1. Introduction

Project Title: SmartSorting–AI-Based Fruit and Vegetable Quality Classification Web App

Domain: ArtificialIntelligence&Machine Learning

Team Members:

- K Sneha – Team Lead, Backend & ML Integration
- P S Enosh – UI/UX Designer
- Manda Thanush Kumar –FrontendDeveloper
- N Leela Rakesh –DataPreprocessing & Testing
- Appala Tharun Kumar Reddy –DatasetHandling & Documentation

Overview:

SmartSorting is an innovative web application designed to automate the classification of fruits and vegetables into "Healthy" or "Rotten" categories using image-based predictions powered by deep learning. This tool aims to assist farmers, vendors, and quality inspectors by reducing manual sorting efforts and enhancing food quality control processes.

---

2. Project Overview

Purpose:

The primary goal of SmartSorting is to streamline the quality assessment process of fruits and vegetables by leveraging AI and ML technologies. By automating the classification, the system minimizes human error, accelerates sorting operations, and ensures consistent quality standards, which is crucial for supply chain efficiency and consumer satisfaction.

Key Features:

- Image Upload & Classification: Users can upload images of fruits or vegetables, and the system will classify them as "Healthy" or "Rotten."
- Real-Time Prediction: Utilizes a pre-trained VGG16 model for swift and accurate predictions.
- User-Friendly Interface: Built with Flask and HTML/CSS, offering an intuitive and responsive design.

- Visual Indicators: Results are color-coded—green for healthy produce and yellow for rotten—to provide immediate visual feedback.

Additional Insights:
Implementing such a system can significantly reduce post-harvest losses by promptly identifying substandard produce, thereby ensuring only quality items reach the market.

---

3. Architecture

Frontend:

- Developed using HTML5, CSS3, and Vanilla JavaScript.
- Emphasizes responsive design to ensure compatibility across various devices.
- Facilitates image uploads and dynamically displays prediction results.

Backend:

- Powered by a Flask-based Python server.
- Manages routing, file uploads, and interactions with the ML model.
- Ensures secure and efficient processing of user data.

Machine Learning Model:

- VGG16 architectureemployed through transfer learning.
- Fine-tuned on a dataset comprising approximately 30,000 images across 29 classes.
- Model is saved in `.h5` format and loaded dynamically during prediction.

Data Storage:

- Currently, images are temporarily stored locally during the prediction process.
- Future iterations may integrate cloud storage solutions for scalability and persistent data management.

Extended Explanation:
The choice of VGG16, known for its depth and performance in image classification tasks, ensures high accuracy in distinguishing subtle differences between healthy and rotten produce. The modular architecture allows for easy updates and scalability.

4. Setup Instructions

Prerequisites:

- Python 3.10+
- pip (Pythonpackage installer)
- Virtualenv (recommended for environment isolation)

Installation Steps:

```bash
# Clone the repository
git clone https://github.com/YOUR_GITHUB_USERNAME/smartsorting.git
cd smartsorting

#Create a virtual environment
python -m venv venv

#Activate the virtual environment
# For Windows:
venv\Scripts\activate
# For Mac/Linux:
source venv/bin/activate

# Install dependencies
pip install -r requirements.txt

#Run the Flaskapplication
python app.py
```

Note:
Using a virtual environment ensures that dependencies are managed effectively, preventing conflicts with other projects.

5. Folder Structure

```cpp
```

```
smartsorting/
├── app.py
├── fruit_classifier_model_v5.h5
├── requirements.txt
├── templates/
│   ├── index.html
│   ├── predict.html
│   ├── result.html
│   └── contact.html
├── static/
│   └── uploads/
└── output_dataset/ (optional for retraining purposes)
```

Clarification:

The `output_dataset/` directory is optional and can be utilized for retraining the model with new data to improve accuracy or adapt to different types of produce.

---

6. Running the Application

To Launch the Application:

```bash
python app.py
```

Access via Browser:

```cpp
http://127.0.0.1:5000
```

Additional Information:
Upon launching, the application initializes the ML model, ensuring it's ready to process incoming image data for classification.

---

7. API Documentation

Endpoint: `POST /predict`

Accepts:

- Multipart form-data containing an image file.

Returns:

- JSON response with the predicted class.

Example Response:

```json
{
  "prediction": "Strawberry_Healthy"
}
```

Usage Note:
This API endpoint can be integrated into other applications or systems, allowing for seamless incorporation of the classification functionality into broader workflows.

---

8. Authentication

Current Status:

- No authentication system is implemented.

Future Enhancements:

- Implement user login/signup functionality.
- Enable tracking of user upload history and prediction results.

Rationale:
Introducing authentication will allow for personalized user experiences, data tracking, and enhanced security, especially important for commercial deployments.

---

9. User Interface

Screens Included:

- Home (`index.html`): Introduction and navigation.
- Predict: Uploadinterfaceand result display.
- About & Contact Pages: Information about the project and team.

Result Display:

- Green Box: Indicateshealthy produce.
- Yellow Box: Indicatesrotten produce.

Design Philosophy:

The UI is designed to be intuitive and accessible, ensuring users with varying technical backgrounds can effectively utilize the application.

---

10. Testing

Methodology:

- Conducted manual testing using a diverse set of fruit and vegetable images.

Test Dataset:

- Comprised of 4,384 images spanning 29 classes.

Accuracy Achieved:

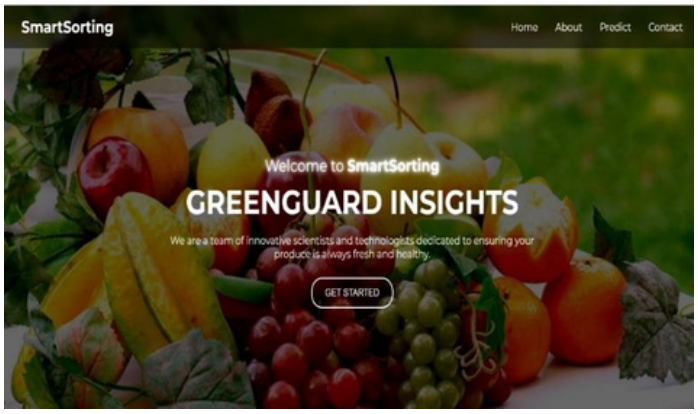
- Approximately 94% ,demonstrating the model's robustness and reliability.

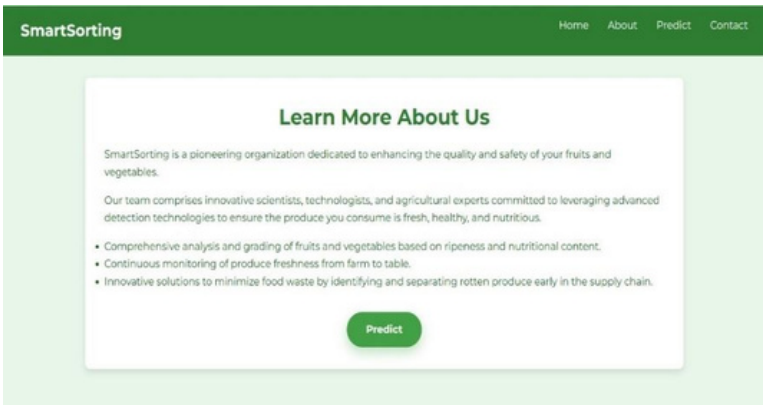
Insight:
High accuracy in testing suggests the model's potential effectiveness in real-world scenarios, though continuous evaluation is recommended as new data becomes available.

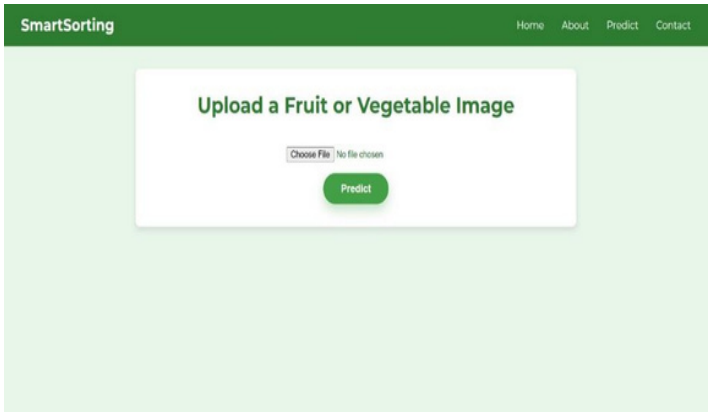
---

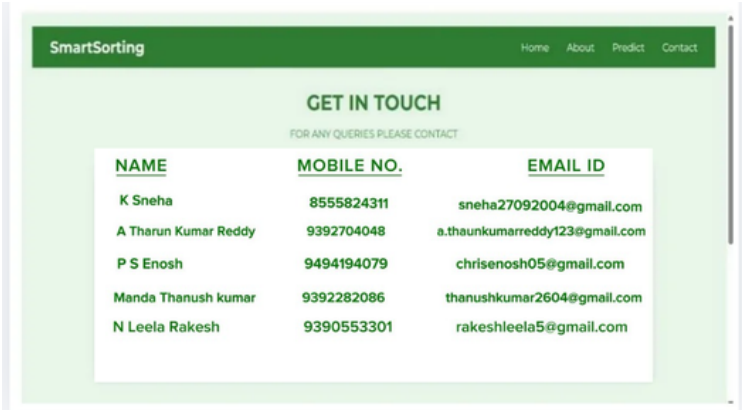
11. Screenshots or Demo

Visual Aids:









Video Link: https://drive.google.com/file/d/1orHeJW3eJbQDJ1jTvKqSVZF-UkUfh55p/view?usp=sharing

Purpose:

Visual demonstrations provide users with a clear understanding of the application's functionality and user experience.

---

12. Known Issues

- Model File Size:

  The `.h5` model file exceeds 25MB, which may pose challenges for hosting on platforms with file size limitations.
- Initial Load Time:

  The first prediction may experience a delay of 2-3 seconds due to model loading time.

Considerations:

Optimizing the model size and load time will enhance user experience, especially for applications requiring rapid predictions.

---

13. Future Enhancements

- Deployment:

  Host the web application on platforms like Render.com for broader accessibility.

- User Management:

  Implement login/signup features to personalize user interactions and maintain history.

- Batch Processing:
  Enable users to upload and classify multiple images simultaneously, increasing efficiency.

- Cloud Integration:
  Utilize cloud storage solutions (e.g., AWS S3) for scalable and persistent image storage.

- API Expansion:
  Develop a RESTful API to facilitate integration with mobile applications and other

systems.

Vision:

These enhancements aim to transform SmartSorting into a comprehensive tool adaptable to various use cases in the agricultural and retail sectors.

GitHub Repository

https://github.com/Sneha-Kongara/Smart-Sorting-Transfer-Learning-for-Identifying-Rotten-Fruits-and-Vegetables

K Sneha
Gmail: sneha27092004@gmail.com

Feel free to reach out for further assistance or collaboration opportunities!