Spotify Audio Feature Analysis & Popularity Prediction

VINAYAK VEMULA

CSIT 558

Data Mining Project

Objective

- Analyze Spotify audio features to understand factors influencing song popularity.
- Apply data mining techniques including classification, regression, and clustering.
- Compare model performance through parameter tuning and multiple seed experiments.
- Identify key patterns and correlations within audio feature data.
- Develop a simple rule-based classifier for user interaction and prediction.

Dataset

- Dataset: Spotify Audio Features Dataset
- Features include: danceability, energy, loudness, valence, tempo, etc.
- Target: Popularity score
- Preprocessing: Scaling, cleaning, feature selection.

```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.cluster import KMeans
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

import warnings
warnings.filterwarnings("ignore")

df = pd.read_csv("spotify_tracks_dataset.csv")

df.head()
```

Dataset

	Unnamed: 0	track_id	artists	album_name	track_name	popularity	duration_ms	explicit	danceability	energy	...
0	0	5SuOikwiRyPMVoIQDJUgSV	Gen Hoshino	Comedy	Comedy	73.0	230666.0	False	0.676	0.4610	...
1	1	4qPNDBW1i3p13qLCt0Ki3A	Ben Woodward	Ghost (Acoustic)	Ghost - Acoustic	55.0	149610.0	False	0.420	0.1660	...
2	2	1iJBSr7s7jYXzM8EGcbK5b	Ingrid Michaelson;ZAYN	To Begin Again	To Begin Again	57.0	210826.0	False	0.438	0.3590	...
3	3	6lfxq3CG4xtTiEg7opyCyx	Kina Grannis	Crazy Rich Asians (Original Motion Picture Sou...	Can't Help Falling In Love	71.0	201933.0	False	0.266	0.0596	...
4	4	5vjLSffimiIP26QG5WcN2K	Chord Overstreet	Hold On	Hold On	82.0	198853.0	False	0.618	0.4430	...

5 rows × 21 columns

Predictive Techniques

- **Decision Tree** – Simple baseline model, easy to interpret.
- **Random Forest** – More accurate and robust; best performance.
- **Linear Regression** – Predicts continuous popularity scores.
- **K-Means Clustering** – Finds natural song groups based on audio features.

Parameter Experiments

- **Decision Tree**

  Tested max_depth values: **3, 5, 8**

- **Random Forest**

  Tested number of trees (n_estimators): **50, 100, 200**

- **Model Robustness (Multiple Seeds)**

  Evaluated with random seeds: **0, 21, 42**

Experiment Results (Classification)

1. Decision Tree Results

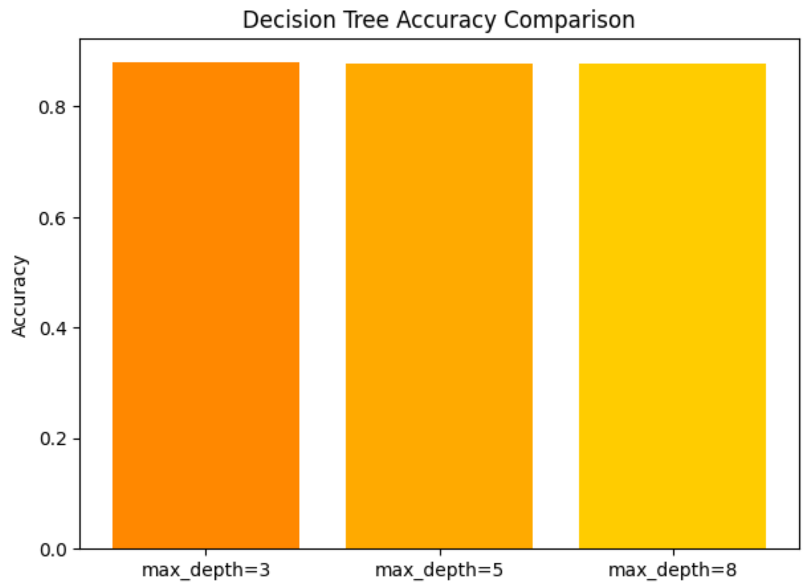
- Accuracy varies with depth
- Moderate performance, serves as a baseline model

```
results_dt = {}

for depth in [3, 5, 8]:
    model = DecisionTreeClassifier(max_depth=depth, random_state=42)
    model.fit(X_train, y_train)
    pred = model.predict(X_test)
    acc = accuracy_score(y_test, pred)
    results_dt[f"max_depth={depth}"] = acc

results_dt
```

```
{'max_depth=3': 0.8791552019037704,
 'max_depth=5': 0.879006469844575,
 'max_depth=8': 0.8778166133710121}
```



Experiment Results (Classification)

2. Random Forest Results

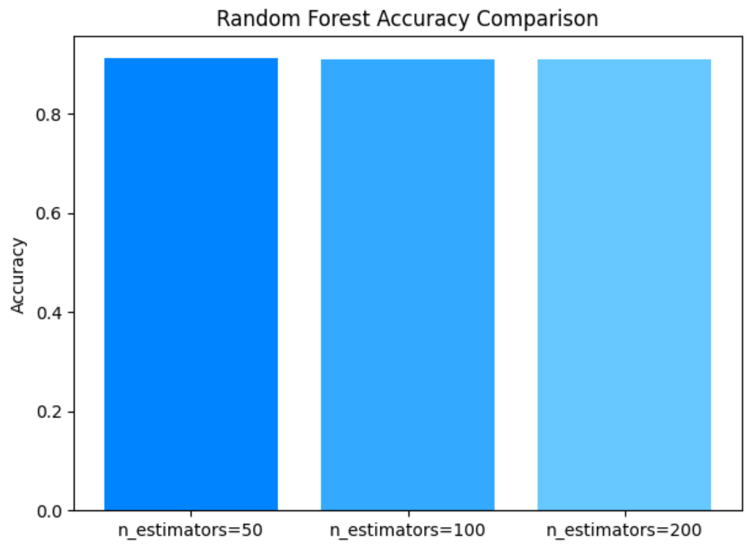
- Highest accuracy among classifiers
- Performance improves with more trees (up to 200)
- Most stable and robust model

```
results_rf = {}

for n in [50, 100, 200]:
    model = RandomForestClassifier(n_estimators=n, random_state=42)
    model.fit(X_train, y_train)
    pred = model.predict(X_test)
    acc = accuracy_score(y_test, pred)
    results_rf[f"n_estimators={n}"] = acc

results_rf
```

```
{'n_estimators=50': 0.9106120324235889,
 'n_estimators=100': 0.9105376663939913,
 'n_estimators=200': 0.9105376663939913}
```



Experiment Results (Classification)

3. Seed Robustness

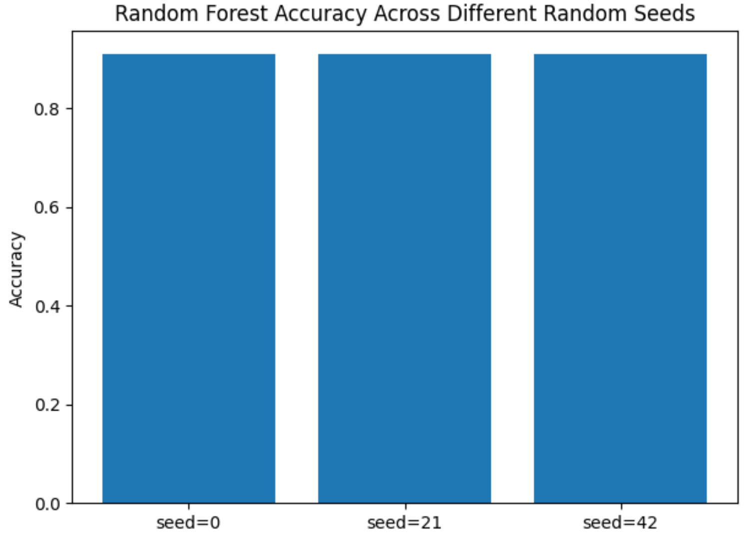
- Accuracy remains consistent across seeds (0, 21, 42)
- Confirms reliability of Random Forest

```
rf_seed_results = {}

# Using the best-performing n_estimators (100) with different random seeds
for seed in [0, 21, 42]:
    model = RandomForestClassifier(n_estimators=100, random_state=seed)
    model.fit(X_train, y_train)
    pred = model.predict(X_test)
    acc = accuracy_score(y_test, pred)
    rf_seed_results[f"seed={seed}"] = acc

rf_seed_results
```

```
{'seed=0': 0.9107607644827843,
 'seed=21': 0.9107607644827843,
 'seed=42': 0.9105376663939913}
```



Regression Results

Linear Regression

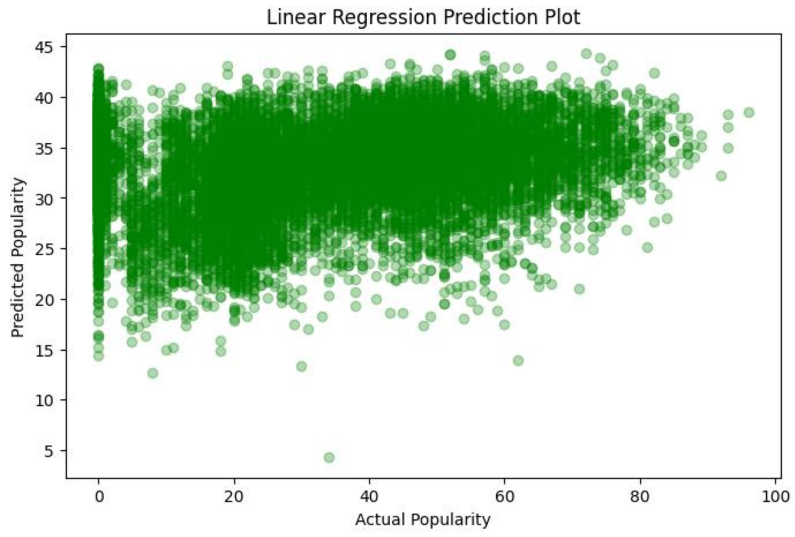
- Predicts continuous popularity values
- Shows moderate correlation between features and popularity
- RMSE indicates prediction error is reasonable for music data
- Useful for understanding numeric relationships

```
lr = LinearRegression()
lr.fit(X_train_scaled, df.loc[X_train.index, "popularity"])
pred_lr = lr.predict(X_test_scaled)
```

```
from sklearn.metrics import mean_squared_error

mse = mean_squared_error(df.loc[X_test.index, "popularity"], pred_lr)
rmse = mse ** 0.5
rmse
```

```
21.25136228779178
```



Clustering Results (K-Means)

Elbow Method Analysis

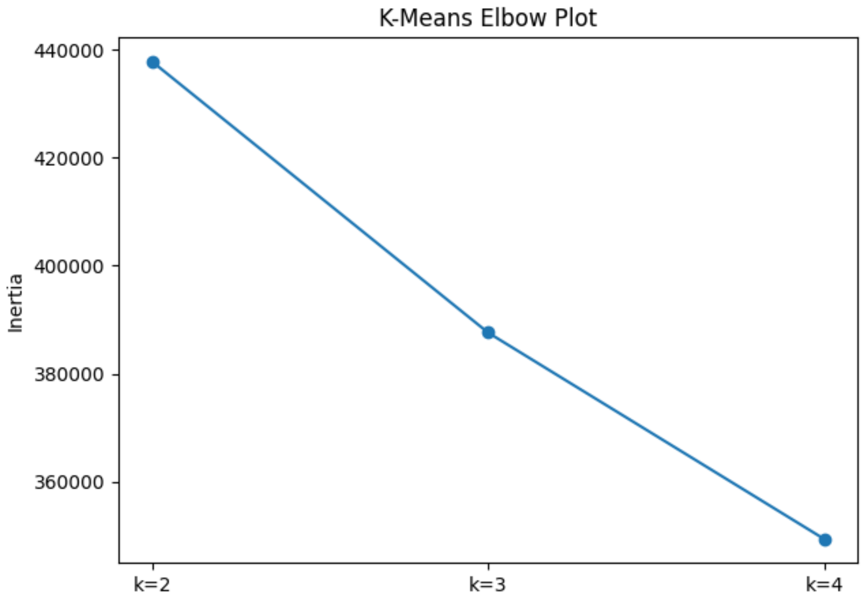
- Tested multiple values of *k* to identify optimal clusters
- Inertia decreased steadily, with **k = 3** showing a balanced point

```
cluster_scores = {}

for k in [2, 3, 4]:
    km = KMeans(n_clusters=k, random_state=42)
    km.fit(X_scaled)
    cluster_scores[f"k={k}"] = km.inertia_

cluster_scores
```

```
{'k=2': 437881.1416230503,
 'k=3': 387595.97000393976,
 'k=4': 349315.67708857916}
```



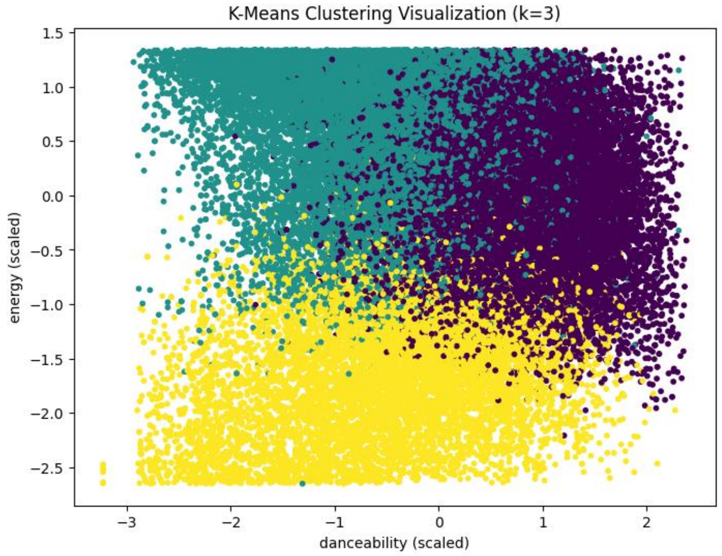
K-Means Visualization

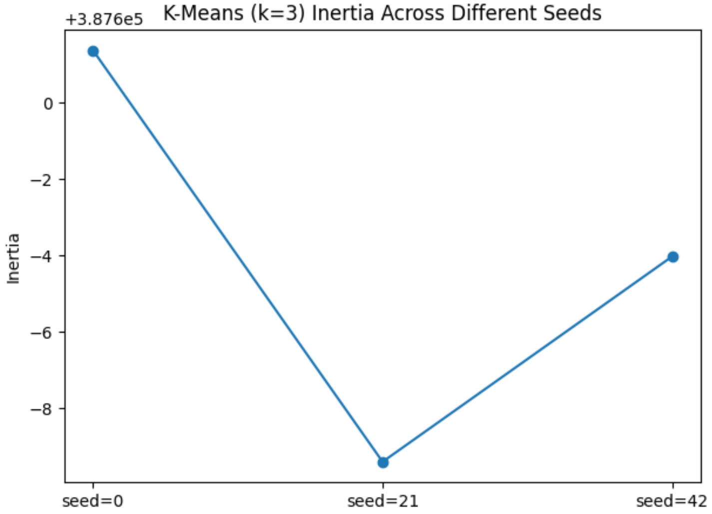
Cluster Visualization

- Songs form **3 distinct groups** based on scaled audio features
- Indicates natural separation in feature space (e.g., energy, danceability)

Robustness Across Seeds

- Inertia values remain consistent for seeds **0, 21, 42**
- Confirms clustering stability and reliable structure in the dataset





Mini Classification Tool

Purpose

- A simple rule-based classifier created using insights from model results
- Allows users to input audio features and receive a popularity prediction

Logic Used

- High danceability + high energy → **Hit**
- High acousticness → **Not a Hit**
- Otherwise → **Maybe a Hit**

```python
def classify_song(danceability, energy, acousticness):
    if energy > 0.65 and danceability > 0.60:
        return "Hit Song"
    if energy < 0.40 and acousticness > 0.60:
        return "Not a Hit"
    return "Maybe a Hit"
```

User Interaction Demo

Purpose

- Demonstrates how the mini classification tool responds to new, unseen inputs
- Shows real predictions based on user-entered audio feature values

3 User Interaction Runs

```
classify_song(0.72, 0.80, 0.10)
```

'Hit Song'

```
classify_song(0.40, 0.35, 0.75)
```

'Not a Hit'

```
classify_song(0.55, 0.50, 0.25)
```

'Maybe a Hit'

Conclusion

Key Findings

- **Random Forest** achieved the highest and most stable classification accuracy
- **Decision Tree** served as a simple baseline with moderate performance
- **Linear Regression** showed reasonable prediction ability for popularity scores
- **K-Means Clustering** revealed **3 natural song groups** based on audio features

Model Robustness

- Parameter tuning and multiple seed tests confirmed consistent performance
- Random Forest proved to be the most reliable model across experiments

Overall Summary

- Audio features strongly influence song popularity
- The **rule-based mini classifier** effectively demonstrates learned insights
- Combines classification, regression, and clustering for a complete analysis

References

Dataset Source

- Spotify Audio Features Dataset (Kaggle)

ML Libraries

- Scikit-learn, Pandas, NumPy, Matplotlib, Seaborn

Preprocessing Reference

- Han, J., Kamber, M., & Pei, J. (2012). *Data Mining: Concepts and Techniques.*

Music ML Research

- Schedl, M. et al. (2014). *Music Information Retrieval using Audio Features.*

Hyperparameter Tuning

- Bergstra, J., & Bengio, Y. (2012). *Random Search for Hyper-Parameter Optimization.*