Book Reviews

Elinor Cohen

Today, almost every book has a rating and a review. We can use this information to determine whether people like a book or not

There are 5 options:

1. it was amazing ★★★★★
2. really liked it ★★★★
3. liked it ★★★
4. it was ok ★★
5. did not like it ★

RESEARCH QUESTION:

Did the person like the book based on his review?

CRAWLING

1 SIGN IN

First, I loaded the file that contained my username and password

```
# getting the username and password
text_file = open("keys.txt", "r")
keys = text_file.read().splitlines()
text_file.close()
```

```
sign_in=driver.find_element(By.XPATH,'/html/body/div[2]/div[1]/div/header/div[1]/div/ul/li[1]/a')
sign_in.click()
username = WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.CSS_SELECTOR, "input[name='user[email]']")))
password = WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.CSS_SELECTOR, "input[name='user[password]']")))

username.clear()
username.send_keys(keys[0])
password.clear()
password.send_keys(keys[1])
time.sleep(3)
log_in = WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.CSS_SELECTOR, "input[type='submit']")))
log_in.click()
```

Second, I sent them to their right places and I logged in

This step was necessary to prevent the crawling from getting stuck

NOTE: I had to use selenium to crawl the data properly

2 Book data and reviews link

Book's total rate	4.24
Pages	287
Ratings number	10030
Reviews number	181
Genre	Fiction
Book's month	September
Book's year	2019
Book's cover	Kindle Edition
Author's followers	1,134

Review link	https://www.goodreads.com/review/show/2905270011

Review rate	really liked it
Review date	Jul 21, 2019

- I went through 100 pages that each contained 100 books
- From each page I collected all the information about the book, in addition to some comments' links and its details
- I went through all the comments' links that each contained the full review

3 REVIEWS DATA

- I reduced the amount of reviews to 150,000 due to lack of time.
- The information was divided into 6 so the code could run on 6 different computers.
- Several pieces of information were left out, but I still had enough data to continue.

After crawling: 124,846 rows X 13 columns


DATA CLEANING

Dropping NaN and duplicated rows

Normalization of columns with a wide range of values

Reduction of categories in categorial columns

Natural Language Processing

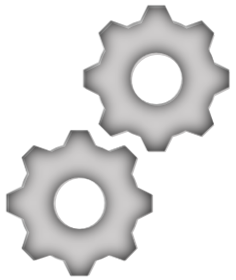
Adding new columns

Convert an object columns to numeric

Genres

- There were 87 genres before I categorized them
- After categorizing:
  Fiction, Nonfiction, Young, Adult, Children, Short, Sequential, New

```
genres_dict = {'Fiction':1, 'Nonfiction':2, 'Young':3, 'Adult':4, 'Childrens':5, 'Short':6, 'Sequential':7, 'New':8}
```



Cover

- There were 19 cover types before I categorized them
- After categorizing:
  Hardcover, Paperback, online, Audiobook

```
{'Hardcover':1, 'Paperback':2, 'online':3, 'Audiobook':4}
```

Year

- There are 2 columns of year, each has a different range
- Reviews year: 2000 – 2022
- Book's year: 1900 – 2022

```
bins=[1900,1950,2000,2005,2010,2015,2030]
labels=[1,2,3,4,5,6]
df['Year_of_book']=pd.cut(df['Year_of_book'],bins,labels=labels)
```

```
bins=[2000,2005,2010,2015,2030]
labels=[1,2,3,4]
df['Review_year']=pd.cut(df['Review_year'],bins,labels=labels)
```

Creating the target column:

- Rating less than 4 – Dislike
- Rating greater than 3 – Like

```
Like = []
for rate in df['Review_rate']:
    if rate<=3:
        Like.append(0)
    if rate>3:
        Like.append(1)
df['Book_like'] = Like
```

0 : Dislike

1 : Like

FOR MACHINE LEARNING:

There were 86,858 values of 1 and 32,044 values of 0. In order to equalize the amounts of values 1 and 0 I dropped randomly 55,000 rows with the 'Book_like' value 1.

BEFORE

1	86858
0	32044

AFTER

0	32044
1	31858


NLP

STEPS TO TAKE:

Word tokenize

Remove “stop words”

Count tags and significant words or groups of words

Make sure the words are in English

Word tagging (Noun, Verb, Adj., Adv.)

Vectorizing top common words

Estimate the word count for each review

Lemmatization

Positive and negative words count

Lemmatizing with assistance function

```
lemmitizer = WordNetLemmatizer()
lemma = []
words_tags = nltk.pos_tag(words)
wordnet_tagged = list(map(lambda x: (x[0], pos_tagger(x[1])), words_tags))
for (word, tag) in wordnet_tagged:
        if tag is None:
            lemma.append(lemmitizer.lemmatize(word))
        else:
            lemma.append(lemmitizer.lemmatize(word,tag))
```

```
def pos_tagger(nltk_tag):
    if nltk_tag.startswith('J'):
        return wordnet.ADJ
    elif nltk_tag.startswith('V'):
        return wordnet.VERB
    elif nltk_tag.startswith('N'):
        return wordnet.NOUN
    elif nltk_tag.startswith('R'):
        return wordnet.ADV
    else:
        return None
```

Counting the persons' names

```
nlp = spacy.load('en_core_web_sm')
Persons = []
for rev in df['Reviews']:
    Persons_Count = 0
    n = nlp(rev).ents
    for x in n:
        if x.label_ == 'PERSON':
            Persons_Count += 1
    Persons.append(Persons_Count)
df['Num_of_names'] = Persons
```

```
pos = []
neg = []
for rev in df_c['Tokenized']:
    pos_count = 0
    neg_count = 0
    for word in rev:
        if word in negative_words:
            neg_count += 1
        elif word in positive_words:
            pos_count += 1
    pos.append(pos_count)
    neg.append(neg_count)
df_c['Positive'] = pos
df_c['Negative'] = neg
```

- I created two lists of words : one of positive words and one of negative ones.
- I added columns to the dataframe for each positive and negative words' amount found in each review.

NOTE: Some of the words could appear in different contexts, which would hinder the prediction result

COUNTVECTORIZER

Excludes stop words

Range of up to 2 words

5 most common words

```
count_vec = CountVectorizer(stop_words='english',ngram_range=(1, 2), max_features= 5)
count_vec.fit(df['Updated_Review'])
text_vectorized = pd.DataFrame(count_vec.transform(df['Updated_Review']).toarray(),
                               columns=count_vec.get_feature_names())
```

I created a dataframe listing the 5 most common words in the text and how many times each appears, and attached it to the existing dataframe

A column with numeric values is essentially a collection of words that share an attribute

15	Num_of_names	63890 non-null	int64
16	Tokenized	63890 non-null	object
17	Words_amount	63890 non-null	int64
18	Noun	63890 non-null	int64
19	Adjective	63890 non-null	int64
20	Adverb	63890 non-null	int64
21	Verb	63890 non-null	int64
22	Other_words	63890 non-null	int64
23	Updated_Review	63890 non-null	object
24	Book_like	63890 non-null	int64
25	Positive	63890 non-null	int64
26	Negative	63890 non-null	int64
27	not_good	63890 non-null	int64
28	i_love_this_book	63890 non-null	int64
29	i_loved_this_book	63890 non-null	int64
30	love_this_book	63890 non-null	int64
31	loved_this_book	63890 non-null	int64
32	i_hate_this_book	63890 non-null	int64
33	one_of_the_worst_books	63890 non-null	int64
34	one_of_the_best_books	63890 non-null	int64
35	worst_book	63890 non-null	int64
36	best_book	63890 non-null	int64

book	63890 non-null	int64
like	63890 non-null	int64
love	63890 non-null	int64
make	63890 non-null	int64
read	63890 non-null	int64
story	63890 non-null	int64
time	63890 non-null	int64

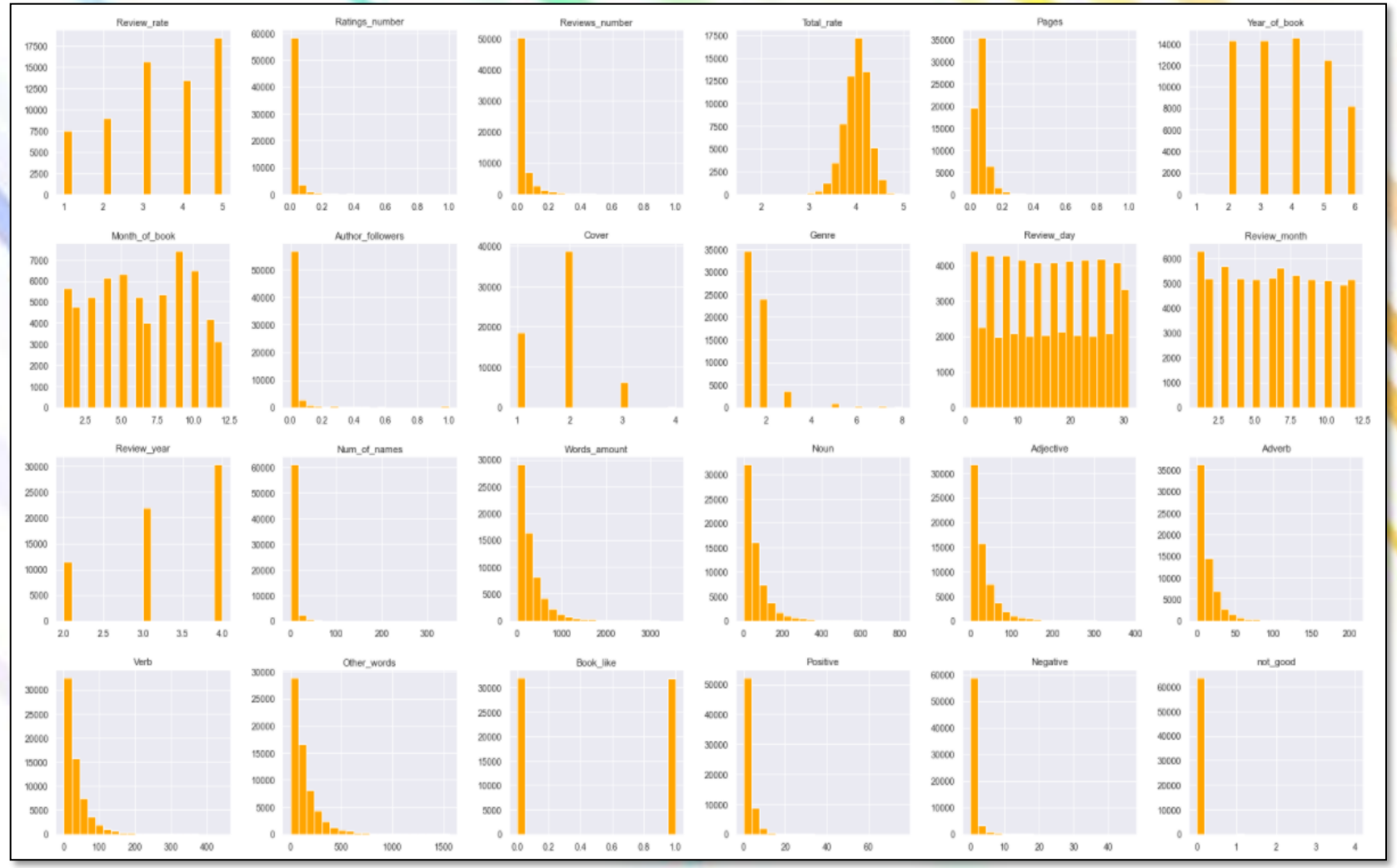
The columns that were created by vectorization

Total : 44 columns, 63,890 rows


EDA

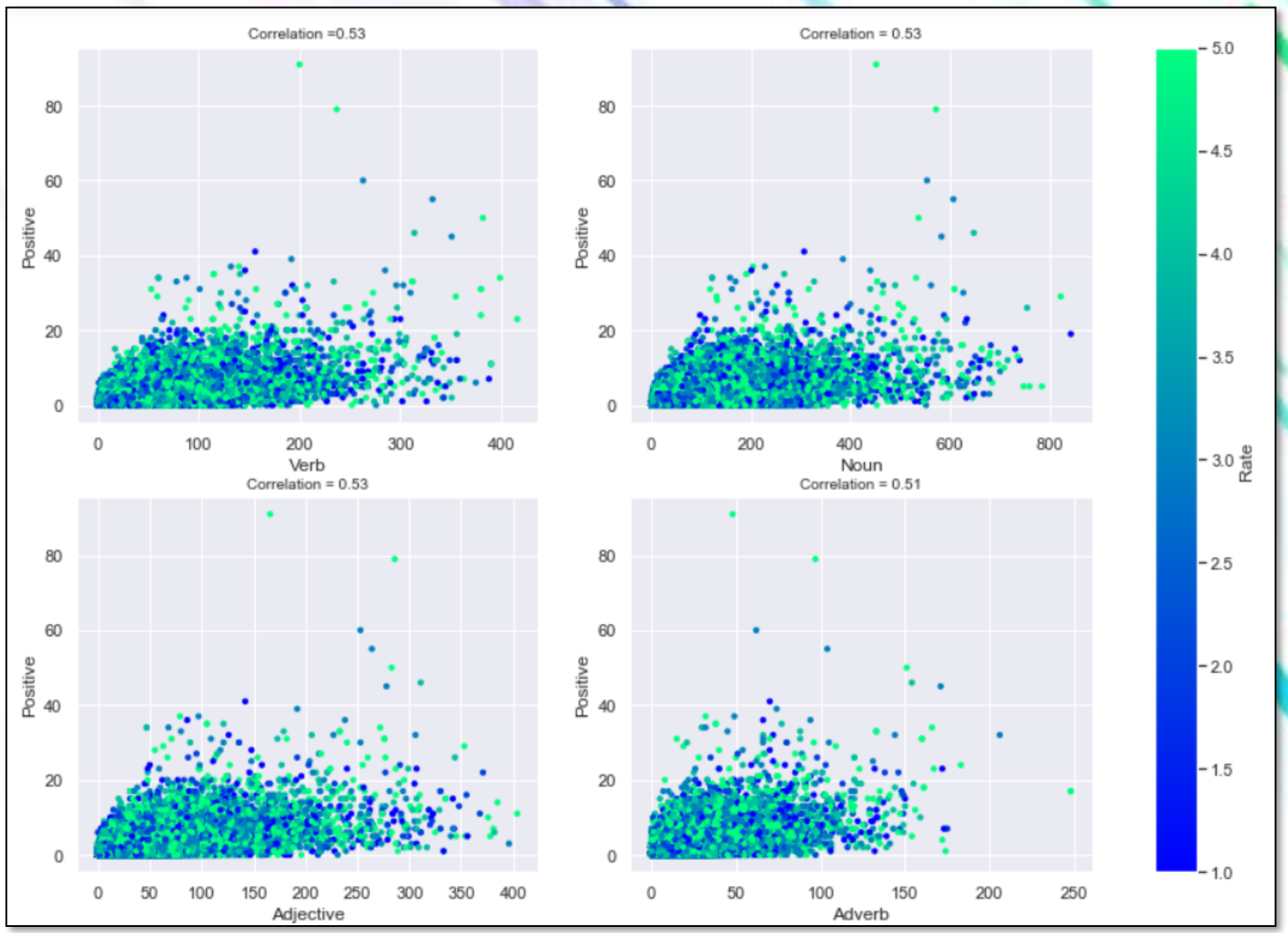
DISTRIBUTIONS

The data is not normally distributed, except for one figure – 'Total_Rate'. Therefore, I used Spearman method and not Pearson method for the correlation test.



NOTE : The display does not include all information

CORRELATIONS

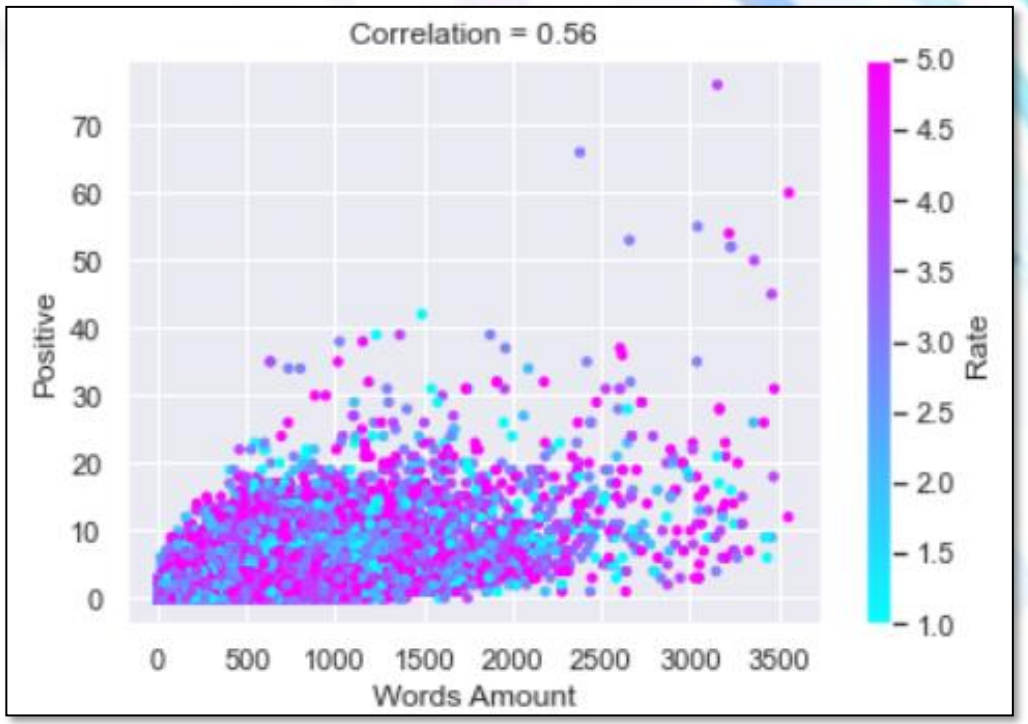


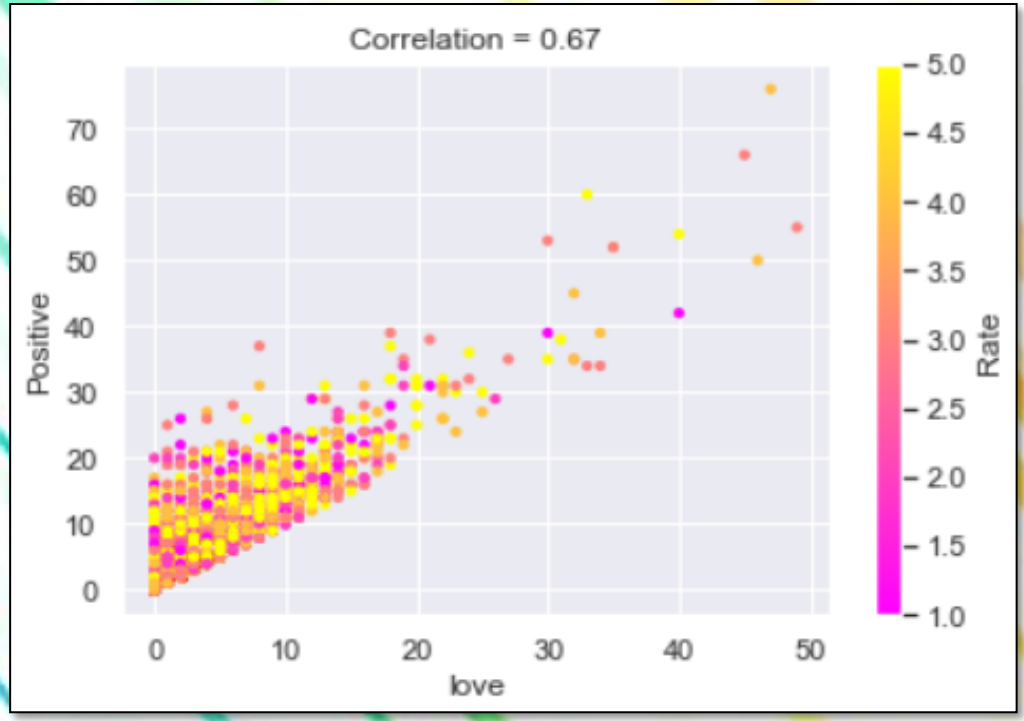
Each dot represents a row in the dataframe.

The dot color represents its rate.

Based on the results, one may conclude that the amount of positive words in the review is positively related to each of the lexical categories

This chart shows that, in general, the number of words in a review affects the number of positive words - although not always. However, there is still a linear relationship between them



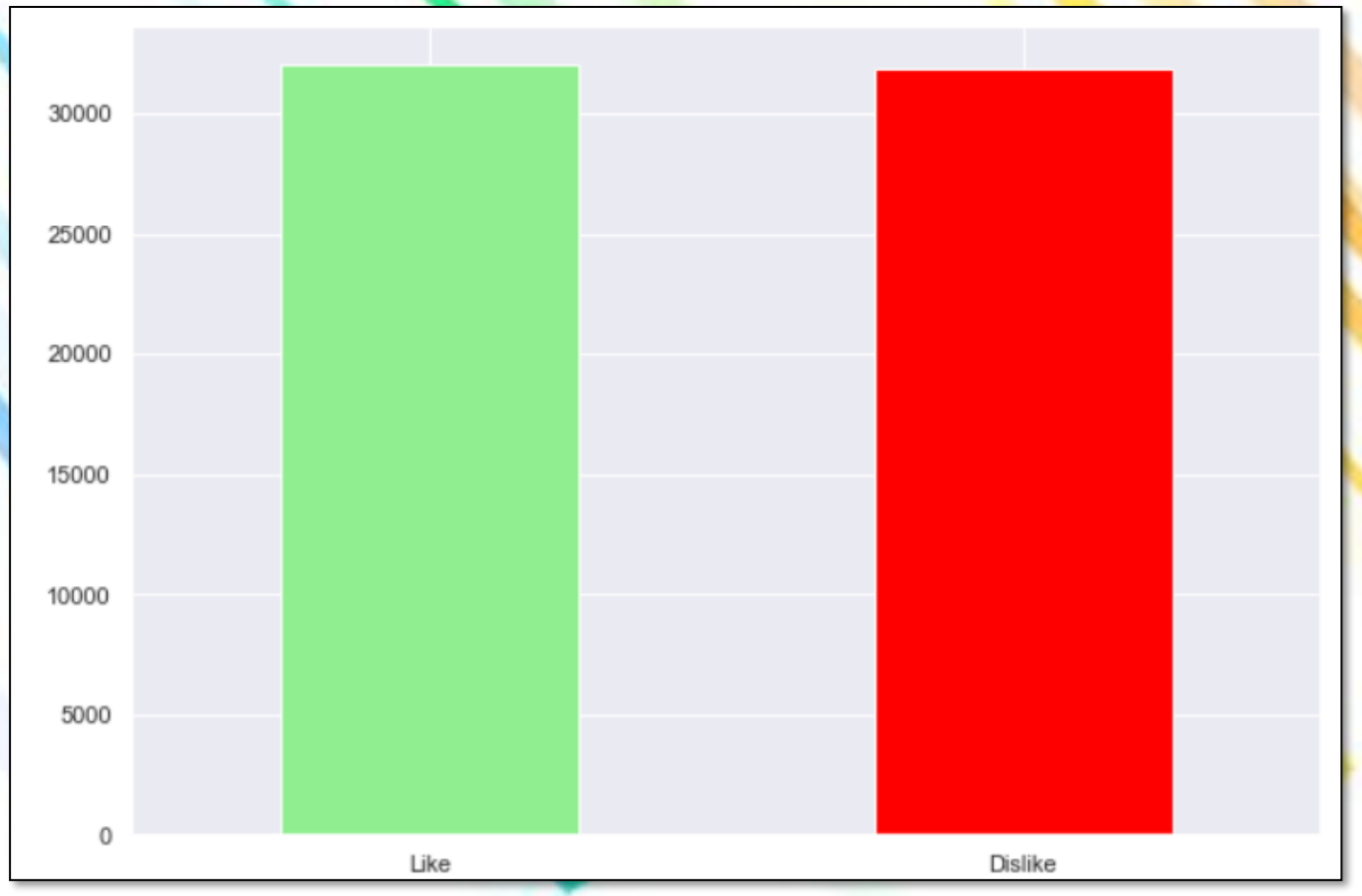


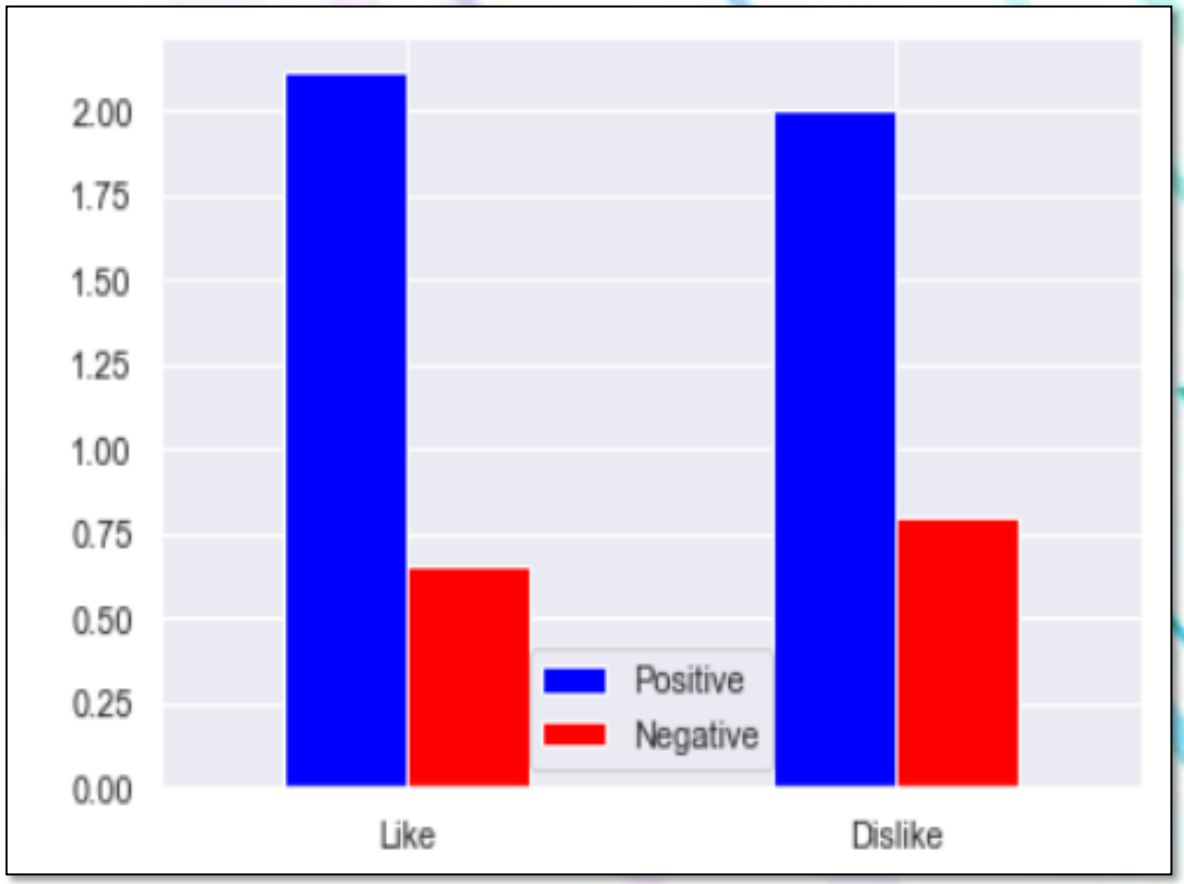
This graph shows that the more the word love appears in the review, the more positive words there are in it

DATA VISUALIZATION

This bar plot represents the amount of each value in the target label.

The amounts are almost equal according to this graph



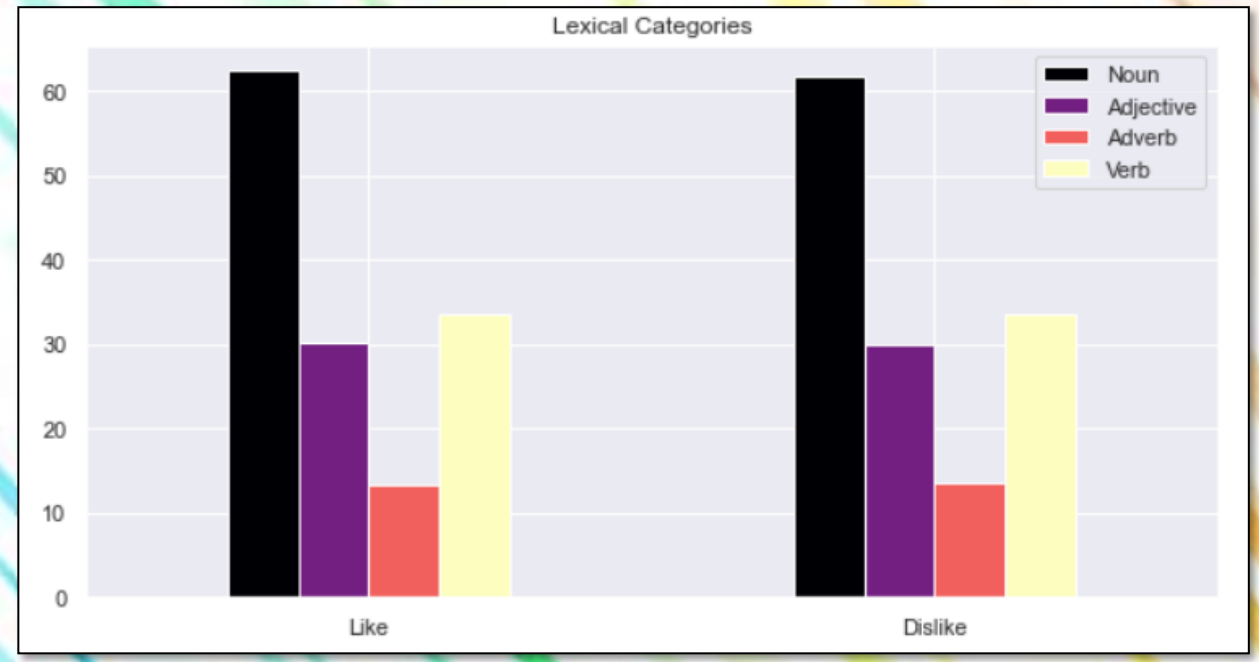


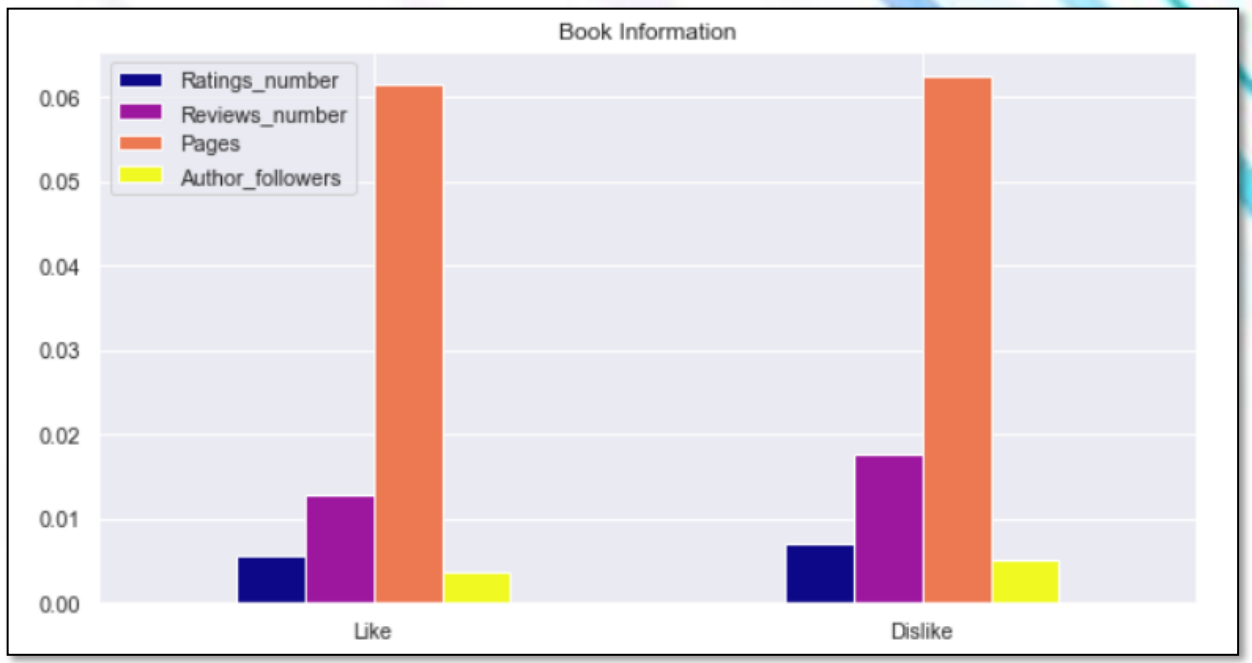
The graph displays the average number of positive words and the average number of negative words for each type of review.

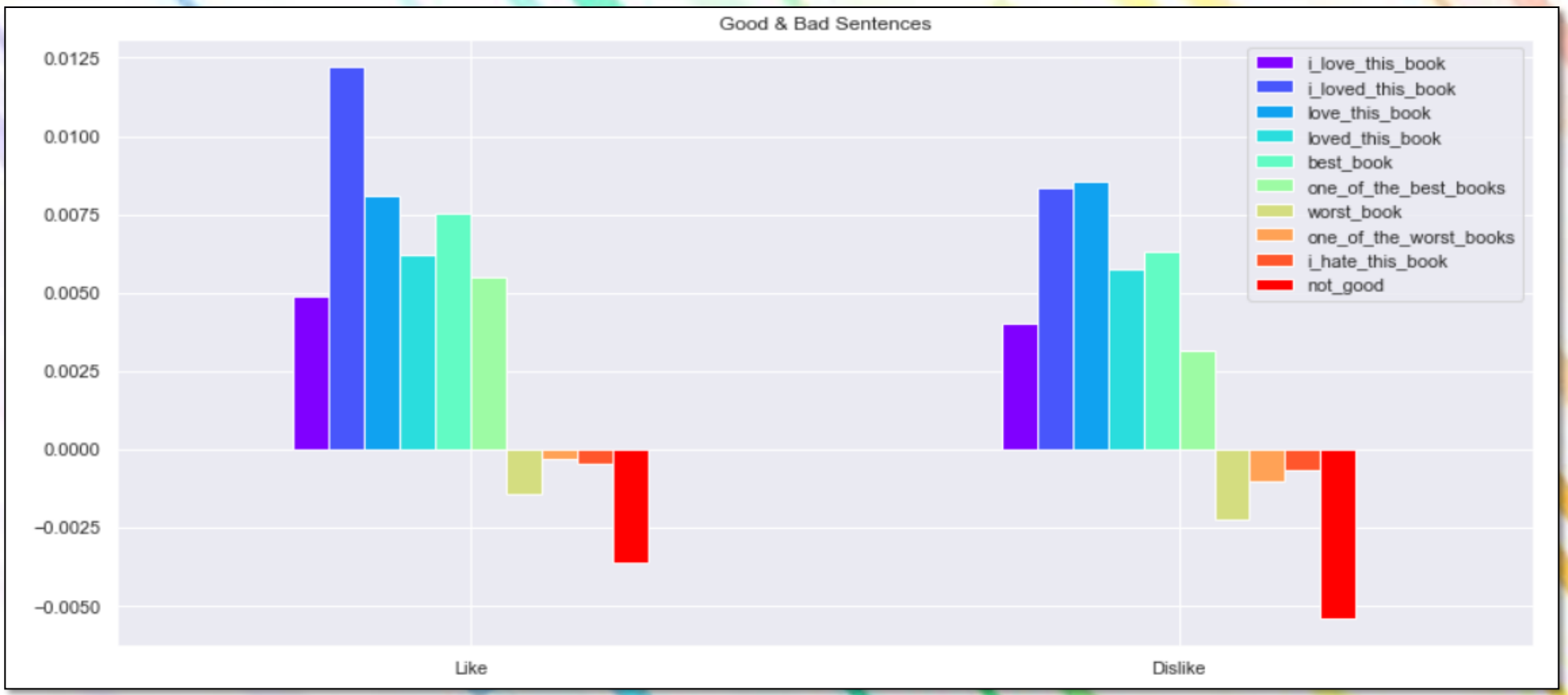
There is a slight difference between the average of the positive and negative words in the good reviews and the averages in the bad reviews

The reason for this is that there are words - both positive and negative - that may appear in contexts that are opposite of what the word indicates.

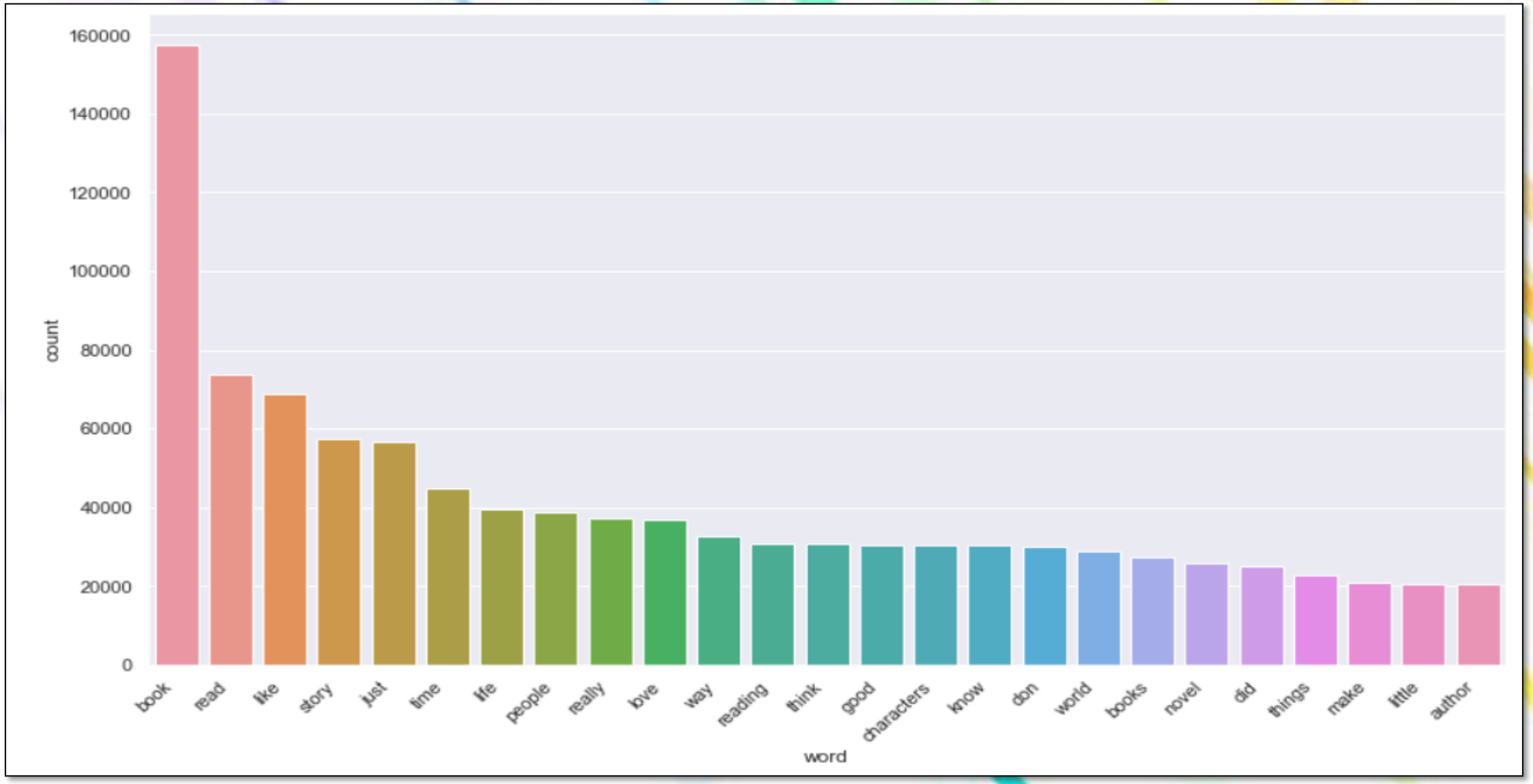
Both graphs show no significant difference between the good and bad reviews





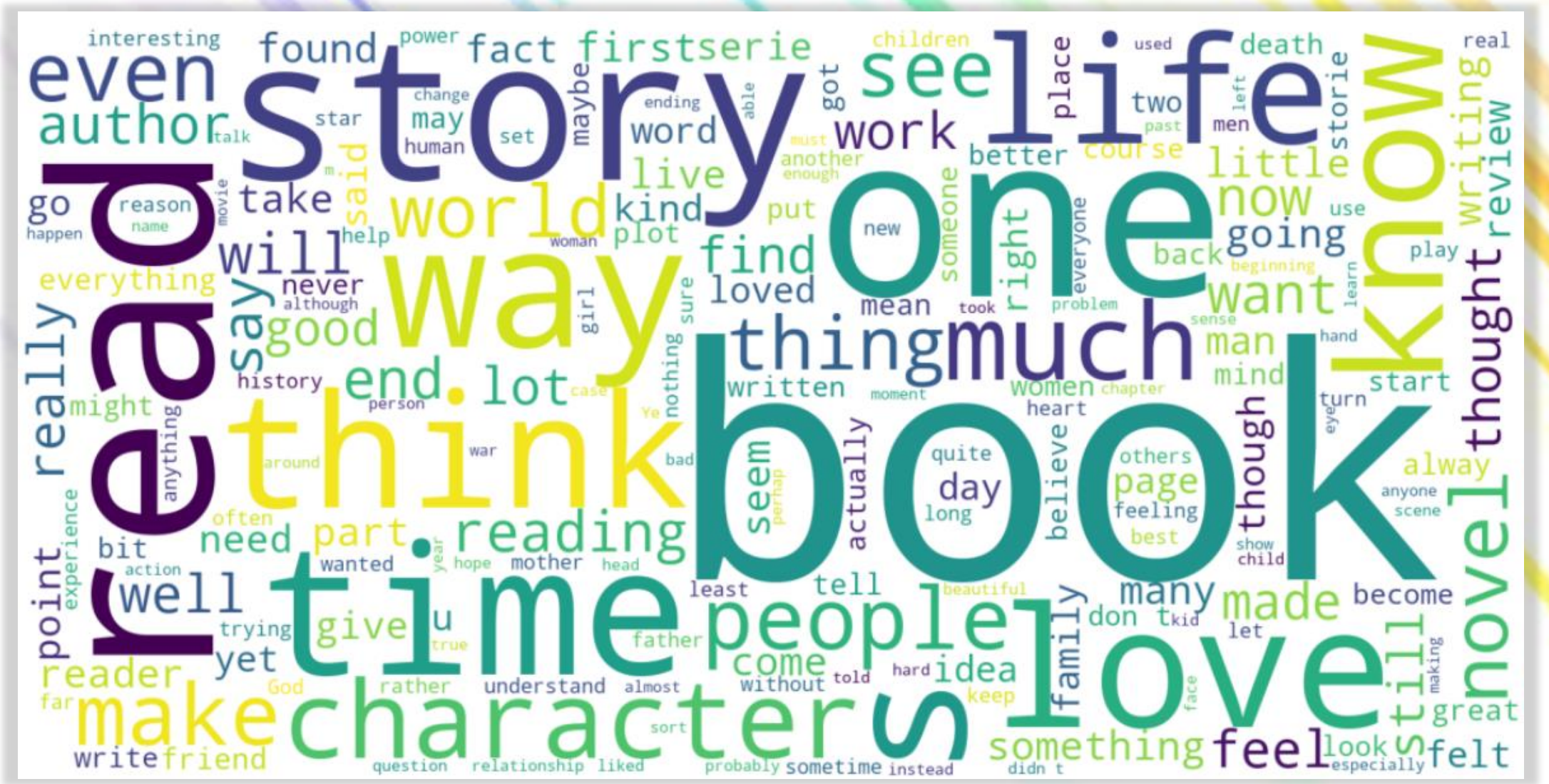


- I chose the word groups in the tokenization process that will appear as groups.
- A negative word group is represented in the negative part of the graph, while a positive word group is represented in the positive part of the graph.
- The displayed number is the average of the number of instances of each group, with a minus attached to the negative ones.
- Positive and negative reviews differ from each other, although the differences are not significant

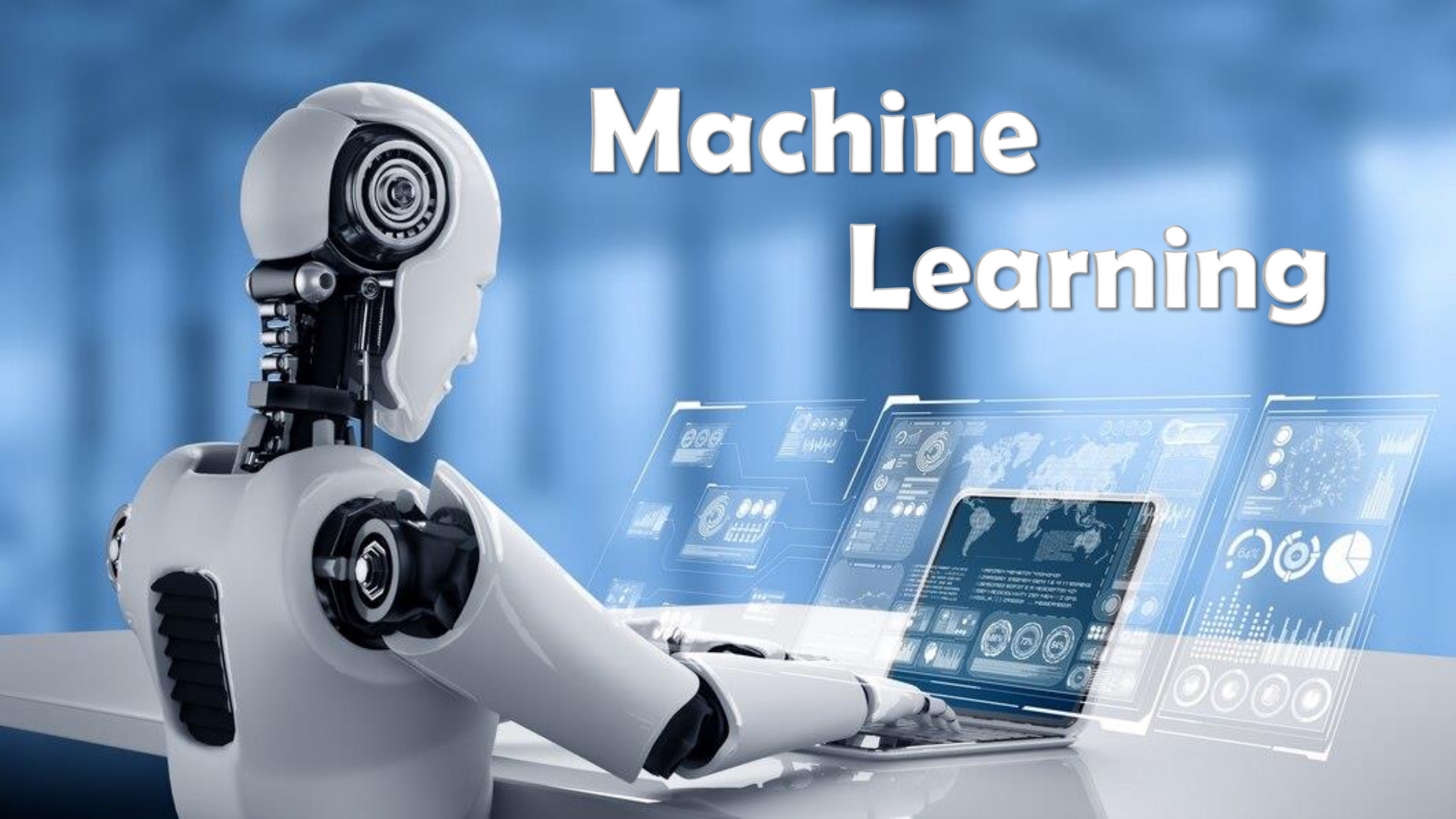


In descending order, these are the 25 most common words of all the reviews

WORDCLOUD



*The most common words


Machine
Learning

- The target label: ‘Book_Like’

- The machine learning model:

  AdaBoostClassifier – supervised learning

- Reprocessing of the data was performed

ML INFORMATION

STEPS:

1. I Dropped all the string columns:

   'Reviews', 'Reviews_link', 'Tokenized', 'Updated_Review'.

2. I Dropped the columns that directly related to the target column:

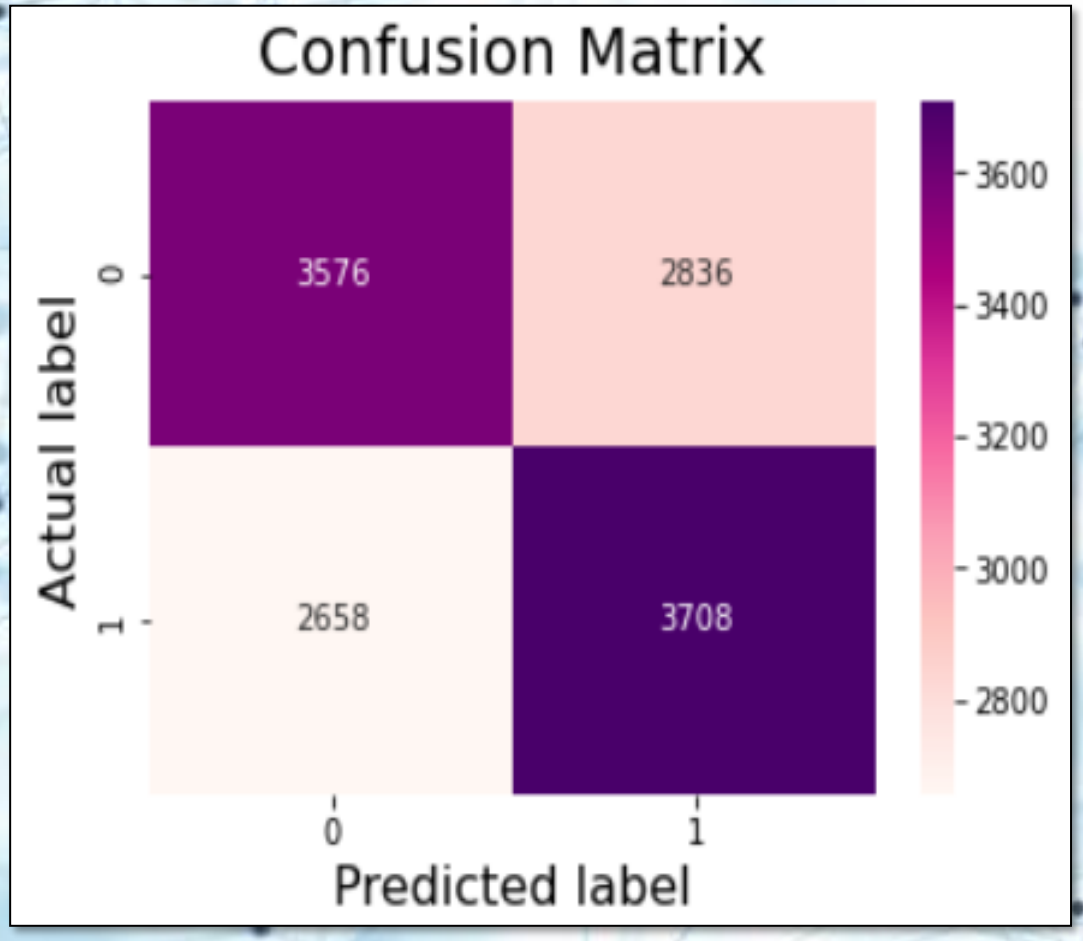
   'Review_rate', 'Total_rate', 'Ratings_number', 'Reviews_number'

3. Calculations and considerations were done on the columns containing information on positive and negative words

MACHINE LEARNING PROCESS:

1. The data frame was divided into 2: target column and features.
2. The divided data frame was divided to 80% train and 20% test.
3. Fit and predict

RESULTS:



Accuracy is: 0.5700422601346063
Precision is: 0.5666259168704156
Recall is: 0.5824693685202639
F1 is: 0.5744384198295894

ACCURACY SCORE : 57%


External
Resources
Stackoverflow
Scikit-learn
Neptune
Goodreads