FLIP ROBO

Ratings Prediction Project

Submitted by:

AKSHAY SHAH

ACKNOWLEDGMENT

I express my sincere gratitude to Flip Robo Technologies for giving me the opportunity to work on this project on Malignant Comment Classifier using machine learning algorithms and NLTK suite of libraries and also, for providing me with the requisite datasets for training and testing prediction accuracies of the models.

INTRODUCTION

Business Problem Framing

A website has a forum for writing technical reviews of products and consists of repository of 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. An application to predict the rating by seeing the review is required to be built.

Therefore, a predictive model to accurately predict a user's rating based on input review is required to be made.

Conceptual Background of the Domain Problem

Predictive modelling, Classification algorithms are some of the machine learning techniques used along with the various libraries of the NLTK suite for Classification of comments. Using NLTK tools, the frequencies of malignant words occurring in textual data were estimated and given appropriate weightage, whilst filtering out words, and other noise which do not have any impact on the semantics of the comments and reducing the words to their base lemmas for efficient processing and accurate classification of the comments.

Review of Literature

A Research paper titled: "Review-Based Rating Prediction" by Tal Haddad was reviewed and studied to gain insights into the importance of contextual information of user sentiments in determining the rating of products, the role of natural language processing tools and techniques in identifying the user sentiments towards various products based on their reviews and ratings

It is learnt that Contextual information about a user’s opinion of a product can be explicit or implicit and can be inferred in different ways such as user score ratings and textual reviews.

Motivation for the Problem Undertaken

Ratings are an important metric in e-commerce application to determine a product’s quality, consumer demand, worth and profitability. The sentiment of a user towards a product is reflected in their rating score and their review of the product. This helps determine how the product is perceived by the consumers and in turn gives an idea about the acceptance of the product by the consumers. There is a strong positive correlation between the rating of a product and its consumer demand. Therefore, it is necessary to build a predictive model which can, with good accuracy predict what rating a user might give a particular product based on the user review. This helps understand user sentiment towards a product and determine the product’s worth and acceptance by consumers.

Analytical Problem Framing

Mathematical/ Analytical Modelling of the Problem

Various Classification analysis techniques were used to build predictive models to understand the relationships that exist between user review and the corresponding user rating.

The user reviews are collected, processed and normalized. Based on the context of the reviews on various items, with similar ratings, prediction of the rating for a given review can be made based on similar reviews which already have corresponding ratings.

In order to predict ratings for user revies, models such as Logistic regression, Random Forest Classifier Boost Classifier, Extreme Gradient Boost Classifier, Multinomial Naïve Bayes Classifier, Complement Naïve Bayes Classifier and Passive Aggressive Classifier were used.

Data Sources and their formats

The Dataset was compiled by scraping User review and rating Data for various products from https://amazon.in and https://www.flipkart.com/

The data was converted into a Pandas Data frame under variousComment and Ratings columns and saved as a .csv file and excel file.

Dataset Description

```
df = pd.read_csv("Rating_Predicts.csv")
```

```
df.head()
```

	Unnamed: 0	Ratings	Review
0	0	5.0	This is the best laptop in this range.I reciev...
1	1	5.0	Good product as used of now.... Everything is ...
2	2	5.0	AWESOME LAPTOP. It supports many high spec gam...
3	3	4.0	For the peoples who R going to buy R they buye...
4	4	5.0	It's good gameing laptop in this price\nDispla...

Exploratory Data Analysis (EDA):

```
df.shape
```

```
(21088, 3)
```

The columns are:

- Review: User review of a product.
- Ratings: Corresponding user rating score for a user review

Data Preprocessing Done

- Rows with null values were removed.
- Columns: Unnamed: 0(just a series of numbers) was dropped since it doesn't contribute to building a good model for predicting the target variable values.
- The train and test dataset contents were then converted into lowercase.
- Punctuations, unnecessary characters etc. were removed, currency symbols, phone numbers, web URLs, email addressesetc were replaced with single words
- Tokens that contributed nothing to semantics of the messages were removed as Stop words. Finally retained tokens were lemmatized using WordNetLemmatizer().
- The string lengths of original comments and the cleaned comments were then compared.

Data Inputs- Logic- Output Relationships

The comment tokens so vectorized using TfidVectorizer are input and the corresponding rating is predicted based on their context as output by classification models.

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

The comment content made available in Dataset is assumed to be written in English Language in the standard Greco-Roman script. This is so that the Stop word package and WordNetLemmatizer can beeffectively used.

Hardware and Software Requirements and Tools Used

Hardware Used:

- Processor: Intel i5 10th Generation
- Physical Memory: 8.0GB (3200MHz)
- GPU: Nvidia RTX 3060, 4GB DDR6 VRAM.

Software Used:

- Windows 11 Operating System

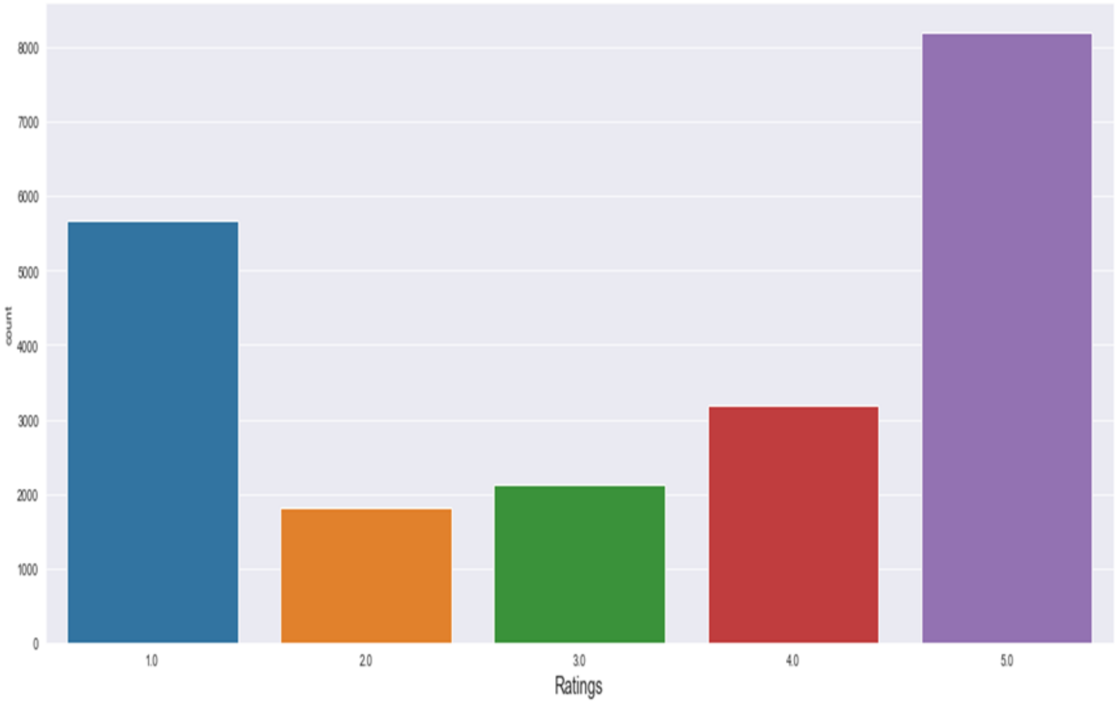
- Python Libraries used:
    - Pandas: For carrying out Data Analysis, Data Manipulation, Data Cleaning etc.
    - NumPy: For performing a variety of operations on thedatasets.
    - matplotlib. pyplot, Seaborn: For visualizing Data and various relationships between Feature and Label Columns
    - imblearn. oversampling: To employ SMOTE technique for balancing out the classes. Stats models: For performing statistical analysis
    - sklearn for Modelling Machine learning algorithms, Data Encoding, Evaluation metrics, Data Transformation, Data Scaling, Component analysis, Feature selection etc.
    - re, string: To perform regex operations
    - Word cloud: For Data Visualization
    - NLTK: To use various Natural Language Processing Tools.
    -

Exploratory Data Analysis

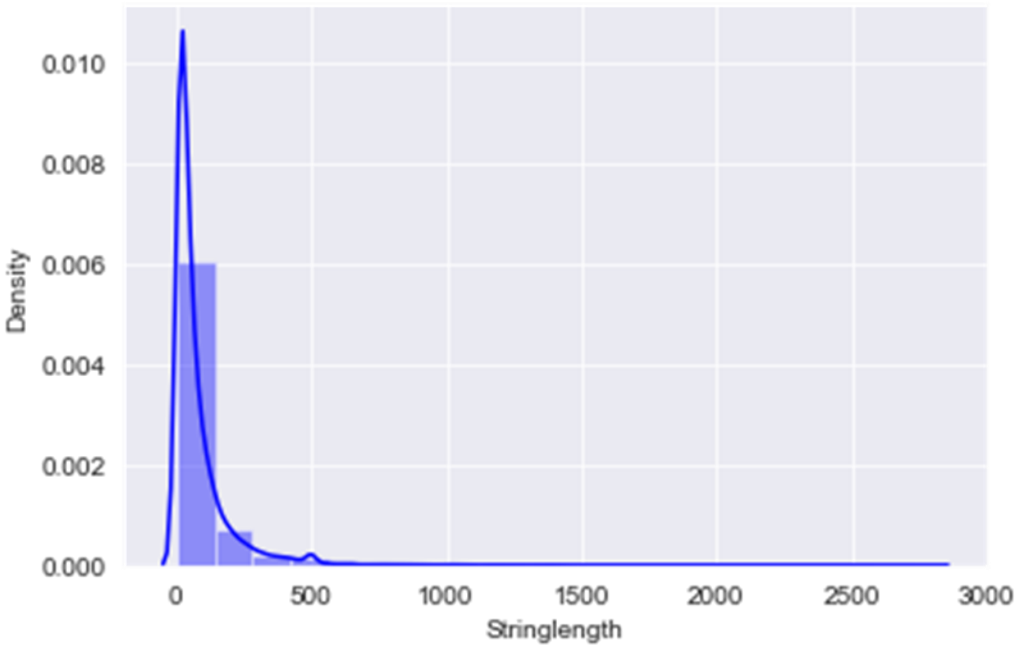
Visualizations

Barplots, Countplots, Distplots, WordClouds were used to visualize the data of all the columns and their relationships with Target variable.

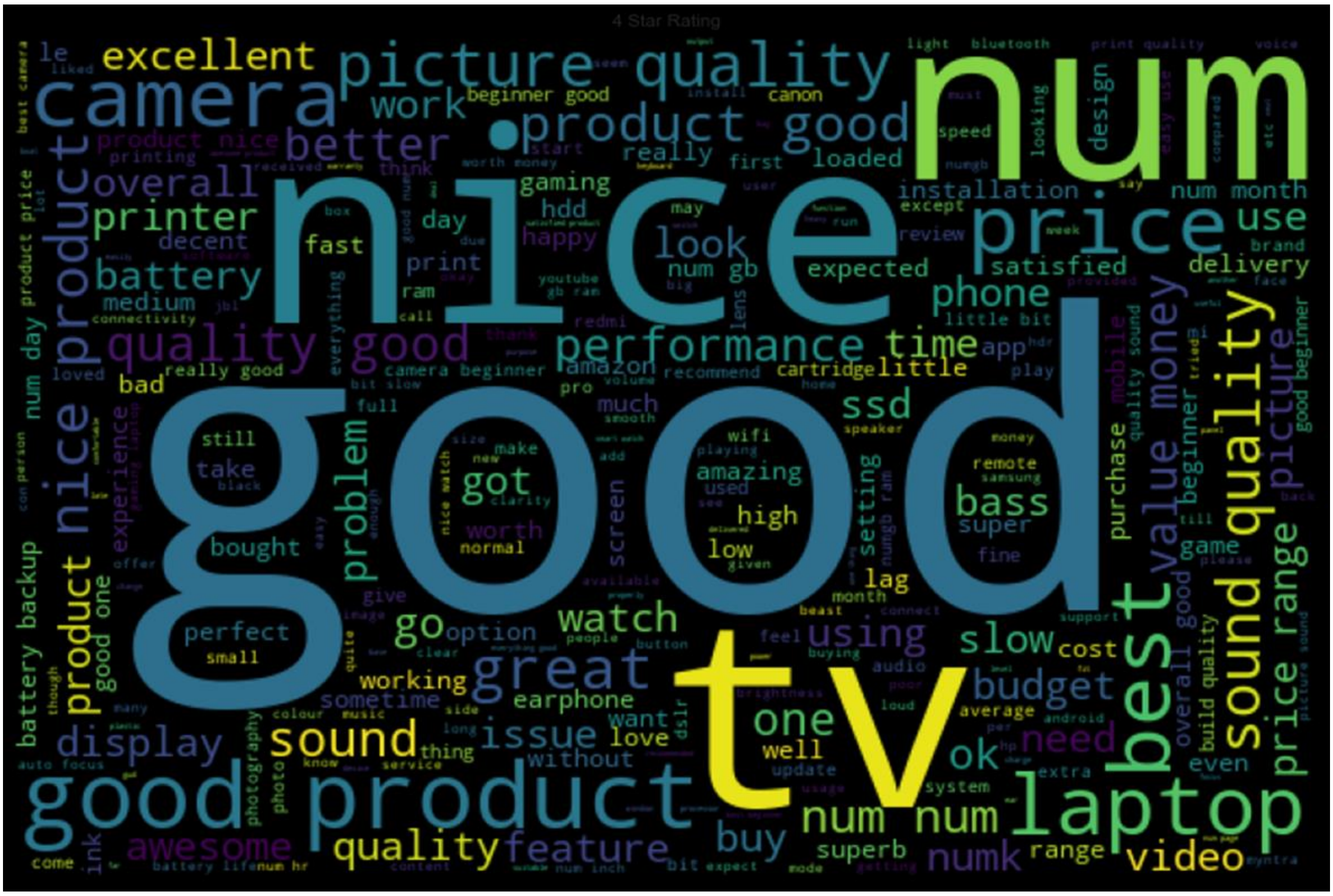
Analyzing the Target Variable



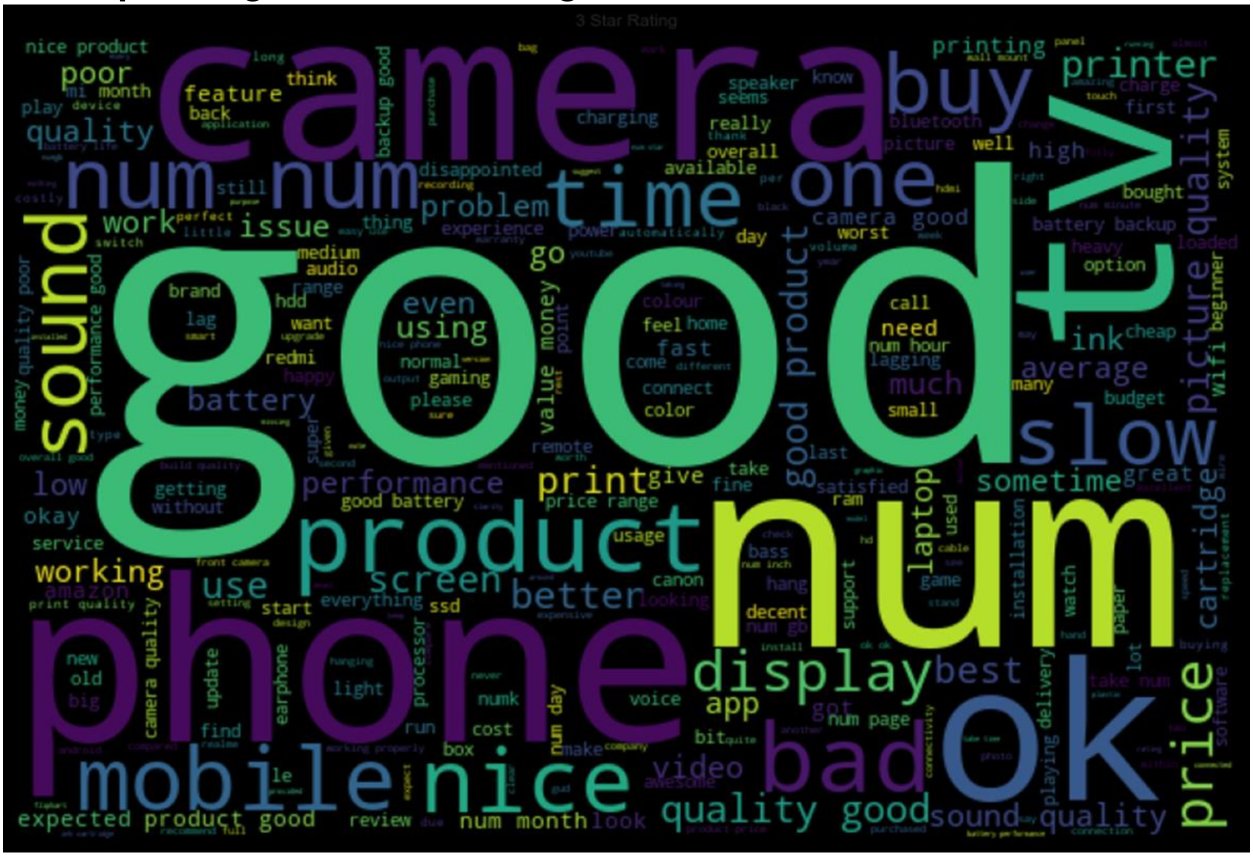
The rating classes 1.0-4.0 are balanced, the 5.0 class represents the highest number of reviews.

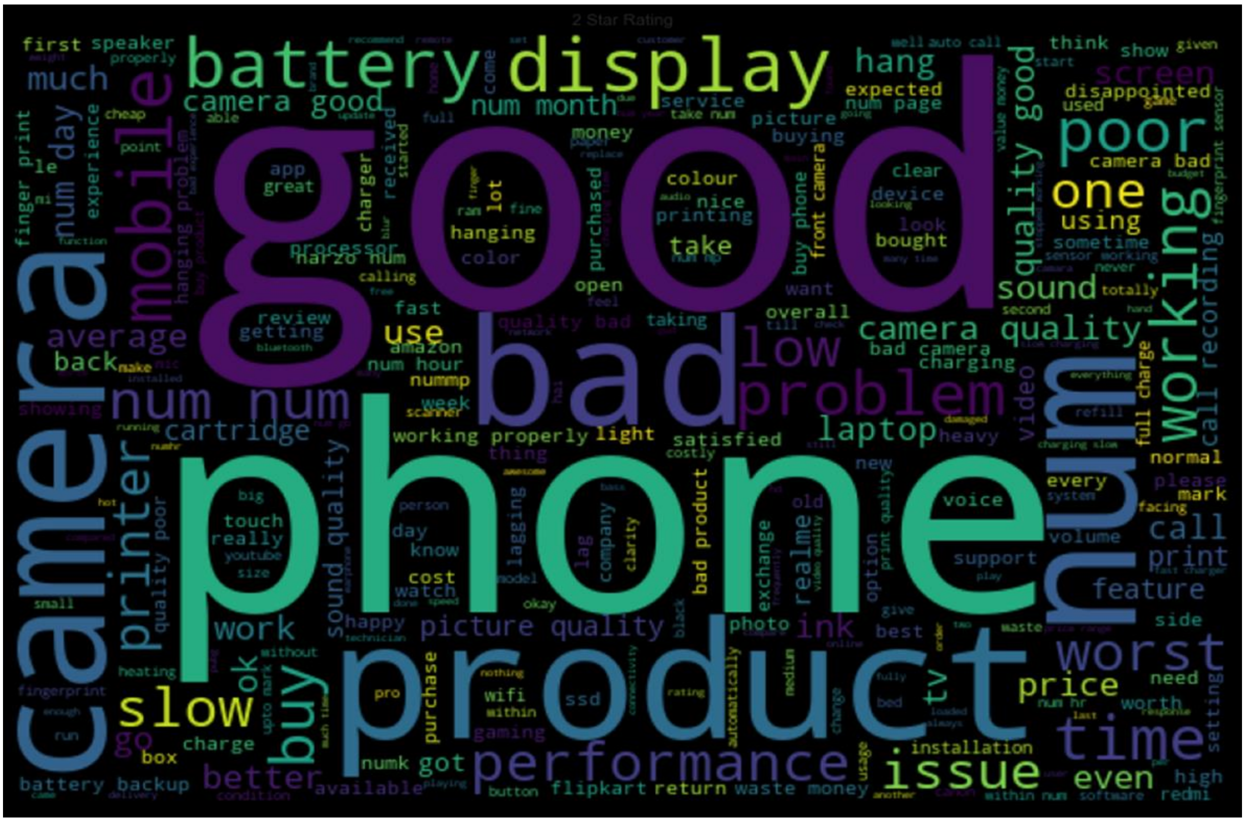


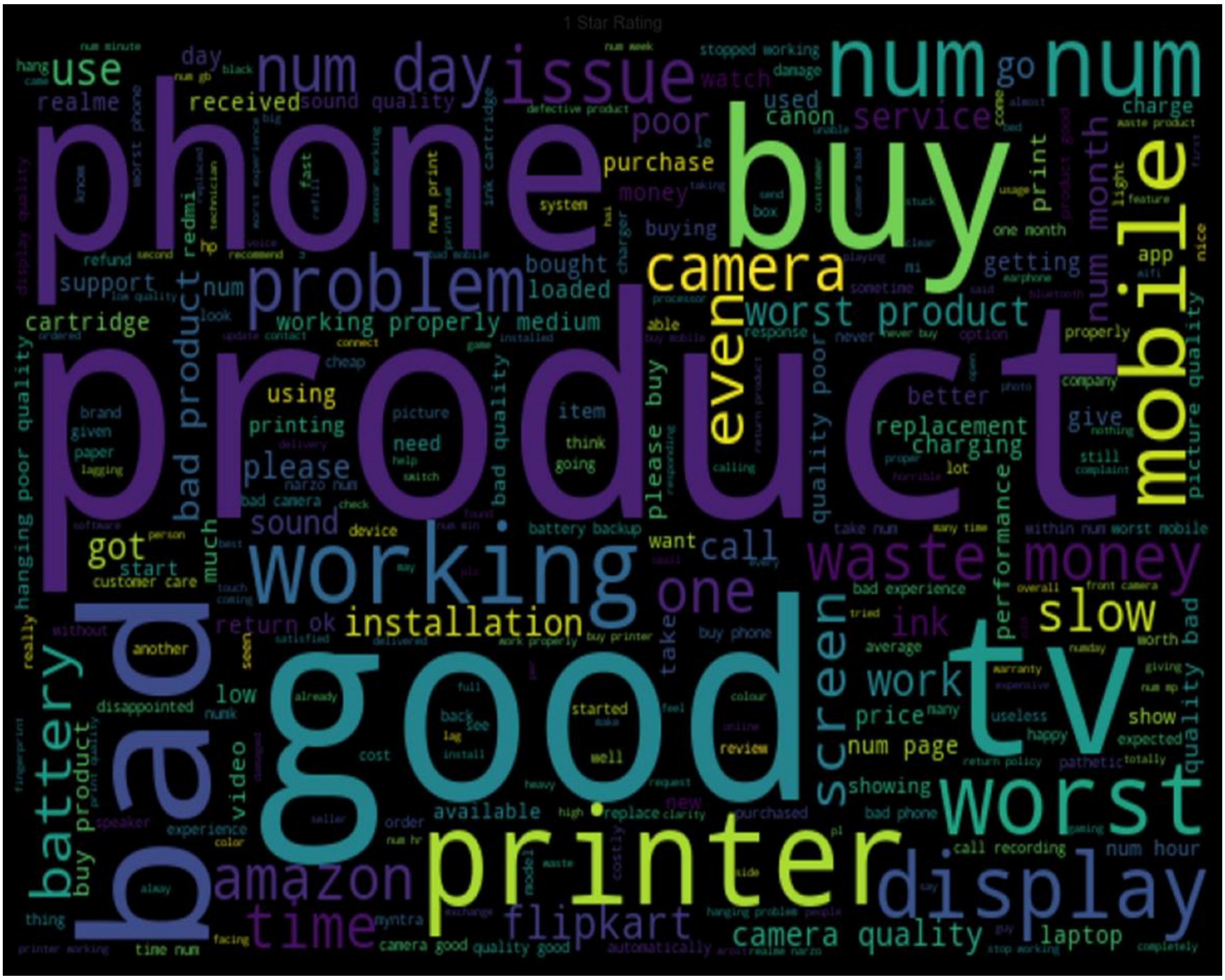
Word Clouds of the most frequent words used in reviews corresponding to various Rating Scores.



Word Clouds of the most frequent words used in reviews corresponding to various Rating Scores.






1 Star Rating
product
phone
good
bad
buy
tv
printer
num
mobile
issue
num day
problem
working
camera
display
worst
waste money
amazon
time
flipkart
installation
battery
slow
screen
worst product
use
even
one
service
poor
call
camera quality
working properly medium
replacement
charging
cartridge
purchase
received
sound quality
laptop

Top 10 words and their corresponding Ratings, along with their counts

.

	Five Star Words	Four Star Words	Three Star Words	Two Star Words	One Star Words
0	(num, 1577)	(good, 855)	(num, 675)	(num, 617)	(num, 2102)
1	(good, 1457)	(num, 845)	(good, 491)	(good, 311)	(product, 1000)
2	(quality, 1419)	(quality, 634)	(quality, 359)	(quality, 289)	(quality, 715)
3	(product, 1303)	(product, 443)	(camera, 196)	(camera, 243)	(buy, 582)
4	(sound, 749)	(sound, 345)	(tv, 181)	(phone, 228)	(phone, 581)
5	(camera, 582)	(price, 265)	(product, 172)	(product, 145)	(working, 551)
6	(price, 559)	(tv, 247)	(time, 148)	(bad, 145)	(camera, 484)
7	(picture, 502)	(picture, 215)	(sound, 143)	(working, 141)	(good, 478)
8	(best, 499)	(camera, 201)	(phone, 135)	(battery, 134)	(bad, 461)
9	(tv, 452)	(laptop, 178)	(battery, 132)	(time, 132)	(time, 447)

Balancing out the classes using SMOTE technique.

SMOTE Technique for balancing classess

```
from imblearn.over_sampling import SMOTE as stm
```

```
smt_x,smt_y = stm().fit_resample(X,y)
```

Train-Test Split.

Checking Best random state

```
maxAcc = 0
maxRS=0
for i in range(0,100):
    x_train,x_test,y_train,y_test = train_test_split(smt_x, smt_y,test_size = .30, random_state = i)
    RF =  RandomForestClassifier()
    RF.fit(x_train,y_train)
    pred = RF.predict(x_test)
    acc  = accuracy_score(y_test,pred)
    if acc>maxAcc:
        maxAcc=acc
        maxRS=i
print(f"Best Accuracy is: {maxAcc} on random_state: {maxRS}")
```

```
Best Accuracy is: 0.7166314420419444 on random_state: 12
```

```
x_train,x_test,y_train,y_test = train_test_split(smt_x,smt_y,test_size = .30,random_state = 35)
```

Model Development

LogisticRegression

```
lr = LogisticRegression(solver='liblinear')
lr.fit(x_train,y_train)
```

```
LogisticRegression(solver='liblinear')
```

```
lr.score(x_train,y_train)
```

```
0.6040483573145664
```

```
pred_lr = lr.predict(x_test)
```

```
print('Classification Report:',classification_report(y_test, pred_lr))
print('Confusion Matrix:',confusion_matrix(y_test,pred_lr))
```

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

         1.0       0.63      0.62      0.63      2424
         2.0       0.50      0.63      0.56      2413
         3.0       0.48      0.47      0.47      2423
         4.0       0.50      0.37      0.43      2492
         5.0       0.57      0.58      0.57      2550

    accuracy                           0.54     12302
   macro avg       0.53      0.54      0.53     12302
weighted avg       0.53      0.54      0.53     12302

Confusion Matrix: [[1514  585  200   71   54]
 [ 471 1526  292   82   42]
 [ 225  496 1145  262  295]
 [  99  313  417  934  729]
 [ 101  109  347  526 1467]]
```

RandomForestClassifier

```
rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
```

```
RandomForestClassifier()
```

```
rfc.score(x_train,y_train)
```

```
0.8783054036163467
```

```
pred_rfc = rfc.predict(x_test)
```

```
print('Classification Report:',classification_report(y_test, pred_rfc))
print('Confusion Matrix:',confusion_matrix(y_test,pred_rfc))
```

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

         1.0       0.79      0.76      0.77      2424
         2.0       0.80      0.81      0.80      2413
         3.0       0.65      0.73      0.69      2423
         4.0       0.62      0.67      0.64      2492
         5.0       0.69      0.57      0.62      2550

    accuracy                           0.70     12302
   macro avg       0.71      0.71      0.70     12302
weighted avg       0.71      0.70      0.70     12302

Confusion Matrix: [[1833  255  159   74  103]
 [ 180 1946  192   48   47]
 [ 130  126 1771  269  127]
 [  65   70  312 1668  377]
 [ 123   43  303  638 1443]]
```

XGBClassifier

```
xgb = XGBClassifier()
xgb.fit(x_train,y_train)
```

```
[16:42:14] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly s
et eval_metric if you'd like to restore the old behavior.

XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=8,
              num_parallel_tree=1, objective='multi:softprob', predictor='auto',
              random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=None,
              subsample=1, tree_method='exact', validate_parameters=1,
              verbosity=None)
```

```
xgb.score(x_train,y_train)
```

```
0.67759467651465
```

```
pred_xgb = xgb.predict(x_test)
```

```
print('Classification Report:',classification_report(y_test, pred_xgb))
print('Confusion Matrix:',confusion_matrix(y_test,pred_xgb))
```

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

         1.0       0.60      0.64      0.62      2424
         2.0       0.54      0.60      0.57      2413
         3.0       0.52      0.49      0.50      2423
         4.0       0.51      0.50      0.51      2492
         5.0       0.61      0.55      0.57      2550

    accuracy                           0.56     12302
   macro avg       0.56      0.56      0.55     12302
weighted avg       0.56      0.56      0.55     12302

Confusion Matrix: [[1547  521  183  101   72]
 [ 525 1451  269  122   46]
 [ 264  392 1186  368  213]
 [ 109  195  357 1252  579]
 [ 121  114  300  621 1394]]
```

AdaBoostClassifier

```
ad = AdaBoostClassifier()
ad.fit(x_train,y_train)
```

```
AdaBoostClassifier()
```

```
ad.score(x_train,y_train)
```

```
0.4261227049437341
```

```
pred_ad = ad.predict(x_test)
```

```
print('Classification Report:',classification_report(y_test, pred_ad))
print('Confusion Matrix:',confusion_matrix(y_test,pred_ad))
```

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

         1.0       0.52      0.52      0.52      2424
         2.0       0.34      0.48      0.40      2413
         3.0       0.39      0.25      0.31      2423
         4.0       0.37      0.41      0.39      2492
         5.0       0.54      0.46      0.50      2550

    accuracy                           0.42     12302
   macro avg       0.43      0.42      0.42     12302
weighted avg       0.43      0.42      0.42     12302

Confusion Matrix: [[1258  833  155  122   56]
 [ 656 1167  362  176   52]
 [ 267  708  609  593  246]
 [ 101  417  325 1028  621]
 [ 124  301  113  850 1162]]
```

MultinomialNB

```
mnb = MultinomialNB()
mnb.fit(x_train,y_train)
```

```
MultinomialNB()
```

```
mnb.score(x_train,y_train)
```

```
0.6059645333240428
```

```
pred_mt = mnb.predict(x_test)
```

```
print('Classification Report:',classification_report(y_test, pred_mt))
print('Confusion Matrix:',confusion_matrix(y_test,pred_mt))
```

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

         1.0       0.59      0.67      0.63      2424
         2.0       0.60      0.59      0.59      2413
         3.0       0.55      0.42      0.47      2423
         4.0       0.45      0.48      0.47      2492
         5.0       0.54      0.58      0.56      2550

    accuracy                           0.55     12302
   macro avg       0.55      0.55      0.54     12302
weighted avg       0.55      0.55      0.54     12302

Confusion Matrix: [[1620  457  151  110   86]
 [ 542 1414  242  151   64]
 [ 299  309 1006  468  341]
 [ 160  108  243 1196  785]
 [ 102   73  181  711 1483]]
```

PassiveAggressiveClassifier

```
pac = PassiveAggressiveClassifier()
pac.fit(x_train,y_train)
```

```
PassiveAggressiveClassifier()
```

```
pac.score(x_train,y_train)
```

```
0.6992648852036373
```

```
pred_pac = pac.predict(x_test)
```

```
print('Classification Report:',classification_report(y_test, pred_pac))
print('Confusion Matrix:',confusion_matrix(y_test,pred_pac))
```

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

         1.0       0.67      0.65      0.66      2424
         2.0       0.63      0.68      0.65      2413
         3.0       0.52      0.63      0.57      2423
         4.0       0.56      0.47      0.51      2492
         5.0       0.57      0.52      0.54      2550

    accuracy                           0.59     12302
   macro avg       0.59      0.59      0.59     12302
weighted avg       0.59      0.59      0.59     12302

Confusion Matrix: [[1567  408  245  123   81]
 [ 397 1639  255   91   31]
 [ 167  258 1535  200  263]
 [  79  164  446 1176  627]
 [ 122  129  474  510 1315]]
```

ComplementNB

```
cp = ComplementNB()
cp.fit(x_train,y_train)
```

```
ComplementNB()
```

```
cp.score(x_train,y_train)
```

```
0.6199700379751245
```

```
pred_cp = cp.predict(x_test)
```

```
print('Classification Report:',classification_report(y_test, pred_cp))
print('Confusion Matrix:',confusion_matrix(y_test,pred_cp))
```

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

         1.0       0.57      0.69      0.63      2424
         2.0       0.61      0.58      0.60      2413
         3.0       0.60      0.40      0.48      2423
         4.0       0.48      0.47      0.48      2492
         5.0       0.52      0.62      0.57      2550

    accuracy                           0.55     12302
   macro avg       0.56      0.55      0.55     12302
weighted avg       0.56      0.55      0.55     12302

Confusion Matrix: [[1673  412  138  100  101]
 [ 595 1399  203  126   90]
 [ 320  304  967  441  391]
 [ 178   95  172 1177  870]
 [ 145   75  136  618 1576]]
```

Analyzing Accuracy of The Models

Classification Report consisting of Precision, Recall, Support and F1- score were the metrics used to evaluate the Model Performance. Precision is defined as the ratio of true positives to the sum of true and false positives. Recall is defined as the ratio of true positives to the sum of true positives and false negatives. The F1 is the weighted harmonic mean of precision and recall. The closer the value of the F1 score is to 1.0, the better the expected performance of the model is. Support is the number of actual occurrences of the class in the dataset. It doesn't vary between models; it just diagnoses the performance evaluation process.

Model Cross Validation

Cross validation is a technique for assessing how the statistical analysis generalizes to an independent data set. It is a technique for evaluating machine learning models by training several models on subsets of the available input data and evaluating them on the complementary subset of the data. Using cross-validation, there are high chances that we can detect over-fitting with ease. Model Cross Validation scores were then obtained for assessing how the statistical analysis generalizes to an independent data set. The models were evaluated by training several models on subsets of the available input data and evaluating them on the complementary subset of the data.

AdaBoostClassifier

```
print(cvs(ad,smt_x,smt_y,cv=5).mean())
```

```
0.3887818558712352
```

MultinomialNB

```
print(cvs(mnb,smt_x,smt_y,cv=5).mean())
```

```
0.4974759175710279
```

ComplementNB

```
print(cvs(cp,smt_x,smt_y,cv=5).mean())
```

```
0.5161321790025607
```

PassiveAggressiveClassifier

```
print(cvs(pac,smt_x,smt_y,cv=5).mean())
```

```
0.5694427508840385
```

RandomForestClassifier

```
print(cvs(rfc,smt_x,smt_y,cv=5).mean())
```

```
0.6894037312522864
```

LogisticRegression

```
print(cvs(lr,smt_x,smt_y,cv=5).mean())
```

```
0.49923180099987813
```

XGBClassifier

```
print(cvs(xgb,smt_x,smt_y,cv=5).mean())
```

```
[16:58:14] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly s
et eval_metric if you'd like to restore the old behavior.
[16:58:34] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly s
et eval_metric if you'd like to restore the old behavior.
[16:58:55] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly s
et eval_metric if you'd like to restore the old behavior.
[16:59:15] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly s
et eval_metric if you'd like to restore the old behavior.
[16:59:34] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly s
et eval_metric if you'd like to restore the old behavior.
0.5226191927813681
```

ROC AUC Scores

The score is used to summarize the trade-off between the true positive rate and false positive rate for a predictive model using different probability thresholds. The AUC value lies between 0.5 to 1 where 0.5 denotes a bad classifier and 1 denotes an excellent classifier.

RandomForestClassifier

```
roc_auc_score(y_test,rf_prob,multi_class='ovo')
```

```
0.9176995358026778
```

LogisticRegression

```
roc_auc_score(y_test,lr_prob,multi_class='ovo')
```

```
0.8314843298397572
```

XGBClassifier

```
roc_auc_score(y_test,xgbc_prob,multi_class='ovo')
```

```
0.8528766237214093
```

AdaBoostClassifier

```
roc_auc_score(y_test,adbc_prob,multi_class='ovo')
```

```
0.7403546886753338
```

MultinomialNB

```
roc_auc_score(y_test,mnb_prob,multi_class='ovo')
```

```
0.8268459472917649
```

ComplementNB

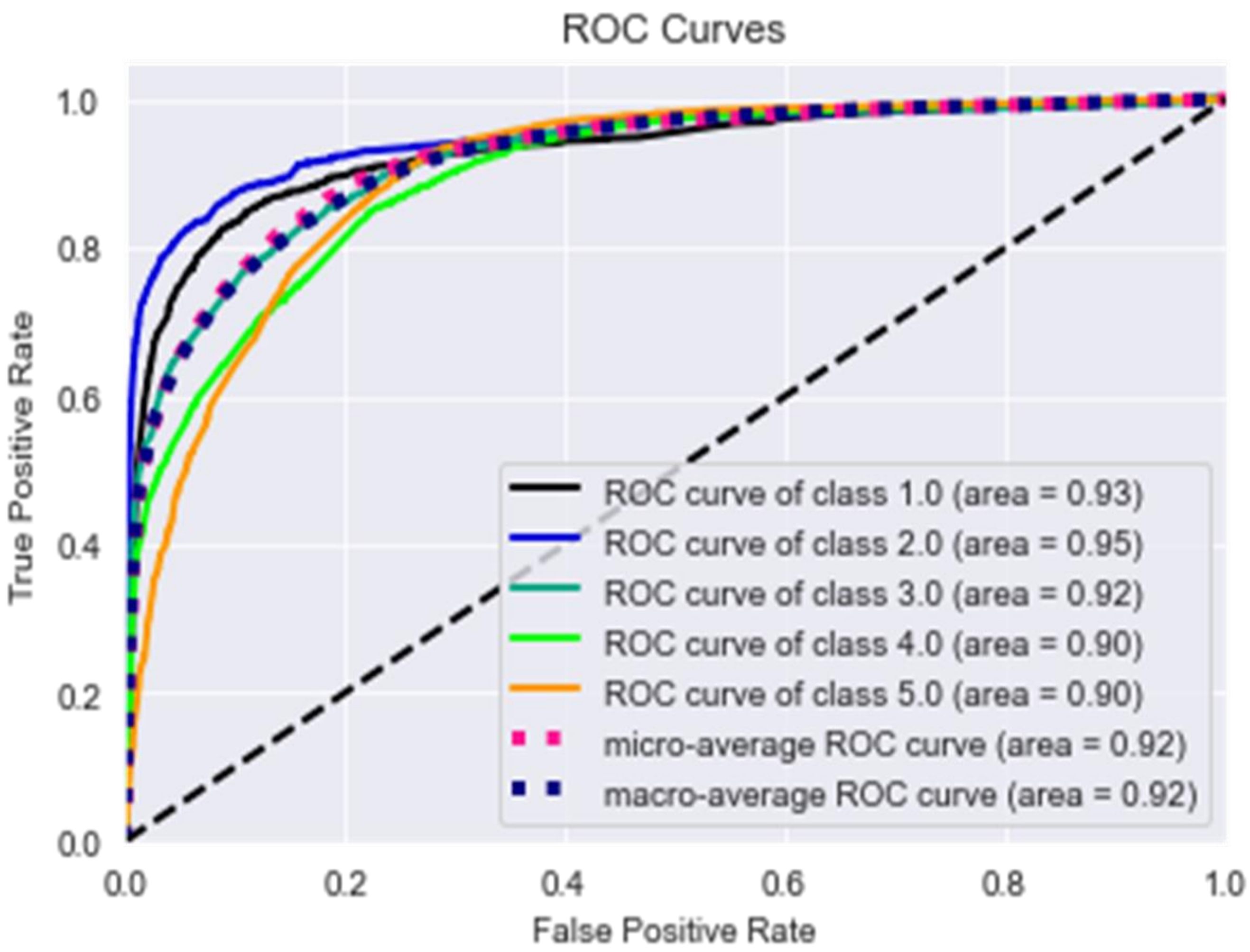
```
roc_auc_score(y_test,cnb_prob,multi_class='ovo')
```

```
0.8332886077417576
```

ROC AUC Curve

The AUC-ROC curve helps us visualize how well our machine learning classifier is performing. ROC curves are appropriate when the observations are balanced between each class.

Random Forest Classifier



Hyper Parameter Tuning

GridSearchCV was used for Hyper Parameter Tuning of the Random Forest Classifier model.

Hyper Parameter Tuning

```
params = {'n_estimators':[400,500,600],'max_depth': [80,90,95],'min_samples_leaf':[2,5,30],'min_samples_split':[1,2,5],'crit
```

```
GridCV = GridSearchCV(RandomForestClassifier(),params,cv=5,n_jobs = -1,verbose = 1)
```

```
GridCV.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 486 candidates, totalling 2430 fits

GridSearchCV(cv=5, estimator=RandomForestClassifier(), n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [80, 90, 95],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'min_samples_leaf': [2, 5, 30],
                         'min_samples_split': [1, 2, 5],
                         'n_estimators': [400, 500, 600]},
             verbose=1)
```

```
GridCV.best_params_
```

```
{'criterion': 'entropy',
 'max_depth': 95,
 'max_features': 'sqrt',
 'min_samples_leaf': 2,
 'min_samples_split': 2,
 'n_estimators': 600}
```

```
:lassifier(n_estimators = 500,criterion = 'gini', max_depth= 95, max_features = 'auto',min_samples_leaf = 2, min_samples_split = :
:rain)
:t(x_test)
```

```
acc = accuracy_score(y_test,rfpred)
conf_matrx = confusion_matrix(y_test,rfpred)
```

```
print('Accurcay:',acc*100)
print('Confusion Matrix:',conf_matrx)
```

```
Accurcay: 58.3157210209722
Confusion Matrix: [[1640  404  203   70  107]
 [ 455 1495  283  105   75]
 [ 265  267 1303  255  333]
 [ 122  114  391 1110  755]
 [ 110   40  339  435 1626]]
```

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 texts 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 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.

Area of Improvement

- Less time complexity
- More computational power can be given
- More accurate reviews can be given
- Many more permutations and combinations in hyper parameter tuning can be used to obtain better parameter list.
- 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