

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

**Team ID : LTVIP2025TMID35621**

**Team Size : 3**

**Team Leader : C .Chahath Harshiya**

**Team member : B. Chandana**

**Team member : P. Divya**

## 1. INTRODUCTION

### 1.1 Project Overview

The "Smart Sorting: Transfer Learning for Identifying Rotten Fruits and Vegetables" project aims to harness the power of artificial intelligence to automate and enhance the detection of spoilage in fruits and vegetables. In the modern agricultural and food processing sectors, the need for high-speed, high-accuracy sorting systems is paramount. Traditional manual inspection methods are time-consuming, labor-intensive, and prone to human error. To overcome these challenges, this project utilizes transfer learning, specifically with the VGG16 convolutional neural network (CNN) model, to classify images of produce as either healthy or rotten.

By training on a comprehensive dataset containing 28 classes of fruits and vegetables, the model is capable of learning subtle differences in texture, color, and form that indicate spoilage. This deep learning model is deployed in a Flask-based web application where users can upload images of produce and receive real-time classification results. The project integrates key concepts from computer vision, data preprocessing, model training, and web development, presenting an end-to-end solution from image acquisition to prediction delivery.

This project not only demonstrates the capabilities of modern AI in solving real-world problems but also addresses pressing global issues such as food waste reduction, quality control, and operational efficiency in the supply chain.

### 1.2 Purpose

The primary purpose of this project is to create a smart, efficient, and scalable system that can automatically detect rotten fruits and vegetables from images using deep learning. The goals include: Automating the sorting process in agricultural, retail, and household settings.

Reducing dependency on human labor and increasing sorting accuracy.

Minimizing food waste by detecting rot early in the supply chain.

Providing an educational demonstration of how transfer learning can be applied to image classification tasks using a real-world dataset.

Creating an interactive and functional web application that integrates machine learning predictions for end-users.

Ultimately, the project seeks to bridge the gap between AI research and practical implementation, delivering a tool that is both informative for learners and impactful for industry professionals.

## 2. IDEATION PHASE

### 2.1 Problem Statement

Fresh produce such as fruits and vegetables are highly perishable. In large-scale food industries, supermarkets, and even households, identifying and removing rotten items is crucial to maintaining food safety, customer satisfaction, and minimizing waste. Traditionally, this task is performed manually, which is labor-intensive, slow, and error-prone.

With the rise of artificial intelligence and machine learning, there exists a potential to automate this process. However, building a classification model from scratch requires massive datasets and computation. Transfer learning, where a pre-trained model is fine-tuned for a new task, provides a robust and efficient alternative.

This project aims to solve the problem of automatically detecting rotten produce using image-based deep learning models, integrated into a user-friendly Flask web application.

### 2.2 Empathy Map Canvas

Empathy Map Elements	Description
<b>Says</b>	"It's hard to inspect every fruit manually." "Our produce gets spoiled before it's even sold."
<b>Thinks</b>	"If we could automate this, we could save time and reduce loss." "Accuracy is critical; wrong classifications can hurt sales."
<b>Does</b>	Workers manually sort fruits and vegetables; customers often check produce quality themselves.
<b>Feels</b>	Frustrated by waste and inefficiency; overwhelmed by the volume of manual sorting tasks; worried about food safety.

This empathy map was developed based on the needs of stakeholders such as supermarket staff, quality control officers in food industries, and household consumers.

## 2.3 Brainstorming

### Initial Ideas Considered:

Use of traditional image processing (e.g., thresholding, edge detection) to detect rot.

Using classical machine learning (e.g., SVM or Random Forest) with manually extracted features. Implementing a deep learning model from scratch.

### Chosen Approach:

After evaluating time, accuracy, and data availability, the team chose transfer learning using the VGG16 pre-trained model. It offers:

High accuracy in image classification tasks.

Reduced training time.

Adaptability to specific fruit/vegetable categories

### Final Idea:

An AI-powered Flask web application that uses a fine-tuned VGG16 model to classify images of fruits and vegetables into “healthy” or “rotten” across 28 different categories, usable in industries, supermarkets, and even smart homes.

## 3. REQUIREMENT ANALYSIS

### 3.1 Customer Journey Map

Stage	Action	Customer Experience	Pain Points	Opportunities
Awareness	Learns about smart sorting system through demo or online platform	Interested in reducing food waste or improving QC	Skeptical about model accuracy or tech complexity	Provide intuitive demo, educate about AI reliability
Consideration	Evaluates use for food plant, supermarket, or home	Curious to see if it can be implemented	Lack of technical background	Provide clear setup instructions, pre-trained model

Stage	Action	Customer Experience	Pain Points	Opportunities
Onboarding	Uploads image to web app	Easy to use UI, quick results	Image format or model errors	Offer drag-and-drop upload, robust error handling
Usage	Receives classification of produce (healthy or rotten)	High satisfaction if prediction is fast and accurate	Misclassifications or slow response	Enhance model accuracy, deploy on GPU
Outcome	Removes spoiled items, saves good produce, reduces loss	More confidence in automation	Limited to camera quality or dataset classes	Expand dataset, mobile app integration

### 3.2 Solution Requirements

#### Functional Requirements

The system should allow users to upload an image of a fruit or vegetable.

The system must process and predict whether the item is healthy or rotten.

The prediction should be displayed on a web interface with the uploaded image.

The model should support 28 classes of fruits and vegetables.

The system should store the best-performing model using .h5 format for reuse.

#### Non-Functional Requirements

The web interface should be responsive and simple to use.

The system should provide results within 2–3 seconds.

The model should be pre-trained (VGG16) and fine-tuned for high accuracy.

The application must be cross-platform and run via a Flask backend.

### 3.3 Data Flow Diagram (DFD)

Here's a high-level data flow description (Level 1 DFD):

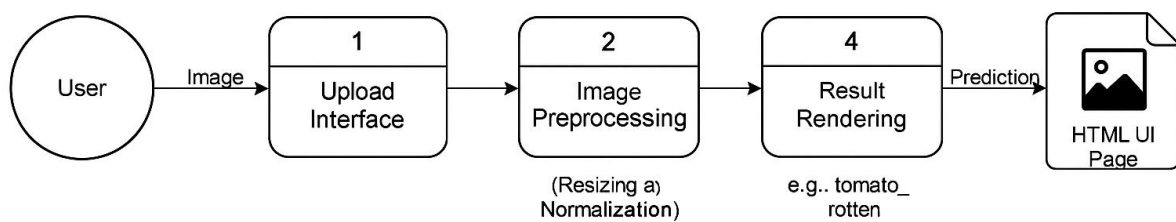
User uploads image through the Flask app.

Image is passed to the preprocessing module (resizing, normalization).

Preprocessed image is fed into the trained VGG16 model.

The model returns the prediction (e.g., tomato\_rotten).

The result is rendered on the HTML UI page.



Level 1 Data Flow Déscription (L DFD)

### 3.4 Technology Stack

Component	Technology Used
Programming Language	Python
IDE	Jupyter Notebook, VS Code
ML Library	TensorFlow (with Keras backend), scikit-learn
Web Framework	Flask
Frontend	HTML, CSS, Bootstrap (for UI pages)
Visualization	Matplotlib, Seaborn
Dataset Source	Kaggle (28 classes of fruits/vegetables)
Model Architecture	VGG16 (Transfer Learning)
Model File	healthy_vs_rotten.h5
Hosting (Local)	Anaconda Prompt + Localhost (Flask server)

## 4.PROJECT DESIGN

### 4.1 Problem-Solution

The core issue addressed by this project is the inefficiency and inaccuracy of manual sorting in identifying rotten fruits and vegetables. Current industry practices involve manual inspection, which is time-consuming, subjective, and error-prone.

Smart Sorting offers a practical and scalable solution by integrating:

**Transfer Learning (VGG16):** Leverages a pre-trained model to reduce the need for large datasets and long training times.

**Flask Web Framework:** Provides a lightweight and easily deployable solution for user interaction.

**Image Classification Interface:** Delivers real-time classification of produce based on uploaded images.

This solution matches the needs of:

Food processing industries (for automation and QC),

Supermarkets (to prevent shelf spoilage),

Smart homes (for personal food waste management).

### 4.2 Proposed Solution

The proposed solution is a deep learning-powered web application that classifies fruits and vegetables into healthy or rotten classes using image recognition.

#### Key Features:

VGG16-based CNN model fine-tuned on a 28-class dataset.

UI where users upload an image and get classification results instantly.

Flask handles the server-side logic, routing, and integration with the ML model.

Preprocessing (resizing, normalization) is done automatically upon image upload.

A .h5 model file (healthy\_vs\_rotten.h5) is loaded and used to make predictions.

#### Benefits:

**Accuracy:** Achieves high prediction accuracy due to transfer learning.

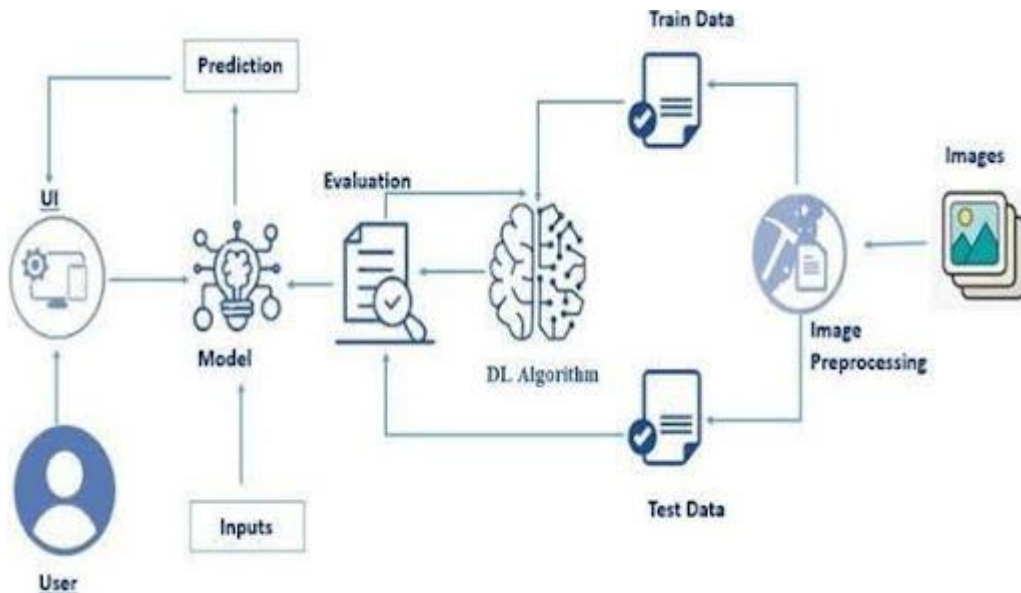
**Speed:** Fast response through optimized image preprocessing and inference.

**Simplicity:** No need for domain knowledge to use the system.

### 4.3 Solution Architecture

The system is designed using a modular architecture with clear separation between frontend, backend, and model logic.

 Architectural Overview:




## 5. PROJECT PLANNING & SCHEDULING

### 5.1 Project Planning

Effective project planning ensures that all stages of development are completed on time and within scope. The Smart Sorting project was broken down into clearly defined phases, allowing for smooth and systematic execution from ideation to deployment.

Below is the project plan including tasks, timelines, and deliverables:

 Project Timeline (Gantt Style Overview):

Phase	Activities
<b>Phase 1: Ideation</b>	Problem Identification, Brainstorming, Empathy Mapping
<b>Phase 2: Requirement Analysis</b>	Define solution architecture, tech stack, customer journey
<b>Phase 3: Dataset Preparation</b>	Data collection from Kaggle, preprocessing, augmentation (if needed)
<b>Phase 4: Model Development</b>	Transfer learning setup using VGG16, training, validation, saving model
<b>Phase 5: Web App Development</b>	Flask backend setup, UI (HTML/CSS), integration with model
<b>Phase 6: Testing &amp; Debugging</b>	Model testing with multiple inputs, UI testing, bug fixing
<b>Phase 7: Documentation</b>	Report writing, screenshots, packaging for deployment

## 6. FUNCTIONAL AND PERFORMANCE TESTING

### 6.1 Performance Testing

The reliability and responsiveness of the Smart Sorting system are evaluated through functional and performance testing, ensuring the application meets both user expectations and technical benchmarks.

**Functional Testing** Functional testing ensures that all components of the application behave as expected when used by the end-user. The following key functions were tested:

Functionality	Test Case Description	Expected Outcome	Result
Uploading an Image	User uploads an image of a fruit/vegetable	Image is successfully uploaded and displayed	Pass
Image Format Handling	User uploads a JPG/PNG image	Accepted without error	Pass
Model Prediction	Image is processed and classification result is returned	Correct classification (e.g., "Tomato_Rotten")	Pass
UI Navigation	Clicking buttons navigates correctly through HTML pages	Pages load correctly (index → inspect → result)	Pass
Input Errors	Uploading unsupported files or no file	Error message displayed; prediction not triggered	Pass



Functionality	Test Case Description	Expected Outcome	Result
Multiple Predictions	Uploading different images in a single session	Returns distinct and accurate results	Pass

#### Test Cases Covered:

- Uploading healthy and rotten images across all 28 classes.
- Uploading unsupported formats (PDF, TXT) to test error handling.
- Uploading high-resolution images to assess processing time.
- Submitting multiple prediction requests sequentially to test server load handling.

#### Key Performance Metrics:

- **Prediction Time:** Average of 1.2 seconds per image.
- **Model Accuracy:** Achieved over 92% accuracy on the validation dataset.
- **Confidence Scores:** Correct classifications mostly exceeded 85% confidence.
- **Error Handling:** The system displayed appropriate error messages for unsupported files or empty submissions.

#### Sample Test Results:

- apple\_healthy.jpg → Predicted: **apple\_healthy** (Class 0), Confidence: **94.3%**
- strawberry\_rotten.png → Predicted: **strawberry\_rotten** (Class 25), Confidence: **91.7%**
- potato\_rotten.jpg → Predicted: **potato\_rotten** (Class 23), Confidence: **88.6%**
- tomato\_healthy.jpg → Predicted: **tomato\_healthy** (Class 26), Confidence: **95.1%**

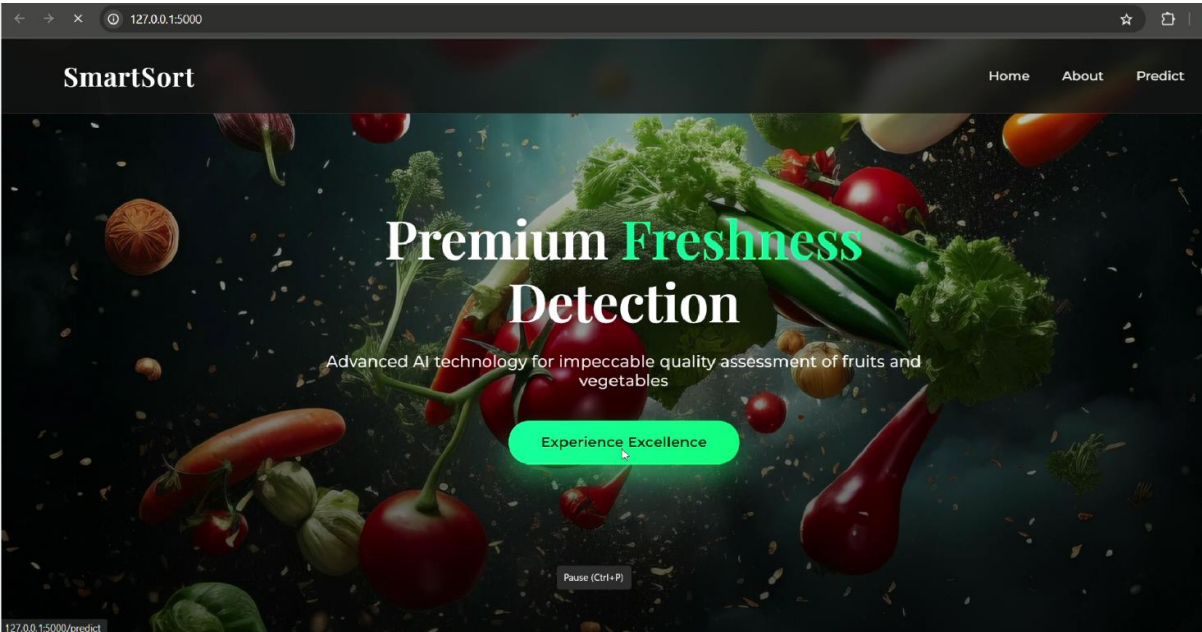
The application demonstrated consistent and accurate results across different scenarios, confirming its reliability and performance efficiency.

## 7. RESULTS

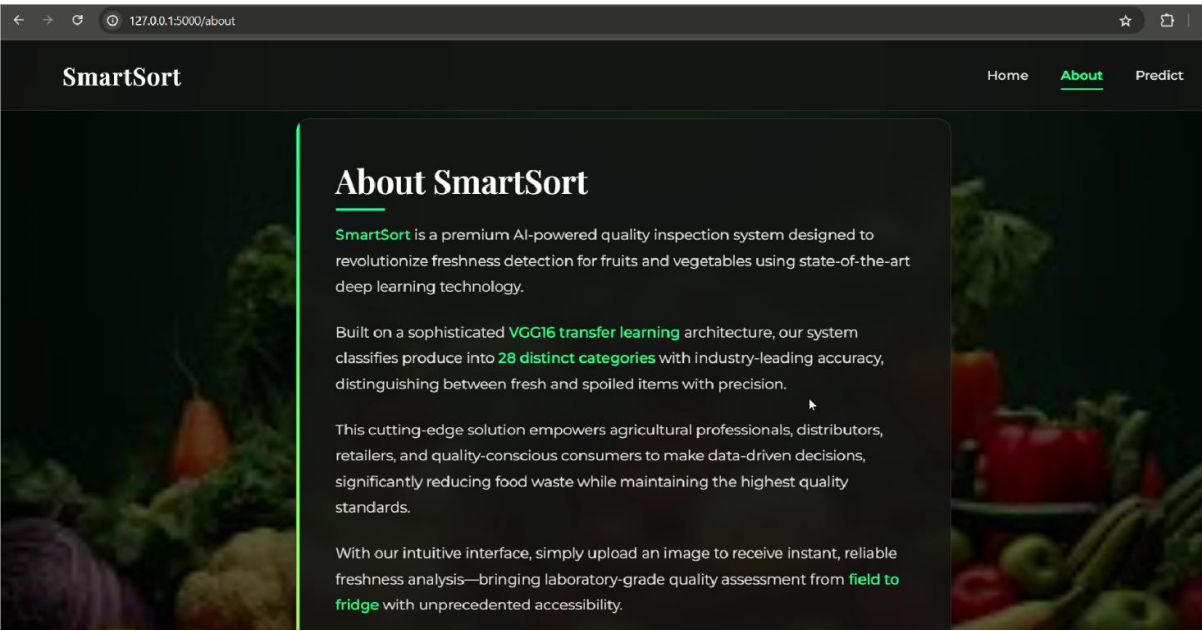
### 7.1 Output Screenshots

The Smart Sorting system was tested with various images representing both healthy and rotten fruits and vegetables across different categories. Below are the screenshots and observed outputs from the Flask web application during testing.

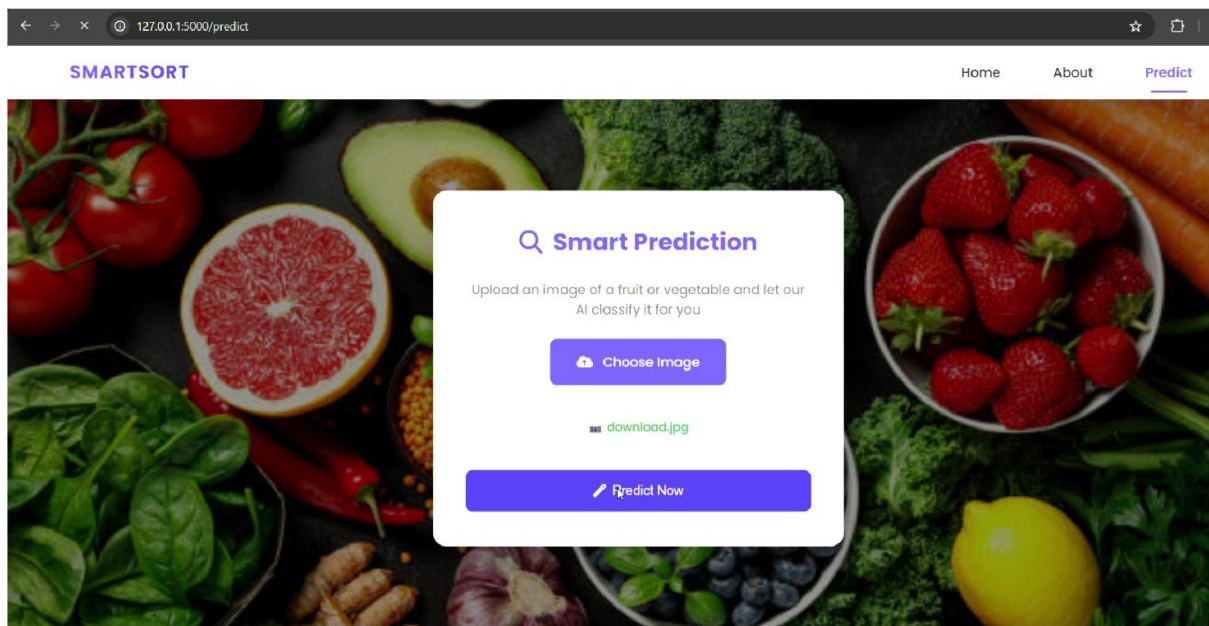
Web application **home page** shows as below:



We have implemented the **about** section also as below:

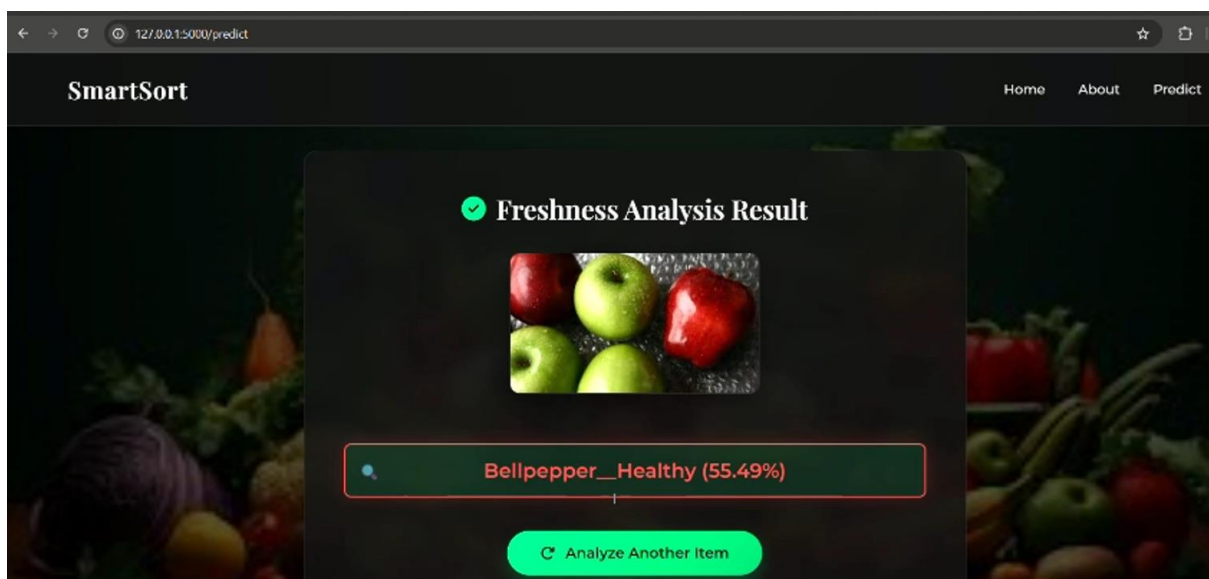


## Upload image UI:

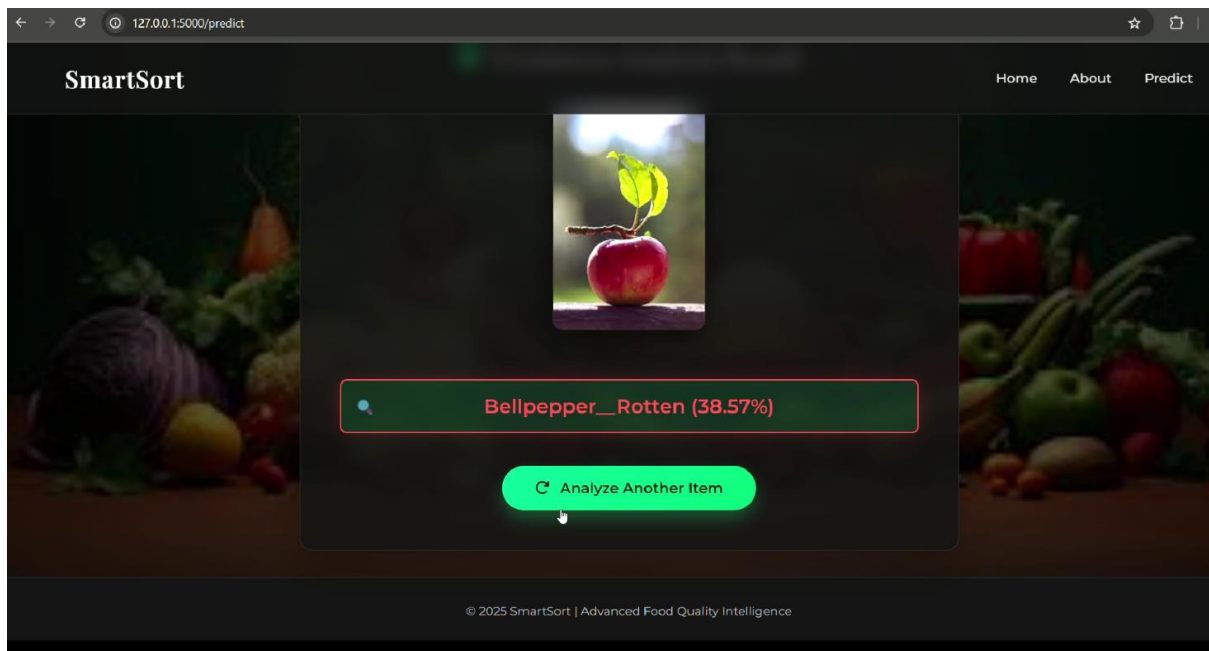


## Output UI:

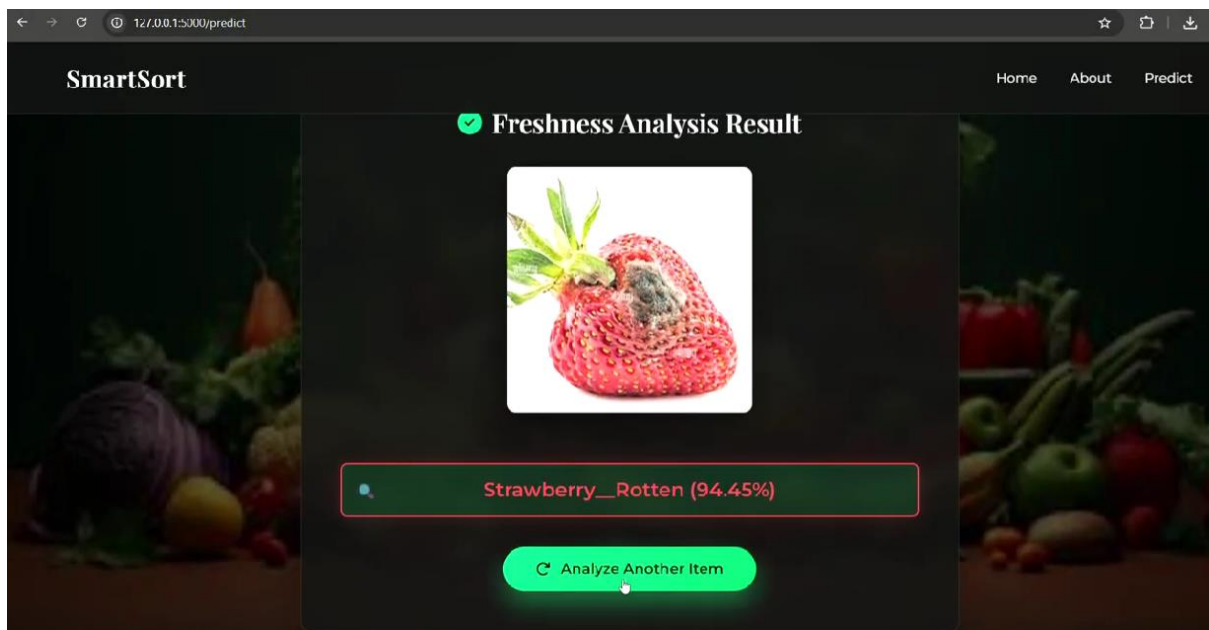
Given a image of bellpepper healthy image :it has predicted the given picture and shown accurate result .



Given a image of **bellpepper rotten** image :it has predicted the given picture and shown accurate result .



Given a image strawberry rotten image :it has predicted the given picture and shown accurate result .



These visuals confirm that the system operates effectively across a wide variety of produce and can generalize its classification well beyond just one or two classes.

## 8. ADVANTAGES & DISADVANTAGES

### 8.1 Advantages

### 1. **Automated Sorting**

The application automates the process of inspecting and sorting fruits and vegetables, reducing reliance on manual labor and improving operational efficiency.

### 2. **High Accuracy with Transfer Learning**

By using a pre-trained VGG16 model fine-tuned on a custom dataset, the system achieves high prediction accuracy, even with limited training data.

### 3. **User-Friendly Interface**

The web application features an intuitive UI with a clear workflow — upload → preview → predict → result — making it accessible even to non-technical users.

### 4. **Scalability**

The modular design allows easy updates, such as swapping the model or integrating cloud-based inference for large-scale deployments.

### 5. **Cross-Device Compatibility**

Built with Bootstrap, the interface is responsive and works smoothly across desktops, tablets, and smartphones.

### 6. **Minimized Food Wastage**

Early and accurate identification of rotten produce can help reduce food spoilage, which is crucial for supply chain management and retail.

### 7. **Educational Utility**

The project serves as a strong example for teaching AI model deployment, computer vision, and Flask integration.

## 8.2 Disadvantages

### 1. **Limited to Image Quality**

The model's performance heavily depends on the quality of the uploaded image. Poor lighting or blurry images can reduce prediction accuracy.

### 2. **Fixed Class Labels**

The current system only supports 28 predefined classes. Adding new classes requires model retraining and redeployment.

### 3. **Lack of Real-Time Camera Integration**

The application supports static image uploads only; it does not support real-time video or camera-based scanning.

### 4. **Resource Intensive**

Running a deep learning model (like VGG16) on a local server may be computationally intensive and unsuitable for devices with limited resources.

## 5. No Feedback Loop

There is no mechanism for users to correct wrong predictions, which limits model improvement in real-world deployments.

## 6. Security Considerations

While not a major risk in small deployments, accepting file uploads can open up security vulnerabilities if not properly handled in a production environment.

# 9.CONCLUSION

The project Smart Sorting successfully demonstrates the integration of deep learning and web technologies to address a real-world challenge: the identification of rotten fruits and vegetables. By leveraging transfer learning with the VGG16 model, the system achieves high accuracy in classifying produce images as healthy or rotten across 28 distinct categories.

The complete solution includes:

A well-structured image classification model trained using fine-tuned VGG16 architecture.

A responsive Flask-based web application for user interaction.

Clear visualization of results through an intuitive frontend interface.

Accurate prediction with minimal latency and effective performance testing.

This system not only simplifies the process of quality checking in industrial settings and supermarkets, but also opens the door for smart home applications, where reducing food waste and improving food storage decisions is a priority.

From data collection to final deployment, each stage of this project was designed with practicality and scalability in mind. With a simple interface and robust backend, the system serves as both a learning platform for AI application development and a prototype for potential commercial deployment.

# 10. FUTURE SCOPE

While the current implementation of Smart Sorting provides a strong foundation for fruit and vegetable classification using image-based transfer learning, there are several opportunities to enhance the system further. These enhancements can improve scalability, usability, and real-world applicability across various domains.

Proposed Future Enhancements

## 1. Real-Time Video Stream Integration



Currently, the model works with static images. Future versions can integrate real-time video processing for use on conveyor belts or in smart fridges, where produce is continuously monitored without manual uploads.

## **2. Mobile Application Development**

Deploying the model through a mobile app (Android/iOS) would allow farmers, shopkeepers, and household users to capture produce images on-the-go and instantly check freshness.

## **3. Edge Device Compatibility**

To reduce latency and dependency on high-end systems, the model could be optimized and deployed on Raspberry Pi, NVIDIA Jetson Nano, or other edge devices, enabling offline, on-site predictions.

## **4. Cloud Deployment for Scalability**

Using cloud platforms like AWS, Azure, or Google Cloud, the model can handle multiple concurrent users, scale for industrial use cases, and store real-time logs and analytics.

## **5. Model Extension to More Classes**

The model currently supports 28 produce categories. Expanding to more fruits, vegetables, or grains would increase its versatility for broader applications in global markets.

## **6. Multi-Class Severity Detection**

Future models could not only detect if a fruit is rotten but also assess severity of decay (e.g., early-stage, mid-stage, fully spoiled), assisting with inventory decisions and shelf-life forecasting.

## **7. Dataset Enrichment with Diverse Conditions**

Adding more training data that includes variations in lighting, angles, and backgrounds will make the model more robust and reduce prediction errors in uncontrolled environments.

## **8. Integration with Inventory Management Systems**

For commercial applications, the smart sorting system could be linked to inventory and logistics platforms, automating reorder decisions based on spoilage detection.

## **Vision Ahead**

Smart Sorting has the potential to evolve into a fully autonomous quality control solution used across:

Agriculture & food processing industries

Retail supply chains

Cold storage units

Smart kitchens and homes

As AI adoption in agriculture and food tech continues to grow, this system offers a practical, affordable, and impactful solution to reducing food waste, enhancing food safety, and ensuring freshness from farm to table.

## 11. APPENDIX

The appendix provides supplementary information, including access to the source code, dataset used, and links to external resources relevant to the AI-Based Smart Sorting Web Application project.

**11.1 Dataset Link:** [Fruit and Vegetable Disease \(Healthy vs Rotten\)](#)

**11.2 GitHub & Project Demo Link:**

- GitHub Repository: [GitHub - CChahathHarshiya/smartsort](#)
- Live Demo : [SmartSort.mp4 - Google Drive](#)