Building a Recommendation System for ModCloth

Newton Tran

INTRODUCTION

Recommendation systems are a type of machine learning technique that utilizes data to help predict suggestions to a user. Some examples include:

- Netflix recommending movies to watch
- Amazon suggesting which products to purchase
- Spotify recommending other songs
- YouTube recommending other videos
- TikTok's For You page
- Instagram's Explore page

With so many options to choose from, recommendation systems can help streamline the consumer's experience. More importantly, from a business standpoint, they can help drive sales since they:

- Increase personalization for the consumer (which also means higher customer retention)
- Lead to more cross-selling
- Lower marketing costs

Recommendation systems are implemented through two main methods: collaborative filtering and content-based filtering. The former utilizes user similarity to recommend items, whereas the latter utilizes the item's features and it only focuses on one specific user's preferences rather than a group or type. Collaborative filtering is further split into two types: user-based and item-based. Here, the former focuses on similarity between users, where if any given two users have similar preferences or behaviors, they are likely to prefer similar items. On the other hand, the latter focuses on similarity between items that users have already interacted with.

Quite notably, Amazon and Netflix have hugely capitalized on recommendation systems. In 2009, Netflix held the Netflix Prize where $1 million was awarded to any team that could come up with the best collaborative filtering algorithm that could predict user ratings for films, based on previous ratings without any other information about the users or films; the winning team, Belkor's Pragmatic Chaos, beat Netflix's algorithm by 10%. For Amazon, they have especially implemented item-based collaborative filtering in their recommendation system that accounts for around 35% of their sales.

PROBLEM STATEMENT

ModCloth is a brand that specializes in vintage-inspired women's apparel, particularly in dresses. The goal of this project is to build a recommendation system that recommends three products to a given user; the recommendation system utilizes item-based collaborative filtering.

THE DATA

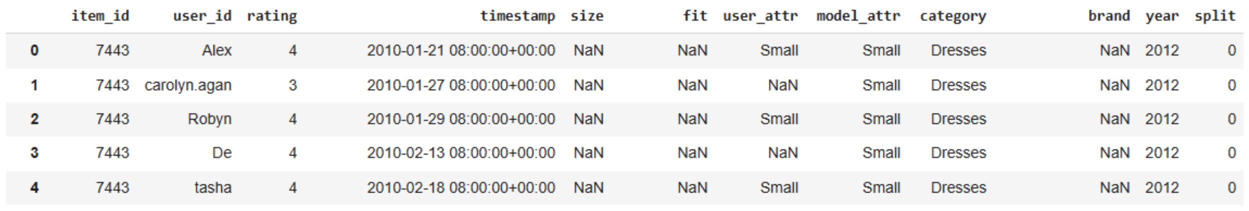
The dataset, 'df_modcloth.csv,' was compiled by the University of California, San Diego's machine learning research lab. It contains 99,893 product reviews, where there are 1020 unique items and 44,783 unique users. The reviews span over nine years from 2010 to 2019. There are a total of 12 features, five of which are numerical and seven are of object-type. The five numerical features include:

```
['item_id', 'rating', 'size', 'year', 'split']
```

The seven object-type features include:

```
['user_id', 'timestamp', 'fit', 'user_attr', 'model_attr', 'category', 'brand']
```

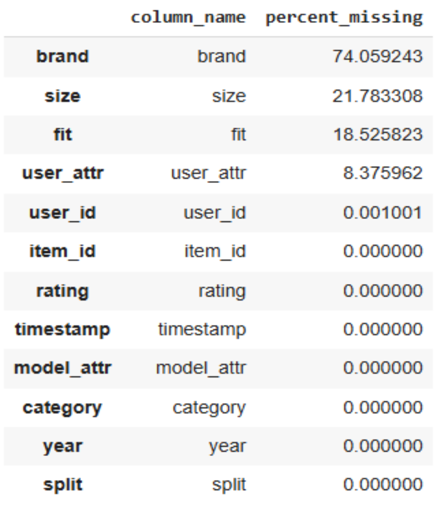
A preview of the data is shown below:



	item_id	user_id	rating	timestamp	size	fit	user_attr	model_attr	category	brand	year	split
0	7443	Alex	4	2010-01-21 08:00:00+00:00	NaN	NaN	Small	Small	Dresses	NaN	2012	0
1	7443	carolyn.agan	3	2010-01-27 08:00:00+00:00	NaN	NaN	NaN	Small	Dresses	NaN	2012	0
2	7443	Robyn	4	2010-01-29 08:00:00+00:00	NaN	NaN	Small	Small	Dresses	NaN	2012	0
3	7443	De	4	2010-02-13 08:00:00+00:00	NaN	NaN	NaN	Small	Dresses	NaN	2012	0
4	7443	tasha	4	2010-02-18 08:00:00+00:00	NaN	NaN	Small	Small	Dresses	NaN	2012	0

EXPLORATORY DATA ANALYSIS

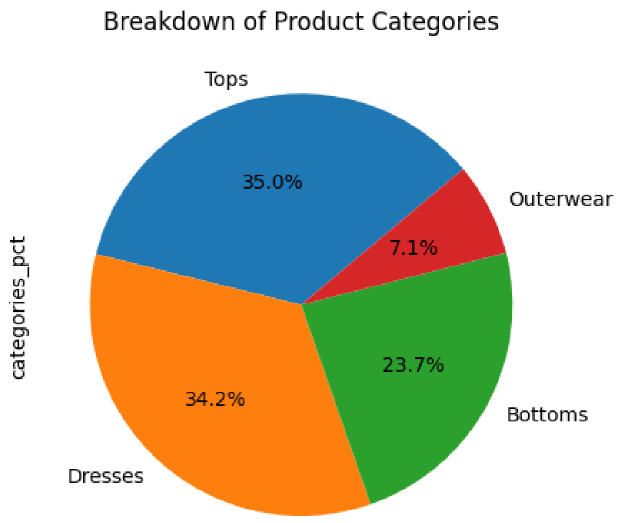
The dataset was first checked for missing values:



	column_name	percent_missing
brand	brand	74.059243
size	size	21.783308
fit	fit	18.525823
user_attr	user_attr	8.375962
user_id	user_id	0.001001
item_id	item_id	0.000000
rating	rating	0.000000
timestamp	timestamp	0.000000
model_attr	model_attr	0.000000
category	category	0.000000
year	year	0.000000
split	split	0.000000

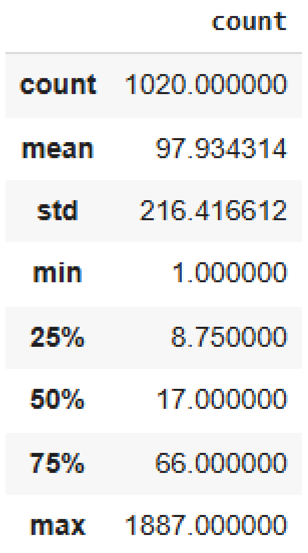
The five features brand, size, fit, user_attr, and user_id contained nulls.

Next, since there were multiple categories for the items, the distribution of the item categories was checked:



Tops, dresses, and bottoms were the three most prominent item categories.

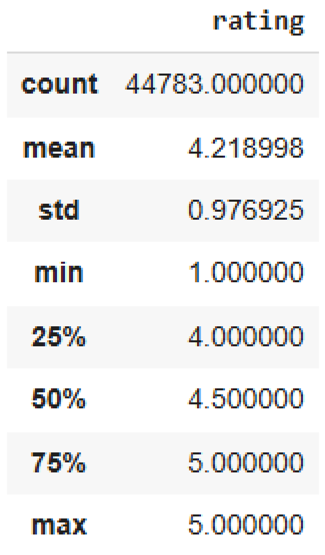
The most important part of the exploratory data analysis is describing the distribution of the item ratings. A set of summary statistics is shown below for the number of item ratings for each of the 1020 unique products:



	count
count	1020.000000
mean	97.934314
std	216.416612
min	1.000000
25%	8.750000
50%	17.000000
75%	66.000000
max	1887.000000

The average number of item ratings was about 98, which starkly contrasts to the median number of 17 ratings. Also, this highly varied with a standard deviation of about 216 ratings and a range of 1886, varying between 1 to 1887 reviews.

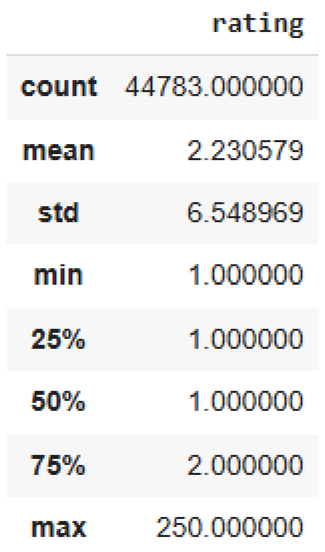
Besides the ratings of the items, another thing to consider is the ratings submitted by the users themselves. A set of summary statistics is shown below for the average ratings submitted for each of the 44,783 unique users:



	rating
count	44783.000000
mean	4.218998
std	0.976925
min	1.000000
25%	4.000000
50%	4.500000
75%	5.000000
max	5.000000

Here, the average user rating is around 4.22, which slightly deviates from the median rating of 4.5. The standard deviation is 0.98, which is more-or-less a whole rating.

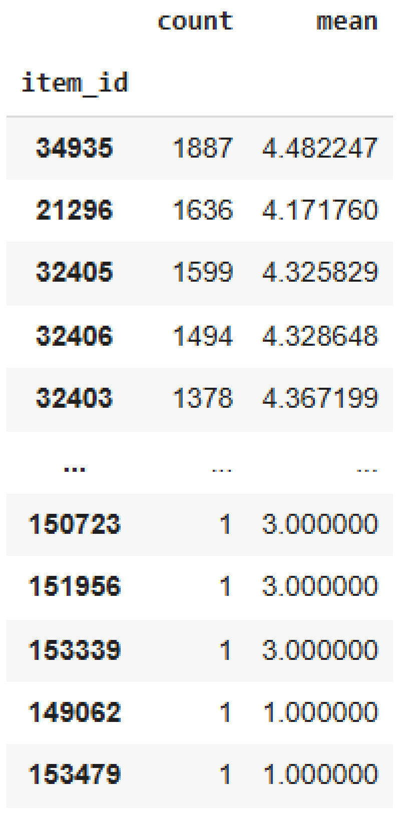
In addition to the average user ratings, another thing to consider is the number of ratings that user has submitted. A set of summary statistics is shown below for the number of ratings submitted:



	rating
count	44783.000000
mean	2.230579
std	6.548969
min	1.000000
25%	1.000000
50%	1.000000
75%	2.000000
max	250.000000

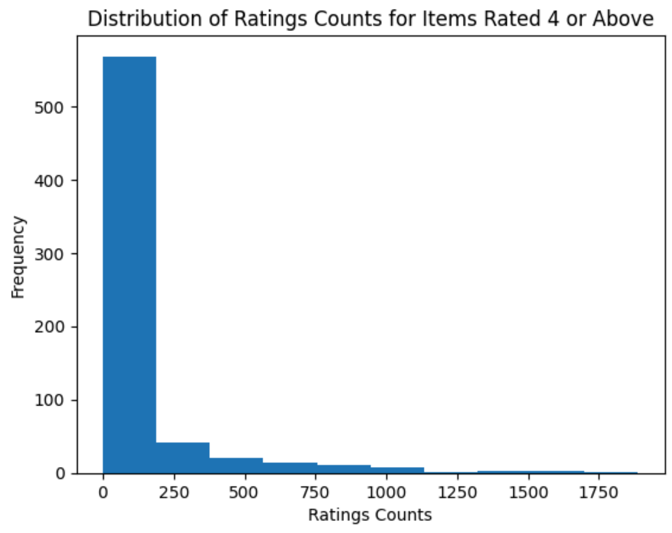
It is evident that some users are more proactive in submitting reviews than others, especially considering that the average number of reviews is about 2. Similar to the distribution of the number of ratings for items, the number of ratings that users submitted also highly varies from 1 to 250.

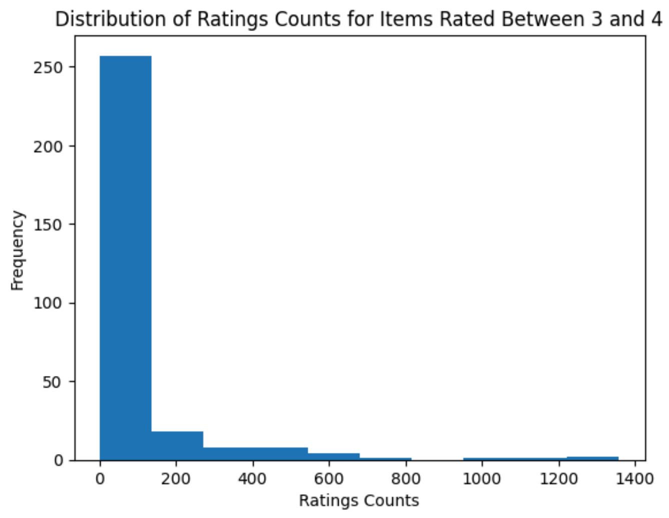
Most importantly, this exemplifies the possibility that some items that are highly or negatively-rated can have little to several reviews. To further understand this possibility, the dataset can be aggregated based on the item's number of ratings and its number of ratings:

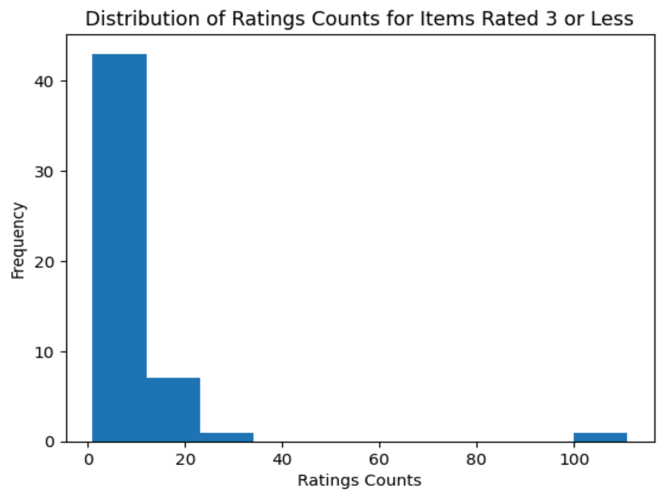


	count	mean
item_id		
34935	1887	4.482247
21296	1636	4.171760
32405	1599	4.325829
32406	1494	4.328648
32403	1378	4.367199
...	...	...
150723	1	3.000000
151956	1	3.000000
153339	1	3.000000
149062	1	1.000000
153479	1	1.000000

This can then be further split into three cases where we look at items rated 4 or above, between 3 and 4, and those rated 3 or below:







In all of these three cases, we observe right-skewness. More so, given each of their right-skewness, we see that the ratings counts for items rated 3 or above differ significantly from those rated below 3.

DATA WRANGLING AND PREPROCESSING

Four features were extracted from the entire dataset which include:

- item_id
- user_id
- rating
- category

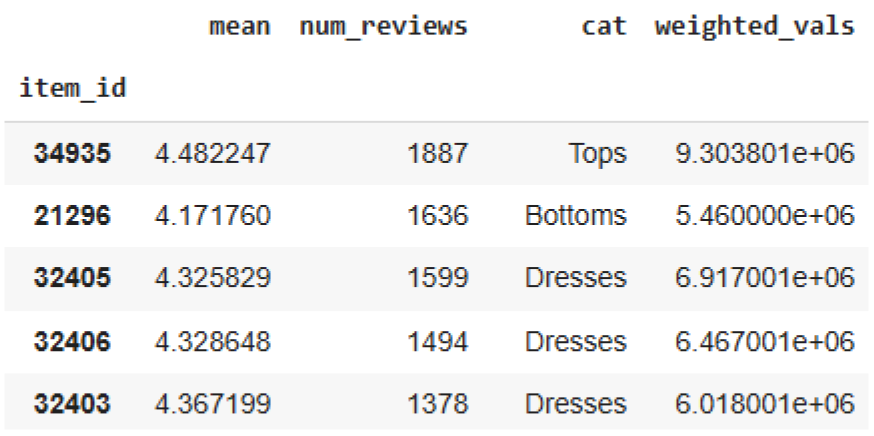
Out of these features, user_id had one missing value, which was imputed by filling it with ‘Unknown.’

Next, it was important to establish the notion of what is considered a “good” item, as this is crucial to the implementation of the recommendation system. In this case, a “good” item fulfilled these two criteria:

- Its average rating was 4 or above. While ratings are arbitrary, a higher rating tends to an overall better item
- It contained at least 24 reviews. As aforementioned, it is possible that a positively-rated item has either a single, few, tens, or even hundreds of reviews; the same applies for negatively-rated items.

After the data was filtered, the dataset was reduced to 93,915 rows, containing 435 unique items and 43,470 unique users. This filtered dataset was subsequently saved as ‘df_modcloth_filtered.csv.’

Additionally, a separate dataset, ‘pop_items.csv,’ was created to include the most popular items through basic aggregation (i.e. generic recommendations), which was grouped by item_id and then sorted by each product’s average rating and number of reviews, each in descending order:



	mean	num_reviews	cat	weighted_vals
item_id				
34935	4.482247	1887	Tops	9.303801e+06
21296	4.171760	1636	Bottoms	5.460000e+06
32405	4.325829	1599	Dresses	6.917001e+06
32406	4.328648	1494	Dresses	6.467001e+06
32403	4.367199	1378	Dresses	6.018001e+06

Weighted_vals is calculated by multiplying mean and num_reviews together, then multiplying by a value between 700 to 1600 (higher rated items are given much higher weights) or dividing by 1000 if the item's mean rating is below 4, and then adding a random float generated between 0 and 1. Adding a random float ensures that the mapping of each item_id to its corresponding weighted_vals is bijective. In other words, it is possible that two distinct items might have the same exact weighted_vals values, but when applying the inverse mapping (i.e. weighted_vals to item_id), the two distinct items cannot be re-obtained. This bijective mapping comes into special play during the final implementation of the recommendation system.

MODELING

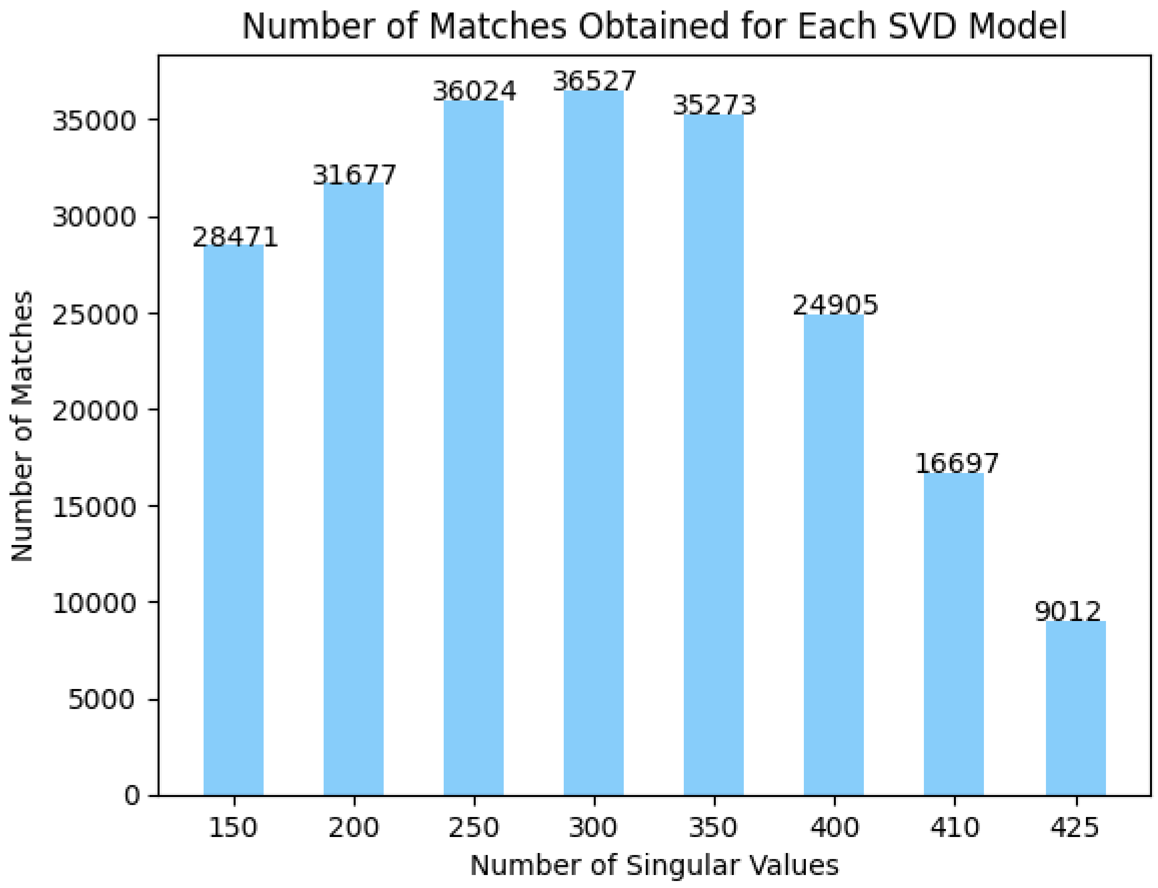
The recommendation system was split into two cases:

- A user's average rating is 4 or above
- Otherwise, a user's average rating is below 4

Since the recommendation system should only recommend "good" items, establishing the threshold of a rating of 4 is crucial. In the first case, item-based collaborative filtering was applied, whereas in the second case, generic recommendations from the most popular items were given.

The dataset was train-test split, using 50-50 stratification. Then, a utility matrix that contains the users as the rows and the items as the columns was constructed out of the training data. In this particular case, item-based collaborative filtering was applied because there were outstandingly much more users than items; it was computationally not feasible to store a similarity matrix of 43,470 users compared to 435 items. In this specific case, since the ratings were rank-based, using Spearman's correlation was preferred.

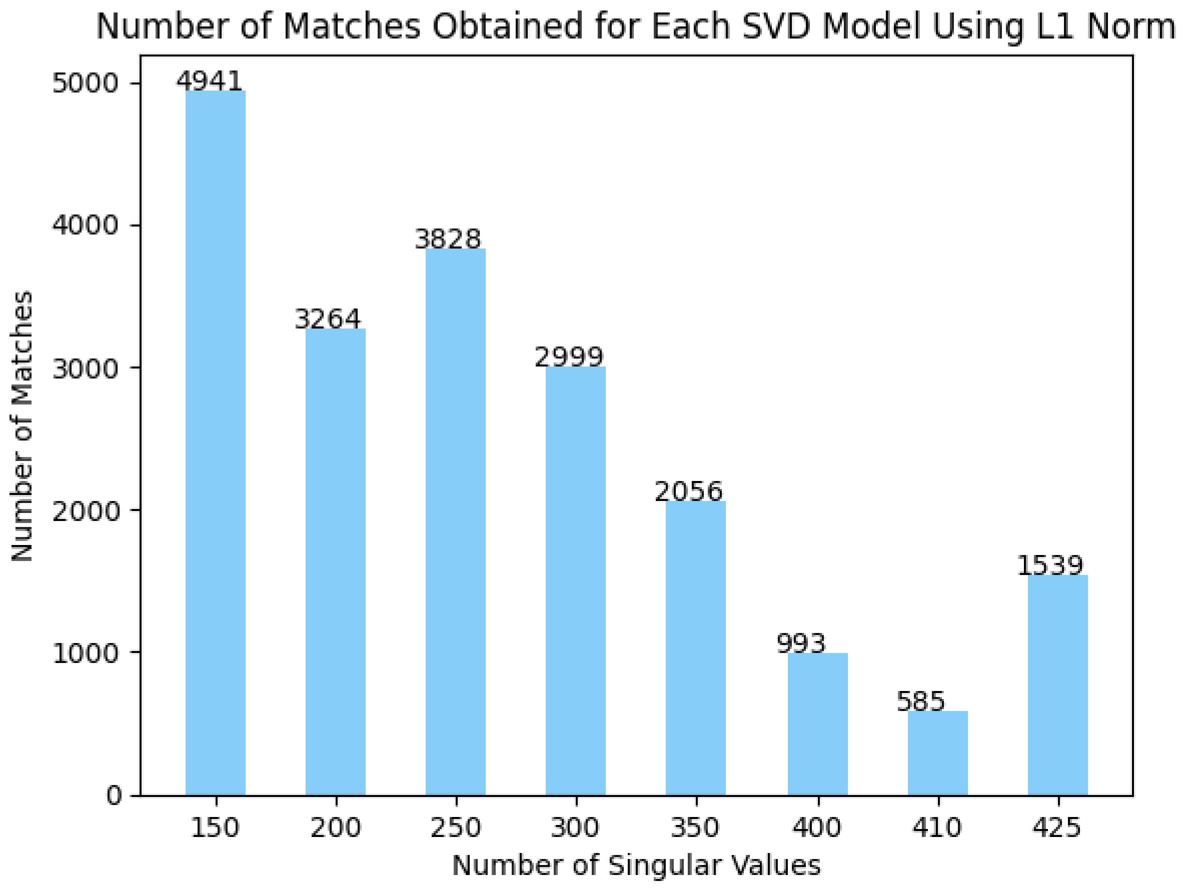
Subsequently, singular value decomposition (SVD) was applied to the utility matrix. The number of singular values tested include: 150, 200, 250, 300, 350, 400, 410, and 425. From that reduced utility matrix, a k-nearest neighbors model of k = 10 was then constructed. In this case, k = 10 was chosen since it was essential that the model could capture a sufficient, but not overly-excessive number of similar items; this is also to take into account the possibility that the k-nearest neighbors model might predict items that the user has already reviewed. Metric-wise, the best model was chosen based on the number of matches (a simple metric) each k-nearest neighbors model obtained per different number of singular values relative to the test data.



In this case, using 300 singular values yielded the highest number of matches. The root mean squared error (RMSE) between the utility matrix and its reduced version for each SVD value is shown below:

```
{150: 0.128778485257336,
 200: 0.1019257880438485,
 250: 0.08059295336375218,
 300: 0.061961617379721494,
 350: 0.04433289670423865,
 400: 0.02500844225436355,
 410: 0.020314003417670517,
 425: 0.011552216980782718}
```

In the k-nearest neighbors model, L2 norm (Euclidean distance) is used by default. However, interestingly enough, using L1 norm (Manhattan distance) yielded significantly less matches.



CONCLUSION

Overall, the data showed signs of right-skewness, which influenced the results of our recommendation system since there was much more emphasis placed on items rated 4 or above, while heavily penalizing those rated below 4. This can be problematic because it is possible that items rated between 3.7 to 3.9, for example, can actually be considered “good” items. Furthermore, trying to establish a generalized notion of what is considered a “good” item is not straightforward because ratings can be highly subjective and inconsistent, especially relative to the number of reviews any given product has.

Additionally, one way we can further understand the behavior of our recommendation system – and the bias in our data – is to perhaps apply sentiment analysis on the size and fit features because these two are arguably one of the more important factors influencing a consumer’s purchases, thereby also influencing the product rating and the performance of the recommendation system.

REFERENCES

Addressing Marketing Bias in Product Recommendations
Mengting Wan, Jianmo Ni, Rishabh Misra, Julian McAuley
WSDM, 2020

https://www.ibm.com/think/topihttps://evdelo.com/amazons-recommendation-algorithm-drives-35-of-its-sales/cs/collaborative-filtering

https://evdelo.com/amazons-recommendation-algorithm-drives-35-of-its-sales/

https://www.crn.com/news/applications-os/220100498/researchers-solve-netflix-challenge-win-1-million-prize