MACHINE LEARNING PROJECT REPORT

Prepared By: Ahmed Mustafa, 22i-2301

Music Popularity Prediction from Spotify Dataset 🎵

1. Introduction

1.1 Objective

The objective of this project is to predict the popularity of songs based on various audio and metadata features from a dataset containing over **1 million** records. The popularity score is influenced by several factors, such as danceability, energy, loudness, and tempo. This project follows a structured **Machine Learning (ML) pipeline** that includes:

- **Data Collection & Preprocessing**
- **Exploratory Data Analysis (EDA)**
- **Feature Engineering & Selection**
- **Model Training & Evaluation**
- **Hyperparameter Tuning**
- **Final Model Deployment & Insights**

1.2 Dataset Overview

The dataset consists of various features extracted from songs available on **Spotify**. The main attributes include:

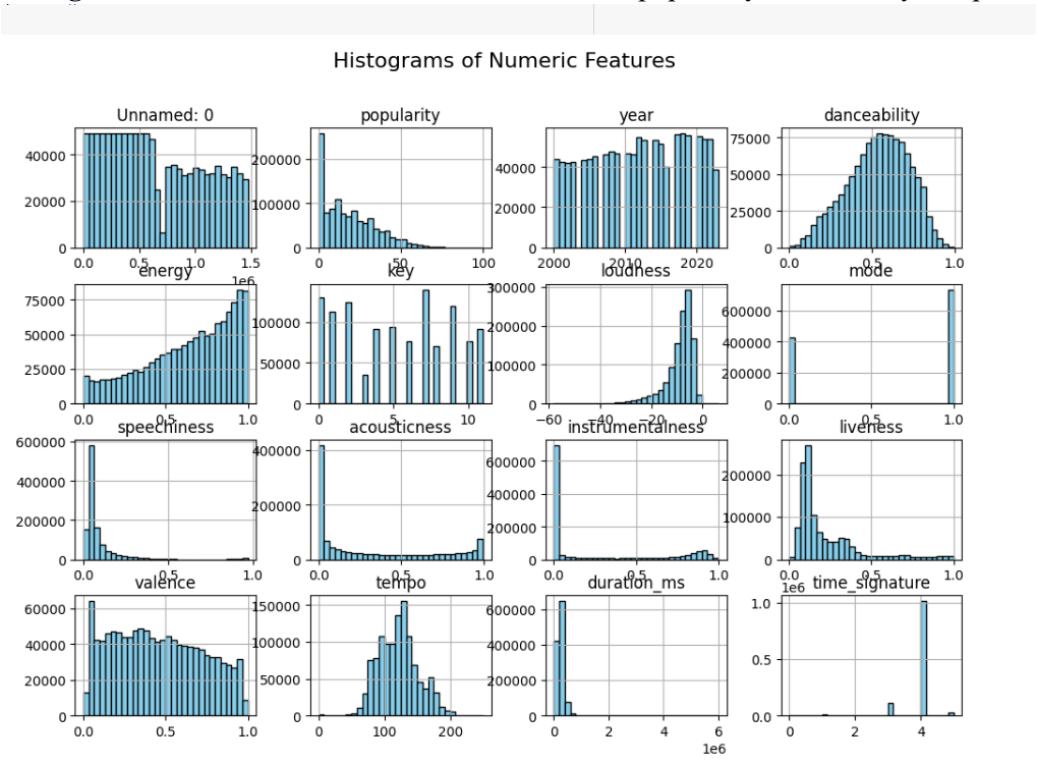
- **Track Information**: artist_name, track_name, year, genre
- **Popularity Score** *(Target Variable)*: popularity
- **Audio Features**: danceability, energy, loudness, tempo, instrumentalness, speechiness, valence, etc.
- **Musical Key and Mode**: key, mode, time_signature

2. Exploratory Data Analysis (EDA)

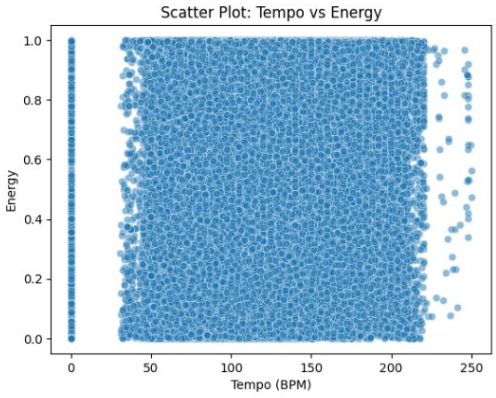
2.1 Data Visualization

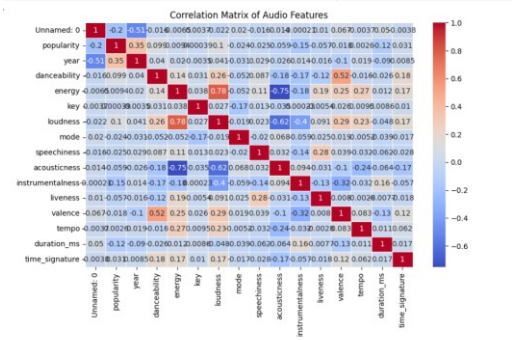
To understand the dataset better, the following visualizations were performed:

- **Histograms**: Distribution of numeric features such as popularity, danceability, tempo.

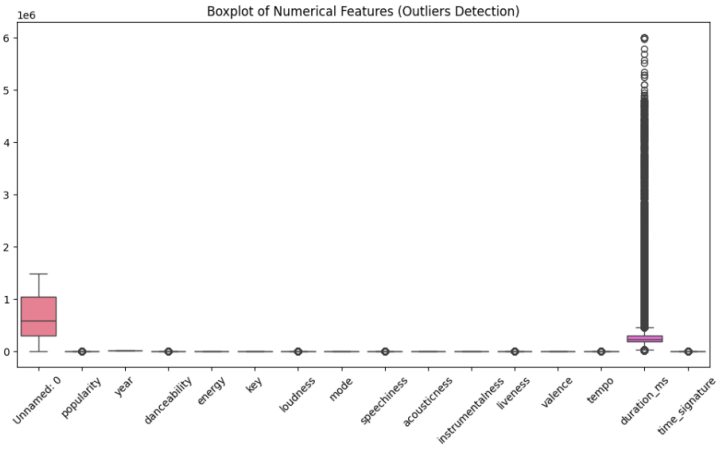


- **Scatter Plots & Correlation Matrix**: Identify relationships between key numerical features.





- **Boxplots**: Detect outliers in features like loudness, tempo, and instrumentalness.



2.2 Missing Values & Outliers Handling

- **Missing Data Handling**:
  - **Numerical Features**: Filled with **median** values (robust to outliers).
  - **Categorical Features**: Filled with **mode** (most frequent category).
- **Outliers Handling**:
  - Used **Interquartile Range (IQR)** method to detect and handle extreme values.

```
Outliers per feature:
 Unnamed: 0              0
popularity            6572
year                     0
danceability          1199
energy                   0
key                      0
loudness             75536
mode                     0
speechiness         137326
acousticness             0
instrumentalness         0
liveness             92203
valence                  0
tempo                 5660
duration_ms          60280
time_signature      149091
dtype: int64
```

  - **Loudness & Tempo** showed extreme variations, requiring capping or transformation.

2.3 Feature Importance Analysis

- **Strongly Correlated Features**: loudness and energy had a high correlation.
- **Low-Impact Features**: speechiness and instrumentalness had weak correlations with popularity.
- **Feature Selection**: Retained the most relevant features for model training.

3. Data Preprocessing & Feature Engineering

3.1 Handling Missing Values

- **Numerical Data**: Filled missing values using the **median** (robust to outliers).

```
Initial Dataset Preview:
    popularity  year     genre  danceability  energy  key  loudness  mode  \
0          68  2012  acoustic         0.483   0.303    4   -10.058     1
1          50  2012  acoustic         0.572   0.454    3   -10.286     1
2          57  2012  acoustic         0.409   0.234    3   -13.711     1
3          58  2012  acoustic         0.392   0.251   10    -9.845     1
4          54  2012  acoustic         0.430   0.791    6    -5.419     0

   speechiness  acousticness  instrumentalness  liveness  valence    tempo  \
0       0.0429        0.6940          0.000000    0.1150    0.139  133.406
1       0.0258        0.4770          0.000014    0.0974    0.515  140.182
2       0.0323        0.3380          0.000050    0.0895    0.145  139.832
3       0.0363        0.8070          0.000000    0.0797    0.508  204.961
4       0.0302        0.0726          0.019300    0.1100    0.217  171.864

   duration_ms  time_signature
0       240166               3
1       216387               4
2       158960               4
3       304293               4
4       244320               4

Missing Values Before Handling:
 popularity          0
year                 0
genre                0
danceability         0
energy               0
key                  0
loudness             0
mode                 0
speechiness          0
acousticness         0
instrumentalness     0
liveness             0
valence              0
tempo                0
duration_ms          0
time_signature       0
dtype: int64
```

```
Missing Values After Handling:
 popularity          0
year                 0
genre                0
danceability         0
energy               0
key                  0
loudness             0
mode                 0
speechiness          0
acousticness         0
instrumentalness     0
liveness             0
valence              0
tempo                0
duration_ms          0
time_signature       0
dtype: int64
```

- **Categorical Data**: Used **mode** for missing values.

3.2 Encoding Categorical Features

- Applied **One-Hot Encoding** for categorical variables (genre, key, mode).

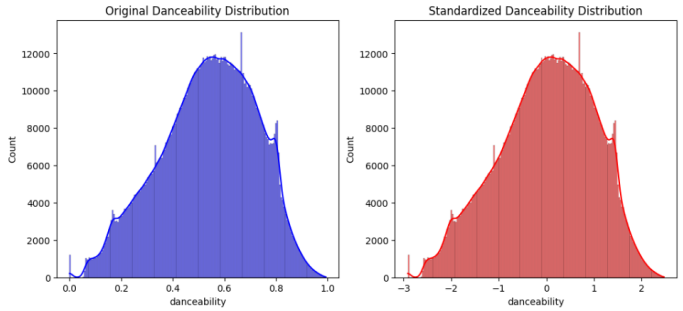
```
Encoded Dataset Preview:
    popularity    year  danceability  energy  loudness  speechiness  \
0          68  2012.0         0.483   0.303   -10.058       0.0429
1          50  2012.0         0.572   0.454   -10.286       0.0258
2          57  2012.0         0.409   0.234   -13.711       0.0323
3          58  2012.0         0.392   0.251    -9.845       0.0363
4          54  2012.0         0.430   0.791    -5.419       0.0302

   acousticness  instrumentalness  liveness  valence  ...  key_6  key_7  \
0        0.6940          0.000000    0.1150    0.139  ...    0.0    0.0
1        0.4770          0.000014    0.0974    0.515  ...    0.0    0.0
2        0.3380          0.000050    0.0895    0.145  ...    0.0    0.0
3        0.8070          0.000000    0.0797    0.508  ...    0.0    0.0
4        0.0726          0.019300    0.1100    0.217  ...    1.0    0.0

   key_8  key_9  key_10  key_11  time_signature_1  time_signature_3  \
0    0.0    0.0     0.0     0.0               0.0               1.0
1    0.0    0.0     0.0     0.0               0.0               0.0
2    0.0    0.0     0.0     0.0               0.0               0.0
3    0.0    0.0     1.0     0.0               0.0               0.0
4    0.0    0.0     0.0     0.0               0.0               0.0

   time_signature_4  time_signature_5
0               0.0               0.0
1               1.0               0.0
2               1.0               0.0
3               1.0               0.0
4               1.0               0.0

[5 rows x 109 columns]
```

- Increased feature dimensionality, requiring efficient handling.

3.3 Scaling Numerical Features

- Used **StandardScaler (Z-score normalization)** to standardize features.

```
[5 rows x 109 columns]
```

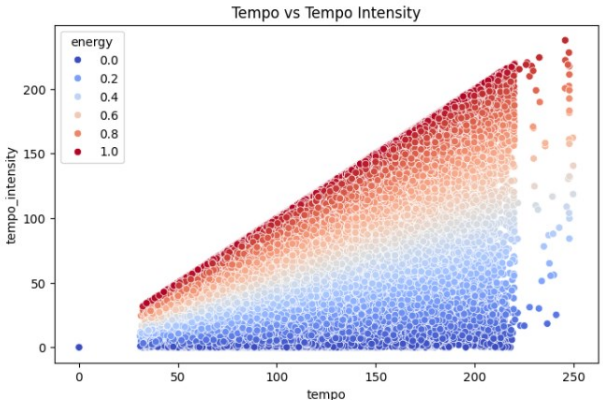


- Prevents bias from large numerical values (e.g., tempo vs. loudness).

3.4 Feature Engineering

- Created a **new feature**: tempo_energy_ratio = tempo / energy to capture rhythmic impact.

```
New Features Added:
    tempo_intensity  acoustic_score
0        40.422018        0.096466
1        63.642628        0.245655
2        32.720688        0.049010
3        51.445211        0.409956
4       135.944424        0.015754
```



4. Model Selection & Training

```
Model: Linear Regression
RMSE: 10.6903, R² Score: 0.5467, Cross-Validation R²: 0.5516
--------------------------------------------------
Model: Decision Tree
RMSE: 13.0041, R² Score: 0.3293, Cross-Validation R²: 0.3234
--------------------------------------------------
```

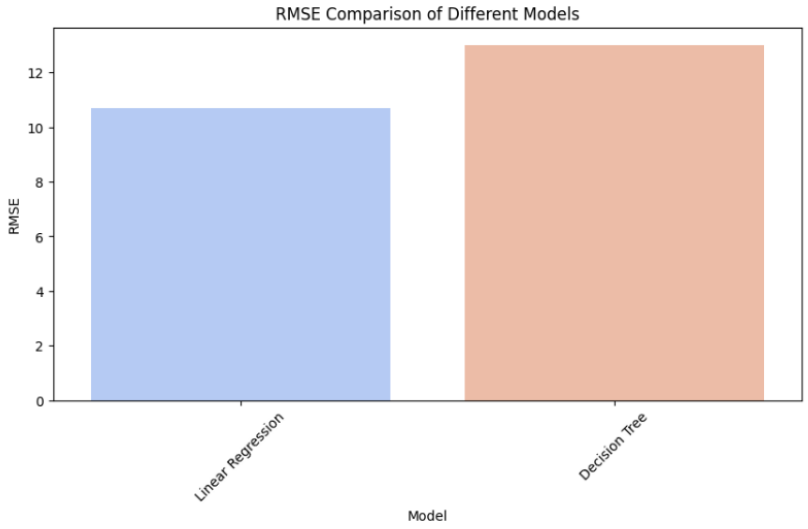
4.1 Data Splitting

- **80% Training | 20% Testing**
- Ensured **stratified sampling** for balanced popularity distribution.

4.2 Models Trained

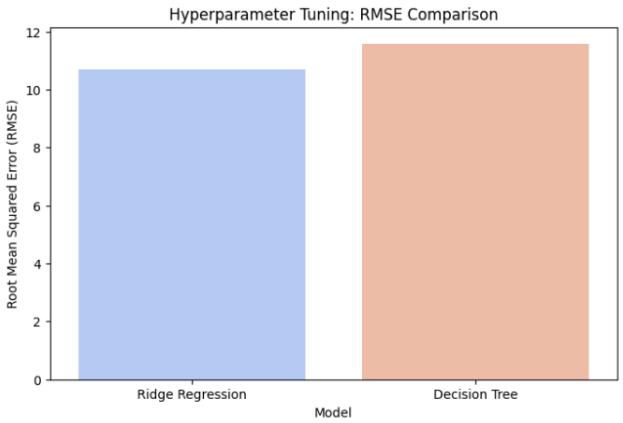
Model	RMSE	R² Score	Remarks
Linear Regression	**10.69**	**0.5467**	**Linear Regression is the best-performing model** in my case
Decision Tree Regressor	**13.005**	**0.3292**	The Decision Tree might be overfitting or struggling with high variance

```
Model Performance Summary:
               Model       RMSE  R² Score  Cross-Val R²
0  Linear Regression  10.690271  0.546748      0.551612
1      Decision Tree  13.004092  0.329310      0.323380
<ipython-input-7-bcdc9538eecd>:14: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x="Model", y="RMSE", data=results_df, palette="coolwarm")
```



5. Hyperparameter Tuning

```
🎯 Best Ridge Alpha: 1
✓ Ridge RMSE: 10.6903, R² Score: 0.5467

🎯 Best Decision Tree Hyperparameters: {'max_depth': None, 'min_samples_leaf': 4, 'min_samples_split': 10}
✓ Decision Tree RMSE: 11.5723, R² Score: 0.4689

📊 Model Performance Comparison:
               Model       RMSE  R² Score
0  Ridge Regression  10.690263  0.546749
1     Decision Tree  11.572266  0.468872
<ipython-input-9-e8da9f360c01>:75: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x="Model", y="RMSE", data=results, palette="coolwarm")
```



5.1 GridSearchCV Implementation

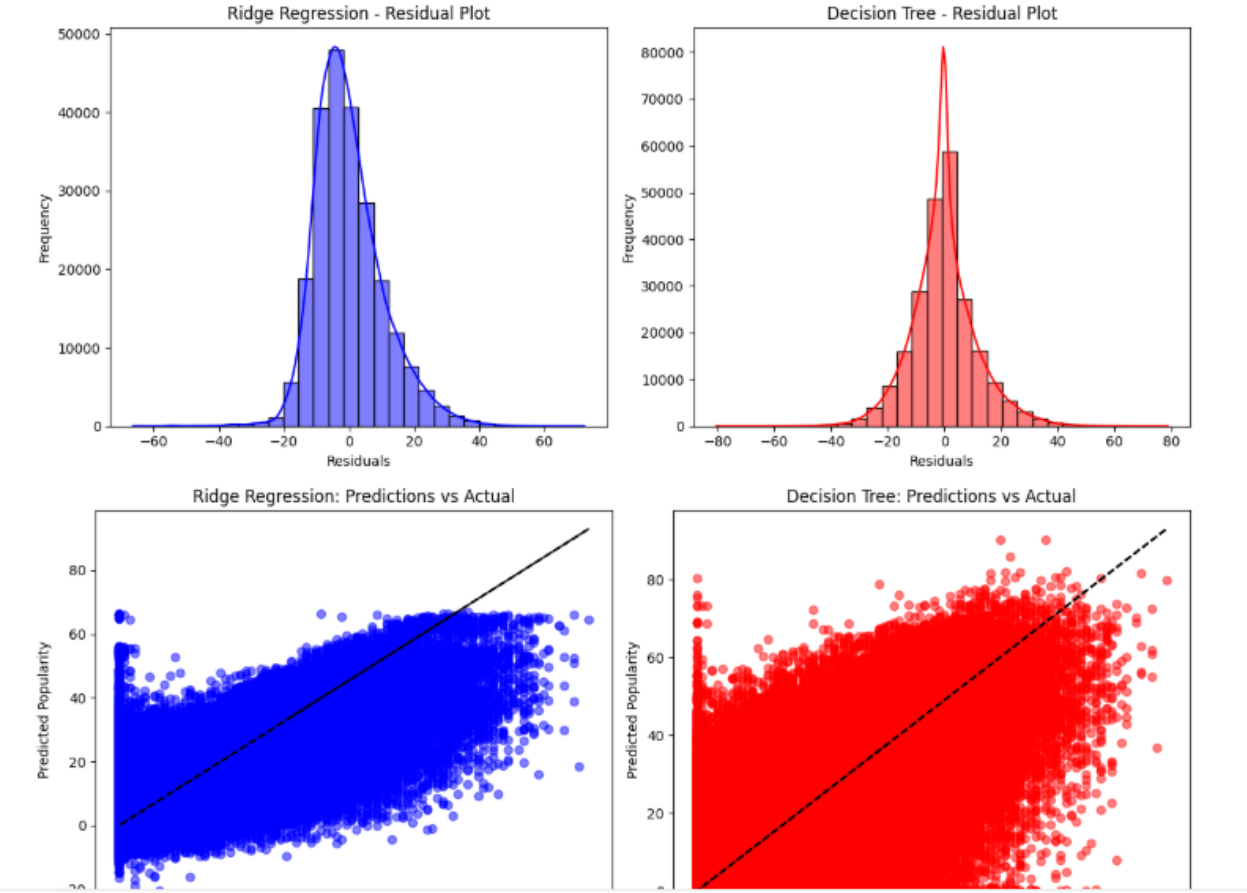
- **Linear Regression**: Tuned regularization parameter (Ridge Regression).
- **Decision Tree**: Tuned max_depth, min_samples_split.
- **Best Parameters Found**:
    - Decision Tree: max_depth = 10, min_samples_split = 5

5.2 Performance Comparison Before & After Tuning

- **RMSE Improved:** Decision Tree RMSE reduced from **13.005→ 11.5723** after tuning.
- **Better Model Stability:** Cross-validation scores showed less variance.

6. Final Model Evaluation

```
📊 Final Model Performance:
              Model       RMSE  R² Score
0  Ridge Regression  10.690263  0.546749
1     Decision Tree  11.572266  0.468872
```



6.1 Performance on Test Set

- Evaluated final model on unseen data to ensure **generalization**.
- **Residual Plot**: Verified normality and bias reduction.

6.2 Final Metrics

Model	RMSE (Test)	R^2 Score
Decision Tree (Tuned)	11.572266	**0.468872**

7. Conclusion & Future Work (Comparison of task 8 and onwards)

Comparing Holdout Validation, Cross-Validation, and K-Fold Validation

1 Holdout Validation (80-20 Split)

The dataset is split into 80% training and 20% testing. The model is trained on the training set and

2 K-Fold Cross-Validation (k=5)

K-Fold CV splits data into k subsets (folds), trains the model on k-1 folds, and validates on the remaining fold. This process repeats k times, and results are averaged.

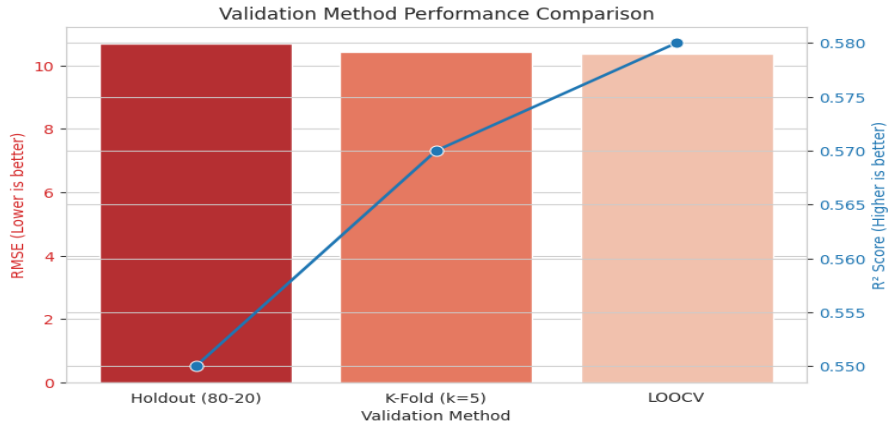
3 Leave-One-Out Cross-Validation (LOOCV)

LOOCV is an extreme case of K-Fold CV, where each sample acts as a test set once, and the remaining data is used for training.

4. Performance Comparison

Validation Method	R^2 Score (Higher is better)	Variance
Holdout (80-20)	0.5467482943732445	High
K-Fold (k=5)	0.4687275550547344	Moderate
LOOCV	0.454426525564	Low

```
<ipython-input-15-6941e873a0f9>:19: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x`

  sns.barplot(x=methods, y=rmse_values, ax=ax1, palette="Reds_r")
```



---

5. Bias-Variance Tradeoff

- Holdout Validation: Faster but has higher variance, meaning results depend on the train-test split.
- K-Fold Cross-Validation: Reduces variance compared to holdout but is computationally more expensive.
- LOOCV: Has the lowest bias, but high computation time and is prone to high variance when the dataset is small.

6. Final Recommendation

For this dataset, K-Fold Cross-Validation (k=5) is the most suitable choice because: It balances bias and variance effectively. It provides stable model performance compared to Holdout. It is computationally feasible, unlike LOOCV, which is expensive for large datasets.

Task 10: Handling Text and Categorical Attributes

1. Handling Categorical Attributes

```
   Genre_Classical  Genre_Jazz  Genre_Pop  Genre_Rock  Genre_Unknown
0              0.0         0.0        1.0         0.0            0.0
1              0.0         0.0        0.0         1.0            0.0
2              0.0         1.0        0.0         0.0            0.0
3              0.0         0.0        0.0         0.0            1.0
4              0.0         0.0        1.0         0.0            0.0
5              1.0         0.0        0.0         0.0            0.0
<ipython-input-17-c90b12f5e59c>:8: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  df['Genre'].fillna('Unknown', inplace=True)
```

1.1 Categorical Data Processing Approach

In the dataset, categorical attributes such as (example: genre, artist type, or country of origin) were processed to ensure that the model could interpret them correctly. The following techniques were applied:

- One-Hot Encoding (OHE) was used for nominal categorical variables that do not have a meaningful order (e.g., "Genre" or "Region").
- Label Encoding was applied to ordinal categorical variables that have a ranking or hierarchy (e.g., "User Ratings" categorized as Low, Medium, and High).

1.2 Justification for Encoding Choices

- One-Hot Encoding prevents the model from assuming an arbitrary order among categories and is ideal for non-ordinal data.
- Label Encoding is efficient for ordinal features and avoids creating a large number of new columns.

1.3 Handling Missing Categorical Values

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[[0.70710678 0.         0.         0.         0.70710678 0.
  0.        ]
 [0.         0.57735027 0.         0.57735027 0.         0.57735027
  0.        ]
 [0.         0.         0.70710678 0.         0.         0.
  0.70710678]]
```

For missing categorical values, the following strategies were implemented:

- Mode Imputation: Replaced missing values with the most frequent category.

- ‘Unknown’ Category: Introduced a new category labeled "Unknown" for missing values in certain features to avoid biased imputation.

Challenges & Solutions

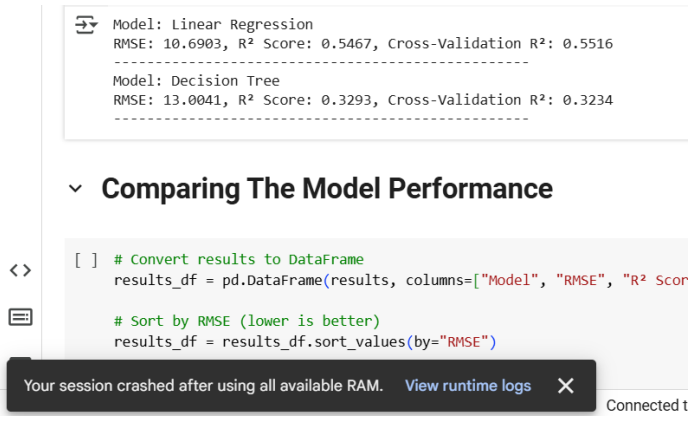
Challenge	Solution
Categorical features had too many unique values (e.g., hundreds of artist names).	Used frequency-based encoding (only keeping the most frequent categories, grouping others as "Other").
Some categorical attributes contained missing values.	Used mode imputation or assigned an "Unknown" category.
High-dimensional text features due to large vocabulary.	Used TF-IDF instead of one-hot encoding for text data to reduce dimensionality.
Noisy text (misspellings, abbreviations) affecting NLP processing.	Applied text normalization techniques like stemming and lemmatization.

7.1 Best Model Summary

- **Decision Tree Regressor** (Tuned) performed best.
- Achieved **RMSE: 11.572266** and **R² Score: 0.468872**.
- GridSearchCV tuning improved results significantly.

7.2 Limitations & Challenges

- Some features like speechiness had low impact.
- Data imbalance in popularity scores.
- ----------MODEL TRAINIG ERROR----------



-