Amazon Product Recommendation System

ML/Data Science Program

October 2023

Presentation By: Ayomikun C. Adeniran

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

Business Problem and Data Overview

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

Today, information is growing exponentially with volume, velocity and variety throughout the globe. This has lead to information overload, and too many choices for the consumer of any business. It represents a real dilemma for these consumers and they often turn to denial. Recommender Systems are one of the best tools that help recommending products to consumers while they are browsing online. Providing personalized recommendations which is most relevant for the user is what's most likely to keep them engaged and help business.

- First, let's see how the data looks. Below is a table consisting of the first 5 rows.

	user_id	prod_id	rating
1310	A3LDPF5FMB782Z	1400501466	5.0
1322	A1A5KUIIIHFF4U	1400501466	1.0
1335	A2XIOXRRYX0KZY	1400501466	3.0
1451	AW3LX47IHPFRL	1400501466	5.0
1456	A1E3OB6QMBKRYZ	1400501466	1.0

- `No of rows: 65290`
  `No of columns: 3`

- The dataset has 65,290 rows and 3 columns.

Data Overview

There is no missing value. The source of the data is high-level so the data seems to be very clean.

```
Summary
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 given to products was 4.29/5.00 which means majority of customers were satisfied with the products they purchased.
•The median rating was 5/5 and the 25th percentile was 4/5 which means that 75% of products received a very high rating of 4 or 5.

Business Problem - Objective

As a Data Science Manager at Amazon, we have been given the task of building a recommendation system to recommend products to customers based on their previous ratings for other products. We have a collection of labeled data of Amazon reviews of products.

The goal is:

- to extract meaningful insights from the data, and
- build a recommendation system that helps in recommending products to online consumers.

Business Problem and Data Overview

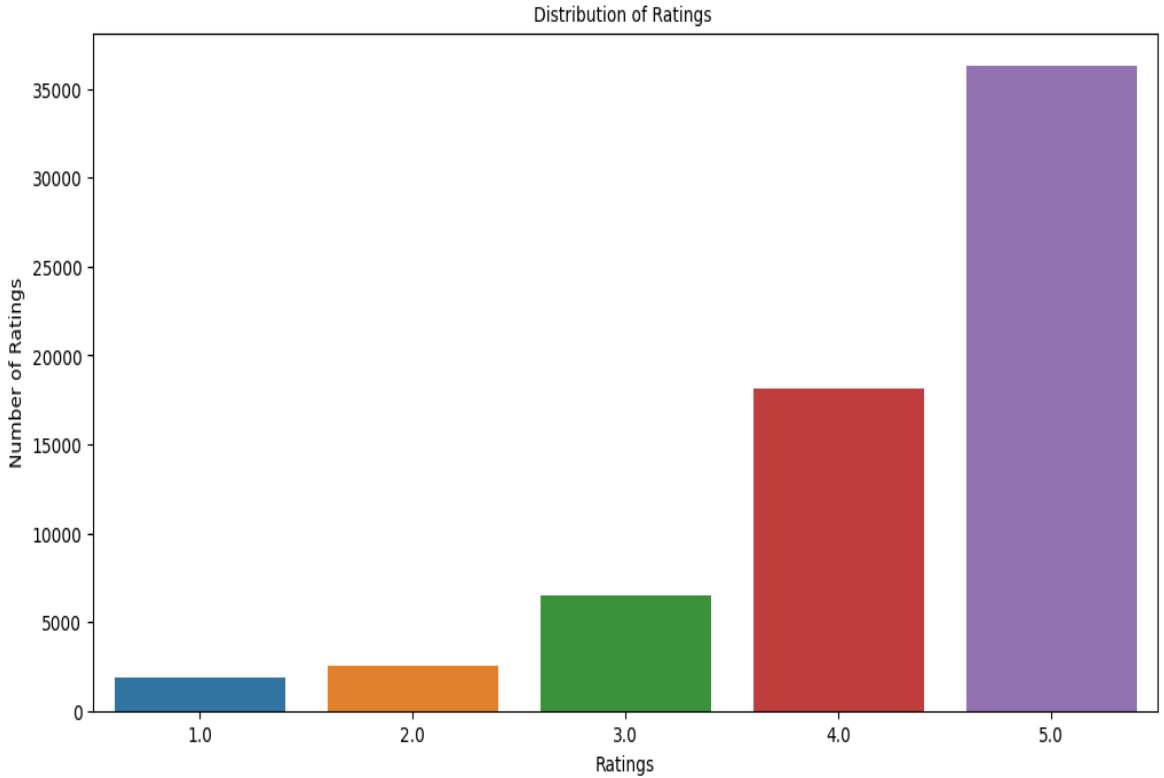
- Overview of the dataset.

The Amazon dataset contains the following attributes:
•**userId:** Every user identified with a unique id
•**productId:** Every product identified with a unique id
•**Rating:** The rating of the corresponding product by the corresponding user
•**timestamp:** Time of the rating. We **will not use this column** to solve the current problem.

The dataset is very large and has 7,824,482 observations, it is not computationally possible to build a model using this. Moreover, many users have only rated a few products and also some products are rated by very few users. Hence, we can reduce the dataset by considering certain logical assumptions:

- Here, we will consider users who have given at least 50 ratings,
- We also consider products that have at least 5 ratings, as when people shop online, they prefer to have some number of ratings of a product.

Exploratory Data Analysis



- Majority of products (greater than 50,000) received a rating of 4 or 5.
- Only about 12,500 products received a rating of 3 or less.
- So, we can use 3.5 as our threshold value.

Exploratory Data Analysis

- The number of observations in the final data = 65290
- Number of unique USERS in Raw data = 1540
- Number of unique ITEMS in Raw data = 5689

Top 10 users based on the number of ratings

user_id	rating
ADLVFFE4VBT8	295
A3OXHLG6DIBRW8	230
A1ODOGXEYECQQ8	217
A36K2N527TXXJN	212
A25C2M3QF9G7OQ	203
A680RUE1FDO8B	196
A22CW0ZHY3NJH8	193
A1UQBFCERIP7VJ	193
AWPODHOB4GFWL	184
A3LGT6UZL99IW1	179

Remark: The highest number of **ratings by a user is 295** which is far from the actual number of products present in the data. We can build a recommendation system to recommend products to users which they have not interacted with.

Model 1: Rank-Based

- To build the Rank-based Model, we take the following steps:

- Create the Final rating DataFrame
- Next, we write the code to create the function top_n_products
- Recommend top 5 products with 50 minimum interactions based on popularity
- Recommend top 5 products with 100 minimum interactions based on popularity

Rank-based Model

- Final Rating Dataframe

prod_id	avg_rating	rating_count
B00LGQ6HL8	5.0	5
B003DZJQQI	5.0	14
B005FDXF2C	5.0	7
B00I6CVPVC	5.0	7
B00B9KOCYA	5.0	8

Rank-based Model (contd)

- Top 5 products with 50 minimum interactions (based on popularity) recommended:

1. 'B001TH7GUU'
2. 'B003ES5ZUU'
3. 'B0019EHU8G'
4. 'B006W8U2MU'
5. 'B000QUUFRW'

- Top 5 products with 100 minimum interactions (based on popularity) recommended:

1. 'B003ES5ZUU'
2. 'B000N99BBC'
3. 'B007WTAJTO'
4. 'B002V88HFE'
5. 'B004CLYEDC'

Similarity-based Models

- We build **similarity-based recommendation systems** using cosine similarity and using **KNN to find similar users** which are the nearest neighbor to the given user.
- We used a new library, called surprise, to build the models.

Some useful terminology:

- **Precision@k** - the **fraction of recommended items that are relevant in top k predictions**. The value of k is the number of recommendations to be provided to the user.
- **Recall@k** - the **fraction of relevant items that are recommended to the user in top k predictions**.
- **F1-score@k** - the **harmonic mean** of Precision@k and Recall@k. When precision@k and recall@k both seem to be important**,** then it is useful to use this metric because it is representative of both of them.
- To compute precision and recall, a **threshold of 3.5** and **k=10** are used**.**

Model 2: User-User Similarity-based Model

Observations about the model performance (the model with default parameters)

```
RMSE: 1.0012 Precision: 0.855 Recall: 0.858 F_1 score: 0.856
```

- We observe that the baseline model has RMSE=1.001 on the test set.
- We are getting a recall of ~0.86, which means out of all the relevant products, 86% are recommended.
- We are getting a precision of ~ 0.86, which means out of all the recommended books, 86% are relevant.
- Here F_1 score of the baseline model is ~0.86. It indicates that most recommended products were relevant and that most relevant products were recommended.
- We can try to improve the performance by using GridSearchCV to tune different hyperparameters of the algorithm.

Model 2: User-User Similarity-based Model

Observations about the model performance (after hyperparameter tuning)

```
RMSE: 0.9553 Precision: 0.855 Recall: 0.885 F_1 score: 0.87
```

- After tuning the hyperparameter values,
- the root mean squared error has decreased to 0.955.
- precision stayed roughly the same while recall has increased to ~88.5%.
- this means our F1-score has also increased a bit to 87%.
- So, the model performed slightly better after tuning.

Model 2: User-User Similarity-based Model

Before Tuning		After Tuning	
User ID: **A3LDPF5FMB782Z**	r_ui = 5.00 est = 3.40	User ID: **A3LDPF5FMB782Z**	r_ui = 5.00 est = 3.40
Product ID: **1400501466**		**Product ID:** **1400501466**	
User ID: **A34BZM6S9L7QI4**	r_ui = None est = 4.29 for the	User ID: **A34BZM6S9L7QI4**	r_ui = None est = 4.29
Product ID: **1400501466**		**Product ID:** **1400501466**	

Observations about the model performance:

The predicted ratings did not change for each of these two users from the predictions given by the initial model.

Model 2: User-User Similarity-based Model

- **Identifying similar Users to a given User (nearest neighbors)**

the 5 most similar users to the first user are:

```
[6, 7, 17, 26, 32]
```

- **Predicting top 5 products for user "A3LDPF5FMB782Z" with Model 2**

The top 5 product recommendations (each with a predicted rating of 5) for user_id "A3LDPF5FMB782Z" using a similarity-based recommendation engine are:

- B00005LENO
- B000067RT6
- B00006HSML
- B00006I53X
- B00006I5J7

Model 3: Item-Item Similarity-based Model

```
RMSE: 0.9950 Precision: 0.838 Recall: 0.845 F_1 score: 0.841
```

- We observe that the item-item similarity-based collaborative filtering model has RMSE=0.9950 on the test set.
- We are getting a recall of ~0.84, which means out of all the relevant products, 84% are recommended.
- We are getting a precision of ~ 0.84, which means out of all the recommended books, 84% are relevant.
- The F_1 score of the baseline model is ~0.84. It indicates that most recommended products were relevant and that most relevant products were recommended.
- We can try to improve the performance by using GridSearchCV to tune different hyperparameters of the algorithm.

Model 3: Item-Item Similarity-based Model

```
RMSE: 0.9576 Precision: 0.839 Recall: 0.88 F_1 score: 0.859
```

After tuning the hyperparameter values,

- the root mean squared error has decreased to 0.958.
- precision stayed roughly the same while recall has increased to ~88%.
- this means our F1-score has also increased a bit to 86%.
- So the model performed better after tuning.

Model 3: Item-Item Similarity-based Model

Before Tuning		After Tuning	
User ID: **A3LDPF5FMB782Z**	r_ui = 5.00 est = 4.27	User ID: **A3LDPF5FMB782Z**	r_ui = 5.00 est = 4.67
Product ID: **1400501466**		**Product ID:** **1400501466**	
User ID: **A34BZM6S9L7QI4**	r_ui = None est = 4.29 for the	User ID: **A34BZM6S9L7QI4**	r_ui = None est = 4.29
Product ID: **1400501466**		**Product ID:** **1400501466**	

Observations about the model performance:

- Even though the predicted rating for the second user hasn't changed, tuning the parameter has really improved the rating for the first user (from 4.27 to 4.67) and the prediction is now much closer to the actual user rating than that given by the baseline model.

Model 3: Item-Item Similarity-based Model

Using the model,

- **Identifying similar Users to a given User (nearest neighbors)**

The IDs of the 5 most similar users to the first user (user 0) are:

```
[29, 53, 67, 106, 151]
```

- **Predicting top 5 products for user "A3LDPF5FMB782Z" with Model 3**

Using our similarity-based recommendation system, the top 5 product recommendations are:

- 1400532655
- 1400599997
- 9983891212
- B00000DM9W
- B00000J1V5

  Each of these products have a predicted rating of 4.29.

Model 4: Matrix Factorization based Model

Singular Value Decomposition (SVD)

- SVD is used to **compute the latent features** from the **user-item matrix**. But SVD does not work when we **miss values** in the **user-item matrix**.

```
RMSE: 0.8882 Precision: 0.853 Recall: 0.88 F_1 score: 0.866
```

Observations:

- We observe that the svd model has RMSE=0.888 on the test set.
- We are getting a recall of ~0.88, which means out of all the relevant products, 88% are recommended.
- We are getting a precision of ~0.85, which means of all the recommended books, 85% are relevant.
- The F_1 score of the baseline model is ~0.87. It indicates that most recommended products were relevant and that most relevant products were recommended.
- We can try to improve the performance by using GridSearchCV to tune different hyperparameters of the algorithm.

Model 4: Matrix Factorization based Model

- `RMSE: 0.8808 Precision: 0.854 Recall: 0.878 F_1 score: 0.866`

After tuning the hyperparameter values,

- the root mean squared error has decreased to 0.88.
- precision stayed roughly the same while recall has increased to ~88%.
- this means our F1-score has also increased a bit to ~87%.
- So the model performed better after tuning.

Model 4: Matrix Factorization based Model

Before Tuning		After Tuning	
User ID: **A3LDPF5FMB782Z**	r_ui = 5.00 est = 4.08	User ID: **A3LDPF5FMB782Z**	r_ui = 5.00 est = 4.13
Product ID: 1400501466		**Product ID: 1400501466**	
User ID: **A34BZM6S9L7QI4**	r_ui = None est = 4.40 for the	User ID: **A34BZM6S9L7QI4**	r_ui = None est = 4.22
Product ID: 1400501466		**Product ID: 1400501466**	

Observations about the model performance:

- The baseline SVD model performed well and predicted 4.08 and 4.40 for the first user and the second user respectively.
- There was a slight improvement using the optimized SVD model (the predicted rating for the first user rose to 4.13 while the second user reduced to 4.22).

Conclusion and Recommendations

- In this case study, we built recommendation systems using four different algorithms. They are as follows:
  - Rank-based using averages
  - User-user similarity-based collaborative filtering
  - Item-item similarity-based collaborative filtering
  - Model-based (matrix factorization) collaborative filtering
- To demonstrate "user-user similarity-based collaborative filtering", "item-item similarity-based collaborative filtering", and "model-based (matrix factorization) collaborative filtering", *surprise library* was used. For these algorithms, grid search cross-validation was used to find the optimal hyperparameters for the data, and improve the performance of the model.
- **For performance evaluation** of these models, **precision@k and recall@k** are used. Using these two metrics, the F_1 score is calculated for each working model.

Conclusion and Recommendations

- Overall, the **optimized matrix factorization collaborative filtering** has given the **best performance** in terms of the F1-Score (~0.87)
- Collaborative Filtering searches for neighbors based on similarity of products preferences and recommends products that those neighbors purchase while Matrix factorization works by decomposing the user-item matrix into the product of two lower-dimensional rectangular matrices.
- Matrix Factorization has lower RMSE (0.88) due to the reason that it assumes that both products and users are present in some low-dimensional space describing their properties and recommends a product based on its proximity to the user in the latent space. Implying it accounts for latent factors as well.
- We can try to even further improve the performance of these models using hyperparameter tuning.
- We can also try to combine different recommendation techniques to build a more complex model like hybrid recommendation systems.