AASD 4004
Machine Learning - II

Applied AI Solutions Developer Program

Module 16
Audio Use Cases

Vejey Gandyer

Agenda

- Audio Classification
- Unsupervised Audio Segmentation
- Speaker Diarization
- Speech to Text
- Text to Speech

Audio Classification

Audio Classification

Classify audio clips into different musical genres

WAV files are organized in two folders representing two genres

Audio features to be extracted, and a classifier model to be built on these features

Audio Classification

Steps

Load the WAV files from folders into Numpy arrays

Extract segment-level feature statistics from them

Choose any classifier algorithm (SVM)

Train a SVM model

Apply the model on the data

Plot the predictions (decision boundary to split two genres)

Audio Classification

Load the WAV files from folders into Numpy arrays

```
dirs = ["data/music/classical", "data/music/metal"]
class_names = [os.path.basename(d) for d in dirs]
```

Extract segment-level feature statistics from them

```
features = []
for d in dirs: # get feature matrix for each directory (class)
    f, files, fn = aF.directory_feature_extraction(d, m_win, m_step, s_win, s_step)
    features.append(f)
```

```
f1 = np.array([features[0][:, fn.index('spectral_centroid_mean')],
               features[0][:, fn.index('energy_entropy_mean')]])
f2 = np.array([features[1][:, fn.index('spectral_centroid_mean')],
               features[1][:, fn.index('energy_entropy_mean')]])
```

Audio Classification

Train a SVM model

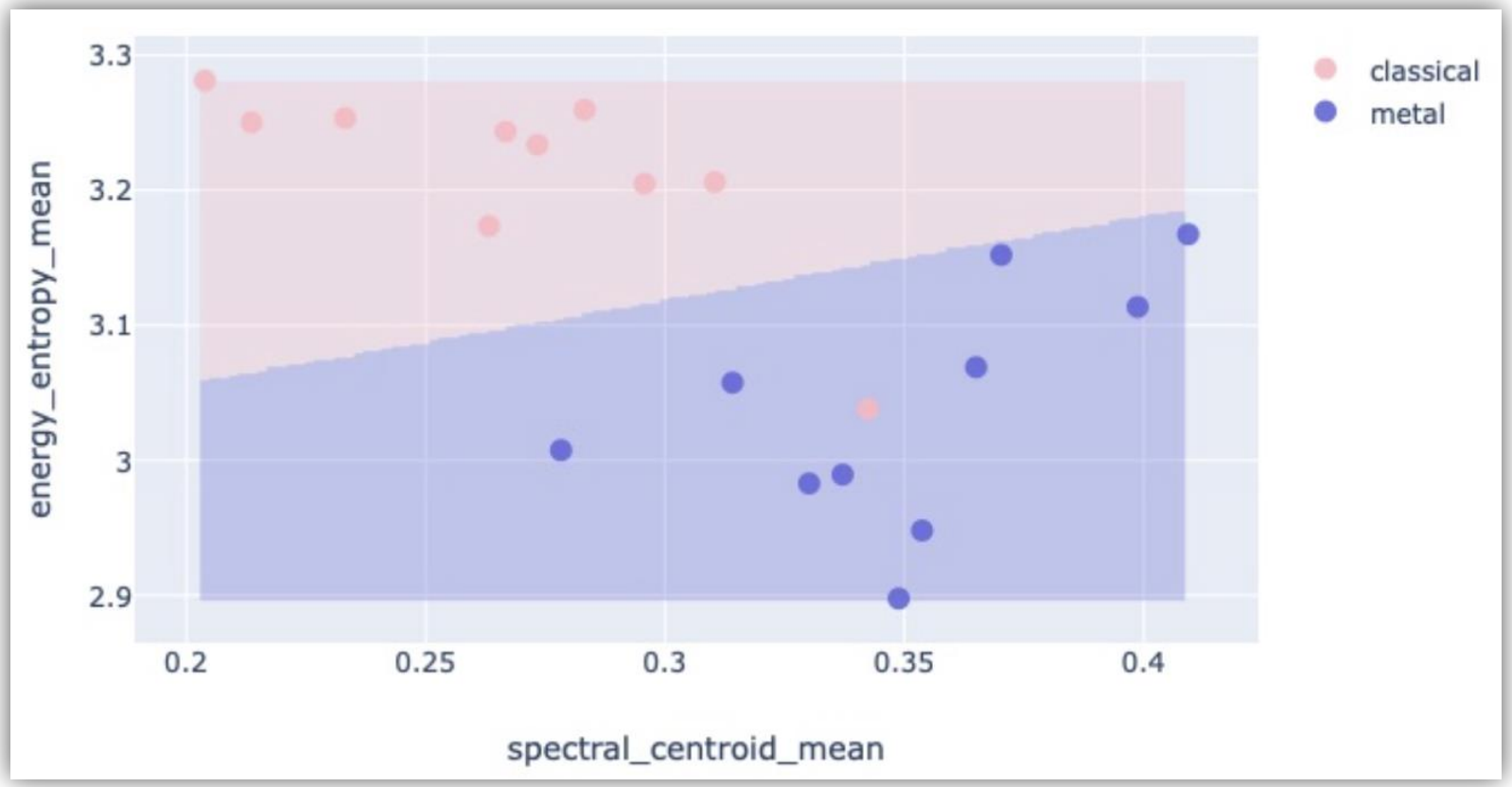
```
cl = SVC(kernel='rbf', C=20)
cl.fit(f, y)
```

Apply the model on the data

```
x_ = np.arange(f[:, 0].min(), f[:, 0].max(), 0.002)
y_ = np.arange(f[:, 1].min(), f[:, 1].max(), 0.002)
xx, yy = np.meshgrid(x_, y_)
Z = cl.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape) / 2
```

Audio Classification

Plot the predictions (decision boundary to split two genres)



Audio Classification

PyAudioAnalysis Wrapper - Feature Extraction + Classifier

```
from pyAudioAnalysis.audioTrainTest import extract_features_and_train
mt, st = 1.0, 0.05
dirs = ["data/music/classical", "data/music/metal"]
extract_features_and_train(dirs, mt, mt, st, st, "svm_rbf", "svm_classical_metal")
```

Audio Classification

Best Classifier with C = 5

	classical			metal			OVERALL	
C	PRE	REC	f1	PRE	REC	f1	ACC	f1
0.001	79.4	81.0	80.2	80.6	79.0	79.8	80.0	80.0
0.010	77.2	78.0	77.6	77.8	77.0	77.4	77.5	77.5
0.500	76.5	75.0	75.8	75.5	77.0	76.2	76.0	76.0
1.000	88.2	75.0	81.1	78.3	90.0	83.7	82.5	82.4
5.000	100.0	83.0	90.7	85.5	100.0	92.2	91.5	91.4
10.000	100.0	78.0	87.6	82.0	100.0	90.1	89.0	88.9
20.000	100.0	75.0	85.7	80.0	100.0	88.9	87.5	87.3

Confusion Matrix:

	cla	met
cla	41.50	8.50
met	0.00	50.00

Selected params: 5.00000

Audio Classification

Predictions on Unseen testing data

```
from pyAudioAnalysis import audioTrainTest as aT
files_to_test = ["data/music/test/classical.00095.au.wav",
                 "data/music/test/metal.00004.au.wav",
                 "data/music/test/rock.00037.au.wav"]
for f in files_to_test:
    print(f'{f}:')
    c, p, p_nam = aT.file_classification(f, "svm_classical_metal","svm_rbf")
    print(f'P({p_nam[0]}={p[0]})')
    print(f'P({p_nam[1]}={p[1]})')
```

```
data/music/test/classical.00095.au.wav:
P(classical=0.6365171558566964)
P(metal=0.3634828441433037)

data/music/test/metal.00004.au.wav:
P(classical=0.1743387880715576)
P(metal=0.8256612119284426)

data/music/test/rock.00037.au.wav:
P(classical=0.27573023692414 49)
P(metal=0.7242697630758552)
```

Audio Segmentation

Audio Segmentation

Assumption: Unknown audio clips always belong to a single label

Real-world recordings are not segments of homogeneous content but sequences of segments of different labels

Audio Segmentation is about segmenting a long audio recording to a sequence of segments that are of homogenous content

- Supervised segmentation
- Unsupervised segmentation

Audio Segmentation

Supervised

Supervised Audio Segmentation

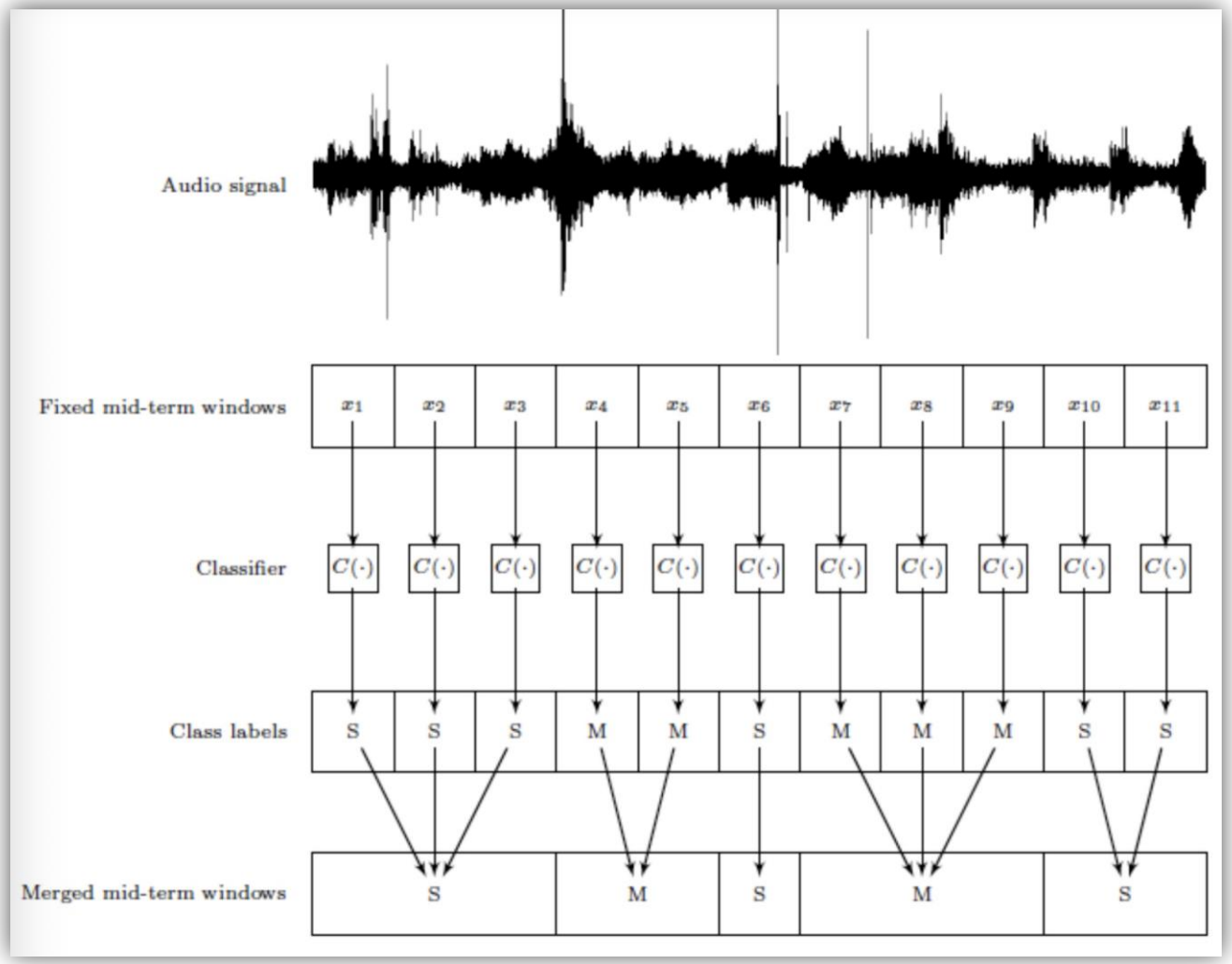
Based on a pretrained segment model that can classify homogenous segments

Pretrained segment model is needed

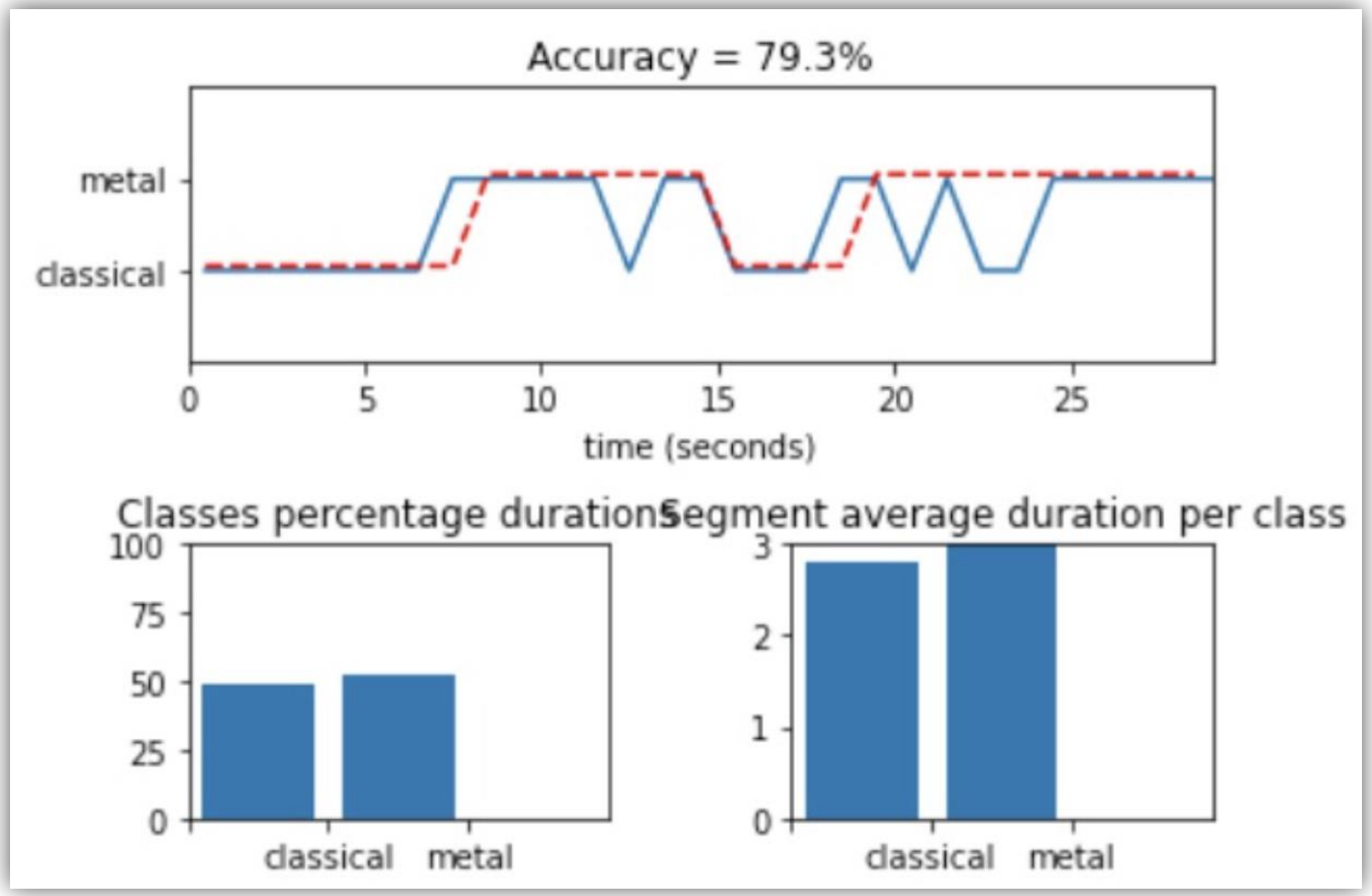
Supervised Audio Segmentation

Steps

1. Split audio clips into non-overlapping fix-sized segments
2. Classify each fix-sized segment using trained model
3. Merge successive segments that contain same class label



Supervised Audio Segmentation



Supervised Audio Segmentatio

Errors are mainly due to:

1. Misclassifications of segment classifier
2. Time resolution issues

```
Segments:
segment 0 0.0 sec - 7.0 sec: classical
segment 1 7.0 sec - 12.0 sec: metal
segment 2 12.0 sec - 13.0 sec: classical
segment 3 13.0 sec - 15.0 sec: metal
segment 4 15.0 sec - 18.0 sec: classical
segment 5 18.0 sec - 20.0 sec: metal
segment 6 20.0 sec - 21.0 sec: classical
segment 7 21.0 sec - 22.0 sec: metal
segment 8 22.0 sec - 24.0 sec: classical
segment 9 24.0 sec - 29.0 sec: metal
```

```
Overall Accuracy: 0.79

Fix-sized segments:
fix-sized segment 0: classical
fix-sized segment 1: classical
fix-sized segment 2: classical
fix-sized segment 3: classical
fix-sized segment 4: classical
fix-sized segment 5: classical
fix-sized segment 6: classical
fix-sized segment 7: metal
fix-sized segment 8: metal
fix-sized segment 9: metal
fix-sized segment 10: metal
fix-sized segment 11: metal
fix-sized segment 12: classical
fix-sized segment 13: metal
fix-sized segment 14: metal
fix-sized segment 15: classical
fix-sized segment 16: classical
fix-sized segment 17: classical
fix-sized segment 18: metal
fix-sized segment 19: metal
fix-sized segment 20: classical
fix-sized segment 21: metal
fix-sized segment 22: classical
fix-sized segment 23: classical
fix-sized segment 24: metal
fix-sized segment 25: metal
fix-sized segment 26: metal
fix-sized segment 27: metal
fix-sized segment 28: metal
fix-sized segment 29: metal
```

Audio Segmentation

Unsupervised

Unsupervised Audio Segmentation

Pretrained segment model is needed for supervised approach

In real-world, no data to train such a pretrained segment model is possible

Can you group song segments so that segments of the same group sound like they belong to the same song part?

Unsupervised Audio Segmentation

Grouping data points --> K-Means Clustering

Clusters of song segments may correspond to stuctural song elements if appropriate audio features are used

Unsupervised Audio Segmentation

Cluster 1

Cluster 2

Cluster 3

Cluster 4

Speaker Diarization

Who speaks when?

Speaker Diarization

Given a long audio sample with different speakers, group audio segments from a same speaker

Almost Unsupervised Audio Segmentation problem

Speaker Diarization

Read the long audio file

```
input_file = "data/diarization_example.wav"
fs, x = read_audio_file(input_file)
```

Extract features

```
[mt_feats, st_feats, _] = mT(x, fs, mt_size * fs, mt_step * fs,
                             round(fs * st_win), round(fs * st_win * 0.5))
(mt_feats_norm, MEAN, STD) = normalize_features([mt_feats.T])
mt_feats_norm = mt_feats_norm[0].T
```

Speaker Diarization

Create clusters on the extracted features

```
n_clusters = 4
x_clusters = [np.zeros((fs, )) for i in range(n_clusters)]
k_means = sklearn.cluster.KMeans(n_clusters=n_clusters)
k_means.fit(mt_feats_norm.T)
cls = k_means.labels_
```

Speaker Diarization

Original Example

Hillary Clinton

Kamalla Harris

Donald Trump

Barrack Obama

Speech To Text (Speech Recognition)

Speech-To-Text (Speech Recognition)

Speech Recognition is a system that translates the language being spoken into digitized text format

Features

Streaming speech to text in real-time: the API is capable of processing real-time audio signals from the device microphone or take an audio file as input and convert it into text also.

Different models based on the domain: you can choose from different trained models depending on the requirements of the project. For example, for converting audio from a telephone, the enhanced phone call model can be used.

Adaptation: you can customize the API to understand rare words, currency, numbers etc by making these as additional classes.

Speech-To-Text (Speech Recognition)

Install the necessary library

```
pip install SpeechRecognition
```

Import the library

```
import speech_recognition as sr
```

Create the Speech-To-Text Recognizer object

```
recognizer = sr.Recognizer()
```

Speech-To-Text (Speech Recognition)

Listen to the microphone and convert speech to text

```
with sr.Microphone() as inputs:
    print("Please speak now")
    listening = recognizer.listen(inputs)
    print(recognizer.recognize_google(listening))
```

Listen to the audio file and convert speech to text

```
with sr.AudioFile('myvoice.wav') as inputs:
    file_audio = recognition.listen(inputs)
    try:
        convert_text = recognition.recognize_google(file_audio)
```

Text to Speech

Text to Speech

Google Text to Speech gTTS API

Features

- Customizable speech-specific sentence tokenizer that allows for unlimited lengths of text to be read, all while keeping proper intonation, abbreviations, decimals
- Customizable text pre-processors which can, for example, provide pronunciation corrections
- Automatic retrieval of supported languages

Text-To-Speech

Install the necessary library

```
pip install gTTS
```

Import the library

```
from gtts import gTTS
```

Create the Text-To-Speech Recognizer object

```
text='Hi ,Welcome to GBC AI Program'
language = 'en'
speech = gTTS(text = text, lang = language, slow = False)
```

Import the library

```
speech.save('medium_1.wav')
```

Further Reading

PyAudioAnalysis

https://github.com/tyiannak/pyAudioAnalysis

Feature Extraction

https://github.com/tyiannak/pyAudioAnalysis/wiki/3.-Feature-Extraction

https://github.com/tyiannak/basic_audio_analysis/blob/master/notebobok.ipynb

https://hackernoon.com/intro-to-audio-analysis-recognizing-sounds-using-machine-learning-qy2r3ufl

Further Reading

PyPDF2

https://pypi.org/project/PyPDF2/

Pyttsx3

https://pyttsx3.readthedocs.io/en/latest/