FLIP ROBO

PROJECT REPORT

ON

Rating Prediction Project

SUBMITTED BY

V Samyukta

ACKNOWLEDGMENT

It gives me immense pleasure to deliver this report. Working on this project was a great learning experience. I express my sincere gratitude to Flip Robo Technologies for giving me the opportunity to work on this project, Ratings Prediction using machine learning algorithms and NLTK suite.

Flip Robo Technologies (Bangalore) provided all of the necessary information, required for the completion of the project.

I express my gratitude to my SME, Ms. Khushboo Garg, for providing the directions for carrying out this project
.

My heartfelt gratitude to DataTrained institute for providing me this internship opportunity. Last but not least to my sincere thanks to my family and all those who helped me directly or indirectly in completion this project.

CONTENTS

1.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 Hadad 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. A user may express in his review(s), their satisfaction / dissatisfaction with a product, based on its quality, features performance, and monetary worth and they may then give the product a rating score based on their opinion of it. These reviews have contextual data based on users' experiences with the products and their opinions of them. The user ratings have a strong correlation with the contextual data carried in their reviews. Thus, comparing the similarity in the reviews with the similarity in the scores based on those reviews can be a basis for predicting user ratings based on the context, inference and semantics of their 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.

2.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 normalised. 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 reviews, 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/ are taken. The data was converted into a Pandas Dataframe under attributes, Comment and Ratings columns and saved as a .xlsx file.
  The data set contains nearly 20469 samples with 2 features. Since **Ratings** is my target column and it is a categorical column with 5 categories so this problem is a **Multi Classification Problem**. The Ratings can be 1, 2, 3, 4 or 5, which represents the likely ness of the product to the customer. The data set includes:
- Comment: Text Content of the Review.
- Ratings: Ratings out of 5 stars.

This project is more about exploration, feature engineering and classification that can be done on this data. Since the data set is huge and includes multi classification of ratings, we can do good amount of data exploration and derive some interesting features using the Review column available.
We need to build a model that can predict Ratings of the reviewer.

Features Information:

- **Data Pre-processing 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 addresses etc 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 vectorised using TfidVectorizer are input and the corresponding rating is predicted based on their context as output by classification models.

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

  Hardware technology being Used:-

  - CPU: HP Pavilion
  - Chip: intel core13 8th Gen
  - RAM: 8 GB

  Software Technology being Used:-

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

  Libraries/Packages Used:-

  Pandas, NumPy, matplotlib, seaborn, scikit-learn and pandas_profiling.

  - Windows 10 Operating System
    • Anaconda Package and Environment Manager: Anaconda is a distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows and provides a host of tools and environment for conducting Data Analytical and Scientific works. Anaconda provides all the necessary Python packages and libraries for Machine learning projects.
    • Jupyter Notebook: The Jupyter Notebook is an open-source web application that allows data scientists to create and share documents that integrate live code, equations, computational output, visualizations, and other multimedia resources, along with explanatory text in a single document.
    • Python3: It is open source, interpreted, high level language and provides great approach for object-oriented programming. It is one of the best languages used for Data Analytics and Data science projects/application. Python provides numerous libraries to deal with mathematics, statistics and scientific function.
    • Python Libraries used:
  - Pandas: For carrying out Data Analysis, Data Manipulation, Data Cleaning etc
  - Numpy: For performing a variety of operations on the datasets.
  - matplotlib.pyplot, Seaborn: For visualizing Data and various relationships between Feature and Label Columns
  - imblearn.over_sampling: To employ SMOTE technique for balancing out the classes.
  - Statsmodels: 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
- Wordcloud: For Data Visualization
- NLTK: To use various Natural Language Processing Tools

3.DATA ANALYSIS AND VISUALIZATION

- **Identification of possible problem-solving approaches (methods):**
  I have converted text into feature vectors using TF-IDF vectorizer and separated our feature and labels. Also, before building the model, I made sure that the input data is cleaned and scaled before it was fed into the machine learning models. Just making the Reviews more appropriate so that we'll get less word to process and get more accuracy. Removed extra spaces, converted email address into email keyword, and phone number etc. Tried to make Reviews small and more appropriate as much as possible.

- **Testing of Identified Approaches (Algorithms):**

In this nlp based project we need to predict Ratings which is a multiclassification problem. I have converted the text into vectors using TFIDF vectorizer and separated our feature and labels then build the model using One Vs Rest Classifier. Among all the algorithms which I have used for this purpose I have chosen SGDClassifier as best suitable algorithm for our final model as it is performing well compared to other algorithms while evaluating with different metrics I have used following algorithms and evaluated them

- MultinomialNB
- LogisticRegression
- RandomForestClassifier
- AdaBoostClassifier
- PassiveAggressiveClassifier
- ComplementNB

From all of these above models RandomForestClassifier was giving me good performance with less difference in accuracy score and cv score.

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

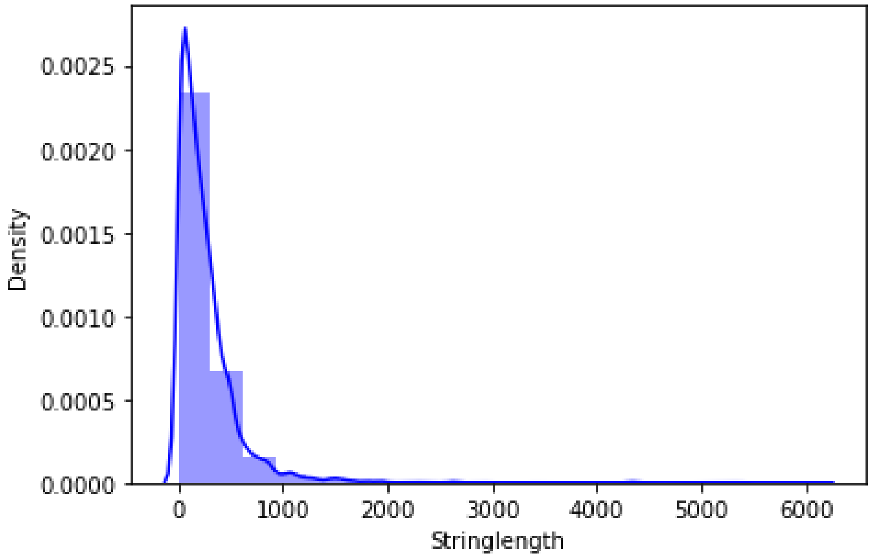
I have used the following metrics for evaluation:

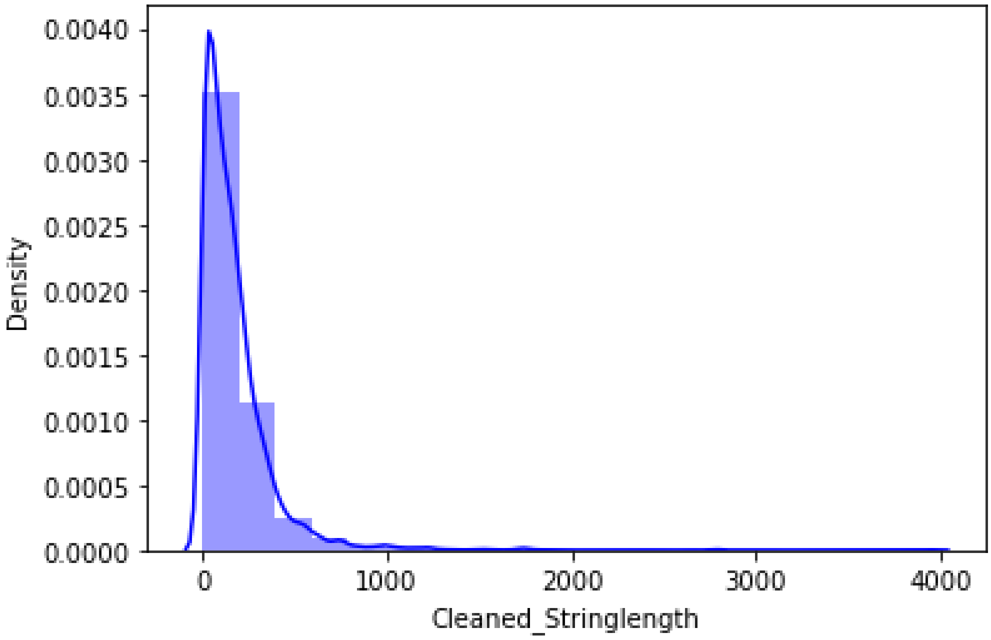
- I have used f1_score, precision_score, recall_score, multilabel_confusion_matrix and hamming loss all these evaluation metrics to select best suitable algorithm for our final model.

- **Precision** can be seen as a measure of quality; higher precision means that an algorithm returns more relevant results than irrelevant ones.
- **Recall** is used as a measure of quantity and high recall means that an algorithm returns most of the relevant results.
- **Accuracy score** is used when the True Positives and True negatives are more important. Accuracy can be used when the class distribution is similar.
- **F1-score** is used when the False Negatives and False Positives are crucial. While F1-score is a better metric when there are imbalanced classes.

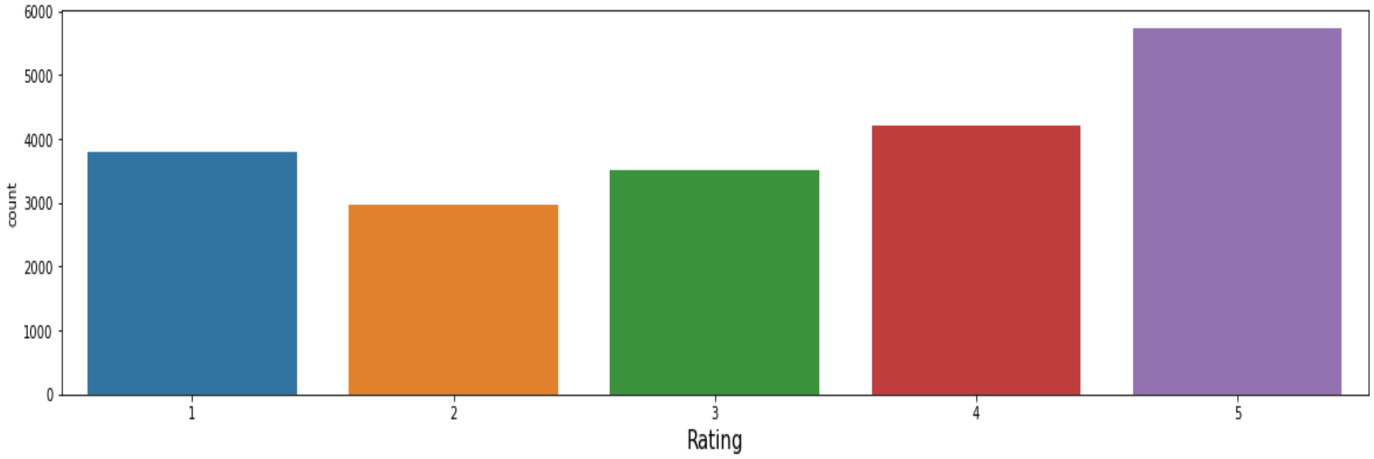
- **Visualization:**
  I have used bar plots to see the relation of categorical feature and I have used dist plots for numerical features.

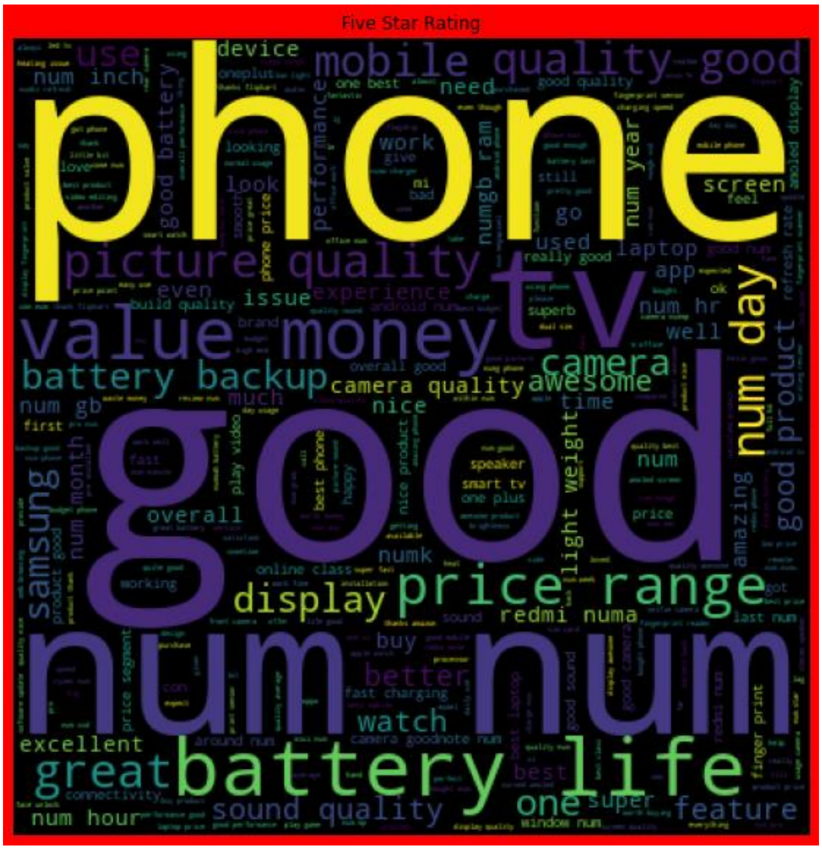
➢ **Visualization of numerical features:**

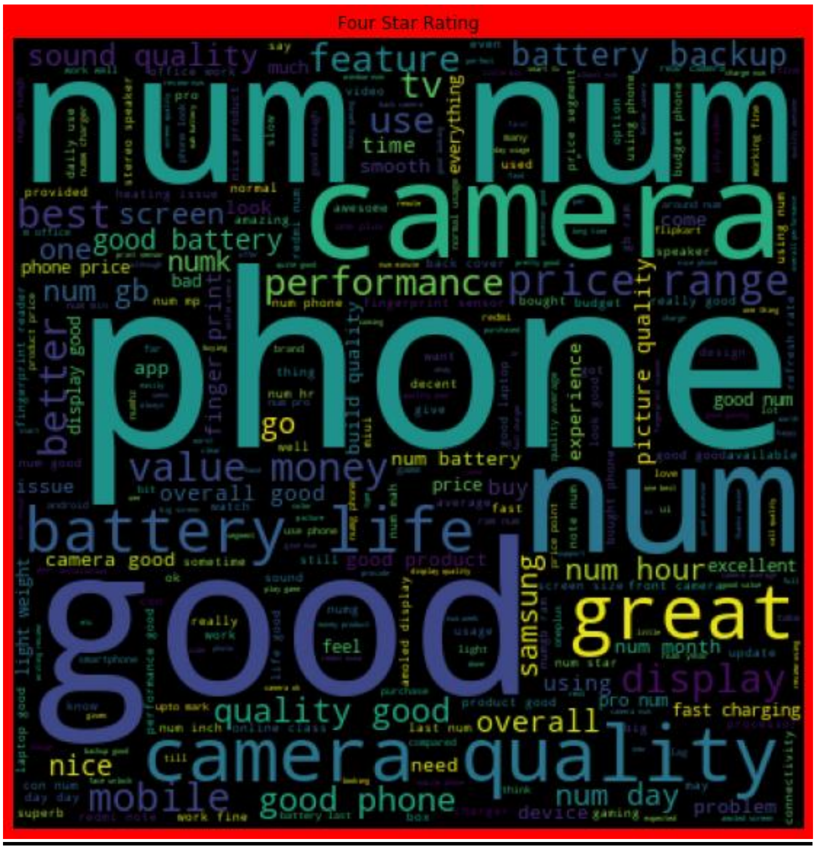


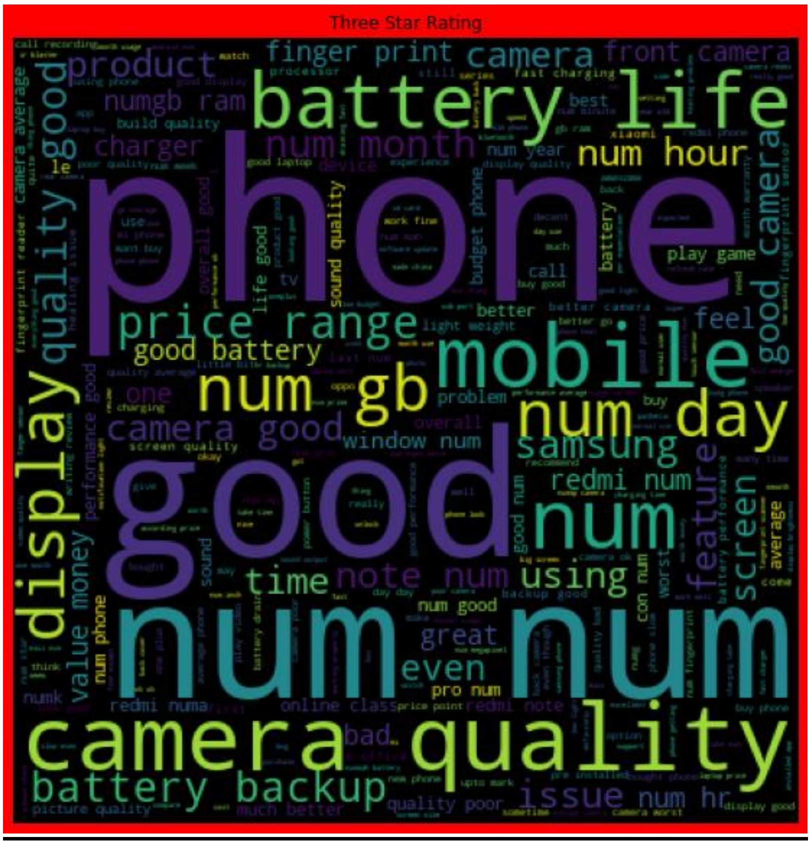


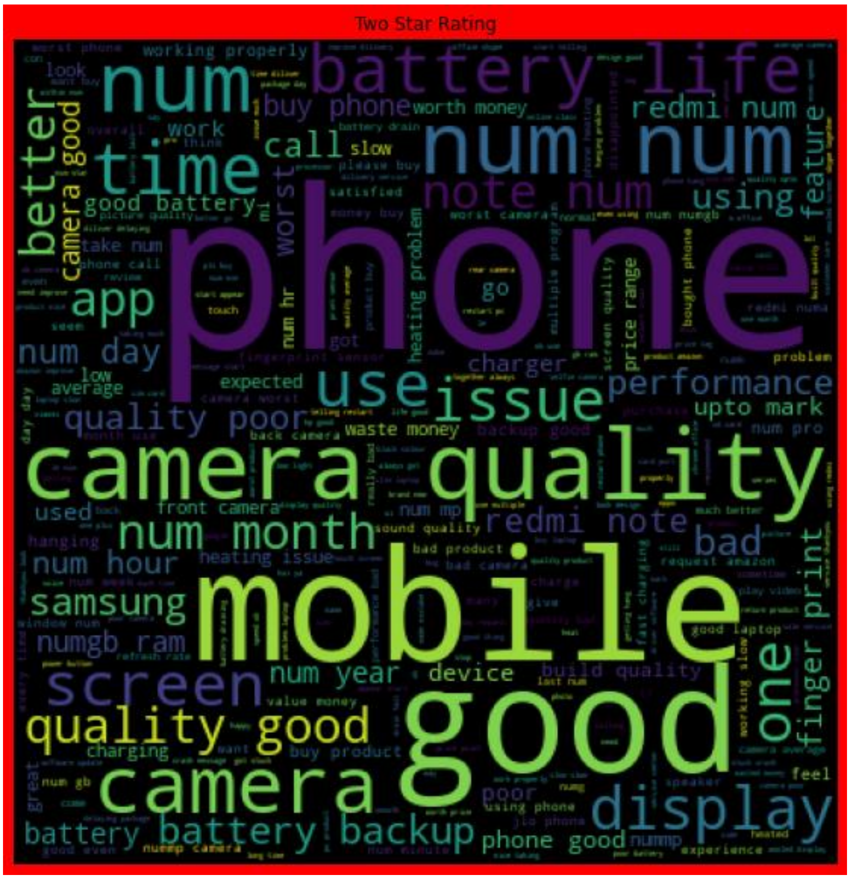
2. Visualization of categorical features with target:

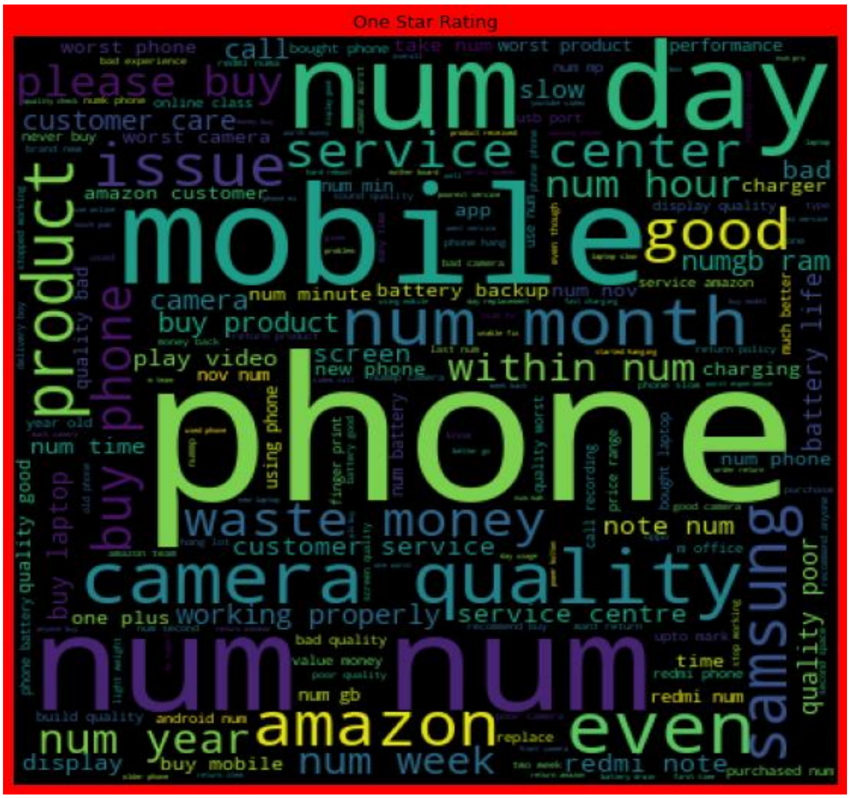





Four Star Rating


Three Star Rating

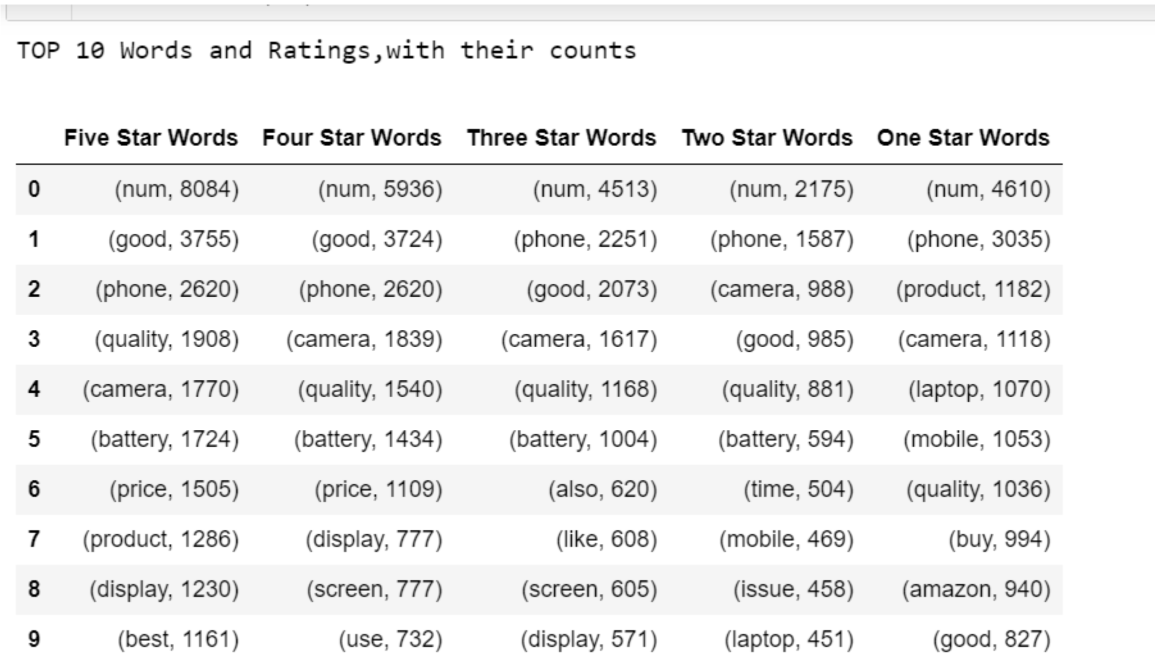




Observations:

- The length of most of the comments were between 0 to 800 before cleaning the data. After cleaning data, it came down to between 0 - 500

From the graphs above the following observations are made:
• Reviews corresponding to 5.0 rating frequently carry words like: ‘good’,’excellent’,’quality’, ‘value money’ etc indicating very high customer satisfaction and high quality product.
• Reviews corresponding to 4.0 rating frequently carry words like: ‘good’, ’great’, ’performance’, ’features’, ‘quality’, ’price’ etc indicating high customer satisfaction and good quality product.
• Reviews corresponding to 3.0 rating frequently carry words like: ‘good’, ’well’, ’average, ’quality’, ’issue’ etc indicating customer dissatisfaction and average to below average product quality.
• Reviews corresponding to 2.0 rating frequently carry words like: ‘problem’, ‘bad’,’issues’, ’waste money’,’poor quality’ etc indicating high customer dissatisfaction and below average product quality.
• Reviews corresponding to 1.0 rating frequently carry words like: ‘stopped’, ’working’, ’cheap’, ’return’, ’issue’, ’waste money’, ’poor quality’,’ customer care’, ’bad’, ‘poor build quality’ etc indicate very high customerdissatisfaction and poor quality product.



TOP 10 Words and Ratings,with their counts

	Five Star Words	Four Star Words	Three Star Words	Two Star Words	One Star Words
0	(num, 8084)	(num, 5936)	(num, 4513)	(num, 2175)	(num, 4610)
1	(good, 3755)	(good, 3724)	(phone, 2251)	(phone, 1587)	(phone, 3035)
2	(phone, 2620)	(phone, 2620)	(good, 2073)	(camera, 988)	(product, 1182)
3	(quality, 1908)	(camera, 1839)	(camera, 1617)	(good, 985)	(camera, 1118)
4	(camera, 1770)	(quality, 1540)	(quality, 1168)	(quality, 881)	(laptop, 1070)
5	(battery, 1724)	(battery, 1434)	(battery, 1004)	(battery, 594)	(mobile, 1053)
6	(price, 1505)	(price, 1109)	(also, 620)	(time, 504)	(quality, 1036)
7	(product, 1286)	(display, 777)	(like, 608)	(mobile, 469)	(buy, 994)
8	(display, 1230)	(screen, 777)	(screen, 605)	(issue, 458)	(amazon, 940)
9	(best, 1161)	(use, 732)	(display, 571)	(laptop, 451)	(good, 827)

- **Run and evaluate selected models**

<u>1.Model building:</u>

The model algorithms used were as follows:

- Logistic Regression: It is a classification algorithm used to find the probability of event success and event failure. It is used when the dependent variable is binary(0/1, True/False, Yes/No) in nature. It supports categorizing data into discrete classes by studying the relationship from a given set of labelled data. It learns a linear relationship from the given dataset and then introduces a nonlinearity in the form of the Sigmoid function. It not only provides a measure of how appropriate a predictor(coefficient size)is, but also its direction of association (positive or negative).

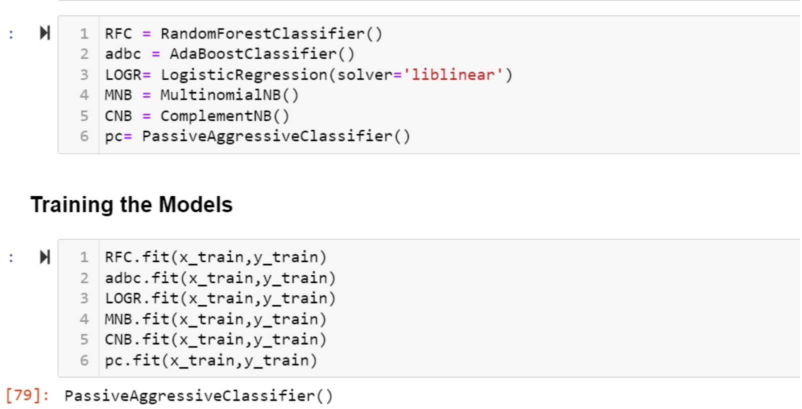
Multinomial Naïve Bayes Classifier: Multinomial Naive Bayes algorithm is a probabilistic learning method that is mostly used in Natural Language Processing (NLP). The algorithm is based on the Bayes theorem. It calculates the probability of each tag for a given sample and then gives the tag with the highest probability as output.

- RandomForestClassifier: A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. A random forest produces good predictions that can be understood easily. It reduces overfitting and can handle large datasets efficiently. The random forest algorithm provides a higher level of accuracy in predicting outcomes over the decision tree algorithm.

- Complement Naïve Bayes Classifier: Complement Naïve Bayes is somewhat an adaptation of the standard Multinomial Naive Bayes algorithm. Complement Naive Bayes is particularly suited to work with imbalanced datasets. In complement Naive Bayes, instead of calculating the probability of an item belonging to a certain class, we calculate the probability of the item belonging to all the classes.

- Passive Aggressive Classifier: Passive-Aggressive algorithms do not require a learning rate and are called so because if the prediction is correct, keep the model and do not make any changes. i.e., the data in the example is not enough to cause any changes in the model. If the prediction is incorrect, make changes to the model. i.e., some change to the model may correct it.

- AdaBoost Classifier: The basis of this algorithm is the Boosting main core: give more weight to the misclassified observations. the meta-learner adapts based upon the results of the weak classifiers, giving more weight to the misclassified observations of the last weak learner. The individual learners can be weak, but as long as the performance of each weak learner is better than random guessing, the final model can converge to a strong learner (a learner not influenced by outliers and with a great generalization power, in order to have strong performances on unknown data)

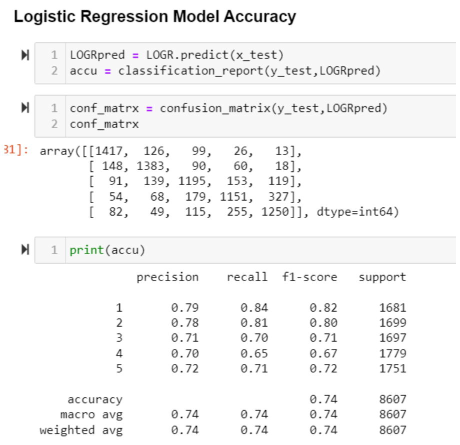


```
RFC = RandomForestClassifier()
adbc = AdaBoostClassifier()
LOGR= LogisticRegression(solver='liblinear')
MNB = MultinomialNB()
CNB = ComplementNB()
pc= PassiveAggressiveClassifier()
```

Training the Models

```
RFC.fit(x_train,y_train)
adbc.fit(x_train,y_train)
LOGR.fit(x_train,y_train)
MNB.fit(x_train,y_train)
CNB.fit(x_train,y_train)
pc.fit(x_train,y_train)
```

```
[79]: PassiveAggressiveClassifier()
```



Logistic Regression Model Accuracy

```
LOGRpred = LOGR.predict(x_test)
accu = classification_report(y_test,LOGRpred)
```

```
conf_matrx = confusion_matrix(y_test,LOGRpred)
conf_matrx
```

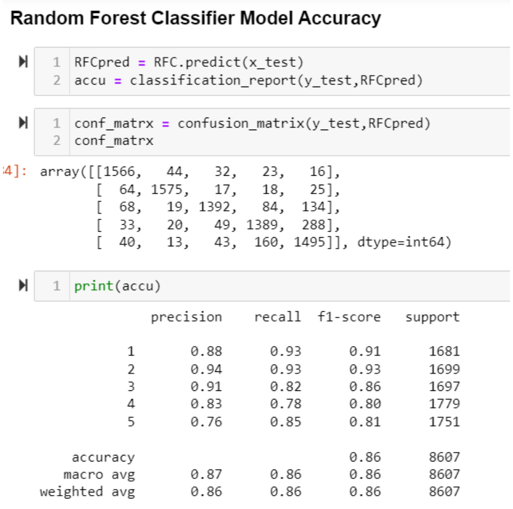
```
31]: array([[1417,  126,   99,   26,   13],
            [ 148, 1383,   90,   60,   18],
            [  91,  139, 1195,  153,  119],
            [  54,   68,  179, 1151,  327],
            [  82,   49,  115,  255, 1250]], dtype=int64)
```

```
print(accu)
```

```
              precision    recall  f1-score   support

           1       0.79      0.84      0.82      1681
           2       0.78      0.81      0.80      1699
           3       0.71      0.70      0.71      1697
           4       0.70      0.65      0.67      1779
           5       0.72      0.71      0.72      1751

    accuracy                           0.74      8607
   macro avg       0.74      0.74      0.74      8607
weighted avg       0.74      0.74      0.74      8607
```



Random Forest Classifier Model Accuracy

```
RFCpred = RFC.predict(x_test)
accu = classification_report(y_test,RFCpred)
```

```
conf_matrx = confusion_matrix(y_test,RFCpred)
conf_matrx
```

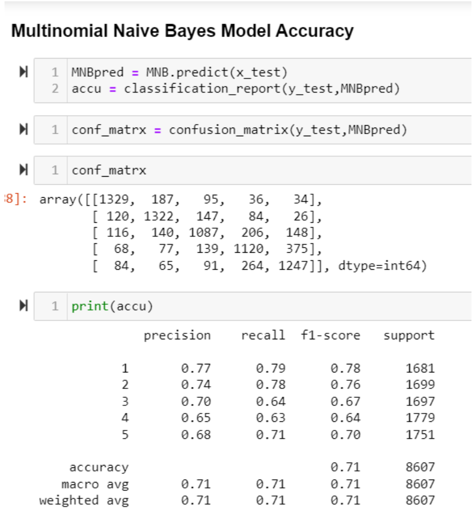
```
4]: array([[1566,   44,   32,   23,   16],
           [  64, 1575,   17,   18,   25],
           [  68,   19, 1392,   84,  134],
           [  33,   20,   49, 1389,  288],
           [  40,   13,   43,  160, 1495]], dtype=int64)
```

```
print(accu)
```

```
              precision    recall  f1-score   support

           1       0.88      0.93      0.91      1681
           2       0.94      0.93      0.93      1699
           3       0.91      0.82      0.86      1697
           4       0.83      0.78      0.80      1779
           5       0.76      0.85      0.81      1751

    accuracy                           0.86      8607
   macro avg       0.87      0.86      0.86      8607
weighted avg       0.86      0.86      0.86      8607
```



Multinomial Naive Bayes Model Accuracy

```
MNBpred = MNB.predict(x_test)
accu = classification_report(y_test,MNBpred)
```

```
conf_matrx = confusion_matrix(y_test,MNBpred)
```

```
conf_matrx
```

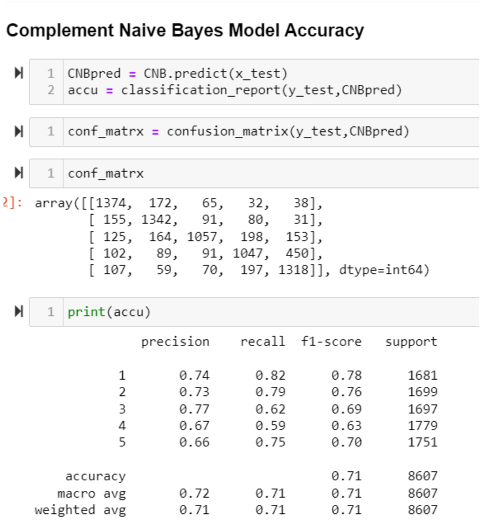
```
8]: array([[1329,  187,   95,   36,   34],
           [ 120, 1322,  147,   84,   26],
           [ 116,  140, 1087,  206,  148],
           [  68,   77,  139, 1120,  375],
           [  84,   65,   91,  264, 1247]], dtype=int64)
```

```
print(accu)
```

```
              precision    recall  f1-score   support

           1       0.77      0.79      0.78      1681
           2       0.74      0.78      0.76      1699
           3       0.70      0.64      0.67      1697
           4       0.65      0.63      0.64      1779
           5       0.68      0.71      0.70      1751

    accuracy                           0.71      8607
   macro avg       0.71      0.71      0.71      8607
weighted avg       0.71      0.71      0.71      8607
```



Complement Naive Bayes Model Accuracy

```
CNBpred = CNB.predict(x_test)
accu = classification_report(y_test,CNBpred)
```

```
conf_matrx = confusion_matrix(y_test,CNBpred)
```

```
conf_matrx
```

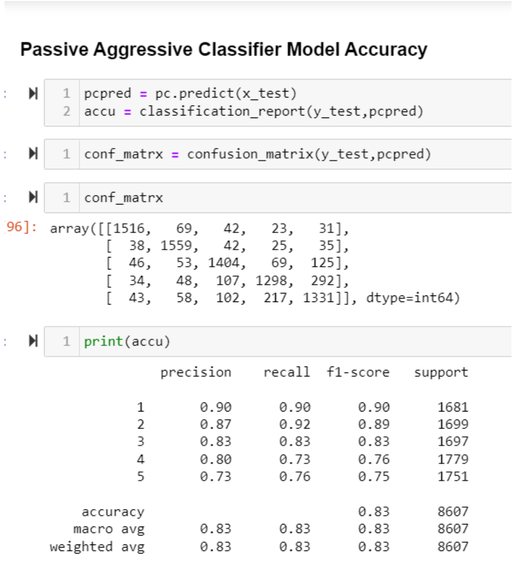
```
2]: array([[1374,  172,   65,   32,   38],
           [ 155, 1342,   91,   80,   31],
           [ 125,  164, 1057,  198,  153],
           [ 102,   89,   91, 1047,  450],
           [ 107,   59,   70,  197, 1318]], dtype=int64)
```

```
print(accu)
```

```
              precision    recall  f1-score   support

           1       0.74      0.82      0.78      1681
           2       0.73      0.79      0.76      1699
           3       0.77      0.62      0.69      1697
           4       0.67      0.59      0.63      1779
           5       0.66      0.75      0.70      1751

    accuracy                           0.71      8607
   macro avg       0.72      0.71      0.71      8607
weighted avg       0.71      0.71      0.71      8607
```



Passive Aggressive Classifier Model Accuracy

```
pcpred = pc.predict(x_test)
accu = classification_report(y_test,pcpred)
```

```
conf_matrx = confusion_matrix(y_test,pcpred)
```

```
conf_matrx
```

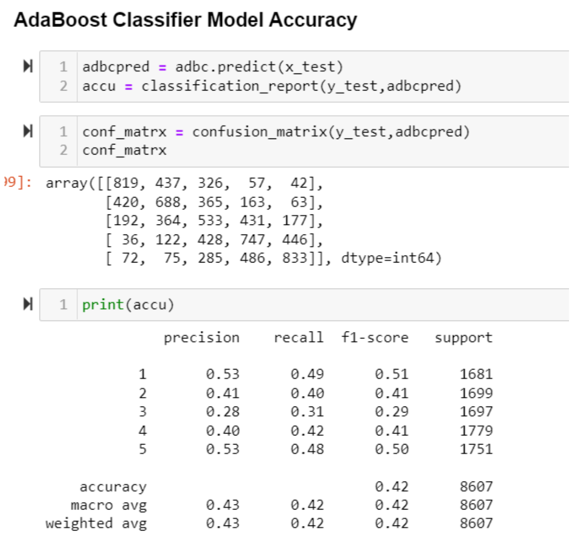
```
96]: array([[1516,   69,   42,   23,   31],
            [  38, 1559,   42,   25,   35],
            [  46,   53, 1404,   69,  125],
            [  34,   48,  107, 1298,  292],
            [  43,   58,  102,  217, 1331]], dtype=int64)
```

```
print(accu)
```

```
              precision    recall  f1-score   support

           1       0.90      0.90      0.90      1681
           2       0.87      0.92      0.89      1699
           3       0.83      0.83      0.83      1697
           4       0.80      0.73      0.76      1779
           5       0.73      0.76      0.75      1751

    accuracy                           0.83      8607
   macro avg       0.83      0.83      0.83      8607
weighted avg       0.83      0.83      0.83      8607
```



AdaBoost Classifier Model Accuracy

```
adbcpred = adbc.predict(x_test)
accu = classification_report(y_test,adbcpred)
```

```
conf_matrx = confusion_matrix(y_test,adbcpred)
conf_matrx
```

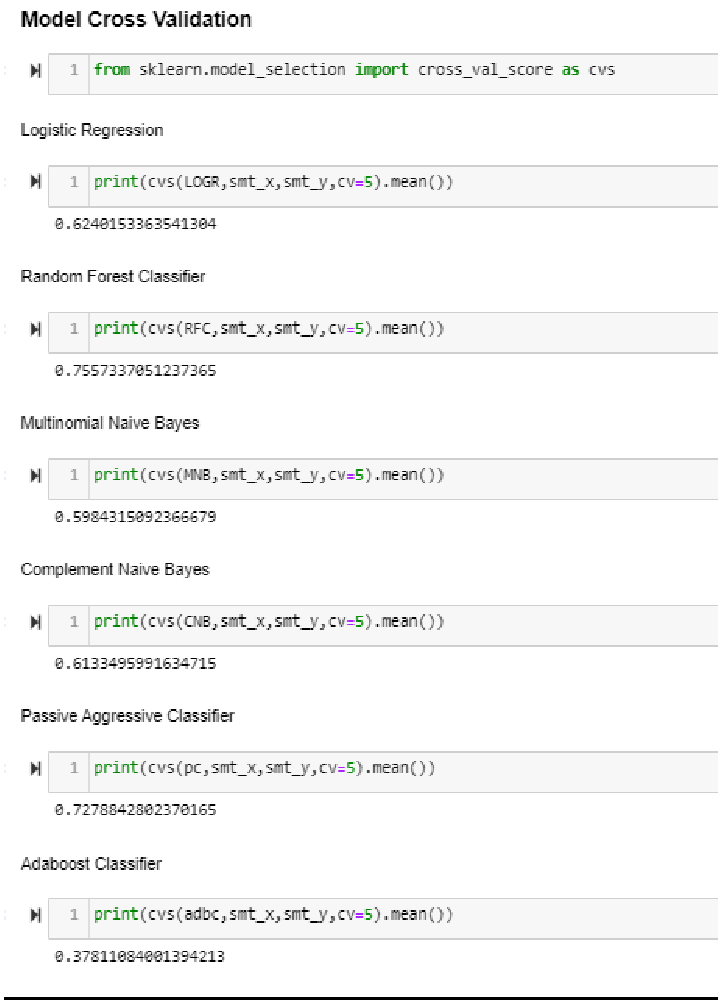
```
9]: array([[819, 437, 326,  57,  42],
           [420, 688, 365, 163,  63],
           [192, 364, 533, 431, 177],
           [ 36, 122, 428, 747, 446],
           [ 72,  75, 285, 486, 833]], dtype=int64)
```

```
print(accu)
```

```
              precision    recall  f1-score   support

           1       0.53      0.49      0.51      1681
           2       0.41      0.40      0.41      1699
           3       0.28      0.31      0.29      1697
           4       0.40      0.42      0.41      1779
           5       0.53      0.48      0.50      1751

    accuracy                           0.42      8607
   macro avg       0.43      0.42      0.42      8607
weighted avg       0.43      0.42      0.42      8607
```



Model Cross Validation

```
from sklearn.model_selection import cross_val_score as cvs
```

Logistic Regression

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

```
0.6240153363541304
```

Random Forest Classifier

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

```
0.7557337051237365
```

Multinomial Naive Bayes

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

```
0.5984315092366679
```

Complement Naive Bayes

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

```
0.6133495991634715
```

Passive Aggressive Classifier

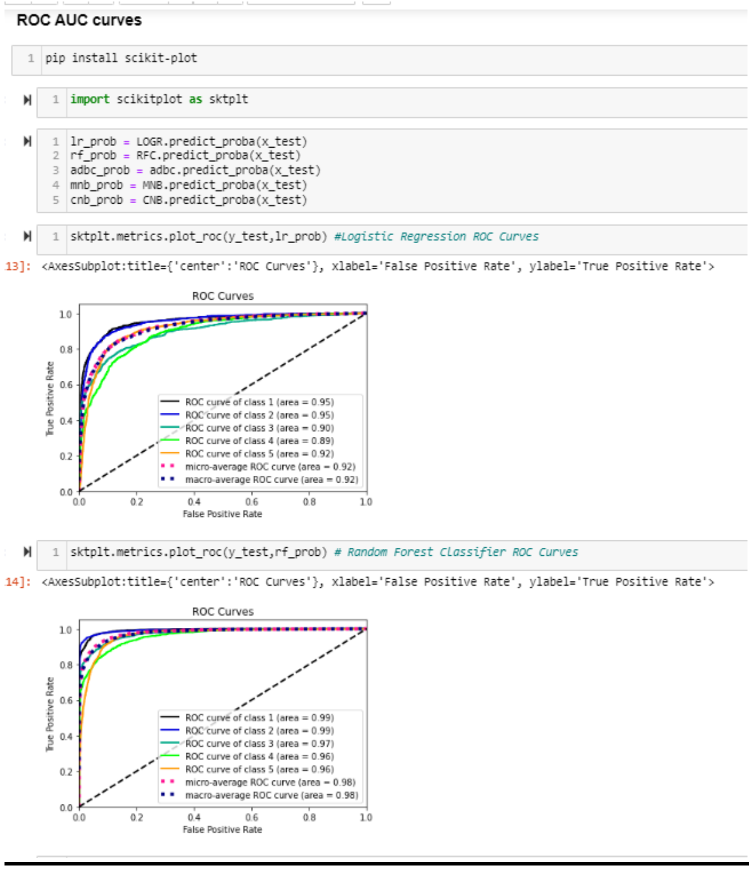
```
print(cvs(pc,smt_x,smt_y,cv=5).mean())
```

```
0.7278842802370165
```

Adaboost Classifier

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

```
0.37811084001394213
```



ROC AUC curves

```
pip install scikit-plot
```

```
import scikitplot as sktplt
```

```
lr_prob = LOGR.predict_proba(x_test)
rf_prob = RFC.predict_proba(x_test)
adbc_prob = adbc.predict_proba(x_test)
mnb_prob = MNB.predict_proba(x_test)
cnb_prob = CNB.predict_proba(x_test)
```

```
sktplt.metrics.plot_roc(y_test,lr_prob) #Logistic Regression ROC Curves
```

```
13]: <AxesSubplot:title={'center':'ROC Curves'}, xlabel='False Positive Rate', ylabel='True Positive Rate'>
```

ROC Curves
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve of class 1 (area = 0.95)
ROC curve of class 2 (area = 0.95)
ROC curve of class 3 (area = 0.90)
ROC curve of class 4 (area = 0.89)
ROC curve of class 5 (area = 0.92)
micro-average ROC curve (area = 0.92)
macro-average ROC curve (area = 0.92)

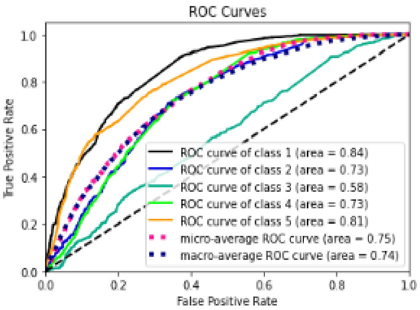
```
sktplt.metrics.plot_roc(y_test,rf_prob) # Random Forest Classifier ROC Curves
```

```
14]: <AxesSubplot:title={'center':'ROC Curves'}, xlabel='False Positive Rate', ylabel='True Positive Rate'>
```

ROC Curves
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve of class 1 (area = 0.99)
ROC curve of class 2 (area = 0.99)
ROC curve of class 3 (area = 0.97)
ROC curve of class 4 (area = 0.96)
ROC curve of class 5 (area = 0.96)
micro-average ROC curve (area = 0.98)
macro-average ROC curve (area = 0.98)

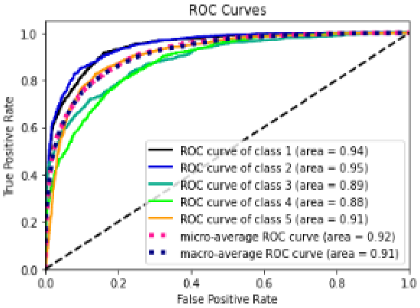
```
sktplt.metrics.plot_roc(y_test,adbc_prob) # Ada Boost Classifier ROC Curve
```

```
]: <AxesSubplot:title={'center':'ROC Curves'}, xlabel='False Positive Rate', ylabel='True Positive Rate'>
```



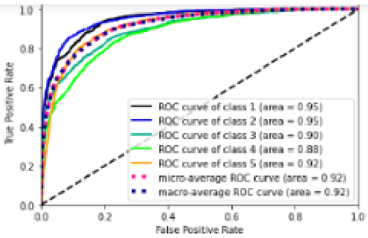
```
sktplt.metrics.plot_roc(y_test,mnb_prob) #Multinomial Naive Bayes ROC Curves
```

```
]: <AxesSubplot:title={'center':'ROC Curves'}, xlabel='False Positive Rate', ylabel='True Positive Rate'>
```



```
sktplt.metrics.plot_roc(y_test,cnb_prob) #Complement Naive Bayes ROC Curves
```

```
]: <AxesSubplot:title={'center':'ROC Curves'}, xlabel='False Positive Rate', ylabel='True Positive Rate'>
```



ROC AUC Scores

Logistic Regression

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

```
8]: 0.9230065019198236
```

Multinomial Naive Bayes

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

```
9]: 0.915030889138626
```

Complement Naive Bayes

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

```
0]: 0.9195214324772051
```

Adaboost Classifier

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

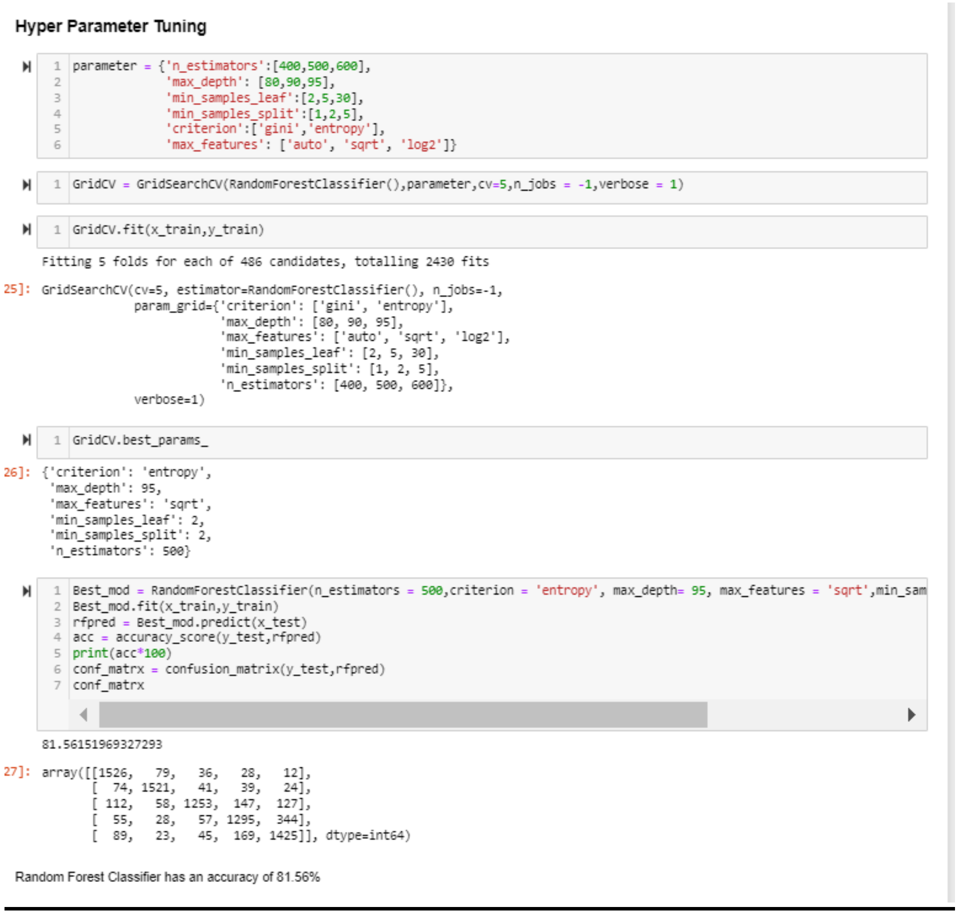
```
1]: 0.7389051854130201
```

Random Forest Classifier

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

```
2]: 0.9765221611655281
```

Based on comparing the above graphs, roc_auc_scores,Precision, Recall, Accuracy Scores with Cross validation scores, it is determined that Random Forest Classifier is the best model for the dataset.



Hyper Parameter Tuning

```
parameter = {'n_estimators':[400,500,600],
             'max_depth': [80,90,95],
             'min_samples_leaf':[2,5,30],
             'min_samples_split':[1,2,5],
             'criterion':['gini','entropy'],
             'max_features': ['auto', 'sqrt', 'log2']}
```

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

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

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

25]: 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_
```

```
26]: {'criterion': 'entropy',
      'max_depth': 95,
      'max_features': 'sqrt',
      'min_samples_leaf': 2,
      'min_samples_split': 2,
      'n_estimators': 500}
```

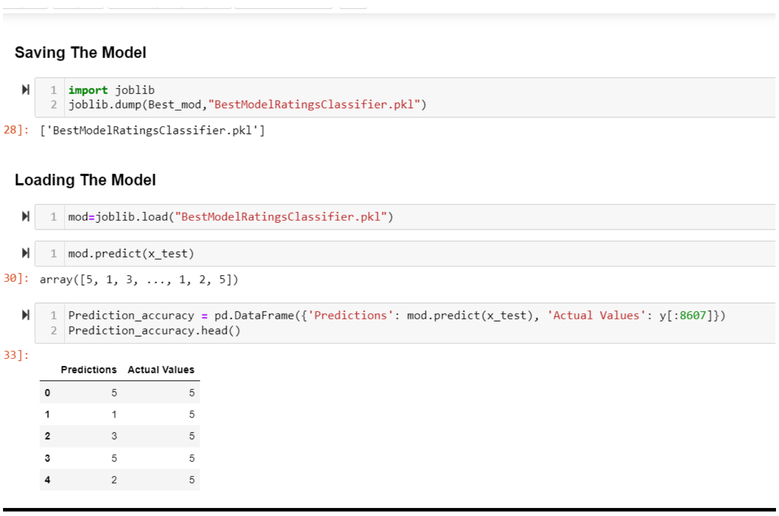
```
Best_mod = RandomForestClassifier(n_estimators = 500,criterion = 'entropy', max_depth= 95, max_features = 'sqrt',min_sam
Best_mod.fit(x_train,y_train)
rfpred = Best_mod.predict(x_test)
acc = accuracy_score(y_test,rfpred)
print(acc*100)
conf_matrx = confusion_matrix(y_test,rfpred)
conf_matrx
```

```
81.56151969327293

27]: array([[1526,   79,   36,   28,   12],
            [  74, 1521,   41,   39,   24],
            [ 112,   58, 1253,  147,  127],
            [  55,   28,   57, 1295,  344],
            [  89,   23,   45,  169, 1425]], dtype=int64)
```

Random Forest Classifier has an accuracy of 81.56%

3. Saving the model and Predicting Rating for test data:



Saving The Model

```
import joblib
joblib.dump(Best_mod,"BestModelRatingsClassifier.pkl")
```

```
28]: ['BestModelRatingsClassifier.pkl']
```

Loading The Model

```
mod=joblib.load("BestModelRatingsClassifier.pkl")
```

```
mod.predict(x_test)
```

```
30]: array([5, 1, 3, ..., 1, 2, 5])
```

```
Prediction_accuracy = pd.DataFrame({'Predictions': mod.predict(x_test), 'Actual Values': y[:8607]})
Prediction_accuracy.head()
```

33]:

	Predictions	Actual Values
0	5	5
1	1	5
2	3	5
3	5	5
4	2	5

- I have saved my best model using .pkl as follows**.**
- Now loading my saved model and predicting the test values.

4.CONCLUSION

- **Key Findings and Conclusions of the Study:**
- In this project I have collected data of reviews and ratings for different products from amazon.in and flipkart.com.
- we have tried to detect the Ratings in commercial websites on a scale of 1 to 5 on the basis of the reviews given by the users. We made use of natural language processing and machine learning algorithms in order to do so.
- 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 RandomForest Classifier for our final model.
- Finally by doing hyperparameter tuning we got optimum parameters for our final model.

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

  Data cleaning was a very important step in removing null values from the dataset. Visualising data helped identify class composition of label column and the most frequently occurring words in reviews corresponding to each of the rating scores. The various data pre-processing and feature engineering steps in the project lent cognizance to various efficient methods for processing textual data. The NLTK suite is very useful in pre-processing text-based data and building classification models.

- **Limitations of this work and Scope for Future Work:**
  First drawback is scrapping the data as it is a fluctuating process. Followed by raw data which is not in format to analyse. Also, this study will not cover all Regression algorithms instead, it is focused on the chosen algorithm, starting from the basic ensemble techniques to the advanced ones.

5.REFERENCE

www.amazon.com

www.flrpkart.com