Machine Learning - Web Design Project

Project 4- Group 6

Adam Loux, Misha Mambully Muralidharan, Cecilia Rocha & Willian Ruiz

Introduction

Music is something that connects people together beyond borders and cultures. Our team really believes in this sentiment and decided to focus in on it.In our project we are exploring a spotify dataset from kaggle to predict the danceability of the songs with the help of several audio features using machine learning .We are also examining the song's popularity trends and patterns through Data Visualization. We have used Jupyter Notebooks to clean the data and create the ML models.The model was created with Python using libraries such as pandas, scikit-learn.The model was then integrated into a web application using Flask.We used javascript,HTML,css for the front end.Data visualization was done through Tableau.

Key Research Questions

1) Can machine learning models accurately classify songs into danceability categories (Low, Medium, High) based on audio features in the dataset?

2) What mood categories correlate with the highest streams and the most popular songs?

3) How do audio features like energy, tempo, loudness and danceability vary across music genres?

Dataset Overview

DataSet Source :Kaggle
https://www.kaggle.com/datasets/maharshipandya/-spotify-tracks-dataset/data

https://www.kaggle.com/datasets/abdulszz/spotify-most-streamed-songs

Data Cleaning and Feature Engineering

In this project, We worked with multiple datasets. Each dataset had specific preprocessing steps tailored to its purpose. Below is a breakdown of how we processed the datasets.

Data Preparation for Machine Learning

Removed unnecessary columns which were not required for ML.

```
# Drop unwanted columns
cols_to_drop = ["Unnamed: 0", "track_id","artists" ,"album_name","track_name",'track_genre',"duration_ms","explicit"]
df = df.drop(columns=cols_to_drop)
```

Many machine learning algorithms cannot handle missing values, so it was required to fill them in before training the model.
For the numerical columns, We used the SimpleImputer from sklearn, which replaced missing values with the median of each column.

```
# Define preprocessing for numeric features (impute and scale)
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
```

We binned our target variable ‘danceability’ into low, medium and high values for our classification.

```
# We will first define the bins and labels
bins = [0.0, 0.33, 0.66, 1.0]
labels = ['Low', 'Medium', 'High']

# Then we Apply the  binning to the popularity column
df['danceability'] = pd.cut(df['danceability'], bins=bins, labels=labels, right=True, include_lowest=True)
```

Data Cleaning and Feature Engineering for Tableau Visualization

Renamed columns:For clarity one of the columns was renamed.

```
# Renaming the column "Unnamed: 0" to "id"
df.rename(columns={'Unnamed: 0': 'id'}, inplace=True)
```

We filled out the missing columns with values.

```
# # 2. Filling out the  missing values in artists, album_name, and track_name
df['artists'].fillna('Unknown Artist', inplace=True)
df['album_name'].fillna('Unknown Album', inplace=True)
df['track_name'].fillna('Unknown Track', inplace=True)
```

Machine Learning : Danceability Classification Model for Songs

Initially, we tried to predict song popularity using regression models. However, during the exploratory phase, the results of various models indicated that the R squared values were poor, meaning the regression models were not adequately predicting the popularity of the songs. This led us to reconsider our approach.

After that we decided to shift our focus on predicting popularity to classifying the danceability of songs. The main reason for this shift was that classifying danceability is a more unique and interesting challenge. Predicting the danceability of a song based on features like energy, tempo, and loudness offers more room for innovation.

For the classification task, we initially used several machine learning models to see which one would perform best on the danceability task.

We applied the following algorithms:
Logistic Regression
Random Forest
Gradient Boosting
LightGBM (LGBM)
XGBoost (XGB)

After training the models, we found that the performance of all models was not that great.Random Forest was giving us perfect scores with the AUC scores giving a perfect 1 for train metrics and 0.94 for test metrics clearly indicating a case of overfitting.LightGBM was only model that was worth considering after looking at the values.

We can look at the Train Metrics and Test Metrics below.

```
TRAIN METRICS
    Confusion Matrix:
    [[19887    46  7680]
 [   99  4626  3936]
 [ 5474   985 42767]]

    AUC: 0.9127782735566777

    Classification Report:
                  precision    recall  f1-score   support

            High       0.78      0.72      0.75     27613
             Low       0.82      0.53      0.65      8661
          Medium       0.79      0.87      0.83     49226

        accuracy                           0.79     85500
       macro avg       0.80      0.71      0.74     85500
    weighted avg       0.79      0.79      0.78     85500
```

The model performs best at predicting songs with Medium danceability, both in terms of precision -79%, recall -87%, and F1-score -83%. This could be because the distribution of the Medium class in the data is more prominent, and the model is better at learning from it.The Low class had the lowest recall (53%), indicating that many low-danceability tracks were misclassified as medium or high.The precision for this class was also 82%, which means that when the model predicted Low, it was accurate. But, the lower recall suggests it missed many of the low-danceability tracks.

The High class performed decently, with a precision of 78% and recall of 72%, showing the model's ability to identify high-danceability tracks, though there is still room for improvement in correctly classifying them.

```
TEST METRICS
    Confusion Matrix:
    [[ 6366    24  2759]
 [    37  1481  1468]
 [ 1992   405 13968]]

    AUC: 0.8942835902806459

    Classification Report:
                  precision    recall  f1-score   support

        High       0.76      0.70      0.73      9149
         Low       0.78      0.50      0.60      2986
      Medium       0.77      0.85      0.81     16365

    accuracy                           0.77     28500
   macro avg       0.77      0.68      0.71     28500
weighted avg       0.77      0.77      0.76     28500
```

On the test set, the model achieved an accuracy of 77%, which is slightly lower than the training set.This is because the model might be slightly overfitting in the training data.The Medium class again showed strong performance with a recall of 85% and an F1-score of 81%. The model does a good job of identifying medium danceability tracks on the test set as well. The Low class suffered from lower recall of 50% and precision of 78%, which indicates that the model still struggles with correctly identifying low danceability tracks on new, unseen data.The High class again performed reasonably well with a precision of 76% and recall of 70%.

We decided to use cross-validation to assess the LightGBM model's stability, and found the AUC Scores: [0.8838561, 0.87676837, 0.87162605, 0.89344772, 0.86793699] and the Mean AUC: 0.879.The AUC scores are relatively consistent across the five folds, ranging from 0.8679 to 0.8934. This suggests that the model's

performance is stable.The AUC scores are all above 0.85, which indicates that the model is performing well.

Since LightGBM provided the best results when compared with all the other models we tried , we retrained the model on the full dataset using LGBM. The final model was evaluated and saved for deployment in the web application.

We also did a recommender for our project.For our recommender system, we used the K-Nearest Neighbors algorithm to recommend tracks to users based on a given track's features like popularity, danceability ,energy, loudness, speechiness, acousticness. KNN works by finding the most similar data points to a given query point, based on a distance metric, and using these neighbors to make predictions or recommendations.

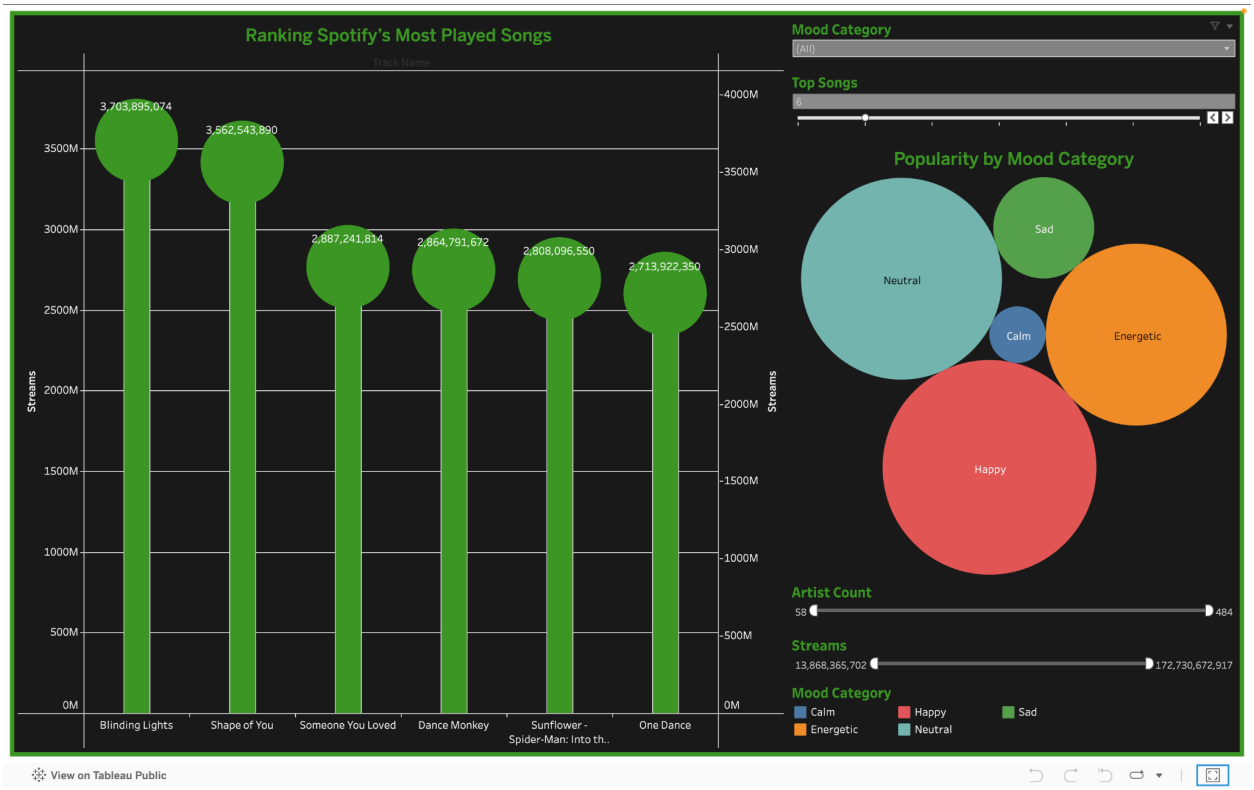
After preprocessing the features, we fit the KNN model to the dataset. The model uses the Cosine distance metric to find tracks that are similar to the target track.The idea was to make the user input the track name and the artist and then recommender would recommend 5 similar tracks.Because of time constraints we did not use our model in our website but we did have recommended page that used the dataset directly.

We implemented a content based recommender system using the KNN algorithm, which recommends tracks based on the features of the tracks themselves (such as popularity, danceability, energy, etc.), another widely used technique in recommender systems is collaborative filtering.

Collaborative filtering can offer more personalized recommendations, as it is based on the preferences of users with similar tastes.We could use collaborative filtering in future iterations of the recommender system by incorporating user interactions such as track ratings or play history. Combining content based and collaborative filtering could further enhance the recommendation quality by considering both item characteristics and user preferences.

Tableau Visualizations

1. This dashboard offers a better analysis of how mood categories affect song popularity and performance metrics on Spotify.



Lollipop Chart:

The lollipop chart displays the streams of the top 30 songs based on their popularity. Each point represents a song's stream count, providing a visual representation of their rank. You can also filter the data by mood category (happy, sad, energetic, calm, neutral). This allows us to explore how songs in different mood categories perform in terms of streams.

The distribution of moods in the top 30 songs reveals interesting insights into listener preferences. Energetic songs make up only 6 out of the top 30, indicating that while upbeat tracks have a place in popular music, they are not the most dominant mood. Happy songs are even less represented, with only 4 songs in the top 30, suggesting that while happy music is impactful,

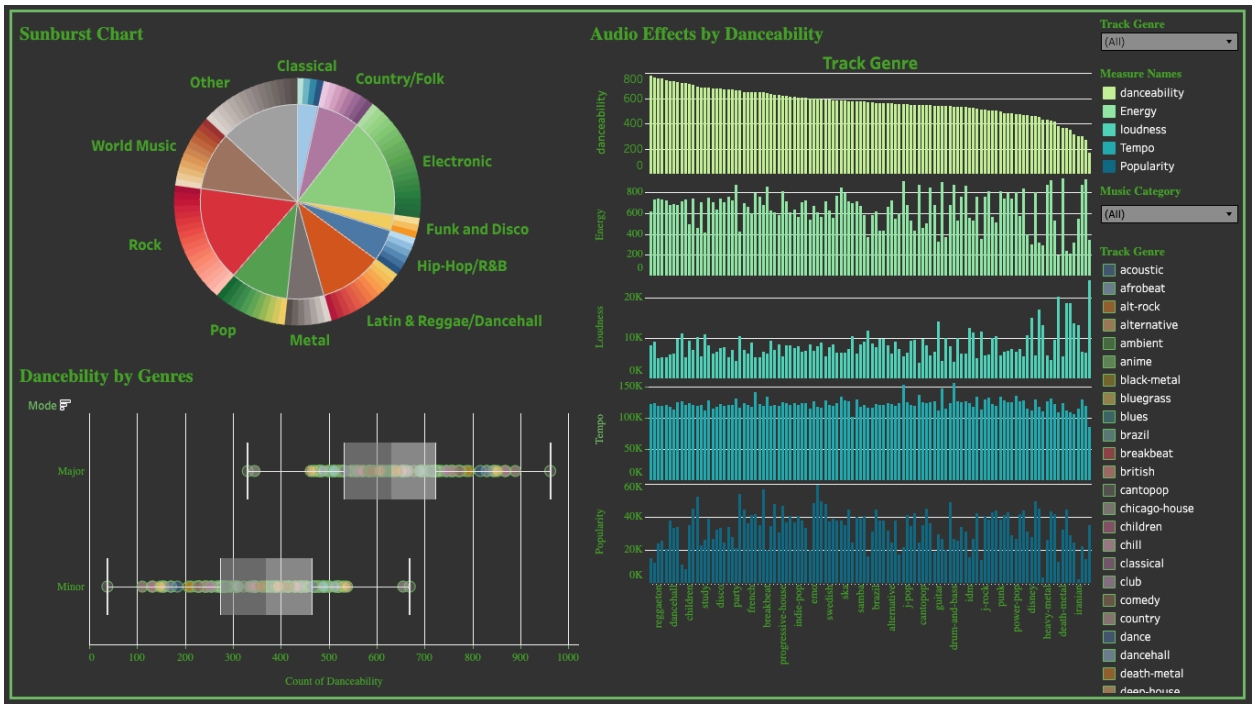
it competes with other emotional tones. The largest portion, with 12 songs, falls under the neutral mood category, indicating that neutral tracks are the most popular. These songs are versatile, appealing to a wide audience in various contexts. Finally, 8 songs are classified as sad, showing that melancholic or reflective music also holds significant appeal, often resonating with listeners during emotional moments. Overall, this distribution highlights that while energetic and happy songs are important, neutral and sad moods seem to dominate in terms of popularity.

Bubble Chart:

The bubble chart visualizes the mood categories with respect to artist count, streams, and Spotify playlist appearances. Each mood is represented as a bubble, and the size of the bubble represents the number of artists and streams in that category. Filters allow for a better view of how many artists are associated with a particular mood and the streams within each mood category.

The data from Spotify's entire catalog shows that neutral and happy songs dominate in terms of overall streams, with neutral tracks leading by a significant margin. Energetic songs also perform well, though not as strongly as the neutral and happy categories. Sad songs, despite having a smaller number of artists, still accumulate a notable number of streams, indicating their continued relevance in the music landscape. Calm songs, while still heard, have the smallest number of streams, suggesting they appeal to a more specific audience or purpose, such as relaxation or background music. This distribution highlights the diverse emotional spectrum of Spotify's music catalog, where neutral and happy tracks resonate most with a wide range of listeners.

2. This Tableau dashboard is designed to provide a comprehensive analysis on one of our high-level questions: How do audio features like energy, tempo, loudness and danceability vary across music genres*?*



Sunburst Chart:

This sunburst chart visually represents how each genre fits within a broader music category, offering insights into the distribution and relationships between different genres in the dataset. To enhance clarity, we first grouped related music genres under a new parent category called 'Music Category,' which includes broad classifications such as Rock, Pop, and Hip-Hop.

However, due to the size and diversity of the Rock genre, we introduced an additional parent category specifically for the 'Metal' subgenre group. While Metal falls under Rock, its complexity and prominence warranted a separate distinction.

Danceability Box-and-Whisker Plot:

The Danceability box-and-whisker plot was created to explore the relationship between music genres and their danceability scores, with a

focus on understanding how different genres perform based on musical mode. To prepare the data, we used the ‘Mode’ and ‘Genre’ fields, which were key in categorizing and analyzing the danceability of tracks.

The 'Mode' field, initially represented by binary values (1 and 0), indicates whether a track is in a major or minor key. We transformed these binary values into more intuitive labels, changing "1" to "Major" and "0" to "Minor," making it easier to interpret the differences in danceability between major and minor keys.

Overall, this box plot was used to identify potential trends and outliers in danceability scores. However, the results showed that no particular genre exhibited a significant trend in danceability, suggesting that danceability is not strongly influenced by genre or key alone.

Audio Effects Bar Chart:

We also created a series of bar charts to highlight key audio features such as danceability, energy, loudness, tempo, and popularity, and to examine how these features vary across different music genres. Using the 'Track Genre' field to segment the data, we aimed to uncover patterns in the characteristics that influence how music is perceived and categorized.

Initially, we focused on popularity as the primary variable and sorted the chart based on this feature. However, the results showed that popularity scores did not vary significantly across most genres, making it less informative for drawing meaningful conclusions.

As we shifted focus to the additional audio features, distinct genre-related patterns began to emerge:

- **Tempo:** Measured as the speed or pace of a track and derived from the average beat duration, showed no noticeable effect on distinguishing genres. It appeared to be a relatively neutral characteristic across the board.

- **Loudness:** Measures the overall volume of a track in decibels. Surprisingly, the data revealed that classical music tends to be louder than other genres, despite its traditionally wide dynamic range.

- **Energy**: Is a perceptual measure (ranging from 0.0 to 1.0) representing the intensity and activity of a track. Energetic tracks often feel fast, loud, and noisy. Our analysis showed a clear contrast, with classical music on the lower end of the energy scale and metal on the higher end—an outcome that aligns with the nature of these genres.

- **Danceability:** Reflects how suitable a track is for dancing, based on a combination of tempo, rhythm stability, beat strength, and overall regularity. The results followed a logical trend: genres like electronic, Latin, and reggae scored higher, while classical music ranked lower on the danceability scale.

Overall, the bar charts provided a more nuanced understanding of how audio features vary by genre. While some features like tempo and popularity did not show strong differentiation, others—especially energy, loudness, and danceability—offered meaningful insights into genre-specific characteristics.

Bias and Limitations

The dataset may have imbalances in class, where one of the danceability categories is overrepresented compared to the others.
This can introduce class bias and as a result model may show higher accuracy in predicting majority class and poorly in minority class.In our model we could see a high number of instances where 'low' class was being wrongly predicted as another class.he features used in the model (such as loudness, energy, tempo) are based on numerical audio features, which may not capture all the nuances of what makes a song danceable.

Another limitation in the dataset is that each genre is represented by a fixed number of tracks—1000 per genre. While this helps to standardize the data across genres, it might not fully capture the diversity or richness of each genre. Genres that have a larger number of tracks available on Spotify could be underrepresented, while smaller genres may not have as many tracks to choose from, which could skew the analysis.

Popularity is a key feature in the dataset, but it's calculated based on an algorithm that relies heavily on the number of plays a track has received and how recent those plays are. While this method works for identifying currently popular tracks, it introduces a few limitations: if a person continuously listens to a specific track, it could skew the popularity score and the algorithm treats duplicate tracks (such as versions from albums, singles, or remixes) as independent entities. This can lead to inflated popularity scores for the same song in different formats.

Conclusion

This project demonstrates the application of machine learning techniques to classify danceability by using the Spotify dataset.
It enables the prediction of how likely a song's danceability is to be categorized as low, medium and high based on various audio features.The visualizations generated through Tableau complement the analysis by exploration of music trends and genre-specific patterns.The combination of machine learning and data visualization thus provides a comprehensive understanding of music danceability trends with the help of audio features.

Works cited

Spotify most streamed songs. Kaggle. *Unveiling Streaming: A Comprehensive Analysis of Spotify's Most Streamed Songs*
https://www.kaggle.com/datasets/abdulszz/spotify-most-streamed-songs
ChatGPT