Smart Image Classification Using Ensemble Learning

Introduction

This project delivers a lightweight and efficient image classification web application using **ensemble learning**, specifically a **stacking model** that aggregates multiple machine learning classifiers. The system is designed to classify images based on color histogram features and provides a user-friendly interface for uploading images and viewing results in real time.

Objectives

- Develop an ensemble learning model for image classification.
- Pre-process image data and extract relevant features.
- Build a web interface for users to upload images.
- Perform predictions using a trained model and return the results to the user.

System Architecture

The project is modularly structured into the following components:

`stacking/stacking.py`

- Contains the stacking model definition.
- Integrates:
    - **Base Learners**: K-Nearest Neighbours (KNN), Decision Tree, Support Vector Machine (SVM).
    - **Meta-Learner**: Logistic Regression.
- Function: `train_stacking_model()`
    - Loads `image_features.csv`.
    - Encodes class labels.
    - Trains and returns the ensemble model and label encoder.

`main.py`

- Handles pre-processing and inference.
- Core functions:

- Reads images using OpenCV.
- Extracts **colour histograms** as features.
- Loads the saved model and label encoder.
- Predicts the class and decodes the result.

`app.py`

- Implements the **Flask web server**.
- Accepts image uploads via an HTML form.
- Triggers prediction logic and returns the result on the same page.
- Stores uploaded files in a designated `uploads/` directory.

Functional Workflow

1. **User Upload**: The web UI allows users to upload an image.
2. **Pre-processing**: Image is resized and converted to a colour histogram feature vector.
3. **Prediction**:
   - Base classifiers predict individually.
   - Meta-classifier combines these predictions for a final decision.
4. **Result**: The predicted class label is rendered on the web page.

Feature Engineering

- **Colour Histogram Extraction**:
  - Images are resized to a fixed dimension.
  - Histograms are computed across RGB channels.
  - These are flattened into a feature vector for classification.

Dataset

- The model is trained on features extracted from a dataset of labelled images.
- These features and labels are stored in a CSV file (`image_features.csv`).

Dependencies

To run the project, ensure the following libraries are installed:

```
pip install flask opencv-python scikit-learn pandas numpy
```

Running the Application

```
python app.py
```

Then visit: `http://127.0.0.1:5000` to interact with the web interface.

Potential Improvements

- Extend feature extraction to include shape or texture features.
- Use deep learning-based embeddings (e.g., from a pre-trained CNN).
- Add user authentication for persistent usage.
- Implement real-time training capability.

Conclusion

This image classification project demonstrates a practical implementation of ensemble learning using stacking. By combining multiple models and serving predictions through a web interface, it offers a powerful yet accessible tool for real-time image recognition tasks.