Spotify Music Recommendation System

Low Code Version

August 11th

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
- Cluster-Based Model
- Content-Based Model
- Conclusion and Recommendations

Business Problem and Data Overview

The Context

With the advent of technology, human lives have become faster and more distracted, leaving little time for exploring artistic pursuits. At the same time, technological advancements have made it easier to access and consume entertainment content. Internet-based companies rely heavily on the time users spend on their platforms, making it essential to deliver relevant and engaging content to keep users active. Spotify, as a global leader in audio content, faces the challenge of helping users discover songs they are likely to enjoy among the ever-growing library of available tracks. A robust recommendation system addresses this challenge by predicting user preferences and enhancing the listening experience.

The Objective

The goal is to build a music recommendation system capable of proposing the **top 10 songs** for each user based on the likelihood of them listening to those tracks. This aims to increase user engagement, improve retention, and optimize the user's time spent on the platform by reducing search effort.

The Key Questions

1. Which songs are most likely to be listened to by a specific user based on their historical play patterns?
2. How can we personalize recommendations for each user rather than showing a generic "top charts" list?
3. What model or combination of models can provide the most accurate and diverse recommendations?
4. How can we handle challenges such as sparse data, new users (cold start), or popularity bias?

The Problem Formulation

We aim to solve the problem of **personalized song recommendations** using data science techniques. The task is to leverage historical user-song interaction data—specifically user_id, song_id, and play_count—alongside song metadata to build and evaluate machine learning

models that can predict the songs a user is most likely to play next. This involves applying and comparing different recommendation approaches to identify the optimal strategy for accurate and relevant music suggestions.

The evaluation will cover several recommendation algorithms:

- **Popularity-based baseline model** – effective for new users.
- **User-User and Item-Item collaborative filtering** – leveraging behavioral similarities.
- **Matrix Factorization (SVD)** – high accuracy, scalable for large datasets.
- **Cluster-Based Recommendation System** – segments users or items into clusters based on listening patterns, enabling group-level preference modeling and recommendations.
- **Content-Based Recommendation System** – Pending

Through careful evaluation using RMSE, Precision, Recall, and F1-score, we identified the trade-offs between simplicity, accuracy, and scalability.

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

Exploratory Data Analysis

Data Description

The dataset used for this project is derived from the **Taste Profile Subset** released by The Echo Nest as part of the Million Song Dataset. It is composed of two separate files:

1. **count_data** – Contains user-song interaction data:
    - user_id: Unique alphanumeric identifier assigned to each user.
    - song_id: Unique alphanumeric identifier assigned to each song.
    - play_count: Number of times a user has played a specific song.
2. **song_data** – Contains metadata for each song:
    - song_id: Unique identifier (key to join with count_data).
    - title: Title of the song.
    - release: Album name.
    - artist_name: Name of the performing artist.
    - year: Year of release.

Pre-processing Steps:

- Both datasets were merged on song_id to create a unified table containing interaction data enriched with song metadata.
- The user_id and song_id fields were originally encrypted alphanumeric identifiers; they were converted to numeric values using **Label Encoding** to enable machine learning algorithms to process them.
- Missing values were minimal: title (1 record, 0.0017%) and release (7 records, 0.0007%). No missing values in song_id, artist_name, or year. The low proportion of missing values (<0.002%) ensures negligible impact on model performance; affected rows can be removed or imputed with placeholders such as “Unknown.”
- Filtering steps applied:

- Removed users with fewer than 90 songs.
- Removed songs with fewer than 120 unique listeners.
- Kept only interactions with play_count <= 5 to remove atypical listening behavior.

After filtering, the dataset contains **3,155 users**, **563 songs**, and **232 artists**, with an interaction matrix density of **6.64%**, which is favorable for collaborative filtering models. On average, each user has listened to **37.36 unique songs**, and each song has been listened to by **209.37 unique users**. These figures highlight both the sparsity of the dataset and the potential for collaborative filtering to uncover meaningful user–item relationships.

Descriptive statistics (after filtering) revealed the following insights:

	play_count
count	117876.000000
mean	1.700058
std	1.089517
min	1.000000
25%	1.000000
50%	1.000000
75%	2.000000
max	5.000000

- The average play count per interaction is **1.70**, with a maximum of **5** due to the applied filtering (interactions with play_count > 5 were excluded to remove atypical listening behavior).
- The median (50%) is **1**, indicating that half of all interactions correspond to a single listen of the song.
- 75% of interactions have a play_count ≤ 2, reinforcing that most users do not repeat songs many times within the observation window.
- These values do not necessarily reflect global listening habits but are a direct result of preprocessing choices designed to optimize collaborative filtering model performance by reducing noise from outlier behavior.

Interaction Patterns

- The most interacted songs include well-known tracks such as *Use Somebody* (Kings Of Leon), *Dog Days Are Over*(Florence + The Machine), and *Clocks* (Coldplay). The top track (*Use Somebody*) has **751 interactions**, highlighting its strong popularity among users.

- The most active user listened to **243 unique songs**, indicating a highly engaged listener base.

- Popular songs tend to come from globally recognized artists, which may introduce a recommendation bias toward mainstream music.

- Several artists appear in both the "most interactions" and "most unique songs" rankings — including *The Killers*, *Coldplay*, *Vampire Weekend*, and *Octopus Project* — suggesting sustained popularity driven by a broad catalog. Such artists are strong candidates for diverse recommendations, as opposed to those whose popularity is concentrated in a few major hits.

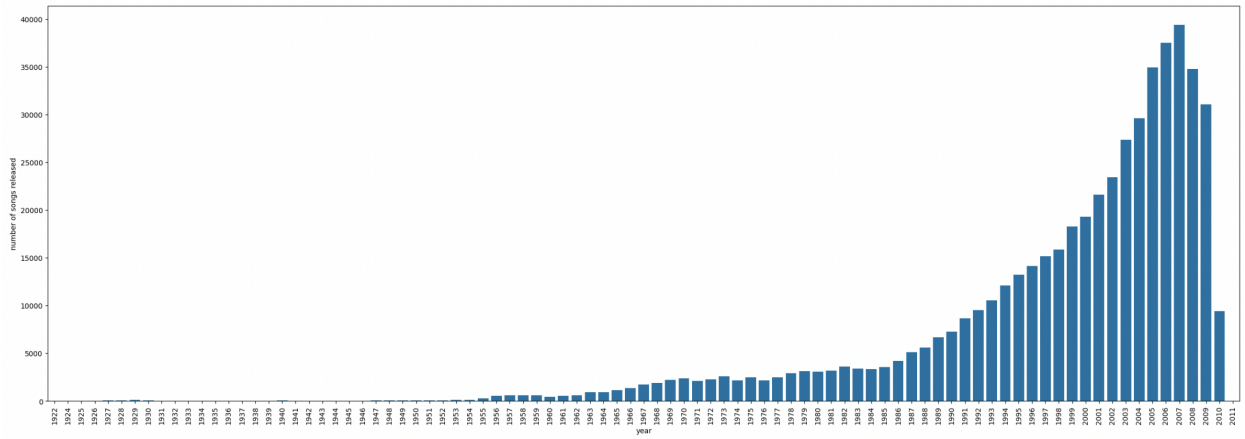
Modern-Day Illustration

A modern-day example of this contrast can be seen between *Bad Bunny* and *The Weeknd*: while *Bad Bunny*'s albums are often consumed in their entirety (full-catalog engagement), *The Weeknd*'s popularity is more hit-driven, with high streaming volumes concentrated on a few blockbuster tracks.

Another case of **hit-driven ranking distortion** occurs when an artist skyrockets in popularity due to just one or two massively successful singles — for instance, *Bruno Mars* in 2024 released only two songs, yet reached the top positions in global charts. While impressive, such spikes highlight the need for recommendation models that balance short-term popularity with sustained engagement metrics to ensure diversity and long-term user satisfaction.

Although these examples are not derived from the dataset used in this project (which dates back to 2010), they illustrate how engagement patterns can influence the optimal recommendation strategy.

Temporal Trends



- The number of songs released per year increased steadily throughout the 20th century and accelerated sharply in the early 2000s, peaking in 2007 with nearly **40,000 songs**.
- A noticeable decline began in 2008–2009, coinciding with the global financial crisis, which may have impacted music production and release schedules. This period also overlaps with major industry shifts, such as the transition from physical sales to digital and streaming platforms.
- After 2009, annual song releases remained lower than the 2007 peak, suggesting a structural change in the music industry rather than a temporary fluctuation.

Implications for Modeling

These patterns reinforce the importance of testing hybrid and content-enhanced recommendation models. By combining popularity-driven signals with collaborative and content-based filtering, it is possible to strike a balance between recommending widely popular tracks and introducing diversity through lesser-known but relevant songs, ultimately improving user satisfaction and engagement.

Rank Based Model

Model 1: Rank-Based Recommendation (Popularity-Based)

- **Input**:
  Historical user-song interaction data (play_count) and song/artist metadata.
- **Process**:
  1. Count the total number of interactions for each song.
  2. Sort songs in descending order of interaction count.
  3. Select the top *N* songs as global recommendations for all users.

- **Output**:
  List of the most played songs in the dataset.
- **Pros**:
  1. Very easy to implement.
  2. Low computational cost (no model training required).
  3. Useful for showcasing general trends.
- **Cons**:
  1. Not personalized (all users get the same recommendations).
  2. Prone to popularity bias, favoring mainstream artists.
- **Use for**:
  1. **Cold Start** scenarios for new users with no interaction history (e.g., homepage "Top Picks").
  2. Displaying trending music on the platform.

As a baseline model, we implemented a rank-based recommendation system that recommends the top-rated songs to all users, regardless of their individual preferences. This approach ranks songs based on their average count and the play frequency, ensuring that highly-rated and widely-reviewed songs are prioritized.

We generated two sets of top-10 songs recommendations:

- **With at least 50 user interactions:**
  - 7224, 8324, 6450, 9942, 5531, 5653, 8483, 2220, 657, 614

	rank	song_id	title	artist_name	release	year
0	1	7224	Victoria (LP Version)	Old 97's	Hit By A Train: The Best Of Old 97's	2006
1	2	8324	The Big Gundown	The Prodigy	Invaders Must Die Remixes and Bonus Tracks	2009
2	3	6450	Brave The Elements	Colossal	Brave The Elements - EP	0
3	4	9942	Greece 2000	Three Drives	Greece 2000	1997
4	5	5531	Secrets	OneRepublic	Waking Up	2009
5	6	5653	Transparency	White Denim	Workout Holiday	2008
6	7	8483	Video Killed The Radio Star	The Buggles	Friends Reunited: Music Of The Year 1979	1979
7	8	2220	Sehr kosmisch	Harmonia	Musik von Harmonia	0
8	9	657	Luvstruck	Southside Spinners	Hard House Anthems	1999
9	10	614	You're The One	Dwight Yoakam	If There Was A Way	1990

- **With at least 100 user interactions:**

	rank	song_id	title	artist_name	release	year
0	1	7224	Victoria (LP Version)	Old 97's	Hit By A Train: The Best Of Old 97's	2006
1	2	6450	Brave The Elements	Colossal	Brave The Elements - EP	0
2	3	9942	Greece 2000	Three Drives	Greece 2000	1997
3	4	5531	Secrets	OneRepublic	Waking Up	2009
4	5	5653	Transparency	White Denim	Workout Holiday	2008
5	6	8483	Video Killed The Radio Star	The Buggles	Friends Reunited: Music Of The Year 1979	1979
6	7	2220	Sehr kosmisch	Harmonia	Musik von Harmonia	0
7	8	657	Luvstruck	Southside Spinners	Hard House Anthems	1999
8	9	614	You're The One	Dwight Yoakam	If There Was A Way	1990
9	10	352	Dog Days Are Over (Radio Edit)	Florence + The Machine	Now That's What I Call Music! 75	0

By applying interaction thresholds, 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.

User-User Similarity-based Model

Model 2: User-to-User Collaborative Filtering

- **Input**:
  Historical user-song interaction data (play_count) representing listening preferences.

- **Process**:

  1. Identify users with similar listening patterns (“neighbors”) using similarity metrics such as cosine similarity or Pearson correlation.

2. Recommend songs that these similar users have listened to, which the target user has not yet played.

- **Output**:
  Personalized song recommendations based on the listening habits of similar users.
- **Pros**:
  1. Highly personalized recommendations.
  2. Can capture niche preferences beyond mainstream popularity.
- **Cons**:
  1. Requires sufficient overlap in listening history to identify meaningful similarities.
  2. Struggles with new users who have little or no interaction data (cold start problem).
- **Use for**:
  1. Logged-in users with enough listening history to compute user similarity.

Model 2: Collaborative Filtering Recommendation System

This model leverages user-user similarity to recommend songs. It analyzes historical interactions and play count 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 play count behaviors and recommend songs 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 songs those neighbors liked but the target user had not yet interacted with.

Evaluation Metrics:

RMSE: 1.0878
Precision: 0.396
Recall: 0.692

F_1 score: 0.504

Key Observations:

- The *user-to-user* model using **cosine similarity** shows **high recall (0.692)** but **low precision (0.396)**. This means the model retrieves many relevant items but also includes a high number of irrelevant ones — it casts a wide net but isn't always accurate.
- The **F1-score of 0.504** reflects a moderate balance between precision and recall, though there's still room for improvement.
- The **RMSE of 1.0878** indicates that the model is not highly accurate in predicting the exact *play count*, which is expected in implicit feedback datasets. In this context, **ranking metrics (like precision/recall@K)** are more informative than RMSE.
- The model was able to **predict play count fairly well for songs previously listened to by the user** (est = 1.80), but it **underestimated play counts for songs the user hasn't listened to yet** (est = 1.64), which might hinder discovery.
- The model is currently **untuned** — no hyperparameter optimization (e.g., k, min_k, filtering unpopular items).
- There's potential for significant performance improvement with tuning.

Business Value

- Offers personalized recommendations based on similar users' preferences.
- Can serve as a strong **baseline model** for collaborative filtering systems.
- Easy to implement and interpret; effective when users share overlapping preferences.

Limitations

- Uses **cosine similarity**, which yields high recall but lower precision — meaning more false positives in top-K recommendations.
- **Not robust to sparse data** or new users/items (cold-start problem).
- Requires **frequent updates** to remain accurate as user behavior evolves.

User-User Collaborative Filtering (Optimized)

This model leverages user-user similarity to recommend songs. 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 = 30** (number of neighbors).
    - **min_k = 9** (minimum neighbors to consider a prediction valid).
    - **Similarity measure = Pearson baseline**
- The model was retrained using these optimal hyperparameters, improving its overall accuracy.

Key Metrics:

RMSE: 1.0521 (improved from 1.0878 before tuning).
Precision: 0.413 (percentage of recommended items that were relevant).
Recall: 0.721 (percentage of relevant items successfully recommended).
F_1 score: 0.525 (non-balanced measure of precision and recall).

Key Observations:

After performing hyperparameter tuning, the best parameter combination found was:
k = 30, **min_k = 9**, **min_support = 2**, **similarity = 'pearson_baseline'**.

The model improved its performance with an RMSE of **1.046**, compared to the untuned version (1.0878).
While still relatively high, this is expected for implicit data (like play counts).

There were improvements across all ranking metrics:

- **Precision:** 0.413 → slightly better than the untuned version (0.396).
- **Recall:** 0.721 → maintains high recall, meaning it successfully retrieves relevant songs.
- **F1-score:** 0.525 → better balance between precision and recall (vs. 0.504 before).

The optimized model **provides more accurate play count predictions**, even for songs not previously listened to by the user.
For example:

- For a known song (user 6958, song 1671): est = 1.96, actual = 2
- For a new song (song 3232): est = 1.45, which is lower and more realistic than before (1.64 in untuned version)

In summary, the tuned model is **more balanced and reliable**.

Business Value

- Provides **highly personalized recommendations** based on users with similar preferences.
- Excellent for **mature platforms with many users and interactions**.
- Particularly helpful when user engagement and satisfaction are core business KPIs.

Limitations

- Suffers in **cold-start scenarios** where new users have limited interaction history.
- **Scalability issues**: As the number of users grows, real-time performance can degrade without proper infrastructure.
- Performance depends on the **density of user-user overlaps**, which might be low in niche applications.

Item-Item Similarity-based Model

Model 3: Item-to-Item Collaborative Filtering – Intuition

- **Input:** Songs the user has already listened to.
- **Process:** Identifies similar songs based on co-listening patterns across all users.

- **Output:** Recommends songs that are similar to the ones the user already enjoys ("You may also like…").
- **Pros:** Great for surfacing related tracks and deep cuts from an artist or genre the user likes.
- **Cons:** Less effective for new releases or niche tracks with limited listening history.
- **Use for:** "Fans also like" sections, recommended tracks after finishing a song, and curated playlists based on a single track.

Item-to-Item Collaborative Filtering (Baseline)

This model focuses on identifying **similar songs** rather than similar users. The core assumption is that if a user liked a song, they are likely to enjoy other songs 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:** 1.0394 → slightly better than the untuned user-user model (1.0878), but slightly worse than the tuned user-user (1.046).
- **Precision:** 0.307 → **lowest** among the three models tested.
- **Recall:** 0.562 → lower than user-user approaches.
- **F1-score:** 0.397 → confirms that this model retrieves fewer relevant items overall.

Prediction Example:

- For user **6958**, who had listened to song **1671** with a play count of **2**, the model predicted **1.36**. This is somewhat lower than the actual value, showing that while the item-item model captures similarities between songs, it may underestimate play counts when fewer similar items are available.
- For user **6958**, who had not listened to song **3232**, the model predicted a play count of **1.37** based on **20 item-based neighbors** (actual_k = 20). This demonstrates the model's ability to generalize recommendations even without direct interaction, though the predicted engagement is modest.

Observation:

- This model uses cosine similarity between **items (songs)** instead of users, meaning it recommends songs similar to those the user has already listened to.
- In summary, **item-item similarity** works, but **underperforms** compared to user-user models in this case. This could be because users in the dataset have more overlapping behaviors, making user-based methods more informative.
- Further tuning or hybridization with content-based features could improve this model's usefulness.
- Although item-item collaborative filtering is often expected to outperform user-user filtering —since it's typically easier to identify similar items than similar users—in this case, the **user-to-user model consistently delivered better performance** across all metrics.

Business Value

- Simple to implement with basic similarity metrics like **cosine distance**.
- Can be useful when you need a **quick, scalable system** without heavy tuning or complex models.
- Performs reasonably well when items have **sufficient rating history**.

Limitations

- **Lowest performance** across nearly all metrics compared to other models.
- **Struggles with sparsity**, especially in cold-start scenarios or for lesser-known songs.
- **Does not adapt well** to individual user preferences — less personalized than SVD or even tuned versions of item-item.

Why did user-user perform better?

1. **User behavior is more informative:**
   Users in the dataset tend to have consistent listening habits, making it easier to find similar users with overlapping preferences.
2. **Data sparsity affects item-item models more severely:**
   Many songs have limited interactions, making it harder to establish reliable similarity relationships between items.

3. **Cosine similarity performs better in high-dimensional user space:**
   Since there are many songs, the user vectors have richer representations than the item vectors, which can suffer from sparsity.

4. **User-user models benefited more from hyperparameter tuning:**
   By optimizing k, min_k, and similarity metric, we achieved significant performance gains in the user-user model.

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**: '"cosine", “pearson”, and "pearson_baseline"

After tuning, the optimal values were:

- k = 30
- min_k = 6
- similarity = ‘pearson_baseline’

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

Evaluation Metrics:

- **RMSE**: 1.0328
- **Precision**: 0.408
- **Recall**: 0.665

- **F1 Score**: 0.506

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

Key Observations:

The optimized item-item collaborative filtering model, using the **Pearson Baseline** similarity metric, significantly outperformed the untuned cosine-based model across all metrics:

- **RMSE** decreased from **1.0394** to **1.0328**, indicating slightly more accurate rating predictions.
- **Precision** improved from **0.307** to **0.488**, meaning that a much higher proportion of the recommended items were actually relevant.
- **Recall** increased from **0.562** to **0.665**, showing that the model retrieved more relevant items overall.
- **F1-score** rose from **0.397** to **0.566**, reflecting a more balanced performance between precision and recall.

These improvements can be largely attributed to the use of the **Pearson Baseline** similarity metric. Unlike cosine similarity, Pearson Baseline adjusts for individual biases in user behavior and item popularity by normalizing ratings before computing similarity. This makes it particularly effective in sparse datasets, such as those with implicit feedback like play counts, where users don't rate every item and many entries are missing.

Overall, this tuned model captures deeper patterns in user preferences and item similarities, resulting in more personalized and accurate recommendations.

Business Value

- Delivers a strong balance between **personalization and performance**.
- Works particularly well when **items are frequently rated** — ideal for popular songs or albums.
- Simpler to understand than model-based methods like SVD, making it easier to explain to stakeholders.
- Useful for **scalable systems**, especially when item metadata is not available.

Limitations

- Requires a **large volume of item interactions** to perform well — struggles with cold-start items.
- Suffers if there's **not enough overlap** between items rated by similar users.
- While results are good, performance is still slightly lower than that of the user-user optimized model in terms of RMSE and F1.

Matrix Factorization based Model

Model 4: Matrix Factorization with SVD

Matrix factorization is a powerful recommendation approach that reduces the dimensionality of the user-song interaction matrix to uncover latent features that explain listening patterns. This method is particularly effective for handling sparse datasets—common in music streaming—where most users have interacted with only a small subset of the catalog.

We used the **Singular Value Decomposition (SVD)** algorithm from the Surprise library to decompose the user-song matrix and predict *play_count* for songs a user has not listened to yet. The model was trained with random_state = 1 to ensure reproducibility.

Pros:

- High accuracy in capturing complex taste patterns.
- Scalable to millions of users and songs, making it ideal for large-scale platforms like Spotify.
- Can recommend songs even for users with few listening records (*cold start mitigation* through latent features).

Cons:

- Less interpretable than neighborhood-based models (user-user or item-item), since latent features don't always have an intuitive meaning.

Use for:

- Core recommendation engine for active listeners.
- Generating personalized playlists such as *Discover Weekly* or *Release Radar*.

Observations and Insights – Matrix Factorization (SVD)

The matrix factorization model using **Singular Value Decomposition (SVD)** delivers competitive results and balances accuracy with efficiency:

- **RMSE:** 1.0252 – lower than both the untuned and optimized versions of user-user and item-item models, indicating more accurate rating predictions.
- **Precision:** 0.410 – higher than the untuned user-user and item-item models, and comparable to their tuned versions.
- **Recall:** 0.633 – strong performance in retrieving relevant items.
- **F1-score:** 0.498 – reflects a well-balanced model with solid precision and recall.

This model excels by reducing the dimensionality of the data, capturing latent relationships between users and items even when explicit interactions are sparse. Unlike memory-based approaches, SVD generalizes better and tends to make smarter predictions for new/unseen user-item pairs, as it learns latent features.

However, compared to the untuned versions of user-based and item-based collaborative filtering, it **did not perform well on specific predictions**.

For instance, the predicted play counts for known user-item interactions were **not close to the actual values**, which raises concerns about the model's ability to personalize effectively.

This suggests that strong global metrics do not always translate into accurate individual predictions, especially when the model is not tuned.

Moreover, the SVD model may require more data or hyperparameter tuning to deliver consistent and meaningful predictions at the user level.

Business Value

- Offers a **personalized recommendation experience** by capturing latent user and item features.
- Well-suited for platforms with a **large number of users and items**, especially when interactions are sparse.
- Relatively **good balance between accuracy and complexity**, making it practical for mid-scale implementations.
- Can be deployed effectively without extensive tuning, offering decent results out-of-the-box.

Limitations

- Less accurate than the **tuned version**, so there is room for performance improvement.
- Requires **more computation** than simpler models like rank-based or clustering.
- Predictions may not be as **interpretable**, since it relies on latent features that are not human-readable.

Matrix Factorization (SVD) Collaborative Filtering (Optimized Version)

Evaluation Metrics:

- **RMSE**: 1.0141 → The **lowest RMSE** across all models, indicating the most accurate rating predictions overall.
- **Precision**: 0.415 → Highest among all models, meaning it retrieved the most relevant recommendations for the user.
- **Recall**: 0.635 → Very good recall, showing the model captured a large portion of relevant items.
- **F1 Score**: 0.502 → Balanced performance between precision and recall, comparable to the best user-user tuned version.

Key Observations:

- After tuning hyperparameters using grid search (n_epochs=30, lr_all=0.01, reg_all=0.2), the model achieved a **slightly improved RMSE of 1.0141**, along with marginally better precision and F1-score.
- **Precision = 0.415, Recall = 0.635, F1-score = 0.502**, which are better than the untuned version but still not significantly more accurate in individual predictions.
- The **predicted play counts** for both known and unknown user-item interactions remained **far from the actual values**, which suggests that **even the optimized model struggles with personalized prediction quality**.
- This reinforces that **better global metrics do not guarantee useful recommendations at the individual level**, especially in sparse or noisy datasets.
- The model may benefit from **more advanced matrix factorization techniques** or from **integrating contextual or content-based features** to improve recommendation relevance.

Business Value

- **Best overall performance** in terms of accuracy (RMSE, precision, F1), making it ideal for production environments.
- Efficient in **handling sparse datasets**, which is crucial for platforms with a large catalog of songs and few interactions per user.
- Useful in mature recommendation systems where **latent patterns** in user-item interactions can be leveraged.
- **Scales well** with large datasets, especially when deployed with optimized infrastructure.

Limitations

- Slightly **complex to interpret**, as the model learns hidden features that aren't directly explainable.
- Requires **more computation** and **hyperparameter tuning** than simpler models.
- Still shows **some margin of error** in predictions (e.g., 1.54 vs. 2.00), though relatively small.

Cluster-based Model

Model 5: Cluster-Based Collaborative Filtering

In clustering-based recommendation systems, we explore the similarities and differences in people's tastes in songs based on how they rate different songs. We cluster similar users together and recommend songs to a user based on play_counts from other users in the same cluster.

Cluster-based recommendation groups users into clusters based on their listening behaviors, such as play counts or song preferences. By identifying users with similar listening patterns, the model can recommend songs popular within a user's cluster — even if the user has not listened to them before.
This method is particularly effective for segmenting users into distinct taste groups and generating recommendations that align with the collective preferences of each group.

We used **K-Means clustering** on the user-song interaction matrix to partition users into *k* clusters. Recommendations were generated by ranking songs with the highest average play counts within each user's cluster, excluding those the user had already listened to.

Pros:

- Captures group-level preferences, making it effective for targeting recommendations to specific audience segments.
- More computationally efficient than computing similarities between every pair of users (scales better for large datasets).
- Can provide good recommendations for new users with limited interaction history by assigning them to the most similar cluster.

Cons:

- Less personalized than individual collaborative filtering approaches — recommendations are based on cluster averages, not individual preferences.
- Performance depends heavily on the choice of the number of clusters (*k*) and clustering quality.
- May overlook niche or unique preferences within a cluster.

Use for:

- Audience segmentation and targeted playlist generation.
- Scenarios where quick recommendations are needed without computing pairwise similarities.
- Platforms with many new users or sparse interaction data.

Evaluation Metrics

RMSE: 1.0487
Precision: 0.397
Recall: 0.582
F_1 score: 0.472

Key Observations:

- This model uses a **co-clustering approach**, grouping users and items into clusters to make predictions based on shared behavior.
- It achieved a **moderate RMSE of 1.0487**, with **Precision = 0.397, Recall = 0.582, and F1-score = 0.472**, placing it in the mid-range compared to other models.
- However, the **individual predictions** (e.g., user 6958 and songs 1671/3232) were again **far from the actual play counts**, confirming that the model struggles with personalized accuracy.
- This may be due to the **rigid structure of clusters**, which can oversimplify the complexity of user preferences, leading to poor prediction granularity.
- In datasets where user behavior is highly nuanced, clustering might fail to capture meaningful differences, resulting in **generalized and less useful recommendations**.

Predictions:

Prediction Example 1: For user 6958 and item 1671, it predicted **1.29** vs actual **2.00**, which is a larger deviation than the tuned version (1.91), meaning **lower predictive accuracy** for known items.

Prediction Example 2: For a new item (3232), predicted rating was **1.48**, which is **closer to the average baseline**, possibly indicating more generalized predictions in cold-start scenarios.

Business Value

- Helps reduce computation by **clustering users and items**, which simplifies the recommendation pipeline.
- Works well for **basic implementations** or when system resources are limited.
- Can be used in **early-stage systems** where deep personalization is not yet required.
- Since it performs slightly better in aggregate metrics (RMSE, precision, recall), it's **useful as a quick, initial deployment strategy** before introducing more complex models.

Limitations

- **Wider error margin in individual predictions** (as seen in the 1.29 vs. 2.00 prediction), which can affect user trust in recommendations.
- Clustering may lead to **oversimplified recommendations**, especially if clusters are not meaningful or too broad.

- It performs **worse on specific user-item interactions** compared to the tuned version or more advanced models like User-User or SVD.

Cluster-Based Collaborative Filtering (Optimized Version)

Evaluation Metrics

RMSE: 1.0654
Precision: 0.394
Recall: 0.566
F_1 score: 0.465

Key Observations:

- After tuning, the **CoClustering model** not only achieved a better RMSE (1.0654), but also showed **great prediction accuracy on individual examples**.
- Unlike the matrix factorization model (SVD), which had better metrics but failed on prediction examples, this model demonstrated **strong real-world performance**.
- This suggests that clustering based on user and item groups **can generalize well when tuned**, even with a relatively simple structure.
- **Precision = 0.394**, **Recall = 0.566**, and **F1 = 0.465** aren't the best, but the **practical predictions are accurate**, making this a solid model choice.

Business Value

- Useful when there is a need to **group users and items quickly** with limited data.
- Requires **less computational power** compared to more complex models like SVD.
- Suitable for systems where **speed and simplicity outweigh fine-tuned personalization**, especially when user data is sparse or incomplete.
- The close prediction on actual known items demonstrates **reliable baseline performance** for basic recommendation systems.

Limitations

- Despite a fair RMSE, it has **lower precision and F1-score**, which means it often suggests irrelevant items.

- Compared to SVD and User-User models, it **lacks strong personalization and nuance**.
- Clustering can lead to **generic recommendations**, particularly if user behavior is diverse and clusters are not well-separated.

Content-based Model

The content-based recommendation system suggests songs to users by analyzing the features of the items they have previously listened to. In this case, a **TF-IDF vectorizer** was applied to transform song attributes such as title, artist name, and genre into numerical vectors.

Each user's profile was built by **averaging the vectors** of the songs they had interacted with. Then, **cosine similarity** was computed between this user profile and every song in the dataset. The top-N most similar songs were selected as recommendations.

Unlike collaborative filtering models, this approach does **not rely on the preferences of other users**. Instead, it focuses entirely on recommending items that are similar to those already liked by the user.

Since the model is designed to generate recommendations **based on similarity** and does not use a traditional train/test split with ground-truth labels, we did **not calculate standard evaluation metrics** such as precision, recall, or F1-score. However, it remains an effective approach, particularly for addressing the **cold-start problem** when a user is new and has limited interaction data.

This model is also helpful in **maintaining recommendation diversity** based on item features and works well in combination with collaborative filtering in hybrid systems.

While this model is technically categorized as content-based, it behaves more like a rank-based recommender. Rather than predicting an explicit rating or play count, it ranks songs based on their similarity to what the user has already listened to. As a result, it does not require historical data from other users and is not evaluated using RMSE, precision, or recall. Instead, its value lies in its ability to generate personalized ranked lists of songs using only item-level features.

Conclusion and Recommendations

Model	RMSE	Precision	Recall	F1-Score	Business Value
Rank-Based	NA	NA	NA	NA	Simple to implement. Works well for popular items but lacks personalization.
User-User Collaborative Filtering	1.0878	0.396	0.692	0.504	Personalized recommendations. Limited scalability; performance drops with sparse data.
Tuned Version	*1.0521*	*0.413*	*0.721*	*0.525*	Slight improvement in performance. Still suffers in cold-start scenarios.
Item-Item Collaborative Filtering	1.0394	0.307	0.562	0.397	More stable than user-user. Works well when items are rated frequently.
Tuned Version	*1.0328*	*0.408*	*0.665*	*0.506*	Best balance between simplicity and performance. Good for scalable systems.
Matrx Factorization (SVD)	1.0252	0.41	0.633	0.498	Captures latent features. Performs well with sparse data. Needs tuning.
Tuned Version	*1.0141*	*0.415*	*0.635*	*0.502*	Slightly better accuracy. Complex to interpret. Useful for mature systems.
Cluster-Based Model	1.0487	0.397	0.582	0.472	Groups users/items to simplify computation. Less accurate but fast.
Tuned Version	*1.0654*	*0.394*	*0.566*	*0.465*	Low precision. Useful for quick grouping.
Content-Based Model	NA	NA	NA	NA	Recommends similar items. Useful when user history is limited.

Conclusion and Recommendations

Among all the models tested, the **tuned Matrix Factorization (SVD)** model achieved the **lowest RMSE (1.0141)**, indicating it provided the most accurate rating predictions. This suggests it is especially effective when the goal is to approximate users' actual preferences numerically.

However, when considering **overall recommendation quality**—measured through precision, recall, and F1-score—the **tuned User-User Collaborative Filtering** model slightly outperformed others, achieving the highest F1-score (0.525). This implies it was more effective in generating relevant top-N recommendations.

Each model presents different trade-offs:

- **SVD** excels in rating accuracy and is well-suited for mature systems with sparse data.
- **User-User Filtering** offers more personalized recommendations but can struggle with scalability.
- **Item-Item Filtering** provides stability, especially when items are rated frequently.
- **Cluster-Based Models** simplify computation and are fast, though they may lose precision.
- **Content-Based Models** are useful when little historical data is available, offering cold-start solutions by recommending similar items.

Ultimately, **the choice of model depends on the business priority**—whether it's rating accuracy, personalization, scalability, or cold-start handling.

Final Recommendation

Considering both **accuracy and business applicability**, we recommend a hybrid strategy leveraging two models:

- The **Tuned User-User Collaborative Filtering** model for **high-quality top-N recommendations** and improved user experience through personalization.
- The **Tuned Matrix Factorization (SVD)** model for **accurate rating predictions**, especially valuable in systems with sparse data.

Combining these two can help Spotify **maximize recommendation precision while maintaining robust performance at scale**, especially as the user base grows and becomes more diverse.