**FRUIT FRESHNESS DETECTION**

**GROUP MEMBERS :**

* RASHI TYAGI
* DIVIZA SRIVASTAVA
* AANYA MAURYA

**COURSE NAME / CODE :** EXPLORATION AND SEMINARS IN AI / ABAI1009P

**PROJECT DESCRIPTION :** This project focuses on whether fruits are fresh or rotten using computer vision . By analysing colour , texture and shape features of fruit images , the system classifies them as either fresh or rotten .

It uses basic image processing and visualization techniques in python . The project uses a dataset of fruit images categorized according to their freshness levels. These images are preprocessed — resized, normalized, and augmented — to improve model performance.

**PROBLEM STATEMENT :** In the food industry , fruit freshness is critical for quality control . Manual inspection is often subjective and time consuming . This project aims to automate the process of fruit freshness detection using image – based AI techniques . With the growing demand for automation in food quality monitoring, there is a need for an intelligent system that can **accurately and automatically assess fruit freshness**. However, building such a system is challenging due to **variations in fruit type, color, texture, lighting, and spoilage patterns**.

Hence, the problem addressed in this project is:

**To design and implement a Python-based automated system capable of detecting and classifying the freshness level of fruits (fresh, partially fresh, or spoiled) using image processing and machine learning techniques.**

**LIMITATIONS :**

* The model’s accuracy may decrease under poor lighting or if fruits are partially occluded.
* The project focuses on visual features only; other freshness indicators like smell or firmness are not considered.

**DATASET :**

**NAME**: Fresh and rotten fruits dataset

**SOURCE** : Kaggle / Self- captured images

**CLASSES** :

* Fresh apple
* Rotten apple
* Fresh banana
* Rotten banana
* Fresh oranges
* Rotten oranges

The dataset is divided into training (70%), validation (20%), and testing (10%) sets.

**SCRIPT :**

# ======================================

# STEP 1: LOAD DATASET FROM FOLDER

# ======================================

import os

import cv2

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.preprocessing import StandardScaler

from mpl\_toolkits.mplot3d import Axes3D

# Define the dataset directory

data\_dir = "/content/fruits\_dataset/train" # change if your folder is elsewhere

# Check folder structure

print("Classes found:", os.listdir(data\_dir))

# ======================================

# STEP 2: COUNT IMAGES IN EACH CLASS

# ======================================

classes = os.listdir(data\_dir)

data\_summary = {}

for cls in classes:

path = os.path.join(data\_dir, cls)

data\_summary[cls] = len(os.listdir(path))

# Create a DataFrame summary

df\_summary = pd.DataFrame(list(data\_summary.items()), columns=['Class', 'Image Count'])

print(df\_summary)

# Plot the counts

plt.figure(figsize=(8,5))

plt.bar(df\_summary['Class'], df\_summary['Image Count'], color='orange')

plt.title('Number of Images per Class')

plt.xticks(rotation=45)

plt.xlabel('Class Name')

plt.ylabel('Count')

plt.show()

# ======================================

# STEP 3: DISPLAY SAMPLE IMAGES

# ======================================

plt.figure(figsize=(10,8))

for i, cls in enumerate(classes[:6]):

path = os.path.join(data\_dir, cls)

img\_name = os.listdir(path)[0]

img = plt.imread(os.path.join(path, img\_name))

plt.subplot(2,3,i+1)

plt.imshow(img)

plt.title(cls)

plt.axis('off')

plt.suptitle("Sample Images from Each Class", fontsize=14)

plt.show()

# ======================================

# STEP 4: BASIC IMAGE FEATURE EXTRACTION

# ======================================

# Read one image to analyze features

sample\_path = os.path.join(data\_dir, classes[0], os.listdir(os.path.join(data\_dir, classes[0]))[0])

img = cv2.imread(sample\_path)

img\_rgb = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

print("Image Shape:", img\_rgb.shape)

print("Pixel Value Range:", img\_rgb.min(), "to", img\_rgb.max())

# Convert to grayscale

gray = cv2.cvtColor(img\_rgb, cv2.COLOR\_RGB2GRAY)

# Compute basic pixel statistics

mean\_val = np.mean(gray)

std\_val = np.std(gray)

print(f"Mean Pixel Intensity: {mean\_val:.2f}, Standard Deviation: {std\_val:.2f}")

# Visualize color channels

fig, ax = plt.subplots(1, 3, figsize=(12,4))

ax[0].imshow(img\_rgb[:,:,0], cmap='Reds')

ax[0].set\_title("Red Channel")

ax[1].imshow(img\_rgb[:,:,1], cmap='Greens')

ax[1].set\_title("Green Channel")

ax[2].imshow(img\_rgb[:,:,2], cmap='Blues')

ax[2].set\_title("Blue Channel")

for a in ax: a.axis('off')

plt.show()

# ======================================

# STEP 5: OPTIONAL - FEATURE SCALING DEMO

# ======================================

# Create small feature array [Mean, Std]

features = np.array([[mean\_val, std\_val]])

scaler = StandardScaler()

scaled\_features = scaler.fit\_transform(features)

print("Scaled Feature Values:", scaled\_features)

# ======================================

# STEP 6: OPTIONAL - 3D VISUALIZATION (DEMO)

# ======================================

# Simulated feature points for 3D plot

np.random.seed(0)

X = np.random.rand(20, 3)

fig = plt.figure(figsize=(6,5))

ax = fig.add\_subplot(111, projection='3d')

ax.scatter(X[:,0], X[:,1], X[:,2], color='green')

ax.set\_title("3D Visualization Example")

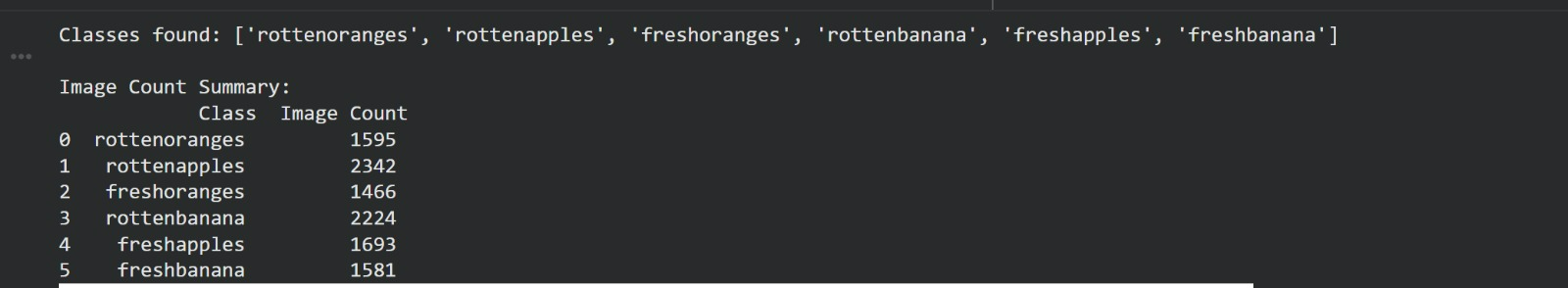
ax.set\_xlabel("Feature 1")

ax.set\_ylabel("Feature 2")

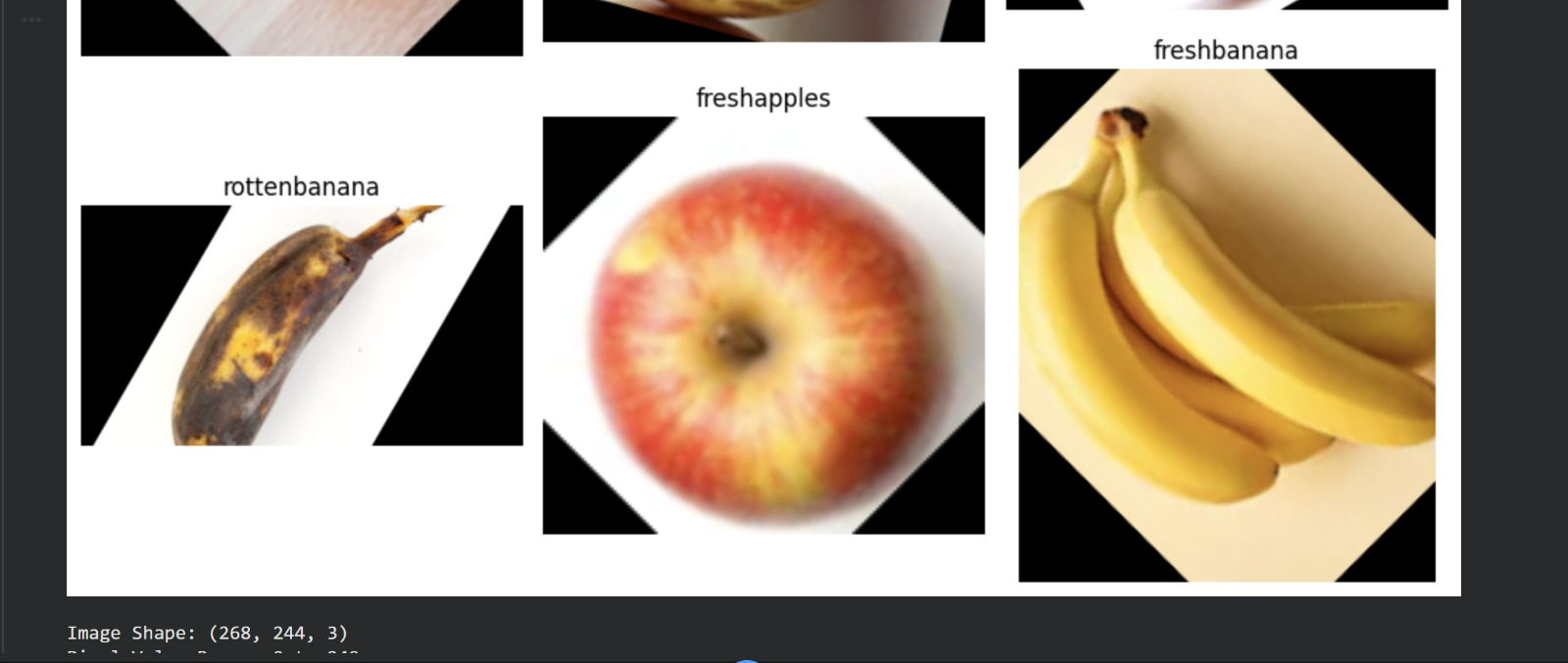
ax.set\_zlabel("Feature 3")

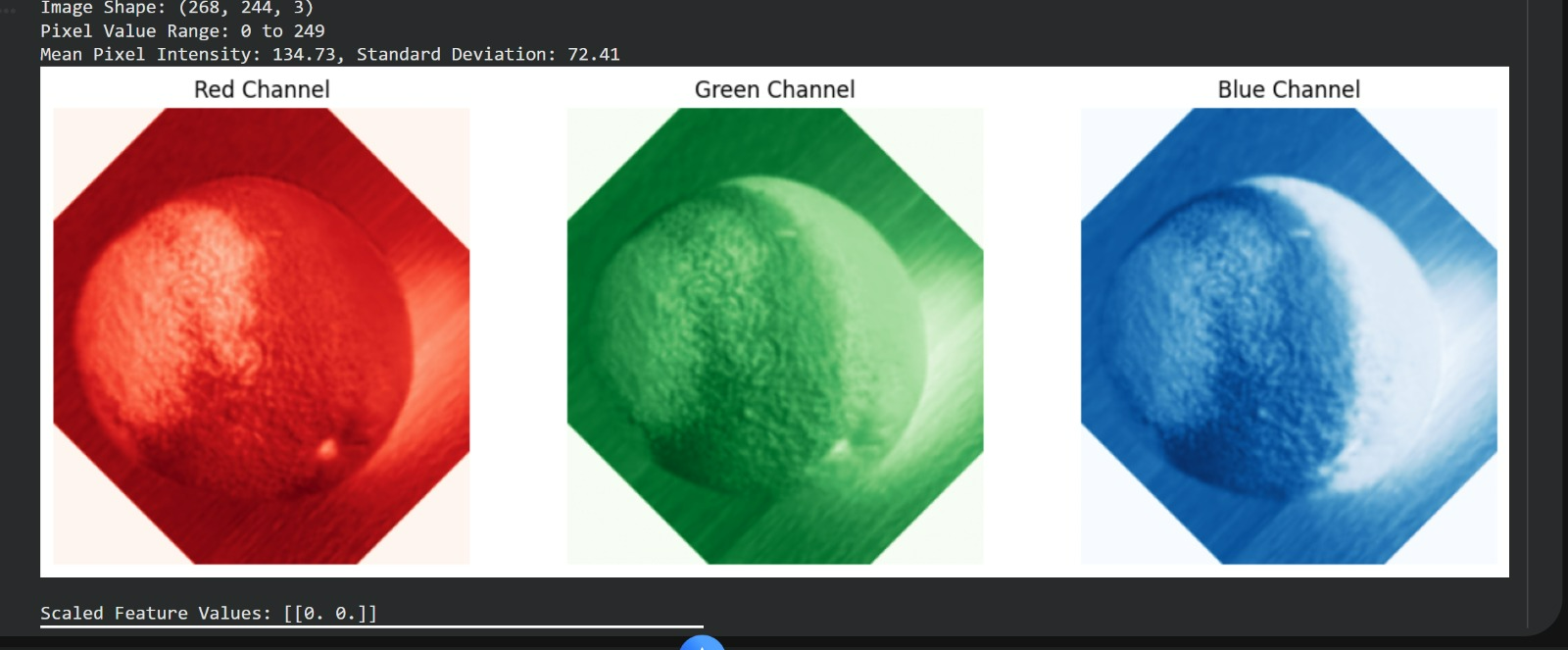
plt.show()

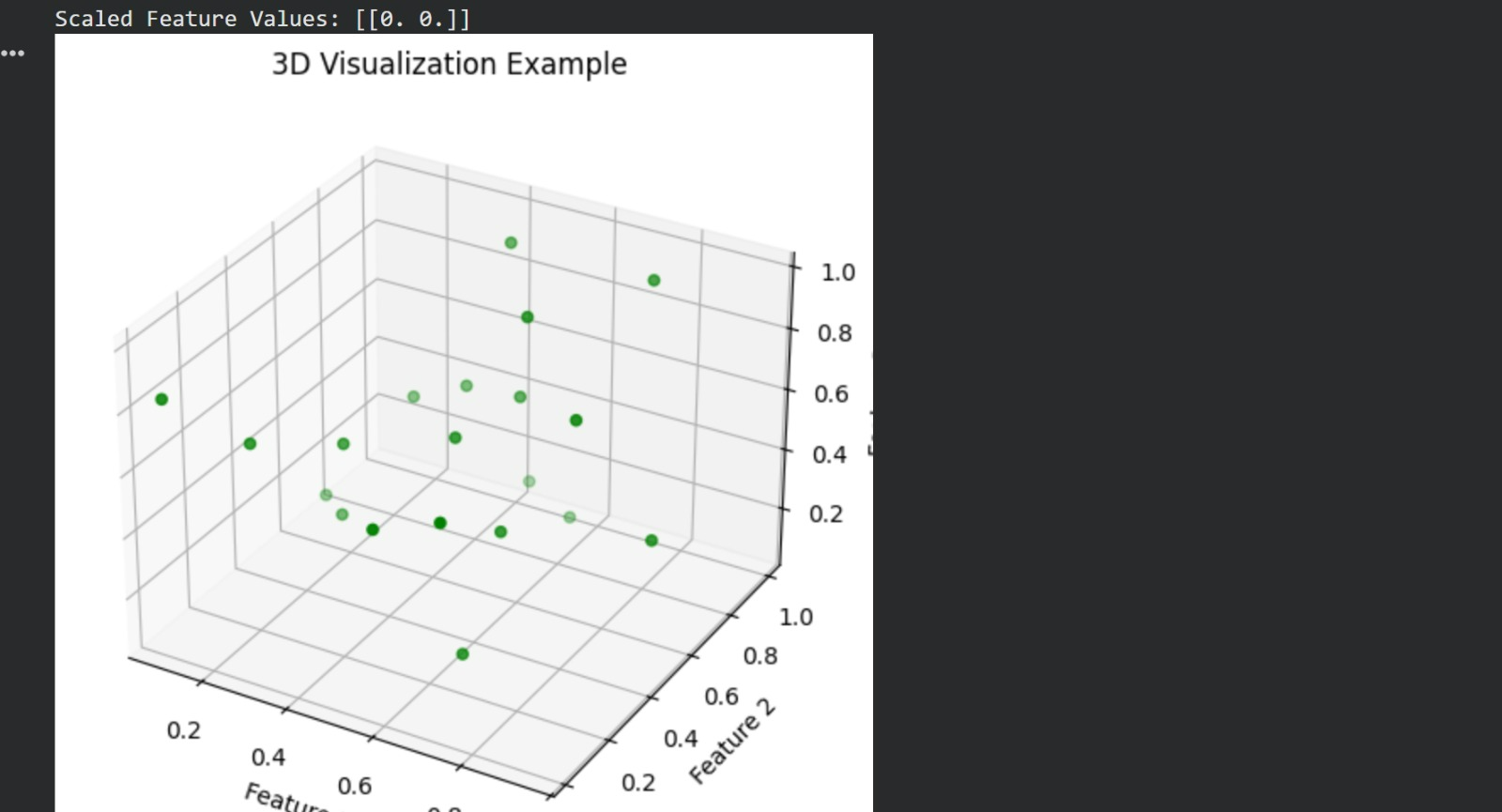
**OBSERVATION :**

****

****







**PYTHON FUNCTIONS :**

