Smart Sorting - Project Code Documentation

# 1. Dataset Structure

Organizeour dataset as follows:  
  
output\_dataset/  
├── train/  
│ ├── Apple\_Healthy/  
│ ├── Apple\_Rotten/  
│ ├── Banana\_Healthy/  
│ ├── ...  
└── test/  
 ├── Apple\_Healthy/  
 ├── Apple\_Rotten/  
 ├── Banana\_Healthy/  
 ├── ...  
Each folder should contain images for that class.

# 2. Downloading Dataset (Kaggle)

!pip install -q kaggle  
!mkdir ~/.kaggle  
!cp kaggle.json ~/.kaggle/ # Upload your kaggle.json file  
!chmod 600 ~/.kaggle/kaggle.json  
!kaggle datasets download -d kritikseth/fruit-and-vegetable-image-recognition  
!unzip -q fruit-and-vegetable-image-recognition.zip -d ./output\_dataset

# 3. Load Dataset using Keras

from tensorflow.keras.preprocessing.image import ImageDataGenerator  
  
train\_path = "output\_dataset/train"  
test\_path = "output\_dataset/test"  
  
train\_datagen = ImageDataGenerator(  
 rescale=1./255,  
 zoom\_range=0.2,  
 shear\_range=0.2,  
 horizontal\_flip=True,  
 rotation\_range=20  
)  
test\_datagen = ImageDataGenerator(rescale=1./255)  
  
train = train\_datagen.flow\_from\_directory(  
 train\_path,  
 target\_size=(224, 224),  
 batch\_size=32,  
 class\_mode='categorical'  
)  
  
test = test\_datagen.flow\_from\_directory(  
 test\_path,  
 target\_size=(224, 224),  
 batch\_size=32,  
 class\_mode='categorical'  
)

# 4. Visualize a Batch

import matplotlib.pyplot as plt  
  
x, y = next(train)  
plt.figure(figsize=(10, 10))  
for i in range(9):  
 plt.subplot(3, 3, i+1)  
 plt.imshow(x[i])  
 plt.title(f"Label: {y[i].argmax()}")  
 plt.axis("off")  
plt.tight\_layout()  
plt.show()

# 5. Flask Backend - app.py

from flask import Flask, render\_template, request  
from keras.models import load\_model  
from keras.preprocessing.image import load\_img, img\_to\_array  
from keras.applications.vgg16 import preprocess\_input  
import numpy as np  
import os  
  
app = Flask(\_\_name\_\_)  
model = load\_model("Healthy\_vs\_rotten.h5")  
class\_names = [f"Class {i}" for i in range(28)]  
  
@app.route('/')  
def index():  
 return render\_template('index.html')  
  
@app.route('/predict', methods=['POST'])  
def predict():  
 if 'image' not in request.files:  
 return "No file uploaded.", 400  
 image\_file = request.files['image']  
 if image\_file.filename == '':  
 return "No image selected.", 400  
 image\_path = os.path.join("static", image\_file.filename)  
 image\_file.save(image\_path)  
  
 image = load\_img(image\_path, target\_size=(224, 224))  
 image = img\_to\_array(image)  
 image = preprocess\_input(image)  
 image = np.expand\_dims(image, axis=0)  
 predictions = model.predict(image)  
 predicted\_class = class\_names[np.argmax(predictions)]  
 return render\_template('result.html', prediction=predicted\_class, image\_path=image\_path)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app.run(debug=True)

# 6. HTML - index.html

<!DOCTYPE html>  
<html>  
<head>  
 <title>Smart Sorting - Upload Image</title>  
</head>  
<body>  
 <h2>Upload a Fruit or Vegetable Image</h2>  
 <form action="/predict" method="post" enctype="multipart/form-data">  
 <input type="file" name="image" required>  
 <input type="submit" value="Predict">  
 </form>  
</body>  
</html>

# 7. HTML - result.html

<!DOCTYPE html>  
<html>  
<head>  
 <title>Prediction Result</title>  
</head>  
<body>  
 <h2>Prediction: {{ prediction }}</h2>  
 <img src="/{{ image\_path }}" width="300">  
 <br><a href="/">Try Another Image</a>  
</body>  
</html>