Smart Sorting: Transfer Learning For Identifying Rotten Fruits And Vegetables

# Introduction

**Project Title**: *Smart Sorting: Transfer Learning For Identifying Rotten Fruits And Vegetables*

**Team ID**: LTVIP2025TMID41465

**Team Size**: 4

* **Team Leader**: K V Supriya – Project Lead & Model Developer
* **Team Member**: C K Madhu Hasitha – Data Engineer
* **Team Member**: Gavve Sravani – Frontend & Backend Developer
* **Team Member** : C Vani Sree – Evaluation & Deployment Specialist

# Project Overview

## Purpose

**Smart Sorting** is an innovative project that enhances the detection of rotten fruits and vegetables using advanced transfer learning techniques. It leverages pre-trained deep learning models tailored to specific produce datasets to automate sorting and improve quality control. To reduce human error, increase sorting accuracy, and minimize food waste in agriculture and food industries.

* 1. **Features**
     + Transfer Learning with Pre-trained Models
     + Real-Time Image Analysis
     + Multi-Environment Compatibility
     + User Alerts And Notifications
     + Integration with IOT Devices
     + Customizable and Explandable Dat

# Architecture

## Frontend

The user interface is developed using Streamlit, a Python-based UI framework.

Key features of the frontend:

📷 A file uploader for uploading fruit/vegetable images.

🧠 Real-time display of model predictions (Healthy or Rotten).

📊 Confidence score for each prediction.

🎉 Visual effects like balloons for healthy produce and error messages for rotten predictions.

💡 Styled layout using Streamlit's markdown and CSS for better UI experience.

The frontend dynamically updates based on user input without requiring page reloads.

## Backend

The backend is implemented in Python using:

TensorFlow and Keras for deep learning model loading and predictions.

PIL and NumPy for image processing.

Backend responsibilities:

📥 Receive and preprocess the uploaded image (resize to 224×224, normalize).

🧠 Load the .h5 trained model and run prediction on the input image.

🔍 Map predicted output to class labels (like Apple\_Healthy, Tomato\_Rotten).

🔁 Send results to the Streamlit frontend for display.

## Dataset:

* + - The system uses the “Fruits and Vegetables Diseases Dataset” containing:
    - 28 classes (14 fruits/vegetables X 2 conditions: healthy/rotten)
    - 200 images per class (5600 total images)

## Model

MobileNetV2

## Model Training Pipeline:

* + - Model trained in Google Colab using Keras + TensorFlow
    - Transfer Learning model saved in .h5 format for easy deployment
    - Supports binary classification (fresh vs rotten)

## Data Preprocessing:

* + - Image resizing to model-required dimensions (e.g., 224x224)
    - Normalization to scale pixel values
    - Augmentation (rotation, flipping, zoom) used during training
    - Dataset split into training, validation, and test sets

## Database

* + - No persistent database used in this prototype
    - All predictions are performed in memory from uploaded images

# Setup Instructions

## Prerequisites

* + - Python 3.10+
    - Streamlit
    - TensorFlow / kERAS
    - Numpy,OpenCV,Pillow
    - Google Colob (or Jupyter Notebook)

## Installation

git clone <repo-link> cd smart\_sorting\_app/

pip install -r requirements.txt python app.py

# Folder Structure

SmartSortingAI/

│

├── 📁 Project Files/

│ ├── app1.py

│ ├── fruitveg\_model.h5 # Trained Keras model

│ ├── requirements.txt

│ └── Dockerfile

│

├── 📁 Documents/ # All documentation and planning

│ ├── Ideation Phase/

│ │ ├── Brainstorming-\_Idea\_Generation-\_Prioritization\_Template.docx

│ │ └── Define\_Problem\_Statements\_Template.docx

│ ├── Empathy\_Map\_Canvas.docx

│ ├── Performance\_Testing\_Report.docx

│ ├── Project\_Planning.docx

│ ├── Requirement\_Analysis.docx

│ └── Smart\_Sorting\_Project\_Report.docx

│

├── 📁 Presentations/

│ └── Smart\_Sorting\_System.pptx # Your pitch or final presentation

│

├── 📁 Video Demo/

│ └── Smart\_Sorting\_Demo.mp4 # Recorded demo

│

├── 📄 README.md # Overview of your project and how to run it

├── 📄 .gitignore # Files to exclude from Git (like \*.pyc, \_pycache\_)

# Running the Application

To run the Streamlit app:

streamlit run app.py

Then open: [http://127.0.0.1:5000](http://127.0.0.1:5000/) in your browser.

# API Documentation

This application primarily works through image upload via form, but internally uses:

## POST /

* **Route:** /
* **Method:** POST
* **Input:** Uploaded image (jpg/png) of fruit/vegetable

## Processing:

* + Image resized and normalized
  + Passed to the CNN model (smart\_sorting\_model.h5)

## Output:

* + Prediction: "Fresh" or "Rotten"
  + Result rendered back on the UI

# Authentication

This is a prototype application built for demonstration purposes. It does not implement any user login or authentication. The app is publicly accessible and designed for easy use without requiring user accounts or credentials.

# User Interface

The interface is built using Streamlit, offering a single-page minimalist design where users can:

Upload an image of a fruit or vegetable directly on the main page.

View the prediction results immediately below the upload section, including:

The uploaded image preview.

The classification result (e.g., "Apple\_Healthy" or "Banana\_Rotten").

Confidence score percentage.

Visual badges highlighting "Healthy" or "Rotten" status.

Positive feedback effects such as balloons for healthy predictions.

UI Highlights:

Soft, clean color scheme with custom CSS to enhance readability and user focus.

Responsive layout that works well on desktop and mobile browsers.

Clear and simple interaction flow—just upload and view results with no navigation between pages.

Error and warning messages guide the user if the image upload or prediction fails.

An About Us section briefly explains the project purpose and team.

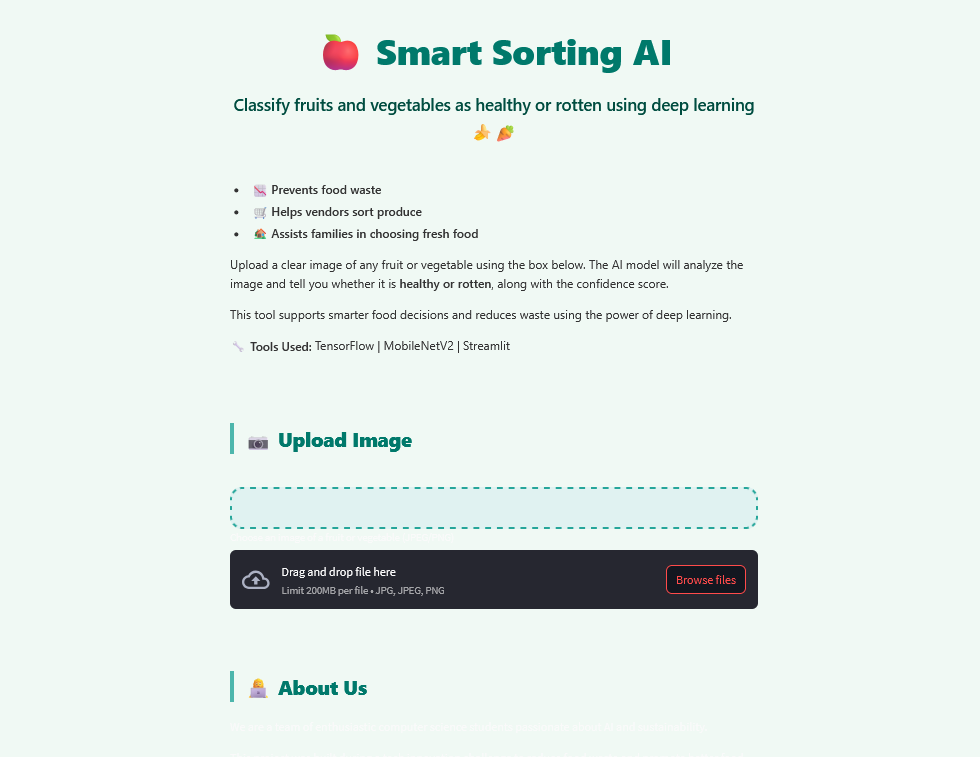
This design ensures a user-friendly experience with minimal training needed, suitable for vendors, families, and food industry stakeholders looking for quick quality checks.

# Testing

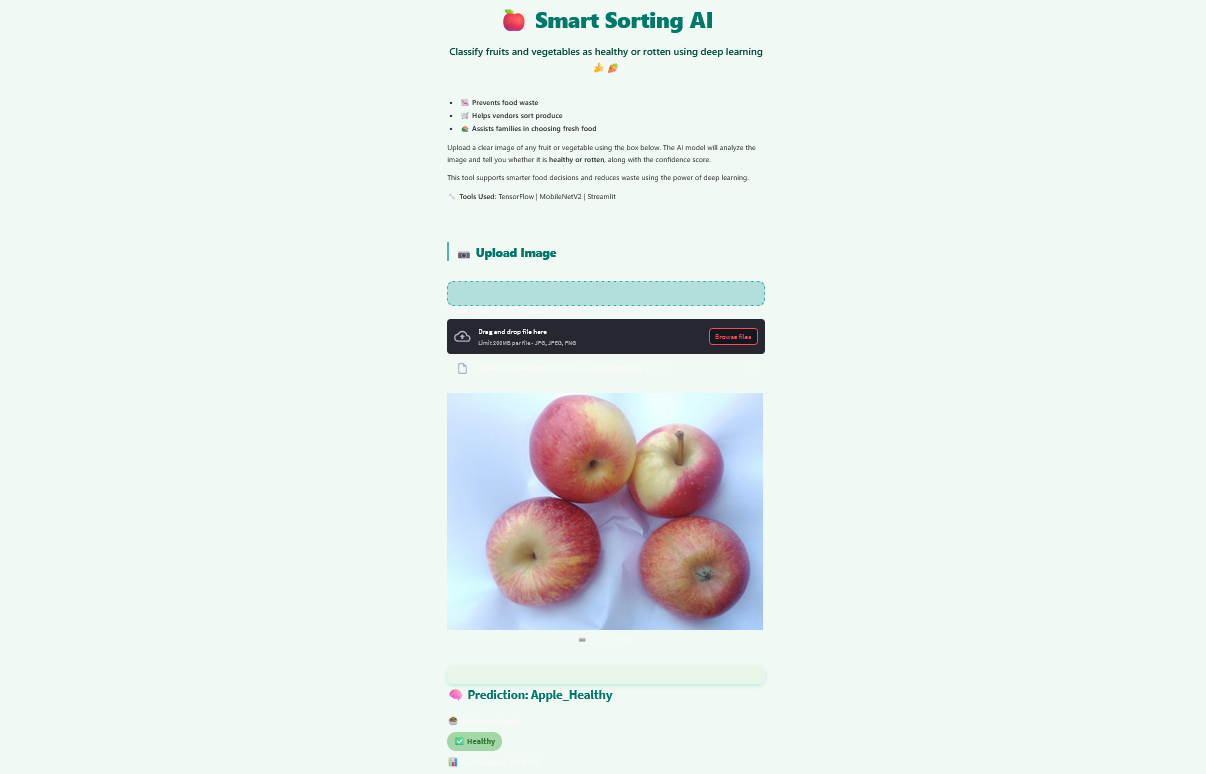
* Accuracy, Precision, and Recall calculated using test dataset
* Confusion matrix plotted for model evaluation
* Manual testing done for multiple image types and quality conditions
* Validated robustness using augmented images

# Screenshots

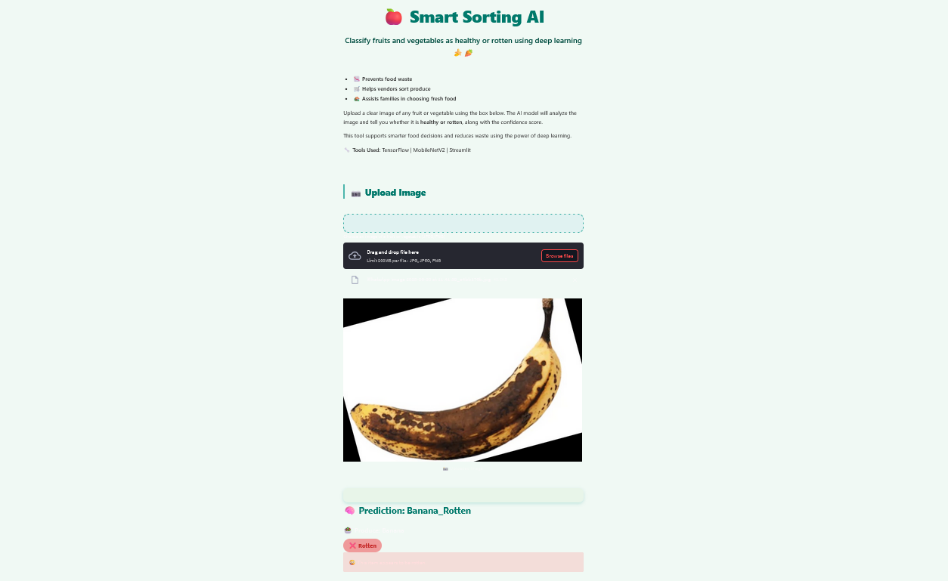
* This application allows users to upload an image of a fruit or vegetable



* After uploading, the system displays the result as “Prediction: Fresh”, if the uploaded fruit or vegetable is fresh.



* After uploading, the system displays the result as “Prediction: Rotten”, if the uploaded fruit or vegetable is rotten.



**12.Demo Video**

https://drive.google.com/file/d/1eHd4ClEfwaowreIdgKxdO-OlmMUFizPS/view?usp=drivesdk

**13.Known Issues**

* Performance may vary for poorly lit or blurred images
* No batch prediction or history tracking
* Requires internet if hosted using Colab backend

# 14.Future Enhancements

* Integrate database for storing image history and predictions
* Add multi-class classification (e.g., "Half Rotten", "Mild Spots")
* Deploy on Raspberry Pi for edge detection
* Add voice command support and native mobile app UI
* Improve dataset diversity and train on more fruit/vegetable types

**15. Deployment**

The Streamlit app for Smart Sorting is deployed and accessible online at:

https://share.streamlit.io/vanisree2204-sys/smart-sorting-ai-fresh-or-rotten-classifier/main/app.py

Users can visit this URL to upload images and get real-time classification without local setup.

This enhances accessibility and ease of use for end-users and stakeholders.