IR_A3 REPORT

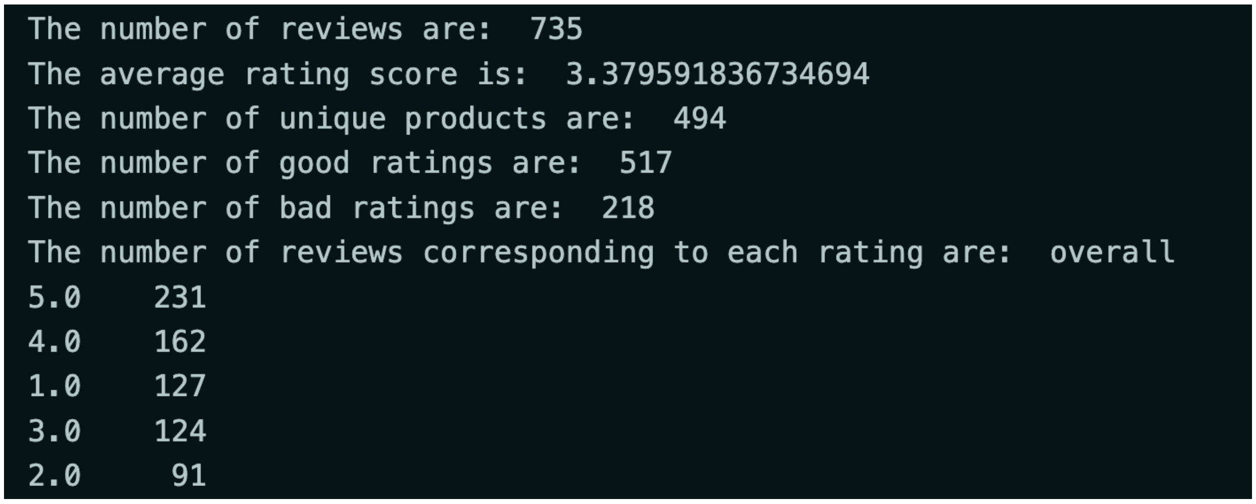
1) Loading the Data:

In order to load the data, I thought of 2 approaches. I could use chunking on my local system to load the data and then mirror my operations on each of these data chunks to extract the product data frame. This was quite tedious so I just used a server with higher computational power and loaded the entire data at once.

2) Choosing a Product:

I chose the product ‘headphones’. I used a regex-based match for the term ‘headphone’ or ‘headphones’ in the title field of the metadata. I extracted these asins and looked for reviews of this asin to extract headphone review data and headphone metadata from the entire data.

3) Below are the stats of the product:



```
The number of reviews are:  735
The average rating score is:  3.379591836734694
The number of unique products are:  494
The number of good ratings are:  517
The number of bad ratings are:  218
The number of reviews corresponding to each rating are:  overall
5.0    231
4.0    162
1.0    127
3.0    124
2.0     91
```

4) Pre-processing the text

I used regex to get rid of the HTML tags.

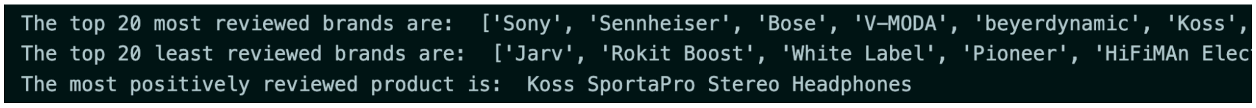
I used the unicodedata.normalize() function to get rid of accented characters.

I built my own acronyms dictionary for a few instances and used it to replace acronyms with their equivalent expansion.

I used regex to remove special characters.

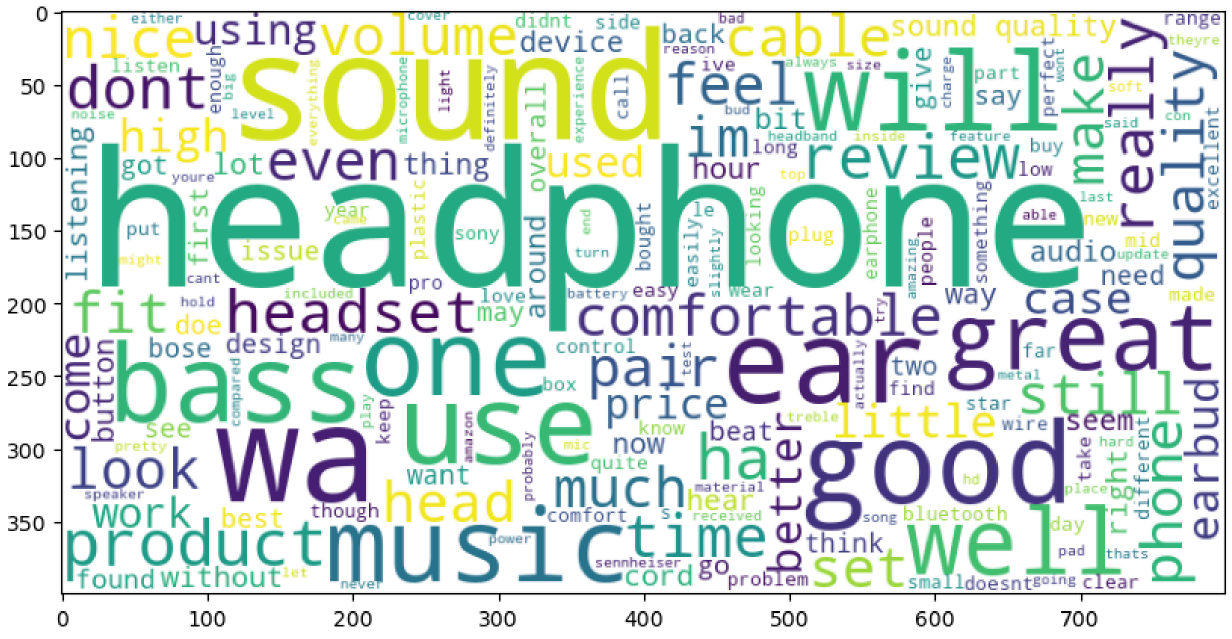
I used the WordNetLemmatizer() from the nltk library to lemmatize the text and then normalized it using the .lower() function.

5) More review statistics are present in the python notebook, here are some samples

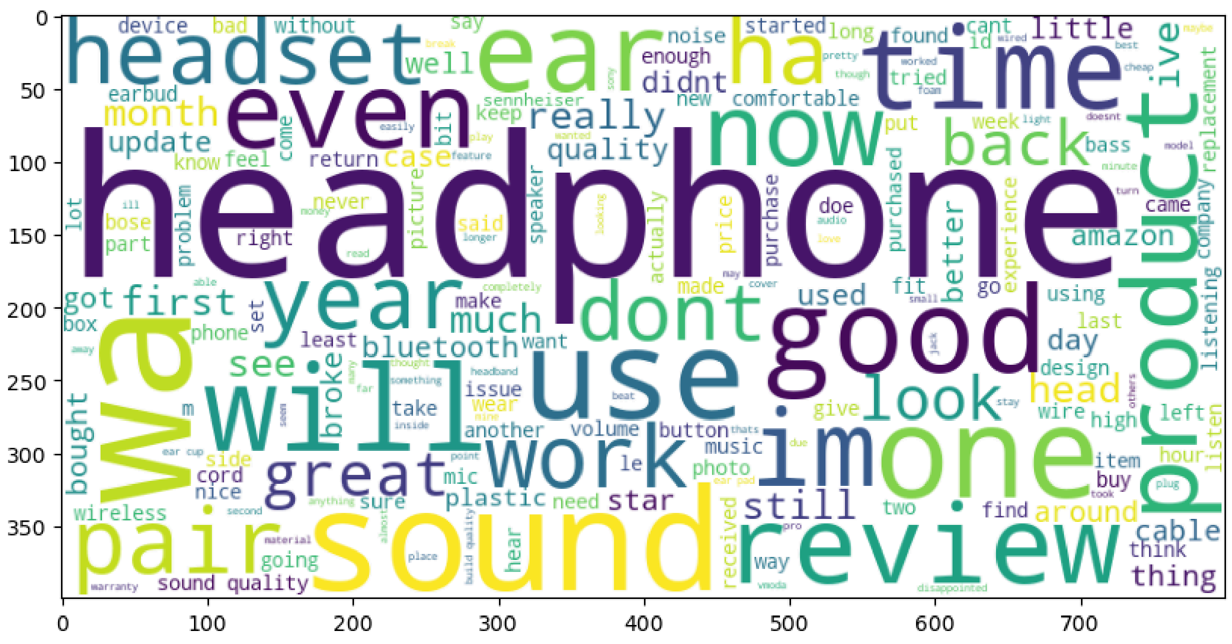


```
The top 20 most reviewed brands are:  ['Sony', 'Sennheiser', 'Bose', 'V-MODA', 'beyerdynamic', 'Koss',
The top 20 least reviewed brands are:  ['Jarv', 'Rokit Boost', 'White Label', 'Pioneer', 'HiFiMAn Elec
The most positively reviewed product is:  Koss SportaPro Stereo Headphones
```

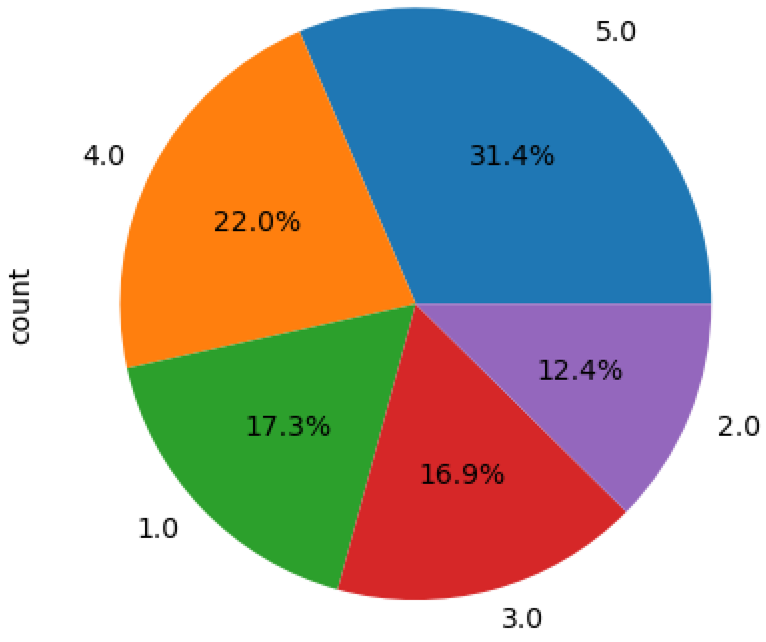
good wordcloud



bad wordcloud



6) The rating distribution piechart is as follows:



7) Word embeddings

For generating word embeddings, I used pre-trained word2vec embeddings with an embedding size of 100. word2vec here was pre-trained on the google-news-300M corpus.

The reason for using word2vec here was that it captures the contextual information from the text as well.

8) Rating Classes

The ratings were divided into three different classes on the basis of the given criteria.
>3 for good (encoded with → 1)
=3 for average (encoded with → 0)
<3 for bad (encoded with → -1)

9) Running ML models

I ran 5 models and printed the classification report for the same. The results are reported in the main.ipynb file. I used the sklearn library for this

10) Collaborative filtering

The pipeline is as follows:
(i) Start by creating the user-item matrix where:

(users → rows) = (number of unique reviewer ids)
and
(items → columns) = (number of unique asins)

(ii) Fill NaN values with 0

(ii) Make a 5-fold split

(iii) For the train split, for each train user… we find out the N nearest neighbors.

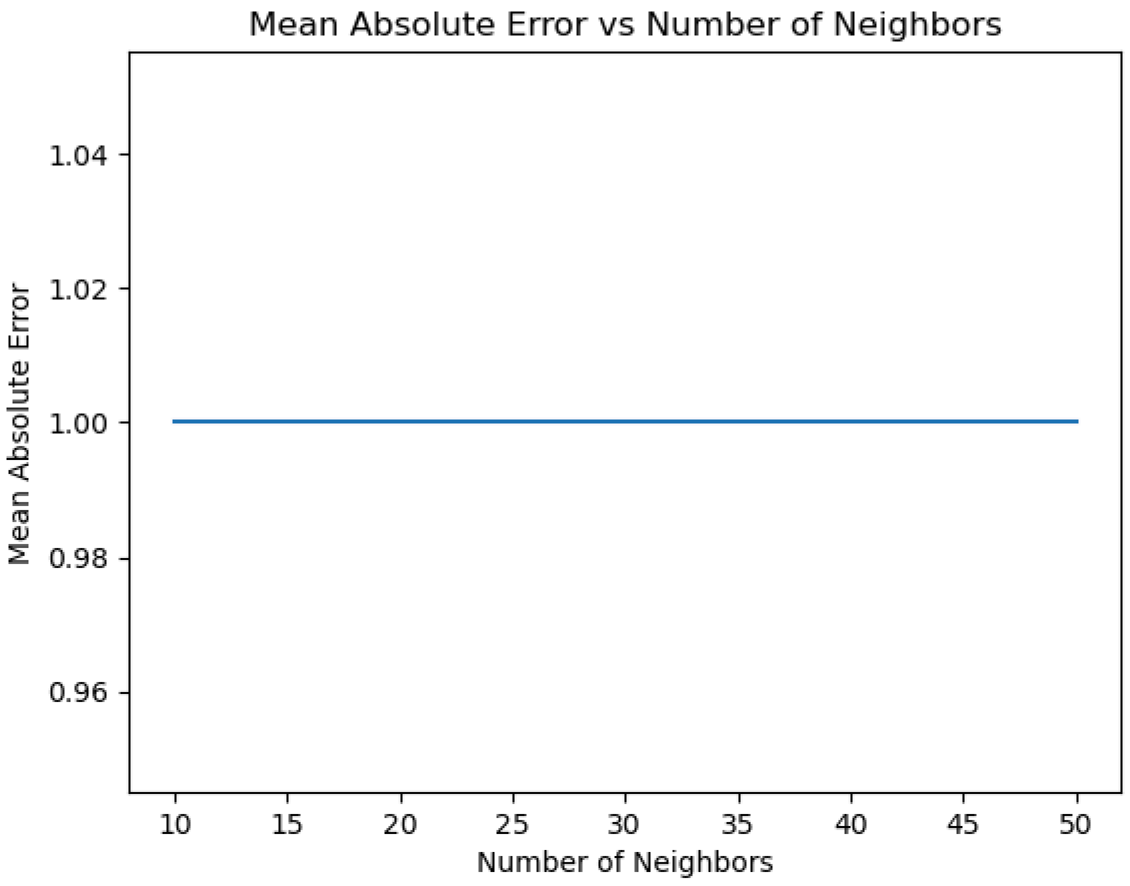
(iv) For all zero values of a user, I fill the value with zero value with mean of the product rating in the neighbors.

(v) For the test set, we calculate the mean absolute error for each (train_user, test_user) pair.

(vi) Report the mean absolute error and plot.

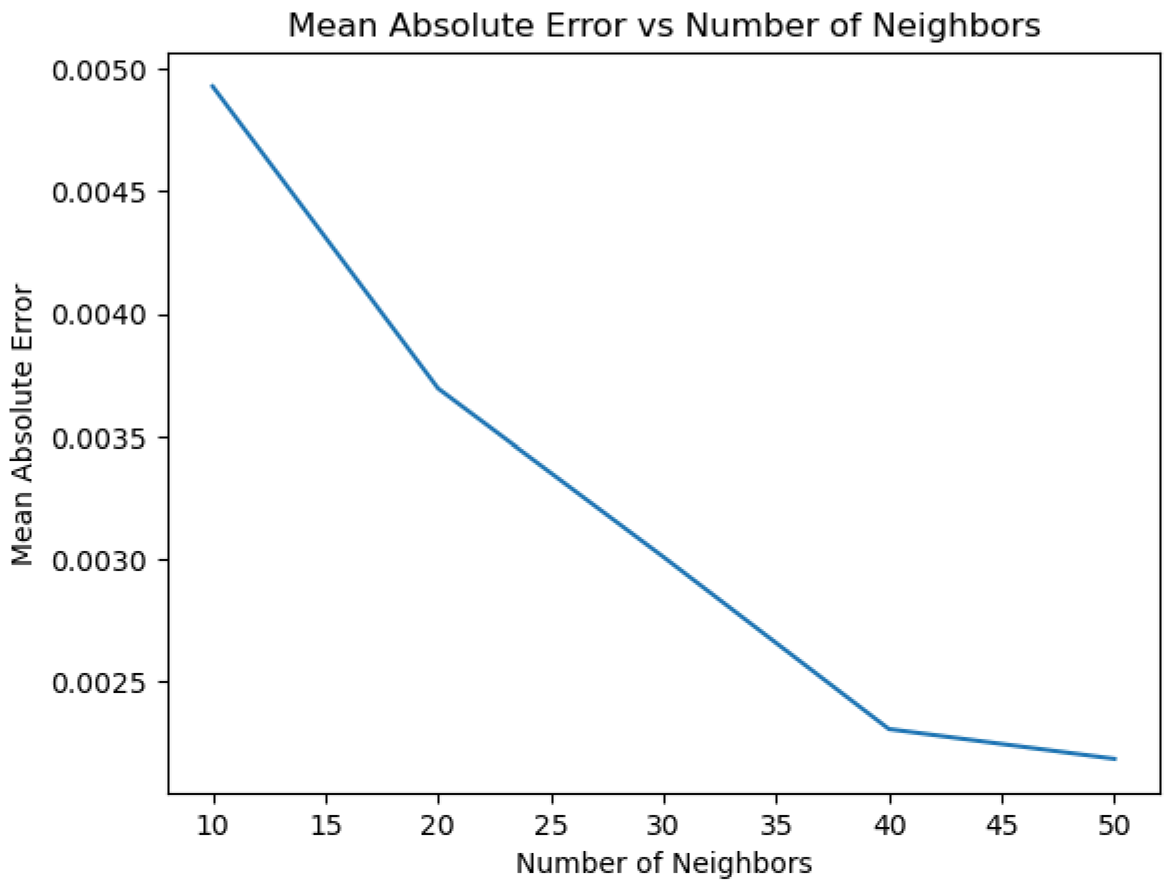
11) Inference for the Mean Absolute Error

(i) User-User Collaborative Filtering:



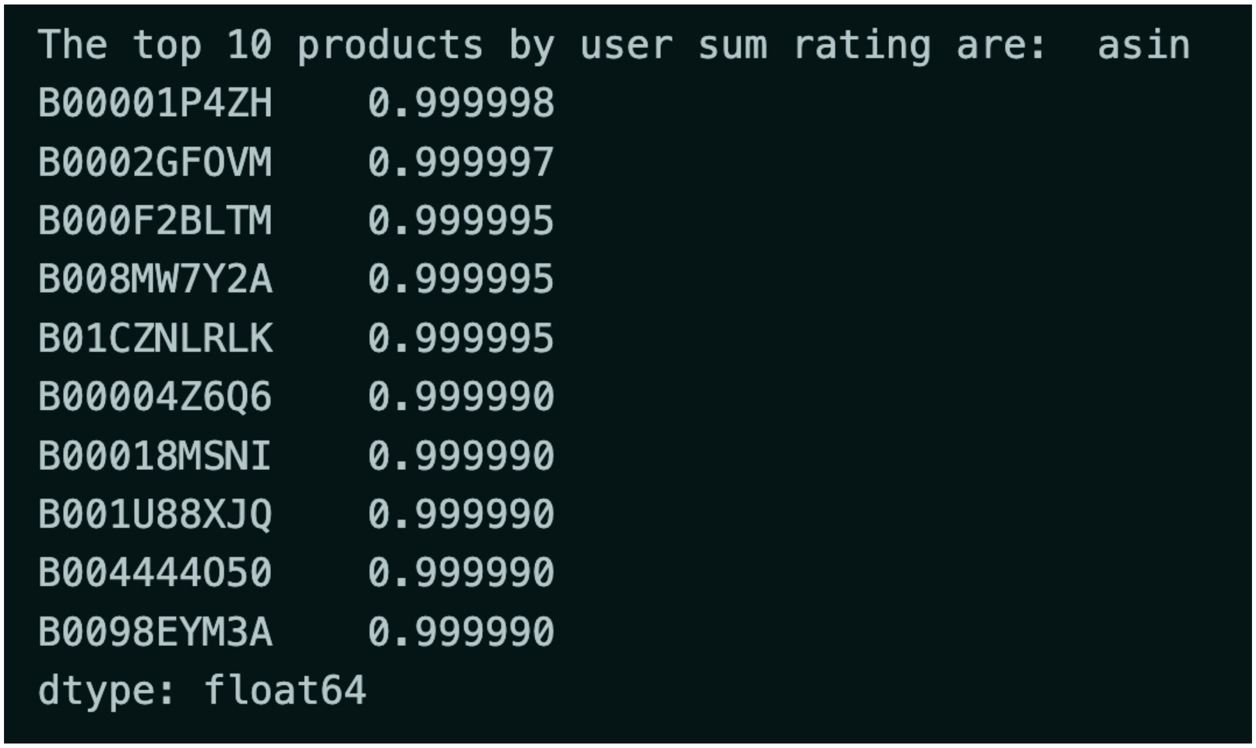
I notice that even after increasing the number of neighbours, the mean absolute error doesn't decrease. This is because the user-item matrix is extremely sparse and with a large number of zeros, the similarity between two users (row-wise) may not have a big difference with the other users.
Moreover, while calculating the mean to fill the zero values, in most cases I arrive at zero mean. Hence, even after increasing the number of neighbours, I observe that there is no drop in the mean absolute error.

(ii) Item-Item:



For item-item similarity (column-wise similarity), while comparing with the nearest neighbours, we are able to find vectors with non-zero rating values. Hence, upon broadening the number of nearest neighbours, we are able to fill more zero values with a non-zero value after calculating the mean of the nearest neighbours for a particular rating field. This results in a drop in the mean absolute error upon increasing the number of errors.

12) Top10 user sum ratings



```
The top 10 products by user sum rating are:  asin
B00001P4ZH    0.999998
B0002GFOVM    0.999997
B000F2BLTM    0.999995
B008MW7Y2A    0.999995
B01CZNLRLK    0.999995
B00004Z6Q6    0.999990
B00018MSNI    0.999990
B001U88XJQ    0.999990
B004444050    0.999990
B0098EYM3A    0.999990
dtype: float64
```