FLIP ROBO

Ratings Prediction Project

Submitted by:

Sumit Santra

ACKNOWLEDGMENT

I Would to like to express my Special thanks of gratitude to my SME **<u>Shubham Yadav</u>**, Who gave me thr golden opportunity to do this wonderful project of **<u>Rating Prediction Project</u>**

Who also helped me in completing my project. I came to know about so many new things , I am really thankful to them.

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

  ➔ For this project we should know the concepts behind the Rating which are good ratings and which one are Average, Fair, Bad, Very Good and Excellent by knowing the concepts behind the ratings we will be able to distinguish between those words and sentences.

  ➔ We Should also know some of the people deliberately gives less ratings to the product for there personal reasons.

  ➔ If you know the basic rating system you will understand the project.

- ## Review of Literature

  ➔ First I Started researching for Some E-commerce website because we need some E-commerce website to scrape the reviews and ratings from those website and use it for our Model Building.

  ➔ Then after that I Started Searching for products and started reading the reviews of those product so that I can find the products which has more number of reviews and has good number of 1 Star, 2 Star, 3 Star , 4 Star and 5 Star Ratings so that we can scrape equal number of reviews and Ratings for that product and dataset becomes balanced.

➔ After that I made the list of the product from different E-commerce website so that it will be easy for me to do it while scraping the dataset.

- **<u>Motivation for the Problem Undertaken</u>**

➔ My motivation to undertake this Project is my mentor Shubham Yadav Sir & to get the knowledge about the Web Scraping and Natural Language Processing So that it will be easy for me to undertake this kind of project easily as I know the Concepts behind the NLP and how to pre process the data and predict the dependent variable.

Analytical Problem Framing

- **<u>Mathematical/ Analytical Modeling of the Problem</u>**

➔ In NLP we have used Vectorization technique to give words a special vector according to there weight in the particular sentence.

➔ The Vectorization is done by different techniques I used **<u>Term frequency and Inverse Document frequency.</u>** It gives unique vectors to every words in a sentence and save it as an array.

Scikit-Learn

- $\mathrm{IDF}(t) = \log\frac{1+n}{1+\mathrm{df}(t)} + 1$

Standard notation

- $\mathrm{IDF}(t) = \log\frac{n}{\mathrm{df}(t)}$

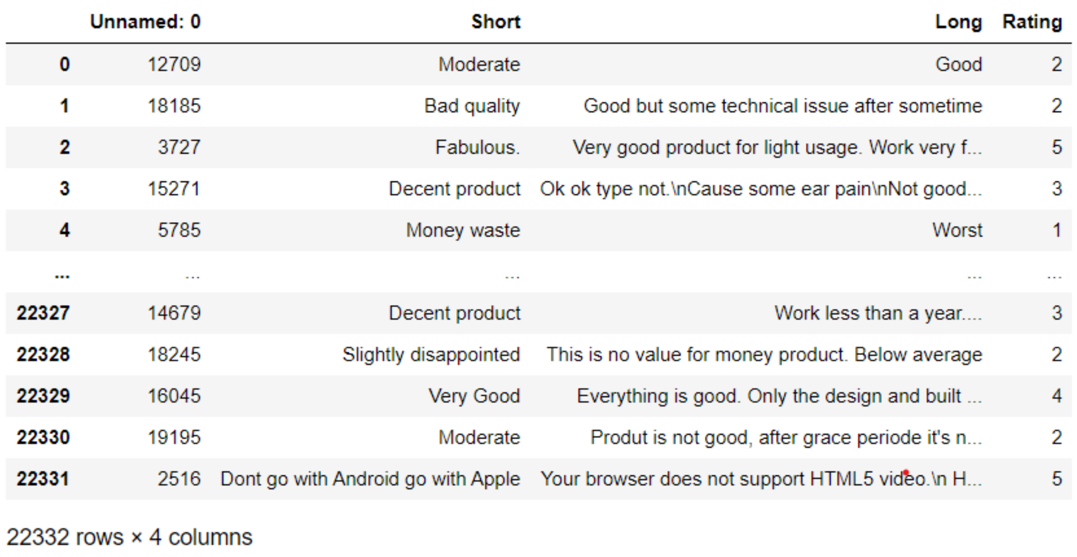
➔ From above we can see the formula for Term frequency and inverse document frequency.

- ## <u>Data Sources and their formats</u>

➔ Data Sources are different ecommerce website from which I scraped all the important reviews and ratings. They where in html format I converted them into list and later converted them into dictionary and later to dataframe format.

➔ The necessary things in this data are the Reviews and the Ratings of the product from which we have to predict the ratings from the Reviews of the product.

➔ After Scraping the Product across 2 E-commerce website and 10-15 products I scraped about 23000 rows of data including the reviews and the ratings of the product.



	Unnamed: 0	Short	Long	Rating
0	12709	Moderate	Good	2
1	18185	Bad quality	Good but some technical issue after sometime	2
2	3727	Fabulous.	Very good product for light usage. Work very f...	5
3	15271	Decent product	Ok ok type not.\nCause some ear pain\nNot good...	3
4	5785	Money waste	Worst	1
...	...	...	...	...
22327	14679	Decent product	Work less than a year....	3
22328	18245	Slightly disappointed	This is no value for money product. Below average	2
22329	16045	Very Good	Everything is good. Only the design and built ...	4
22330	19195	Moderate	Produt is not good, after grace periode it's n...	2
22331	2516	Dont go with Android go with Apple	Your browser does not support HTML5 video.\n H...	5

22332 rows × 4 columns

- ## <u>Data Preprocessing Done</u>

➔ From the above data first I removed the un-used columns such as Unnamed: 0 and Short as this is a long review problem so I removed the Short reviews and only kept Long reviews as that column was the only important columns in the whole dataset.

- ➔ After dropping both the columns I checked for Null values in the dataset and found the dataset has 2 missing values so I removed those two null values using dropna method in pandas dataframe.
- ➔ After removing the null data from the dataset I removed punctuations as they don't speak more above the sentence and It will increase the length of the dataset without a prefect meaning after removing the Punctuation I moved to the words.
- ➔ Here in the reviews we have mixed upper case and lower case words that has same meaning like Amazing and amazing has the same meaning but the program cannot distinguish between those so I lowered every word in the dataset so that we get the same meaning for both the words.
- ➔ After lowering the words we need to remove the stopwords to reduce the dimension of the dataset. Stopwords are the words which don't gives meaning to the sentence and has been written repeated times in a sentence after removing the stopwords we are with our final data.

• Data Inputs- Logic- Output Relationships

- ➔ The relation between the Independent variable and the dependent variable is that from a sentence we have to predict the ratings of the product.
- ➔ If there is a negative word in a sentence then the rating of the product decreases and Vice Versa

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

- ➔ My presumptions were that I will only get English reviews after scraping all the reviews I've seen many words like accha, akele, acchi and I thought how to remove it I came with an idea by removing those words with the help of stop words I added those hindi words and remove them using stopwords

➔ Then I even got kannada, Chinese words in the dataset so I removed those by using regular expression by selecting only a-zA-Z and rest I replace with a White Space and I got rid of those unwanted words in the dataset as well as the model.

- ## Hardware and Software Requirements and Tools Used

Hardware Used:

➔ CPU – AMD RYZEN 5800HX
➔ RAM – 16GB
➔ STORAGE – 1 TB SSD

Software Used:

➔ OS – Windows 10
➔ IDE – JUPYTER NOTEBOOK

Python Libraries Used:

➔ Sklearn – Machine Learning Library
➔ Pandas – Panel and Data
➔ Numpy – Numeric Python
➔ Seaborn – Visualization
➔ Matplotlib.pyplot -- Visualization
➔ Nltk – Natural Language Processing toolkit
➔ Joblib – Saving the model
➔ XGBOOST – Boosting Algorithm

Model/s Development and Evaluation

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

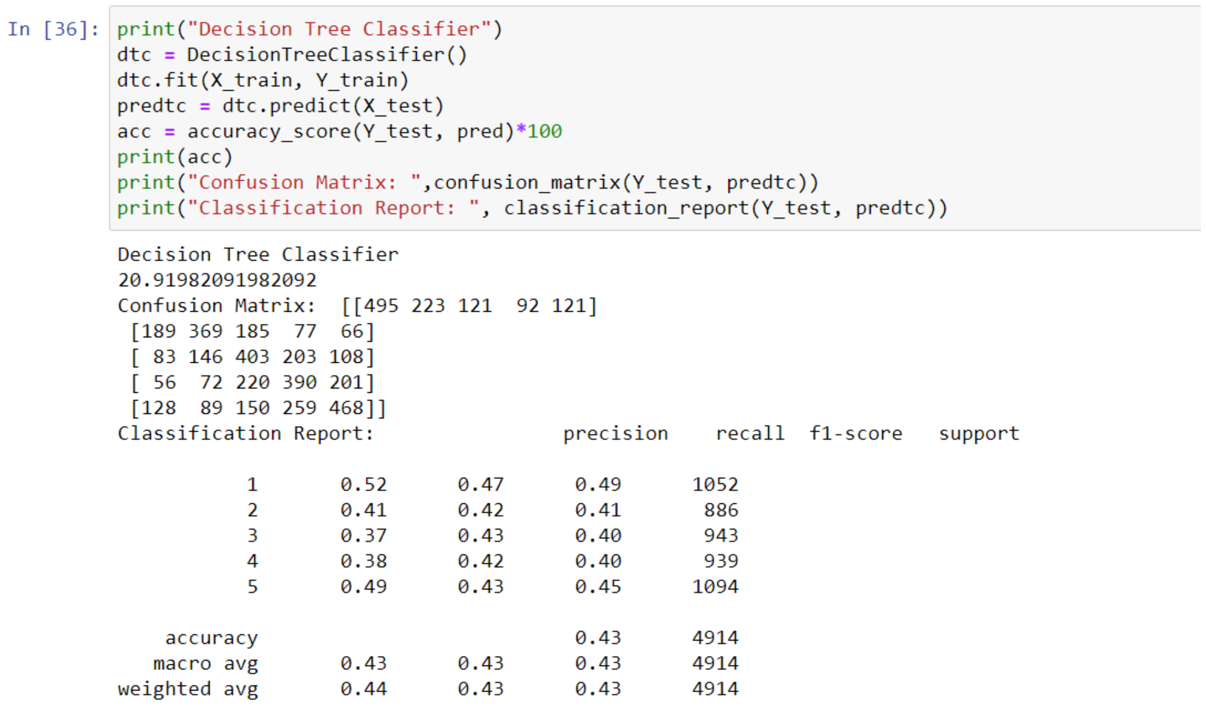
- ➔ I started by looking at the dataset and removing the null values present in the dataset after removing the null data from the columns
- ➔ After removing the null values I left with the raw data and found that there where more data which has same meaning but different cases such as Big and big has same meaning but there case differs and because of which the dataset get bigger after performing vectorization.
- ➔ So I converted all the data in the rows to lower case and then I performed Stopwords removal from the dataset as stopword has no meaning so we can remove those.
- ➔ After removing stopwords my main focus was removing punctuations from the dataset as it will increase the size of the dataset.
- ➔ After removing the punctuations I performed Term Frequency and inverse document frequency so that I can get unique vectors for each and every word
- ➔ After tfidf using for loop I changed the max_columns parameter in rfidf and found max_feature = 11000 has the highest accuracy so I used that for each and every model.

• <u>**Testing of Identified Approaches (Algorithms)**</u>

- ➔ Decision Tree Classifier
- ➔ Extra Tree Classifier
- ➔ Random Forest Classifier
- ➔ Adaboost Classifier
- ➔ Support Vector Classifier
- ➔ Kneighbours Classifier
- ➔ XGBoost Classifier
- ➔ XGBRF Classifier

- **<u>Run and Evaluate selected models</u>**

➔ Decision Tree:



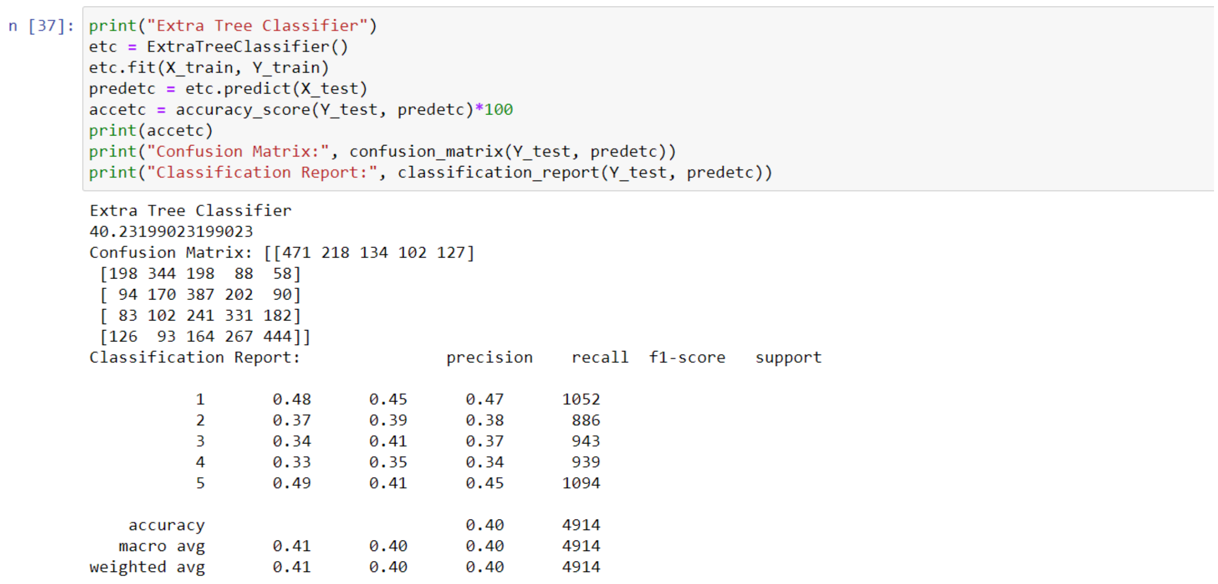
```
In [36]: print("Decision Tree Classifier")
         dtc = DecisionTreeClassifier()
         dtc.fit(X_train, Y_train)
         predtc = dtc.predict(X_test)
         acc = accuracy_score(Y_test, pred)*100
         print(acc)
         print("Confusion Matrix: ",confusion_matrix(Y_test, predtc))
         print("Classification Report: ", classification_report(Y_test, predtc))
```

```
Decision Tree Classifier
20.91982091982092
Confusion Matrix:  [[495 223 121  92 121]
 [189 369 185  77  66]
 [ 83 146 403 203 108]
 [ 56  72 220 390 201]
 [128  89 150 259 468]]
Classification Report:                precision    recall  f1-score   support

           1       0.52      0.47      0.49      1052
           2       0.41      0.42      0.41       886
           3       0.37      0.43      0.40       943
           4       0.38      0.42      0.40       939
           5       0.49      0.43      0.45      1094

    accuracy                           0.43      4914
   macro avg       0.43      0.43      0.43      4914
weighted avg       0.44      0.43      0.43      4914
```

➔ Extra Tree Classifier:



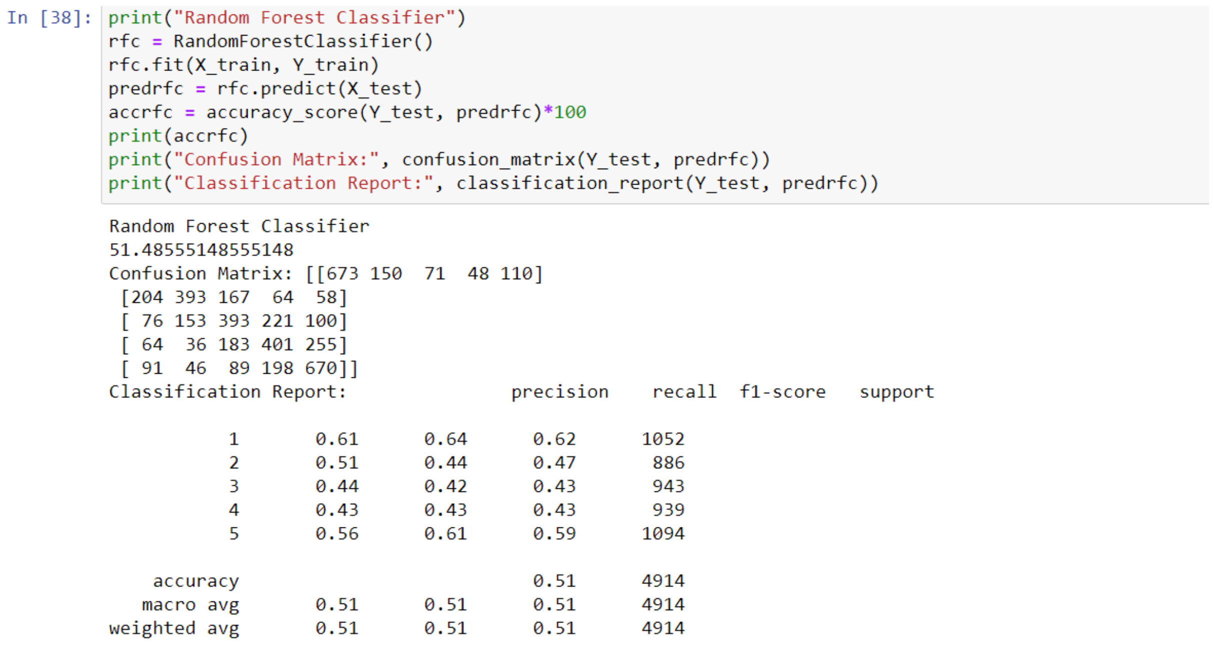
```
n [37]: print("Extra Tree Classifier")
        etc = ExtraTreeClassifier()
        etc.fit(X_train, Y_train)
        predetc = etc.predict(X_test)
        accetc = accuracy_score(Y_test, predetc)*100
        print(accetc)
        print("Confusion Matrix:", confusion_matrix(Y_test, predetc))
        print("Classification Report:", classification_report(Y_test, predetc))
```

```
Extra Tree Classifier
40.23199023199023
Confusion Matrix: [[471 218 134 102 127]
 [198 344 198  88  58]
 [ 94 170 387 202  90]
 [ 83 102 241 331 182]
 [126  93 164 267 444]]
Classification Report:               precision    recall  f1-score   support

           1       0.48      0.45      0.47      1052
           2       0.37      0.39      0.38       886
           3       0.34      0.41      0.37       943
           4       0.33      0.35      0.34       939
           5       0.49      0.41      0.45      1094

    accuracy                           0.40      4914
   macro avg       0.41      0.40      0.40      4914
weighted avg       0.41      0.40      0.40      4914
```

➔ Random Forest Classifier:



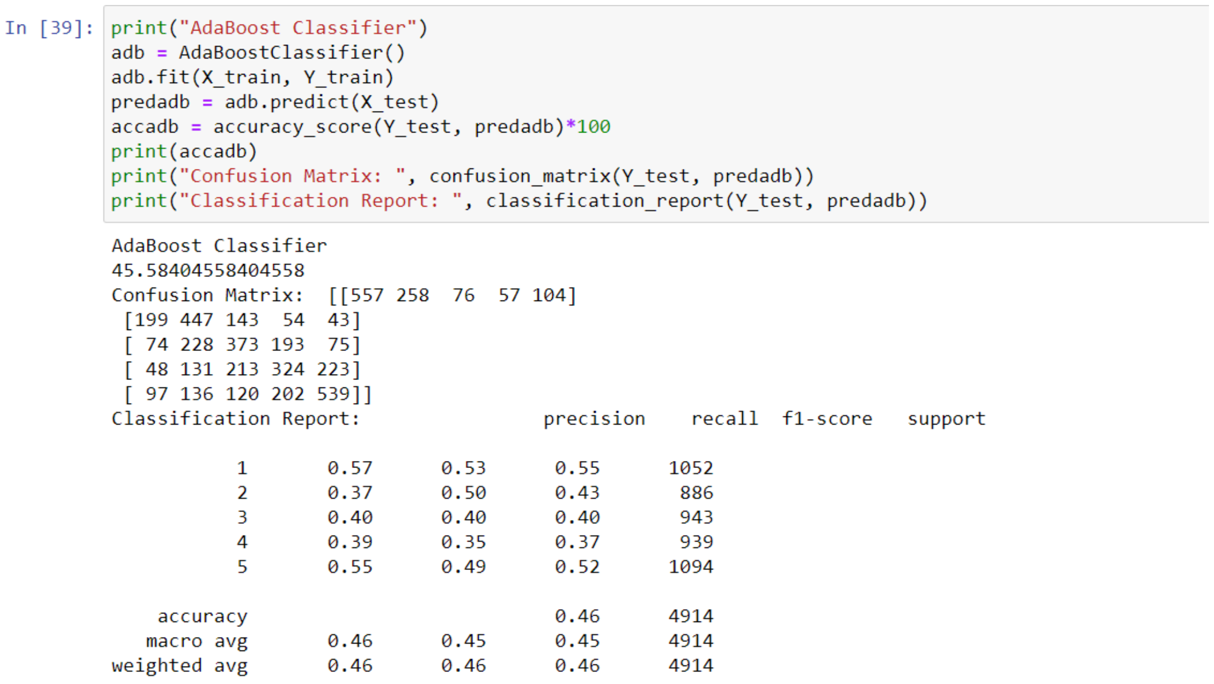
```
In [38]: print("Random Forest Classifier")
         rfc = RandomForestClassifier()
         rfc.fit(X_train, Y_train)
         predrfc = rfc.predict(X_test)
         accrfc = accuracy_score(Y_test, predrfc)*100
         print(accrfc)
         print("Confusion Matrix:", confusion_matrix(Y_test, predrfc))
         print("Classification Report:", classification_report(Y_test, predrfc))
```

```
Random Forest Classifier
51.48555148555148
Confusion Matrix: [[673 150  71  48 110]
 [204 393 167  64  58]
 [ 76 153 393 221 100]
 [ 64  36 183 401 255]
 [ 91  46  89 198 670]]
Classification Report:               precision    recall  f1-score   support

           1       0.61      0.64      0.62      1052
           2       0.51      0.44      0.47       886
           3       0.44      0.42      0.43       943
           4       0.43      0.43      0.43       939
           5       0.56      0.61      0.59      1094

    accuracy                           0.51      4914
   macro avg       0.51      0.51      0.51      4914
weighted avg       0.51      0.51      0.51      4914
```

➔ Adaboost Classifier:



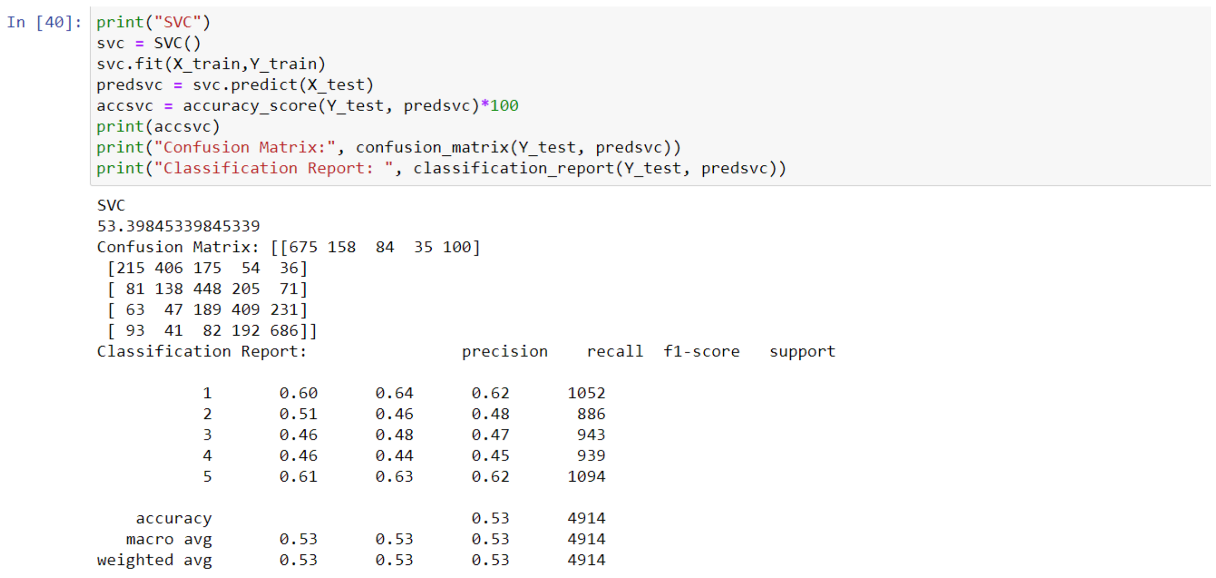
```
In [39]: print("AdaBoost Classifier")
         adb = AdaBoostClassifier()
         adb.fit(X_train, Y_train)
         predadb = adb.predict(X_test)
         accadb = accuracy_score(Y_test, predadb)*100
         print(accadb)
         print("Confusion Matrix: ", confusion_matrix(Y_test, predadb))
         print("Classification Report: ", classification_report(Y_test, predadb))
```

```
AdaBoost Classifier
45.58404558404558
Confusion Matrix:  [[557 258  76  57 104]
 [199 447 143  54  43]
 [ 74 228 373 193  75]
 [ 48 131 213 324 223]
 [ 97 136 120 202 539]]
Classification Report:                precision    recall  f1-score   support

           1       0.57      0.53      0.55      1052
           2       0.37      0.50      0.43       886
           3       0.40      0.40      0.40       943
           4       0.39      0.35      0.37       939
           5       0.55      0.49      0.52      1094

    accuracy                           0.46      4914
   macro avg       0.46      0.45      0.45      4914
weighted avg       0.46      0.46      0.46      4914
```

➔ SVC :



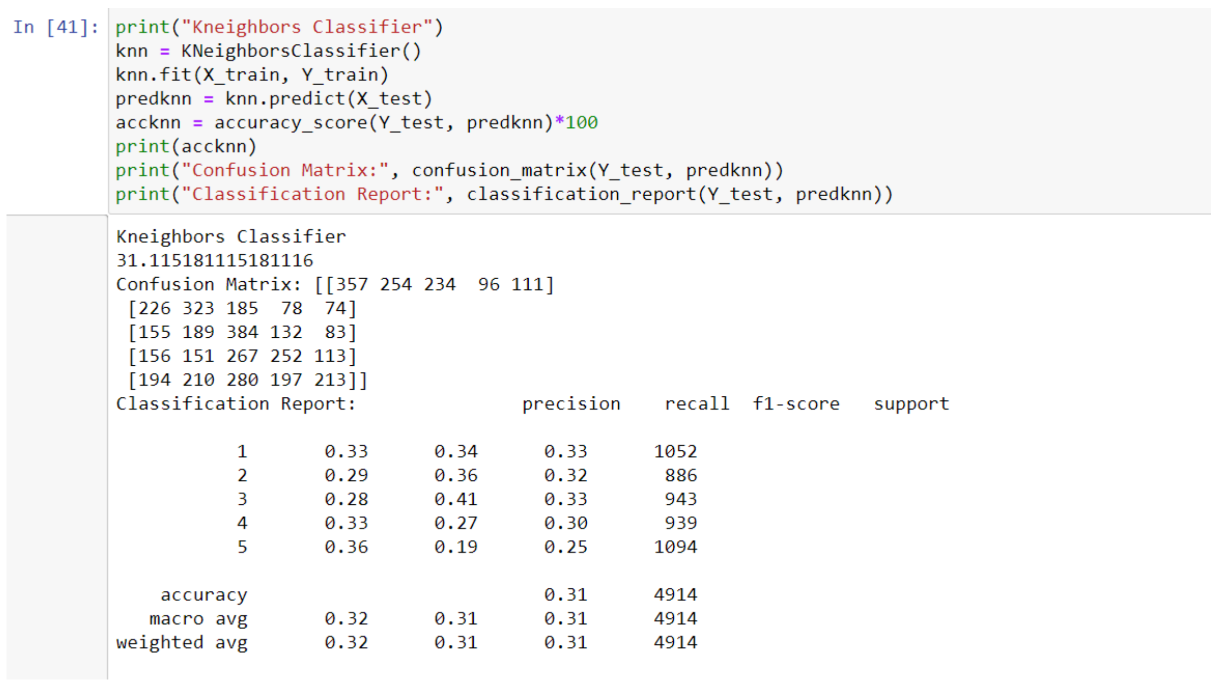
```
In [40]: print("SVC")
         svc = SVC()
         svc.fit(X_train,Y_train)
         predsvc = svc.predict(X_test)
         accsvc = accuracy_score(Y_test, predsvc)*100
         print(accsvc)
         print("Confusion Matrix:", confusion_matrix(Y_test, predsvc))
         print("Classification Report: ", classification_report(Y_test, predsvc))
```

```
SVC
53.39845339845339
Confusion Matrix: [[675 158  84  35 100]
 [215 406 175  54  36]
 [ 81 138 448 205  71]
 [ 63  47 189 409 231]
 [ 93  41  82 192 686]]
Classification Report:                precision    recall  f1-score   support

           1       0.60      0.64      0.62      1052
           2       0.51      0.46      0.48       886
           3       0.46      0.48      0.47       943
           4       0.46      0.44      0.45       939
           5       0.61      0.63      0.62      1094

    accuracy                           0.53      4914
   macro avg       0.53      0.53      0.53      4914
weighted avg       0.53      0.53      0.53      4914
```

➔ Kneighbours Classifier:



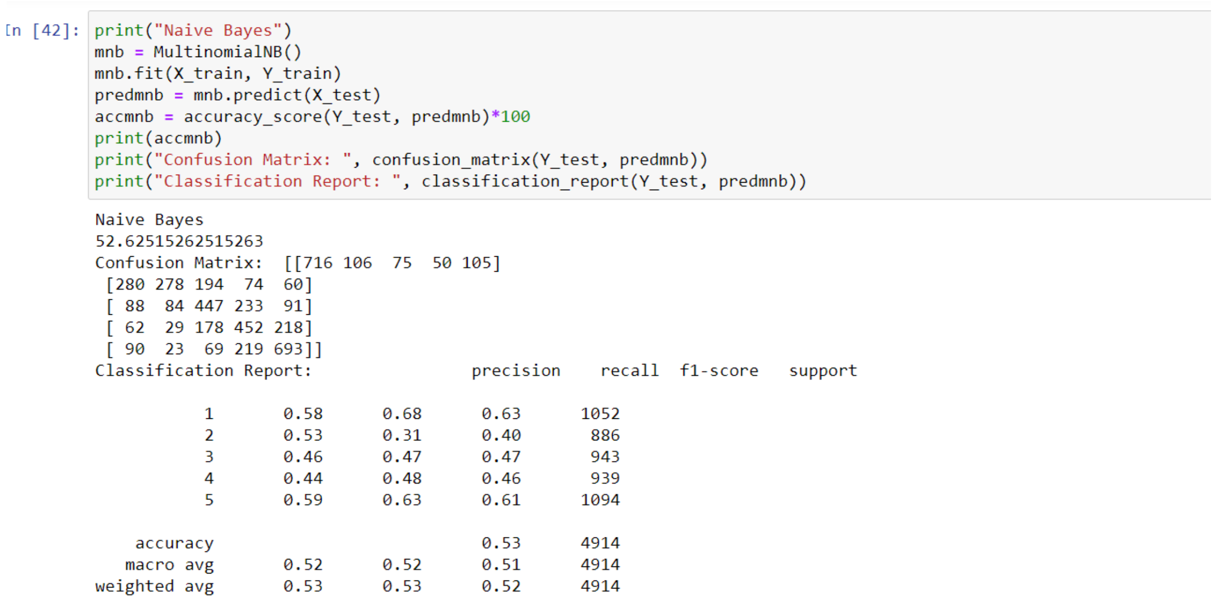
```
In [41]: print("Kneighbors Classifier")
         knn = KNeighborsClassifier()
         knn.fit(X_train, Y_train)
         predknn = knn.predict(X_test)
         accknn = accuracy_score(Y_test, predknn)*100
         print(accknn)
         print("Confusion Matrix:", confusion_matrix(Y_test, predknn))
         print("Classification Report:", classification_report(Y_test, predknn))
```

```
Kneighbors Classifier
31.115181115181116
Confusion Matrix: [[357 254 234  96 111]
 [226 323 185  78  74]
 [155 189 384 132  83]
 [156 151 267 252 113]
 [194 210 280 197 213]]
Classification Report:               precision    recall  f1-score   support

           1       0.33      0.34      0.33      1052
           2       0.29      0.36      0.32       886
           3       0.28      0.41      0.33       943
           4       0.33      0.27      0.30       939
           5       0.36      0.19      0.25      1094

    accuracy                           0.31      4914
   macro avg       0.32      0.31      0.31      4914
weighted avg       0.32      0.31      0.31      4914
```

➔ Naïve Bayes:



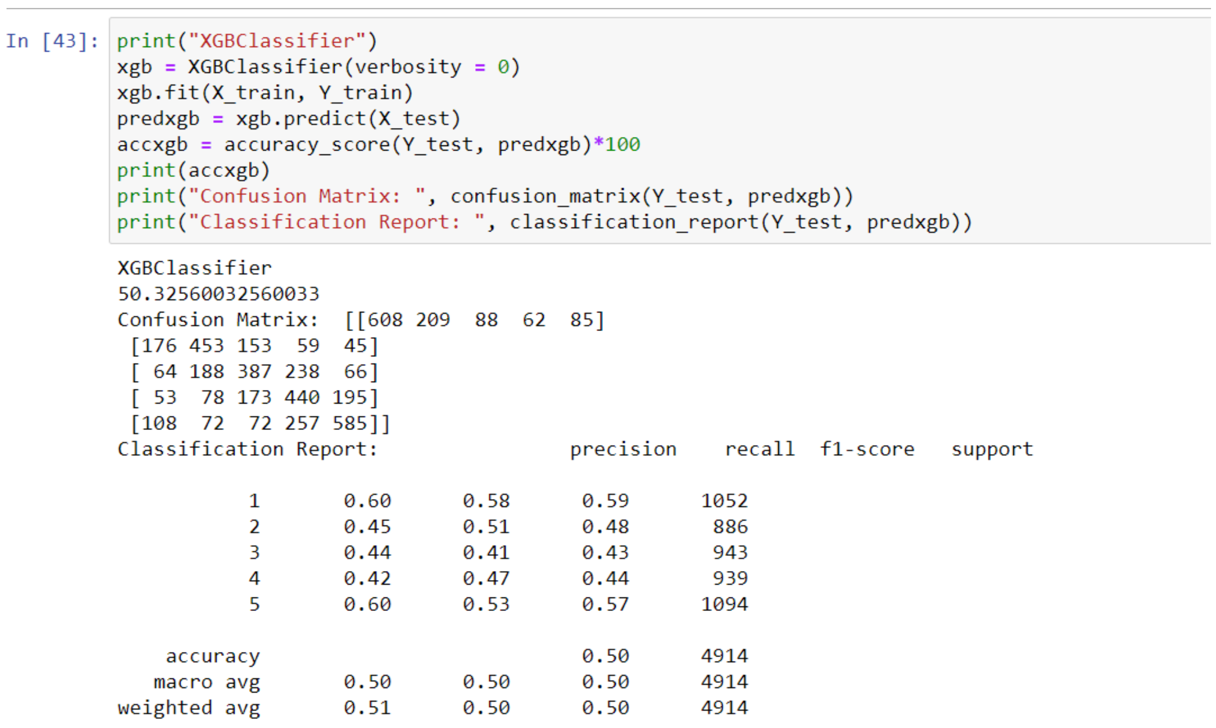
```
In [42]: print("Naive Bayes")
         mnb = MultinomialNB()
         mnb.fit(X_train, Y_train)
         predmnb = mnb.predict(X_test)
         accmnb = accuracy_score(Y_test, predmnb)*100
         print(accmnb)
         print("Confusion Matrix: ", confusion_matrix(Y_test, predmnb))
         print("Classification Report: ", classification_report(Y_test, predmnb))
```

```
Naive Bayes
52.62515262515263
Confusion Matrix:  [[716 106  75  50 105]
 [280 278 194  74  60]
 [ 88  84 447 233  91]
 [ 62  29 178 452 218]
 [ 90  23  69 219 693]]
Classification Report:                precision    recall  f1-score   support

           1       0.58      0.68      0.63      1052
           2       0.53      0.31      0.40       886
           3       0.46      0.47      0.47       943
           4       0.44      0.48      0.46       939
           5       0.59      0.63      0.61      1094

    accuracy                           0.53      4914
   macro avg       0.52      0.52      0.51      4914
weighted avg       0.53      0.53      0.52      4914
```

➔ XGBoost Classifier



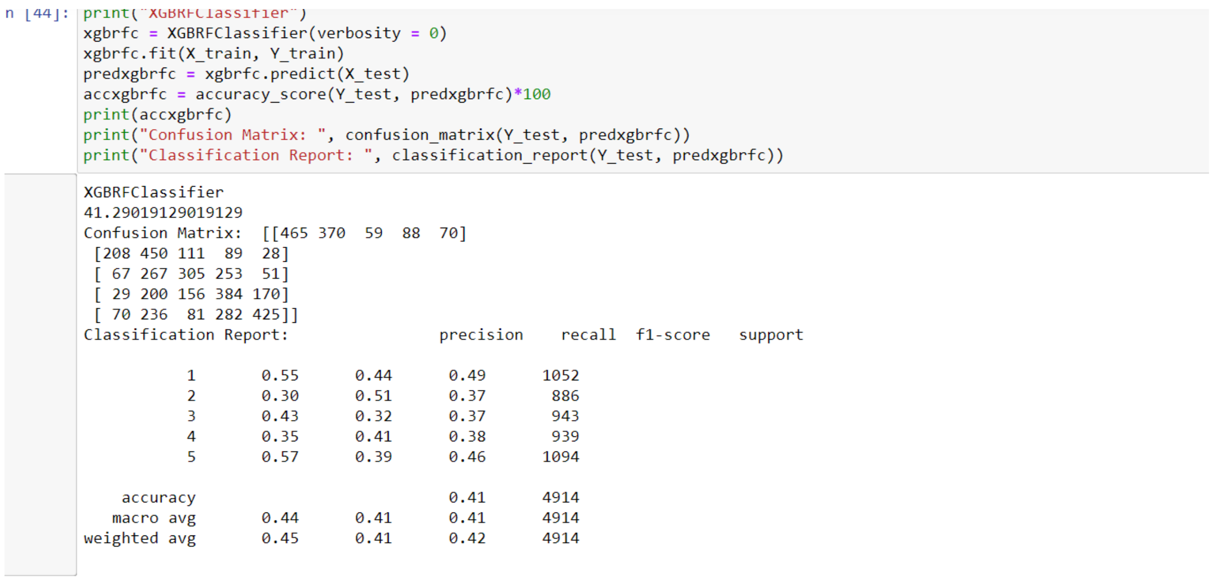
```
In [43]: print("XGBClassifier")
         xgb = XGBClassifier(verbosity = 0)
         xgb.fit(X_train, Y_train)
         predxgb = xgb.predict(X_test)
         accxgb = accuracy_score(Y_test, predxgb)*100
         print(accxgb)
         print("Confusion Matrix: ", confusion_matrix(Y_test, predxgb))
         print("Classification Report: ", classification_report(Y_test, predxgb))
```

```
XGBClassifier
50.32560032560033
Confusion Matrix:  [[608 209  88  62  85]
 [176 453 153  59  45]
 [ 64 188 387 238  66]
 [ 53  78 173 440 195]
 [108  72  72 257 585]]
Classification Report:                precision    recall  f1-score   support

           1       0.60      0.58      0.59      1052
           2       0.45      0.51      0.48       886
           3       0.44      0.41      0.43       943
           4       0.42      0.47      0.44       939
           5       0.60      0.53      0.57      1094

    accuracy                           0.50      4914
   macro avg       0.50      0.50      0.50      4914
weighted avg       0.51      0.50      0.50      4914
```

➔ XGBRF Classifier:


```
n [44]: print("XGBRFClassifier")
        xgbrfc = XGBRFClassifier(verbosity = 0)
        xgbrfc.fit(X_train, Y_train)
        predxgbrfc = xgbrfc.predict(X_test)
        accxgbrfc = accuracy_score(Y_test, predxgbrfc)*100
        print(accxgbrfc)
        print("Confusion Matrix: ", confusion_matrix(Y_test, predxgbrfc))
        print("Classification Report: ", classification_report(Y_test, predxgbrfc))
```

```
XGBRFClassifier
41.29019129019129
Confusion Matrix:  [[465 370  59  88  70]
 [208 450 111  89  28]
 [ 67 267 305 253  51]
 [ 29 200 156 384 170]
 [ 70 236  81 282 425]]
Classification Report:                precision    recall  f1-score   support

           1       0.55      0.44      0.49      1052
           2       0.30      0.51      0.37       886
           3       0.43      0.32      0.37       943
           4       0.35      0.41      0.38       939
           5       0.57      0.39      0.46      1094

    accuracy                           0.41      4914
   macro avg       0.44      0.41      0.41      4914
weighted avg       0.45      0.41      0.42      4914
```

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

➔ I used three metrics for this problem

1) Confusion Matrix
2) Classification Report
3) Accuracy Score

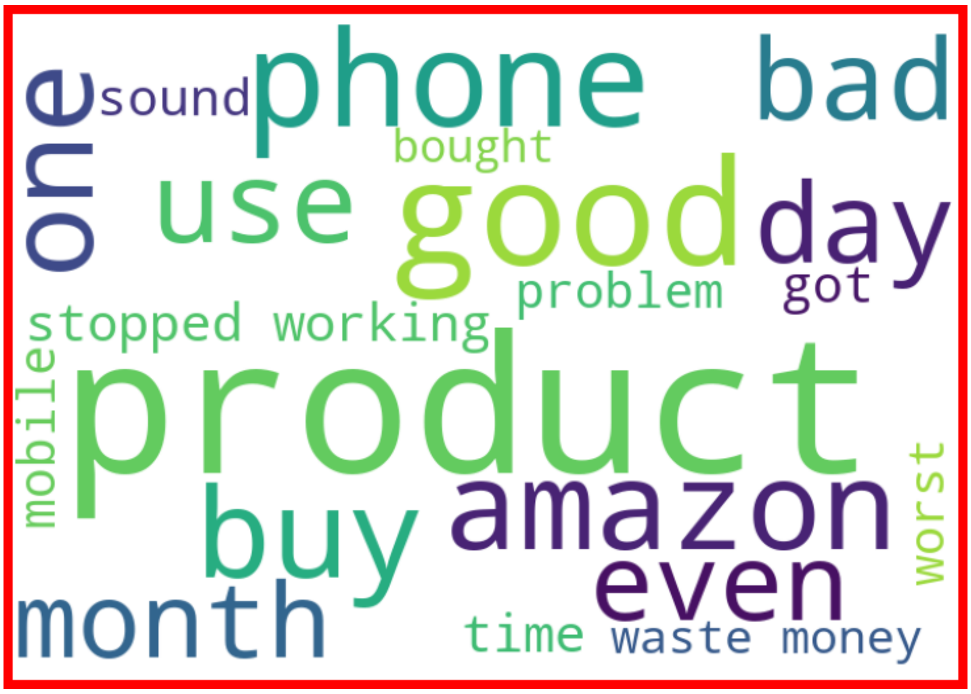
a) **Accuracy Score:-** Accuracy classification score. In multilabel classification, this function computes subset accuracy: the set of labels predicted for **a sample must exactly match** the corresponding set of labels in y_true. Ground truth (correct) labels. Predicted labels, as returned by a classifier.

b) **Classificafication Report:**- A classification report is a performance evaluation metric in machine learning. It is used to show the precision, recall, F1 Score, and support of your trained classification model.

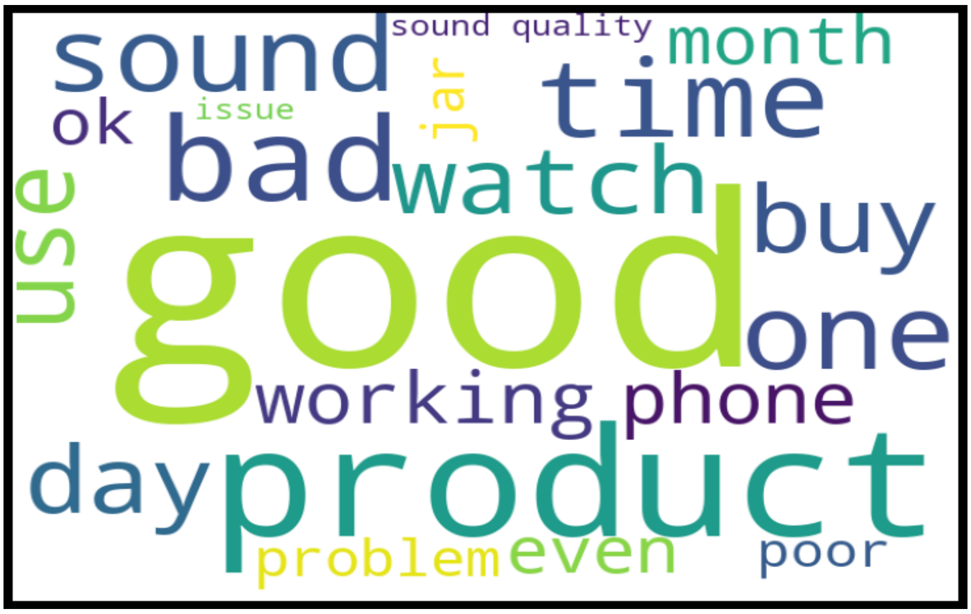
c) **Confusion Matrix:-** The confusion matrix is a matrix used to determine the performance of the classification models for a given set of test data. It can only be determined if the true values for test data are known. The matrix itself can be easily understood, but the related terminologies may be confusing.

- **Visualizations**

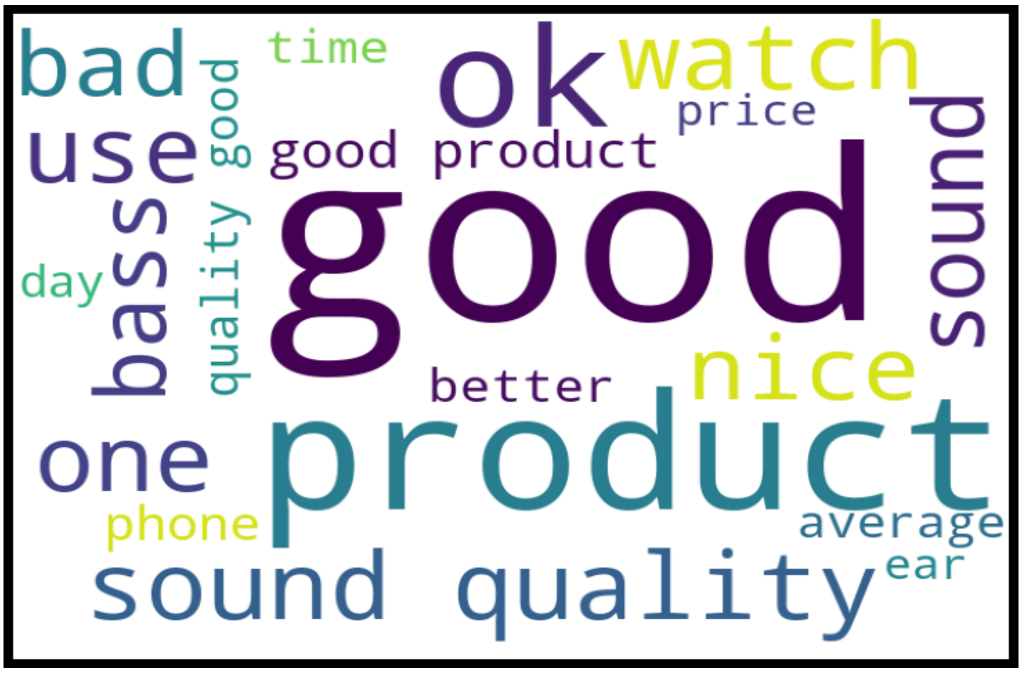
➔ Visualization for 1 Star Reviews:



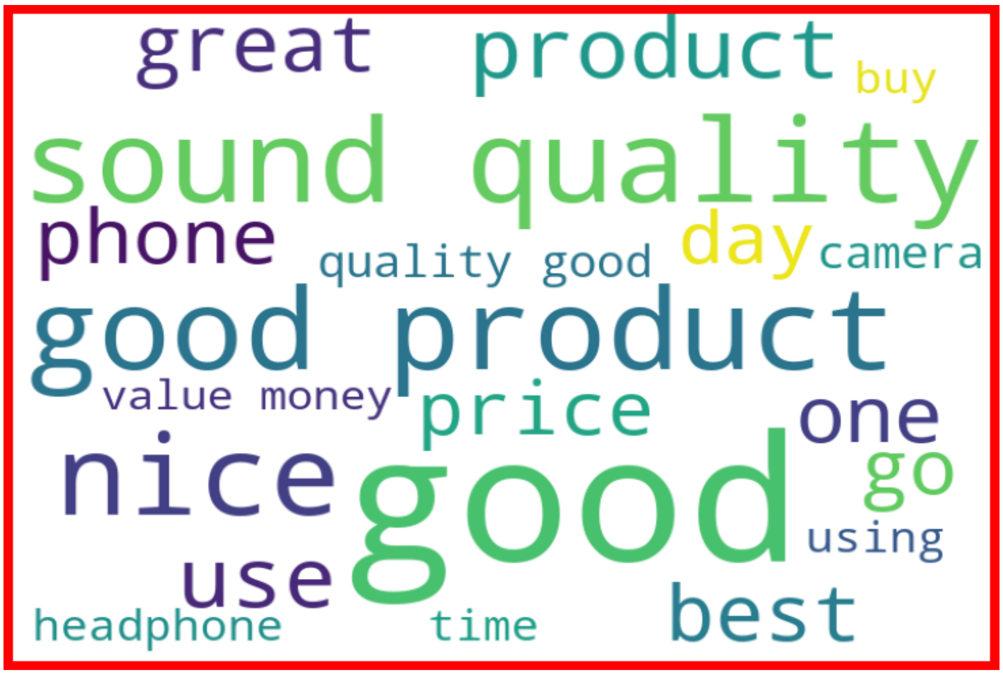
➔ Visualization for 2 Star Reviews:



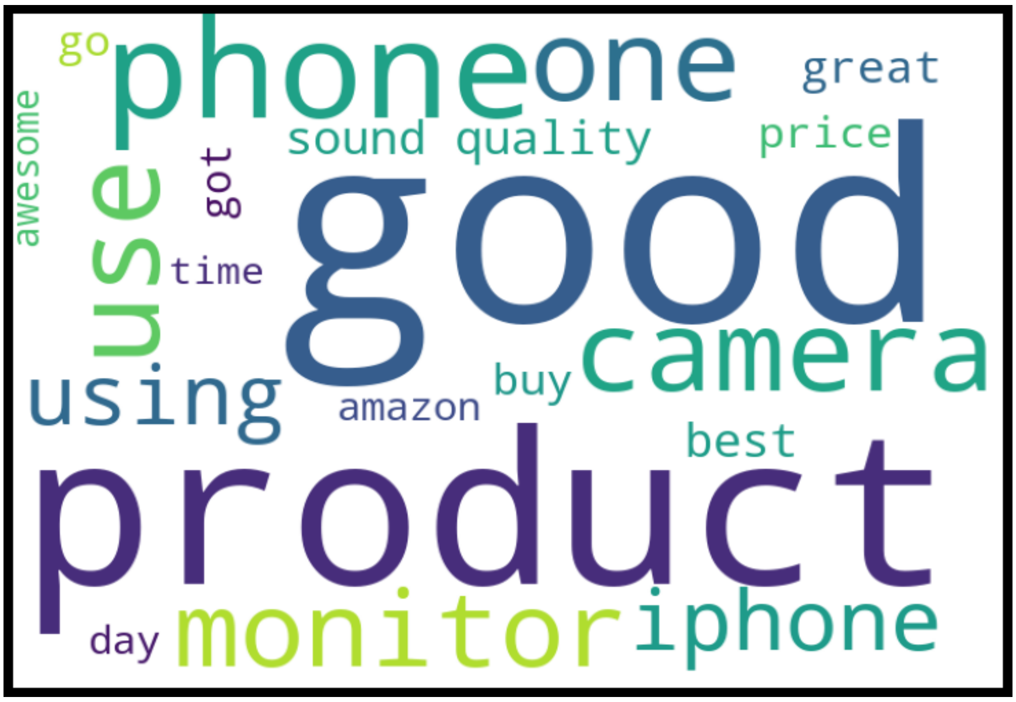
➔ Visualization for 3 Star Reviews:



➔ Visualization for 4 Star reviews:



➔ Visualization for 5 Star reviews:



- Interpretation of the Results

  ➔ From above visualization we can see the words that are in the ratings for a particular rating in a dataset.
  ➔ For 1 star reviews we can see bad reviews
  ➔ For 2 Star reviews we can see some average reviews
  ➔ For 3 Star reviews we can see some good reviews from this rating we can see the good reviews for the product
  ➔ For 4 star reviews we can see Very Good reviews for the Product
  ➔ For 5 Star reviews we can see Excellent reviews for the product and this is the highest review for the product one can get it on a ecommerce website.

CONCLUSION

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

  ➔ From above problem I observe that we need huge data for doing this type of task as Natural Language has many words in present and for building a good model we need huge data so that our model can learn well and predict it with a good accuracy.

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

  ➔ I learnt many things from the problem such as I thought that all the reviews will only have English words but I got many other languages in the reviews such as hindi, telegu, malyalam and Chinese so for removing those I used stop word the I used Regular Expression to remove other words.

- **<u>Limitations of this work and Scope for Future Work</u>**

  - ➔ There are some limitations in this work as we have only trained our model on 22000 rows and 11000 column and there are many other English word which model has not trained on so if that word appears in that reviews the model may mis interpret and give wrong reviews .
  - ➔ As we have trained the model only on English language so if the comments are in Hindi or any other language then also the model will mis interpret and gives wrong results.
  - ➔ To Improve the model we can add more columns and train them on those columns and if we want to make a model which has Multi language reviews then we must also train out model with those language reviews and we are good to go.