Artificial Intelligence & Machine Learning

Project Documentation

**1. Introduction**

**1.1 Project Title**

Smart Sorting: Transfer Learning for Identifying Rotten Fruits and Vegetables

**1.2 Team Members & Their Roles**

|  |  |
| --- | --- |
| **Team Members** | **Roles** |
| Yamuna .G | Data Preparation |
| Chandana .G | Model Testing |
| Vijaya sree | Flask & UI Setup |
| Pallavi | Testing & Feedback |

**2. Project Overview**

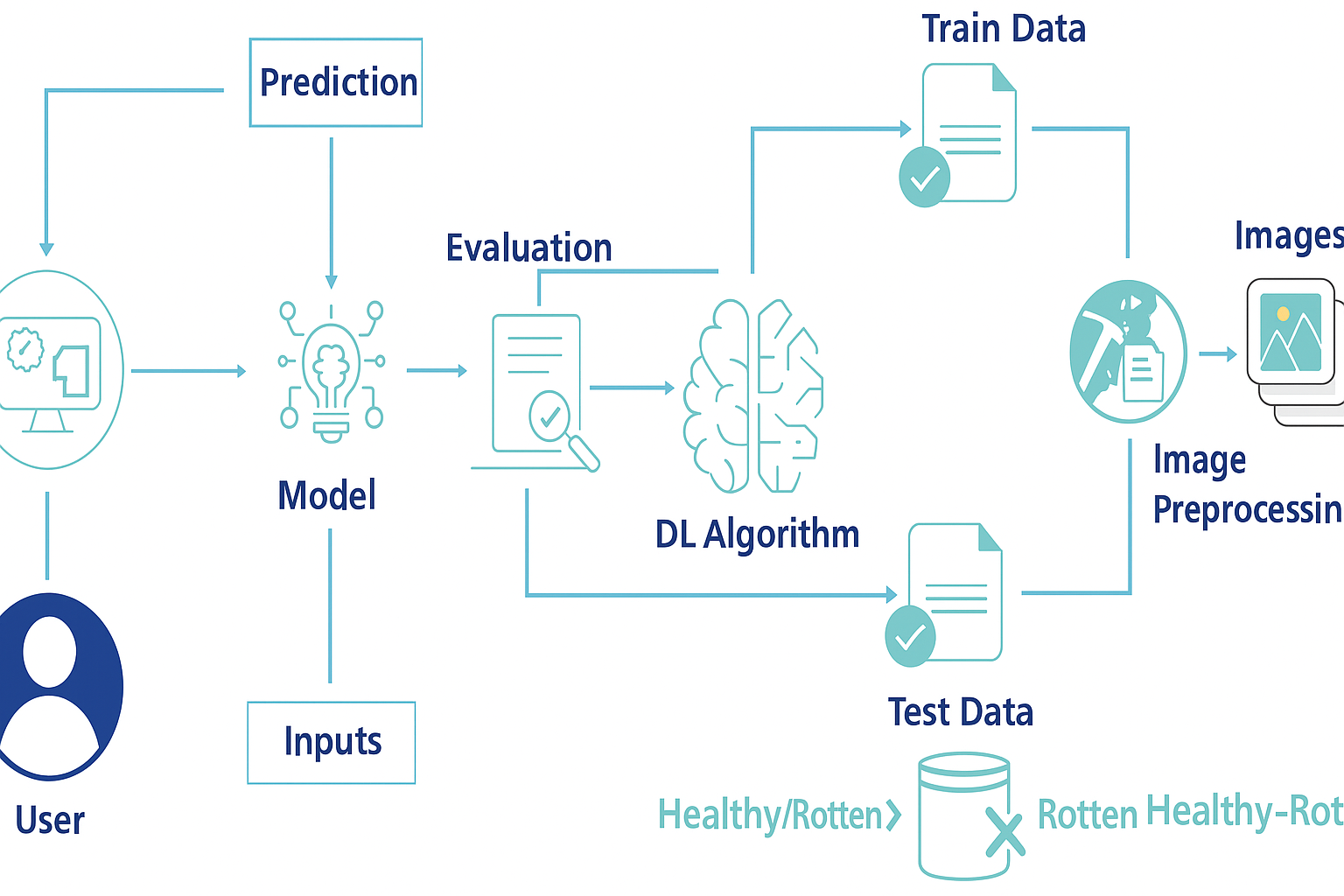
**2.1 Purpose:**

To develop an AI-powered web application that classifies uploaded images of fruits or vegetables into Healthy or Rotten categories, using Transfer Learning with VGG16.

**2.2 Features:**

1. Image upload with real-time classification
2. Real-time inference using VGG16 (Transfer Learning)
3. Confidence score display
4. "Good to Eat"/"Don't Eat" recommendation
5. Multi-class support (28 classes from real-world dataset)
6. Lightweight and optimized predictions
7. Feedback system for continuous learning
8. Error handling (invalid input, corrupt files, etc.)

**3. Architecture**



**3.1 Backend (Flask)**

* **Framework:** Flask (Python 3.9)
* **Model:** fruit\_veg\_disease\_model.keras trained using VGG16 with Transfer Learning
* **Preprocessing:**
  + Image resizing, scaling, normalization
  + Data augmentation applied during training phase
* **Modules:**
  + **app.py:** Flask routes for prediction and feedback
  + **class\_names.json:** JSON labels for 28 fruit/veg classes
  + **feedback\_data:** Stores feedback in .json format

**3.2 Frontend**

* Built using HTML, Internal CSS & Jinja Templates
* **Pages:**
  + **index.html –** Upload interface
  + **result.html –** Prediction results with score and recommendation
  + **feedback.html –** Collects feedback

**3.3 Model**

* **Model:** VGG16 pretrained on ImageNet, fine-tuned with 28-class dataset
* **Libraries:** 
  + Keras 2.10
  + TensorFlow 2.10
* **Optimizer:** Adam
* **Loss:** Categorical Crossentropy
* **Evaluation:** 
  + Accuracy
  + F1 Score
* **Multiple versions:**
  + best\_model.keras
  + final\_model.keras
  + model\_checkpoint.h5
  + model\_checkpoint.keras
  + fruit\_veg\_disease\_model.h5
  + healthy\_vs\_rotten\_model.keras

**3.4 Dataset**

* **Source:** Kaggle Fruit and Vegetable Disease Dataset
* **Structure:**
  + **train:** Healthy & Rotten classes
  + **validation:** For evaluation
* **Preprocessing Techniques Used:**
  + Image resizing & normalization
  + Data augmentation
  + One-hot encoding for labels
  + Feature scaling

**4. Setup Instructions**

**4.1 Prerequisites:**

* Python 3.9+
* Flask 2.2.5
* TensorFlow 2.10.0
* Keras 2.10.0
* Other Python libraries (see requirements.txt)

**4.2 Installation:**

Step-by-step guide to clone, install dependencies, and set up the environment variables.

1. **Create a virtual environment:**
2. conda create -n smart-sorting python=3.9
3. conda activate smart-sorting
4. **Install dependencies:**
   1. pip install -r backend/requirements.txt
5. **Run the Flask server:**
   1. python backend/app.py

**5. Folder Structure**

* + **Client:** Describe the structure of the **HTML, With Internal CSS** frontend.
  + **Server:** Explain the organization of the **Flask** backend.

**6. Running the Application**

* **Start Flask server:** python backend/app.py

**7. API Documentation**

| **Route** | **Method** | **Description** |
| --- | --- | --- |
| / | GET | Load upload form |
| /predict | POST | Handle image upload & return prediction |
| /feedback | POST | Save user feedback to local JSON |

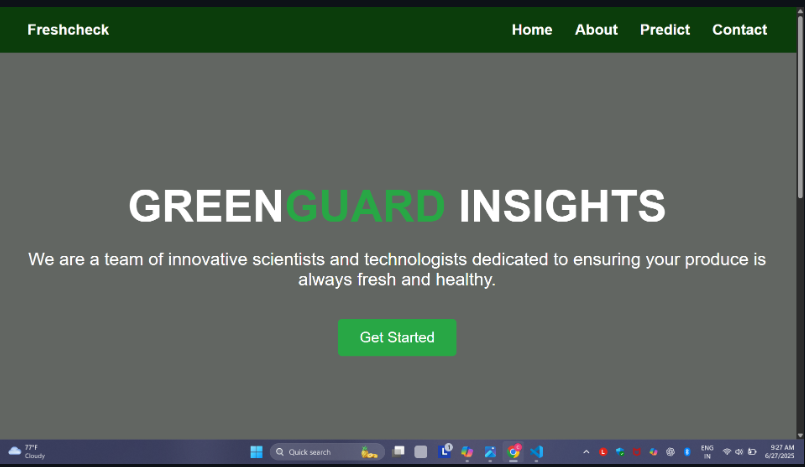
**8. Authentication**

* Not implemented.
* **Future Scope**: Added admin panel to review feedback and retrain models.

**9. User Interface**

**9.1 By index.html code**

* Starting page

****

* Content Page

A screenshot of a website

AI-generated content may be incorrect.

**9.2 By result.html**

* Choose a file from the trained dataset to predict Rotten or Health

A screenshot of a computer

AI-generated content may be incorrect.

* Predicted Healthy – Good to Eat

An orange on a white surface

AI-generated content may be incorrect.

**10. Testing**

**10.1 Data Preparation & Sample Validation**

* + **Performed by**: *yamuna*
  + **Responsibilities**:
    - Cleaned and structured dataset from Kaggle
    - Organized class-wise folders for healthy/rotten images
    - Manually checked 28-class image distribution for balance
  + **Validation**:
    - Verified preprocessed images
    - Helped ensure training/validation split had proper class representation

**10.2 Model Testing**

* + **Performed by**: *chandana*
  + **Responsibilities**:
    - Validated model performance on validation data
    - Measured accuracy, confidence score, and class prediction reliability
  + **Results**:
    - **Training Accuracy**: 88.49%
    - **Validation Accuracy**: 88.26%
  + **Tools & Libraries**:
    - TensorFlow 2.10,
    - Keras 2.10,
    - Scikit-learn
  + **Techniques Used**:
    - Confusion matrix
    - Precision, Recall, F1 Score
    - Manual verification using test images

**10.3 Flask & UI Integration Testing**

* + **Performed by**: *Vijaya sree*
  + **Responsibilities**:
    - Integrated Flask with HTML templates
    - Ensured routing between /, /predict, and /feedback
    - Handled static assets (uploaded images, result images)
  + **Tools Used**:
    - Flask debug server
    - Browser-based testing with real-time uploads

**10.4 Web Application Testing & Feedback**

* + **Performed by**: *palavi*
  + **Responsibilities**:
    - Functional testing of the entire prediction and feedback flow
    - Validated UI pages and error handling
  + **Test Cases**:
    - Upload valid/invalid image formats
    - Verify result and feedback routing
    - Test data storage in feedback\_data/ (folder)

**12. Known Issues**

* By split and copied the all the trained images and finally, I have fixed the issues.

**A screenshot of a computer

AI-generated content may be incorrect.**

* At last, all the trained dataset images are saved Succesfully

A screenshot of a computer

AI-generated content may be incorrect.

**13. Future Enhancements**

* + Responsive frontend using React.js
  + Admin dashboard to review feedback
  + Feedback-based model retraining
  + Real-time camera integration
  + Model compression for mobile support