SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

DATE	20-02-2026
TEAM ID	LTVIP2026TMIDS90622
PROJECT NAME	SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES
MAXIMUM MARKS	2 MARKS

6.2 DATA COLLECTION

Collect the data-set:

It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

Activity 1: Download the dataset

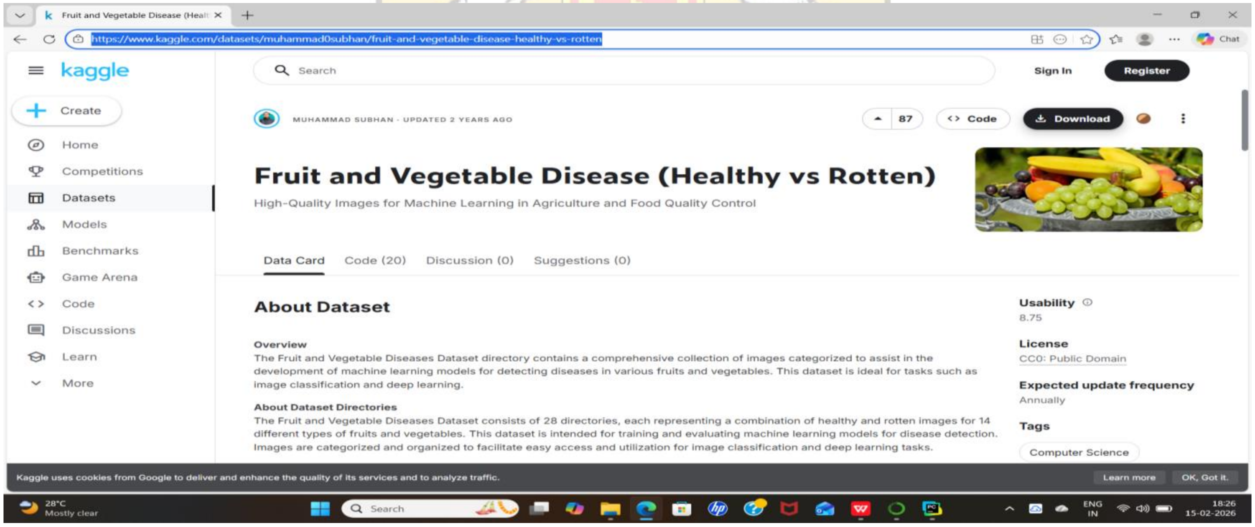
There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.In this project, we have used 28 classes of fruits and vegetables data. This data is downloaded from kaggle.com or can be connected by using API. Please refer to the link given below to download the dataset.

Link: Dataset

[Fruit and Vegetable Disease (Healthy vs Rotten)]

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.

Note: There are several techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.



Fruit and Vegetable Disease (Healthy vs Rotten)

High-Quality Images for Machine Learning in Agriculture and Food Quality Control

About Dataset

Overview

The Fruit and Vegetable Diseases Dataset directory contains a comprehensive collection of images categorized to assist in the development of machine learning models for detecting diseases in various fruits and vegetables. This dataset is ideal for tasks such as image classification and deep learning.

About Dataset Directories

The Fruit and Vegetable Diseases Dataset consists of 28 directories, each representing a combination of healthy and rotten images for 14 different types of fruits and vegetables. This dataset is intended for training and evaluating machine learning models for disease detection. Images are categorized and organized to facilitate easy access and utilization for image classification and deep learning tasks.

Usability: 8.75

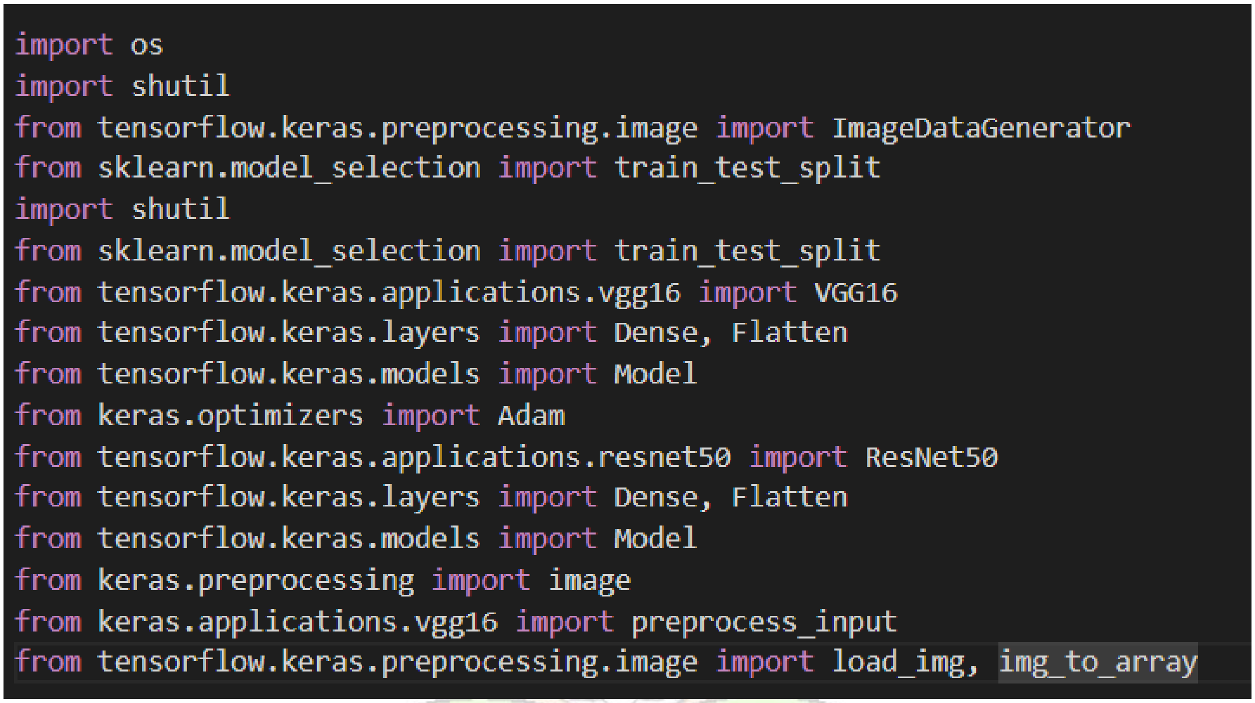
License: CC0: Public Domain

Expected update frequency: Annually

Tags: Computer Science

Kaggle uses cookies from Google to deliver and enhance the quality of its services and to analyze traffic.

IMPORT THE FOLLOWING LIBRARIES :

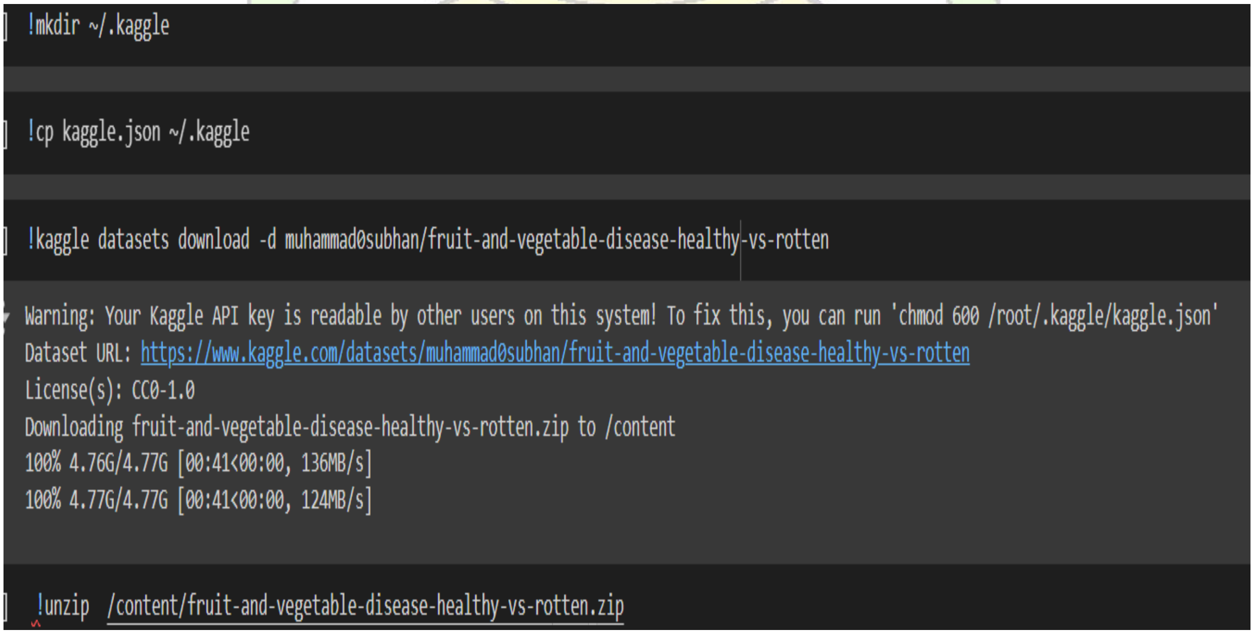


```
import os
import shutil
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
import shutil
from sklearn.model_selection import train_test_split
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Model
from keras.optimizers import Adam
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Model
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
from tensorflow.keras.preprocessing.image import load_img, img_to_array
```

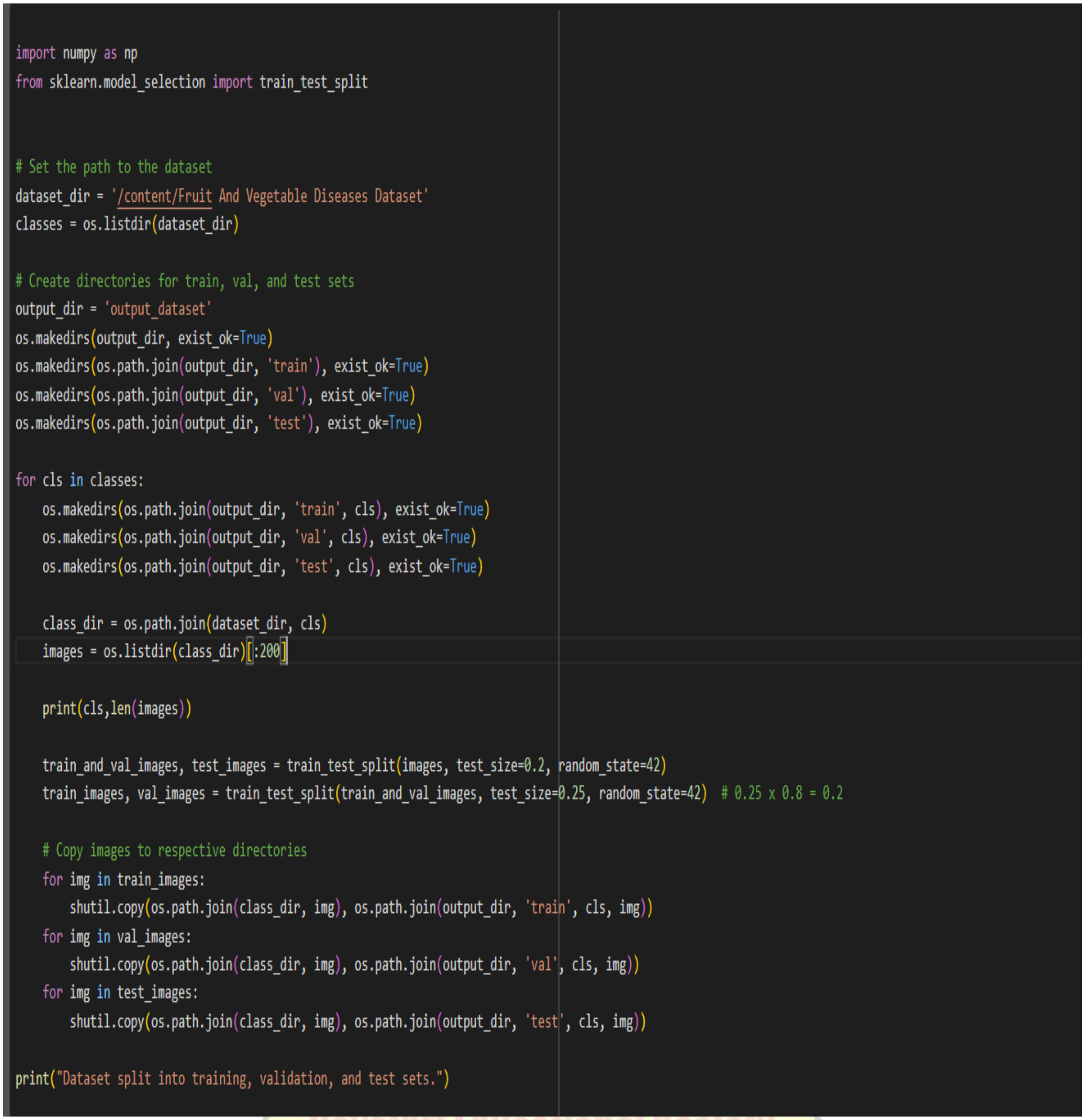
READ THE DATASET :

Our dataset format might be in .csv, excel files, .txt, .json, or zip files, etc. We can read the dataset with the help of pandas.

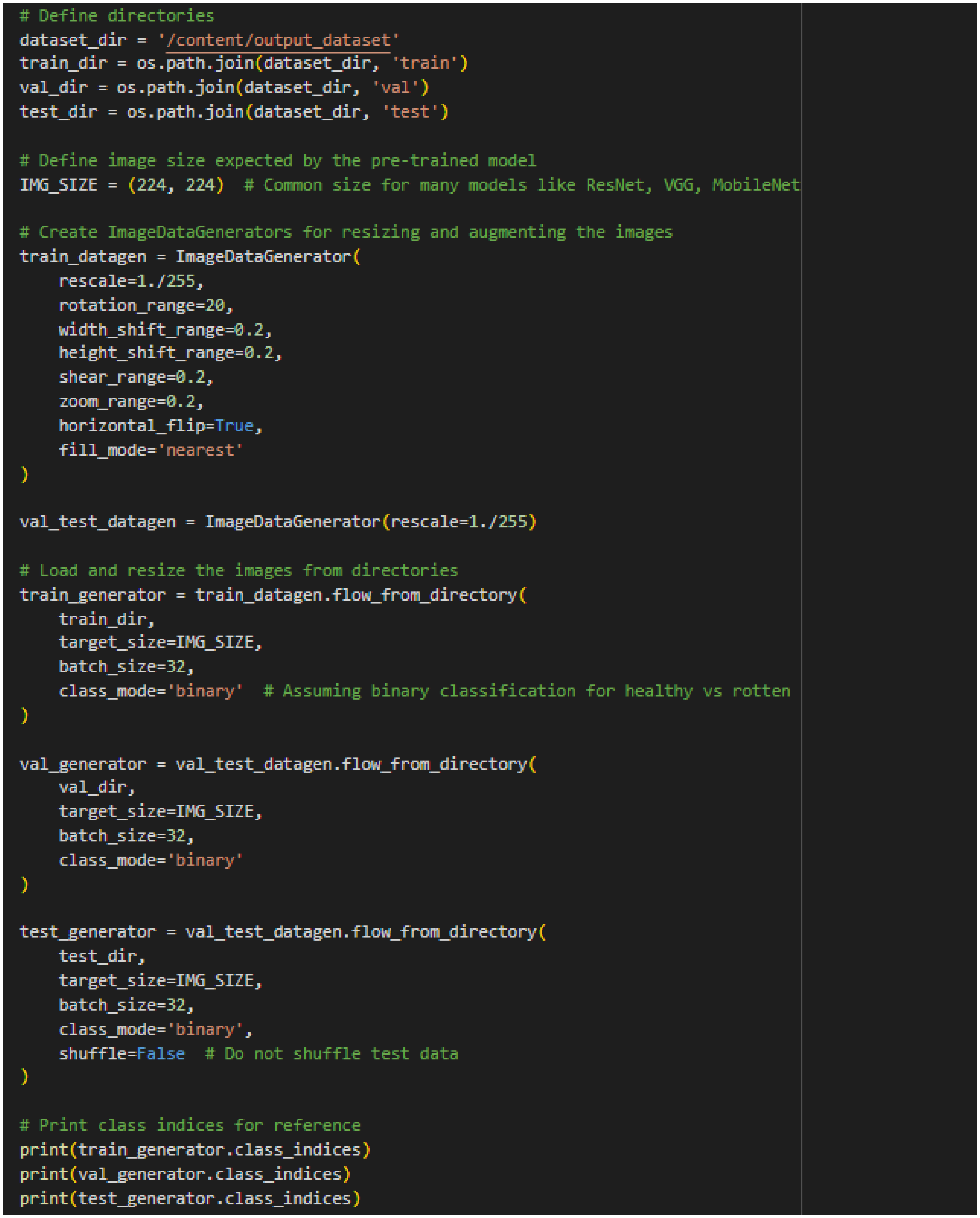
At first, unzip the data and convert it into a pandas data frame.



```
!mkdir ~/.kaggle

!cp kaggle.json ~/.kaggle

!kaggle datasets download -d muhammad0subhan/fruit-and-vegetable-disease-healthy-vs-rotten

Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run 'chmod 600 /root/.kaggle/kaggle.json'
Dataset URL: https://www.kaggle.com/datasets/muhammad0subhan/fruit-and-vegetable-disease-healthy-vs-rotten
License(s): CC0-1.0
Downloading fruit-and-vegetable-disease-healthy-vs-rotten.zip to /content
100% 4.76G/4.77G [00:41<00:00, 136MB/s]
100% 4.77G/4.77G [00:41<00:00, 124MB/s]

!unzip  /content/fruit-and-vegetable-disease-healthy-vs-rotten.zip
```



```
import numpy as np
from sklearn.model_selection import train_test_split


# Set the path to the dataset
dataset_dir = '/content/Fruit And Vegetable Diseases Dataset'
classes = os.listdir(dataset_dir)

# Create directories for train, val, and test sets
output_dir = 'output_dataset'
os.makedirs(output_dir, exist_ok=True)
os.makedirs(os.path.join(output_dir, 'train'), exist_ok=True)
os.makedirs(os.path.join(output_dir, 'val'), exist_ok=True)
os.makedirs(os.path.join(output_dir, 'test'), exist_ok=True)

for cls in classes:
    os.makedirs(os.path.join(output_dir, 'train', cls), exist_ok=True)
    os.makedirs(os.path.join(output_dir, 'val', cls), exist_ok=True)
    os.makedirs(os.path.join(output_dir, 'test', cls), exist_ok=True)

    class_dir = os.path.join(dataset_dir, cls)
    images = os.listdir(class_dir)[:200]

    print(cls,len(images))

    train_and_val_images, test_images = train_test_split(images, test_size=0.2, random_state=42)
    train_images, val_images = train_test_split(train_and_val_images, test_size=0.25, random_state=42)  # 0.25 x 0.8 = 0.2

    # Copy images to respective directories
    for img in train_images:
        shutil.copy(os.path.join(class_dir, img), os.path.join(output_dir, 'train', cls, img))
    for img in val_images:
        shutil.copy(os.path.join(class_dir, img), os.path.join(output_dir, 'val', cls, img))
    for img in test_images:
        shutil.copy(os.path.join(class_dir, img), os.path.join(output_dir, 'test', cls, img))

print("Dataset split into training, validation, and test sets.")
```



```
# Define directories
dataset_dir = '/content/output_dataset'
train_dir = os.path.join(dataset_dir, 'train')
val_dir = os.path.join(dataset_dir, 'val')
test_dir = os.path.join(dataset_dir, 'test')

# Define image size expected by the pre-trained model
IMG_SIZE = (224, 224)  # Common size for many models like ResNet, VGG, MobileNet

# Create ImageDataGenerators for resizing and augmenting the images
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

val_test_datagen = ImageDataGenerator(rescale=1./255)

# Load and resize the images from directories
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=IMG_SIZE,
    batch_size=32,
    class_mode='binary'  # Assuming binary classification for healthy vs rotten
)

val_generator = val_test_datagen.flow_from_directory(
    val_dir,
    target_size=IMG_SIZE,
    batch_size=32,
    class_mode='binary'
)

test_generator = val_test_datagen.flow_from_directory(
    test_dir,
    target_size=IMG_SIZE,
    batch_size=32,
    class_mode='binary',
    shuffle=False  # Do not shuffle test data
)

# Print class indices for reference
print(train_generator.class_indices)
print(val_generator.class_indices)
print(test_generator.class_indices)
```