Is Amazon Lying To Me?

CSC1143 Data Management & Visualisation

Student Details:
1. Reeha Althaf (A00050820) reeha.althaf2@mail.dcu.ie
2. Janavi Ravi (A00048806) janavi.ravi2@mail.dcu.ie

Table of Contents

Abstract

This project examines Amazon's displayed ratings and whether they accurately reflect the opinions of verified purchasers across different price points. It also explores how this 'trust gap' evolves as categories accumulate non-verified reviews. The aim is to visualise how rating inflation varies by product price and category, potentially revealing targeted manipulation of products.

Dataset

The datasets from McAuley-Lab/Amazon-Reviews-2023 [1], available on Hugging Face, comprise Amazon's User Reviews, which are roughly 275.41 GB in size, and Product Metadata, which is approximately 101.4 GB. The User Review dataset comprises approximately 571.54 million reviews from May 1996 to September 2023, and the Item Metadata dataset contains approximately 35 million items listed. The above-mentioned datasets meet the volume and variety requirements of big data. Below is the schema for the datasets:

For User Review:

Field	Item	Explanation
rating	float	Rating of the product (from 1.0 to 5.0).
title	str	Title of the user review.
text	str	Text body of the user review.
images	list	Images that users post after they have received the product. Each image has different sizes (small, medium, large), represented by the small_image_url, medium_image_url, and large_image_url, respectively.
asin	str	ID of the product.
parent_asin	str	Parent ID of the product. Note: Products with different colors, styles, and sizes usually belong to the same parent ID. The “asin” in previous Amazon datasets is actually the parent ID. Please use the parent ID to find product meta.
user_id	str	ID of the reviewer
timestamp	int	Time of the review (unix time)
verified_purchase	bool	User purchase verification
helpful_vote	int	Helpful votes of the review

For Item Metadata:

Field	Type	Explanaion
main_category	str	Main category(i.e., domain) of the product
title	str	Name of the product
average_rating	float	Rating of the product shown on the product page
rating_number	int	Number of ratings in the product
features	list	Bullet-point format features of the product
description	list	Description of the product
price	float	Price in US dollars (at time of crawling)
images	list	Images of the product. Each image has different sizes (thumb, large, hi_res). The "variant" field shows the positions of the image
videos	list	Videos of the product, including title and URL
store	str	Store name of the product
categories	list	Hierarchical categories of the product
details	dict	Product details, including materials, brand, sizes etc
parent_asin	str	Parent ID of the product
bought_togther	list	Recommended bundles from the websites

The data is stored in a distributed format that we access through PySpark. This setup supports scalable operations, such as grouping, filtering, and aggregating, over more than 500 million rows.

Data Exploration

In the initial exploration, we compute basic summary statistics, such as rating counts per score. This indicates that most reviews fall between 1 and 5 stars, but a small number (approximately 151) have a rating of 0.0, which is not a valid rating on Amazon. A detailed inspection reveals that these zero-rated reviews contain standard review text, and the related products have non-zero ratings on the live site. This indicates that the zeros likely stem from errors during the crawling process. Since zero-rated rows comprise a tiny portion of the total (approximately 151 out of more than 571 million records), we decided to remove them and limit the dataset to ratings between 1 and 5.

```
+------+---------+
|rating|    count|
+------+---------+
|   0.0|      151|
|   1.0| 58930529|
|   2.0| 28106622|
|   3.0| 40329433|
|   4.0| 72502401|
|   5.0|371675761|
+------+---------+
```

Figure 1 Distribution of the ratings

The user review dataset had a Boolean type value for the verified_purchase column, indicating whether the review left by the user was actually based on a purchase of the product or whether the review was left without the product.

```
+-----------------+---------+
|verified_purchase|    count|
+-----------------+---------+
|             true|512356685|
|            false| 59188212|
+-----------------+---------+
```

Figure 2 Distribution of the verified_purchase

For the Item Metadata dataset, during extraction, we encountered difficulties loading the data and therefore decided to define the schema beforehand to ease the data loading process. We also discovered that several rows were missing the "Main Category" column. Since the data available from Hugging Face is organised by category, we created a new column called "Category From File" that contains the name of the category as specified in the file from which the particular row originated.

```
# Define schema manually to avoid reading 'details' field
schema = StructType([
    StructField("parent_asin", StringType(), True),
    StructField("title", StringType(), True),
    StructField("price", DoubleType(), True),
    StructField("average_rating", DoubleType(), True),
    StructField("rating_number", IntegerType(), True),
    StructField("main_category", StringType(), True),
    StructField("store", StringType(), True),
])
```

Figure 3 Manually defined schema for Item Metadata

```
# Extract category from filename
filename = os.path.basename(file)  # e.g., "meta_Electronics.jsonl"

# Remove "meta_" and ".jsonl", then clean up
category_from_file = filename.replace('meta_', '') \
                             .replace('.jsonl', '') \
                             .replace('_', ' ') \
                             .replace('...', ' ')  # Handle truncated names
```

Figure 4 Category Extraction from the File

We then proceeded to clean up our Item Metadata dataset, dropping rows that had null values for the price andaverage_rating columns.
We then used the parent_asin column to join both datasets. We used parent_asin instead of asin because products of different colours or sizes have the same asin but a different parent_asin.

Upon joining the dataset, we split the verified_purchase and its related data into two categories (non-verified purchase-related columns and verified purchase-related columns) and filtered them such that each category has at least five verified and unverified purchase counts.
We extracted the following columns from the dataset to create a parquet file that we could visualize using Tableau:

1. parent_asin
2. price
3. main_category
4. category_from_file
5. user_id
6. verified_avg_rating
7. verified_count
8. nonverified_avg_rating
9. nonverified_count
10. total_reviews
11. trust_gap_stars
12. trust_gap_pct

Note: While visualizing the new sub-dataset, we noticed that the formula used for 'trust_gap_stars' was incorrect. Therefore, we created a calculated field in Tableau called 'True gap' to recalculate the gap using the correct formula.

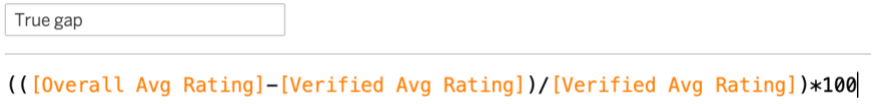

Figure 5 Updated formula calculated using Tableau

Visualization

During exploration, we examine rating distributions through tabular summaries and counts ordered by rating value to check for skew and identify anomalies. Sample outputs include tables listing each distinct rating with its respective count, which highlights the prevalence of 4- and 5-star reviews compared to lower scores. Small sample tables of zero-rated reviews are also shown to illustrate the anomaly and offer reasons for removing those rows from the dataset.

We wanted not only to visualize the trust gap (the percentage difference between Amazon's displayed rating and the verified purchaser average) but also to examine its relationship with product price.

We initially considered using a bar chart to illustrate the trust gap and a line chart to display the average price per category. We then further developed our ideas until we arrived at our final chart.

Sketches

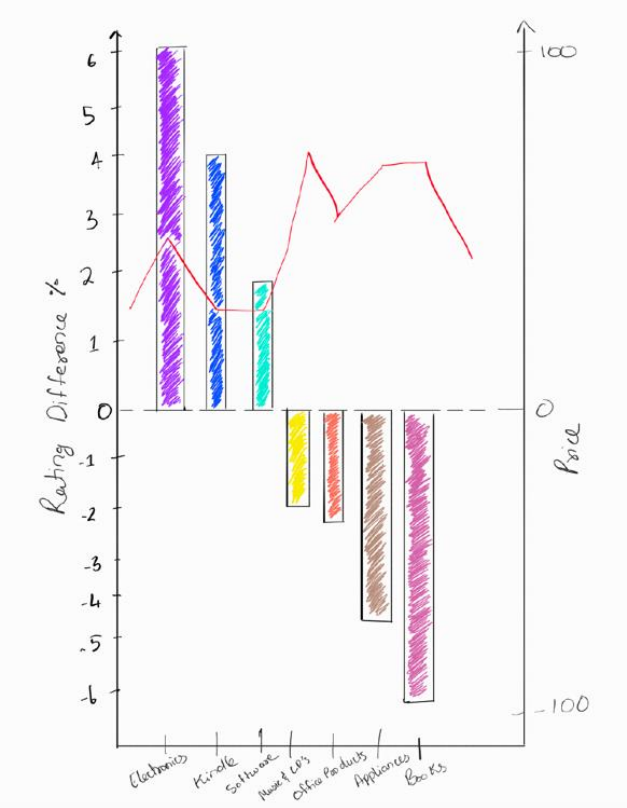

Figure 6 Initial sketch for our visualization

We initially considered using a bar chart to illustrate the trust gap and a line chart to display the average price per category. However, we were not really convinced by the visual aspect of the chart. The combination of the bar and line chart appears rather messy when one column (Rating Inflation) contains both positive and negative values. In contrast, the other column (Price) contains only positive values.

We then wondered if a combination of the bar chart and circles to represent the average price per category would be a better and less messy option.

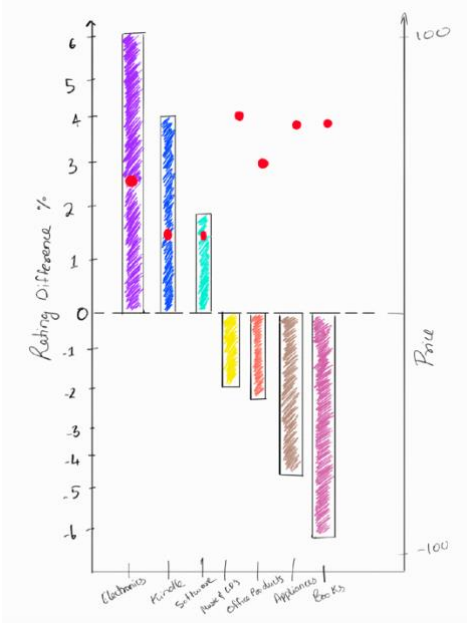

Figure 7 Second sketch

Although this did not look as messy, it did not convey the information we wanted to visualize effortlessly.

After much deliberation, we decided that instead of having the 'Rating Difference %' and 'Price' as the y-column and the 'Categories' on the x-column, we decided to have 'Rating Difference %' on the y-column and 'Price' as the x-column, and then differentiate between the categories by using the colour detail. We also wanted the size of the bubbles to differ based on the number of non-verified reviews that category has. Below is the visualization we decided on.

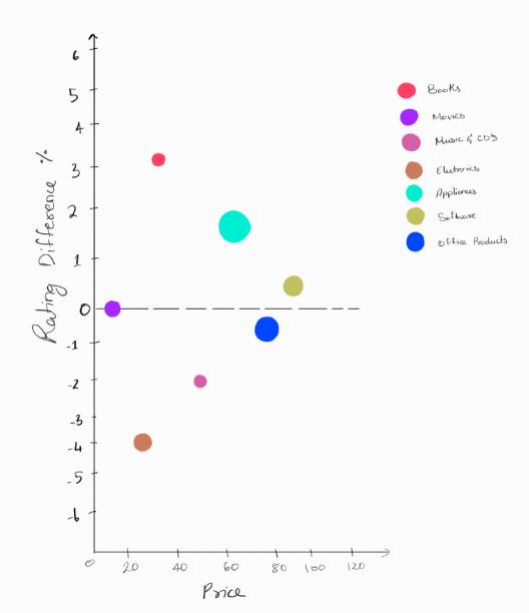

Figure 8 3rd sketch - Scatter Plot

This visualization on sketch looked perfect; however, when implemented on Tableau, it was not the best visualization due to how small some of the bubbles looked.

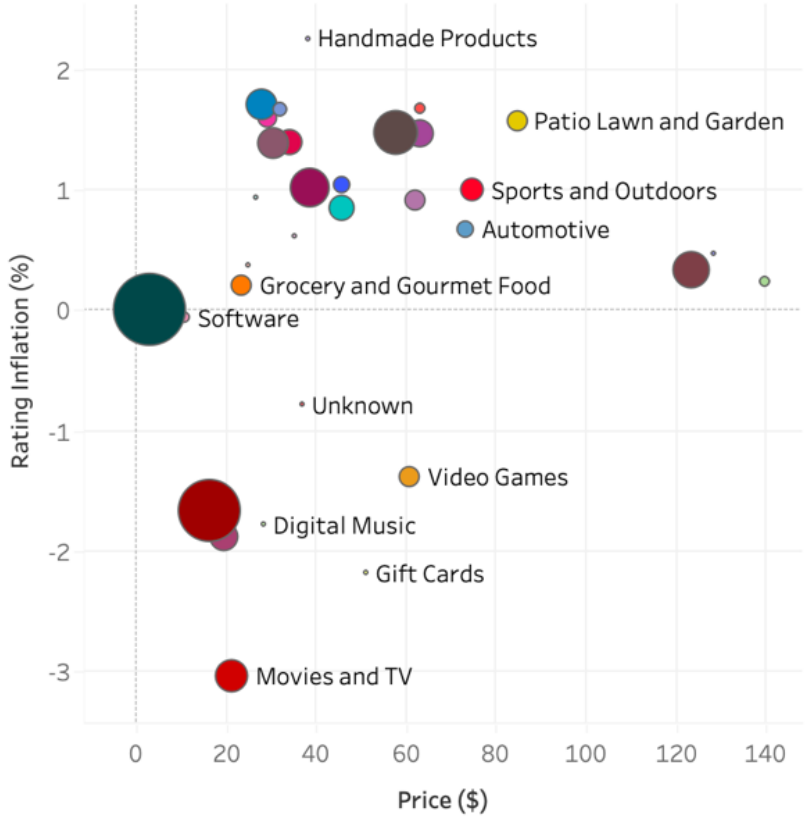

Figure 9 Third sketch as visualized on Tableau

We wanted to show viewers the number of unverified reviews in each category so they could better understand the impact. However, as we can see above, some of the bubbles were too small to be visible. The above visualization could be beneficial if the viewer were viewing it on their phone, as they could then zoom in so that they do not miss anything. However, a strong visualization is easy to view and understand.

Thus, eventually, we decided that instead of having the size of the bubbles vary according to the number of unverified comments in that particular category, we could animate the visualization such that as the number of unverified reviews increases, the categories show up and then leave a ghost trail as and when a new category shows up.

Since we cannot embed a visualization in this document, we have inserted the final frame of our visualization below.

Our Chart

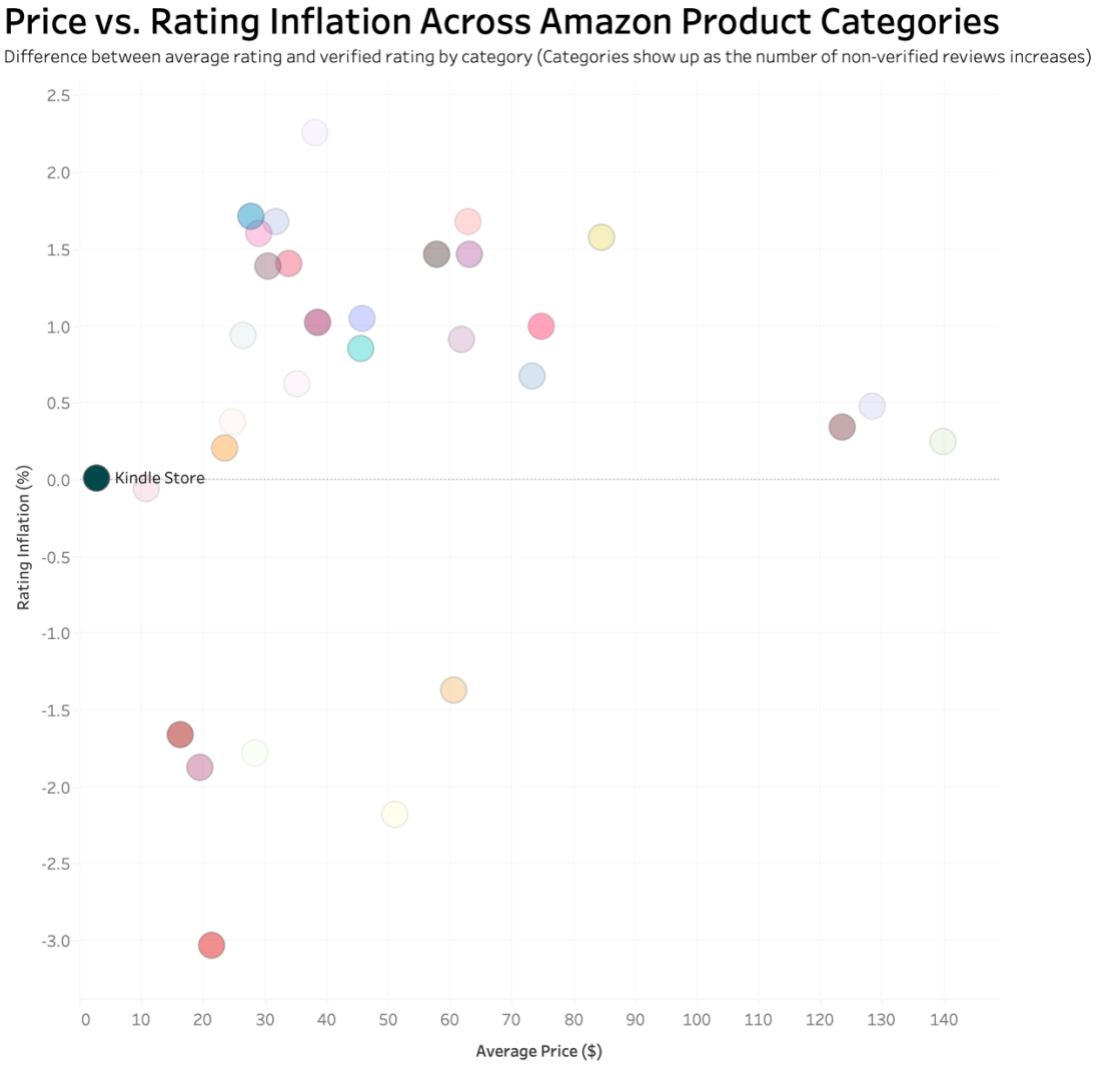

Figure 10 Final Visualization on Tableau

we needed a visualization that could show correlation between two continuous variables (price and trust gap), differentiate between the different product categories and reveal how patterns emerge as non-verified reviews accumulate hence why we chose a scatter plot.

Design Choice

We made the following design choices for our visualization, guided by the Data Visualisation Checklist by Stephanie Evergreen & Ann K. Emery [2]:

1. Grid Lines: 40% opacity to reduce clutter while maintaining spatial reference.
2. Axis Labels: X-axis shows "Average Price ($)" for clarity; Y-axis displays formula to explain positive (overselling) vs. negative (underselling) values.
3. Title/Subtitle: Clear heading and explanatory subtitle communicate purpose immediately.

4. Strategic Colour Gradient Tableau's default colour palette provides fewer than 30 distinct colours after which it repeats the same colours for the categories. More importantly, random colour assignment would create visual chaos during animation. We implemented a custom color strategy:
    a. Sorted categories by total non-verified review count (ascending)
    b. b. Generated hex codes using coolors [3].
    c. c. Manually assigned colors in Tableau: lightest shades to categories with lowest non-verified counts, darkest shades to highest non-verfied counts.

    This ensures that as categories appear sequentially during animation and earlier categories fade into "ghost trails" (via Show History feature), the light-colored ghosts remain visible but recede into the background, while newly-appearing dark-colored categories naturally draw attention. This creates a visual hierarchy aligned with temporal progression

5. Uniform Size: Equal-sized bubbles prevent smaller ones from becoming invisible when faded, prioritizing visibility over encoding additional dimensions.
6. Animation Configuration We used Tableau's Pages shelf to create temporal animation:
    a. Dimension: Sum of non-verified review count.
    b. Show History: Enabled with "All" marks visible, creating ghost trails of previous states.
7. Dynamic Labels: Names appear when current, disappear when ghosted, preventing clutter.
8. Reference Line: Dashed line at Y=0 separates positive inflation (above) from negative (below).

Conclusion

We used PySpark for processing 375GB of Amazon data and Tableau for visualization with animation effects. The main challenge was data scale, addressed through intelligent filtering while maintaining validity. Our light-to-dark color strategy required iteration to ensure ghost trail visibility.

The visualization successfully reveals that Amazon's ratings are generally reliable but vary systematically by price and category, with inflation intensifying as non-verified reviews accumulate.

References

[1] McAuley-Lab, “McAuley-Lab/Amazon-Reviews-2023 · Datasets at Hugging Face,” huggingface.co, Mar. 31, 2024. https://huggingface.co/datasets/McAuley-Lab/Amazon-Reviews-2023

[2] “Data Visualization Checklist.” Available: https://stephanieevergreen.com/wp-content/uploads/2016/10/DataVizChecklist_May2016.pdf

[3] Coolors, “Coolors,” Coolors.co, 2018. https://coolors.co