

---

# **Predict whether a Spotify track will be popular or not based on its track-level and artist-level features**

**Nisha R S(A86605224258)**

**Datset name : Spotify Data Clean**

**Model used : Random Forest Classifier**

**Course: AI/ML Lab**

# Problem Statement

What prediction / classification problem are you solving?

This project solves a binary classification problem that predicts whether a Spotify track will be popular or not popular using track-level and artist-level features available in the Spotify data clean dataset.

Why is this problem relevant / important?

Predicting track popularity is important for music streaming platforms, artists, and record labels because it helps:

- Improve music recommendation systems
- Identify tracks with high potential popularity
- Support data-driven decisions in music production and promotion
- Accurate predictions enhance user engagement and content discovery.

What question(s) does your model answer?

The model answers the following questions:

Will a Spotify track become popular based on its features?

Which track and artist attributes influence popularity?

Can popularity be predicted using historical metadata?

# Dataset Overview

Dataset Source - Kaggle and contains cleaned Spotify track and artist metadata.

Number of rows - 8583 & columns - 16

Type of Data : Mixed data type

- Numerical:
- Categorical / Binary:

Target Variable : Track Popularity (Popular / Not Popular) - Converted into a binary target variable

Why this Dataset Was Chosen

1. Size Suitability:

- The dataset is large enough to train and evaluate machine learning models effectively, but still manageable for academic analysis.

2. Relevance to the Problem:

- It directly contains features related to track popularity, which aligns with the goal of predicting song success.

3. Ease of Modeling:

- The dataset is clean, structured, and contains mostly numeric features, making it suitable for classification models like Random Forest.

# Data Dictionary

Feature Name	Type	Description	Why Important for Popularity Prediction
track_duration_ms	Numerical	Duration of the track in milliseconds	Track length affects listener engagement; extremely short or long tracks may be less popular.
explicit	Binary (0/1)	Indicates whether the track contains explicit content	Explicit content can influence reach, audience size, and platform.
track_number	Numerical	Track's position in the album	Early tracks are often promoted more and may receive higher listener attention.
artist_popularity	Numerical	Popularity score of the artist on Spotify	Popular artists tend to attract more listeners, increasing track popularity.
artist_followers	Numerical	Number of followers the artist has	A larger follower base increases initial streams and visibility of tracks.
album_total_tracks	Numerical	Total number of tracks in the album	Album size can affect how much attention each track receives.
track_popularity	Numerical (Target source)	Popularity score assigned by Spotify	Used to derive the binary target variable (Popular / Not Popular).
popular	Binary (Target)	Indicates whether a track is popular or not	This is the target variable the model predicts.

- **Track Duration (track\_duration\_ms)** - Songs that are too long or too short may not be replayed often, which affects popularity.
- **Explicit Content (explicit)** - Explicit songs may have limited audience reach, while clean songs can be played by everyone.
- **Track Number (track\_number)** - Songs placed early in an album are usually listened to more than later tracks.
- **Artist Popularity (artist\_popularity)** - Popular artists already have many listeners, so their songs become popular faster.
- **Artist Followers (artist\_followers)** - More followers mean more people will listen to the song when it is released.
- **Album Total Tracks (album\_total\_tracks)** - If an album has many songs, each song may get less attention.
- **Track Popularity (track\_popularity)** - This score shows how popular a song is based on listener activity.
- **Popular (Target Variable)** - This is what the model predicts: whether the song is popular or not.

## Distribution Plots

- Used to understand data spread and check skewness in features like artist followers and track duration.

## Basic Statistics (Mean, Min, Max, Std)

- Used to summarize feature values and understand their range and variability.

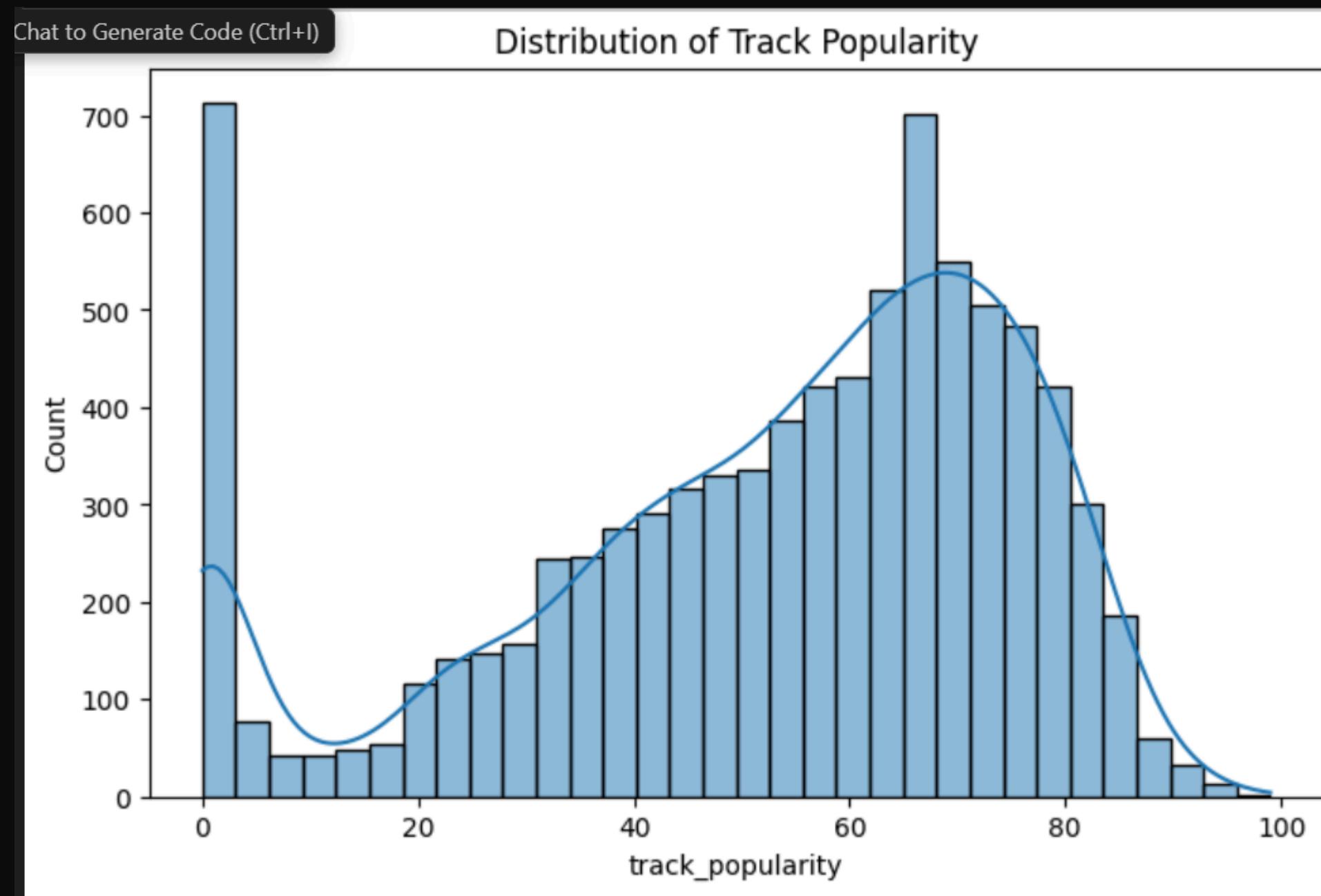
## Correlation Matrix / Heatmap

- Used to identify relationships between features and detect multicollinearity.

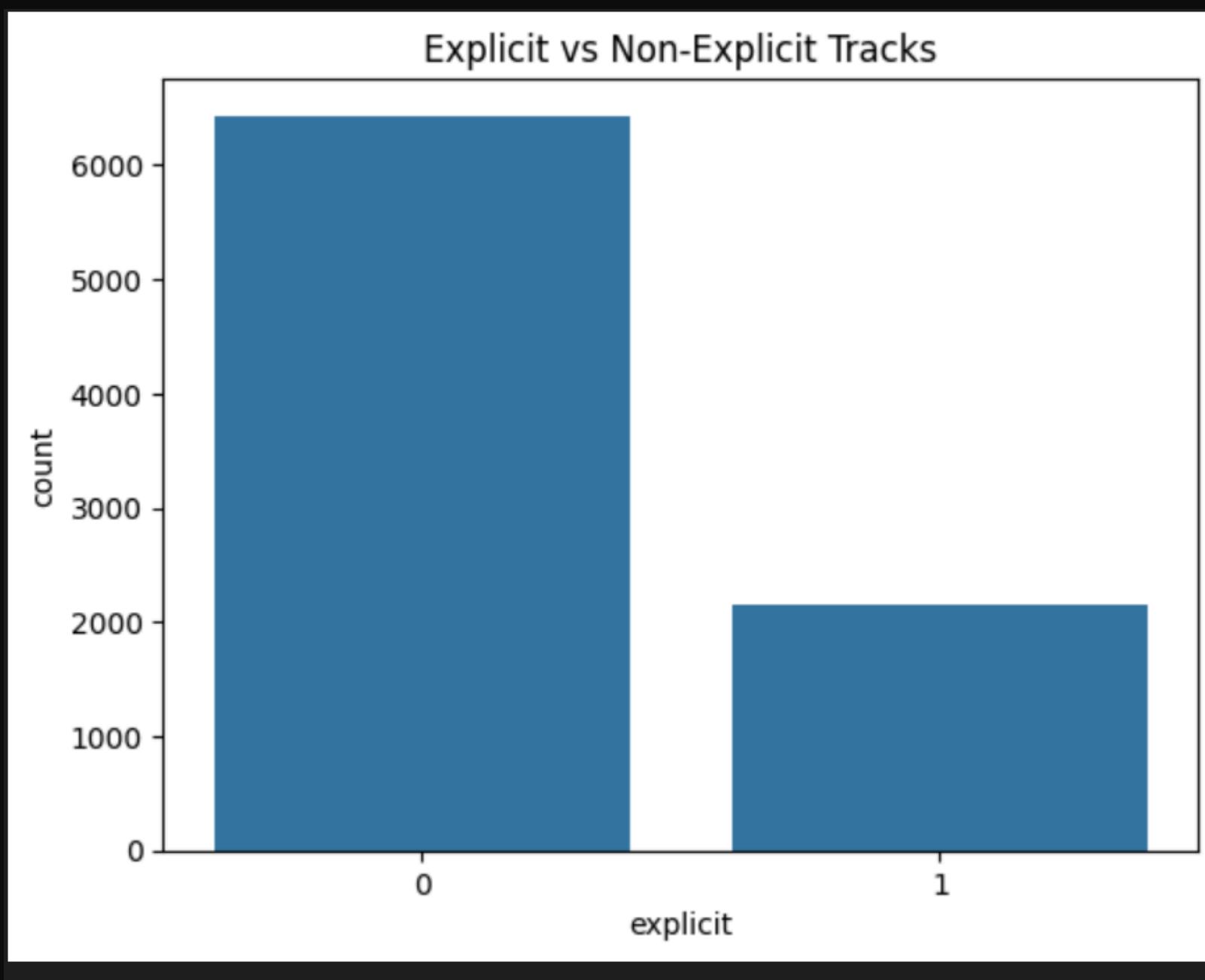
## Outlier Checks

- Used to detect extreme values that could affect model performance.

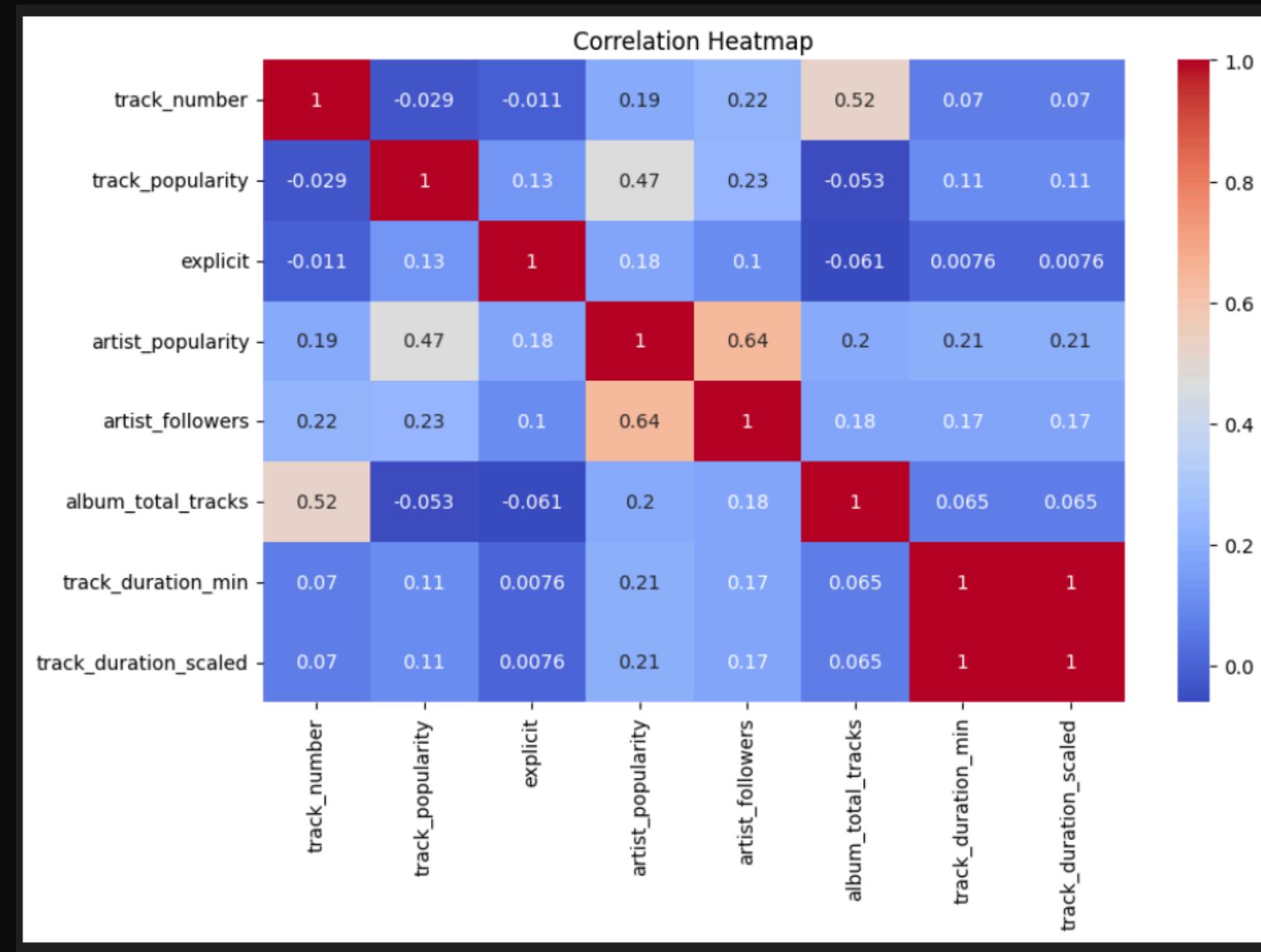
# Distribution of Track Popularity



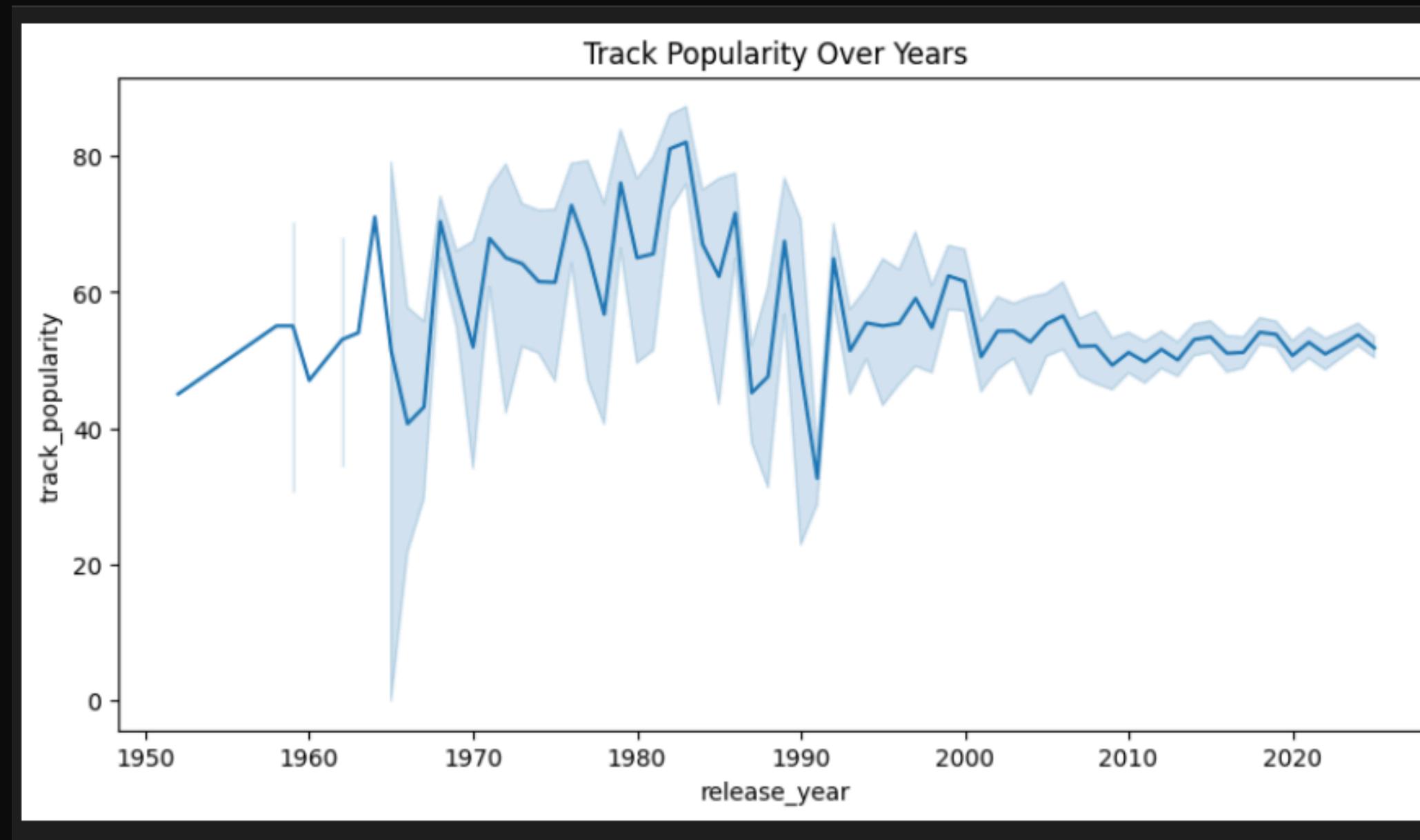
# Explicit vs Non-Explicit Tracks



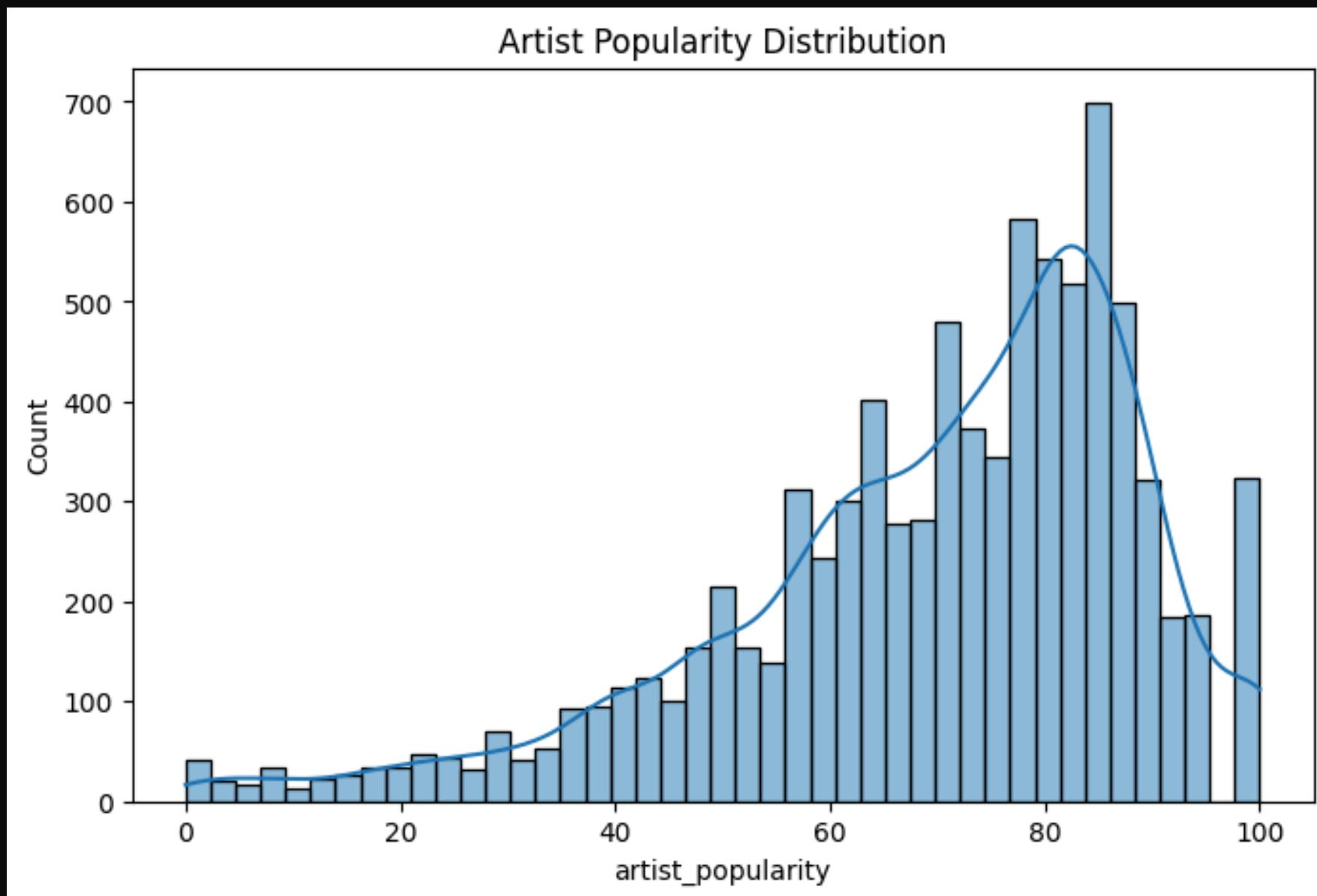
# Correlation Heatmap



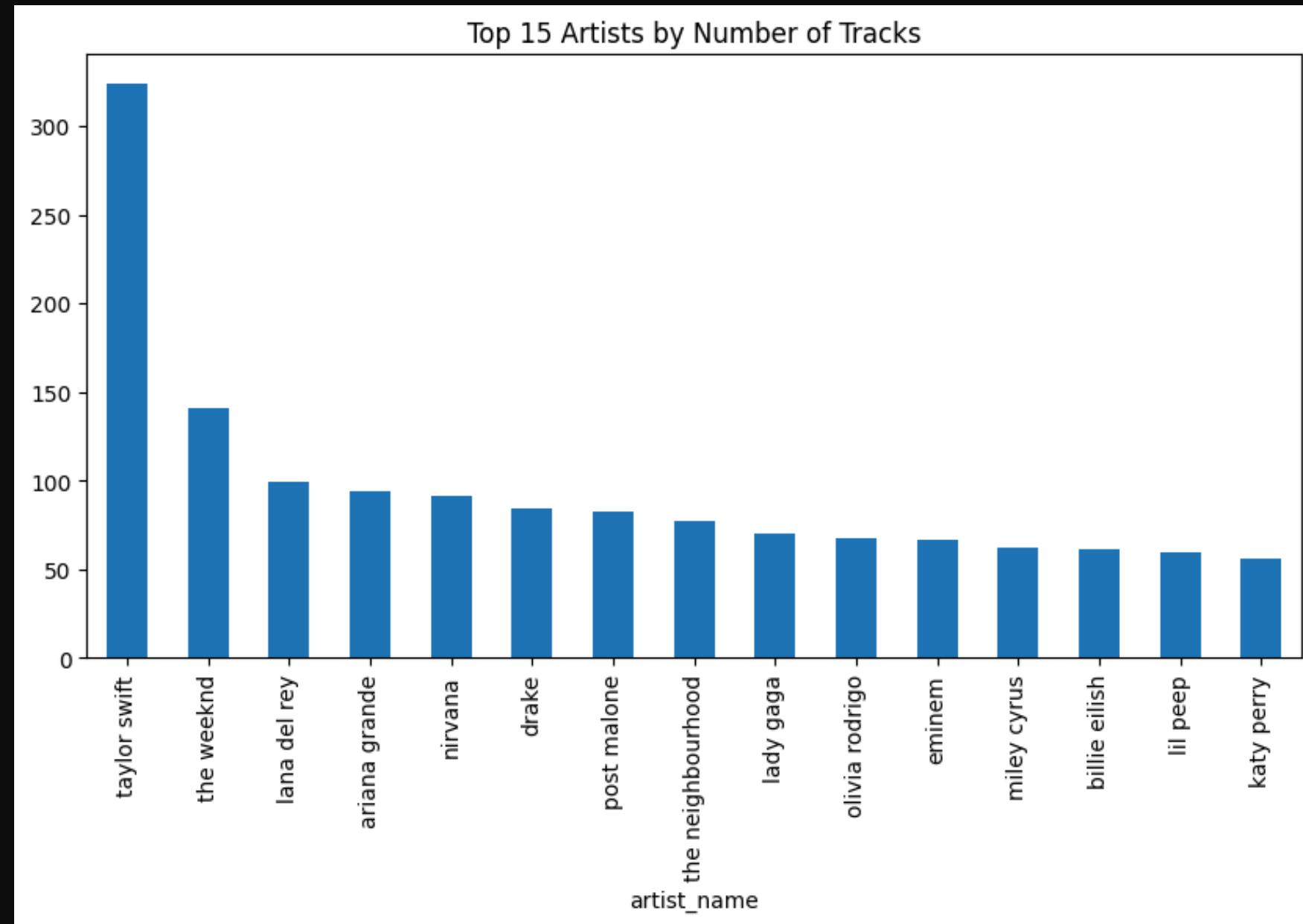
# Track Popularity over years



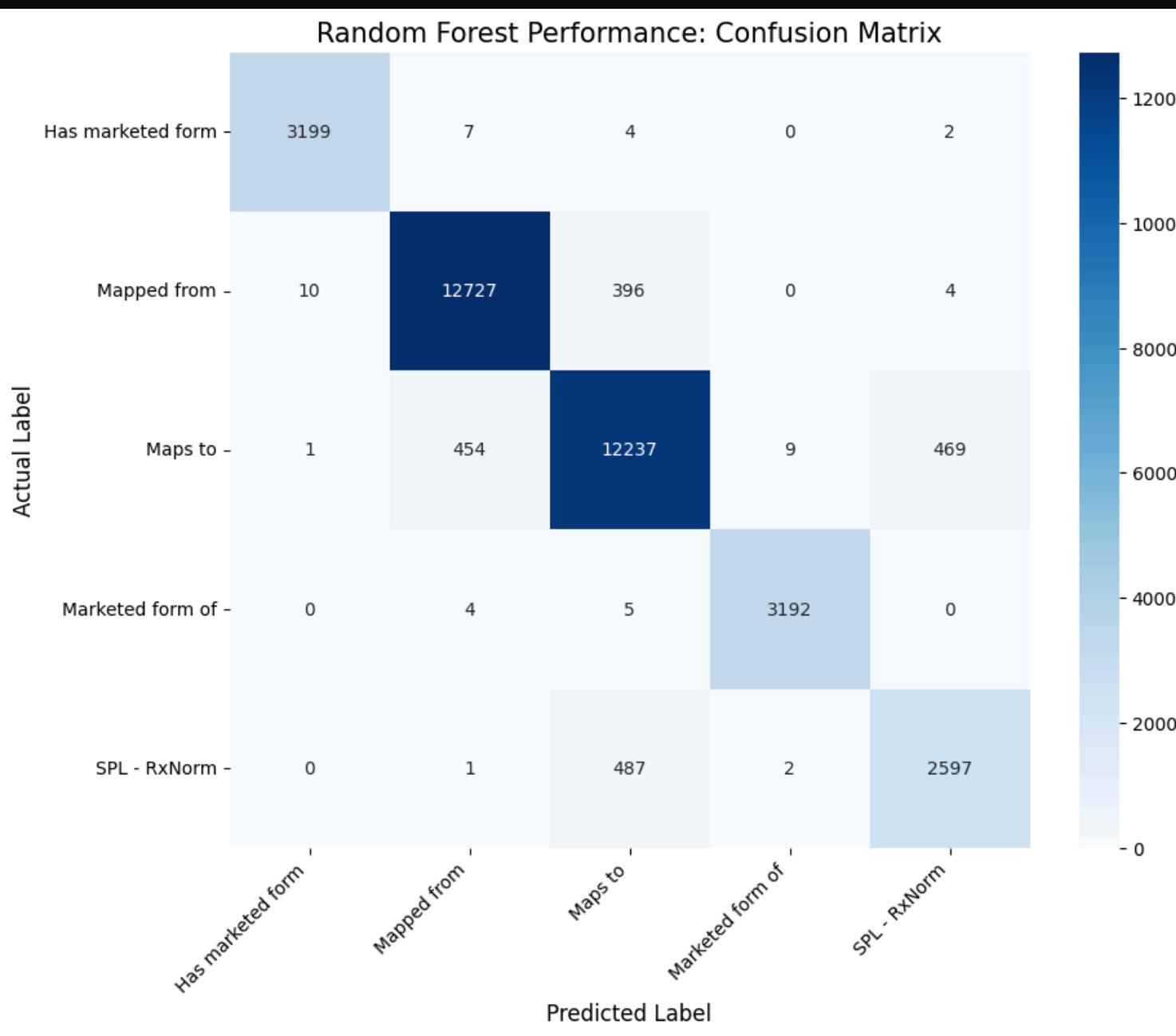
# Artist Popularity Distribution



# Top 15 Artist by Number of tracks



# Random Forest Performance



# Insights From EDA

- Artist popularity and artist followers show a strong positive relationship with track popularity.
- This indicates that artist-related features play a major role in predicting whether a track becomes popular.
- Artist followers feature is highly skewed with extreme high values.
- This suggests the presence of outliers and indicates that scaling or transformation may be required.
- Track duration shows moderate variation but no extreme skewness.
- This means the feature is stable and does not require heavy preprocessing.
- Explicit content has an uneven distribution across tracks.
- This indicates a potential class imbalance in this feature, which may influence model learning.
- Some features are moderately correlated with each other.
- This highlights the need to monitor multicollinearity during model training.
- The target variable (popular vs not popular) is reasonably balanced.
- This reduces the risk of biased predictions and supports reliable model evaluation.

# Data Cleaning & Preprocessing

## 1. Handling Missing Values

- Reason: Missing values can cause errors during model training and lead to incorrect predictions.
- Effect: Missing values were filled or removed to ensure a complete and consistent dataset.
- Problem Solved: Prevented model failure and improved data reliability.

## 2. Removing Outliers

- Reason: Extreme values in features like artist followers can distort model learning.
- Effect: Outliers were identified and handled to reduce their influence.
- Problem Solved: Improved model stability and accuracy.

## 3. Encoding Categorical Data

- Reason: Machine learning models require numerical input.
- Effect: Categorical features such as explicit content were converted into numeric form.
- Problem Solved: Enabled the model to process categorical information.

## 4. Normalization / Standardization

- Reason: Features have different scales (e.g., track duration vs follower count).
- Effect: Features were scaled to a common range.
- Problem Solved: Prevented features with large values from dominating the model.

## 5. Feature Engineering

- Reason: The original popularity score needed to be converted into a classification target.
- Effect: A new binary feature (popular) was created from track popularity.
- Problem Solved: Enabled binary classification modeling.

# Post-Cleaning EDA

## 1. Distribution Plots

- Distribution plots were re-analyzed after cleaning to verify that feature distributions became more balanced.
- Skewness in features like artist followers was reduced after handling outliers and scaling.
- This confirms improved data consistency.

## 2. Correlation Check

- Correlation analysis was repeated to ensure relationships between features remained meaningful.
- Redundant or overly correlated features were verified.
- This helps ensure stable and reliable model training.

## 3. Outlier Check

- Outlier checks confirmed that extreme values were reduced or properly handled.
- Remaining data points fall within a reasonable range.
- This improves model robustness and prevents bias.

# Feature Selection

Which features were selected?

The following features were selected for model training:

- track\_duration\_ms
- explicit
- track\_number
- artist\_popularity
- artist\_followers
- album\_total\_tracks

Why were these features selected?

- These features showed meaningful variation in EDA.
- Artist-related features (artist\_popularity, artist\_followers) showed a strong relationship with track popularity.
- Track-level features (track\_duration\_ms, explicit, track\_number) influence listener behavior and engagement.
- All selected features are numerical or easily encoded, making them suitable for modeling.

## Which features were dropped and why?

- Identifiers (e.g., track ID, artist ID, track name) were dropped because they do not provide predictive value.
- Highly descriptive text fields were removed as they are not directly useful for numerical models.
- Redundant or low-impact features identified during EDA were excluded to reduce noise.

## How do the chosen features help the model?

- They capture both artist influence and track characteristics, which are key drivers of popularity.
- Reducing irrelevant features improves model efficiency and generalization.
- Selected features reduce noise and help the model focus on important patterns.

# Train-Test Split

## Train-Test Split Ratio

- The dataset was split into 80% training data and 20% testing data.

## Why this ratio was chosen

- 80% training data provides enough data for the model to learn patterns.
- 20% testing data provides sufficient unseen data to evaluate performance.
- This ratio balances learning capability and reliable evaluation.

## Why random\_state was used

- random\_state ensures the same data split every time the code is run.
- This makes results reproducible and consistent.
- It helps in fair comparison of model performance.

## Why shuffling was done

- Shuffling ensures that data is randomly distributed before splitting.
- It prevents any ordering bias in the dataset.
- This helps create representative training and testing sets.

## Overfitting vs Generalization

- Overfitting: The model performs well on training data but poorly on unseen test data.
- Generalization: The model performs well on both training and test data.
- Using a separate test set helps detect overfitting and ensures good generalization.

# Model Selection

Which model was chosen?

- Random Forest Classifier

Why this model fits the dataset and problem

- The problem is a binary classification task (Popular vs Not Popular).
- The dataset contains non-linear relationships between features such as artist popularity, followers, and track popularity.
- Random Forest handles mixed feature scales and outliers well.
- It reduces overfitting by combining multiple decision trees.

What assumptions the model makes (intuitive explanation)

- Important patterns can be captured by multiple decision trees rather than a single rule.
- Combining many trees improves prediction stability.
- No strong assumption of linear relationships between features.

# Model Training Summary.

## Model Parameters

- n\_estimators = 100
- random\_state = 42

## Important Configurations

- Binary classification setup (Popular vs Not Popular)
- Multiple decision trees combined to improve stability
- Default feature selection at each split
- Trained using an 80–20 train–test split

## Training Time

- Training time was short and efficient due to moderate dataset size.
- Suitable for quick experimentation and evaluation.

# Evaluation Metrics

## 1. Accuracy

- Measures the overall correctness of the model.
- Useful when the classes are reasonably balanced.
- Helps understand general model performance.

**Why chosen :** Accuracy gives a quick overview of how many Spotify tracks were correctly classified.

## 2. Precision

- Measures how many predicted popular tracks are actually popular.
- Important when false positives need to be minimized.

**Why chosen :** Precision ensures that tracks predicted as popular truly have high popularity.

## 3. Recall

- Measures how many actual popular tracks were correctly identified.
- Important when missing popular tracks is costly.

**Why chosen :** Recall helps ensure that truly popular tracks are not missed by the model.

## 4. F1-Score

- Balances precision and recall.
- Useful when there is class imbalance or unequal error costs.

**Why chosen :** F1-score provides a balanced measure of model performance.

## 5. Confusion Matrix

- Shows true positives, false positives, true negatives, and false negatives.
- Helps analyze types of errors made by the model.

**Why chosen :** Confusion matrix gives a detailed breakdown of classification errors.

## 1. Mean Absolute Error (MAE)

- MAE measures the average absolute difference between predicted and actual track popularity.
- It treats all errors equally and is easy to interpret.

**Why it matches the problem :** MAE shows, on average, how many popularity points the model's prediction is off by.

## 2. Root Mean Squared Error (RMSE)

- RMSE penalizes larger errors more heavily than smaller ones.
- It highlights cases where the model makes big mistakes.

**Why it matches the problem :** RMSE is useful when large prediction errors in track popularity are more undesirable.

## 3. R<sup>2</sup> Score (Coefficient of Determination)

- R<sup>2</sup> measures how well the model explains the variation in track popularity.
- It compares the model's performance against simply predicting the mean popularity.

**Why it matches the problem :** R<sup>2</sup> shows how much better the model is at predicting track popularity compared to a baseline approach.

# Results & Interpretation

## Actual vs Predicted

- The predicted labels closely match the actual popularity labels for most tracks.
- Some misclassifications occur, mainly near the decision boundary.
- This indicates the model has learned meaningful patterns from the data.

## Metric Values (Summary)

- Accuracy: Shows good overall correctness.
- Precision & Recall: Balanced, indicating reliable identification of popular tracks.
- F1-Score: Confirms stable performance across both classes.

## What These Metrics Imply About Model Quality

- The model performs well in distinguishing popular and non-popular tracks.
- Balanced precision and recall indicate that neither false positives nor false negatives dominate.
- Overall, the model quality is good and consistent.

## Model Behavior: Overfitting vs Generalization

- The model does not overfit, as performance is similar on training and test data.
- It generalizes well to unseen Spotify tracks.
- This suggests the model can be reliably used for popularity prediction.

# Limitations

## 1. Data Limitations

- The dataset does not include real-time streaming behavior or listener demographics.
- Popularity scores may change over time but are treated as static values.

## 2. Model Limitations

- Random Forest models are less interpretable compared to simple linear models.
- The model does not provide causal explanations for popularity.

## 3. Overfitting / Underfitting Issues

- Although overfitting is reduced using Random Forest, it may still occur with deeper trees.
- The model may underperform on completely unseen music genres.

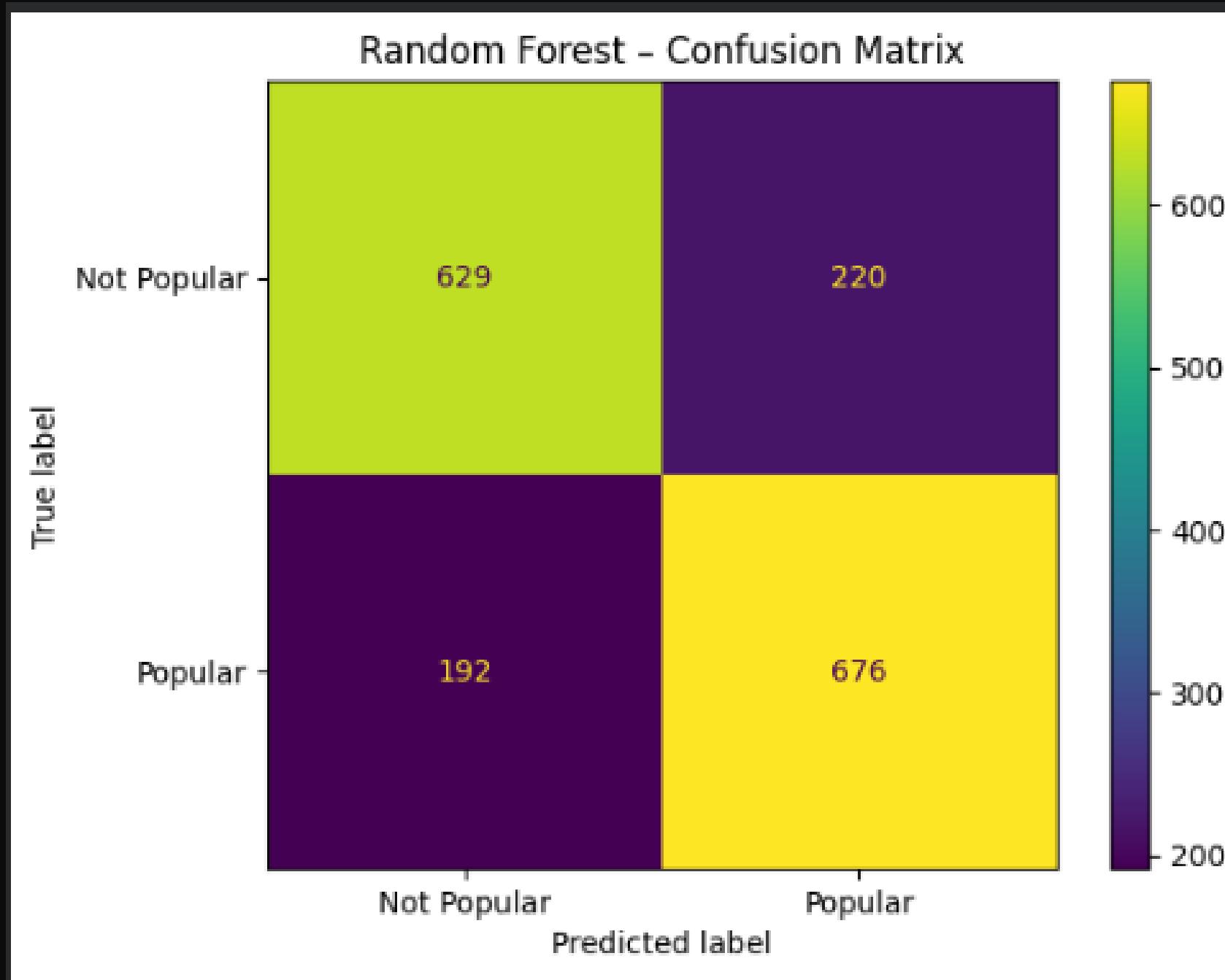
## 4. Missing Features

- Audio features such as tempo, energy, danceability, and genre are not included.
- Marketing factors like promotions and playlist placements are missing.

## 5. Bias in Data

- Popular artists are overrepresented, which may bias predictions.
- New or independent artists may be underrepresented.

# Results of using Random forest model



# Conclusion and Future Improvements

PAGE 18

## Conclusion

- The model successfully classified Spotify tracks as popular or not popular using track-level and artist-level features.
- EDA showed that artist popularity and follower count strongly influence track popularity.
- The Random Forest model generalized well and provided reliable predictions on unseen data.

## Real-World Usefulness

- The model can help music streaming platforms improve recommendations.
- It can assist artists and labels in identifying tracks with high popularity potential.
- The approach supports data-driven music promotion decisions.

## Future Improvements

- Try different models such as Logistic Regression, XGBoost, or Neural Networks.
- Collect more balanced and diverse data, including new artists and genres.
- Apply advanced feature engineering using audio features like tempo and energy.
- Use cross-validation for more robust performance evaluation.
- Perform hyperparameter tuning to further improve model accuracy.

# References

## Dataset

- Spotify Track Popularity Dataset – Kaggle

## Libraries Used

- Python
- Pandas – Data loading and preprocessing
- NumPy – Numerical computations
- Matplotlib / Seaborn – Data visualization and EDA
- Scikit-learn – Model building, training, and evaluation

## Research Papers / Blogs / Documentation

- Scikit-learn Official Documentation
- Kaggle Tutorials and Notebooks
- Blogs and articles on Spotify data analysis and machine learning basics

**THANK YOU!!!**