Analysis and review of Amazon Dataset

Kerschel James

I. Introduction

The task of analyzing big data has become a prevalent issue as the world has become very data intensive. Researchers need to be able to analyze large amounts of data and provide reasonable predictions and show trends that can be used to further the profitability of the market but optimize buying. In this paper we analyze data from Amazon provided by **Julian McAuley**, UCSD. The dataset we used contains approximately 83 million reviews of products bought and reviewed on Amazon and our task was to analyze and generate interesting reports on the dataset along with another file with metadata of the products. The lesson to be learnt from this experiment is how to deal with big data in organizations as the traditional database sometimes cannot query the information in a manner that is reliable and useful.

II. Method of Analysis

A. Resources used

1) Programming language

The programming language used was python as it has many libraries such as pandas and sklearn that makes the reading and analysis of data simpler in terms of coding. Sklearn was used to aid the analysis of datapoints and plot it onto a graph for visuals.

2) Google Cloud VM

In order to complete this task, we made use of Google Cloud's VM service. We created a VM with 52GB ram and 8vCPU's in order to have a dedicated system that can run the code written without interruption.

3) Power Bi Desktop (Microsoft)

Power Bi is a very powerful visualization tool that can perform analysis on very large files. It was used in order to create the kmeans cluster and exploratory analysis on the data. Power BI in comparison to Excel was much more powerful due to the fact that Excel cannot open files with millions of rows.

B. Coding

Processing a large file such as this can take quite a long and depending on the method used it may be impossible to run on a normal low-end machine. Storing all the data at once then processing it would not be feasible for this task therefore pandas was used to read the file and process it piece by piece even though a high-end VM was used.

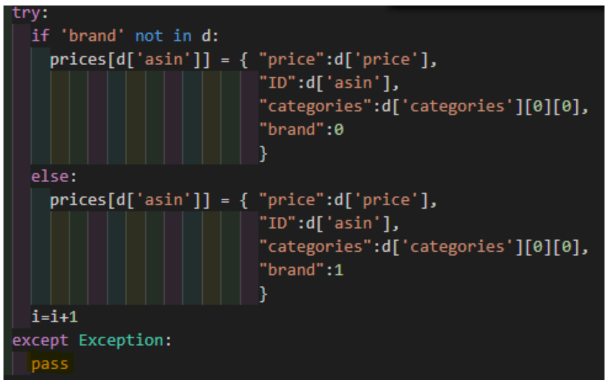
```
def parse(path):
  g = gzip.open(path, 'r')
  for l in g:
    yield eval(l)
```

Figure 1. read gzip file in parts

The file was also read in its compressed format which was of size 18GB. Python provided support to do this. When reading the data from the files, it was noted that even though the file was initially cleaned by the producer of the dataset there was still some inconsistencies in it. Our strategy to counteract this was selecting elements from the schema that we believe were essential for the analysis based on the requirements of the project.

Figure 2. try catch logic used

We also used "try catches" statements to ensure that the program does not crash unexpectedly when an error is found. An example of this was in the metadata.json.gz file where some items had an ID and some did not. Products without ID's were treated as corrupted data and was ignored. The method for doing this is shown in Figure 2. We were able to



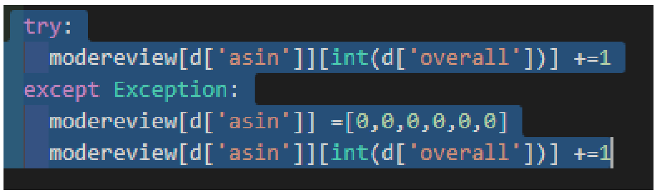
```
try:
  if 'brand' not in d:
    prices[d['asin']] = { "price":d['price'],
                          "ID":d['asin'],
                          "categories":d['categories'][0][0],
                          "brand":0
                          }
  else:
    prices[d['asin']] = { "price":d['price'],
                          "ID":d['asin'],
                          "categories":d['categories'][0][0],
                          "brand":1
                          }
  i=i+1
except Exception:
  pass
```

reduce the size of the file by using a dictionary and storing only the unique products in it since the file had multiple reviews for each product. The aggressive_dedup.json.gz file had 85million records and approximately only 9.5million unique products. The metadata.json.gz file had 9.5million records but after the filtering was done on the set it was narrowed down to 5.5million records. This was due to mal formatted data and missing ID for products. Therefore the final result file records were 9.5million reviews and 5.5 million records of metadata.

III. Results

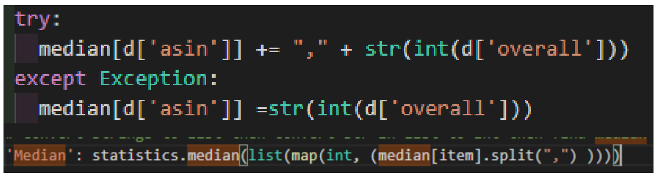
A. Product popularity report

As mentioned previously from the records we filtered each unique product and calculated the Mean, Mode and Median. The approach to calculate Mean was each product had a separate key in a dictionary which was the ASID code. For each ASID code the rating of the review was added to the dictionary value and a count was kept. The average value was calculated through each iteration of the product.
On the other hand, the Mode and Median were only calculated at the end of reading all the data. Each rating of each product was stored in a dictionary array being appended each time a review is found. The to find the mode each product had an array of size 1-5 where the index corresponds to the rating received as in the code below. At the end, all that was needed was to find the array of each product with the highest value.



```
try:
    modereview[d['asin']][int(d['overall'])] +=1
except Exception:
    modereview[d['asin']] =[0,0,0,0,0,0]
    modereview[d['asin']][int(d['overall'])] +=1
```

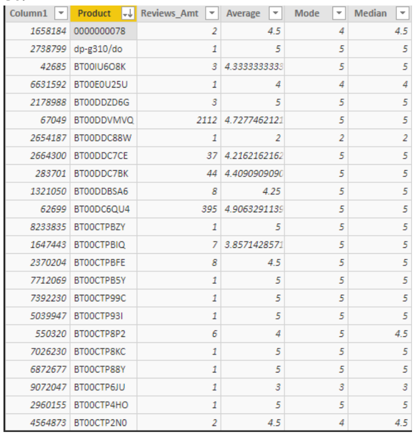
For the median each rating of a product was appended to a comma delimited string. It was then split and sorted to find the median.



```
try:
    median[d['asin']] += "," + str(int(d['overall']))
except Exception:
    median[d['asin']] =str(int(d['overall']))
```

```
'Median': statistics.median(list(map(int, (median[item].split(",") ))))
```

Figure 3 shows a snapshot of the Mean Median and Mode of each product.



Column1	Product	Reviews_Amt	Average	Mode	Median
1658184	0000000078	2	4.5	4	4.5
2738799	dp-g310/do	1	5	5	5
42685	BT00IU6O8K	3	4.333333333	5	5
6631592	BT00E0U25U	1	4	4	4
2178988	BT00DDZD6G	3	5	5	5
67049	BT00DDVMVQ	2112	4.7277462121	5	5
2654187	BT00DDC88W	1	2	2	2
2664300	BT00DDC7CE	37	4.2162162162	5	5
283701	BT00DDC7BK	44	4.4090909090	5	5
1321050	BT00DDBSA6	8	4.25	5	5
62699	BT00DC6QU4	395	4.9063291139	5	5
8233835	BT00CTPBZY	1	5	5	5
1647443	BT00CTPBIQ	7	3.8571428571	5	5
2370204	BT00CTPBFE	8	4.5	5	5
7712069	BT00CTPB5Y	1	5	5	5
7392230	BT00CTP99C	1	5	5	5
5039947	BT00CTP93I	1	5	5	5
550320	BT00CTP8P2	6	4	5	4.5
7026230	BT00CTP8KC	1	5	5	5
6872677	BT00CTP88Y	1	5	5	5
9072047	BT00CTP6JU	1	3	3	3
2960155	BT00CTP4HO	1	5	5	5
4564873	BT00CTP2N0	2	4.5	4	4.5

Figure 3. Mean,Median,Mode

B. *Type of Distribution*

We used a simple mathematical relation to determine the type of distribution from the data retrieved. When most points converge to a particular value the graph usually has a peak at that point. It can be said that the points convergence is the mean of the values. The mode value is the one which was selected the most. The relationship between the mode and the mean is as follows:
If mode == mean then the distribution is symmetrical
If the mean > mode the distribution is positively skewed
If the mean < mode the distribution is negatively skewed

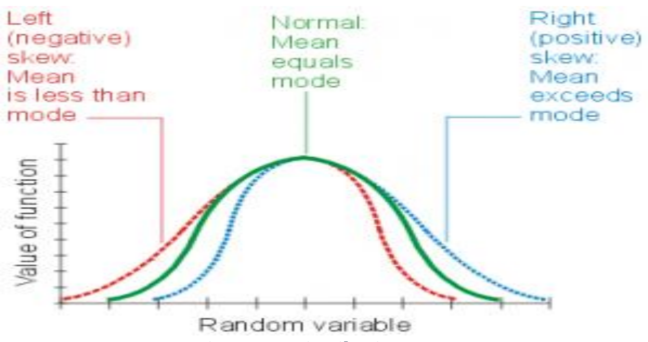


Figure 4. Distribution types

This strategy was used to determine the type of distribution as is Figure 4. From the previous section we calculated the mean of each item so all we did was we found the average across all the items.

Mean = 4.15
The mode across the whole dataset was then calculated and found to be:
Mode = 5
We were able to conclude that Mean < Mode therefore the distribution was negatively skewed. This could also be done by plotting the values on a line graph and looking at the curve, the calculation method was simpler for us since we had pieces on the calculation already from the previous section.

C. *Product features*

a) Most reviewed product

Product ID: B0054JZC6E was found to have the most reviews of 25366.

b) User who gave the most reviews

The person who review the most items was A14OJS0VWMOSWO and they had a total review count of 39448 and the average rate they gave each product was 5.

c) Most helpful reviews

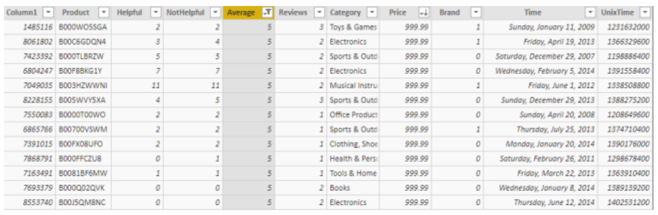
A14OJS0VWMOSWO had the most helpful reviews. The helpfulness was not used as a ratio we only took a summation of how many thumbs up the rating was given. This more models the way the system is in Amazon now since the dislike was removed.

d) Cheapest and most expensive high review

We defined a high review product as one which has a review greater than 4. We sorted the records according to price and review average and took highest one according to the sorting of the product ID since multiple items had the same high price, low price and rating.

- Expensive item

There were many items with the same expensive and cheap price with the same review but as mentioned before the top one alone was selected. From Figure 5 B000WO55GA was selected with a price of 999.99 and a rating of 5.

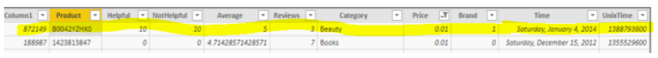


Column1	Product	Helpful	NotHelpful	Average	Reviews	Category	Price	Brand	Time	UnixTime
1485116	B000WO55GA	2	2	5	3	Toys & Games	999.99	1	Sunday, January 11, 2009	1231632000
8061802	B00C6GDQN4	3	4	5	2	Electronics	999.99	1	Friday, April 19, 2013	1366329600
7423392	B000TLBRZW	5	5	5	2	Sports & Outd	999.99	0	Saturday, December 29, 2007	1198886400
6804247	B00F8BKG1Y	7	7	5	2	Electronics	999.99	0	Wednesday, February 5, 2014	1391558400
7049035	B003HZWWNI	11	11	5	2	Musical Instru	999.99	1	Friday, June 1, 2012	1338508800
8228155	B005WVY5XA	4	5	5	3	Sports & Outd	999.99	0	Sunday, December 29, 2013	1388275200
7550083	B0000T00WO	2	2	5	1	Office Product	999.99	0	Sunday, April 20, 2008	1208649600
6865766	B00700VSWM	2	2	5	1	Sports & Outd	999.99	1	Thursday, July 25, 2013	1374710400
7391015	B00FX08UFO	2	2	5	1	Clothing, Shoe	999.99	0	Monday, January 20, 2014	1390176000
7868791	B000FFCZU8	0	1	5	1	Health & Pers	999.99	0	Saturday, February 26, 2011	1298678400
7163491	B0081BF6MW	1	1	5	1	Tools & Home	999.99	0	Friday, March 22, 2013	1363910400
7693379	B000Q02QVK	0	0	5	2	Books	999.99	0	Wednesday, January 8, 2014	1389139200
8553740	B00J5QM8NC	0	0	5	2	Electronics	999.99	0	Thursday, June 12, 2014	1402531200

Figure 5. Most expensive high review items

- Cheapest item

The cheapest item with the highest review was a Book and it received an average rating of 5 the ID of this item was B00DSNTDJ0 but the price was $0.00. I believe that this disqualifies it from being the cheapest item, we assumed that items free items were a bonus gift so we found the item that persons actually had to pay for. The other cheapest item was with a review of 5 was B0042YZHX0 with a price of $0.01. It was in the Beauty category.



Column1	Product	Helpful	NotHelpful	Average	Reviews	Category	Price	Brand	Time	UnixTime
872149	B0042YZHX0	10	10	5	3	Beauty	0.01	1	Saturday, January 4, 2014	1388793600
188987	1423813847	0	0	4.71428571428571	7	Books	0.01	0	Saturday, December 15, 2012	1355529600

Figure 6. Cheapest high review item

IV. Predictive Analysis

To perform the analysis on the data Power Bi Desktop was used. It as discussed previously is a very powerful tool which also was able to do kmeans clustering. The k for the clusters was automatically found by Power BI but we also plotted an elbow curve graph to verify the results.

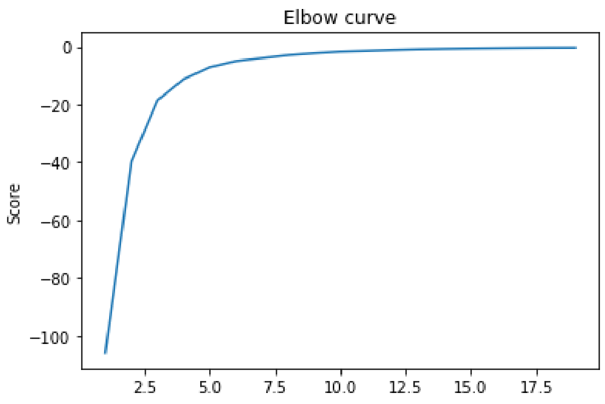


Figure 7. Elbow curve

Before the discovery of Power BI, sklearn was used to find the clusters and PCA to reduce the large set of variables to a small set that still contains most of the information in the large set. Power BI was much more visual and allowed more flexibility for a novice in machine learning techniques.

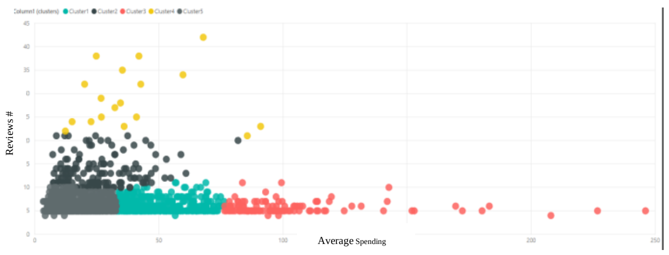


Figure 8. Reviews VS Average Spending

For the kmeans clustering the attributes chosen were # of reviews from user and the average amount of money they spent. From Figure 8 it can be seen that majority of persons who on average spend little money review items more and if we can make an assumption that everyone who buys an item reviews it then the cheaper the item is, the more persons buy it.

It is believed that this can be used to provide a recommendation to basically sell cheaper items because the overall income after selling a lot of cheap items could amount to more than selling a small quantity expensive items.

V. Pearsons'r

In order to determine if there was a correlation between the score and the helpfulness Pearson's r Product-Moment Correlation was used.

This test is a measure of the strength of a linear association between two variables and is denoted by r. How it works is that it attempts to find the best fit line through the data points and the variance from the line to each point is found using the formula below.

$$r = \frac{\sum XY - \frac{\sum X \sum Y}{N}}{\sqrt{\left(\sum X^2 - \frac{\left(\sum X\right)^2}{N}\right)}\sqrt{\left(\sum Y^2 - \frac{\left(\sum Y\right)^2}{N}\right)}}$$

Figure 9. Pearson's R formula

The formula in Figure 9 is used to find the sum of squared difference between the two variables in order to determine Pearson's r, r is the correlation product that determines how much effect one variable has on the other. These are the values of r and their interpretation:

R close to 1 means that there is a very strong positive relation between the two variables; as one increases so does the other.

R approximately 0 means that there is absolutely no relation between them.

R approximately -1 means that there is a negative relation between the two variables, meaning that as one variable increase the other decreases.

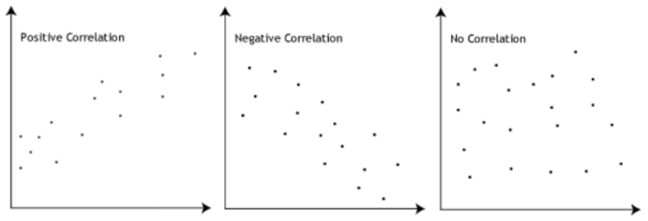


Figure 10. Correlation meaning

From our dataset we had to determine the relationship between helpfulness and score. The approach taken was we tallied the helpfulness ignoring the not helpful feature and looped our dataset passing each X and Y value through the formula. We found that r was -0.021 which means there is absolutely no correlation between the helpfulness and the score.

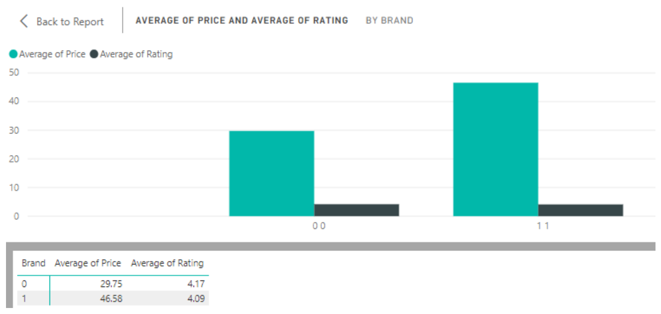
We also decided to implement our own Pearson's r since the libraries in python required us to load the whole dataset into memory to calculate it and with so many records this would not be a very feasible approach.

```
# Pearson's r
  # Ex^2 and Ey^2
  x_sqred += (d['overall'] * d['overall'])
  y_sqred += (d['helpful'][0]**2)
#   to be squared lower down
    x_sum_sq += d['overall']
    y_sum_sq += d['helpful'][0]
# E x*y
SumXbyY += d['overall'] * d['helpful'][0]
# Ex and Ey
  SumX = x_sum_sq
  SumY = y_sum_sq
  count = count +1
```

```
# (Ex)^2 and (Ey)^2
  x_sum_sq = x_sum_sq * x_sum_sq
  y_sum_sq = y_sum_sq ** 2
  PearsonR = (SumXbyY -
((SumX*SumY)/count)) / ( sqrt( (x_sqred  -
(x_sum_sq/count))  ) * (sqrt( (y_sqred  -
(y_sum_sq/count) )) ))
```

VI. Exploratory data

An interesting discovery was found in terms of brand, price and average rating of a product. Branded items were represented by 1 and non-branded were 0. From Figure 11 it was noted that brand named products have a much higher price than non-brand-named products which makes sense but their average rating compared to the other is approximately the same. Non-branded items even had a higher rating of 4.17 compared to branded items which got 4.09. It can be said that buying brand name items probably does not really have a great benefit other than self-satisfaction if you can get almost the same appreciation from the non-branded ones. The results can also be mean that persons who went for unbranded products were satisfied with it for the lower price even through it was not as good as the branded one, they still liked it for the price gave it a great review.



Brand	Average of Price	Average of Rating
0	29.75	4.17
1	46.58	4.09

Figure 11. Brand vs Non-Brand items rating and price

Further investigation was done to determine which category products in the branded showed the drastic difference in price. We separated each category and it was found that every branded item had a higher price than the non-branded version. It can be seen distinctly in the category "Furniture and Decorations" since the price of both the branded and non-branded was high which separated it from them other categories. On average the price difference for a branded compared to non-branded item was found to be approximately $17.00

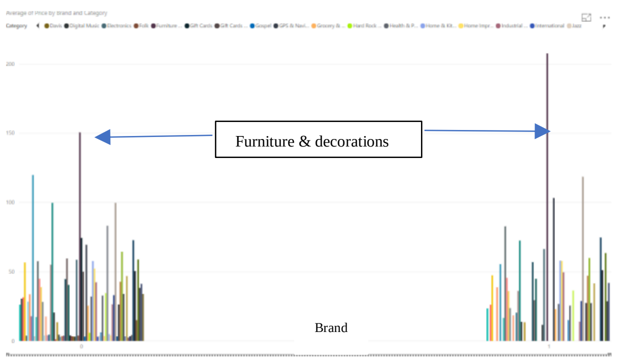


Figure 12. Categories of Band price difference

We also plotted a clustering chart of # of reviews Vs Average rating of product. The dataset was quite large and patterns were not as distinct with the large set so we focused on the category of Electronics. As mentioned previously Power BI automatically finds the best k with the use of the elbow curve method and put the points into distinct clusters. We noticed that each cluster represented a popularity/average bracket. As you go up the average axis the higher the average rating of the products the more reviews, they in turn receive. Also, someone cannot review an item without purchasing it first on Amazon so we can also say that the higher the rating of the product the more sales it will receive.

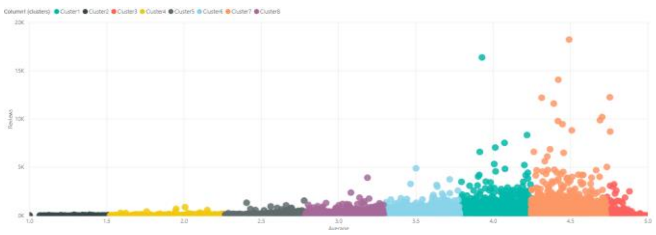


Figure 13 # of reviews vs Average rating

VII. Future work to consider

The findings from the kmeans clustering graph was not knowledge that was already know through general economics. In consideration of this if different attributes were chosen such as helpfulness and average review or review count, a better recommendation for future trends may be observed.

VIII. Appendices

Amazon.py – Python file for reading the amazon files and creating the csv files used for processing

Results.txt – File created from amazon.py with metadata about the results.

Trends.py – file used to extract information from products in metadata file as shown in figure 5. This was used for the exploratory analysis.

means.py - used to generate the elbow curve. Initial implementation of kmeans before the use of Power BI
Trends.csv – contains information extracted from trends.py such as average rating, brand, category ,helpfulness, nonhelpfulness, price, productID,# of review, Time, and Unixtime of each product.

Users.csv – Each unique userID and the average amount of money they spent per product along with the number of products they bought.

Link for all the Files -
https://drive.google.com/open?id=1r10fuA7-_261Hh9Qqz6vrRIUA7GkvwbS