SI 618 Project 1 Report: Sales Analytics for Fashion Products on Amazon by Using PySpark

Yonnie Chan (yonniech@umich.edu)
October 23, 2022

1 Motivation

Online apparel and accessory sales in the United States are expected to surpass $118 billion in 2019, according to eMarketer, making it the largest product category by e-commerce sales (eMarketer, "The State of US Apparel Shopping in Five Charts", April 2018). Amazon is the largest online platform in the US, which is widely known as a go-to retail destination with a wide variety of item types.

Based on up to 2.6M products in the clothing, shoes & jewelry category in the US, the analysis starts with data manipulation and data preprocessing, including data cleaning, dataset merging, etc. Then, I will conduct sales data analytics, data visualization, and finally, gather business insights from the data analysis process.

In this project, I intend to explore datasets by PySpark to analyze various aspects of sales: top selling in this category, how top-performing sellers are winning on Amazon, and figuring out the key factors to becoming a top-sellers. By discovering the secret to success, I believe it would be helpful for third-party brands and retailers or those who are in E-commerce industries to improve their sales and to raise the market share of engaged customers on Amazon.

2 Datasets

For this project, I had access to two different datasets available on Amazon review data (2018) (https://jmcauley.ucsd.edu/data/amazon_v2/index.html). The two datasets are in the fashion category and are both given in **.**json format. I will elaborate on the details of the features I extracted from each dataset.

2.1 Products Meta Data

Total Records Count: 2,685,059.

This dataset contains the total product records in the fashion category on Amazon in 2018. The columns include the following information:

```
RangeIndex: 186637 entries, 0 to 186636
Data columns (total 16 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
```

```
 0   title            186632 non-null  object
 1   brand            139957 non-null  object
 2   feature          123875 non-null  object
 3   rank             180222 non-null  object
 4   date             185001 non-null  object
 5   asin             186637 non-null  object
 6   imageURL         132017 non-null  object
 7   imageURLHighRes  132017 non-null  object
 8   description      15869 non-null   object
 9   price            17799 non-null   object
 10  also_view        11595 non-null   object
 11  also_buy         21642 non-null   object
 12  fit              4831 non-null    object
 13  details          885 non-null     object
 14  similar_item     317 non-null     object
 15  tech1            97 non-null      object
dtypes: object(16)
```

2.2 Reviews Data

Total Records Count: 32,292,099.

The dataset shows the reviews of products in the fashion category on Amazon in 2018. The columns include the following information:

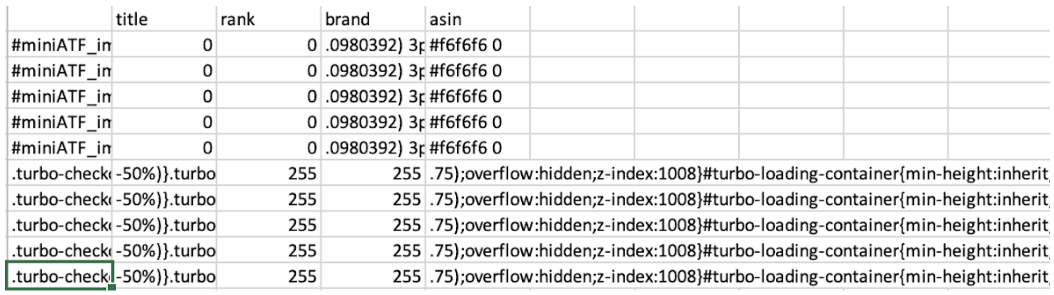
```
RangeIndex: 883636 entries, 0 to 883635
Data columns (total 12 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   overall         883636 non-null  float64
 1   verified        883636 non-null  bool
 2   reviewTime      883636 non-null  object
 3   reviewerID      883636 non-null  object
 4   asin            883636 non-null  object
 5   reviewerName    883544 non-null  object
 6   reviewText      882403 non-null  object
 7   summary         883103 non-null  object
 8   unixReviewTime  883636 non-null  int64
 9   vote            79900 non-null   object
 10  style           304569 non-null  object
 11  image           28807 non-null   object
dtypes: bool(1), float64(1), int64(1), object(9)
```

3 Data Manipulation

The datasets were not in a clean and preferred format, so I needed to conduct data manipulations process to prepare the dataset for the next analysis and visualization step. The source code of data cleaning can be found in **Data_Cleaning.ipynb**.

Step 1: Perform data cleaning

Firstly, I kept only the necessary variables that I'm interested in. In the products dataset, there were originally 16 fields and 2.6M products with missing values in this json file. When I tried to extract four fields (*title, rank, brand, and asin*) for further research, I found that there were several missing, messy, and unwanted records in *rank*.

For example:
```
- 13,052,976inClothing,Shoesamp;Jewelry(
- >#250,901inElectronics
-
>#9,005,799inHome&Kitchen(Seetop100)>#21,622inHome&Kitchen>Storage&Organiz
ation>Baskets,Bins&Container
- And also some web-crawling format errors, like:
```


	title	rank	brand	asin
#miniATF_in	0	0	.0980392) 3p	#f6f6f6 0
#miniATF_in	0	0	.0980392) 3p	#f6f6f6 0
#miniATF_in	0	0	.0980392) 3p	#f6f6f6 0
#miniATF_in	0	0	.0980392) 3p	#f6f6f6 0
#miniATF_in	0	0	.0980392) 3p	#f6f6f6 0
.turbo-check	-50%)}.turbo	255	255	.75);overflow:hidden;z-index:1008}#turbo-loading-container{min-height:inherit
.turbo-check	-50%)}.turbo	255	255	.75);overflow:hidden;z-index:1008}#turbo-loading-container{min-height:inherit
.turbo-check	-50%)}.turbo	255	255	.75);overflow:hidden;z-index:1008}#turbo-loading-container{min-height:inherit
.turbo-check	-50%)}.turbo	255	255	.75);overflow:hidden;z-index:1008}#turbo-loading-container{min-height:inherit
.turbo-check	-50%)}.turbo	255	255	.75);overflow:hidden;z-index:1008}#turbo-loading-container{min-height:inherit

Those records would become noices and largely mislead our direction through analysis process. I split the string by "in" words to separate the rank (data type: integer) and the category (data type: string). Then, I replace abnormal words such as "amp;" or ">#" and remove those that are not in the category I want to include. I gave several condition commands to make sure only data in clothing, shoes & jewelry can be left and have to be in the right format, which is supposed to be only an integer in every line. The information of records after cleaning is listed below.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 133281 entries, 0 to 133280
Data columns (total 4 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   title   133281 non-null  object
 1   rank    133281 non-null  int64
 2   brand   133281 non-null  object
 3   asin    133281 non-null  object
dtypes: int64(1), object(3)
```

In reviews dataset, there were originally 12 fields and 32M reviews records with missing value in this json file. The main fields I want to extract from this dataset are *overall(rating), asin, reviewText, and vote*. Moreover, I imputed missing values by replacing null values with zero in *vote*, and deal with *reviewText* to make them in the right format. Also, I filter out those reviews that were not verified. The information of records after cleaning is listed below.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 827527 entries, 0 to 827526
```

```
Data columns (total 4 columns):
 #   Column      Non-Null Count   Dtype  
---  ------      --------------   -----  
 0   overall     827527 non-null  float64
 1   asin        827527 non-null  object 
 2   reviewText  827527 non-null  object 
 3   vote        827527 non-null  int64  
dtypes: float64(1), int64(1), object(2)
```

Step 2: Join the two datasets by shared '*asin'* column

Next, I intend, using Spark SQL, to join two datasets on their sharing columns with Spark SQL to obtain a completed dataset. The overall concept of the merging task would be: combining the Product dataset and reviews dataset on the basis of *asin*. After aggregating these datasets, I plan to manipulate, transform, and analyze my data using PySpark.

4 Analysis and Visualization

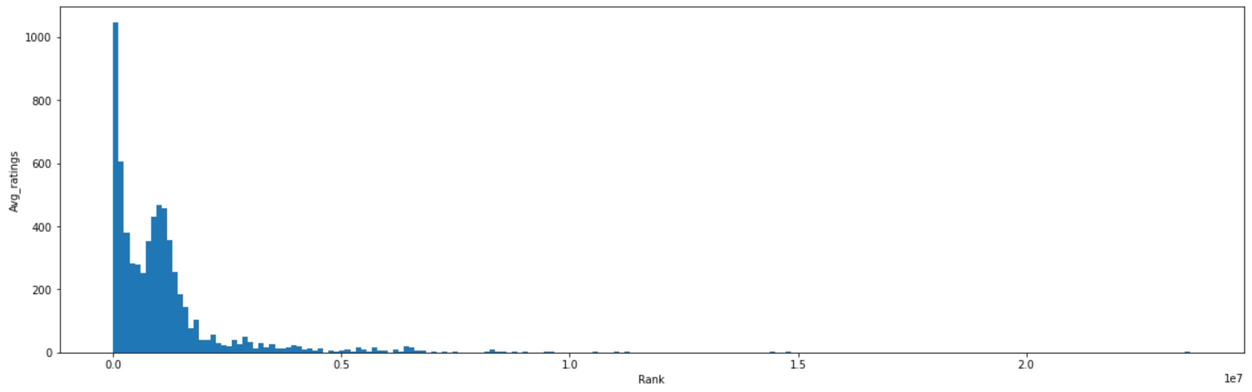
The three tasks in this part were accomplished by using large-scale computation techniques PySpark. The source code can be found in **si_618_project1.py**.

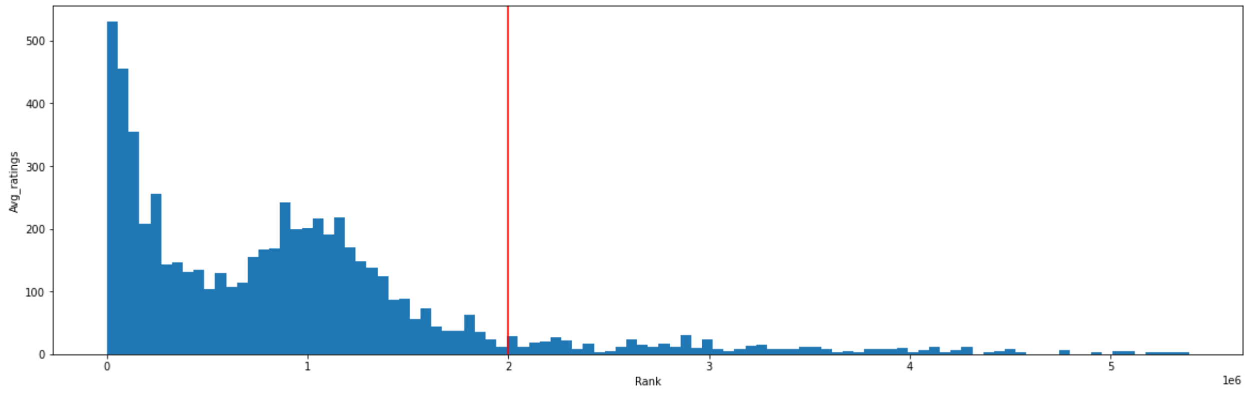
4.1 Whether *Num_of_Reviews, Avg_ratings, and Rating_power* affect the product ranking, and find the top products

To determine what kinds of top products are representative, there were several indicators in my mind. It would be an easy and convenient task for me to choose the highest ranking in the category, but top100, top 1,000, or top 10,000? It was also feasible to filter by the number of reviews because products with high review counts are supposed to have more discussion volume.

To figure out this question, I use Spark SQL to create the columns. After converting the pyspark.sql.dataframe into pandas.dataframe, I used ".corr()" command to calculate the correlations between *Num_of_Reviews* by the Pearson method. However, I found that the correlation was not large enough. I decided to add more variables such as *Avg_ratings* and *Rating_power (Num_of_Reviews*Avg_ratings)* to make a correlation analysis, then I returned to Spark SQL to create the columns I may need in this task.

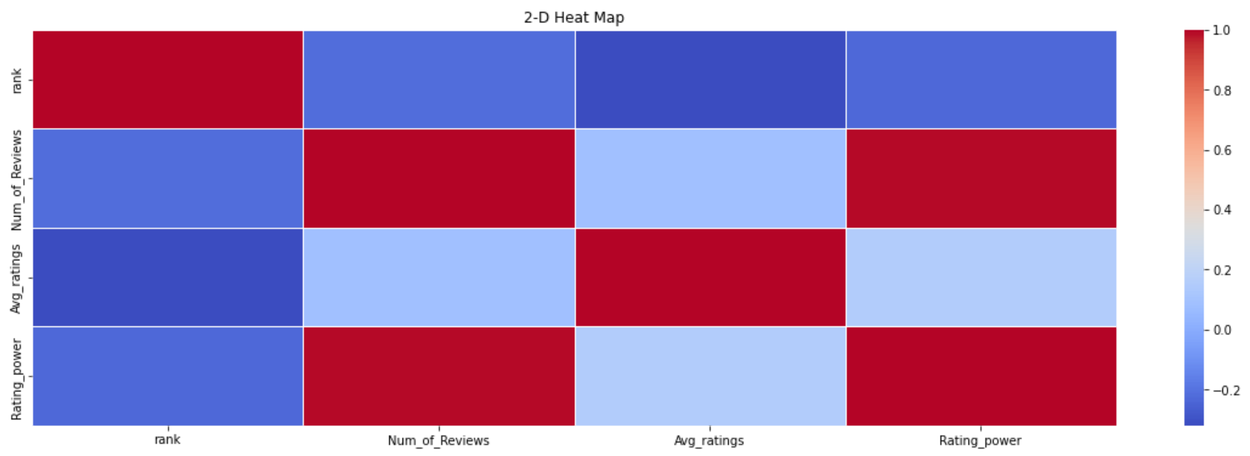
After creating a correlation matrix and finding the most significant factor, which is *Avg_ratings*, I generated a histogram for *Avg_ratings* and *rank* by pyplot package to decide the range of ranking that have the most representative data.





These two plots demonstrate a much smoothy pattern after 2e6 in the x-axis. Thus, I fine-tune the number of the dataset I extract for correlation analysis. Then I generated the following correlation matrix, and heatmap to observe correlations between these four variables. The data is representative enough to make a conclusion for this part of the analysis. The correlation coefficient between *rank* and *Avg_ratings* is the largest, which means the average ratings of the product could affect the rank. The higher the average of ratings is, the lower the rank is (more best-selling).

	rank	**Num_of_Reviews**	**Avg_ratings**	**Rating_power**
rank	1.000000	-0.218502	-0.320289	-0.232901
Num_of_Reviews	-0.218502	1.000000	0.084074	0.992350
Avg_ratings	-0.320289	0.084074	1.000000	0.151516
Rating_power	-0.232901	0.992350	0.151516	1.000000



4.2 Take *num_of_helpful_reviews* into consideration to see if it affects the product ranking, and refine the weighted-rating formula by *vote*

Number of helpful reviews may be weighted when calculating the average ratings. I desire to dig out the best formula that could best represent the ranking of the products. Therefore, I refine the rating as a weighted rating and calculate by Spark SQL:

$$New\,Weighted\,Rating\;=\;\sum_{i=1}^{n}\left(overall(rating)_i * vote_i\right) / \sum_{i=1}^{n}\left(vote_i\right)$$

$$\{\,vote_i\;=\;1,\;if\;vote_i\;=\;0\;vote_i,\;if\;vote_i\neq\;0$$

For example, if a product has the reviews from different customers as following:

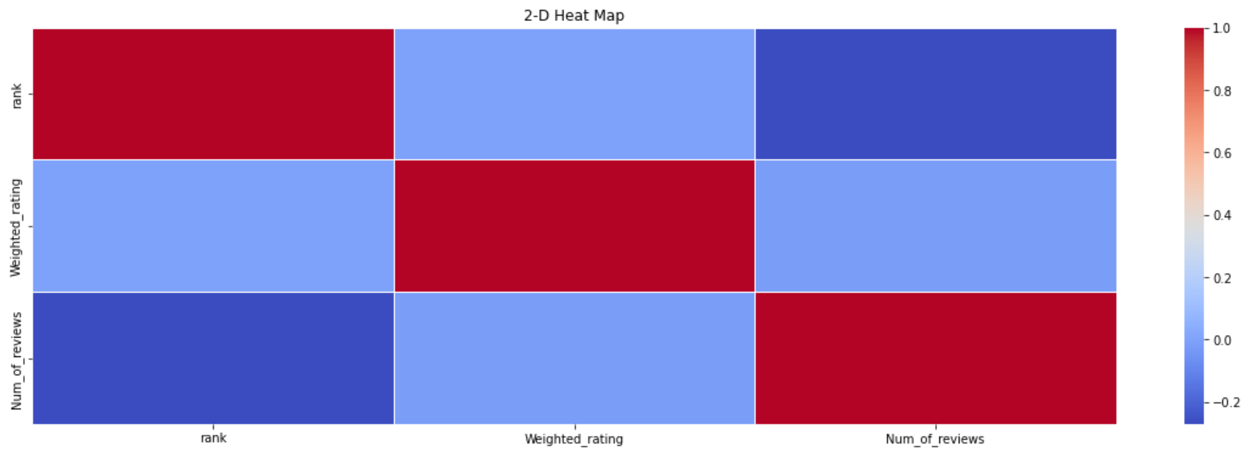
Id	overall	vote
1	5	2
2	2	4
3	1	0

The new weighted rating would be (5*2+2*4+1*1)/(2+4+1)

With the similar process, I generate a correlation matrix to check whether the formula is suitable. However, I found that the correlation coefficient between *rank* and *weighted_rating* is super low. That means the formula I created couldn't be a good indicator for ranking, and it should be optimized a little more.

	rank	Weighted_rating	Num_of_reviews
rank	1.000000	-0.004044	-0.270408

	rank	Weighted_rating	Num_of_reviews
Weighted_rating	-0.004044	1.000000	-0.018467
Num_of_reviews	-0.270408	-0.018467	1.000000



4.3 Using the Bayesian Average in Ranking

After several trials, I decided to try Bayesian Average (https://www.algolia.com/doc/guides/managing-results/must-do/custom-ranking/how-to/bayesian-average/), to see whether the formula could have a better result.

$$bayesAvg = (productRatingsCount + C * m)/(productRatingsCount + C)$$

I made a calculation of bayesian average in an inner layer in Spark SQL, and pair them with the products dataset. After calculation and correlation matrix observation, the formula ended up failing to have a higher correlation coefficient between *rank* and *bayes_avg*.

	rank	bayes_avg
rank	1.000000	-0.167957
bayes_avg	-0.167957	1.000000

5 Challenges

There are several bottlenecks during the process of the project. Firstly, when gathering the suitable datasets, I found sales or revenue of the products or brands (eg. number of items sold, cost) is considered confidential data of a company, which is practically impossible to assess. However, I need an indicator to be served as a target variable to verify the success of the product. I then fine-tune my topic to match the datasets I could collect. Therefore, I made an assumption that rank (best-seller) is considered to represent the success of the products in this project.

Moreover, because of the limited access to the dataset, if I could add more dimensions for consideration (eg. promotion rate) would be better. Then I could also have more probability to try out more formulas to obtain a better result.