# Introduction

The music industry is a highly competitive field, where artists constantly release songs with the hope of achieving widespread success. However, only a small percentage of these songs manage to gain significant popularity. One of the major challenges artists face is understanding the factors that contribute to a song's success. Being able to identify these factors can be invaluable for artists, producers, and marketers. This project aims to develop a predictive model that forecasts the popularity of songs based on various data inputs. By analyzing trends and patterns, the model will help artists recognize the key elements that make songs successful. The primary goal is to provide valuable insights into what drives a song’s popularity and how these insights can guide the creation of future hits. This research could revolutionize the industry by offering data-driven tools that increase the likelihood of success for new music. Ultimately, the model seeks to bridge the gap between artistic creation and market trends, empowering artists to craft music that resonates with broader audiences. The significance of this project lies in its potential to transform music production by combining data analysis with artistic creation. By accurately predicting the factors that drive song popularity, the project offers valuable insights that can help artists, producers, and marketers make more informed decisions. This approach could enhance both the creative and commercial aspects of the industry, making it easier for artists to align their work with audience preferences and improve their chances of success in the competitive music market.

# Methodology

## Data Collection

The dataset used was a Spotify dataset retrieved from Kaggle. This dataset was chosen due to the large amount of data it contained. The dataset contains information about songs, including attributes such as valence, energy, danceability, and track genre, alongside their popularity. The dataset was loaded as a pandas Dataframe.

## Data Cleaning

After loading the data, the DataFrame was displayed to observe the data it contained. This showed that the first five column; track ID, artist name, track name, album name, and an unknown column were irrelevant to the goal of the project. To focus on the relevant data, the dataset was filtered by removing the 5 irrelevant columns. After removing the columns, the new dataset was inspected using head(), describe(), and dtypes() functions to understand its structure, data types, and summary statistics. This step was to ensure familiarity with the dataset and identify any inconsistencies or missing values. No missing values could be seen from that inspection hence a more robust method was used to check whether there were any missing or NaN values in the dataset. This was the isnull().sum() method combination. This step was crucial for ensuring that incomplete data did not bias the model. After running this method, the result revealed that there were no missing values in dataset hence there was no missing value handling performed. After identifying no missing data, the two categorical columns, ‘explicit’ and ‘track\_genre’, were encoded using ‘LabelEncoder’. Label encoding assigns a unique integer to each category, which is suitable for algorithms that can interpret ordinal relationships or treat encoded labels as discrete categories. The encoded data was split into independent variables (music\_X) and dependent variables(music\_y) (Popularity)

## Exploratory Data Analysis (EDA)

**1. Top Genres by Popularity Analysis**

* **Visualization Insight:** A grouped bar chart highlights the distribution of features such as valence, energy, danceability, and acousticness across various track genres.
* **Key Observations:**
  + **Dance & Pop genres** dominate the metrics, particularly in energy and danceability. These genres are strongly associated with high-energy and rhythmic qualities, making them more appealing for mainstream audiences.
  + Genres like reggaeton and hip-hop also exhibit higher values in danceability and energy, indicating their popularity in dynamic and upbeat tracks.
  + Acoustic genres, such as piano, display lower danceability but higher acousticness, reflecting their soothing and instrumental nature.
  + Genres like indie, alternative, and folk have balanced features, appealing to niche audiences with a blend of rhythm and acoustic elements.

**2. Explicit Content and Popularity**

* **Visualization Insight:** The bar chart compares the average popularity of songs labeled as explicit versus non-explicit.
* **Key Observations:**
  + Songs with explicit content have **higher average popularity** compared to non-explicit ones. This suggests that explicit lyrics may resonate more with certain audiences, potentially due to their bold and unfiltered expression.
  + The gap, however, is not overwhelmingly significant, indicating that explicit content is only one of many factors influencing popularity.

**3. Top 20 Genres by Average Popularity**

* **Visualization Insight:** A horizontal bar chart ranks the top 20 track genres based on average popularity.
* **Key Observations:**
  + **Pop-film** and **K-pop** genres top the list, reflecting their global appeal and strong fan bases. The cinematic and high-production-value nature of these genres contributes to their widespread success.
  + Other highly ranked genres like chill, sad, and grunge suggest a growing trend toward emotionally resonant music, which deeply connects with listeners.
  + Niche genres like anime and emo indicate the influence of subcultures on music preferences, which often lead to dedicated followings.
  + Traditional genres such as sertanejo and regional genres like brazil and pagode show that cultural music continues to hold a significant place in global popularity.
  + Genres like metal and ambient rank lower, suggesting a more limited but dedicated audience base.

## Feature Selection and Correlation Analysis

In the feature selection and correlation analysis, the correlation between each independent variable and the target variable, popularity, was calculated. The features were then sorted based on their absolute correlation values to identify the most influential ones. A bar plot was used to visualize these correlations, emphasizing the key features that impact popularity. Given the small number of features, all were initially considered in the analysis. To improve model performance, feature scaling was performed using MinMaxScaler, which normalized the selected features to a range between 0 and 1. This step was crucial for models like Gradient Boosting and Random Forests, as scaling ensures that no single feature dominates due to its scale, thus enhancing the effectiveness of the models.

## Data Scaling

The ‘MinMaxScaler’ was applied to the selected features, transforming each feature to a range between 0 and 1. This ensured uniformity across all features, which is essential for distance-based models and algorithms sensitive to feature magnitude. Scaling prevents features with large ranges from dominating those with smaller ranges, thus improving model convergence and accuracy.

After scaling, the dataset was split into training and testing subsets using an 80-20 split with ‘train\_test\_split’. The training set is used to fit the models, while the testing set evaluates performance. The random state was set to 42 to ensure reproducibility of results.

## Model Training

Three regression models were trained in-order to identify the best performing model

1. **Random Forest Regressor**
2. **Gradient Boosting Regressor**
3. **An Enhanced Gradient Boosting Regressor**

**Random Forest Regressor**

The Random Forest Regressor was configured with the following parameters:

* **Number of trees (n\_estimators=100):** This determines the number of decision trees built in the forest.
* **Fixed random seed (random\_state=42):** Ensures reproducibility of results by controlling randomness.
* **Parallel processing (n\_jobs=-1):** Utilizes all available CPU cores to speed up the training process.

This model uses the randomness in tree construction and prediction aggregation. By averaging predictions from multiple trees, it reduces overfitting and improves the model’s ability to generalize to unseen data.

**Gradient Boosting Regressor**

The standard Gradient Boosting Regressor was configured with:

* **500 boosting stages (n\_estimators=500):** Determines the number of sequential trees added.
* **Maximum tree depth (max\_depth=15):** Controls the complexity of individual trees to prevent overfitting.
* **Minimum samples per split (min\_samples\_split=15):** Ensures splits occur only when there are sufficient samples, promoting simpler and more stable splits.
* **Learning rate (learning\_rate=0.05):** Regulates the contribution of each tree to the final prediction, striking a balance between accuracy and computational efficiency.

The Gradient Boosting model iteratively optimizes residual errors, where each tree corrects the errors made by the previous ones. This sequential refinement makes it highly effective for capturing complex patterns.

**Enhanced Gradient Boosting Regressor**

An improved variant of the Gradient Boosting Regressor was also tested, where:

* **The number of boosting stages was increased to 1000 (n\_estimators=1000):** This enabled the model to learn more complex relationships within the dataset.  
  This extended training enhances the model’s capacity to handle intricate data structures but comes with increased computational costs.

**Ensemble Approach: Voting Regressor**

To capitalize on the strengths of the individual models, a **Voting Regressor** was implemented. This ensemble technique combines predictions from the Random Forest and Gradient Boosting models, aggregating them into a single output. The rationale behind this approach is to leverage the diversity of the base models, resulting in a final model that is more robust and accurate.

**Performance Optimization via Parameter Variation**

* **Number of trees (n\_estimators):** Adjusting this value demonstrated how increasing the number of trees in Random Forest and Gradient Boosting impacted generalization and overfitting.
* **Maximum tree depth (max\_depth):** Modifications to this parameter showed how deeper trees allowed for capturing complex interactions at the risk of overfitting.
* **Learning rate (learning\_rate):** Experimenting with this value revealed the trade-off between model convergence speed and final performance.

# Detailed Analysis of Results

## Random Forest Regressor (RF)

The **Random Forest Regressor** demonstrated moderate performance with a test score of **0.8331** and a **Root Mean Squared Error (RMSE)** of **9.1134**. While the model captured patterns effectively, its performance declined as the number of estimators increased. This behavior aligns with the tendency of Random Forest models to face diminishing returns beyond a certain number of trees, as excessive estimators do not significantly enhance accuracy but increase computational demands.

* **Behavior with Increased Estimators:** When the number of estimators was set to **500**, the model's performance worsened. This outcome is attributed to the increased likelihood of overfitting to noise in the training data and the escalating computational cost of training and prediction. The results indicate that beyond a certain threshold, additional trees add redundancy rather than improving generalization.
* **Resource Consumption:** The RF model required substantial memory and processing time as the number of estimators increased. This trade-off highlights the need to carefully balance performance and resource constraints, particularly when scaling models to large datasets.

## Gradient Boosting Regressor (GBR)

The **Gradient Boosting Regressor** exhibited improved performance compared to the RF model, achieving a test score of **0.8759** and an RMSE of **7.8583**. GBR is inherently more focused on optimizing performance by sequentially correcting errors made by earlier trees, which explains its higher accuracy.

* **Effect of Parameters:**
  + **n\_estimators (500):** The large number of estimators provided the model sufficient iterations to learn complex relationships, contributing to its strong performance. However, a higher number of estimators increases computational time.
  + **max\_depth (15):** A deeper tree allowed the model to capture intricate patterns, but it also risked overfitting. In this case, the selected depth struck a balance between capturing relationships and generalizing well.
  + **learning\_rate (0.05):** A lower learning rate ensured incremental improvements at each stage, preventing the model from converging prematurely to suboptimal solutions.
* **Limitations:** Despite its strong performance, the GBR model is computationally intensive and sensitive to hyperparameter tuning. Misconfigured parameters can lead to underfitting or overfitting, emphasizing the importance of validation during model development.

## Improved Gradient Boosting Regressor (Improved GBR)

The **Improved GBR** was a fine-tuned version of the GBR and delivered the best performance among the individual models, achieving a test score of **0.8899** and an RMSE of **7.4006**. The enhancements included optimized hyperparameters, which allowed the model to learn more effectively.

* **Parameter Optimization:**
  + **n\_estimators (1000):** A larger number of estimators provided the model with more opportunities to refine predictions. However, this also significantly increased training time.
  + **max\_depth (15):** The depth was retained to allow complex relationships to be captured.
  + **min\_samples\_split (15):** Increasing this parameter reduced overfitting by requiring a minimum number of samples to split nodes, ensuring the model learned robust patterns.
  + **learning\_rate (0.05):** A conservative learning rate allowed gradual convergence, minimizing the risk of overshooting optimal solutions.
* **Strengths and Challenges:** Improved GBR performed well due to precise parameter tuning, but its computational demands were notably higher, requiring careful consideration in real-world applications with limited resources.

## Ensemble Voting Regressor

The **Voting Regressor**, which combined the predictions of the Improved GBR and RF models, achieved a test score of **0.8750** and an RMSE of **7.8861**. While the ensemble model balanced the strengths of its components, its performance was slightly inferior to the Improved GBR alone.

* **Model Averaging:** The ensemble’s slightly reduced performance suggests that the strong predictions of the Improved GBR may have been diluted by combining it with the RF model, which had a lower accuracy.
* **Resource Implications:** As an ensemble model, the Voting Regressor required more computational resources than individual models. However, it provided more robust predictions by leveraging complementary strengths.

## Insights

1. **Data Sensitivity:** The models generally showed high accuracy, with the Improved GBR achieving the best performance. However, variations in parameter configurations (e.g., number of estimators) highlighted the sensitivity of results to tuning.
2. **Resource Evaluation:**
   * The RF model became less efficient with higher estimators, emphasizing the importance of choosing optimal values for computational feasibility.
   * Both GBR and Improved GBR were computationally expensive but yielded better results due to their ability to capture complex relationships.
   * The ensemble model increased computational cost while balancing predictions but did not outperform the best individual model.
3. **Error Cases:**
   * Predictions with the RF model had the highest error variance, indicating potential weaknesses in capturing intricate patterns in the data.
   * GBR and Improved GBR effectively reduced error, highlighting the importance of sequential learning in gradient boosting algorithms.
4. **Bottlenecks and Recommendations:**
   * **Resource Bottlenecks:** Computational demands increased significantly with the number of estimators, particularly for GBR and Improved GBR.
   * **Data Limitations:** Models may have encountered challenges due to data sparsity in certain genres or features. Augmenting the dataset with more diverse samples could further improve accuracy.
   * **Algorithmic Challenges:** The trade-off between model complexity and generalization was evident, necessitating careful tuning to optimize performance without overfitting.

# Conclusion

Among the models trained, the Improved Gradient Boosting Regressor was the best performing model achieving a test score of 0.8899 and the lowest RMSE of 7.4006. The results demonstrate that sequential learning and fine-tuned parameters significantly enhance model accuracy while balancing computational efficiency. The insights from this project show the importance of data-driven decision-making in music production and marketing. Artists and producers can use these findings to align their creative efforts with market trends, optimizing both artistic and commercial outcomes.