Data Wrangling

Soo Hyeon Kim

September 2018

1. Data Obtaining

Data was found through web search and acquired at the repository of Dr. McAuley, Assistant Professor at UCSD (University of California San Diego) with his permission (http://jmcauley.ucsd.edu/data/amazon/index.html). I have chosen most exhaustive and inclusive dataset which holds around 82 million reviews from May 1996 to July 2014 without duplicated data, together with metadata which includes descriptions, price, sales-rank, brand name and co-purchasing links of 9.4 million products. Both files were formatted in one-review-per-line loose JSON (list of JSONs). The reason it is loose JSON is unlike normal JSON files, I could not load file conveniently and directly using json.load() from Python's json library due to '\n' (newline character) and double quote - single quote issues. In this regard, I had issues to tackle: 1) how to import large JSON files onto Python Jupyter Notebook framework, 2) manipulating data so that it can be more easily used, 3) cleaning data, especially about missing values, 4) converting certain features to something more handy 5) if exists, how to deal with data skewness, 6) normalizing numerical features and 7) finalizing data. Before we move onto next session, see below examples of the format.

```
{
  "reviewerID": "A2SUAM1J3GNN3B",
  "asin": "0000013714",
  "reviewerName": "J. McDonald",
  "helpful": [2, 3],
  "reviewText": "I bought this for my husband who plays the
piano.  He is having a wonderful time playing these old hymns.
The music  is at times hard to read because we think the book
was published for singing from more than playing from.  Great
purchase though!",
  "overall": 5.0,
  "summary": "Heavenly Highway Hymns",
  "unixReviewTime": 1252800000,
  "reviewTime": "09 13, 2009"
}
```

<Figure 1. Review data example>

Where

1. reviewerID - ID of the reviewer, e.g. A2SUAM1J3GNN3B
2. asin - ID of the product, e.g. 0000013714
3. reviewerName - name of the reviewer
4. helpful - helpfulness rating of the review, e.g. 2/3 (2 people think this review is helpful out of 3 people who voted)
5. reviewText - text of the review

6. overall - rating of the product
7. summary - summary of the review
8. unixReviewTime - time of the review (unix time)
9. reviewTime - time of the review (raw)

```
{
  "asin": "0000031852",
  "title": "Girls Ballet Tutu Zebra Hot Pink",
  "price": 3.17,
  "imUrl":
"http://ecx.images-amazon.com/images/I/51fAmVkTbyL._SY300_.jpg",
  "related":
  {
    "also_bought": ["B00JHONN1S", "B002BZX8Z6", "B00D2K1M3O",
"0000031909", "B00613WDTQ", "B00D0WDS9A", "B00D0GCI8S",
"0000031895", "B003AVKOP2", "B003AVEU6G", "B003IEDM9Q",
"B002R0FA24", "B00D23MC6W", "B00D2K0PA0", "B00538F5OK"],
    "also_viewed": ["B002BZX8Z6", "B00JHONN1S", "B008F0SU0Y",
"B00D23MC6W", "B00AFDOPDA", "B00E1YRI4C", "B002GZGI4E",
"B003AVKOP2", "B00D9C1WBM", "B00CEV8366", "B00CEUX0D8",
"B0079ME3KU", "B00CEUWY8K", "B004FOEEHC", "0000031895",
"B00BC4GY9Y", "B003XRKA7A", "B00K18LKX2", "B00EM7KAG6"],
    "bought_together": ["B002BZX8Z6"]
  },
  "salesRank": {"Toys & Games": 211836},
  "brand": "Coxlures",
  "categories": [["Sports & Outdoors", "Other Sports", "Dance"]]
}
```

<Figure 2. Meta data example>

Where

1. asin - ID of the product, e.g. 0000031852
2. title - name of the product
3. price - price in US dollars (at time of crawl)
4. imUrl - url of the product image
5. related - related products (also bought, also viewed, bought together, buy after viewing)
6. salesRank - sales rank information
7. brand - brand name
8. categories - list of categories the product belongs to

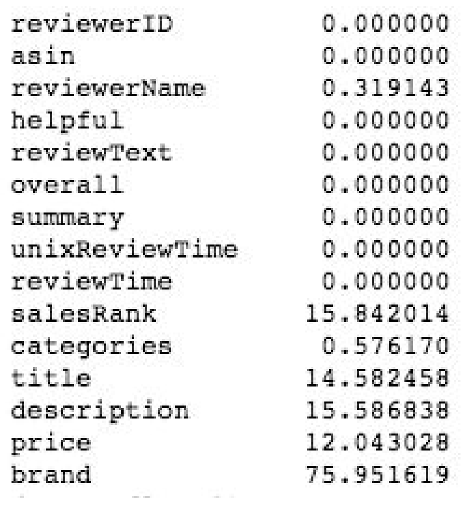
2. Importing Data

Fortunately, the author provided a reference for how to read the data into Panda's DataFrame. The only problem was the data itself is too large to be run on local machine. Thus, I decided to "chunk-ify" data into set of data which are somewhat more manageable. Basically, I splitted the review data into 10 sub data frame.

3. Manipulating / Filtering Data

As shown in <Figure 1>, 'helpful' feature is in such a format as [2, 3] which is equivalent to 2/3 and interpreted as 2 people think this review is helpful out of entire 3 people who participated in voting helpfulness for the product. By construction, I will call 2 "helpful numerator" and 3 "helpful denominator". Therefore, helpful [2, 3] is read as 'helpful numerator 2 and helpful denominator 3'. I decided to focus on reviews with more than 9 of helpful denominator and less than 10,000 of helpful denominator, for small number of participants is hard to believe the model prediction is well generalized and more than 10,000 helpful denominator is rare so I regarded those entries as outlier. You can think of helpful denominator as voter population and helpful numerator as number of people who voted for a particular review. Filtering each 10 chunks of dataset took around 5 minutes on my 32 GB RAM macbook pro. After each chunk was manipulated, those 10 data frames were concatenated into one data frame. As a side note, applying Pandas to_dateframe method to 'reviewTime' column first takes longer time than filtering dataframe with map method as string type. After filtering, review dataset reduced to 4,944,438 rows. As for metadata, I dropped 'related' and 'imUrl' columns because our goal is predicting helpfulness ratings. At this stage, I had two separate data frames: review data frame and metadata data frame. However, they both have 'asin' (product ID) column and therefore I could attempt to merge them together. After an examination, I noticed that asin is unique in metadata data frame in contrast to review data frame. Hence, when merging, I set 'validate' parameter as 'many_to_one'. Now we have merged 4,753,979 rows and 15 columns and I noticed that merging brought somewhat positive effect in reducing missing values in 'reviewerName'.

4. Cleaning Data



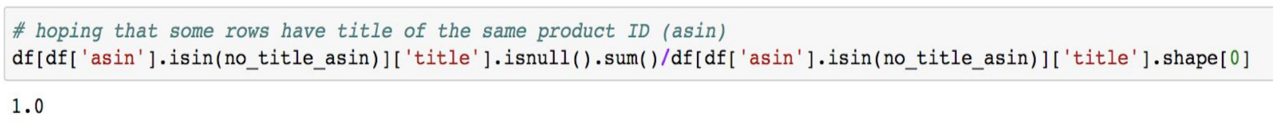
```
reviewerID         0.000000
asin               0.000000
reviewerName       0.319143
helpful            0.000000
reviewText         0.000000
overall            0.000000
summary            0.000000
unixReviewTime     0.000000
reviewTime         0.000000
salesRank         15.842014
categories         0.576170
title             14.582458
description       15.586838
price             12.043028
brand             75.951619
```

<Figure 3. Missing proportion (%)>

There are 5 features which have over 12% missings and 2 features with under 0.6%. Especially, 'brand' has around 76% missing value proportion. We can assume that brand is not very important feature and thus drop the column. Also, both unixReviewTime and reviewTime mean the same and the only difference is format. Hence, we drop unixReviewTime column.

Finally, we do not need both reviewerName and reviewerId since we can identify a reviewer with his/her ID.

Regarding missings, I tackled ‘title’, ‘description’ and ‘price’ first and ‘saleRank’ and ‘categories’ later. First, I checked if I can fill the missing spots using product id (i.e. asin) from rows of same asin. I first extracted sub data frame with no title, description, and price separately and using unique() method, I collected their product id’s - no_title_asin, no_description_asin, and no_price_asin. Then I extract another sub data frame of which row asin is same as asin’s of rows with a missing value from previous step. Finally, I investigated how many rows are absent of title, description, and price out of all those rows of missing values in title, description and price. I provided code snippet for this part below.

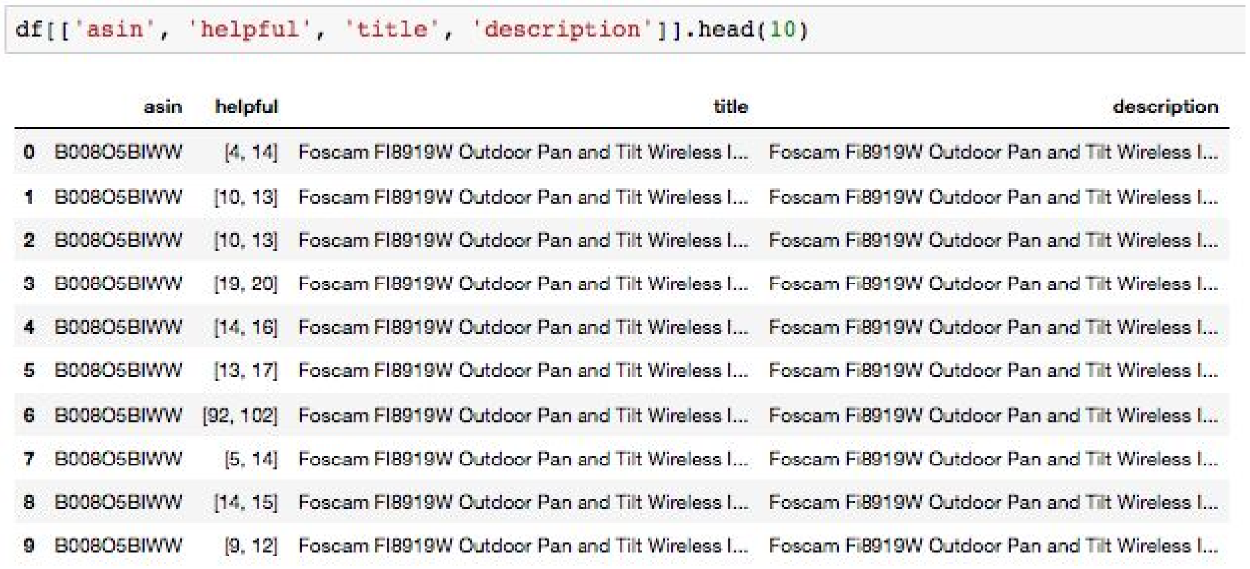


```
# hoping that some rows have title of the same product ID (asin)
df[df['asin'].isin(no_title_asin)]['title'].isnull().sum()/df[df['asin'].isin(no_title_asin)]['title'].shape[0]
```

```
1.0
```

<Figure 4. Checking the ratio of missing values for ‘title’ out of those which do not have title>

Interestingly all three features’ missing value proportion was 1.0 which means not a single row of same product id preserves title, description and price.



```
df[['asin', 'helpful', 'title', 'description']].head(10)
```

	asin	helpful	title	description
0	B008O5BIWW	[4, 14]	Foscam FI8919W Outdoor Pan and Tilt Wireless I...	Foscam Fi8919W Outdoor Pan and Tilt Wireless I...
1	B008O5BIWW	[10, 13]	Foscam FI8919W Outdoor Pan and Tilt Wireless I...	Foscam Fi8919W Outdoor Pan and Tilt Wireless I...
2	B008O5BIWW	[10, 13]	Foscam FI8919W Outdoor Pan and Tilt Wireless I...	Foscam Fi8919W Outdoor Pan and Tilt Wireless I...
3	B008O5BIWW	[19, 20]	Foscam FI8919W Outdoor Pan and Tilt Wireless I...	Foscam Fi8919W Outdoor Pan and Tilt Wireless I...
4	B008O5BIWW	[14, 16]	Foscam FI8919W Outdoor Pan and Tilt Wireless I...	Foscam Fi8919W Outdoor Pan and Tilt Wireless I...
5	B008O5BIWW	[13, 17]	Foscam FI8919W Outdoor Pan and Tilt Wireless I...	Foscam Fi8919W Outdoor Pan and Tilt Wireless I...
6	B008O5BIWW	[92, 102]	Foscam FI8919W Outdoor Pan and Tilt Wireless I...	Foscam Fi8919W Outdoor Pan and Tilt Wireless I...
7	B008O5BIWW	[5, 14]	Foscam FI8919W Outdoor Pan and Tilt Wireless I...	Foscam Fi8919W Outdoor Pan and Tilt Wireless I...
8	B008O5BIWW	[14, 15]	Foscam FI8919W Outdoor Pan and Tilt Wireless I...	Foscam Fi8919W Outdoor Pan and Tilt Wireless I...
9	B008O5BIWW	[9, 12]	Foscam FI8919W Outdoor Pan and Tilt Wireless I...	Foscam Fi8919W Outdoor Pan and Tilt Wireless I...

<Figure 5. Part of data frame>

As you can see in <Figure 5>, all rows of same asin share title and description, which means it does not have any difference among these rows and it is no use for analysis. Therefore, we drop title and description columns. However, I believe price can have impact on helpfulness score in that if a product is very expensive, more people would like to find helpful reviews and evaluate them. For this reason, I instead dropped rows without price.

Finally, I tried to simplify categories. At first glance, salesRank and categories columns includes similar information. Therefore, it is natural to attmept filling missing values in categories column with the help of salesRank. However, after a study, I came to conclusion that salesRank does not help with filling the missing spot in categories column. Furthermore, since categories column's missing rate is very small (~0.6%) and that of salesRank is over 15%. Therefore, I dropped salesRank column and removed rows without categories value. Since one product can have many categories and it seemed they are in order of how general it is - more general categories come before less general ones. In other words, first category in the list is the most general category that embraces subsequent categories and therefore, I used first item in the list as representative category of the product. At the same time I removed odd names such as '[', '#508510' together with missing value symbol: ''. Now we have pure 4,153,006 rows free from missing values.

5. Converting Certain Columns

One might notice that overall rating is on a scale of 0 to 5; however helpfulness is in such a format as [4, 14] which is equivalent to saying 4 people find it helpful out of 14 people who voted for helpfulness to this product. This is difficult to utilize to our end and therefore, I converted this metric to 0-5 rating metric same as overall rating. First, I split the column into helpful numerator and helpful denominator columns.

$$helpful\_num : helpful\_den = rating : 5$$
$$\therefore rating = 5 \times \frac{helpful\_num}{helpful\_den}$$

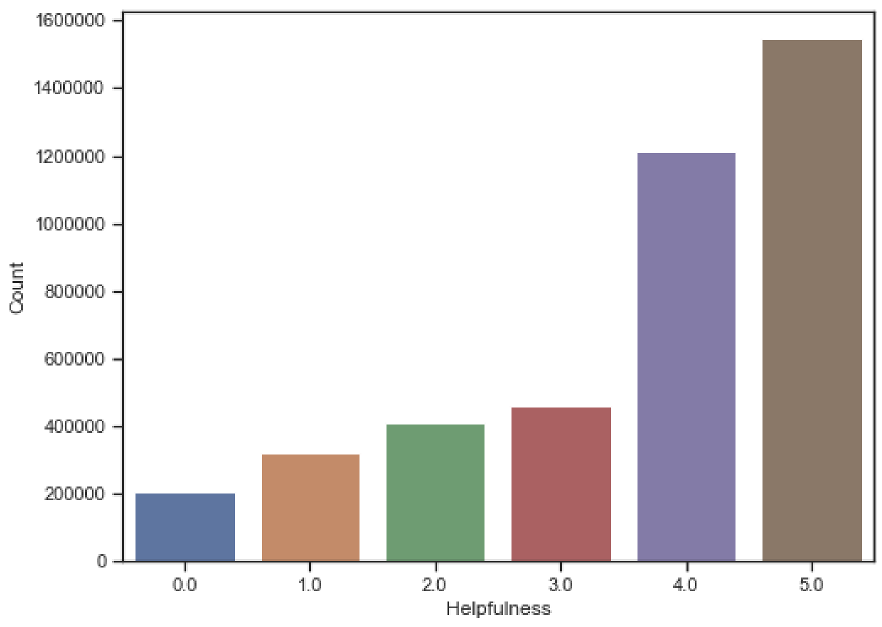
<Eq. 1. Rating Calculation>

I applied <Eq. 1> using these two columns resulting in seemingly continuous 0-5 rating values. To keep up with overall rating format and later use it for classification label, I rounded those values and could achieve {0.0, 1.0, 2.0, 3.0, 4.0, 5.0} distinct rating groups.

As I checked at Amazon website, and from this data, summary and reviewText are written up in seperate boxes. Therefore, I regarded summary as part of review and combined them together in the fashion that summary is the first sentence(s) and I created new column "review" and dropped the two columns.

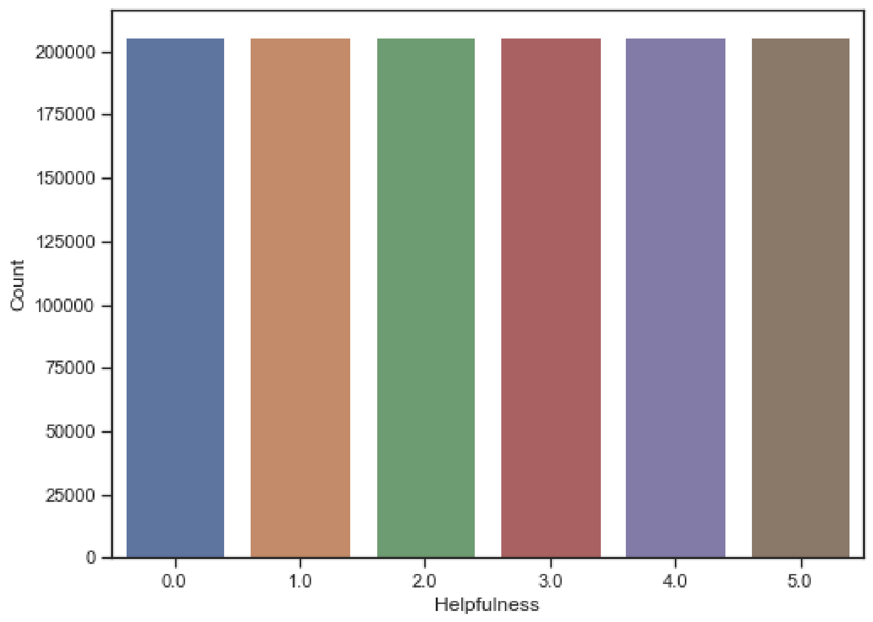
6. Appeasing Data Skewness

Helpfulness is our target column and therefore we should be more careful. Using skewed data can bring about negative results: 1) you can build bad model and your learned classifier will not be able to correctly identify the proper class; 2) the training process will invest a lot of time and effort in tuning "uninteresting parameters" as they seem to discriminate between the classes; and 3) your model can miss good features and emphasize useless features. All in all, skewed data could lead to models that are biased towards to the majority labels.



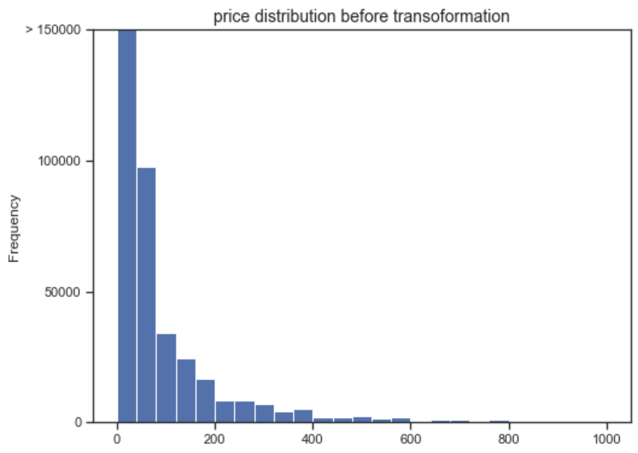
<Figure 6. Helpfulness rating distribution>

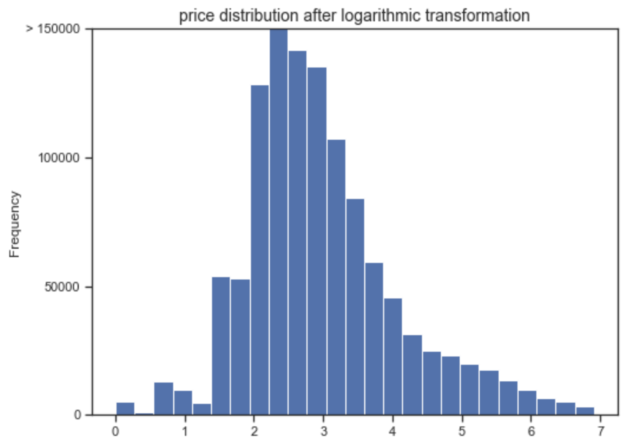
As you can see in <Figure 6>, helpfulness was heavily left-skewed. I adopted stratified sampling to mitigate the skewness. Since smallest helpfulness label group is 0 and its size is 205,906, in each label groups, I randomly sampled 206,000. Now, data is uniformly distributed when it comes to helpfulness rating, and our data size is reduced from 4.15M rows to about 1.24M rows.



<Figure 7. Helpfulness rating distribution after stratified sampling>

We may as well check price distribution because price can vary a lot.



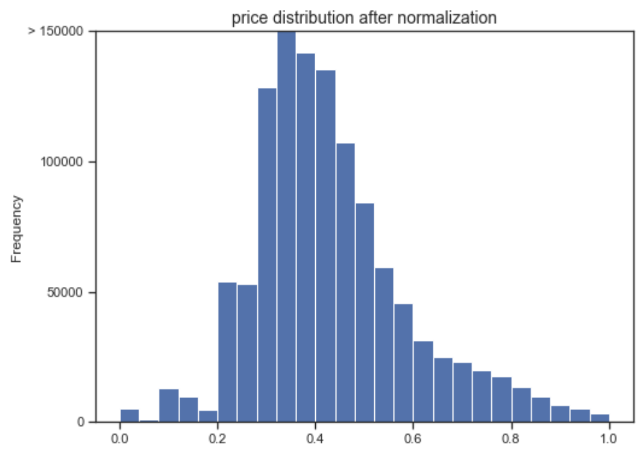


<Figure 8 & 9. Price distribution before (left) / after (right) transform>

We can tell from <Figure 8> price is highly right skewed. According to wikipedia (https://en.wikipedia.org/wiki/Data_transformation_(statistics)), when there is evidence of substantial skew in the data, it is common to transform the data to a symmetric distribution. Using logarithmic transformations will help the data spread more evenly and the very large and very small values do not negatively affect the performance of a learning algorithm. Using a logarithmic transformation significantly reduces the range of values caused by outliers. Care must be taken when applying this transformation however: The logarithm of 0 is undefined, so I translate the values by a small amount above 0 to apply the the logarithm successfully. As you can see in <Figure 9> price is now distributed more symmetrically.

7. Normalizing Numerical Features

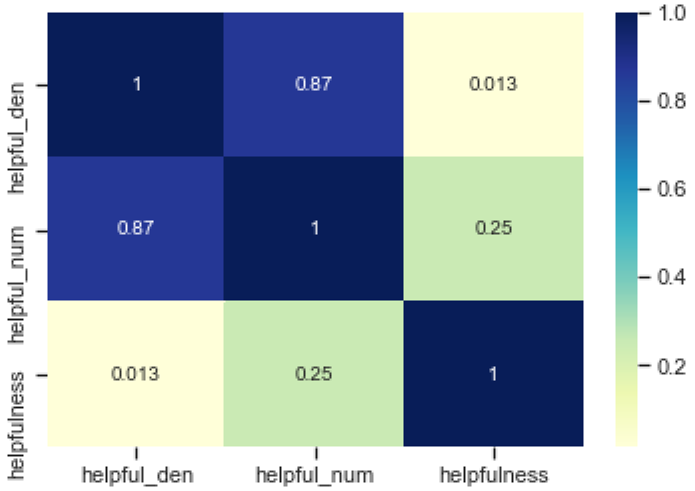
In addition to performing transformations on features that are highly skewed, it is often good practice to perform some type of scaling on numerical features. Applying a scaling to the data does not change the shape of each feature's distribution. However, normalization ensures that each feature is treated equally when applying supervised learners. For these reasons, I normalized price. Note that once scaling is applied, observing the data in its raw form will no longer have the same original meaning.



<Figure 10. Price distribution after normalization>

8. Finalizing Data

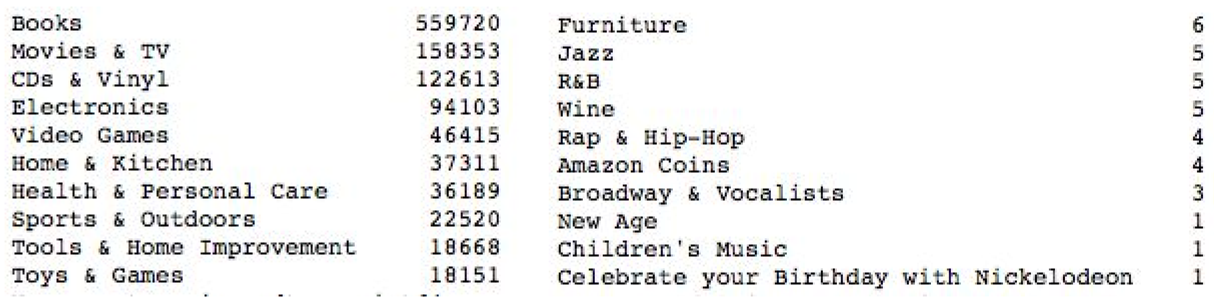
In this section, I will briefly discuss whether we should keep help_num and helpful_den columns and category distribution.



<Figure 11. Correlation heatmap among helpful_den, helpful_num and helpfulness>

<Figure 11> shows that helpful_num has 0.25 correlation to helpfulness and on the other hand, helpful_den has rather low correlation which is 0.013. This seems natural outcome because no matter how many people voted, it does not change the groundtruth - to get higher helpfulness rating, more people need to vote for the review. Therefore, we keep helpful_num and drop helpful_den.

From earlier steps, we learned that there are 72 categories. If there are many categories, it gets cumbersome when we later process and we do not want to learn our model on small size categories. Like how we did at Step 3. Filtering Data. I cut categories off at 400 because some categories only have 1, 3, or small number of products.



```
Books                         559720    Furniture                                6
Movies & TV                   158353    Jazz                                     5
CDs & Vinyl                   122613    R&B                                      5
Electronics                    94103    Wine                                     5
Video Games                    46415    Rap & Hip-Hop                            4
Home & Kitchen                 37311    Amazon Coins                             4
Health & Personal Care         36189    Broadway & Vocalists                     3
Sports & Outdoors              22520    New Age                                  1
Tools & Home Improvement       18668    Children's Music                         1
Toys & Games                   18151    Celebrate your Birthday with Nickelodeon 1
```

<Figure 12. 10 most (left) and least (right) frequent categories>

Cutting off at count 400, I ended up with 31 categories, dropped rows only takes up 0.21%.