FLIP ROBO

RATING PREDICTION PROJECT

Submitted by:

Amrutha Dhondale

ACKNOWLEDGMENT

I would like to thank Flip Robo Technologies for providing me with the opportunity to work on this project from which I have learned a lot. I would also like to thank my mentor in Fliprobo, Swati Mahaseth, for providing me with the dataset and problem statement for performing this wonderful task.

Some of the resources that helped me to complete this project are as follows:

- Internet/web
- Stack overflow
- Analytics Vidhya
- Wikipedia

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 rating. So, we must build an application which can predict the rating by seeing the review.

- ## Conceptual Background of the Domain Problem

  Nowadays, a massive number of reviews is available online. Besides offering a valuable source of information, these informational contents generated by users, also called User Generated Contents (UGC) strongly impact the purchase decision of customers.
  As a matter of fact, a recent survey (Hinckley, 2015) revealed that 67.7% of consumers are effectively influenced by online reviews when making their purchase decisions. More precisely, 54.7% recognized that these reviews were either fairly, very, or absolutely important in their purchase decision making. Relying on online reviews has thus become a second nature for consumers.
  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.

- **Review of Literature**

The rapid development of Web 2.0 and e-commerce has led to a proliferation in the number of online user reviews. Online reviews contain a wealth of sentiment information that is important for many decisionmaking processes, such as personal consumption decisions, commodity quality monitoring, and social opinion mining. Mining the sentiment and opinions that are contained in online reviews has become an important topic in natural language processing, machine learning, and Web mining.

We have used different machine learning models to predict the above. Since we have categorical target data so classification model algorithms have been used. We will start our project with the train dataset which contains categories of offensive words.

We will observe all the features with following goals in mind:

- Relevance of the features
- Distribution of the features
- Data Cleaning of the features
- Visualization of the features

After having gone through all the features and cleaning the dataset, we will move on to machine learning classification modelling.

- Pre-processing of the dataset for models
- Testing multiple algorithms with multiple evaluation metrics

• Select evaluation metric as per our specific business application
• Doing hyper-parameter tuning using GridSearchCV for the best model
• Finally saving the best model

- **Motivation for the Problem Undertaken**

  It’s a multi-label classification problem to solve which is a completely different project that I tried so far. So, it will be exciting for me to get my hands on NLP Concepts which is required in this kind of projects.

  Every day we come across various products through digital medium we swipe across hundreds of product choices under one category. It will be tedious for the customer to make selection. Here comes 'reviews' where customers who have already got that product leave a rating after using them and brief their experience by giving reviews.

  Many product reviews are not accompanied by a scale rating system, consisting only of a textual evaluation. In this case, it becomes daunting and time-consuming to compare different products to eventually make a choice between them.

  Therefore, models able to predict the user rating from the text review are critically important. Getting an overall sense of a textual review could in turn improve consumer experience.

Analytical Problem Framing

- **Mathematical/ Analytical Modelling of the Problem**

  The dataset contains csv file: There are only two features that were scraped from e-commerce website that are ratings and reviews which were for different items like laptops, Phones, Headphones, smart watches, Professional Cameras, Printers, monitors, home theatre, router, and printer.

  In our scrapped dataset, our target variable "Rating " is a categorical variable i.e., it can be classified as ‘1.0’, ‘2.0’,’3.0’,’4.0’,’5.0’. Therefore, we will be handling this modelling problem as classification.

  Fetched equal number of reviews for each rating. From an initial statistical overview of the dataset, we infer that our target variable is of multi label binary classification.

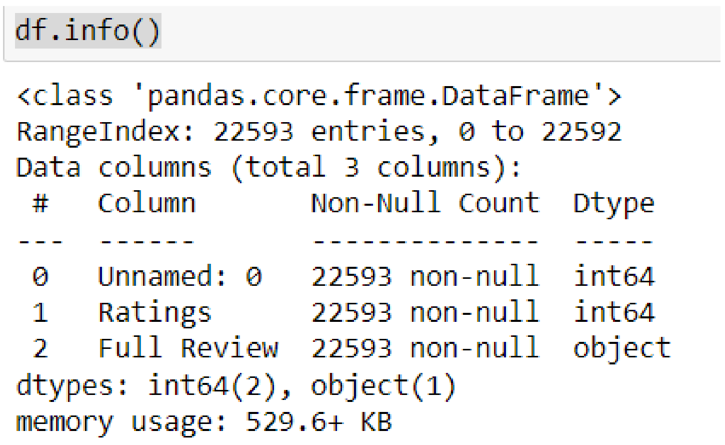
- **Data Sources and their formats**

  The data was in CSV format (Comma Separated Values). After reading the data by using function df=pd.read_csv (‘ --- file path --- ’) in jupyter notebook there were 22593 rows and 3 columns.

  Dataset/Attributes Description:

  1)Ratings: It is the Label column, which includes ratings in the form of integers from 1 to 5.

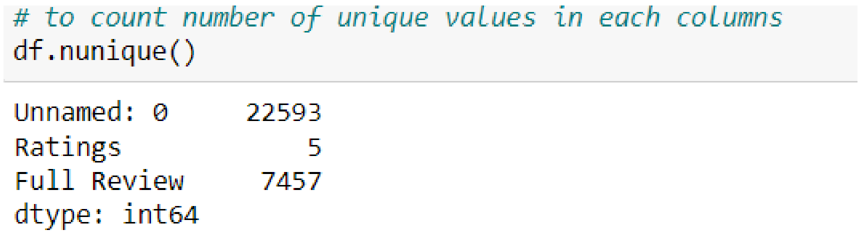
  2)Full review: It contains text data on the basis of which we have to build a model to predict ratings.



```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22593 entries, 0 to 22592
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Unnamed: 0   22593 non-null  int64 
 1   Ratings      22593 non-null  int64 
 2   Full Review  22593 non-null  object
dtypes: int64(2), object(1)
memory usage: 529.6+ KB
```

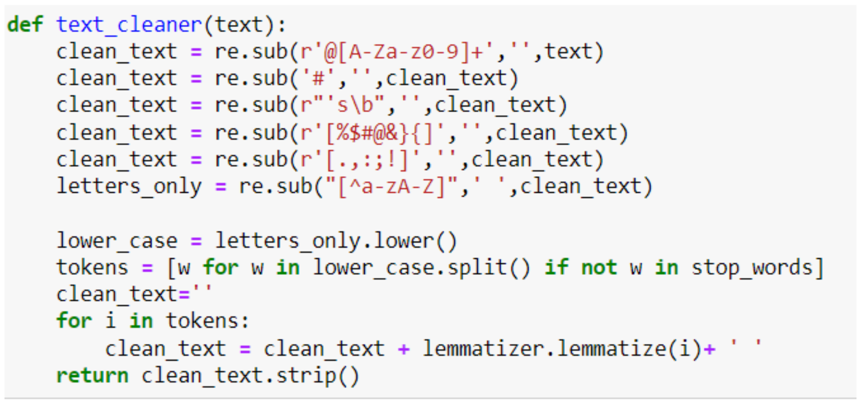
Number of unique elements present in each column



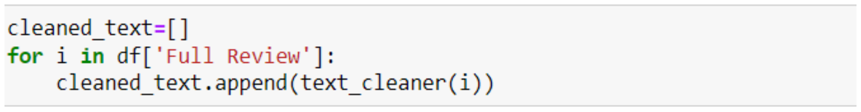
```
# to count number of unique values in each columns
df.nunique()
```

```
Unnamed: 0     22593
Ratings            5
Full Review     7457
dtype: int64
```

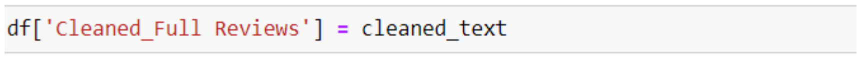
- **Data Preprocessing Done**
  Imported RegEx (regular expression) to identify sequence of characters that specifies a search pattern, installed nltk library for words classification, tokenizing, stemming, parse tree visualization etc and wordcloud for visualizing loudwords used in the comments.



```
def text_cleaner(text):
    clean_text = re.sub(r'@[A-Za-z0-9]+','',text)
    clean_text = re.sub('#','',clean_text)
    clean_text = re.sub(r"'s\b",'',clean_text)
    clean_text = re.sub(r'[%$#@&}{]','',clean_text)
    clean_text = re.sub(r'[.,:;!]','',clean_text)
    letters_only = re.sub("[^a-zA-Z]",' ',clean_text)

    lower_case = letters_only.lower()
    tokens = [w for w in lower_case.split() if not w in stop_words]
    clean_text=''
    for i in tokens:
        clean_text = clean_text + lemmatizer.lemmatize(i)+ ' '
    return clean_text.strip()
```



```
cleaned_text=[]
for i in df['Full Review']:
    cleaned_text.append(text_cleaner(i))
```



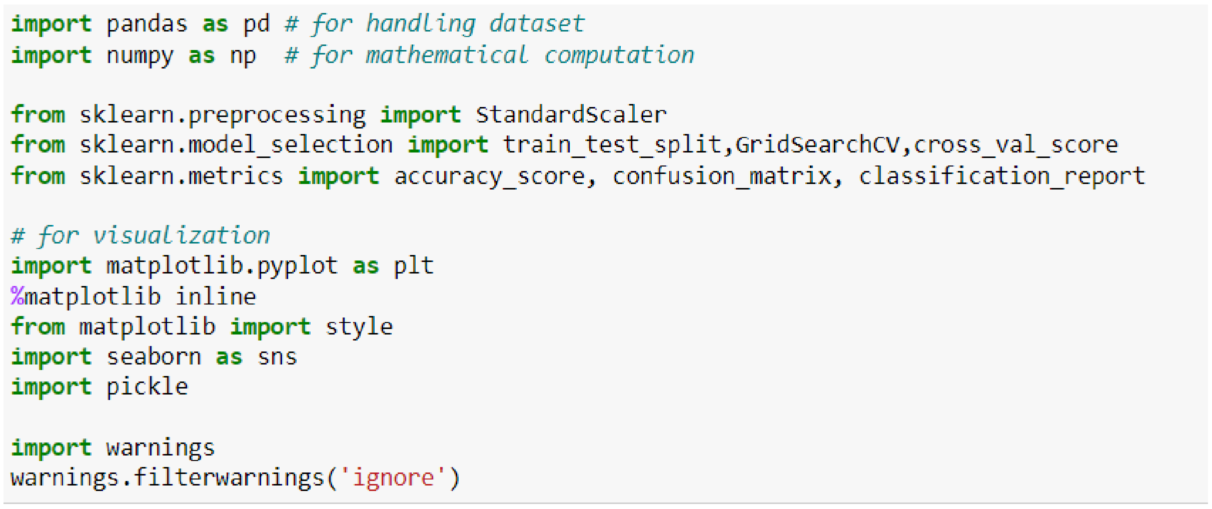
```
df['Cleaned_Full Reviews'] = cleaned_text
```

After doing all data cleaning on Full Reviews column, then storing it to a empty list named cleaned text and then storing to a dataframe cleaned full reviews.

- **Hardware and Software Requirements and Tools Used**

For doing this project, the hardware used is a laptop with a stable internet connection. While coming to software part, I have used Jupyter notebook to do my python programming and analysis.
We also need Google chrome webdriver to scrap data.
Libraries Used:

- scikit-learn
- matplotlib
- pandas
- numpy
- Selenium



```
import pandas as pd # for handling dataset
import numpy as np  # for mathematical computation

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split,GridSearchCV,cross_val_score
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# for visualization
import matplotlib.pyplot as plt
%matplotlib inline
from matplotlib import style
import seaborn as sns
import pickle

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

Libraries and Packages used:

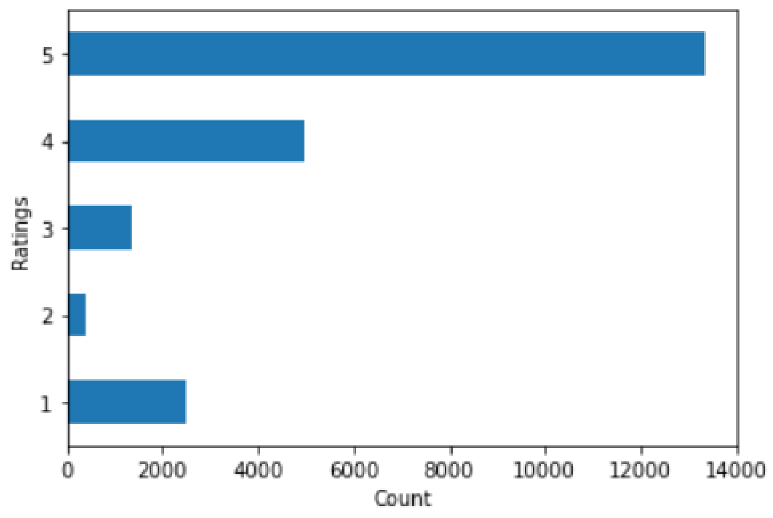
- Pickle for saving the machine learning model.
- GridSearchCV for Hyper-parameter tuning.
- Cross validation score to cross check if the model is overfitting or not.

• Seaborn and matplotlib for visualization. • Wordcloud to visualize sense of loud words used • Lemmatizing for converting words to its base form

Model/s Development and Evaluation

- ### Identification of possible problem-solving approaches (methods)

```
df.groupby('Ratings')['Ratings'].count().plot(kind='barh'
plt.xlabel('Count')
plt.show()
```



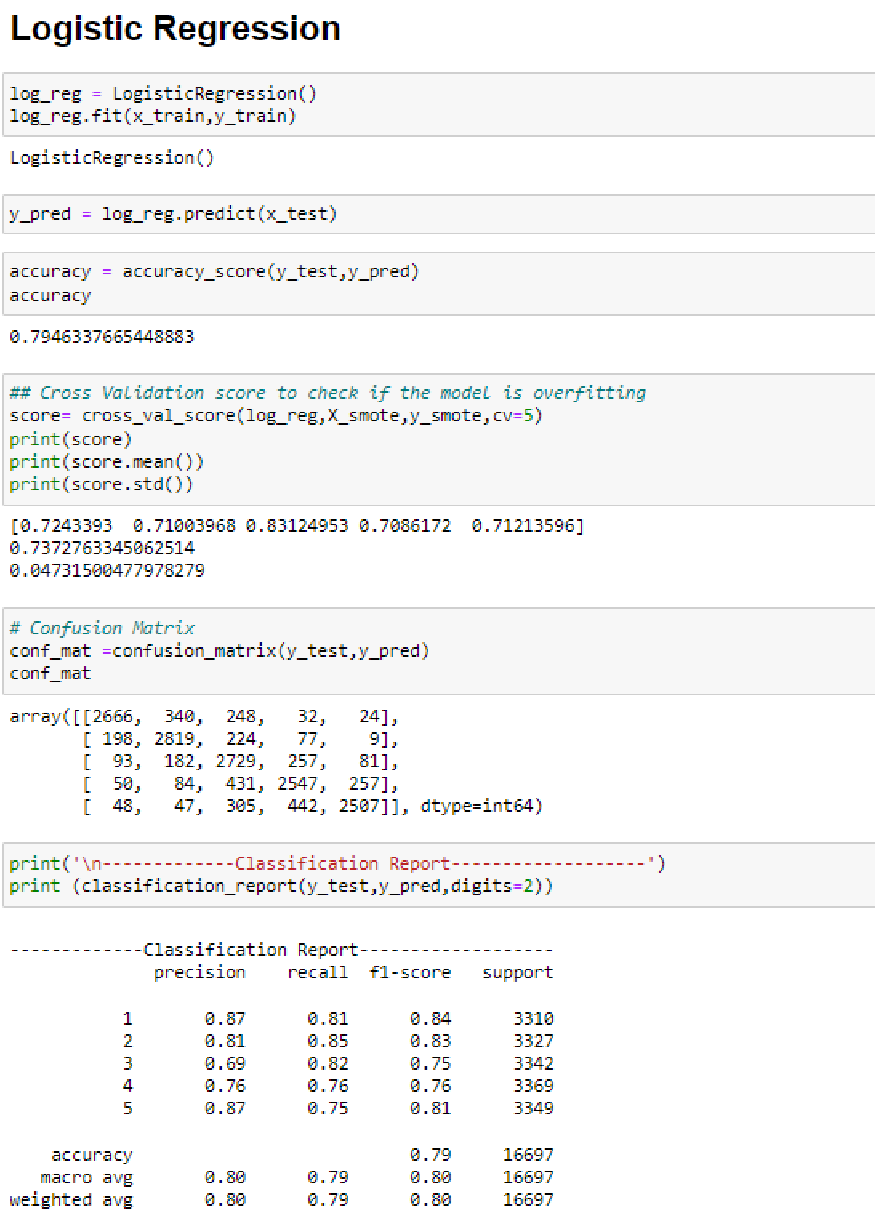
From the above we can observe that the dataset was highly imbalanced so we have used SMOTE to balance the dataset.

- ### Testing of Identified Approaches (Algorithms)

Algorithms used:

• Logistic Regression
• Decision Tree Classifier
• Random Forest Classifier
• KNN Classifier
• Support Vector Machines
• Gradient Boosting Classifier

- **Run and Evaluate selected models**



Logistic Regression

```
log_reg = LogisticRegression()
log_reg.fit(x_train,y_train)
```

```
LogisticRegression()
```

```
y_pred = log_reg.predict(x_test)
```

```
accuracy = accuracy_score(y_test,y_pred)
accuracy
```

```
0.7946337665448883
```

```
## Cross Validation score to check if the model is overfitting
score= cross_val_score(log_reg,X_smote,y_smote,cv=5)
print(score)
print(score.mean())
print(score.std())
```

```
[0.7243393  0.71003968 0.83124953 0.7086172  0.71213596]
0.7372763345062514
0.04731500477978279
```

```
# Confusion Matrix
conf_mat =confusion_matrix(y_test,y_pred)
conf_mat
```

```
array([[2666,  340,  248,   32,   24],
       [ 198, 2819,  224,   77,    9],
       [  93,  182, 2729,  257,   81],
       [  50,   84,  431, 2547,  257],
       [  48,   47,  305,  442, 2507]], dtype=int64)
```

```
print('\n-------------Classification Report------------------')
print (classification_report(y_test,y_pred,digits=2))
```

```

-------------Classification Report------------------
              precision    recall  f1-score   support

           1       0.87      0.81      0.84      3310
           2       0.81      0.85      0.83      3327
           3       0.69      0.82      0.75      3342
           4       0.76      0.76      0.76      3369
           5       0.87      0.75      0.81      3349

    accuracy                           0.79     16697
   macro avg       0.80      0.79      0.80     16697
weighted avg       0.80      0.79      0.80     16697
```

DecisionTree Classifier

```
dt_clf = DecisionTreeClassifier()
dt_clf.fit(x_train,y_train)
pred=dt_clf.predict(x_train)
dt_clf_report = pd.DataFrame(classification_report(y_train,pred, output_dict=True))

print("\n==================Train Result=====================")

print(f"Accuracy Score: {accuracy_score(y_train,pred) * 100:.2f}%")
print("_______________________________________________")
print(f"CLASSIFICATION REPORT:\n{dt_clf_report}")
print("_______________________________________________")
print(f"Confusion Matrix:\n {confusion_matrix(y_train,pred)}\n")

#*********************** Test Score ************************

pred = dt_clf.predict(x_test)
dt_clf_report = pd.DataFrame(classification_report(y_test,pred, output_dict=True))
print("\n==================Test Result=====================")
print(f"Accuracy Score: {accuracy_score(y_test,pred) * 100:.2f}%")
print("_______________________________________________")
print(f"CLASSIFICATION REPORT:\n{dt_clf_report}")
print("_______________________________________________")
print(f"Confusion Matrix:\n {confusion_matrix(y_test,pred)}\n")
```

```
==================Train Result=====================
Accuracy Score: 91.85%
_______________________________________________
CLASSIFICATION REPORT:
                      1             2             3            4  \
precision      0.983047      0.972422      0.816192     0.921515
recall         0.923460      0.956231      0.942187     0.876952
f1-score       0.952322      0.964259      0.874676     0.898682
support    10047.000000  10030.000000  10015.000000  9988.000000

                      5  accuracy     macro avg  weighted avg
precision      0.919761  0.918483      0.922588      0.922642
recall         0.893385  0.918483      0.918443      0.918483
f1-score       0.906381  0.918483      0.919264      0.919311
support    10008.000000  0.918483  50088.000000  50088.000000
```

```
Confusion Matrix:
 [[9278  149  589   16   15]
 [ 133 9591  299    2    5]
 [  17   67 9436  222  273]
 [   4   37  701 8759  487]
 [   6   19  536  506 8941]]


===================Test Result=====================
Accuracy Score: 84.31%
_______________________________________________
CLASSIFICATION REPORT:
                     1            2            3            4            5  \
precision     0.929768     0.909475     0.748452     0.816646     0.831307
recall        0.871903     0.905921     0.868043     0.774711     0.796059
f1-score      0.899906     0.907695     0.803824     0.795126     0.813301
support    3310.000000  3327.000000  3342.000000  3369.000000  3349.000000

           accuracy     macro avg  weighted avg
precision  0.843086      0.847129      0.846859
recall     0.843086      0.843327      0.843086
f1-score   0.843086      0.843970      0.843714
support    0.843086  16697.000000  16697.000000
_______________________________________________
Confusion Matrix:
 [[2886  130  224   34   36]
 [ 103 3014  175   18   17]
 [  44   77 2901  177  143]
 [  31   54  329 2610  345]
 [  40   39  247  357 2666]]
```

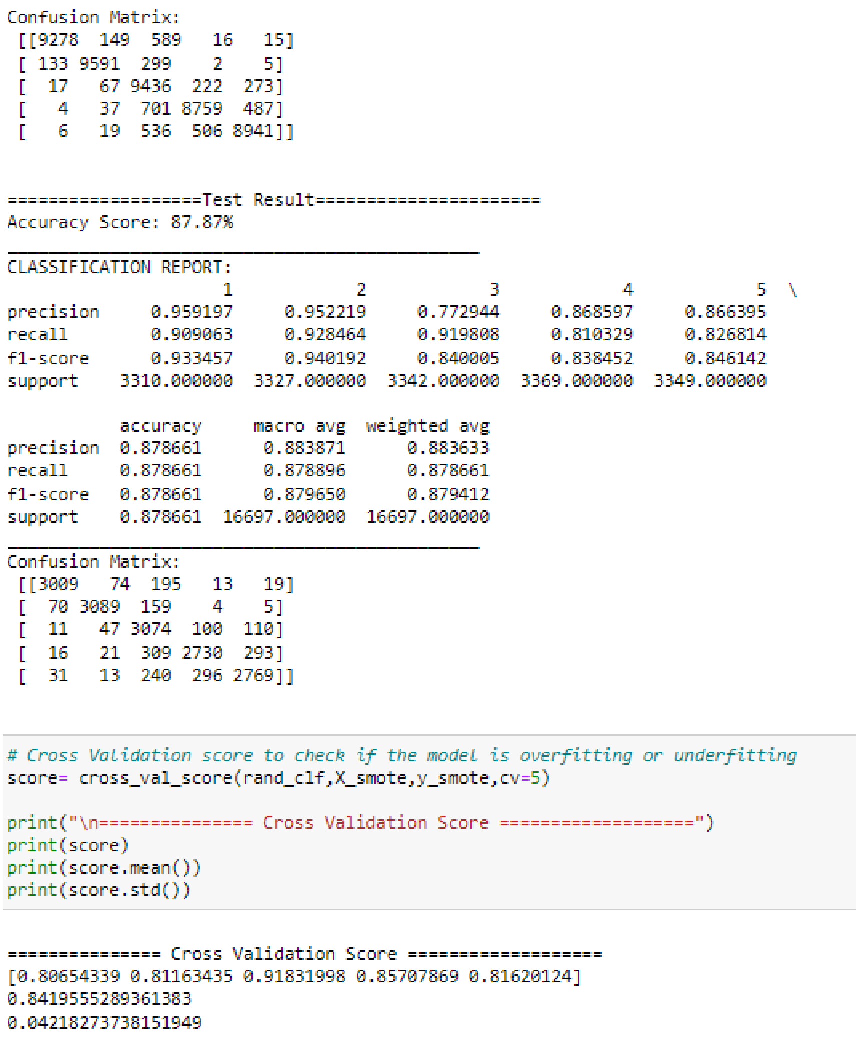
```
# Cross Validation score to check if the model is overfitting or underfitting
score= cross_val_score(dt_clf,X_smote,y_smote,cv=5)

print("\n=============== Cross Validation Score ===================")
print(score)
print(score.mean())
print(score.std())
```

```
=============== Cross Validation Score ===================
[0.7473235  0.76866063 0.88530359 0.78760201 0.77397619]
0.7925731826008835
0.04814318722929633
```

Random Forest Classifier

```
rand_clf = RandomForestClassifier(random_state=51)
rand_clf.fit(x_train,y_train)
pred=dt_clf.predict(x_train)
rand_clf_report = pd.DataFrame(classification_report(y_train,pred, output_dict=True))

print("\n===================Train Result=====================")

print(f"Accuracy Score: {accuracy_score(y_train,pred) * 100:.2f}%")
print("_______________________________________________")
print(f"CLASSIFICATION REPORT:\n{rand_clf_report}")
print("_______________________________________________")
print(f"Confusion Matrix:\n {confusion_matrix(y_train,pred)}\n")

#*********************** Test Score ************************

pred = rand_clf.predict(x_test)
rand_clf_report = pd.DataFrame(classification_report(y_test,pred, output_dict=True))
print("\n===================Test Result=====================")
print(f"Accuracy Score: {accuracy_score(y_test,pred) * 100:.2f}%")
print("_______________________________________________")
print(f"CLASSIFICATION REPORT:\n{rand_clf_report}")
print("_______________________________________________")
print(f"Confusion Matrix:\n {confusion_matrix(y_test,pred)}\n")
```

```
===================Train Result=====================
Accuracy Score: 91.85%
_______________________________________________
CLASSIFICATION REPORT:
                      1             2             3            4  \
precision      0.983047      0.972422      0.816192     0.921515
recall         0.923460      0.956231      0.942187     0.876952
f1-score       0.952322      0.964259      0.874676     0.898682
support    10047.000000  10030.000000  10015.000000  9988.000000

                      5  accuracy     macro avg  weighted avg
precision      0.919761  0.918483      0.922588      0.922642
recall         0.893385  0.918483      0.918443      0.918483
f1-score       0.906381  0.918483      0.919264      0.919311
support    10008.000000  0.918483  50088.000000  50088.000000
```

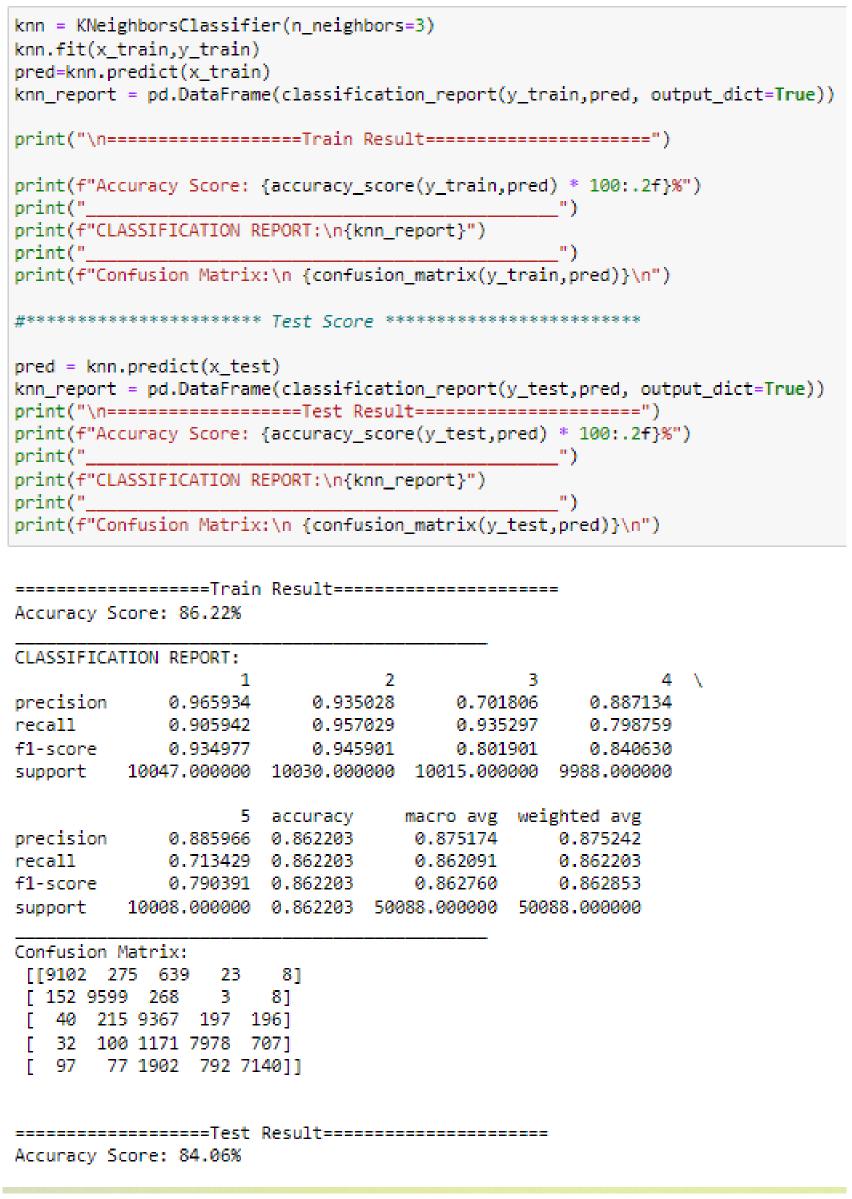


```
Confusion Matrix:
 [[9278  149  589   16   15]
 [ 133 9591  299    2    5]
 [  17   67 9436  222  273]
 [   4   37  701 8759  487]
 [   6   19  536  506 8941]]


==================Test Result=====================
Accuracy Score: 87.87%
_______________________________________________
CLASSIFICATION REPORT:
                     1            2            3            4            5  \
precision     0.959197     0.952219     0.772944     0.868597     0.866395
recall        0.909063     0.928464     0.919808     0.810329     0.826814
f1-score      0.933457     0.940192     0.840005     0.838452     0.846142
support    3310.000000  3327.000000  3342.000000  3369.000000  3349.000000

           accuracy     macro avg  weighted avg
precision  0.878661      0.883871      0.883633
recall     0.878661      0.878896      0.878661
f1-score   0.878661      0.879650      0.879412
support    0.878661  16697.000000  16697.000000
_______________________________________________
Confusion Matrix:
 [[3009   74  195   13   19]
 [  70 3089  159    4    5]
 [  11   47 3074  100  110]
 [  16   21  309 2730  293]
 [  31   13  240  296 2769]]
```

```
# Cross Validation score to check if the model is overfitting or underfitting
score= cross_val_score(rand_clf,X_smote,y_smote,cv=5)

print("\n=============== Cross Validation Score ===================")
print(score)
print(score.mean())
print(score.std())
```

```
=============== Cross Validation Score ===================
[0.80654339 0.81163435 0.91831998 0.85707869 0.81620124]
0.8419555289361383
0.04218273738151949
```

KNN Classifier



```python
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(x_train,y_train)
pred=knn.predict(x_train)
knn_report = pd.DataFrame(classification_report(y_train,pred, output_dict=True))

print("\n==================Train Result=====================")

print(f"Accuracy Score: {accuracy_score(y_train,pred) * 100:.2f}%")
print("______________________________________________")
print(f"CLASSIFICATION REPORT:\n{knn_report}")
print("______________________________________________")
print(f"Confusion Matrix:\n {confusion_matrix(y_train,pred)}\n")

#*********************** Test Score ************************

pred = knn.predict(x_test)
knn_report = pd.DataFrame(classification_report(y_test,pred, output_dict=True))
print("\n==================Test Result=====================")
print(f"Accuracy Score: {accuracy_score(y_test,pred) * 100:.2f}%")
print("______________________________________________")
print(f"CLASSIFICATION REPORT:\n{knn_report}")
print("______________________________________________")
print(f"Confusion Matrix:\n {confusion_matrix(y_test,pred)}\n")
```

```
==================Train Result=====================
Accuracy Score: 86.22%
______________________________________________
CLASSIFICATION REPORT:
                      1             2             3            4  \
precision      0.965934      0.935028      0.701806     0.887134
recall         0.905942      0.957029      0.935297     0.798759
f1-score       0.934977      0.945901      0.801901     0.840630
support    10047.000000  10030.000000  10015.000000  9988.000000

                      5  accuracy     macro avg  weighted avg
precision      0.885966  0.862203      0.875174      0.875242
recall         0.713429  0.862203      0.862091      0.862203
f1-score       0.790391  0.862203      0.862760      0.862853
support    10008.000000  0.862203  50088.000000  50088.000000
______________________________________________
Confusion Matrix:
 [[9102  275  639   23    8]
 [ 152 9599  268    3    8]
 [  40  215 9367  197  196]
 [  32  100 1171 7978  707]
 [  97   77 1902  792 7140]]


==================Test Result=====================
Accuracy Score: 84.06%
```

```
_____________________________________________
CLASSIFICATION REPORT:
                     1            2            3            4            5  \
precision     0.953085     0.918134     0.681394     0.852209     0.866266
recall        0.896073     0.940487     0.918312     0.761650     0.688564
f1-score      0.923700     0.929176     0.782309     0.804389     0.767260
support    3310.000000  3327.000000  3342.000000  3369.000000  3349.000000

           accuracy     macro avg  weighted avg
precision   0.84063      0.854217      0.853972
recall      0.84063      0.841017      0.840630
f1-score    0.84063      0.841367      0.841039
support     0.84063  16697.000000  16697.000000
_____________________________________________
Confusion Matrix:
 [[2966  107  222    6    9]
 [  65 3129  128    2    3]
 [  20   94 3069   78   81]
 [  21   46  473 2566  263]
 [  40   32  612  359 2306]]
```

```python
# Cross Validation score to check if the model is overfitting or underfitting
score= cross_val_score(knn,X_smote,y_smote,cv=5)

print("\n=============== Cross Validation Score ===================")
print(score)
print(score.mean())
print(score.std())
```

```
=============== Cross Validation Score ===================
[0.69416785 0.74927005 0.84262933 0.72291682 0.72366549]
0.7465299094107959
0.0511170264557282
```

Support-Vector Machines

```
svc = SVC(kernel = 'rbf',C=1)
svc.fit(x_train,y_train)
pred=svc.predict(x_train)
svc_report = pd.DataFrame(classification_report(y_train,pred, output_dict=True))

print("\n==================Train Result=====================")

print(f"Accuracy Score: {accuracy_score(y_train,pred) * 100:.2f}%")
print("_______________________________________________")
print(f"CLASSIFICATION REPORT:\n{svc_report}")
print("_______________________________________________")
print(f"Confusion Matrix:\n {confusion_matrix(y_train,pred)}\n")

#*********************** Test Score ************************

pred = svc.predict(x_test)
svc_report = pd.DataFrame(classification_report(y_test,pred, output_dict=True))
print("\n==================Test Result=====================")
print(f"Accuracy Score: {accuracy_score(y_test,pred) * 100:.2f}%")
print("_______________________________________________")
print(f"CLASSIFICATION REPORT:\n{svc_report}")
print("_______________________________________________")
print(f"Confusion Matrix:\n {confusion_matrix(y_test,pred)}\n")
```

```
==================Train Result=====================
Accuracy Score: 88.23%
_______________________________________________
CLASSIFICATION REPORT:
                      1             2             3            4  \
precision      0.961132      0.913924      0.773349     0.889300
recall         0.886036      0.927318      0.902846     0.817982
f1-score       0.922057      0.920572      0.833095     0.852151
support    10047.000000  10030.000000  10015.000000  9988.000000

                      5  accuracy     macro avg  weighted avg
precision      0.898669  0.882347      0.887275      0.887327
recall         0.877298  0.882347      0.882296      0.882347
f1-score       0.887855  0.882347      0.883146      0.883198
support    10008.000000  0.882347  50088.000000  50088.000000
_______________________________________________
Confusion Matrix:
 [[8902  429  636   61   19]
 [ 266 9301  399   48   16]
 [  60  283 9042  339  291]
 [  16   99 1039 8170  664]
 [  18   65  576  569 8780]]


==================Test Result=====================
Accuracy Score: 85.97%
_______________________________________________
CLASSIFICATION REPORT:
                     1            2            3            4            5  \
precision     0.946975     0.908873     0.763585     0.859947     0.841966
recall        0.879456     0.911332     0.887193     0.767290     0.854285
f1-score      0.911967     0.910101     0.820761     0.810980     0.848081
support    3310.000000  3327.000000  3342.000000  3369.000000  3349.000000

           accuracy     macro avg  weighted avg
precision  0.859675      0.864269      0.864054
recall     0.859675      0.859911      0.859675
f1-score   0.859675      0.860378      0.860149
support    0.859675  16697.000000  16697.000000
_______________________________________________
Confusion Matrix:
 [[2911  155  197   23   24]
 [  99 3032  171   21    4]
 [  30   94 2965  117  136]
 [  19   36  356 2585  373]
 [  15   19  194  260 2861]]
```

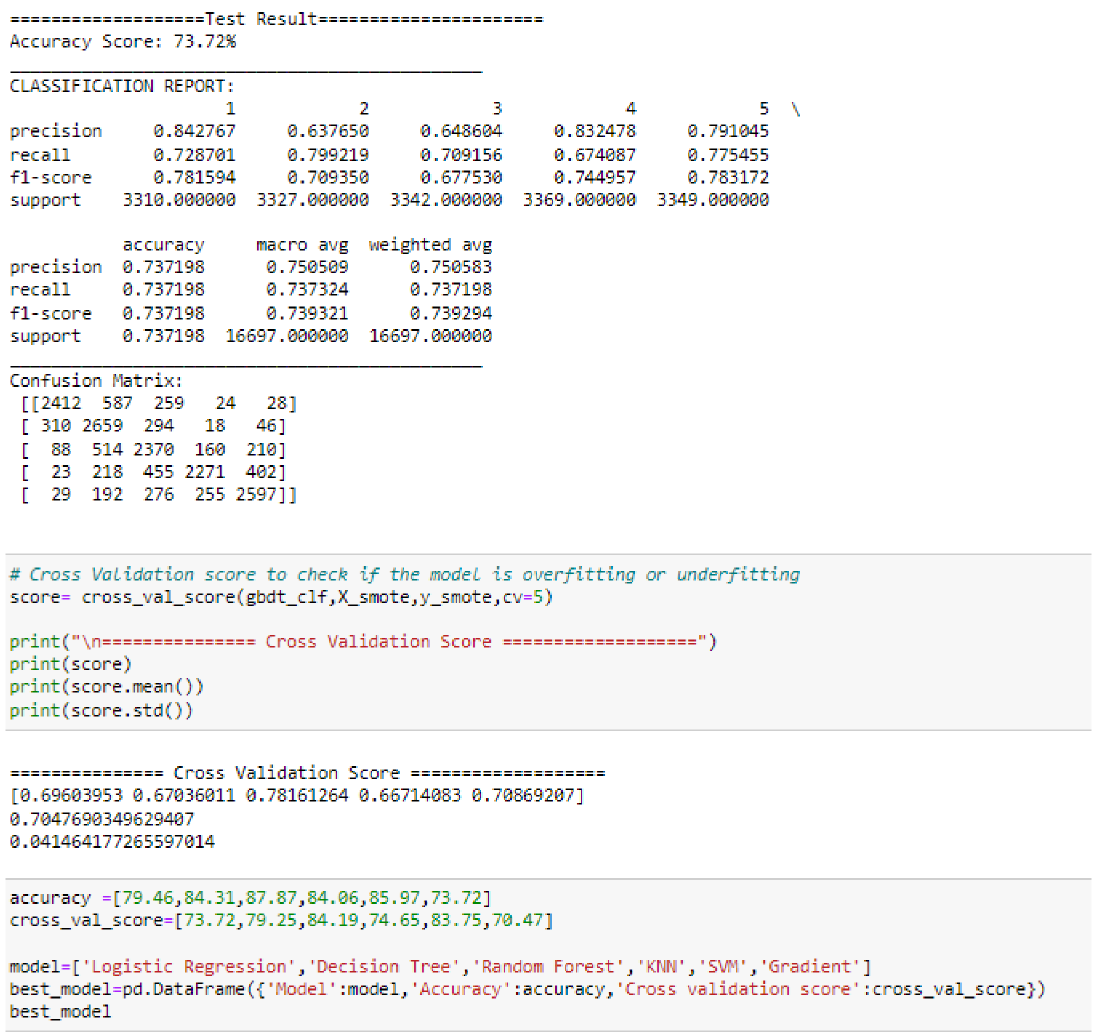
```
# Cross Validation score to check if the model is overfitting or underfitting
score= cross_val_score(svc,X_smote,y_smote,cv=5)

print("\n=============== Cross Validation Score ==================")
print(score)
print(score.mean())
print(score.std())
```

```
=============== Cross Validation Score ==================
[0.83012653 0.79725986 0.89593472 0.84652242 0.81799805]
0.8375683162386764
0.03331654066153871
```

Gradient Boosting Classifier

```
gbdt_clf = GradientBoostingClassifier()
gbdt_clf.fit(x_train,y_train)
pred=gbdt_clf.predict(x_train)
gbdt_clf_report = pd.DataFrame(classification_report(y_train,pred, output_dict=True))

print("\n===================Train Result=====================")

print(f"Accuracy Score: {accuracy_score(y_train,pred) * 100:.2f}%")
print("_______________________________________________")
print(f"CLASSIFICATION REPORT:\n{gbdt_clf_report}")
print("_______________________________________________")
print(f"Confusion Matrix:\n {confusion_matrix(y_train,pred)}\n")

#*********************** Test Score ************************

pred = gbdt_clf.predict(x_test)
gbdt_clf_report = pd.DataFrame(classification_report(y_test,pred, output_dict=True))
print("\n===================Test Result=====================")
print(f"Accuracy Score: {accuracy_score(y_test,pred) * 100:.2f}%")
print("_______________________________________________")
print(f"CLASSIFICATION REPORT:\n{gbdt_clf_report}")
print("_______________________________________________")
print(f"Confusion Matrix:\n {confusion_matrix(y_test,pred)}\n")
```

```
===================Train Result=====================
Accuracy Score: 74.89%
_______________________________________________
CLASSIFICATION REPORT:
                      1             2             3            4  \
precision      0.851281      0.652047      0.662812     0.852882
recall         0.737235      0.827119      0.732501     0.675611
f1-score       0.790164      0.729223      0.695916     0.753966
support    10047.000000  10030.000000  10015.000000  9988.000000

                      5  accuracy     macro avg  weighted avg
precision      0.797604  0.748902      0.763325      0.763295
recall         0.771783  0.748902      0.748850      0.748902
f1-score       0.784481  0.748902      0.750750      0.750762
support    10008.000000  0.748902  50088.000000  50088.000000
_______________________________________________
Confusion Matrix:
 [[7407 1683  800   72   85]
 [ 891 8296  691   39  113]
 [ 237 1423 7336  382  637]
 [  71  667 1377 6748 1125]
 [  95  654  864  671 7724]]
```



```
===================Test Result=====================
Accuracy Score: 73.72%

_______________________________________________
CLASSIFICATION REPORT:
                     1            2            3            4            5  \
precision     0.842767     0.637650     0.648604     0.832478     0.791045
recall        0.728701     0.799219     0.709156     0.674087     0.775455
f1-score      0.781594     0.709350     0.677530     0.744957     0.783172
support    3310.000000  3327.000000  3342.000000  3369.000000  3349.000000

           accuracy     macro avg  weighted avg
precision  0.737198      0.750509      0.750583
recall     0.737198      0.737324      0.737198
f1-score   0.737198      0.739321      0.739294
support    0.737198  16697.000000  16697.000000
_______________________________________________
Confusion Matrix:
 [[2412  587  259   24   28]
 [ 310 2659  294   18   46]
 [  88  514 2370  160  210]
 [  23  218  455 2271  402]
 [  29  192  276  255 2597]]
```

```
# Cross Validation score to check if the model is overfitting or underfitting
score= cross_val_score(gbdt_clf,X_smote,y_smote,cv=5)

print("\n=============== Cross Validation Score ===================")
print(score)
print(score.mean())
print(score.std())
```

```
=============== Cross Validation Score ===================
[0.69603953 0.67036011 0.78161264 0.66714083 0.70869207]
0.7047690349629407
0.041464177265597014
```

```
accuracy =[79.46,84.31,87.87,84.06,85.97,73.72]
cross_val_score=[73.72,79.25,84.19,74.65,83.75,70.47]

model=['Logistic Regression','Decision Tree','Random Forest','KNN','SVM','Gradient']
best_model=pd.DataFrame({'Model':model,'Accuracy':accuracy,'Cross validation score':cross_val_score})
best_model
```

- **Key Metrics for success in solving problem under consideration**

  Precision: It is the ratio between the True Positives and all the Positives. It can be seen as a measure of quality; higher precision means that an algorithm returns more relevant results than irrelevant ones

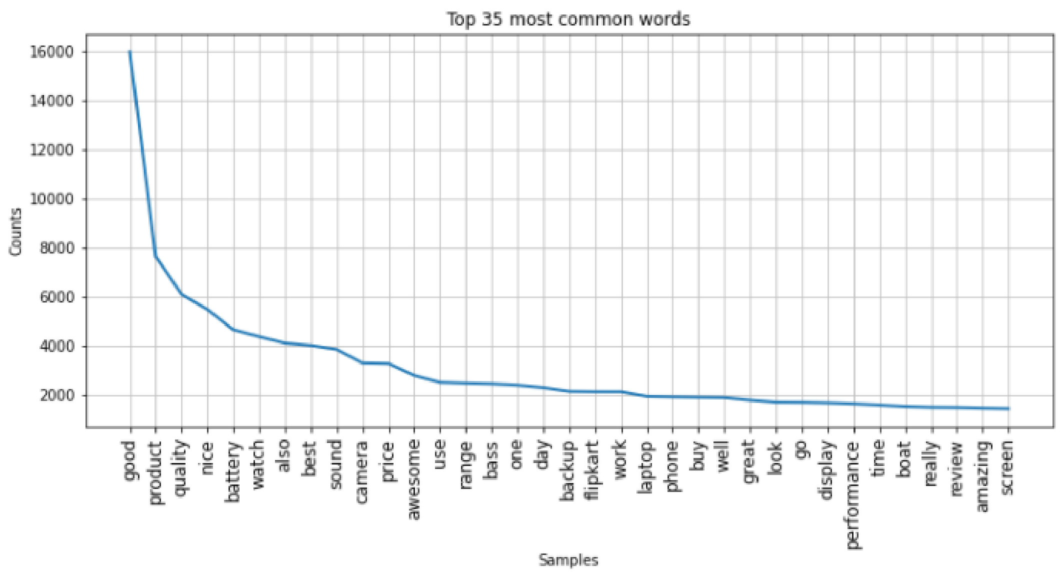
  Recall: It is the measure of our model correctly identifying True Positives. It is used as a measure of quantity and high recall means that an algorithm returns most of the relevant results.

  Accuracy score: It is the ratio of the total number of correct predictions and the total number of predictions. It is used when the True Positives and True negatives are more important. Accuracy can be used when the class distribution is similar

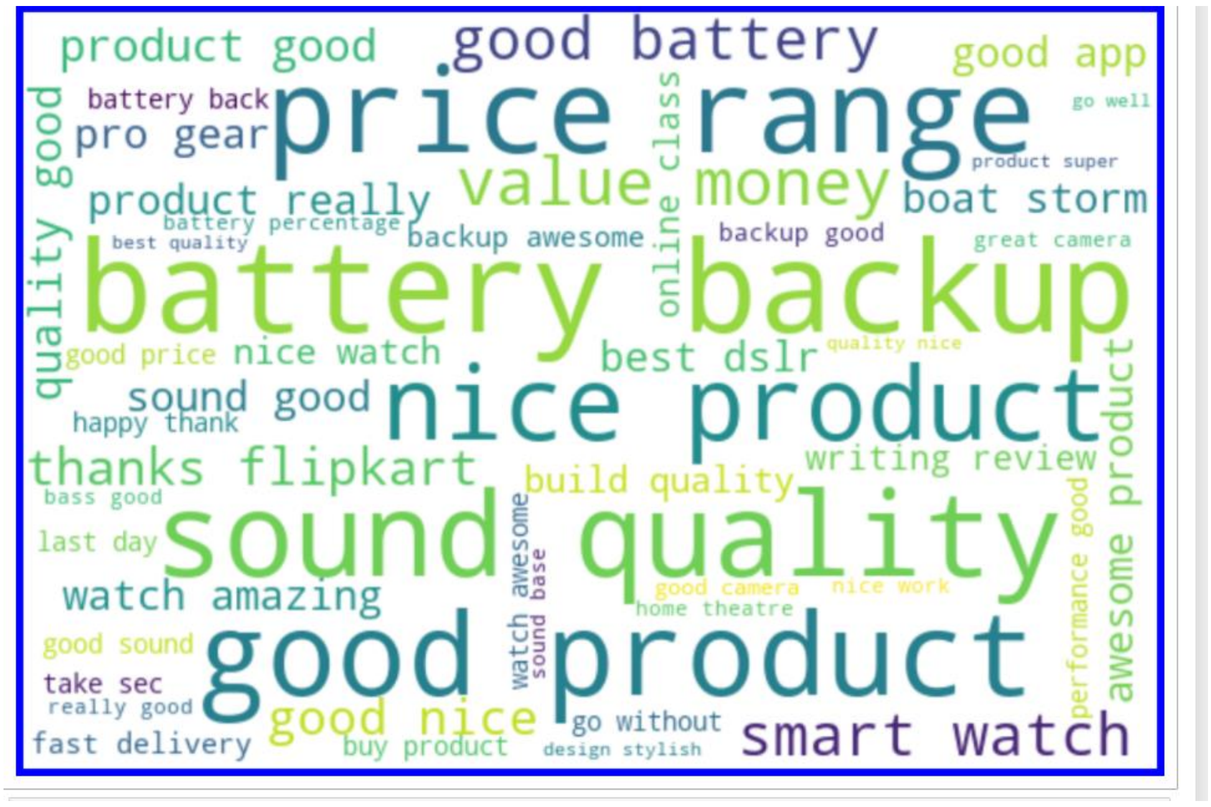
F1-score: It is used when the False Negatives and False Positives are crucial. Hence F1-score is a better metric when there are imbalanced classes.

Cross_val_score: To run cross-validation on multiple metrics and to return train scores, fit times and score times. Get predictions from each split of cross- validation for diagnostic purposes. Make a scorer from a performance metric or loss function

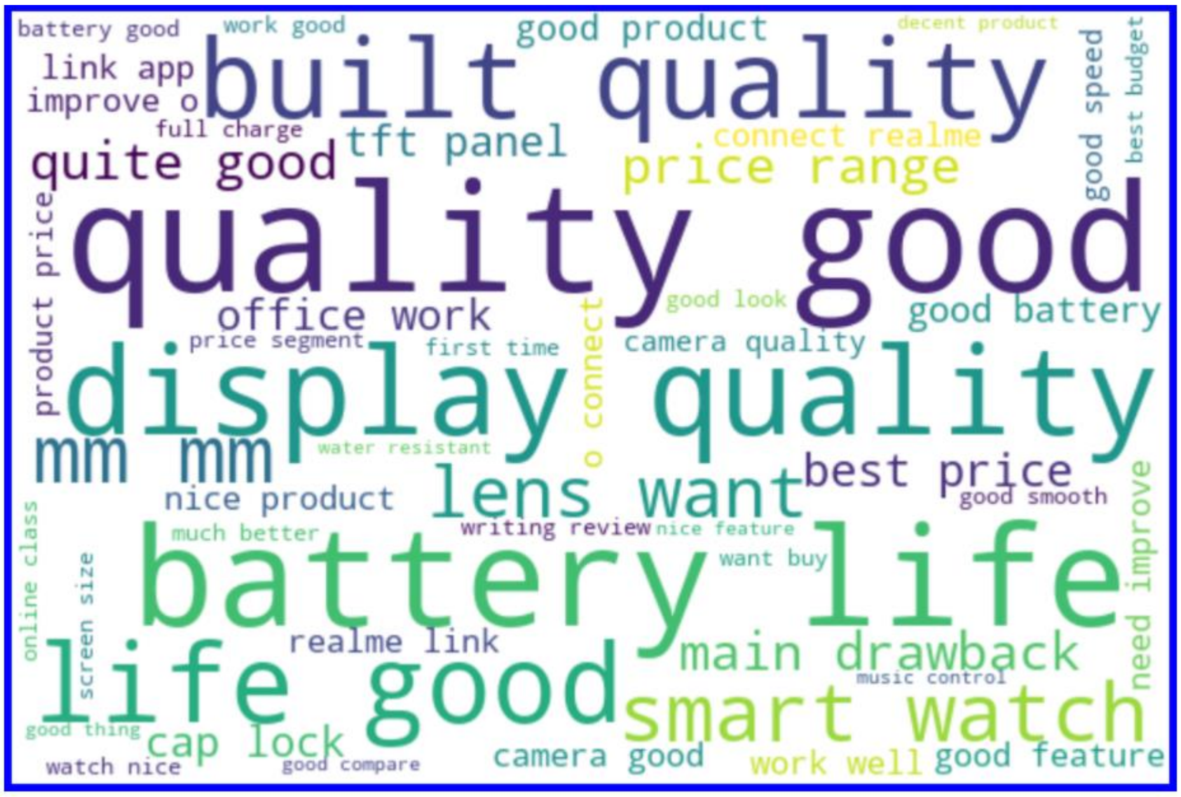
- **Visualizations**



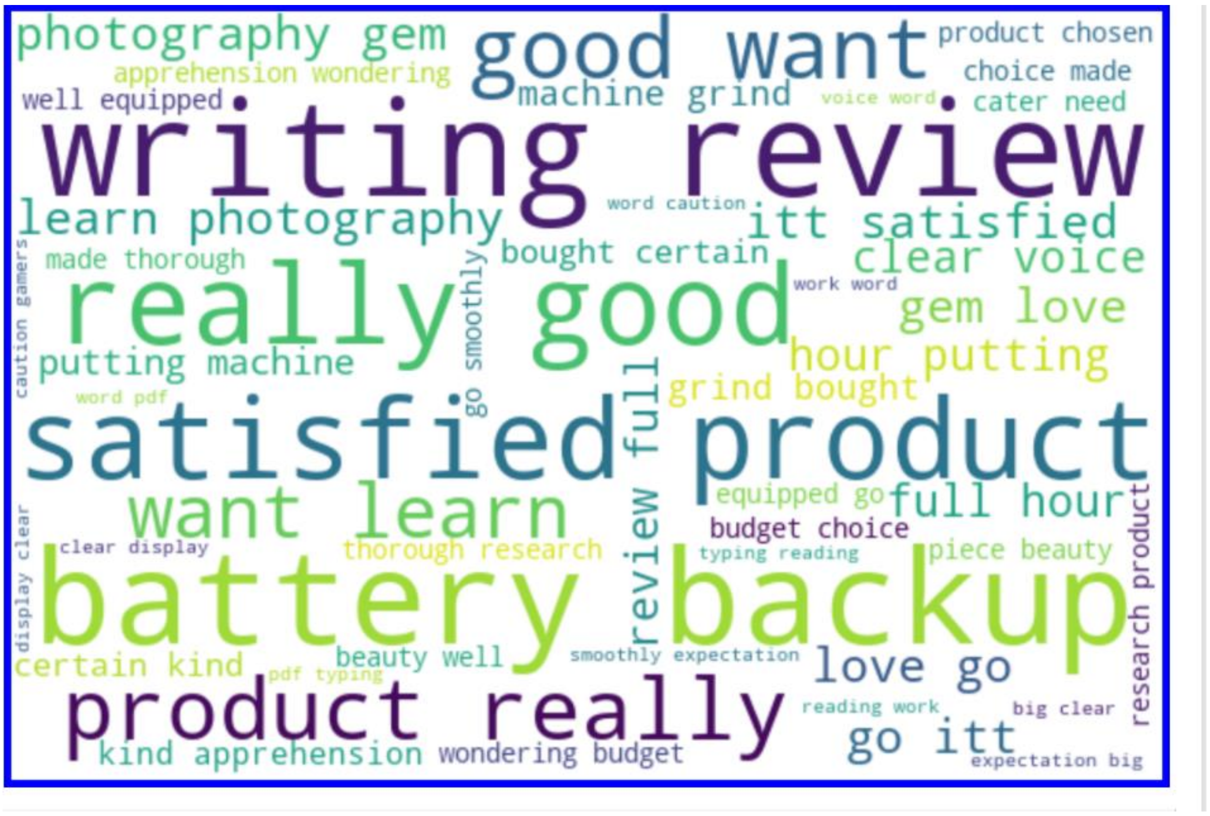
Words appearing more frequently in Cleaned_Full Reviews column that is given rating five by consumers



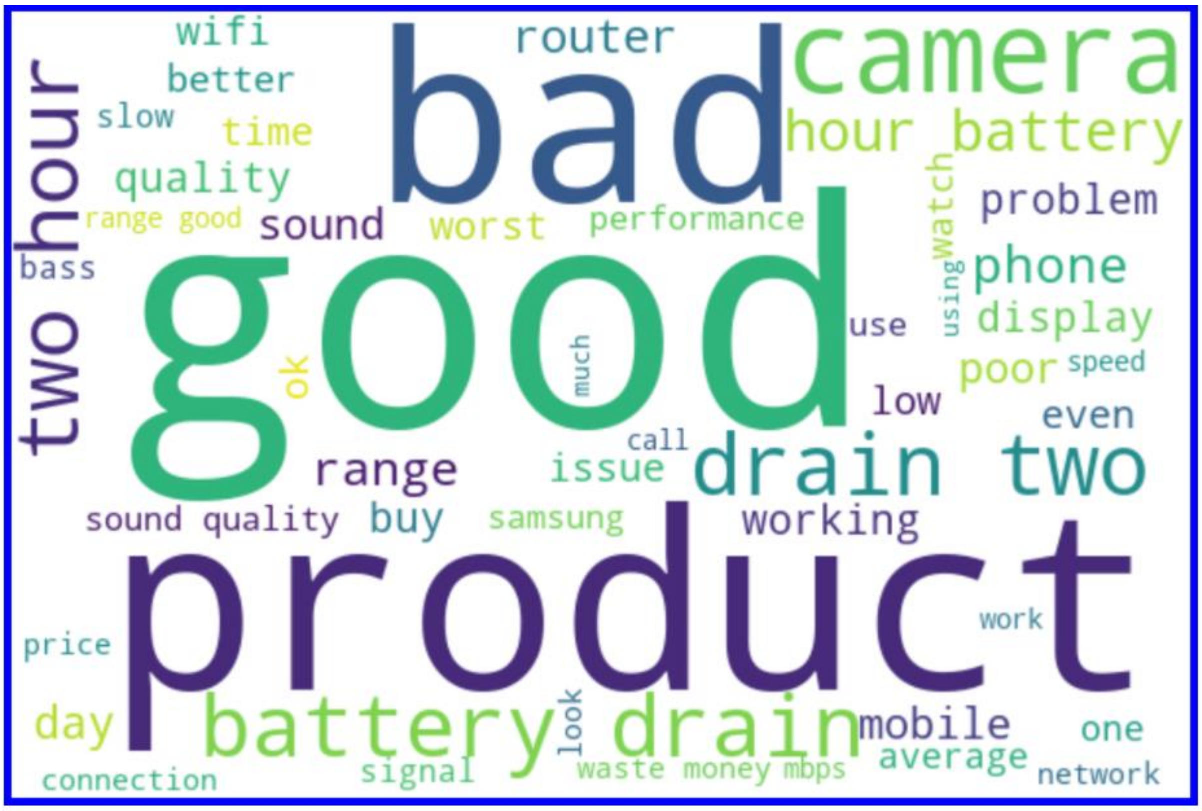
Words appearing more frequently in Cleaned_Full Reviews column that is given rating four by consumers



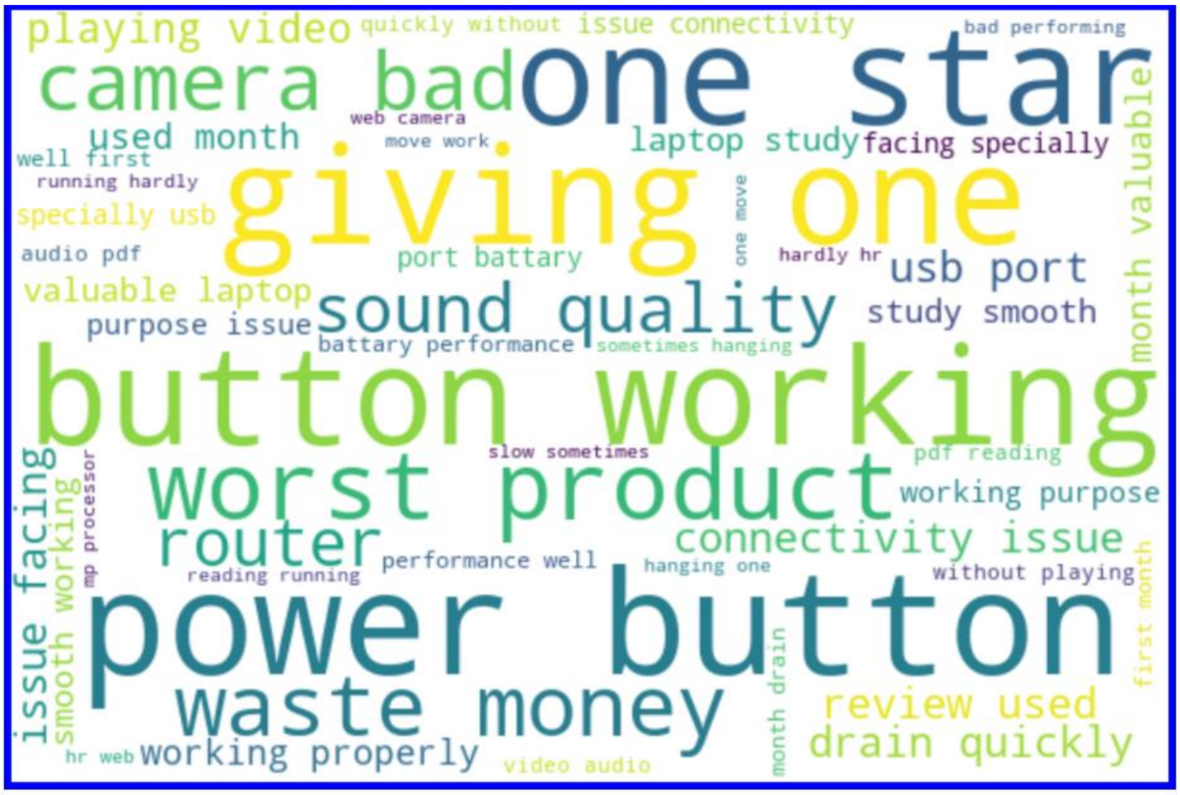
Words appearing more frequently in Cleaned_Full Reviews column that is given rating three by consumers



Words appearing more frequently in Cleaned_Full Reviews column that is given rating two by consumers



Words appearing more frequently in Cleaned_Full Reviews column that is given rating one by consumers



- **Interpretation of the Results**

  • Our model performs reasonably well on 1,4 and 5-star ratings and relatively bad on 2 and 3-star ratings.

  Ratings 5: Words like perfect, nice, best, excellent, good has been mostly used.

  Ratings 4: Words like good, nice, best, better has been mostly used

  Ratings 3: Words like average, low, poor, difficult has been mostly used

  Ratings 2: Words like slow, poor, barely, problem has been mostly used

  Ratings 1: Words like bad, disappointed, poor, one star has been mostly used

  • The analysis also reveals that the user reviews are inconsistent with user numeric ratings, and that numeric ratings are higher than user reviews might suggest.

  • Some of the reviews were bad and the texts had more wrong information about the product.

CONCLUSION

- **Key Findings and Conclusions of the Study**

  From the whole project evaluation these are the inferences that I could draw from the visualization of data.
  • The numeric ratings given for products may be biased and overrated because higher ratings given by users potentially attract several new users disproportionately.
  • However, the large increase in review-based data implies a need to focus also on performing predictions. This process is

challenging yet fruitful, as user reviews are qualitative while ratings are essentially quantitative.
• WordCloud was not showing proper text which had more positive and negative weightage.
• User reviews are limited to identifying polarity and subjectivity.
• It is very difficult to model a customer's rating behavior properly. The correctness of a model highly depends on what kind of information the dataset provides and the structure of the model.

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

Through Visualization it was clear that the target variable was an imbalanced dataset. Also, the features that depicted a lot of storytelling when visualization was done for all of them.

Visualization gives meaning to a data which helps drawing inference from it. This project allowed me to work with two different deep learning models and additionally, I was able to implement them on a Natural Language Processing use-case. The various data pre-processing and feature engineering steps in the project made me cognizant of the efficient methods that can be used to clean textual data.

Random Forest classifier was the best model to be deployed in production because its difference between its accuracy and CV value was least among all models.

Through this project we were able to learn various Natural language processing techniques like lemmatization, stemming, removal of stopwords. This project has demonstrated the

importance of sampling effectively, modelling and predicting data.
The dataset was highly imbalanced so we balanced the dataset using smote technique. We converted text data into vectors with the help of vectorizer.

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

However, our method will not be able to judge common affection of user in capturing the sentiment of review texts.

For further improvement of results, Recurrent neural networks offer extremely high performance on natural language processing problems, and if the architecture for the inference step were implemented efficiently the computational overhead would be minimal.

Thank You