🎧 Emotion Detection from Audio using Deep Learning and Django

Project Overview

In this project, we will build a **deep learning model** trained for **emotion detection** — or more precisely, **emotion classification from audio data**.
After developing and training the model, we will integrate it into a **web application built with the Django framework**, and finally, **host it on AWS** for online accessibility.

Project Workflow

1. **Introduction & Overview**
   We begin with an overview of the problem — detecting human emotions from audio signals using deep learning.
   The project involves:
   - Building a deep learning model for audio-based emotion classification
   - Developing a web interface using Django
   - Deploying the web application on AWS
2. **Model Development**
   The core of the project lies in building and training the emotion classification model.
   - The model is trained on an **emotion dataset** to classify the emotion expressed in an audio file.
   - We use an extract_feature() function that leverages **Librosa** — a powerful Python library for audio analysis — to extract meaningful features from the input audio.
   - The chosen model architecture is an **MLP (Multi-Layer Perceptron) Classifier**, a type of feedforward neural network suitable for classification tasks.
3. **Web Application using Django**
   Once the model is ready, we create a **Django-based website** to allow users to upload audio files and get emotion predictions in real-time.
   - **Django** is a free and open-source web application framework written in Python.
   - It provides powerful built-in features such as:
     - User authentication and session management
     - Admin panel
     - Contact forms and comment boxes
     - File upload system
     - Backend–frontend communication support
   - Instead of building these components from scratch, Django allows us to **reuse and configure** them according to our project requirements.

    - Through Django’s structured framework, we can easily integrate the trained emotion detection model into the website and handle user interactions efficiently.

4. **Hosting on AWS**
   The final step is **deployment** — hosting our Django web application on **Amazon Web Services (AWS)**.
    - We will use an **EC2 instance (T2 Micro)** to deploy and serve the website.
    - This allows us to understand how **virtual servers** and **cloud hosting** work in practice.
    - Once hosted, the website and model become accessible **from anywhere on the internet**, enabling real-world testing and usage.

Conclusion

By the end of this project, we will have:

- A trained **deep learning model** capable of detecting emotions from audio data
- A **fully functional Django website** for user interaction
- A **cloud-hosted application** running on AWS

This project combines **machine learning**, **web development**, and **cloud deployment**, providing an end-to-end understanding of building and deploying AI-powered web applications.