

# Audio Processing for ML Readiness

Kalp Shah

November 29, 2023

**Aim:** Process audio recordings to prepare them for machine learning analysis.

## 1 Introduction

This project involves recording audio from the microphone, saving it as a WAV file, and extracting key audio features using Python. The primary goal is to demonstrate basic audio processing capabilities using popular libraries such as sounddevice, numpy, pydub, and librosa.

## 2 Tool & Technology

**programming language:**python

**platform:**pycharm

**library:**sounddevice, numpy, pydub, librosa

## 3 Methods

Extract audio file from the devices and convert it into WAV file name as recorded\_audio.wav. After that extract the feature like volume ,pitch score from the audio file.

- **record\_audio Method:** Using library like sounddevice,pydub,numpy the recording of audio from microphone and convert it into WAV file and save in computer memory.

```

Codeium: Refactor | Explain | Generate Docstring
def record_audio(file_path, duration=5, sample_rate=44100):
    print("Recording...")

    # Record audio from the microphone
    audio_data = sd.rec(int(duration * sample_rate), samplerate=sample_rate, channels=1, dtype=np.int16)
    sd.wait()

    # Convert to AudioSegment
    audio_segment = AudioSegment(audio_data.tobytes(), frame_rate=sample_rate, sample_width=2, channels=1)

    # Save the recorded audio
    audio_segment.export(file_path, format="wav")
    print(f"Audio recorded and saved at: {file_path}")

# Specify the file path where you want to save the recorded audio
wav_file_path = "recorded_audio.wav"

# Record audio from the microphone and save to a file
record_audio(wav_file_path, duration=5, sample_rate=44100)

# Play the saved audio file using playsound
print("Playing the recorded audio...")
# playsound(wav_file_path)

```

Figure 1: Code Of Audio Recording

- **extract\_audio\_features Method:** By the help of library like librosa we can extract the feature like volume of audio, average pitch score from the audio.

```

# Define a function to extract audio features
Codeium: Refactor | Explain | Generate Docstring
def extract_audio_features(file_path):
    # Load audio file using librosa
    y, sr = librosa.load(file_path, sr=None)

    # Calculate volume
    volume = np.mean(librosa.feature.rms(y=y))

    # Calculate MFCC features
    mfccs = librosa.feature.mfcc(y, sr=sr, n_mfcc=13)

    # Calculate pitch (fundamental frequency)
    pitch, _ = librosa.core.piptrack(y=y, sr=sr)
    pitch_frequency = np.nanmean(librosa.pitch_tuning(pitch))

    return volume, pitch_frequency

# Specify the path to the audio file
audio_file_path = "recorded_audio.wav"

# Extract features
volume, pitch_frequency = extract_audio_features(audio_file_path)

# Display the extracted features
print("Audio features:")
print(f'Volume: {volume}')
# print(f'MFCCs shape: {mfccs.shape}')
print(f'Average Pitch: {pitch_frequency} Hz')

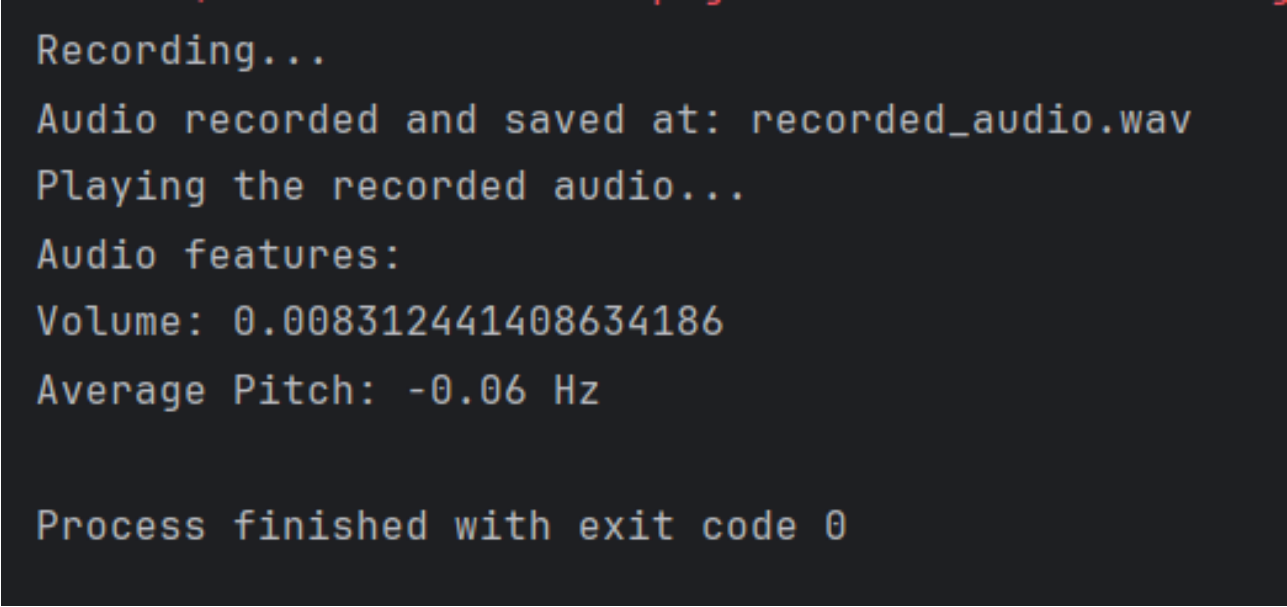
```

Figure 2: Code For Feature Extraction

## 4 Result

Overall code will show the step that it is performing right now after all of that it will show the value of volume and average accuracy.

For our sample,  
volume=0.0083 pitch=-0.06hz

A screenshot of a terminal window with a dark background and light-colored text. The text shows the steps of a program: recording audio, saving it as 'recorded\_audio.wav', playing it back, and then displaying audio features. The features shown are Volume: 0.008312441408634186 and Average Pitch: -0.06 Hz. The process ends with 'Process finished with exit code 0'.

```
Recording...
Audio recorded and saved at: recorded_audio.wav
Playing the recorded audio...
Audio features:
Volume: 0.008312441408634186
Average Pitch: -0.06 Hz

Process finished with exit code 0
```

Figure 3: Output of the code