Processing and Visualizing Audio Data

**1. Importing Libraries and Modules:**

The code starts by importing necessary libraries and modules for audio processing, data manipulation, visualization, and machine learning.

**2. Loading Audio Data:**

The code loads audio data from a file using a given UUID (Universal Unique Identifier) and extracts the audio signal, sample rate, and other relevant information.

**3. Zero Padding:**

The loaded audio signal is then zero-padded to a target duration. This ensures that all audio signals have the same length for consistency in further processing.

**4. Calculating Spectral Features:**

The Short-Time Fourier Transform (STFT) and Mel-Frequency Cepstral Coefficients (MFCC) are computed from the audio signal. These features provide information about the frequency content and spectral characteristics of the audio.

**5. Calculating Power Spectrum:**

The power spectrum is calculated using Welch's method, which estimates the power spectral density of the audio signal.

**6. Plotting Raw Audio Signal:**

The raw audio signal is plotted over time using Plotly. The x-axis represents time in seconds, and the y-axis represents amplitude.

**7. Plotting STFT Spectrogram:**

The STFT spectrogram, which displays the frequency content over time, is plotted using Librosa and Matplotlib. The x-axis represents time, the y-axis represents log-scaled frequencies, and color represents magnitude.

**8. Plotting MFCC Spectrogram:**

The MFCC spectrogram, which highlights the significant features of the audio signal, is plotted in a similar manner to the STFT spectrogram.

**9. Plotting Power Spectrum Density:**

The power spectrum density is plotted, showing the frequency distribution of the audio signal's power.

**10. Splitting Train-Test Data:**

The dataset is split into training and testing sets, with features extracted from the audio data. The split is stratified to maintain class distribution.

**11. Selecting Features and Labels:**

The features for classification are selected, including various audio characteristics and spectrogram features. The label is the 'STATUS' column, indicating the health status of the audio sample.

**12. Normalizing Data:**

The training and testing features are normalized using Min-Max scaling. The label (status) is encoded using LabelEncoder.

**13. Setting Up Logistic Regression:**

A Logistic Regression model is set up for multiclass classification. The 'multi\_class' parameter indicates multinomial classification, and the model is fitted to the training data.

**14. Evaluating Logistic Regression:**

The logistic regression model is evaluated using accuracy, precision, recall, and confusion matrix on both the training and testing datasets.

**15. Setting Up XGBoost Classifier:**

An XGBoost classifier is configured with specified parameters for multiclass classification. The model is fitted to the training data using the XGBoost library.

**16. Evaluating XGBoost Classifier:**

The XGBoost classifier's performance is evaluated similarly to the logistic regression, including accuracy, precision, recall, and confusion matrix, for both the training and testing datasets.

**17. Plotting Feature Importance:**

The top 10 most important features for the XGBoost classifier are plotted using a horizontal bar chart.

**18. Plotting Loss Evolution:**

The evolution of training and testing log loss is plotted over the iterations (number of trees) for the XGBoost classifier.

**19. Merging Predictions:**

The predictions of both the logistic regression and XGBoost classifiers are merged back into the original dataset using UUIDs for comparison.

These two parts of the code work together to process and analyze audio data, extract meaningful features, train and evaluate machine learning models, and visualize various aspects of the data and model performance.

**Methodology**

In this section, we outline the methodology adopted to process and analyze the audio dataset, extract relevant features, and train machine learning models for classification. The code provided below serves as a step-by-step guide for the entire process.

**Data Preparation and Feature Extraction**

The initial phase of the project involves preparing the audio dataset and extracting relevant audio features. The code begins by loading the required libraries and defining necessary parameters. It then proceeds to import audio data, convert it to a suitable format, and perform zero-padding to achieve a consistent duration. Subsequently, the Short-Time Fourier Transform (STFT) and Mel Frequency Cepstral Coefficients (MFCC) are calculated to extract spectral features from the audio signals. Additionally, the power spectrum density using Welch's method is computed. The extracted features are then visualized through plots showcasing the raw audio signal, STFT, MFCC spectrograms, and power spectrum density.

**Digital Signal Processing (DSP) Workflow**

In the context of this project, digital signal processing plays a pivotal role in transforming raw audio data into meaningful features that are suitable for machine learning classification. The following DSP workflow outlines the steps taken to extract relevant features from the audio signals:

**Audio Data Preprocessing**

The raw audio data is initially loaded and converted into a suitable format for processing. This involves decoding the audio files, typically in the form of WebM format, and converting them into a digital representation that can be manipulated. The digital audio samples are typically represented as discrete values over time.

**Zero Padding**

To ensure consistency in the duration of audio samples, zero padding is applied. Zero padding involves adding zeroes to the audio samples to achieve a desired duration. This step is crucial for maintaining uniformity across audio clips and facilitating further processing.

**Short-Time Fourier Transform (STFT)**

The Short-Time Fourier Transform (STFT) is a fundamental DSP technique used to analyze the frequency content of audio signals over short time intervals. By applying a windowing function and computing the discrete Fourier transform for each window, the STFT provides a representation of how the frequency content of the signal evolves over time. This technique is particularly useful for capturing changes in frequency components that are crucial for distinguishing audio characteristics.

**Mel Frequency Cepstral Coefficients (MFCC)**

MFCC is a widely used technique in DSP for feature extraction from audio signals. It mimics the human auditory system's perception of sound by extracting key features such as timbral and spectral characteristics. The process involves converting the power spectrum obtained from the STFT into a logarithmic Mel frequency scale and then applying Discrete Cosine Transform (DCT) to capture relevant coefficients. These coefficients effectively represent the spectral envelope of the audio signal and are commonly used in audio analysis tasks.

**Power Spectrum Density Estimation**

The Power Spectrum Density (PSD) provides valuable insight into the distribution of signal power across different frequency components. In this project, Welch's method is employed to estimate the PSD. Welch's method involves dividing the signal into overlapping segments, computing the periodograms for each segment, and averaging them to obtain a smooth and accurate estimate of the PSD.

**Visualization**

Visualizing the extracted features aids in understanding the transformed audio data. Plots showcasing the raw audio signal, STFT, MFCC spectrograms, and power spectrum density provide intuitive representations of the audio characteristics. These visualizations offer insights into how different features evolve over time and frequency.

By incorporating these DSP techniques, the project effectively extracts relevant features from raw audio signals, transforming them into a format suitable for machine learning classification. This DSP workflow bridges the gap between raw audio data and meaningful representations that form the basis for health status classification.

**Data Splitting and Preprocessing**

To enable training and testing of machine learning models, the dataset is split into training and testing subsets. This split is stratified based on class distribution to preserve small classes. Relevant features for classification are selected for both the training and testing datasets. Furthermore, the dataset's status labels are encoded using label encoding, and data normalization is performed using Min-Max scaling to avoid data .

**Benchmark Model: Logistic Regression**

A logistic regression model is implemented as a benchmark for classification. The logistic regression classifier is trained on the normalized training data, and its performance is evaluated using various metrics including accuracy, precision, recall, and a confusion matrix.

**Advanced Model: XGBoost Classification**

To achieve improved classification accuracy, an advanced model using XGBoost is developed. The XGBoost classifier is trained using the training dataset and its performance is assessed on both the training and testing datasets. Cross-validation is applied to evaluate the model's consistency across multiple folds. The evolution of the loss function is visualized as the number of estimators increases. The importance of features in the XGBoost model is quantified and visualized.