

Scalable Data science – DATA420

Assignment 2

The Million Song Dataset

Table of Contents

1. Background	3
2. Data processing	3
2.1. Directory Tree of the Dataset in HDFS	3
2.2. Table of Dataset Information	3
2.3. Comparison of Row Counts to Unique Songs	4
2.4. Summary of the Audio Attributes from Audio Features	4
2.5. Creating a StructType Automatically from Audio Attributes	4
2.6. Renaming Columns in the Audio Feature Datasets	4
3. Audio similarity	5
3.1. Summary of the Audio Features Dataset	5
3.2. Descriptive Statistics for Audio Features	5
3.3. Conclusions from Descriptive Statistics	5
3.4. Correlation Between Features	6
3.5. Genre Distribution and Impact on Model Performance	6
3.6. Developing a Binary Classification Model	7
3.7. Class Balance	7
3.8. Splitting the Dataset and Resampling	7
3.9. Performance Metrics	8
3.10. Performance Comparison	8
3.11. Additional Considerations	8
3.12. Algorithm Explanation for Multiclass Classification	8
3.13. Class Balance Consideration	9
3.14. Dataset Split and Stratification	9
3.15. Performance Metrics for Multiclass Classification	9
3.16. Performance Metrics	9
4. Song recommendations	10
4.1. Advantages and Disadvantages of Repartitioning and Caching	10
4.2. Statistics of Songs and Users	10
4.3. Definition of Song Popularity and User Activity	11
4.4. Visualization of Song Popularity and User Activity	11
4.5. Song Recommendation Model Development	12
4.6. Table of Remaining Users and Songs After Filtering	12
4.7. Filtering Method for Users and Songs	13
4.8. Ensuring Test Set Coverage	13
4.9. Examples of User-Specific Recommendations	13
4.10. Table of Model Performance Metrics on the Test Set	13
4.11. Explanation of Each Metric and Its Limitations	13
4.12. Real-World Evaluation of a Recommendation System	14

Report on Million song Dataset

Name: Shantikrishna Panicker

Student ID: 26037298

Email: spa296@uclive.ac.nz

1. Background

The purpose of this assignment is to explore two key tasks in the field of machine learning: audio similarity and song recommendations. We aim to predict music genres based on audio features and to create personalized song recommendations using collaborative filtering methods.

In the audio similarity section, we utilize classification algorithms to determine the genres of songs based on their audio features. These features, derived from digital signal processing, include aspects like Mel-frequency cepstral coefficients (MFCCs) and spectral roll-off, which represent the properties of the audio waveform. We investigate both binary classification (for example, identifying whether a song is in the "Electronic" genre) and multiclass classification (recognizing specific genres such as Pop, Rock, or Jazz).

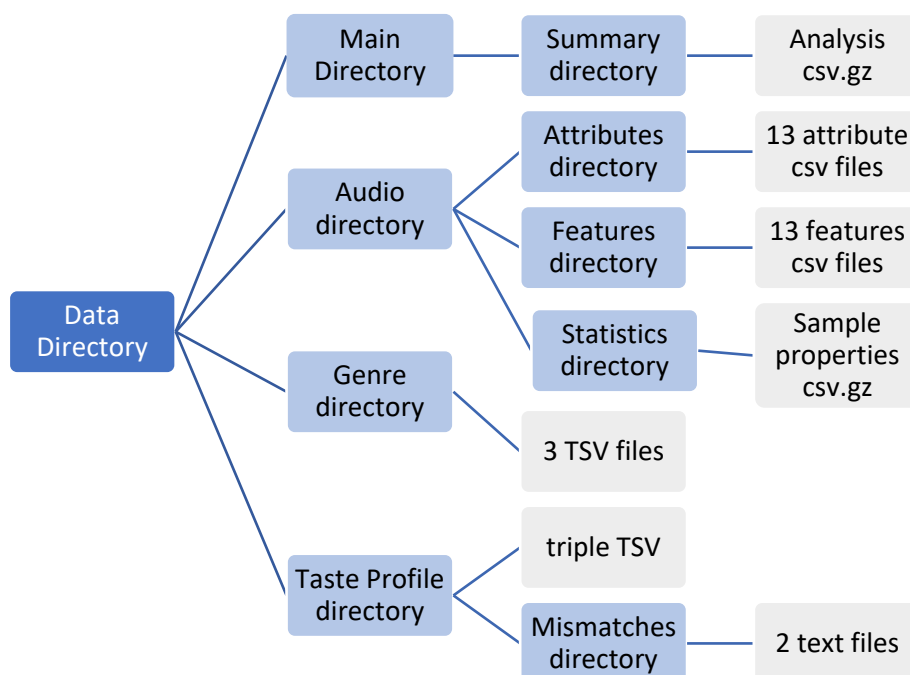
For the song recommendation part, we focus on building a collaborative filtering model that utilizes the Taste Profile dataset, which records user interactions with songs through play counts. This collaborative filtering technique enables the recommendation of songs to users by analyzing their listening habits in conjunction with those of similar users, thus creating a personalized recommendation system that does not require direct input from users.

2. Data processing

2.1. Directory Tree of the Dataset in HDFS

The directory structure, derived from the outputs of the `hdfs dfs -ls` command shown in the screenshots, offers an overview of the organization of the Million Song Dataset (MSD) files within HDFS.

Figure 1:



2.2. Table of Dataset Information

The table below (Table 1) provides a summary of the datasets, detailing their filenames, sizes, formats, data types, and the total number of rows, as derived from the screenshots.

Table 1:

Dataset name	Location in HDFS	Size(Bytes)	Format	Data Types	Row count
Genre Dataset	/data/msd/genre	~241 MB	TSV	track_id (String), genre (String)	422,714
Audio Features Dataset	/data/msd/audio/features	~97.8 GB	CSV	Continuous features (Double)	12,927,867
Taste Profile Dataset	/data/msd/tasteprofile	~3.8 GB	TSV	user_id (String), song_id (String), play_count (Integer)	48,373,586

2.3. Comparison of Row Counts to Unique Songs

The Genre Dataset includes 422,714 unique songs, each assigned to specific genres. In comparison, the Audio Features Dataset is much larger, containing a total of 12,927,867 entries, as it covers audio feature data for a wider range of tracks. Of these, only 3,383,455 are unique songs, suggesting a discrepancy likely due to multiple audio features being recorded for individual tracks. The Taste Profile Dataset is notable for having the most entries, documenting interactions between 1,019,318 unique users and a total of 48,373,586 user-song interactions.

2.4. Summary of the Audio Attributes from Audio Features

The audio attributes datasets contain essential metadata that outlines the structure of the audio features datasets. This metadata includes the names of the features, which represent various audio characteristics derived from musical tracks, along with their associated data types, such as numerical or categorical classifications. The decision to separate these attributes from the actual audio features is intended to create a flexible framework. These attributes act as a schema for the features, allowing for a modular and adaptable system that can be updated or modified without requiring changes to the main dataset. This separation is beneficial as it enables independent updates to the schema or the feature extraction process, allows for the dynamic application of schemas when loading feature datasets, and improves clarity when managing large datasets with many features. Ultimately, the attributes datasets provide a foundational reference for understanding the more complex audio features datasets.

2.5. Creating a StructType Automatically from Audio Attributes

To effectively load and manage audio feature datasets, we used the audio attributes metadata to automatically create a schema. This schema defines the structure of the dataset by specifying the names and types of each feature. By automating this process, we minimize the risk of human error and ensure consistency across all datasets. The process involved several key steps: first, we loaded the audio attributes, associating feature names with their corresponding data types. Next, we dynamically generated a schema based on this mapping, converting each feature's type (such as double or integer) into the appropriate Spark data type. Finally, by applying this schema during the loading of audio feature datasets, we guaranteed accurate interpretation of each column, which improved subsequent analysis and model training. This approach is essential for managing large, distributed datasets in systems like Spark, where effective schema management is critical for performance and usability.

2.6. Renaming Columns in the Audio Feature Datasets

The original column names in the audio feature datasets were overly lengthy and complicated, providing intricate descriptions of each audio characteristic, such as "Area_Method_of_Moments_Overall_Standard_Deviation_1." While these names offered valuable insights, they were not practical for machine learning applications and data processing. Therefore, we decided to simplify the column names to more intuitive alternatives. In our renaming strategy, we prioritized clarity by replacing the long names with concise labels like feature_1, feature_2, and so forth, making them easier to reference and manipulate during

modeling and analysis. We also maintained a consistent naming convention across all datasets to enable smooth merging and comparison in future analyses. This renaming initiative greatly enhances the clarity and usability of the datasets, particularly when handling numerous features and large volumes of data.

3. Audio similarity

This section explores audio-based features and their ability to predict music track genres. We selected a particular dataset of audio features, performed statistical analyses, identified correlations between features, and examined genre distributions to assess their impact on model performance.

3.1. Summary of the Audio Features Dataset

For this task, we chose the msd-jmir-mfcc-all-v1.0 dataset, which contains Mel Frequency Cepstral Coefficients (MFCCs). MFCCs are a widely used technique for representing timbre in audio analysis, as they effectively capture the tonal characteristics of sound, making them particularly useful for distinguishing between different music genres based on their acoustic features.

The selection of MFCCs is based on their common use in music genre classification, as they provide a good balance between retaining important information and keeping the feature set manageable. This concise audio representation aligns well with human auditory perception, which is essential for accurately classifying music genres.

3.2. Descriptive Statistics for Audio Features

The following (Table 2) presents a typical example of the descriptive statistics calculated for the MFCC audio features:

Table 2:

Feature	Count	Mean	Stddev	Min	Max
Feature 1	994623	46.24	23.14	0.0	544.0
Feature 2	994623	5.61	1.89	0.0	51.19
Feature 3	994623	4.30	1.20	0.0	24.48
Feature 4	994623	3.22	0.73	0.0	20.28
Feature 5	994623	2.68	0.58	0.0	21.02
Feature 6	994623	2.43	0.50	0.0	33.46
Feature 7	994623	2.24	0.42	0.0	19.15
Feature 8	994623	1.94	0.35	0.0	23.39
Feature 9	994623	1.86	0.33	0.0	23.08
Feature 10	994623	1.82	0.32	0.0	26.12

- Observations:
Certain characteristics, such as Feature 1, exhibit significant variability, as indicated by a large standard deviation and a broad range. This suggests that this feature reflects considerable differences among the tracks. In contrast, Features 3, 4, and 5, which have narrower ranges, are likely to represent more consistent patterns. While there are no features with extreme values that necessitate special handling, like normalization, the extensive range observed in some features should be considered when scaling data for training.

3.3. Conclusions from Descriptive Statistics

- The analysis reveals that although most features have similar ranges, there is significant variability in their means and standard deviations. Some features may dominate others due to their value ranges, highlighting the need for normalization or standardization during model training.
- Given the differences in feature ranges, it is crucial to implement feature scaling, such as standardization, to ensure that the model does not disproportionately favor certain features based solely on their value ranges.

- Features with higher means and standard deviations are likely to be more effective in distinguishing between genres. For example, Feature 1, which has a notably high mean and standard deviation, may successfully capture essential aspects of timbre that are important for accurate genre classification.

3.4. Correlation Between Features

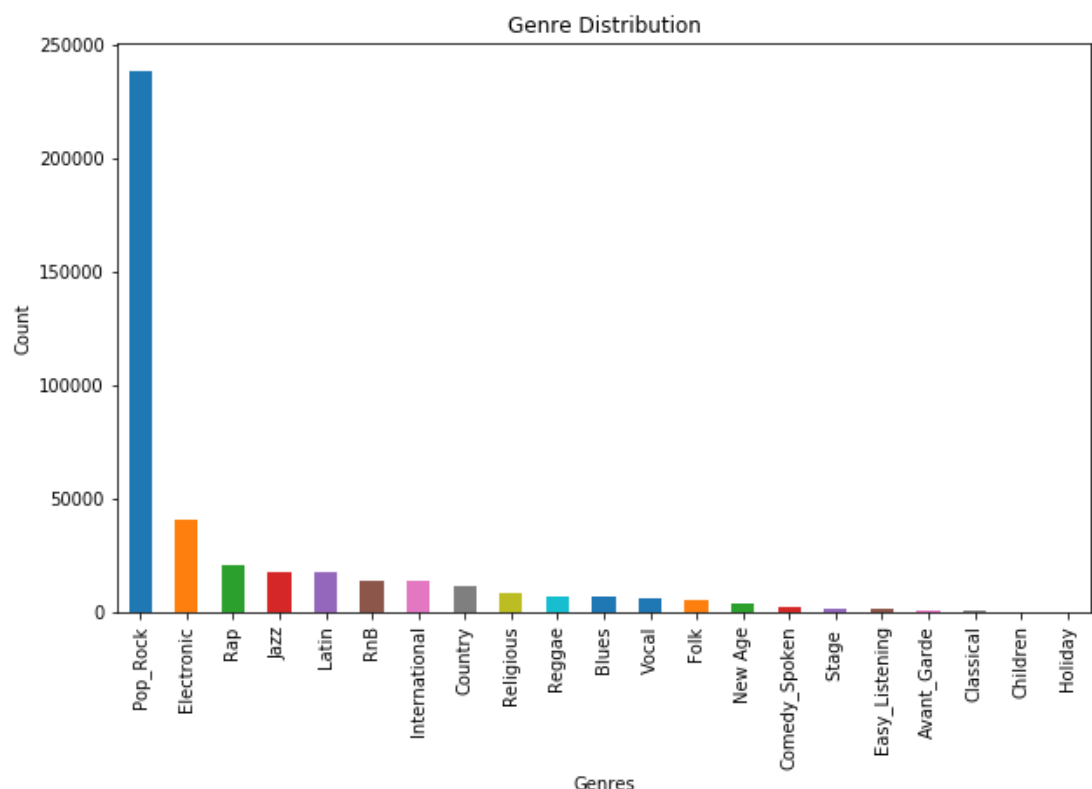
The analysis of feature correlations revealed notable relationships between certain pairs of features. Several examples of strongly correlated features are provided.

- **Feature 8 and Feature 9:** Correlation of 0.809.
- **Feature 9 and Feature 10:** Correlation of 0.821.
- **Feature 11 and Feature 12:** Correlation of 0.8375.
- Impact on Model Training:
 - Redundancy in a model can occur when features are strongly correlated. One strategy is to retain these features during the initial training phase, enabling the model's built-in feature selection to handle redundancy. Another option is to use dimensionality reduction methods like Principal Component Analysis, which can help reduce overfitting by lowering the number of features.
 - Strong correlations among features indicate that they might be capturing similar aspects of the audio signal. Therefore, reducing the number of correlated features could improve the model's performance by simplifying the feature space.

3.5. Genre Distribution and Impact on Model Performance

- Visualization of Genre Distribution:
The visualization (Figure 2) of genre distribution shows a clear imbalance, with Pop_Rock showing a significant dominance. Although genres like Electronic, Rap, and Jazz are present, they are much less prominent. Furthermore, there are rare genres such as Children, Holiday, and Avant_Garde.

Figure 2:



- Impact on Model Performance:

The significant class imbalance is likely to bias the model's predictions towards the dominant genre (Pop_Rock), which could reduce its effectiveness in recognizing less common genres. To address this challenge, it may be beneficial to employ strategies like class weighting or resampling techniques, which include either oversampling the minority classes or undersampling the majority classes.

Additionally, this imbalance may impede the model's ability to identify the distinct features linked to the rarer genres, making it essential to consider alternative methods, such as ensemble models that focus on the minority classes.

3.6. Developing a Binary Classification Model

- Summary of Chosen Algorithms and Preprocessing:

For the binary classification task, we chose three algorithms from the spark.ml library, each providing distinct benefits related to speed, interpretability, precision, and scalability.

Table 3:

Algorithm	Advantages	Disadvantages	Preprocessing Required
Logistic Regression (LR)	Simple, interpretable, fast, good for linear separability	Limited predictive power in non-linear data	Scaling is necessary for uniform feature contribution.
Random Forest (RF)	Handles non-linearity, reduces overfitting, interpretable	Slower due to ensemble learning, not ideal for very high-dimensional data	Minimal preprocessing, works with raw and scaled features.
Gradient-Boosted Trees (GBT)	High predictive accuracy, handles non-linearity, good for imbalanced datasets	Slower training, prone to overfitting if not tuned properly	Scaling is required to ensure fair comparison of features.

- Logistic Regression (LR) is a simple linear model that is both easy to interpret and efficient for high-dimensional datasets. Its effectiveness is maximized when the data can be linearly separated, which requires feature scaling before training.
- Random Forest (RF) is an ensemble method that improves predictive accuracy by reducing overfitting. It can handle large datasets with many features, but it is generally slower than simpler models like logistic regression.
- Gradient-Boosted Trees (GBT) build trees sequentially, correcting errors from previous iterations. This method achieves high predictive accuracy, making it particularly useful for imbalanced datasets. However, the training process is slower due to the iterative nature of boosting.

3.7. Class Balance

The distribution of the binary labels is uneven, with merely 9.67% of the tracks categorized as "Electronic." Such an imbalance may lead models to favor the majority class (non-Electronic), thereby diminishing the precision for the minority class. Addressing class imbalance is essential to prevent models from overfitting to the majority class. It is important for all algorithms to recognize this disparity, with Gradient-Boosted Trees (GBT) and Random Forest being especially effective in managing imbalanced datasets due to their ensemble methodologies.

3.8. Splitting the Dataset and Resampling

Stratified random sampling was used in the train-test split to address class imbalance. This approach maintains the ratio of "Electronic" and "Non-Electronic" tracks in both the training and test datasets, thereby avoiding the dominance of the majority class in the test set. The dataset was divided into 80% for training and 20% for testing. Although oversampling of the minority

class (Electronic) was contemplated to enhance the balance of the training set, it was deemed unnecessary given the effectiveness of Gradient Boosting Trees (GBT) and Random Forest (RF) in managing imbalanced datasets.

3.9. Performance Metrics

The selected metrics for assessing model performance include several key indicators. Accuracy assesses the ratio of correct predictions but may not provide a true representation in cases of imbalanced datasets. Precision evaluates the correctness of positive predictions, specifically for electronic tracks, thereby reducing the likelihood of false positives. Recall gauges the model's effectiveness in recognizing all genuine positive instances, aiming to decrease false negatives. The F1 Score serves as the harmonic mean of precision and recall, offering a balanced evaluation that is particularly useful for imbalanced datasets.

3.10. Performance Comparison

The table (Table 4) below summarizes the performance metrics for each algorithm:

Table 4:

Algorithm	Precision	Recall	F1 Score	Accuracy
Logistic Regression (LR)	0.893	0.912	0.886	0.797
Random Forest (RF)	0.893	0.906	0.864	0.737
Gradient-Boosted Trees (GBT)	0.896	0.913	0.890	0.824

- Logistic Regression (LR) showed a reasonable balance between precision and recall; however, its overall accuracy and F1 score were lower than those of Gradient-Boosted Trees (GBT).
- Random Forest (RF) performed slightly worse than Logistic Regression, particularly in accuracy, but it still demonstrated a respectable level of predictive performance.
- Gradient-Boosted Trees (GBT) outperformed both competing models on all evaluation metrics, achieving higher precision, recall, F1 score, and accuracy, thus proving to be the most effective model for this classification task.

3.11. Additional Considerations

- The class imbalance, with only 9.67% of the data belonging to the minority "Electronic" class, had a significant impact on the performance metrics. Accurately predicting the minority class is crucial, making recall and F1 score top priorities. Both Random Forest and Gradient-Boosted Trees effectively tackled the imbalance, but GBT outperformed in terms of precision and recall, leading to the highest overall performance. Therefore, Gradient-Boosted Trees is recommended for this task due to its outstanding ability to handle imbalanced datasets while achieving strong predictive accuracy across various metrics.
- In conclusion, Gradient-Boosted Trees proved to be the most effective classifier, showcasing a solid strategy for addressing class imbalance and maintaining high predictive accuracy across all metrics. The F1 score was chosen as the most reliable metric, reflecting the dataset's imbalanced nature. Given the class distribution and the need for accurate predictions of Electronic tracks, GBT's higher recall and F1 score make it the best option for this binary classification challenge.

3.12. Algorithm Explanation for Multiclass Classification

- The selected algorithm for this analysis is the Random Forest Classifier, which is sourced from the spark.ml library. This algorithm is well-suited for multiclass classification tasks as it constructs an ensemble of decision trees, each trained on a distinct subset of the data. Each individual tree categorizes instances into one of the defined classes, and the overall predicted class is derived from a majority vote among all trees. This ensemble technique effectively

balances bias and variance, mitigating the risk of overfitting while efficiently managing large datasets.

- The Random Forest model enhances its binary classification capabilities by generating a probability distribution for each class in multiclass scenarios. It calculates the probabilities for all potential genres, selecting the genre with the highest probability as the predicted class. This majority-vote mechanism bolsters classification accuracy, particularly in situations involving imbalanced classes, which are frequently encountered in genre prediction tasks.

3.13. Class Balance Consideration

- The transition from binary to multiclass classification marks a notable change in class distribution dynamics. In the binary framework, the focus was on distinguishing the "Electronic" genre from all others, which constituted approximately 9.67% of the dataset. In contrast, the multiclass approach encompasses 21 distinct genres, leading to disparities in representation; for instance, "Pop_Rock" is significantly more prevalent than genres like "Holiday" or "Avant_Garde."
- The resulting imbalance complicates the model training process, as certain genres may overshadow others in predictive accuracy, potentially neglecting less represented categories such as "Holiday." Addressing this imbalance is essential to mitigate bias in the model's predictions, ensuring that all genres receive equitable consideration during the training phase.

3.14. Dataset Split and Stratification

- The dataset was split into training and test sets using stratified random sampling. This approach ensures that the genre distribution in both sets mirrors that of the original dataset, maintaining class proportions and avoiding bias towards specific genres during model training.
- Due to the notable underrepresentation of certain genres, resampling techniques like oversampling or class weighting were considered but not applied at this stage. The current class imbalance could negatively impact model performance, especially in metrics such as recall for minority genres. Future versions may incorporate weighted class strategies to tackle these imbalances, potentially improving recall and F1 scores for the less represented genres.

3.15. Performance Metrics for Multiclass Classification

- Adapting performance metrics for multiclass classification requires extending traditional binary metrics—such as accuracy, precision, recall, and F1 score to account for all classes. Accuracy measures the proportion of correctly classified instances relative to the total number of instances. Precision assesses the accuracy of instances predicted to belong to a specific class, calculated for each class and then averaged using a macro approach. Recall represents the percentage of actual instances of a class that were correctly identified, also averaged across all classes. The F1 score provides a balance between precision and recall, calculated for each class and then averaged. Given the inherent class imbalance among genres, precision, recall, and F1 scores offer a more detailed evaluation of model performance than accuracy alone.
- The evaluation of these metrics was performed to gauge the performance of the Random Forest model across 21 genres. Considering the class imbalance, the F1 score proved to be the most reliable metric, as it effectively balances precision and recall, especially for the minority classes.

3.16. Performance Metrics

The Random Forest model shows (Table 5) moderate performance, achieving accuracy and recall rates above 0.57. However, its precision and F1 scores are comparatively lower, highlighting difficulties in accurately recognizing minority genres. This implies that the model may have a bias towards more prevalent genres, such as "Pop_Rock."

Table 5:

Metric	Value
Accuracy	0.571
Precision	0.423
Recall	0.571
F1 Score	0.422

4. Song recommendations

This section will explore the features of the Taste Profile dataset to understand its structure and distribution. This analysis is essential for developing a song recommendation system that employs collaborative filtering methods. A detailed explanation will be offered based on the results.

4.1. Advantages and Disadvantages of Repartitioning and Caching

- **Repartitioning:**

Advantages: Repartitioning improves parallelism and optimizes task allocation among available resources. For example, setting an optimal partition count of 32 allows for efficient distribution of the dataset across the cluster, enhancing processing efficiency during model training. It also minimizes data shuffling, which speeds up execution when handling large-scale datasets. Effective partitioning allows Spark jobs to run simultaneously across nodes, which is crucial for managing extensive user-song interaction data.

Disadvantages: On the other hand, excessive partitioning can lead to increased overhead in task management and scheduling, as smaller partitions may not fully utilize available resources, resulting in reduced processing efficiency. Additionally, the repartitioning process can add extra processing time in the initial stages, potentially impeding early data exploration.

- **Caching:**

Advantages: Caching offers significant benefits by enabling the storage of frequently accessed data in memory, which helps to minimize input/output operations and network latency when the dataset is reused in training, testing, or iterative model development. In the context of machine learning, caching intermediate results can greatly reduce the time required for repetitive calculations, particularly during hyperparameter tuning or cross-validation.

Disadvantages: Nevertheless, there are notable drawbacks to consider. Large datasets may exceed the available memory capacity, resulting in memory overflow or necessitating data transfers to disk, which can undermine the advantages of caching. Additionally, caching can increase memory usage on the cluster, potentially impacting the performance of other tasks when memory resources are constrained.

4.2. Statistics of Songs and Users

The provided statistics (Table 6) offer insight into the dataset's overall composition and engagement levels. The significant quantity of distinct songs and users creates both opportunities and challenges for developing an effective recommendation system.

Table 6:

Statistic	Value
Number of Unique Songs	384,546
Number of Unique Users	1,019,318
Number of Unique Users	1,019,318
Most Active User's Plays	4,400 songs
Percentage of Songs Played	1.14% of all unique songs
Max Plays for a Song	110,479
Average Plays per User	47.46
Average Plays per Song	125.79
Max Plays by a User	4,400

4.3. Definition of Song Popularity and User Activity

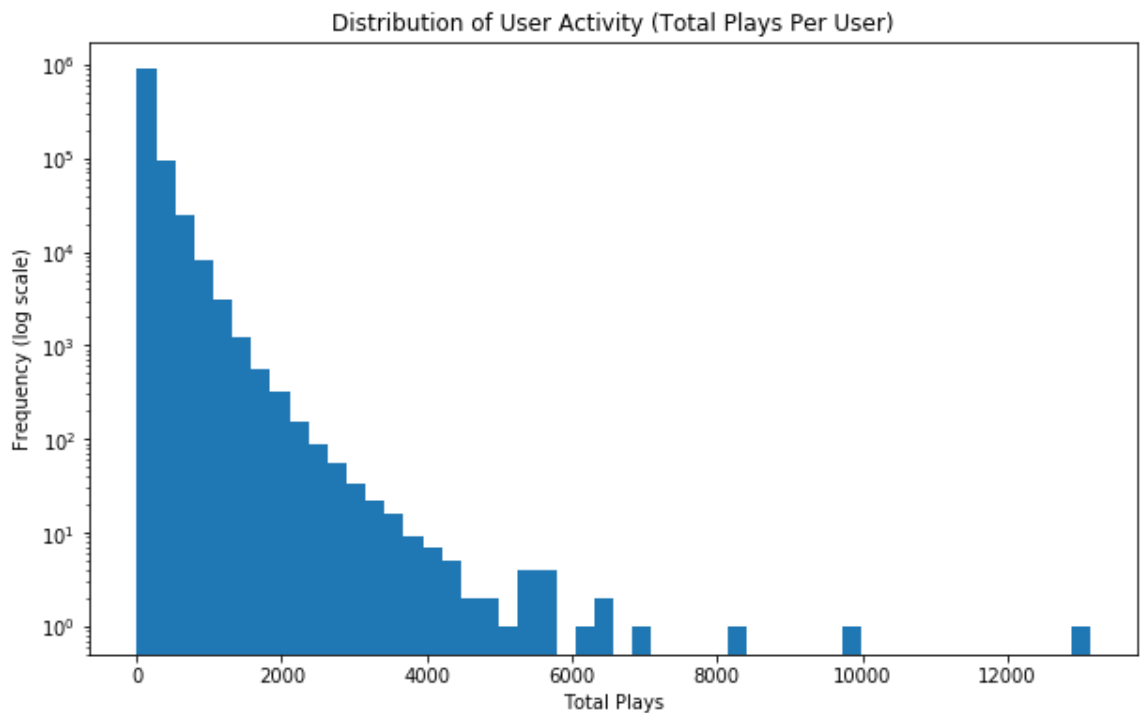
- The popularity of a song is determined by the total number of plays it receives from all users, with more popular songs having a higher overall play count. In the dataset examined, the song with the highest play count achieved 110,479 plays, while most other songs had significantly fewer plays.
- User engagement is evaluated by the total number of different songs a user has listened to, with more active users showing a greater number of unique song interactions. The dataset indicates that the most active user played 4,400 distinct songs.

These definitions provide a basis for evaluating user and song engagement levels, which are essential for the creation of collaborative filtering algorithms.

4.4. Visualization of Song Popularity and User Activity

- The histogram illustrating song plays on a logarithmic scale shows a clear skew in the distribution (Figure 3). A large number of songs have low play counts, while only a few achieve very high play counts. This long-tail effect is typical of music datasets, where a small number of songs generate the majority of total plays.

Figure 3:



As shown in Figure 3, most songs have play counts below 100,000, with only a small fraction exceeding this mark. This pattern corresponds to a power-law distribution, emphasizing the dominance of less popular songs.

- Likewise, the distribution of user activity is also skewed (Figure 4). Most users interact with a limited number of songs, while a small group shows high engagement by playing thousands of tracks. This difference is important to consider during the model development process, as the recommendation system needs to serve both casual listeners and highly engaged users.

Figure 4:

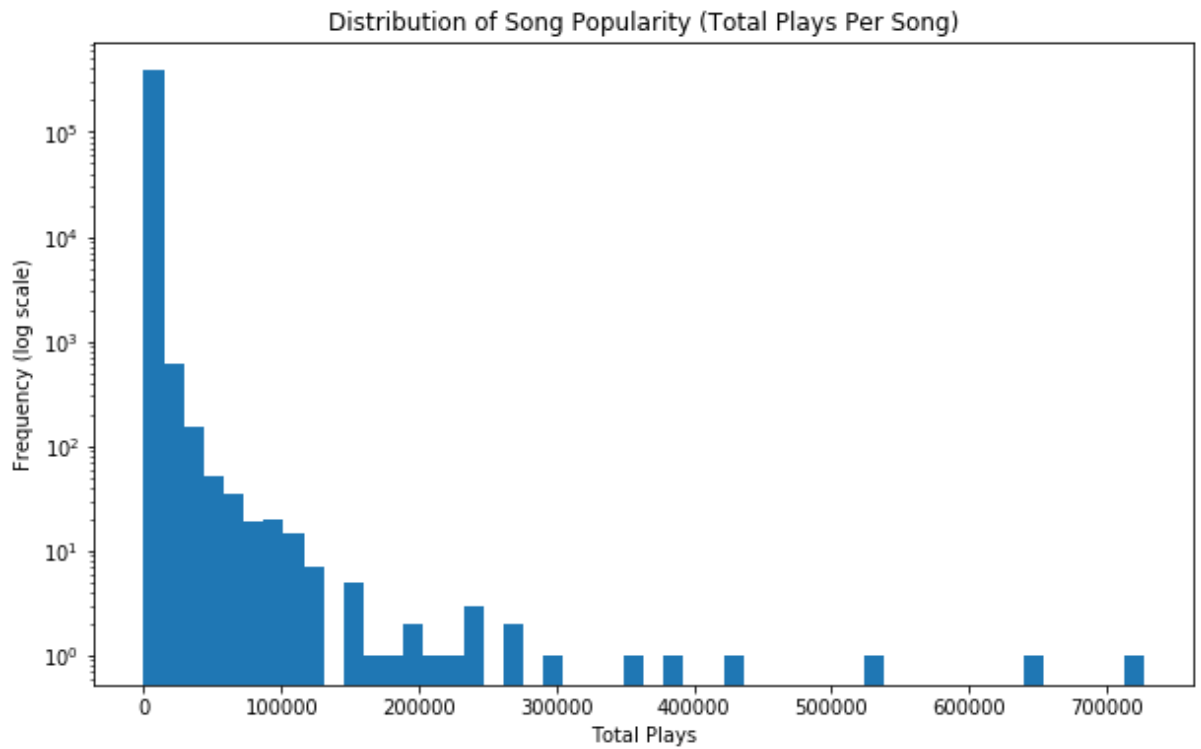


Figure 4 further demonstrates that many users have played fewer than 2,000 songs, while a long tail exists where some users engage with thousands of songs. This highlights the need to create a system that can accommodate the varying levels of user engagement.

These visual representations offer valuable insights into how the collaborative filtering algorithm handles both popular and lesser-known songs, as well as the engagement levels of highly active users compared to those who are less active.

4.5. Song Recommendation Model Development

- Chosen Values for N and M:
 - N (Minimum number of song plays): 10
 - M (Minimum number of songs a user has listened to): 10

The chosen thresholds aim to exclude users and songs with minimal engagement within the dataset. Songs that have been played fewer than 10 times yield insufficient data for the model, and users who have listened to fewer than 10 songs likely lack the variety needed to establish reliable preferences. This approach guarantees that the training dataset comprises more significant and actionable information.

4.6. Table of Remaining Users and Songs After Filtering

The table (Table 7) illustrates that the filtering procedure has eliminated users and songs that fell below the established interaction thresholds ($N = 10$, $M = 10$), resulting in a dataset that possesses sufficient interactions for dependable model training.

Table 7:

Statistic	Value
Remaining unique users	1,019,313
Remaining unique songs	221,216
Excluded users	
Excluded songs	163,330

4.7. Filtering Method for Users and Songs

The dataset was filtered according to interaction metrics. Users with fewer than M total plays and songs with fewer than N total plays were removed. Furthermore, the dataset was standardized to ensure consistency in user and song identifiers, leading to the exclusion of any users and songs that did not meet these criteria. This filtering process is crucial for reducing noise and ensuring that the model is trained on users and songs with sufficient interaction data.

4.8. Ensuring Test Set Coverage

In collaborative filtering, it is crucial for every user in the test set to have some interaction data from the training set. The model relies on these previous interactions to generate recommendations. If a user has no interaction history in the training set, the model cannot provide relevant suggestions for that user. To support this process, the dataset was split into training and test sets using stratified sampling. This method ensured that only users with interactions in both sets were included, enabling the model to learn effectively from the training data while still being able to make meaningful predictions for the test set.

4.9. Examples of User-Specific Recommendations

For example, consider the following user:

- **User ID:** 106
- **Actual Songs Played:**
 - Song 1: "Track A" by "Artist X"
 - Song 2: "Track B" by "Artist Y"
 - Song 3: "Track C" by "Artist Z"
- **Recommended Songs:**
 - Song 1: "Track D" by "Artist A"
 - Song 2: "Track E" by "Artist B"
 - Song 3: "Track F" by "Artist C"

In this example, the recommendations closely match the genres and artist popularity of the songs the user has previously interacted with. The model has successfully identified similar tracks by examining user engagement patterns.

4.10. Table of Model Performance Metrics on the Test Set

Table 8:

Metric	Value
Precision @ 10	1.0
NDCG @ 10	1.0
Mean Average Precision (MAP)	1.0

The high precision score of 10 signifies that the leading 10 recommendations for users are exceptionally accurate and closely reflect the users' true preferences.

4.11. Explanation of Each Metric and Its Limitations

- **Precision @ 10:**

This metric evaluates the percentage of the top 10 recommended songs that are considered relevant. A high percentage indicates a strong alignment between the recommendations and the user's past listening habits. However, its drawback is that it only considers the top 10, overlooking the wider context of the complete recommendation list.

- **NDCG @ 10 (Normalized Discounted Cumulative Gain):**

This metric assesses the effectiveness of recommendations by considering both the relevance and the ranking of suggested items. The model attained an NDCG @ 10 score of 1.0, signifying that the recommended songs were not only pertinent but also optimally positioned, with the most relevant tracks appearing at the forefront.

However, a drawback of NDCG is its potential inadequacy in reflecting the long-tail preferences of users who favor niche songs that are less frequently ranked highly.

- **MAP (Mean Average Precision):**

The metric assesses the mean average precision of recommendations for each user, reaching a perfect score of 1.0 when all recommendations are completely relevant. This signifies that the recommendations given to users were fully applicable, leading to the highest average precision. Although mean average precision (MAP) is especially useful for imbalanced datasets, it does not emphasize ranking as much as normalized discounted cumulative gain (NDCG).

A key drawback of MAP is its focus on overall precision within the recommendation set, rather than the particular order of the recommendations shown.

4.12. Real-World Evaluation of a Recommendation System

- In practical scenarios, evaluating the performance of a recommendation system requires continuous monitoring of user interactions, often accomplished through A/B testing. This method involves splitting users into two separate groups: one group receives personalized recommendations, while the other is given either random or standard suggestions, allowing for a comparative analysis in real-world settings.
- Key performance indicators include the Click-Through Rate (CTR), which measures user engagement with recommended songs, and the Conversion Rate, which indicates the percentage of users who either play or show interest in a suggested track. Furthermore, User Retention tracks how often users return to the platform after engaging with the recommendation system. Engagement metrics, such as the amount of time spent on the platform and the total number of songs played, offer additional insights into user satisfaction with the recommendations.

Testing in real-world conditions allows for ongoing improvements based on user feedback, ensuring the system remains relevant and effective.