```
!pip install ultralytics
```

```
Requirement already satisfied: ultralytics in /usr/local/lib/python3.12/dist-packages (8.3.202)
Requirement already satisfied: numpy>=1.23.0 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (2.0.2)
Requirement already satisfied: matplotlib>=3.3.0 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (3.10.0)
Requirement already satisfied: opencv-python>=4.6.0 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (4.12.0.88)
Requirement already satisfied: pillow>=7.1.2 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (11.3.0)
Requirement already satisfied: pyyaml>=5.3.1 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (6.0.2)
Requirement already satisfied: requests>=2.23.0 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (2.32.4)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (1.16.1)
Requirement already satisfied: torch>=1.8.0 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (2.8.0+cu126)
Requirement already satisfied: torchvision>=0.9.0 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (0.23.0+cu126)
Requirement already satisfied: psutil in /usr/local/lib/python3.12/dist-packages (from ultralytics) (5.9.5)
Requirement already satisfied: polars in /usr/local/lib/python3.12/dist-packages (from ultralytics) (1.25.2)
Requirement already satisfied: ultralytics-thop>=2.0.0 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (2.0.17)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=3.3.0->ultralytics
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=3.3.0->ultralytics) (0
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=3.3.0->ultralytic
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=3.3.0->ultralytic
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=3.3.0->ultralytics)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=3.3.0->ultralytics
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=3.3.0->ultraly
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests>=2.23.0->ultr
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests>=2.23.0->ultralytics) (3
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests>=2.23.0->ultralytic
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests>=2.23.0->ultralytic
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultralytics) (3.19.1)
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultraly
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultralytics) (75.2.0)
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultralytics) (1.13
Requirement already satisfied: networkx in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultralytics) (3.5)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultralytics) (3.1.6)
Requirement already satisfied: fsspec in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultralytics) (2025.3.0)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->u
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0-
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.6.80 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->u
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultr
Requirement already satisfied: nvidia-cublas-cu12==12.6.4.1 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultr
Requirement already satisfied: nvidia-cufft-cu12==11.3.0.4 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultra
Requirement already satisfied: nvidia-curand-cu12==10.3.7.77 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ult
Requirement already satisfied: nvidia-cusolver-cu12==11.7.1.2 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ul
Requirement already satisfied: nvidia-cusparse-cu12==12.5.4.2 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ul
Requirement already satisfied: nvidia-cusparselt-cu12==0.7.1 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ult
Requirement already satisfied: nvidia-nccl-cu12==2.27.3 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultralyt
Requirement already satisfied: nvidia-nvtx-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultraly
Requirement already satisfied: nvidia-nvjitlink-cu12==12.6.85 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ul
Requirement already satisfied: nvidia-cufile-cu12==1.11.1.6 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultr
Requirement already satisfied: triton==3.4.0 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultralytics) (3.4.0
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7->matplotlib>=3.3
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.12/dist-packages (from sympy>=1.13.3->torch>=1.8.0-
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from jinja2->torch>=1.8.0->ultralyti
```

```
# Step 1: Install Libraries
from ultralytics import YOLO
import sys, os
import shutil
import random
import zipfile
from google.colab import drive
```

```
# Step 2: Mount Drive & Extract Dataset
drive.mount('/content/drive')

zip_path = "/content/drive/MyDrive/Pictures/all-leaves.zip"
extract_path = "/content/drive/MyDrive/Pictures/all-leaves/all_leaves_unzip"

# Extract dataset if not already extracted
if not os.path.exists(extract_path):
    print("⏳ Extracting dataset...")
    with zipfile.ZipFile(zip_path, 'r') as zip_ref:
        zip_ref.extractall(extract_path)
    print("✅ Dataset extracted!")
else:
    print("✅ Dataset already extracted.")

# Count images
dry_path = os.path.join(extract_path, "all_dry")
healthy_path = os.path.join(extract_path, "all_healthy")

dry_count = len([f for f in os.listdir(dry_path) if f.lower().endswith((".jpg",".jpeg",".png"))])
healthy_count = len([f for f in os.listdir(healthy_path) if f.lower().endswith((".jpg",".jpeg",".png"))])

print(f"Dry: {dry_count}, Healthy: {healthy_count}")
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True)
✅ Dataset already extracted.
Dry: 2025, Healthy: 2005
```

```
# Step 3: Prepare YOLO Dataset Folders

yolo_dataset = "/content/leaf_dataset"
train_images = os.path.join(yolo_dataset, "train/images")
val_images = os.path.join(yolo_dataset, "val/images")
train_labels = os.path.join(yolo_dataset, "train/labels")
val_labels = os.path.join(yolo_dataset, "val/labels")

# Clean previous dataset if exists
if os.path.exists(yolo_dataset):
    shutil.rmtree(yolo_dataset)

os.makedirs(train_images)
os.makedirs(val_images)
os.makedirs(train_labels)
os.makedirs(val_labels)

# Split data: 80% train, 20% val
def copy_images(src_folder, class_id):
    files = [f for f in os.listdir(src_folder) if f.lower().endswith((".jpg",".jpeg",".png"))]
    random.shuffle(files)
    split = int(0.8 * len(files))

    for i, file in enumerate(files):
        src = os.path.join(src_folder, file)
        if i < split:
            dst_img = os.path.join(train_images, file)
            dst_lbl = os.path.join(train_labels, file.replace(".jpg", ".txt").replace(".png", ".txt").replace(".jpeg", ".txt"
        else:
            dst_img = os.path.join(val_images, file)
            dst_lbl = os.path.join(val_labels, file.replace(".jpg", ".txt").replace(".png", ".txt").replace(".jpeg", ".txt"))

        shutil.copy(src, dst_img)

        # YOLO label format: class_id center_x center_y width height (normalized)
        with open(dst_lbl, "w") as f:
            f.write(f"{class_id} 0.5 0.5 1 1\n")  # Whole image = one class

# Class 0 = dry, Class 1 = healthy
copy_images(dry_path, 0)
copy_images(healthy_path, 1)

print("✅ Dataset ready in YOLO format!")
```

```
✅ Dataset ready in YOLO format!
```

```
# Step 4: Create Data.yaml File

data_yaml = "/content/leaf_dataset/data.yaml"
with open(data_yaml, "w") as f:
    f.write(f"path: {yolo_dataset}\n")
    f.write("train: train/images\n")
    f.write("val: val/images\n")
    f.write("names:\n")
    f.write("  0: dry\n")
    f.write("  1: healthy\n")

print("✅ data.yaml created!")
```

```
✅ data.yaml created!
```

```
# Step 5: Train YOLOv8 Model

model = YOLO("yolov8n.pt")  # Nano version for speed, use yolov8s.pt for better accuracy

model.train(
    data=data_yaml,
    epochs=20,
    imgsz=224,
    batch=16,
    name="leaf_model",
    project="/content/leaf_results"
)
```

```
                   all        806        795      0.927      0.886      0.965      0.952

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       5/20         0G     0.1669     0.5213     0.8904          4        224: 100% ━━━━━━━━━━━━ 202/202 0.6it/s 5:25
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 26/26 0.9it/s 29.
                   all        806        795       0.79      0.927      0.964      0.937

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       6/20         0G     0.1588     0.4975     0.8872          4        224: 100% ━━━━━━━━━━━━ 202/202 0.6it/s 5:31
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 26/26 0.9it/s 28.
                   all        806        795      0.875      0.849      0.945      0.932

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       7/20         0G     0.1487     0.4643     0.8891          4        224: 100% ━━━━━━━━━━━━ 202/202 0.6it/s 5:26
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 26/26 0.9it/s 30.
                   all        806        795      0.962      0.938      0.982       0.98

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       8/20         0G     0.1394     0.4227     0.8877          4        224: 100% ━━━━━━━━━━━━ 202/202 0.6it/s 5:26
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 26/26 0.9it/s 30.
                   all        806        795      0.951      0.928      0.976      0.961

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       9/20         0G      0.131     0.3999     0.8869          4        224: 100% ━━━━━━━━━━━━ 202/202 0.6it/s 5:30
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 26/26 0.9it/s 28.
                   all        806        795      0.959      0.931      0.981      0.979

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      10/20         0G     0.1173     0.3928     0.8825          3        224: 100% ━━━━━━━━━━━━ 202/202 0.6it/s 5:29
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 26/26 0.9it/s 29.
                   all        806        795      0.966      0.973      0.992      0.977
Closing dataloader mosaic
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_limit=(3, 7)), ToGray(p=0.01, method='weighted_avera

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      11/20         0G     0.1166      0.346     0.8636          1        224: 100% ━━━━━━━━━━━━ 202/202 0.6it/s 5:24
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 26/26 0.9it/s 29.
                   all        806        795       0.94      0.943      0.983      0.967

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      12/20         0G    0.09998     0.2736     0.8642          1        224: 100% ━━━━━━━━━━━━ 202/202 0.6it/s 5:24
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 26/26 0.9it/s 28.
                   all        806        795      0.977      0.982      0.994       0.99

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      13/20         0G    0.09207     0.2417     0.8572          1        224: 100% ━━━━━━━━━━━━ 202/202 0.6it/s 5:20
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 26/26 0.9it/s 29.
                   all        806        795      0.954      0.984      0.991       0.99

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      14/20         0G    0.08021     0.2029     0.8575          1        224: 100% ━━━━━━━━━━━━ 202/202 0.6it/s 5:22
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 26/26 0.9it/s 28.
                   all        806        795      0.958      0.966      0.993      0.987

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      15/20         0G    0.07466     0.1766      0.857          1        224: 100% ━━━━━━━━━━━━ 202/202 0.6it/s 5:21
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 26/26 0.9it/s 28.
                   all        806        795      0.985      0.992      0.995      0.983
```

```
# Step 6: Test on New Images
from contextlib import redirect_stdout
import pandas as pd

test_folder = "/content/drive/MyDrive/Pictures/testing_images"

# Suppress most output during prediction
with open(os.devnull, 'w') as f, redirect_stdout(f):
    results = model.predict(source=test_folder, save=True, conf=0.5)

# Collect results into a dataframe(optional)
predictions = []
for r in results:
    boxes = r.boxes
    if len(boxes) > 0:
        for i in range(len(boxes)):
            class_id = int(boxes.cls[i])
            class_name = r.names[class_id]
            predictions.append([os.path.basename(r.path), class_name])
    else:
        predictions.append([os.path.basename(r.path), "No detection"])

# Save to CSV
csv_path = "/content/drive/MyDrive/Pictures/predictions.csv"
pd.DataFrame(predictions, columns=["Image", "Prediction"]).to_csv(csv_path, index=False)
```

```
print(f"Predictions saved at: {results[0].save_dir}")
print(f"CSV with results saved at: {csv_path}")
```

```
image 1/35 /content/drive/MyDrive/Pictures/testing_images/1067.jpg: 224x224 1 healthy, 45.1ms
image 2/35 /content/drive/MyDrive/Pictures/testing_images/1146.jpg: 224x224 1 dry, 40.1ms
image 3/35 /content/drive/MyDrive/Pictures/testing_images/1211.jpg: 192x224 1 dry, 33.4ms
image 4/35 /content/drive/MyDrive/Pictures/testing_images/14.jpg: 192x224 1 dry, 35.3ms
image 5/35 /content/drive/MyDrive/Pictures/testing_images/2019.jpg: 192x224 1 dry, 32.8ms
image 6/35 /content/drive/MyDrive/Pictures/testing_images/2090.jpg: 224x224 1 healthy, 41.1ms
image 7/35 /content/drive/MyDrive/Pictures/testing_images/2144.jpg: 192x224 1 dry, 47.6ms
image 8/35 /content/drive/MyDrive/Pictures/testing_images/2737.jpg: 192x224 1 dry, 32.4ms
image 9/35 /content/drive/MyDrive/Pictures/testing_images/2809.jpg: 192x224 1 dry, 32.0ms
image 10/35 /content/drive/MyDrive/Pictures/testing_images/2892.jpg: 224x224 1 healthy, 37.4ms
image 11/35 /content/drive/MyDrive/Pictures/testing_images/4588.jpg: 192x224 1 healthy, 32.4ms
image 12/35 /content/drive/MyDrive/Pictures/testing_images/4868.jpg: 192x224 1 healthy, 32.4ms
image 13/35 /content/drive/MyDrive/Pictures/testing_images/5086.jpg: 192x224 1 healthy, 32.1ms
image 14/35 /content/drive/MyDrive/Pictures/testing_images/5485.jpg: 192x224 1 healthy, 31.9ms
image 15/35 /content/drive/MyDrive/Pictures/testing_images/5_jpg.rf.9c5f9b73134c8b5ef944f9330601b25f.jpg: 224x224 1 dry, 37.1ms
image 16/35 /content/drive/MyDrive/Pictures/testing_images/6149_iOS.jpg: 224x224 1 dry, 59.5ms
image 17/35 /content/drive/MyDrive/Pictures/testing_images/6567.jpg: 224x224 1 dry, 57.3ms
image 18/35 /content/drive/MyDrive/Pictures/testing_images/7090.jpg: 224x224 1 dry, 56.3ms
image 19/35 /content/drive/MyDrive/Pictures/testing_images/7153.jpg: 224x224 1 healthy, 56.5ms
image 20/35 /content/drive/MyDrive/Pictures/testing_images/754.jpg: 192x224 1 dry, 51.9ms
image 21/35 /content/drive/MyDrive/Pictures/testing_images/9237.jpg: 224x224 1 healthy, 58.8ms
image 22/35 /content/drive/MyDrive/Pictures/testing_images/Dried_0864.jpg: 160x224 1 dry, 44.7ms
image 23/35 /content/drive/MyDrive/Pictures/testing_images/Dried_128.jpg: 160x224 1 dry, 44.8ms
image 24/35 /content/drive/MyDrive/Pictures/testing_images/Fresh_0581.jpg: 160x224 1 healthy, 47.4ms
image 25/35 /content/drive/MyDrive/Pictures/testing_images/Fresh_106.jpg: 160x224 1 healthy, 45.8ms
image 26/35 /content/drive/MyDrive/Pictures/testing_images/IMG_9099.jpg: 224x224 1 dry, 58.4ms
image 27/35 /content/drive/MyDrive/Pictures/testing_images/IMG_9117.jpg: 224x224 1 dry, 60.6ms
image 28/35 /content/drive/MyDrive/Pictures/testing_images/image077.jpg: 224x192 1 healthy, 51.8ms
image 29/35 /content/drive/MyDrive/Pictures/testing_images/image284.jpg: 224x192 1 dry, 51.5ms
image 30/35 /content/drive/MyDrive/Pictures/testing_images/image316.jpg: 224x192 1 dry, 48.1ms
image 31/35 /content/drive/MyDrive/Pictures/testing_images/lea9.jpeg: 160x224 1 dry, 49.7ms
image 32/35 /content/drive/MyDrive/Pictures/testing_images/leaf1.jpeg: 224x160 1 dry, 51.0ms
image 33/35 /content/drive/MyDrive/Pictures/testing_images/leaf2.jpg: 192x224 1 dry, 1 healthy, 51.4ms
image 34/35 /content/drive/MyDrive/Pictures/testing_images/leaf3.jpg: 224x192 1 dry, 49.3ms
image 35/35 /content/drive/MyDrive/Pictures/testing_images/leaf7.webp: 224x192 1 healthy, 53.3ms
Speed: 1.1ms preprocess, 45.5ms inference, 1.0ms postprocess per image at shape (1, 3, 224, 192)
Results saved to /content/leaf_results/leaf_model25
Predictions saved at: /content/leaf_results/leaf_model25
CSV with results saved at: /content/drive/MyDrive/Pictures/predictions.csv
```

```
"/content/leaf_results/leaf_model/weights/best.pt"
```

```
'/content/leaf_results/leaf_model/weights/best.pt'
```

```
os.environ["EMAIL_PASSWORD"] = "YOUR EMAIL PASSWORD"
```

```
# Step 7
import cv2
import numpy as np
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.base import MIMEBase
from email.mime.text import MIMEText
from email import encoders
from google.colab.patches import cv2_imshow

#CONFIGURATION
image_folder = "/content/drive/MyDrive/Pictures/testing_images"
dryness_threshold = 40  # %dryness above which email alert will be sent

sender_email = "Mention sender_email ID"
receiver_email = "Mention receiver_email ID"
password = os.environ.get("EMAIL_PASSWORD")
```

```
# Step 8:calculate_leaf_health
def calculate_leaf_health_fixed(image_path):
    img = cv2.imread(image_path)
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    # Convert to HSV for leaf segmentation
    hsv = cv2.cvtColor(img_rgb, cv2.COLOR_RGB2HSV)

    # Step 1: Leaf segmentation (green + dry)
    lower_leaf = np.array([15, 30, 30])
    upper_leaf = np.array([110, 255, 255])
    leaf_mask = cv2.inRange(hsv, lower_leaf, upper_leaf)
    leaf_only = cv2.bitwise_and(img_rgb, img_rgb, mask=leaf_mask)
```

```
    # Step 2: Healthy vs Dry detection inside leaf only
    lower_green = np.array([35, 40, 40])
    upper_green = np.array([90, 255, 255])

    lower_brown = np.array([5, 50, 50])
    upper_brown = np.array([25, 255, 200])

    hsv_leaf = cv2.cvtColor(leaf_only, cv2.COLOR_RGB2HSV)
    green_mask = cv2.inRange(hsv_leaf, lower_green, upper_green)
    brown_mask = cv2.inRange(hsv_leaf, lower_brown, upper_brown)

    green_pixels = np.sum(green_mask > 0)
    brown_pixels = np.sum(brown_mask > 0)
    total_pixels = green_pixels + brown_pixels

    if total_pixels == 0:
        return 0, 0, img

    health_percent = (green_pixels / total_pixels) * 100
    dryness_percent = (brown_pixels / total_pixels) * 100

    #Dynamic Text Box
    output_img = img.copy()
    label = "DRY" if dryness_percent > health_percent else "HEALTHY"
    text = f"{label}: H:{health_percent:.2f}% D:{dryness_percent:.2f}%"

    font = cv2.FONT_HERSHEY_SIMPLEX
    font_scale = 0.8
    thickness = 2

    (text_width, text_height), baseline = cv2.getTextSize(text, font, font_scale, thickness)
    padding_x, padding_y = 20, 20
    start_x, start_y = 10, 10
    end_x = start_x + text_width + padding_x
    end_y = start_y + text_height + padding_y

    cv2.rectangle(output_img, (start_x, start_y), (end_x, end_y), (0, 0, 0), -1)
    cv2.putText(output_img, text, (start_x + 10, start_y + text_height + 5),
                font, font_scale, (0, 255, 0) if health_percent > dryness_percent else (0, 0, 255), thickness)

    return health_percent, dryness_percent, output_img
```

```
# Step 9: EMAIL SENDING FUNCTION
def send_email(dry_images, dryness_report):
    if not dry_images:
        print("No dry leaves above threshold. No email sent.")
        return

    msg = MIMEMultipart()
    msg["From"] = sender_email
    msg["To"] = receiver_email
    msg["Subject"] = "Irrigation Alert: Dry Leaves Detected!"

    # Email body with irrigation instructions
    body = f"""
    🌿 Irrigation Alert: Dry Leaves Detected!

    The average leaf dryness has exceeded {dryness_threshold}%.

    💧 Action Required:
    Please turn on the irrigation tap or start the watering system to maintain soil moisture.

    Dry Leaves Report:
    {dryness_report}

    Ensure proper water supply to prevent further crop damage.
    """
    msg.attach(MIMEText(body, "plain"))

    for img_path in dry_images:
        part = MIMEBase("application", "octet-stream")
        with open(img_path, "rb") as f:
            part.set_payload(f.read())
        encoders.encode_base64(part)
        part.add_header("Content-Disposition", f"attachment; filename={os.path.basename(img_path)}")
        msg.attach(part)

    try:
        with smtplib.SMTP("smtp.gmail.com", 587) as server:
            server.starttls()
            server.login(sender_email, password)
```

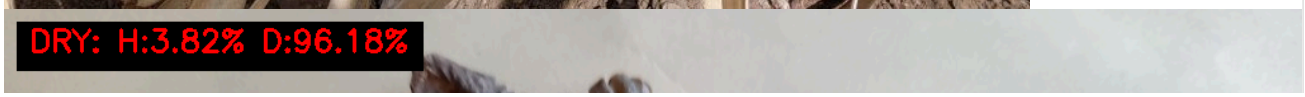
```
            server.send_message(msg)
        print("Email sent successfully with dry leaf images!")
    except Exception as e:
        print("Email sending failed:", e)
```

```
# Step 10: PROCESS ALL IMAGES
dry_images = []
dryness_report = ""
all_images = [f for f in os.listdir(image_folder) if f.lower().endswith(('.jpg', '.png', '.jpeg'))]

for img_name in all_images:
    img_path = os.path.join(image_folder, img_name)
    health, dryness, out_img = calculate_leaf_health_fixed(img_path)

    # Show all images in Colab
    cv2_imshow(out_img)

    # Collect dry images above threshold
    if dryness > dryness_threshold:
        save_path = f"/content/dry_{img_name}"
        cv2.imwrite(save_path, out_img)
        dry_images.append(save_path)
        dryness_report += f"{img_name} -> Health: {health:.2f}%, Dryness: {dryness:.2f}%\n"
 # Step 11
if dry_images:
    send_email(dry_images, dryness_report)
else:
    print("All leaves are healthy. No email sent.")
```

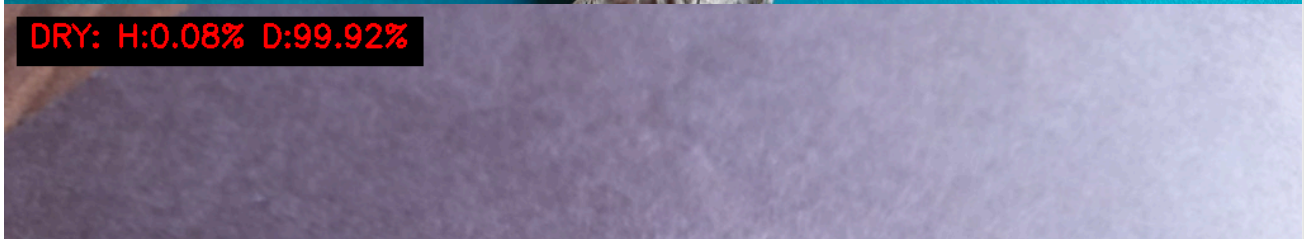

HEALTHY: H:99.75% D:0.25%

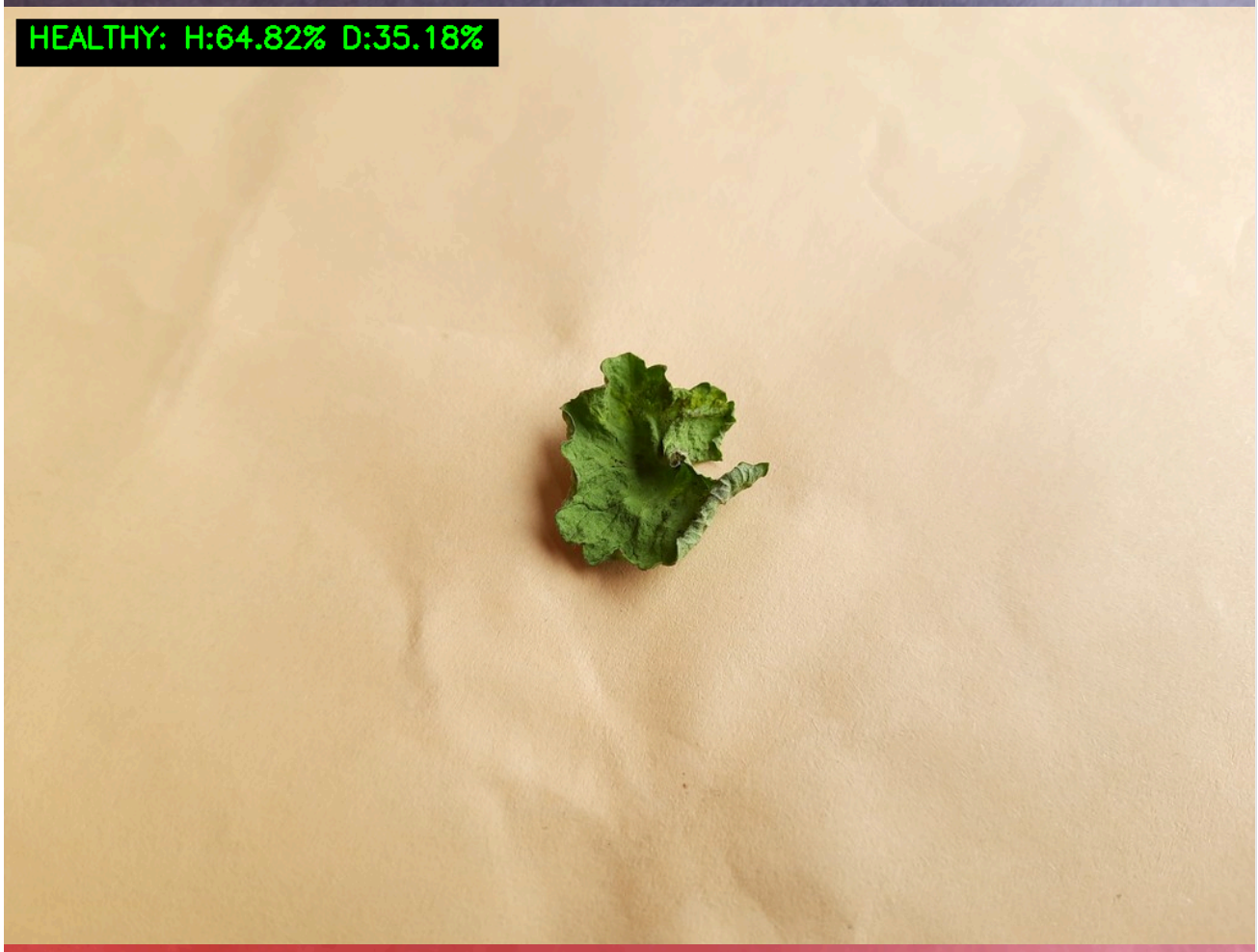

DRY: H:0.00% D:100.00%


DRY: H:42.09% D:57.91%

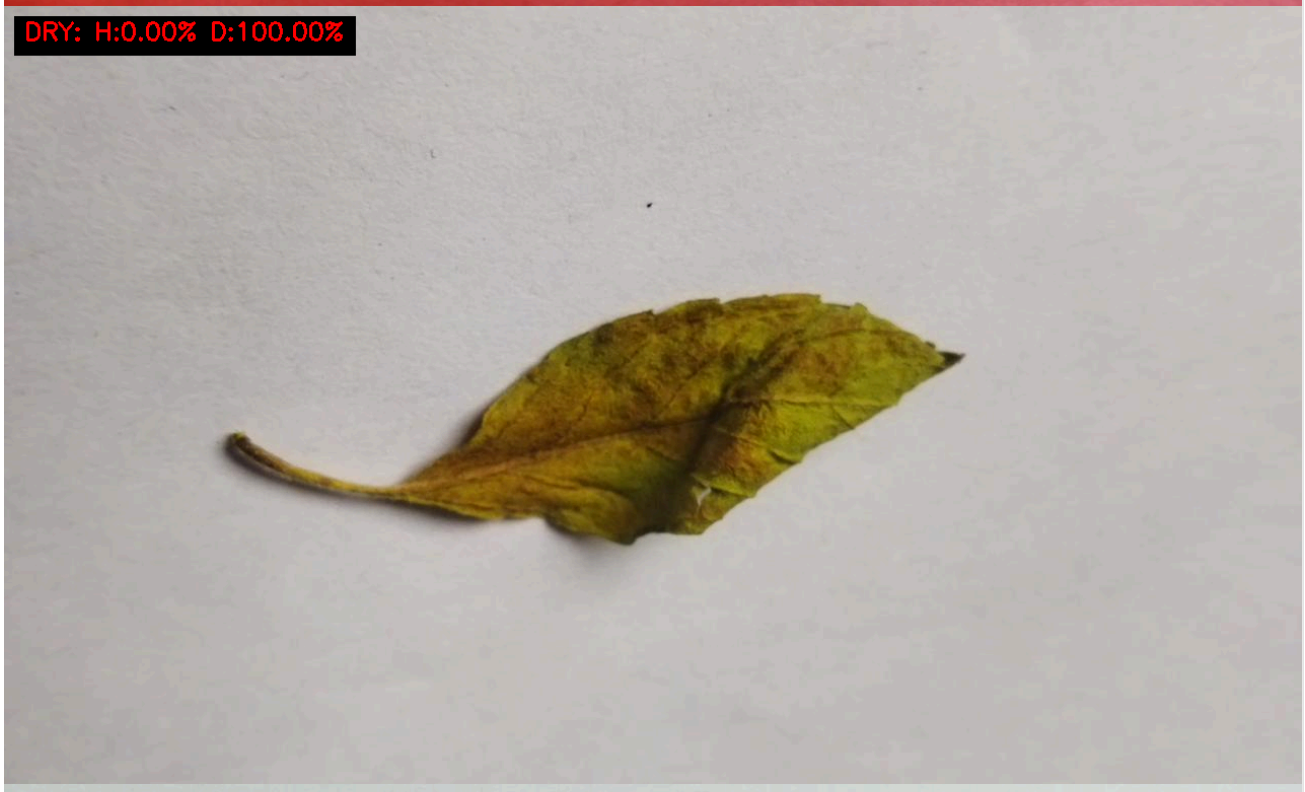

DRY: H:3.82% D:96.18%

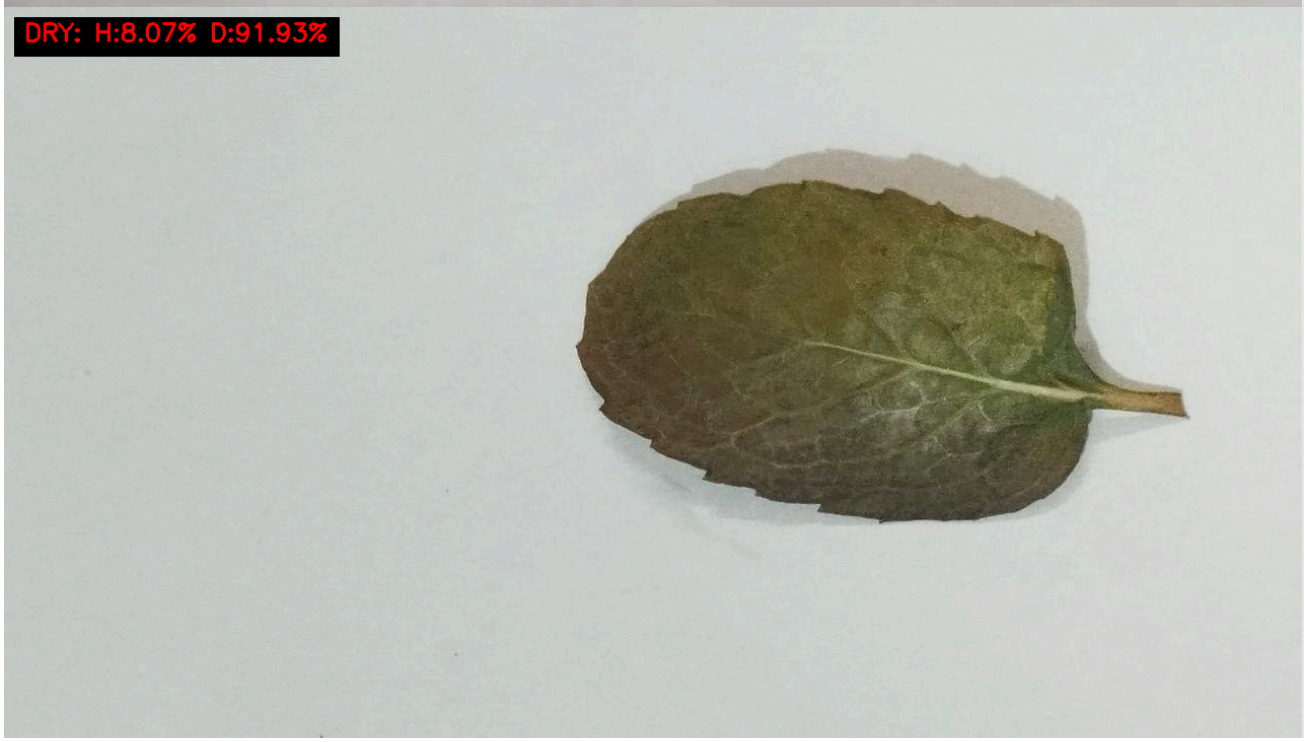

DRY: H:16.67% D:83.33%


DRY: H:0.08% D:99.92%


HEALTHY: H:64.82% D:35.18%


DRY: H:40.38% D:59.62%

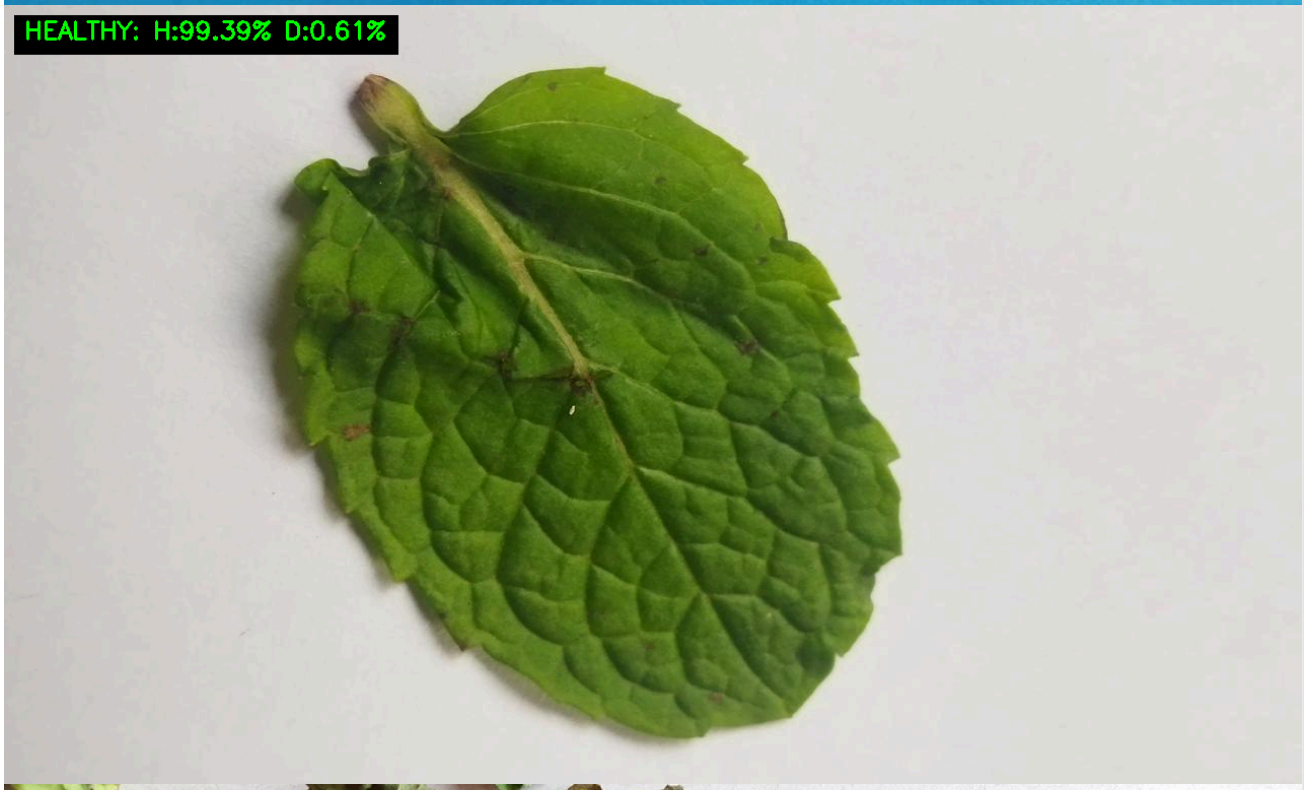

DRY: H:0.00% D:100.00%


DRY: H:8.07% D:91.93%

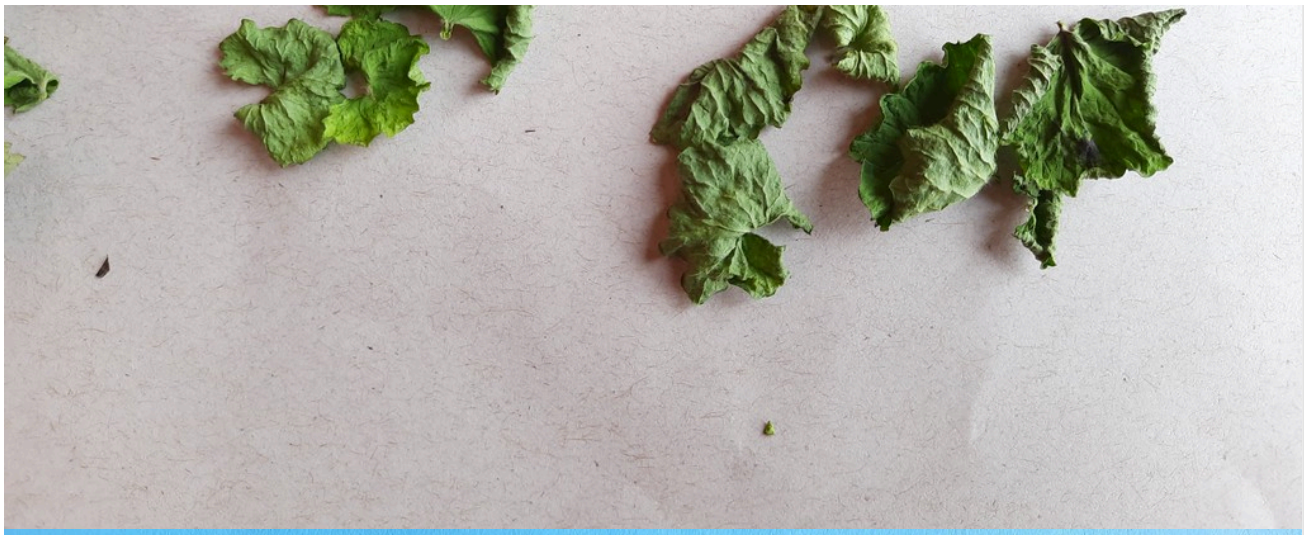

HEALTHY: H:98.83% D:1.17%


HEALTHY: H:80.94% D:19.06%


HEALTHY: H:100.00% D:0.00%

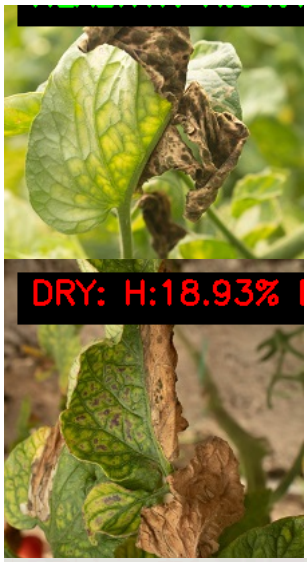

HEALTHY: H:99.39% D:0.61%


HEALTHY: H:88.73% D:11.27%




DRY: H:28.16% D:71.84%

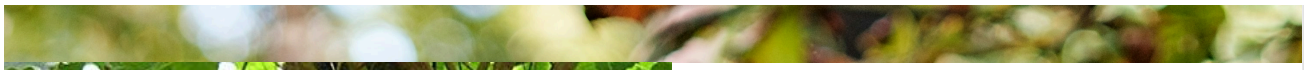

HEALTHY: H:96.02% D:3.98%


DRY: H:18.93%


HEALTHY: H:99.83% D:0.17%

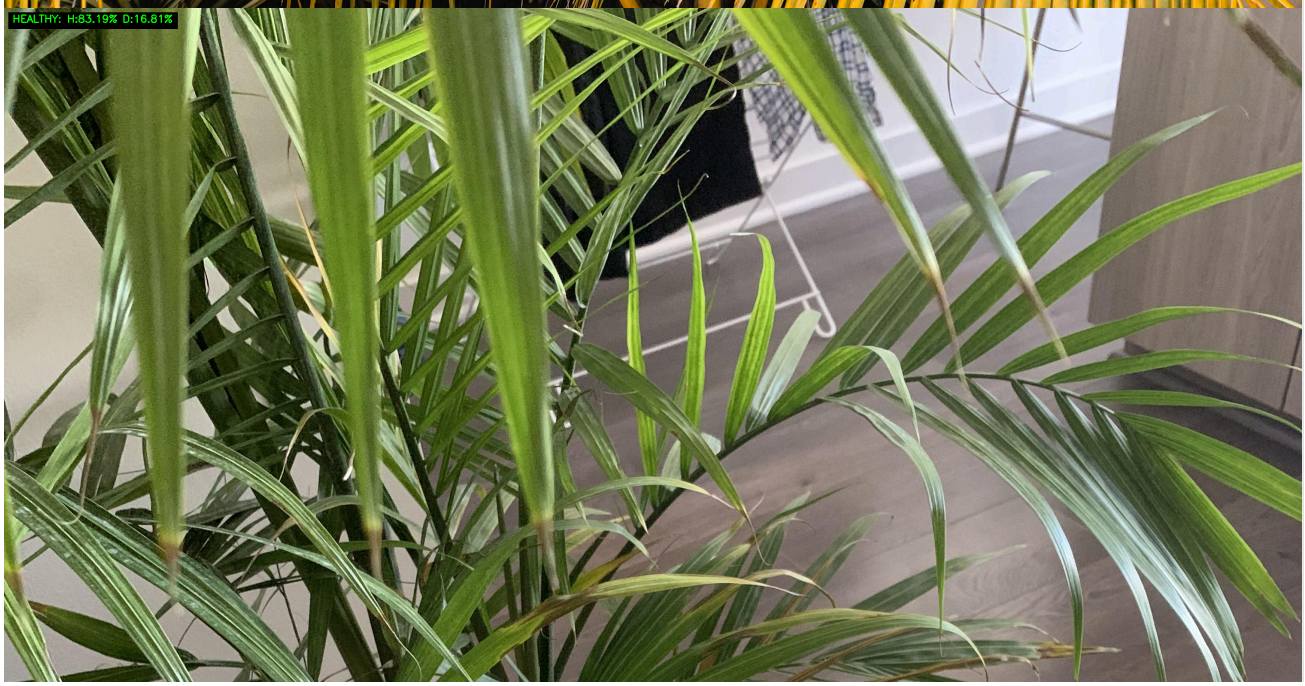

HEALTHY: H:93.23% D:6.77%

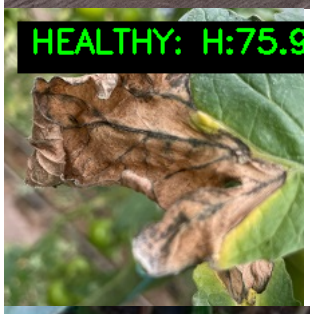

HEALTHY: H:74.33% D:25.67%

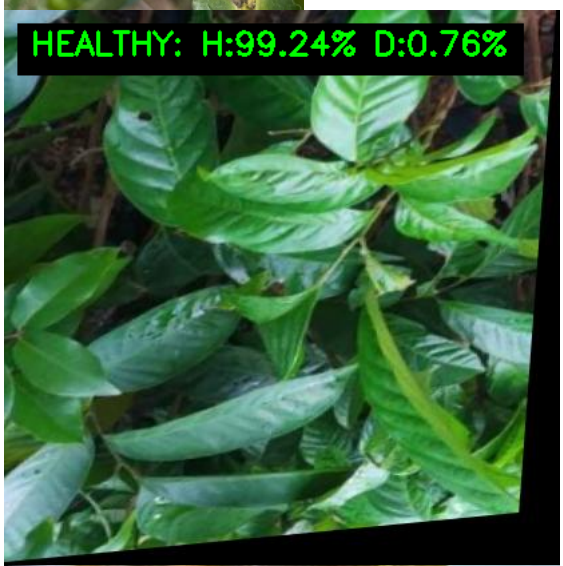



HEALTHY: H:95.49% D:4.51%

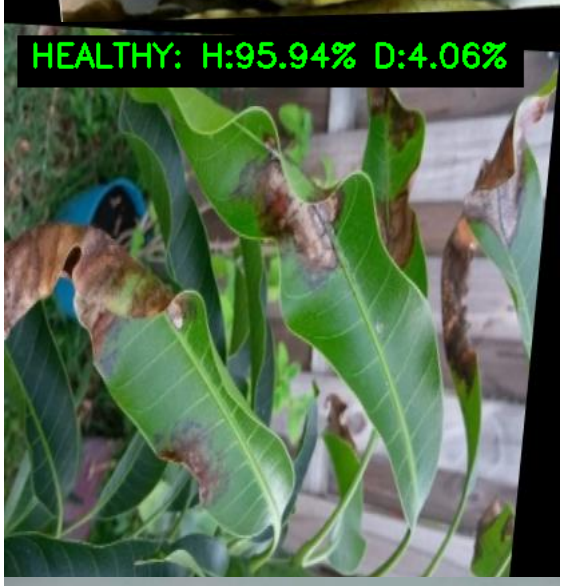

HEALTHY: H:66.17% D:33.83%


HEALTHY: H:83.19% D:16.81%


HEALTHY: H:75.9


HEALTHY: H:99.24% D:0.76%


DRY: H:4.08% D:95.92%


HEALTHY: H:95.94% D:4.06%


HEALTHY: H:100.00% D:0.00%


HEALTHY: H:100.00% D:0.00%


DRY: H:0.00% D:100.00%


```
Email sent successfully with dry leaf images!
```

Start coding or generate with AI.