Task 3 Report — Audio-Based Playlist Clustering

Pavlo Kukurik

This report describes approach to solving Task 3 (optional) from the It-Jim internship challenge. The goal was to automatically group a set of music tracks into playlists based on how they sound, without relying on tags, metadata or genres.

I implemented two versions of the clustering tool:

- Version 1: based on classical audio features
- Version 2: based on deep learning audio embeddings

Both scripts take a folder with .mp3 files and generate a .json file with clustered playlists.

Version 1 — Classical Audio Features

Method

For the first version, I extracted a set of low-level audio features using the librosa library:

- 13 MFCCs (Mel-frequency cepstral coefficients), averaged across the track
- Tempo, estimated from the beat pattern

These features were combined into a 14-dimensional vector for each song. I then standardized the features and clustered the tracks using KMeans with 3 clusters.

Output

The output is saved to playlists_v1.json, where each playlist corresponds to one of the KMeans clusters.

Version 2 — Deep Learning Embeddings

Method

For the second version, I used YAMNet — a pretrained deep audio embedding model published by Google. It maps audio signals to a 1024-dimensional space based on training on a large corpus of labeled audio events.

For each track:

- I resampled the audio to 16 kHz (YAMNet requirement)
- Extracted frame-level embeddings and averaged them
- Applied KMeans clustering on the resulting embeddings

Output

The clustered playlists are saved in playlists_v2.json, using the same structure as in version 1.

Comparison and Observations

	Version 1	Version 2
Feature type	MFCCs + tempo (14-dim)	YAMNet embeddings (1024-dim)
External models	None	TensorFlow Hub (YAMNet)
Processing speed	Fast	Slightly slower
Quality of grouping	Acceptable	More consistent and nuanced

- Version 1 was fast and easy to interpret, but limited in what it could capture.
- Version 2 was more expressive and seemed to form more meaningful clusters based on overall timbre and ambiance.

Conclusion

Both approaches successfully grouped songs into playlists based on sound. The classical method worked well as a lightweight baseline, but the deep embedding approach produced noticeably better groupings. It required more compute but provided more nuanced similarity judgments.