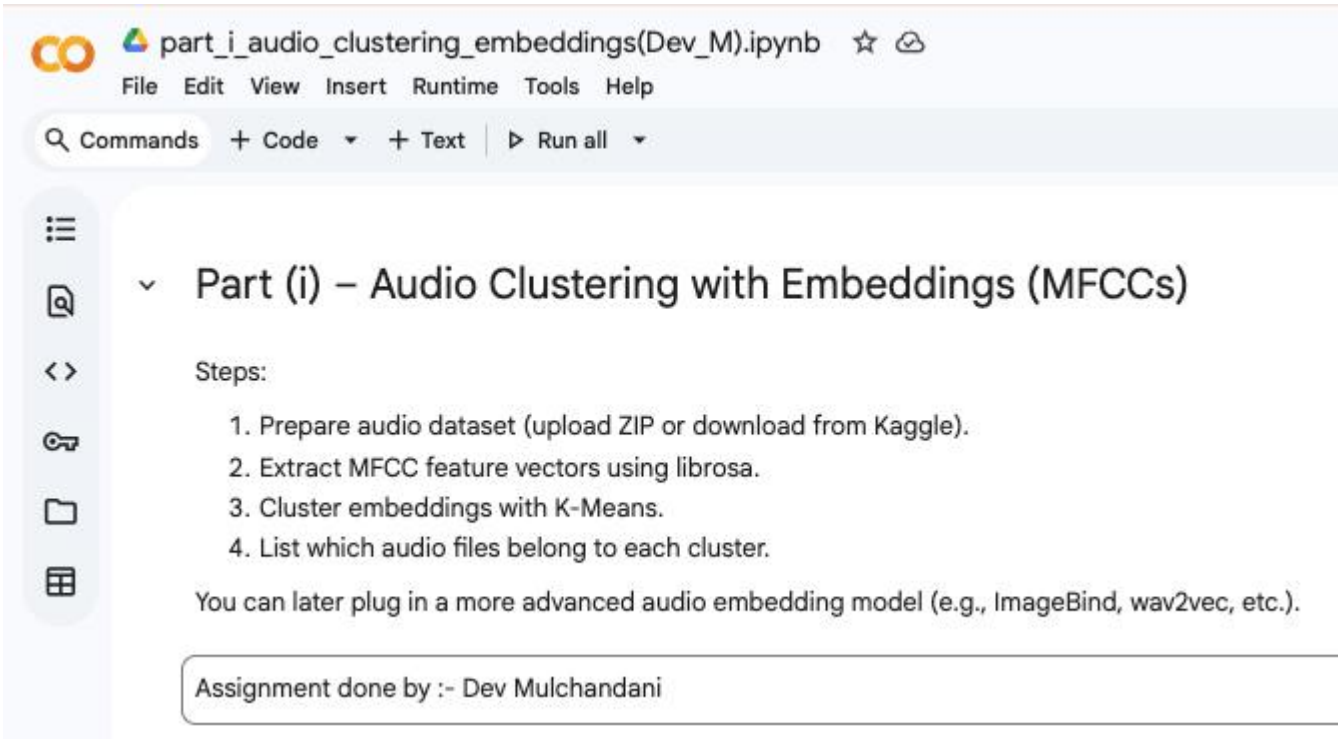
Clustering Assignments

❖ Assignment done by :- Dev Mulchandani

❖ Part-I :–Audio Clustering Using MFCC Embeddings

In Part I, I worked with audio files and extracted meaningful features using Mel-Frequency Cepstral Coefficients (MFCCs), which summarize the shape of the sound spectrum. After computing MFCC-based embeddings for all audio samples, I applied K-Means clustering to group audio clips with similar acoustic characteristics. I evaluated the model using silhouette score and examined sample files from each cluster to understand the types of sounds grouped together. This section highlighted how audio embeddings can be used for sound classification, audio retrieval, and organization of large audio collections.

❖ Screenshots:-



Prepare audio dataset (upload or Kaggle)

[1]
✓ 16s

```python
# @title Prepare audio dataset (upload or Kaggle)
import os, zipfile
from pathlib import Path

try:
    from google.colab import files  # type: ignore
    IN_COLAB = True
except Exception:
    IN_COLAB = False

data_dir = Path("audio")
data_dir.mkdir(exist_ok=True)

print("Option 1: upload a ZIP with .wav / .flac / .ogg / .mp3 files.")
print("Option 2: use Kaggle to download an audio dataset into 'audio/'.")
mode = input("Type 'upload' or 'kaggle' (or 'skip' if already prepared): ").strip().lower()

if mode == "upload":
    if not IN_COLAB:
        raise RuntimeError("Upload only works in Colab.")
    uploaded = files.upload()
    for fn in uploaded:
        if fn.lower().endswith(".zip"):
            with zipfile.ZipFile(fn, "r") as zf:
                zf.extractall(data_dir)
elif mode == "kaggle":
    if IN_COLAB:
        print("Please upload kaggle.json from your Kaggle account.")
        uploaded = files.upload()
        kaggle_path = Path("~/.kaggle").expanduser()
        kaggle_path.mkdir(parents=True, exist_ok=True)
        for fn in uploaded:
            Path(fn).replace(kaggle_path / "kaggle.json")
        os.chmod(kaggle_path / "kaggle.json", 0o600)

    !pip -q install kaggle
    DATASET_SLUG = input("Enter Kaggle dataset slug (e.g. 'mousavi310/esc50'): ").strip()
    !kaggle datasets download -d $DATASET_SLUG -p audio
    for zpath in data_dir.glob("*.zip"):
        with zipfile.ZipFile(zpath, "r") as zf:
            zf.extractall(data_dir)
else:
    print("Skipping dataset download; make sure 'audio/' contains audio files.")
```

```
Option 1: upload a ZIP with .wav / .flac / .ogg / .mp3 files.
Option 2: use Kaggle to download an audio dataset into 'audio/'.
Type 'upload' or 'kaggle' (or 'skip' if already prepared): upload
Choose files  sample_audio_partI.zip
sample_audio_partI.zip(application/zip) - 884882 bytes, last modified: 02/12/2025 - 100% done
Saving sample_audio_partI.zip to sample_audio_partI.zip
```

Extract audio features with librosa (MFCCs)

[2] ✓ 22s

```
# @title Extract audio features with librosa (MFCCs)
!pip -q install librosa

import librosa
import numpy as np
from pathlib import Path

audio_paths = [p for p in Path("audio").rglob("*") if p.suffix.lower() in [".wav", ".flac", ".ogg", ".mp3"]]
print("Found", len(audio_paths), "audio files.")

embeddings = []
used_paths = []
for p in audio_paths:
    try:
        y, sr = librosa.load(p, sr=None, mono=True)
        mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=20)
        emb = mfcc.mean(axis=1)
        embeddings.append(emb)
        used_paths.append(p)
    except Exception as e:
        print("Failed to load", p, ":", e)

if not embeddings:
    raise ValueError("No valid audio files found in 'audio/'.")

embeddings = np.vstack(embeddings)
print("Embeddings shape:", embeddings.shape)
```

```
Found 20 audio files.
Embeddings shape: (20, 20)
```

Cluster audio embeddings

[3] ✓ 2s

```
# @title Cluster audio embeddings
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from collections import defaultdict

k = 5
kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
labels = kmeans.fit_predict(embeddings)

score = silhouette_score(embeddings, labels)
print(f"Silhouette score: {score:.3f}")

clusters = defaultdict(list)
for path, lab in zip(used_paths, labels):
    clusters[lab].append(str(path))

for lab, files_in_cluster in clusters.items():
    print("=" * 70)
    print("Cluster", lab)
    for f in files_in_cluster[:15]:
        print("-", f)
```

```
Silhouette score: 0.513
======================================================================
Cluster 2
- audio/audio_9.wav
- audio/audio_4.wav
- audio/audio_14.wav
======================================================================
Cluster 4
- audio/audio_19.wav
- audio/audio_11.wav
- audio/audio_0.wav
- audio/audio_2.wav
- audio/audio_15.wav
======================================================================
Cluster 3
- audio/audio_17.wav
- audio/audio_12.wav
======================================================================
Cluster 0
- audio/audio_6.wav
- audio/audio_10.wav
- audio/audio_3.wav
- audio/audio_18.wav
======================================================================
Cluster 1
- audio/audio_1.wav
- audio/audio_13.wav
- audio/audio_8.wav
- audio/audio_5.wav
- audio/audio_16.wav
- audio/audio_7.wav
```