Amazon Product Recommendation System

Low Code Version

August 3rd 2025

Aída Fuentes

Contents / Agenda

- Business Problem and Data Overview
- Exploratory Data Analysis
- Rank Based Model
- User-User Similarity-based Model
- Item-Item Similarity-based Model
- Matrix Factorization based Model
- Conclusion and Recommendations

Data Analysis Business Problem and Data Overview

- **Overview of the business problem and what we are trying to solve.**

In today's digital landscape, consumers are overwhelmed with an abundance of choices. The exponential growth of online content and product catalogs has led to decision fatigue, where users often abandon purchases due to lack of direction. To address this, leading e-commerce platforms such as Amazon, Walmart, and Etsy rely on intelligent recommendation systems to enhance user experience and drive conversions.

As part of Amazon's Data Science team, we were tasked with developing a recommendation system capable of suggesting relevant products to users based on their past ratings. Using a dataset of Amazon product reviews, we explored multiple modeling approaches to extract actionable insights and generate personalized recommendations.

Our analysis began with a thorough exploration of the dataset, followed by the implementation and evaluation of several recommendation algorithms:

- A **popularity-based baseline model**, effective for new users.
- **User-User and Item-Item collaborative filtering**, leveraging behavioral similarities.
- And a **Matrix Factorization model (SVD)**, which delivered the most accurate and personalized predictions.

Through careful evaluation using RMSE, precision, recall, and F1-score, we identified the trade-offs between simplicity, accuracy, and scalability. The final SVD model proved to be the most effective, outperforming traditional models by capturing latent patterns in user behavior.

This report outlines the methodology, key findings, and model comparisons, and concludes with business-oriented recommendations for deploying a scalable and effective product recommendation system at Amazon.

- **Overview of the dataset.**

Exploratory Data Analysis

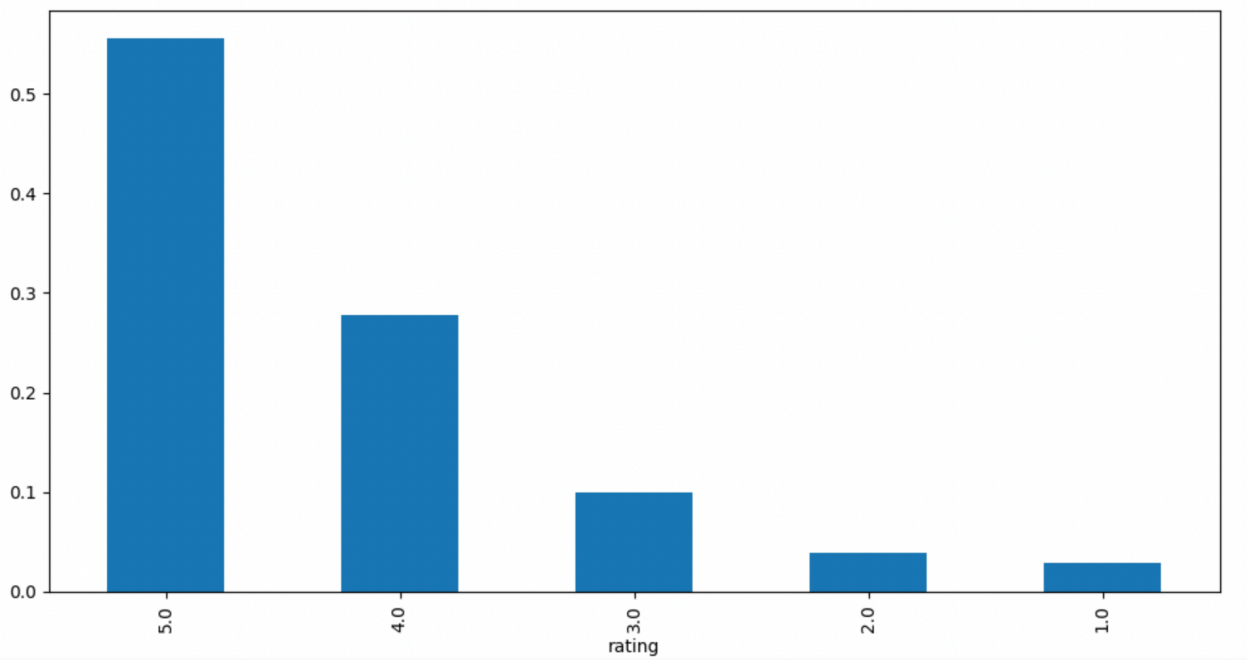
The dataset consists of **65,290 user-product interactions**, capturing product ratings on a scale from **1 to 5**. It includes **1,540 unique users** and over 5,689 **unique products**.

A preliminary data quality assessment confirmed that there are **no missing values—all three columns (user_id, prod_id, and rating) have 0% null values**, ensuring the dataset is fully complete and ready for modeling.

Descriptive statistics revealed the following insights:

	rating
count	65290.000000
mean	4.294808
std	0.988915
min	1.000000
25%	4.000000
50%	5.000000
75%	5.000000
max	5.000000

- The **average rating is 4.29**, indicating that most products are rated positively.
- Ratings **range from 1 to 5**, showing that users do occasionally give low scores.
- The **standard deviation is 0.99**, suggesting moderate variability—some users rate products significantly higher or lower than the average.



- The **median rating is 5**, meaning that at least half of all ratings are at the maximum score.

This highly skewed distribution points to a likely **self-selection bias**, where users are more inclined to leave reviews for products they liked. Moreover, a large portion of products received very few ratings, while a smaller group accumulated high volumes of feedback.

To account for this imbalance and ensure robust results, later models filtered products based on **minimum interaction thresholds**.

This exploratory analysis provided essential context for modeling decisions, revealing key characteristics of user behavior, rating dynamics, and data sparsity.

Rank Based Model

Model 1: Rank-Based Recommendation Intuition

- **Input:** Historical data on product ratings and sales.
- **Process:** Calculates the average rating or most frequently purchased products.
- **Output:** Recommends top-rated or most popular items to all users.
- Pros: Easy to implement
- Cons: Not personalized

Use for: New users with no interaction history (e.g., homepage “Top Picks”)

As a baseline model, we implemented a rank-based recommendation system that recommends the top-rated products to all users, regardless of their individual preferences. This approach ranks products based on their average rating and the number of times they have been rated, ensuring that highly-rated and widely-reviewed products are prioritized.

We generated two sets of top-5 product recommendations:

- **With at least 50 user interactions:**
    - 'B001TH7GUU', 'B003ES5ZUU', 'B0019EHU8G', 'B006W8U2MU', 'B000QUUFRW'
- **With at least 100 user interactions:**
    - 'B003ES5ZUU', 'B000N99BBC', 'B007WTAJTO', 'B002V88HFE', 'B004CLYEDC'

By applying interaction thresholds (e.g., 50 or 100), we ensure that the recommendations are not biased by a small number of high ratings, thereby increasing trustworthiness. However, this model does not personalize results and will recommend the same products to every user.

Key Findings:

- The top 5 recommended products all had average ratings close to 5 and had been rated by a large number of users.
- This model is particularly useful for cold-start scenarios, such as new users with no prior activity, since it does not rely on personalization.
- However, its main limitation is that all users receive the same set of recommendations, which may not align with individual interests or preferences.

User-User Similarity-based Model

User-to-User Collaborative Filtering Intuition:

- **Input:** Users' past ratings and preferences.
- **Process:** Finds users with similar behavior ("neighbors") and recommends items they liked.
- **Output:** Personalized recommendations based on similar users.
- Pros: Personalized
- Cons: Requires sufficient user overlap

Use for: Logged-in users with sufficient rating history

Model 2: Collaborative Filtering Recommendation System

This model leverages user-user similarity to recommend products. It analyzes historical interactions and ratings to find users with similar preferences and predicts ratings for products that a user has not yet seen, based on what similar users liked.

In this model, we used K-Nearest Neighbors (KNNBasic) to identify users with similar rating behaviors and recommend products that similar users liked. The core idea is that users who have rated products similarly in the past are likely to agree again in the future.

To measure similarity, we applied the cosine similarity metric on user rating vectors. Once a group of similar users (neighbors) was identified for a given user, the algorithm recommended items those neighbors liked but the target user had not yet interacted with.

Evaluation Metrics:

- **RMSE:** 1.0012
- **Precision:** 0.855
- **Recall:** 0.858
- **F1-score:** 0.856

Prediction Examples:

- For user **A3LDPF5FMB782Z**, who had previously rated product **1400501466** with a score of **5**, the model predicted a rating of **3.40**. While slightly lower than the actual rating, this is acceptable given the RMSE (~1.0), showing the model behaves consistently.
- For user **A2U0HALGF2X7TQ**, who had **not rated** product **1400501466**, the model predicted a rating of **5.0**. This strong recommendation aligns with the fact that **similar users rated this product very highly**, indicating the model is able to leverage neighborhood information effectively. However, this prediction was based on **only one similar user** (actual_k = 1), which limits its reliability. Although the result appears promising, the low number of neighbors suggests that this recommendation may be **less robust**, **and should be interpreted with caution.**

Key Observations:

- The user-user collaborative filtering model (KNN with cosine similarity) performed well with a precision of 85.5% and a recall of 85.8%, resulting in an F1 score of 0.856. This suggests that the model is effective at recommending relevant products, although there is still some room for reducing the prediction error (RMSE = 1.00).
- This test demonstrates that the model can effectively approximate user preferences, which is essential for generating personalized product recommendations. As more data is collected, the accuracy of the predictions is expected to improve.
- The user-to-user collaborative filtering model is built only on explicit ratings, which are often biased towards positive reviews. Since most users tend to rate only when they are satisfied, the model frequently predicts high ratings (e.g., close to 5), which might not

accurately reflect true user preferences. Incorporating implicit feedback or more diverse rating data would enhance the accuracy and robustness of the recommendations.

- To enhance the accuracy of the similarity-based recommendation system, we fine-tuned its hyperparameters (number of neighbors, minimum required neighbors, and similarity metrics). This process aims to reduce prediction errors and ensure the system is better aligned with user behavior patterns.

User-User Collaborative Filtering (Optimized)

This model leverages user-user similarity to recommend products. It analyzes historical interactions and ratings to find users with similar preferences and predicts ratings for products that a user has not yet seen, based on what similar users liked.

Approach and Optimization:

- We implemented a **KNN-based collaborative filtering algorithm** using cosine and MSD similarity measures.
- A **grid search with cross-validation** was used to tune the hyperparameters:
    - **k = 50** (number of neighbors).
    - **min_k = 6** (minimum neighbors to consider a prediction valid).
    - **Similarity measure = MSD**.
- The model was retrained using these optimal hyperparameters, improving its overall accuracy.

Key Metrics:

- **RMSE:** 0.9530 (improved from 1.0012 before tuning).
- **Precision:** 84.7% (percentage of recommended items that were relevant).
- **Recall:** 89.3% (percentage of relevant items successfully recommended).
- **F1 Score:** 0.869 (balanced measure of precision and recall).

Business Value:

- Unlike popularity-based recommendations, this model offers true personalization, which enhances user satisfaction and increases the likelihood of engagement and repeat visits.

- By uncovering niche products liked by similar users, it can promote lesser-known items and help users discover more relevant content.
- As the system collects more interaction data, its performance is expected to improve further, making it a valuable tool for customer retention and conversion.
- Using the similarity-based collaborative filtering model, we generated the top 5 personalized product recommendations for user **A3LDPF5FMB782Z**. The predicted rating for each recommended product was 5, indicating a high level of confidence from the model based on the preferences of similar users. This result highlights the strength of the collaborative filtering approach in identifying **relevant products that the user has not yet interacted with**.

Limitations Observed in Optimized Model

Despite the improvements in evaluation metrics after tuning, the optimized user-user collaborative filtering model failed to produce valid predictions for some cases. For instance, when attempting to predict ratings for both known and unknown interactions involving product **1400501466**, the model returned 'was_impossible': True due to 'Not enough neighbors'. This indicates that, even with optimized hyperparameters, the model still suffers from **sparsity issues**, especially when the data for certain user-item combinations is limited. These scenarios highlight the importance of data density in neighborhood-based models, and suggest that combining approaches (e.g., hybrid or model-based techniques) may be necessary to ensure robust recommendations across the entire catalog.

Item-Item Similarity-based Model

Model 3: Item-to-Item Collaborative Filtering Intuition

- **Input:** Products the user has already rated or interacted with.
- **Process:** Identifies similar products based on co-rating behavior across users.
- **Output:** Recommends similar items (“You may also like…”).
- **Pros**: Great for cross-selling
- **Cons**: Less effective for new or niche items

Use for: Product detail pages and cart suggestions

Item-to-Item Collaborative Filtering (Baseline)

This model focuses on identifying **similar products** rather than similar users. The core assumption is that if a user liked a product, they are likely to enjoy other products that are similar in terms of rating patterns across the user base.

Instead of comparing users, we compare **items** based on how they were rated by different users. Using **K-Nearest Neighbors (KNNBasic)** with **cosine similarity**, we computed item-item similarities and generated recommendations accordingly.

Evaluation Metrics (Baseline Model):

- **RMSE:** 0.9950
- **Precision:** 0.838
- **Recall:** 0.845
- **F1-score:** 0.841

Prediction Example:

- For user **A3LDPF5FMB782Z**, who had rated product **1400501466** with a 5, the model predicted **4.27**. This is reasonably close to the true rating and suggests that the model can capture item similarities effectively.
- For user **A2UOHALGF2X77Q**, who had not interacted with product **1400501466**, the model predicted a rating of **4.00** based on a **single item-based neighbor** (`actual_k = 1`). While less robust than predictions based on larger neighbor sets, this result still shows the model's ability to generalize even in sparse data conditions.

Observations:

- The item-to-item approach performed similarly to the user-user model in terms of RMSE, but had slightly lower precision and recall.
- Because this model relies on finding **similar items**, it is generally more stable for users with limited interaction history.

- However, if an item has few ratings, the model may struggle to compute meaningful similarity scores, leading to default or missing predictions.
- In user-user, we observed that the model **defaulted to the global mean (≈4.29)** when it couldn't find enough similar items for a prediction — a known limitation of KNN when data is sparse.

Business Value:

While not personalized to the user, this model can **effectively recommend alternatives or complements** to previously rated products. It is especially valuable for:

- Recommending similar products on product pages ("Customers who bought this also bought…"),
- Enhancing product discovery when users are browsing specific items,
- Mitigating the impact of limited user data by focusing on item behavior.

Item-Item Collaborative Filtering (Optimized Version)

In this model, we use item-to-item collaborative filtering, which recommends products by analyzing the similarity between items based on how users have rated them. The assumption is that if a user liked item A, they are likely to enjoy similar items B, C, or D.

Approach and Optimization:

We implemented the model using the KNNBasic algorithm with the user_based=False parameter to focus on item similarities. To improve prediction accuracy, we tuned the model using grid search cross-validation over the following hyperparameters:

- **k** (number of neighbors): [10, 20, 30]
- **min_k** (minimum neighbors required to make a prediction): [3, 6, 9]
- **Similarity measures**: 'msd' and 'cosine'

After tuning, the optimal values were:

- k = 20
- min_k = 6
- similarity = msd

The model was retrained with these optimal parameters, resulting in better performance.

Evaluation Metrics:

- **RMSE**: 0.9578
- **Precision**: 0.839
- **Recall**: 0.88
- **F1 Score**: 0.859

These results show improved predictive performance compared to the baseline item-to-item model.

Prediction Examples:

- For user **A3LDPF5FMB782Z**, who rated product **1400501466** with a 5, the optimized model predicted **4.71**, closer to the actual value than previous models.
- For user **A2UOHALGF2X77Q**, who had not rated the product, the model was **unable to make a prediction** due to a lack of sufficient neighbors (actual_k < min_k). The system returned an error indicating **'Not enough neighbors'**, which highlights one limitation of item-based collaborative filtering: it may struggle in sparse data scenarios.

Key Observations:

- The optimized item-to-item model performs well when enough data is available, generating **accurate, personalized recommendations**.
- In cold-start or sparse contexts, the model fails to predict, suggesting that additional strategies (e.g., hybrid models or metadata-based recommendations) may be necessary to improve coverage.

Business Implication:

This model is suitable for **recommending similar products based on a user's past interactions**, which aligns with how companies like **Amazon** surface related items ("Customers who bought this also bought..."). Its strength lies in promoting relevant alternatives and enhancing the discovery of new items, which can **increase cross-selling and time-on-site**.

Limitations Observed in Optimized Model

The similarity-based recommendation system generated the same predicted rating **(4.292024)** for all top-5 product recommendations for user **A1A5KUIIIHFF4U**. This indicates that the model could not find sufficient neighboring items with relevant interactions and therefore defaulted to the average rating. This highlights a common limitation of similarity-based models in sparse datasets

Matrix Factorization based Model

Model 4: Matrix Factorization (SVD)

- **Input:** Entire user-product interaction matrix.
- **Process:** Learns hidden patterns and latent features to predict ratings.
- **Output:** Highly accurate personalized recommendations, even for sparse profiles.
- **Pros**: High accuracy, scalable for large datasets
- **Cons:** Less interpretable than neighborhood models

Use for: Core recommendation engine for active users

Model 3: Matrix Factorization with SVD

Matrix factorization is a powerful recommendation approach that reduces the dimensionality of the user-item interaction matrix to uncover **latent features** that explain observed ratings. It is especially effective in handling **sparse datasets** and is widely used in production environments, including by Amazon and Netflix.

We used the **Singular Value Decomposition (SVD)** algorithm from the Surprise library to decompose the user-item matrix and predict ratings for unrated products. The model was trained with a fixed `random_state = 1` to ensure reproducibility.

Evaluation Metrics:

- **RMSE:** 0.8882
- **Precision:** 0.853

- **Recall:** 0.88
- **F1-score:** 0.866

This model outperformed both collaborative filtering models in **RMSE and F1-score**, indicating it makes more accurate predictions and better balances precision and recall.

Prediction Examples:

- For user **A3LDPF5FMB782Z**, who had rated product **1400501466** with a 5, the model predicted **4.08** — relatively close, and more accurate than the user-user model.
- For **A2UOHALGF2X77Q**, who had not rated that product, the model predicted **4.16**, showing that matrix factorization can provide solid predictions **even for new users** without direct neighbors.

These examples highlight that SVD can generalize better and avoid the 'Not enough neighbors' limitation seen in KNN-based models.

Key Observations:

- Matrix factorization captured **underlying patterns in the data**, leading to improved performance across all metrics.
- It is especially useful in **sparse environments** where neighbor-based models fail due to insufficient overlap.
- While it is less interpretable than user-user or item-item methods, its predictive power and scalability make it an ideal choice for large-scale recommendation systems.

Business Value:

This model offers highly accurate recommendations across a broader range of users and products. It supports **personalization at scale** and can boost customer engagement by delivering relevant suggestions, even when little direct interaction data is available. For a company like Amazon, SVD's ability to handle massive datasets while maintaining performance is a significant strategic advantage.

Conclusion and Recommendations

Model	RMSE	Precision	Recall	F1- Score	Business Value
Rank- Based	N/A	N/A	N/A	N/A	Simple to implement; useful as a baseline but lacks personalization.
User-User Collaborative Filtering	1.0012 -> 0.9530	0.855 -> 0.893	0.858 -> 0.869	0.856 -> 0.869	Personalized but suffers in sparse data; limited if few similar users exist.
Item- Item Collaborative Filtering	0.9950 -> 0.95578	0.838 -> 0.839	0.845 -> 0.88	0.841->0.859	Effective for "similar items"; performs well if item ratings are dense.
Matrix Factorization (SVD)	0.8882	0.853	0.88	0.866	Best overall accuracy; scalable and suitable for large e-commerce platforms.

- The **Matrix Factorization (SVD)** model outperformed all others in terms of **prediction accuracy and robustness**, making it the best candidate for implementation in a large-scale setting like Amazon. It is especially suitable for **cold-start problems** and **sparse data** environments.
- **User-user** and **item-item collaborative filtering** models are intuitive and interpretable. When sufficient user or item similarity exists, they perform well — especially after **hyperparameter tuning** — but they may fail to make predictions in low-data scenarios.
- The **popularity-based** approach, while simplistic, is a useful **fallback option** for first-time users or when no interaction data is available. It ensures that recommendations are always available, albeit not personalized.

Business Recommendation

We recommend implementing a **hybrid system**, combining:

- **Matrix Factorization (SVD)** for general recommendations across the platform.

- **Item-based filtering** for suggesting similar products on product detail pages ("Customers also bought...").
- **Popularity-based model** for onboarding new users or visitors with no prior activity.

This blended approach will ensure **coverage, accuracy, and user satisfaction**, leading to increased **engagement, retention, and sales conversion**.

Appendix

• User-User Optimized Prediction

Predicting top 5 products for userId = "A3LDPF5FMB782Z" with similarity based recommendation system

```
# Making top 5 recommendations for user_id "A3LDPF5FMB782Z" with a similarity-based recommendation engine
recommendations = get_recommendations(df_final, "A3LDPF5FMB782Z", 5, sim_user_user_optimized)
```

```
# Building the dataframe for above recommendations with columns "prod_id" and "predicted_ratings"
pd.DataFrame(recommendations, columns = ['prod_id', 'predicted_ratings'])
```

	prod_id	predicted_ratings
0	B000067RT6	5
1	B000BQ7GW8	5
2	B001TH7GUU	5
3	B004RORMF6	5
4	B005ES0YYA	5

Write your observations here: Using the similarity-based collaborative filtering model, we generated the top 5 personalized product recommendations for user A3LDPF5FMB782Z. The predicted rating for each recommended product was 5, indicating a high level of confidence from the model based on the preferences of similar users. This result highlights the strength of the collaborative filtering approach in identifying relevant products that the user has not yet interacted with.

• Item-Item Optimized Prediction

Predicting top 5 products for userId = "A1A5KUIIIHFF4U" with similarity based recommendation system.

```
# Making top 5 recommendations for user_id A1A5KUIIIHFF4U with similarity-based recommendation engine.
recommendations = get_recommendations(df_final, "A1A5KUIIIHFF4U", 5, sim_item_item_optimized)
```

```
# Building the dataframe for above recommendations with columns "prod_id" and "predicted_ratings"
pd.DataFrame(recommendations, columns = ['prod_id', 'predicted_ratings'])
```

	prod_id	predicted_ratings
0	1400532655	4.292024
1	1400599997	4.292024
2	9983891212	4.292024
3	B00000DM9W	4.292024
4	B00000J1V5	4.292024

Write your observations here: The similarity-based recommendation system generated the same predicted rating (4.292024) for all top-5 product recommendations for user A1A5KUIIIHFF4U. This indicates that the model could not find sufficient neighboring items with relevant interactions and therefore defaulted to the average rating. This highlights a common limitation of similarity-based models in sparse datasets.