Smart Sorting: Transfer Learning For Identifying Rotten Fruits And Vegetables

1. Introduction

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

Team ID: LTVIP2025TMID46028**Team Size**: 4

- **Team Leader**: T Yaswanth– Project Lead & Model Developer
- **Team Member**:– K DinessData Engineer
- **Team Member**:Konduru Naresh– Frontend & Backend Developer

2.Project Overview

2.1. 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.

2.2. 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

3. Architecture

3.1. Frontend

- The frontend is designed using **HTML5** and **CSS3**, creating a clean, responsive user interface.
- The prediction result page (prediction.html) features:
  - A full-screen background (hero.jpg) with a **centered container**.
  - Display of prediction output (Fresh or Rotten) using the {{ prediction }} variable from Flask.
  - Display of the **uploaded fruit/vegetable image** using {{ image }} variable from Flask.
  - A styled button link to return to the homepage and try another prediction.
- The page uses **inline CSS styles** for layout, colors, image formatting, and responsive design.
- The layout is optimized for **usability** and **aesthetic appeal**, with a dark transparent background and rounded elements for better focus.

3.2. Backend

- Developed in Python using Flask
- Loads the .h5 model (smart_sorting_model.h5) at runtime
- Handles image preprocessing, resizing, and model inference
- Displays real-time predictions in the web interface

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

VGG16-pretrained convolutional neural network

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)

3.3. 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

3.4. Database

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

4. Setup Instructions

4.1. Prerequisites

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

4.2. Installation

```
git clone <repo-link>
cd smart_sorting_app/
pip install -r requirements.txt
python app.py
```

5. Folder Structure

5.1. Smartsorting app

```
├── templates/
├── index.html          # Image upload form
└──result.html            # Displays prediction result (provided above)
├── static/
├── hero.jpg          # Background image for result page
├── uploads/           # Stores uploaded images temporarily
```

5.2. Server (Flask Backend)

```
├── app.py            # Main Flask app
├── smart_sorting_model.h5
Requiremts.txt                    # Trained CNN model
```

6. Running the Application

To run the Flask app:

```
cd smart_sorting_app/
python app.py
```

Then open: http://127.0.0.1:5000 in your browser.

7. 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

8. Authentication

Not applicable – This is a prototype application.This application does not implement user login or authentication.

9. User Interface

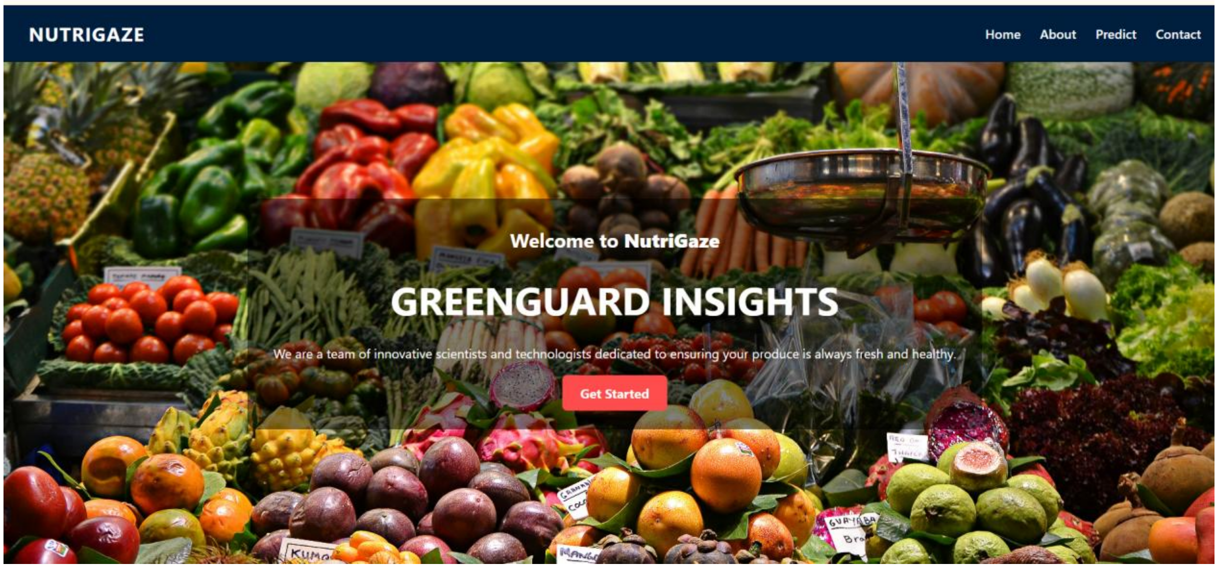
- Minimalist interface with two key pages:
  - **index.html**: Allows the user to upload an image.
  - **prediction.html**: Displays model result with uploaded image preview and reset option.
- The prediction result is clearly displayed along with the uploaded image, helping users validate visually.
- UI Highlights:
  - Dark themed result container for visual clarity
  - Rounded, responsive design with smooth UX
  - Simple retry functionality via "Try Another" button
- Ideal for industrial or low-training-required environments.

10. 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

11. Screenshots or Demo

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

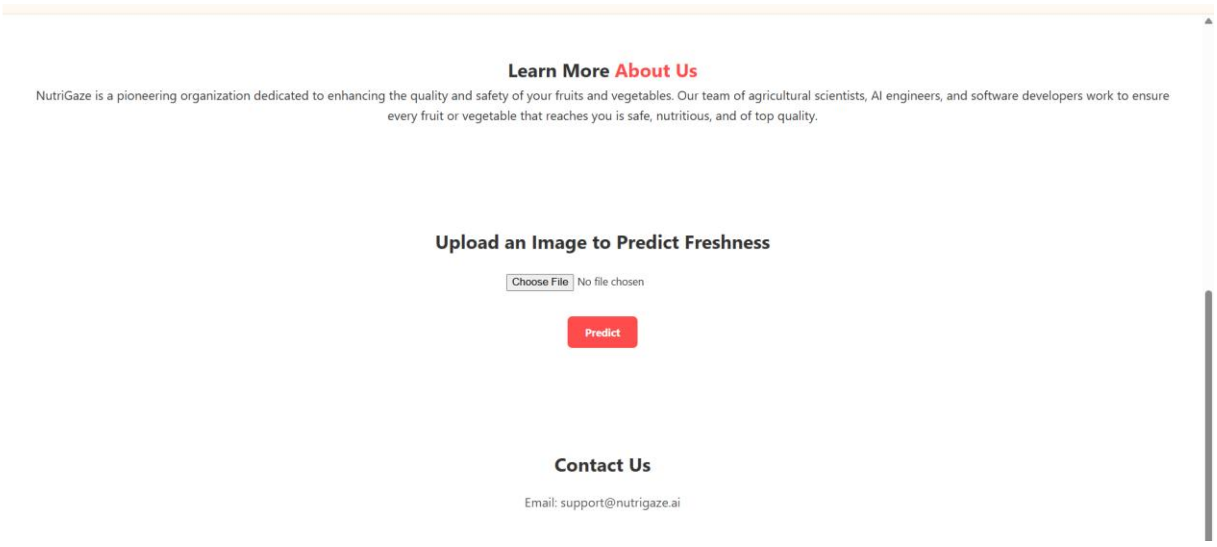


Welcome to NutriGaze

GREENGUARD INSIGHTS

We are a team of innovative scientists and technologists dedicated to ensuring your produce is always fresh and healthy.

- After uploading, the system displays the result as "Prediction: Fresh", if the uploaded fruit or vegetable is fresh.



Learn More About Us

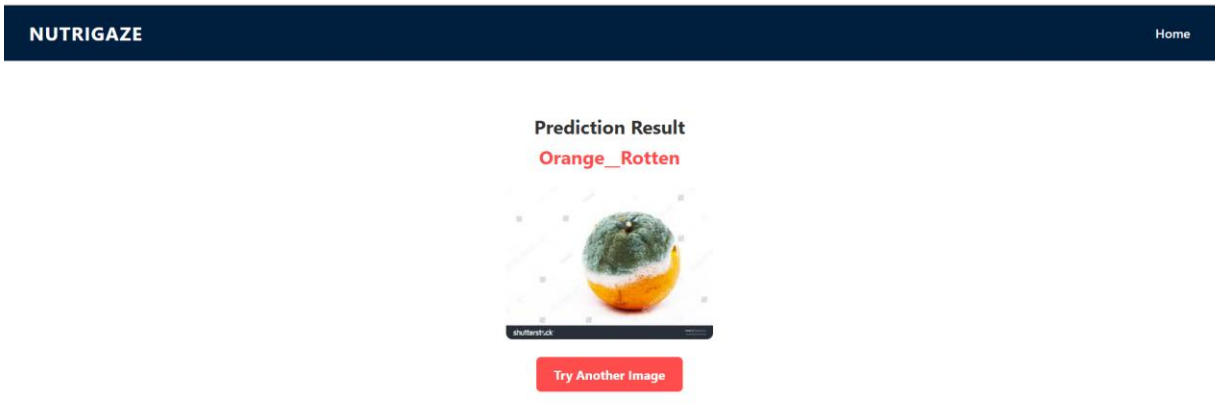
NutriGaze is a pioneering organization dedicated to enhancing the quality and safety of your fruits and vegetables. Our team of agricultural scientists, AI engineers, and software developers work to ensure every fruit or vegetable that reaches you is safe, nutritious, and of top quality.

Upload an Image to Predict Freshness

Contact Us

Email: support@nutrigaze.ai

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



Demo Link:

https://drive.google.com/file/d/1g2s25vemKb2CwkzCY0v5-KSnADq25-RX/view?usp=drivesdk

12. Known Issues

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

13. 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