Université Paris 1 Panthéon-Sorbonne

École d'économie de la Sorbonne | UFR - Economie (EES)

Scoring and Machine Learning Project

What Makes a Hit?

Exploring and Predicting Spotify Streams with Machine Learning Models

Bertrand K. Hassani

12 January 2025

Submitted by:

Jessie Cameron

Gabriela Moravcikova

Kateryna Zaichenko

Table of Contents

List of Figures

List of Tables

What Makes a Hit?

Exploring and Predicting Spotify Streams with Machine Learning Models

1. Introduction

The purpose of this study is to explore which key factors influence the number of streams for Spotify's most popular songs in 2023. It aims to both explain and predict streaming success by leveraging machine learning models, including LASSO regression, decision tree, random forest and gradient boosting. The analysis evaluates the performance and interpretability of these models, with findings discussed from both a data science perspective and a decision-making standpoint.

2. Data Sources and Description

The Spotify data is sourced from Kaggle (Elgiriyewithana, 2023) and provides a comprehensive list of the most popular songs on Spotify in 2023. The dataset includes 952 songs by 644 unique artists and contains a variety of variables offering insights into each track's attributes. These variables span basic metadata such as the track and artist names, release dates, and platform-related metrics (e.g., playlist and chart inclusions), as well as detailed audio features such as tempo, danceability, and energy.

Table 1 presents summary statistics for these variables. The average number of streams per song is approximately 514 million, with a wide range from 2,762 to over 3.7 billion. The number of contributing artists per song ranges from 1 to 8, with solo tracks being the most common. While the majority of songs were released in recent years (predominantly in 2019), the release years span from as early as 1930 to 2023. Songs were evenly distributed across all 12 months, with June being the average release month, and the typical release date falling around the 14th of the month.

In terms of platform metrics, songs appear far more frequently in Spotify charts than in playlists. On average, a song appeared in Spotify charts 5,022 times compared to just 12 times in playlists. Apple and Deezer platforms show lower chart and playlist inclusions by comparison.

The audio features provide further insights into the musical characteristics of these tracks. The average tempo is 122.6 beats per minute (BPM), ranging from 65 BPM (slower tracks) to 206 BPM (fast-paced tracks). The average danceability and energy scores are both around 65%, indicating that most tracks are rhythmically engaging and high-energy, catering to listener enjoyment and movement. Conversely, the average acousticness (27%) and instrumentalness (1.6%) are low, highlighting the dominance of electronically produced, vocal-centric tracks in the dataset.

Table 1 – Summary Statistics

Variable	**Mean**	**Std. dev.**	**Min.**	**Q1**	**Median**	**Q3**	**Max.**
Streams	514.14	566.86	0	141.64	290.53	673.87	3703.90
Artist count	1.56	0.89	1	1	1	2	8
Release date							
Released year	2018.29	11.01	1930	2020	2022	2022	2023
Released month	6.04	3.56	1	3	6	9	12
Released day	13.94	9.20	1	6	13	22	31
Playlists and charts							
In Spotify charts	5202.57	7901.40	31	874.50	2216.50	5573.75	52898
In Spotify playlists	12.02	19.58	0	0	3	6	147
In Apple charts	51.96	50.63	0	7	38.5	87	275
In Apple playlists	67.87	86.47	0	13	34	88	672
In Deezer charts	385.54	1131	0	13.0	44	164	12367
In Deezer playlists	2.67	6	0	0	0	2	58
In Shazam charts	56.91	157.51	0	0.00	2.00	33.25	1451
Audio features							
Bpm	122.6	28	65	99.75	121	140.25	206
Danceability (%)	67	15	23	57	69	78	96
Energy (%)	64.27	16.56	9	53	66	77	97
Acousticness (%)	27.08	26.00	0	6	18	43	97
Instrumentalness (%)	1.58	8.41	0	0	0	0	91
Liveness (%)	18.21	13.72	3	10	12	24	97
Speechiness (%)	10.14	9.92	2	4	6	11	64
Valence (%)	51.41	23.48	4	32	51	70	97

Notes: This Table presents the descriptive statistics of the variables related to Spotify streams. The sample consists of 952 songs representing the most popular tracks on Spotify in 2023.
Source*:* Kaggle

Figure 1 illustrates the top 10 most streamed songs on Spotify. The most streamed song, *Blinding Lights* by The Weeknd, surpassed 3.5 billion streams. Other notable tracks include *Shape of You* by Ed Sheeran and *Someone You Loved* by Lewis Capaldi, both exceeding 2 billion streams.

Figure 1 – Top 10 Songs by Spotify Streams

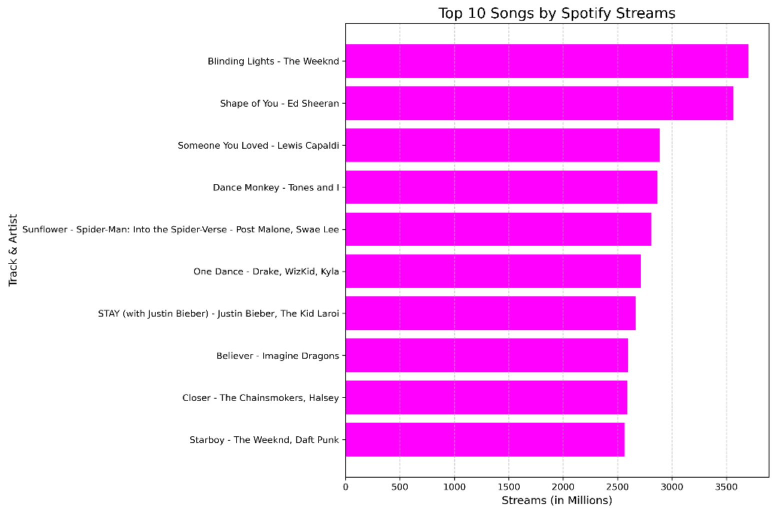
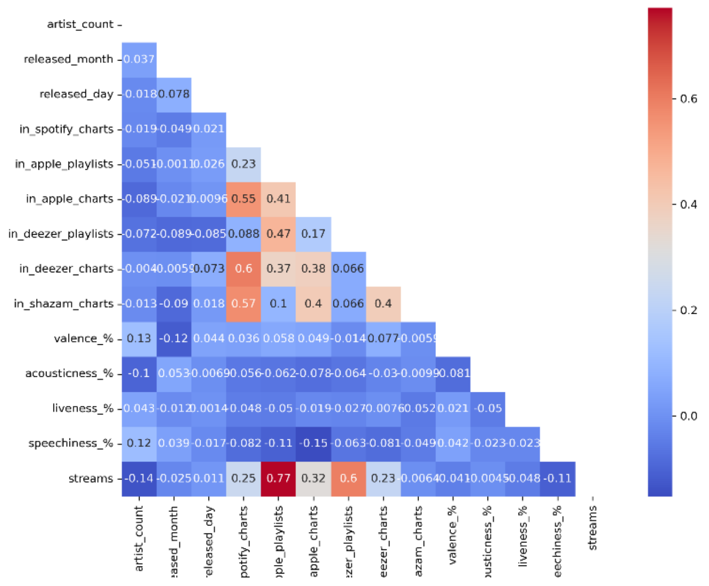


Figure 2 presents the correlation matrix between various features and the target variable. A strong positive correlation of 0.77 is observed between *in_apple_playlists* and streams, suggesting that tracks featured in Apple playlists are likely to experience higher streaming numbers. *in_spotify_playlists* also shows a moderate positive correlation with streams (0.32), reinforcing the impact of playlist inclusion on streaming numbers. Additionally, there is considerable multicollinearity among playlist-related features, with *in_spotify_playlists*, *in_apple_playlists*, and *in_deezer_playlists* exhibiting high correlations with one another.

Figure 2 – Correlation Matrix



Multicollinearity is a common challenge in regression analysis, occurring when two or more independent variables are highly correlated. This issue inflates the standard errors of regression coefficients, resulting in unreliable estimates, reduced statistical significance, and potentially misleading conclusions. One of the most widely used methods for detecting multicollinearity is the Variance Inflation Factor (VIF), which quantifies the severity of multicollinearity by measuring how much the variance of a regression coefficient is inflated due to correlations between predictor variables. Generally, a VIF value exceeding 5 indicates a high level of multicollinearity. Table 2 presents the VIF scores for all features in the dataset. After analyzing the correlation matrix and VIF scores, the following features were removed: *released_year*, *bpm*, *danceability_%, energy_%,* and *in_spotify_playlists*.

Table 2 – Multicollinearity Test

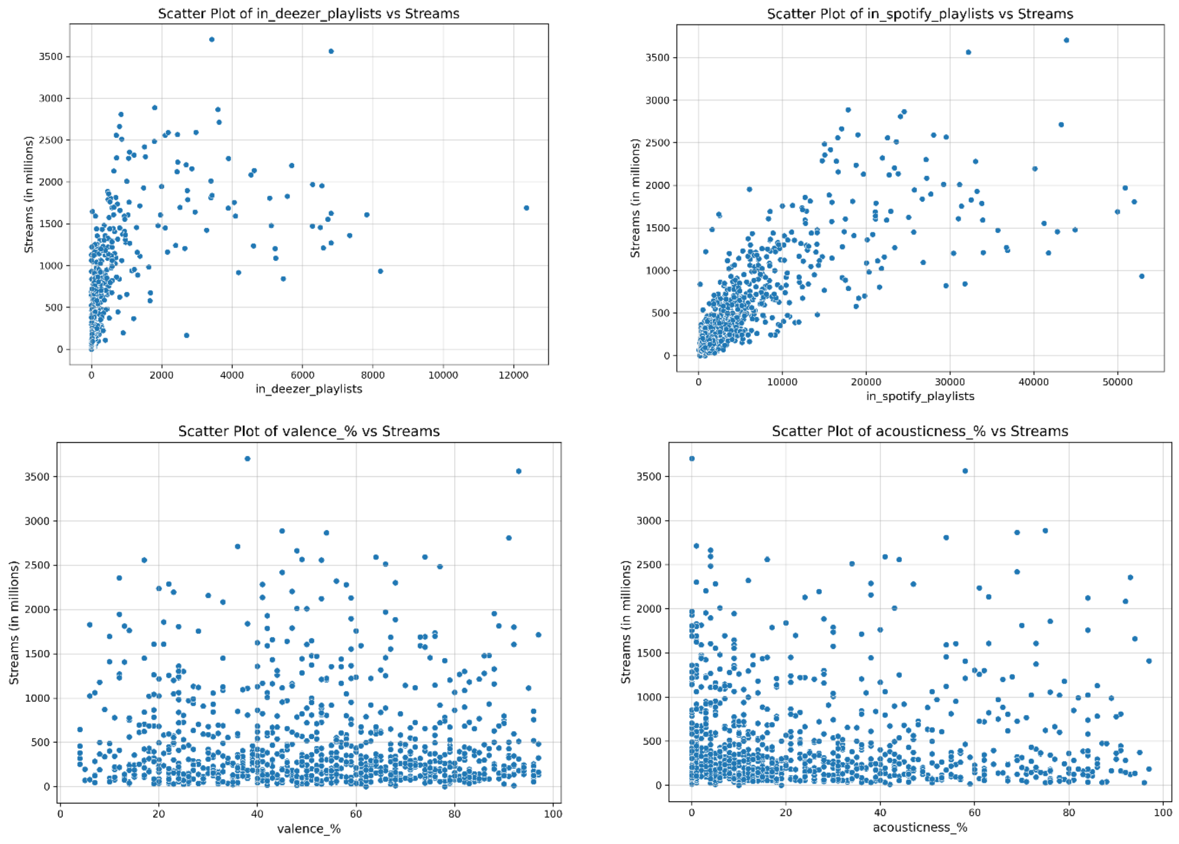
Feature	VIF Before	VIF After
artist_count	4.40	3.77
released_year	99.91	-
released_month	4.13	3.46
released_day	3.42	3.07
in_spotify_playlists	8.29	-
in_spotify_charts	3.21	3.13
in_apple_playlists	4.65	2.78
in_apple_charts	3.56	3.42
in_deezer_playlists	3.92	1.49
in_deezer_charts	2.20	2.12
in_shazam_charts	1.78	1.76
bpm	21.24	-
danceability_%	32.10	-
valence_%	8.57	4.77
energy_%	29.56	-
acousticness_%	3.45	1.92
instrumentalness_%	1.08	1.05
liveness_%	2.88	2.52
speechiness_%	2.22	2.04

The scatter plots illustrating the relationship between Spotify streams and selected feature variables are illustrated in Figure 3, with additional feature variables generated and available in the code. The relationship between *in_deezer_playlists* and streams suggests a non-linear trend. For instance, at lower values of *in_deezer_playlists*, there is a modest upward trend, indicating that streams generally increase as the number of playlists featuring a song rises. However, beyond approximately 2,000 playlists, the trend begins to flatten, reflecting diminishing returns where additional playlist inclusions result in less significant increases in streams. This relationship may be further influenced by a few outliers with extremely high playlist counts and varied stream levels, adding complexity to the observed trend.

Similarly, the relationship between *in_spotify_playlists* and streams also shows signs of tapering off at higher values, suggesting that the marginal impact of additional playlist inclusions diminishes as the number of playlists increases. However, the relationship appears stronger than that of *in_deezer_playlists*, as evidenced by the more pronounced upward trend, where higher playlist counts on Spotify are more consistently associated with increased streams.

In contrast, no clear trends are observed between *valence_%* and *acousticness_%* and the number of Spotify streams. The scatter plots for these variables display data points randomly distributed across their respective ranges, indicating that these features alone do not strongly influence streams. While there may be some clustering of lower streams at lower levels of *acousticness_%,* the lack of a discernible pattern suggests that any influence these features may have is likely minor or dependent on interactions with other variables.

Figure 3 – Relationship between Spotify Streams and Feature Variables



3. Model Selection

This section describes the machine learning models employed to identify and analyse the key factors influencing the number of Spotify streams.

3.1 Champion Model

Gradient boosting is selected as the champion model for this study because of its suitability for structured datasets, such as the Spotify data used in this analysis, and its ability to capture complex, non-linear relationships with high precision. It effectively models the non-linear relationship observed between *in_deezer_playlists* and streams, including the diminishing returns observed as playlist inclusions increase, as well as the weaker non-linearity associated with *in_spotify_playlists*. By accounting for these complexities, gradient boosting handles the varied impact of platform-specific features, which is a challenge for simpler linear models.

In addition to its predictive capabilities, gradient boosting provides valuable insights into feature importance. For example, features such as *in_apple_playlists*, which show a strong positive correlation with streams (0.77), and *in_spotify_playlists*, which exhibit moderate correlation (0.32), can be identified as key predictors. Moreover, its tree-based approach makes gradient boosting less sensitive to multicollinearity compared to traditional regression models, as it naturally selects the most relevant feature splits, mitigating the effects of highly correlated variables such as playlist inclusions across different platforms.

The model's customizability further enhances its suitability for this study. Hyperparameters like the number of trees, learning rate, and tree depth can be fine-tuned to balance predictive performance and computational efficiency. This flexibility is particularly beneficial for the Spotify dataset, where descriptive statistics reveal a wide variability in streaming numbers and the influence of diverse song attributes and platform metrics.

Gradient boosting works by combining the predictions of multiple "weak learners" (typically shallow decision trees) to form a strong predictive model. Each tree is trained to correct the residuals of the previous model, enabling the algorithm to refine its predictions iteratively. The model minimizes a loss function (e.g. Mean Squared Error for regression tasks) using gradient descent. At each iteration t, the algorithm updates the model by adding a new tree $h_t(x)$ that minimizes the loss:

$$F_t(x) = F_{t-1}(x) + \eta \cdot h_t(x)$$

Where:

- $F_t(x)$: The model at iteration t.
- $F_{t-1}(x)$: The model at the previous iteration.
- η: The learning rate, which controls the step size of each update.
- $h_t(x)$: The new tree fitted to the negative gradient of the loss function at iteration t.

While gradient boosting is a power and versatile model, it has some limitations that need to be considered:

- Computational Intensity: Gradient boosting builds trees sequentially, which can increase training time, particularly with larger datasets. However, this limitation is manageable in the study since Spotify dataset is of moderate size.
- Sensitivity to Hyperparameters: Proper tuning of hyperparameters like the learning rate (η) and the number of trees ($n_{estimators}$) is critical to avoid overfitting or underfitting. This is especially relevant to this study, as varied performance across configurations are observed.
- Reduced Interpretability: While gradient boosting provides feature importance rankings, the model itself is less interpretable than simpler methods like LASSO or decision tree. This limitation is less significant in this study, as feature importance still offers sufficient insights for our analysis goals.

3.2 Challenger Models

This study evaluates three challenger models, detailed further below: LASSO regression, decision tree, and random forest.

3.2.1 LASSO Regression

The Least Absolute Shrinkage and Selection Operator (LASSO) regression is a regularization technique that enhances traditional linear regression by introducing a L1 penalty. This penalty term helps to mitigate overfitting by shrinking the coefficients of less significant predictors toward zero, effectively performing automatic feature selection. The LASSO objective function is:

$$min \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Where y_i is the observed value; $\hat{y}_i$ is the predicted value for each data point i; λ is the regularization parameter; and β_j are the coefficients of the predictors. The strength of the penalty is controlled by λ, with larger values resulting in more coefficients being shrunk to zero, thereby retaining only the most relevant predictors.

LASSO is particularly effective in addressing multicollinearity, as seen in the strong correlations between playlist features (Figure 2). By shrinking the coefficients of less relevant variables, LASSO reduces the number of predictors, simplifying the model. This not only enhances interpretability but also helps prevent overfitting. Such simplification is particularly valuable when predicting a target like *streams,* as it facilitates clearer and more actionable insights in a business context, where decisions must be based on easily understandable models. Although LASSO is typically used in high-dimensional datasets with more predictors than observations, it can still be highly beneficial in this study, where there are 14 predictors and 952 observations.

However, a potential issue with LASSO is selection bias, as it may arbitrarily choose one variable from a set of highly correlated predictors, potentially disregarding other important variables. Additionally, LASSO is sensitive to the scale of the variables, which can impact its performance. This concern is mitigated in the current analysis by standardizing all predictors before fitting the model. Finally, LASSO is susceptible to the influence of outliers, which can distort coefficient estimates and affect feature selection. Therefore, proper handling of outliers is essential to ensure the robustness and reliability of the results.

3.2.2 Decision Tree

A decision tree is a supervised learning algorithm that uses a tree-like structure to make decisions or predictions by recursively splitting the dataset into subsets based on feature values. It consists of:

- Decision nodes: Representing decisions or tests based on attributes.

- Branches: Indicating the outcomes of these decisions.
- Leaf nodes: Providing the final predictions or outcomes.

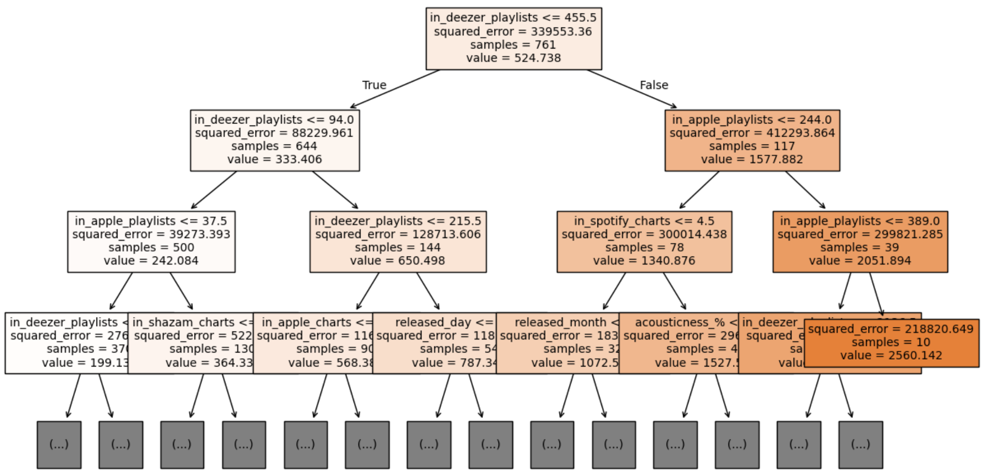
At each node, the algorithm selects the feature and threshold that best separate the data into target categories or minimizes the prediction error. Figure 4 presents a visual representation of the decision tree derived from this study’s data.

Decision trees are widely used because they closely mirror the human decision-making process, making them highly interpretable and easy to understand. They effectively model non-linear relationships and do not require data normalization or scaling, which simplifies the preprocessing steps. Additionally, similar to LASSO regression, decision trees perform implicit feature selection by prioritizing the most important splits based on criteria such as Gini impurity or information gain. They are also less sensitive to outliers, as splits rely on thresholds rather than the magnitude of feature values.

Despite these advantages, decision trees have notable limitations, including a tendency to overfit the training data. Moreover, they are sensitive to small changes in the dataset, which can lead to significant different tree structures, which reduces the model robustness.

These challenges can be addressed by using ensemble methods such as random forests and gradient boost, which improve predictive performance and robustness by combining multiple trees into a single, more reliable model.

Figure 4 – Decision Tree*


in_deezer_playlists <= 455.5
squared_error = 339553.36
samples = 761
value = 524.738
True
False
in_deezer_playlists <= 94.0
squared_error = 88229.961
samples = 644
value = 333.406
in_apple_playlists <= 244.0
squared_error = 412293.864
samples = 117
value = 1577.882
in_apple_playlists <= 37.5
squared_error = 39273.393
samples = 500
value = 242.084
in_deezer_playlists <= 215.5
squared_error = 128713.606
samples = 144
value = 650.498
in_spotify_charts <= 4.5
squared_error = 300014.438
samples = 78
value = 1340.876
in_apple_playlists <= 389.0
squared_error = 299821.285
samples = 39
value = 2051.894
squared_error = 218820.649
samples = 10
value = 2560.142
(...)

Notes: *Depth limited to three levels.

3.2.3 Random Forest

Random forest is an ensemble learning method that enhances predictive performance by combining the outputs of multiple decision trees. Each tree is trained on a randomly sampled subset of the data and considers a random subset of features when splitting nodes. This randomness reduces the risk of overfitting and improves the model's ability to generalize to new data. For regression tasks like this study, the final prediction is obtained by averaging the predictions across all trees. The random forest objective function is based on minimizing the Mean Squared Error (MSE) for regression tasks:

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

Where: y_i is the observed value, $\hat{y}_i$ is the predicted value for each data point i, and n is the number of observations.

Unlike linear models, random forest does not assume a specific functional relationship between predictors and the target variable, making it highly flexible for capturing complex patterns. This flexibility is particularly beneficial for the Spotify dataset, where variables such as playlist inclusions and song attributes interact in non-linear ways.

Random forest is particularly effective in handling datasets with a mix of numeric and categorical variables, allowing it to model the diverse characteristics of the Spotify data. For example, the dataset includes a mix of continuous variables like streams, percentages such as valence, as well as categorical-like features such as chart of playlist inclusion.

Additionally, random forest provides insights into feature importance by evaluating how much each feature reduces the impurity (e.g. MSE) across all trees in the ensemble. For this study, this is particularly useful for identifying the most significant drivers of streaming success. Another advantage of random forest is its robustness to multicollinearity and outliers. Unlike LASSO, random forest can handle correlated features such as playlist inclusions from different platforms. Furthermore, its ability to handle outliers is beneficial in predicting streams, which may have extreme values for popular tracks such as *Blinding Lights* and *Shape of You*.

Despite its strengths, random forest has some limitations. It can be computationally intensive, especially when the number of trees or features is large, as each tree must be trained independently. However, given the moderate size of the Spotify dataset, this limitation is manageable. Additionally, while random forest provides feature importance rankings, it lacks the straightforward interpretability of linear models, making it less suitable for applications where simple, easily explainable models are preferred. This limitation may be relevant if the results are presented to management or stakeholders who prefer simpler, more explainable models for decision-making.

4. Results and Analysis

This section presents the results of the machine learning models from a data science perspective. Key metrics, visualizations, and comparisons are used to evaluate the performance of the champion model and challengers.

4.1 Gradient Boosting

Gradient boosting combines the strengths of weak learners (shallow decision trees) to optimize predictive performance. Key hyperparameters – such as the number of boosting stages, the learning rate, and the maximum tree depth – were fine-tuned to achieve the best results:

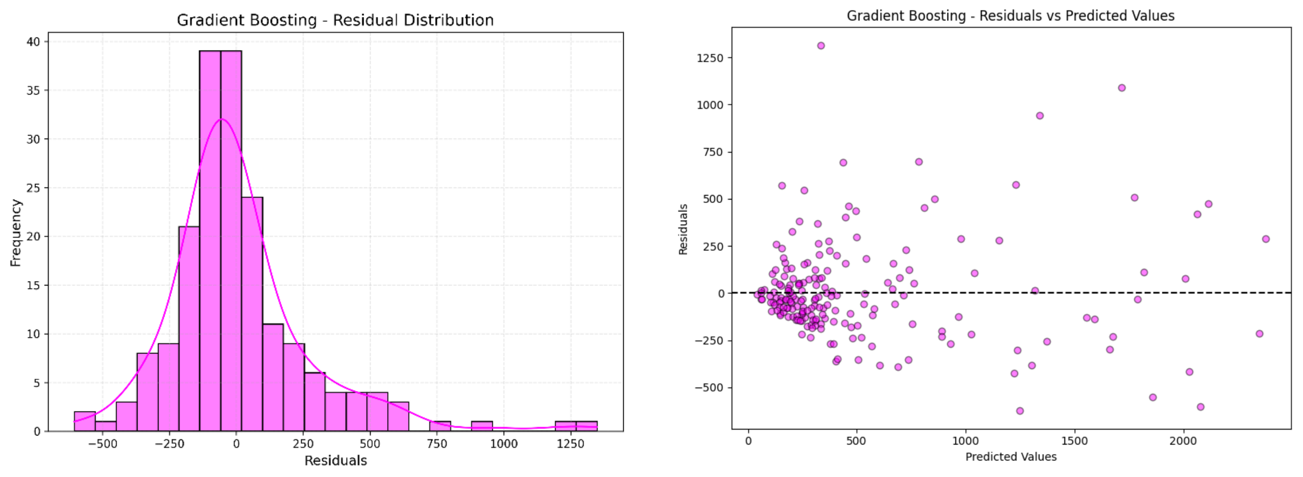
- ***n_estimators*****:** Increases the number of boosting iterations, improving accuracy but increasing computational cost.
- ***learning_rate*****:** Controls the contribution of each tree to the final model, ensuring gradual learning to avoid overfitting, though it requires a higher number of boosting stages to maintain accuracy.
- ***max_depth*****:** Limits the depth of individual trees to strike a balance between capturing complexity and avoiding overfitting.

A five-fold cross-validation procedure was performed to select the optimal hyperparameters. This involves dividing the training set into five folds, training the model on four folds, and validating it on the fifth. The process is repeated for all folds, and the RMSE is averaged to evaluate model performance and avoid overfitting. The final gradient boosting model was configured as follows:

- n_estimators = 300
- learning_rate = 0.05
- max_depth = 3

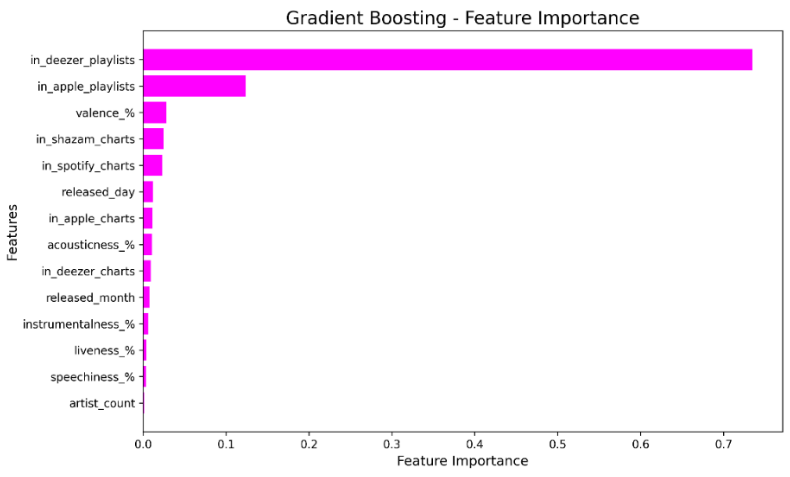
Residual analysis was conducted to evaluate the gradient boosting model's predictive accuracy and to check for systematic errors (Figure 5). These diagnostics suggest that the model is robust and appropriately balanced in terms of prediction accuracy and generalization. The left panel shows residuals that are approximately normally distributed and centred around zero. This indicates unbiased predictions with no systematic over- or underprediction. The right panel displays a random scatter of residuals around zero, with slight heteroscedasticity for higher predicted values. This suggests that while the model is well-specified for most cases, it struggles slightly with extreme streaming numbers.

Figure 5 – Gradient Boosting: Residual Diagnostics



Gradient boosting provides feature importance rankings by evaluating how much each feature reduces the error in predictions. Figure 6 highlights the dominant role of *in_deezer_playlists*, which accounts for 72.44% of total importance, emphasizes the critical role of playlist visibility on streaming numbers. Following this, *in_apple_playlists* contributes for 12.85% to the model, showing its significant but smaller influence compare to Deezer playlists.

Figure 6 – Gradient Boosting: Feature Importance



Interestingly, valence_% represents 2.65% of the importance. This indicates that the emotional tone of a song also impacts streaming behavior, albeit to a lesser degree. On the other hand, features such as *in_shazam_charts* and *in_spotify_charts* contributed moderately to the model, while variables like *artist_count* and *speechiness_%* had minimal impact.

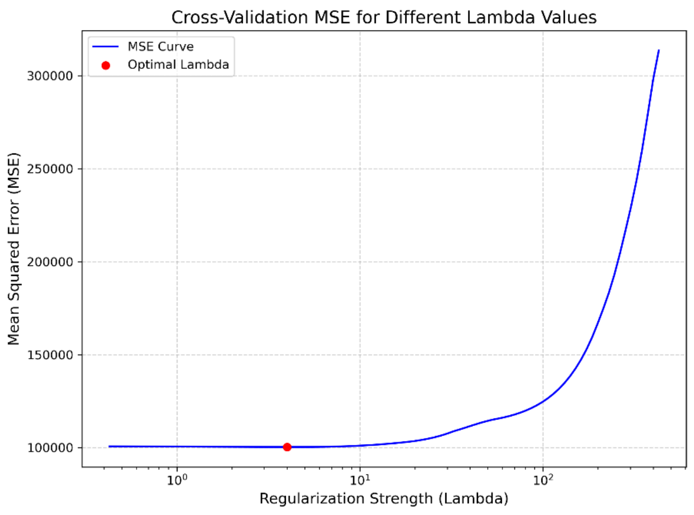
4.2 LASSO Regression

Choosing the optimal λ is critical to achieving the right balance between model complexity and prediction accuracy in LASSO regression. The λ parameter controls the strength of regularization, influencing how the model penalizes coefficients:

- Small λ: Weak regularization allows the model to fit the data more freely. While this can capture intricate patterns, it also risks overfitting, where the model learns noise instead of general trends.

- Large λ: Strong regularization excessively shrinks coefficients toward zero. While this reduces model complexity, it may lead to underfitting, where the model becomes too simplistic to explain the data adequately.

Figure 7 – LASSO Regression: Cross Validation MSE



To determine the optimal λ, a five-fold cross-validation procedure was performed. This technique divides the dataset into five equal parts (folds). The model is trained on four folds and validated on the remaining fold. This process is repeated across all folds, and the mean squared error (MSE) is averaged for each λ value. This ensures a robust estimate of the model's performance and reduces the likelihood of overfitting.

Figure 6 shows the relationship between λ and the cross-validated MSE. The MSE reaches its lowest point at the optimal λ = 4.0061, indicating a moderate level of regularization. At this value, the LASSO model effectively shrinks less important coefficients to zero while retaining the significant ones, representing the best trade-off between reducing overfitting and avoiding underfitting.

Residual diagnostics were conducted to evaluate the performance of the LASSO regression model (Figure 8). The residual distribution is centered around zero, indicating unbiased predictions. The residuals vs. predicted values plot shows a random scatter around zero, indicating no systematic bias in the model. The residuals exhibit relatively constant variance across predicted values, with minor signs of heteroscedasticity at larger predictions. These results suggest the model is well-specified and appropriately balances prediction accuracy with generalization.

Figure 8 – LASSO Regression: Residual Diagnostics

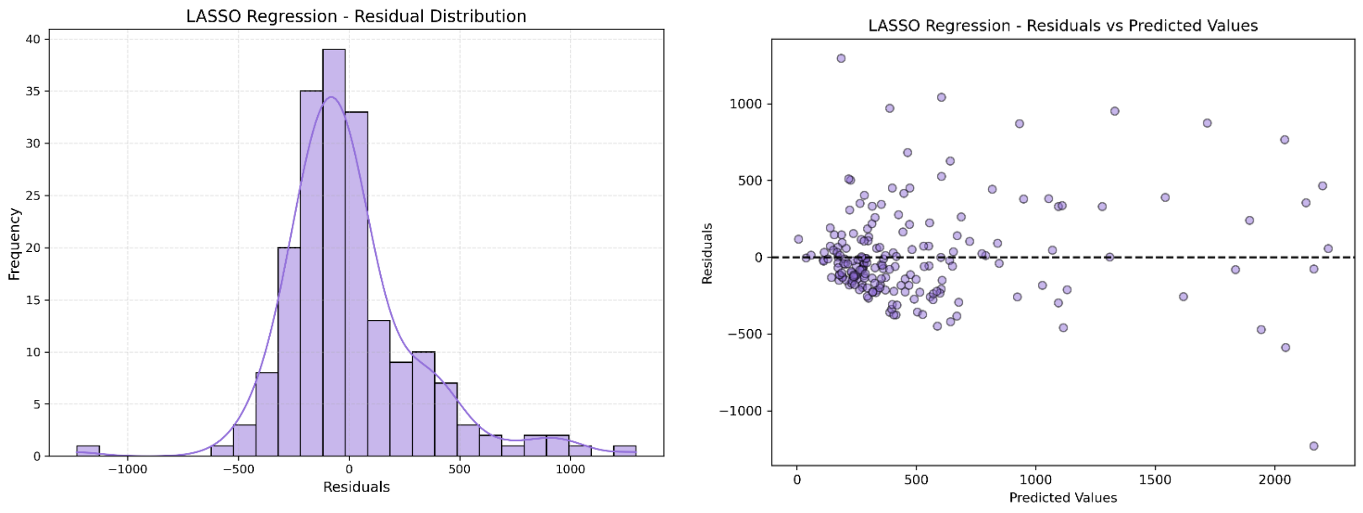
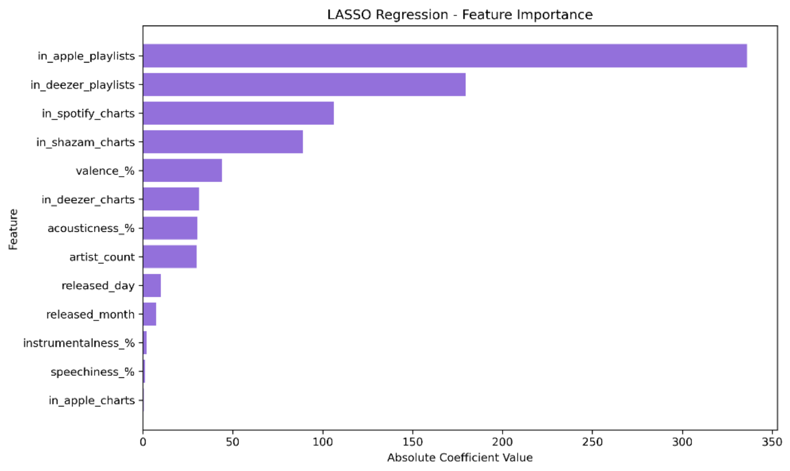


Figure 9 illustrates the absolute magnitude of the coefficients for each feature, highlighting their relative importance in predicting streams. The results indicate that *in_apple_playlists* has the largest absolute coefficient, making it the most influential predictor of streams. This is followed closely by *in_deezer_playlists*, *in_spotify_charts*, and *in_shazam_charts*, emphasizing the significant role of playlist and chart inclusions in driving streaming numbers. In contrast, features such as *instrumentalness_%, speechiness_%,* and *in_apple_charts* have coefficients near zero, suggesting they have minimal or negligible impact on the target variable.

Figure 9 – LASSO Regression: Feature Importance



4.3 Decision Tree

To determine the optimal specification for the decision tree model, a hyperparameter tuning process was conducted. Various combinations of key parameters, such as tree depth, minimum samples per leaf, and minimum samples for a split, were tested to identify a set of hyperparameters that minimized error and enhanced generalization. By using a grid search method with cross-validation, the final model specification was selected to represent the optimal combination of parameters.

To assess the model's ability to generalize to unseen data, R^2 scores were calculated for both the training and test datasets, as shown in Table 3. The training R^2 score of 0.8140 indicates that the model explains approximately 81% of the variance in the training data, demonstrating strong performance on the data it was trained on. However, the test R^2 was slightly lower at 0.6871, reflecting a drop in performance when applied to unseen data. This discrepancy suggests slight overfitting observed in the residual analysis.

Table 3 – Decision Tree R^2 Scores

Training (%)	81.40
Test (%)	68.71

To evaluate the performance and reliability of the decision tree model, a residual analysis was performed. The left panel of Figure 10 shows the distribution of residuals, which are centered around zero and follow an approximately normal distribution. This indicates that the model's predictions are unbiased, with errors distributed symmetrically. The right panel displays the residuals plotted against the predicted values. While many residuals are scattered randomly around the zero line, there are distinct gaps and extreme outliers, particularly for higher predicted values. This uneven distribution suggests potential overfitting, which is characteristic of decision trees.

Figure 10 – Decision Tree: Residual Diagnostics

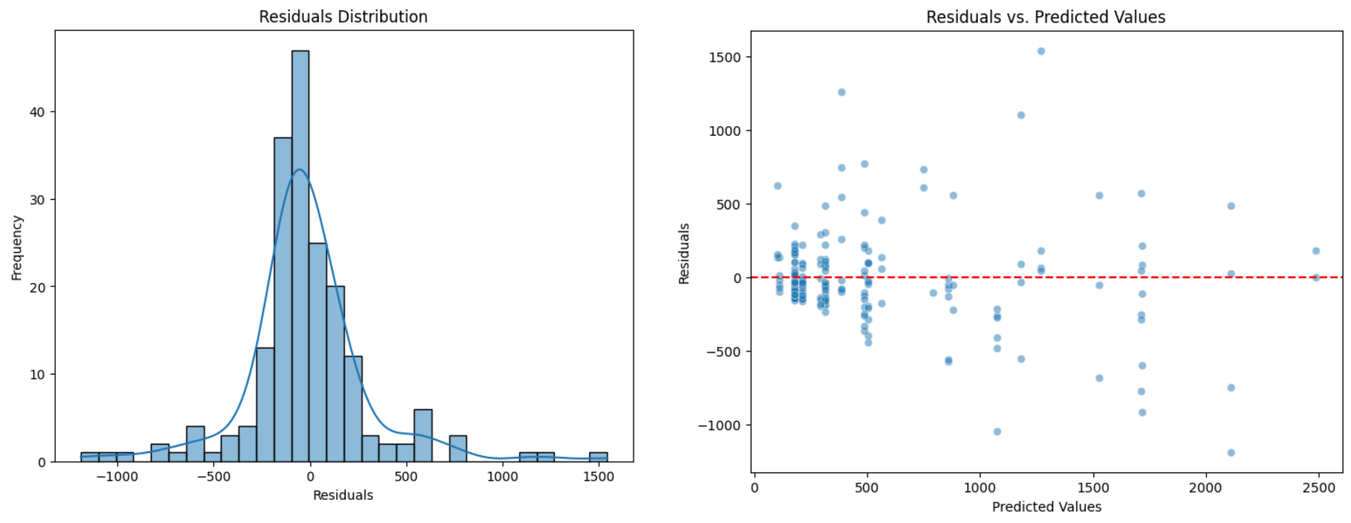
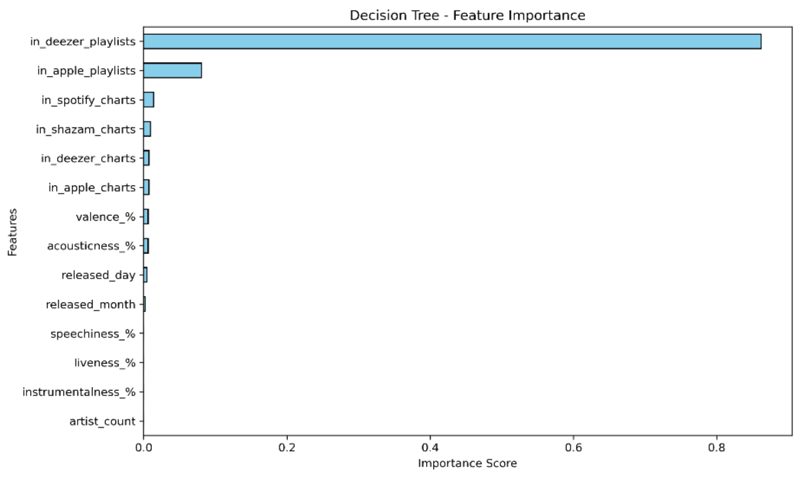


Figure 11 shows the most important features influencing the predictions of the decision tree model. The feature *in_deezer_playlists* is the most influential predictor, consistent with results observed in the gradient boosting model. Following this, *in_apple_playlists* and *in_spotify_charts* also display high importance scores, reflecting their strong contribution to predicting the number of streams.

Figure 11 – Decision Tree: Feature Importance



4.4 Random Forest

Choosing the optimal hyperparameters is critical for balancing model complexity and prediction accuracy in random forest regression. Hyperparameters such as the number of trees, the maximum depth of trees, and the number of features considered at each split determine the model's flexibility and performance.

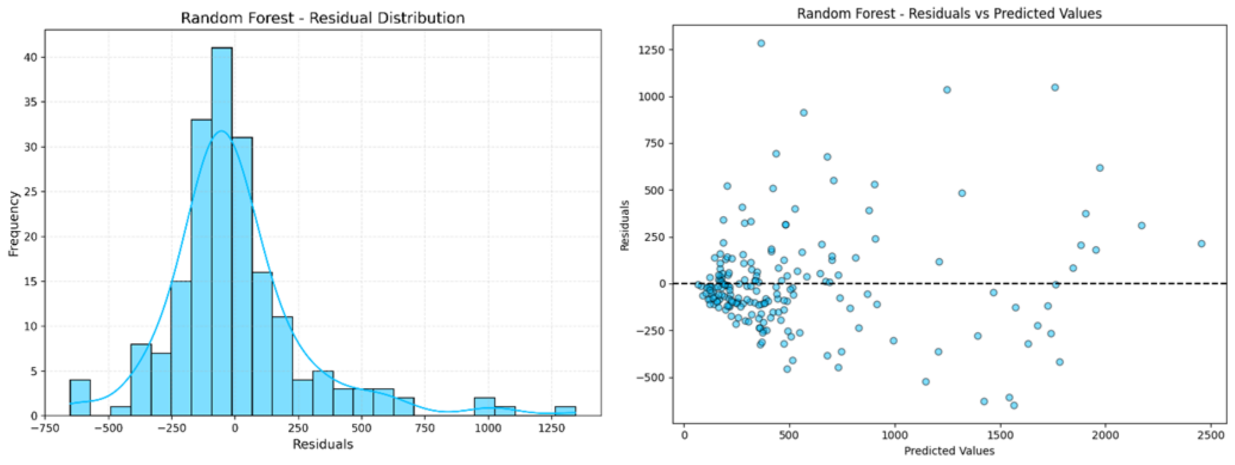
- Large Number of Trees (*n_estimators*)**:** Increasing the number of trees improves predictive accuracy but increases computational cost.
- Shallow Trees (low *max_depth*): Reduces the risk of overfitting but may lead to underfitting if the trees fail to capture data complexity.
- High (*max_features)*: Allows trees to consider more variables at each split, capturing complex relationships but potentially increasing overfitting.

Hyperparameter optimization for the Random Forest model utilized a five-fold cross-validation approach, as outlined before. This ensured reliable parameter selection and model performance. The final model was trained with the following specifications:

- *n_estimators* = 100
- *max_depth* = none
- *max_features* = auto

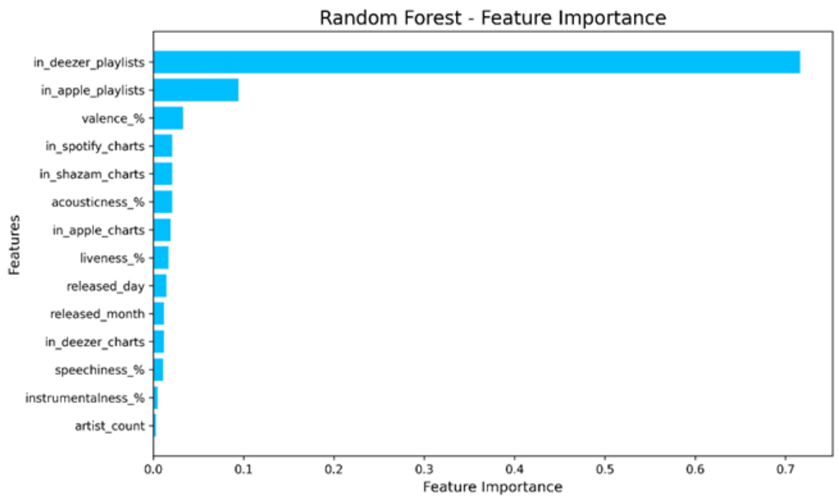
Residual diagnostics were conducted to evaluate the model's fit (Figure 12). The histogram of residuals is centered around zero. However, it exhibts a slight right skew, suggesting that the model occasionaly underpredicts for some instances with higher residual values (left panel). The right panel shows that most residuals are randomly scattered around the zero line, indicating no clear patterns or systematic errors in the models predictions. However, larger residuals are observed for higher predicted values, suggesting that the model struggles slightly with extreme cases or tracks with unusually high streaming numbers.

Figure 12 – Random Forest: Residual Diagnostics



Random forest provides insights into feature importance by evaluating how much each feature reduces impurity across all trees, as shown in Figure 13. The results highlight *in_deezer_playlists*, *in_apple_playlists*, and *valence_%* as the most important predictors of streaming success. Notably, the strong alignment between random forest and gradient boosting results underscores the robustness of these findings, with both models identifying *in_deezer_playlists* and *in_apple_playlists* as key drivers of streaming success. *ŭ*

Figure 13 – Random Forest: Feature Importance



4.2 Model Comparison

Table 4 presents the performance metrics across champion model, gradient boosting, and challenger models: LASSO, decision tree, and random forest. The results demonstrate that gradient boosting is the best-performing model for predicting Spotify streams. It achieves the highest R-squared value (79.19%), which indicates its ability to explain the largest proportion of variance in the target variable, Spotify streams. Additionally, it records the lowest Mean Squared Error (MSE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE), highlighting its superior predictive accuracy and reliability.

These results validate the selection of gradient boosting as the champion model. It's iterative learning process, which sequentially corrects errors from previous models, allows it to refine predictions and achieve high performance. Moreover, gradient boosting excels at capturing non-linear relationships, handling feature interactions, and maintaining robustness in the presence of noise. This makes the model particularly well-suited for the complexities of Spotify data.

In comparison, random forest performs well, achieving the second highest R-squared value (77.86%) and relatively low error metrics. This demonstrates its effectiveness in capturing non-linear relationships and interactions between features, a key strength of ensemble methods. However, gradient boosting surpasses random forest due to its ability to fine-tune predictions through iterative boosting, reducing residual errors and enhances model precision.

On the other hand, LASSO and decision tree exhibit weaker performance. Decision tree records the lowest R-squared value (68.71%) and the highest error metrics, reflecting its limited ability in capturing the complexities of the data. Decision Trees are prone to overfitting when their complexity is not controlled, and their reliance on a single tree structure limits their predictive power compared to ensemble methods. Similarly, LASSO, while simpler, less computationally intensive and effective for feature selection, struggles to capture non-linear relationships and interaction between variables. Consequently, LASSO does not achieve the same level of predictive accuracy as ensemble-based models.

These results highlight the advantage of ensemble methods, like gradient boosting, in capturing complex patterns and minimizing errors. This makes them well-suited for scenarios requiring high accuracy and reliability, such as this dataset.

Table 4 – Performance Metrics

Metric	LASSO	Decision Tree	Random Forest	Gradient Boosting
MAE	211.31	205.31	181.21	179.52
MSE	92761.36	103964.06	73546.34	69129.66
RMSE	303.85	322.435	271.19	262.92
R^2 (%)	72.08	68.71	77. 86	79.19

5 Discussion

These business implications are derived from the results of the champion model, which demonstrated superior performance across all metrics and showed consistency across different model specifications. The variable *in_deezer_playlists* emerged as the most influential variable, followed by *in_apple_playlists*, underscoring the important role of playlist incluisions in amplifying a song's exposure to listeners. For artists and producers, these results suggest that efforts should focus on engaging with playlist curators to increase visibility on platforms like Deezer, Apple Music, and Spotify. Playlist placement strategies, such as targeting popular or genre-specific playlists, can dramatically impact a track's reach and streaming numbers.

The emotional tone of a song, represented by *valence_%,* was identified as a moderate contributor to streaming success. This suggests that listeners may gravitate toward songs with specific emotional tones, which can influence streaming patterns. Artists and producers can use this insight to align song production with listener preferences, potentially increasing a track's likelihood of being streamed. For example, upbeat tracks with higher valence scores may be better suited for inclusion in mood-specific playlists, such as workout or party playlists.

While release dates did not dominate the feature importance rankings, they remain an essential consideration for marketing strategies. Descriptive statistics revealed a tendency for tracks to be released in recent years and an even distribution of release months, suggesting no single optimal timing. Although no single optimal timing emerged, artists and producers could benefit from analyzing seasonal trends or aligning releases with major events to maximize exposure

Further, features like *in_spotify_charts* and *in_shazam_charts* also contributed to streaming success, though to a lesser extent than playlist-related features. Chart inclusions indicate popularity and social proof, making them a secondary but still valuable factor for exposure. Spotify could use this information to improve its charting algorithms, ensuring placements genuinely reflect listener engagement and enhancing their effectiveness in driving streams.

While these results provide valuable insights, they should be interpreted with caution. Alhtough gradient boosting and random forest are effective in capturing non-linear relationships, their performance may decline when applied to significantly different Spotify datasets or in response to shifts in music consumption trends. Models trained on historical data may fail to adapt to shifts in listener behavior or platform algorithms. For this reason, artists and producers should view model predictions as helpful guides rather than guarantees, supplementing them with industry expertise and market knowledge.

In addition, these models are influenced by historical playlisting and streaming trends, which may not adequately represent emerging artists or new platforms. This limitation highlights the importance of continuously updating models and incorporating diverse datasets to reflect evolving trends and market dynamics.

Finally, external factors like cultural impact, social media trends, and marketing efforts, which can significantly influence streaming success, are not captured in this analysis. Decision-makers should remain aware of these unquantifiable factors and incorporate them into broader strategic planning.

6 Conclusion

This study examined the key factors influencing Spotify streaming success using machine learning models, with gradient boosting identified as the top performer across all metrics. The analysis revealed that playlist visibility, particularly *in_deezer_playlists* and *in_apple_playlists*, is the most critical driver of streaming numbers, followed by secondary factors like emotional tone *(valence_%)* and chart inclusions. These insights provide actionable recommendations for artists and producers to prioritize playlist placements, align song attributes with listener preferences, and strategically time releases.

While the models offer valuable insights, limitations such as overfitting, reliance on historical data, and the exclusion of external factors like social media trends and cultural influences must be acknowledged. Future studies could address these gaps by incorporating more comprehensive datasets and exploring adaptive algorithms to account for evolving listener behavior and platform dynamics.

References

Elgiriyewithana, N., 2023. Most Streamed Spotify Songs 2023 [Data set]. Kaggle. Available at: https://doi.org/10.34740/KAGGLE/DSV/6367938