RATINGS PREDICTION

Submitted By :

Shivani Angadi

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my trainers at Data Trained “Dr. Deepika Sharma” and my trainer for Natural Language Processing for being such a fantastic coach as well as my SME at Flip Robo Technologies “Mr. Shubham Yadav " who gave me the golden opportunity to do this wonderful project on the topic “Ratings Prediction”, which also helped me in doing a lot of research and exploring so many new things. Also, I would like to show appreciation to “Mr. Shubham Yadav” and “Mr. Sajid Choudhary” for guiding me throughout and helping me solve issues regarding this project.

I am really thankful to Flip Robo Technologies for giving me this prime opportunity of interning in their organization and enhancing my skills.

Secondly, I want to thank Data Trained Academy for giving me the best possible study notes, online platforms such as www.medium.com, www.stackoverflow.com, www.scikit-learn.org and www.github.com for providing me with the resources when I stumbled and fell along the way.
It helped me increase my knowledge and skills.

THANKS AGAIN TO ALL WHO SUPPORTED.

Introduction

Business Problem

The client owns a website where people write reviews on different technical products they purchase, and now wishes to add a new feature to it by adding “Ratings” to every “Review” that the consumer writes on their website. Now, it is mandatory to add ratings as well with the reviews. Since, the old feature only consisted of reviews, the client would like to build a machine learning model that can help them predict ratings for the reviews which were written in the past and do not have a rating. The ratings should be in such a way that it has only 5 options available i.e. 1,2,3,4 & 5.

Background Domain Knowledge

An e-commerce website is an extensive software that lets industries manage all functions regarding online sales and services of products. E-commerce sites are trending and have become the digital hub for product sales and storage of customer data allowing customers to make online payments through their websites at ease and select their means of payment as per the convenience. E-commerce sites are mostly designed for B2C sales. These sites consist of a feature where the customers can make an entry as a review of how much they like the product after making purchases and also put a negative remark in case they dislike it.

Literature Review

NLP (Natural Language Processing) is a sub-area of Artificial Intelligence which helps computers and gives the ability to understand human languages and communicate with them in their own language. This whole concept is about dealing with texts or speech in human understandable language.

Project Motivation

The motivation is to build a machine learning model with the help of Natural Language Processing that will be used by the company management to figure out the ratings for each review that was being posted on their website in the past which in return will help the organization in increasing their revenue.

Analytical Problem Framing

Analytical Modeling of Problem

The “Ratings Prediction” project consists of the reviews and ratings of technical products such as laptops, Phones, Headphones, smart watches, Professional Cameras, Printers, Monitors, Home theater, Router from different ecommerce websites such as Amazon and Flipkart. The data set collected had 20,649 records and 2 variables i.e. “Reviews” & “Ratings”. The “Reviews” data is in the form of a text which is processed by making use of **Natural Language Processing**. The use case of this task is that of a **Sentiment Analysis** problem. Problem Solving was handled in 5 parts:

1. **Project Domain Research -** Domain research was made in the beginning of the task to have an understanding of the task and its business problem in depth. This helps in building a model in accordance with the use case and focuses on the main purpose of the task.
2. **Collecting Data -** Data Collection is made from different e-commerce sites such as Amazon and Flipkart for various technical products. The data is scrapped by making use of tools such as Selenium and Beautifulsoup.
3. **Text Preprocessing -** Text data contains huge ambiguity which needs to be processed by making use of the NLP preprocessing pipeline. Use of nltk tools have been made to preprocess data.
4. **Vectorizing Text Data -** After preprocessing of text, these texts need to be converted into vectors in order to make the machine learning model understand the data passed during the model building phase.
5. **Interpreting Solutions -** Once the data is preprocessed and vectorized, we interpret it’s result using different classification algorithms. Making comparisons, the model which attains highest accuracy is selected.

Data Sources

The data is scrapped from e-commerce websites such as Amazon and Flipkart and is saved in the form of an excel sheet. Data is scrapped by making use of tools such as **Selenium** and **Beautifulsoup**. Reviews and Ratings are scrapped for different technical products . The data set contained an unimportant variable called “Unnamed: 0” which was removed from the data set. A pictorial representation of

how the initial data set looked like is shown below:

	Unnamed: 0	Reviews	Ratings
0	0	Best for this price range but camera is ok per...	5
1	1	Good budget phone. Best battery and best perfo...	5
2	2	Nice entry level smartphone.. too good for fam...	5
3	3	Phone is good but camera quality is not so goo...	4
4	4	Very nice mobile in this price Battery perform...	5

Data Pre-processing

Data Cleaning and processing plays a major role when it comes to text comprehensions. The data ambiguity present in text data is much higher than that present in a normal data set. Cleaning of these unnecessary data helps the machine learning model to understand data more accurately. The approach data preprocessing are :

- Dropping the "Unnamed: 0" column as it is an extra element in a data set which is not important. Hence, we remove the column.

Pre Processing Of Text

- Removing Punctuations from the text present in the data set.

```
# Remove punctuations from data set
df['Reviews'] = df['Reviews'].str.replace(r'[^\w\d\s]', ' ') #removing punctuations
df['Reviews'] = df['Reviews'].str.replace(r'[?:\d+(?:\.\d*)?|\.\d+]', ' ')  #Removing numerical data
df['Reviews'] = df['Reviews'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$',' ') #removing websites
df.head()
```

- Removal of Emoticons

```
import re
def remove_emoji(string):
    emoji_pattern = re.compile("["
                               u"\U0001F600-\U0001F64F" # emoticons
                               u"\U0001F300-\U0001F5FF" # symbols & pictographs
                               u"\U0001F680-\U0001F6FF" # transport & map symbols
                               u"\U0001F1E0-\U0001F1FF" # flags (iOS)
                               u"\U00002702-\U000027B0"
                               u"\U000024C2-\U0001F251"
                               "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r'', string)
```

```
df['Reviews'] = df['Reviews'].apply(remove_emoji)
```

- Setting a list of our own stop words as in consideration with the use case. List of self created stop words is shown below:

```
#Setting our own stop words as per requirements
stop_words = ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd",
              'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers',
              'herself', 'it', "it's", 'its', 'itself','they', 'them', 'their', 'theirs', 'themselves', 'what', 'which',
              'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been',
              'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if',
              'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between',
              'into', 'through', 'during', 'before', 'after', 'to', 'from', 'up', 'down', 'in', 'out', 'over', 'under',
              'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both',
              'each','other','such', 'own', 'same', 'so','than','s', 't', 'can', 'will', 'just', 'should', "should've",
              'now', 'd', 'll', 'm', 'o', 're', 've', 'y','ma','printer','phone','mobile','router','earphone','earphones',
              'headphone','headphones','smartphone','product','home theater','smart watch','laptop','laptops','camera',
              'cameras']

```

- Removing Stopwords & Converting all texts into lower case and also removing words with length less than 2 because it is considered not important in consideration with the goal of the task.

```
#stopwords removal
df['Reviews']=df['Reviews'].apply(lambda words: ' '.join(word.lower() for word in words.split()
                                                         if word not in stop_words and len(word)>2))
```

```
df.head()
```

	Reviews	Ratings	length
0	best price range performance enough lite use d...	5	195.0
1	good budget best battery best performance came...	5	232.0
2	nice entry level too good family use used ligh...	5	124.0
3	phone good quality not good battery performanc...	4	124.0
4	very nice price battery performance good camer...	5	160.0

- Removing Extra Spaces

```
# Replace whitespace between terms with a single space
df['Reviews']= df['Reviews'].str.replace(r'\s+', ' ')

# Remove leading and trailing whitespace
df['Reviews'] = df['Reviews'].str.replace(r'^\s+|\s+?$', '')
```

- Tokenization and Lemmatization of Texts

```
#Creatinf function to lemmatize words in a text
import nltk

w_tokenizer = nltk.tokenize.WhitespaceTokenizer()
lemmatizer = nltk.stem.WordNetLemmatizer()

def lemmatize_text(text):
    return [lemmatizer.lemmatize(w,'v') for w in w_tokenizer.tokenize(text)]
```

```
#Applying Lemmatization on the column
df['Reviews'] = df['Reviews'].apply(lemmatize_text)
```

```
#The texts have been first converted into tokens and then Lemmatized
df.Reviews
```

```
0         [best, price, range, performance, enough, lite...
1         [good, budget, best, battery, best, performanc...
2         [nice, entry, level, too, good, family, use, u...
3         [phone, good, quality, not, good, battery, per...
4         [very, nice, price, battery, performance, good...
5         [good, like, performance, great, display, good...
6         [camera, and, battery, best, perform, super, p...
7         [fully, colourful, display, battery, backup, a...
8         [this, mark, camera, bad, performance, wise, n...
9         [the, very, good, photo, also, very, good, and...
10        [camera, not, good, performance, wise, display...
11        [best, high, end, graphic, game, pubg, availab...
12        [pro, large, battery, last, long, time, lag, d...
13        [more, efficiency, more, power, nice, one, rea...
14                                [best, quality, product]
```

It is clearly noticeable from the output that the strings have been converted into tokens and then lemmatized.

Comparing Original & Cleaned Text

	Reviews	Ratings	length	clean_length
0	[best, price, range, performance, enough, lite...	5	195.0	18
1	[good, budget, best, battery, best, performanc...	5	232.0	23
2	[nice, entry, level, too, good, family, use, u...	5	124.0	14
3	[phone, good, quality, not, good, battery, per...	4	124.0	11
4	[very, nice, price, battery, performance, good...	5	160.0	17

```
# Total length removal
print ('Origian Length', df.length.sum())
print ('Clean Length', df.clean_length.sum())
```

```
Origian Length 2582308.0
Clean Length 267798
```

The ambiguity present in the text can be easily seen from the above findings. The original length of text is 25,82,308 and the length of processed text is 2,67,798. Removal of unwanted data helps in training the model better and helps in attaining better accuracy in turn reducing the runtime.

Assumptions

- An assumption of not attaining a good accuracy for the model was made looking at the imbalance nature of the data set. This makes the model partial for a particular output class.
- It was expected to have a long runtime considering the length of the data set.
- Multinomial Naive Bayes Classifier was expected to give the best results in case of an NLP Sentiment Analysis Classification.

Hardware and Software Requirements

Hardware

1. 16 GB RAM - used for data storage and training models.

2. CPU - used for executing algorithms.

Software

1. Pandas - A fast, powerful and flexible open source tool used for data manipulation and data analysis.

2. Numpy - Numpy is an open source library used for mathematical and computational analysis of data and works best with multidimensional arrays and matrices.

3. Scikit-Learn - Sklearn is a free Machine Learning library used to run various algorithms and consists of scientific libraries like NumPy and SciPy.

4. Seaborn - Seaborn is a library used to plot graphs. It also visualizes random distributions.

5. Matplotlib - It works like MATLAB and helps pyplot function in making some changes to a figure such as creating a figure, creating plotting area, etc.

6. Pickle - Pickle library is used to save models which can be used for both “dumping” and “loading” purposes.The model can be used again to read and write whenever required.

7. Nltk - It is a foremost platform in coding python programs to deal with text data and human understandable language.

8. Regex - Regex library is a regular expression that helps in narrowing down the appearance of texts with the help of certain expressions that can be used in Analytics.

Model Development & Evaluation

Problem Solving Approach

- The first stage was to have an understanding of the data and see how the data looks i.e. describing data, checking it's length and shape.
- Second phase includes text pre-processing.
- After preprocessing, all the cleaned text present in the data were converted into vectors i.e. machine understandable form.
- 4 different Classification models were used and compared in order to see which algorithm gives the best results in case of text evaluation.
- Model which attains the best accuracy was selected as our model.

Algorithms Used

1. MultinomialNB
2. DecisionTreeClassifier
3. KNeighborsClassifier
4. RandomForestClassifier

Evaluation of Selected Models

- MultinomialNB

```
Accuracy score of MultinomialNB() is:
0.5349854227405247
[[ 622    4    1    1  443]
 [ 105   26    0    1  139]
 [  76    2    0    0  282]
 [  63    1    0   21  622]
 [ 172    0    0    2 1533]]
              precision    recall  f1-score   support

           1       0.60      0.58      0.59      1071
           2       0.79      0.10      0.17       271
           3       0.00      0.00      0.00       360
           4       0.84      0.03      0.06       707
           5       0.51      0.90      0.65      1707

    accuracy                           0.53      4116
   macro avg       0.55      0.32      0.29      4116
weighted avg       0.56      0.53      0.44      4116
```

- DecisionTreeClassifier

```
Accuracy score of DecisionTreeClassifier() is:
0.4793488824101069
[[ 578   59   54   85  295]
 [ 106   54   23   25   63]
 [  76   22   45   41  176]
 [ 111   23   50  149  374]
 [ 229   41   88  202 1147]]
              precision    recall  f1-score   support

           1       0.53      0.54      0.53      1071
           2       0.27      0.20      0.23       271
           3       0.17      0.12      0.15       360
           4       0.30      0.21      0.25       707
           5       0.56      0.67      0.61      1707

    accuracy                           0.48      4116
   macro avg       0.36      0.35      0.35      4116
weighted avg       0.45      0.48      0.46      4116
```

- KNeighborsClassifier

```
Accuracy score of KNeighborsClassifier() is:
0.43731778425655976
[[ 514   38   93  101  325]
 [ 100   46   24   21   80]
 [  79   18   41   74  148]
 [ 106   20   72  189  320]
 [ 276   39  103  279 1010]]
              precision    recall  f1-score   support

           1       0.48      0.48      0.48      1071
           2       0.29      0.17      0.21       271
           3       0.12      0.11      0.12       360
           4       0.28      0.27      0.28       707
           5       0.54      0.59      0.56      1707

    accuracy                           0.44      4116
   macro avg       0.34      0.32      0.33      4116
weighted avg       0.43      0.44      0.43      4116
```

- RandomForestClassifier

```
Accuracy score of RandomForestClassifier() is:
0.5493197278911565
[[ 678   22   13   11  347]
 [ 132   48    9    8   74]
 [ 101    4   26    7  222]
 [  98    2   15   99  493]
 [ 211    8   19   59 1410]]
              precision    recall  f1-score   support

           1       0.56      0.63      0.59      1071
           2       0.57      0.18      0.27       271
           3       0.32      0.07      0.12       360
           4       0.54      0.14      0.22       707
           5       0.55      0.83      0.66      1707

    accuracy                           0.55      4116
   macro avg       0.51      0.37      0.37      4116
weighted avg       0.53      0.55      0.50      4116
```

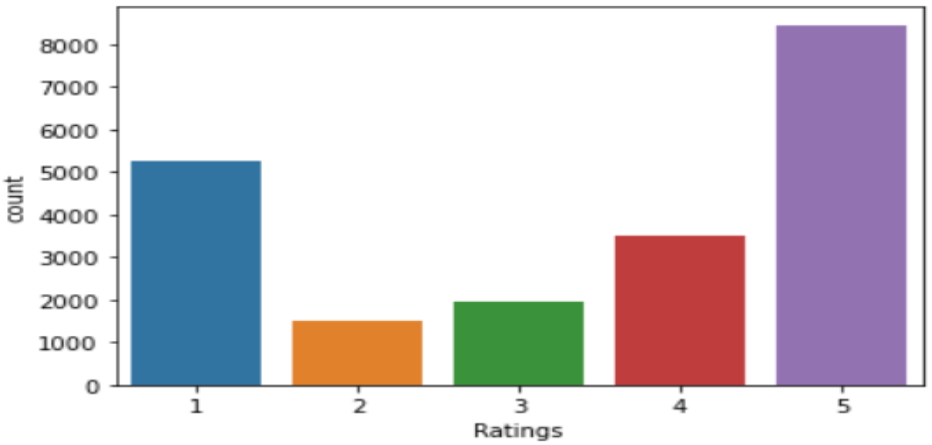
Conclusion -

We get an accuracy of 53.49% in MultinomialNB, 47.93% Decision Tree, 43.73% KNeighbors Classifiers and 54.93% for RandomForestClassifier() . But looking at the accuracy and other metrics such as precision, recall and f1 score, the model is not performing well because of the imbalanced nature of the data set. However, we cannot use sampling techniques to overcome this issue in case of NLP problems as Sample techniques in nlp can distort the data. Hence, we can overcome this issue only by collecting more data in a form that the data set is balanced. We'll still try to hyper parameter tune the model at which we achieved the best accuracy and try increasing the accuracy. In this case, we have achieved maximum accuracy for RandomForestClassifier.

Visualizations

- **Countplot -**

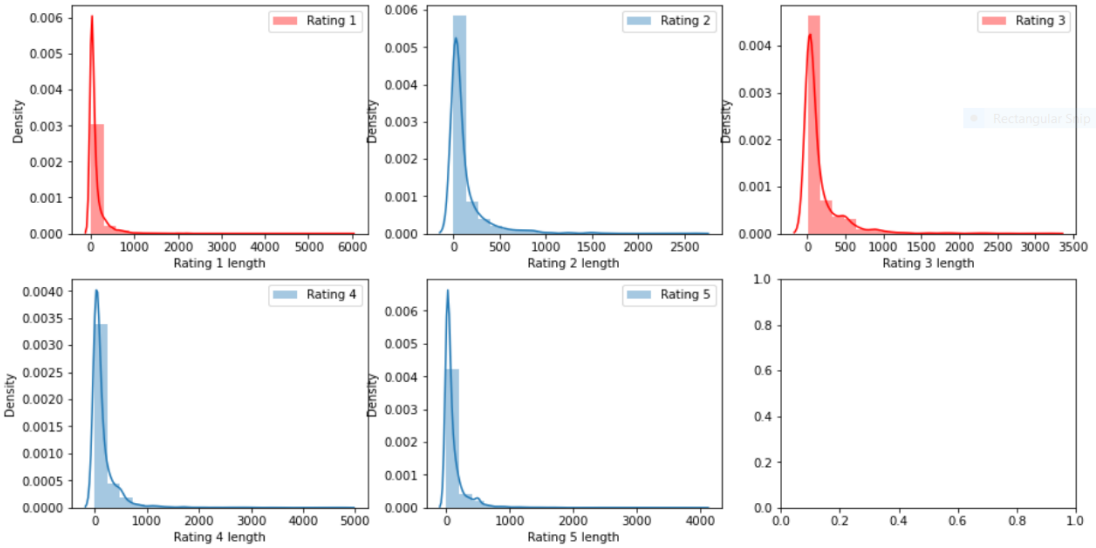
```
5     8451
1     5243
4     3501
3     1951
2     1503
Name: Ratings, dtype: int64
```



Maximum records are for 'Rating 5' and 'Rating 1' and least for 'Rating 2' which indicates that the data set is imbalanced in nature.

- **Density Graph -**

Before Data Cleaning



The length of original text for each Rating is as follows:

Rating 1 = 6000 texts

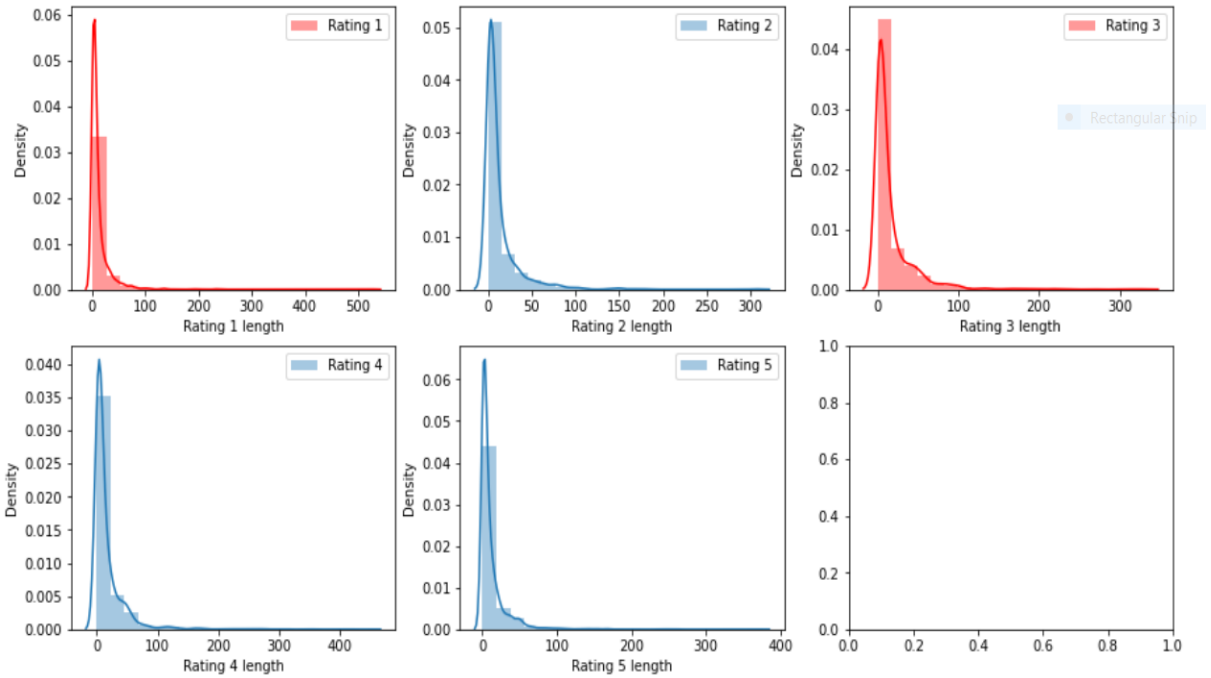
Rating 2 = More than 2500 texts

Rating 3 = Around 3500 texts

Rating 4 = 5000 texts

Rating 5 = More than 4000 texts

After Data Cleaning



The length of cleaned text for each Rating is as follows:

Rating 1 = More than 500 texts

Rating 2 = More than 300 texts

Rating 3 = More than 300 texts

Rating 4 = More than 400 texts

Rating 5 = Less than 400 texts

Conclusion

- Facing all the obstacles in data scraping, cleaning data, analyzing data, comparing different algorithms and hypertuning the best one, Random Forest Classifier without hypertuning it concluded to be the best model for the problem.

- Several challenges regarding installing libraries, resolving errors, data cleaning were faced during model building as it was very challenging to identify the right preprocessing elements and steps to be performed for this use case. Lot of time was consumed resolving errors and installing libraries and also few libraries were unable to install because of the current version used. Hence, various changes and computations were required to be done to make few libraries, attributes and functions work. The task faced too many hurdles in its way but still achieving a good accuracy became unsuccessful due to unbalanced data set scrapped. Therefore, this project needs further improvisation to it by adding more data and making the data set balanced and then trying the model on it. The assumption of attaining a good accuracy is made if a few balanced records are added to the existing data.

- The limitation of this project is that use of **Sampling Techniques** cannot be made to solve the imbalance nature of the data set as it can distort the data. Hence, if the data set is imbalanced, achieving a high accuracy becomes difficult.