Data Engineer Assignment: Language & Dialect Dataset Engineering for Speech Classification Model Fine-Tuning

Description

Fine-tuning speech recognition models for **language and dialect classification** requires well-structured and diverse audio datasets.
Most public models (e.g. Whisper, Wav2Vec2, Silero) handle language detection decently, but often fail on **underrepresented dialects.**

In this assignment, your goal is to **demonstrate your data engineering approach** by preparing a small but structured and documented dataset suitable for fine-tuning such models on the language/dialect classification task.

This is **not** a model training task. Focus on **dataset design, preparation, structure, and quality.**

Core Requirements

- **Dataset sourcing:** Use any public dataset that contains **spoken audio** in **multiple languages and dialects** (e.g. Common Voice, SAMI corpus, [Arabic dialect corpora], etc.). Use a **small representative subset** (e.g. 10 languages/dialects, ~5 mins in total per dialect) for demonstration purposes.
- **Preprocessing pipeline:** Build a script or notebook that:
    - Downloads and filters the audio dataset(s)
    - Cleans metadata
    - Removes invalid/missing/empty samples
    - Converts all audio to consistent format (mono WAV, 16 kHz)
    - Structures the data in train/val/test split (even if small)
- **Labeling structure:** Ensure audio samples are clearly labeled with both **language** and **dialect**. Use a clear naming convention or metadata schema (language, dialect, speaker_id, duration, etc.).
- **File structure:** Organize data in a clean folder hierarchy or a single CSV/JSON manifest with paths and labels. Include spectrograms or waveform plots (optional) for visual inspection of diversity.
- **Quality checks:** Include code that:
    - Prints basic dataset stats (number of samples, avg duration, distribution per dialect)
    - Plots duration histogram
    - Verifies label consistency and encoding
- **Documentation:** Explain your decisions via code comments/docstrings and a README.md file. Include:
    - What data you selected and why
    - Your dialect coverage
    - Limitations of the dataset
    - How your pipeline could scale to support 100+ dialects

Optional Enhancements *(pick at least one)*

- **Data augmentation module:** Include optional pitch-shift, speed-up/slow-down, or background noise injection to simulate more realistic training scenarios.
- **Dialect balancing:** Implement simple balancing logic to make sure each dialect has roughly equal sample count.

General Optional Enhancements *(pick any)*

- **Spectrogram dashboard:** Create a Streamlit or Jupyter-based mini-dashboard to browse spectrograms and hear samples with their metadata.
- **MLOps-ready design:** Use modular scripts, logging, and config-driven design to enable scaling and reuse in future ingestion pipelines.
- **Multi-format export:** Allow exporting the dataset manifest in multiple formats (CSV, JSON, HF-compatible Dataset, etc.)

ChatGPT/LLM Usage
You may use ChatGPT or other LLMs during the assignment. If you do:

- Mention which parts were LLM-assisted.
- Briefly explain how you verified/refined the outputs.

Transparency in your README.md is enough.

Delivery
Please publish your solution in a public GitHub repo containing:

- Python scripts or notebooks to create the dataset
- Sample output (a few MB of processed audio samples + metadata)
- README.md with:
    - Setup instructions
    - Overview of your approach
    - Dataset description and statistics
    - Optional improvements, if any

Any creative extra feature is a plus. Show us how you'd bring this into the real world!