# Rating Prediction for Health and Personal Care Products

Siddhartha Sumant Mysore, Krithika Suwarna

## 1. ABSTRACT

This paper proposes a comprehensive methodology for developing a predictive model to estimate product ratings specifically for Health and Personal Care products on Amazon, utilizing user-generated review data. In this paper, we employ two models to predict the ratings of new user reviews: first, a Singular Value Decomposition (SVD) latent factor model, and second, an Embedding Based Similarity Collaborative Filtering (E-SCF) model

## 2. DATASET

In this paper we use a dataset of Amazon product reviews of health and personal care products. Amazon is a leading multinational E-commerce company. Registered users can purchase products, leave reviews, and rate items sold on the website. The components of a review are a Header - A brief, attention-grabbing summary of the review, Subject - A more in-depth explanation of the user's experience, highlighting likes, dislikes, and any specific features and Star Rating - Users rate the product on a scale of 1 to 5 stars, reflecting their satisfaction (1 being the lowest and 5 the highest). The dataset is particularly useful for rating prediction models, as it includes user ratings and review content.

### 2.1 Dataset Structure Overview

The dataset consists of 494,121 rows representing individual product review submissions, with 8 key attributes (columns) that capture essential information about the reviews. The detailed information of each row of the record is provided in Table 1 below. This table outlines the key attributes that make up each review entry, offering a clear understanding of the dataset's structure and content.

Table 1: Data Formula

| | |
|---|---|
| Rating | Numeric score (1-5) indicating user satisfaction. |
| Title | Brief summary of the reviewer's opinion. |
| Text | Detailed content of the review, explaining the user's experience. |
| ASIN | Unique product identifier. |
| User ID | Unique identifier for the reviewer. |
| Timestamp | The time when the review was posted, epoch format. |
| Helpful Vote | Number of users who found the review helpful. |
| Verified Purchase | Indicates if the review is from a verified purchase. |

In this study, we use 400,000 samples for training, 50,000 samples for validation to test multiple models and identify the best performing one, and the remaining data is reserved for testing and evaluating the model's performance.

## 2.2 Exploratory Data Analysis

In this section, we perform an exploratory data analysis (EDA) on the health and personal care product dataset. EDA is a crucial step in understanding the structure, patterns, and relationships within the dataset before applying any machine learning models.

### 2.2.1 Review Text Analysis

First, we examine the review text by utilizing a word cloud to visually represent the most frequent words. Figure 1 displays the word cloud for the review text.


**Figure 1: Word Cloud for Review Text.**

The word cloud analysis reveals key insights: frequent words like "product," "use," and "love" suggest regular use and satisfaction. Positive sentiments such as "great," "good," and "recommend" indicate overall satisfaction. Terms like "easy," "comfortable," and "soft" highlight appreciation for product comfort, particularly in personal care. Words like "price," "buy," and "brand" reflect consumer behavior, while negative terms like "problem" and "disappointed" point to areas for improvement.

The scatter plot in figure 2 illustrates that there is no strong correlation between review length (word count) and the rating given.

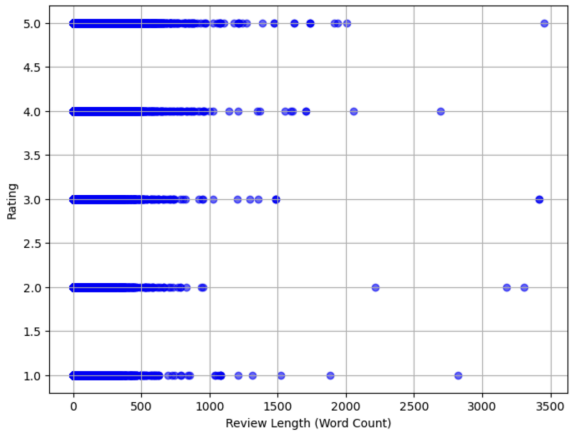


**Figure 2 : Review Length vs Rating**

Most reviews are relatively short (under 1000 words), and both high (5 stars) and moderate (3-4 stars) ratings appear across a similar range of review lengths. While some longer reviews are associated with higher ratings, this is not consistent, as shorter reviews can also receive high ratings. The distribution indicates that review length is not a reliable predictor for ratings, making it unsuitable for use in a linear regression model aimed at predicting ratings.

### 2.2.1 User and Product Analysis

In this analysis, we focus on examining the users and the products they have interacted with.

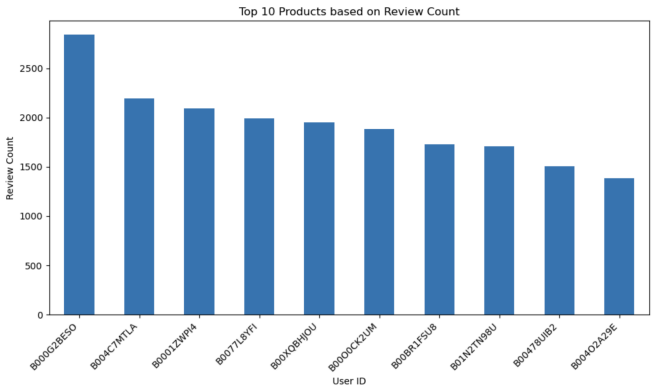


**Figure 3: Graph of top 10 products**

The graph in Figure 3 displays the top 10 products based on review count, with the x-axis representing product IDs and the y-axis showing the number of reviews. The product "B000G2BESO" has the highest review count, exceeding 2,500 reviews, while the other products show varying levels of engagement.

. We see that not all products have the same number of reviews, as indicated by the varying frequencies of reviews for each product. Although the dataset contains a total of 494,121 reviews, the reviews are not evenly distributed across the 62,597 unique products.The number of reviews per product varies, with some products receiving significantly more attention and feedback than others, highlighting the differences in consumer interest across products.

The figure 4 shows the graph of average ratings of the most popular product from 2008 to 2024, highlighting fluctuations over time. The ratings peaked around 2017-2018 but sharply declined after 2020, especially in 2024, indicating a drop in customer satisfaction.

By this, we see that over the years, the type of users has changed, likely influencing the product's ratings. Earlier positive ratings may reflect loyal customers, while recent ones likely come from a more diverse user base with varied experiences.This shift could be due to factors like changes in the product's target audience, market trends, or the introduction of competing products.

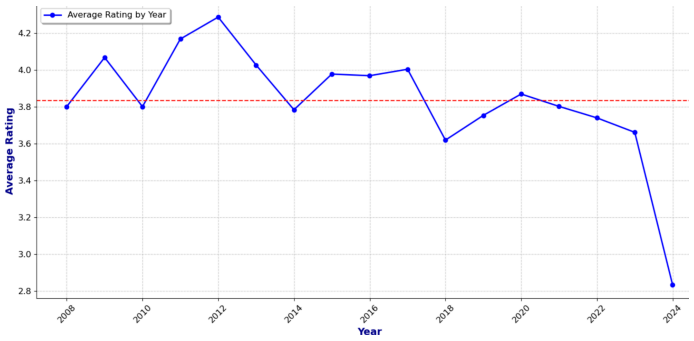


**Figure 4: Change in Ratings over Time**

Due to the notable differences between recent and past reviews, we intend to assign distinct weightages to each group. This approach will ensure that the model accounts for the evolving nature of user feedback, allowing it to better capture current trends and enhance prediction accuracy.

# 3. PREDICTIVE TASK

In this study, we aim to predict ratings for new reviews using two distinct methods. The first method utilizes past reviews to forecast ratings for new ones, while the second method predicts ratings based on the content of the reviews by analyzing the comments and their summaries.

## 3.1 Method 1

This method relies on past reviews by analyzing the historical ratings provided by reviewers for different products.

The objective is to estimate the rating a reviewer would give to a product they have not yet rated, based on their historical ratings. The primary features for this prediction include the overall rating (the score given by the reviewer), reviewerID (a unique identifier for each reviewer), and asin (a unique identifier for each product). This is framed as a collaborative filtering problem, where we utilize the interactions between reviewers and products to predict missing ratings for unseen product-reviewer combinations.

$$u_i^T \cdot v_j \text{ -> } R_{ij}$$

To achieve this, we create a rating matrix, where rows represent reviewers, columns represent products, and the matrix values represent the ratings given by the reviewers. Since each reviewer typically rates only a small subset of products, the matrix is sparse.

## 3.1 Method 2

The goal of this task is to predict the rating for a new review by leveraging the ratings the user has provided for similar products.

This method uses **cosine distances** on embedded vectors to compute the distance between two items in terms of similarity.

***Cosine Distance=1−Cosine Similarity***

The methodology begins with preparing user-item interaction data, organizing it into sequences where each user's interactions are treated as a sentence, enabling the use of embedding models such as Word2Vec to learn dense vector representations for items. These embeddings encode co-occurrence patterns and semantic similarities between items in a continuous vector space.

Once trained, item-item similarities are computed using cosine distances derived from the embeddings. For rating prediction, the model aggregates ratings of similar items that the user has already interacted with, weighting them by their computed similarity to the target item. about the rating a reviewer might give.

This method is particularly useful when working with large amounts of unstructured text data, where traditional collaborative filtering techniques may not be applicable.

# 4. MODEL

## 4.1 Baseline Model

In this study, initially we implemented a baseline model to predict ratings based on the observed preferences of users. We utilized a collaborative filtering approach where the prediction for a user-item pair is derived from the user's historical ratings.

The model was designed in two stages:

**Data Preprocessing**: The data was divided into training and validation sets, with the training set containing the first 400,000 entries. The validation set was split into two parts: one from indices 400,000 to 450,000 and another from 450,000 onwards. User-specific and global averages were computed from the training data for predicting ratings.

**Prediction Generation**: For each user-item pair in the test set, the model predicts a rating using the user's average rating if available; otherwise, it defaults to the global average, which is computed from all ratings in the dataset.

**Evaluation**: Evaluation: The performance of the model was assessed using the Mean Squared Error (MSE), which resulted in a value of 2.4, indicating a relatively high error.

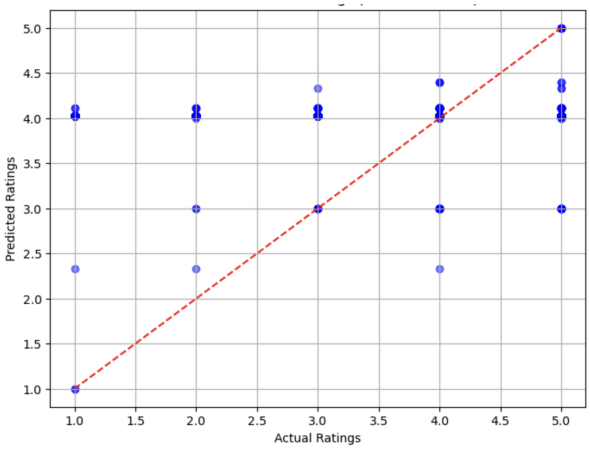


**Figure 4: Predicted vs Actual Rating - Baseline**

The baseline model, using global and user-specific averages, shows significant weaknesses as predicted ratings deviate widely from actual ratings, indicating poor accuracy. The model oversimplifies by not considering individual user preferences or item characteristics. This results in a less personalized approach, making it unsuitable for complex rating patterns. More advanced techniques are needed to better capture user-item interactions.

## 4.2 Singular Value Decomposition (SVD)

In this study, we employ SVD, a powerful matrix factorization technique, to predict ratings for new reviews based on historical user-item interactions.

Specifically, we use the SVD model from the Surprise library, which is designed to uncover hidden patterns in user preferences and product characteristics by decomposing the rating matrix into latent factors. The key parameters used in our implementation include **latent factors**, **epochs**, and **regularization** terms, which control the complexity and the learning process of the model.

The rating prediction task involves predicting the rating a reviewer would give to a product they have not yet rated. In this case, the input to the model is a **rating matrix**, where rows represent **reviewers** (or users) and columns represent **products** (or items). The values in the matrix correspond to the **ratings** (overall score) provided by reviewers for products. The matrix is typically sparse, with each reviewer rating only a subset of the available products. The goal of the SVD model is to fill in the missing entries of this matrix, predicting ratings for unseen product-reviewer pairs.

During the study, our primary challenge was the limitation of computational resources required to run the SVD model. Higher values for **n_factors** allow the model to capture more nuanced relationships in the data, and more **n_epochs** can help in converging to a better solution. However, these improvements require significant computational resources. This also affected the **optimization** process of the SVD model as we were unable to run a **grid search** to optimize hyperparameters due to computational constraints.

## 4.3 Embedding-Based Similarity Collaborative Filtering (E-SCF)

In this study we employ an E-SCF model to predict the ratings for new reviews by organizing user-item interactions into **Per User** and **Per Item** groupings. The model captures relationships between items by organizing the items rated by each user into **sequences**. These sequences are treated like sentences, where the "words" are items, and their co-occurrence represents relationships.

**Word2Vec** is then used to transform each item into a dense vector in a high-dimensional space to capture **latent relationships** between items, such as similarity in user preferences. The rating prediction process involves assessing the **similarity** between the target item and the items previously rated by the user. The predicted rating is then calculated using the **weighted average** of these ratings. If no similar items are found, a global average rating is used as the prediction.

# 5. LITERATURE

The dataset we used is from github (https://amazon-reviews-2023.github.io/) This dataset have been previously used in the paper "*Bridging Language and Items for Retrieval and Recommendation Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, Julian McAuley arXiv. "*

The paper introduces BLAIR (Bridging Language and Items for Retrieval and Recommendation), a series of pre-trained sentence embedding models designed for recommendation scenarios. BLAIR is trained to understand correlations between item metadata and natural language contexts, facilitating improved item retrieval and recommendation tasks. The model is trained on a new large-scale dataset, AMAZON REVIEWS 2023, which includes over 570 million reviews and 48 million items across 33 categories, making it significantly larger than previous datasets.

BLAIR leverages a contrastive learning objective to pair user reviews with item metadata, effectively linking items with natural language. It demonstrates strong performance in multiple recommendation tasks, including sequential recommendation and complex product search, which involves retrieving items based on detailed natural language queries. The study highlights BLAIR's strong ability to generalize across domains and outperform existing models in tasks like product retrieval and recommendation.

# 6. RESULT AND CONCLUSION

## 6.1 Baseline Model

The MSE obtained by the baseline model was **2.4589**.The baseline model serves as a simple starting point for evaluating recommender system performance. While its MSE provides a reference for assessing other more sophisticated models, it lacks personalization and adaptability.

## 6.2 Singular Value Decomposition (SVD)

The matrix factorization model using SVD from the surprise library achieved an MSE of **1.8618.** This represents a significant improvement over the baseline model.

Key parameters used for the SVD model:

- n_factors: 10
- n_epochs: 250
- Learning Rate: 0.001
- Regularization: 0.001

The SVD-based matrix factorization model outperforms the baseline model by effectively capturing latent user and item interactions. Key observations include:

- **Improved Performance**: The lower MSE indicates the model's ability to predict ratings more accurately by identifying underlying patterns in user-item interactions.
- **Scalability**: While the training time increased due to the higher complexity, the improvement in accuracy justifies the added computational cost.
- **Model Tuning**: The parameters chosen (e.g., n_factors and n_epochs) play a crucial role in achieving this improvement. Further hyperparameter tuning could potentially lower the MSE even more.

## 6.3 Embedding-Based Similarity Collaborative Filtering (E-SCF)

The E-SCF model achieved an MSE of **2.20422.**

**Key features of the model:**

Word2Vec embedding: The embedded vector was created using the following parameters:

- **min_count=10**: Items with fewer than 10 instances were ignored.
- vector_size=100: 100-dimensional embeddings were used.
- **window=3:** A context window of size 3 was applied.
- **sg=1**: The skip-gram architecture was used for training.

The Word2Vec-based approach demonstrates moderate improvement over the baseline model but does not outperform the SVD model. The strength of the model is that it captures item similarity based on learned embeddings, which is beneficial for sparse datasets. Its weakness is that it relies heavily on Relies heavily on the quality of the Word2Vec embeddings, which depend on sufficient data and appropriate parameter tuning.