**Audio Classification**

**Audio Classification** is a machine learning task that involves identifying and tagging audio signals into different classes or categories. The goal of audio classification is to enable machines to automatically recognize and distinguish between different types of audio, such as music, speech, and environmental sounds.

Visuals and sounds are two of the more common things that humans perceive. Both of these senses seem quite trivial for most people to analyze and develop an intuitive understanding of. Similar to how problems related to natural processing (NLP) are straightforward enough for humans to deal with, the same cannot be said for machines, as they have struggled to achieve desirable results in the past. However, with the introduction and rise of deep learning models and architectures over the last decade, we have been able to compute complex computations and projects with a much higher success rate. Audio classification - just like with text - assigns a class label output from the input data. The only difference is instead of text inputs, you have raw audio waveforms. Some practical applications of audio classification include identifying speaker intent, language classification, and even animal species by their sounds. It also has other numerous applications, including speech recognition, music genre classification, environmental sound identification, and even detecting abnormal sounds in industrial machinery. With the rapid advancement of deep learning techniques, audio classification has witnessed significant improvements in accuracy and efficiency.

Audio classification is the process of analyzing and identifying any type of audio, sound, noise, musical notes, or any other similar type of data to classify them accordingly. The audio data that is available to us can occur in numerous forms, such as sound from acoustic devices, musical chords from instruments, human speech, or even naturally occurring sounds like the chirping of birds in the environment. Modern deep learning techniques allow us to achieve state-of-the-art results for tasks and projects related to audio signal processing.

**Artificial Neural Network (ANN)**

Artificial Neural Networks (ANNs), a subset of deep learning algorithms, are inspired by the structure and functioning of the human brain. ANNs consist of interconnected layers of neurons that learn to recognize patterns in data by adjusting the weights of their connections. In audio classification, these networks can be trained to identify features in audio signals that correspond to specific categories.

The raw audio signals are typically preprocessed into meaningful representations, such as Mel spectrograms or Mel Frequency Cepstral Coefficients (MFCCs), to facilitate efficient learning by ANNs. These representations help in capturing the time-frequency characteristics of the audio, which are crucial for distinguishing between different classes.

In this project, an Artificial Neural Network is employed to classify audio signals into their respective categories. The choice of ANN for this task is motivated by its simplicity, flexibility, and capability to model complex, nonlinear relationships in data. While more advanced architectures like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are often employed in audio processing, ANNs offer a foundational and computationally efficient approach suitable for many audio classification tasks.

Sample Rate: Represents the precision of the audio signal in time. Higher sample rates provide better sound quality but take up more data.

Audio Data (Waveform): The raw representation of the sound's amplitude, sampled at the given sample rate.

**Steps for this Projects**

1. **Data Collection**: Download and extract the UrbanSound8K dataset.
   * Obtain the dataset and inspect its structure, including the audio files and metadata.
2. **Exploratory Data Analysis (EDA)**: Analyze the dataset to understand its structure and distribution.
   * Examine the metadata (e.g., number of classes, class distribution, and file paths).
   * Visualize the class distribution to identify any imbalances in the dataset.
   * Analyze individual audio files for duration, sampling rate, and amplitude variations.
   * Plot example audio signals and their spectrograms/MFCCs to visualize patterns.
3. **Data Preprocessing**: Prepare the audio data for model training.
   * Extract features such as MFCCs or Mel spectrograms from the audio files.
   * Normalize the feature values to ensure consistency.
   * Split the dataset into training, validation, and testing sets.
4. **Model Design**: Build the architecture of your Artificial Neural Network (ANN).
   * Include an input layer, hidden layers with appropriate activation functions, and an output layer with softmax activation.
5. **Model Training**: Train the ANN model using the preprocessed features and corresponding labels.
   * Monitor training and validation loss to detect underfitting or overfitting.
6. **Model Evaluation**: Evaluate the trained model on the testing dataset.
   * Use metrics such as accuracy, precision, recall, F1-score, and confusion matrices to measure performance.
7. **Model Optimization**: Improve model performance through:
   * Hyperparameter tuning (e.g., learning rate, batch size, number of layers).
   * Regularization techniques like dropout.
   * Early stopping to prevent overfitting.
8. **Model Deployment**: Save and deploy the trained model.
   * Export the model for future use to classify new or unseen audio files.
9. **Data Collection**

For this project, the dataset used for audio classification tasks is sourced from [https://www.google.com/url?q=https://goo.gl/8hY5ER&sa=D&source=editors&ust=1732719826279105&usg=AOvVaw1sl2UOGXINm4j6hEx8bFHI](https://www.google.com/url?q=https%3A%2F%2Fwww.google.com%2Furl%3Fq%3Dhttps%3A%2F%2Fgoo.gl%2F8hY5ER%26sa%3DD%26source%3Deditors%26ust%3D1732719826279105%26usg%3DAOvVaw1sl2UOGXINm4j6hEx8bFHI). This dataset is well-suited for classifying audio signals into predefined categories, containing diverse and representative samples for training and evaluation.

**Key Features of the Dataset**:

1. **Source**: The dataset is publicly available and accessible via the provided link. It is curated to support audio classification and machine learning tasks.
2. **Content**: It includes 8,732 audio samples categorized into 10 distinct classes. Examples of categories are [list specific categories, e.g., 'Dog Bark', 'Car Horn', 'Siren' etc.].
3. **Format**: The audio files are in **.wav** format.
4. **Annotations**: Each audio file is labeled with its respective category, enabling supervised learning approaches.
5. **Size**: The dataset occupies approximately 5.4 GB of storage.

**Purpose and Relevance**: The dataset provides a balanced and diverse representation of audio categories, ensuring effective training of the Artificial Neural Network (ANN) model. It is ideal for feature extraction techniques such as Mel spectrograms b or MFCCs, which are critical for building a robust classification system.

1. **Exploratory Data Analysis (EDA)**

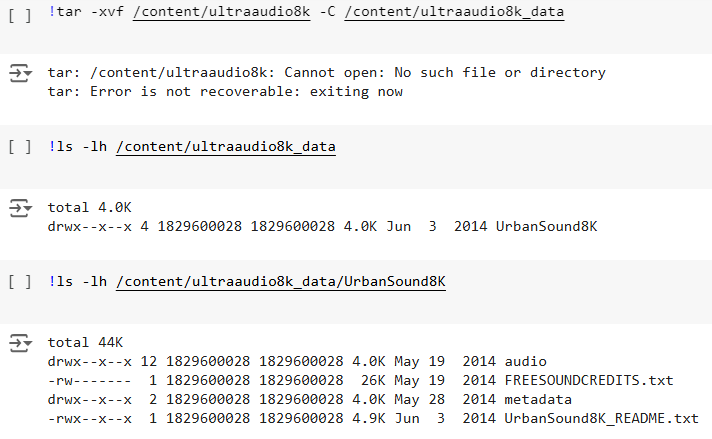
Exploratory Data Analysis (EDA) is the process of analyzing and summarizing datasets to understand their structure, patterns, and key characteristics. In the context of audio classification, EDA helps you explore your dataset before preprocessing and model training. The goal is to uncover insights, identify potential issues (e.g., imbalanced data, noisy samples), and make informed decisions for the preprocessing and modeling phases.

**Why is EDA Important in Audio Classification?**

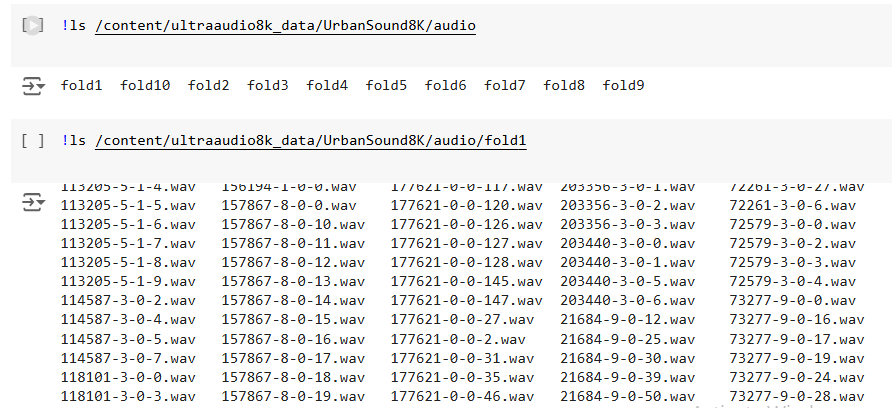
* **Understand the Data**: Ensures you are aware of data characteristics before starting model training.
* **Identify Challenges**: Highlights issues like imbalance, noisy data, or inconsistent features.
* **Guide Preprocessing**: Helps design effective preprocessing steps tailored to the dataset.
* **Improve Model Results**: A well-understood dataset leads to better feature extraction and more accurate classification.

**These are some steps we perform in EDA**

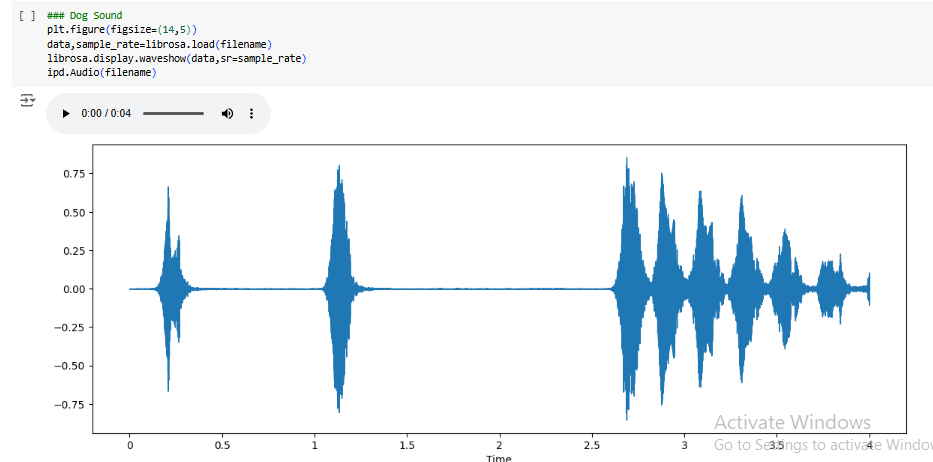
The first step involved extracting the dataset from a compressed .tar file into the designated directory (/content/ultraaudio8k\_data). After extraction, we listed the files and directories to verify the dataset structure. This is useful to ensure the correct extraction and to understand where the audio files and metadata are located.



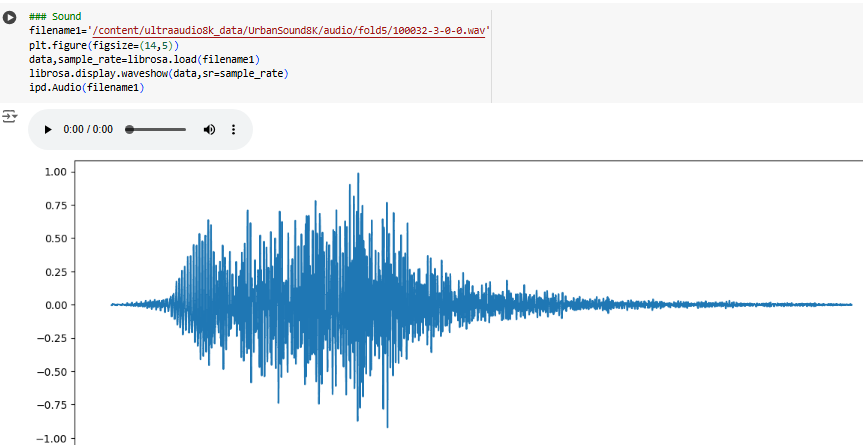
Here, we listed the contents of the audio directory and a specific subfolder (fold1) to get an overview of the audio file organization. The **UrbanSound8K dataset** is divided into multiple folds (each containing a subset of audio files), and this inspection confirms the file structure before proceeding with further analysis.



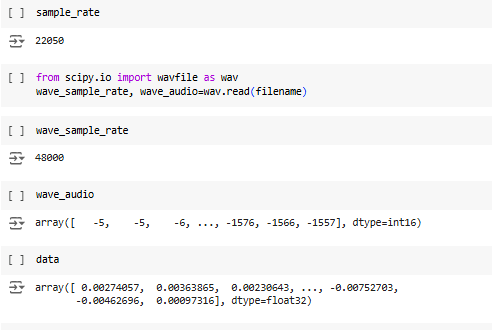
We selected an audio file from the dataset and visualized its waveform using librosa.display.waveshow. This step allows us to visually inspect the raw audio data and observe its amplitude variations over time. We also play the audio to better understand its content. This process is repeated for multiple audio files to inspect different sound types (e.g., dog barking, sirens).



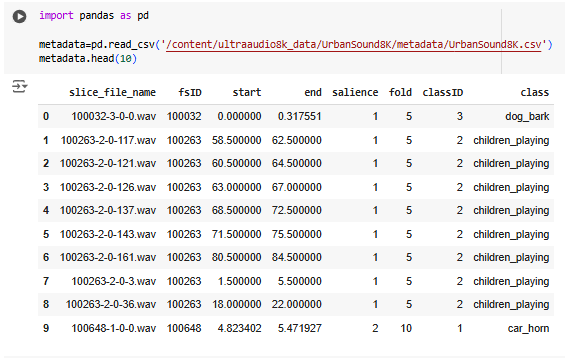
We examined additional audio files from different folds to visualize and compare their waveforms. This helps to identify common patterns in the audio signals, such as noise, silences, or characteristic patterns for specific sound categories.

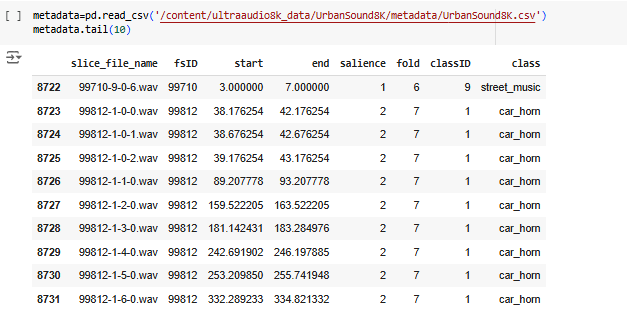


The sample rate of an audio file refers to the number of samples per second in the recording. We checked the sample rate using scipy.io.wavfile.read to ensure consistency across files. The analysis also confirms that the audio data is correctly loaded, and we can extract both the sample rate and the raw audio data for further processing.

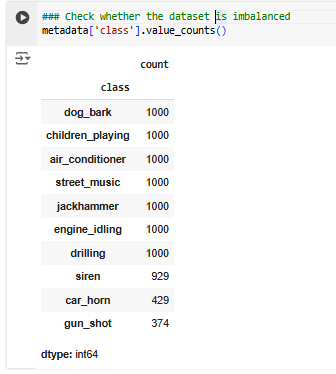


The metadata file provides important information about each audio file, such as its class label, fold number, and file name. By inspecting the first and last few entries of the metadata, we gain an understanding of the dataset's organization and how labels are assigned to the audio files.





To assess the balance of the dataset, we analyzed the distribution of audio samples across the different classes using value\_counts(). This step identifies if some classes are overrepresented (e.g., one class having many more samples than others) or underrepresented, which could impact the performance of the classification model. If the dataset is imbalanced, strategies like data augmentation or class balancing may be needed.



**Data Preprocessing**

**Libraries Used**:

librosa is a Python library used for analyzing and processing audio data.

Scipy is an open-source Python library used for scientific and technical computing, providing tools for optimization, signal processing, statistics, linear algebra, and more, built on top of NumPy.

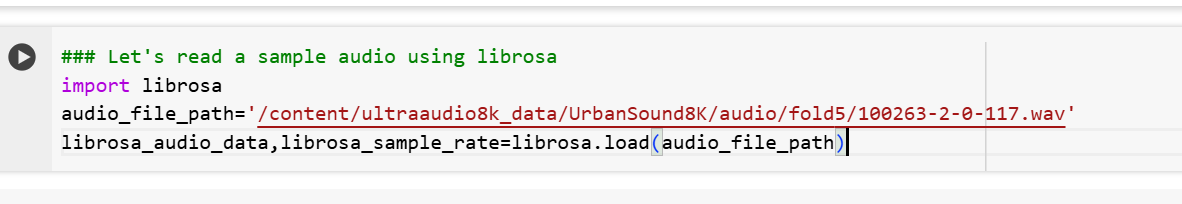
**Resampy used:** Resampy is used in preprocessing to handle audio files with varying sample rates, ensuring consistency across the dataset. Many audio datasets contain files recorded at different sample rates, which can affect feature extraction and model performance. By using resampling, Resampy converts all audio files to a standard sample rate (e.g., 22,050 Hz), making it easier to process the data uniformly. This is especially important when working with tools like Librosa, as consistent sample rates improve the reliability of features like MFCCs and help the model learn

**Audio Loading**:

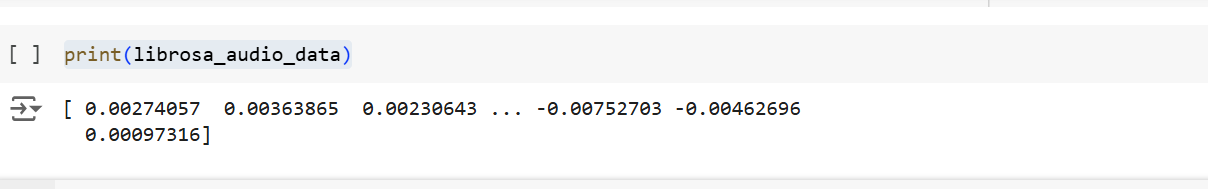
librosa.load() loads the audio file from the specified path.

librosa\_audio\_data: A NumPy array representing the amplitude values of the audio signal.

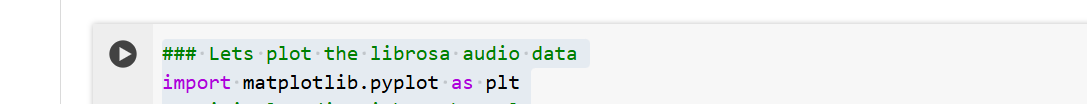
librosa\_sample\_rate: The sampling rate of the audio (default is 22050 Hz if not specified).

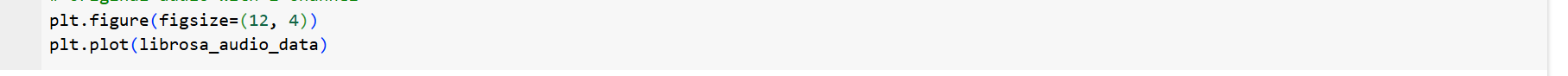


When you print the variable librosa\_audio\_data, it will display the raw audio data as a NumPy array containing amplitude values of the audio signal. These values are typically normalized between -1.0 and 1.0 by default.



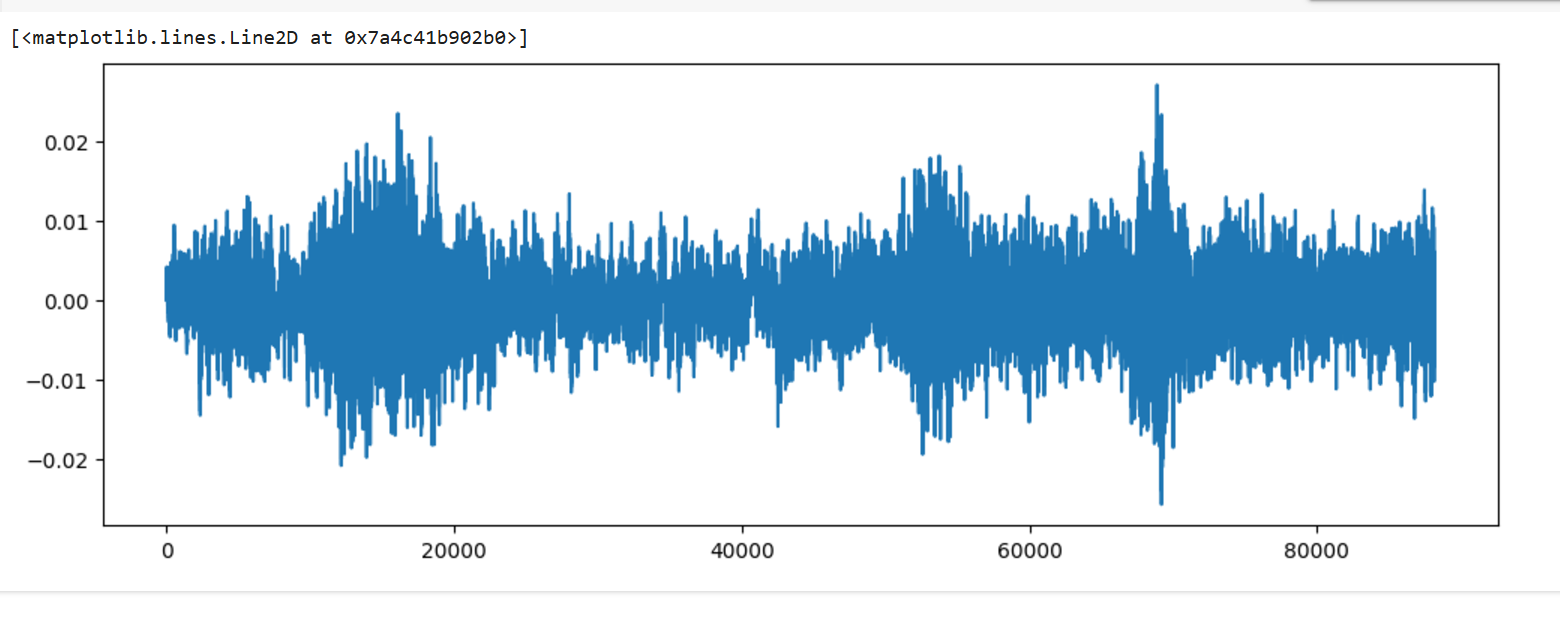
The matplotlib.pyplot module is imported to enable plotting functionality.



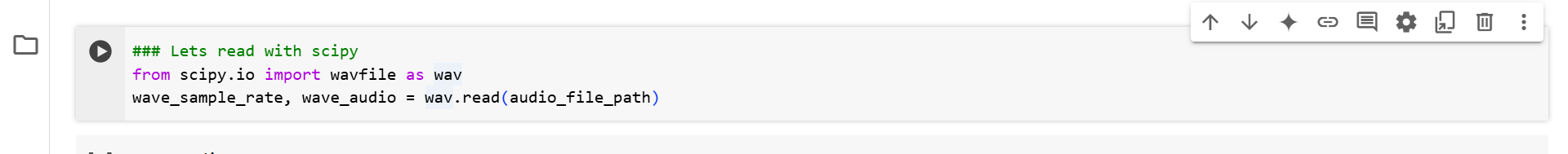
A figure is created with dimensions 12 units wide and 4 units tall to provide a clear and wide view of the audio waveform. 

The waveform of the audio signal stored in librosa\_audio\_data is plotted as a 2D line graph.

The x-axis represents the time in terms of sample indices.

The y-axis represents the amplitude of the audio signal, which indicates the loudness.

The result is a visual representation of the audio signal's waveform. It provides insight into the audio's structure, such as variations in loudness, silences, and the overall shape of the signal.

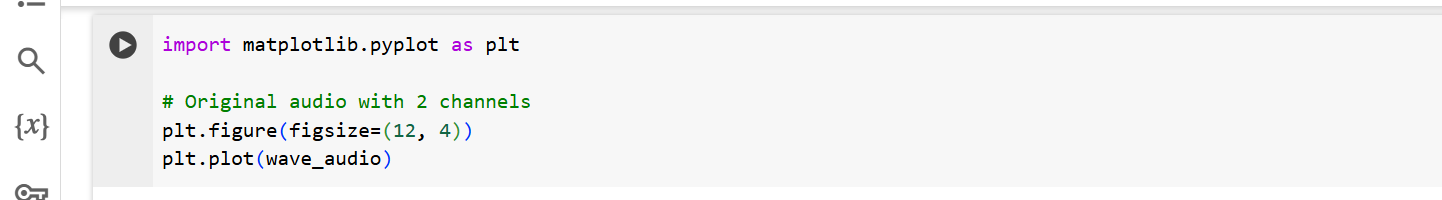


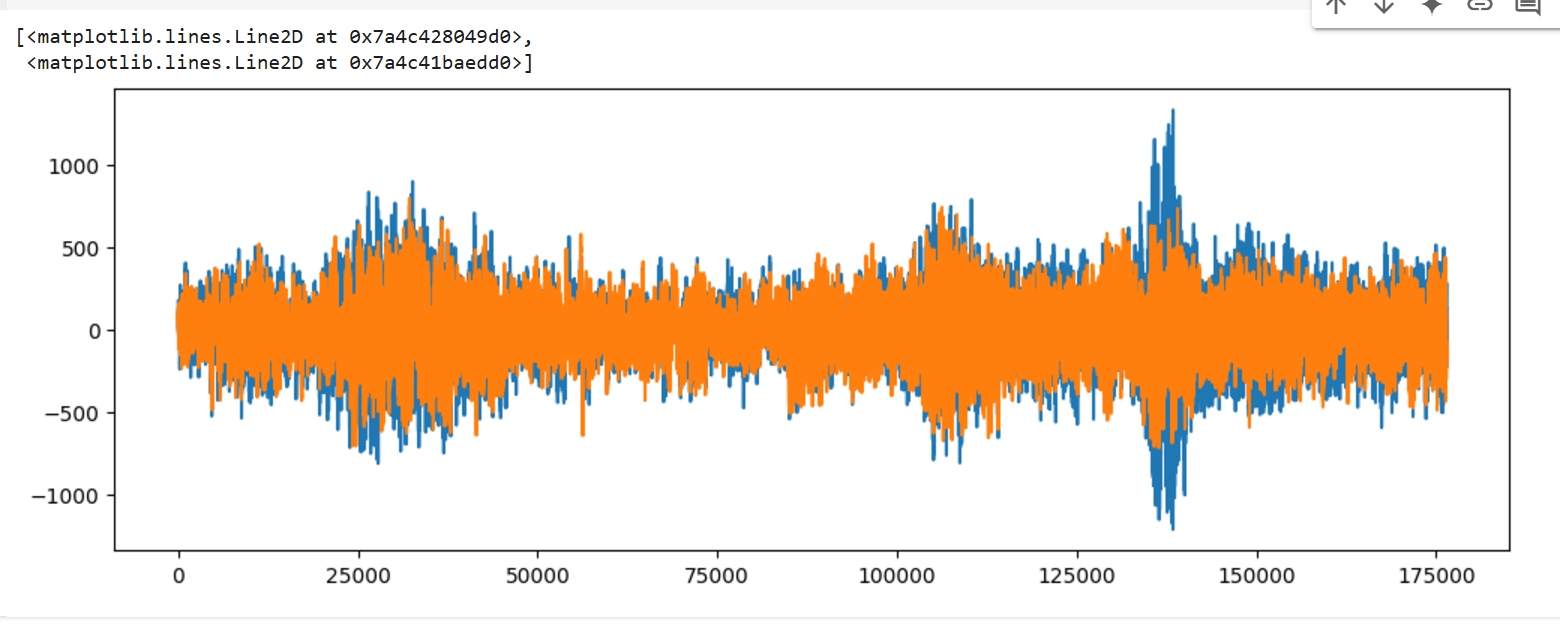
This line imports the wavfile module from the scipy.io library. This module is used to read and write WAV (Waveform Audio File) files. The as wav part creates an alias to simplify the usage of the module in the rest of the code.

wav.read() is used to load the audio data from the file specified by the audio\_file\_path.

wave\_sample\_rate: This variable stores the sample rate of the audio file. The sample rate is the number of samples per second of audio, typically measured in Hertz (Hz).

wave\_audio: This variable contains the actual audio data, which is a numpy array of amplitude values representing the audio waveform. Each value corresponds to a sample in the audio signal, and the array will have the same number of samples as the length of the audio.





The plot generated in the output represents the audio waveform of a stereo sound file. The graph consists of two distinct signals, one for each channel (left and right), with the blue and orange lines representing the waveform data for each channel, respectively.

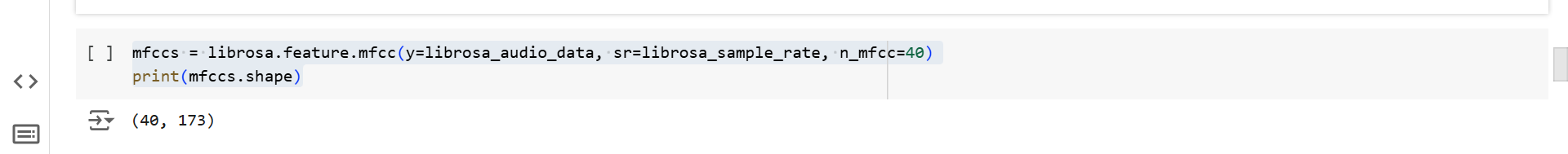
**Key observations from the graph:**

**Stereo Channels**: The audio has two channels, and both are plotted on the same graph. The top portion shows variations in amplitude over time for each channel, with the blue and orange lines representing different audio signals.

**Amplitude Fluctuations**: The large fluctuations in the waveform represent loud sounds, while smaller fluctuations indicate softer sounds. This waveform is useful for visualizing how the sound signal varies in amplitude, showing the overall loudness or intensity of the audio at each point.

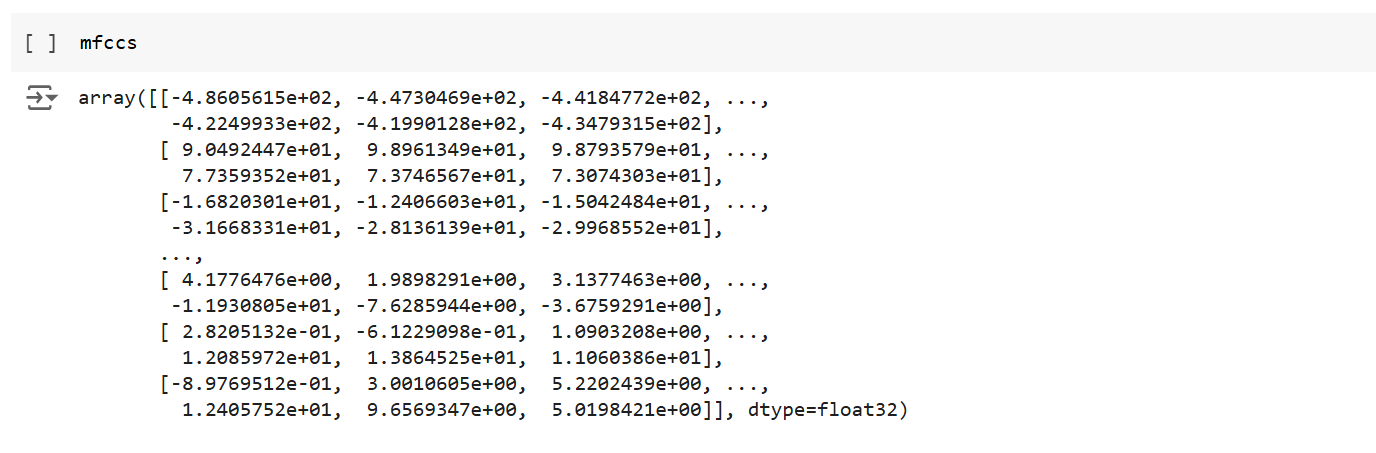
**Time on the X-axis:** The x-axis represents time, with a range of values indicating how long the audio lasts (in samples or frames).

**Amplitude on the Y-axis**: The y-axis shows the amplitude of the audio signal, which varies between positive and negative values, reflecting the pressure variations in the air created by sound waves. Negative values represent a downward displacement, and positive values represent an upward displacement in the waveform.

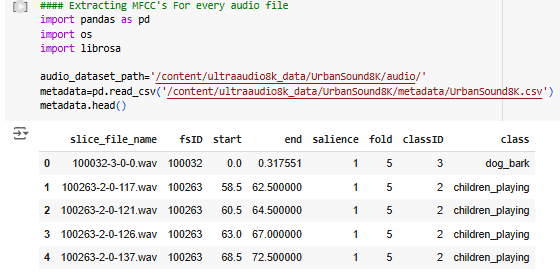


**MFCCs (Mel-Frequency Cepstral Coefficients)** are a representation of audio signals that focus on how humans perceive sound, emphasizing the most relevant frequencies based on the Mel scale. They are derived by analyzing the short-term power spectrum of an audio signal, breaking it into frames, applying a Mel filter bank, and finally computing the cepstral coefficients.

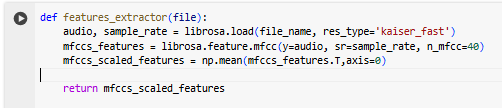
They are used in audio classification projects because they provide a compact, noise-robust representation of the audio, focusing on frequency components that are critical for identifying patterns. MFCCs reduce high-dimensional audio data into a smaller set of features, making them computationally efficient and effective for machine learning. Their ability to capture spectral and temporal dynamics of audio makes them a popular choice for tasks like speech recognition, music genre classification, and emotion detection.



The metadata file (UrbanSound8K.csv) is loaded using pandas, which contains details about the audio files, such as their file paths and labels.

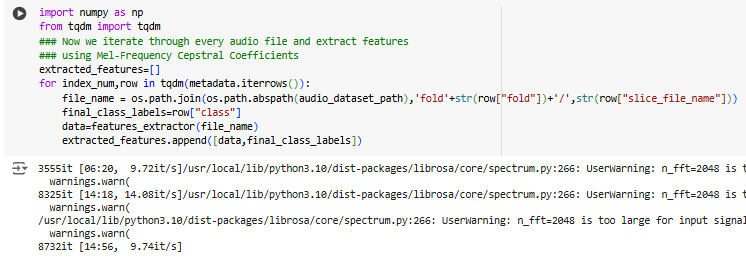


* A function features\_extractor() is defined to process audio files:
  1. **Audio Loading**: Each audio file is loaded using librosa.load(), which converts it into a time-domain signal. The parameter res\_type='kaiser\_fast' is used for efficient resampling.
  2. **MFCC Computation**: MFCCs are computed using librosa.feature.mfcc(), generating 40 coefficients that capture the spectral characteristics of the audio.
  3. **Feature Scaling**: The mean value of MFCCs across all frames is calculated using np.mean(). This results in a compact representation of the audio file.



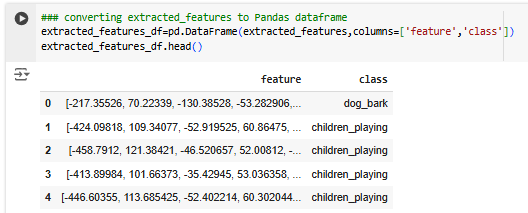
Using a loop with tqdm for progress tracking, the features\_extractor() function is applied to every audio file listed in the metadata:

* The file\_name is constructed dynamically based on metadata information (e.g., fold number and file name).
* The class label for each file is retrieved.
* The extracted MFCC features and corresponding class labels are appended to a list for further processing.



The extracted features are converted into a pandas DataFrame with two columns:

* feature: Contains the MFCC feature arrays.
* class: Contains the corresponding class labels.
* The features (X) and labels (y) are separated into two arrays for model training:
  + X: A NumPy array of the MFCC feature lists.
  + y: The corresponding class labels encoded as one-hot vectors using pd.get\_dummies().



 The dataset is divided into training and testing sets using train\_test\_split():

* X\_train and y\_train: Training data.
* X\_test and y\_test: Testing data.

 The split ratio is set to 80:20, ensuring a randomized yet consistent split with random\_state=0.

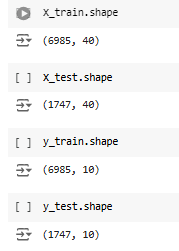


 **X\_train**: Features for training data.

 **X\_test**: Features for testing data.

 **y\_train**: Encoded labels for training.

 **y\_test**: Encoded labels for testing.



**References:**

* <https://www.digitalocean.com/community/tutorials/audio-classification-with-deep-learning>
* <https://towardsdatascience.com/audio-deep-learning-made-simple-sound-classification-step-by-step-cebc936bbe5>  
  <https://www.analyticsvidhya.com/blog/2022/04/guide-to-audio-classification-using-deep-learning/>
* <https://github.com/jeffprosise/Deep-Learning/blob/master/Audio%20Classification%20(CNN).ipynb>
* <https://ieeexplore.ieee.org/document/10258355/>
* <https://medium.com/@oluyaled/audio-classification-using-deep-learning-and-tensorflow-a-step-by-step-guide-5327467ee9ab>
* <https://paperswithcode.com/task/audio-classification>
* <https://jespublication.com/uploads/2024-V15I506.pdf>

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**