UNIVERSITI MALAYSIA PAHANG AL-SULTAN ABDULLAH

GROUP PROJECT
BSD2513 ARTIFICIAL INTELLIGENCE
SEMESTER II 2022/2023

TITLE:

PALMSCAN: SMART MONITORING FOR PALM PLANTATION QUALITY

PREPARED FOR:
DR KU MUHAMMAD NAIM KU KHALIF

GROUP NAME: FARMTECH

MATRIC ID	NAME	SECTION
SD22016	SRI SHAMNEE SAI A/P RAJAH	02G
SD22041	TANUSHALANI A/P MUNIANDY	02G
SD22065	VIENOSHA A/P SELVARAJU	02G
SD22063	VINITA A/P GNYANASUNTHARAM	02G
SD22013	AQILAH MAISARAH BINTI AZIZI	01G

TABLE OF CONTENT

EXECUTIVE SUMMARY

The focus of this project is to develop an AI-powered oil palm scanner to boost the oil palm industry in Malaysia. Since oil palm exportation has become the significance to Malaysia’s economy so the need to improve the harvesting process is important. By accelerating the checking and quality analysis process, the suggested solution seeks to achieve high productivity and economic benefits.

The main problem issued by this project is the low efficacy and lack of accuracy of manual inspection. Mistakes made by humans and the inefficient nature of manual assessments contribute to a process delay and risk of quality problems. To solve these problems, we are going to introduce AI scanners in this field, so the time taken will be less by shortening the process and maintained quality standards.

We exported the data in image form from Roboflow. We classify it into two main paths: the happy path, it consists only of the images of ripe oil palm fruits while the second path is the challenging path, it consists of under ripe, unripe, over ripe, and empty oil palm fruits. The AI model will be trained with those data of images to achieve greater accuracy in determining and observing the quality of the oil palms and will help in Malaysia’s economic growth as it simplifies the export procedures and will decrease the cost of manpower.

SUMMARY OF THE PROJECT CONTEXT AND OBJECTIVES

In this project, we will work on several intricate, connected tasks to create a complex application that uses image recognition to determine the freshness of palm oil crops. First, we will collect a large collection of photos of palm oil fruit and carefully classify each one using maturity categories as ripe, unripe, underripe, overripe, and empty. To guarantee the robustness of the model, this dataset will be varied and representative of different ripening phases and climatic circumstances. Next, we will concentrate on classifying the maturity of palm oil fruits using an appropriate machine learning model. To achieve this, experimentation with various methods will be necessary to maximize the accuracy and performance of the model. Using a different dataset, we will evaluate the model using cross-validation and other methods to make sure the model is resilient and generalizable. Integration of the trained model into a user-friendly graphical user interface (GUI) will be done. The application will handle image processing.

Objectives:

- Develop an automated system that correctly classifies palm oil fruits into the appropriate categories and integrates it into a graphical user interface (GUI) .
- Reduce the dependence on human inspection to increase the accuracy and efficiency of the palm oil harvesting and processing process.
- Increasing total industry productivity and product quality in the palm oil sector

METHODOLOGY

3.1 Data Collection and preparation

To develop an accurate and reliable palm fruit recognition system, we collected and prepared image datasets. The team members search on the internet to collect various fruit pictures of different stages of palm fruit to create a comprehensive dataset.

The dataset includes photos, which enables the system to handle various kinds of data. The images collected have different stages of palm fruit, which are ripe, empty, unripe, under ripe and over ripe. The system can be trained on these photos to improve its performance in identifying masks in static images. To further enhance the functionality of the system, a real-time streaming system was developed. The system can identify the stage of palm fruit in real time, enabling the team to identify whether it is ready to harvest or not.

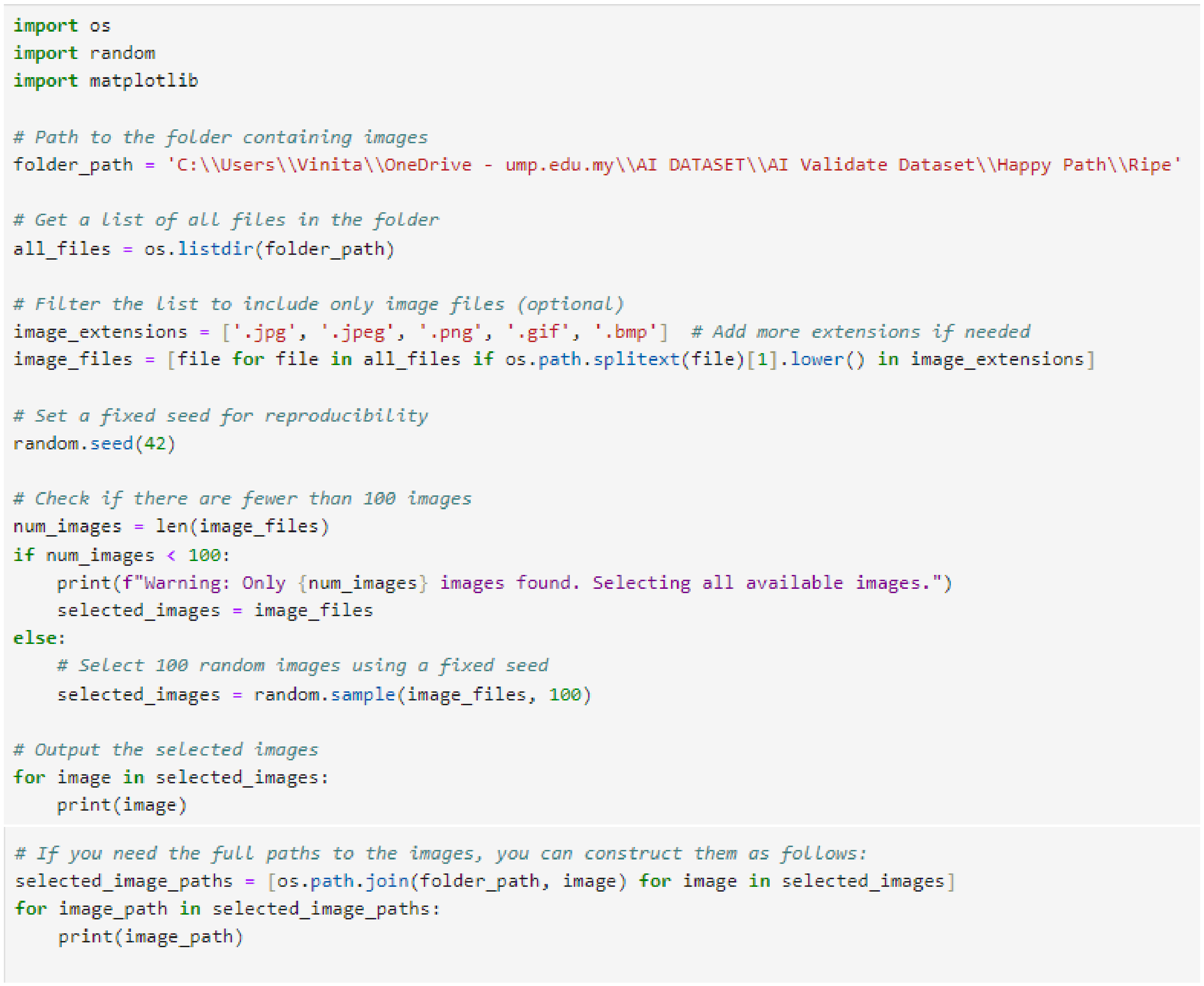
The collected photos are then carefully divided according to the classes. We analyze each picture one by one and put them into respective classes accordingly. Then, we prepared and used to train palm fruit detection using a teachable machine. We just insert 100 random pictures selected from each stage of fruit and insert them into a teachable machine. Thanks to the teachable machine it can train the data in fractions of seconds and provide an accuracy table and confusion matrix. Then we use code to valid and test data with the remaining pictures and get accurate confusion matrix with F1 score.

The dataset, along with the annotations, is made available for further research and development purposes. It can be accessed through a Google Drive link provided by the team, ensuring transparency and reproducibility in the development of palm fruit detection system.

By leveraging this curated dataset, the project aims to train models that are robust, accurate, and able to detect palm fruit in images and live video. The diversity of the dataset in terms of fruit, demographics and environment factors contributes to the system's ability to handle various scenarios and generalize well to unseen data.

3.2 Coding of project without GUI

Firstly, we create a folder where a folder has Happy path and Challenging path. In the happy path we have ripe fruit pictures. In challenging path we have Empty, Over Ripe, under ripe and Unripe. We use Jupyter Notebook to create coding that can choose random 100 pictures from those folders.

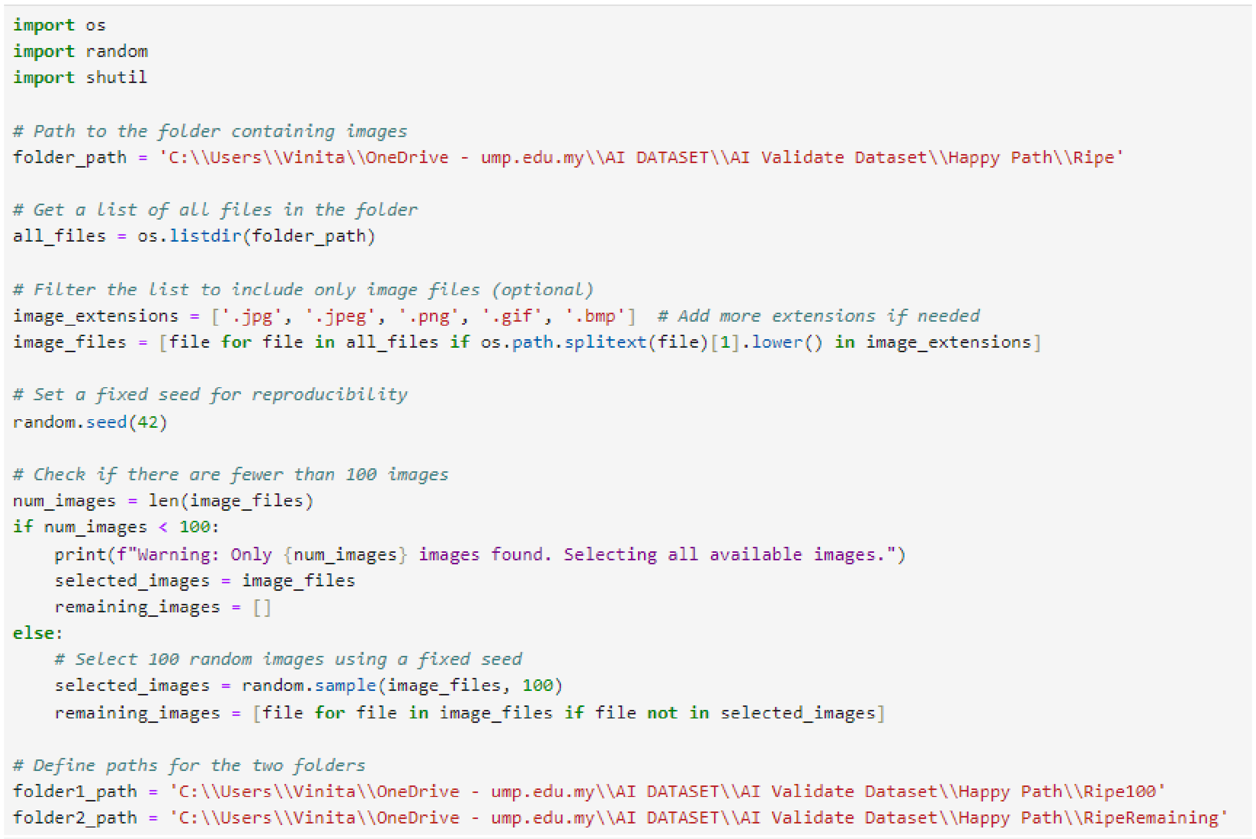


```python
import os
import random
import matplotlib

# Path to the folder containing images
folder_path = 'C:\\Users\\Vinita\\OneDrive - ump.edu.my\\AI DATASET\\AI Validate Dataset\\Happy Path\\Ripe'

# Get a list of all files in the folder
all_files = os.listdir(folder_path)

# Filter the list to include only image files (optional)
image_extensions = ['.jpg', '.jpeg', '.png', '.gif', '.bmp']  # Add more extensions if needed
image_files = [file for file in all_files if os.path.splitext(file)[1].lower() in image_extensions]

# Set a fixed seed for reproducibility
random.seed(42)

# Check if there are fewer than 100 images
num_images = len(image_files)
if num_images < 100:
    print(f"Warning: Only {num_images} images found. Selecting all available images.")
    selected_images = image_files
else:
    # Select 100 random images using a fixed seed
    selected_images = random.sample(image_files, 100)

# Output the selected images
for image in selected_images:
    print(image)
```

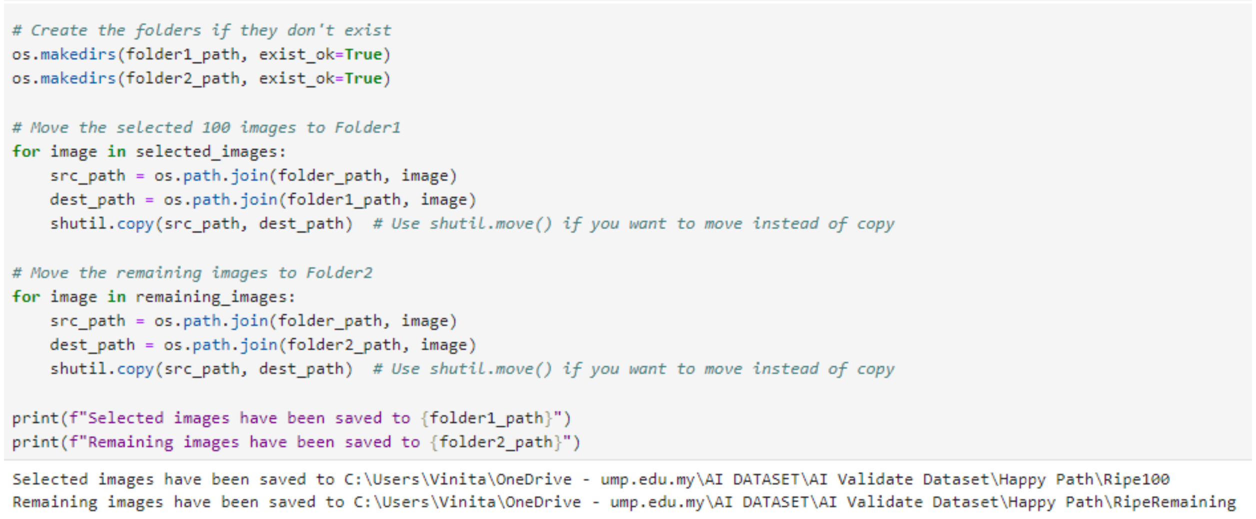
```python
# If you need the full paths to the images, you can construct them as follows:
selected_image_paths = [os.path.join(folder_path, image) for image in selected_images]
for image_path in selected_image_paths:
    print(image_path)
```

This one is the code that we use separate the ripe pictures from the folder and create two folders from it and do folder for remaining and the random 100.

This code organizes a set of image files from a specified folder into two separate folders. It begins by defining the path to the folder containing the images and then retrieves a list of all files in that folder. The code filters this list to include only image files based on a set of common image file extensions. Using a fixed random seed for reproducibility, it checks if there are fewer than 100 images; if so, it selects all available images. If there are 100 or more images, it randomly selects 100 images and separates the remaining images. The code then creates two new folders if they don't already exist: one for the selected 100 images and another for the remaining images. Finally, it copies the selected 100 images to the first folder and the remaining images to the second folder, ensuring the original files remain intact. It concludes by printing messages indicating where the selected and remaining images have been saved.



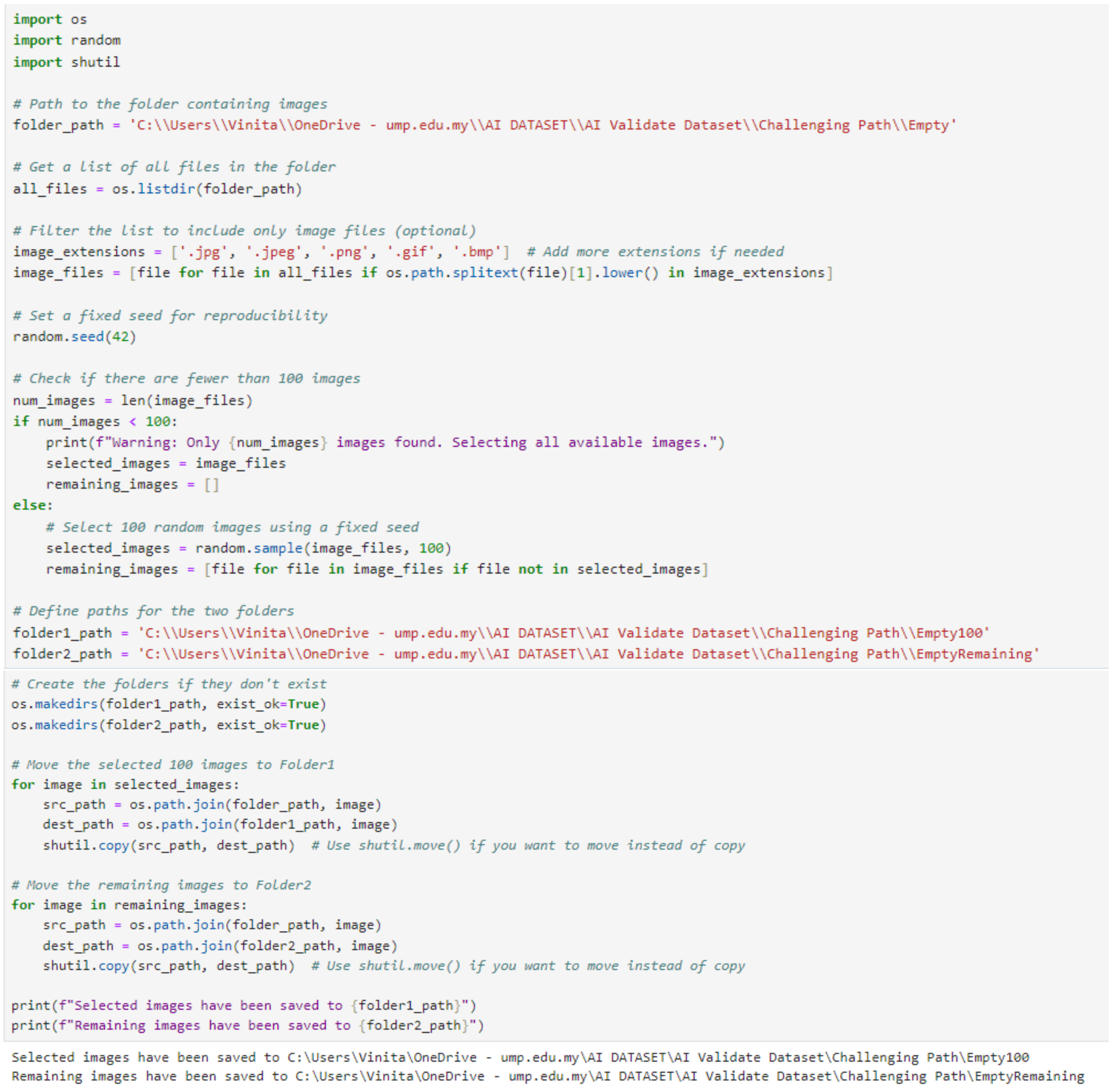
```python
import os
import random
import shutil

# Path to the folder containing images
folder_path = 'C:\\Users\\Vinita\\OneDrive - ump.edu.my\\AI DATASET\\AI Validate Dataset\\Happy Path\\Ripe'

# Get a list of all files in the folder
all_files = os.listdir(folder_path)

# Filter the list to include only image files (optional)
image_extensions = ['.jpg', '.jpeg', '.png', '.gif', '.bmp']  # Add more extensions if needed
image_files = [file for file in all_files if os.path.splitext(file)[1].lower() in image_extensions]

# Set a fixed seed for reproducibility
random.seed(42)

# Check if there are fewer than 100 images
num_images = len(image_files)
if num_images < 100:
    print(f"Warning: Only {num_images} images found. Selecting all available images.")
    selected_images = image_files
    remaining_images = []
else:
    # Select 100 random images using a fixed seed
    selected_images = random.sample(image_files, 100)
    remaining_images = [file for file in image_files if file not in selected_images]

# Define paths for the two folders
folder1_path = 'C:\\Users\\Vinita\\OneDrive - ump.edu.my\\AI DATASET\\AI Validate Dataset\\Happy Path\\Ripe100'
folder2_path = 'C:\\Users\\Vinita\\OneDrive - ump.edu.my\\AI DATASET\\AI Validate Dataset\\Happy Path\\RipeRemaining'
```



```python
# Create the folders if they don't exist
os.makedirs(folder1_path, exist_ok=True)
os.makedirs(folder2_path, exist_ok=True)

# Move the selected 100 images to Folder1
for image in selected_images:
    src_path = os.path.join(folder_path, image)
    dest_path = os.path.join(folder1_path, image)
    shutil.copy(src_path, dest_path)  # Use shutil.move() if you want to move instead of copy

# Move the remaining images to Folder2
for image in remaining_images:
    src_path = os.path.join(folder_path, image)
    dest_path = os.path.join(folder2_path, image)
    shutil.copy(src_path, dest_path)  # Use shutil.move() if you want to move instead of copy

print(f"Selected images have been saved to {folder1_path}")
print(f"Remaining images have been saved to {folder2_path}")
```

```
Selected images have been saved to C:\Users\Vinita\OneDrive - ump.edu.my\AI DATASET\AI Validate Dataset\Happy Path\Ripe100
Remaining images have been saved to C:\Users\Vinita\OneDrive - ump.edu.my\AI DATASET\AI Validate Dataset\Happy Path\RipeRemaining
```

This one is the code that we use to separate the empty pictures from the folder and create two folders from it and do folder for remaining and the random 100.

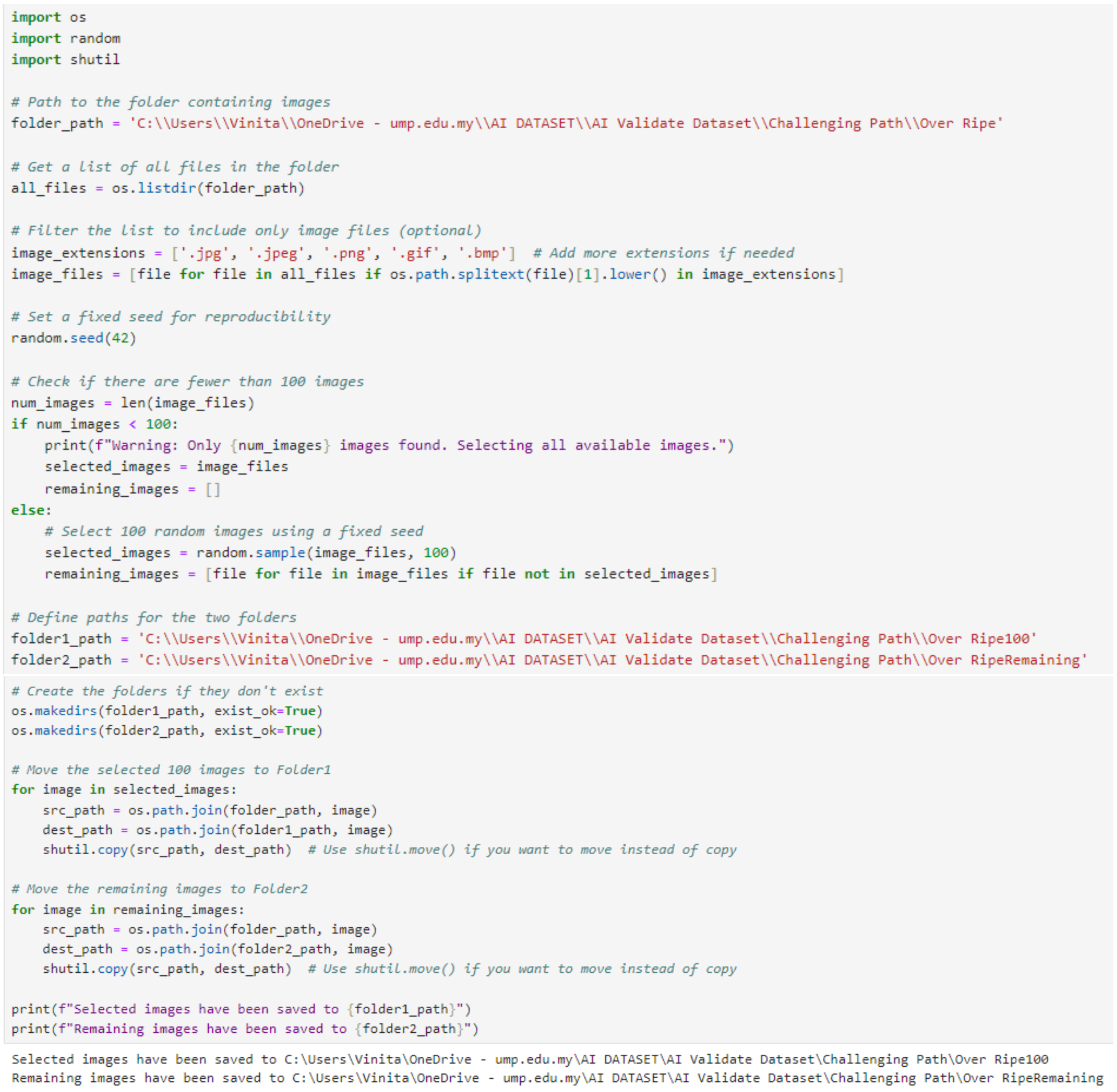


```python
import os
import random
import shutil

# Path to the folder containing images
folder_path = 'C:\\Users\\Vinita\\OneDrive - ump.edu.my\\AI DATASET\\AI Validate Dataset\\Challenging Path\\Empty'

# Get a list of all files in the folder
all_files = os.listdir(folder_path)

# Filter the list to include only image files (optional)
image_extensions = ['.jpg', '.jpeg', '.png', '.gif', '.bmp']  # Add more extensions if needed
image_files = [file for file in all_files if os.path.splitext(file)[1].lower() in image_extensions]

# Set a fixed seed for reproducibility
random.seed(42)

# Check if there are fewer than 100 images
num_images = len(image_files)
if num_images < 100:
    print(f"Warning: Only {num_images} images found. Selecting all available images.")
    selected_images = image_files
    remaining_images = []
else:
    # Select 100 random images using a fixed seed
    selected_images = random.sample(image_files, 100)
    remaining_images = [file for file in image_files if file not in selected_images]

# Define paths for the two folders
folder1_path = 'C:\\Users\\Vinita\\OneDrive - ump.edu.my\\AI DATASET\\AI Validate Dataset\\Challenging Path\\Empty100'
folder2_path = 'C:\\Users\\Vinita\\OneDrive - ump.edu.my\\AI DATASET\\AI Validate Dataset\\Challenging Path\\EmptyRemaining'
```

```python
# Create the folders if they don't exist
os.makedirs(folder1_path, exist_ok=True)
os.makedirs(folder2_path, exist_ok=True)

# Move the selected 100 images to Folder1
for image in selected_images:
    src_path = os.path.join(folder_path, image)
    dest_path = os.path.join(folder1_path, image)
    shutil.copy(src_path, dest_path)  # Use shutil.move() if you want to move instead of copy

# Move the remaining images to Folder2
for image in remaining_images:
    src_path = os.path.join(folder_path, image)
    dest_path = os.path.join(folder2_path, image)
    shutil.copy(src_path, dest_path)  # Use shutil.move() if you want to move instead of copy

print(f"Selected images have been saved to {folder1_path}")
print(f"Remaining images have been saved to {folder2_path}")
```

```
Selected images have been saved to C:\Users\Vinita\OneDrive - ump.edu.my\AI DATASET\AI Validate Dataset\Challenging Path\Empty100
Remaining images have been saved to C:\Users\Vinita\OneDrive - ump.edu.my\AI DATASET\AI Validate Dataset\Challenging Path\EmptyRemaining
```

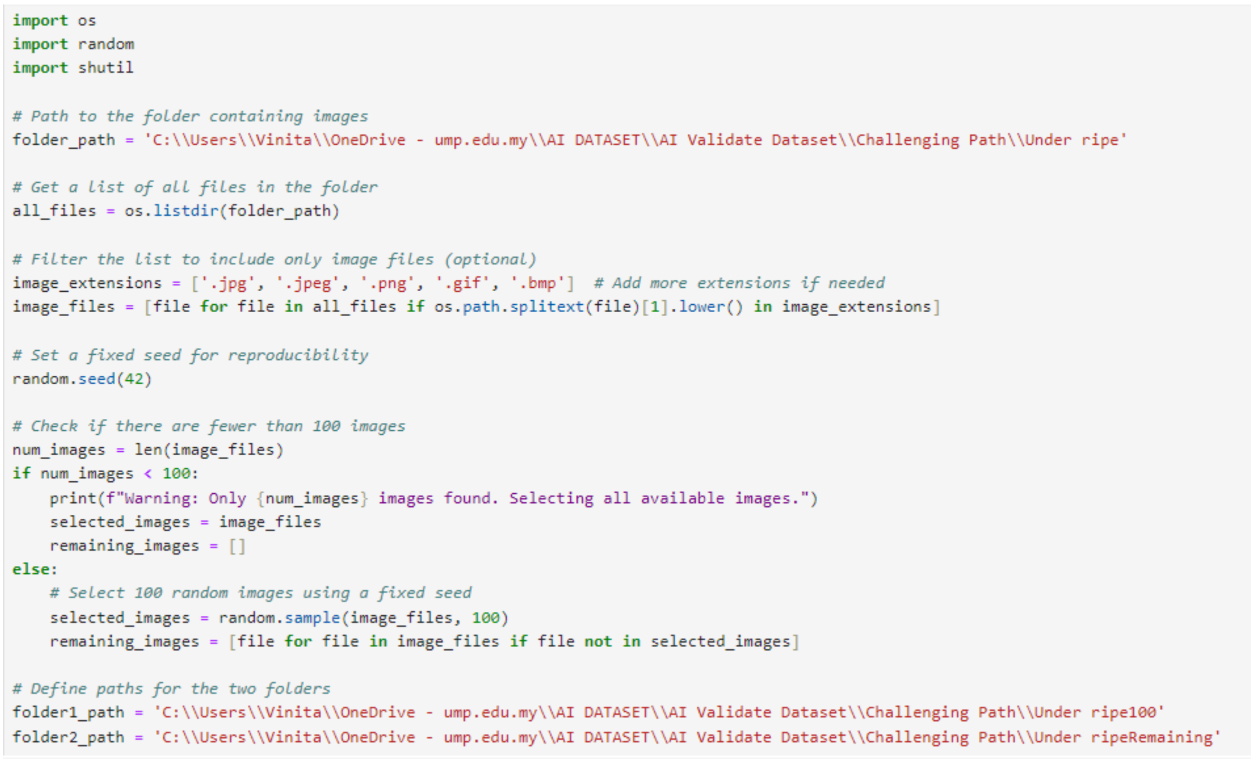
This one is the code that we use separate the Over ripe pictures from the folder and create two folders from it and do folder for remaining and the random 100.



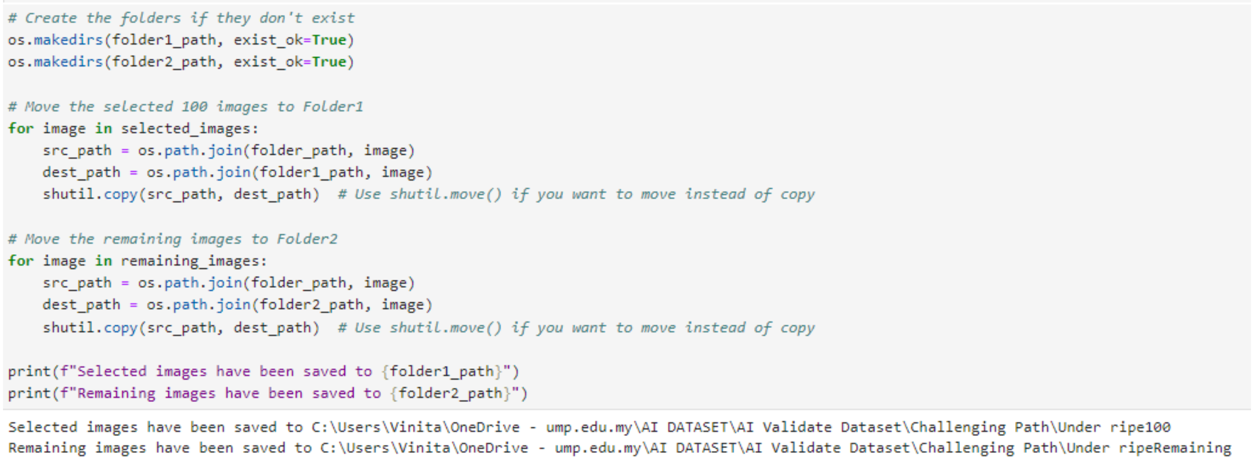
```python
import os
import random
import shutil

# Path to the folder containing images
folder_path = 'C:\\Users\\Vinita\\OneDrive - ump.edu.my\\AI DATASET\\AI Validate Dataset\\Challenging Path\\Over Ripe'

# Get a list of all files in the folder
all_files = os.listdir(folder_path)

# Filter the list to include only image files (optional)
image_extensions = ['.jpg', '.jpeg', '.png', '.gif', '.bmp']  # Add more extensions if needed
image_files = [file for file in all_files if os.path.splitext(file)[1].lower() in image_extensions]

# Set a fixed seed for reproducibility
random.seed(42)

# Check if there are fewer than 100 images
num_images = len(image_files)
if num_images < 100:
    print(f"Warning: Only {num_images} images found. Selecting all available images.")
    selected_images = image_files
    remaining_images = []
else:
    # Select 100 random images using a fixed seed
    selected_images = random.sample(image_files, 100)
    remaining_images = [file for file in image_files if file not in selected_images]

# Define paths for the two folders
folder1_path = 'C:\\Users\\Vinita\\OneDrive - ump.edu.my\\AI DATASET\\AI Validate Dataset\\Challenging Path\\Over Ripe100'
folder2_path = 'C:\\Users\\Vinita\\OneDrive - ump.edu.my\\AI DATASET\\AI Validate Dataset\\Challenging Path\\Over RipeRemaining'
# Create the folders if they don't exist
os.makedirs(folder1_path, exist_ok=True)
os.makedirs(folder2_path, exist_ok=True)

# Move the selected 100 images to Folder1
for image in selected_images:
    src_path = os.path.join(folder_path, image)
    dest_path = os.path.join(folder1_path, image)
    shutil.copy(src_path, dest_path)  # Use shutil.move() if you want to move instead of copy

# Move the remaining images to Folder2
for image in remaining_images:
    src_path = os.path.join(folder_path, image)
    dest_path = os.path.join(folder2_path, image)
    shutil.copy(src_path, dest_path)  # Use shutil.move() if you want to move instead of copy

print(f"Selected images have been saved to {folder1_path}")
print(f"Remaining images have been saved to {folder2_path}")
```

```
Selected images have been saved to C:\Users\Vinita\OneDrive - ump.edu.my\AI DATASET\AI Validate Dataset\Challenging Path\Over Ripe100
Remaining images have been saved to C:\Users\Vinita\OneDrive - ump.edu.my\AI DATASET\AI Validate Dataset\Challenging Path\Over RipeRemaining
```

This one is the code that we use separate the Under ripe pictures from the folder and create two folders from it and do folder for remaining and the random 100.



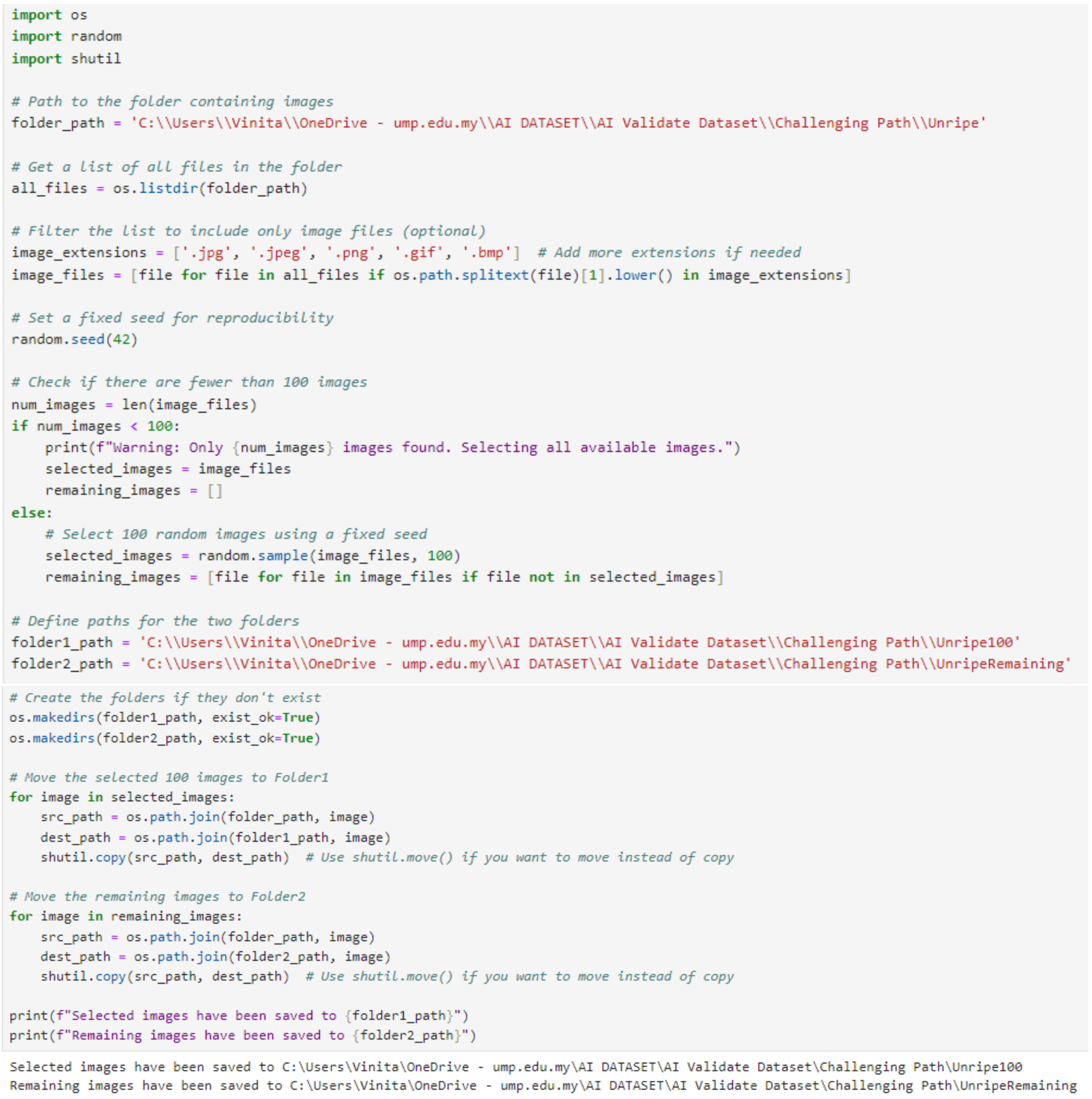
```python
import os
import random
import shutil

# Path to the folder containing images
folder_path = 'C:\\Users\\Vinita\\OneDrive - ump.edu.my\\AI DATASET\\AI Validate Dataset\\Challenging Path\\Under ripe'

# Get a list of all files in the folder
all_files = os.listdir(folder_path)

# Filter the list to include only image files (optional)
image_extensions = ['.jpg', '.jpeg', '.png', '.gif', '.bmp']  # Add more extensions if needed
image_files = [file for file in all_files if os.path.splitext(file)[1].lower() in image_extensions]

# Set a fixed seed for reproducibility
random.seed(42)

# Check if there are fewer than 100 images
num_images = len(image_files)
if num_images < 100:
    print(f"Warning: Only {num_images} images found. Selecting all available images.")
    selected_images = image_files
    remaining_images = []
else:
    # Select 100 random images using a fixed seed
    selected_images = random.sample(image_files, 100)
    remaining_images = [file for file in image_files if file not in selected_images]

# Define paths for the two folders
folder1_path = 'C:\\Users\\Vinita\\OneDrive - ump.edu.my\\AI DATASET\\AI Validate Dataset\\Challenging Path\\Under ripe100'
folder2_path = 'C:\\Users\\Vinita\\OneDrive - ump.edu.my\\AI DATASET\\AI Validate Dataset\\Challenging Path\\Under ripeRemaining'
```



```python
# Create the folders if they don't exist
os.makedirs(folder1_path, exist_ok=True)
os.makedirs(folder2_path, exist_ok=True)

# Move the selected 100 images to Folder1
for image in selected_images:
    src_path = os.path.join(folder_path, image)
    dest_path = os.path.join(folder1_path, image)
    shutil.copy(src_path, dest_path)  # Use shutil.move() if you want to move instead of copy

# Move the remaining images to Folder2
for image in remaining_images:
    src_path = os.path.join(folder_path, image)
    dest_path = os.path.join(folder2_path, image)
    shutil.copy(src_path, dest_path)  # Use shutil.move() if you want to move instead of copy

print(f"Selected images have been saved to {folder1_path}")
print(f"Remaining images have been saved to {folder2_path}")
```

```
Selected images have been saved to C:\Users\Vinita\OneDrive - ump.edu.my\AI DATASET\AI Validate Dataset\Challenging Path\Under ripe100
Remaining images have been saved to C:\Users\Vinita\OneDrive - ump.edu.my\AI DATASET\AI Validate Dataset\Challenging Path\Under ripeRemaining
```

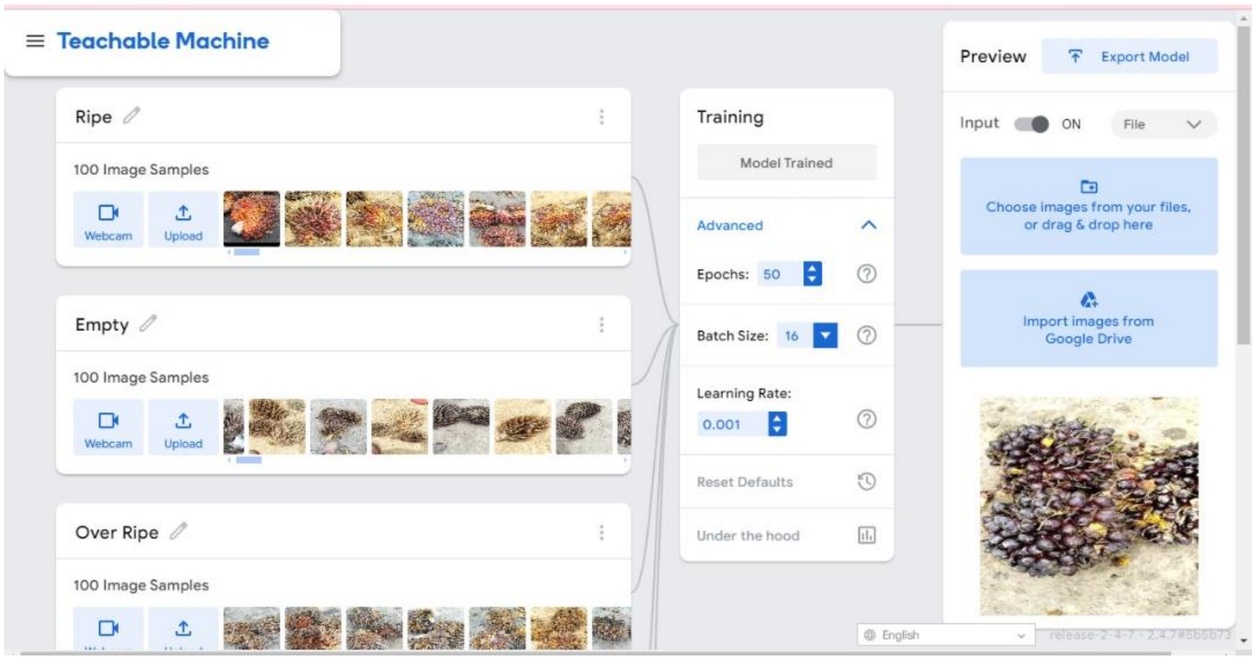
This one is the code that we use separate the Unripe pictures from the folder and create two folders from it and do folder for remaining and the random 100.



```python
import os
import random
import shutil

# Path to the folder containing images
folder_path = 'C:\\Users\\Vinita\\OneDrive - ump.edu.my\\AI DATASET\\AI Validate Dataset\\Challenging Path\\Unripe'

# Get a list of all files in the folder
all_files = os.listdir(folder_path)

# Filter the list to include only image files (optional)
image_extensions = ['.jpg', '.jpeg', '.png', '.gif', '.bmp']  # Add more extensions if needed
image_files = [file for file in all_files if os.path.splitext(file)[1].lower() in image_extensions]

# Set a fixed seed for reproducibility
random.seed(42)

# Check if there are fewer than 100 images
num_images = len(image_files)
if num_images < 100:
    print(f"Warning: Only {num_images} images found. Selecting all available images.")
    selected_images = image_files
    remaining_images = []
else:
    # Select 100 random images using a fixed seed
    selected_images = random.sample(image_files, 100)
    remaining_images = [file for file in image_files if file not in selected_images]

# Define paths for the two folders
folder1_path = 'C:\\Users\\Vinita\\OneDrive - ump.edu.my\\AI DATASET\\AI Validate Dataset\\Challenging Path\\Unripe100'
folder2_path = 'C:\\Users\\Vinita\\OneDrive - ump.edu.my\\AI DATASET\\AI Validate Dataset\\Challenging Path\\UnripeRemaining'
# Create the folders if they don't exist
os.makedirs(folder1_path, exist_ok=True)
os.makedirs(folder2_path, exist_ok=True)

# Move the selected 100 images to Folder1
for image in selected_images:
    src_path = os.path.join(folder_path, image)
    dest_path = os.path.join(folder1_path, image)
    shutil.copy(src_path, dest_path)  # Use shutil.move() if you want to move instead of copy

# Move the remaining images to Folder2
for image in remaining_images:
    src_path = os.path.join(folder_path, image)
    dest_path = os.path.join(folder2_path, image)
    shutil.copy(src_path, dest_path)  # Use shutil.move() if you want to move instead of copy

print(f"Selected images have been saved to {folder1_path}")
print(f"Remaining images have been saved to {folder2_path}")
```

```
Selected images have been saved to C:\Users\Vinita\OneDrive - ump.edu.my\AI DATASET\AI Validate Dataset\Challenging Path\Unripe100
Remaining images have been saved to C:\Users\Vinita\OneDrive - ump.edu.my\AI DATASET\AI Validate Dataset\Challenging Path\UnripeRemaining
```

Furthermore, we use Teachable Machine to train our data using the random 100 pictures that we choose for each classes of picture.


Teachable Machine

Ripe

100 Image Samples

Empty

100 Image Samples

Over Ripe

100 Image Samples

Training

Model Trained

Advanced

Epochs: 50

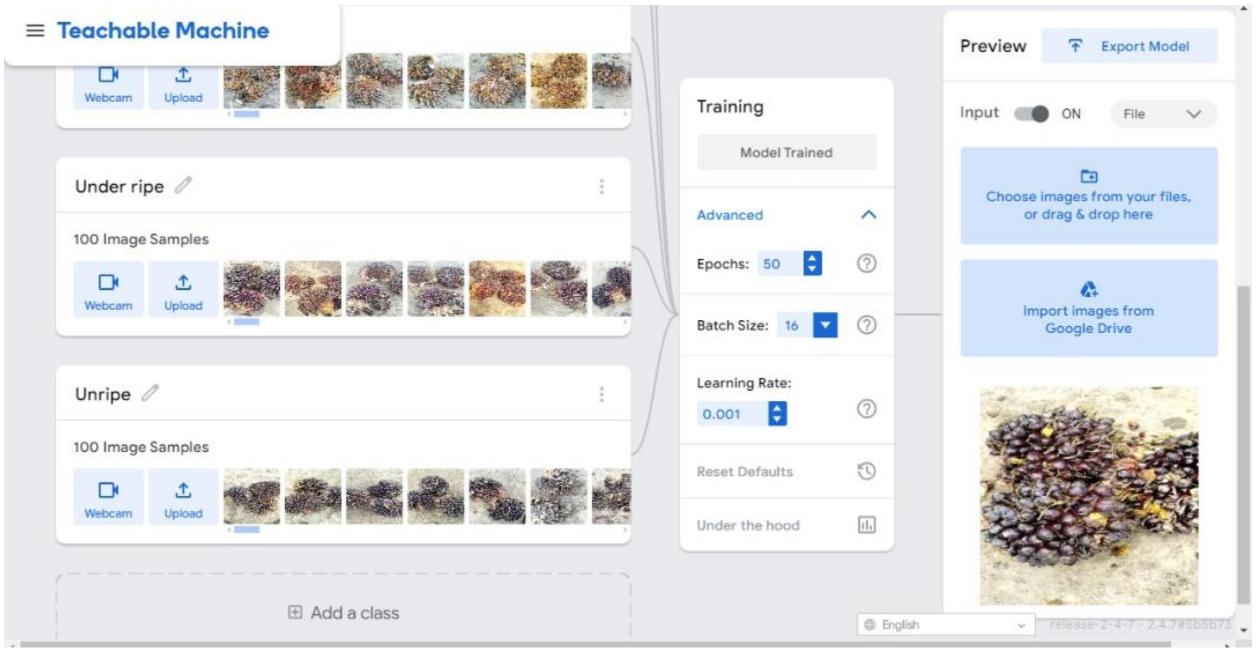
Batch Size: 16

Learning Rate: 0.001

Preview

Input ON File

Webcam
Upload
Reset Defaults
Under the hood
Export Model
Choose images from your files, or drag & drop here
Import images from Google Drive
English


Teachable Machine

Under ripe

100 Image Samples

Unripe

100 Image Samples

Training

Model Trained

Advanced

Epochs: 50

Batch Size: 16

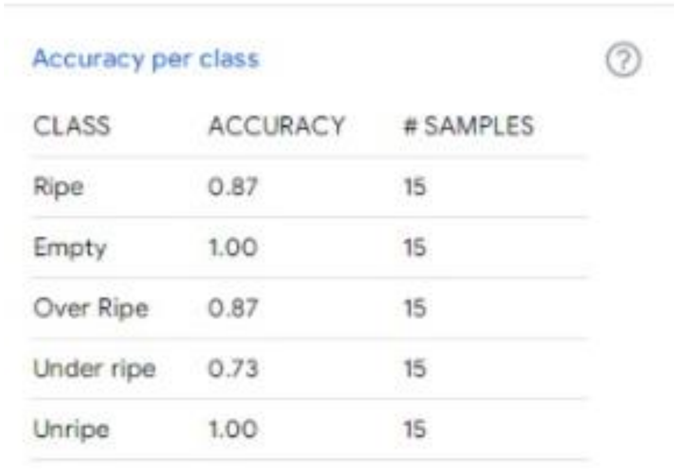
Learning Rate: 0.001

Preview

Input ON File

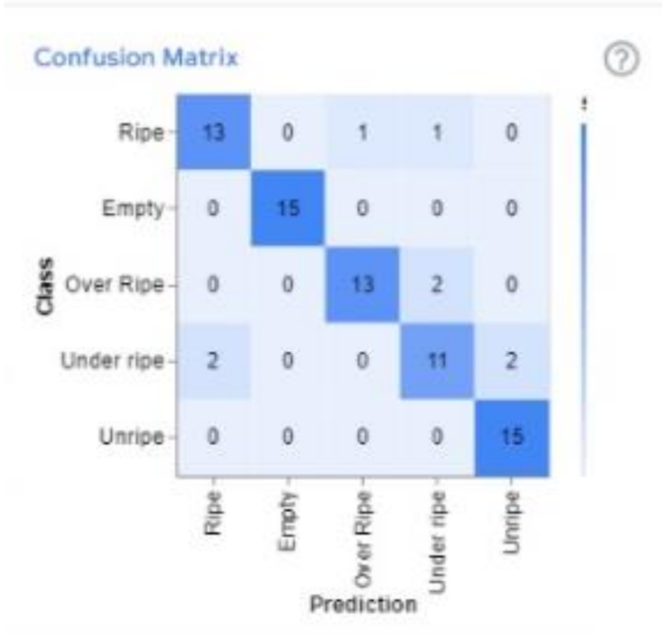
Webcam
Upload
Add a class
Reset Defaults
Under the hood
Export Model
Choose images from your files, or drag & drop here
Import images from Google Drive
English

After we train the data, we get an accuracy table and confusion matrix from teachable machine itself. But we want to make our validation and test data more accurate. Not only that we also never get an F! Score here, so we did some coding to know the F1 score.



Accuracy per class

CLASS	ACCURACY	# SAMPLES
Ripe	0.87	15
Empty	1.00	15
Over Ripe	0.87	15
Under ripe	0.73	15
Unripe	1.00	15

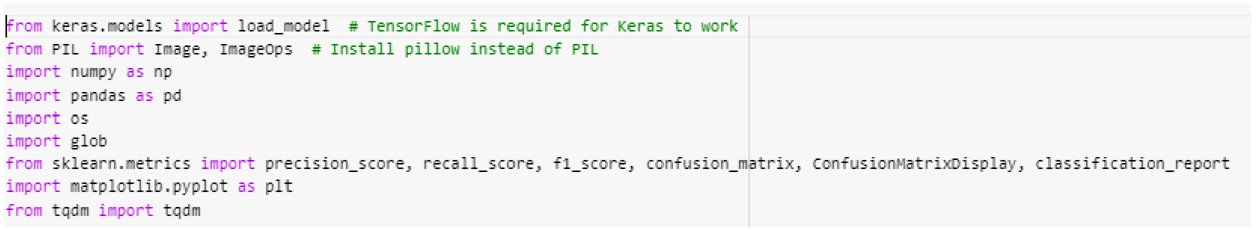


Confusion Matrix

Class \ Prediction	Ripe	Empty	Over Ripe	Under ripe	Unripe
Ripe	13	0	1	1	0
Empty	0	15	0	0	0
Over Ripe	0	0	13	2	0
Under ripe	2	0	0	11	2
Unripe	0	0	0	0	15

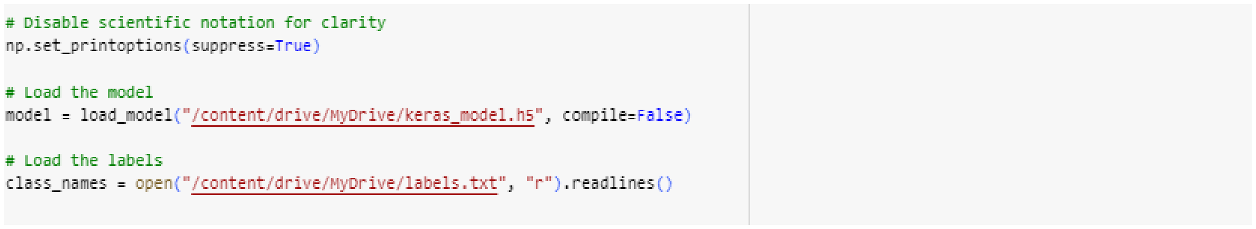
For validating and testing data we use the remaining pictures that we saved in different folders. For that we download the keras model and label text from teachable machine and upload it in this code. So, there are a few libraries that should be used and make sure the libraries will work completely. The coding used by 9 libraries: -

- keras.model : Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow. The keras.models module contains functions and classes to create, configure, train, and evaluate deep learning models.
- PIL : PIL is a library that adds image processing capabilities to your Python interpreter. It allows you to open, manipulate, and save many different images file formats.
- numpy : NumPy is a fundamental package for scientific computing with Python. It provides support for arrays, matrices, and many mathematical functions to operate on these data structures.
- pandas : Pandas is a powerful data manipulation and analysis library for Python. It provides data structures like Series (1-dimensional) and DataFrame (2-dimensional) that are efficient for handling and analyzing structured data.
- os : The os module provides a way of using operating system-dependent functionality, such as reading or writing to the filesystem, handling paths, and executing system commands.
- glob : The glob module finds all the pathnames matching a specified pattern according to the rules used by the Unix shell, although it operates independently of the shell.
- sklearn.metrices : sklearn.metrics is a module within Scikit-Learn, a machine learning library, that provides functions to measure and evaluate the performance of machine learning models.
- matplotlib.pyplot : matplotlib.pyplot is a collection of functions that make Matplotlib work like MATLAB. It provides a state-machine interface to the underlying plotting library in Matplotlib.
- tqdm : tqdm is a library that allows you to add a progress bar to your loops in Python with minimal code changes.



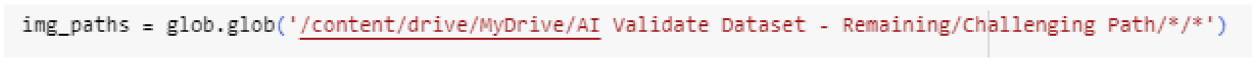
```
from keras.models import load_model  # TensorFlow is required for Keras to work
from PIL import Image, ImageOps  # Install pillow instead of PIL
import numpy as np
import pandas as pd
import os
import glob
from sklearn.metrics import precision_score, recall_score, f1_score, confusion_matrix, ConfusionMatrixDisplay, classification_report
import matplotlib.pyplot as plt
from tqdm import tqdm
```

Then, this code will load keras model that we download from the teachable machine to valid the data. We also use the label text file that been download from teachable machine as well.



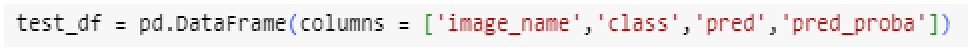
```
# Disable scientific notation for clarity
np.set_printoptions(suppress=True)

# Load the model
model = load_model("/content/drive/MyDrive/keras_model.h5", compile=False)

# Load the labels
class_names = open("/content/drive/MyDrive/labels.txt", "r").readlines()
```

This function is to open the dataset from google drive that we saved.



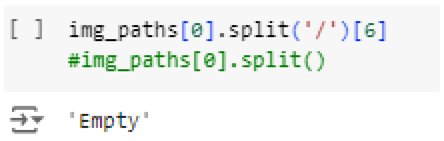
```
img_paths = glob.glob('/content/drive/MyDrive/AI Validate Dataset - Remaining/Challenging Path/*/*')
```

This code is to create a data frame of 4 attributes which is image name, class, prediction and prediction probability.



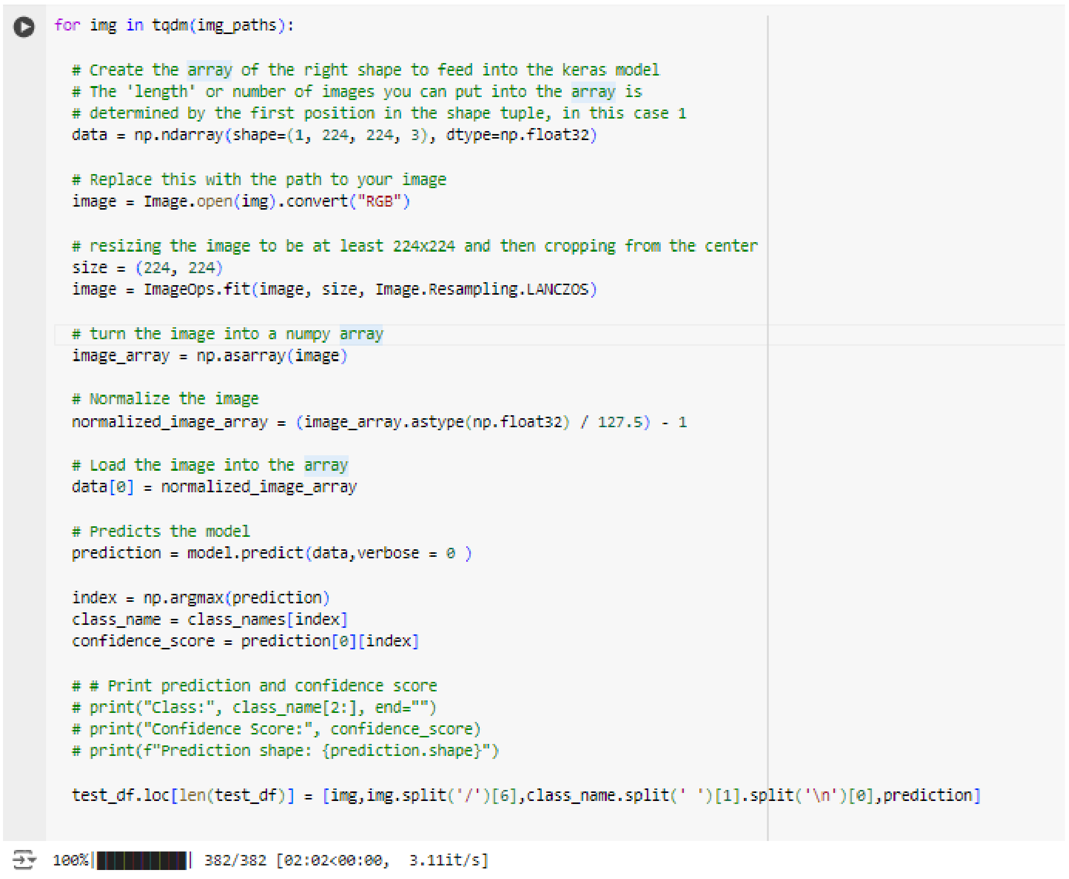
```
test_df = pd.DataFrame(columns = ['image_name','class','pred','pred_proba'])
```

This code checks if we took the data from the folder.



```
[ ]  img_paths[0].split('/')[6]
     #img_paths[0].split()

     'Empty'
```

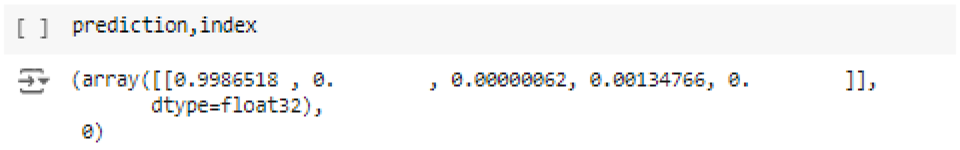
This code snippet processes a series of images and uses a pre-trained Keras model to predict their classes. For each image in the `img_paths` list, it first initializes a NumPy array `data` with the shape required by the Keras model (1, 224, 224, 3). It then loads each image, converts it to RGB, resizes it to 224x224 pixels, and ensures the image is centered and cropped correctly. The image is converted into a NumPy array, normalized by scaling its values to the range [-1, 1], and placed into the `data` array. The Keras model then makes a prediction on this processed image, outputting a set of probabilities for each class. The code identifies the class with the highest probability and records the class name and confidence score. Finally, it appends the image path, a specific part of the path, the predicted class name, and the raw prediction array to a DataFrame `test_df` for further analysis. This loop is wrapped in a tqdm progress bar to track its progress.



```
for img in tqdm(img_paths):

  # Create the array of the right shape to feed into the keras model
  # The 'length' or number of images you can put into the array is
  # determined by the first position in the shape tuple, in this case 1
  data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)

  # Replace this with the path to your image
  image = Image.open(img).convert("RGB")

  # resizing the image to be at least 224x224 and then cropping from the center
  size = (224, 224)
  image = ImageOps.fit(image, size, Image.Resampling.LANCZOS)

  # turn the image into a numpy array
  image_array = np.asarray(image)

  # Normalize the image
  normalized_image_array = (image_array.astype(np.float32) / 127.5) - 1

  # Load the image into the array
  data[0] = normalized_image_array

  # Predicts the model
  prediction = model.predict(data,verbose = 0 )

  index = np.argmax(prediction)
  class_name = class_names[index]
  confidence_score = prediction[0][index]

  # # Print prediction and confidence score
  # print("Class:", class_name[2:], end="")
  # print("Confidence Score:", confidence_score)
  # print(f"Prediction shape: {prediction.shape}")

  test_df.loc[len(test_df)] = [img,img.split('/')[6],class_name.split(' ')[1].split('\n')[0],prediction]
```

```
100%|██████████| 382/382 [02:02<00:00,  3.11it/s]
```

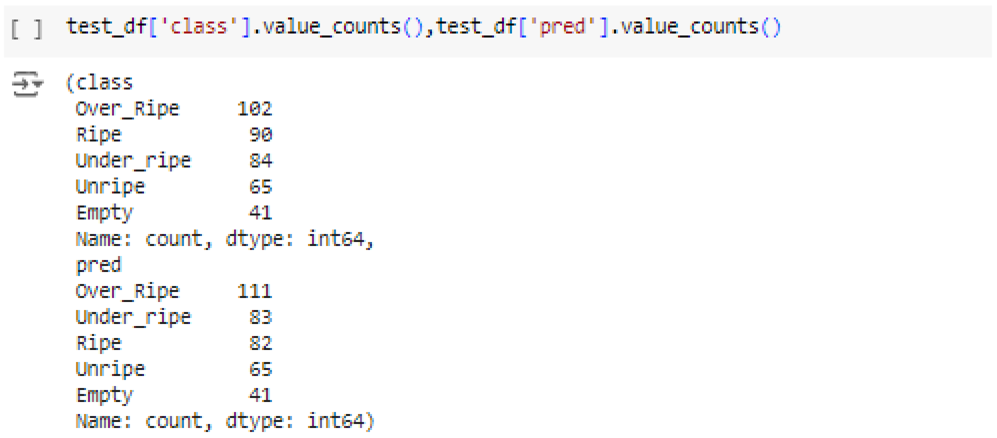
This function is to identify the prediction index which is the prediction for each class.



```
[ ]  prediction,index
```

```
(array([[0.9986518 , 0.        , 0.00000062, 0.00134766, 0.        ]],
      dtype=float32),
 0)
```

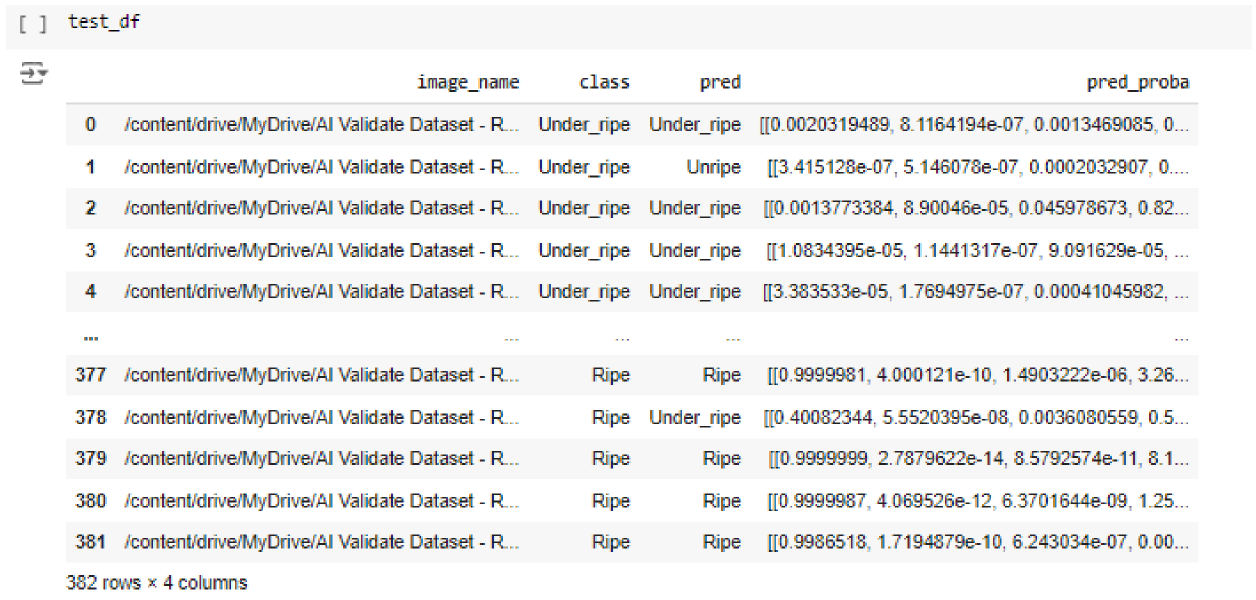
This function is used to test the data frame.



```
[ ]  test_df['class'].value_counts(),test_df['pred'].value_counts()
```

```
(class
 Over_Ripe     102
 Ripe           90
 Under_ripe     84
 Unripe         65
 Empty          41
 Name: count, dtype: int64,
 pred
 Over_Ripe     111
 Under_ripe     83
 Ripe           82
 Unripe         65
 Empty          41
 Name: count, dtype: int64)
```

This is the data frame that we created, it shows the image name, class, prediction and prediction probability.

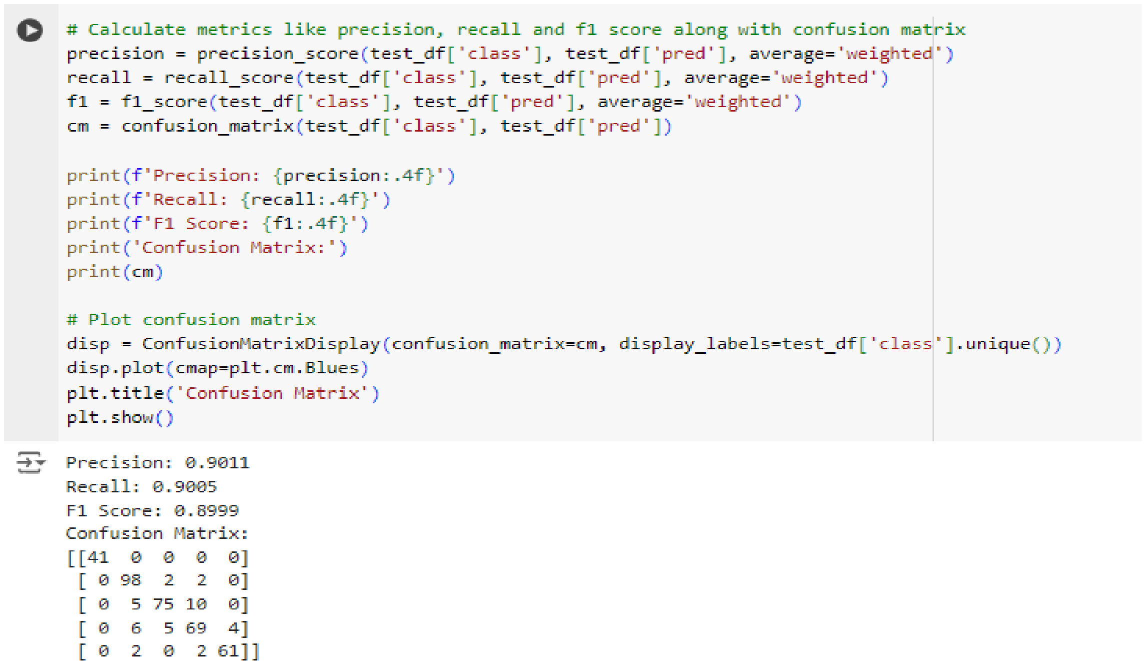


```
[ ]  test_df
```

	image_name	class	pred	pred_proba
0	/content/drive/MyDrive/AI Validate Dataset - R...	Under_ripe	Under_ripe	[[0.0020319489, 8.1164194e-07, 0.0013469085, 0...
1	/content/drive/MyDrive/AI Validate Dataset - R...	Under_ripe	Unripe	[[3.415128e-07, 5.146078e-07, 0.0002032907, 0....
2	/content/drive/MyDrive/AI Validate Dataset - R...	Under_ripe	Under_ripe	[[0.0013773384, 8.90046e-05, 0.045978673, 0.82...
3	/content/drive/MyDrive/AI Validate Dataset - R...	Under_ripe	Under_ripe	[[1.0834395e-05, 1.1441317e-07, 9.091629e-05, ...
4	/content/drive/MyDrive/AI Validate Dataset - R...	Under_ripe	Under_ripe	[[3.383533e-05, 1.7694975e-07, 0.00041045982, ...
...	...	...	...	...
377	/content/drive/MyDrive/AI Validate Dataset - R...	Ripe	Ripe	[[0.9999981, 4.000121e-10, 1.4903222e-06, 3.26...
378	/content/drive/MyDrive/AI Validate Dataset - R...	Ripe	Under_ripe	[[0.40082344, 5.5520395e-08, 0.0036080559, 0.5...
379	/content/drive/MyDrive/AI Validate Dataset - R...	Ripe	Ripe	[[0.9999999, 2.7879622e-14, 8.5792574e-11, 8.1...
380	/content/drive/MyDrive/AI Validate Dataset - R...	Ripe	Ripe	[[0.9999987, 4.069526e-12, 6.3701644e-09, 1.25...
381	/content/drive/MyDrive/AI Validate Dataset - R...	Ripe	Ripe	[[0.9986518, 1.7194879e-10, 6.243034e-07, 0.00...

382 rows × 4 columns

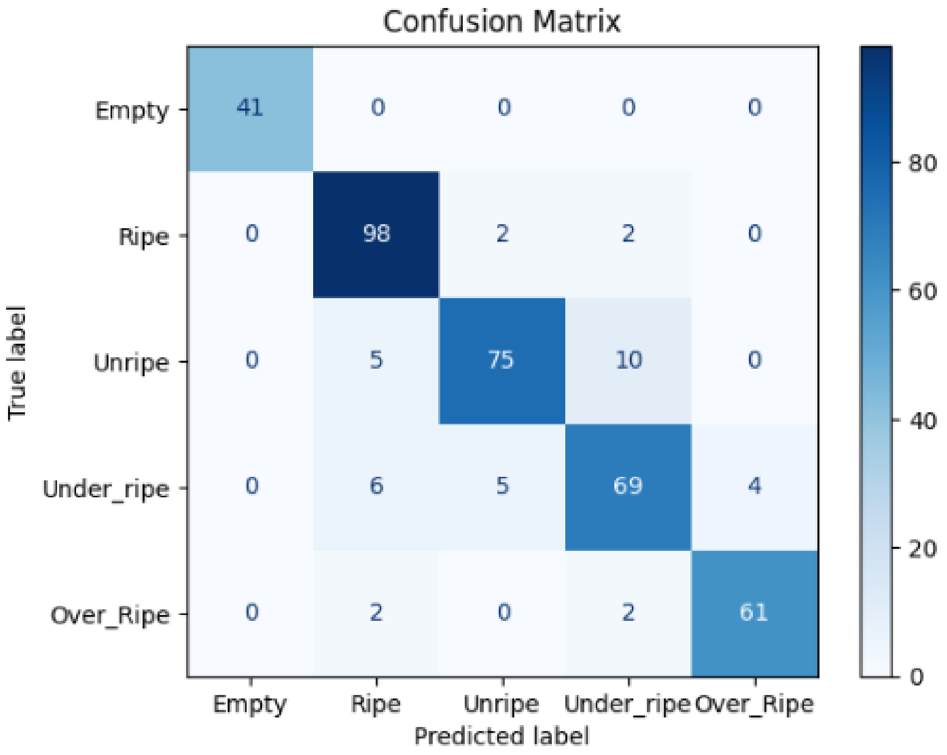
This code calculates and displays several classification metrics for a machine learning model's prediction. It first computes precision, recall, and F1 score using the `precision_score`, `recall_score`, and `f1_score` functions from the `sklearn.metrics` module, with the 'weighted' average parameter to account for class imbalance. These metrics are calculated based on the actual classes (`test_df['class']`) and the predicted classes (`test_df['pred']`). It then computes the confusion matrix, which provides a detailed breakdown of the model's performance by showing the counts of true positive, true negative, false positive, and false negative predictions for each class. The precision, recall, and F1 score are printed with four decimal places. The confusion matrix is visualized using the `ConfusionMatrixDisplay` class from `sklearn.metrics`, which displays the matrix as a color-coded plot with the class labels. This visual representation helps in understanding the model's performance across different classes. The plot is shown using Matplotlib's `plt.show()` function.



```python
# Calculate metrics like precision, recall and f1 score along with confusion matrix
precision = precision_score(test_df['class'], test_df['pred'], average='weighted')
recall = recall_score(test_df['class'], test_df['pred'], average='weighted')
f1 = f1_score(test_df['class'], test_df['pred'], average='weighted')
cm = confusion_matrix(test_df['class'], test_df['pred'])

print(f'Precision: {precision:.4f}')
print(f'Recall: {recall:.4f}')
print(f'F1 Score: {f1:.4f}')
print('Confusion Matrix:')
print(cm)

# Plot confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=test_df['class'].unique())
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.show()
```

```
Precision: 0.9011
Recall: 0.9005
F1 Score: 0.8999
Confusion Matrix:
[[41  0  0  0  0]
 [ 0 98  2  2  0]
 [ 0  5 75 10  0]
 [ 0  6  5 69  4]
 [ 0  2  0  2 61]]
```

This confusion matrix shows that empty picture is 100% accurate, ripe has 4 inaccurate, unripe has 57 accurate pictures and 15 inaccurate pictures, under ripe has 69 pictures while 15 pictures are inaccurate. For over ripe has 61 accurate pictures while 4 inaccurate pictures. The overall F1 score is 89.99%.



3.3 Coding of project with GUI

```
!pip install Pillow numpy keras
```

```
pip install gradio
```

To ensure that all required libraries are available for image processing, numerical operations, handling machine learning models and creating graphical user interfaces (GUIs). Specifically:

Pillow: For image processing.

numpy: For handling numeric data and arrays.

keras: To load and use a trained machine learning model.

gradio: To create a web-based GUI for user interaction.

```
import numpy as np
from keras.models import load_model
from PIL import Image, ImageOps
import gradio as gr
```

```
# Load the model
model_path = "/content/keras_model.h5"  # Replace with your model path
model = load_model(model_path, compile=False)
```

```
# Load the class names
labels_path = "/content/labels.txt"  # Replace with your labels path
with open(labels_path, "r") as file:
    class_names = [line.strip() for line in file]
```

- Importing the necessary libraries enables various operations such as image manipulation, numerical calculations and interfacing with pre-trained models.
- Loading a pre-trained model from the given path ensures that we can use it to make predictions.

- Reading class names from a text file helps map model output to human-readable labels, important for interpreting results.

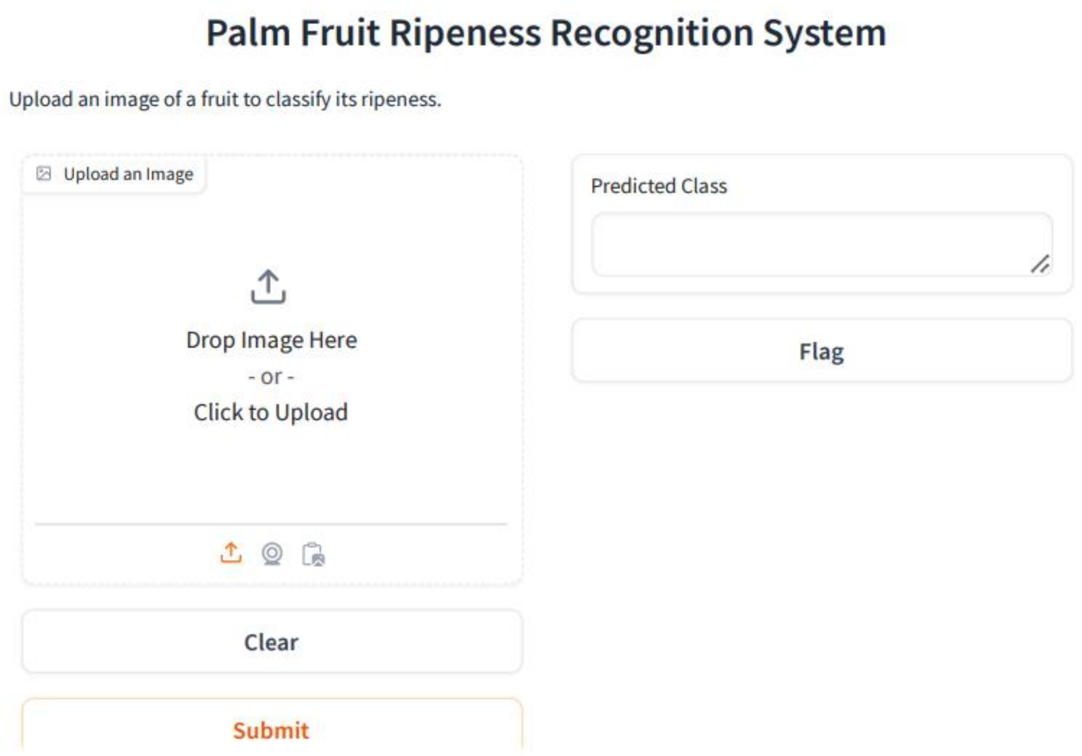
```python
# Function to process and predict an image
def predict_image(img):
    # Open the image file
    image = Image.fromarray(img)
    # Resize the image to the required input size of the model
    image = ImageOps.fit(image, (224, 224), Image.ANTIALIAS)
    # Convert the image to a numpy array
    image_array = np.asarray(image)
    # Normalize the image array
    normalized_image_array = (image_array.astype(np.float32) / 127.5) - 1
    # Load the image into a 4D tensor (1, 224, 224, 3)
    data = np.expand_dims(normalized_image_array, axis=0)
    # Predict the image
    prediction = model.predict(data)
    # Get the predicted class
    predicted_class = class_names[np.argmax(prediction)]
    return predicted_class.strip()
```

- **Open Images**: Convert images uploaded from NumPy arrays to PIL images for easier manipulation.
- **Resizing**: Ensures the image matches the input size expected by the model (224x224 pixels).
- **Converting to NumPy Array**: Simplifies the numerical operations required for normalization.
- **Normalize**: Scales pixel values to the range [-1, 1], which deep learning models often need for better performance.
- **Expand Dimensionality**: Converts the image array to a 4D tensor (cluster size, height, width, channel), as the model expects this shape.
- **Predicting**: Using models to predict image classes.
- **Mapping to Class Name**: Find the class with the highest prediction probability and map it to a readable label.

```python
# Create a Gradio interface
interface = gr.Interface(
    fn=predict_image,
    inputs=gr.Image(type="numpy", label="Upload an Image"),
    outputs=gr.Textbox(label="Predicted Class"),
    title="Palm Fruit Ripeness Recognition System",
    description="Upload an image of a fruit to classify its ripeness."
)

# Launch the interface
interface.launch(share=True)  # share=True creates a public link to the interface
```

- **Gradio Interface**: Provide a user-friendly interface that allows users to upload images and view predicted classes.
- **Function Binding**: Binds the 'predict_image' function to the interface, defining it as an operation to be performed when an image is uploaded.
- **Input and Output**: Specifies that the input will be an image and the output will be a text box displaying the predicted class.
- **Interface Metadata**: Added titles and descriptions to make the interface more informative and user-friendly.
- **Launch**: Launches the web interface, making it accessible via a public link if 'share=True', allowing others to use the tool without having to run the code themselves.

Output of the GUI:

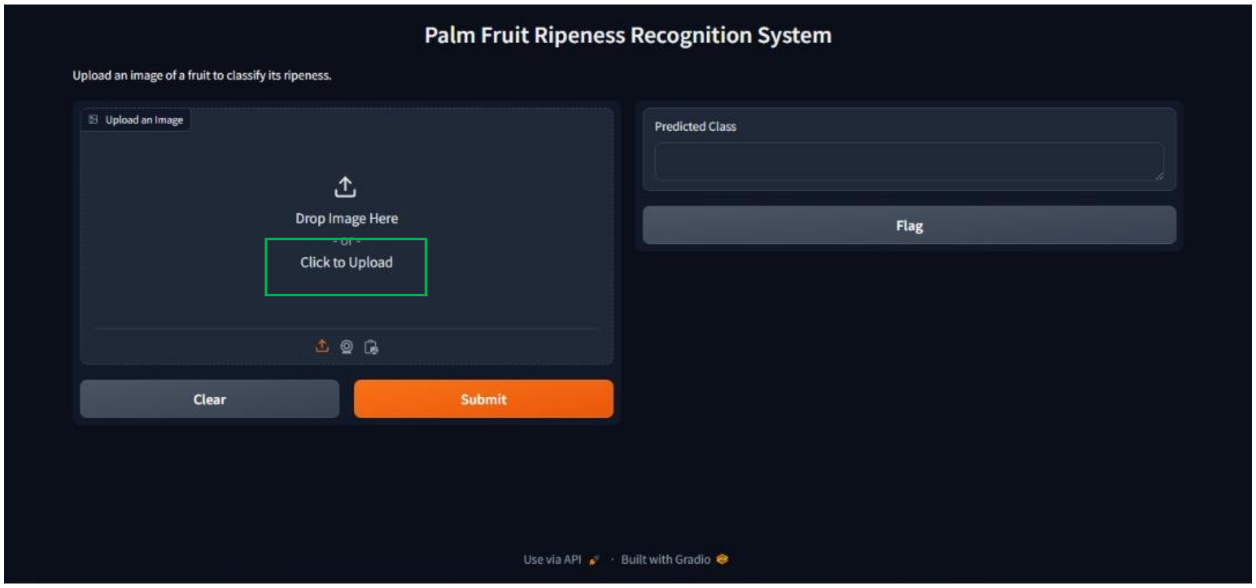


Palm Fruit Ripeness Recognition System

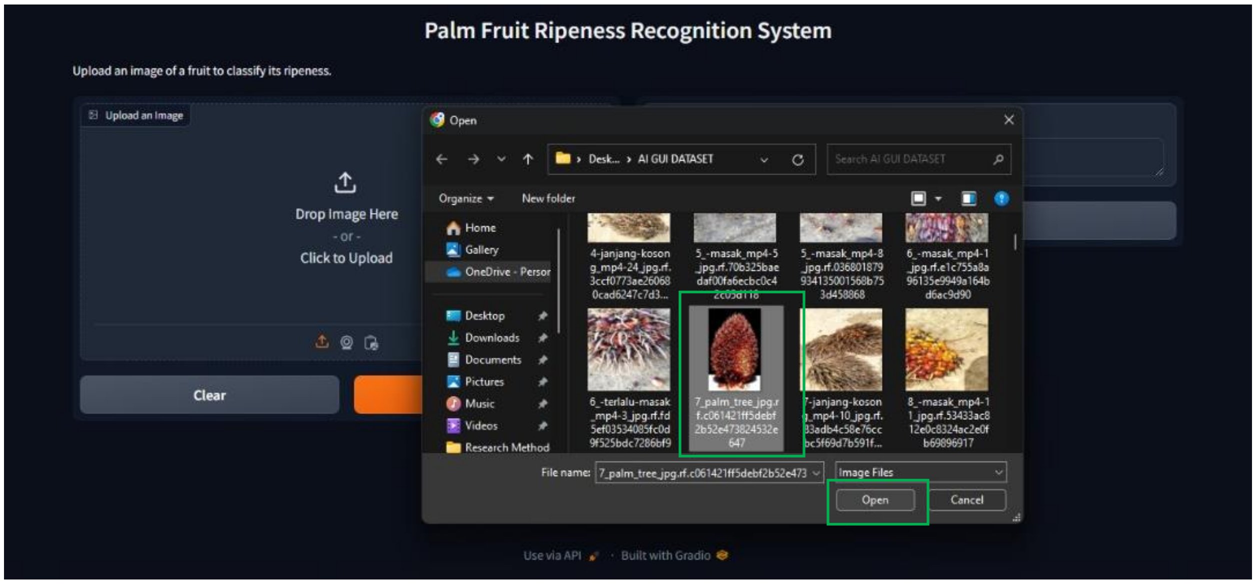
Upload an image of a fruit to classify its ripeness.

RESULTS AND DISCUSSION

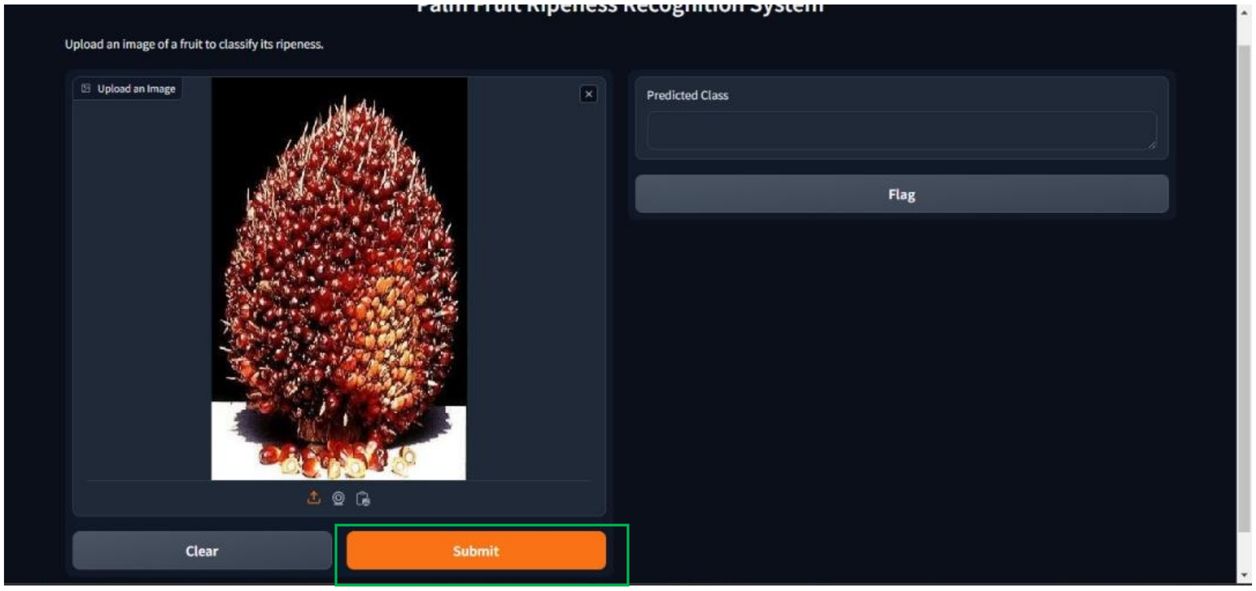
When the code is run successfully, the link to the GUI will be provided in the output. Upon clicking on the link, a new website will open a system with the title Palm Fruit Ripeness Recognition System.


Palm Fruit Ripeness Recognition System
Upload an image of a fruit to classify its ripeness.
Upload an Image
Drop Image Here
- or -
Click to Upload
Predicted Class
Flag
Clear
Submit
Use via API · Built with Gradio

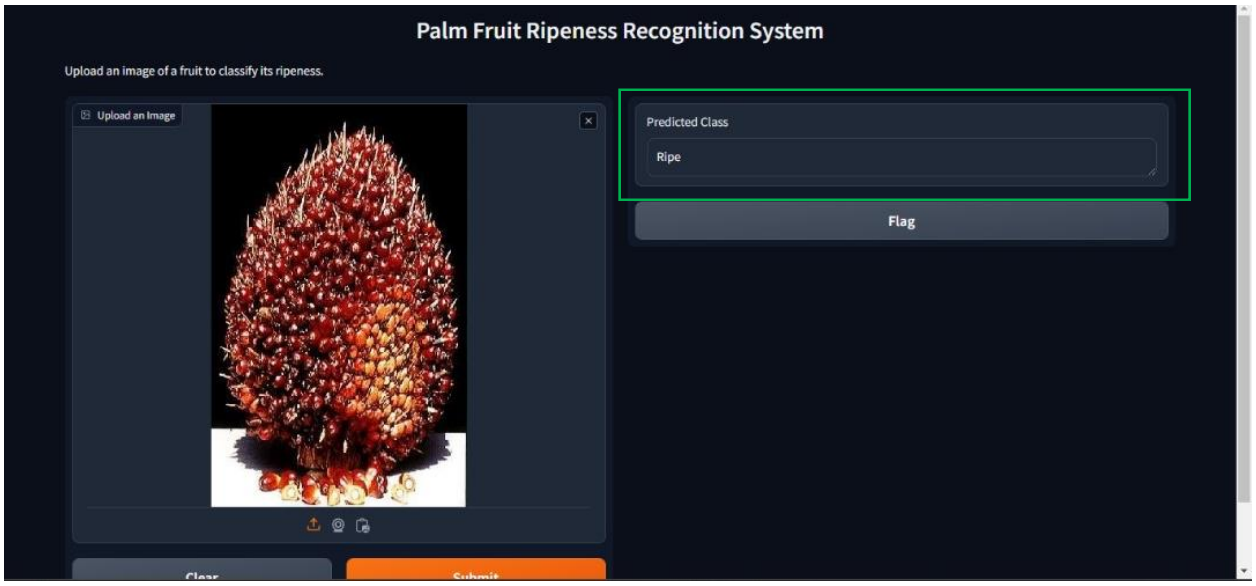
The user can select “Click to Upload” to upload palm fruit picture from the device to identify its ripeness. When the user clicks “Click to Upload”, the file directory opens and user may select which palm fruit picture to detect. For example, a picture was chosen randomly.



After the chosen picture has been uploaded, the user may click “Submit” button to check the ripeness of the palm fruit displayed in the picture.

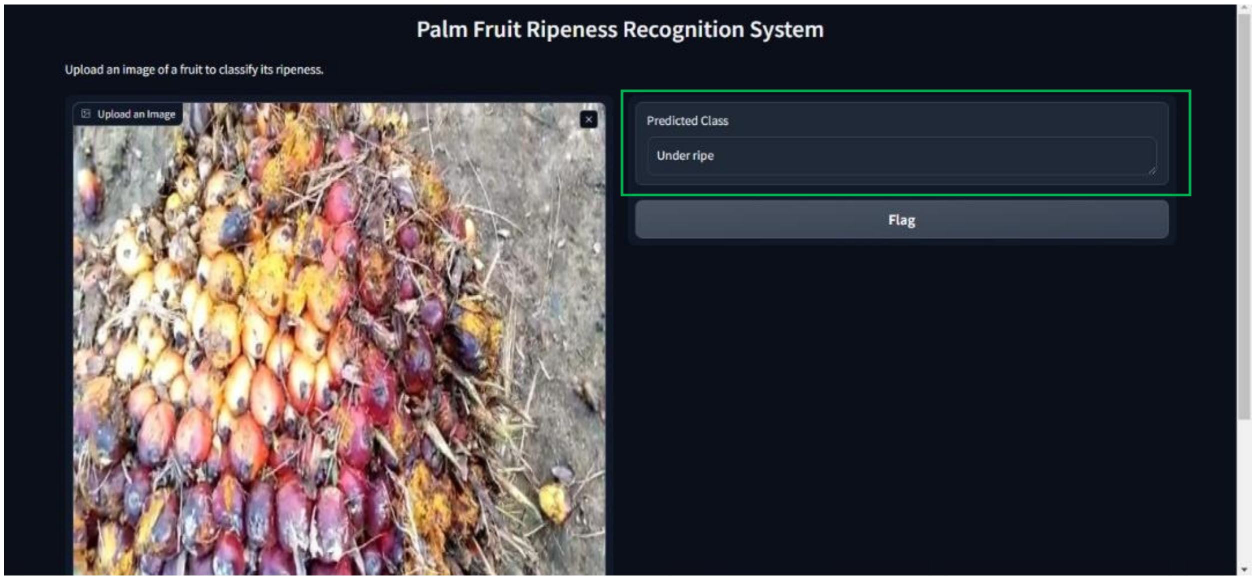

Palm Fruit Ripeness Recognition System
Upload an image of a fruit to classify its ripeness.
Upload an Image
Predicted Class
Flag
Clear
Submit

After submitting, the output will be obtained in the predicted class. In the predicted class, it will show whether the fruit displayed is ripe, unripe, under ripe, over ripe or empty. For this result, it shows the palm fruit is ripe. The system identify the ripeness of the fruit caused by the colour factor where if the fruit is ripe, its colour will be orange-red, over ripe will be in orange, under ripe is red-purple in colour. The unripe palm fruit will be in purple-black colour. These are the classification used in this system to identify and determine the ripeness of the palm fruit.


Palm Fruit Ripeness Recognition System
Upload an image of a fruit to classify its ripeness.
Upload an Image
Flag
Clear
Submit

Predicted Class
Ripe

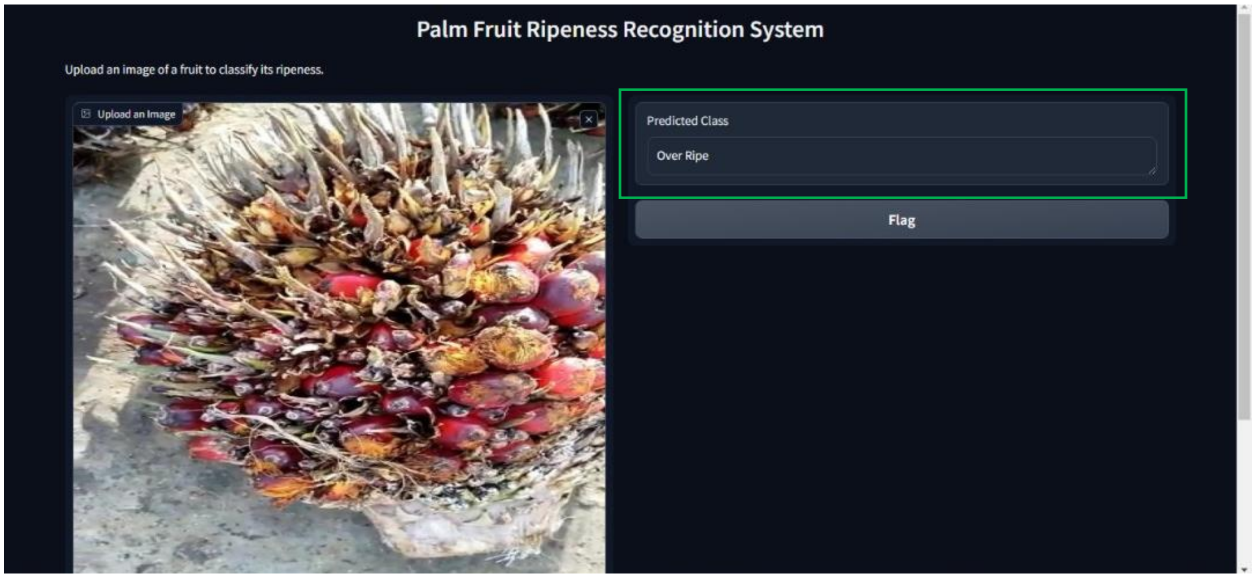
Next, picture 2 is chosen by selecting “Clear” button to remove the picture 1 and select “Click to Upload” again to upload pictures from file directory. The output shown below identified the palm fruit as under ripe.


Palm Fruit Ripeness Recognition System
Upload an image of a fruit to classify its ripeness.
Upload an Image
Flag

Predicted Class

Under ripe

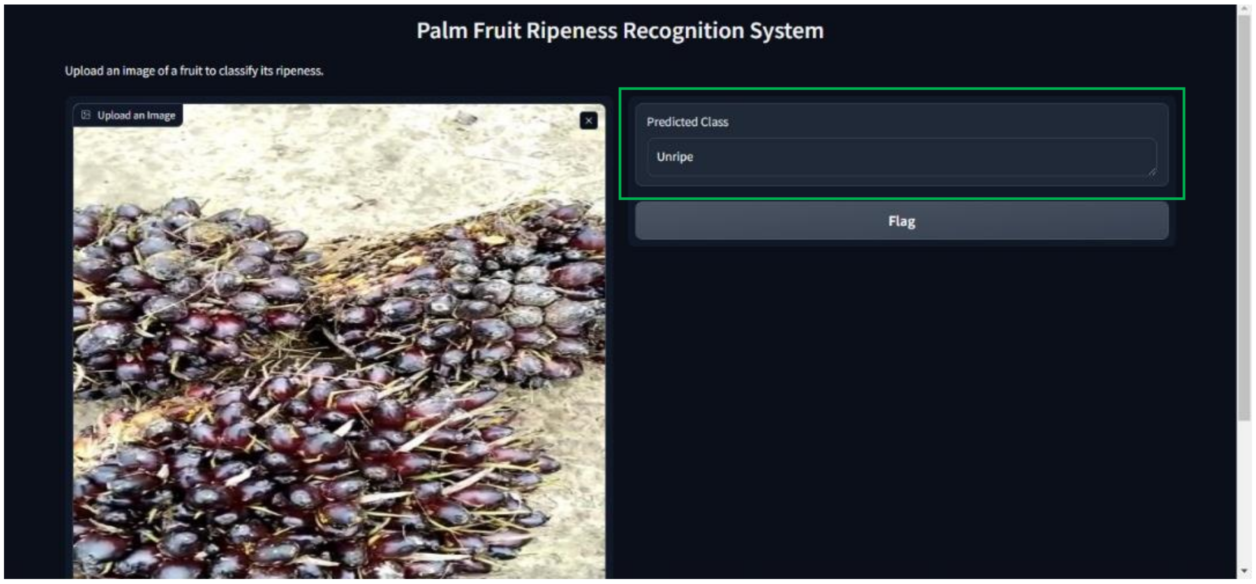
Picture 3 is chosen as another example of palm fruit ripeness. The picture displayed below is detected as over ripe.


Palm Fruit Ripeness Recognition System
Upload an image of a fruit to classify its ripeness.
Upload an Image
Flag

Predicted Class

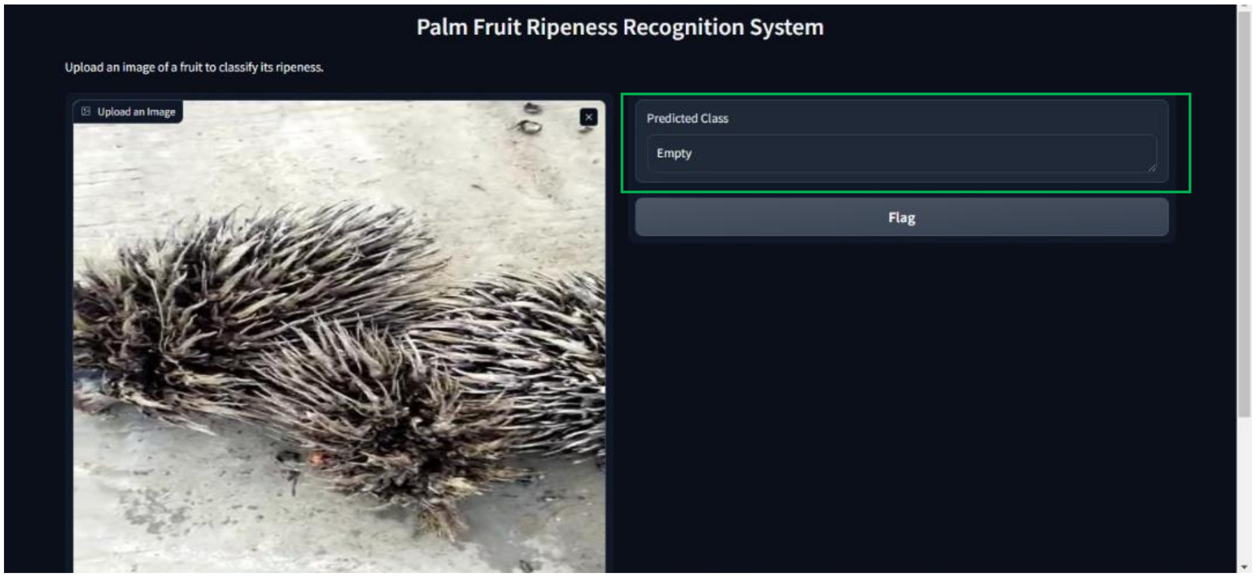
Over Ripe

Picture 4 is chosen as fourth example of palm fruit ripeness recognition. The palm fruit below is identified as unripe based on the colour of the fruit.


Palm Fruit Ripeness Recognition System
Upload an image of a fruit to classify its ripeness.
Upload an Image
Predicted Class
Flag

Unripe

Picture 5 is chosen and in this output, the system detects the selected palm fruit image as empty.


Palm Fruit Ripeness Recognition System
Upload an image of a fruit to classify its ripeness.
Upload an Image
Predicted Class
Flag

Empty

CONCLUSION

In summary, our project, "PalmScan: Smart Monitoring for Palm Plantation Quality," has effectively created an AI-driven system to improve the precision and efficiency of Malaysian palm oil fruit inspection. Through the application of sophisticated image recognition methods and machine learning models, we have developed a strong framework that can identify several types of palm fruits, including ripe, unripe, underripe, overripe, and empty. The substantial difficulties presented by manual inspection which is frequently rife with mistakes and inefficiencies are addressed by this approach.

We collected a diverse dataset of palm fruit images, representing different stages of ripeness and environmental conditions to ensure the model's robustness. The training and validation processes, enhanced by the use of tools like Teachable Machine and Keras, resulted in a model with a high accuracy rate. The implementation of a graphical user interface (GUI) makes it easy for users to interact with the system, whether through a desktop application, enabling real-time monitoring and decision-making.

This project's impact goes beyond improvements in technology. It greatly lessens the need for manual labor by automating the inspection process for palm fruit, which raises overall productivity and guarantees constant product quality. This innovation lowers labor costs and streamlines export procedures, which boosts economic growth in addition to improving operational efficiency.

Additionally, the system's real-time monitoring feature offers timely insights that allow for rapid response, enhancing the management and harvesting procedures in palm plantations. This research serves as an excellent example of how artificial intelligence and computer vision technology may be used to promote sustainable practices, solve practical agricultural problems, and maximize resource usage.

Hence, the PalmScan project, which offers a scalable and effective way to raise the caliber and output of the palm oil sector, essentially marks a significant advancement in the application of AI in agriculture. This technology not only increases economic rewards but also advances agricultural practices and sustainability by improving the speed and accuracy of quality assessments.

REFERENCES

1) Computer Vision Project. (2023). Oil palm ripeness dataset. Roboflow. https://universe.roboflow.com/fyp-mxrwa/oil-palm-ripeness-dataset

2) Google. (n.d.). Teachable Machine. Teachable Machine with Google. https://teachablemachine.withgoogle.com/

3) Palm Fruit Classification. (n.d.). Palm-fruit-ripeness-classificationcnn dataset and pre-trained model. Roboflow. https://universe.roboflow.com/palm-fruit-classification/palm-fruit-ripeness-classificationcnn/images/ytGKrT4Nyy2R0y0TDBmK?queryText=&pageSize=50&startingIndex=0&browseQuery=true

4) Suharjito, N., Adeta, F., Junior, Koeswandy, Y. P., Debi, N., Nurhayati, P. W., Asrol, M., & Marimin, N. (2023). Annotated datasets of oil palm fruit bunch piles for ripeness grading using deep learning. Scientific Data, 10(1). https://doi.org/10.1038/s41597-023-01958-x

5) Suharjito, N., Elwirehardja, G. N., & Prayoga, J. S. (2021). Oil palm fresh fruit bunch ripeness classification on mobile devices using deep learning approaches. Computers and Electronics in Agriculture, 188, 106359. https://doi.org/10.1016/j.compag.2021.106359

6) VGG Image Annotator (VIA). (n.d.). Visual Geometry Group - University of Oxford. https://www.robots.ox.ac.uk/~vgg/software/via/

APPENDIX

AI Coding File

https://drive.google.com/drive/folders/1CH2SZP0uQO7J1-oEqQxeJ6Q9JQoc6W7p?usp=sharing

Images File for Training, Validating and GUI Process

AI GUI DATASET

https://drive.google.com/drive/folders/1AIE-NuRd6WSjiO8A77L8LPBAMrXjsVT5?usp=sharing

AI Validate Dataset

https://drive.google.com/drive/folders/1dFiKoHttGqw6g8Ji9eSrFeLZcIoMiw06?usp=drive_link

AI Validate Dataset - 100 train

https://drive.google.com/drive/folders/1_VDrkc7BszsJnvRAYZjujTzKztslu9i2?usp=drive_link

AI Validate Dataset - Remaining

https://drive.google.com/drive/folders/1eWIABnA6U3BLe1meb-JS2NF7Ffzexz_0?usp=drive_link