Addition to the second part of the project.
I would like to bring some computations to evaluate the success of the provided algorithm.
All of the results can be found in the attached file “result.txt”. (for all 900 images)
In my documentation the algorithm was named as Second task First approach.

First of all, I import the multi label which is the ground truth and rgbm which is the prediction.
My labeling result from Second approach. The results are taken from folder ‘result2’.
The next step is detecting all distinct colors in each image and removing black.

```
# reading label
label = cv2.imread('multi_label/label_0' + image_title + '.png',
                   cv2.IMREAD_COLOR)
# reading rgb prediction mask result
rgbm = cv2.imread('project1/result2/rgbm_0' + image_title + '.png')

# detecting colors in the multi labels
colorinlabel = []
colorsinrgbm = []
for i in label:
    for j in i:
        if j.tolist() not in colorinlabel:
            colorinlabel.append(j.tolist())
# detecting colors in the predicted labels
for i in rgbm:
    for j in i:
        if j.tolist() not in colorsinrgbm:
            colorsinrgbm.append(j.tolist())

# removing black from both
colorinlabel.remove([0, 0, 0])
colorsinrgbm.remove([0, 0, 0])
```

After that thresholding images to get binary mask for each prediction and ground truth.

```
# thresholding each color in the ground truth
# and getting the desired mask
truth = colorinlabel.copy()
for i in range(len(colorinlabel)):
    color = np.array(colorinlabel[i])
    truth[i] = cv2.inRange(label, color, color)

# thresholding in predictions
predictions = colorsinrgbm.copy()
for i in range(len(colorsinrgbm)):
    color = np.array(colorsinrgbm[i])
    predictions[i] = cv2.inRange(rgbm, color, color)
```

In the next snippet of the code I do computations for each label I have in predictions. Since I would like to have an automated way of calculating results and interfering as less as possible I decided to check each target label in the ground truth labels.
I create the list of calculated assessment metrics and choose the maximum value for each leaf label.

```
# computation of assessment metrics
iouforpredictions = []
diceforpredictions = []
for i in predictions:
    ioufori = []
    dicefori = []
    for j in truth:
        # iou
        intersection = np.logical_and(i, j)
        union = np.logical_or(i, j)
        iou_score = np.sum(intersection) / np.sum(union)
        ioufori.append(iou_score)

        # dice
        im1 = np.asarray(i).astype(np.bool)
        im2 = np.asarray(j).astype(np.bool)
        intersection1 = np.logical_and(im1, im2)
        dice = 2. * intersection1.sum() / (im1.sum() + im2.sum())
        dicefori.append(dice)

    iouforpredictions.append(max(ioufori))
    diceforpredictions.append(max(dicefori))
```

As the final step I create a dictionary for a better collection and ordering of results.

```
# creating a dictionary consisting of rgb colors as keys and
# success of labeling as values [IoU, Dice]
score_for_each_mask = {}
sumdice = 0
sumiou = 0
for i in range(len(iouforpredictions)):
    sumdice = sumdice + diceforpredictions[i]
    sumiou = sumiou + iouforpredictions[i]
    score_for_each_mask[str(colorsinrgbm[i])] = [iouforpredictions[i],
diceforpredictions[i]]

n = len(iouforpredictions)
score_for_plant = [sumiou / n, sumdice / n]
```

So by the provided code I was able to assess my algorithm.
All the results can be found in result.txt.
I would like to mention here some of them. The total IoU and total dice coefficients are 0.566 and 0.681 correspondingly.

Plant	IoU	Dice
1 (indexed as 0)	0.662	0.768
2 (indexed as 1)	0.445	0.567
3 (indexed as 2)	0.652	0.754
4 (indexed as 3)	0.495	0.631
5 (indexed as 4)	0.575	0.683

Besides, I would like to provide the IoU and Dice scores for the image that is in the documentation.

For rgbm_02_00_000_00.png:

```
{
'[0, 255, 0]': [0.7153392330383481, 0.8340498710232158],
'[255, 0, 0]': [0.7601990049751244, 0.8637648388920294]
}
```

For rgbm _ 00_02_008_02.png:

```
{
'[0, 0, 255]': [0.945748449955713, 0.972117901445317],
'[255, 255, 0]': [0.8903953189667475, 0.9420202325230258],
'[0, 255, 0]': [0.8024766721897619, 0.8904155982389085],
'[255, 0, 0]': [0.9059835869651821, 0.9506730206504473]
}
```

Result per leaf mask in 1_00_007_02

```
{
'[0, 0, 255]': [0.7058591282375237, 0.8275702448733973],
'[255, 255, 0]': [0.21286419359974526, 0.35101076398004727],
'[0, 255, 0]': [0.6675076211500053, 0.8006051818697598],
'[255, 0, 0]': [0.8626699629171817, 0.9262724799256752]
}
```

The whole code can be found in assessment.py.