FLIP ROBO

Project Report On:

"Rating Prediction Project"

SUBMITTED BY

Mr.A.J.Wakchaure

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. Khushboo Garg (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 Mr. Sajid Choudhary (SME Flip Robo) who has inspired me in so many aspects and also encouraged me a lot with his valuable words and with his unconditional support I have ended up with a beautiful Project.

A huge thanks to my academic team “Data trained” who are the reason behind what I am today. Last but not least my parents who have been my backbone in every step of my life. And also thank you for many other persons who has helped me directly or indirectly to complete the project.

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. 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 33758 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:

- Review : 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.
- ✓ Visualized each feature using seaborn and matplotlib libraries by plotting distribution plot .
- ✓ 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 1 features with a label. The features are independent and label is dependent as our label varies the values(text) of our independent variables changes.

- Got to know the frequently occuring and rare occuring word with the help of count plot.

- And was able to see the words in the Review text with reference to there ratings using word cloud.

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 i53RAM: 4 GB ROM/SSD: 1 TB	Distribution: Anaconda Navigator Programming language: Python Browser based language shell: Jupyter Notebook

Libraries required :-

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

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import sent_tokenize,word_tokenize
import string
import re
from sklearn.datasets import make_classification
from imblearn.over_sampling import SMOTE
from sklearn.naive_bayes import MultinomialNB,GaussianNB
from sklearn.model_selection import train_test_split,cross_val_score,GridSearchCV
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report,
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier,

- ✓ **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.

<u>3.Data Analysis and Visualization</u>

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

- ➢ LogisticRegression
- ➢ RandomForestClassifier
- ➢ DecisionTreeClassifier
- ➢ KNN
- ➢ AdaBoost Classification

Testing Accuracy:

- Logistic Regression=71.79089026915114
- Random Forest.=75.38302277432712
- Decision Tree=71.5631469979296
- AdaBoost =70.53830227743271
- KNN=69.35817805383023

Cross Validation Score

- Logistic Regression.=68.87893998677743
- Random Forest.=68.05593942639399
- Decision Tree=63.257813187117584
- AdaBoost =68.56215909455796
- KNN=66.06221454497904

Compare Accuracy score and CV score

- Logistic Regression=71.790-68.878=2.912
- Random Forest.=75.383-68.055=7.32
- Decision Tree=71.563-63.257=8.30
- AdaBoost =70.538-68.562=1.97
- KNN=69.358-66.062=3.29

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

3.3 Hyperparameter Tuning

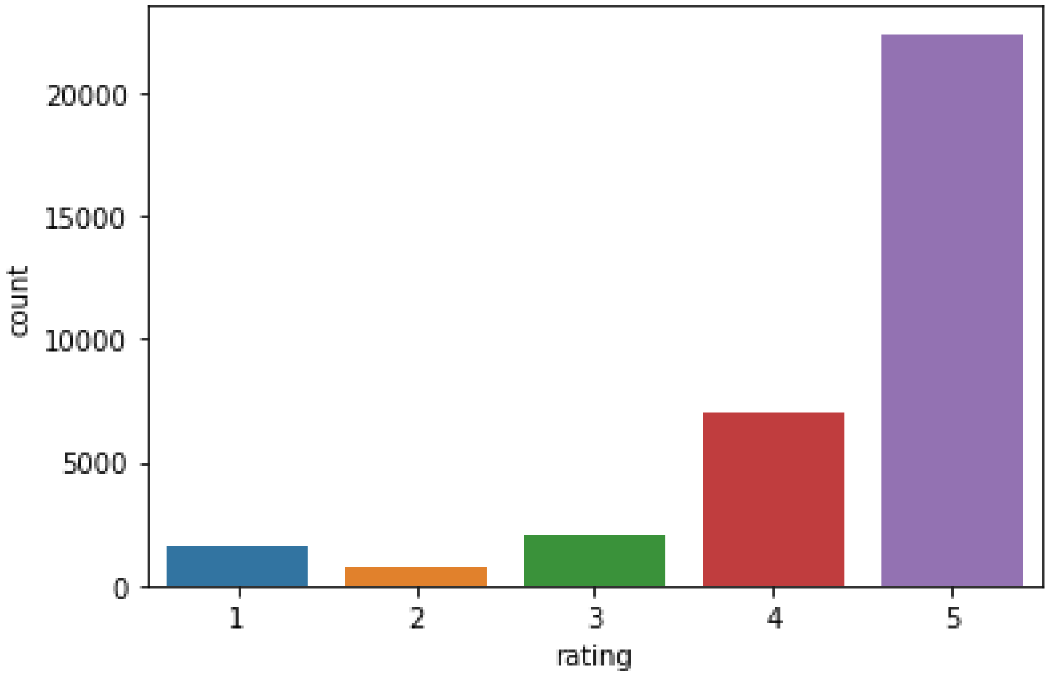
Hyperparameter tuning is used for getting improved accuracy,we can use Randomized Search CV or Grid search CV for hyperparameter tuning.

```
params={'n_estimators':[10,12,13],
    'max_depth':[10,15],
    'min_samples_leaf':[5,6],
    }
Final_RF=RandomForestClassifier(max_depth=10,min_samples_leaf=
5,n_estimators=10)
Final_RF.fit(x_train,y_train)
final_pred=Final_RF.predict(x_test)
final_score=accuracy_score(y_test,final_pred)
print(final_score*100)
```

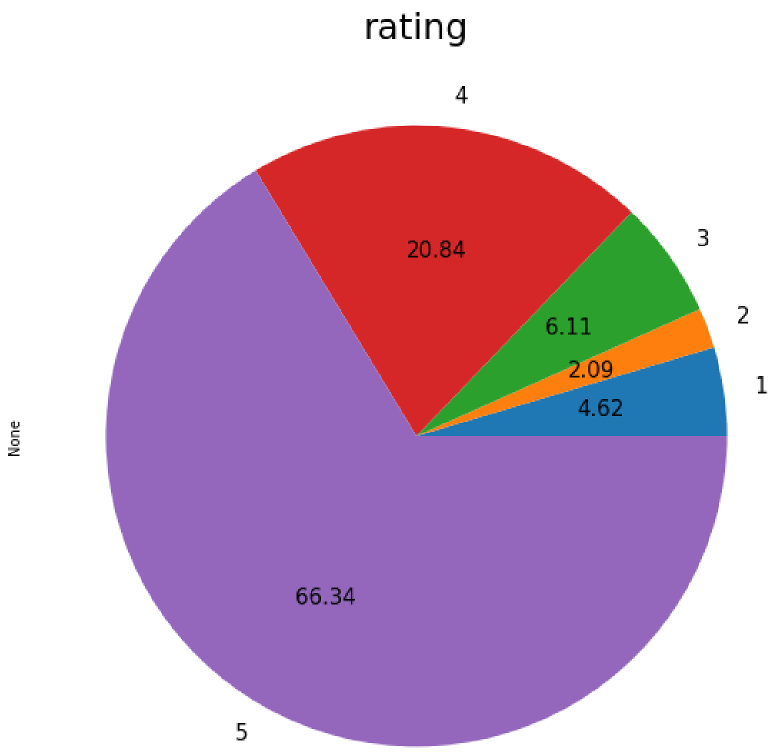
Here we more improvised accuracy score 70.41 which is treated as final accuracy score

3.4 Visualizations

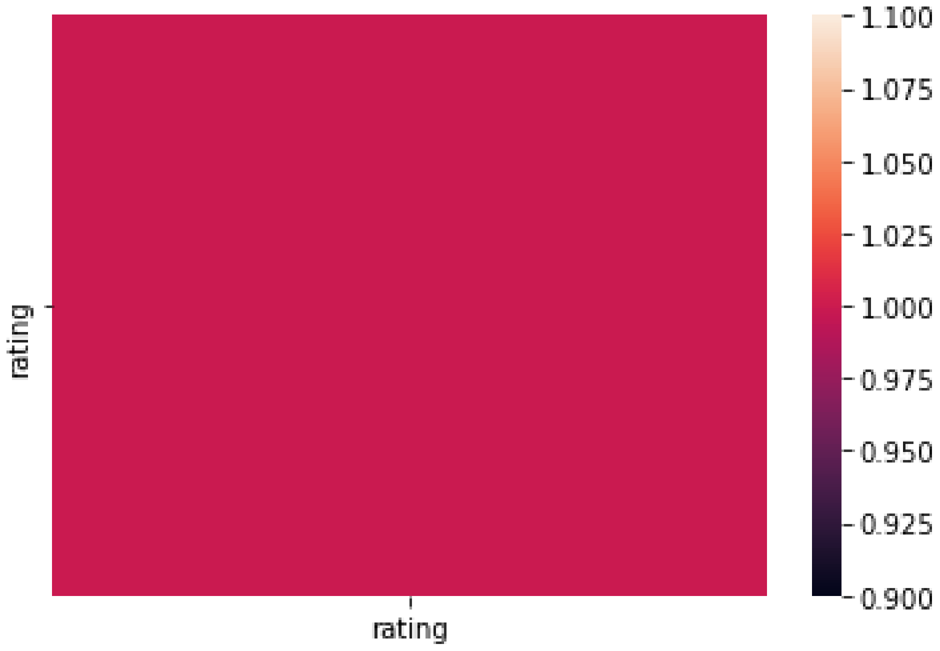
1) Count plot for Rating.



2) Pie Chart for Rating

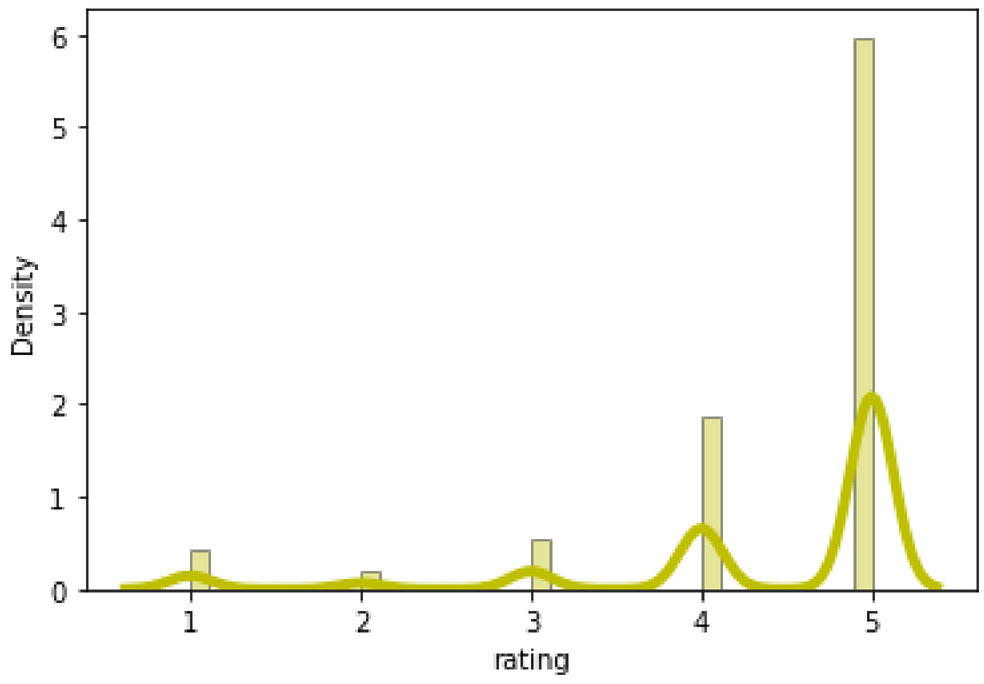


3) Multivariate Analysis



Observations:

- ✓ As we know that some of the review are too lengthy, so i have to treat them as outliers and remove them using z_score method. After removing the outliers the word count and character count looks as below.
- ✓ After plotting histograms for word counts and character counts and after removing outliers we can see we are left out with good range of number of words and characters.



By seeing the above plot we can see that Good, product, quality are occurring frequently. And the second plot shows rarely occuring words.

3.5 Run and Evaluate selected models

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.

Logistic Regression:

```
Training accuracy is 0.7431233362910381
Test accuracy is 0.7179089026915114
[[  11   34   41  127]
[   7   66   83  455]
[   2   42  228 1762]
[   0   23  149 6630]]
precision    recall  f1-score   support

2       0.55      0.05      0.09       213
3       0.40      0.11      0.17       611
4       0.46      0.11      0.18      2034
5       0.74      0.97      0.84      6802

accuracy                           0.72      9660
macro avg       0.54      0.31      0.32      9660
weighted avg       0.65      0.72      0.64      9660
```

Decision Tree Classifier:

```
Training accuracy is 0.889751552795031
Test accuracy is 0.715631469979296
[[  61   38   48   66]
[  26  189   94  302]
[  20   90  602 1322]
[  37  127  577 6061]]
         precision    recall  f1-score   support

         2       0.42      0.29      0.34       213
         3       0.43      0.31      0.36       611
         4       0.46      0.30      0.36      2034
        5       0.78      0.89      0.83      6802

accuracy                            0.72      9660
macro avg       0.52      0.45      0.47      9660
weighted avg    0.68      0.72      0.69      9660
```

Random Forest Classifier:

```
Training accuracy is 0.889751552795031
Test accuracy is 0.7538302277432712
[[  56   21   27  109]
[   9  174   48  380]
[   7   22  434 1571]
[   6   27  151 6618]]
             precision    recall  f1-score   support

        2       0.72      0.26      0.38       213
        3       0.71      0.28      0.41       611
        4       0.66      0.21      0.32      2034
        5       0.76      0.97      0.86      6802

 accuracy                            0.75      9660
 macro avg       0.71      0.43      0.49      9660
weighted avg     0.74      0.75      0.70      9660
```

AdaBoostClassifier:

```
Training accuracy is 0.6982253771073647
Test accuracy is 0.7053830227743271
[[  18   52    4  139]
[  16   63   34  498]
[  20   57   73 1884]
[  22   34   86 6660]]
        precision    recall  f1-score   support

         2       0.24      0.08      0.12       213
         3       0.31      0.10      0.15       611
         4       0.37      0.04      0.07      2034
         5       0.73      0.98      0.83      6802

accuracy                           0.71      9660
macro avg       0.41      0.30      0.29      9660
weighted avg    0.61      0.71      0.61      9660
```

KNeighborsClassifier:

```
Training accuracy is 0.7106921029281278
Test accuracy is 0.6935817805383023
[[  17   17   14  165]
[  38   45   42  486]
[  27   39  113 1855]
 [  40   36  201 6525]]
        precision    recall  f1-score   support

        2       0.14      0.08      0.10       213
        3       0.33      0.07      0.12       611
        4       0.31      0.06      0.09      2034
        5       0.72      0.96      0.82      6802

accuracy                              0.69      9660
macro avg          0.37      0.29      0.29      9660
weighted avg       0.60      0.69      0.61      9660
```

<u>2. Cross Validation score:</u>

- ✓ Great all our algorithms are giving good cv scores. Among these algorithms I am selecting SGD Classifier as best fitting algorithm for our final model as it is giving least difference between accuracy and cv 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,x,y, cv = 3).mean()*100
   print("Cross validation score :", scores)
for model in [LG,DT,RF,ada,knn]:
   cross_val(model)
```

```
******************************LogisticRegression******************************
Cross validation score : 68.87893998677743
******************************DecisionTreeClassifier******************************
Cross validation score : 63.257813187117584
******************************RandomForestClassifier******************************
Cross validation score : 68.05593942639399
******************************AdaBoostClassifier******************************
Cross validation score : 68.56215909455796
******************************KNeighborsClassifier******************************
Cross validation score : 66.06221454497904
```

<u>3. Hyper Parameter Tunning:</u>

I have did hyperparameter tuning for Random Forest Classifier for the parameters like 'n_estimator','max_depth','min_samples_leaf'.

```
params={'n_estimators':[10,12,13],
 'max_depth':[10,15],
'min_samples_leaf':[5,6],
 }
```

```
Final_RF=RandomForestClassifier(max_depth=10,min_samples_leaf= 5,n_estimators=10)
Final_RF.fit(x_train,y_train)
final_pred=Final_RF.predict(x_test)
final_score=accuracy_score(y_test,final_pred)
print(final_score*100
```

- ✓ And after doing hyperparameter tuning I got above parameters as best suitable parameters for our final model.
- ✓ I have trained my final model using these parameters and it was unable to increase the accuracy of the model.

- ✓ After training and building our final model I saved this model into .pkl file.

3. <u>Saving the model and Predictions</u>

```
import joblib
joblib.dump(RF,"Rating prediction using NLP.pkl")
```

4.CONCLUSION

4.1 Key Findings and Conclusions of the Study

- ✓ In this project I have collected data of reviews and ratings for different products from 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.
- ✓ 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 analysed 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 RF Classifier for our final model.
- ✓ Finally by doing hyperparameter tuning we got optimum parameters for our final model.

4.2 Learning Outcomes of the Study in respect of Data Science

I have scrapped the reviews and ratings of different technical products from flipkart.com and amazon.in websites. Improvement in computing technology has made it possible to examine social information that cannot previously be captured, processed and analysed. New analytical techniques(NLP) of machine learning can be used in property research. The power of visualization has helped us in understanding the data by graphical representation it has made me to understand what data is trying to say. Data cleaning is one of the most important steps to remove unrealistic values, punctuations, urls, email address and stop words. This study is an exploratory attempt to use 6 machine learning algorithms in estimating Rating, and then compare their results.

To conclude, the application of NLP in Rating classification is still at an early stage. We hope this study has moved a small step ahead in providing some methodological and empirical contributions to crediting institutes, and presenting an alternative approach to the valuation of Ratings.

4.3 Limitations of this work and Scope for 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.

While we couldn't reach out goal of maximum accuracy in Ratings prediction project, we did end up creating a system that can with some improvement and deep learning algorithms get very close to that goal. As with any project there is room for improvement here. The very nature of this project allows for multiple algorithms to be integrated together as modules and their results can be combined to increase the accuracy of the final result.

.

Thank you