Major Research Project

On

Prediction of Product Rating on E-commerce using Text Mining, Analytics & NLP

Submitted By:

Ruchika Mehta
2k21/DMBA/104

Under the Guidance of
Dr. P.K. Suri

DELHI SCHOOL OF MANAGEMENT
Delhi Technological University
Bawana Road, Delhi- 110042

CERTIFICATE

This is to certify that **Ruchika Mehta 2K21/DMBA/104** has submitted the project report titled "**Prediction of Product Rating on E-commerce using Text Mining, Analytics & NLP**" in partial fulfilment of the requirements for the award of degree of Master of Business Administration (MBA) from Delhi School of Management, Delhi Technological University New Delhi during the academic year 2022-23.

Dr. P.K Suri
Professor

Dr. Archana Singh
Head of the Department

DECLARATION

I, Ruchika Mehta, hereby declare that the presented report of major research project titled "**Prediction of Product Rating on E-commerce using Text Mining, Analytics & NLP"** is uniquely prepared by me.

I also confirm that the report is only prepared for my academic requirement, not for any other purpose. It might not be used with the interest of the opposite party of the corporation.

…….

Ruchika Mehta
2k21/DMBA/104
MBA
Delhi School of Management
Delhi Technological University

ACKNOWLEDGEMENT

I, Ruchika Mehta would like to convey my gratitude to the Head of Department (HOD) of Delhi School of Management, Delhi Technological University for emphasizing on the major research project.

I am greatly indebted to my mentor and Hod for providing their valuable guidance at all stages of the study, their advice, constructive suggestions, positive and supportive attitude and continuous encouragement, without which it would have not been possible to complete the project.

I would also like to thank Prof. P.K. Suri, who in spite of busy schedule has co-operated with me continuously and indeed, his valuable contribution and guidance have been certainly indispensable for my project work.
I extend my warm gratitude and regards to everyone who helped me during internship.

…….

Ruchika Mehta
2k21/DMBA/104
MBA
Delhi School of Management
Delhi Technological University

EXECUTIVE SUMMARY

In this digital era, multiple terabytes/petabytes of data are being generated every day. With the boost in e-commerce industries over past few years, it has become important for those organizations to provide best services to their customers in order to retain existing customers and to bring new customers. Customers generates plenty of data every day in form of emails, reviews, feedbacks, enquiries, etc. Most of such data falls in category of Text, hence it is has become important for an organization to analyze such data in most optimal manner. It will help them to target specific customers, increase sales, make improvements in products and services.

Text Analytics has shown its evident importance in recent few years. It has uncovered various opportunities for the companies to grow in various verticals. This project is focused on how human natural language can be interpreted to perform analytics on it to discover new ideas and improvements to help companies to satisfy and target their customers in an effective manner. The algorithms used in this project will represent their power to understand the sentiments of the customer reviews and/or will categorize the most important text of the reviews generated for e- commerce company.

Contents

CHAPTER 1: INTRODUCTION

1.1 Background

In modern era, where more than 80% of data is generated in form of plain text i.e., unstructured data from emails, messages, blogs, reviews, etc. It has become important to make machines understand human language and analyze the voluminous data that is being generated. Text Analytics enable users to transform unstructured data and uncovers hidden insights in the data.

By understanding Natural Language, businesses can gain insights about the people sentiments towards different products. They can also understand which product is being discussed the most on social platforms. In past few years, text analytics has enabled Business users to grow their business at large extent.

1.2 Introduction

Humans speaks and writes in different languages based on the geographical and their region accepted languages. It is difficult to draw insights from the regional languages however text processing, with the help of Natural Language Understanding has become matured in field of text analytics to process English language which is widely accepted in most of the regions of the world. It enables users to understand the text written by customers/users of products or services from various channels such as blogs, emails, reviews, feedbacks, enquiries and many more.

The rating of stars, i.e., 1 to 5 stars on various e-commerce websites, has their corresponding text content which explains overall product quality overview. This numerical information is the first element used by the consumers to compare products before making their purchase decision. However, the reviews are entered by human.
which can be quite different from the rate of text content. Really, users have different levels and do not measure the product in the same way. For example, a user can

rate the product well and give it a 5-star rating while another user can write similar comments and give only 3 stars. Additionally, customer reviews may contain anecdotal, not necessarily true or reliable, based on personal accounts rather than true facts or research.

In subsequent chapters, we will learn about how text data can be used to perform analytics with the help of Machine Learning algorithms and natural language understanding. It will also represent how text of the reviews written by customers can be helpful for the companies to categorize sentiments of the customers. Using natural language processing, companies can also understand if the rating provided by the customers against the review text they have written is aligned with the magnitude of the words they have used.

1.3 Objectives of the Study

- Understand the sentiments of people towards a product
- Calculate the future rating of the product and predict deviations from actual rating.

1.4 Methodology

First of all, we will clean the data and data preprocessing will be done. In this type of data, multi classification is there, so with the help of previously used algorithms can be extended to a particular case of having more than two classes, we will address this problem. Some classifiers or algorithms are specifically designed to solve multi-class problems such as Naïve Bayes and SVM. Then the algorithm will the highest accuracy will be selected.

CHAPTER 2: NATURAL LANGUAGE PRE- PROCESSING

The language is a natural thing. But mastering a new language at first is difficult. If anyone has ever chosen to speak a language other than their mother tongue, they will understand! There are so many layers of peeling and syntax to consider - it's quite a tough challenge.

In any Natural Language Processing project, the goal is to train a model which can analyse and make predictions. However, before training any machine learning model, one needs to perform Data Pre-processing which is an essential step. The quality and testing metrics are highly impacted by the level of data cleaning or data pre-processing performed on the data.

There are various text pre-processing steps to clean the data and get better results at the time of prediction:

2.1 Tokenization:

In order to train computers to understand text or sentences, there is a need to break down the sentences into a way that computers can understand. This is where the concept of tokenization comes into place which converts the sentences into collection of words to make machines understand. This is a crucial step to be performed while working with any Natural Language project.

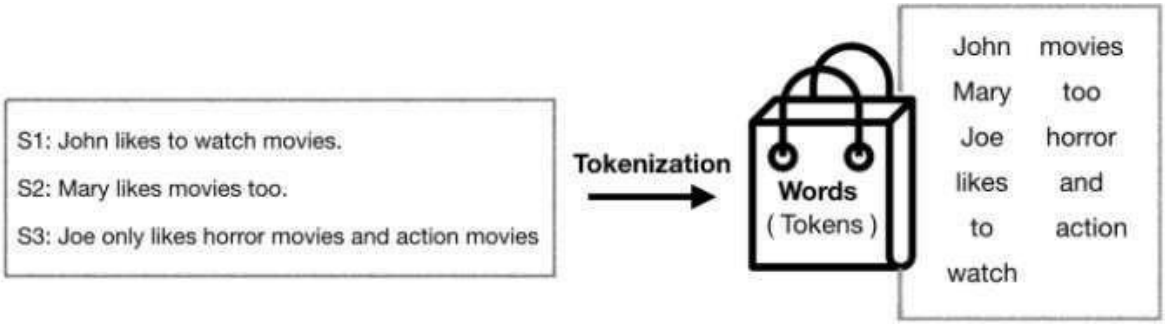
Tokenization is a process of dividing a piece of text into smaller parts known as tokens. Tokens can be words, collection of words, or characters. Tokenization can be broadly classified into 3 categories i.e., word tokenization, collection of words tokenization and character tokenization.

Example:

"This is the report of NLP project"

After applying tokenization on the above sentence, it will be:

["This", "is", "the", "report", "of", "NLP", "project"]



2.2 Lower Casing:

When we work on raw text, same words can be repeated multiple times by a same user or different users by using different cases of the text. In order to reduce the size of the vocabulary of our text data and to do parsing of the text data, conversion of data into uniform case is another important step in text pre-processing. As a general practice, words are converted to lower-case however they can be converted into upper case as well.

As an example, using python, conversion of text into lowercase can be done:

```
def lowercase_text(text):
    return text.lower()

input_str = "This is a BITS NLP dissertation project by Kashish Gakkar"
lowercase_text(input_str)
```

```
'this is a bits nlp dissertation project by kashish gakkar'
```

2.3 Removal of Numbers:

Another step in pre-processing is to remove numbers from the data. Since numbers do not play significant role in playing with textual data, numbers can be removed or converted into textual representations. Regular expressions(re) can be used to remove the numbers.

Example:

```
# For Removing numbers
import re
def remove_num(text):
    result = re.sub(r'\d+', '', text)
    return result

input_s = "There are 5 semesters in BITS Msc. Business Analytics course and 4 semesters are completed"
remove_num(input_s)
```

```
'There are  semesters in BITS Msc. Business Analytics course and  semesters are completed'
```

To convert numbers into text:

```
# convert number into text
def convert_num(text):
    # split strings into list of texts
    temp_string = text.split()
    # initialise empty list
    new_str = []

    for word in temp_string:
        # if text is a digit, convert the digit
        # to numbers and append into the new_str list
        if word.isdigit():
            temp = q.number_to_words(word)
            new_str.append(temp)

        # append the texts as it is
        else:
            new_str.append(word)

    # join the texts of new_str to form a string
    temp_str = ' '.join(new_str)
    return temp_str

input_str = 'There are 5 semesters in BITS Msc. Business Analytics course and 4 semesters are completed'
convert_num(input_str)
```

```
'There are five semesters in BITS Msc. Business Analytics course and four semesters are completed'
```

2.4 Removal of Punctuations:

Punctuations are removed from textual data because they are treated as noise in the data. If we don't remove punctuations such as ‘,‘ ‘.’, etc., they will be treated separately and may affect the accuracy during prediction. In other words, punctuation can decrease the signification of the actual verbs or nouns used in the text.

```
# let's remove punctuation
import string
def rem_punct(text):
    translator = str.maketrans('', '', string.punctuation)
    return text.translate(translator)

input_str = "There are 5 semesters in BITS Msc. Business Analytics course, and 4 semesters are completed!!"
rem_punct(input_str)
```

```
'There are 5 semesters in BITS Msc Business Analytics course and 4 semesters are completed'
```

2.5 Removal of Stop Words:

Stop words are words that do not contribute to the meaning of the sentence. Hence, they can be safely removed without causing any change in the meaning of a sentence. The NLTK (Natural Language Toolkit) library has the set of stop words and we can use these to remove stop words from our text and return a list of word tokens. However, additional stop words as per knowledge of the data can also be added in the list.

Below is the example which demonstrates how to remove stop words and apply tokenization:

```
# importing nltk library
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import nltk

nltk.download('stopwords')
nltk.download('punkt')

# remove stopwords function
def rem_stopwords(text):
    stop_words = set(stopwords.words("english"))
    word_tokens = word_tokenize(text)
    filtered_text = [word for word in word_tokens if word not in stop_words]
    return filtered_text

ex_text = "Data is the new oil. A.I is the last invention"
rem_stopwords(ex_text)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\kashi\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\kashi\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
['Data', 'new', 'oil', '.', 'A.I', 'last', 'invention']
```

It can be observed from the output list that stop words such as ‘is’, ‘the’ are removed from the given text.

2.6 Parts of Speech (POS) tagging:

The pos (parts of speech) explains how a word is used in a sentence. In a sentence, a word can have different contexts and semantic meanings. The basic natural language processing (NLP) models like bag-of-words(bow) fails to identify these relations between the words. For that we use pos tagging to mark a word to its pos tag based on its context in the data. Pos is also used to extract relationship between the words.

Example:

```
# importing tokenize library
from nltk.tokenize import word_tokenize
from nltk import pos_tag
nltk.download('averaged_perceptron_tagger')

# convert text into word_tokens with their tags
def pos_tagg(text):
    word_tokens = word_tokenize(text)
    return pos_tag(word_tokens)

pos_tagg('This is a BITS project.')
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\kashi\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping taggers\averaged_perceptron_tagger.zip.
```

```
[('This', 'DT'),
 ('is', 'VBZ'),
 ('a', 'DT'),
 ('BITS', 'NNP'),
 ('project', 'NN'),
 ('.', '.')]
```

In the above example DT stands for Determiner, VBZ stands for Verb (3rd person singular present), NNP stands for Proper noun, PRP stands for personal noun, NN as Noun. We can get all the details pos tags using the Penn Treebank tagset.

```
# downloading the tagset
nltk.download('tagsets')

# extract information about the tag
nltk.help.upenn_tagset('NN')
```

```
NN: noun, common, singular or mass
    common-carrier cabbage knuckle-duster Casino afghan shed thermostat
    investment slide humour falloff slick wind hyena override subhumanity
    machinist ...
```

2.7 Stemming:

Stemming is the process of getting the root form of a word. Root or Stem is the part to which in flexional affixes (like -ed, -ize, etc.) are added. We would create the stem words by removing the prefix of suffix of a word. Hence, stemming a word may not result in actual words of the language.

If the sentences are not in tokens, then they need to be converted it into tokens. Once strings of text are converted into tokens, then those word tokens can be converted into their root form. These are various types of stemmers available:

- Porter stemmer
- Snowball stemmer, and
- Lancaster Stemmer

However, Porter stemmer is mostly used amongst all. Example of porter stemmer:

```
#importing nltk's porter stemmer
from nltk.stem.porter import PorterStemmer
from nltk.tokenize import word_tokenize
stem1 = PorterStemmer()

# stem words in the list of tokenised words
def s_words(text):
    word_tokens = word_tokenize(text)
    stems = [stem1.stem(word) for word in word_tokens]
    return stems

text = 'There are 5 semesters in BITS Msc. Business Analytics course, and 4 semesters are completed!!'
s_words(text)
```

```
['there',
 'are',
 '5',
 'semest',
 'in',
 'bit',
 'msc',
 '.',
 'busi',
 'analyt',
 'cours',
 ',',
 'and',
 '4',
 'semest',
 'are',
 'complet',
 '!',
```

From above result, ‘semester’ is converted to ‘semest’ however it is not an actual English word.

2.8 Lemmatization:

As stemming, lemmatization do the same task but the only difference is that lemmatization ensures that root word belongs to the language itself. To get the valid words, lemmatization should be used. In NLTK (Natural language Toolkit), WordLemmatizer is used to get the lemmas of words. Context for the lemmatization also needs to be provided. Hence, pos(parts-of-speech) is added as a parameter.

2.9 Named Entity Recognition

It is the process to extract information from unstructured text. It is used to classify the entities which is present in the text into categories like a person, organization, event, places, etc. This provides detailed knowledge about the text and the relationship between the different entities.

Example:

In this chapter, we have covered most of the pre-processing techniques for processing natural language data. However, there can be some other concepts which are not in scope for this project.

Also, another important point to note here is that one needs to apply above preprocessing steps as per the domain of the data and not all the steps are required to be used in all types of natural language processing problems.

CHAPTER 3: DATA COLLECTION AND EXPLORATION

3.1 Data Collection

With the boost in e-commerce industries over past few years, it has become important for those organizations to provide best services to their customers in order to retain existing customers and to bring new customers. Customers generates plenty of data every day in form of emails, reviews, feedbacks, enquiries, etc. Most of such data falls in category of Text, hence it is has become important for an organization to analyze such data in most optimal manner. It will help them to target specific customers, increase sales, make improvements in products and services.

Almost every organization these days generate plenty of data each day which consists of more than 80% of textual data in form of reviews, enquiries, feedbacks, etc.

In this project, we will work on reviews text generated on an e-commerce platform.

Data is collected from a public repository i.e., Kaggle. The data is focused on 'Cell phones and Accessories' category of an e-commerce platform. This data will be used to train the model and test the model.

https://www.kaggle.com/zainabhaddad/reviews-cell-phones-and-accessories-5

Data Format: JSON

Each product review in the file is provided with the following labels, below are the description:

1.4.1 reviewerID: the ID of the reviewer

1.4.2 asin (Amazon Standard Identification Number): the product ID of the item being reviewed

1.4.3 reviewer Name: the name of the reviewer

1.4.4 helpful: helpfulness rating of the review (fraction of users who found the review helpful)

1.4.5 reviewText: the text of the review corresponding to the comment of the reviewer

1.4.6 overall: the rating of the product, out of five stars

1.4.7 summary: the summary of the review

1.4.8 unixReviewTime: time of the review (unix time)

1.4.9 reviewTime: time of the review in mm/dd/yyyy

3.2 Data Exploration

Exploratory data analysis is one of the most important parts of any machine learning task and Natural Language Processing also requires plenty of data exploration to solve any business problem.

Looking at the head (top 5 rows) of the dataset, we have 9 columns in total:

```
reviews_df.head()
```

	reviewerID	asin	reviewerName	helpful	reviewText	rating	summary	unixReviewTime	reviewTime
0	A30TL5EWN6DFXT	120401325X	christina	[0, 0]	They look good and stick good! I just don't li...	4.0	Looks Good	1400630400	05 21, 2014
1	ASY55RVNIL0UD	120401325X	emily l.	[0, 0]	These stickers work like the review says they ...	5.0	Really great product.	1389657600	01 14, 2014
2	A2TMXE2AFO7ONB	120401325X	Erica	[0, 0]	These are awesome and make my phone look so st...	5.0	LOVE LOVE LOVE	1403740800	06 26, 2014
3	AWJ0WZQYMYFQ4	120401325X	JM	[4, 4]	Item arrived in great time and was in perfect ...	4.0	Cute!	1382313600	10 21, 2013
4	ATX7CZYFXI1KW	120401325X	patrice m rogoza	[2, 3]	awesome! stays on, and looks great. can be use...	5.0	leopard home button sticker for iphone 4s	1359849600	02 3, 2013

Below are the columns that are in the training dataset:

```
reviews_df.columns
```

```
Index(['reviewerID', 'asin', 'reviewerName', 'helpful', 'reviewText', 'rating',
       'summary', 'unixReviewTime', 'reviewTime'],
      dtype='object')
```

```
Data columns (total 10 columns):
reviewerID        194439 non-null object
asin              194439 non-null object
reviewerName      190920 non-null object
helpful           194439 non-null object
reviewText        194439 non-null object
rating            194439 non-null float64
summary           194439 non-null object
unixReviewTime    194439 non-null int64
reviewTime        194439 non-null object
```

In this project, we will predict ratings using the text of the reviews. Let us understand few characteristics of ratings variable:

	rating
count	194439.000000
mean	4.129912
std	1.222499
min	1.000000
25%	4.000000
50%	5.000000
75%	5.000000
max	5.000000

Mean value in above figure represents that people have rated the products better under 'Cell phones and accessories category'.

Visualizations can help understand the data well. Below are the text statistics visualizations which are easy to understand and generate meaningful insights. These visualizations help in **exploration of fundamental characteristics** of the text data.

Understand the distribution of ratings present in the given data:

```
reviews_df['rating'].value_counts()
```

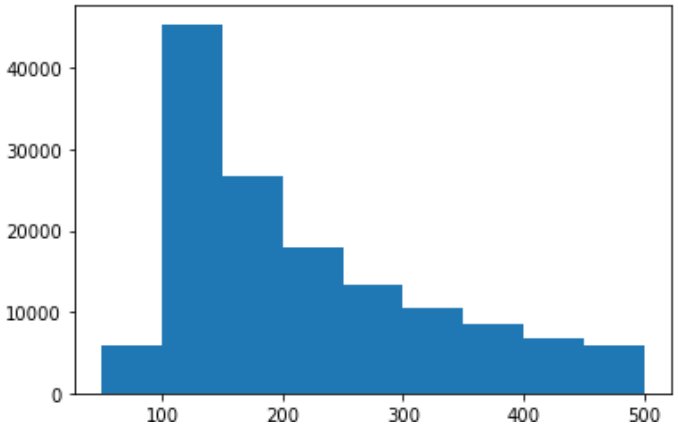
```
5.0    108664
4.0     39993
3.0     21439
1.0     13279
2.0     11064
```

Understand the distribution of number of characters present in reviews text in the given data:

```
reviews_df['reviewTextLen'] = reviews_df['reviewText'].str.len()
```

```
plt.hist(reviews_df['reviewTextLen'], bins = [50,100,150,200,250,300,350,400,450,500])
```

```
(array([ 5846., 45395., 26611., 18019., 13259., 10446.,  8498.,  6753.,
         5914.]),
 array([ 50, 100, 150, 200, 250, 300, 350, 400, 450, 500]),
 <a list of 9 Patch objects>)
```

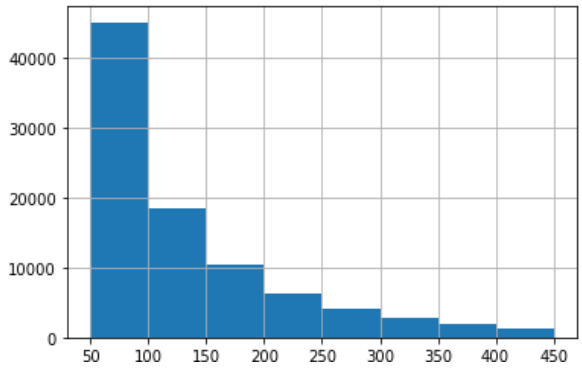


It can be observed that the data is positively skewed which clearly depicts that a greater number of people put reviews length between 100-200 characters and a small subset of people is adding reviews up to 500 characters as well.

Now, performing data exploration at a word-level. Plotting the number of words appearing in each text review.

```
reviews_df['reviewText'].str.split().\
    map(lambda x: len(x)).\
    hist(bins = [50,100,150,200,250,300,350,400,450])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ac095eee08>
```

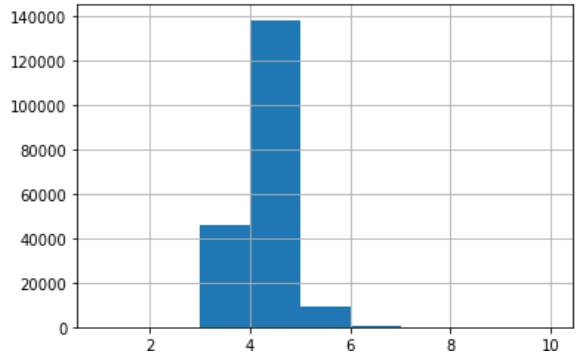


Above figure depicts that more than 40% of population is using 50 words per review and the data is highly skewed towards right.

Looking at the average word length used in reviews, it is found that on an average word length between 3 to 6 is used in the reviews:

```
reviews_df['reviewText'].str.split().\
   apply(lambda x : [len(i) for i in x]). \
   map(lambda x: np.mean(x)).hist(bins = [1,2,3,4,5,6,7,8,9,10])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ac09634c88>
```



Question: *The average word length ranges between 3 to 6 with 5 being the most common length. Does it mean that people are using really short words in e-commerce reviews?*

One reason why this is not true is with stop words. Stop words are the most commonly used words in any language such as “the”, “a”,” an”, etc. Since these words are probably shorter in length these words may have caused the above graph to be impacted.

Analyzing the number and types of stop words can give us a better understanding of the details.

To find a corpus containing stop words you can use the nltk library. Nltk contains stop words from many languages. Since in this project, we are dealing with English reviews only, fetching out English words from the corpus.

```
from collections import defaultdict

corpus=[]
new = reviews_df['reviewText'].str.split()
new=new.values.tolist()
corpus=[word for i in new for word in i]

dic=defaultdict(int)

for word in corpus:
    if word in stop_words:
        dic[word]+=1
```

```
dic
```

```
defaultdict(int,
            {'and': 476462,
             'just': 55600,
             "don't": 28829,
             'the': 861412,
             'because': 31273,
             'was': 107405,
             'it': 352038,
             'up': 37639,
             "won't": 7193,
             'a': 446772,
```

From above figure, it can be evidently seen that stop words such as “the”,” and”, “to”, etc. dominating average length of words in reviews text as the count of each stop word is quite high in numbers. Below are the top 10 dominating stop words:

```
sorted(dic, key=dic.get, reverse=True)[:10]
```

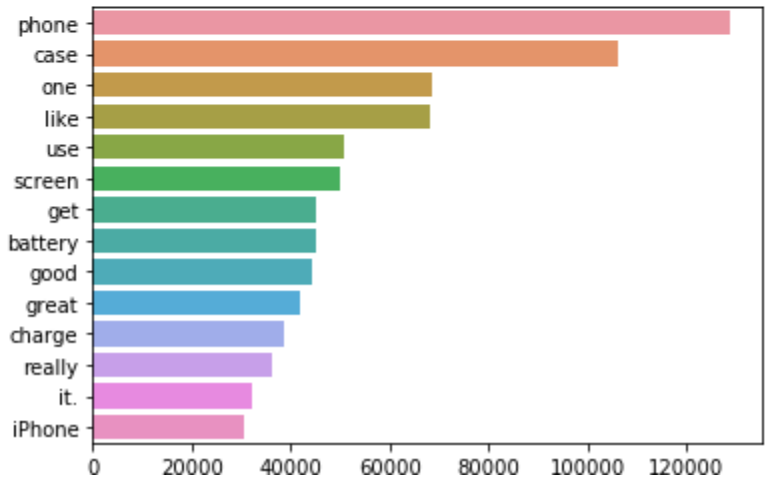
```
['the', 'and', 'to', 'a', 'it', 'is', 'of', 'for', 'my', 'this']
```

So now that we know which stop words appear more often in reviews text, let's check which words besides these stop words appear more often.

```
from collections import Counter
counter=Counter(corpus)
most=counter.most_common()
x, y= [], []
for word,count in most[:70]:
    if (word.lower() not in stop_words):
        x.append(word)
        y.append(count)

sns.barplot(x=y,y=x)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1acaa180548>
```



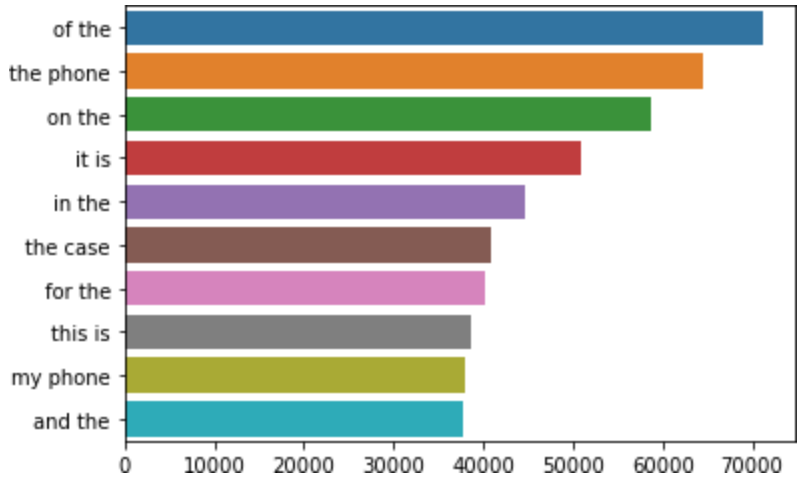
It is evident that the data we are analyzing is for phones and its accessories and from above figure also, we are able to see highest frequency for words like ‘phone’, ‘case’, ‘screen’, etc.

N-Grams

N-grams are simply **contiguous collection of n words**. For example “Bits Pilani”,” Natural Language Processing” etc.
If the number of words in the collection of words is two, it is called bigram. For 3 words in a collection, it is called a trigram and so on.

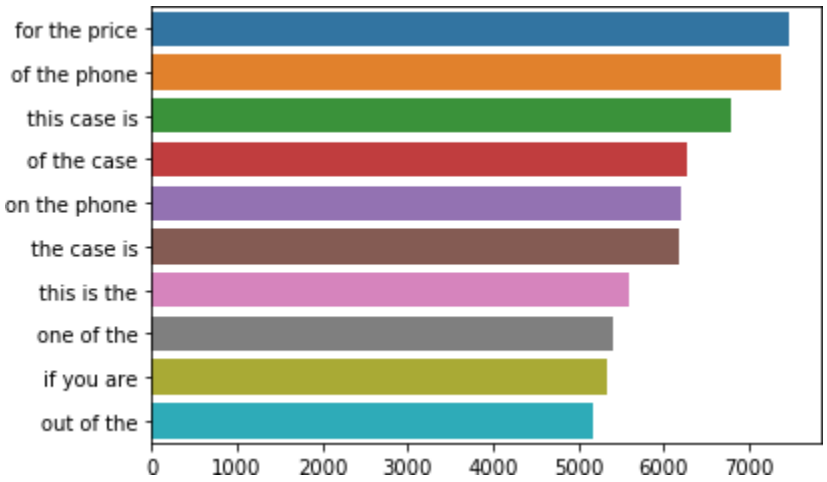
To build a representation of the vocabulary used in the available reviews data, there are various techniques available however we will use *Countvectorizer*. ***Countvectorizer* is a technique used for tokenization, vectorization and represent the corpus in an appropriate form.**

Analyzing bi-grams in reviews data by finding top 10 bi-grams that are used most frequently in the available reviews:



Few of the most dominating bi-grams are related to phone i.e. ‘the phone’ and case i.e. ‘the case’.

Let’s analyze tri-grams now:



Tri-grams have provided interesting results that the most reviews have talked about price which can be seen at the top tri-gram i.e. ‘for the price’.

CHAPTER 4: DATA PRE-PROCESSING

As we discussed in chapter 2 (Natural Language Pre-processing), there are several pre-processing steps to be performed while working with data which consists Natural Language. In this chapter, we will look at different steps that have been taken to clean reviews data.

The data being used in this project contains various reviews which talks about different related products as well. In various reviews, users can provide URLs of various websites, email addresses, some special characters, emoticons, etc.

Any natural language content can contain many fillers and text which does not hold significance to perform analysis on data as we have already discussed in the previous chapters.

4.1 Removal of junk characters

As an initial step, we will remove all the junk characters, email addresses, URLs, etc. from the reviews text.

	reviewerID	asin	reviewerName	helpful	reviewText	rating	summary	unixReviewTime	reviewTime	reviewTextLen	processedReviewText
0	A30TL5EWN6DFXT	120401325X	christina	[0, 0]	They look good and stick good! I just don't li...	4.0	Looks Good	1400630400	05 21, 2014	189	they look good and stick good i just do not li...
1	ASY55RVNIL0UD	120401325X	emily l	[0, 0]	These stickers work like the review says they ...	5.0	Really great product.	1389657600	01 14, 2014	160	these stickers work like the review says they ...
2	A2TMXE2AFO7ONB	120401325X	Erica	[0, 0]	These are awesome and make my phone look so st...	5.0	LOVE LOVE LOVE	1403740800	06 26, 2014	166	these are awesome and make my phone look so st...

In above data, we can observe that *reviewText* has been converted to processedReviewText which does not contain any characters such as exclamation marks, commas, etc. It has also removed all set of html tags as well. This step is a crucial step before applying other data cleansing techniques and before fitting any model for prediction.

4.2 Tokenization and Lemmatization

Tokenization and Lemmatization needs to be applied on data to generate set of words which plays important role in predicting ratings. Tokenization and Lemmatization concepts are already discussed in chapter 2.

Each sentence will be broken into set of words and any word with same English language significance will be converted into their base word such as:
'Run', 'Ran', 'Runs', 'Running' -> 'Run'
All above words have the same significance but are different because of the tense of the text. Using Lemmatization, we will convert such words to 'Run' which is the base words for all of above words.

4.3 Stop words and punctuations removal

Although data is pretty much in clean state however, it still contains all the stop words and punctuation marks which are not required for data analysis. We will remove these stop words by using inbuilt set of English words under NLTK library in python.

After performing 4.1, 4.2 and 4.3, a given review will be converted into below:

Original text:

```
They look good and stick good! I just don't like the rounded shape because I was always bumping it and Siri kept popping up and
it was irritating. I just won't buy a product like this again
```

Converted text:

```
look good stick good like rounded shape always bump pron siri keep pop pron irritate buy product like
```

4.4 Vectorization

Vectorization is an approach to transform set of words aka. Documents into set of real numbers. There are various methods to transform text to vectors however, in this project, we will discuss about Tf-Idf vectorization.

Term Frequency(TF)

It can be defined as the number of times a word appears in a document divided by the total number of words in this document. Every text has its term frequency.

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}}$$

Inverse Data Frequency (IDF)

It is calculated by taking log of Total number of documents divided by the number of documents containing the word (w). The Inverse data frequency determines the weight of less occurring words in all documents in the corpus.

$$idf(w) = log(\frac{N}{df_t})$$

Tf-Idf is computed by the multiplication of TF and IDF i.e., tf*idf:

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

4.5 Pre-processing of train and test data

All above pre-processing steps will be applied to training and testing data in the following chapter. This will make sure that we are following consistency with the train data and test data by removing in-significant words from the data and only retaining important words for predictions.

CHAPTER 5: MODEL TRAINING

Prediction of ratings from text reviews cannot be achieved by people given the large number of data sets available. Fortunately, machine learning and text classification Strategies that have been widely studied over the years have enabled automated learning from raw data.

The purpose of this section is to describe the various methods and techniques that have been developed recently to take on the challenge of predicting ratings from the text review itself. This section focuses on different approaches to benefit from this text-classification function.

In this chapter, we will attempt to predict the exact score of the rating based on its text content. We have already discussed numerous features from the text in order to enhance the performance of the classifiers in the previous chapters.

Multi-class Classification aims to classify values into one more than two classes. Output value can increase to k -class. As shown in the figure below, each possible value prediction corresponds to one different category. Hence, there are five classes which are assigned as per different review text. Classes have a meaning that is equally consistent with that
each event is assigned to one category only. For example, text update cannot be measured with 3 and 4 stars at a time. In short, this method of separating the classes aims to accurately predict the category in which reviews are assigned.

To address this problem of classification, previously used algorithms can be extended to a particular case of having more than two classes. Some classifiers or algorithms are specifically designed to solve multi-class problems such as Naïve Bayes and SVM.

In order to classify text, several defined steps are required to be followed which are depicted in below figure:

Data gathering is done as specified in section 3.1 of this project.

Out of the available columns, only following elements are extracted for analysis:

4.5.1 text reviews

4.5.2 corresponding overall rating of the text review

The remaining columns in the file were considered as insignificant given in framework of text classification, only the text information is considered as useful. To retrieve these pieces of information from the json files, python and Excel were used.

5.1 Choice of thevariables:

As stated in the title of this dissertation, the aim is to predict ratings of Amazon reviews. Therefore, the dependent variable is the numerical rating of a specific review rated by a consumer. The popular Amazon's rating system works along with a five-star system and allocates a certain number of stars proportionately to the satisfaction of the customers. As illustrated in below figure star corresponds to the lowest rating whereas 5 stars are equivalent to the best rating.

As mentioned in the introduction of this dissertation, the purpose is to predict Amazon reviews. Therefore, the dependent variable is the numerical variable of a particular review. The popular Amazon rating system works in conjunction with a five-star system and offers a certain number of stars equally to customer satisfaction.

5.2 Training a classifier

In the training phase, the algorithm assigns the class to each of the reviews given in the dataset. These steps allow for classification to be read in contexts so that later they can accurately distinguish some new reviews. We see the different classes that can be assigned according to the chosen method. To use this training step, Python's programming language, its nltk, pandas and Scikit-learn libraries are used. First, the training corpus (reviews and related classes) developed in previous steps was treated as a bag-of-words (BoW) and was converted into portable number symbols using the CountVectorizer method. Indeed, the latter allows us to enter a token to obtain a feature dictionary and term document matrix. In order to give each feature a more relevant index than its appearance, the TF-IdfVectorizer is used to determine its tf-idf values. Tf-Idf allows to capture the significance of words (tf) while considering the value (discriminatory power) by assigning them various weights (idf).

Finally, various algorithms will be used to determine which of them was the most appropriate for this text-splitting function. Note that various methods (Naïve Bayes, Random Forest and Support Vector Machines) are trained with the available data set. The code written in Python for text editing can be found in ***Annexure***.

5.3-Evaluation

This step is able to measure performance and evaluate the performance of trained models. In other words, we can see if the classifier has learnt some common terms and is able to predict the correct outcome in new unseen situations.

Different classifiers are tested using two approaches i.e., train-test-split where dataset is divided into 70:30 ratio as training and testing datasets. Another approach is 10-fold cross validation. In addition, classification reports are generated to gauge the performance of the trained models which shows a few basic test metrics like precision, recall and f1 scores.

These are metrics for accuracy so we can compare them afterwards. In the end, confusion matrix is also calculated to get an overall view of the actual values compared to the predicted values for each classifier.

The results of all the classifiers are described in the next chapter

CHAPTER 6: MODEL RESULTS AND SELECTION

6.1 Naïve Bayes

Multinomial Naïve Bayes model is trained with the train vectorized dataset and got below results:

Classification report:

	precision	recall	f1-score	support
1.0	0.00	0.00	0.00	3990
2.0	0.00	0.00	0.00	3342
3.0	0.75	0.00	0.00	6328
4.0	0.50	0.00	0.00	12064
5.0	0.56	1.00	0.72	32608
accuracy			0.56	58332
macro avg	0.36	0.20	0.14	58332
weighted avg	0.50	0.56	0.40	58332

Confusion Matrix:

	Predicted 1	Predicted 2	Predicted 3	Predicted 4	Predicted 5
Actual 1	0	0	0	0	3990
Actual 2	0	0	1	0	3341
Actual 3	0	0	3	1	6324
Actual 4	0	0	0	4	12060
Actual 5	0	0	0	3	32605

10-k Cross validation scores:

```
[0.55896469 0.55896469 0.55906052 0.55906052 0.55906052 0.55906052
 0.55906052 0.55906052 0.55888908 0.55888908]
```

6.2 Support Vector Machines

Support vector classifier is trained with the train vectorized dataset and got below results:

Classification report:

	precision	recall	f1-score	support
1.0	0.43	0.58	0.49	3990
2.0	0.20	0.24	0.22	3342
3.0	0.32	0.32	0.32	6328
4.0	0.41	0.30	0.35	12064
5.0	0.77	0.80	0.78	32608
accuracy			0.60	58332
macro avg	0.43	0.45	0.43	58332
weighted avg	0.59	0.60	0.59	58332

Confusion Matrix:

	Predicted 1	Predicted 2	Predicted 3	Predicted 4	Predicted 5
Actual 1	2317	679	377	184	433
Actual 2	958	817	753	340	474
Actual 3	701	934	2024	1303	1366
Actual 4	475	736	1772	3671	5410
Actual 5	923	833	1472	3350	26030

10-k Cross validation scores:

```
[0.57850531 0.59718889 0.59129093 0.59746271 0.5902623  0.58546203
 0.57894737 0.57500429 0.58786216 0.57654723]
```

6.3 Random Forest

Random Forest classifier is trained with 100 trees using the train vectorized dataset and got below results:

Classification report:

	precision	recall	f1-score	support
1.0	0.74	0.16	0.26	3990
2.0	0.43	0.00	0.01	3342
3.0	0.55	0.04	0.07	6328
4.0	0.46	0.03	0.06	12064
5.0	0.58	0.99	0.73	32608
accuracy			0.58	58332
macro avg	0.55	0.25	0.23	58332
weighted avg	0.55	0.58	0.45	58332

Confusion Matrix:

	Predicted 1	Predicted 2	Predicted 3	Predicted 4	Predicted 5
Actual 1	626	5	37	51	3271
Actual 2	147	13	77	83	3022
Actual 3	46	6	233	210	5833
Actual 4	12	2	62	412	11576
Actual 5	18	4	17	137	32432

6.4 Discussion

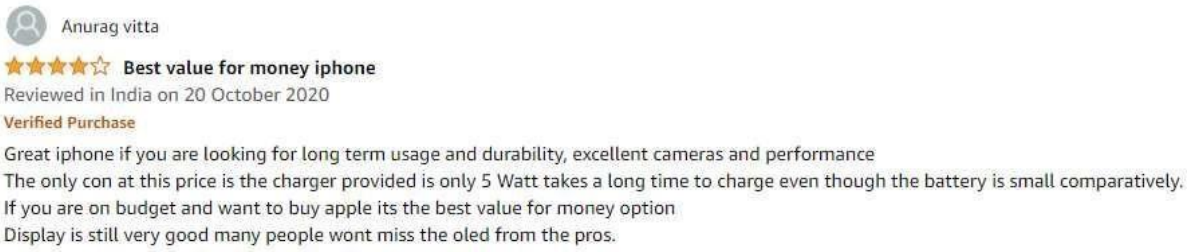
It is observed from the above results that Naïve Bayes model has not performed well on the given dataset. However, precision can be seen higher in the model trained by Random Forest algorithm however recall is poor. The model that has performed the best out of 3 models we have trained is support vector machines which has given 60% accuracy in predictions with balanced precision and recall.

In SVC results, precision and recall are less for classes 2,3 and 4 as compared to classes 1 and 5. This has happened because of imbalanced dataset which have extrapolated the results.

Hence, SVC will be chosen for the predictions of any reviews that needs to be classified in future.

6.5 Predictions

Let us make some predictions using SVC trained classifier and see the results. A random review is picked from Amazon website under ‘cell phones’ category:



The review text is used for making prediction of rating. The review text is pre-processed just like the training dataset. After pre-processing, below sample of words are retained from the review for prediction:

```
txt
```

```
['great',
 'iphone',
 'pron',
 'look',
 'long',
 'term',
 'usage',
 'durability',
 'excellent',
 'camera',
 'performance',
 'con',
 'price',
 'charger',
 'provide',
```

```
pred = executePreocessing("""Great iphone if you are looking for long term usage and durability, excellent cameras and performa
The only con at this price is the charger provided is only 5 Watt takes a long time to charge even though the battery is small
If you are on budget and want to buy apple its the best value for money option
Display is still very good many people wont miss the oled from the pros.""", r'D:\BITS\Semester 5\Final\SVCmodelForPrediction.s
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\kashi\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
pred
```

```
array([4.])
```

It is found that the model has predicted rating of 4 which is matching the actual rating of the review on Amazon’s website.

6.6 Fulfil Business use-case

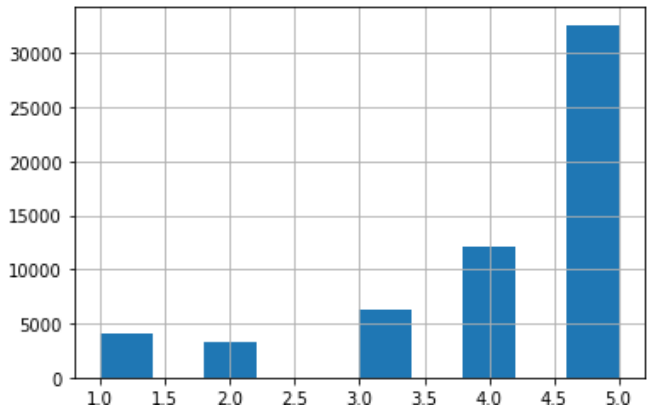
After making predictions using the trained model, organizations can take help to find out the deviations between the actual ratings and the predicted ratings because for a particular product, there are high chances that a user can rate the product well and give it a 5-star rating while another user can write similar comments and give only 3 stars.

Customers can rate similar products differently with similar text of reviews as per their inclination towards product brand, category of the products, etc.

Distribution of actual ratings:

```
final_df['og_rating'].hist()
```

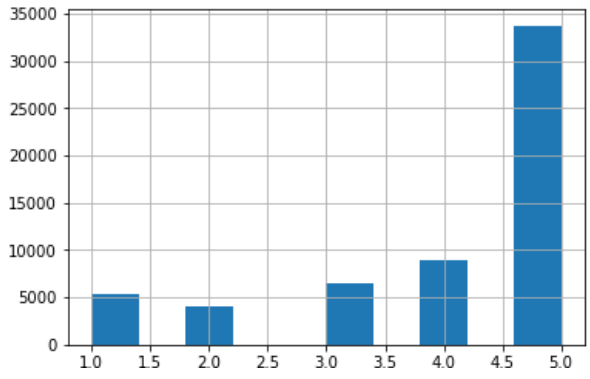
```
<matplotlib.axes._subplots.AxesSubplot at 0x1acd3de4408>
```



Distribution of predicted ratings:

```
final_df['pred_rating'].hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ace2a06b48>
```



Below are the pivot tables that are created to analyze the deviations between actual and predicted ratings for each product ASIN. Pivot table using actual ratings:

```
final_df.pivot_table(index=['asin'], columns='og_rating', aggfunc='size', fill_value=0)
```

og_rating	1.0	2.0	3.0	4.0	5.0
asin					
120401325X	0	0	0	1	1
3998899561	0	0	0	0	2
6073894996	0	2	1	4	5
7532385086	0	0	0	0	1
7887421268	1	0	0	0	0
...	...	...	...	...	...
B00LH1QZR8	0	0	0	0	1
B00LH1R3C4	0	0	0	0	1
B00LH1R5HW	0	0	0	0	2
B00LH1R6SK	0	0	0	0	1
B00LORXVUE	0	0	0	0	4

Pivot table using predicted ratings

```
final_df.pivot_table(index=['asin'], columns='pred_rating', aggfunc='size', fill_value=0)
```

pred_rating	1.0	2.0	3.0	4.0	5.0
asin					
120401325X	0	0	1	0	1
3998899561	0	0	0	0	2
6073894996	2	2	0	1	7
7532385086	0	0	0	0	1
7887421268	0	1	0	0	0
...	...	...	...	...	...
B00LH1QZR8	0	0	1	0	0
B00LH1R3C4	0	0	0	0	1
B00LH1R5HW	1	0	0	0	1
B00LH1R6SK	0	0	0	0	1
B00LORXVUE	1	1	0	0	2

It can be observed that ASIN # B00LORXVUE has overall rating of 5.0 if computed using actual ratings however using predicted ratings, we found actual rating of 3.25.

Depending on the number of stars, they can easily point out poorly rated reviews and therefore improve the quality of their products according to customer requirements.

CHAPTER 7: CONCLUSION

Depending on the positive or negative phrases stored as a number code in nlp can help determine the sentiment towards a product. For e.g. for our product, it is a positive sentiment.

As discussed in discussion section, due to less ratings, product rating was 5.0 but the actual predicted raring is 3.25.

At last, in this project, we have studied the various models that successfully predict the number of user numbers from the content of a review text. To do so, the text fragmentation, which allows the automatic splitting of a document into a set of classes after training on predefined data, is applied to set of reviews from Amazon. The data set relates to two different product categories: cell phones and accessories products. A well-documented speculative task has been investigated for the purpose of predicting the exact amount of each review.

Appendices

Below are the functions used for pre-processing of training and test data:

```
def data_preprocessing_predict(text):
#     text_list = text.split()
    from nltk.corpus import stopwords
    import nltk
    nltk.download('stopwords')
    stop_words=set(stopwords.words('english'))
    nlp = en_core_web_sm.load()  # preprocessing library spacy
    pattern = "@\S+|https?:\S+|http?:\S|[^A-Za-z0-9]+"

    clean_data = []
    doc = nlp(text)
    for token in doc:
        clean = re.sub(pattern, '', str(token.lemma_).lower())
        if clean not in string.punctuation:
            if clean not in stop_words:
                clean_data.append(clean)
    return clean_data
```

```
def executePreocessing(text, modelPath, vectorPath):
    global txt, prediction
    txt = data_preprocessing_predict(text)
    txt1 = ' '.join(txt)
    with open(vectorPath, 'rb') as f:
        vectorizer = pickle.load(f)
    with open(modelPath, 'rb') as f:
        model = pickle.load(f)
    pred_vector_ = vectorizer.transform([txt1])
    prediction = model.predict(pred_vector_)
    if 'svc' not in modelPath.lower():
        predictedProbability = model.predict_proba(pred_vector_)
        if list(predictedProbability.flatten())[0] == list(predictedProbability.flatten())[1] == list(predictedProbability.flatte
            return "UNKNOWN"
        elif list(predictedProbability.flatten())[np.argmax(predictedProbability)] > .5:
            return prediction
        else:
            return "UNKNOWN"
    else:
        return prediction
```

Understanding original vs predicted sentiments

```
final_df['og_sentiment'] = ''
final_df['pred_sentiment'] = ''
og_sentiment = []
pred_sentiment = []
for row in final_df.index:
    if final_df['og_rating'][row] < 3.0 :    og_sentiment.append('negative')
    elif final_df['og_rating'][row] > 3.0:   og_sentiment.append('positive')
    elif final_df['og_rating'][row] == 3.0:  og_sentiment.append('neutral')

    if final_df['pred_rating'][row] < 3.0 :    pred_sentiment.append('negative')
    elif final_df['pred_rating'][row] > 3.0:   pred_sentiment.append('positive')
    elif final_df['pred_rating'][row] == 3.0:  pred_sentiment.append('neutral')

final_df['og_sentiment'] = og_sentiment
final_df['pred_sentiment'] = pred_sentiment
```

```
final_df.head()
```

	review	og_rating	pred_rating	og_review	reviewer_name	asin	og_sentiment	pred_sentiment
156187	iBolt xProDock Active Car DockHolderMount Sams...	5.0	4.0	IBOLT xProDock Active Car Dock/Holder/Mount fo...	LO127	B00C0ZKI9Q	positive	positive
102252	pouch everything look off box commuter case ar...	5.0	3.0	This pouch is everything that I was looking fo...	yogitim	B008FQVO5G	positive	neutral

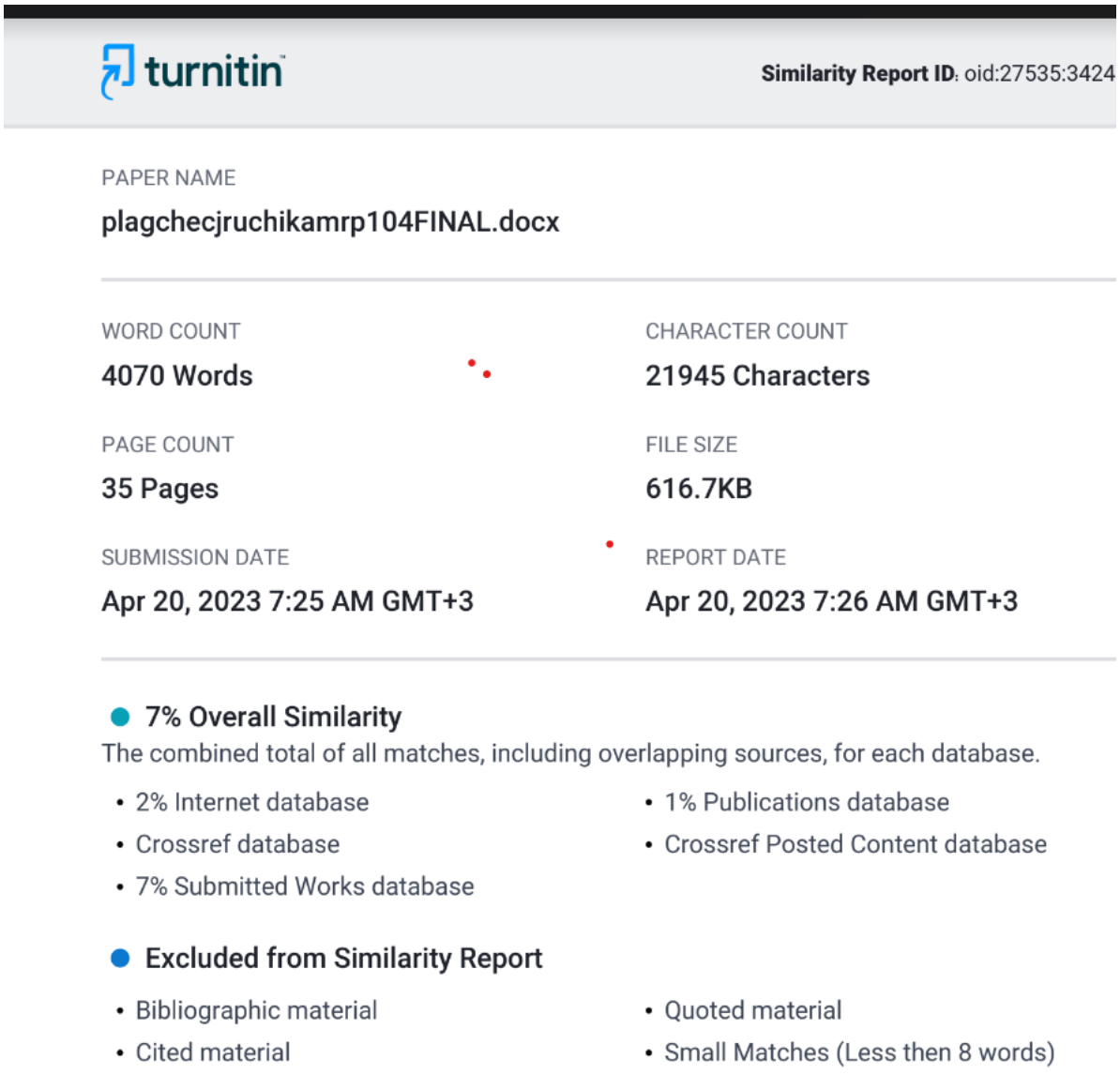
Glossary

- BoW : Bag-of-Words
- FN : False Negative
- FP : False Positive
- NB: Naive-Bayes
- POS: part-of-speech
- SVM: Support Vector Machine
- tf-idf: term frequency-inverse document frequency
- TN: True Negative
- TP: True Positive

References

[1] *Alexander Clark, Chris Fox and Shalom Lappin: Computational linguistics and Natural Language Processing*

Similarity Report



turnitin

Similarity Report ID: oid:27535:3424

PAPER NAME

plagchecjruchikamrp104FINAL.docx

WORD COUNT	CHARACTER COUNT
4070 Words	**21945 Characters**
PAGE COUNT	FILE SIZE
35 Pages	**616.7KB**
SUBMISSION DATE	REPORT DATE
Apr 20, 2023 7:25 AM GMT+3	**Apr 20, 2023 7:26 AM GMT+3**

7% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 2% Internet database
- Crossref database
- 7% Submitted Works database
- 1% Publications database
- Crossref Posted Content database

Excluded from Similarity Report

- Bibliographic material
- Cited material
- Quoted material
- Small Matches (Less then 8 words)