
Dog Emotion Classification

This project classifies dog emotions from audio and video inputs using machine learning. It includes a Flask backend, a Dockerized deployment on Hostinger VPS, and a Flutter-based UI for video predictions.

Features

- **Audio Classification**: Classifies dog barks into predefined categories (aggressive, happy, howling, pain, unknown, whining).
- **Video Classification**: Predicts dog emotions and behaviors using a Flutter UI.
- **Admin Dashboard**: Web-based interface to manage datasets, train models, and view metrics.
- **Dockerized Deployment**: Simplifies the deployment process.

Prerequisites

- **Hostinger VPS**: Configured with Docker.
- **Python**: Version 3.12.
- **Flutter SDK**: For running the mobile app locally.

- ## Dog Emotion Classification Model Setup and Run Instructions on Hostinger VPS

1. Access the VPS

Connect to your VPS via SSH:

```
ssh root@82.112.236.11
```

2. Navigate to the Project Directory

```
cd ../home/Final-Dog-Emotion-Docker/
```

3. Build the Docker Image

```
docker build -t dog-emotion-classification-model .
```

4. Run the Docker Container

```
docker run -p 5003:5000 dog-emotion-classification-model
```

Access the Application

- **Admin Dashboard**: Open a browser and navigate to `http://82.112.236.118:5003`.
- **Flutter App**: Configure the Flutter application to connect to `http://82.112.236.118:5003`.

Folder Structure

Here is the detailed folder structure of the project:

```
/home/Final-Dog-Emotion-Docker/
├── app.py                  # Main Flask app
├── app2.py                 # Alternative Flask app
├── datasets/               # Raw audio datasets
│   ├── aggressive/         # Aggressive dog barks
│   ├── happy/              # Happy dog barks
│   ├── howling/            # Howling sounds
│   ├── pain/               # Painful barks
│   ├── Unknown/            # Unknown sounds
│   └── whining/            # Whining sounds
├── Dockerfile              # Docker configuration file
├── dog_model.ipynb         # Model development notebook
├── model/                  # Pretrained models and metrics
│   ├── dog_bark_classifier.pkl
│   ├── performance_metrics.pkl
│   └── rfe_selector.pkl
├── processed_dataset/      # Preprocessed datasets
```

```
│   ├── aggressive/
│   ├── happy/
│   ├── howling/
│   ├── pain/
│   ├── Unknown/
├── requirements.txt       # Python dependencies
├── templates/             # HTML templates for Flask
│   ├── dashboard.html     # Admin dashboard
│   └── login.html         # Login page
├── uploads/               # Uploaded files
└── README.md              # Project documentation (this file)
```

Docker Workflow

Commands

Navigate to the Project Directory:
```
cd ../home/Final-Dog-Emotion-Docker/
```

Build the Docker Image:
```
docker build -t dog-emotion-classification-model .
```

Run the Docker Container:
```
docker run -p 5003:5000 dog-emotion-classification-model
```

Environment Variables

Variable	Description
`FLASK_APP`	Main Flask app entry point (app.py).
`FLASK_RUN_HOST`	Host address for Flask (`0.0.0.0`).

`FLASK_RUN_PORT`	Port Flask runs on (default: 5000).

Steps To Upload Dataset And Train Model

1. **Clone the Repository**

Clone the project from the repository to your local machine:

```
git clone <repository-url>
cd Dog-Emotion-Classificayion-Model
```

2. **Create and Activate Virtual Environment**

Create a virtual environment:

```
python -m venv .venv
```

a. Activate the virtual environment:

On Windows:

```
.venv\Scripts\activate
```

On macOS/Linux:

```
source .venv/bin/activate
```

3. **Install Dependencies**

Install the necessary Python packages from `requirements.txt`:

```
pip install -r requirements.txt
```

4. **Run the Flask App**

Start the Flask app:

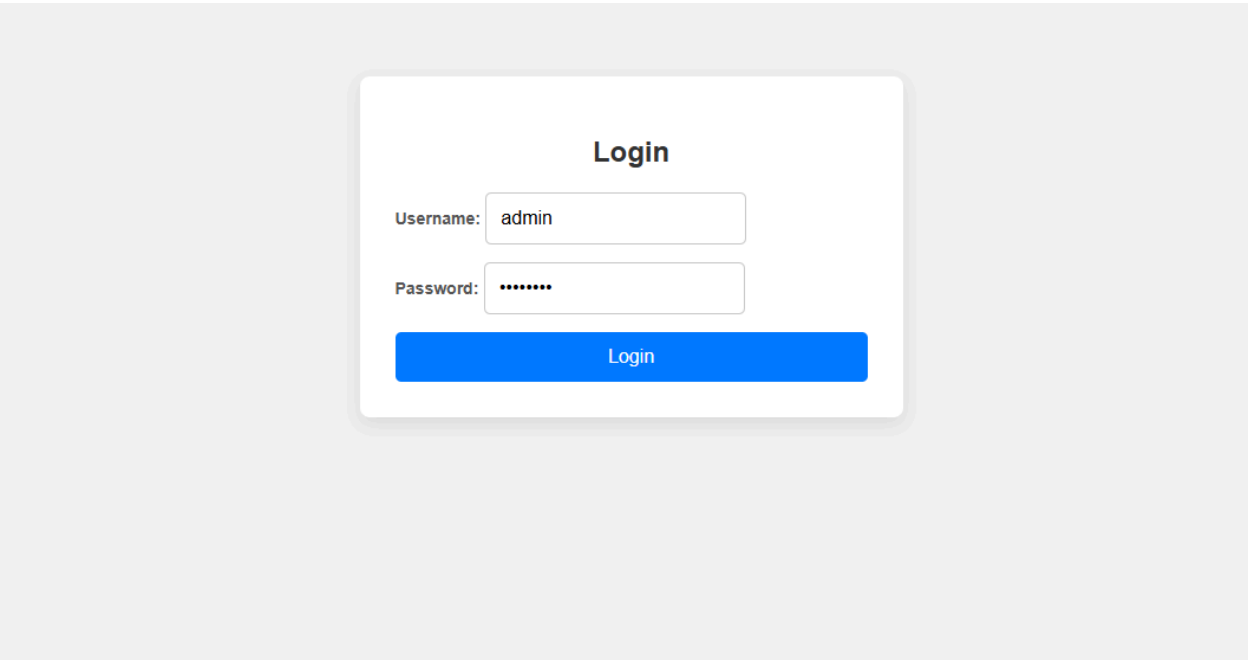
```
flask run
```

This will run the app on `http://127.0.0.1:5000`. You should see output similar to:

```
* Running on http://127.0.0.1:5000
Click on the link
```

```
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [06/Dec/2024 11:20:20] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [06/Dec/2024 11:20:20] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [06/Dec/2024 11:25:04] "POST / HTTP/1.1" 302 -
127.0.0.1 - - [06/Dec/2024 11:25:04] "GET /dashboard HTTP/1.1" 200 -
```

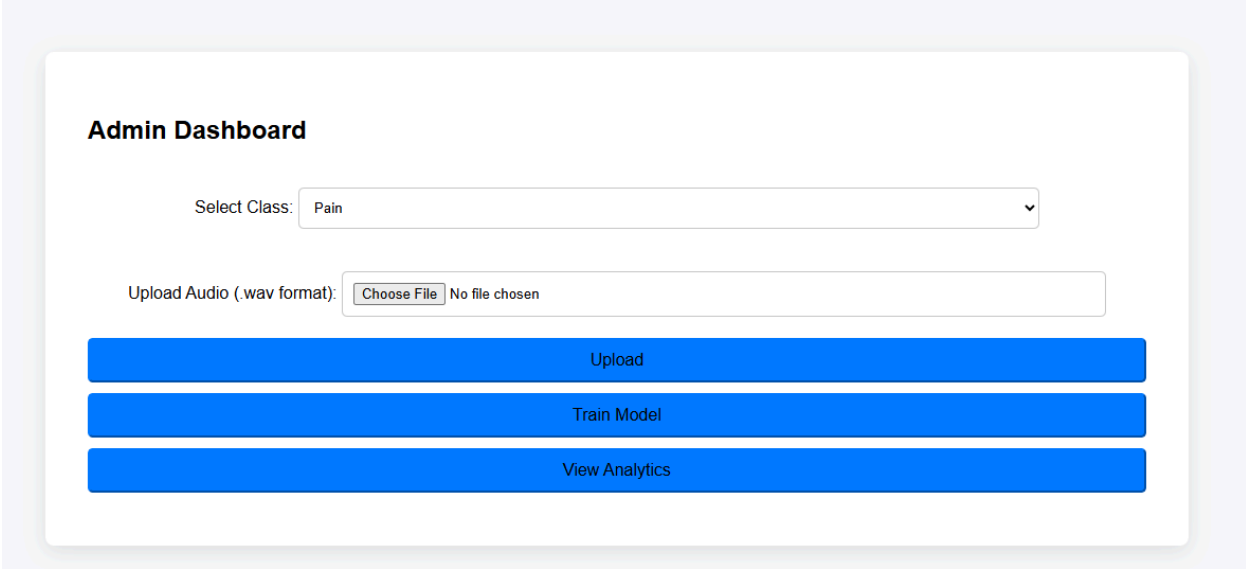
5. **There will be Interface like this**



username : admin

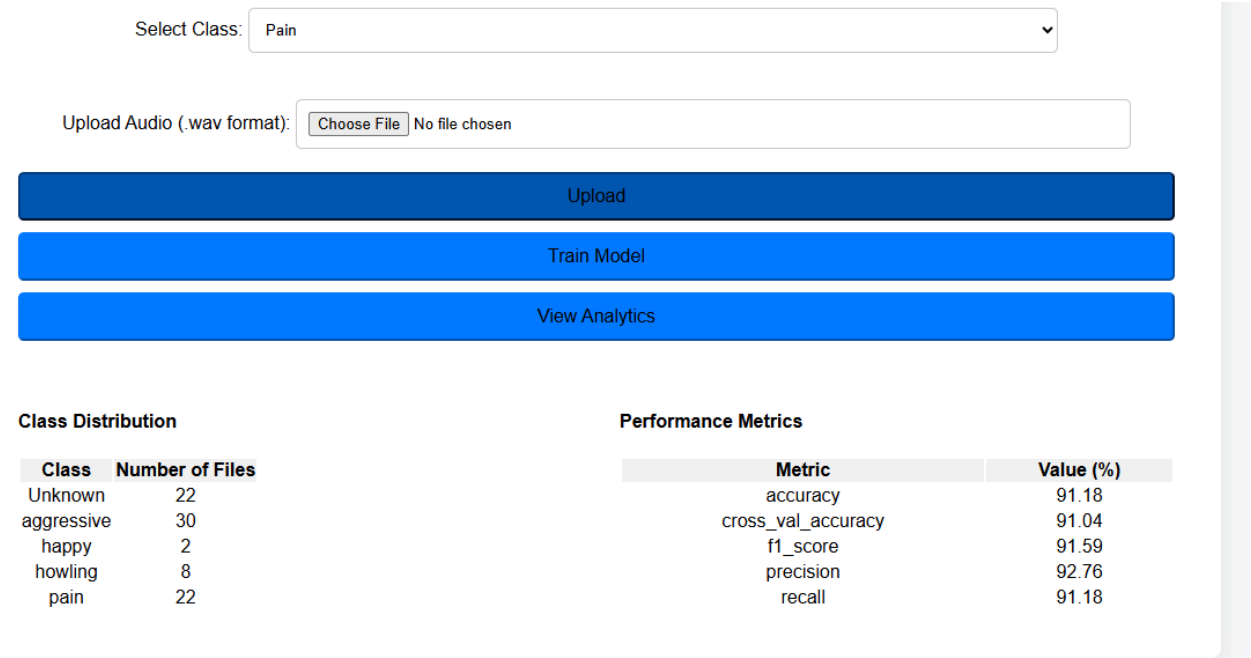
Password : admin123

6. **Upload Dataset**
   a. You can now upload `.wav` audio files through the Flask app.
   b. Select upload button to upload the audio .
      The audio will be placed in processed_dataset folder under the folder you have selected

c. 

d. Press Train to train the model. Wait for 1 to 2 mins till it completely.

7. **You can click of View Analytics Button to check the accuracy of the model and other statistics.**



- ## Configuring and Running the Flutter Application

1. Navigate to the `flutter_app` directory.

   cd flutter2

2. Open `lib/main.dart` and locate the `uri.parse` line.

```
try {
  final uri = Uri.parse("http://82.112.236.118:5003/predict");
  final request = http.MultipartRequest('POST', uri);
  request.files
      .add(await http.MultipartFile.fromPath('videofile', videoFile.path));
```

3. Replace the placeholder URI with the copied Flask server address from the previous step.

```
var uri = Uri.parse("http://<your_flask_server_address>:port");
```

Connect Mobile Device

1. Connect your mobile device to your PC via USB.
2. Enable **USB Debugging** on the mobile device.

Run the Flutter Application

1. Ensure the Conda environment is still activated in the VS Code terminal.
2. Run the Flutter app using:
   `flutter run`
3. Follow the on-screen instructions to test the app on your device.