Ratings Prediction Project Presentation

Submitted To: Khusboo Garg (SME of Internship Batch no-28)

Submitted By: Abhishek Behera
Internship Batch no-28

INTRODUCTION

- This is a Machine Learning Project performed on customer reviews. Reviews are processed using common NLP techniques.
- Millions of people use Amazon and Flipkart to buy products. For every product, people can rate and write a review. If a product is good, it gets a positive review and gets a higher star rating, similarly, if a product is bad, it gets a negative review and lower star rating. My aim in this project is to predict star rating automatically based on the product review.
- The range of star rating is 1 to 5. That means if the product review is negative, then it will get low star rating (possibly 1 or 2), if the product is average then it will get medium star rating (possibly 3), and if the product is good, then it will get higher star rating (possibly 4 or 5).
- This task is similar to Sentiment Analysis, but instead of predicting the positive and negative sentiment (sometimes neutral also), here we need to predict the rating.

PROBLEM STATEMENT

- 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, Flipkart etc.

- There are two main methods to approach this problem. The first one is based on review text content analysis and uses the principles of natural language process (the NLP method). This method lacks the insights that can be drawn from the relationship between costumers and items. The second one is based on recommender systems, specifically on collaborative filtering and focuses on the reviewer’s point of view.

DATA COLLECTION PHASE

- You have to scrape at least 20000 rows of data. You can scrape more data as well, it's up to you. More the data better the model. In this section you need to scrape the reviews of different laptops, Phones, Headphones, smart watches, Professional Cameras, Printers, monitors, home theatre, router from different e-commerce websites.

- Basically, we need these columns:

  1) reviews of the product.

  2) rating of the product.

- Fetch an equal number of reviews for each rating, for example if you are fetching 10000 reviews then all ratings 1,2,3,4,5 should be 2000. It will balance our data set. Convert all the ratings to their round number as there are only 5 options for rating i.e., 1,2,3,4,5. If a rating is 4.5 convert it 5.

MODEL BUILDING PHASE

- After collecting the data, you need to build a machine learning model. Before model building do all data pre-processing steps involving NLP. Try different models with different hyper parameters and select the best model. Follow the complete life cycle of data science. Include all the steps mentioned below:

1. Data Cleaning

2. Exploratory Data Analysis and Visualization

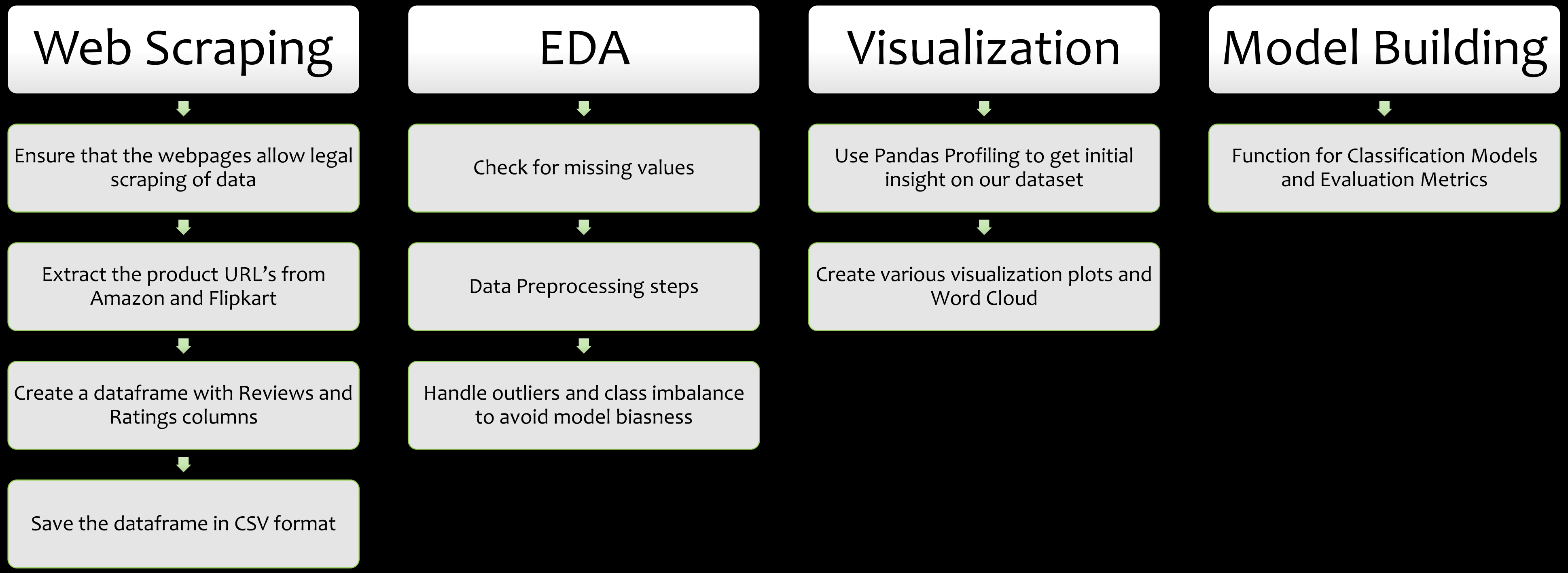
3. Data Pre-processing

4. Model Building

5. Model Evaluation

6. Selecting the Best classification model

PROJECT FLOW



Web Scraping

Ensure that the webpages allow legal scraping of data

Extract the product URL’s from Amazon and Flipkart

Create a dataframe with Reviews and Ratings columns

Save the dataframe in CSV format

EDA

Check for missing values

Data Preprocessing steps

Handle outliers and class imbalance to avoid model biasness

Visualization

Use Pandas Profiling to get initial insight on our dataset

Create various visualization plots and Word Cloud

Model Building

Function for Classification Models and Evaluation Metrics

HARDWARE AND SOFTWARE USED

- Hardware technology being used.

RAM	: 8 GB
CPU	: AMD Ryzen 5 3550H with Radeon Vega Mobile Gfx 2.10 GHz
GPU	: AMD Radeon ™ Vega 8 Graphics and NVIDIA GeForce GTX 1650 Ti

- Software technology being used.

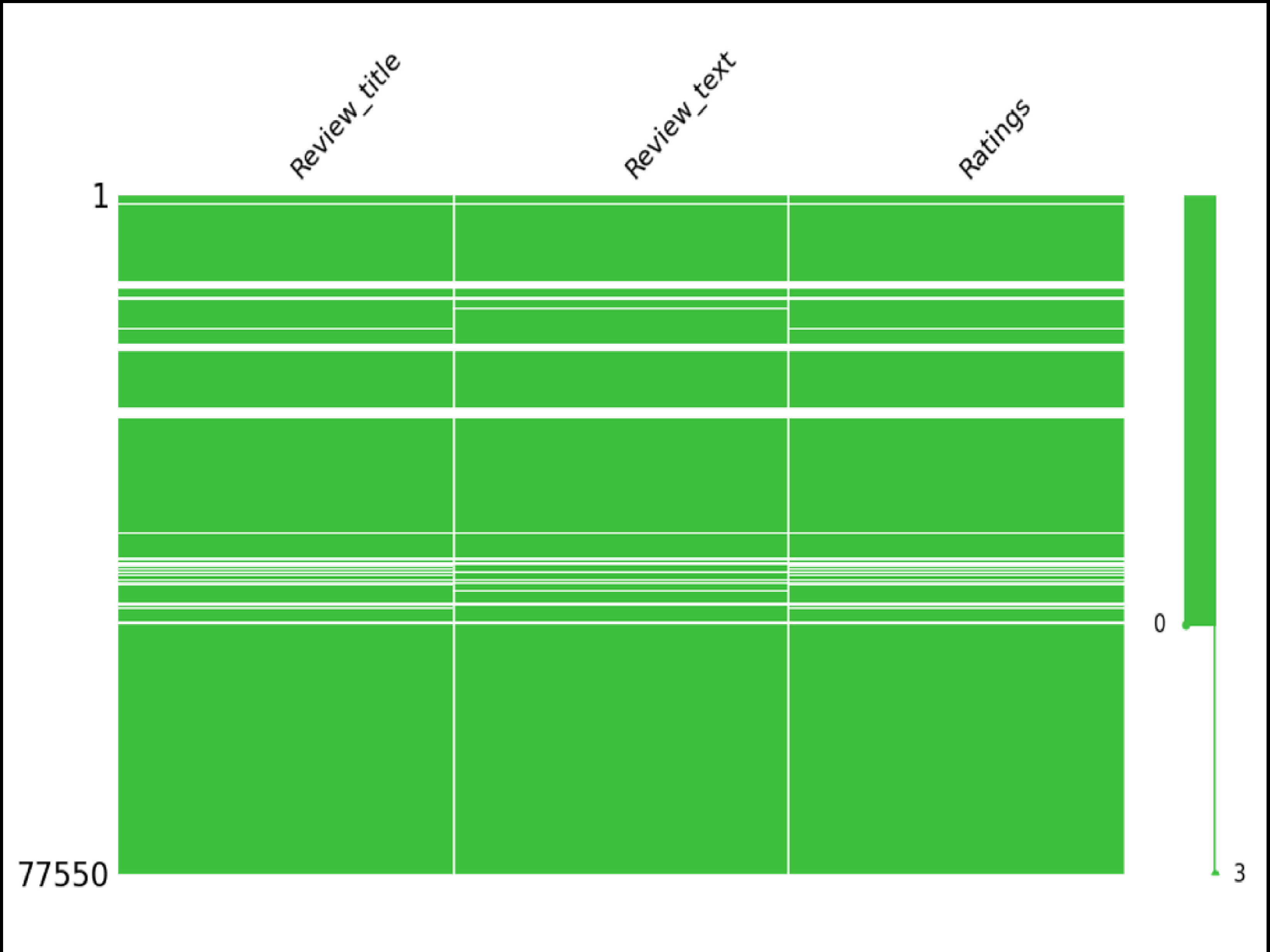
Programming language	: Python
Distribution	: Anaconda Navigator
Browser based language shell	: Jupyter Notebook

- Libraries/Packages specifically being used.

Pandas, NumPy, matplotlib, seaborn, scikit-learn, pandas-profiling, missingno, NLTK

DATA PREPROCESSING

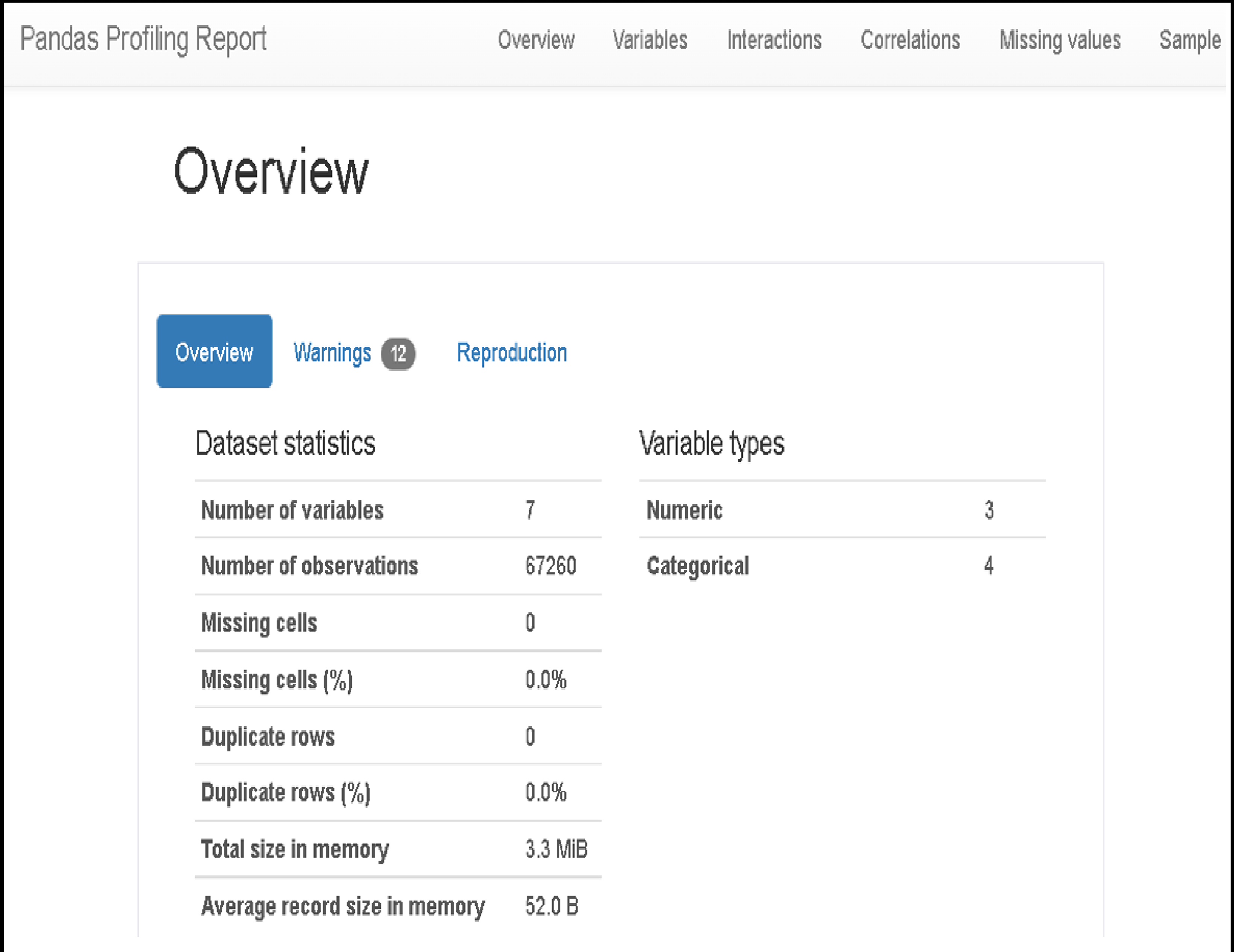
- Importing the necessary libraries/dependencies
- Checking dataset dimensions and null value details
- Taking a look at various label categories using the Unique method
- Performing data cleaning and then visualization steps
- Making Word Clouds for loud words in each label class
- Handling the class imbalance issue manually and fixing it
- Converting text into vectors using the TF-IDF Vectorizer
- Splitting the dataset into train and test to build classification models
- Evaluating the classification models with necessary metrics



MISSING VALUES

I used the missingno matrix feature to get a visual on all the NaN values present in our dataset and then decided to drop them all so that we were left with meaningful information.

PANDAS PROFILING



Overview

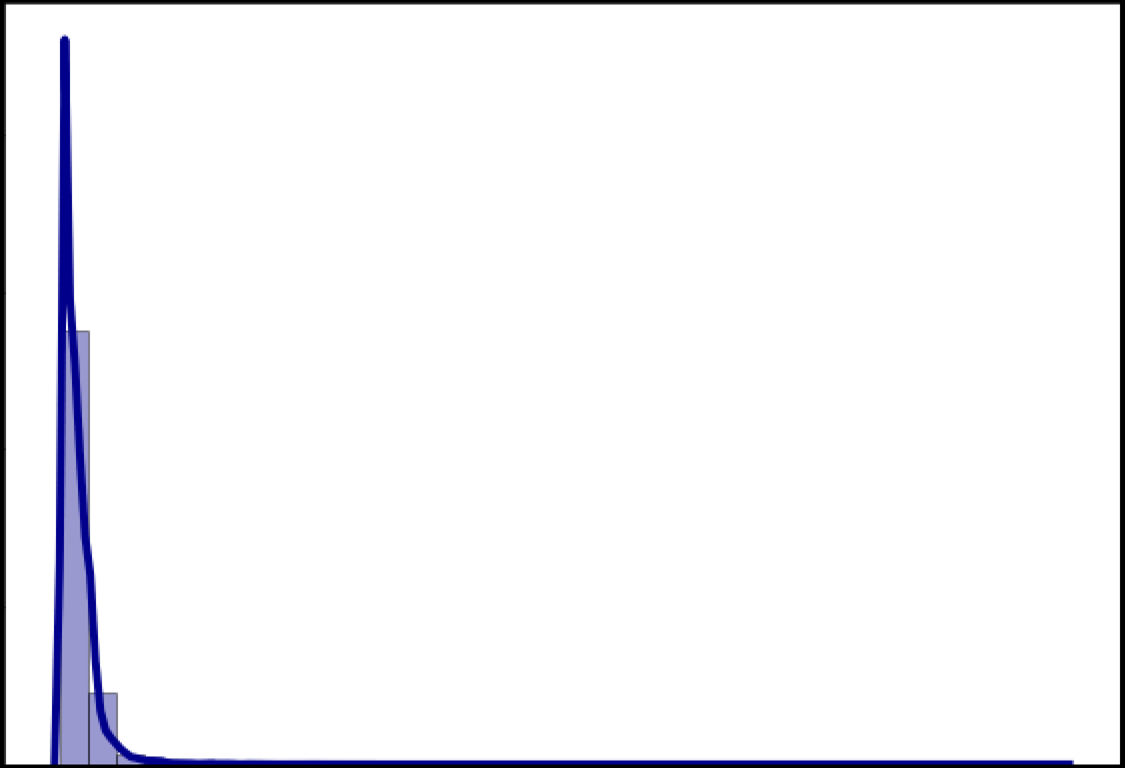
Dataset statistics

Number of variables	7
Number of observations	67260
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	3.3 MiB
Average record size in memory	52.0 B

Variable types

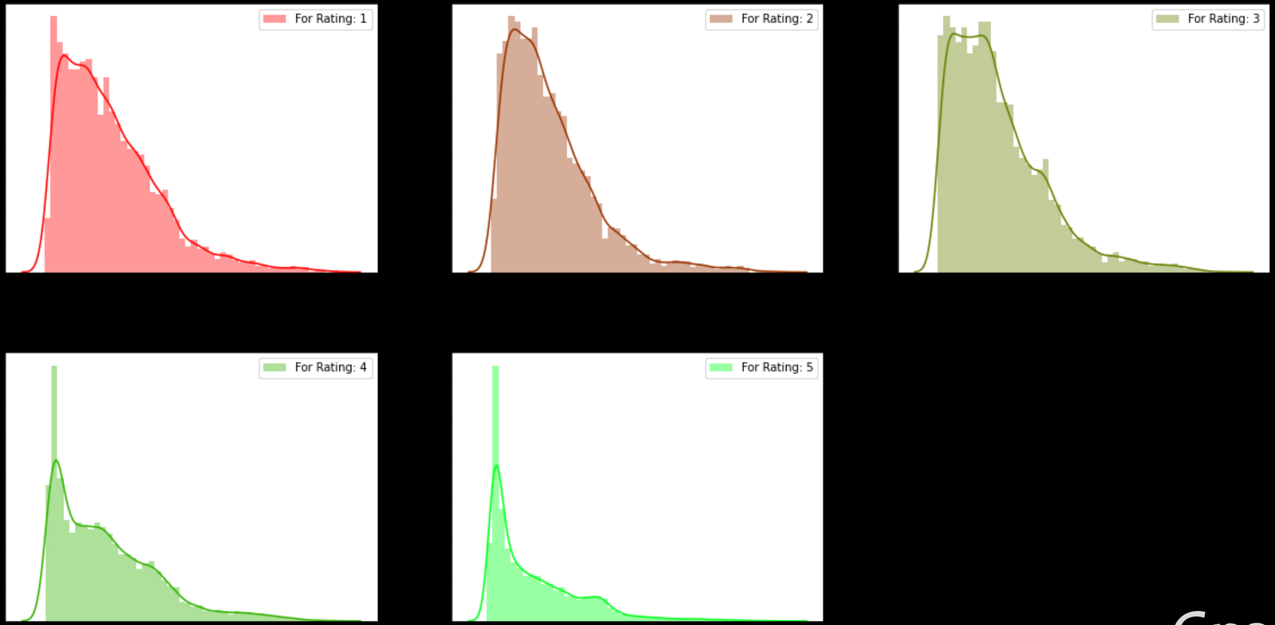
Numeric	3
Categorical	4

I used the pandas-profiling feature to get an insight on the initial dataset details and check out the application of all the data preprocessing steps on it.



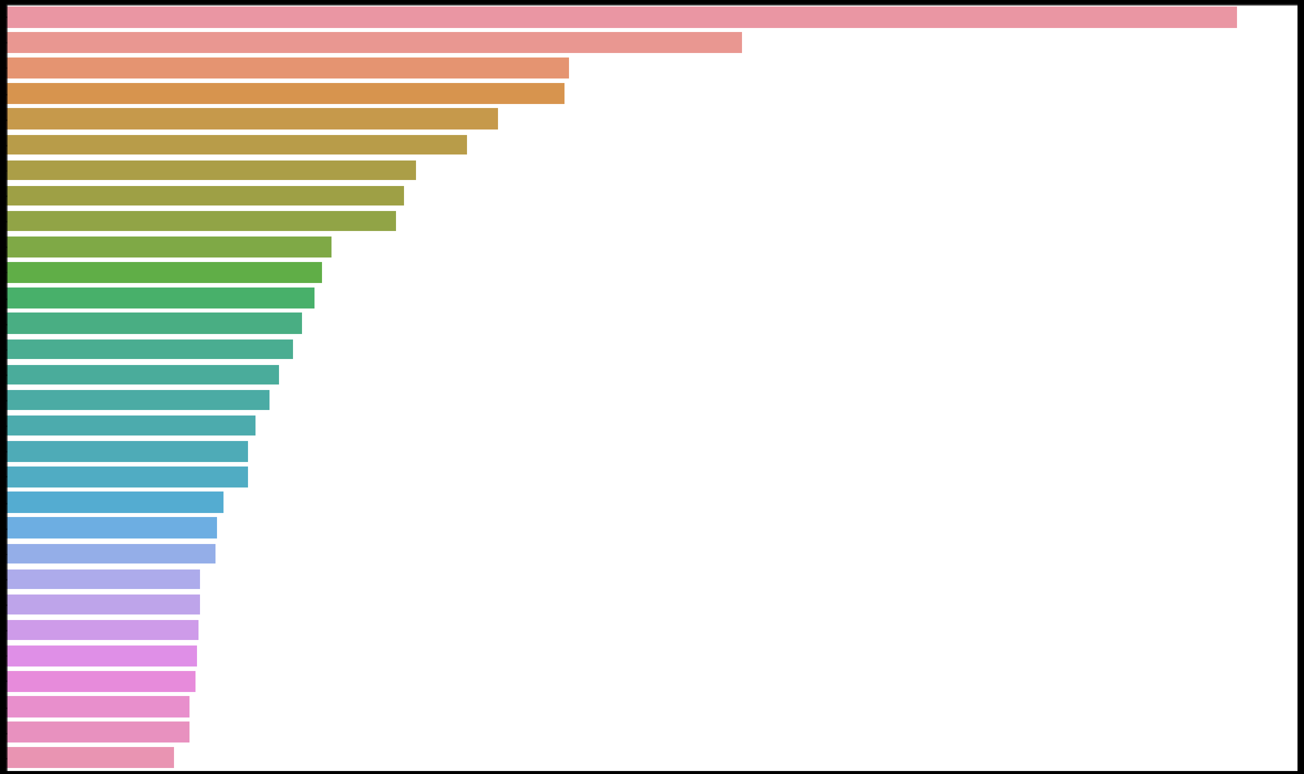
WORD AND CHARACTER COUNT

Created the histogram + distribution plots for Word Counts and Character Counts before and after cleaning the text data. We basically removed all the stop words, punctuations, smiley, special characters, white spaces etc.



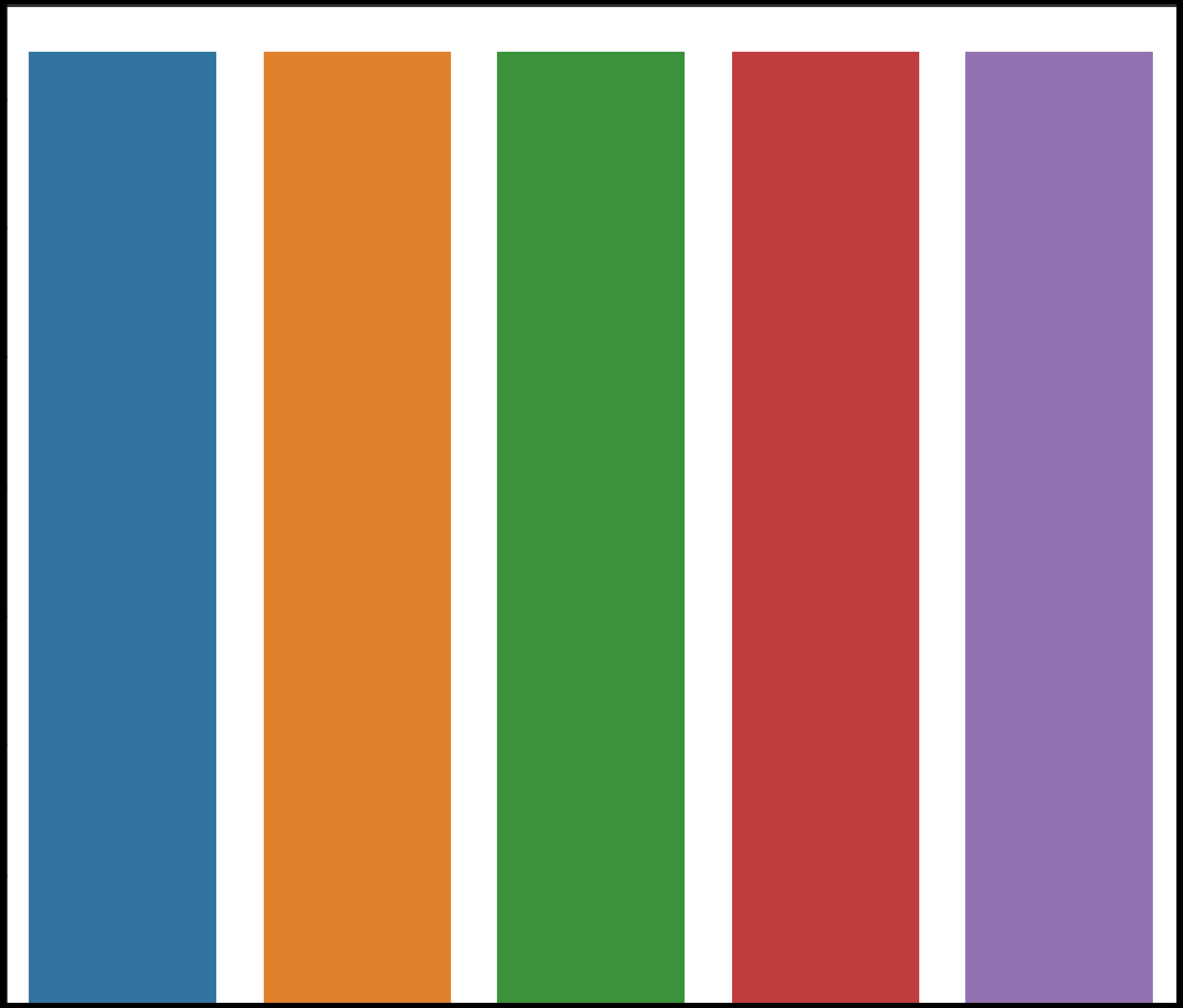
RATINGS PLOT

Created the histogram + distribution plots for our target label and observed each and every rating class for word counts as well as their character counts.



BAR PLOTS

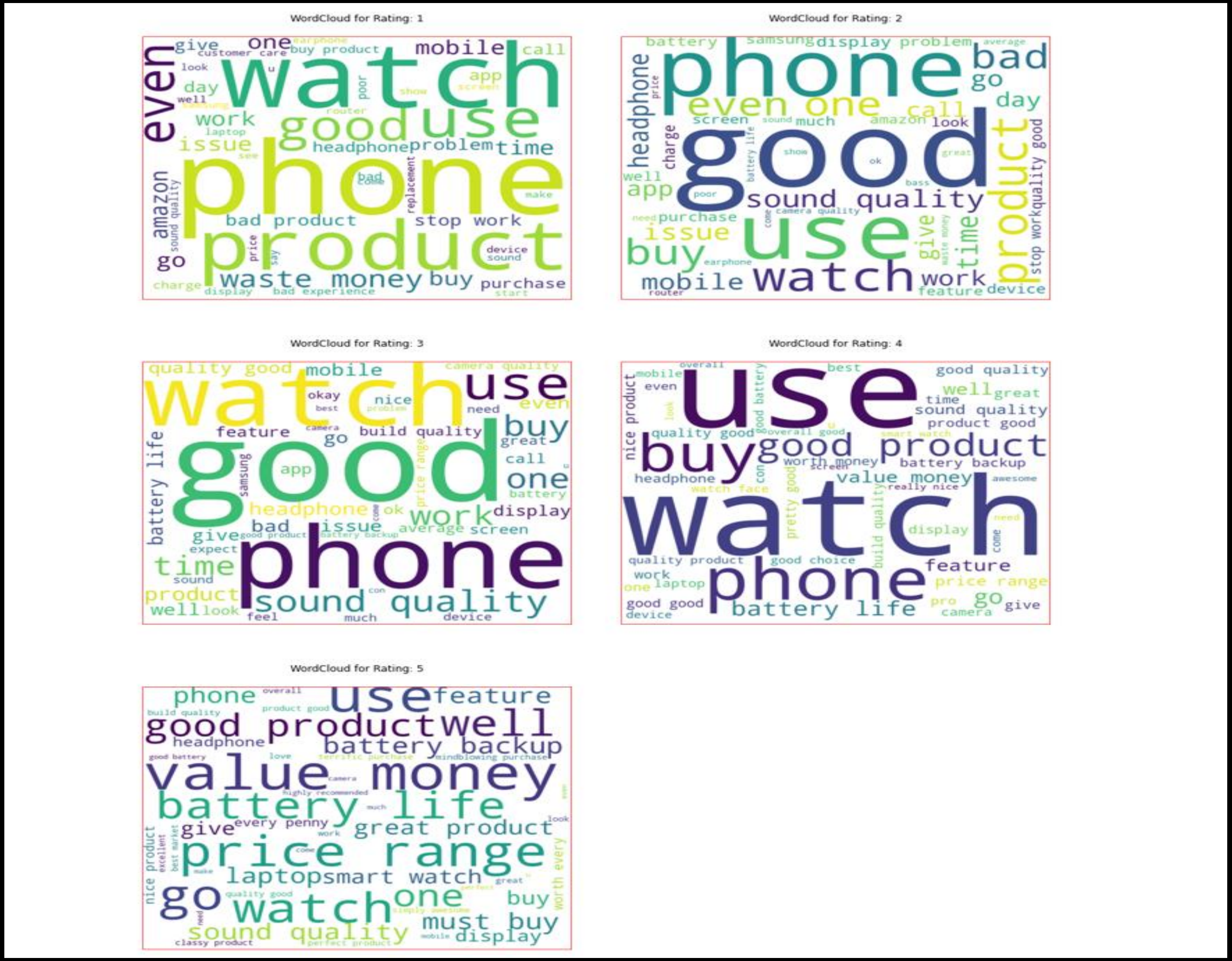
Generated these bar plots for most frequently used words in review summary and least or rarely used words in a review summary by any customer in our dataset.



Count Plots

Generated these count plots before and after handling the data imbalance concern where we notice that the dataframe consisted of different number of rating reviews that needed to be equalized.

WORD CLOUD



WordCloud for Rating: 1

WordCloud for Rating: 2

WordCloud for Rating: 3

WordCloud for Rating: 4

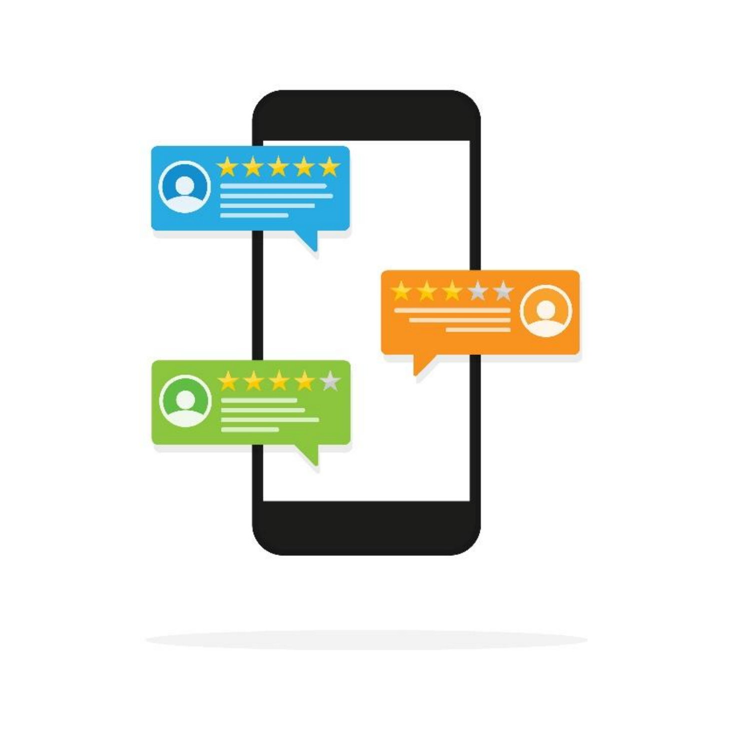
WordCloud for Rating: 5

Word Cloud as the name suggests is a cloud of words. It is a visualization technique for text data wherein each word is picturized with its importance in the context or its frequency.

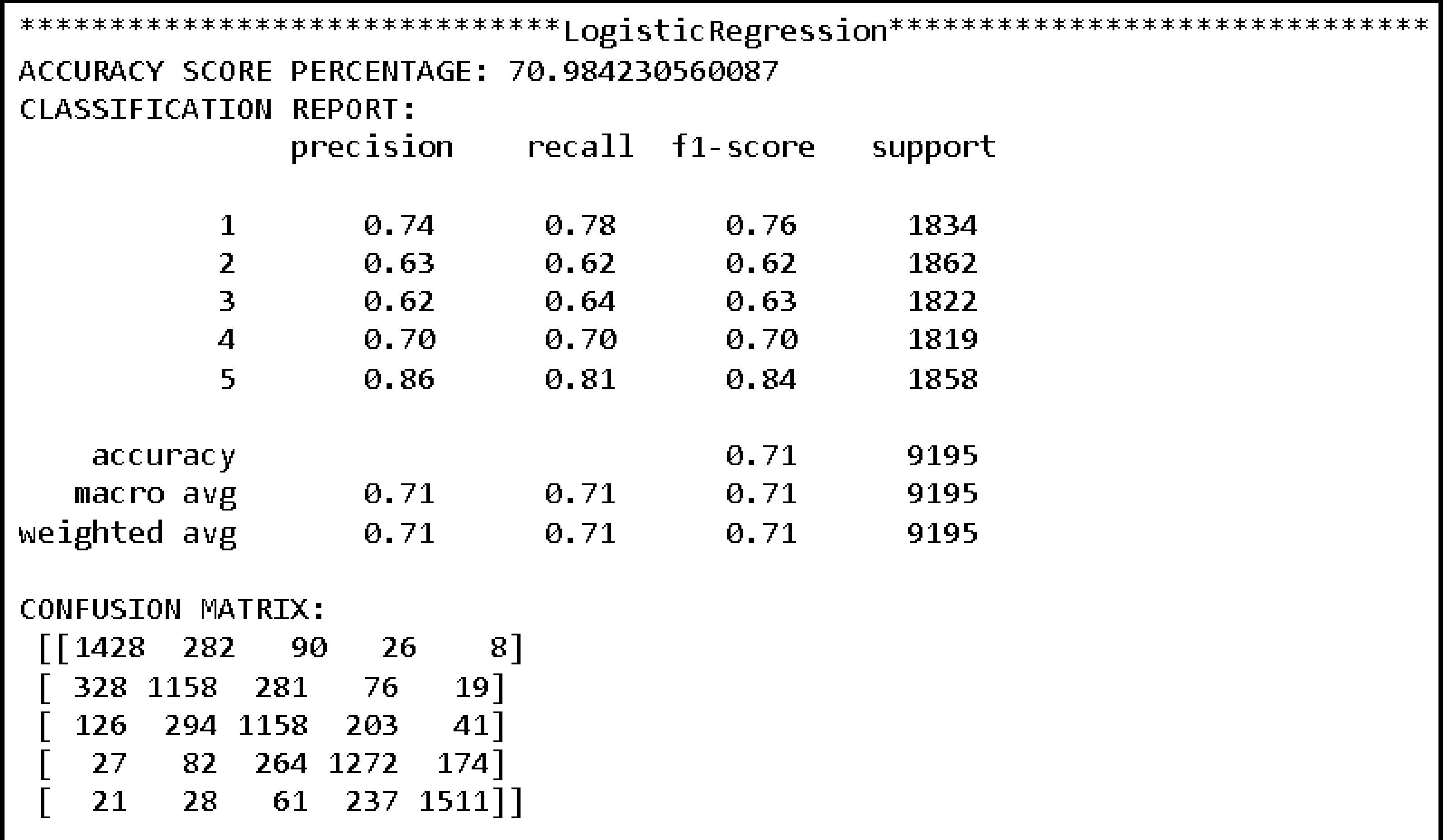
MODEL DEVELOPMENT ALGORITHMS

The complete list of algorithms that were used in training and testing the classification model are listed below:

1. Logistic Regression
2. Linear Support Vector Classifier
3. Random Forest Classifier
4. Bernoulli Naïve Bayes
5. Multinomial Naïve Bayes
6. Stochastic Gradient Descent Classifier
7. LGBM Classifier
8. XGB Classifier

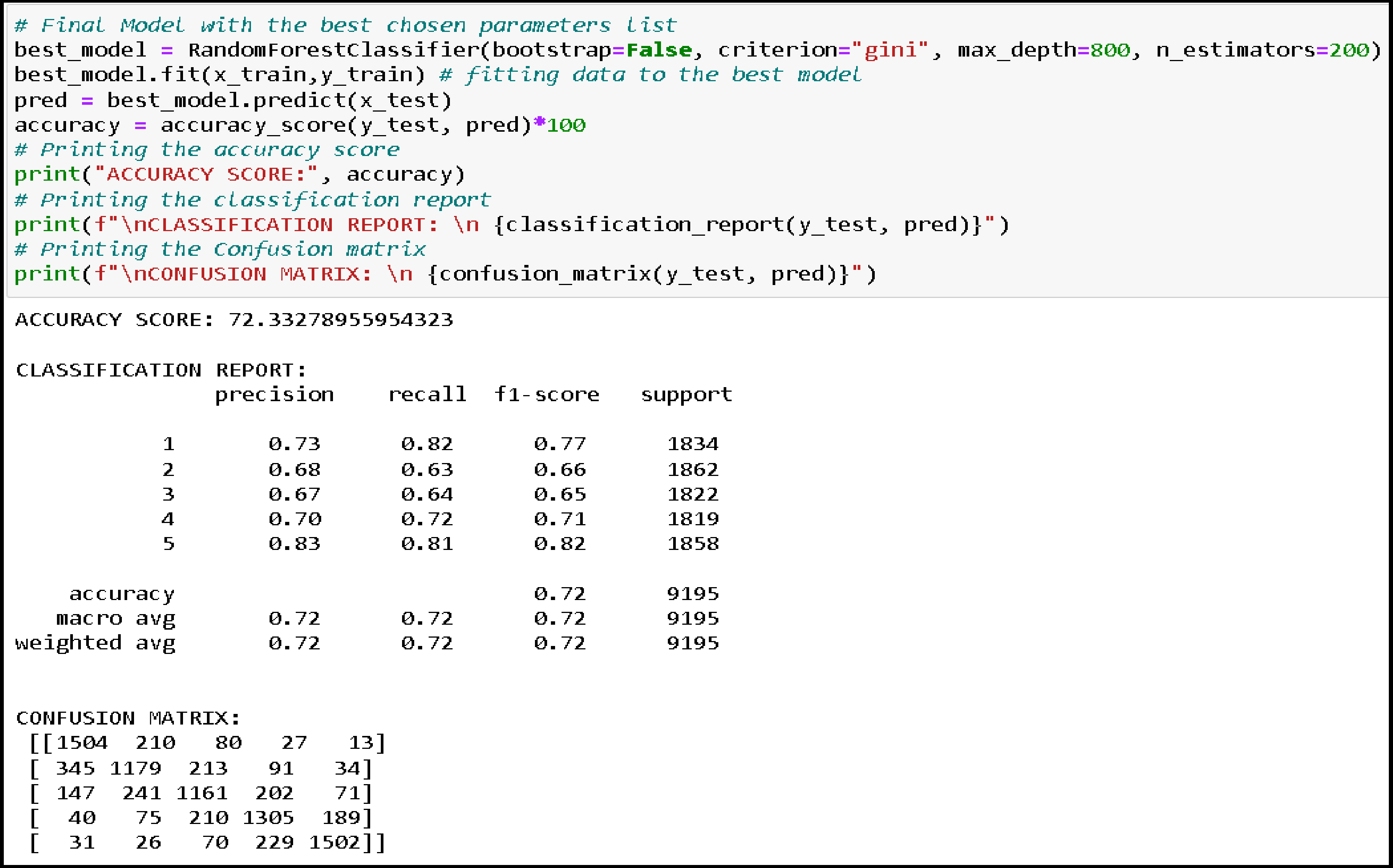


MODEL CREATION AND EVALUATION



```
******************************LogisticRegression******************************
ACCURACY SCORE PERCENTAGE: 70.984230560087
CLASSIFICATION REPORT:
               precision    recall  f1-score   support

           1       0.74      0.78      0.76      1834
           2       0.63      0.62      0.62      1862
           3       0.62      0.64      0.63      1822
           4       0.70      0.70      0.70      1819
           5       0.86      0.81      0.84      1858

    accuracy                           0.71      9195
   macro avg       0.71      0.71      0.71      9195
weighted avg       0.71      0.71      0.71      9195

CONFUSION MATRIX:
 [[1428  282   90   26    8]
 [ 328 1158  281   76   19]
 [ 126  294 1158  203   41]
 [  27   82  264 1272  174]
 [  21   28   61  237 1511]]
```

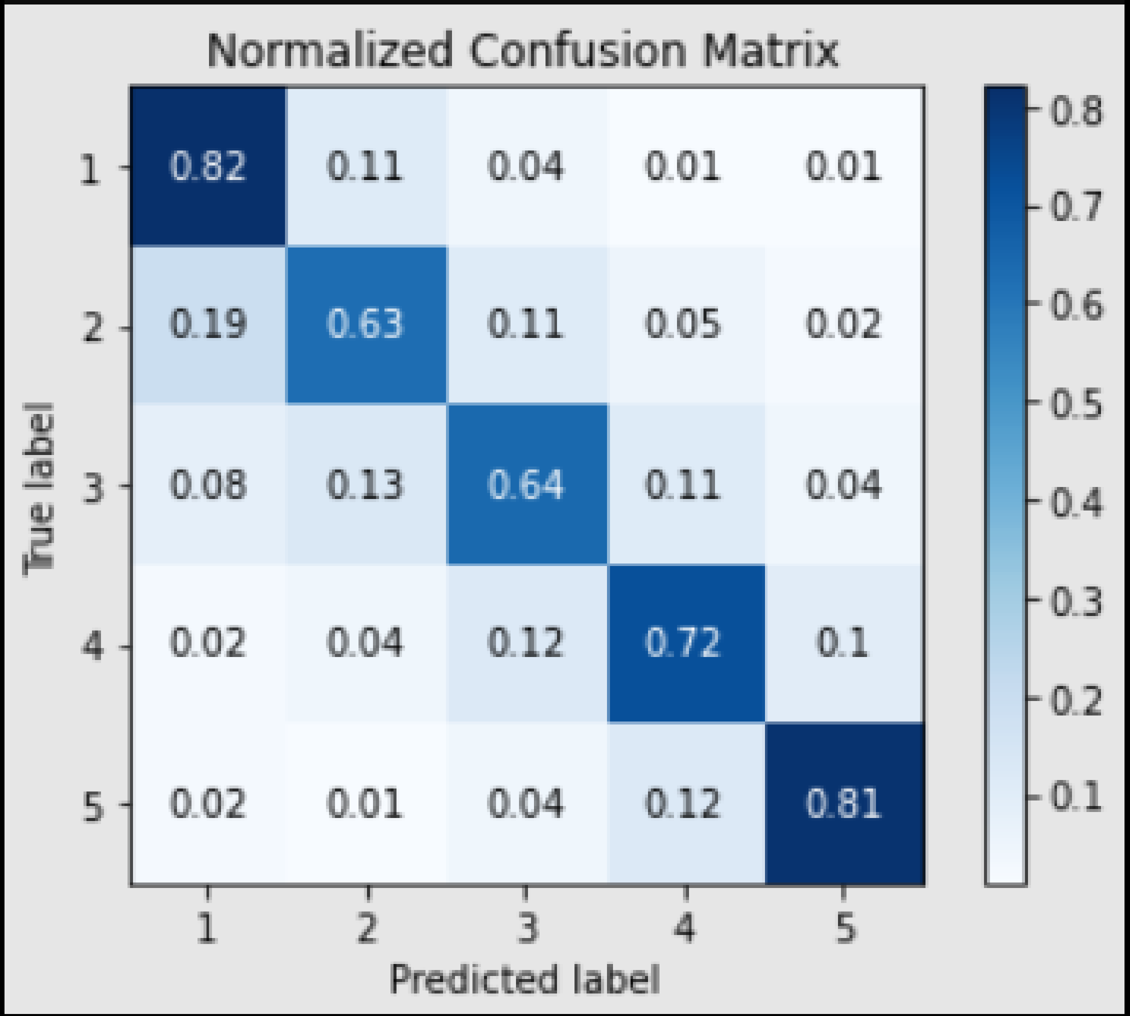
FINAL MODEL



```
# Final Model with the best chosen parameters list
best_model = RandomForestClassifier(bootstrap=False, criterion="gini", max_depth=800, n_estimators=200)
best_model.fit(x_train,y_train) # fitting data to the best model
pred = best_model.predict(x_test)
accuracy = accuracy_score(y_test, pred)*100
# Printing the accuracy score
print("ACCURACY SCORE:", accuracy)
# Printing the classification report
print(f"\nCLASSIFICATION REPORT: \n {classification_report(y_test, pred)}")
# Printing the Confusion matrix
print(f"\nCONFUSION MATRIX: \n {confusion_matrix(y_test, pred)}")
```

```
ACCURACY SCORE: 72.33278955954323

CLASSIFICATION REPORT: 
               precision    recall  f1-score   support

           1       0.73      0.82      0.77      1834
           2       0.68      0.63      0.66      1862
           3       0.67      0.64      0.65      1822
           4       0.70      0.72      0.71      1819
           5       0.83      0.81      0.82      1858

    accuracy                           0.72      9195
   macro avg       0.72      0.72      0.72      9195
weighted avg       0.72      0.72      0.72      9195


CONFUSION MATRIX: 
 [[1504  210   80   27   13]
 [ 345 1179  213   91   34]
 [ 147  241 1161  202   71]
 [  40   75  210 1305  189]
 [  31   26   70  229 1502]]
```

NORMALIZED CONFUSION MATRIX



Normalized Confusion Matrix

True label \ Predicted label	1	2	3	4	5
1	0.82	0.11	0.04	0.01	0.01
2	0.19	0.63	0.11	0.05	0.02
3	0.08	0.13	0.64	0.11	0.04
4	0.02	0.04	0.12	0.72	0.1
5	0.02	0.01	0.04	0.12	0.81

CONCLUSION

- Key findings of the study: In this project I have collected data of reviews and ratings for different products from amazon.in and flipkart.com. Then I have done different text processing for reviews column and chose equal number of text from each rating class to eliminate problem of imbalance. By doing different EDA steps I have analyzed the text. We have checked frequently occurring words in our data as well as rarely occurring words. After all these steps I have built function to train and test different algorithms and using various evaluation metrics I have selected Random Forest Classifier as our final model. Finally by doing hyperparameter tuning we got optimum parameters for our final model. And finally we got improved accuracy score for our final model.

- Limitations of this work and scope for the future work: As we know the content of text in reviews is totally depends on the reviewer and they may rate differently which is totally depends on that particular person. So it is difficult to predict ratings based on the reviews with higher accuracies. Still we can improve our accuracy by fetching more data and by doing extensive hyperparameter tuning.

CONCLUSION

- Areas of improvement:

I. Less time complexity

II. More computational power can be given

III. More accurate reviews can be given

IV. Many more permutations and combinations in hyper parameter tuning can be used to obtain better parameter list

- Final Remarks: After applying the hyper parameter tuning the best accuracy score obtained was 72.33278955954323% which can be further improved by obtaining more data and working up through other parameter combinations.

- We were able to create a rating prediction model that can be used to identify rating details just by evaluating the comments posted by a customer.

THANK YOU