Rate Price Prediction project

Submitted by

Fenny Denny

ACKNOWLEDGMENT

This includes mentioning of all the references, research papers, data sources, professionals and other resources that helped me and guided me in completion of the project.

I wish to express my sincere gratitude to Miss. Khushboo Garg, SME for providing me an opportunity to do my internship and project work in “FLIP ROBO”.

It gives me immense pleasure in presenting this project report on “Micro Credit Defaulter Model”. It has been my privilege to have a team of project guide who have assisted me from the commencement of this project. The success of this project is a result of sheer hard work, and determination put in by me with the help of You Tube videos, references taken from Kaggle.com, skikit-learn.org.. To know more about micro finance, I read

https://www.geeksforgeeks.org/

https://github.com/

https://www.mckinsey.com/

https://www.counterpointresearch.com/

I hereby take this opportunity to add a special note of thanks for to Miss. Khushboo Garg, who undertook to act as my mentor despite his many other professional commitments. Her wisdom, knowledge and commitment to the highest standards inspired and motivated me. Without his insight, support this project wouldn’t have reached fruitfulness.

The project is dedicated to all those people of Fliprobo, Datatrained who helped me while doing this project.

INTRODUCTION

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

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'sreviews 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.

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.

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.

Analytical Problem Framing

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.

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.

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.

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. 9

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

Hardware & Software Requirements & Tools Used While taking up the project

Data Science task should be done with sophisticated machine with high end machine configuration. The machine which I'm currently using is powered by intel core i5 processor with 8GB of RAM. With this above-mentioned configuration, I managed to work with the data set in Jupyter Notebook which help us to write Python codes. As I'm using low configuration machine so it took more time then usual to execute codes. The library used for the assignment are Numpy, Pandas, Matplotlib, Seaborn, Scikit learn

Librariesrequired :-

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

```
# Preprocessing
import numpy as np
import pandas as pd

# Visualization
import seaborn as sns
import matplotlib.pyplot as plt
import os
import scipy as stats

import nltk
from nltk.corpus import stopwords
import re
import string
from nltk import FreqDist
from nltk.tokenize import word_tokenize
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import wordnet
from nltk import FreqDist

# Evaluation Metrics
from sklearn import metrics
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.metrics import roc_curve,accuracy_score,roc_auc_score,hamming_loss, log_loss

# Warning
import warnings
%matplotlib inline
warnings.filterwarnings('ignore')
```

```
from sklearn.svm import LinearSVC
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
```

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

➢ 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. 12 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.

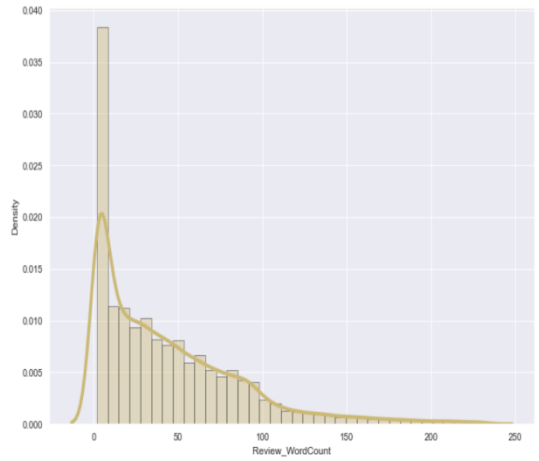
• Accuracy score is used when the True Positives and True negatives are more important. Accuracy can be used when the class distribution is similar.

• 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.

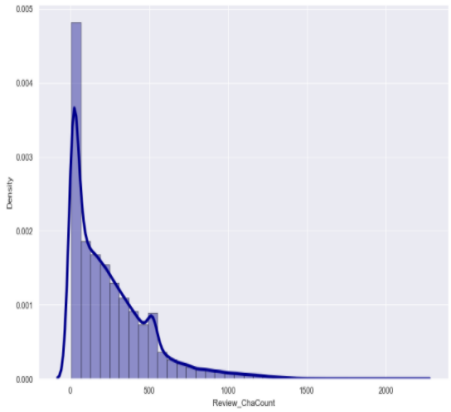
Visualizations

i)Plotting word count and character count using hist plot

```
# density plot and histogram of Review word count
sns.distplot(df['Review_WordCount'], hist = True, kde = True,
             bins = int(180/5), color = 'y',
             hist_kws = {'edgecolor':'black'},
             kde_kws = {'linewidth':4})
plt.show()
```




```
# density plot and histogram of all character count
sns.distplot(df['Review_ChaCount'], hist = True, kde = True,
             bins = int(180/5), color = 'darkblue',
             hist_kws = {'edgecolor':'black'},
             kde_kws = {'linewidth':4})
plt.show()
```



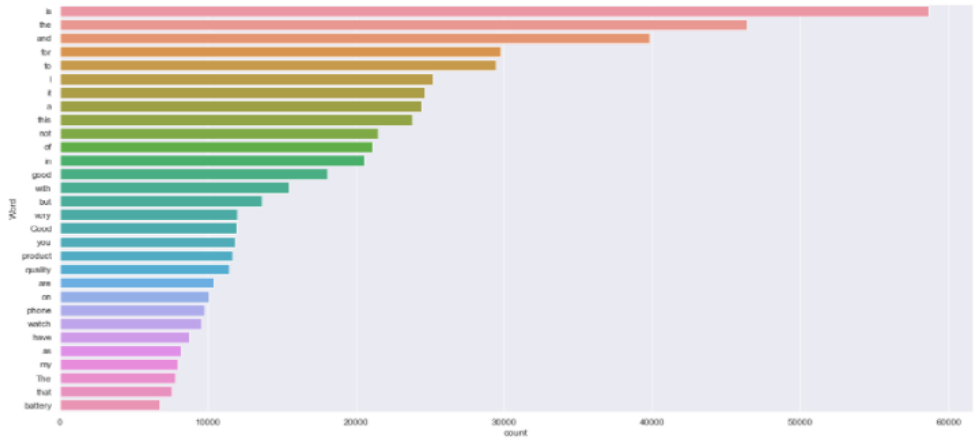
After plotting histograms for word counts and character counts and after removing outliers we can see we are left out with good range of number of words and characters.

Top 30 most frequently occuring words:

In [36]:
```
#importing nltk libraries
import nltk
from nltk.corpus import stopwords
import re
import string
from nltk import FreqDist
from nltk.tokenize import word_tokenize
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import wordnet
from nltk import FreqDist
```

In [37]:
```
#function to plot most frequent terms
def freq_words(x, terms = 30):
    all_words = ' '.join([text for text in x])
    all_words = all_words.split()
    fdist = FreqDist(all_words)
    words_df = pd.DataFrame({'word':list(fdist.keys()),
                             'count':list(fdist.values())})
    #selecting top 30 most freq words
    d = words_df.nlargest(columns = 'count', n = terms)
    plt.figure(figsize = (20,10))
    ax = sns.barplot(data = d, x='count', y='word')
    ax.set(ylabel = 'Word')
    plt.show()
```

In [38]:
```
freq_words(df['Review'])
```

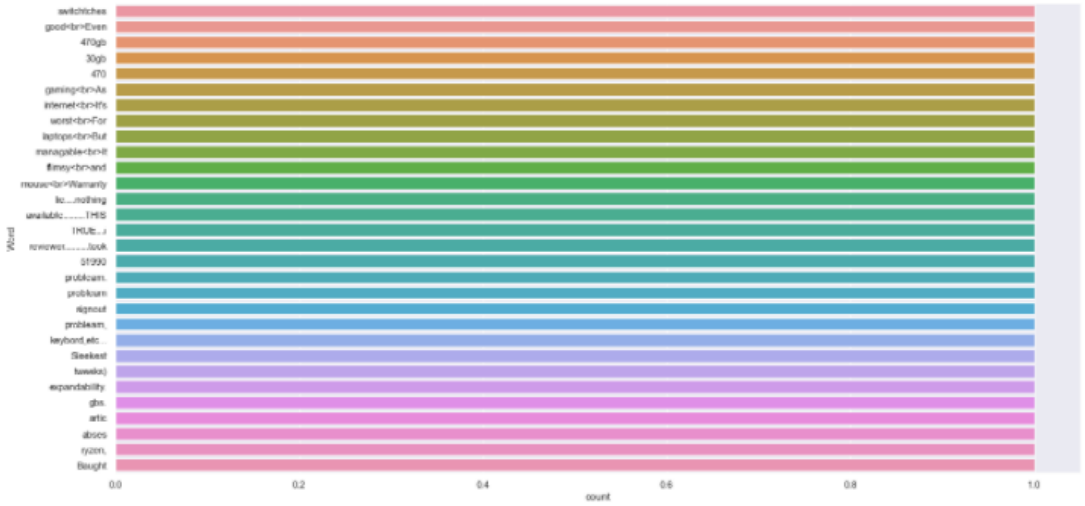


By seeing the above plot we can see that Good, prodout, quality.....are occurring frequently.

Top 30 Rare words:

n [39]:
```
#function to plot least frequent terms
def rare_words(x, terms = 30):
    all_words = ' '.join([text for text in x])
    all_words = all_words.split()
    fdist = FreqDist(all_words)
    words_df = pd.DataFrame({'word':list(fdist.keys()),
                             'count':list(fdist.values())})
    #selecting top 30 most freq words
    d = words_df.nsmallest(columns = 'count', n = terms)
    plt.figure(figsize = (20,10))
    ax = sns.barplot(data = d, x='count', y='word')
    ax.set(ylabel = 'Word')
    plt.show()
```

n [40]:
```
rare_words(df['Review'])
```



Above list of words are have rare occurance in Review

Run and Evaluate selected models

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.

```
The number of classes after fit Counter({5: 37833, 3: 37833, 1: 37833, 2: 37833, 4: 37833})
```

```python
from sklearn.svm import LinearSVC
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
# defining the algorithms
rf = RandomForestClassifier()
DTC = DecisionTreeClassifier()
svc = LinearSVC()
lr = LogisticRegression(solver='lbfgs')
mnb = MultinomialNB()
bnb = BernoulliNB()
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")
```

```python
# Running multiple algorithms
for model in [lr,svc,DTC,sgd,rf]:
    BuiltModel(model)
```

```
******************************LogisticRegression******************************
Accuracy Score: 67.95176320116938
--------------------------------------------------
CLASSIFICATION REPORT :
               precision    recall  f1-score   support

           1       0.67      0.72      0.70      1682
           2       0.39      0.45      0.42      1152
           3       0.47      0.40      0.43      1401
           4       0.55      0.60      0.57      1946
           5       0.89      0.83      0.86      4765

    accuracy                           0.68     10946
   macro avg       0.59      0.60      0.60     10946
weighted avg       0.69      0.68      0.68     10946

Confusion Matrix :
 [[1210  315   95   45   17]
 [ 334  524  188   74   32]
 [ 156  326  567  273   79]
 [  41  128  237 1160  380]
 [  54   59  120  555 3977]]

******************************LinearSVC******************************
Accuracy Score: 67.87867714233509
--------------------------------------------------
CLASSIFICATION REPORT :
               precision    recall  f1-score   support

           1       0.68      0.71      0.70      1682
           2       0.38      0.37      0.38      1152
           3       0.44      0.44      0.44      1401
           4       0.57      0.58      0.57      1946
           5       0.86      0.85      0.86      4765

    accuracy                           0.68     10946
   macro avg       0.59      0.59      0.59     10946
weighted avg       0.68      0.68      0.68     10946
```

```
Confusion Matrix :
 [[1201  269  131   51   30]
 [ 329  429  256   85   53]
 [ 150  263  616  253  119]
 [  42  103  246 1121  434]
 [  46   56  145  455 4063]]

*****************************DecisionTreeClassifier*****************************
Accuracy Score: 60.679700347158786
--------------------------------------------------
CLASSIFICATION REPORT :
               precision    recall  f1-score   support

           1       0.58      0.58      0.58      1682
           2       0.30      0.39      0.34      1152
           3       0.37      0.36      0.36      1401
           4       0.51      0.50      0.50      1946
           5       0.83      0.79      0.81      4765

    accuracy                           0.61     10946
   macro avg       0.52      0.52      0.52     10946
weighted avg       0.62      0.61      0.61     10946

Confusion Matrix :
 [[ 982  379  167   79   75]
 [ 289  447  194  133   89]
 [ 208  302  501  207  183]
 [ 110  181  256  966  433]
 [ 112  177  234  496 3746]]

*****************************SGDClassifier*****************************
Accuracy Score: 67.2209026128266
--------------------------------------------------
CLASSIFICATION REPORT :
               precision    recall  f1-score   support

           1       0.64      0.76      0.70      1682
           2       0.35      0.51      0.41      1152
           3       0.54      0.29      0.38      1401
           4       0.54      0.61      0.57      1946
           5       0.90      0.82      0.86      4765

    accuracy                           0.67     10946
   macro avg       0.59      0.60      0.58     10946
weighted avg       0.69      0.67      0.67     10946

Confusion Matrix :
 [[1276  308   38   40   20]
 [ 367  582   92   87   24]
 [ 189  445  405  289   73]
 [  70  216  150 1178  332]
 [  80  111   67  590 3917]]

*****************************RandomForestClassifier*****************************
Accuracy Score: 68.08879956148365
--------------------------------------------------
CLASSIFICATION REPORT :
               precision    recall  f1-score   support

           1       0.64      0.75      0.69      1682
           2       0.38      0.49      0.43      1152
           3       0.56      0.31      0.40      1401
           4       0.57      0.58      0.58      1946
           5       0.86      0.85      0.85      4765

    accuracy                           0.68     10946
   macro avg       0.60      0.60      0.59     10946
weighted avg       0.69      0.68      0.68     10946

Confusion Matrix :
 [[1263  285   51   43   40]
 [ 353  570   99   80   50]
 [ 196  399  436  244  126]
 [  67  168  125 1132  454]
 [  80   79   67  487 4052]]
```

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 = 3).mean()*100
    print("Cross validation score :", scores)
```

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

```
******************************LogisticRegression******************************
Cross validation score : 59.950444986217065
******************************LinearSVC******************************
Cross validation score : 58.86521662502546
******************************DecisionTreeClassifier******************************
Cross validation score : 50.927553170680554
******************************SGDClassifier******************************
Cross validation score : 60.06634942720653
******************************RandomForestClassifier******************************
Cross validation score : 58.5613730104476
```

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, scoring='accuracy',n_jobs=-1,verbose=2)
```

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

```
Fitting 3 folds for each of 12 candidates, totalling 36 fits
```

77]:
```
GridSearchCV(cv=3, estimator=SGDClassifier(), n_jobs=-1,
             param_grid={'loss': ['hinge', 'squared_hinge'], 'n_jobs': [-1, 1],
                         'penalty': ['l2', 'l1', 'elasticnet']},
             scoring='accuracy', verbose=2)
```

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

78]:
```
{'loss': 'hinge', 'n_jobs': -1, 'penalty': 'l2'}
```

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 : 63.89017716352843


Confusion Matrix :
 [[1014  302  162   96   62]
 [ 297  370  238  136   86]
 [ 173  238  546  265  175]
 [  68  108  253 1014  443]
 [  69   75  200  447 3944]]


CLASSIFICATION REPORT :
              precision    recall  f1-score   support

           1       0.63      0.62      0.62      1636
           2       0.34      0.33      0.33      1127
           3       0.39      0.39      0.39      1397
           4       0.52      0.54      0.53      1886
           5       0.84      0.83      0.84      4735

    accuracy                           0.64     10781
   macro avg       0.54      0.54      0.54     10781
weighted avg       0.64      0.64      0.64     10781
```

Saving the model and Predictions:

I have saved my best model using .pkl as follow

Model Saving:

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

```
30]: ['Ratings_RP.pkl']
```

Finally I have saved the model into .pkl file.

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.

✓ 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 SGDClassifier 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

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 22 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.

Thank you

****