Amazon Reviews Machine Learning Project (Group L)

Morgan Simmons[1] Rachel Kang[2] Mark Haller[3]

1. Data

All data, reports, and code will be in the provided link below:

https://github.com/yxs4yt/machine_learning_project

Research Model: Predicts at the time a review is posted whether it will receive >5 helpful votes using the Summary data.

1.1. Variables[2]

- *overall*: The overall product rating (from 1-5)
- *verified:* If the review was verified by Amazon
- *reviewTime:* Time of the review (month, date, year format)
- *reviewerID:* Reviewer ID
- *asin:* Product ID
- *style:* A dictionary of the product metadata
- *reviewerName:* Name of the reviewer
- *reviewText:* Text of the review
- *summary:* Summary of the review
- *unixReviewTime:*Time of the review Unix time
- *vote:* How many "helpful" votes the review has
- *image:* List of images the user posts with their review after they have received the product

The *vote* variable is a key variable to the model because it is the outcome of the question. It is a direct signal in the data of how many people gave the product a good review.

.

1.2. Challenges[2]

One of the biggest practical challenges was data size. Several raw files are extremely large, which strains local storage and memory and slows down simple operations (previewing, counting rows, or building features). A second challenge was version control and hosting: GitHub enforces a hard 100 MB limit per file. Large datasets must be tracked with Git Large File Storage (LFS) or stored outside the repository and fetched at runtime. These constraints shaped how we had to pick data. Overall, the key variables are well-defined and aligned with the research question, but scale and hosting limitations were the main challenges in the reading portion for analysis.

Challenges with cleaning hinged on choosing what to keep vs. drop and making the thresholds explicit. We removed near-empty or low-relevance columns and kept only rows with core IDs/timestamps.

The main challenge of preparing our data was making sure the data could answer our question at the time a review is posted. We limited features to what's available, treated missing votes as 0, set a clear helpfulness cutoff (≥5), and used a time-based split in hopes that our model learns from earlier reviews and is tested on later ones which avoids leaks and keeping the prediction context realistic.

1.3. Cleaning the Data[1]

When looking at our data, we came across some issues concerning three main variables. One of these variables was *image*. The concern with this variable was that we didn't foresee this variable being of note for future modelling or examination, as well as many reviews were missing had missing entries for this variable, so we ended up removing it overall from the dataset. The same rationale was used for the *style* variable, as keeping them would increase the computational power needed to traverse and model using the dataset. The other variable that needed to be cleaned was *vote*. As previously stated, vote is a key variable that we're using for our model, but there were issues when it had missing entries. To mitigate this, we transformed all missing entries in the *vote* column to 0. This was because all those missing entries are meant to represent that the review has 0 "helpful" reviews, so we changed it to better represent this and so we can

1.3. Visualizing Preliminary Data[1]

In the codebook, the first graph shows the class balance of reviews, divided into two categories: those with fewer than 5 reviews and those with 5 or more reviews. The tall bar on the left indicates that the vast majority of cases fall into the "<5 Reviews" category, while only a small portion belong to the "≥5 Reviews" category. This imbalance suggests that most reviews in the dataset receive very little engagement, and only a minority achieve higher visibility or interaction. For modeling purposes, this imbalance is important to note because it can bias predictions toward the dominant class unless techniques like resampling or class weighting are applied.

The second graph shows the relationship between text length and helpful votes on a logarithmic scale. Most reviews cluster at shorter lengths (under about 5,000 characters) and receive relatively few helpful votes (often fewer than 10). However, some reviews that are longer do occasionally attract more helpful votes, with a few outliers reaching into the hundreds or thousands. The log scale on the y-axis makes it easier to see this variation, since without it, the small differences among the majority of reviews would be compressed at the bottom of the graph. The pattern suggests that while longer reviews may have a better chance of being judged as helpful, length alone does not guarantee more helpful votes, many long reviews still receive very few.

Together, these graphs provide insight into the nature of the dataset. The class balance graph emphasizes the rarity of widely reviewed items, while the scatterplot highlights that helpfulness is influenced by more than just the length of the review. Both plots indicate that the dataset has skewed distributions, which has implications for data analysis and modeling approaches. This will help give us insight into which patterns we should look at when we start to develop a predictive model.

All superscripts next to headers indicate which author wrote which part.

References

OpenAI. (2025). *ChatGPT* [Large language model]. https://chat.openai.com/

Ni, J., Li, J., & McAuley, J. (2019). *Justifying recommendations using distantly-labeled reviews and fine-grained aspects*. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.