**1. Name of the Data**

Spotify Top 1000 Tracks Dataset

**2. Source of the Data**

This dataset originates from Kaggle Datasets.

**3. Link to the Original Data**

<https://www.kaggle.com/yamaerenay/spotify-dataset-19212020-160k-tracks>

**4. Data Explanation**

The dataset contains audio features and metadata for the top 1,000 tracks on Spotify including:

* Track name, artist, album
* Release date
* Popularity score (0-100)
* Duration in minutes

For regression tasks, we predict the continuous popularity score. For classification, we categorize popularity into: Low (0-60), Medium (60-75), High (75-90), and Very High (90-100).

**5. Type of Problem**

* Regression: Predict continuous popularity score
* Classification: Categorize popularity into 4 classes

**6. Number of Attributes**

8 features after preprocessing:  
['track\_name\_encoded', 'artist\_encoded', 'album\_encoded', 'release\_year', 'duration\_min']

**7. Number of Samples**

1,000 tracks (800 training, 200 testing)

**8. Properties of the Data (Statistics)**

* Minimum popularity: 0
* Maximum popularity: 100
* Mean popularity: 58.3
* Standard Deviation: 18.7
* Most common release year: 2018 (127 tracks)

**9. Missing Data**

No missing values after cleaning.

**10. Data Visualization**

Figure 1:Distribution of popularity scores

**11. Normalization or Standardization**

Applied StandardScaler to all features because:

* SVM is sensitive to feature scales
* Neural networks require normalized inputs
* Ensures fair feature comparison

**12. Preprocessing Applied**

1. Parsed 'release\_date' as datetime
2. Encoded categorical variables (artist, track)
3. Generated 'release\_year' feature
4. Removed irrelevant columns (URLs, IDs)
5. Applied feature scaling
6. Split data into train/test sets

**13. Train-Test Split**

* 80% training (800 samples)
* 20% testing (200 samples)
* Stratified sampling for classification

**14. Machine Learning Models and Performance**

Regression Models

|  |  |  |
| --- | --- | --- |
| Model | MSE | R² |
| Random Forest | 585.17 | 0.040 |
| Linear Regression | 558.84 | 0.083 |
| Neural Network | 587.55 | 0.036 |

Classification Models

|  |  |  |
| --- | --- | --- |
| Model | Accuracy | F1-Score |
| Random Forest | 52.2% | 0.52 |
| SVM | 46.7% | 0.43 |
| Neural Network | 43.4% | 0.44 |

**Best Performing**: Random Forest  
**Worst Performing**: Naive Bayes

**15. Accuracy and Figures**

Figure 2: Random Forest regression predictions

**16. Advanced Visualization**

Figure 3: Key predictors of song popularity

**17. Explanation**

I selected the Spotify dataset to understand what makes songs popular in the streaming era. The data required significant preprocessing including date parsing and categorical encoding. Feature engineering revealed release year as the strongest predictor.

Random Forest outperformed other models due to its ability to handle non-linear relationships. The classification task proved challenging with only 52.2% accuracy, suggesting popularity depends on factors beyond our features.

Standardization was crucial for SVM and neural networks. Visualizations highlighted right-skewed popularity distribution, meaning most tracks cluster in mid-range scores.

This project demonstrated that while machine learning can identify trends, predicting "viral" hits remains complex. The insights could help artists optimize release timing but should complement creative decisions.

Future work could incorporate audio features like tempo or valence. The exercise reinforced that data quality often matters more than model selection.

**18. Project Structure Note**

<https://github.com/ReemAlgethami/Predicting-Song-Popularity-on-Spotify-Using-Machine-Learning>

Predicting-Song-Popularity-on-Spotify-Using-Machine-Learning /

├── ProjectML.py # Main analysis script

├── Data/

│ ├── train\_data.csv # Training set (features + target)

│ ├── test\_data.csv # Testing set (features + target)

│ └── Results/

├── 1\_Core\_Analysis/

│ ├── popularity\_dist.png # Popularity score distribution

│ ├── class\_distribution.png # Popularity class distribution

│ └── release\_years.png # Track distribution by release year

│

├── 2\_Feature\_Analysis/

│ ├── feature\_importance.png # Feature importance from Random Forest

│ └── correlation\_heatmap.png # Feature correlation heatmap

│

├── 3\_Regression\_Results/

│ ├── CSV\_Files/

│ │ ├── Decision Tree\_regression.csv

│ │ ├── KNN\_regression.csv

│ │ ├── Linear Regression\_regression.csv

│ │ ├── Neural Network\_regression.csv

│ │ ├── Random Forest\_regression.csv

│ │ └── SVM\_regression.csv

│ │

│ └── Visualizations/

│ ├── Decision Tree\_regression.png

│ ├── KNN\_regression.png

│ ├── Linear Regression\_regression.png

│ ├── Neural Network\_regression.png

│ ├── Random Forest\_regression.png

│ └── SVM\_regression.png

│

├── 4\_Classification\_Results/

│ ├── CSV\_Files/

│ │ ├── Decision Tree\_classification.csv

│ │ ├── KNN\_classification.csv

│ │ ├── Naive Bayes\_classification.csv

│ │ ├── Neural Network\_classification.csv

│ │ ├── Random Forest\_classification.csv

│ │ └── SVM\_classification.csv

│ │

│ └── Confusion\_Matrices/

│ ├── Decision Tree\_confusion\_matrix.png

│ ├── KNN\_confusion\_matrix.png

│ ├── Naive Bayes\_confusion\_matrix.png

│ ├── Random Forest\_confusion\_matrix.png

│ └── SVM\_confusion\_matrix.png

│

└── 5\_Reports/

└── summary\_report.txt # Analysis summary

**19.Results**

19.1 Regression Performance

Figure 4: Regression Performance

Figure 5: Comparative Analysis of Regression Models for Spotify Popularity Prediction

**Key Observations**:

* **Linear Regression** surprisingly achieved the lowest MSE (558.84), suggesting simpler models may perform adequately for this task.
* All models struggled with extreme popularity values (scores <20 or >80), as seen in the prediction scatterplots.

**19.2 Classification Performance**