Decoding Playlist Success

Bogdan Bîndilă
m.bindila@student.utwente.nl

Cristina Racoviță
c.racovita@student.utwente.nl

Sarang Nirwan
sarangnirwan@student.utwente.nl

Smriti Dangi
s.dangi@student.utwente.nl

Abstract

This report presents a predictive analysis of playlist popularity using the Spotify Million Playlist Dataset and retrieved data from Spotify API. Handling this big data with PySpark and Random Forest Regressor, we focused on forecasting the number of followers for a given playlist based on audio features and song genres.

Our findings indicate that audio features have a limited impact on playlist popularity, challenging common assumptions in the field. The results suggest that factors beyond audio features play a more significant role in determining playlist success, such as artist popularity.

This research contributes to the understanding of playlist dynamics and provides insights to music enthusiasts. The report concludes with implications for future research directions in the realm of big data analytics and playlist popularity prediction.

1 Introduction

The music consumption trend shows that in the past five years, users of streaming platforms increasingly favor listening to playlists. In 2018 The Annual Music Report had revealed that 54% of consumers replaced albums by playlists and the 2023 Fan Engagement Report showed that 88% of them listen to customized playlists. This shifting paradigm in music consumption underscores the growing importance of playlist popularity as a key metric in understanding user behavior and preferences.

In the evolving landscape of digital media, big data has emerged as a pivotal element, particularly in the music streaming industry. Platforms like Spotify, harnessing vast amounts of data, offer unprecedented opportunities for in-depth analysis and insights into user preferences and trends.

The ubiquity of music streaming services like Spotify has led to the availability of extensive datasets, including a notable one featuring 1 million playlists. Alongside this, Spotify's API provides rich data, including detailed artist information and a set of 12 characteristic audio features for each track. These resources form the backbone of our data-driven investigation.

While there is a plethora of research centered on analyzing individual songs, which we will discuss in Section 2, our project diverges by targeting playlists. The primary challenge addressed in this study is the development of a predictive model for determining the popularity of a playlist, measured by its follower count, based on its constituent audio features and genres. This approach marks a significant shift from a song to a playlist-centric perspective.

Our methodology encompasses a comprehensive process beginning with data retrieval from both the Spotify API and the 1 million playlist dataset. This is followed by meticulous data-cleaning procedures. We then engaged in exploratory data analysis, employing both univariate and bivariate techniques, which paved the way for effective feature selection. The culmination of this process was the development and training of a predictive model.

This study has the potential to significantly impact several facets of the music streaming industry. It can offer insights into user behavior and preferences on a macro scale, guide playlist creation strategies, and enhance personalized user experiences. Furthermore, it contributes to the broader field of big data analytics by introducing innovative methodologies and perspectives.

This report is structured to provide a comprehensive overview of our project. Following this introduction, the subsequent sections will detail our methodology, present our findings, discuss the implications and limitations of our study, and finally, conclude with key takeaways and potential directions for future research.

2 Related Work

Hit song prediction (HSP), one of the emerging fields in music information retrieval (MIR), remains a considerable challenge. Many scientific papers research to predict the popularity of songs using the Spotify API data. Many of them expand upon the available features by introducing additional ones, such as sentence similarity coefficients, MEL spectrograms, and MEL frequency cepstral coefficients in (Martín-Gutiérrez et al., 2020). Nevertheless, the majority of these papers focus on developing machine-learning models aimed at forecasting the popularity of individual songs. The papers referenced by our group provide a broad overview of the research in this field.

In (Martín-Gutiérrez et al., 2020), attention is given to augmenting the retrieval of music information and the prognostication of the popularity of songs. It initiates the introduction of the SpotGenTrack Popularity Dataset (SPD) which serves the purpose of comparing and endorsing models. The paper proceeds to present HitMusicNet, an elaborate deep-learning framework that accommodates multiple modes of input to predict popularity. This framework outperforms previous investigations by incorporating audio, lyrics, and metadata as different modalities.

In (Khan et al., 2022), the paper focuses on the feature selection process when building a machine learning model to predict individual song popularity. Feature selection enhances the accuracy of music popularity classification through the utilization of machine learning techniques. The filter feature selection approach was employed to identify pertinent features. The paper determined that models that underwent feature selection exhibited superior accuracy rates in comparison to models that did not undergo feature selection. There are many other related works similar to the ones cited in this report that continue to experiment with different approaches to improve model accuracy in determining song popularity.

In the exploratory analysis of over 100,000 songs from Spotify (Devor, 2019), Matt Devor identified a significant challenge with the highly right-skewed distribution of the popularity scores assigned to songs. The skewness indicated that while a few songs enjoyed extremely high popularity, the vast majority did not, posing a challenge for predictive modeling. Devor's initial use of linear regression was hampered by this skewness, as reflected by low R^2 values, suggesting that the model struggled to capture the variance in the popularity scores. To address the imbalance in the dataset, undersampling techniques were employed to create a more balanced representation of popular and unpopular songs. This approach, although it improved model metrics to some extent, also highlighted the trade-off between improving model fit and maintaining the significance of the predictors. Devor's exploration revealed the nuanced difficulty of predicting song popularity based on Spotify's audio features and underscored the importance of considering data distribution in predictive modeling.

An interesting approach is presented in (Alonso-Jiménez et al., 2023), where the idea of creating a music recommendation system is extended by using playlist data as a source of music similarity information. There, a neural network is trained with playlist information to demonstrate that these data are valuable for music classification and similarity. This study concludes that superior performance can be achieved with consumption metadata consisting of playlist information by relying on track representations obtained from a Word2Vec model trained on the playlist sequences.

However, no paper was found that aimed to aggregate features on a playlist level and build a machine-learning model based on those features. Building a model to predict playlist popularity is the additional contribution that our group has experimented with in this project.

3 Dataset

This section details the dataset used to answer the proposed research questions, illustrates the retrieval process, and explains the data cleaning operations.

3.1 Data Description

Our study is based on the dataset employed for The Spotify Million Playlist Dataset Challenge. The data at hand contains 1 million playlists created by US users between January 2010 and November 2017. They were randomly sampled from a population of 4 billion playlists and filtered based on content quality and offensiveness. Overall, they contain 66,346,428 tracks, and in terms of structure, the dataset is composed of 1,000 JSON files, each containing 1,000 playlists, totaling 31.2 GB unzipped.

The playlist dataset was downloaded from the challenge website and as shown in Figure 2, includes playlist metadata namely playlist id/name/description/duration/date of modification, number of albums/artists/songs/followers and if more than one user collaborates in creating it, along with information about the tracks themselves, such as the artist name/URI, album name/URI and track name/URI/duration/position.

Since the focus of this project is built around the music itself, without considering the popularity of individual songs, artists, or albums, we also improved the existing attributes by retrieving the audio characteristics of each song and the music genres associated with the artists.

The audio features dataset contains the track URI, needed to join this table with the playlist data, and 12 columns of interest describing the audio characteristics, shown in Figure 2 and detailed in Figure 1. As a consequence of the retrieval procedure, they are stored in 229 JSON files, occupying 1 GB of storage space.

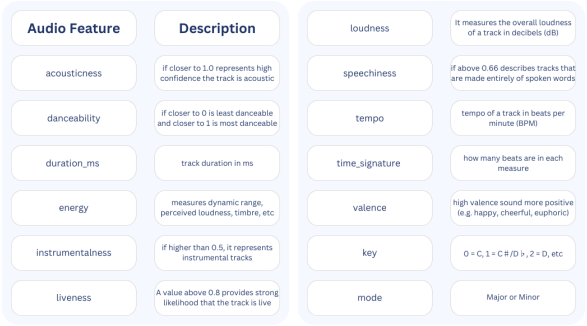

Figure 1: Description of Audio Features

Last but not least, we retrieved for each artist name, id, and URL the associated genres. This information, along with other metadata such as the artist URI, is shown in Figure 2. This table is 35.1 MB, and because

of how the retrieval process was configured, it is split into 60 JSON files.

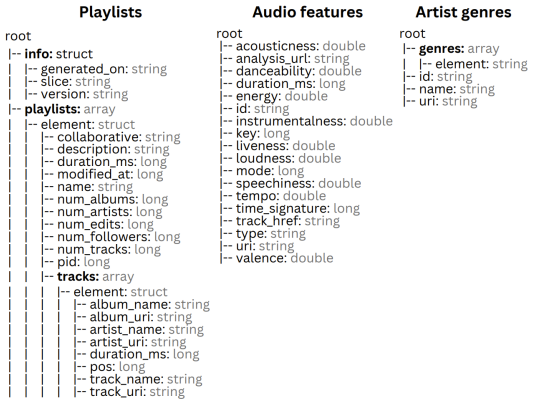

Figure 2: Schemas of the raw tables

The large number of files stored per dataset is minimized during the data cleaning phase when the data will be repartitioned according to the data size to optimize the processing time required for downstream operations.

3.2 Retrieval Process

We retrieved the audio features of the songs and the genres associated with the artists using the Spotify API. Namely, the invoked endpoints are *https://api.spotify.com/v1/audio-features/* and *https://api.spotify.com/v1/artists/*.

Acquiring a large amount of data through this API is challenging because it has an unknown limit of calls per second, monitored with a 30-second window. Because of this, we had to find the maximum number of calls that would allow us to avoid getting a 429 error. After much experimentation, it turned out that a safe threshold is 50 requests per minute. However, even following this rule, after 2,000 API calls, a 429 error code is received for the next 24 hours. Thus, by using three different Spotify accounts, the process took a total of five days.

Listed below are the steps taken to obtain, process, and store the aforementioned datasets:

1. gets unique artist or track IDs from the playlist dataset and sorts them to maintain the retrieval order
2. retrieve batches of 50 artists or 100 songs per API call
3. from each batch remove unwanted features and convert data types to ensure consistency
4. provide the schema and combine 2,000 batches of data into a single PySpark DataFrame
5. repartition the data frame and store it as a single JSON file

The data retrieval script was executed directly on the cluster to minimize write time and avoid moving data. Due to API limits, only one call was made at a time.

In total, out of 2,262,292 unique songs, for 193 entries, the audio features were not found. When it comes to artists, of the 295,860 distinct entries, no information was available for 3,346.

3.3 Data Cleaning

We applied four main steps to be sure our retrieved data is clean. These are described below in depth, together with what changes in the data structure we needed to make to keep the records in a more suitable-to-work-with form. We tried to implement the most important stages that are presented in (Ridzuan and Zainon, 2019): read data, detect and correct abnormal data (like null IDs, weird artists' names, etc), fill missing data (this step does not apply in our case), broadcast the updated value, and solve deduplication and conflict resolution (interval and duplication analysis).

3.3.1 Null Checking

Firstly, we identified the fields that should not be null in every DataFrame: Artists, Playlists, and Audio Features. Also, some fields were tested only to understand the data better (such as artist name, playlist name, and playlist description). After completing this step, our conclusions are:

- Artist ID, name, and URL are non-null. The genre array from this table contains empty arrays in the proportion of 58%.
- Audio Feature URI, track href, and analysis URI is non-null.
- Playlist name, id, and track URI are non-null. The description field is empty 98% of rows; therefore, we drop it, inspiring from this paper (**?**).

3.3.2 Duplicate Checking

The second step was to find data duplicates, mostly (but not only) for fields that have the primary key role. We take into consideration every table the: ID, URI, and name. Almost all of them were unique, except for the playlist and artist names, which turned out to reflect the real world.

3.3.3 Interval Analysis

The interval analysis was done for quality checking since many of these values were already provided from Spotify documentation. Audio Features' intervals are detailed in Figure 3. Besides these fields, we also verified the playlist number of albums, artists, edits, followers, and tracks, and we found out that:

- Number of albums is between two and 244.
- Number of artists is between three and 238.
- Number of edits is between one and 201.
- Number of followers is between one and 71643 (with 620 unique values).
- Number of tracks is between five and 376.

3.3.4 Structure Changes

After looking over these DataFrames, we deleted the fields that were not needed for our project. These include constant-value fields, duplicate columns, and use-

Figure 3: Intervals of Audio Features

less information. The structure of the final data can be seen in Figure 4.

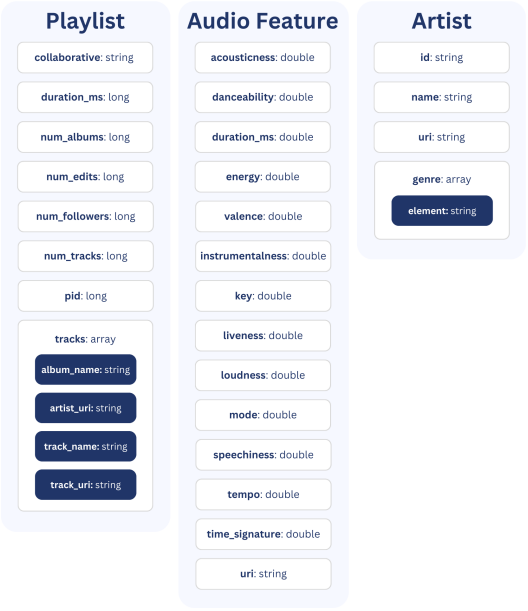

Figure 4: Final Schema after Cleaning

4 Method

4.1 Exploratory Data Analysis: *Univariate Analysis*

Before commencing our analysis, our team engaged in a comprehensive discussion to dissect the core research question into focused sub-questions, facilitating a structured approach to our exploratory data analysis.

4.1.1 Sub-categorization of Research Questions

These questions were separated by the dataset and focused on the artist genre information, audio features, and playlist information respectively. The list of questions that we decided on are as follows:

Artist Genre Data Questions

1. What are the most popular genres across the dataset?
2. How many genres are generally associated with each artist?

The corresponding data for these questions included a count of the genres listed in the artist data,

Audio Feature Data Questions

1. What is the distribution of the audio features?
2. What is the distribution of the duration of tracks?

Playlist Data Questions

1. What is the distribution of playlist duration?
2. How often are the playlists edited?
3. What is the overall composition of a playlist?
4. What are the popular words in the title of playlists?
5. What is the distribution of follower counts across the playlists?

4.1.2 Data Extraction

The data extraction for this analysis started with analyzing the schema of the cleaned data to write the script to then run on PySpark, hoping to export the resulting CSV files onto the HDFS server, and subsequently move to a local machine to visualize the data using the following libraries - Pandas, Matplotlib, and Seaborn.

4.1.3 Artists Data Visualisation

The first set of analyses was conducted on the artist data which focused on genre information. The bar chart of the top 20 most popular genres (Figure 5) displays a diverse musical taste among the playlists' creators. However, certain genres like 'norteno' and 'filmi' appear more frequently, indicating potential trends or cultural influences in the dataset.

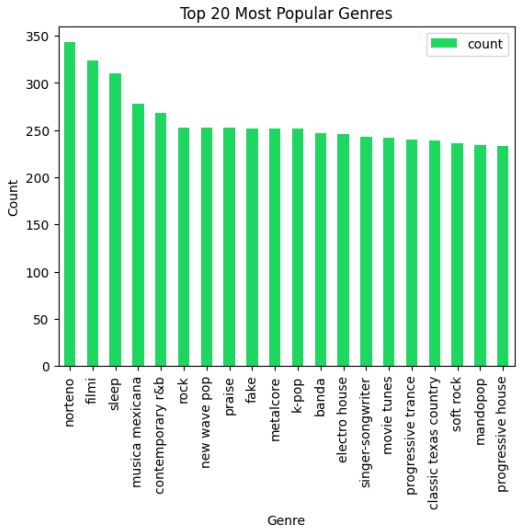

Figure 5: Top 20 Most Popular Genres

Figure 6 illustrates the number of genres that artists are typically associated with, showing a predominantly right-skewed distribution. Most artists fall within one to three genres, indicating a tendency toward genre specialization. This pattern is essential for modeling as it reflects the genre diversity an artist brings to a playlist, potentially affecting its popularity.

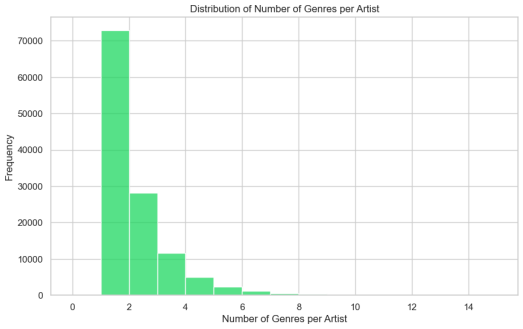

Figure 6: Number of Genres per Artist

4.1.4 Audio Features Visualisation

Secondly, audio feature analyses were done on the general distribution of the technical audio aspects as well as the duration of the songs. The box plots of audio features (Figure 7) provide a comprehensive overview of the distribution of each feature. Across features such as danceability, energy, and valence, there is a relatively even spread with few outliers, suggesting a diverse range of music characteristics within the playlists.

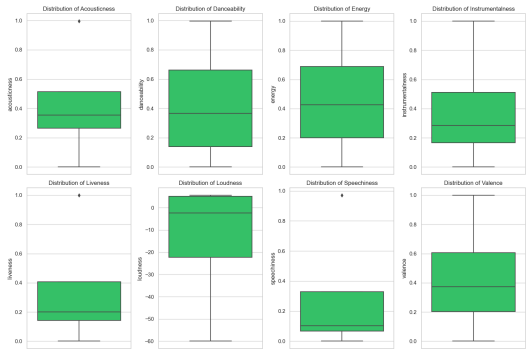

Figure 7: Distribution of Audio Features

The histogram of song duration (Figure 8) reveals a right-skewed distribution, with the majority of songs falling within the shorter duration range. This may reflect a preference for tracks that fit within a certain length, possibly influenced by listening habits or platform trends.

4.1.5 Playlist Data Visualisation

Lastly, the playlist data was analyzed. The total duration of playlists is another variable with a right-skewed distribution (Figure 9). Most playlists are relatively short, with fewer playlists exceeding a longer duration. This might correlate with listeners' preference for shorter listening periods. Here is also a table of the individual statistics (Table 1). The values of the duration are highly variable and we have summarised the statistics for them below. There is a very big gap between the max and min values. Numerically they are 176 hours, 25 minutes, and 2 minutes respectively.

The number of edits to playlists shows a heavily right-skewed distribution, as depicted in Figure 10.

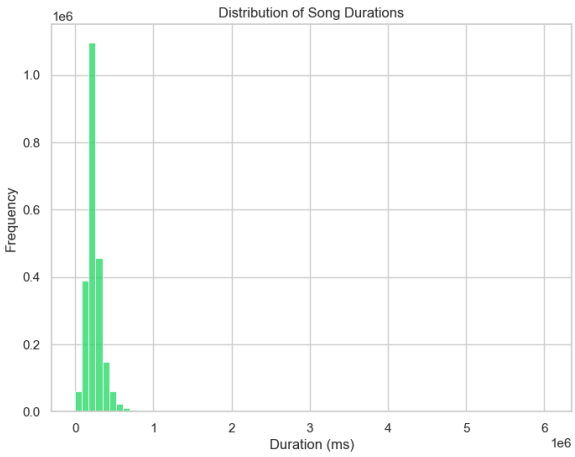

Figure 8: Distribution of Duration of Songs

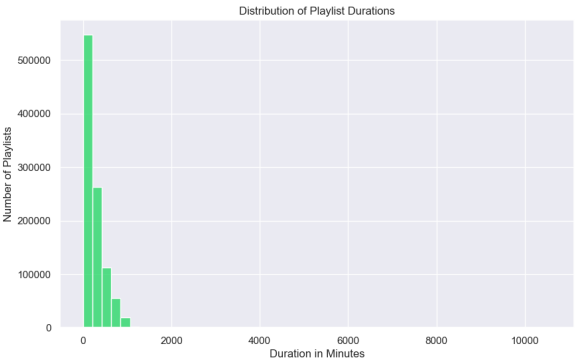

Figure 9: Distribution of Duration of Playlist

Most playlists have been edited a few times, with the frequency sharply decreasing as the number of edits increases, suggesting that most users tend to create playlists and make minimal changes thereafter.

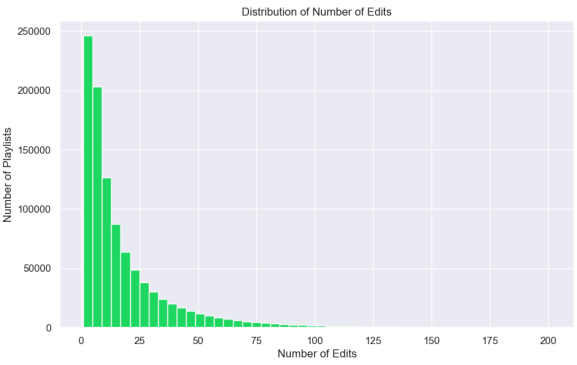

Figure 10: Distribution of Revision of Playlists

Figure 11 illustrates the number of artists, albums, and tracks in the playlists. All three variables show right-skewed distributions, with playlists typically featuring various tracks from a smaller selection of artists and albums.

Playlist name lengths, as seen in Figure 12, also display a right-skewed distribution with the majority of

Statistic	Duration (min)
Mean	259.66
Standard Deviation	214.27
Minimum	1.63
25th Percentile	99.92
50th Percentile (Median)	190.37
75th Percentile	357.23
Maximum	10584.56

Table 1: Basic Statistics for Playlist Duration

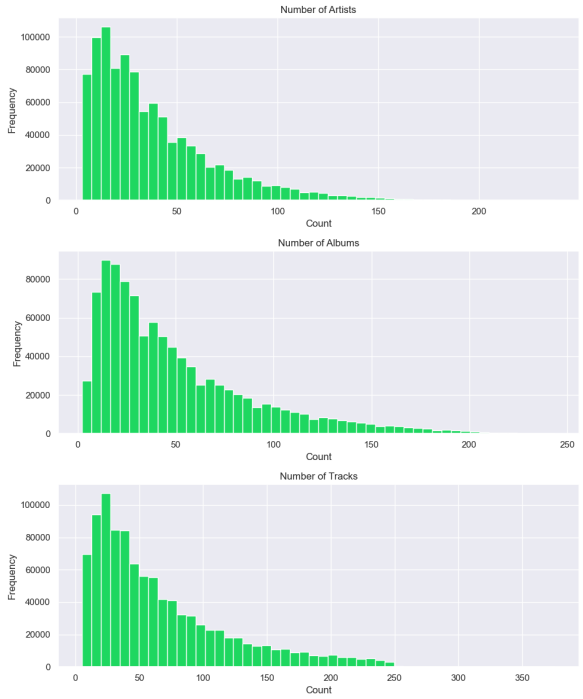

Figure 11: Overall Composition of Playlists

playlist names being relatively short. This suggests that users prefer concise titles for their playlists.

The distribution of followers, both in its raw and log-transformed state (Figure 13), exhibits a significant right skew. This indicates that a small number of playlists have a very high follower count, whereas the vast majority have relatively few. Log transformation did not normalize the distribution, indicating that the high follower counts are extreme outliers.

followers	1	2	3	4-100	101 - 1000	1001 - 71643
data %	75.42	14.96	4.69	4.84	0.064	0.0176

Table 2: Distribution of playlists by follower count

Moreover, as can be seen in Table 2, we have fewer and fewer playlists as the number of followers increases.

4.2 Exploratory Data Analysis: *Bivariate Analysis*

4.2.1 Aggregation

For the bivariate analysis, features from the main dataset as well as from other datasets were aggregated to the

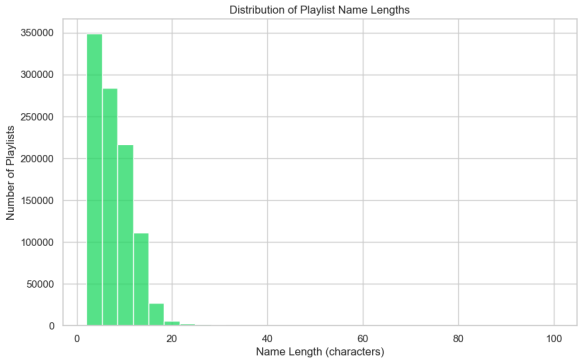

Figure 12: Length of Playlist Names

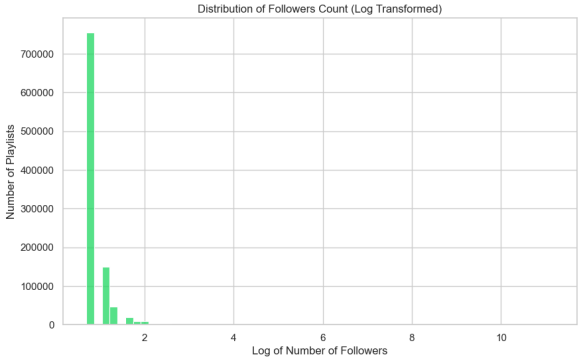

Figure 13: Log-transformed Distribution Followers of Playlists

playlist level first followed by the number of followers. Grouping the dataset to the playlist level still meant that the dataset contained 1 million entries which is impossible to load and visualise. Hence, the dataset had to be further aggregated to the number of followers thus resulting in the number of records in the dataset to reduce to 620. The data was then saved to the HDFS server and moved to the NFS server from which it was copied to a local machine for visualization using Python libraries such as Matplotlib and Seaborn.

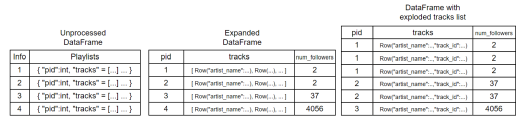

Figure 14: A visual representation of the transformation of the playlist dataset

Relevant features are first selected from the playlists JSON objects (pid, tracks, num_followers), and then the tracks column is exploded to have a unique track_uri per record as shown in Figure 14.

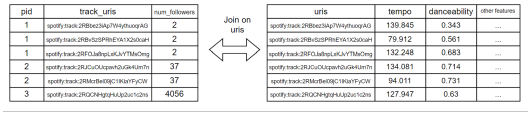

Figure 15: Joining the playlist and audio datasets based on track Uris

After having a unique track_id per row in the playlist dataset, it is joined with the artist's dataset using an inner join on the track_uri column as shown in Figure 15.

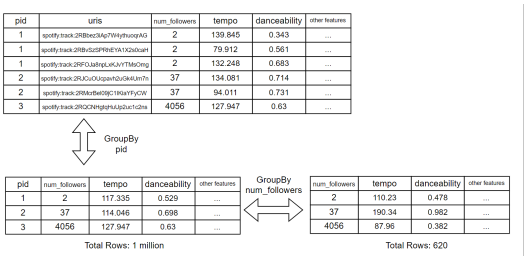

Figure 16: Aggregating data from 1 million records to 620 records

The final data aggregation step involved using two group-by operations to first have a unique pid per record and then to have a unique num_followers count per row as shown in 16.

4.2.2 Visualisations

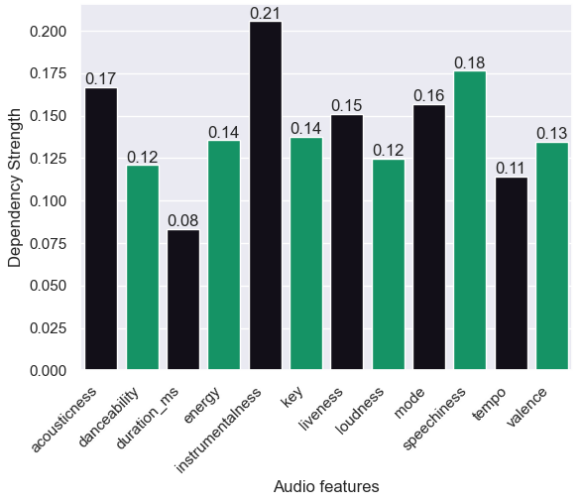

Figure 17: Mutual Information Regression scores for each audio feature

The hypothesis that we had was that the audio features are unlikely to be linearly related to the number of followers. However, there might be some form of clustering occurring in audio features if we break down the playlists into different groups according to their corresponding number of followers.

We first determined whether there might be any non-linear useful audio features using the mutual_info_regression function from Sklearn as seen in Figure 17. The function relies on non-parametric methods based on entropy estimation from k-nearest neighbor distances. However, as seen from the plot the scores from each audio feature are, unfortunately, not particularly impressive.

Figure 18: Instrumentalness feature analysis

The plot of the logged(num_followers) against instrumentalness in Figure 18 can show a visual confirmation of weak clustering as most of the points regardless of which quartile they belong to are around the mean.

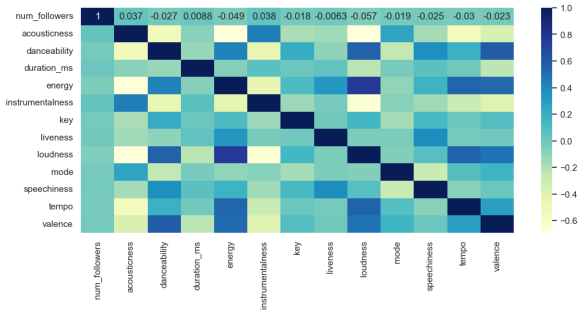

Figure 19: Correlation with audio features

The correlation heat map of the logged(num_followers) with the audio features in Figure 19 also confirms a weak Pearson correlation with every audio feature.

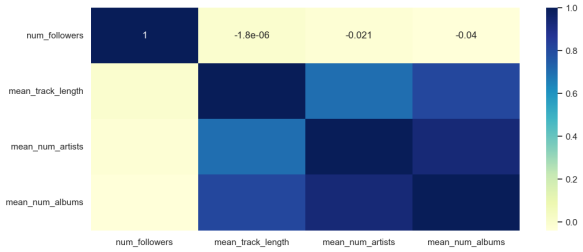

Figure 20: Correlation with main features

The Pearson correlation values of logged(num_followers) with the main features in Figure 20 from the playlist dataset were also not impressive.

However, these correlation values and plots were calculated after aggregating the dataset from 1 million down to 620 by grouping them using the number of followers as shown in Figure 16.

4.3 Data Preprocessing

Having all features analyzed, we continue with the data preprocessing section of our study. We need to prepare the data so that it can be leveraged by a machine-learning model that predicts the number of followers at the playlist level. The information is aggregated at the playlist level using the average function. In this way, we have a reduced number of attributes that give a compact representation of our data.

The dataset is difficult to model because of two main problems: the target column described in Table 2 has a long tail and is highly unbalanced. This issue was fixed by employing three transformations:

- **Remove 0.1% of large outliers** which correspond to the 10,000 playlists that have more than 75 followers. We experimented with different thresholds, but we chose this one because higher values would have removed the large majority of observations with different numbers of followers and we could not model this problem as a regression one.
- **Undersample playlists** represented by equalizing the number of playlists with one and two followers, by randomly removing from the first category. This method eliminates 604,000 data points.
- **Apply log transformation** to the target distribution to remove the skewness effect, placing values in the interval [0, 11.17].

The methods presented above are optional data transformations. They are applied before assembling features into vectors using VectorAssembler, as required by PySpark models, standardizing features with the formula $\frac{x-\mu}{\sigma}$ and splitting data into train-validation and test sets. The features are rescaled to ensure a fair treatment of all of them. Figure 21 illustrates the chain of data transformations.

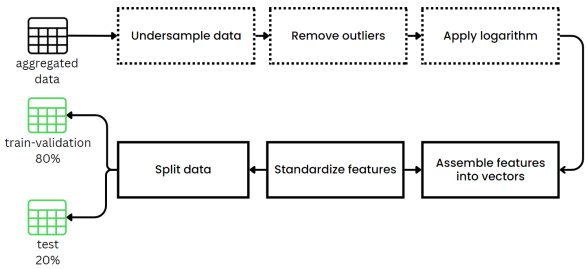

Figure 21: Data preprocessing steps

4.4 Modeling and Optimization

Our model choice is based on two properties: to be trainable in a distributed fashion, so that we can conduct many experiments quickly, and to be non-linear. The second attribute is mandatory because as seen in Figure 19 and Figure 20, there is no linear correlation between features and target. Constrained by the regression models available in PySpark, we picked a Random Forest Regressor (RFR) (Breiman, 2001). Besides meeting our needs, it is robust to overfitting, handles outliers, is interpretable, and performs feature selection intrinsically.

We trained the model and optimized it by performing a grid search over four hyperparameters:

- numTrees - [64, 96, 128]
- subsamplingRate - [0.5, 0.7, 1.0]
- minInstancesPerNode - [1, 5, 10, 15, 20]
- featureSubsetStrategy - ["sqrt", "all"]

We chose the number of a maximum of 128 trees because after this threshold the performance difference between using n of 2n trees is not distinguishable (Oshiro et al., 2012). As shown in (Sun et al., 2019), the minimum number of observations in a leaf node is the most important hyperparameter to be adjusted, so we picked the most values for it. The other two hyperparameters ensure the diversity of the built trees.

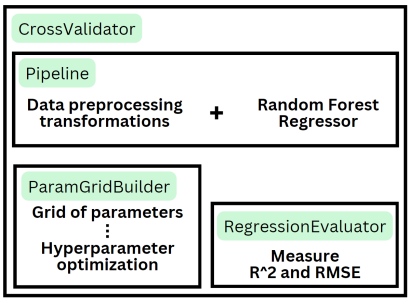

Figure 22: Training and optimization process

Figure 22 shows the PySpark classes, highlighted with green, used to train and optimize the Random Forest Regressor. The data preprocessing and the model are placed into a pipeline. This pipeline, together with the grid of hyperparameters and an evaluator are inserted into a 3-fold cross-validation procedure. This is used to report the accurate performance on the entire train-validation set. We are measuring the coefficient of determination R^2 (Equation 1) and the Root Mean Squared Error (RMSE) (Equation 2), but we are optimizing the model according to the maximization of R^2 because it is more straightforward to compare its values across different versions of the dataset.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \quad (1)$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}} \quad (2)$$

After running this process implemented completely in PySpark, we get the hyperparameters of the best model and the feature importance determined by it.

5 Results

We performed various experiments in which we turned on and off the optional transformations that solve the data distribution issue mentioned in section 4.3. To assess the effectiveness of the model, we compare it with a baseline model that always predicts the average follower count of train-validation playlists.

The experiments were conducted using five executors, each having 4 cores and up to 8GB of RAM. We tried to minimize the variance between experiments and to make them as deterministic as possible by setting a seed. However, due to the distributed nature of PySpark, there still are variations between experiments. For example, the split into train-validation and test data does not yield the same results every time, even if we use a seed.

On average, the model optimization took around 22 minutes, when we undersampled the data and removed the outliers, and 82 minutes when we used the entire dataset.

Figure 23: Performance of the model trained with different versions of the data

Figure 23 shows the R^2 measured when the Random Forest model was trained with different variations of the data. We can observe that the model does not overfit the data in any of the experiments. When the model is trained with the original data, it is worse than the baseline model. Outlier removal or target deskewing help the model perform slightly better. Even though data undersampling makes the model not as good as the baseline, when combined with the other two methods, it raises the metric to 0.058 (all techniques).

The model optimization improves the performance on the test data by around 4.29%, on par with the average expected improvement of 3-5% obtained after optimizing Random Forest models (Olson et al., 2018). Its best hyperparameters, in the order they were presented in section 4.4, are 128, 0.5, 20, and 'all'.

Even when we consider the maximum metric measured, the model is very weak, not being able to learn the data properly. This has to do with the extreme data imbalance and the skewness of the target that cannot be fixed by tested transformations.

When it comes to the $RMSE$ metric, its value depends on the range of values from the target variable, in our case, the number of followers. Table 3 shows the performance of the RFR compared with the baseline in two experiments. Again, when the original data was used, the baseline was better than RFR, but when all data transformations were applied, our model surpassed a naive predictor.

experiment type	RFR cross-val	RFR test	baseline train-val	baseline test	followers range
all techniques + optimization	0.585	0.587	0.603	0.608	0 - 4.21
original data	123.46	130.84	139.45	72.27	0 - 71643

Table 3: RMSE metric for two experiments

Even though based on the results the current dataset cannot be used to train a model, we can leverage the explainability of RFR by having a look at what features it considers important. Figure 24 shows that regardless of the learned version of the data, the most important attributes are the same. It turns out that the number of albums and tracks are the most influential, followed by the audio features valence and danceability. When all transformations were applied and we trained the model with the remaining 386,000 data points, half of the audio features did not influence the model at all and a bigger importance was assigned to the number of albums, meaning that they truly matter.

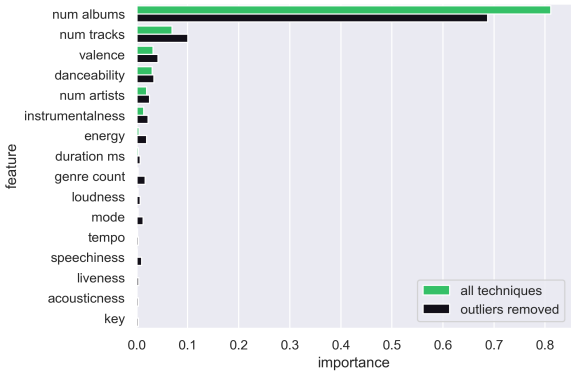

Figure 24: Feature importance given by the RFR

6 Discussion and Conclusions

6.1 Limitations and Challenges

While our report has promising results, it is important to acknowledge certain limitations that impact the predictions of our model. One of them is the skewness of our target data, which makes our job difficult to achieve (our model does not have enough examples with playlists that have many followers). A second limitation is that our audio feature data are too generic and no data pattern was found between them. That is why the high-level aggregated features have no predictive power. One last thing that we need to keep in mind is that our study relies on data obtained through the Spotify API and Spotify Million Playlist dataset. Any inconsistencies and

inaccuracies have an impact on the model's predictive capabilities.

Because the problem with the current dataset cannot be modeled as a regression one, we tried to approach it from a classification perspective. We tried to transform the target column into classes using binning. If we create two classes, one of the playlists with one follower and the other with the rest, there is no value for this type of model that only predicts if a playlist has more than one follower. Moreover, there would be too much variance of features inside each class, leading to a poor-performing classifier. The issue of within-class variance appears even if we try to create more classes, by adding the playlists in different buckets. This would make the problem even harder because of the data imbalance.

6.2 Conclusions

We augmented the Spotify Million Playlist Dataset by storing the artist and audio feature data from Spotify API to decode the success of playlist popularity. After our investigation, we can state that the audio features do not matter so much when it comes to playlist popularity. What seems to be the most important factor is the artists' popularity and the number of albums included in that playlist. The high-level features are too generic to learn some patterns from them; therefore, further investigations may be needed with more detailed audio features. All the code we developed is available at this Github link.

6.3 Future Improvements

Future improvements for this project would involve the usage of audio feature extraction software as the features that we have are too high level and insufficient. For example, in (Martín-Gutiérrez et al., 2020) features such as MEL spectrograms, and MEL frequency cepstral coefficients which could be aggregated to playlist level. In addition to the audio features, other aspects to investigate include, for instance, how prominently a playlist is featured on Spotify. As explored in (Pachali and Datta, 2022), where the authors estimated that being featured on Spotify increased the daily playlist follower count by 0.95%. Another non-audio feature to explore would be the presence of prominent artists in playlists as is explored in (Interiano et al., 2018) as the "superstar" feature. The playlist dataset that we utilized only had data until the year 2017, hence a more updated dataset can also be part of a future approach to predicting playlist popularity.

References

Pablo Alonso-Jiménez, Xavier Favory, Hadrien Foroughmand, Grigoris Bourdalas, Xavier Serra, Thomas Lidy, and Dmitry Bogdanov. 2023. Pre-training strategies using contrastive learning and playlist information for music classification and similarity.

Leo Breiman. 2001. Random forests. *Machine learning*, 45:5–32.

Matt Devor. 2019. Predicting spotify song popularity. `https://github.com/MattD82/Predicting-Spotify-Song-Popularity`.

Myra Interiano, Kamyar Kazemi, Lijia Wang, Jienian Yang, Zhaoxia Yu, and Natalia L. Komarova. 2018. Musical trends and predictability of success in contemporary songs in and out of the top charts. *Royal Society Open Science*, 5(5):171274.

Faheem Khan, Ilhan Tarimer, Hathal Salamah Alwageed, Buse Cennet Karadağ, Muhammad Fayaz, Akmalbek Bobomirzaevich Abdusalomov, and Young-Im Cho. 2022. Effect of feature selection on the accuracy of music popularity classification using machine learning algorithms. *Electronics*, 11(21).

David Martín-Gutiérrez, Gustavo Hernández Peñaloza, Alberto Belmonte-Hernández, and Federico Álvarez García. 2020. A multimodal end-to-end deep learning architecture for music popularity prediction. *IEEE Access*, 8:39361–39374.

Randal S Olson, William La Cava, Zairah Mustahsan, Akshay Varik, and Jason H Moore. 2018. Data-driven advice for applying machine learning to bioinformatics problems. In *Pacific Symposium on Biocomputing 2018: Proceedings of the Pacific Symposium*, pages 192–203. World Scientific.

Thais Mayumi Oshiro, Pedro Santoro Perez, and José Augusto Baranauskas. 2012. How many trees in a random forest? In *Machine Learning and Data Mining in Pattern Recognition: 8th International Conference, MLDM 2012, Berlin, Germany, July 13-20, 2012. Proceedings 8*, pages 154–168. Springer.

Max J. Pachali and Hannes Datta. 2022. What drives demand for playlists on spotify? *SSRN Electronic Journal*.

Fakhitah Ridzuan and Wan Mohd Nazmee Wan Zainon. 2019. A review on data cleansing methods for big data. *Procedia Computer Science*.

Yunlei Sun, Huiquan Gong, Yucong Li, and Dalin Zhang. 2019. Hyperparameter importance analysis based on n-rrelieff algorithm. *International Journal of Computers Communications & Control*, 14(4):557–573.