Guide to the Onboarding task

Motive

Project Echo represents a commitment to leveraging AI/ML tools for bioacoustics analysis, with the ultimate goal of aiding conservationists in monitoring and protecting biodiversity. By focusing on the development of advanced audio classification models, efficient data pipelines, and innovative recording devices, the project addresses real-world ecological challenges through cutting-edge technology.

The onboarding tasks for Project Echo are carefully designed to support the project's objectives while fostering a collaborative and inclusive team environment. These tasks begin with dataset insertion, requiring new members to manually insert the dataset into their local machines. This step encourages exploration of the Project Echo repository, which, in its current state, can be somewhat complex. Through this hands-on interaction, participants familiarize themselves with the repository structure, documentation, and available resources, including the dataset itself.

Subsequent tasks focus on dataset exploration, preprocessing, and feature extraction. These activities are essential for understanding the raw audio data that powers Project Echo's bioacoustics tools. By exploring and processing the dataset, participants develop a deeper understanding of the data's structure and its preparation for machine learning models. Feature extraction further introduces them to techniques for deriving meaningful patterns from audio, directly supporting the project's goal of achieving high model accuracy. Additionally, a team collaboration exercise fosters open communication and cooperation among members, many of whom may not know each other well. This activity sets the tone for a supportive and interactive team environment.

Ideation

The onboarding tasks were developed with the aim of helping new team members integrate into Project Echo while making the process engaging and easy to follow. The tasks were designed to flow naturally, starting with simple activities such as adding and exploring the dataset and gradually progressing to more advanced steps like preprocessing and feature extraction. This incremental approach helps participants build their understanding step by step, ensuring they are not overwhelmed.

To make the experience hands-on and interactive, the onboarding includes small coding exercises that encourage learning by doing. Instant feedback mechanisms are incorporated, providing participants with confidence as they complete each step. All tasks are directly relevant to the project's objectives, such as working with audio data and preparing it for AI models. By tying the tasks to the broader goals of Project Echo, participants gain a clear understanding of how their efforts contribute to conservation efforts.

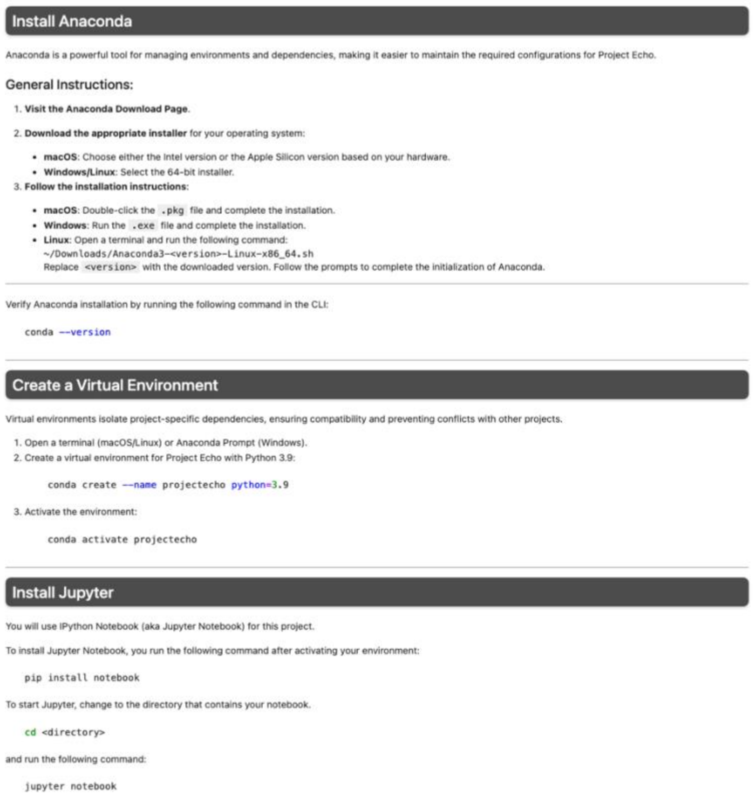
Recognizing the importance of teamwork, the onboarding also incorporates a team-building activity to create a welcoming environment. This exercise emphasizes the value of open communication and shows participants that asking for help is encouraged. The goal is not only to make the onboarding process informative but also enjoyable and inclusive for everyone involved.

Development Environment Setup

The first phase of onboarding focuses on setting up the development environment, a critical step in ensuring participants can contribute effectively to Project Echo. Configuring the environment for a project of this scale can be daunting, with challenges like dependency conflicts and version mismatches. To address these, a detailed, step-by-step setup guide was integrated into the onboarding process. This guide provides comprehensive instructions for installing essential tools like Anaconda, creating virtual environments, and configuring Jupyter Notebooks.

A detailed setup guide significantly reduces the frustrations new members might encounter during the initial setup phase. Common issues such as missing dependencies, version conflicts, and improper configurations can delay progress and demotivate contributors. Providing clear and comprehensive instructions enables new team members to quickly install the required tools and begin contributing without being hindered by technical roadblocks. This proactive approach ensures a smooth and efficient transition into the project.

A standardized development environment is critical for maintaining consistency across team workflows and ensuring reproducibility in scientific tasks. The guide specifies precise versions of tools and libraries, such as Python 3.9, TensorFlow 2.10, and audio processing utilities, eliminating discrepancies that may arise due to variations in individual setups. This uniformity is essential for achieving reliable and reproducible results, particularly in tasks like model training and testing where accuracy is of utmost importance.



Install Anaconda

Anaconda is a powerful tool for managing environments and dependencies, making it easier to maintain the required configurations for Project Echo.

General Instructions:

1. **Visit the Anaconda Download Page**.
2. **Download the appropriate installer** for your operating system:
   - **macOS**: Choose either the Intel version or the Apple Silicon version based on your hardware.
   - **Windows/Linux**: Select the 64-bit installer.
3. **Follow the installation instructions**:
   - **macOS**: Double-click the `.pkg` file and complete the installation.
   - **Windows**: Run the `.exe` file and complete the installation.
   - **Linux**: Open a terminal and run the following command:
     `~/Downloads/Anaconda3-<version>-Linux-x86_64.sh`
     Replace `<version>` with the downloaded version. Follow the prompts to complete the initialization of Anaconda.

Verify Anaconda installation by running the following command in the CLI:

```
conda --version
```

Create a Virtual Environment

Virtual environments isolate project-specific dependencies, ensuring compatibility and preventing conflicts with other projects.

1. Open a terminal (macOS/Linux) or Anaconda Prompt (Windows).
2. Create a virtual environment for Project Echo with Python 3.9:

```
conda create --name projectecho python=3.9
```

3. Activate the environment:

```
conda activate projectecho
```

Install Jupyter

You will use IPython Notebook (aka Jupyter Notebook) for this project.

To install Jupyter Notebook, you run the following command after activating your environment:

```
pip install notebook
```

To start Jupyter, change to the directory that contains your notebook.

```
cd <directory>
```

and run the following command:

```
jupyter notebook
```

Familiarizing Members with Key Tools and Practices

In addition to facilitating technical setup, the guide familiarizes team members with essential tools and best practices used in Project Echo. It introduces libraries such as librosa for audio processing, TensorFlow for machine learning, and Git for version control. Through hands-on exposure, new contributors gain an understanding of how these tools integrate into the project’s broader goals. The guide also emphasizes the importance of practices such as creating virtual environments to isolate dependencies, managing pinned library versions for compatibility, and organizing directories for datasets and scripts. These elements not only streamline the project’s workflows but also equip team members with valuable skills applicable to future endeavours.

Structured to align with the philosophy of gradual learning, the guide breaks down the setup process into manageable steps. New members progress from basic installations to more advanced tasks, such as configuring virtual environments, installing dependencies, and running test scripts. This approach ensures accessibility for all skill levels, fostering confidence and capability in using the tools required for the project.

Dataset Access and Repository Navigation in Project Echo

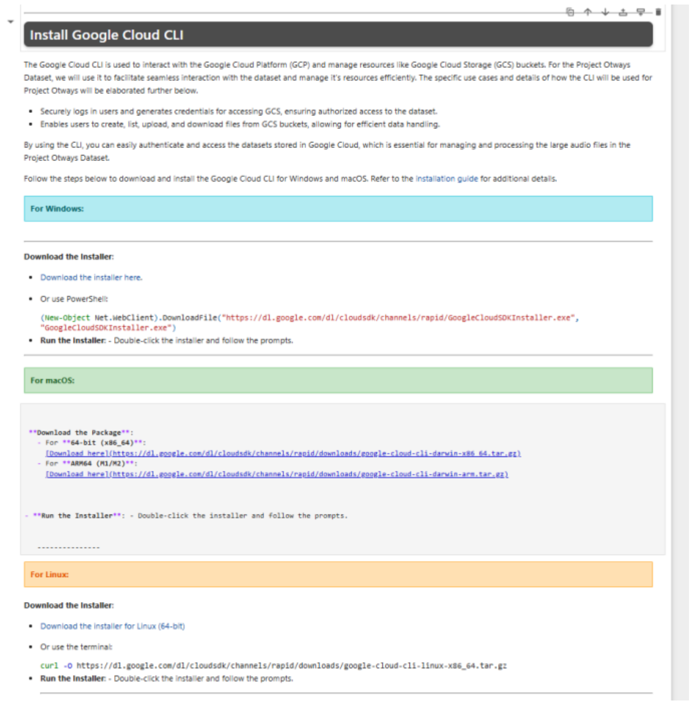
The next phase of onboarding focuses on dataset access and repository navigation. The GitHub repository included the following resources:

- **GoogleCloud_download Notebook**: A Jupyter notebook for accessing and downloading datasets from Google Cloud Storage. Provides details on storage
- buckets, authentication, local directory setup, and data transfers.
- **README.md**: Documentation for the Project Otways Dataset, setup instructions for tools and authentication, dataset download steps.

The original onboarding process lacked both thes GoogleCloud_download notebook and the README.md file, which are critical components of the repository. Without these resources, new users were left without any clear guidance on accessing datasets or understanding the project's structure and purpose. This makes it challenging to set up the environment, navigate the data, or effectively use the available tools. This was not user intuitive at all due to the lack of how to work with the data or understand the purpose of the files in Project Echo. A main goal for our updated onboarding task was to consolidate and improve this information.

Enhanced Documentation for Google Cloud CLI Installation and Authentication

Recognizing that new students may come from diverse technical backgrounds, the guide includes platform-specific instructions for installing and authenticating the Google Cloud CLI on



Install Google Cloud CLI

The Google Cloud CLI is used to interact with the Google Cloud Platform (GCP) and manage resources like Google Cloud Storage (GCS) buckets. For the Project Otways Dataset, we will use it to facilitate seamless interaction with the dataset and manage it's resources efficiently. The specific use cases and details of how the CLI will be used for Project Otways will be elaborated further below.

- Securely logs in users and generates credentials for accessing GCS, ensuring authorized access to the dataset.
- Enables users to create, list, upload, and download files from GCS buckets, allowing for efficient data handling.

By using the CLI, you can easily authenticate and access the datasets stored in Google Cloud, which is essential for managing and processing the large audio files in the Project Otways Dataset.

Follow the steps below to download and install the Google Cloud CLI for Windows and macOS. Refer to the installation guide for additional details.

For Windows:

Download the Installer:

- Download the installer here.
- Or use PowerShell:

```
(New-Object Net.WebClient).DownloadFile("https://dl.google.com/dl/cloudsdk/channels/rapid/GoogleCloudSDKInstaller.exe",
"GoogleCloudSDKInstaller.exe")
```

- **Run the Installer**: - Double-click the installer and follow the prompts.

For macOS:

```
**Download the Package**:
  - For **64-bit (x86_64)**:
    [Download here](https://dl.google.com/dl/cloudsdk/channels/rapid/downloads/google-cloud-cli-darwin-x86_64.tar.gz)
  - For **ARM64 (M1/M2)**:
    [Download here](https://dl.google.com/dl/cloudsdk/channels/rapid/downloads/google-cloud-cli-darwin-arm.tar.gz)



- **Run the Installer**: - Double-click the installer and follow the prompts.


  ---------------
```

For Linux:

Download the Installer:

- Download the installer for Linux (64-bit)
- Or use the terminal:

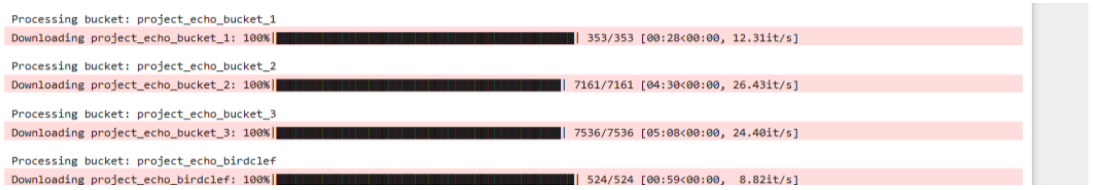
```
curl -O https://dl.google.com/dl/cloudsdk/channels/rapid/downloads/google-cloud-cli-linux-x86_64.tar.gz
```

- **Run the Installer**: - Double-click the installer and follow the prompts.

Linux, macOS, and Windows, significantly improving the setup process by making it more accessible and user-friendly.

Streamlined GCS File Download with Progress Tracking

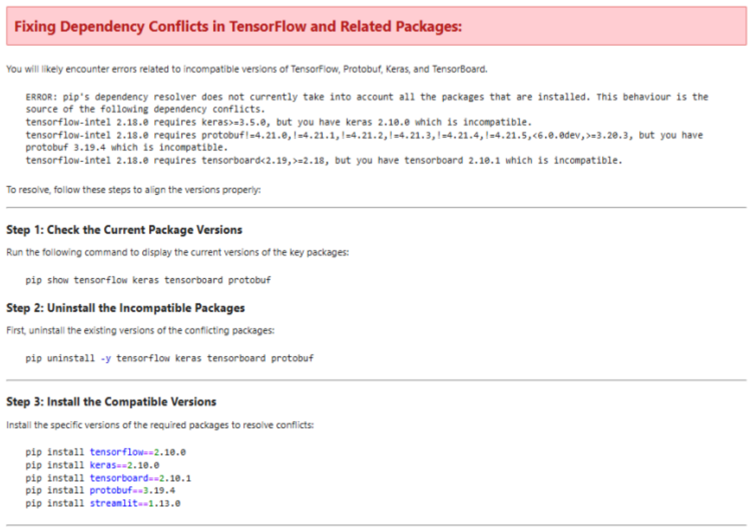
A streamlined GCS file download script was also implemented to organize files into a unified local directory structure. The code downloads files from multiple Google Cloud Storage (GCS) buckets into a unified local directory structure (for easy access and efficiency), organizing nested files and showing a progress bar for each bucket, making it easier to be able to manage and track the download process efficiently.



```
Processing bucket: project_echo_bucket_1
Downloading project_echo_bucket_1: 100%|████████████████████████████████████████| 353/353 [00:28<00:00, 12.31it/s]

Processing bucket: project_echo_bucket_2
Downloading project_echo_bucket_2: 100%|██████████████████████████████████████| 7161/7161 [04:30<00:00, 26.43it/s]

Processing bucket: project_echo_bucket_3
Downloading project_echo_bucket_3: 100%|██████████████████████████████████████| 7536/7536 [05:08<00:00, 24.40it/s]

Processing bucket: project_echo_birdclef
Downloading project_echo_birdclef: 100%|████████████████████████████████████████| 524/524 [00:59<00:00,  8.82it/s]
```

Troubleshooting Guide for Protobuf and TensorFlow Dependency Issues

Troubleshooting common issues, such as TensorFlow-Protobuf dependency conflicts, is also addressed through a dedicated section in the guide, providing step-by-step solutions to significantly improving troubleshooting efficiency for users. This reduces frustrations during setup and ensures a smoother onboarding experience.



Fixing Dependency Conflicts in TensorFlow and Related Packages:

You will likely encounter errors related to incompatible versions of TensorFlow, Protobuf, Keras, and TensorBoard.

```
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the
source of the following dependency conflicts.
tensorflow-intel 2.18.0 requires keras>=3.5.0, but you have keras 2.10.0 which is incompatible.
tensorflow-intel 2.18.0 requires protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3, but you have
protobuf 3.19.4 which is incompatible.
tensorflow-intel 2.18.0 requires tensorboard<2.19,>=2.18, but you have tensorboard 2.10.1 which is incompatible.
```

To resolve, follow these steps to align the versions properly:

Step 1: Check the Current Package Versions

Run the following command to display the current versions of the key packages:

```
pip show tensorflow keras tensorboard protobuf
```

Step 2: Uninstall the Incompatible Packages

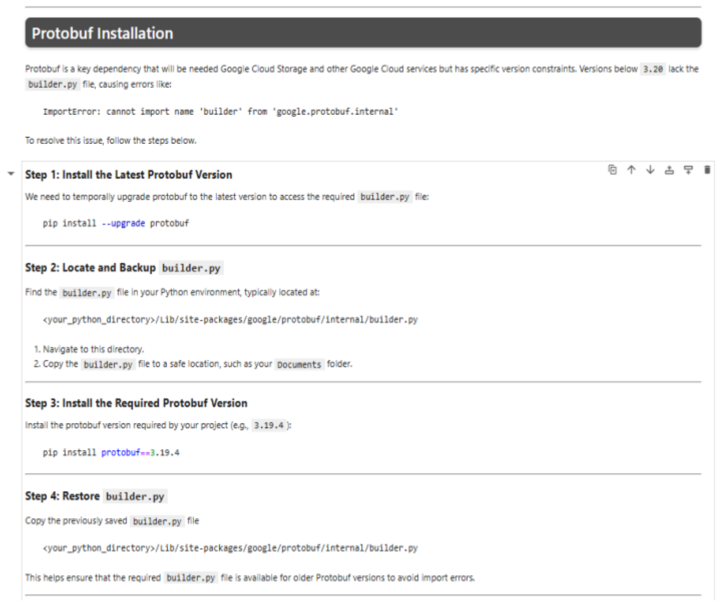
First, uninstall the existing versions of the conflicting packages:

```
pip uninstall -y tensorflow keras tensorboard protobuf
```

Step 3: Install the Compatible Versions

Install the specific versions of the required packages to resolve conflicts:

```
pip install tensorflow==2.10.0
pip install keras==2.10.0
pip install tensorboard==2.10.1
pip install protobuf==3.19.4
pip install streamlit==1.13.0
```



Protobuf Installation

Protobuf is a key dependency that will be needed Google Cloud Storage and other Google Cloud services but has specific version constraints. Versions below `3.20` lack the `builder.py` file, causing errors like:

```
ImportError: cannot import name 'builder' from 'google.protobuf.internal'
```

To resolve this issue, follow the steps below.

Step 1: Install the Latest Protobuf Version

We need to temporally upgrade protobuf to the latest version to access the required `builder.py` file:

```
pip install --upgrade protobuf
```

Step 2: Locate and Backup `builder.py`

Find the `builder.py` file in your Python environment, typically located at:

```
<your_python_directory>/Lib/site-packages/google/protobuf/internal/builder.py
```

1. Navigate to this directory.
2. Copy the `builder.py` file to a safe location, such as your `Documents` folder.

Step 3: Install the Required Protobuf Version

Install the protobuf version required by your project (e.g., `3.19.4`):

```
pip install protobuf==3.19.4
```

Step 4: Restore `builder.py`

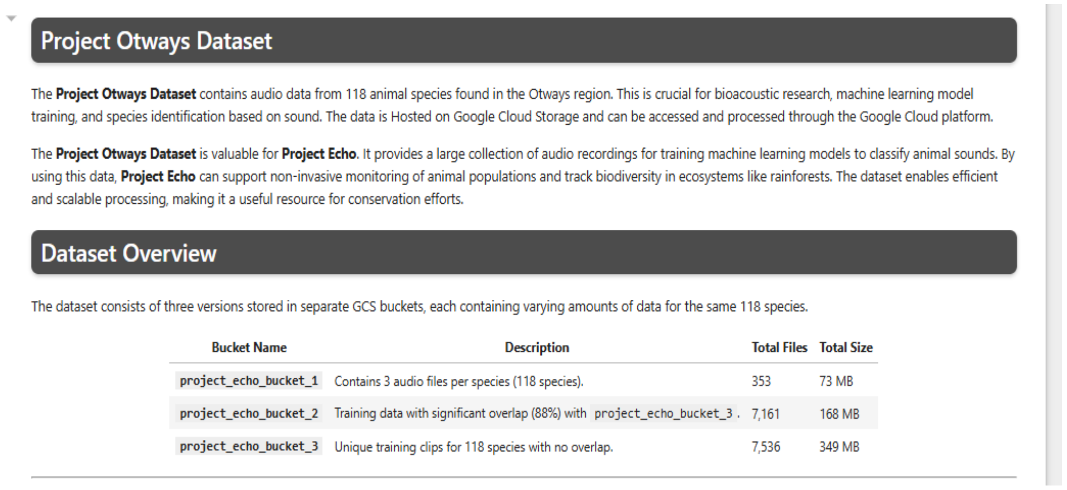
Copy the previously saved `builder.py` file

```
<your_python_directory>/Lib/site-packages/google/protobuf/internal/builder.py
```

This helps ensure that the required `builder.py` file is available for older Protobuf versions to avoid import errors.

Defining the Project Otways Dataset Structure and Purpose

To help students understand the dataset's importance, a new section clearly defines the Project Otways Dataset and its relevance to Project Echo. This section outlines the contents of each storage bucket, explaining how the data supports the project's goals. By providing this context, students can better appreciate the significance of the dataset and how it informs their work.



Project Otways Dataset

The **Project Otways Dataset** contains audio data from 118 animal species found in the Otways region. This is crucial for bioacoustic research, machine learning model training, and species identification based on sound. The data is Hosted on Google Cloud Storage and can be accessed and processed through the Google Cloud platform.

The **Project Otways Dataset** is valuable for **Project Echo**. It provides a large collection of audio recordings for training machine learning models to classify animal sounds. By using this data, **Project Echo** can support non-invasive monitoring of animal populations and track biodiversity in ecosystems like rainforests. The dataset enables efficient and scalable processing, making it a useful resource for conservation efforts.

Dataset Overview

The dataset consists of three versions stored in separate GCS buckets, each containing varying amounts of data for the same 118 species.

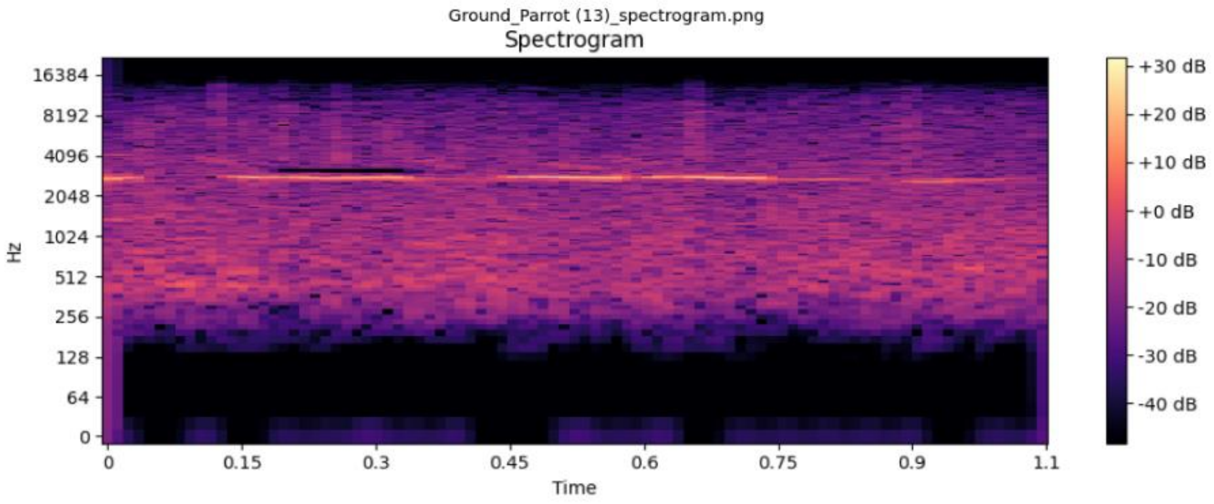
Bucket Name	Description	Total Files	Total Size
`project_echo_bucket_1`	Contains 3 audio files per species (118 species).	353	73 MB
`project_echo_bucket_2`	Training data with significant overlap (88%) with `project_echo_bucket_3`.	7,161	168 MB
`project_echo_bucket_3`	Unique training clips for 118 species with no overlap.	7,536	349 MB

Implementation

Dataset implementation

The third phase involves exploring and preprocessing the dataset, providing students with practical experience in handling raw data. This section of the onboarding task begins with students downloading the dataset and organizing it on their local machines. This step not only familiarizes them with the Project Echo repository but also encourages them to explore its structure and available resources.

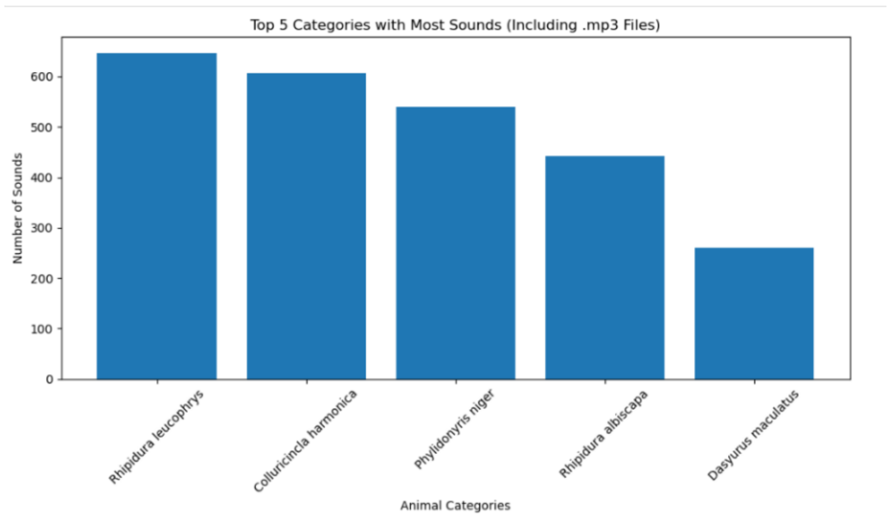
To accommodate low-powered machines and ensure a seamless experience, only 50 random audio files from the dataset are processed. These files are then converted into spectrograms and saved as PNG images. Upon completion, students are prompted to view 10 of the generated spectrograms. This visual feedback enables them to observe the outcomes of their work, reinforcing their understanding of the conversion process and building confidence for future tasks.



(e.g. of the spectrogram created)

Dataset Exploration
The next step introduces students to dataset exploration; They are tasked with identifying the top five categories with the most sound files. A small code snippet is provided, requiring students to specify the file types (e.g., .wav and .mp3). This approach encourages active engagement with the notebook and ensures they thoroughly interact with the content rather than rushing through it.

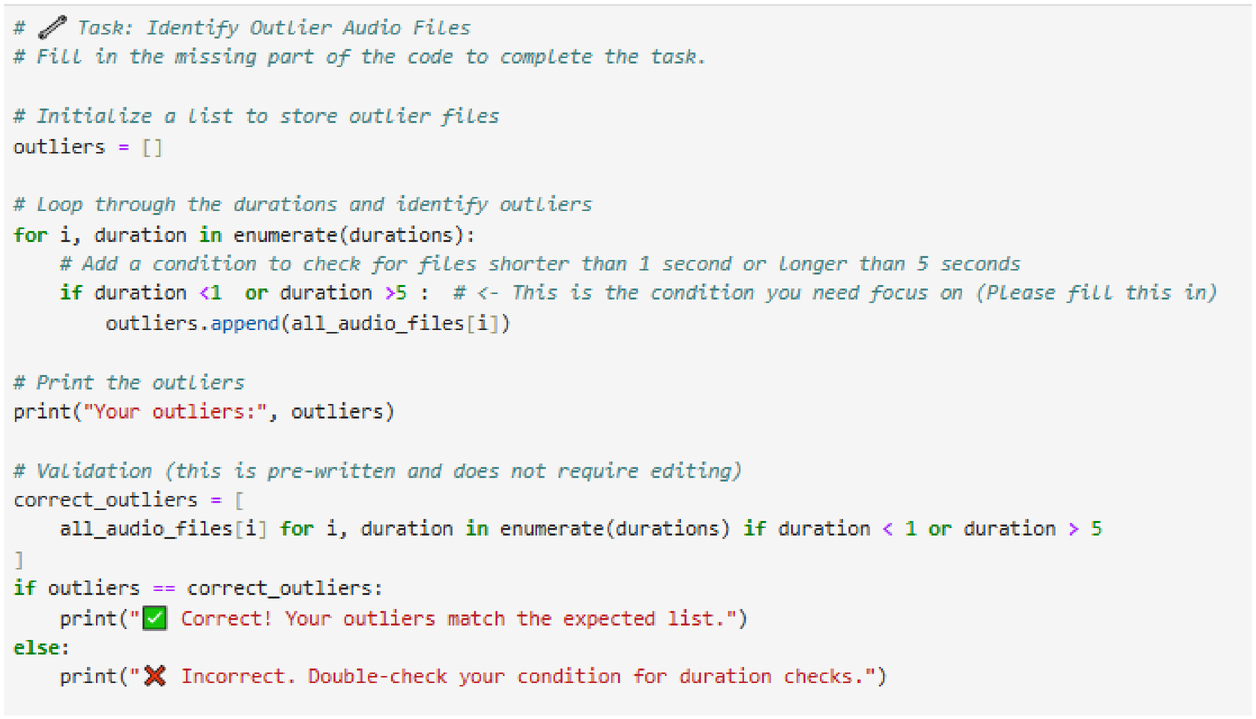
Upon completing the task, the results are displayed in a bar chart. This visualization helps students gain a clearer understanding of the dataset's sound distribution and highlights the importance of data exploration in shaping the project's direction.



(e.g. of the bar graph)

Data Preprocessing
In this step, students are tasked with identifying audio files with unusual durations. While real-world preprocessing often involves multiple criteria for detecting outliers, this task focuses solely on duration for simplicity. A straightforward condition is provided to flag files shorter than one second or longer than five seconds as outliers.

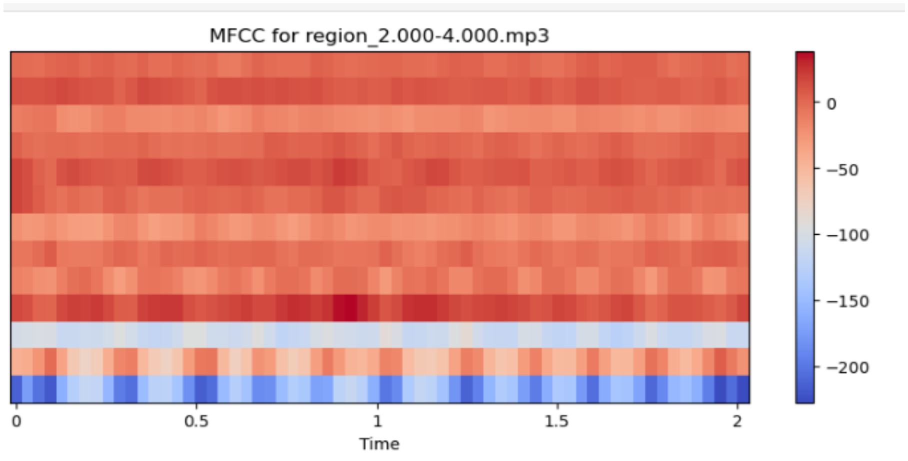
This task offers a clear introduction to the importance of data cleaning in machine learning workflows. By looping through file durations and identifying anomalies, students gain practical experience in handling irregularities within a dataset, building their understanding of essential preprocessing techniques.

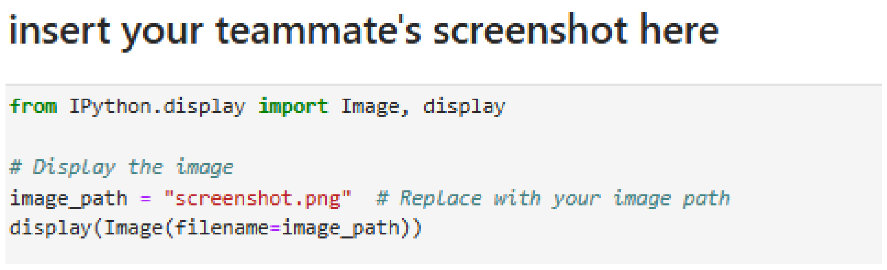


```python
# 🔧 Task: Identify Outlier Audio Files
# Fill in the missing part of the code to complete the task.

# Initialize a list to store outlier files
outliers = []

# Loop through the durations and identify outliers
for i, duration in enumerate(durations):
    # Add a condition to check for files shorter than 1 second or longer than 5 seconds
    if duration <1  or duration >5 :  # <- This is the condition you need focus on (Please fill this in)
        outliers.append(all_audio_files[i])

# Print the outliers
print("Your outliers:", outliers)

# Validation (this is pre-written and does not require editing)
correct_outliers = [
    all_audio_files[i] for i, duration in enumerate(durations) if duration < 1 or duration > 5
]
if outliers == correct_outliers:
    print("✅ Correct! Your outliers match the expected list.")
else:
    print("❌ Incorrect. Double-check your condition for duration checks.")
```

Team Collaboration

The final step of the onboarding task emphasizes teamwork and collaboration. Students are required to run a program to generate and visualize Mel Frequency Cepstral Coefficients (MFCCs) for a random audio file. To promote collaboration, students are encouraged to share a screenshot of their teammate's MFCC heatmap. This simple yet effective activity helps break the ice and fosters an environment where team members feel comfortable communicating and supporting one another.





insert your teammate's screenshot here

```python
from IPython.display import Image, display

# Display the image
image_path = "screenshot.png"  # Replace with your image path
display(Image(filename=image_path))
```

The onboarding process for Project Echo is thoughtfully designed to provide new team members with the resources, knowledge, and confidence they need to contribute meaningfully. It guides them through setting up the development environment, downloading and exploring datasets, and engaging in preprocessing and implementation tasks, with each phase carefully aligned to the project's overarching goals.

By integrating clear troubleshooting guides, accessible documentation, and collaborative activities, the onboarding experience fosters a welcoming and supportive team culture. Each step builds progressively on the previous one, creating a cohesive and engaging pathway that encourages teamwork and open communication. This structured and inclusive approach ensures that students feel empowered to make impactful contributions to Project Echo.