```
from google.colab import files
files.upload()
```

Show hidden output

```
!pip install rasterio scikit-image
```

Show hidden output

```
import numpy as np
import rasterio
from skimage import filters, morphology, exposure

# ------------------------------------------------
# 1. FILENAMES (exact Colab names)
# ------------------------------------------------
FILES = {
    'red': 'red (3).tif',
    'green': 'green (3).tif',
    'blue': 'blue (3).tif',
    'nir': 'nir (3).tif',
    'red_edge': 'red edge (3).tif'
}

# ------------------------------------------------
# 2. LOAD BAND USING RASTERIO
# ------------------------------------------------
def load_band(path):
    with rasterio.open(path) as src:
        data = src.read(1).astype(np.float32)

        # Replace NoData with 0
        if src.nodata is not None:
            data[data == src.nodata] = 0

        # Normalize
        return data / (np.max(data) + 1e-6)

# ------------------------------------------------
# 3. MAIN FUNCTION
# ------------------------------------------------
def calculate_weed_matrix():
    print("--- Starting Weed Detection ---")

    bands = {}

    # Load all bands
    for key, fname in FILES.items():
        try:
            bands[key] = load_band(fname)
            print(f"✓ Loaded {fname} | shape={bands[key].shape}")
        except Exception as e:
            print(f"✗ Failed to load {fname}: {e}")
            return None

    # ------------------------------------------------
    # 4. VEGETATION INDICES
    # ------------------------------------------------
    wsri = (bands['red_edge'] - bands['red']) / (
        bands['red_edge'] - bands['blue'] + 1e-6
    )

    ndre = (bands['nir'] - bands['red_edge']) / (
        bands['nir'] + bands['red_edge'] + 1e-6
    )

    ndvi = (bands['nir'] - bands['red']) / (
        bands['nir'] + bands['red'] + 1e-6
    )

    # ------------------------------------------------
    # 5. VEGETATION MASK
    # ------------------------------------------------
    veg_thresh = filters.threshold_otsu(ndvi)
    veg_mask = ndvi > veg_thresh
```

```
    veg_mask = ndvi > veg_thresh

    if np.sum(veg_mask) == 0:
        print("⚠ No vegetation detected")
        return []

    # ------------------------------------------------
    # 6. WEED SCORE
    # ------------------------------------------------
    weed_score = (wsri * 0.7) + (ndre * 0.3)
    weed_score = np.where(veg_mask, weed_score, 0)

    # ------------------------------------------------
    # 7. THRESHOLD + CLEANUP
    # ------------------------------------------------
    threshold = np.percentile(weed_score[veg_mask], 85)
    weed_mask = (weed_score > threshold) & veg_mask

    weed_mask = morphology.binary_opening(
        weed_mask, morphology.disk(1)
    )

    # ------------------------------------------------
    # 8. EXTRACT MATRIX [row, col, intensity]
    # ------------------------------------------------
    coords = np.argwhere(weed_mask)

    intensity_map = exposure.rescale_intensity(
        weed_score * weed_mask, out_range=(0, 1)
    )

    matrix = []
    for r, c in coords:
        intensity = float(intensity_map[r, c])
        if intensity > 0.1:
            matrix.append([int(r), int(c), round(intensity, 4)])

    return matrix

# ------------------------------------------------
# 9. RUN
# ------------------------------------------------
result = calculate_weed_matrix()

if result:
    print("\n--- RESULT ---")
    print("Total weeds detected:", len(result))
    print("First 10 rows:")
    for row in result[:]:
        print(row)

    with open("weed_detection_matrix.txt", "w") as f:
        f.write(str(result))

    print("\n✓ Saved weed_detection_matrix.txt")
else:
    print("No weeds detected.")
```

```
--- Starting Weed Detection ---
✓ Loaded red (3).tif | shape=(100, 100)
✓ Loaded green (3).tif | shape=(100, 100)
✓ Loaded blue (3).tif | shape=(100, 100)
✓ Loaded nir (3).tif | shape=(100, 100)
✓ Loaded red edge (3).tif | shape=(100, 100)

--- RESULT ---
Total weeds detected: 34
First 10 rows:
[21, 42, 0.1895]
[22, 41, 1.0]
[22, 42, 0.2484]
[22, 43, 0.2416]
[23, 42, 0.2289]
[42, 26, 0.1812]
[42, 27, 0.1817]
[42, 28, 0.1856]
[43, 25, 0.1858]
[43, 26, 0.1995]
[43, 27, 0.2037]
```

```
[43, 28, 0.2487]
[43, 29, 0.2823]
[44, 26, 0.2108]
[44, 27, 0.2274]
[44, 28, 0.3764]
[44, 29, 0.7036]
[44, 30, 0.1969]
[45, 29, 0.3012]
[69, 27, 0.1794]
[69, 28, 0.2066]
[70, 26, 0.2848]
[70, 27, 0.241]
[70, 28, 0.2088]
[70, 29, 0.2308]
[70, 42, 0.2137]
[71, 27, 0.318]
[71, 28, 0.2468]
[71, 29, 0.1928]
[71, 41, 0.3876]
[71, 42, 0.324]
[71, 43, 0.1983]
[72, 28, 0.238]
[72, 42, 0.2393]

✓ Saved weed_detection_matrix.txt
```

```
import numpy as np
import matplotlib.pyplot as plt
import rasterio

def visualize_results(matrix_result):
    print("--- Preparing Visualization ---")

    # ------------------------------------------------
    # 1. LOAD RGB BANDS SAFELY (GeoTIFF)
    # ------------------------------------------------
    try:
        with rasterio.open('red (1).tif') as src:
            r = src.read(1).astype(np.float32)
        with rasterio.open('green (1).tif') as src:
            g = src.read(1).astype(np.float32)
        with rasterio.open('blue (1).tif') as src:
            b = src.read(1).astype(np.float32)
    except Exception as e:
        print(f"Error loading RGB bands: {e}")
        return

    # ------------------------------------------------
    # 2. RGB COMPOSITE + CONTRAST STRETCH
    # ------------------------------------------------
    rgb = np.stack([r, g, b], axis=-1)

    p2, p98 = np.percentile(rgb, (2, 98))
    rgb = np.clip(rgb, p2, p98)
    rgb = (rgb - p2) / (p98 - p2 + 1e-6)

    # ------------------------------------------------
    # 3. EXTRACT WEED POINTS
    # ------------------------------------------------
    if not matrix_result:
        print("No weeds detected.")
        return

    rows = np.array([pt[0] for pt in matrix_result])
    cols = np.array([pt[1] for pt in matrix_result])
    intensities = np.array([pt[2] for pt in matrix_result])  # 0 → 1

    # Normalize intensities for colormap
    intensities = np.clip(intensities, 0, 1)

    # ------------------------------------------------
    # 4. PLOT
    # ------------------------------------------------
    fig, ax = plt.subplots(1, 2, figsize=(20, 10))

    # Left: Original RGB
    ax[0].imshow(rgb)
    ax[0].set_title("Original RGB Image", fontsize=15)
    ax[0].axis('off')
```

```
        # Right: RGB + Weed Points (depth-based color)
        sc = ax[1].imshow(rgb)
        points = ax[1].scatter(
            cols,
            rows,
            c=intensities,        # depth controls color
            cmap='plasma',          # light → dark red
            s=4,                  # point size
            alpha=0.9
        )

        ax[1].set_title(f"Weed Detection (Depth Encoded) | {len(rows)} points", fontsize=15)
        ax[1].axis('off')

        # Colorbar
        cbar = plt.colorbar(points, ax=ax[1], fraction=0.035)
        cbar.set_label("Weed Depth / Intensity", rotation=270, labelpad=15)

        plt.tight_layout()
        plt.show()

    # ------------------------------------------------
    # RUN VISUALIZATION
    # ------------------------------------------------
    visualize_results(result)
```

```
--- Preparing Visualization ---
```