Project Report on
“Ratings Prediction Project”

Project by:

Adi Rajit Mahesh

Acknowledgment:

It is a Great pleasure to express my Gratitude to Team FlipRobo, for giving me opportunity to work on a fantastic project, which helped me in improving my knowledge, coding skills and my analyzation skills.

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.

Background

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 Product Review Data show the effectiveness of the proposed framework

Motivation:

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:

In this particular 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.

Sources:

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:

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

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

Hardware & Software Requirements

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

Processor: core i5
RAM: 8 GB
ROM/HDD: 1 TB

Software/s required

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

Libraries required:

```
# lets import all required libraries for the analysis
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
import scipy as stats
import nltk
from nltk.corpus import stopwords
from nltk import FreqDist
from nltk.tokenize import word_tokenize
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import wordnet
import re
import string
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

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

✓ ***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

Data Analysis and Visualization:

Problem solving approach:

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.

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.

Key Metrics:

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

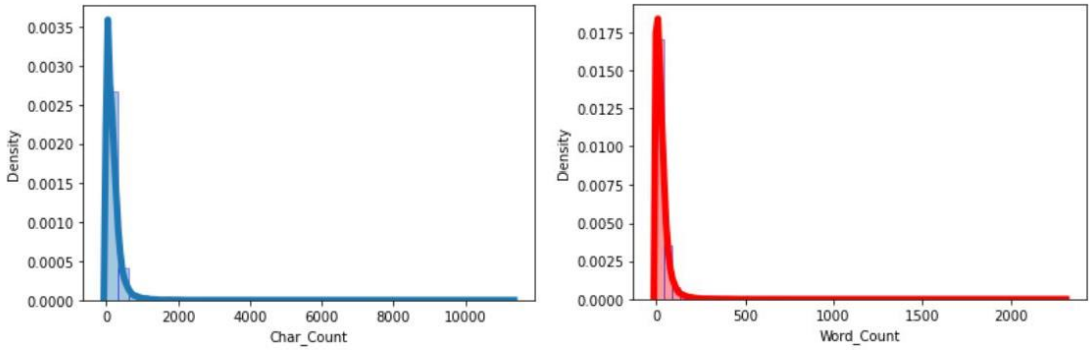
important. Accuracy can be used when the class distribution is similar.

- ***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:

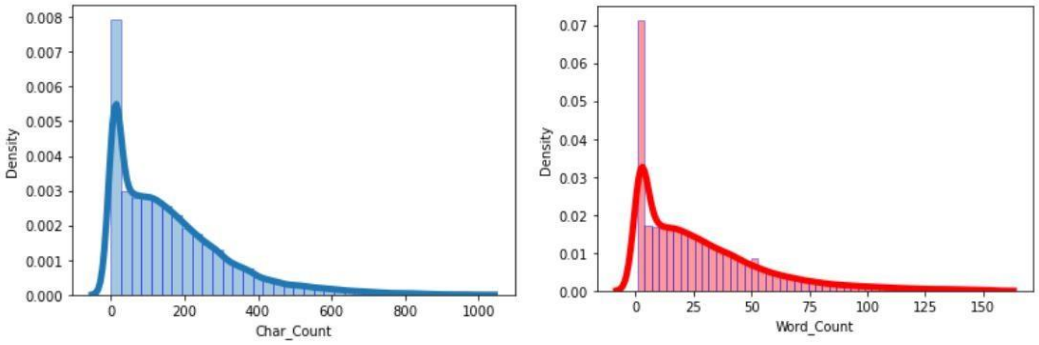


Observations:

✓ By observing the histogram we can clearly see that most of our text is having the number of words in the range of 0 to 200, But some of the reviews are too lengthy which may act like outliers in our data.

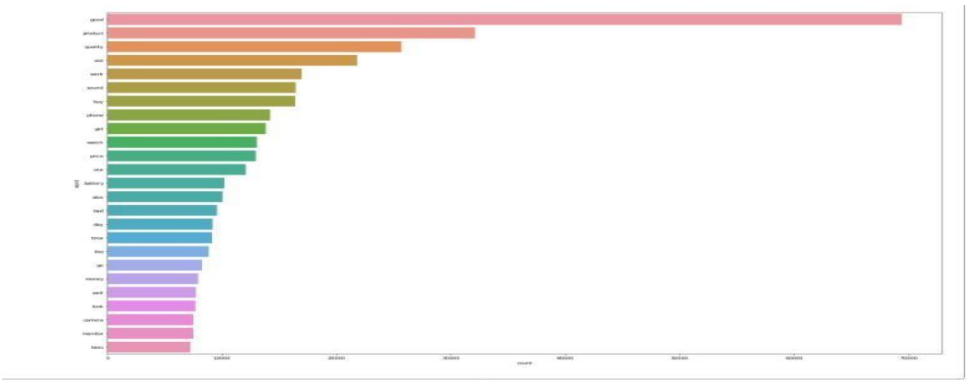
✓ Above plot represents histogram for character count of Review text, which is quite similar to the histogram of word count.

✓ 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.

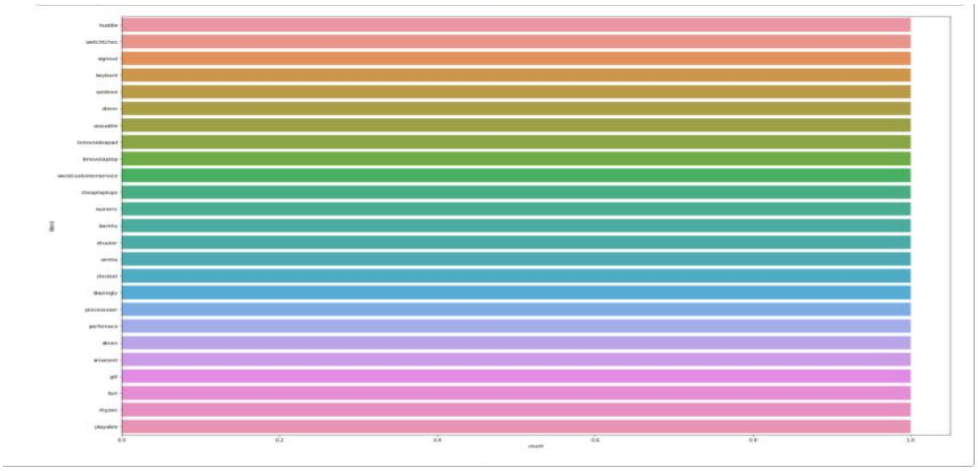


- Here we can clearly see that after removing outliers we can clearly visualize the good range of Count of Characters and Words.

ii) Top 25 most frequently occuring and rarely occuring words:



- After observing above plot we can clearly see that Good is the most frequent word used in reviews, followed by Product, Quality and Use.



- Here we can see the list of all the characters which are used the least or in this case these words are used only 1.

iii) Word Cloud for perticular ratings:

Rating 1:

Rating 2:

Rating 3:

Rating 4:

Rating 5:

✓ We can see the words which are indication of Reviewer's opinion on products.

✓ Most frequent words used for each Rating is displayed in the word cloud.

Model Selection:

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.

In [78]:
```
rfc = RandomForestClassifier()
dtc = DecisionTreeClassifier()
svc = LinearSVC()
lr = LogisticRegression(solver='lbfgs')
mnb = MultinomialNB()
bnb = BernoulliNB()
xgb = XGBClassifier(verbosity=0)
lgb = LGBMClassifier()
sgd = SGDClassifier()
```

In [79]:
```
#lets create a function to train and test the model with evalua
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
    print(f"Confusion Matrix : \n {confusion_matrix(y_test,pred
```

```
In [80]:
for model in [lr,dtc,svc,sgd,rfc,xgb]:
    BuiltModel(model)
```

**********LogisticRegression**********

Accuracy Score: 66.43572930842508

CLASSIFICATION REPORT :

	precision	recall	f1-score	support
1	0.68	0.76	0.72	2853
2	0.52	0.47	0.50	2188
3	0.50	0.49	0.50	2405
4	0.52	0.56	0.54	2783
5	0.85	0.81	0.83	5937
accuracy			0.66	16166
macro avg	0.61	0.62	0.62	16166
weighted avg	0.67	0.66	0.66	16166

**********DecisionTreeClassifier**********

Accuracy Score: 62.66856365210936

CLASSIFICATION REPORT :

	precision	recall	f1-score	support
1	0.65	0.65	0.65	2853
2	0.50	0.52	0.51	2188
3	0.47	0.50	0.48	2405
4	0.50	0.51	0.51	2783
5	0.80	0.76	0.78	5937
accuracy			0.63	16166
macro avg	0.58	0.59	0.59	16166
weighted avg	0.63	0.63	0.63	16166

**********LinearSVC**********

Accuracy Score: 68.97810218978103

CLASSIFICATION REPORT :

	precision	recall	f1-score	support
1	0.70	0.74	0.72	2853
2	0.57	0.55	0.56	2188
3	0.54	0.53	0.53	2405
4	0.56	0.57	0.56	2783
5	0.85	0.84	0.84	5937
accuracy			0.69	16166
macro avg	0.64	0.65	0.65	16166
weighted avg	0.69	0.69	0.69	16166

**********SGDClassifier**********

Accuracy Score: 65.63157243597674

CLASSIFICATION REPORT :

	precision	recall	f1-score	support
1	0.59	0.85	0.70	2853
2	0.56	0.36	0.43	2188
3	0.58	0.39	0.47	2405
4	0.50	0.57	0.53	2783
5	0.83	0.82	0.83	5937
accuracy			0.66	16166
macro avg	0.61	0.60	0.59	16166
weighted avg	0.66	0.66	0.65	16166

**********RandomForestClassifier**********

Accuracy Score: 69.22553507361128

CLASSIFICATION REPORT :

	precision	recall	f1-score	support
1	0.67	0.83	0.74	2853
2	0.64	0.49	0.56	2188
3	0.61	0.52	0.56	2405
4	0.52	0.67	0.59	2783
5	0.87	0.78	0.82	5937
accuracy			0.69	16166
macro avg	0.66	0.66	0.65	16166
weighted avg	0.71	0.69	0.69	16166

**********XGBClassifier**********

Accuracy Score: 65.72435976741309

CLASSIFICATION REPORT :

	precision	recall	f1-score	support
1	0.67	0.76	0.71	2853
2	0.49	0.45	0.47	2188
3	0.51	0.45	0.48	2405
4	0.51	0.57	0.54	2783
5	0.84	0.80	0.82	5937
accuracy			0.66	16166
macro avg	0.61	0.61	0.61	16166
weighted avg	0.66	0.66	0.66	16166

Cross Validation Scores:

```
# lets check cross validation scores of all models as well
def cross_val(model):
    print('*'*25+model.__class__.__name__+'*'*25)
    score=cross_val_score(model,train_features,y,cv=3).mean()*1
    print("Cross Validation Score : ",score)
```

```
for model in [lr,dtc,svc,sgd,rfc,xgb]:
    cross_val(model)
```

```
**********LogisticRegression**********
Cross Validation Score :  58.09504277981278
**********DecisionTreeClassifier**********
Cross Validation Score :  48.41719984082818
**********LinearSVC**********
Cross Validation Score :  55.993392215542805
**********SGDClassifier**********
Cross Validation Score :  57.895548725685906
**********RandomForestClassifier**********
Cross Validation Score :  56.53620283999719
**********XGBClassifier**********
Cross Validation Score :  57.567695323499414
```

- After observing scores of all the models. we found least difference between accuracy and cv score in SGD Classifiers, so, we will take it as our final model and now we will hyper tune our final model.

Hyper Parameter Tuning:

In [83]:
```
# let's select parameters for hyper tuning of our best model
g_params = {
            'penalty':['l2','l1','elasticnet'],
            'loss':['hinge','squared_hinge'],
            'n_jobs':[-1,1]}
```

In [*]:
```
# Lets train our model with the given parameters using GridSear
gcv =  GridSearchCV(sgd, g_params, cv =3 , verbose=10)
gcv.fit(x_train_ns,y_train_ns)
```

In [*]:
```
# lets search for best parameters found by GridSearchCV
GCV.best_params_
```

Conclusion:

✓ In this project I have collected data of reviews and ratings for different products from amazon.in .

✓ 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 performed 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 we analyzed the text.

✓ We have checked frequently occurring words in our data as well as rarely occurring words.

✓ We removed the outilers which were present in our extracted data.

✓ In Model Selection Process we tried 6 models and based on accuracy score and cross validation score, we selected our best model.

✓ 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.