpredictable at all times due to this, the risk in RATING PREDICTION remains an import factor. This Model can predict for a time period and needs to be updated as per need or on yearly basis to stop the variance in ratings. Machine can classify the Reviews of the consumer but the intensity of the user remains unpredictable.

The best way to be future ready to get the model updated once or twice as per the market standard.

Thank-you