



RATINGS PREDICTION USING NLP

Submitted by:

VANISREE P G

ACKNOWLEDGMENT

First I would like to thank the Almighty for his wonderful presence with me throughout this project and helped me to make it as a successful one.

For my internship I had the pleasure of working at FILP ROBO Was a great chance for acquired knowledge, personal and Professional development.

I extend whole hearted thanks to FILP ROBO under whom I worked and learned a lot and for enlightening me with their knowledge and experience to grow with the corporate working.

This is a great pleasure to express my deep sense of gratitude and thanks to SME for his valuable ideas, instantaneous help, effective support and continued encouragement which enabled for the successful completion of the project. I also like to thank the data trained mentors and Technical team members for helping me with technical queries.

And these are the following website which I referred for the reference

1. <https://www.kaggle.com/>
2. <https://scikit-learn.org/>
3. www.stackoverflow.com
4. www.google.com
5. www.geeksforgeeks.org

INTRODUCTION

➤ Business Problem Framing

In recent years, the importance of contextual information has motivated generation of personalized recommendations according to the available contextual information of users. In recent years, we have witnessed a flourish of review websites. It presents a great opportunity to share our view points for various products we purchase. However, we face an information overloading problem. How to mine valuable information from reviews to understand a user's preferences and make an accurate recommendation is crucial. Traditional recommender systems consider some factors, such as user's purchase records, product category, and geographic location. In this work, we propose a sentiment-based rating prediction method to improve prediction accuracy in recommender systems. Firstly, we propose a social user sentimental measurement approach and calculate each user's sentiment on items/products. Secondly, we not only consider a user's own sentimental attributes but also take interpersonal sentimental influence into consideration. Then, we consider product reputation, which can be inferred by the sentimental distributions of a user set that reflect customers' comprehensive evaluation. At last, we fuse three factors user sentiment similarity, interpersonal sentimental influence, and item's reputation similarity into our recommender system to make an accurate rating

prediction. We conduct a performance evaluation of the three sentimental factors on a real-world dataset collected from Yelp. Our experimental results show the sentiment can well characterize user preferences, which helps to improve the recommendation performance

The text of a review is often overlooked in such predictive tasks in favour of features such as the user's and business previous rating history. However, if the sentiment of the text of a user's review can be estimated suitably, it would be what the opinion of the user is about the business in his own words and not a mathematical predictive task. Thus, it is essential to be able to predict what the user feels about a business from the review text and this is the task of rating prediction from review text was chosen.

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 a rating. So, we have to build an application which can predict the rating by seeing the review.

➤ **Conceptual Background of the Domain Problem**

Online product reviews have strong and influential impact on the consumers as they tend to use the reviews for purchasing decisions. The drawback is that a popular product usually has too many reviews for the consumers to read. Additionally, new products might have too few reviews. Thus, the review text, ranking, and rating must be recommended to the consumers. Specifically, review helpfulness plays a vital role in product rankings and recommendations. Another major aspect of modelling these reviews is to capture effective feature information from the dataset. We can relate our problem with finding the truthfulness of any comment used in online platforms as well. In truth assessment and semantic analysis for product review text is performed on Flipkart.com data.

In their research process, consumers want to find useful information as quickly as possible. However, searching and comparing text reviews can be frustrating for users as they feel submerged with information (Ganu, Elhada & Marian, 2009). Indeed, the massive amount of text reviews as well as its unstructured text format prevent the user from choosing a product with ease. The star-rating, i.e. stars from 1 to 5 on Amazon, rather than its text content gives a quick overview of the product quality. This numerical information is the number one factor used in an early phase by consumers to compare products before making their purchase decision.

However, many product reviews (from other platforms than Amazon) 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 (Baccianella, Esuli & Sebastiani, 2009). Getting an

overall sense of a textual review could in turn improve consumer experience. Nevertheless, this predictive task presents some challenges. Firstly, because reviews are human feedbacks, it may be difficult to accurately predict the rating from the text content. Indeed, users all have different standards and do not rate a product the same way. For instance a user may rate a product as good and assign a 5-star score while another user may write the same comment and give only 3 stars. In addition, reviews may contain anecdotal information, which do not provide any helpful information and complicates the predictive task. Finally the vocabulary used can also be very specific according to the product category.

➤ **Review of Literature**

The dataset used for this task is the one available as a part of the flipkart.com dataset challenge. Most of the predictive tasks previously performed on this dataset have rating predictions primarily based on user and business attributes. However research has been carried out, not just in the general area of text mining and sentiment analysis, but in text mining for predictive tasks in review and rating systems. The impact of text derived information has been previously studied at the sentence level, with the help of the topic information on various datasets. Various methods have been adopted in the past, including regression, bag of opinions method and classification. In movie reviews, it has been observed that Naive Bayes' had a slightly better accuracy than the SVM method. However, this was in combination with other features of the dataset. Hence, the results differ from the ones chosen here.

The dataset has been extensively studied as well. Attempts have been made to gauge information from the review text by predicting what the user felt about various aspects of the business, such as service, quality of food and ambiance. If the user experience can be divided into various aspects, then a function of

these can be used to predict the overall rating. Another approach that has been taken is to classify 1 and 2 stars together, 4 and 5 stars together, in order to gauge the general opinion of the user. However, this falsely increases the accuracy of rating prediction. It is an analysis of the user's sentiment but should not be used for rating prediction tasks. Work has been done to take this a step further as well. If a feature for some customization of a user is included, we can treat the reviews of each user as separate entities. There is expected to be a uniformity in the reviews that a user writes and different users have different ways of expressing the same emotion. Other measures of evaluation such as precision and recall have been used for baseline comparison.

We can relate our problem with finding the truthfulness of any comment used in online platforms as well. In, truth assessment and semantic analysis for product review text is performed on flipkart.com data. Aspect based product review helpfulness is proposed. As related products share common aspects such as shipping and warranty, the authors propose an aspect extraction model making use of product category information to balance the aspects of a general category and those of subcategories under them. On top of this, a two layer regressor is trained for helpfulness prediction. Experimental results demonstrate 7% additional prediction accuracy compared to baseline methods on have product category data collected from Amazon. In the authors used both qualitative and quantitative factors as helpfulness prediction of reviews. Their findings suggest that word count has a threshold in its effects on review helpfulness. Additionally, reviewer experience and their impact were not statistically significant predictors for helpfulness. Past helpfulness records tends to predict future helpfulness ratings. Finally they conclude that characteristics of reviewers and review messages have a varying degree of impact on review helpfulness. Most of the previous works either use vector space model or traditional semantic analysis

with document features to improve the prediction. Considering that in mind, we use several linguistic (lexical, semantic), anatomical, and metadata information with word embedding for feature extraction

➤ **Motivation for the Problem Undertaken**

To understand real world problems where Machine Learning and Data Analysis can be applied to help to predict the prices in various domains to make better decisions with the help of which they can gain profit or can be escaped from any loss which otherwise could be possible without the study of data. The project has been provided by the Flip Robo Technologies as a part of the internship programme. This project helps to exposure to real world data. To display my skill in solving a real time problem has been the motivation.

For preparing the desired data a simple code was written in python to remove the useless features. Many features were removed except the summary of the review, the text of the review itself, score and product. The score that is generated by the reviewer includes a number of stars on scales of 1 to 5. Reviews that were rated with one or two stars were considered as negative and those with four or five stars were considered as positive. Reviews with three stars usually contain many mixed reviews and are difficult to be labelled into a positive or negative category.

Analytical Problem Framing

➤ Mathematical/ Analytical Modelling of the Problem

In the whole research process various mathematical, statistical and analytics modelling has been done. There has been reduction of the columns because few of them was not necessary for the problem solving. And few of them was removed due to very less correlation with dependent variable. Since the dataset contains a lot of features hence feature selection has been also done.

Our proposed approach is depicted that consists of five major steps. Firstly, review with star-rating score is fed into the feature generation engine that includes four different functions such as lexical, structural, semantic and combined. Secondly, generated features are used as input to standard machine learning classification algorithm. In the third step, four different trained

Models are built using the classification algorithm. In the fourth and fifth steps, test reviews are evaluated based on the pre-trained classification models for review helpfulness detection. We consider the review helpfulness prediction as a binary classification problem once reviews are labeled as helpful

In our work, all reviews can belong to exactly one class. Thus, our task is that of multi-class classification. It can also be reformulated as a binary classification by grouping the classes, for e.g. {1, 2 stars} and {3, 4, 5 stars}. The case of multi-label is therefore not applicable to us. In order to apply text classification, the unstructured format of text has to be converted into a structured format for the simple reason that it is much easier for computer to deal with numbers than text. This is mainly achieved by projecting the textual contents into Vector Space Model, where text data is converted into

vectors of numbers. In the field of text classification, documents are commonly treated like a Bag-of-Words (BoW), meaning that each word is independent from the others that are present in the document. They are examined without regard to grammar neither to the word order⁴. In such a model, the term-frequency (occurrence of each word) is used as a feature in order to train the classifier. However, using the term frequency implies that all terms are considered equally important. As its name suggests, the term frequency simply weights each term based on their occurrence frequency and does not take the discriminatory power of terms into account. To address this problem and penalize words that are too frequent, each word is given a term frequency inverse document frequency (tf-idf) score Flesch reading ease

➤ Data Sources and their formats

The most critical aspect of this project is the accumulation of knowledge. To prepare the models, the distinct well springs of the data on e-commerce websites flipkart.com. using selenium and saved in csv file.

scrape around 20000 rows of data. Review are collected for this project.

The data descriptions are as follow (26023, 7) rows and columns. To predict rating using Regression. I will start by importing all the necessary libraries that we need for this task and import the dataset.

1) Importing libraries

2) Importing the dataset

Our target is to find the insights of the data and to do thorough data analysis.

1.Uploading Data set

```

In [1]: import pandas as pd
import numpy as np
import matplotlib as mlp
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')

In [2]: trn_data=pd.read_csv('Product_Rating.csv')

In [3]: trn_data.head()

```

| Unnamed: 0 | Unnamed: 0.1 | Product | Brand_name | Price | Rating | Review Summary | Full Review | Review_date |
|------------|--------------|---------|--------------------|---------|--------|----------------|---|-------------|
| 0 | 0 | Phone | SAMSUNG Galaxy F13 | ₹11,999 | 3 | Fair | This mobile is purely for normal use, I don't ... | 1 month ago |

```

In [12]: #the complete information about the dataset
trn_data.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 21365 entries, 0 to 26012
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  -
 0   Product         21365 non-null  object
 1   Brand_name      21365 non-null  object
 2   Price           21365 non-null  object
 3   Rating          21365 non-null  int64
 4   Review Summary  21365 non-null  object
 5   Full Review     21365 non-null  object
 6   Review_date     21365 non-null  object
dtypes: int64(1), object(6)
memory usage: 1.3+ MB

```

➤ Data Pre-processing

Before building model, the data should be properly pre processed and converted to quality, clean data even the resulting machine learning model will be of great quality .The data pre-processing includes three main parts that is data integration, data cleaning, data transformation. In data integration the data collected from various sources are integrated. In data cleaning process the data containing the null values, unnecessary rows with null values are being cleared. The data transformation includes the feature scaling ,categorical data, etc to set the certain range of data.

- The dataset contains 21365 rows and 7 columns
- Rating is our dependent variable.

a) Checking for duplicates and dropping them

```
In [6]: trn_data.duplicated().sum()
Out[6]: 4657

In [7]: #dropping duplicate rows
trn_data.drop_duplicates(inplace=True)

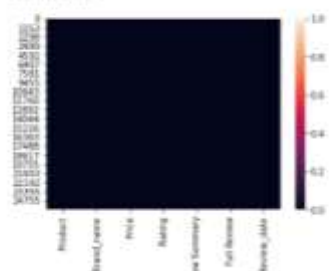
In [8]: trn_data.duplicated().sum()
Out[8]: 0
```

b) Checking missing value from the data set.

```
In [9]: trn_data.isnull().sum()
Out[9]: Product      0
Brand_name    0
Price         0
Rating        0
Review_Summary 1
Full_Review    0
Review_date    0
dtype: int64

In [10]: import seaborn as sns
sns.heatmap(trn_data.isnull())

Out[10]: <seaborn.axis>
```



➤ There were one null value was present in the dataset .we drop it

```
In [11]: trn_data.dropna(subset=['Review_Summary'], inplace = True)
```

```
In [12]: #the complete information about the dataset
trn_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21365 entries, 0 to 26012
Data columns (total 7 columns):
#   column          Non-Null Count  Dtype
---  -
0   Product         21365 non-null  object
1   Brand_name      21365 non-null  object
2   Price           21365 non-null  object
3   Rating          21365 non-null  int64
4   Review_Summary  21365 non-null  object
5   Full_Review     21365 non-null  object
6   Review_date     21365 non-null  object
dtypes: int64(1), object(6)
memory usage: 1.3+ MB
```

Data is pre-processed using following technique:

- Removing punctuation
- Removing numbers
- Converting text to lower case (no capital letters)
- Removing extra whitespace

- e) Removing stop-words (extremely common words which do not provide any analytic information and tend to be of little value i.e. a, and, are etc.)
- f) Tokenization
- g) Stemming
- h) Lemmatization
- i) Apply Text vectorization to convert text to numeric

➤ **Data Inputs- Logic- Output Relationships**

The dataset consists of 2 features columns. The Reviews are independent and rating is dependent as our rating change the value of our independent variable's changes. Using word cloud, we can see most occurring word for different categories.

➤ **Hardware and Software Requirements and Tools Used**

Hardware used for doing the project is a 'Laptop' with high end specification and stable internet connection .while coming to the software part I had used 'python jupyter notebook' for do my python program and data analysis.

Excel file and Microsoft excel are required for the data handling. In jupyter notebook I had imported lot of python libraries are carried to this project.

1.Pandas-a library which is used to read the data ,visualisation and analysis of data.

2.Numpy-used for working with array and various mathematical operations in python.

3. Seaborn- visualization for plotting different type of plot.
4. Matplotlib- It provides an object-oriented API for embedding plots into applications .
5. selenium- web scraping the data.
6. Libraries used for text mining/text analysis are nltk, stopwords, WordLemmatizer, tfidfvectorizer, Word_tokenize.
7. Libraries used for Machine learning model building like LogisticRegression, SVM, RandomForestClassifier.

Model/s Development and Evaluation

➤ Identification of possible problem-solving approaches (methods)

In machine learning, several algorithms are applied to predict rating on review. The algorithms are: Linear regression, Decision tree, SVR, Gradient Boosting Regression, Ridge and Random Forest Algorithm. These models have been implemented using the python library Sklearn. The parameters like accuracy score , confusion matrix Cross Validation Score are considered to check the efficiency of these models. Further Hyper tuning to build more accurate model

Regression Model with following algorithms

- Linear Regression
- Decision Tree Regressor
- Random forest regressor

- SVR
- Gradient Boosting Regressor
- Ridge

Evaluation metrics

- Accuracy Score
- Confusion matrix
- Cross val_ score.

Testing of Identified Approaches (Algorithms)

Listing down all the algorithms used for the training and testing.

- LR=LinearRegression()
- DT=DecisionTreeRegressor()
- rf=RandomForestRegressor()
- svr=SVR()
- R=Ridge()
- GBR=GradientBoostingRegressor()

Run and Evaluate selected models

Describe all the algorithms used along with the snapshot of their code and what were the results observed over different evaluation metrics. Train-test data splits were conducted. In this situation, we split the data into training and test sets, then fit candidate models on the training set, evaluate and select them on the test set.

1. LogisticRegression

```
In [54]: # Creating train_test_split using best random_state
x_train, x_test, y_train, y_test = train_test_split(X, y, random_state=31, test_size=.1)
```

```
In [57]: from sklearn.linear_model import LogisticRegression
# creating the model
model_lg = LogisticRegression()

# feeding the training set into the model
model_lg.fit(x_train, y_train)

# predicting the results for the test set
y_pred = model_lg.predict(x_test)

# calculating the training and testing accuracies
print('training accuracy :', model_lg.score(x_train, y_train))
print('testing accuracy :', model_lg.score(x_test, y_test))
#Accuracy score
print('Accuracy Score of Logistics Regression :', accuracy_score(y_test, y_pred))
# classification report
print('classification Report of Logistics Regression :\n', classification_report(y_test, y_pred))

# confusion matrix
print('Confusion matrix of Logistics Regression :\n', confusion_matrix(y_test, y_pred))

training accuracy : 0.988571723484363
testing accuracy : 0.8823712948517941
Accuracy Score of Logistics Regression : 0.8823712948517941
classification Report of Logistics Regression :
      precision    recall  f1-score   support

     1       0.83     0.17     0.28         925
     2       0.69     0.01     0.03         225
     3       0.44     0.04     0.08         488
     4       0.11     0.02     0.02       1258
     5       0.52     0.98     0.73       2572

 accuracy              0.58         6418

      macro avg       0.55     0.25     0.22         6418
      weighted avg     0.36     0.53     0.46         6418

Confusion matrix of Logistics Regression :
[[ 167   0   7   0  761]
 [ 22   1   1   1  158]
 [  7   2  20   4  417]
 [  2   0  12  15 1289]
 [  2   0   5  27 3538]]
```

```
In [58]: from sklearn.model_selection import cross_val_score
CvScore = cross_val_score(model_lg, X, y, cv =5)
print('%03f'% 'Cross Validation Score', model_lg, '%03f'% CvScore)
print('CvScore : ', CvScore)
print('Mean Cv Score : ', CvScore.mean())
print('Std deviation : ', CvScore.std())

Cross Validation Score LogisticsRegression() :
CvScore : [0.36587877 0.57647293 0.57383571 0.57172046 0.56338447]
Mean Cv Score : 0.5790444652468892
Std deviation : 0.0946791479287934
```


2. Decision Tree Classifier

```
In [44]: from sklearn.tree import DecisionTreeClassifier

# creating model
model_dt = DecisionTreeClassifier()

# feeding the training set into the model
model_dt.fit(x_train, y_train)

# predicting the results for the test set
y_pred = model_dt.predict(x_test)

# calculating the training and testing accuracies
print("training accuracy :", model_dt.score(x_train, y_train))
print("testing accuracy :", model_dt.score(x_test, y_test))

# accuracy score
print("accuracy score of DecisionTreeClassifier :", accuracy_score(y_test, y_pred))

# classification report
print("classification report of DecisionTreeClassifier :", classification_report(y_test, y_pred))

# confusion matrix
print("confusion matrix of DecisionTreeClassifier :", confusion_matrix(y_test, y_pred))

training accuracy : 0.882861186132057
testing accuracy : 0.85479128827613
accuracy score of DecisionTreeClassifier : 0.85479128827613
classification report of DecisionTreeClassifier :
      precision    recall  f1-score   support

     0       0.35       0.35       0.35         815
     1       0.51       0.45       0.48         123
     2       0.42       0.87       0.58         408
     3       0.40       0.86       0.51        1138
     4       0.59       0.97       0.73        1072

    accuracy: 0.85
    avg prec: 0.57
    weighted avg: 0.58       0.60       0.58       4426

Confusion matrix of DecisionTreeClassifier :
[[ 219  0  11  0  666]
 [ 25 17  3  1 179]
 [ 19  3  31  0  194]
 [  2  0  18 77 1140]
 [  4  0  13 184 1411]]

In [45]: from sklearn.model_selection import cross_val_score
cv_score = cross_val_score(model_dt, x, y, cv=10)
print("Cross Validation Score", model_dt, "\n")
print("CV score :", cv_score)
print("Mean CV score :", cv_score.mean())
print("Std Deviation :", cv_score.std())

Cross Validation Score DecisionTreeClassifier() :
CV score : [0.50881403 0.57064404 0.57262732 0.68786763 0.57648801]
Mean CV score : 0.5679525111055117
Std Deviation : 0.061056446229863884
```

3. Random Forest Classifier

```
In [46]: from sklearn.ensemble import RandomForestClassifier

# creating the model
model_rf = RandomForestClassifier(n_estimators = 100)

# feeding the training set into the model
model_rf.fit(x_train, y_train)

# predicting the results for the test set
y_pred = model_rf.predict(x_test)

# calculating the training and testing accuracies
print("training accuracy :", model_rf.score(x_train, y_train))
print("testing accuracy :", model_rf.score(x_test, y_test))

# accuracy score
print("accuracy score of RandomForestClassifier :", accuracy_score(y_test, y_pred))

# classification report
print("classification report of RandomForestClassifier :", classification_report(y_test, y_pred))

# confusion matrix
print("confusion matrix of RandomForestClassifier :", confusion_matrix(y_test, y_pred))

training accuracy : 0.882861186132057
testing accuracy : 0.85479128827613
accuracy score of RandomForestClassifier : 0.85479128827613
classification report of RandomForestClassifier :
      precision    recall  f1-score   support

     0       0.35       0.35       0.35         815
     1       0.50       0.45       0.48         123
     2       0.46       0.87       0.58         408
     3       0.40       0.86       0.51        1138
     4       0.59       0.97       0.74        1072

    accuracy: 0.85
```

| | | | | | |
|--------------|--|------|------|------|------|
| accuracy | | 0.57 | 0.25 | 0.38 | 0.40 |
| macro avg | | 0.57 | 0.25 | 0.38 | 0.40 |
| weighted avg | | 0.59 | 0.48 | 0.50 | 0.49 |

Confusion matrix of RandomForestClassifier :

```
[[ 133  12  14  0  0]
 [ 15  17  3  1  1]
 [ 0  4  12  7  1]
 [ 2  0  13  74 13]
 [ 3  1  7  61 147]]
```

4.Gradient Boosting Classifier

```
In [34]: from sklearn.ensemble import GradientBoostingClassifier

# creating the model
model_gbc = GradientBoostingClassifier()
# feeding the training set into the model
model_gbc.fit(x_train, y_train)

# predicting the results for the test set
y_pred = model_gbc.predict(x_test)

# calculating the training and testing accuracies
print("training accuracy :", model_gbc.score(x_train, y_train))
print("testing accuracy :", model_gbc.score(x_test, y_test))

#Accuracy Score
print("accuracy score of GradientBoostingClassifier :", accuracy_score(y_test, y_pred))

# classification report
print("classification report of GradientBoostingClassifier :\n", classification_report(y_test, y_pred))

# confusion matrix
print("Confusion matrix of GradientBoostingClassifier :\n", confusion_matrix(y_test, y_pred))

Training accuracy : 0.8681000000000001
Testing accuracy : 0.5742000000000001
accuracy score of GradientBoostingClassifier : 0.5742000000000001
classification report of GradientBoostingClassifier :
precision    recall  f1-score   support

 1      0.00      0.12      0.01      120
 2      0.00      0.00      0.00       120
 3      0.00      0.00      0.00       450
 4      0.10      0.41      0.20      1230
 5      0.57      1.00      0.71     1972

 accuracy      0.42      0.25      0.37      3420
 macro avg           0.13      0.31      0.24      3420
 weighted avg           0.54      0.27      0.44      3420

Confusion matrix of GradientBoostingClassifier :
[[ 113  0  0  2  0  0]
 [ 10  0  0  0  0  0]
 [ 2  0  2  1  4  4]
 [ 0  0  0  7 13 1]
 [ 1  0  0 12 26 0]]
```

```
In [ ]: from sklearn.model_selection import cross_val_score
cvscore = cross_val_score(model_gbc, x, y, cv=5)
print('Cross Validation Score', model_gbc, '\n')
print("cvscore :", cvscore)
print("Mean CV Score :", cvscore.mean())
print("Std deviation :", cvscore.std())
```

5.SVC

```
In [44]: from sklearn.svm import SVC

# creating the model
model_svc = SVC()

# feeding the training set into the model
model_svc.fit(x_train, y_train)

# predicting the results for the test set
y_pred = model_svc.predict(x_test)

# calculating the training and testing accuracies
print("training accuracy :", model_svc.score(x_train, y_train))
print("testing accuracy :", model_svc.score(x_test, y_test))

#Accuracy Score
print("accuracy score of SVC :", accuracy_score(y_test, y_pred))

# classification report
print("classification report of SVC :\n", classification_report(y_test, y_pred))

# confusion matrix
print("Confusion matrix of SVC :\n", confusion_matrix(y_test, y_pred))

Training accuracy : 0.8540000000000001
Testing accuracy : 0.5880000000000001
accuracy score of SVC : 0.5880000000000001
classification report of SVC :
precision    recall  f1-score   support

 1      0.00      0.16      0.00      120
 2      0.50      0.01      0.04       120
 3      0.40      0.07      0.12       450
 4      0.40      0.06      0.10      1230
 5      0.50      0.90      0.74     1972
```

```

accuracy          0.59          0.28          0.60          0.60
macro avg         0.59          0.28          0.28          0.60
weighted avg      0.60          0.60          0.60          0.60

Confusion matrix of SVC :
[[ 239   2   12   0  672]
 [  25   9   3   1  191]
 [  10   2  32   6   400]
 [   2   0  13  69  1154]
 [   4   0   7  67  3494]]

```

```

In [63]: from sklearn.model_selection import cross_val_score
CVscore = cross_val_score(model_svc, X, Y, cv=5)
print('Cross Validation Score', model_svc, '\n')
print('CVscore : ', CVscore)
print('Mean CV Score : ', CVscore.mean())
print('Std deviation : ', CVscore.std())

Cross Validation Score SVC() :

CVscore : [0.56821905 0.57758015 0.57363571 0.60379125 0.59840861]
Mean CV Score : 0.5843609553007255
Std deviation : 0.014086829593140657

```

Evaluating the model accuracy is an essential part of the process of creating machine learning models to describe how well the model is performing in its predictions are mainly used to evaluate the prediction error rates and model performance in regression analysis.

the best model choose for hyper parameter tuning are RandomForestRegressor,

➤ RandomForestRegressor

Hyper parameter tuning

Random Forest Regressor

```

In [59]: from sklearn.model_selection import GridSearchCV

In [60]: parameter = { 'max_features': ['auto', 'log2'],
                       'criterion':['gini','entropy'],
                       'n_estimators': [75,100,150]}

In [61]: GCV = GridSearchCV(RandomForestClassifier(),parameter,verbose=10)
GCV.fit(x_train,y_train)

[CV 4/5; 11/12] START criterion=entropy, max_features=log2, n_estimators=100...
[CV 4/5; 11/12] END criterion=entropy, max_features=log2, n_estimators=100; score=0.579 total time= 5.4min
[CV 5/5; 11/12] START criterion=entropy, max_features=log2, n_estimators=100...
[CV 5/5; 11/12] END criterion=entropy, max_features=log2, n_estimators=100; score=0.577 total time= 5.5min
[CV 1/5; 12/12] START criterion=entropy, max_features=log2, n_estimators=150...
[CV 1/5; 12/12] END criterion=entropy, max_features=log2, n_estimators=150; score=0.582 total time= 8.2min
[CV 2/5; 12/12] START criterion=entropy, max_features=log2, n_estimators=150...
[CV 2/5; 12/12] END criterion=entropy, max_features=log2, n_estimators=150; score=0.575 total time= 8.0min
[CV 3/5; 12/12] START criterion=entropy, max_features=log2, n_estimators=150...
[CV 3/5; 12/12] END criterion=entropy, max_features=log2, n_estimators=150; score=0.572 total time= 7.8min
[CV 4/5; 12/12] START criterion=entropy, max_features=log2, n_estimators=150...
[CV 4/5; 12/12] END criterion=entropy, max_features=log2, n_estimators=150; score=0.580 total time= 7.9min
[CV 5/5; 12/12] START criterion=entropy, max_features=log2, n_estimators=150...
[CV 5/5; 12/12] END criterion=entropy, max_features=log2, n_estimators=150; score=0.577 total time= 7.9min

Out[61]: GridSearchCV(estimator=RandomForestClassifier(),
                      param_grid={'criterion': ['gini', 'entropy'],
                                   'max_features': ['auto', 'log2'],
                                   'n_estimators': [75, 100, 150]},
                      verbose=10)

In [62]: GCV.best_params_

Out[62]: {'criterion': 'gini', 'max_features': 'log2', 'n_estimators': 100}

```

```

In [65]: from sklearn.ensemble import RandomForestClassifier

Final_mod = RandomForestClassifier(criterion='gini',n_estimators= 100,max_features='log1')
Final_mod.fit(x_train,y_train)
y_pred=Final_mod.predict(x_test)
print('Final Random Forest Classifier Model')
print('Accuracy Score :'+'\n', accuracy_score(y_test, y_pred))
print('\n')
print('Confusion matrix of Random Forest Classifier :'+ '\n',confusion_matrix(y_test, y_pred))
print('\n')
print('Classification Report of Random Forest Classifier'+'\n',classification_report(y_test, y_pred))

Final Random Forest Classifier Model
Accuracy Score :
0.597971218276756

Confusion matrix of Random Forest Classifier :
[[ 222  12  15   0  649]
 [ 28  17   3   1  179]
 [ 10   3  33   6  399]
 [  2   0  13  70 1153]
 [  4   0   6  79 3481]]

Classification Report of Random Forest Classifier
precision    recall  f1-score   support

      1       0.05       0.25       0.39       525
      2       0.52       0.06       0.13       225
      3       0.40       0.07       0.13       409
      4       0.45       0.06       0.18       1238
      5       0.54       0.97       0.74       2673

 accuracy: 0.598
 macro avg: 0.58      0.28      0.58      6428
weighted avg: 0.59      0.68      0.58      6428

```

The best model after hyper parameter tuning is Random Forest Regressor

10. Saving model

```

In [66]: import joblib
         joblib.dump(Final_mod,'product_Review_Rating_Prediction.pkl')

Out[66]: ['product_Review_Rating_Prediction.pkl']

```

In our project we had implemented various Machine Learning Algorithms such as Logistic Regression, Decision Tree Regression, Random Forest Regression and compared the accuracy of results based on our test data set. Based on the various accuracy levels we find that Random Forest Regression gives the highest accuracy i.e. 60%. Therefore we selected Random Forest Regression and created User Interface based on it.

Visualizations

After cleaning the data, we can visualize data and better understand the relationships between different variables. There are many more visualizations that you can do to learn more about your dataset, like scatterplots, histograms, boxplots.

➤ The Analysis of Rating of product based on review

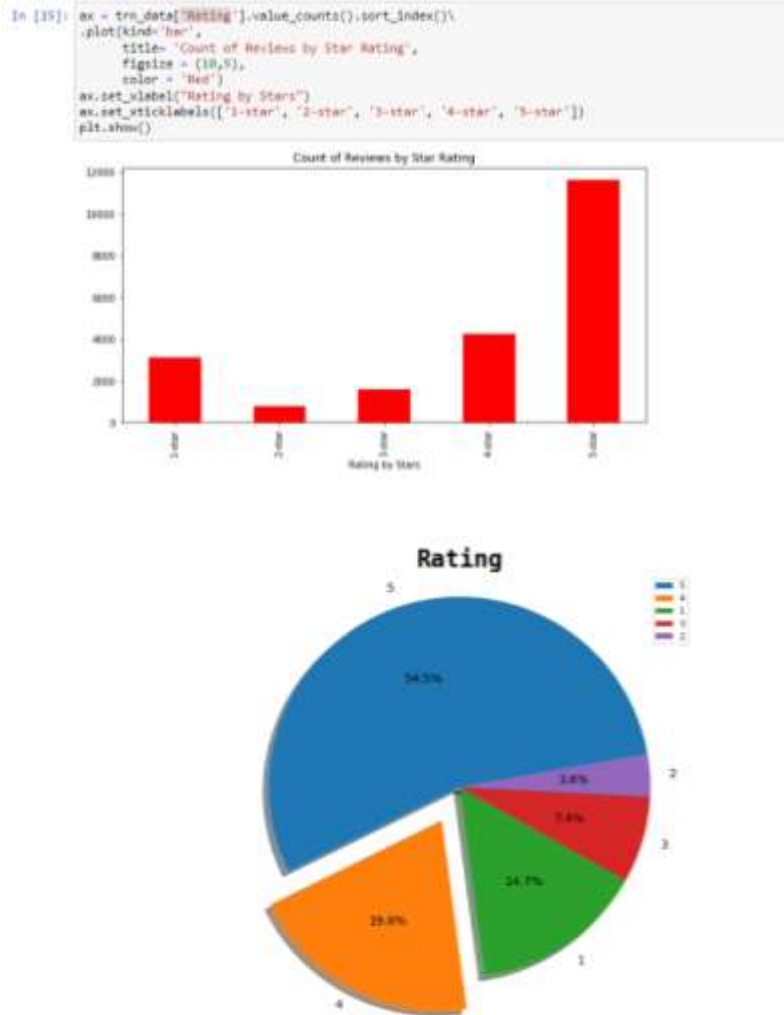


Fig. 1&2: The Analysis Rating of product based on review

1. Around 54.4% customer given 5- star rating followed by 14.7% customer given lowest 1-star rating.
2. Around 19.8% customer given 4- star rating followed by 7.4% customer given lowest 3-star rating.

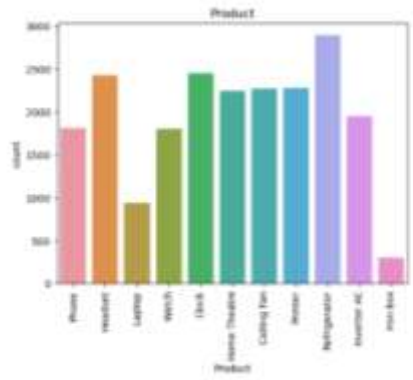


Fig. 3: The Analysis of product

- In observation from fig:3 Refrigerator has high amount review are compare with iron box

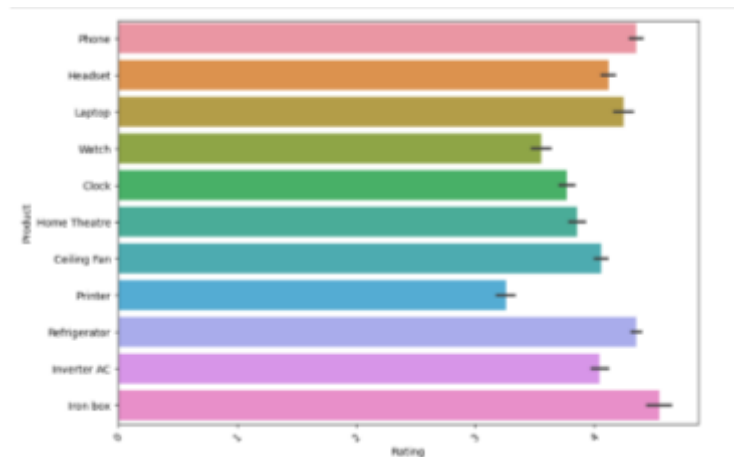


Fig. 4: The Analysis of The Rating and product.

- From fig 4: iron box and phone have above 4 star rating.
- Printer and watch have 3 star rating compare with all product.

word cloud

- A word cloud is a simple yet powerful visual representation object for text processing, which shows the most frequent word with bigger and bolder letters, and with different colors. The smaller the the size of the word the lesser it's important
- Significant textual data points can be highlighted using a word cloud.



CONCLUSION

➤ Key Findings and Conclusions of the Study

From this dataset I get to know that each feature play a very import role to understand the data. Data format plays a very important role in the visualization and Applying the models and algorithms .

There are many systems which uses different machine learning algorithms such as Logitic Regression (LR), Decision Tree, Support Vector Machine (SVM), Random Forest Algorithm, etc for predicting the rating of product. In this ML based system, we are using Random Forest Algorithm which gives more accuracy in predicting the rating.. It is easy to use and it gives more accuracy in prediction. It requires less time for prediction and it helps in reduction of over fitting. The goal of this dissertation is to successfully predict a user's numerical rating from its review text content. To do so, supervised machine learning techniques and more specifically text classification are used. Three distinct approaches are presented, namely binary classification, aiming at predicting the rating of a review as low or high, as well as multi-class classification and logistic regression whose aim is to predict the exact value of the rating for each review. Moreover, three different classifiers (Naïve Bayes, Support Vector Machine and Random Forest) are trained and tested on two different datasets from Amazon. These datasets are divided into two major categories: experience and search products and are characterized by

an imbalanced distribution. We overcome this issue by applying sampling techniques to even out the class distributions. Eventually, the performance of those classifiers is tested and assessed thanks to accuracy metrics, including precision, recall and f1-score.

Learning Outcomes of the Study in respect of DataScience

Learnt how to process the large number of data. Tried and learnt more about distribution of the data. The power of visualization is helpful for the understanding of data into the graphical representation its help me to understand that what data is trying to say, Data cleaning is one of the most important step to remove missing value or null value fill it by mean median or by mode or by 0. Setting a good parameters is more important for the model accuracy. Finding a best random state played a vital roll in finding a better model.

In this paper, a recommendation model is proposed by mining sentiment information from social users' reviews. We fuse user sentiment similarity, interpersonal sentiment influence, and item reputation similarity into a unified matrix factorization frame-work to achieve the rating prediction task. In particular, we use social users' sentiment to denote user preferences. Besides, we build a new relationship named interpersonal

sentiment influence between the user and friends, which reflects how users' friends influence users in a sentimental angle. What is more, as long as we obtain user's textual reviews, we can quantitatively measure user's sentiment, and we leverage items' sentiment distribution among users to infer item's reputation. The experiment results demonstrate that the three sentimental factors make great contributions to the rating prediction. Also, it shows significant improvements over existing approaches on a real-world dataset

Limitations of this work and Scope for Future Work

The techniques to increase the speed of the model need to be constructed. In Upcoming days when huge amount of information is accessed as in detailed information in the dataset, the expected results in future are highly correct. For further research anyone desire to expand upon it ought to request different sources of historical data or be a lot of organized in collection knowledge manually over amount of your time to boot. At last, it is curious to match our model accuracy with that of the business models accuracy offered nowadays.

For conducting the experiments in this thesis, no pre-processing has been done on the data set. Pre-processing is the process where the data is being cleaned and prepared before being fed to the algorithms. Online reviews usually contain many irrelevant and uninformative features

which may not even have an impact on the orientation of them. This process involves removing many steps such as 17 white space removal and stop words removal etc. The results from all experiments implies that both approaches give higher accuracies when they are being applied on the summaries of the reviews. The possible explanation for this result might be the nature of the reviews. The reviews itself contain a large amount of words, which can lead to sparsity in bag of words features. As a result we see that the accuracies of the algorithms for all experiments are higher when applied on the summaries which are more informative and contain limited number of words.