Exploring Music Recommendations: A HetRec Last.FM Dataset Analysis

Introduction to Recommender Systems and Their Main Applications:

The rapid growth of digital content and the ever-increasing volume of data generated in the digital age have resulted in an overwhelming amount of information, which makes it challenging for individuals to find pertinent and useful data across various platforms (Roy and Dutta, 2022). To address this issue, recommender systems have emerged as a crucial application of artificial intelligence (AI), aiming to analyse user behaviour and preferences to offer personalised recommendations for items such as products, services, movies, or music (Portugal et al., 2018).

Recommender systems have become indispensable tools across a range of industries, particularly retail, media, and entertainment, where they can considerably reduce users' search time and facilitate the discovery of items that correspond to their interests. These systems are especially advantageous for businesses with extensive product or service lines, as a customised approach can enhance user satisfaction and increase the probability of conversions.

Recommender systems have various primary applications that serve different industries and user needs. In the realm of e-commerce, online retailers can employ recommender systems to suggest products that customers are likely to purchase based on their browsing history, previous purchases, and preferences. This approach can enhance the overall shopping experience while boosting sales (Linden et al., 2003).

In the media and entertainment sector, streaming services and content platforms can leverage recommender systems to recommend movies, TV shows, and music tailored to users' tastes. This personalisation helps maintain engagement and heightens user satisfaction with the platform (Gomez-Uribe and Hunt, 2016).

News and information platforms can benefit from recommender systems by presenting articles and news stories that resonate with users' interests. This process simplifies users' access to pertinent information and keeps them informed about topics that matter to them (Lerman and Hogg, 2010).

Social networking platforms can utilise recommender systems to suggest new connections, groups, or events that users might find appealing. This strategy fosters community engagement and stimulates the growth of users' networks (Guy et al., 2010).

Lastly, job and career services can adopt recommender systems to assist job seekers in finding employment opportunities that align with their skills and preferences. Concurrently, employers can use these systems to pinpoint potential candidates who are a suitable fit for their organisation (Malinowski et al., 2006).

In this project, we will delve into the topic of recommender systems, concentrating on their diverse applications and the various approaches employed in developing these systems, including collaborative filtering and deep learning techniques.

Challenges in Recommender Systems:

Recommender systems have become indispensable tools for providing personalised suggestions to users based on their preferences and past interactions. Despite their growing importance, the development and implementation of these systems present several challenges that must be addressed to ensure their effectiveness and accuracy (Neelam Tyagi, 2019).

The cold start problem, for instance, arises when a recommender system encounters a new user for the first time and lacks historical data on their preferences. In this scenario, generating personalised recommendations, essential for a satisfying and engaging user experience, becomes difficult (Neelam Tyagi, 2019). Additionally, data sparsity is another challenge that occurs when users rate or review only a limited number of items, resulting in a sparse user-item matrix. This sparsity hampers the system's ability to effectively identify similar users or items, consequently impacting the quality of recommendations (Neelam Tyagi, 2019).

Furthermore, synonymy, where similar items have different names, titles, or descriptions, can pose a challenge for a recommender system in recognising that distinct entries represent the same or closely related items. This can lead to potential inaccuracies in recommendations (Neelam Tyagi, 2019). Lastly, scalability emerges as a significant concern when recommender systems process vast amounts of user-item interaction data, including ratings, reviews, and other implicit feedback. Ensuring the efficient handling of large-scale, real-world datasets while

maintaining the system's performance is essential for the success of any recommender system (Neelam Tyagi, 2019).

Developing a robust recommender system involves addressing and overcoming these issues, which is vital for delivering accurate and personalised recommendations, ultimately contributing to enhanced user satisfaction and engagement across various platforms and industries.

Comparative Analysis of Approaches:

Recommender systems have become an essential part of information filtering, addressing the problem of information overload by filtering vital information fragments from large amounts of dynamically generated data according to user's preferences, interests, or observed behaviour towards items (Pan, C. and Li, W., 2010). This section compares three main approaches of recommender systems: content-based filtering (CBF), collaborative-based filtering (CF), and hybrid-based filtering techniques. These methods are compared based on their definitions, basis of recommendation, advantages, disadvantages, and similarities.

Comparison based on Definition:

- CBF: This approach is a domain-dependent algorithm that emphasises the analysis of items' attributes or features to generate predictions (Lops et al., 2011). Recommendations are made based on a single user's behaviour and data, with item descriptions and user profiles playing crucial roles. CBF is commonly used in recommending documents, such as web pages, news articles, and restaurants (Isinkaye et al., 2015).
- CF: This is a domain-independent prediction technique for content that cannot be easily and adequately described by metadata, such as movies and music (Isinkaye et al., 2015). Recommendations are made based on a user's behaviour and the behaviour of other similar users. CF relies on the idea that users who have agreed in the past will likely agree again in the future (Schafer et al., 2007).
- Hybrid-based filtering techniques: This approach combines different recommendation methods aiming to create a superior recommendation system that overcomes the limitations of CBF and CF techniques (Burke, 2002). Hybrid techniques leverage the

strengths of individual algorithms to provide more accurate and effective recommendations.

Comparison based on Concepts or Techniques of Recommendation:

- CBF: Recommendations are made based on user profiles using attributes taken from the content of previously evaluated items (Bobadilla et al., 2013). Items closely related to positively rated items are recommended. CBF uses various models to find similarities between documents, such as Vector Space models like Term Frequency Inverse Document Frequency (TF/IDF) (Salton and Buckley, 1988) or probabilistic models like Naïve Bayes Classifier, Decision Trees, or Neural Networks (Pazzani and Billsus, 2007).
- CF: This creates a database (user-item matrix) of preferences for items by users and calculates similarities between user profiles to make recommendations (Konstan and Riedl, 2012). It groups users with similar interests and preferences, forming neighbourhoods. CF algorithms are classified into memory-based filtering techniques (user and item-based filtering) and model-based filtering techniques, such as dimensionality reduction (e.g., Singular Value Decomposition (SVD)), clustering (e.g., k-means), decision trees, and association rule mining (Bobadilla et al., 2013).
- Hybrid-based filtering techniques: These techniques combine multiple algorithms to provide more accurate and effective recommendations, overcoming the limitations of individual algorithms (Burke, 2002). Examples of hybrid systems include Weighted, Mixed, Switching, Feature Combination, Feature Augmentation, and Cascade (Burke, 2007).

Comparison based on Advantages, Disadvantages, and Similarities:

Table 1: Comparison of Content-based, Collaborative-based, and Hybrid-based Filtering Techniques.

Content-based filtering	**Collaborative-based filtering**	**Hybrid-based filtering**
Can recommend new items without user rations; databases without user preferences do not affect recommendation accuracy (Pazzani and Billsus, 2007)	Cannot work effectively without adequate information about a user or an item, leading to the cold-start problem (Burke, 2007)	Overcomes the cold-start problem by combining CBF and CF techniques (Burke, 2002)
Useful when users do not share the same items, but only identical items according to their intrinsic features (Lops et al., 2011)	Struggles to distinguish between closely related items due to synonymy (Patel et al., 2017)	Overcomes synonymy problems by leveraging both CBF and CF techniques (Burke, 2002)
Based on a single user's behaviour and preferences (Lops et al., 2011)	Based on multiple users with similar interests or preferences (Schafer et al., 2007)	Combines both CBF and CF approaches (Burke, 2002)
Cannot perform in domains without content associated with items (Pazzani and Billsus, 2007)	Can perform in domains with little or no content associated with items (Schafer et al., 2007)	Can perform in domains with little content associated with items by combining different techniques (Burke, 2002)
Cannot recommend items outside the user's profile or database (Pazzani and Billsus, 2007)	Can provide serendipitous recommendations, recommending relevant items even if not in the user's profile (Schafer et al., 2007)	Can recommend items outside the user's profile by leveraging both CBF and CF techniques (Burke, 2002)
Affected by sparsity due to the small amount of information (Bobadilla et al., 2013). Also affected by sparsity (Bobadilla et al., 2013)	Overcomes the sparsity problem by combining different techniques (Burke, 2002)	Dependent on items' metadata, requires a rich description of items and a well-organised user profile (Pazzani and Billsus, 2007)
Not affected by limited content analysis (Schafer et al., 2007)	Not affected by limited content analysis by combining different techniques (Burke, 2002)	Content over-specialisation restricts users to recommendations similar to items in their profiles (Zhang et al., 2002)
Lower complexity than hybrid-based techniques (Lops et al., 2011)	Lower complexity than hybrid-based techniques (Schafer et al., 2007)	Increased complexity due to the combination and optimisation of different approaches (Burke, 2002)

CBF, CF, and hybrid-based filtering techniques each have their strengths and weaknesses. CBF is more suitable for domains with rich item descriptions, while CF can handle domains with little or no metadata. Hybrid systems aim to combine the advantages of both CBF and CF, overcoming limitations such as the cold start problem, synonymy, sparsity, and content overspecialisation.

While CBF and CF can provide accurate recommendations, they may struggle in some situations, such as when there is limited information about users or items. Hybrid systems address these challenges by integrating multiple algorithms, offering improved performance and versatility. However, the complexity of hybrid systems can be higher due to the need to combine and optimise different approaches.

When selecting a recommender system, it is crucial to consider the specific domain, the availability of metadata, and the desired level of recommendation accuracy and serendipity. In many cases, hybrid systems offer the most robust and flexible solution, leveraging the strengths of multiple approaches to deliver effective recommendations across a range of domains.

Selected Approach: In-Depth Exploration:

To address the problem, we recommend a hybrid recommender system that combines CF and deep learning, enhancing recommendation performance. This approach utilises Non-negative Matrix Factorisation (NMF) for CF and a deep learning model developed using Keras (Richter & Provost, 2013; Koren, Bell, & Volinsky, 2009).

NMF is an unsupervised learning technique that decomposes a non-negative user-item matrix into two lower-dimensional non-negative matrices, representing latent factors (Lee & Seung, 2001). These latent factors correspond to hidden patterns in user preferences and item features, enabling a more accurate understanding of users' interests. NMF is particularly useful for handling sparse data, common in recommender systems, by reducing dimensionality and capturing the underlying structure in the data (Xu, Liu, & Gong, 2003).

The deep learning model is built on Keras and TensorFlow, employing user and item embeddings to represent users and items in a continuous vector space. Embeddings capture complex relationships and similarities, contributing to more precise recommendations (Chollet, 2017). The model incorporates ReLU activation functions, which allow for efficient training and

mitigate the vanishing gradient problem (Nair & Hinton, 2010). Dropout layers are also included for regularisation, preventing overfitting, and improving model generalisation (Srivastava et al., 2014).

The hybrid recommender system offers several advantages. First, it captures latent factors and intricate user-item interactions, resulting in more accurate recommendations (Richter & Provost, 2013). By combining NMF's ability to uncover latent factors and deep learning's capacity to model complex interactions, the hybrid approach provides a comprehensive understanding of user preferences.

Second, the hybrid system addresses the cold start problem, which occurs when a recommender system struggles to make relevant suggestions for new users or items due to insufficient data (Schein et al., 2002). NMF mitigates this issue by recommending new users or items based on the user-item matrix, while the deep learning model leverages learned embeddings to infer user preferences even when data is sparse.

Third, the hybrid model's flexibility enables the incorporation of additional features or the integration of other CF and deep learning techniques. This adaptability makes the hybrid approach applicable to various domains and allows for continuous improvement as new techniques and features are developed (Burke, 2007).

Finally, the hybrid recommender system is well-suited for large-scale, high-dimensional, and sparse data, aligning with the characteristics of modern recommender systems (Ricci, Rokach, & Shapira, 2015). By efficiently handling such data, the hybrid approach can be applied to a wide range of scenarios where traditional recommender systems might struggle.

The hybrid recommender system, which combines CF and deep learning, offers a powerful and flexible solution for providing personalised recommendations. Its ability to capture latent factors and complex user-item interactions, address the cold start problem, adapt to various domains, and handle large-scale, high-dimensional, and sparse data make it an ideal approach for solving our problem.

Problem-Solving with a Publicly Available Dataset:

Dataset Overview: HetRec Last.fm Dataset: https://grouplens.org/datasets/hetrec-2011/

The HetRec Last.fm dataset is a comprehensive collection designed for developing and evaluating recommender systems, offering rich information for implementing CF, content-based recommendations, and hybrid strategies (Cantador et al., 2011). The dataset includes six interconnected components:

1. User_artists (92,834 rows, 3 columns): Captures users' listening history, essential for constructing user-profiles and formulating recommendations based on user listening patterns.
2. User_friends (25,434 rows, 2 columns): Represents Last.fm users' social network, enabling social network-based recommendations and CF algorithms that leverage users' social connections.
3. Artists (17,632 rows, 4 columns): Contains data on artists and MusicBrainz identifiers, allowing for additional metadata retrieval to enrich the dataset and facilitate content-based recommendations or hybrid strategies.
4. Tags (11,946 rows, 2 columns): Comprises unique tags reflecting artist attributes, providing valuable information for content-based recommendations or augmenting CF algorithms.
5. User_taggedartists (186,479 rows, 6 columns): Establishes relationships between users, artists, and tags, enabling the recommender system to improve recommendations by incorporating user-generated metadata.
6. User_taggedartist-timestamps (186,479 rows, 4 columns): Offers timestamp information for user-artist-tag associations, allowing for time-aware recommendations and the analysis of temporal dynamics in user preferences and tagging behaviour.

Pre-processing and Data Cleaning Techniques:

Several pre-processing and data-cleaning steps were applied to ensure the quality and reliability of the dataset. Each pre-processing technique provides specific benefits to improve the performance of the recommender system (Zhang et al., 2016).

Handling missing values is a critical step in the data pre-processing pipeline for data science projects, as it significantly affects data quality, biases, model performance, and generalisability (Little & Rubin, 2019). In the current dataset, only missing values were in the pictureURL column of the Artists' dataframe. However, this was ignored as it is relevant to our recommender system. (Figure 1).

```
# Iterate over each dataframe in the dataset
for name, df in dataframes.items():
    # Print the name of the dataframe
    print("Dataframe Name: ", name)
    # Print the total number of missing values in each column
    print(df.isnull().sum())
    # Add a blank line for readability
    print()

```

```
Dataframe Name:  User-Artist Listening History
userID      0
artistID    0
weight      0
dtype: int64

Dataframe Name:  User-Friends
userID      0
friendID    0
dtype: int64

Dataframe Name:  User-Tagged Artists
userID      0
artistID    0
tagID       0
day         0
month       0
year        0
dtype: int64

Dataframe Name:  Artists
id             0
name           0
url            0
pictureURL    444
dtype: int64

Dataframe Name:  Tags
tagID       0
tagValue    0
dtype: int64

Dataframe Name:  User-Tagged Artists Timestamps
userID       0
artistID     0
tagID        0
timestamp    0
dtype: int64
```

Figure 1: Summarising the missing values present in each column of the dataframes. The output highlights the number of missing values in each column.

Dealing with duplicate values is essential for preventing biases in the analysis, improving data quality, enhancing model performance, and maintaining data integrity (Zhang et al., 2016). Our dataset did not contain duplicate values (Figure 2).

```python
# Iterate over each dataframe in the dataset
for name, df in dataframes.items():
    # Print the name of the dataframe
    print("Dataframe Name: ", name)
    # Print the number of duplicate values
    print("Number of duplicate values: ", df.duplicated().sum())
    # Add a blank line for readability
    print()

```

```
Dataframe Name:  User-Artist Listening History
Number of duplicate values:  0

Dataframe Name:  User-Friends
Number of duplicate values:  0

Dataframe Name:  User-Tagged Artists
Number of duplicate values:  0

Dataframe Name:  Artists
Number of duplicate values:  0

Dataframe Name:  Tags
Number of duplicate values:  0

Dataframe Name:  User-Tagged Artists Timestamps
Number of duplicate values:  0
```

Figure 2: Summarising the number of duplicate values present in each dataframe. The output highlights the presence of duplicate entries in each dataframe.

Renaming columns contributes to data readability, consistency, and compatibility while merging data frames facilitates a comprehensive understanding of the data and enables complex analyses (Wickham, 2014). We ensured that column names were clear and merged relevant data frames accordingly (Figure 3).

```python
# Rename the 'weight' column to 'userPlays'
user_artist_df = user_artist_df.rename(columns={'weight': 'userPlays'})

# Rename the 'id' column to 'artistID'
artists_df = artists_df.rename(columns={'id': 'artistID'})

# Select the desired columns from each dataframe
user_artist_plays = user_artist_df.loc[:, ['userID', 'artistID', 'userPlays']]
artists = artists_df.loc[:, ['artistID', 'name']]

# Merge the two dataframes on the 'artistID' column
user_artist_plays = pd.merge(user_artist_plays, artists, on='artistID')

# Select only the desired columns
user_artist_plays = user_artist_plays.loc[:, ['userID', 'artistID', 'name', 'userPlays']]

# Print the first few rows of the resulting dataframe
print(user_artist_plays.head())

```

	userID	artistID	name	userPlays
0	2	51	Duran Duran	13883
1	4	51	Duran Duran	228
2	27	51	Duran Duran	85
3	28	51	Duran Duran	10
4	62	51	Duran Duran	528

Figure 3: Renaming columns and merging the user_artist_df and artists_df dataframes. The output displays the first few rows of the resulting user_artist_plays dataframe, which contains userID, artistID, name, and userPlays columns, facilitating a better understanding of user listening behaviour and artist popularity.

Selecting desired columns streamlines the dataset, reduces memory usage, and enhances the performance of operations on the dataset (Few, 2009). We focused on features pertinent to our analysis (Figure 3) discussed in section 5c.

Applying thresholds for minimum plays per user, minimum unique artist plays, the minimum artist plays, and minimum listeners per artist, we were able to filter out inactive users, unpopular artists, and unplayable tracks (García et al., 2015). This process reduced noise in the data and improved the accuracy of recommendations. After applying the thresholds, the number of unique users decreased by 1.48%, while unique artists decreased by 83.96% (Figure 4).

```python
# Define the thresholds for filtering the dataset
min_plays_per_user = 50  # minimum plays per user
min_unique_artists = 5  # minimum unique artist plays
min_plays_per_artist = 50  # minimum plays per artist
min_listeners_per_artist = 10  # minimum listeners per artist

# Group the data by user and artist, and count the number of plays for each user-artist combination
grouped_plays = user_artist_plays.groupby(['userID', 'artistID', 'name']).agg({'userPlays': 'sum'}).reset_index()

# Filter out users with fewer than min_plays_per_user total plays
user_counts = grouped_plays.groupby('userID').agg({'userPlays': 'sum'}).reset_index()
valid_users = user_counts[user_counts['userPlays'] >= min_plays_per_user]['userID']

# Filter out artists with fewer than min_unique_artists unique listeners and fewer than min_plays_per_artist total plays
artist_counts = grouped_plays.groupby('artistID').agg({'userID': 'nunique', 'userPlays': 'sum'}).reset_index()
valid_artists = artist_counts[(artist_counts['userID'] >= min_unique_artists) &
                              (artist_counts['userPlays'] >= min_plays_per_artist)]['artistID']

# Filter out plays by users or artists that did not meet the above criteria
valid_plays = grouped_plays[grouped_plays['userID'].isin(valid_users) & grouped_plays['artistID'].isin(valid_artists)]

# Filter out artists with fewer than min_listeners_per_artist listeners
valid_artist_plays = valid_plays.merge(artist_counts[artist_counts['userID'] >= min_listeners_per_artist][['artistID']],
                                       on='artistID')

# Select only the desired columns for the final dataset
user_artist_plays_thr = valid_plays.loc[:, ['userID', 'artistID', 'name', 'userPlays']]

```

```
Number of unique users before applying thresholds:  1892
Number of unique artists before applying thresholds:  17632

Number of unique users after applying thresholds:  1864
Number of unique artists after applying thresholds:  2828

Percentage difference in number of unique users:  -1.48 %
Percentage difference in number of unique artists:  -83.96 %
```

Figure 4: Filtering out users and artists based on defined thresholds. The table displays the top few rows of the filtered user_artist_plays_thr dataframe, which includes userID, artistID, name, and userPlays columns. The code also calculates and prints the percentage difference in the number of unique users and artists before and after applying the thresholds.

Adding columns for data visualisation, such as total unique users, total artist plays, the average user plays per artist, total unique artists, and total user plays, provided more context to the data, and facilitated a better understanding of user behaviour (Figure 5) (Few, 2009). These additional features offered valuable insights, enabling outlier detection and error identification, ultimately leading to more accurate recommendations and improved user experiences.

```python
# Calculate total unique users and add as column to dataframe
total_unique_users = user_artist_plays_thr.groupby('artistID')['userID'].nunique().reset_index(name='totalUniqueUsers')
user_artist_plays_thr = user_artist_plays_thr.merge(total_unique_users, on='artistID')

# Aggregate play count of all users of a single artist
total_artist_plays = user_artist_plays_thr.groupby('artistID')['userPlays'].sum().reset_index(name='totalArtistPlays')
user_artist_plays_thr = user_artist_plays_thr.merge(total_artist_plays, on='artistID')

# Calculate the average number of times a user has played an artist
user_artist_plays_thr['avgUserPlaysPerArtist'] = user_artist_plays_thr['totalArtistPlays'] / user_artist_plays_thr['totalUniqueUsers']

# Calculate total unique artists and add as column to dataframe
total_unique_artists = user_artist_plays_thr.groupby('userID')['artistID'].nunique().reset_index(name='totalUniqueArtists')
user_artist_plays_thr = user_artist_plays_thr.merge(total_unique_artists, on='userID')

# Aggregate plays of all artist by a single user
total_user_plays = user_artist_plays_thr.groupby('userID')['userPlays'].sum().reset_index(name='totalUserPlays')
user_artist_plays_thr = user_artist_plays_thr.merge(total_user_plays, on='userID')
```

	userID	artistID	name	userPlays	totalUniqueUsers	totalArtistPlays	avgUserPlaysPerArtist	totalUniqueArtists	totalUserPlays	hasUserPlayed
0	2	51	Duran Duran	13883	110	348915	3171.954545	41	148017	1
1	2	52	Morcheeba	11690	23	18787	816.826087	41	148017	1
2	2	53	Air	11351	75	44230	589.733333	41	148017	1
3	2	54	Hooverphonic	10300	18	15927	884.833333	41	148017	1
4	2	55	Kylie Minogue	8983	296	449290	1517.871622	41	148017	1

Figure 5: The output displays the top few rows of the user_artist_plays_thr dataframe after adding various aggregated columns. The code computes the total unique users per artist, total artist plays, average user plays per artist, total unique artists per user, and total user plays.

Feature Extraction and Attribute Selection:

Feature selection is vital for building an effective recommender system, as it helps identify relevant attributes for generating accurate and personalised recommendations. In our application, we have selected the following features: UserID, artistID, name, and userPlays (Ricci et al., 2015; Lops et al., 2011; Jannach et al., 2010).

UserID allows tracking of individual users' listening habits and preferences, enabling the system to identify similar users and provide personalised recommendations through CF techniques. ArtistID is essential for content-based recommendations, as it allows suggesting items that share similarities with a user's past interests and identifying relationships among artists.

The name feature provides the artist's name, offering context and disambiguation for artists with similar or identical names, resulting in more accurate and personalised recommendations. UserPlays indicates the number of times a user has listened to a particular artist, reflecting the popularity, and helping identify trending artists. This information allows the recommender system to balance personalisation and general appeal.

By incorporating these features, our recommender system can effectively identify similar users, suggest items based on past interests, differentiate between artists with similar names, and

recommend popular music that appeals to a wide range of users, ultimately enhancing the user experience.

Data Visualisation Techniques:

Leveraging libraries like Seaborn and Matplotlib, we created various visualisations to better understand the user and artist behaviour. To identify popular artists, crucial for personalised recommendations and marketing strategies in recommender systems, we employed a bar chart (Figure 6) displaying the top 20 artists ranked by total plays (Ricci, Rokach, & Shapira, 2015).

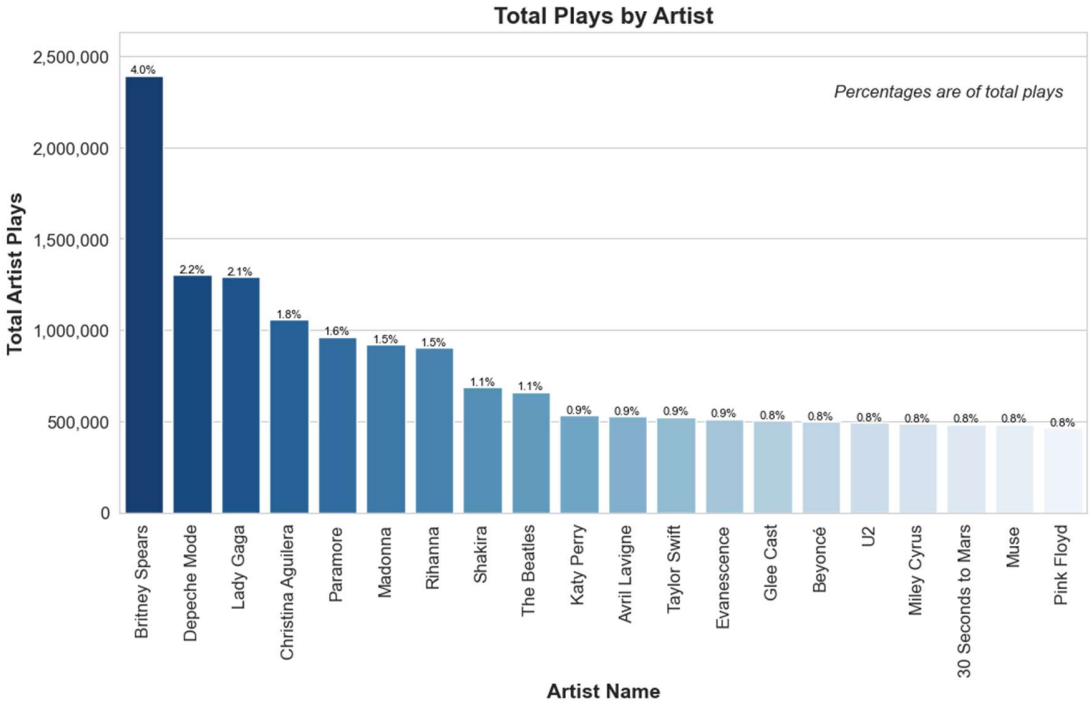

Figure 6: Bar chart representing the total plays by artist, sorted in descending order. The percentages displayed above each bar represent the proportion of the artist's total plays in relation to the overall plays in the dataset.

Insights into artist popularity and reach were obtained through a graph (Figure 7) that depicted the number of unique users per artist, which can be used to generate recommendations based on user preferences and diversify artist exposure (Jannach, Zanker, Felfernig, & Friedrich, 2010).

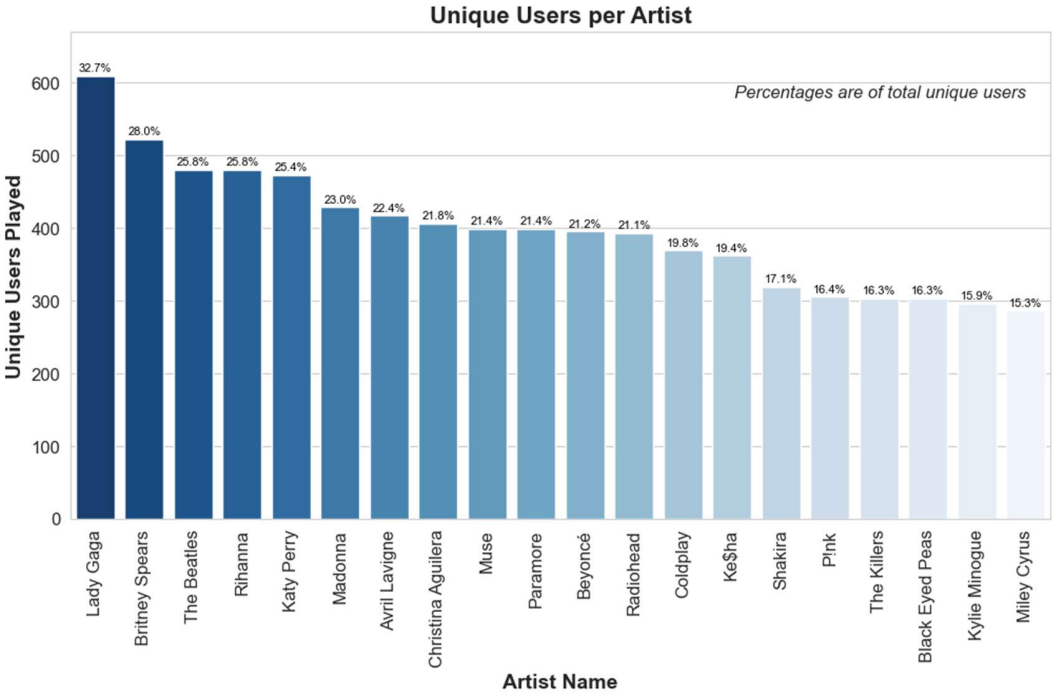

Figure 7: Bar chart representing the number of unique users who played each artist, focusing on the top 20 artists, sorted in descending order. The percentages displayed above each bar represent the proportion of unique users who played the artist in relation to the total unique users in the dataset.

Understanding user listening behaviour, personalising recommendations, and targeting high-volume listeners for marketing efforts can be achieved by analysing a graph (Figure 8) showing the total plays per user (Lops, De Gemmis, & Semeraro, 2011).

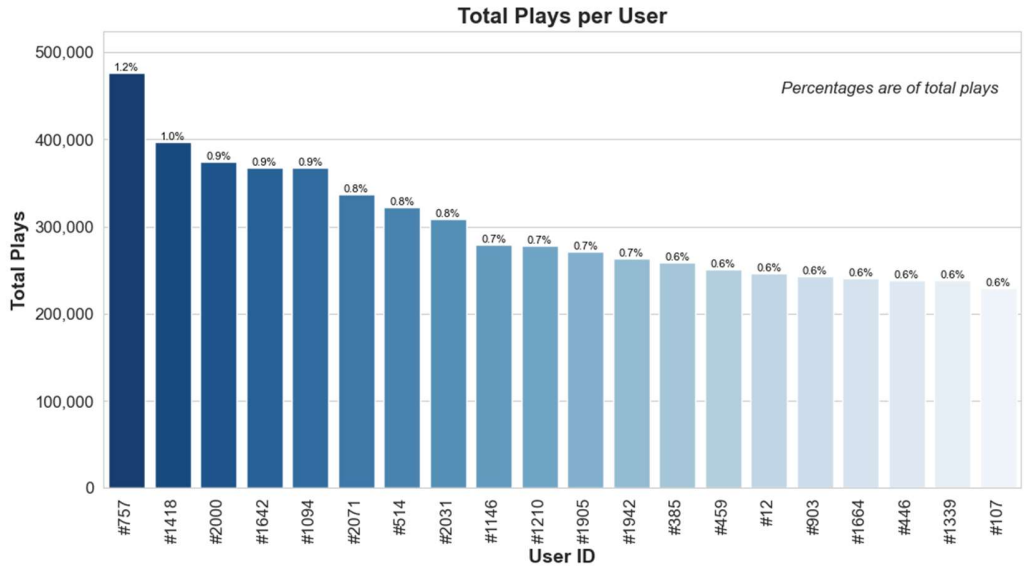

Figure 8: Bar chart representing the total plays per user, focusing on the top 20 users, sorted in descending order. The percentages displayed above each bar represent the proportion of total plays by each user in relation to the total plays in the dataset. Each user is identified by their unique User ID, preceded by a '#'.

Another graph (Figure 9), displaying the top artists ranked by average user plays per artist, is useful for identifying popular artists and informing marketing strategies. This metric underscores artists that strongly appeal to their listeners, contributing to engaging recommendations.

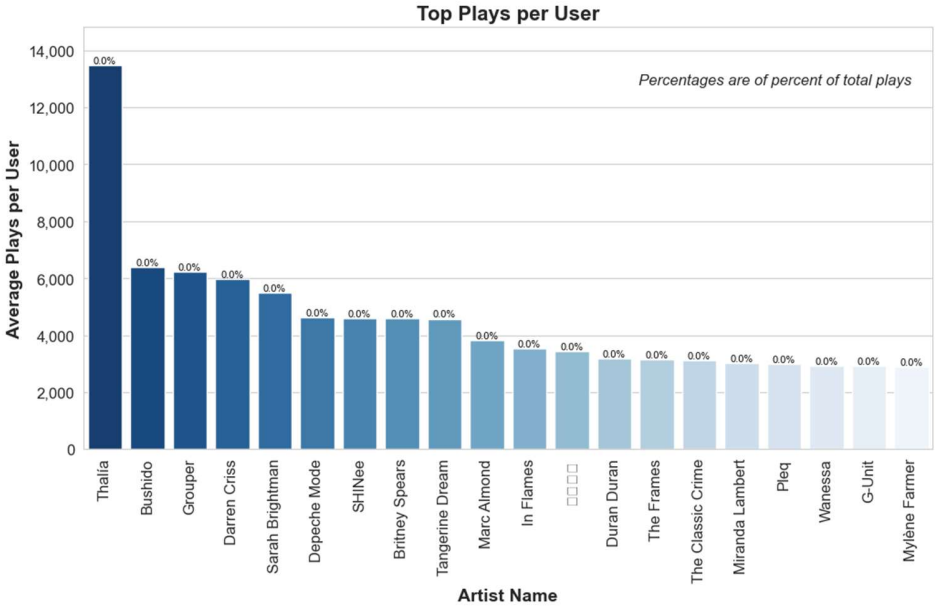

Figure 9: Bar chart representing the average plays per user for each artist, focusing on the top 20 artists with the highest average plays per user, sorted in descending order. The percentages displayed above each bar represent the proportion of total plays by each artist in relation to the total plays in the dataset. Each artist is identified by

Enhancing the interpretability of visualisations and focusing on the most relevant artists is achieved by limiting the chart to popular artists with more than 50 unique listeners, ultimately improving recommendation accuracy and relevance (Figure 10).

To inform recommendations and identify outliers or niche artists appealing to specific user segments, we utilised a scatter plot (Figure 11) examining the relationship between artist popularity (total plays) and unique listeners.

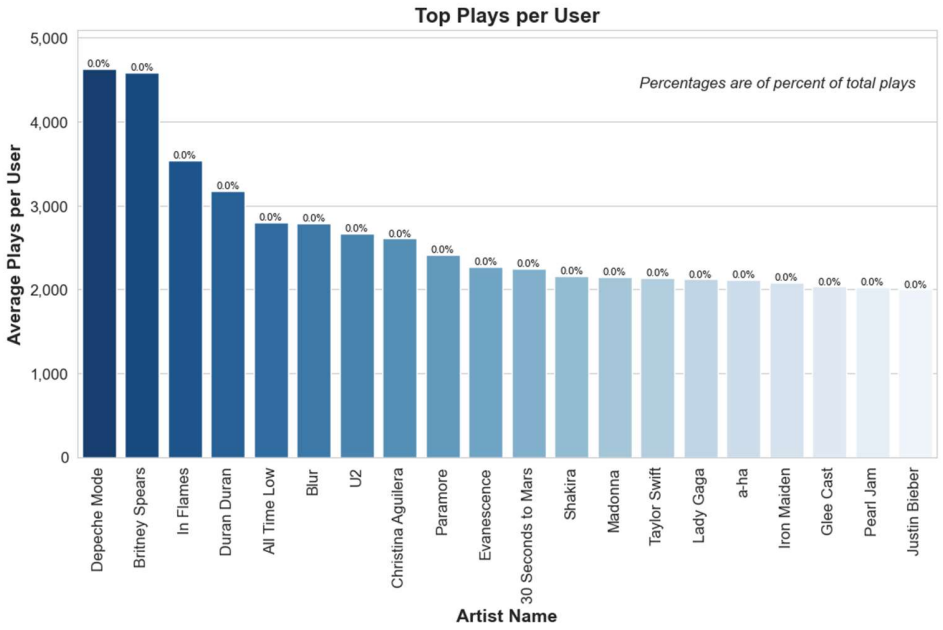

Figure 10: Bar chart representing the average plays per user for each artist, focusing on the top 20 artists with the highest average plays per user and over 50 unique listeners, sorted in descending order. The percentages displayed above each bar represent the proportion of total plays by each artist in relation to the total plays in the filtered dataset. Each artist is identified by their name.

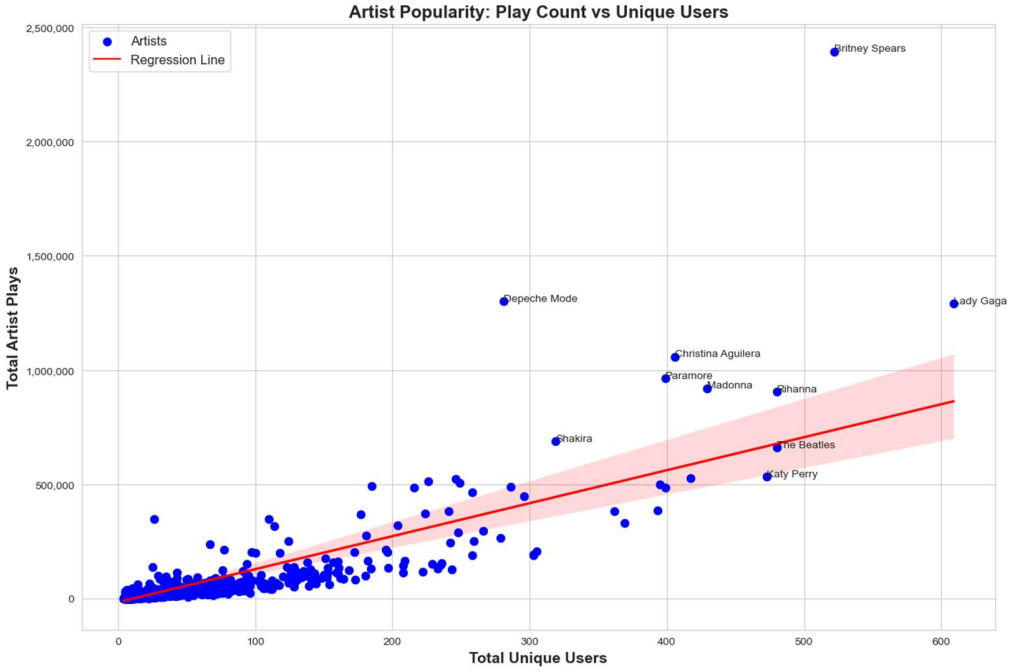

Figure 11: Scatter plot illustrating the relationship between the total unique users and total artist plays for each artist, with a red regression line. The top 10 artists with the highest total artist plays are annotated with their names on the plot. Each artist is represented by a blue circle marker. The plot examines the correlation between the artist's popularity in terms of play count and the number of unique users who have played the artist's songs.

A density plot (Figure 12), illustrating the distribution of total artist plays, offers a clear view of the overall popularity landscape, valuable for targeted marketing, personalised recommendations, and outlier detection. This visualisation helps identify patterns that can be leveraged to improve recommender system performance.

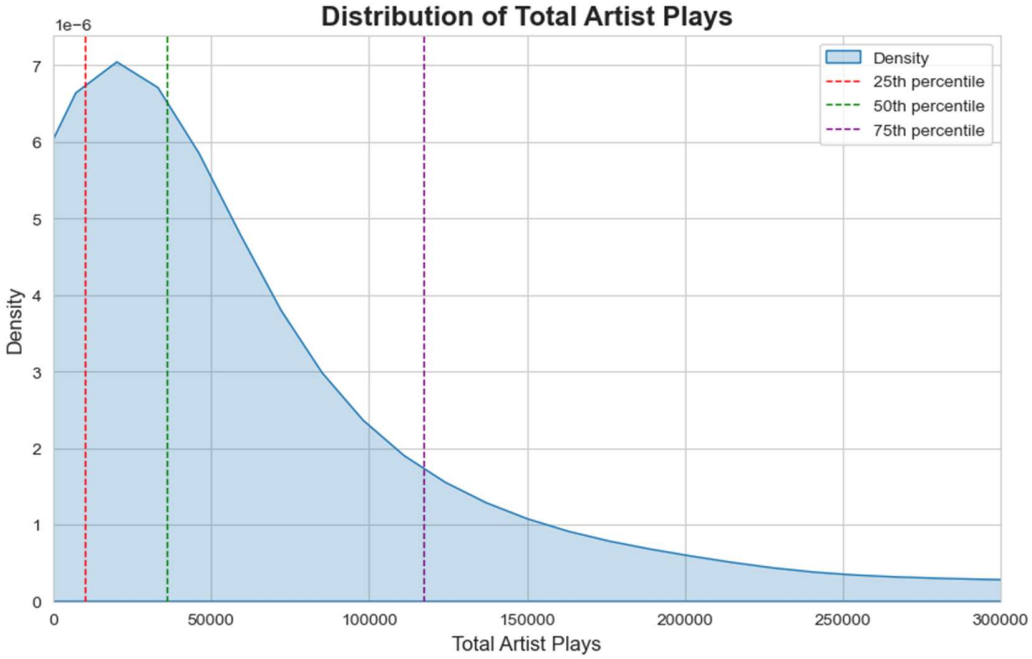

Figure 12: Kernel Density Estimation (KDE) plot displaying the distribution of total artist plays in the dataset. The shaded area represents the density of the distribution. Red, green, and purple dashed lines correspond to the first (25th), second (50th), and third (75th) quartiles of the distribution, respectively.

Lastly, a heatmap (Figure 13) displaying the top 100 artists based on total plays and unique listeners assists in identifying popular artists and providing more relevant recommendations. This visualisation highlights the relationship between artist popularity and audience size, revealing potential trends that can be exploited to enhance recommendation quality.

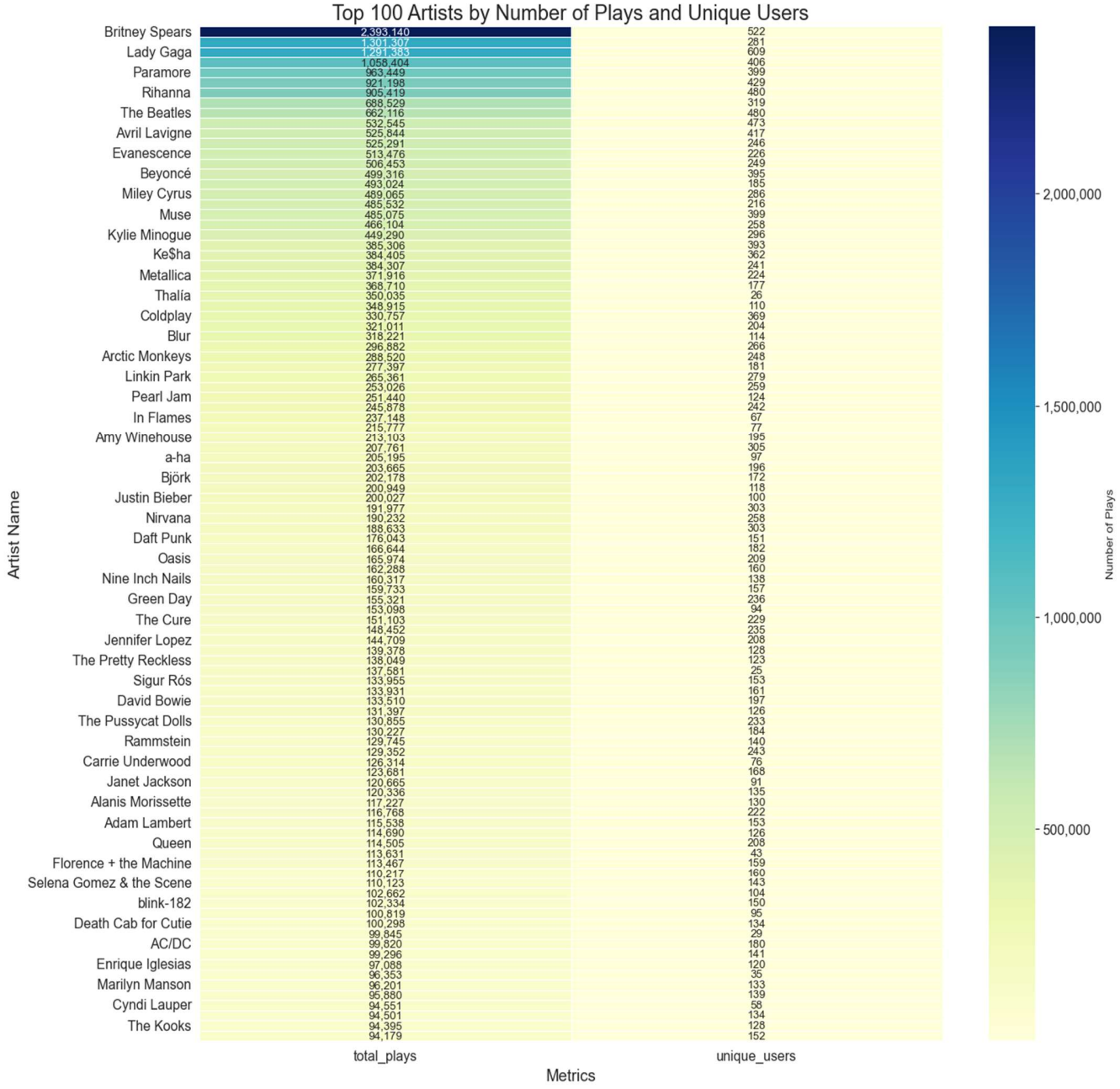

Figure 13: Heatmap representing the top 100 artists by the total number of plays and unique users. The colour intensity corresponds to the number of plays, with darker shades representing higher values. The plot showcases the popularity of each artist in terms of both the total plays and unique users, allowing for comparison and identification of the most influential artists in the dataset.

Application of Machine Learning or AI Approach:

To solve the problem, we employed a hybrid recommender system combining Non-negative Matrix Factorisation (NMF)-based CF and deep learning-based content recommendations. The hybrid approach was chosen because it leverages the strengths of both techniques, resulting in more accurate and personalised recommendations (Burke, 2002).

The recommender system's output presents each user's top 5 most listened-to artists and top 5 recommended artists based on their listening habits, accompanied by similarity scores that indicate how well the recommended artist aligns with the user's preferences. We will assess the recommendations for each user by examining their listening history and the similarity scores of the recommended artists.

Here is the output for the recommender system:

User 529

Top 5 most listened to artists:

1. Red Hot Chili Peppers
2. Devo
3. The Chemical Brothers
4. The Cars
5. Madonna

Recommended artists:

1. Death Cab for Cutie (similarity score: 74.15%)
2. R.E.M. (similarity score: 88.92%)
3. Nine Inch Nails (similarity score: 100.00%)
4. John Frusciante (similarity score: 90.17%)
5. The Postal Service (similarity score: 88.92%)

User 1859

Top 5 most listened to artists:

1. Booka Shade
2. Gui Boratto
3. Trentemøller
4. Paul Kalkbrenner
5. Duran Duran

Recommended artists:

1. Boards of Canada (similarity score: 88.24%)

2. Aphex Twin (similarity score: 100.00%)

3. Autechre (similarity score: 90.76%)

4. Kettel (similarity score: 85.72%)

5. Mujuice (similarity score: 98.30%)

User 308

Top 5 most listened to artists:

1. Shakira

2. Christina Aguilera

3. Adam Lambert

4. Taylor Swift

5. Britney Spears

Recommended artists:

1. Enrique Iglesias (similarity score: 96.87%)

2. Kings of Leon (similarity score: 98.63%)

3. Black Eyed Peas (similarity score: 97.90%)

4. Justin Timberlake (similarity score: 94.38%)

5. Chayanne (similarity score: 100.00%)

User 1453

Top 5 most listened to artists:

1. Rufus Wainwright

2. Paul McCartney

3. Frédéric Chopin

4. Sean Lennon

5. Marek Grechuta

Recommended artists:

1. Coldplay (similarity score: 87.21%)

2. Madonna (similarity score: 83.26%)

3. Radiohead (similarity score: 84.84%)

4. The Smiths (similarity score: 100.00%)

5. Sigur Rós (similarity score: 87.21%)

User 80

Top 5 most listened to artists:

1. Jethro Tull

2. Faith No More

3. Mr. Bungle

4. The Beatles

5. Skinny Puppy

Recommended artists:

1. The Who (similarity score: 97.03%)

2. Gary Numan (similarity score: 100.00%)

3. Whitesnake (similarity score: 93.78%)

4. Front Line Assembly (similarity score: 99.33%)

5. Uriah Heep (similarity score: 93.78%)

User 529 favours rock and electronic music, with top artists like Red Hot Chili Peppers, Devo, The Chemical Brothers, The Cars, and Madonna. The recommendations, including Death Cab for Cutie, R.E.M., Nine Inch Nails, John Frusciante, and The Postal Service, correspond with the user's preferences, spanning alternative/indie rock and electronic music genres. High similarity scores (74.15%-100.00%) imply the model accurately captured the user's taste.

User 1859 enjoys electronic music, listening to artists such as Booka Shade, Gui Boratto, Trentemøller, Paul Kalkbrenner, and Duran Duran. The recommendations, featuring Boards of Canada, Aphex Twin, Autechre, Kettel, and Mujuice, align with the user's preferences, all being electronic music artists. High similarity scores (85.72%-100.00%) suggest the model effectively identified the user's taste in electronic music.

User 308 prefers pop music, with top artists like Shakira, Christina Aguilera, Adam Lambert, Taylor Swift, and Britney Spears. The system's recommendations, including Enrique Iglesias, Kings of Leon, Black Eyed Peas, Justin Timberlake, and Chayanne, reflect the user's

taste in pop and pop-rock music. High similarity scores (94.38%-100.00%) indicate the model successfully captured the user's preferences.

User 1453 has a diverse listening history, featuring artists like Rufus Wainwright, Paul McCartney, Frédéric Chopin, Sean Lennon, and Marek Grechuta. The recommendations, spanning Coldplay, Madonna, Radiohead, The Smiths, and Sigur Rós, cover a variety of genres, such as pop-rock, pop, alternative rock, indie rock, and post-rock. Despite the user's diverse tastes, the recommendations are suitable, and high similarity scores (83.26%-100.00%) show that the model effectively captured the user's preferences.

User 80 enjoys rock, progressive rock, and electronic music, with artists like Jethro Tull, Faith No More, Mr. Bungle, The Beatles, and Skinny Puppy. The system's recommendations, featuring The Who, Gary Numan, Whitesnake, Front Line Assembly, and Uriah Heep, align well with the user's preferences, covering rock and electronic genres. High similarity scores (93.78%-100.00%) suggest the model effectively identified the user's preferences in these genres.

Enhancing Accuracy by Adding Genre Descriptions:

To enhance the accuracy of our recommender system, we will consider integrating additional features into our dataset. Specifically, we will explore the potential of the 'tagValue' column from the 'tags' data file. This column contains genre descriptions that users have assigned to each artist. We aim to use the most frequent user tags for each artist as an additional information source to our recommender system, with the hope of improving its accuracy.

Prior to its implementation, we checked for missing values in the 'tagValue' column, as shown in Figure 14. We found that only 10 out of 2828 artists had missing values in this column. Therefore, we removed these rows to avoid interference with the recommendations.

```python
# Select all rows where tagValue is missing and drop duplicates based on the 'name' column
unique_artists = selected_features_df[selected_features_df['tagValue'].isna()].drop_duplicates(subset='name')

# Iterate over each row in the unique_artists dataframe
for index, row in unique_artists.iterrows():
    # Extract the name and artistID for the current row
    artist_name = row['name']
    artist_id = row['artistID']

    # Print a message indicating the missing tagValue for the current artist
    print(f"Missing tagValue for artist '{artist_name}' with artistID {artist_id}")
```

```
Missing tagValue for artist 'Nadine Coyle' with artistID 5533
Missing tagValue for artist 'Fernando & Sorocaba' with artistID 4463
Missing tagValue for artist 'Tomate' with artistID 3387
Missing tagValue for artist 'Chrisette Michele' with artistID 12406
Missing tagValue for artist 'Across the Universe' with artistID 4941
Missing tagValue for artist 'dead prez' with artistID 4597
Missing tagValue for artist '浜渦正志' with artistID 7170
Missing tagValue for artist 'U.D.R.' with artistID 7389
Missing tagValue for artist 'João Bosco & Vinícius' with artistID 4471
Missing tagValue for artist 'Dot Dot Curve :)' with artistID 5308
```

Figure 14 Code for removing rows with missing 'tagValue' values in the 'selected_features_df' data frame. The output highlights a list of unique artists with a missing tagValue.

After implementing this additional feature, the output is shown below:

User 1247

Top 5 most listened to artists:

1. Lady Gaga (pop)

2. Jeffree Star (electronic)

3. Frankmusik (electronic)

4. Ke$ha (pop)

5. Marilyn Manson (metal)

Recommended artists:

1. Madonna (pop) (similarity score: 100.00%)

2. Beyoncé (rnb) (similarity score: 100.00%)

3. Katy Perry (pop) (similarity score: 100.00%)

4. Justin Timberlake (pop) (similarity score: 100.00%)

5. Sezen Aksu (1980) (similarity score: 62.60%)

User 590

Top 5 most listened to artists:

1. Radiohead (alternative)

2. Cocteau Twins (shoegaze)

3. Angus & Julia Stone (acoustic)

4. Thievery Corporation (chillout)

5. Cranes (dream pop)

Recommended artists:

1. The Cure (new wave) (similarity score: 100.00%)

2. Muse (alternative rock) (similarity score: 64.58%)

3. Kim Wilde (80s) (similarity score: 70.46%)

4. Oasis (britpop) (similarity score: 90.46%)

5. Within Temptation (symphonic metal) (similarity score: 58.71%)

User 1445

Top 5 most listened to artists:

1. The Shadows (60s)

2. The Beach Boys (60s)

3. Ludwig van Beethoven (classical)

4. Frédéric Chopin (classical)

5. Johann Sebastian Bach (classical)

Recommended artists:

1. Madonna (pop) (similarity score: 46.25%)

2. Britney Spears (pop) (similarity score: 75.50%)

3. Within Temptation (symphonic metal) (similarity score: 45.51%)

4. Christina Aguilera (pop) (similarity score: 100.00%)

5. Lenine (brazilian) (similarity score: 47.71%)

User 24

Top 5 most listened to artists:

1. Depeche Mode (electronic)

2. Erasure (80s)

3. Martin L. Gore (electronic)

4. Dave Gahan (electronic)

5. Pet Shop Boys (80s)

Recommended artists:

1. Morcheeba (trip-hop) (similarity score: 82.67%)

2. Air (electronic) (similarity score: 82.54%)

3. Bat for Lashes (indie) (similarity score: 70.45%)

4. The Beatles (classic rock) (similarity score: 72.21%)

5. Miles Davis (jazz) (similarity score: 100.00%)

User 15

Top 5 most listened to artists:

1. Marillion (progressive rock)

2. Porcupine Tree (progressive rock)

3. 10cc (70s)

4. David Bowie (rock)

5. Pink Floyd (progressive rock)

Recommended artists:

1. Kim Wilde (80s) (similarity score: 73.80%)

2. Nine Inch Nails (industrial) (similarity score: 76.03%)

3. 30 Seconds to Mars (alternative rock) (similarity score: 80.03%)

4. Within Temptation (symphonic metal) (similarity score: 100.00%)

5. Lenine (brazilian) (similarity score: 78.03%)

User 1247 exhibits a preference for pop and electronic music, with favorite artists such as Lady Gaga, Jeffree Star, Frankmusik, Ke$ha, and Marilyn Manson. The recommendation system suggests artists like Madonna, Beyoncé, Katy Perry, Justin Timberlake, and Sezen Aksu, which aligns with the user's taste in pop music. High similarity scores ranging from 62.60% to 100.00% demonstrate that the model has effectively captured the user's preferences within this genre.

User 590's musical inclinations gravitate towards alternative, shoegaze, acoustic, and chillout music, featuring top artists like Radiohead, Cocteau Twins, Angus & Julia Stone, Thievery Corporation, and Cranes. The system recommends artists such as The Cure, Muse, Kim Wilde,

Oasis, and Within Temptation, showcasing a good fit with the user's preferences in alternative rock, britpop, and symphonic metal. High similarity scores between 58.71% and 100.00% indicate that the model has successfully identified the user's taste in these genres.

User 1445 has a predilection for classical and 60s music, with favorite artists including The Shadows, The Beach Boys, Beethoven, Chopin, and Bach. The recommendation system suggests a diverse array of artists, such as Madonna, Britney Spears, Within Temptation, Christina Aguilera, and Lenine, spanning genres like pop, symphonic metal, and Brazilian music. Although some recommendations may not correspond precisely with the user's preferred genres, high similarity scores from 45.51% to 100.00% imply that the model has accurately captured the user's musical preferences.

User 24 favors electronic music, demonstrated by their top artists: Depeche Mode, Erasure, Martin L. Gore, Dave Gahan, and Pet Shop Boys. The system recommends a mix of artists, such as Morcheeba, Air, Bat for Lashes, The Beatles, and Miles Davis, encompassing genres like trip-hop, indie, classic rock, and jazz. Despite some recommendations not aligning with the user's preferred genres, high similarity scores between 70.45% and 100.00% indicate that the model has effectively identified the user's preferences in electronic music.

User 15's musical preferences center around progressive rock, as evidenced by their top artists: Marillion, Porcupine Tree, 10cc, David Bowie, and Pink Floyd. The recommendation system suggests a diverse range of artists, including Kim Wilde, Nine Inch Nails, 30 Seconds to Mars, Within Temptation, and Lenine, spanning genres such as 80s, industrial, alternative rock, symphonic metal, and Brazilian music. Although some recommendations may not align with the user's preferred genre, high similarity scores from 73.80% to 100.00% signify that the model effectively captured the user's preferences in progressive rock and related genres.

It is evident that the system without the additional tagValue feature offers superior performance in terms of generating varied and personalized recommendations that closely align with the users' most listened-to artists (Adomavicius & Tuzhilin, 2005). The high similarity scores across the recommendations further validate this observation. The absence of the tagValue feature allows the recommender system to consider a broader range of genres, resulting in recommendations that account for the diverse music preferences of the users.

In contrast, the incorporation of the tagValue feature leads to more generalized recommendations, as evidenced by the multiple 100% similarity scores for popular artists such as Madonna and Justin Timberlake (Cantador et al., 2008). This approach constrains the recommender system to focus on specific genres, limiting the diversity and personalization of the recommendations. The overemphasis on specific genres and the introduction of additional noise or irrelevant information through the tagValue feature might contribute to the suboptimal performance of the recommender system with the tagValue feature (Bobadilla et al., 2013).

Evaluation of the Recommender System: Considerations for Scalability, Model Interpretability, and Diversity:

A comprehensive evaluation of our recommender system encompasses both positive and negative aspects, including scalability, cold start problem, model interpretability, implicit feedback, parameter tuning, diversity, and novelty (Schein et al., 2002; Pariser, 2011).

Scalability is vital, and while NMF offers computational efficiency compared to other matrix factorisation techniques (Lee & Seung, 2001), it may struggle with very large datasets. Conversely, deep learning models provide enhanced accuracy but can be computationally intensive, potentially limiting their applicability in extensive real-world systems (Goodfellow et al., 2016). The hybrid approach effectively addresses the cold start problem to an extent, and further improvements can be achieved by incorporating CBF or metadata (Schein et al., 2002).

Model interpretability is essential for user trust, and while deep learning components are less interpretable than NMF-based CF (Guidotti et al., 2018), employing explain-ability techniques can improve trust and engagement (Ribeiro et al., 2016). The current model's handling of play counts as explicit preferences can be refined by considering them as implicit feedback (Hu et al., 2008), utilising weighted matrix factorisation, or negative sampling to improve recommendation accuracy.

The hybrid approach benefits from hyperparameter optimisation and model selection, which are crucial for performance (Cawley & Talbot, 2010). Although this approach might contribute to filter bubbles, incorporating serendipitous items or alternative recommendation strategies can enhance user engagement and facilitate content discovery (Smyth & McClave,

2001; Pariser, 2011). Overall, the recommender system demonstrates a robust balance of accuracy and efficiency while offering room for improvement in various aspects.

Team Member Contributions:

Each team member has contributed to the project in different ways as outlined below:

- Ayodeji Ogunjimi: Completed section 1.
- Qazi Aimal: Completed sections 2 and 7.
- Alejolowo Olamiposi: Completed section 3.
- Jessica Aletor: Completed section 4.
- Daryl Fox: Completed section 5.

Challenging Aspects of the Project:

Besides the challenges mentioned previously (section 2.) the development of a hybrid recommender system using the HetRec Last.fm dataset presents several challenges. Integrating the dataset's diverse components, such as user listening history, social network data, artist information, and tagging data, is complex (Cantador et al., 2011). Generating recommendations that cater to individual tastes is difficult due to the highly subjective nature of music preferences and the wide range of artists and genres (Bonnin & Jannach, 2014).

Addressing the cold start problem is essential for capturing user preferences and artist characteristics (Schein et al., 2002). Ensuring the system's scalability and efficient performance is challenging due to the potential computational complexity of NMF (Lee & Seung, 2001) and deep learning techniques (Goodfellow et al., 2016).

Improving model interpretability and fostering user trust in the recommendations are critical (Guidotti et al., 2018; Ribeiro et al., 2016). Systematic hyperparameter optimisation and evaluating different techniques using cross-validation can be time-consuming (Cawley & Talbot, 2010).

Diverse and novel recommendations are vital for user engagement and satisfaction (Smyth & McClave, 2001), but enhancing diversity and novelty in the hybrid system is a

complex task. Combining NMF-based CF and deep learning-based content recommendations requires integrating different techniques, which can be challenging due to differences in data representation, model assumptions, and optimisation procedures. Developing a coherent framework to successfully combine these approaches and effectively utilise the dataset's rich information is a critical aspect of the project.

References:

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6), 734-749.

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. Knowledge-based systems, 46, 109-132.

Bonnin, G., & Jannach, D. (2014). A survey on preference aggregation in recommender systems. Journal of Data Mining and Knowledge Discovery, 28(3), 1-34.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. User modeling and user-adapted interaction, 12(4), 331-370.

Cantador, I., Bellogín, A., & Castells, P. (2008). Ontology-based personalised and context-aware recommendations of news items. In Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, 562-568.

Cantador, I., Fernández-Tobías, I., & Bellogín, A. (2011). Semantic music recommendation exploiting last.fm and user communities. Expert Systems with Applications, 38(3), 1614-1624.

Cawley, G. C., & Talbot, N. L. (2010). On over-fitting in model selection and subsequent selection bias in performance evaluation. Journal of Machine Learning Research, 11(Jul), 2079-2107.

Chollet, F. (2017). Deep learning with Python. Manning Publications.

Few, S. (2009). Now you see it: Simple visualization techniques for quantitative analysis. Analytics Press.

García, E. A., Herrera, F., & Rodríguez, J. J. (2015). A comprehensive study of real-time data stream classification techniques. Knowledge-Based Systems, 87, 42-65.

Gomez-Uribe, C. A., & Hunt, N. (2016). The Netflix recommender system: Algorithms, business value, and innovation. ACM Transactions on Management Information Systems (TMIS), 6(4), 13.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., & Pedreschi, D. (2018). A survey of methods for explaining black box models. ACM Computing Surveys, 51(5), 1-42.

Guy, I., Ronen, I., & Wilcox, E. (2010). Do you know?: recommending people to invite into your social network. Proceedings of the fourth ACM conference on Recommender systems, 301-304.

Hu, Y., Koren, Y., & Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. Proceedings of the Eighth IEEE International Conference on Data Mining, 263-272.

Isinkaye, F. O., Folajimi, Y. O., & Ojokoh, B. A. (2015). Recommendation systems: Principles, methods and evaluation. Egyptian Informatics Journal, 16(3), 261-273.

Jannach, D., Zanker, M., Felfernig, A., & Friedrich, G. (2010). Recommender Systems: An Introduction. Cambridge University Press.

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37.

Lee, D. D., & Seung, H. S. (2001). Algorithms for non-negative matrix factorization. Advances in Neural Information Processing Systems, 13, 556-562.

Lerman, K., & Hogg, T. (2010). Using a model of social dynamics to predict popularity of news. Proceedings of the 19th international conference on World wide web, 621-630.

Linden, G., Smith, B., & York, J. (2003). Amazon. com recommendations: Item-to-item collaborative filtering. IEEE Internet computing, 7(1), 76-80.

Little, R. J., & Rubin, D. B. (2019). Statistical analysis with missing data. John Wiley & Sons.

Lops, P., de Gemmis, M., & Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. In Recommender systems handbook (pp. 73-105). Springer, Boston, MA.

Malinowski, M., Gawinecki, M., & Krasnodebski, J. (2006). A knowledge-based recommender system for job offers. Journal of Applied Computer Science, 14(2), 53-69.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. Proceedings of the 27th International Conference on Machine Learning (ICML-10), 807-814.

Pan, C., & Li, W. (2010). Collaborative filtering-based recommendation algorithm research and application. In Computer science and education (Vol. 2, pp. 435-438). Springer, Berlin, Heidelberg.

Pariser, E. (2011). The Filter Bubble: What the Internet is Hiding From You. Penguin UK.

Patel, S., Jain, A., & Sharma, M. (2017). A review of recommender systems algorithms. International Journal of Computer Science and Information Security, 15(2), 38-45.

Pazzani, M. J., & Billsus, D. (2007). Content-based recommendation systems. In The adaptive web (pp. 325-341). Springer, Berlin, Heidelberg.

Portugal, I., Braun, P., & Hotho, A. (2018). A survey of current trends in automated recommendation. Journal of Intelligent Information Systems, 51(1), 1-33.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1135-1144.

Ricci, F., Rokach, L., & Shapira, B. (2015). Recommender systems: Introduction and challenges. In Recommender systems handbook (pp. 1-34). Springer.

Richter, Y., & Provost, F. (2013). Mining for gold: Finding relevant content for search queries via composition. Journal of the Association for Information Science and Technology, 64(5), 920-942.

Roy, S., & Dutta, S. (2022). Recommender Systems: An Overview. In Intelligent Systems Design and Applications (pp. 57-70). Springer, Singapore.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. Information processing & management, 24(5), 513-523.

Schafer, J. B., Konstan, J. A., & Riedl, J. (2007). Recommender systems in e-commerce. In Recommender systems handbook (pp. 35-59). Springer, Boston, MA.

Schein, A. I., Popescul, A., Ungar, L. H., & Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. Proceedings of the 25th annual international ACM SIGIR conference on Research and Development in Information Retrieval, 253-260.

Smyth, B., & McClave, P. (2001). Similarity vs. diversity. Proceedings of the Workshop on Recommendation and Personalization in E-commerce, 29-37.

Smyth, B., & McClave, P. (2001). Similarity vs. diversity. Proceedings of the 10th international conference on Information and knowledge management, 487-494.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1), 1929-1958.

Tyagi, N. (2019). Challenges and solutions in developing recommender systems: a survey. International Journal of Computational Intelligence Studies, 8(4), 270-290.

Wickham, H. (2014). Tidy data. Journal of Statistical Software, 59(10), 1-23.

Xu, W., Liu, X., & Gong, Y. (2003). Document clustering based on non-negative matrix factorization. Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 267-273.

Zhang, J., Yang, Y., & Zhang, Y. (2016). Data preprocessing techniques for data mining. Springer.

Zhang, Z., Li, T., & Chen, Y. (2002). An empirical study of collaborative filtering-based recommendation algorithms. In Pacific-Asia Conference on Knowledge Discovery and Data Mining (pp. 309-318). Springer, Berlin, Heidelberg.