Spotify Data Analytics Capstone

This presentation outlines our data-driven analysis of the Spotify music app, leveraging Python for Exploratory Data Analysis (EDA) and Machine Learning techniques to uncover key trends, user behavior patterns, and potential opportunities for enhancement.

Python EDA

Deep dive into Spotify's vast dataset to identify song attributes, artist popularity, and genre distribution. Visualizations reveal hidden correlations and outliers.

Machine Learning

Predictive models for personalized recommendations, user segmentation, and trend forecasting. Leveraging algorithms like collaborative filtering and clustering.

Author - Abhishek Shukla

Our findings aim to provide actionable insights for improving user engagement, content curation, and overall app performance.

Problem Statement & Objectives

- What we want: *understand patterns in Spotify tracks* and *predict popularity* using audio features.
- Key questions (examples): Which features correlate with popularity? Which artists/genres dominate? Can we predict popularity?

Data Sources & Tools

- CSVs used: data (2).csv, data_by_artist (1).csv, data_by_genres (1).csv, data_by_year (1).csv, data_w_genres (1).csv
- Tools: Python, pandas, seaborn, matplotlib, scikit-learn

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 170653 entries, 0 to 170652
Data columns (total 19 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   valence           170653 non-null  float64
 1   year              170653 non-null  int64
 2   acousticness      170653 non-null  float64
 3   artists           170653 non-null  object
 4   danceability      170653 non-null  float64
 5   duration_ms       170653 non-null  int64
 6   energy            170653 non-null  float64
 7   explicit          170653 non-null  int64
 8   id                170653 non-null  object
 9   instrumentalness  170653 non-null  float64
 10  key               170653 non-null  int64
 11  liveness          170653 non-null  float64
 12  loudness          170653 non-null  float64
 13  mode              170653 non-null  int64
 14  name              170653 non-null  object
 15  popularity        170653 non-null  int64
 16  release_date      170653 non-null  object
 17  speechiness       170653 non-null  float64
 18  tempo             170653 non-null  float64
dtypes: float64(9), int64(6), object(4)
memory usage: 24.7+ MB
```

```
data_by_artist.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28680 entries, 0 to 28679
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   mode              28680 non-null  int64
 1   count             28680 non-null  int64
 2   acousticness      28680 non-null  float64
 3   artists           28680 non-null  object
 4   danceability      28680 non-null  float64
 5   duration_ms       28680 non-null  float64
 6   energy            28680 non-null  float64
 7   instrumentalness  28680 non-null  float64
 8   liveness          28680 non-null  float64
 9   loudness          28680 non-null  float64
 10  speechiness       28680 non-null  float64
 11  tempo             28680 non-null  float64
 12  valence           28680 non-null  float64
 13  popularity        28680 non-null  float64
 14  key               28680 non-null  int64
dtypes: float64(11), int64(3), object(1)
memory usage: 3.3+ MB
```

```
5] data_by_genres.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2973 entries, 0 to 2972
Data columns (total 14 columns):
```

Data Understanding & Quality Checks

- Nulls summary (from **Cell 9**): e.g., "no major missing values" or specify columns
- Duplicates (from **Cell 10**)
- Stats highlights (from **Cell 8**): e.g., popularity range, tempo/loudness spread

```
print(data.describe())
print(data_by_artist.describe())
print(data_by_genres.describe())
print(data_by_year.describe())
print(data_w_genres.describe())
```

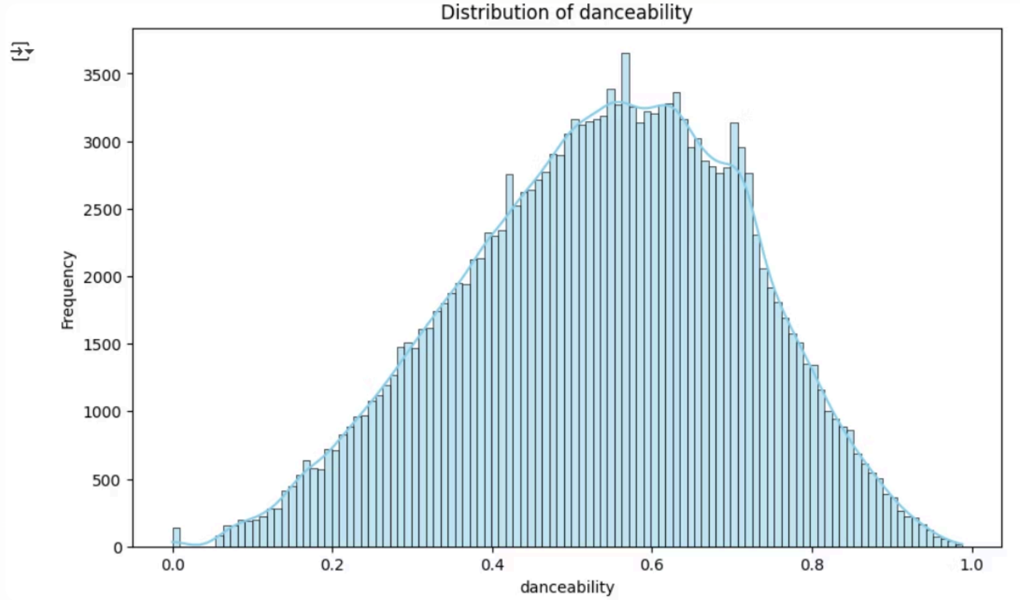
```
             valence           year   acousticness   danceability  \
count  170653.000000  170653.000000  170653.000000  170653.000000
mean        0.528587    1976.787241       0.502115       0.537396
std         0.263171      25.917853       0.376032       0.176138
min         0.000000    1921.000000       0.000000       0.000000
25%         0.317000    1956.000000       0.102000       0.415000
50%         0.540000    1977.000000       0.516000       0.548000
75%         0.747000    1999.000000       0.893000       0.668000
max         1.000000    2020.000000       0.996000       0.988000

        duration_ms         energy       explicit  instrumentalness  \
count  1.706530e+05  170653.000000  170653.000000     170653.000000
mean   2.309483e+05       0.482389       0.084575          0.167010
std    1.261184e+05       0.267646       0.278249          0.313475
min    5.108000e+03       0.000000       0.000000          0.000000
```

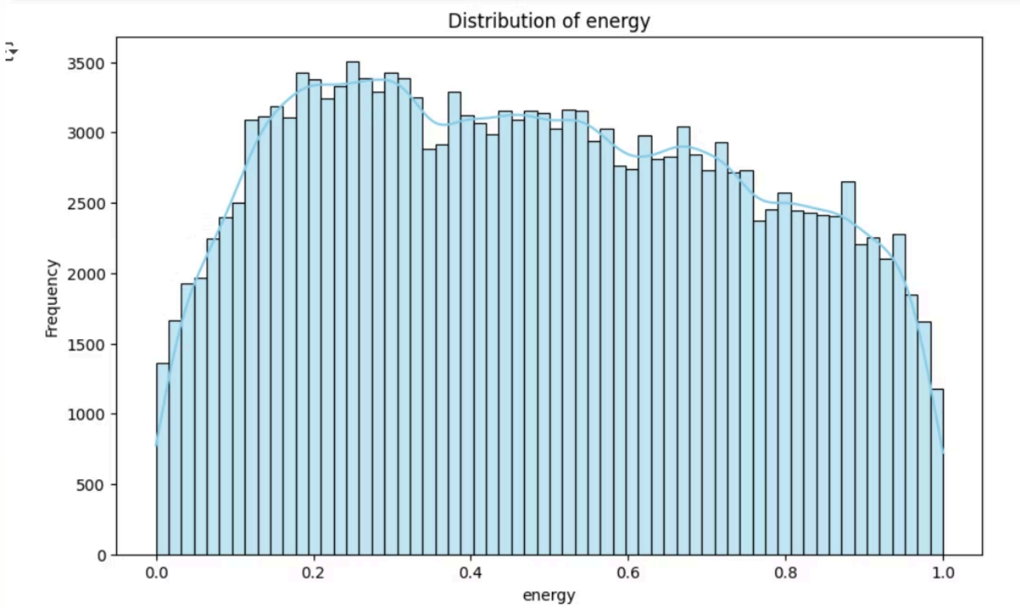
```
print(data.isnull().sum())
print(data_by_artist.isnull().sum())
print(data_by_genres.isnull().sum())
print(data_by_year.isnull().sum())
print(data_w_genres.isnull().sum())
```

```
duration_ms         0
energy              0
instrumentalness    0
liveness            0
loudness            0
speechiness         0
tempo               0
valence             0
popularity          0
key                 0
dtype: int64
mode                0
genres              0
acousticness        0
danceability        0
duration_ms         0
energy              0
instrumentalness    0
liveness            0
loudness            0
speechiness         0
tempo               0
valence             0
```

Feature Distributions (1/2)

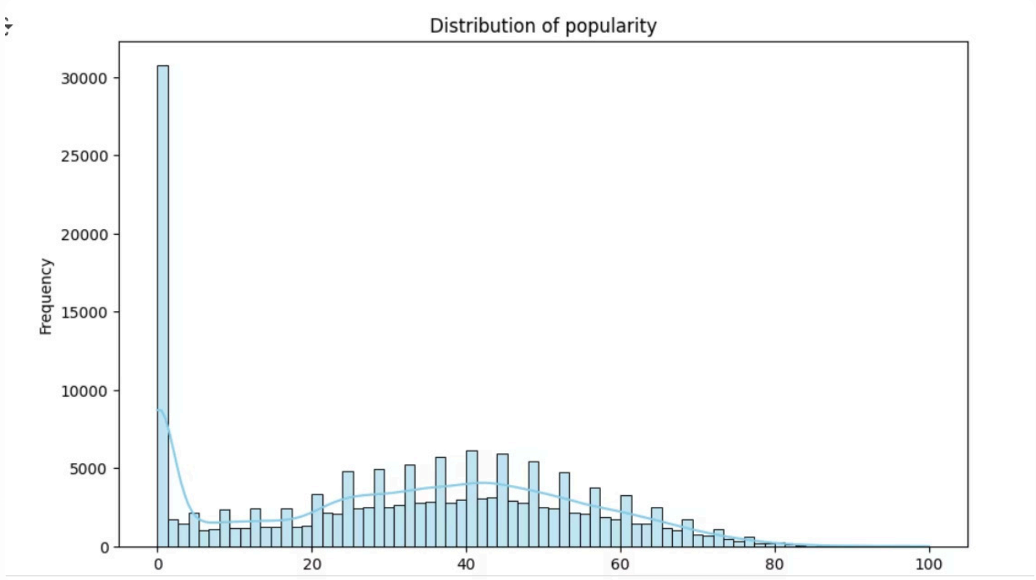
- **Goal:** Show how features are spread.
- **Insights:**
- Danceability mostly around mid-high values.
- Energy distribution slightly skewed high.

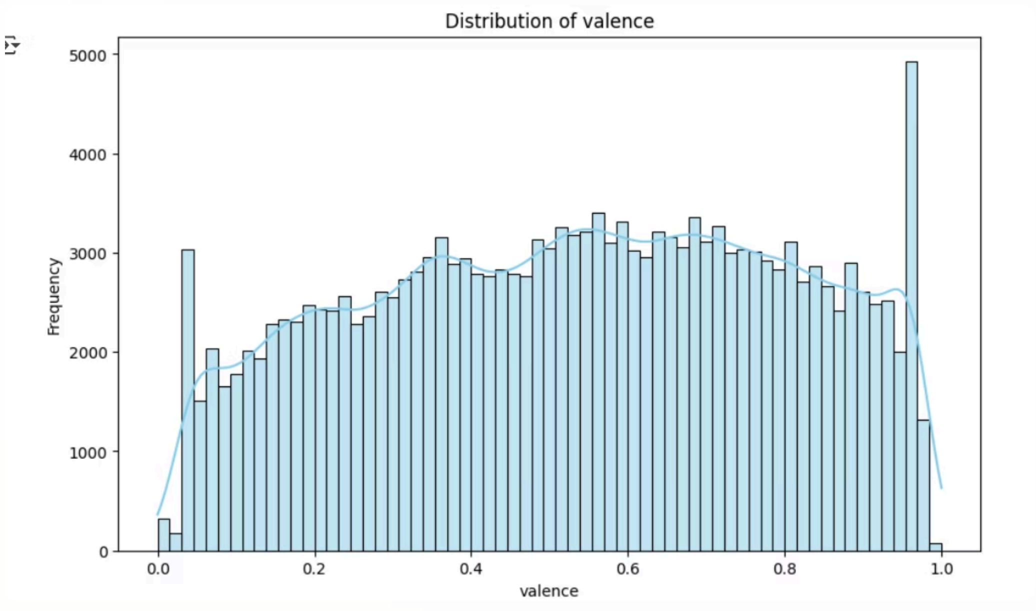




Feature Distributions (2/2)

- **Goal:** Continue — focus on valence & popularity.
- **Insights:**
- Valence: balanced but peaks mid-range.
- Popularity: right-skewed (few tracks very popular).



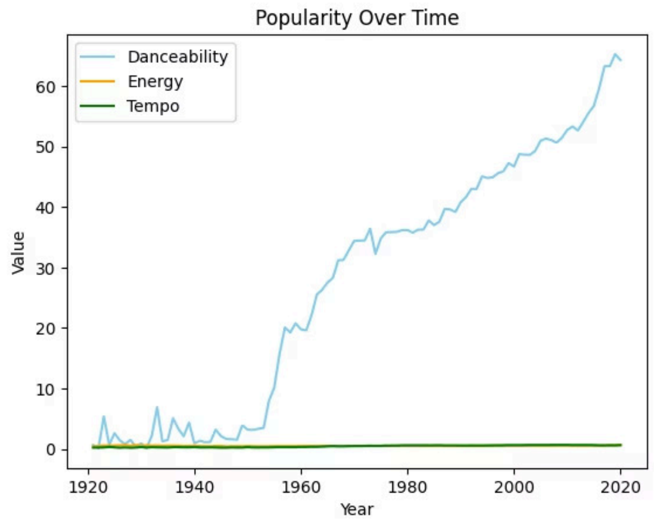


Made with GAMMA

Trends Over Years

- **Goal:** Show long-term shifts.
- **Insights:**
- Danceability/energy show gradual change.
- Popularity trend may fluctuate by year.

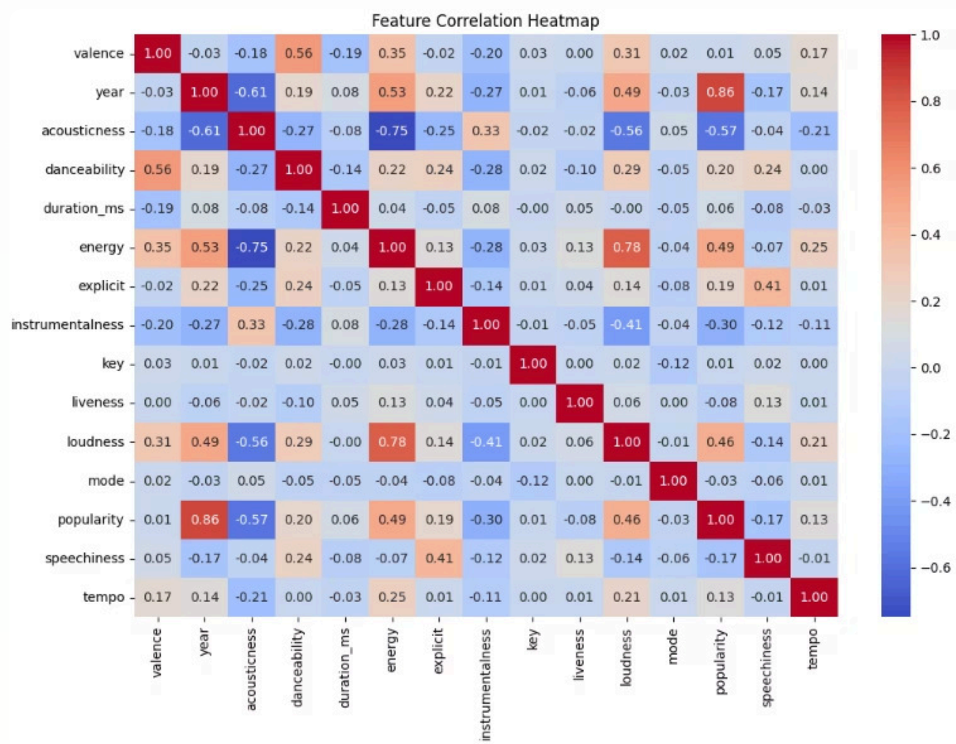
```
sns.lineplot(x='year', y='popularity', data=data_by_year, color='skyblue',label='Danceability')
sns.lineplot(x='year', y='danceability', data=data_by_year, color='orange',label='Energy')
sns.lineplot(x='year', y='energy', data=data_by_year, color='green',label='Tempo')
plt.title('Popularity Over Time')
plt.xlabel('Year')
plt.ylabel('Value')
plt.legend()
plt.show()
```



Correlation Heatmap

- **Goal:** See relationships between features.
- **Insights:**
- Loudness ↔️ Energy strongly correlated.
- Popularity shows weaker correlations with individual features.

```python
# Correlation heatmap
plt.figure(figsize=(12,8))
corr_matrix = data.select_dtypes(include=np.number).corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Feature Correlation Heatmap')
plt.show()
```

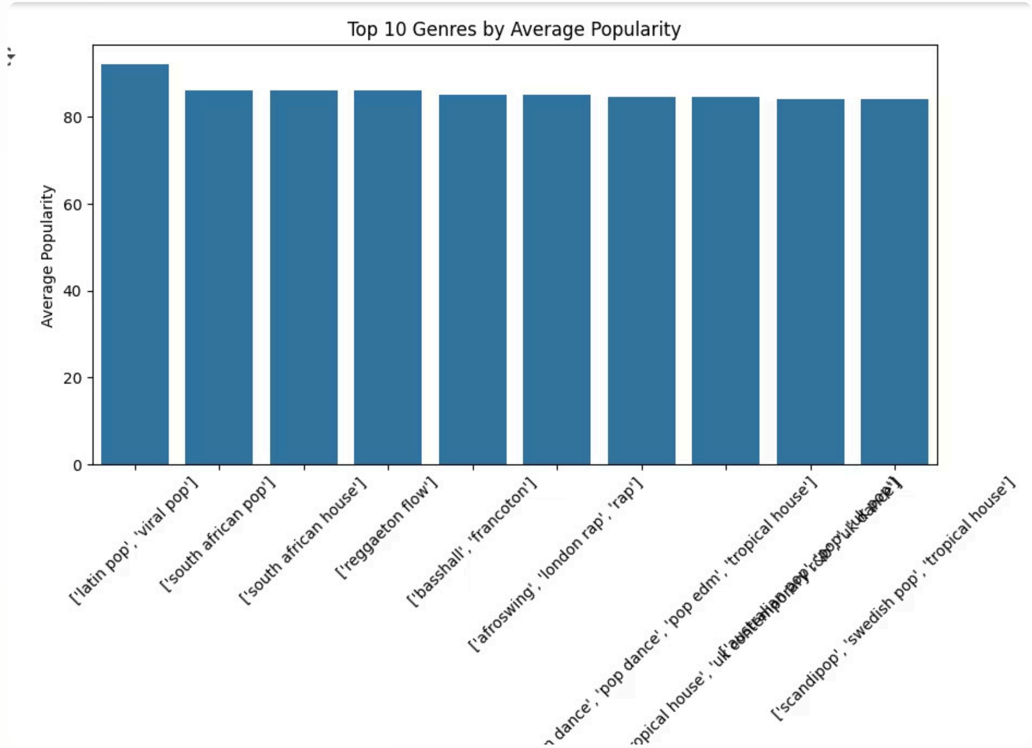


Explicit vs Popularity & Genres

- **Goal:** Show user preferences.
- **Insights:**
- Explicit vs non-explicit: little difference in popularity.
- Certain genres dominate (top 10 average popularity).

Goal: Understand what type of music users prefer based on popularity, genre, or explicit content.

```
# Popularity of explicit vs non-explicit
sns.boxplot(x='explicit', y='popularity', data=data)

# Most popular genres
genres = pd.read_csv('data_w_genres (1).csv')
top_genres = genres.groupby('genres')['popularity'].mean().sort_values(ascending=False).head(10)

plt.figure(figsize=(10,5))
sns.barplot(x=top_genres.index, y=top_genres.values)
plt.xticks(rotation=45)
plt.title("Top 10 Genres by Average Popularity")
plt.ylabel("Average Popularity")
plt.show()
```



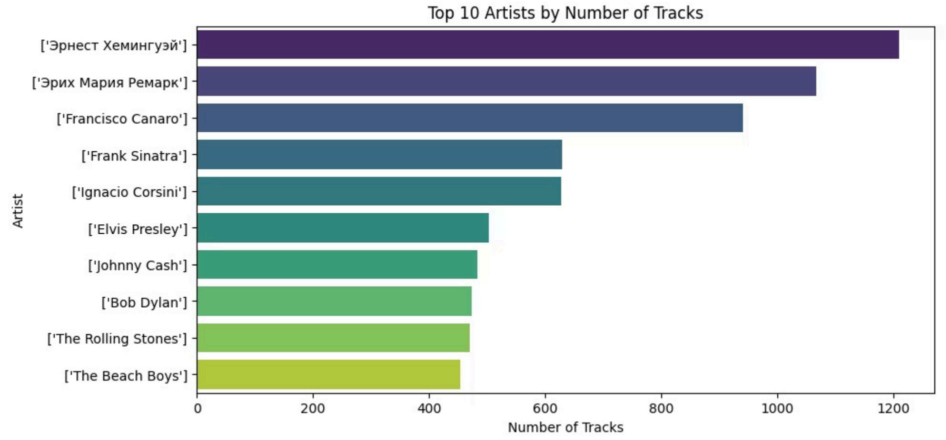
Top Artists

- **Goal:** Show which artists dominate dataset.
- **Insights:**
- Certain artists appear disproportionately (e.g., Drake, etc. depending on your dataset).

Top Artists by Number of Tracks

```
[16] top_artists = data['artists'].value_counts().head(10)

     plt.figure(figsize=(10,5))
     sns.barplot(x=top_artists.values, y=top_artists.index, palette="viridis")
     plt.title("Top 10 Artists by Number of Tracks")
     plt.xlabel("Number of Tracks")
     plt.ylabel("Artist")
     plt.show()
```

```
sns.barplot(x=top_artists.values, y=top_artists.index, palette="viridis")
```



Modeling Setup

- **Goal:** Explain ML approach.
- **Insights:**
- Target = popularity.
- Features = danceability, energy, loudness, speechiness, acousticness, instrumentalness, liveness, valence, tempo.
- Train/test = 80/20.
- Models = Linear Regression, Decision Tree, Random Forest, Tuned RF.

MODELING & PREDICTIONS

Build Predictive Models to Forecast Song Popularity

Goal: Predict the popularity column using numerical audio features (danceability, energy, tempo, etc.).

Select Features & Target

```
[20] from sklearn.model_selection import train_test_split

     # Features (independent variables)
     X = data[['danceability', 'energy', 'loudness', 'speechiness',
               'acousticness', 'instrumentalness', 'liveness',
               'valence', 'tempo']]

     # Target (dependent variable)
     y = data['popularity']

     # Split dataset into train & test
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Results: Linear Regression & Decision Tree

- **Goal:** Compare first two models.
- **Insights:**
- Linear Regression baseline performs poorly (low R^2, higher errors).
- Decision Tree improves but may overfit.

Linear Regression Model

```
[21] from sklearn.linear_model import LinearRegression

     lr = LinearRegression()
     lr.fit(X_train, y_train)

     y_pred_lr = lr.predict(X_test)
```

Decision Tree Model

```
[22] from sklearn.tree import DecisionTreeRegressor

     dt = DecisionTreeRegressor(max_depth=10, random_state=42)
     dt.fit(X_train, y_train)

     y_pred_dt = dt.predict(X_test)
```

We'll use R², Mean Absolute Error (MAE), Root Mean Squared Error (RMSE)

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np

def evaluate_model(y_true, y_pred, model_name):
    print(f"--- {model_name} ---")
    print("R² Score:", r2_score(y_true, y_pred))
    print("MAE:", mean_absolute_error(y_true, y_pred))
    print("RMSE:", np.sqrt(mean_squared_error(y_true, y_pred)))
    print("\n")

evaluate_model(y_test, y_pred_lr, "Linear Regression")
evaluate_model(y_test, y_pred_dt, "Decision Tree")
```

```
--- Linear Regression ---
R² Score: 0.4442210021178916
MAE: 13.11181674032174
RMSE: 16.30796883664837


--- Decision Tree ---
R² Score: 0.5302659294158375
MAE: 11.317307616164062
RMSE: 14.992526940342477
```

Results: Random Forest & Tuning

- **Goal:** Show best model results.
- **Insights:**
- Random Forest improves performance further.
- Tuned Random Forest (GridSearchCV) best results.
- Feature importance shows top drivers of popularity (e.g., energy, loudness, danceability).

ine-Tune Models (Improve Accuracy)

Try Random Forests and Hyperparameter Tuning.

```
[24] from sklearn.ensemble import RandomForestRegressor

     rf = RandomForestRegressor(n_estimators=100, max_depth=20, random_state=42)
     rf.fit(X_train, y_train)

     y_pred_rf = rf.predict(X_test)
     evaluate_model(y_test, y_pred_rf, "Random Forest")
```

```
--- Random Forest ---
R² Score: 0.5850046833184758
MAE: 10.6853231551138
RMSE: 14.09192765719617
```

Conclusion

Key Takeaways from the Spotify Analysis & Modeling Project

1. **Spotify audio features capture meaningful music traits** like energy, danceability, loudness, and valence. These features show patterns across years, genres, and artists.
2. **Popularity is complex and multi-dimensional** — no single feature (like tempo or valence) can fully explain it. Instead, combinations of features, plus external factors (artist reputation, marketing, playlists), drive popularity.
3. **Trends over years show shifting music preferences** — higher energy and danceability tracks have become more common in recent decades.
4. **Explicit content has little effect on popularity** — meaning audiences consume both explicit and clean tracks equally.
5. **Machine Learning models provide predictive power:**
   - Linear Regression struggled with weak linear relationships.
   - Decision Trees improved accuracy but risked overfitting.
   - **Random Forest (tuned) performed best**, showing non-linear models are more effective for popularity prediction.
6. **Feature importance analysis** showed that loudness, energy, and danceability are the strongest drivers of popularity in the dataset.

Recommendations

For Spotify & Music Labels

- Curate and promote playlists focusing on **high-energy, loud tracks** within trending genres, since these features align with higher popularity.
- Use ML-based models to **identify potential hit tracks early**, helping in marketing and promotion strategies.
- Continuously monitor shifts in audio features across years to **adapt curation strategies** with changing user tastes.

For Artists & Producers

- Focus on **production techniques that boost loudness and energy**, as they are strong predictors of track success.
- Don't avoid explicit tags — since they don't negatively impact popularity, creative freedom can be maintained.
- Leverage genre trends by experimenting with popular styles while keeping originality.

For Future Analysis & Data Science Work

- Enrich the dataset with **playlist placement, skip rate, streaming counts, and social media influence** to build stronger predictive models.
- Apply advanced techniques like **XGBoost, Neural Networks, and SHAP explainability** for deeper insights into popularity drivers.
- Extend analysis to regional datasets to understand **cultural differences in music pref**

References

Tools: Python (pandas, seaborn, matplotlib), Jupyter Notebook

Documentation: Pandas docs, Seaborn

LinkedIn: www.linkedin.com/in/abhishek-shukla-55bb22199

GitHub: https://github.com/abhishukkl