Sophia Rybnik
November 2024

I. Project Background

a. *Data Science for Music Recommendation Systems*

Music choices are a direct reflection of an individual's mood, emotions, thoughts, and daily routine. For example, one might listen to upbeat music on a run, as opposed to classical tunes as they unwind for the night. This project is an attempt to apply Machine Learning techniques to identify song "moods" and build playlists of songs that exhibit similar attributes. In my research, I explored how K-means clustering can simplify complex Spotify data describing the audio features of different tracks. I can then use this clustering to recommend songs given user input.

b. *Dataset*

The dataset used in this project was sourced from Kaggle and contains over 600,000 Spotify tracks, with the most recent update occurring in April 2021. The tracks span release dates from 1921 to 2020. Two separate datasets were downloaded:

- **Tracks**: Includes audio features and other information (e.g., artist details, track popularity)
- **Artists**: Lists artists, their music genres, and their popularity.

The audio features in the dataset cover a range of attributes. For example, the valence describes the *musical positiveness* of a track — a high valence indicates that a track sounds positive or cheerful, while a low valence track conveys a negative or dark tone. Attributes like danceability and energy capture differences between faster and slower paced music, while instrumentalness captures a track's focus (or lack of) on vocals. A full list of attributes and their respective definitions is available on the Spotify Developer website. These feature definitions will be critical in interpreting and analyzing the resulting clusters. It's worth mentioning that, in addition to the Kaggle dataset, the Spotify API also provides direct access to track data and audio features. However, using the API is beyond the scope of this research.

c. *Project Outline*

The following steps were undertaken to complete this project:

1. Download, clean, and process the data
2. Explore track audio features and metrics
3. Apply K-means clustering algorithm using to categorize tracks into distinct groups
4. Use PCA to visualize and interpret the clusters as "playlists" or "song moods"
5. Build a recommendation algorithm to suggest songs given user input

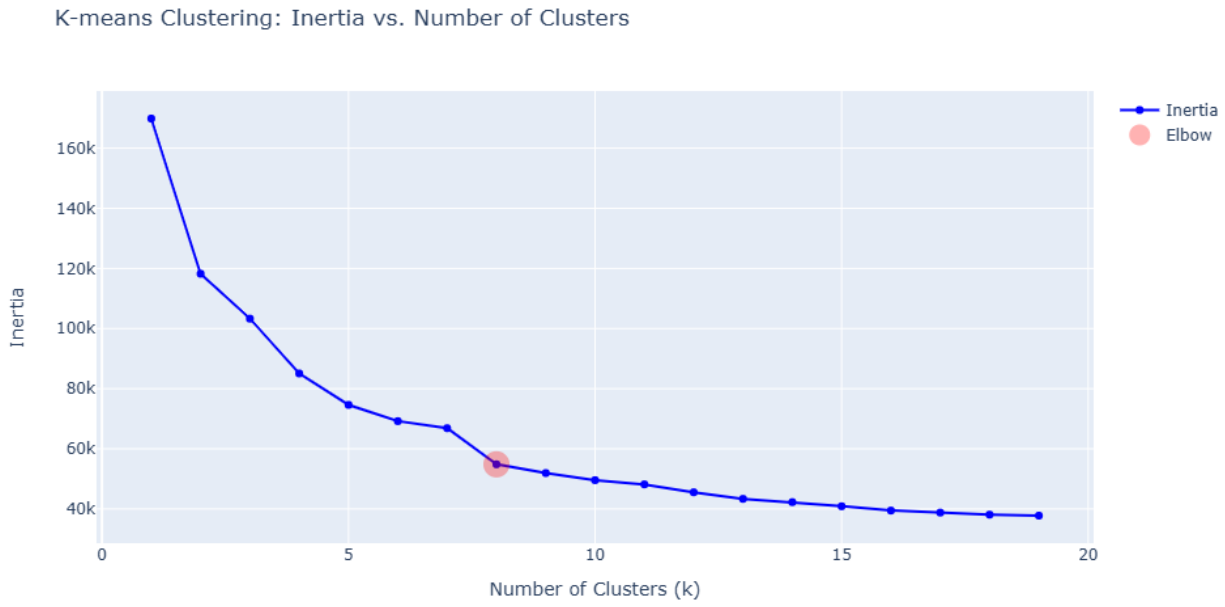
II. Methods

a. *Clustering Algorithm*

The K-means clustering algorithm is a widely used and intuitive machine learning method for classifying unlabeled data. The goal of the algorithm is to identify centroids that meaningfully represent distinct clusters within the data. The process is as follows:

1. **Random Initialization**: Begin by randomly selecting *k* points as the initial centroids.
2. **Assign Points to Nearest Centroid**: Calculate the Euclidean distance between each data point and all centroids, then assign each data point to the closest centroid.
3. **Recompute Centroids**: For each cluster, compute the new centroid as the mean of all the data points assigned to that cluster.
4. **Repeat**: Repeat steps 2 and 3 until the algorithm converges, meaning the assignments of data points to clusters no longer change.

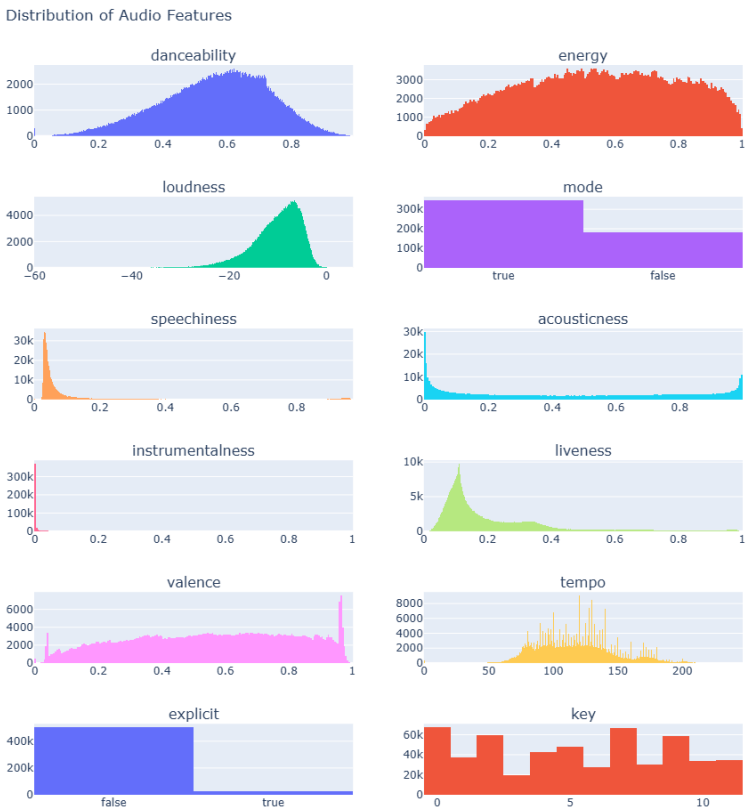
A critical aspect of K-means clustering is selecting the appropriate number of clusters, *k*. There are various methods to determine this value, but for this study, the elbow method was applied. This approach involves running the algorithm with different values of *k* and calculating the inertia) for each. As *k* increases, inertia generally decreases because more clusters can better capture the variation in the data. However, after a certain point, adding more clusters leads to only a small reduction in inertia, indicating that additional clusters may not significantly improve the model and could lead to overfitting [1]. This creates an "elbow" shape in the inertia graph. For the dataset in question, the elbow is observed at k=8, suggesting that 8 clusters is the optimal choice.

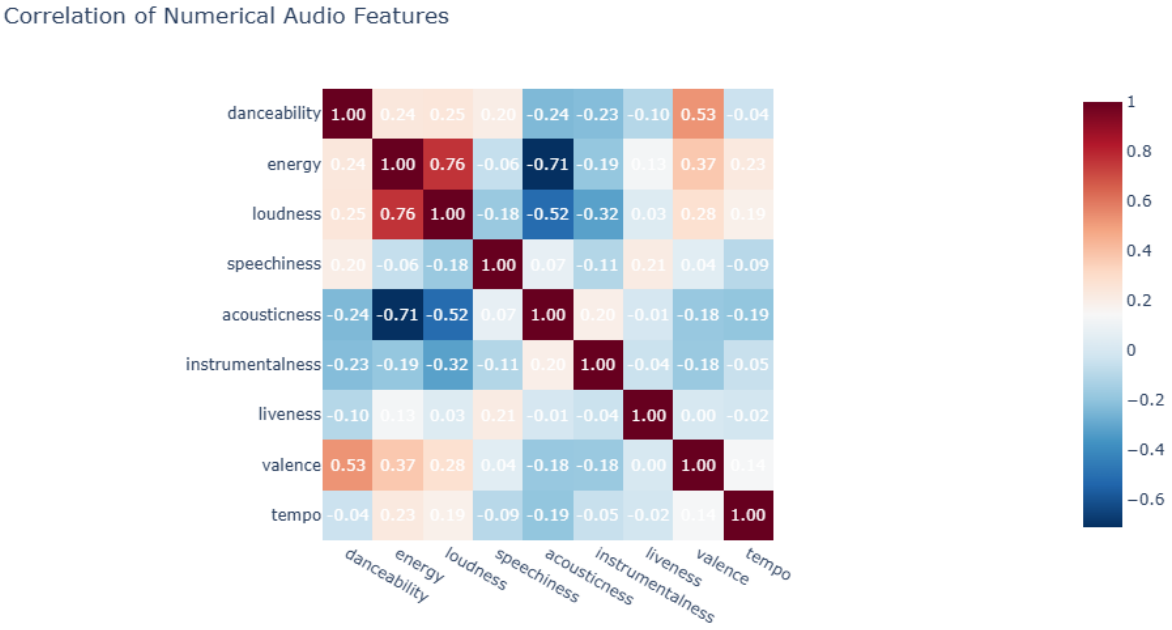


b. Data Considerations

Before applying the clustering algorithm, the data required preprocessing to ensure effective model performance. This included cleaning the data (removing null values, adjusting column types, etc.) and merging the two datasets (tracks and artists). Since a track can have multiple artists, each with distinct genres, the tables had to be merged and engineering thoughtfully. After preparing the dataset, I performed analysis and visualizations to better understand the data (see appendix). There are key considerations kept in mind before applying the clustering algorithm:

- Categorical variables are challenging for K-means since their sample space is discrete while K-means relies on distance-based similarity. As a result, I excluded categorical features (mode, explicit, key, time signature) and focused on numerical ones. Further analysis of the time signature confirmed it didn't add value to clustering (see appendix).
- Some numerical features, like tempo and loudness, required scaling. Features with larger ranges can dominate K-means, skewing results. I applied MinMaxScaler to ensure all features were within the range (0,1) for balanced comparison [2].
- I also analyzed correlations between numerical features. High correlations can lead to overfitting, so I removed the highly correlated "energy" feature (with loudness and acousticness). While there were other significant correlations (e.g., valence-danceability, acousticness-loudness), they weren't strong enough to warrant removal.
- K-means is sensitive to outliers since it uses average calculations to determine centroids. To address this, I applied the Isolation Forest algorithm, which detects anomalies by recursively partitioning data with an ensemble of binary trees. Each data point is scored based on how easily it is separated, and points with abnormal scores are flagged as anomalies [3]. In a personalized dataset, outliers might be less of an issue, since it comes down to personal taste or preference. However, with a generalized dataset, it was important to handle outliers from the outset for effective clustering.





c. Visualization

Principal Component Analysis (PCA) is a technique for reducing the dimensionality of datasets with many features, improving interpretability by creating new uncorrelated variables. These components are linear combinations of the original features that maximize variance, while minimizing information loss. The principal components are derived by solving for the eigenvalues and eigenvectors of a covariance matrix [4]. If S is the sample covariance matrix of the features in a given dataset, then the principal components can be found by solving

$$Sa = \lambda a = 0$$

The largest eigenvalue (λ_1) and its corresponding eigenvector (a_1) capture the greatest variance. While PCA is often used before clustering for dimensionality reduction, it was not necessary in this project due to the large number of tracks compared to the number of features. Instead, PCA was used for the purposes of visualizing the multi-dimensional clusters.

d. Evaluation Strategy

K-Means is an unsupervised algorithm, so in contrast to supervised models you cannot measure the performance with given labels. Alternative evaluation strategies include:

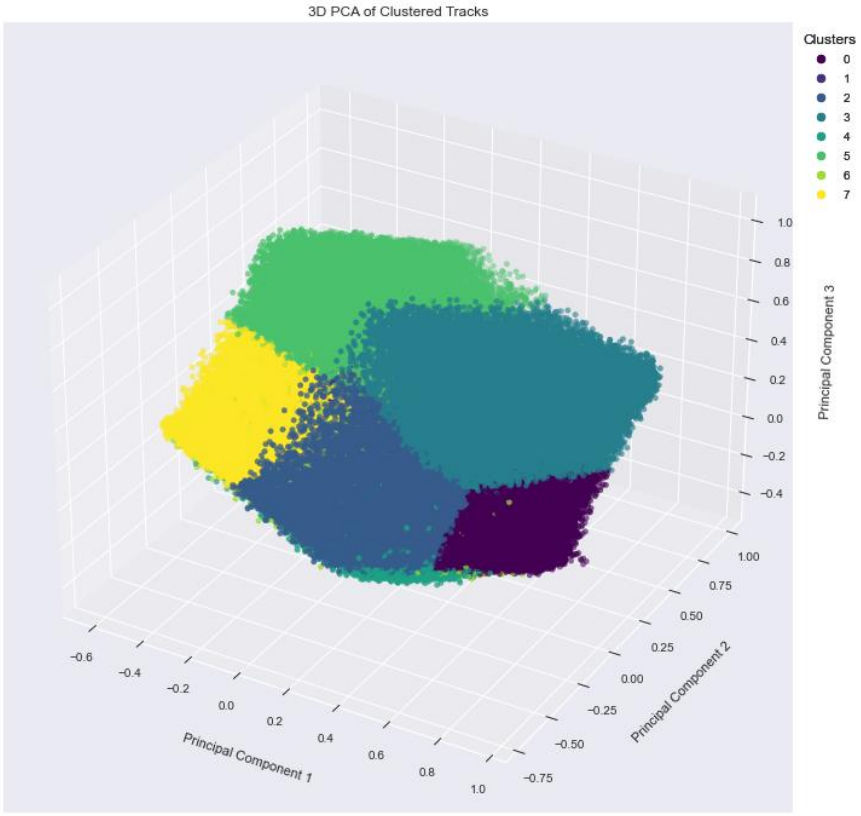
- Visually: After applying PCA, the goal is to observe distinct clusters, where the intra-cluster distances are minimized, and the inter-cluster distances are maximized.
- Silhouette score: A metric that measures overlap between clusters. A score close to 1 indicates well-separated, distinct clusters, while a score close to -1 suggests poor clustering.

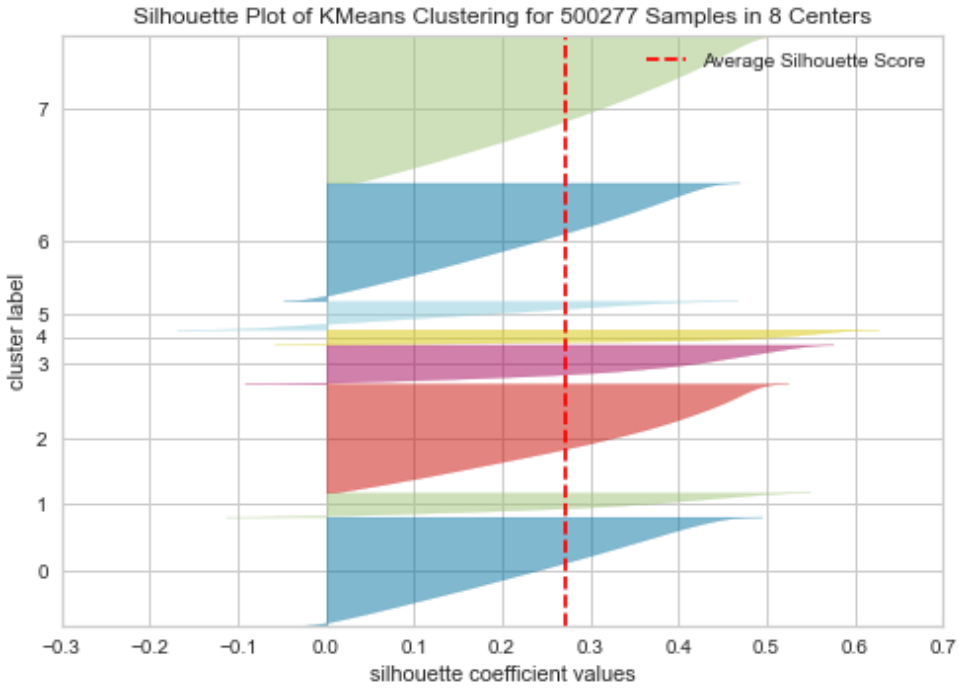
III. Results

a. Cluster Performance

Sophia Rybnik
November 2024

The K-means clustering algorithm is applied to segment the tracks in this dataset into 8 groups, using the numerical audio features of each track. To aid in visualization, PCA is applied to obtain 3D data. Although the inertia is relatively high and the clusters are not perfectly distinct, they are still identifiable as separate groups.

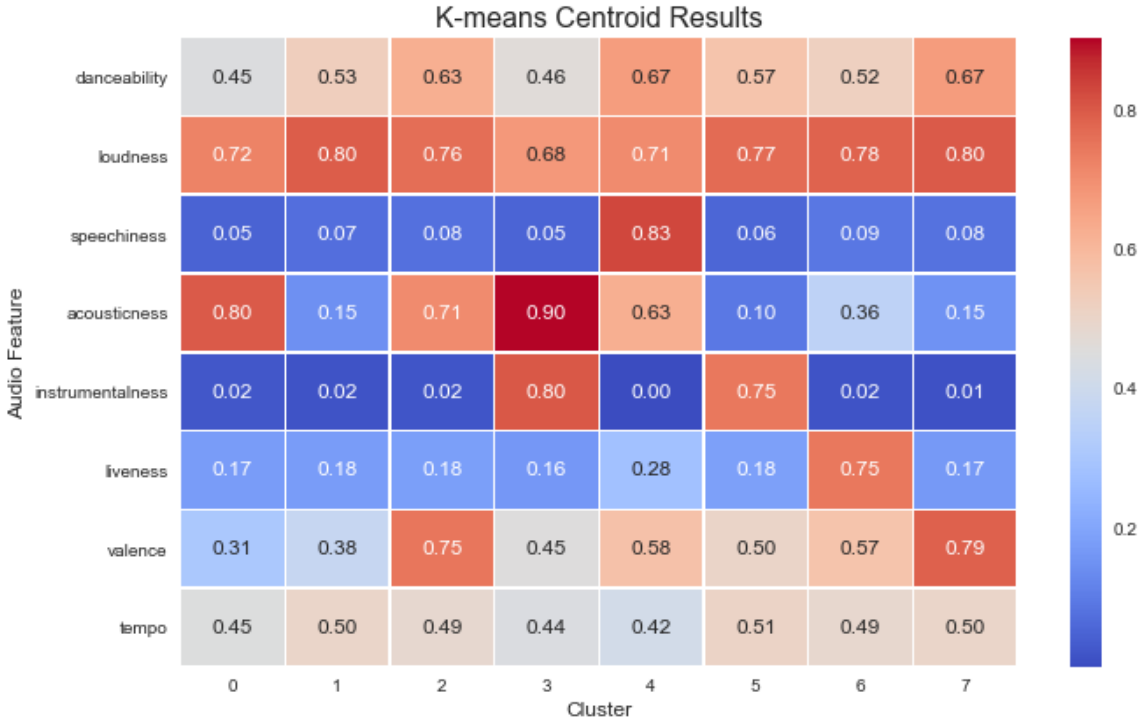




The silhouette score, which ranges from -1 to 1, measures the quality of clustering. A score above 0.5 typically indicates high-quality clusters, while a score below 0.5 suggests lower-quality clusters. Given the complexity and vast scope of the tracks data, it is expected that the clustering won't achieve a silhouette score near 1. However, it is clear that the algorithm with 8 clusters captures the underlying structure of the data reasonably well.

b. Playlists Generated

After analyzing the performance of the clustering model, I took a deeper look at the cluster's audio feature statistics to try and understand the mood of the playlists formed.



There is a clear distinction between the characteristics of the clusters. For instance, cluster 4 exhibits very low instrumentalness, which indicates that the songs in this cluster are likely to be rap (instrumentalness is the likelihood that audio contains no vocal content/spoken word, according to Spotify). On the other hand, cluster 0 consists of acoustic songs with relatively low valence, suggesting that the songs in this cluster have a more somber and melancholic tone. Below, I have provided a random sample of songs from each playlist to further validate my understanding of these audio features.

Playlist 1:

name	artists	label
drivers license	['Olivia Rodrigo']	0
you broke me first	['Tate McRae']	0
deja vu	['Olivia Rodrigo']	0
lovely (with Khalid)	['Billie Eilish', 'Khalid']	0
Heather	['Conan Gray']	0

Sophia Rybnik
November 2024

Playlist 2:

name	artists	label
Peaches (feat. Daniel Caesar & Giveon)	['Justin Bieber', 'Daniel Caesar', 'Giveon']	1
Astronaut In The Ocean	['Masked Wolf']	1
telepatía	['Kali Uchis']	1
Blinding Lights	['The Weeknd']	1
The Business	['Tiësto']	1

Playlist 3:

name	artists	label
Heartbreak Anniversary	['Giveon']	2
positions	['Ariana Grande']	2
What You Know Bout Love	['Pop Smoke']	2
Batom de Cereja - Ao Vivo	['Israel & Rodolffo']	2
Memories	['Maroon 5']	2

Playlist 4:

name	artists	label
everything i wanted	['Billie Eilish']	3
Experience	['Ludovico Einaudi', 'Daniel Hope', 'I Virtuosi Italiani']	3
YKWIM?	['Yot Club']	3
In My Room	['Frank Ocean']	3
Buttercup	['Jack Stauber']	3

Playlist 5:

name	artists	label
Love Yourself	['Justin Bieber']	4
Big Gangsta	['Kevin Gates']	4
CANCIÓN CON YANDEL	['Yandel', 'Bad Bunny']	4
GOSHA	['$NOT']	4
Carry On	['XXXTENTACION']	4

Playlist 6:

name	artists	label
Seven Nation Army	['The White Stripes']	5
Can I Call You Tonight?	['Dayglow']	5
Instant Crush (feat. Julian Casablancas)	['Daft Punk', 'Julian Casablancas']	5
Freaks	['Surf Curse']	5
Around the World	['Daft Punk']	5

Playlist 7:

name	artists	label
Good Days	['SZA']	6
Ella No Es Tuya - Remix	['Rochy RD', 'Myke Towers', 'Nicki Nicole']	6
death bed (coffee for your head)	['Powfu', 'beabadoobee']	6
The Box	['Roddy Ricch']	6
After Party	['Don Toliver']	6

Playlist 8:

name	artists	label
Save Your Tears	['The Weeknd']	7
Leave The Door Open	['Bruno Mars', 'Anderson .Paak', 'Silk Sonic']	7
Bandido	['Myke Towers', 'Juhn']	7
Fiel	['Los Legendarios', 'Wisin', 'Jhay Cortez']	7
Friday (feat. Mufasa & Hypeman) - Dopamine Re-Edit	['Riton', 'Nightcrawlers', 'Mufasa & Hypeman', 'Dopamine']	7

c. *Song Recommendation Algorithm*

Leveraging the K-means clustering results, I built a song recommendation engine that suggests new songs to a user based on their liked songs and the clusters (or playlists) those songs belong to. Here's a breakdown of how it works:

1. **User Preferences:** The user provides a list of songs they like. These songs serve as input for generating recommendations.
2. **Calculate Cluster Weights**: The engine then calculates the weight of each cluster based on how many songs from each cluster the user has liked. A higher weight is assigned to clusters with more liked songs, reflecting the user's preferences.
3. **Generate Recommendations**: The engine selects new songs to recommend by:
    - Filtering out songs the user has already liked.
    - Considering the top 5000 most popular tracks

- Randomly selecting songs from the most liked clusters, giving higher probability to clusters with higher user preference weight.
- Repeating this process until the desired number of recommendations is reached.

For example,

In [181]:
```
# create songlist for user X (since I don't have actual user data: assume as liked songs or playlist)
track_list = ['2h9TDNEXRhcDIV3fsoEVq9', # What Other People Say, ['Sam Fischer', 'Demi Lovato']
              '61KpQadow081I2AsbeLcsb',  # deja vu, ['Olivia Rodrigo']
              '7szuecWAPwGoV1e5vGu8tl',  # In Your Eyes, ['The Weeknd']
              '3UHPGOkUcE4hE7sqBF4Snt',  # Film out, ['BTS']
              '2xLMifQCjDGFmkHkpNLD9h',  # SICKO MODE, ['Travis Scott']
              '5Km4r87BoX2qqtoprYS1gh',  # LOCATION, ['KAROL G', 'Anuel AA', 'J Balvin']
              '0pqnGHJpmpxLKifKRmU6WP',  # Believer, ['Imagine Dragons']
              '4Iedi94TIaB2GGb1nMB68v',  # On Me, ['Lil Baby']
              '1Xi84slp6FryDSCbzq4UCD',  # Arcade, ['Duncan Laurence']
              '7fBv7CLKzipRk6EC6TWHOB',  # The Hills, ['The Weeknd']
             ]
```

In [193]:
```
recommended_songs = generate_recommendations(track_list, tracks_cleaned, num_recommendations=5)
recommended_songs
```

Out[193]:
```
['As Long As You Love Me',
 'Vincent',
 'Narcos',
 'ocean eyes',
 'Hard For The Next (with Future)']
```

Evaluating the performance of recommender systems is challenging because there is no straightforward method for direct evaluation. While I can listen to the recommended songs and express whether I like them, this is not a true evaluation of the system's effectiveness. In practice or at a larger scale, one could conduct a user study.

IV. Conclusions

In conclusion, this research project demonstrates that audio features can be effectively used to organize tracks into playlists that capture a specific mood. While the K-means clustering method provides a straightforward approach, there is potential for further refinement by incorporating additional factors such as song and artist popularity, genre, and lyric sentiment. Additionally, since K-means is a non-deterministic algorithm due to its random initialization of centroids, it would be valuable to train the model multiple times with different seeds to assess the impact of these variations on performance.

The next steps in this project involves developing a more robust song recommendation algorithm based on audio feature similarity rather than random selection. The algorithm should determine the cluster label for each input song using our model, then output a specified number of songs from each cluster, weighted according to the input labels, and ranked by their cosine similarity to the input song. To enhance usability, we should integrate the Spotify Web API, allowing users to search for songs easily and create their personalized song list for more convenient input.

V. References

[1] https://nms.kcl.ac.uk/colin.cooper/teachingmaterial/CSMWAL/CSMWAL/Lectures/ClusterSlides.pdf

[2] https://towardsdatascience.com/a-practical-guide-on-k-means-clustering-ca3bef3c853d

[3] https://medium.com/@limyenwee_19946/unsupervised-outlier-detection-with-isolation-forest-eab398c593b2

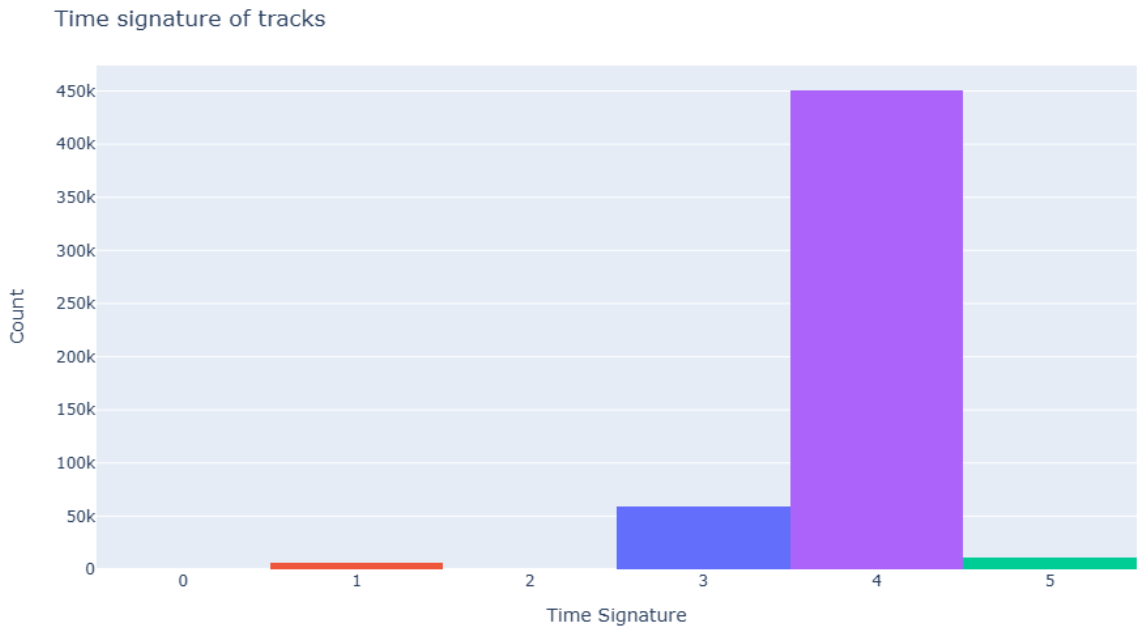
[4] https://royalsocietypublishing.org/doi/10.1098/rsta.2015.0202

Additional sources used:

https://towardsdatascience.com/everything-you-need-to-know-about-min-max-normalization-in-python-b79592732b79

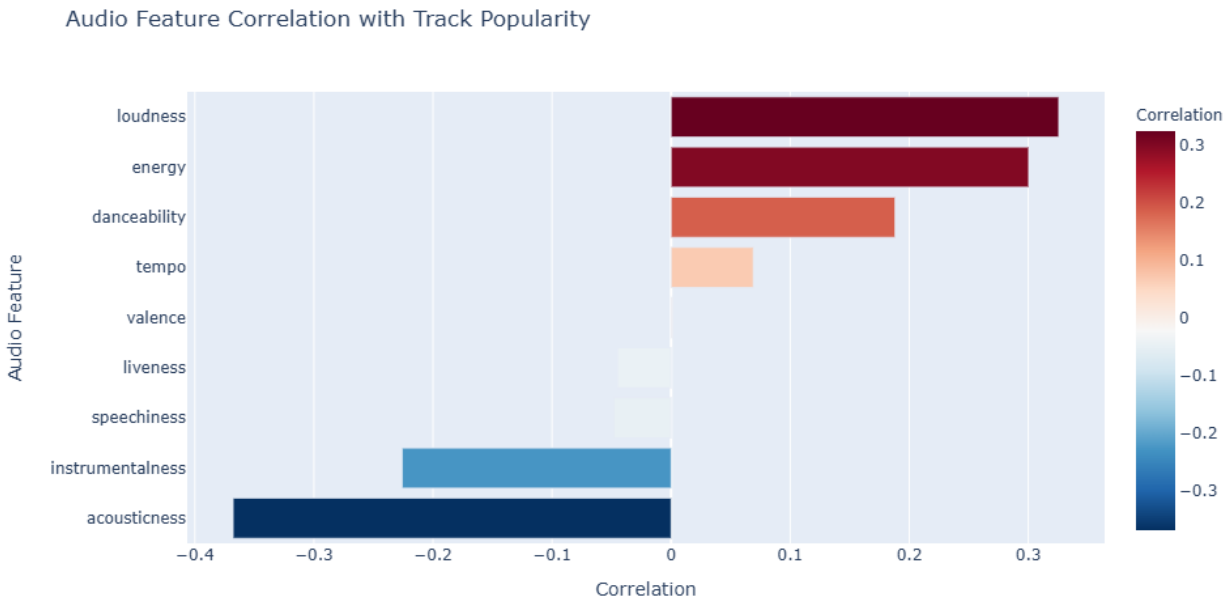
https://medium.com/@datacodedesign/does-removal-of-highly-correlated-features-always-improve-model-performance-8d820d30b71d#:~:text=A%20common%20approach%20to%20this,doesn't%20always%20hold%20true.

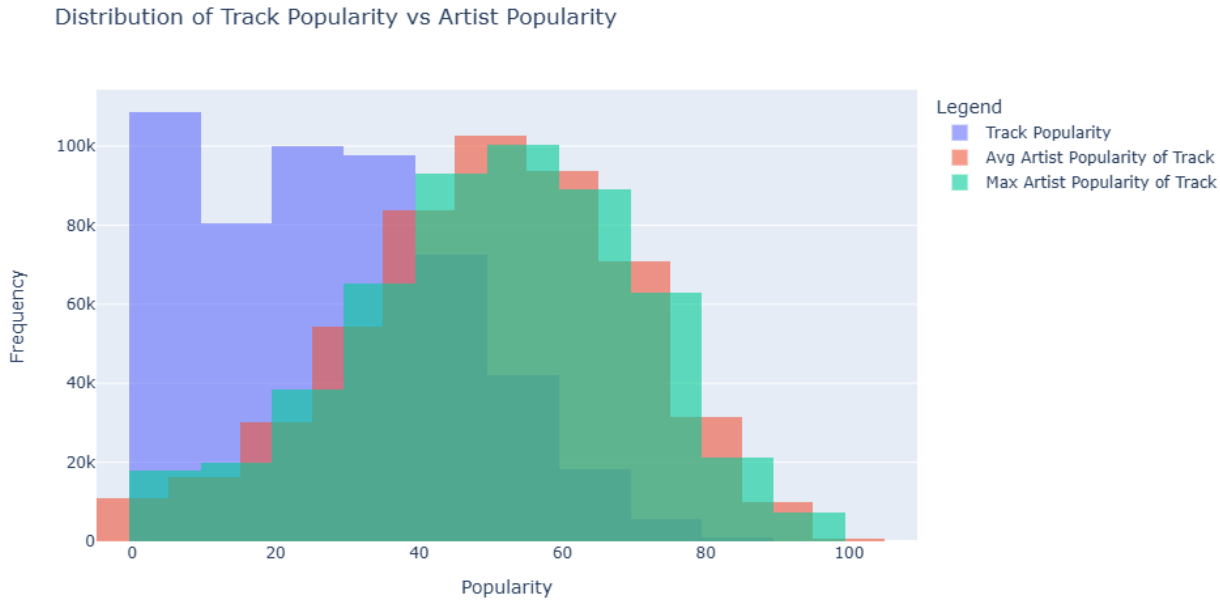
I. Appendix



* Note: According to the Spotify documentation, the Time Signature feature is an estimate of how many beats are in each bar The time signature ranges from 3 to 7 indicating time signatures of "3/4", to "7/4". There is some estimation error or noise in the data, particularly with time signatures that fall into the 0 to 2 range. Additionally, the vast majority of tracks are in a "4/4" time signature. Given the challenges in accurately determining the time signature for tracks labeled as 0, 1, or 2, and the fact that most tracks are

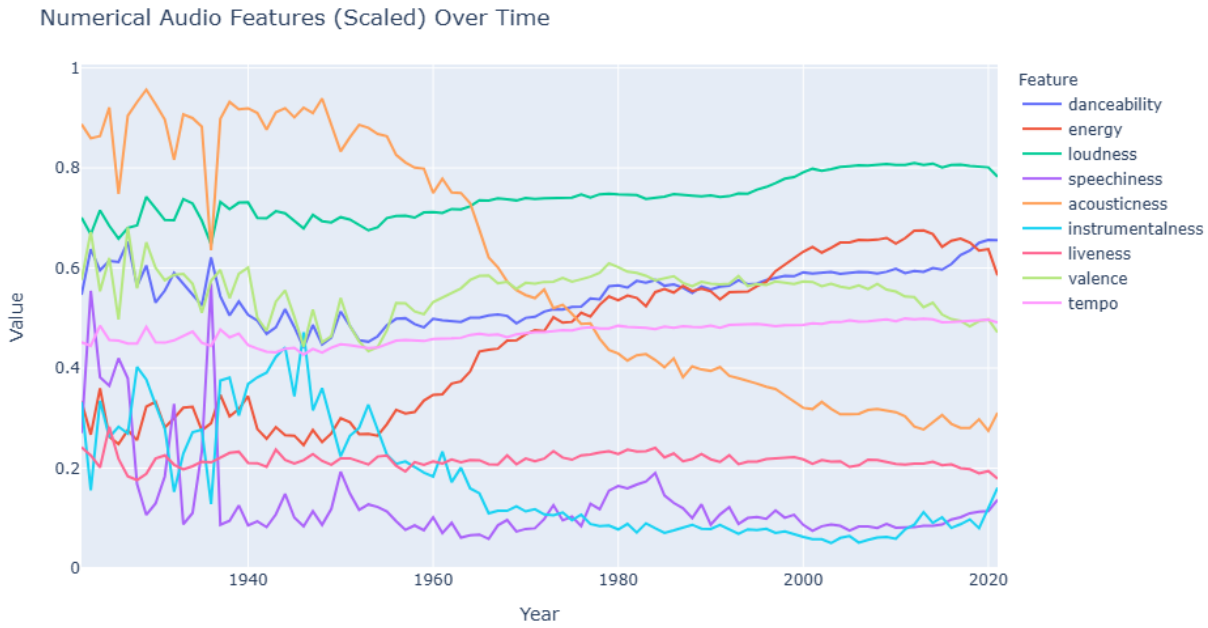
already in "4/4", I decided to exclude this feature. It is unlikely to provide any additional valuable insights.



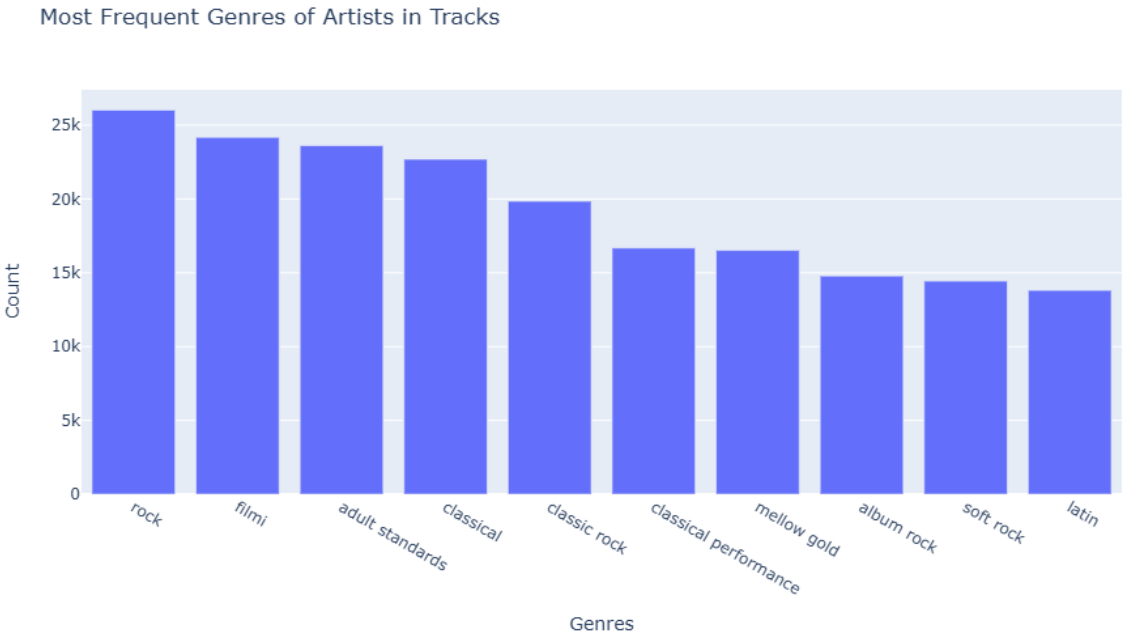


* Note: The track popularity distribution is skewed toward lower values, with while artist popularity is more evenly distributed. Since the measurement scale is from 0 to 100, and it's unclear whether popularity is based on metrics like likes or monthly listeners, this could indicate that few songs are highly popular, distorting the distribution. Without more information on how popularity is measured, it may not be useful to include this feature in a model.

Sophia Rybnik
November 2024



* Note: Many of these features have remained fairly consistent over time. However, there has been a clear decline in the acousticness of songs, coupled with an increase in energy.



* Note: There are over 5000 distinct genre categories, and including them as dummy variables would result in too many columns. Additionally, genres describe the artist rather than the track, so an artist known for pop music might mistakenly have the "country" label applied to all their songs if they release one country track. I will consider the idea of clustering genres as a potential future project expansion.

Sophia Rybnik
November 2024