NAME OF THE PROJECT

Rating prediction

SUBMITTED BY

Reena

ACKNOWLEDGMENT

I would like to express my special gratitude to “Flip Robo” team, who has given me this opportunity to deal with a beautiful dataset and it has helped me to improve my analyzation skills. And I want to express my huge gratitude to Ms sapna (SME Flip Robo), she is the person who has helped me to get out of all the difficulties I faced while doing the project. And also. A huge thanks to my academic team “Data trained” who are the reason behind what I am today.

Contents

1.INTRODUCTION

1.1 Business Problem Framing: Rating prediction is a well-known recommendation task aiming to predict a user's rating for those items which were not rated yet by her. Predictions are computed from users' explicit feedback, i.e. their ratings provided on some items in the past. Another type of feedback are user reviews provided on items which implicitly express users' opinions on items. Recent studies indicate that opinions inferred from users' reviews on items are strong predictors of user's implicit feedback or even ratings and thus, should be utilized in computation. 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 and Yelp!. 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. We have a client who has a website where people write different reviews for technical products. Now they are adding a new feature to their website i.e. the reviewer will have to add stars (rating) as well with the review. The rating is out 5 stars and it only has 5 options available 1 star, 2 stars, 3 stars, 4 stars, 5 stars. Now they want to predict ratings for the reviews which were written in the past and they don't have rating. So we, we have to build an application which can predict the rating by seeing the review.

1.2 Conceptual Background of the Domain Problem

Recommendation systems are an important units in today's e-commerce applications, such as targeted advertising, personalized marketing and information retrieval. In recent years, the importance of contextual information has motivated generation of personalized recommendations according to the available contextual information of users. Compared to the traditional systems which mainly utilize user's rating history, review-based recommendation hopefully provide more relevant results to users. We introduce a review-based recommendation approach that obtains contextual information by mining user reviews. The proposed approach relate to features obtained by analysing textual reviews using methods developed in Natural Language Processing (NLP) and information retrieval discipline to compute a utility function over a given item. An item utility is a measure that shows how much it is preferred according to user's current context. In our system, the context inference is modelled as similarity between the user's reviews history and the item reviews history. As an example application, we used our method to mine contextual data from customer's reviews of technical products and use it to produce review-based rating prediction. The predicted ratings can generate recommendations that are item-based and should appear at the recommended items list in the product page. Our evaluations (surprisingly) suggest that our system can help produce better prediction rating scores in comparison to the standard prediction methods. As far as we know, all the recent works on recommendation techniques utilizing opinions inferred from user's reviews are either focused on the item recommendation task or use only the opinion information, completely leaving user's ratings out of consideration. The approach proposed in this report is filling this gap, providing a simple, personalized and scalable rating prediction framework utilizing both ratings provided by users and opinions inferred from their reviews. Experimental results provided on dataset containing user ratings and reviews from the real world Amazon and Flipkart Product Review Data show the effectiveness of the proposed framework.

1.3 Review of Literature

In real life, people's decision is often affected by friends action or recommendation. How to utilize social information has been extensively studied. Yang et al. propose the concept of "Trust Circles" in social network 6 based on probabilistic matrix factorization. Jiang et al. propose another important factor, the individual preference. some websites do not always offer structured information, and all of these methods do not leverage user's unstructured information, i.e. reviews, explicit social networks information is not always available and it is difficult to provide a good prediction for each user. For this problem the sentiment factor term is used to improve social recommendation. The rapid development of Web 2.0 and e-commerce has led to a proliferation in the number of online user reviews. Online reviews contain a wealth of sentiment information that is important for many decision-making processes, such as personal consumption decisions, commodity quality monitoring, and social opinion mining. Mining the sentiment and opinions that are contained in online reviews has become an important topic in natural language processing, machine learning, and Web mining.

1.4 Motivation for the Problem Undertaken

The project was first provided to me by FlipRobo as a part of the internship program. The exposure to real world data and the opportunity to deploy my skillset in solving a real time problem has been the primary objective. Many product reviews are not accompanied by a scale rating system, consisting only of a textual evaluation. In this case, it becomes daunting and time-consuming to compare different products in order to eventually make a choice between them. Therefore, models able to predict the user rating from the text review are critically important. Getting an overall sense of a textual review could in turn improve consumer experience. However, the motivation for taking this project was that it is relatively a new field of research. Here we have many options but less concrete solutions. The main motivation is to build a prototype of online hate and abuse review classifier which can used to classify hate and good comments so that it can be controlled and corrected according to the reviewer's choice.

2.Analytical Problem Framing

2.1 Mathematical/ Analytical Modeling of the Problem

In this perticular problem the Ratings can be 1, 2, 3, 4 or 5, which represents the likely ness of the product to the customer. So clearly it is a multi classification problem and I have to use all classification algorithms while building the model. We would perform one type of supervised learning algorithms: Classification. Here, we will only perform classification. Since there only 1 feature in the dataset, filtering the words is needed to prevent overfit. In order to determine the regularization parameter, throughout the project in classification part, we would first remove email, phone number, web address, spaces and stops words etc. In order to further improve our models, we also performed TFID in order to convert the tokens from the train documents into vectors so that machine can do further processing. I have used all the classification algorithms while building model then tunned the best model and saved the best model.

2.2 Data Sources and their formats

The data set contains nearly 1,14,491 samples with 3 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:

• Review_Title : Title of the Review.

• Review_Text : 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.

2.3 Data Preprocessing Done

Data pre-processing is the process of converting raw data into a well readable format to be used by Machine Learning model. Data pre-processing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn; therefore, it is extremely important that we pre-process our data before feeding it into our model. I have used following pre-processing steps:

✓ Importing necessary libraries and loading dataset as a data frame.

✓ Checked some statistical information like shape, number of unique values present, info, null values, value counts etc.

✓ Checked for null values and I replaced those null values using imputation method. And removed Unnamed: 0.

✓ Visualized each feature using seaborn and matplotlib libraries by plotting distribution plot and wordcloud for each ratings

. ✓ Done text pre-processing techniques like Removing Punctuations and other special characters, Splitting the comments into individual words, Removing Stop Words, Stemming and Lemmatization.
✓ After getting a cleaned data used TF-IDF vectorizer. It'll help to transform the text data to feature vector which can be used as input in our 6 modelling. It is a common algorithm to transform text into numbers. It measures the originality of a word by comparing the frequency of appearance of a word in a document with the number of documents the words appear in. Mathematically, TF-IDF = TF(t*d)*IDF(t,d)

✓ Balanced the data using SMOTE method.

2.4 Data Inputs-

Logic- Output Relationships The dataset consists of 2 features with a label. The features are independent and label is dependent as our label varies the values(text) of our independent variables changes.

• I checked the distribution of skewness using dist plots and used count plots to check the counts available in each column as a part of univariate analysis.
• Got to know the frequently occuring and rare occuring word with the help of count plot.

2.5 Hardware & Software Requirements & Tools Used

While taking up the project we should be familiar with the Hardware and software required for the successful completion of the project. Here we need the following hardware and software. Hardware required Software/s required Processor: core i5 RAM: 12 GB ROM/SSD: 512 GB Distribution: Anaconda Navigator Programming language: Python Browser based language shell: Jupyter Notebook Librariesrequired :-

✓ To run the program and to build the model we need some basic libraries

✓ import pandas as pd: pandas is a popular Python-based data analysis toolkit which can be imported using import pandas as pd. It presents a diverse range of utilities, ranging from parsing multiple file formats to converting an entire data table into a numpy matrix array. This makes pandas a trusted ally in data science and machine learning.

✓ import numpy as np: NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

✓ import seaborn as sns: Seaborn is a data visualization library built on top of matplotlib and closely integrated with pandas data structures in Python. Visualization is the central part of Seaborn which helps in exploration and understanding of data.

✓ Import matplotlib.pyplot as plt: matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc. With this sufficient libraries we can go ahead with our model building.

3.Data Analysis and Visualization

3.1 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.

3.2 Testing of Identified Approaches (Algorithms)

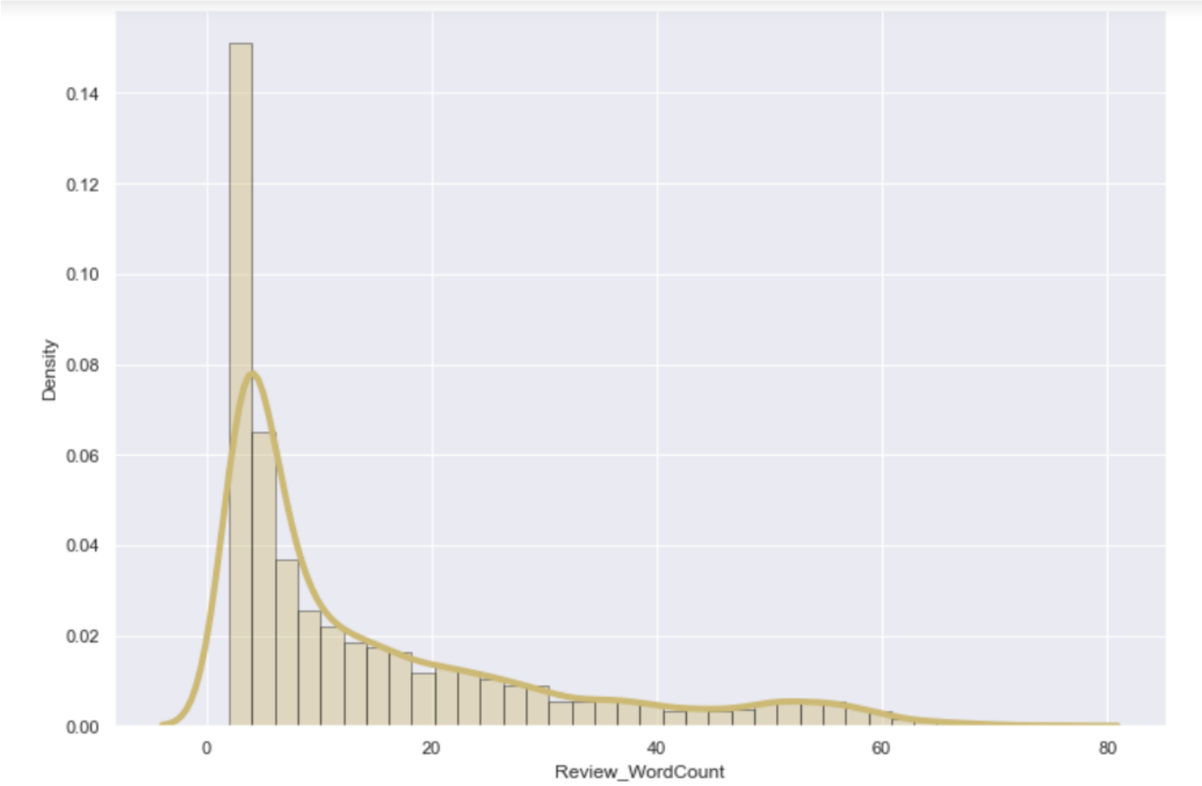
In this nlp based project we need to predict Ratings which is a multiclassification problem. I have converted the text into vectors using TFIDF vectorizer and separated our feature and labels then build the model using One Vs Rest Classifier. Among all the algorithms which I have used for this purpose 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 ➢ LinearSVC ➢ LogisticRegression ➢ RandomForestClassifier ➢ DecisionTreeClassifier ➢ XGBClassifier ➢ SGDClassifier From all of these above models SGDClassifier was giving me good performance with less difference in accuracy score and cv score.

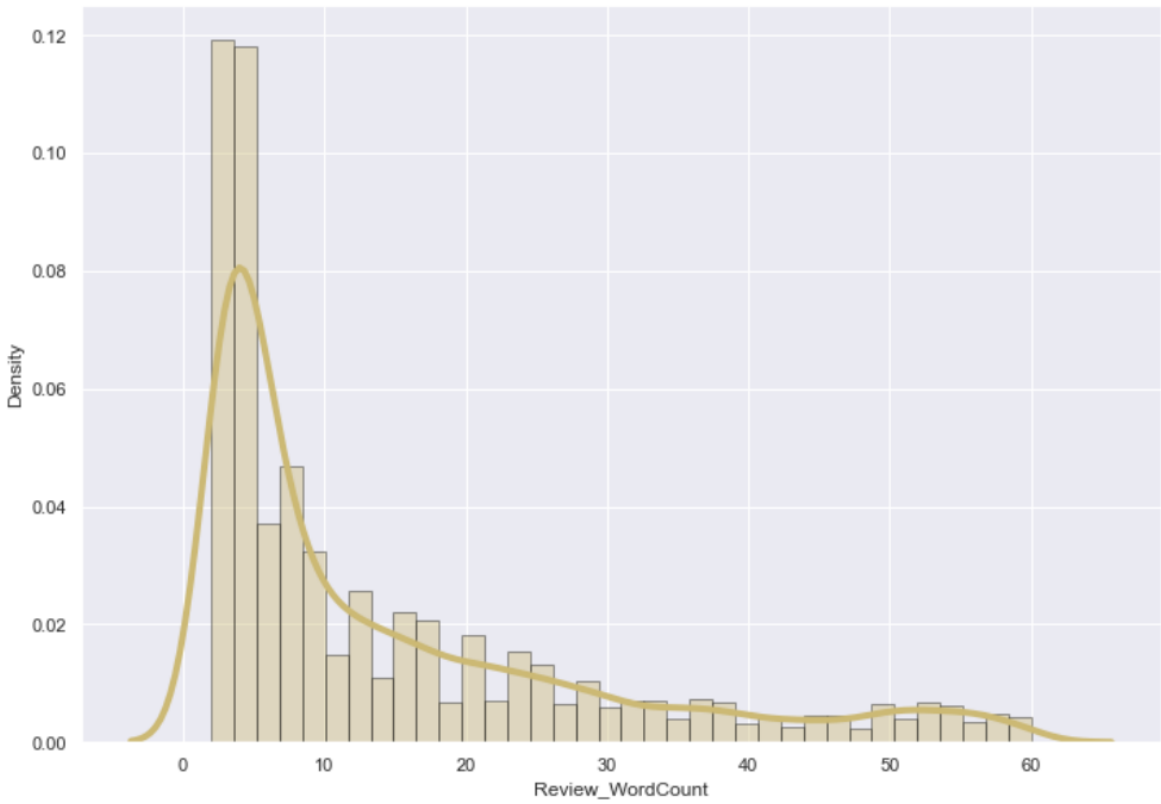
3.3 Key Metrics for success in solving problem under consideration

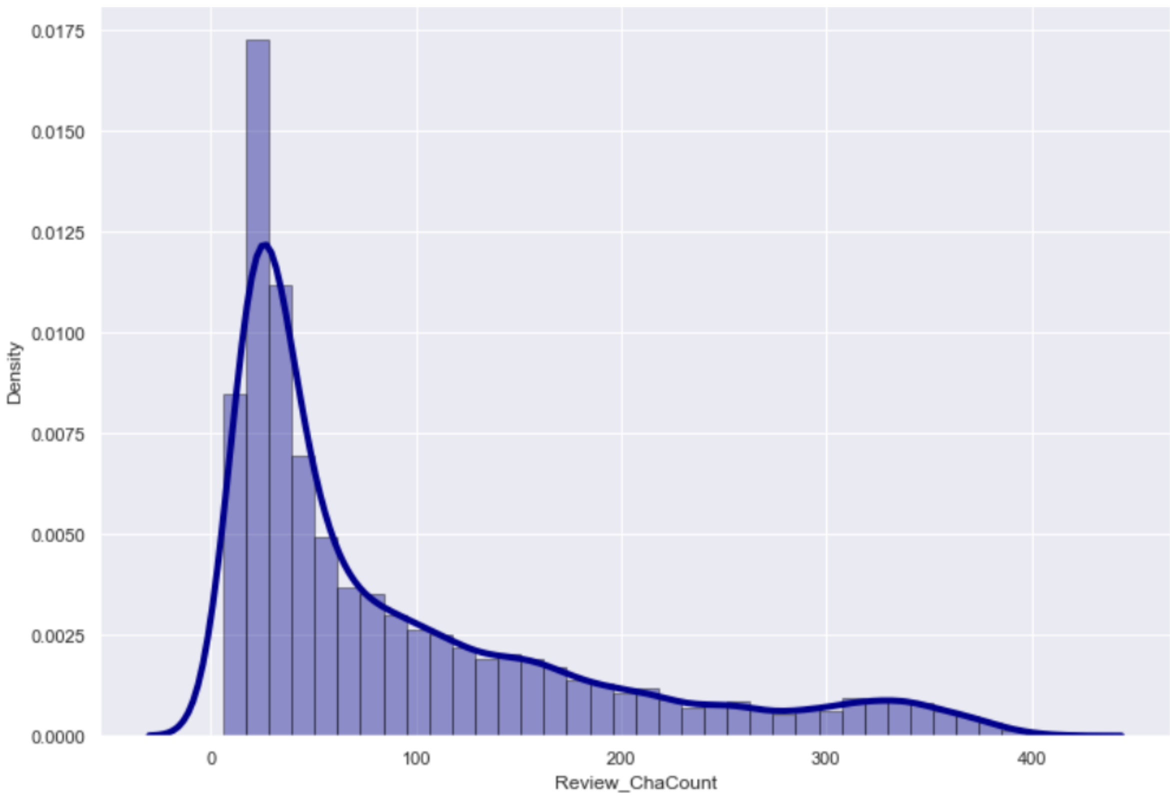
I have used the following metrics for evaluation: • I have used f1_score, precision_score, recall_score, multilabel_confusion_matrix and hamming loss all these evaluation metrics to select best suitable algorithm for our final model. • Precision can be seen as a measure of quality, higher precision means 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.

3.4 Visualizations

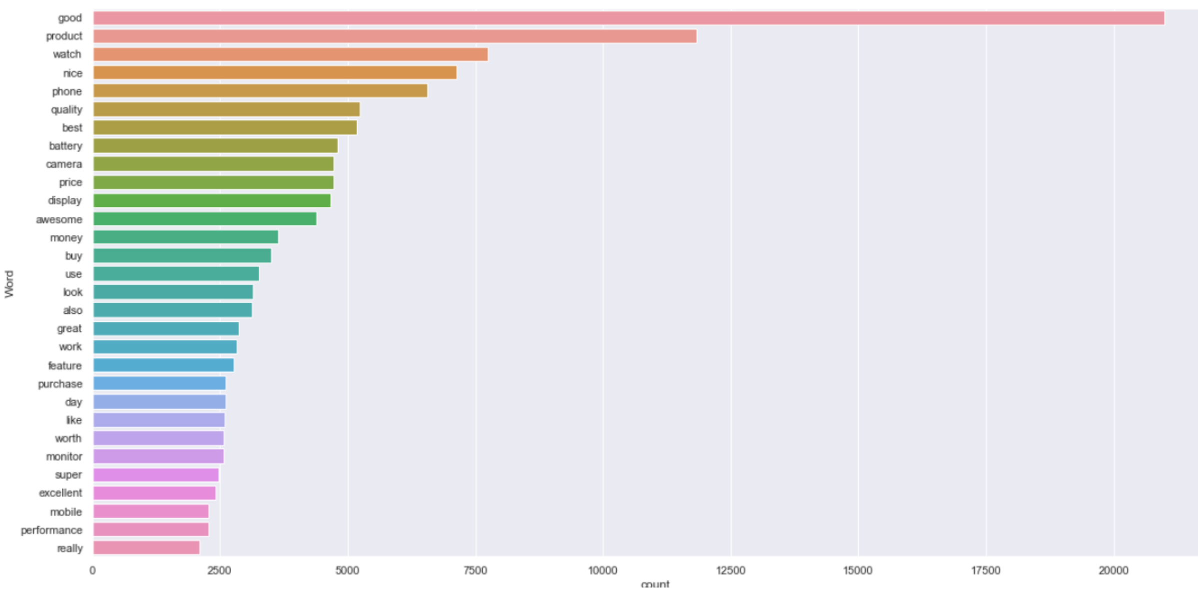
i)Plotting word count and character count using his plot:

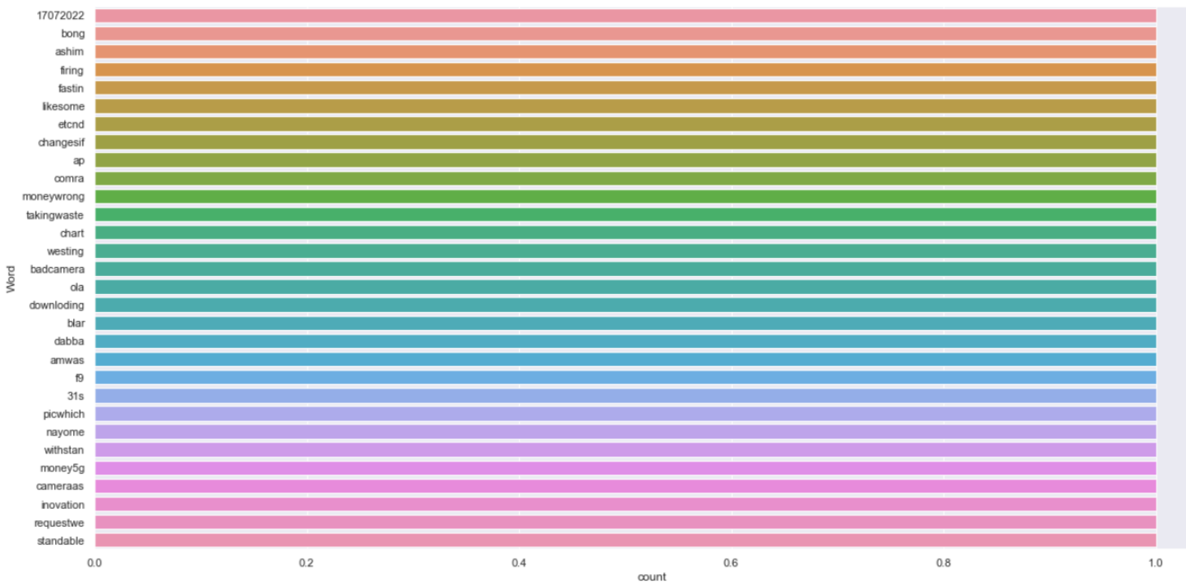



0.12
0.10
0.08
0.06
0.04
0.02
0.00
Density
0
10
20
30
40
50
60
Review_WordCount



ii)Top 30 most frequently occuring and rarely occuring words:





iii) Word Cloud for perticular ratings:

Rating 1:

Rating 2:

Rating 3:

Rating 4:

Rating 5:

1. Model Building:

I have used 6 classification algorithms. First, I have created 6 different classification algorithms and are appended in the variable models. Followed by TFIDF vectorization and data balancing. Then, ran a for loop which contained the accuracy of the models along with different evaluation metrics

Data Balancing:

```
#lets check the shapes of traning and test data
print("x_train", x_train.shape)
print("x_test", x_test.shape)
print("y_train", y_train.shape)
print("y_test", y_test.shape)
```

```
x_train (20685, 150000)
x_test (6895, 150000)
y_train (20685,)
y_test (6895,)
```

```
# Oversample and plot imbalanced dataset with SMOTE
from sklearn.datasets import make_classification
from imblearn.over_sampling import SMOTE

# transforming the dataset
os=SMOTE(sampling_strategy = {1: 37633, 2: 37633, 3: 37633, 4: 37633, 5: 37633})
x_train_ns,y_train_ns=os.fit_resample(x_train,y_train)

print("The number of classes before fit{}".format(Counter(y_train)))
print("The number of classes after fit {}".format(Counter(y_train_ns)))
```

```
The number of classes before fitCounter({5: 12743, 4: 4282, 1: 1931, 3: 1259, 2: 470})
The number of classes after fit Counter({1: 37633, 5: 37633, 3: 37633, 4: 37633, 2: 37633})
```

```
# Importing Libraries for ML Algorithms
from xgboost import XGBClassifier
from sklearn.svm import LinearSVC
#from lightgbm import LGBMClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import GradientBoostingClassifier, AdaBoostClassifier
from sklearn.naive_bayes import MultinomialNB, GaussianNB,BernoulliNB
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV
```

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier

# defining the algorithms
rf = RandomForestClassifier()
DTC = DecisionTreeClassifier()
svc = LinearSVC()
lr = LogisticRegression(solver='lbfgs')
mnb = MultinomialNB()
bnb = BernoulliNB()
xgb = XGBClassifier(verbosity=0)
#lgb = LGBMClassifier()
sgd = SGDClassifier()
```

```python
#creating a function to train and test the model with evaluation
def BuiltModel(model):
    print('*'*30+model.__class__.__name__+'*'*30)
    model.fit(x_train_ns,y_train_ns)
    y_pred = model.predict(x_train_ns)
    pred = model.predict(x_test)

    accuracy = accuracy_score(y_test,pred)*100

    print(f"Accuracy Score:", accuracy)
    print("---------------------------------------------------")

    #confusion matrix & classification report

    print(f"CLASSIFICATION REPORT : \n {classification_report(y_test,pred)}")
    print(f"Confusion Matrix : \n {confusion_matrix(y_test,pred)}\n")
```

Cross validation score:

```
# Defning function cross_val to find cv score of models
def cross_val(model):
    print('*'*30+model.__class__.__name__+'*'*30)
    scores = cross_val_score(model,train_features,y, cv = 5).mean()*100
    print("Cross validation score :", scores)
```

```
for model in [lr,svc,DTC,sgd,rf,xgb]:
    cross_val(model)
```

```
******************************LogisticRegression******************************
Cross validation score : 92.75927788766967
******************************LinearSVC******************************
Cross validation score : 94.22410774109929
******************************DecisionTreeClassifier******************************
Cross validation score : 89.5613674934008
******************************SGDClassifier******************************
Cross validation score : 94.22048494887584
******************************RandomForestClassifier******************************
Cross validation score : 92.44381929127474
******************************XGBClassifier******************************
```

HyperParameter Tuning:

```
# let's selects different parameters for tuning
grid_params = {
                'penalty':['l2','l1','elasticnet'],
                'loss':['hinge','squared_hinge'],
                'n_jobs':[-1,1]
                }
```

```
# Training the model with the given parameters using GridSearchCV
GCV =  GridSearchCV(sgd, grid_params, cv = 3, verbose=10)
GCV.fit(x_train_ns,y_train_ns)
```

```
▸ GridSearchCV
▸ estimator: SGDClassifier
    ▸ SGDClassifier
```

```
# Printing the best parameters found by GridSearchCV
GCV.best_params_
```

```
{'loss': 'squared_hinge', 'n_jobs': -1, 'penalty': 'l1'}
```

Final Model:

```
# Training our final model with above best parameters
model = SGDClassifier(loss = 'squared_hinge', n_jobs = -1, penalty = 'l1')
model.fit(x_train_ns,y_train_ns) #fitting data to model
pred = model.predict(x_test)
accuracy = accuracy_score(y_test,pred)*100

# Printing accuracy score
print("Accuracy Score :", accuracy)

# Printing Confusion matrix
print(f"\nConfusion Matrix : \n {confusion_matrix(y_test,pred)}\n")

# Printing Classification report
print(f"\nCLASSIFICATION REPORT : \n {classification_report(y_test,pred)}")
```

```
Accuracy Score : 97.4329224075417

Confusion Matrix :
 [[ 626    0    4    1    2]
 [   2  157    3    7    0]
 [   4    1  401   18    8]
 [   3    2   17 1319   70]
 [   3    2    2   28 4215]]


CLASSIFICATION REPORT :
               precision    recall  f1-score   support

           1       0.98      0.99      0.99       633
           2       0.97      0.93      0.95       169
           3       0.94      0.93      0.93       432
           4       0.96      0.93      0.95      1411
           5       0.98      0.99      0.99      4250

    accuracy                           0.97      6895
   macro avg       0.97      0.95      0.96      6895
weighted avg       0.97      0.97      0.97      6895
```

Model Saving:

```
import joblib
joblib.dump(model,"Ratings_RP.pkl")
```

```
['Ratings_RP.pkl']
```

Conclusion

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 different algorithms and using various evaluation metrics I have selected SGDClassifier for our final model. Finally by doing hyperparameter tuning we got optimum parameters for our final model. And finally we got good 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.