Step 1: Project Folder and Virtual Environment Creation

Create Project Directory (e.g., on your E: drive) any drive besides C:\:

NB: *using command prompt but you can create manually*

E:

mkdir DeepLearningSwahili

cd DeepLearningSwahili

Create Virtual Environment using Python's built-in venv module:

python -m venv swahili_env (use any name besides "swahili_env")

Activate the Environment:

swahili_env\Scripts\activate

(Your terminal prompt should now show `(swahili_env (the environment name you setup))` at the beginning, indicating the environment is active.)

Step 2: Install and Configure FFmpeg (The External Fix)

FFmpeg is a necessary external tool that Python audio libraries like librosa and soundfile rely on to decode and process audio files. The failure to find it was a core issue.

1. **Install FFmpeg using winget** (Windows Package Manager): Open a PowerShell or Command Prompt with administrator privileges and run:

   winget install ffmpeg

This command downloads and installs the FFmpeg executables.

2. **Verify and Configure System PATH**: For Python libraries to find FFmpeg from any location, the path to its bin folder must be added to your system's environment variables.

- **Verify Installation**: In your terminal (it can be the same terminal where you ran winget), run:

  ffmpeg -version

Xerxes Codes work

If you see the version information, the installation and PATH configuration were successful, and you can skip the manual configuration below.

You can get the capital Ffmpeg path by typing this command within the command prompt

Where Ffmpeg

After obtaining the path which is usually in the C drive you go to it go to the folder of FFmpeg Open it until you reach the bin then copy that path and save it somewhere like a notepad it's what we're going to use when setting up FFmpeg in our notebook otherwise you won't be able to decode audio files

If **Verification Fails**: You must manually add the bin directory of your FFmpeg installation to your Windows **System Path** environment variable.

Step 3: Install Stable Python Dependencies (The Code Fix)

The key to resolving the "install torchcodec" (this error gave me headache) error was to use a stable, previous version of datasets that relies on **librosa** and **soundfile**, rather than the newer, often conflicting torchcodec dependency.

1. **Install Core Libraries with Version Pinning**: Ensure your virtual environment is active ((swahili_env) in the prompt) and run the following command, which pins datasets to a stable version **less than 4.0.0**

   pip install "datasets<4.0.0" transformers accelerate

   ***Crucial Insight**: By pinning datasets to a version <4.0.0, you avoid the need for torchcodec and implicitly enforce the use of the stable audio decoding backend*

2. **Install Audio and Utilities**: Ensure your virtual environment is still active ((swahili_env) in the prompt) and run the following command to Install the specific libraries that the pinned datasets version needs, plus PyTorch for your model

   pip install torch librosa soundfile torchaudio numpy pandas

Step 4: Run the Notebook

Your environment is now fully configured with:

- ✓ A clean, isolated virtual environment.
- ✓ The necessary external tool (FFmpeg) installed and accessible.
- ✓ The correct, non-conflicting Python audio libraries (datasets<4.0.0 and librosa).

1. **Install Jupyter/IPython**:

   pip install jupyter

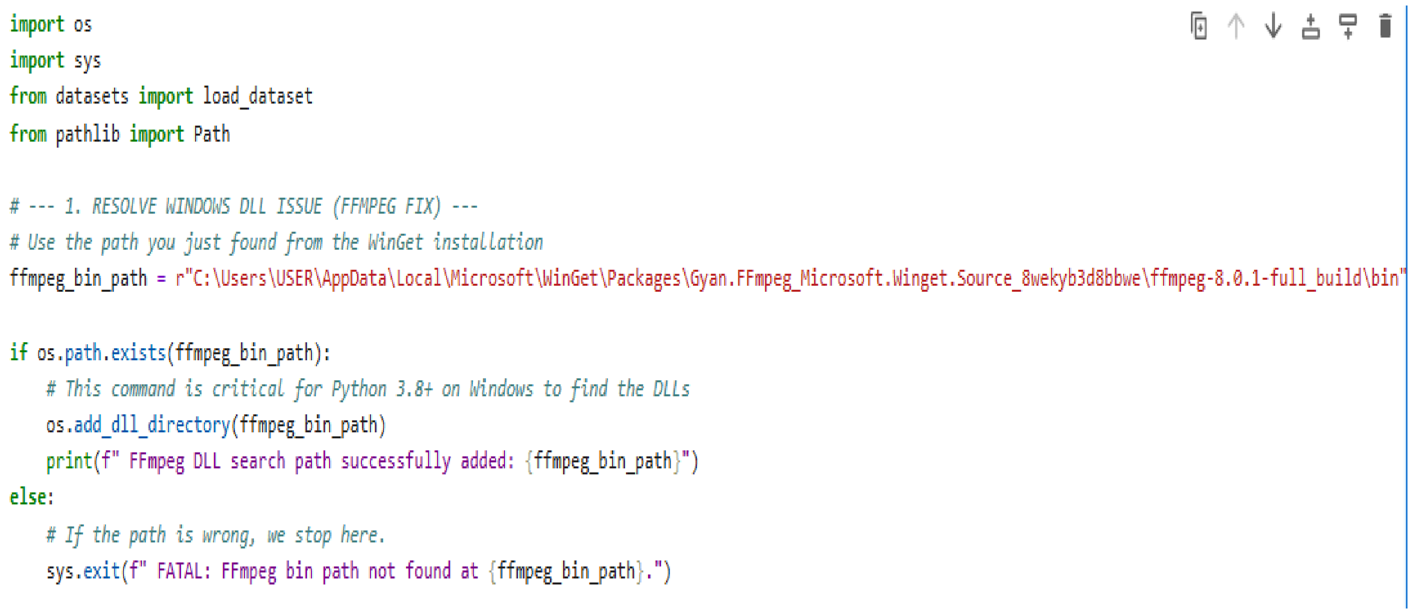
2. **Launch the Notebook**:

   jupyter notebook

Necessary imports for first cell

```
import librosa
import librosa.display
import matplotlib.pyplot as plt
import numpy as np
from collections import Counter
import os
import sys
from pathlib import Path
from datasets import load_dataset
```

Setting up Ffmpeg in notebook

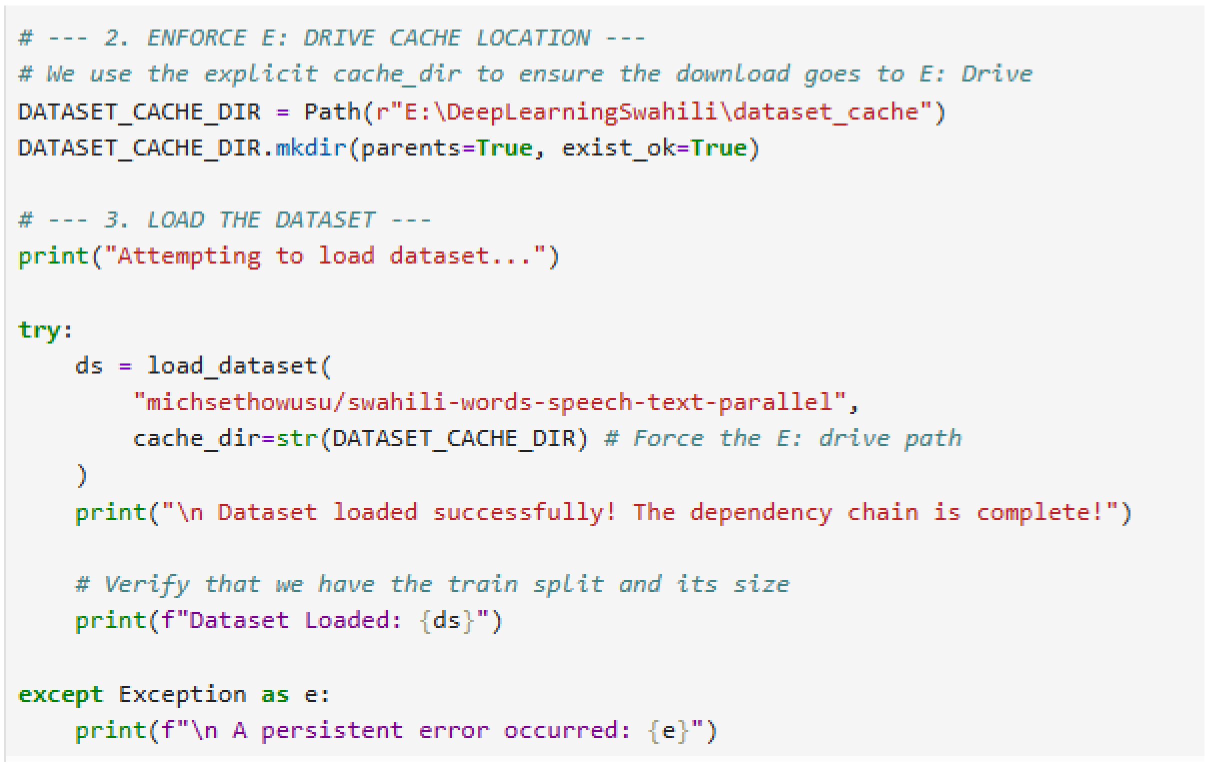


```
import os
import sys
from datasets import load_dataset
from pathlib import Path

# --- 1. RESOLVE WINDOWS DLL ISSUE (FFMPEG FIX) ---
# Use the path you just found from the WinGet installation
ffmpeg_bin_path = r"C:\Users\USER\AppData\Local\Microsoft\WinGet\Packages\Gyan.FFmpeg_Microsoft.Winget.Source_8wekyb3d8bbwe\ffmpeg-8.0.1-full_build\bin"

if os.path.exists(ffmpeg_bin_path):
    # This command is critical for Python 3.8+ on Windows to find the DLLs
    os.add_dll_directory(ffmpeg_bin_path)
    print(f" FFmpeg DLL search path successfully added: {ffmpeg_bin_path}")
else:
    # If the path is wrong, we stop here.
    sys.exit(f" FATAL: FFmpeg bin path not found at {ffmpeg_bin_path}.")
```

Xerxes Codes work

Example: Loading Swahili dataset (Group 0)

By default **huggingface** sends dataset cache when you’re loading the datasets into the C drive, so because I don’t have space in my C drive I had to force it to send the datasets into a file called **huggingface** in my E drive, you can do this in case you don’t have space on your C drive if you have space you can you can skip this step and Justi load the dataset into C drive



```
# --- 2. ENFORCE E: DRIVE CACHE LOCATION ---
# We use the explicit cache_dir to ensure the download goes to E: Drive
DATASET_CACHE_DIR = Path(r"E:\DeepLearningSwahili\dataset_cache")
DATASET_CACHE_DIR.mkdir(parents=True, exist_ok=True)

# --- 3. LOAD THE DATASET ---
print("Attempting to load dataset...")

try:
    ds = load_dataset(
        "michsethowusu/swahili-words-speech-text-parallel",
        cache_dir=str(DATASET_CACHE_DIR) # Force the E: drive path
    )
    print("\n Dataset loaded successfully! The dependency chain is complete!")

    # Verify that we have the train split and its size
    print(f"Dataset Loaded: {ds}")

except Exception as e:
    print(f"\n A persistent error occurred: {e}")
```