Artificial Intelligence and Machine Learning

Smart Sorting: Transfer learning for Identifying Rotten Fruits and Vegetables

1. Introduction

- **Project Title: Smart Sorting: Transfer learning for Identifying Rotten Fruits and Vegetables**
- **Team Members:**
  - Syed Shameem
  - Pannangi Praneetha
  - Vankadari Meghana
  - Yerva Hema Nandini

2. Project Overview

- **Purpose:**
  The purpose of **Smart Sorting** is to develop an intelligent image classification system that automatically identifies whether a fruit or vegetable is **fresh or rotten** using deep learning and transfer learning techniques

  - **Key Features:**
  - Image-based classification (Fresh vs Rotten)
  - Transfer Learning using pre-trained CNN models
  - Real-time prediction through web interface
  - Upload image functionality
  - Display confidence score of prediction
  - Model accuracy evaluation with metrics

3. Architecture

1️⃣ User Layer
- User uploads fruit/vegetable image through browser.

2️⃣ Presentation Layer (Frontend)
- HTML pages inside templates/
- Static files from static/
- Form sends image to Flask backend.

3️⃣ Application Layer (Backend – Flask)
- app.py handles request.
- Saves image in static/uploads/
- Calls preprocessing function.

4️⃣ Data Preprocessing Layer

Resize image (224x224)

- Normalize pixel values
- Convert to NumPy array
- Expand dimensions

5️⃣ AIML Model Layer

- Transfer Learning model:
- MobileNetV2 / ResNet50 / VGG16
- Fine-tuned dense layers
- Binary classification output

6️⃣ Training & Testing Layer

- Dataset split:
- Training set
- Testing set
- Model evaluated using:
- Accuracy
- Precision
- Recall
- F1-Score

Setup Instructions

- **Prerequisites:**
  - Python 3.x
  - TensorFlow / Keras
  - Flask
  - Node.js (if frontend separated)
  - Git

Environment Variables: Create a `.env` file in the backend folder:

```
PORT=5000
MODEL_PATH=fruit_sorting_model.h5
```

Folder Structure

- **Client (Frontend):**

```
project/
│
├── static/
│    ├── assets/
│    ├── forms/
│    └── uploads/
│
├── templates/
│    ├── index.html
│    ├── blog.html
│    ├── blog-single.html
│    └── portfolio-details.html
│
├── app.py
├── healthy_vs_rotten.h5
├── ipython.html
└── Readme.txt
```

4. Running the Application

- git clone link
- cd smart-sorting
- pip install -r requirements.txt
- python app.py
- **Access App:**

  http://127.0.0.1.5000

5. API Documentation

Method	**Endpoint**	**Description**
GET	/	Home route
POST	/predict	Upload image and get prediction

6. Authentication

- Admin login for monitoring predictions
- JWT-based secure access

7. . User Interface Screens include:

- Landing Page
- Upload Image Page
- Prediction Result Page
- Accuracy & Loss Graph Page

8. Testing

- Tested using different fruit and vegetable images
- Verified model accuracy with unseen test data
- Manual UI testing
- API response validation

Screenshots or Demo

- Access all materials here: **SmartSortDrive**

11. Known Issues

- Accuracy depends on image clarity
- Poor lighting may reduce performance
- Model size affects response time

12. Future Enhancements

- Multi-class classification (Apple, Banana, Tomato, etc.)
- Real-time camera integration
- IoT-based smart conveyor sorting
- Google Cloud deployment
- Mobile application version