FLIP ROBO

PROJECT REPORT ON:

"Rating Prediction Project"

SUBMITTED BY

Meet Mehta

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.

.

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 unit 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 provides more relevant results to users. We introduce a review-based recommendation approach that obtains contextual information by mining user reviews. The proposed approach relates 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 turned 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:

- Comments: Review 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 Pre-processing 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 word cloud for each rating.
- ✓ 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 1 label. The features are independent and label is dependent as our label varies the values(text) of our independent variables changes.

- I 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 i5 RAM: 12 GB ROM/SSD: 512 GB	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:

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score,roc_curve
from sklearn.impute import KNNImputer
from sklearn.utils import resample
from lightgbm import LGBMClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.semi_supervised import LabelSpreading,LabelPropagation
from sklearn.ensemble import AdaBoostClassifier
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis,LinearDiscriminantAnalysis
from sklearn.calibration import CalibratedClassifierCV
from sklearn.svm import LinearSVC,NuSVC
from sklearn.linear_model import RidgeClassifierCV,PassiveAggressiveClassifier,Perceptron
from sklearn.naive_bayes import GaussianNB,BernoulliNB
from sklearn.neighbors import NearestCentroid
from sklearn.dummy import DummyClassifier
from io import StringIO
from sklearn.tree import export_graphviz
import pydotplus
from IPython.display import Image
import os
from sklearn.feature_selection import SelectKBest,chi2
from scipy.stats import randint as sp_randint
from scipy.stats import uniform as sp_uniform
from skmultilearn.problem_transform import BinaryRelevance,ClassifierChain, LabelPowerset
from skmultilearn.adapt import MLkNN
from skmultilearn.cluster import LabelCooccurrenceGraphBuilder
from skmultilearn.ensemble import LabelSpacePartitioningClassifier
from skmultilearn.cluster.networkx import NetworkXLabelGraphClusterer
from sklearn.linear_model import SGDClassifier
```

```
from sklearn.multiclass import OneVsRestClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score,hamming_loss
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.preprocessing import MultiLabelBinarizer
import nltk
import matplotlib.pyplot as plt
import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.stem import SnowballStemmer
from os import path
from PIL import Image
import string
import collections
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
import matplotlib.cm as cm
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk import LancasterStemmer
from skmultilearn.ensemble import MajorityVotingClassifier
from skmultilearn.cluster import FixedLabelSpaceClusterer
from skmultilearn.adapt import BRkNNaClassifier,BRkNNbClassifier
from sklearn.multioutput import RegressorChain
from mlxtend.classifier import StackingClassifier
import joblib
import ipyparallel as ipp
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import pandas as pd
import matplotlib.pyplot as plt
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import LabelEncoder
import warnings
warnings.filterwarnings("ignore")
import seaborn as sn
import numpy as np
import scipy.stats as stats
from scipy.stats import zscore
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.metrics import f1_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
import random
from sklearn.model_selection import GridSearchCV,RandomizedSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import RidgeClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.linear_model import LassoLars
from sklearn.linear_model import BayesianRidge
```

- ✓ **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 these 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 different algorithm

- LogisticRegression()
- DecisionTreeClassifier()
- KNeighborsClassifier()
- RandomForestClassifier()
- SVC()
- RidgeClassifier()
- BaggingClassifier()
- GradientBoostingClassifier()
- SGDClassifier()
- LGBMClassifier()
- XGBClassifier()
- ExtraTreesClassifier()
- AdaBoostClassifier()
- CalibratedClassifierCV()
- LinearSVC()
- RidgeClassifierCV()

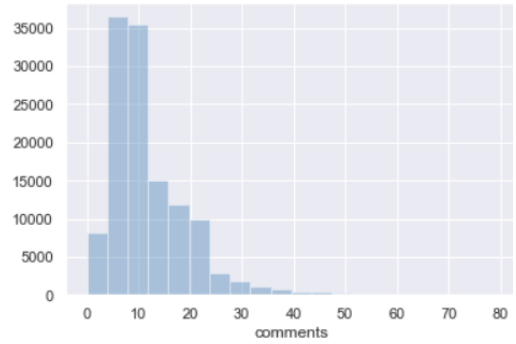
3.3 Key Metrics for success in solving problem under consideration

I have used the following metrics for evaluation:

- I have used accuracy score, cross_val_score, multilabel_confusion_matrix all these evaluation metrics to select best suitable algorithm for our final model.
- **Accuracy score** is used when the True Positives and True negatives are more important. Accuracy can be used when the class distribution is similar.

3.4 Visualizations

i)Plotting comment count using hist plot:



Observations:

- ✓ By observing the histogram we can clearly see that most of our text is having the number of comments in the range of 0 to 50, But some of the reviews are too lengthy which may act like outliers in our data.

ii) Word Cloud for particular ratings:

Rating 1:

Rating 2:

Rating 3:

Rating 4:

Rating 5:

- ✓ From the above plots we can clearly see the words which are indication of Reviewer's opinion on products.
- ✓ Here most frequent words used for each Rating is displayed in the word cloud.

3.5 Run and Evaluate selected models

1. Model Building:

I have used group of classification algorithms, ran a for loop which contained the accuracy of the models along with different evaluation metrics.

```
In [18]: models=[LogisticRegression(),DecisionTreeClassifier(),KNeighborsClassifier(),RandomForestClassifier(),SVC(),
                RidgeClassifier(),BaggingClassifier(),GradientBoostingClassifier(),SGDClassifier(),LGBMClassifier(),XGBClassifier()
                ExtraTreesClassifier(),AdaBoostClassifier(),CalibratedClassifierCV(),LinearSVC(),RidgeClassifierCV()]
```

```
LogisticRegression()
score 0.6586056875428004
Acc score 0.6592791134936192
[[3801  531  377  174  127]
 [ 588 2946  680  748  118]
 [ 240  493 3089  998  189]
 [ 109  170  887 3309  503]
 [  42   85  451 1007 3335]]
**********
DecisionTreeClassifier()
score 0.6618260980971329
Acc score 0.6645997519702365
[[3828  476  484   95  127]
 [ 551 3121 1134  145  129]
 [ 307  453 3834  239  176]
 [ 116  215 1576 2563  508]
 [  54  157  760  682 3267]]
**********
KNeighborsClassifier()
score 0.6355327331502966
Acc score 0.6355562667520103
[[3741  894  245   90   40]
 [ 685 3736  418  147   94]
 [ 387 1442 2802  259  119]
 [ 174 1123  696 2525  460]
 [ 257  629  305  646 3083]]
**********
RandomForestClassifier()
score 0.6673467996897205
Acc score 0.6714805776693203
[[3850  455  488   87  130]
 [ 530 3097 1162  149  142]
 [ 263  435 3886  245  180]
 [ 110  190 1547 2594  537]
 [  43  131  738  650 3358]]
**********
SVC()
score 0.6699671547391775
Acc score 0.6710805296635596
[[3871  476  475   77  111]
 [ 563 3059 1194  144  120]
 [ 247  473 3865  274  150]
 [ 102  202 1556 2608  510]
 [  38  125  738  647 3372]]
**********
RidgeClassifier()
score 0.6575155365213974
Acc score 0.6561187342481097
[[3770  556  371  183  130]
 [ 614 2929  661  752  124]
 [ 243  492 3070  998  206]
 [ 108  172  888 3276  534]
 [  52   87  453  972 3356]]
**********
BaggingClassifier()
score 0.6632462921243367
Acc score 0.6674000880105613
[[3819  484  493   83  131]
 [ 529 3094 1158  157  142]
 [ 277  435 3856  257  184]
 [ 102  201 1553 2594  528]
 [  51  130  744  675 3320]]
**********
GradientBoostingClassifier()
score 0.6387831066949271
Acc score 0.6395967516101932
[[3474 1227   55  165   89]
 [ 456 3642  168  733   81]
 [ 178 1262 2432 1021  116]
 [  65  710  492 3321  390]
 [  27  623  164  987 3119]]
**********
SGDClassifier()
score 0.6352525186008624
Acc score 0.6303956474776973
[[3803  498  189  421   99]
 [ 706 2830  302 1085  157]
 [ 310  479 2090 1858  272]
 [ 164  172  222 3760  660]
 [  77   70  230 1268 3275]]
**********
LGBMClassifier()
score 0.6611259900806303
Acc score 0.6612793535224227
[[3839  752  221   95  103]
 [ 608 3310  861  199  102]
 [ 238  922 3410  288  151]
 [ 103  442 1302 2647  484]
 [  42  354  530  670 3324]]
**********
[17:22:36] WARNING: C:/Users/Admi
1.3.0, the default evaluation met
licitly set eval_metric if you'd
XGBClassifier(base_score=0.5, boc
              colsample_bynode=1,
              importance_type='ga
              learning_rate=0.300
              min_child_weight=1,
              n_estimators=100, n
              objective='multi:so
              reg_lambda=1, scale
              tree_method='exact'
score 0.6619761005955074
Acc score 0.6622794735368244
[[3757  900  175   95   83]
 [ 531 3476  789  203   81]
 [ 211 1031 3358  301  108]
 [  79  533 1261 2667  438]
 [  27  406  501  689 3297]]
**********
```

```
ExtraTreesClassifier()
score 0.6683369237058725
Acc score 0.6719606352762332
[[3905  429  472   81  123]
 [ 555 3122 1128  147  128]
 [ 303  440 3855  240  171]
 [ 113  200 1561 2581  523]
 [  47  143  732  664 3334]]
**********
AdaBoostClassifier()
score 0.6187604678355518
Acc score 0.614553746449574
[[3185 1548   13  183   81]
 [ 399 3789   82  737   73]
 [ 143 1522 1716 1520  108]
 [  46  897   50 3619  366]
 [  11  770   11 1075 3053]]
**********
CalibratedClassifierCV()
score 0.6582856350344741
Acc score 0.6561187342481097
[[3734  566  395  183  132]
 [ 574 2952  677  741  136]
 [ 215  506 3087  996  205]
 [ 103  181  895 3261  538]
 [  41   87  459  966 3367]]
**********
```

```
LinearSVC()
score 0.6582756355347492
Acc score 0.6570388446613593
[[3784  537  377  182  130]
 [ 619 2917  664  752  128]
 [ 246  482 3076 1002  203]
 [ 111  171  885 3278  533]
 [  46   85  447  973 3369]]
**********
RidgeClassifierCV(alphas=array([ 0.1,  1. , 10. ]))
score 0.6574855335210973
Acc score 0.6561187342481097
[[3769  555  373  183  130]
 [ 612 2929  663  752  124]
 [ 242  490 3070 1001  206]
 [ 108  170  888 3279  533]
 [  51   86  455  974 3354]]
**********
```

- ✓ Great all our algorithms are giving good cv scores. Among these algorithms I am selecting ExtraTreesClassifier as best fitting algorithm for our final model as it is giving least difference between accuracy and cv score.

3. Hyper Parameter Tunning:

I have done hyperparameter tuning for **ExtraTreesClassifier** for the parameters like ‘criterion’, ‘max_features’, ‘max_depth’,’class_wieght’.

```
parameters={'criterion':['gini', 'entropy'],'max_features':['auto', 'sqrt', 'log2'],'max_depth':[2,8,16,32,50],'class_weigh
clf = RandomizedSearchCV(ExtraTreesClassifier(), parameters, cv=5,scoring="roc_auc",n_jobs=-1, verbose=1)
clf.fit(x,y)
clf.best_params_
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
```

```
{'max_features': 'sqrt',
 'max_depth': 32,
 'criterion': 'gini',
 'class_weight': 'balanced'}
```

Best Random State w.r.t Best performing Model

```
besttrain(ExtraTreesClassifier(max_depth=32,criterion='gini',max_features='sqrt',class_weight='balanced'),x,y)
```

```
maximum accuracy_score is at random state : 73  and it is : 0.6425571068528223
```

- ✓ 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.

```
Train accuracy 0.6557552481822637
Test accuracy 0.6434772172660719
Confusion matrix
 [[4019  153   27  136  702]
 [ 749 2395  111  693  953]
 [ 440  154 2360  938  976]
 [ 129   65  461 3314 1061]
 [  56   49  136  923 3997]]
classification report
              precision    recall  f1-score   support

           1       0.75      0.80      0.77      5037
           2       0.85      0.49      0.62      4901
           3       0.76      0.48      0.59      4868
           4       0.55      0.66      0.60      5030
           5       0.52      0.77      0.62      5161

    accuracy                           0.64     24997
   macro avg       0.69      0.64      0.64     24997
weighted avg       0.68      0.64      0.64     24997
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 c
```

```
cross_val_score: 0.6311520160367289
accuracy bigger
diff 0.012325201229342997
std: 0.002618881457316295
```

- ✓ After training and building our final model, the model accuracy drops so we choose model without hyperparameter tuning

```
Train accuracy 0.7335653634972547
Test accuracy 0.6781613793655239
Confusion matrix
 [[4019  153   27  136  702]
 [ 749 2395  111  693  953]
 [ 440  154 2360  938  976]
 [ 129   65  461 3314 1061]
 [  56   49  136  923 3997]]
classification report
              precision    recall  f1-score   support

           1       0.75      0.80      0.77      5037
           2       0.85      0.49      0.62      4901
           3       0.76      0.48      0.59      4868
           4       0.55      0.66      0.60      5030
           5       0.52      0.77      0.62      5161

    accuracy                           0.64     24997
   macro avg       0.69      0.64      0.64     24997
weighted avg       0.68      0.64      0.64     24997
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 c
```

```
cross_val_score: 0.6654565081451886
diff 0.021979290879116764
std: 0.001250640988611414
```

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

- I have saved my best model using .pkl as follows.

```
joblib.dump(finalmodel,"Ratings_Pred.pkl")
```

```
['Ratings_Pred.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 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.
- ✓ 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 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 SGDClassifier 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. This model can further be improved with the addition of more algorithms into it. However, the output of these algorithms needs to be in the same format as the others. Once that condition is satisfied, the modules are easy to add as done in the code. This provides a great degree of modularity and versatility to the project.