Regression Analysis of Spotify Songs: From Audio Features to Popularity

September 2025

1 Problem Formulation

Predicting Spotify Track Popularity

The task of this project is to predict the popularity of songs on Spotify using machine learning. Popularity is given as an integer between 0 and 100 and serves as the target variable in our regression problem. Each data point in the dataset corresponds to a single track and includes descriptive attributes such as:

- Audio features: danceability, energy, loudness, speechiness, acousticness, instrumentalness, valence, tempo, duration (in ms), etc.
- Metadata: explicit flag, key, mode, time signature, and genre.

Each track i can be represented by a feature vector

$$\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{in}],$$

and the goal is to learn a mapping

$$f(\mathbf{x}_i) \approx y_i,$$

where y_i is the track's true popularity score.

The dataset originates from the Kaggle dataset *"Spotify Tracks Dataset (125 Genres)"* by Maharshi Pandya [**?**], which contains approximately 114,000 songs across 125 genres. It provides detailed audio and metadata features for each track, making it suitable for regression, recommendation, and classification tasks. Further dataset details and licensing information are provided in Appendix 4.

In summary, the problem is a supervised regression task: given the audio and metadata features of a track, predict its popularity score on Spotify.

2 Methods

Dataset Description and Preprocessing

The dataset used in this project consists of approximately 114,000 Spotify tracks spanning 125 distinct musical genres. Each row in the dataset corresponds to a single track, and each column describes a specific property or metadata field associated with that track.

The main features include both musical characteristics and metadata. The columns `track_id`, `track_name`, `artists`, and `album_name` uniquely identify each recording but do not provide predictive information for modeling, and were therefore removed during preprocessing. The target variable is `popularity`, a numerical value between 0 and 100 assigned by Spotify's internal algorithm based on the number of plays and their recency. Songs that are currently trending or have sustained play counts tend to have higher popularity scores.

The remaining columns describe the acoustic and musical structure of each track. The attributes `danceability`, `energy`, `speechiness`, `acousticness`, `instrumentalness`, `liveness`, and `valence` are perceptual audio features computed by Spotify, each scaled between 0.0 and 1.0. They capture rhythmic regularity, intensity, presence of speech, degree of acoustic instrumentation, and overall emotional tone, respectively. Additional musical properties include `key`, `mode`, `loudness`, `tempo`, and `time_signature`, which represent tonal and rhythmic elements such as musical key, harmonic mode, decibel level, beats per minute, and metrical structure. The feature `duration_ms` denotes the track length in milliseconds, while `explicit` indicates whether the lyrics contain explicit content. The categorical variable `track_genre` specifies the genre of each song (e.g., pop, rock, jazz).

Feature Selection

Before training, several preprocessing steps were applied to ensure the data were suitable for regression analysis. Non-predictive text identifiers were dropped as noted above. Missing numerical values were replaced with zeros, ensuring the model could handle incomplete data. Standard scaling was used only for the MLP regressor. We select features believed to influence popularity, focusing on measurable audio features (`danceability, energy, loudness, speechiness, acousticness, instrumentalness, valence, tempo, duration_ms`) and metadata (`explicit, mode, time_signature`).

2.1 Random Forest Regression

For this project, we use a Random Forest Regression model and compare it to Neural Network Regression. Random Forest is an ensemble method that combines multiple decision trees to capture complex, nonlinear relationships between features and the target variable. Unlike linear regression, it does not assume a linear relationship between audio features and track popularity. It also

reduces overfitting through averaging predictions across many trees and provides measures of feature importance, which can help identify which characteristics most influence popularity.

Loss Function and Optimization

The loss function used is the Mean Squared Error (MSE):

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

MSE penalizes larger prediction errors more strongly than smaller ones, making it suitable for continuous targets like popularity. It encourages the model to produce predictions that are consistently close to the true values and is widely used in regression tasks for its interpretability and sensitivity to large deviations.

Model Validation

The dataset was split into training (70%), validation (15%), and test (15%) sets. This ratio provides a balanced tradeoff between model learning capacity and reliable performance evaluation. With approximately 114,000 samples, 70% ensures sufficient data for the Random Forest to learn complex patterns, while 15% each for validation and testing provides enough data to tune hyperparameters and assess generalization. Stratified sampling was used to preserve the distribution of the popularity scores across all subsets.

2.2 Neural Network Regression (MLP Regressor)

In this project, the MLP model includes three hidden layers with 128, 64 and 32 neurons, both using the ReLU activation function, followed by one output neuron for predicting the popularity score.

The MLP was chosen because it can capture complicated, nonlinear relationships between features. For example, how energy, danceability, and loudness together affect a song's popularity is unlikely to be a simple or linear relationship. Neural networks are well-suited for learning such patterns directly from data.

However, this approach comes with some trade-offs, for example, neural networks take longer to train and require more computing power compared to methods like Random Forests. They also need the input features to be scaled properly, which adds preprocessing steps.

The loss function used for the MLP is Mean Squared Error (MSE) for the same reasons as described above for the Random Forest Regression.

The model validation is also done in the same manner as it is done in the Random Forest Regression.

3 Results

Figures 1 and 2 show the relationship between actual and predicted Spotify track popularity for the Random Forest and MLP regression models, respectively. In both cases, the red dashed line represents perfect predictions ($y = x$). Overall, both models capture the general trend that higher actual popularity corresponds to higher predicted values. The Random Forest predictions align more closely with the ideal line in the mid-range of popularity scores, whereas the MLP exhibits a wider spread and larger deviations, leading to a higher Mean Squared Error (MSE ≈ 409 for MLP vs. 222 for Random Forest).

Both models show substantial scatter around the ideal line, reflecting the noisy and skewed nature of the popularity metric, which is influenced by external factors such as marketing, playlist placement, and release timing. Random Forests, being non-parametric and robust to feature scaling, provide reliable predictions without extensive hyperparameter tuning, but they tend to regress toward the mean, underestimating extremely popular tracks. In contrast, the MLP can theoretically model complex nonlinear interactions among features, but it is sensitive to feature scaling and hyperparameters and struggles when the target distribution is skewed or noisy.

These results suggest that while neural networks offer greater flexibility, ensemble methods like Random Forest provide better practical performance on this dataset, offering robust, interpretable predictions with moderate computational cost.

4 Appendix

Random Forest Regression Code

```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt

data = pd.read_csv("dataset.csv")

X = data.drop(columns=['popularity', 'track_id', 'track_name', '
    artists', 'album_name'], errors='ignore')
X = X.select_dtypes(include=['number']).fillna(0)
y = data['popularity'].fillna(0)

X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size
    =0.3, random_state=42)

X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp,
    test_size=0.5, random_state=42)

rf = RandomForestRegressor(n_estimators=100, max_depth=None,
    random_state=42, n_jobs=-1)
rf.fit(X_train, y_train)
```

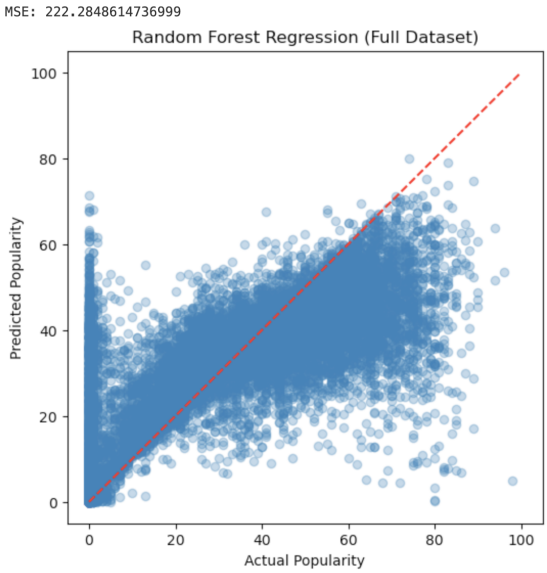

Figure 1: Scatter plot of actual vs. predicted track popularity using Random Forest Regression. The red dashed line represents the ideal prediction line where predicted popularity equals actual popularity.

```
19
20 y_pred = rf.predict(X_test)
21
22 print("MSE:", mean_squared_error(y_test, y_pred))
23
24
25 plt.figure(figsize=(6,6))
26 plt.scatter(y_test, y_pred, alpha=0.3)
27 plt.plot([0,100],[0,100], color='red', linestyle='--')
28 plt.xlabel("Actual Popularity")
29 plt.ylabel("Predicted Popularity")
30 plt.title("Random Forest Regression (Full Dataset)")
31 plt.show()
```

MLP Regressor Code

```
1 from sklearn.preprocessing import StandardScaler
2 from sklearn.model_selection import train_test_split
3 from sklearn.neural_network import MLPRegressor
4 from sklearn.metrics import mean_squared_error
5 import matplotlib.pyplot as plt
6
7 data = pd.read_csv("dataset.csv")
8
9 X = data.drop(columns=['popularity', 'track_id', 'track_name', '
      artists', 'album_name'], errors='ignore')
```

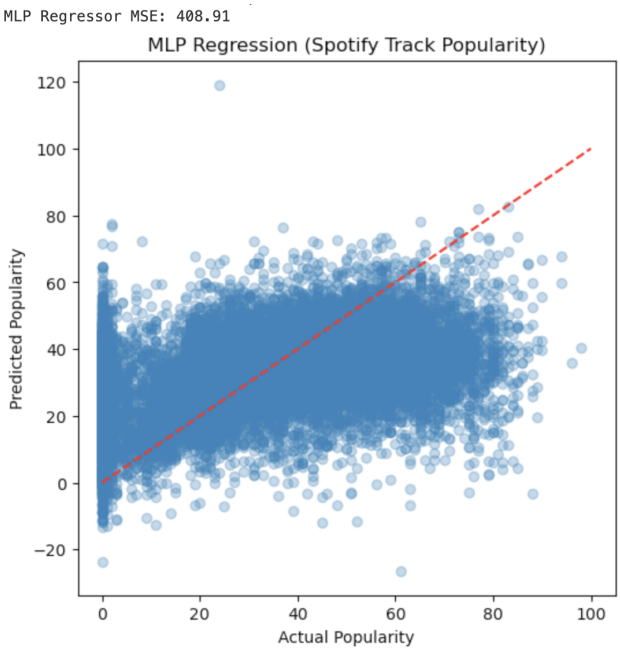

Figure 2: Scatter plot of actual vs. predicted track popularity using MLP Regression. The red dashed line represents the ideal prediction line where predicted popularity equals actual popularity.

```
X = X.select_dtypes(include=['number']).fillna(0)  # Use numeric columns only
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
y = data['popularity'].fillna(0)


X_train, X_temp, y_train, y_temp = train_test_split(X_scaled, y, test_size=0.3, random_state=42, stratify=None)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42, stratify=None)


mlp = MLPRegressor(
    hidden_layer_sizes=(128, 64, 32),  # two hidden layers: 64 and 32 neurons
    activation='relu',
    solver='adam',
    alpha=0.0001,
    batch_size='auto',
    learning_rate='constant',
    learning_rate_init=0.005,
    max_iter=500,
    shuffle=True,
```

```
    random_state=42,
    tol=0.0001,
    verbose=True,
    early_stopping=True,
    validation_fraction=0.1
)

mlp.fit(X_train, y_train)

y_pred = mlp.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
print(f"MLP Regressor MSE: {mse:.2f}")

plt.figure(figsize=(6,6))
plt.scatter(y_test, y_pred, alpha=0.3)
plt.plot([0,100],[0,100], color='red', linestyle='--')
plt.xlabel("Actual Popularity")
plt.ylabel("Predicted Popularity")
plt.title("MLP Regression (Spotify Track Popularity)")
plt.show()
```

Dataset Reference

The dataset used in this study is publicly available on Kaggle:

> Maharshi Pandya, *Spotify Tracks Dataset (125 Genres)*, 2020.
> Available at: https://www.kaggle.com/datasets/maharshipandya/spotify-tracks-dataset
> License: Database — Open Database; Contents — © Original Authors.