186

Activity 13: Logistic Regression

Mary Chris Go / 2014 11122

Goal

To be able to apply logistic regression to train an artificial neuron to give the degree of ripeness of that the fruit

Mary Chris Go / 2014 11122

Perceptron Algorithm

This algorithm is known to perform classification through supervised learning given that the classes are linearly separable in feature space. As I only made a 2D feature space, we expect the decision boundary to be a line.

Logistic Regression

In here, we introduce the logit function which is:

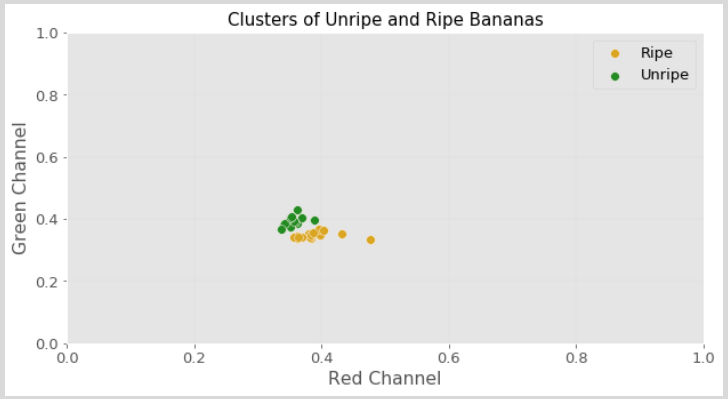
$$logit(p) = \log \frac{p}{1-p}$$

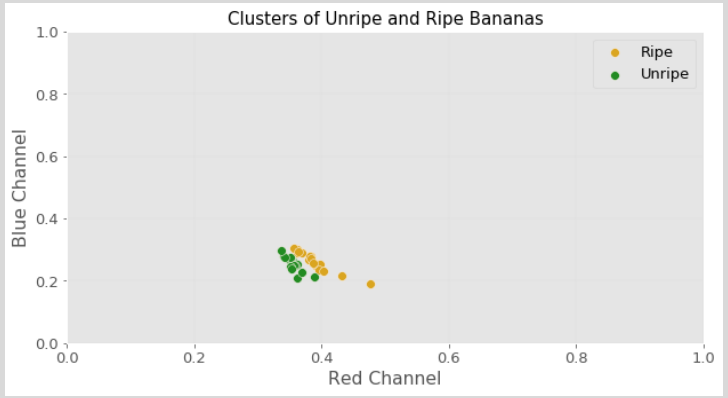
The output is only one of two states (1, 0). We wanted something that output the probability that the input belongs to a certain class.

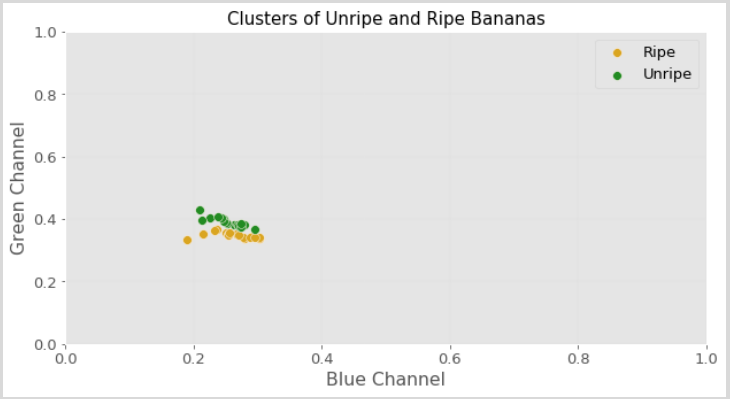
```
for j in range(3):
    filenames = os.listdir(dirs[j])
    for i,f in enumerate(filenames):
        if i == 50:
            break
        #obtaining eccentricity through otsu's method
        fruit_img = cv.imread(dirs[j] + f)
        fruit_img_gray = cv.cvtColor(fruit_img, cv.COLOR_BGR2GRAY)
        threshold, out = cv.threshold(fruit_img_gray, 127, 255, cv.THRESH_OTSU
        out = (fruit_img_gray < threshold).astype(float)

        #Labeling per fruit
        fruit_img_label = meas.label(out)
        props = meas.regionprops(fruit_img_label)
        ecc = props[0]['eccentricity']

        # getting l*a* b*
        img_cielab = cv.cvtColor(img, cv.COLOR_BGR2Lab).astype(float)
        img_cieab /= img_cielab[:,:,0].max()
        fruit_img_L, fruit_img_a, fruit_img_b = cv.split(img_cielab)

        #Append values in the array
        a_fruit[j].append(fruit_img_a.mean())
        b_fruit[j].append(fruit_img_b.mean())
        eccentricity[j].append(ecc)
```

```
def get_line_params(self):
    W = self.W
    A, B, C = W[1], W[2], -W[0]
    m = -A/B
    b = C/B
    return m, b
```

Feature Space





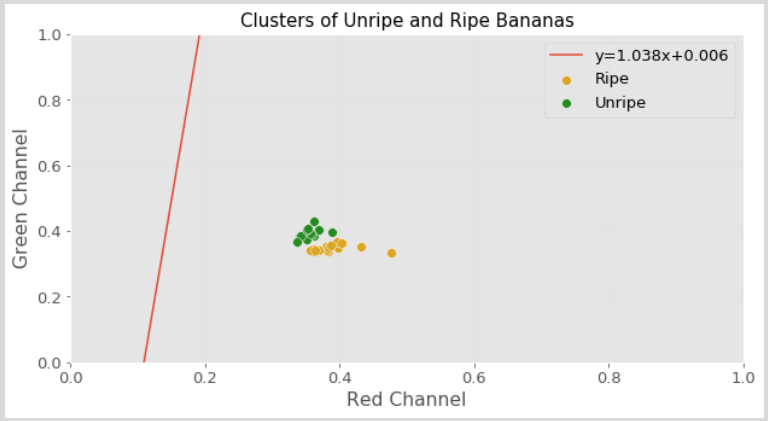


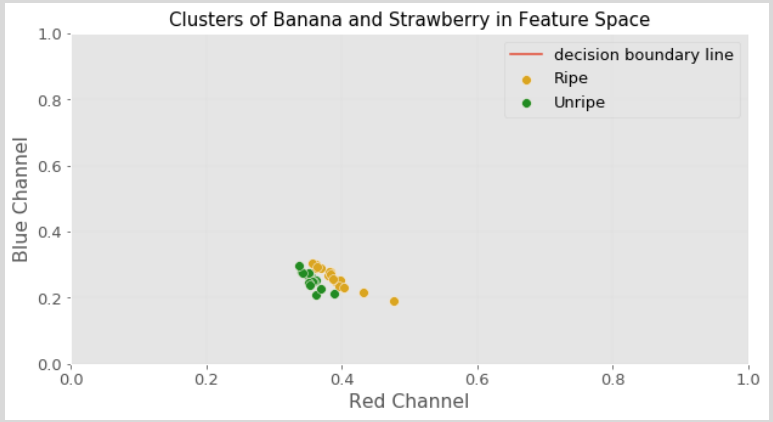

```
for file in glob.glob(location_rp):
    im_rp = cv2.cvtColor(cv2.imread(file), cv2.COLOR_BGR2RGB).astype(float)
    c_rp = NCC(im_rp)
    r2_ave = c_rp[0].mean()
    g2_ave =c_rp[1].mean()
    b2_ave = c_rp[2].mean()
    r_rp.append(r2_ave)
    g_rp.append(g2_ave)
    b_rp.append(b2_ave)
```

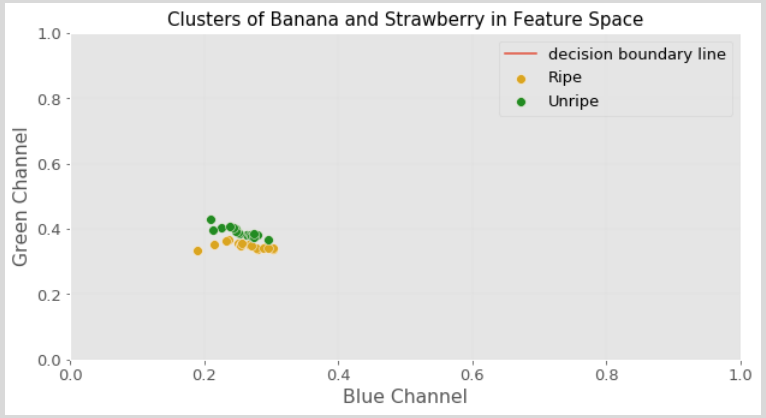
The code was to determine the placement of rgb channels on the feature space.

Mary Chris Go / 2014 11122

Banana and Mango



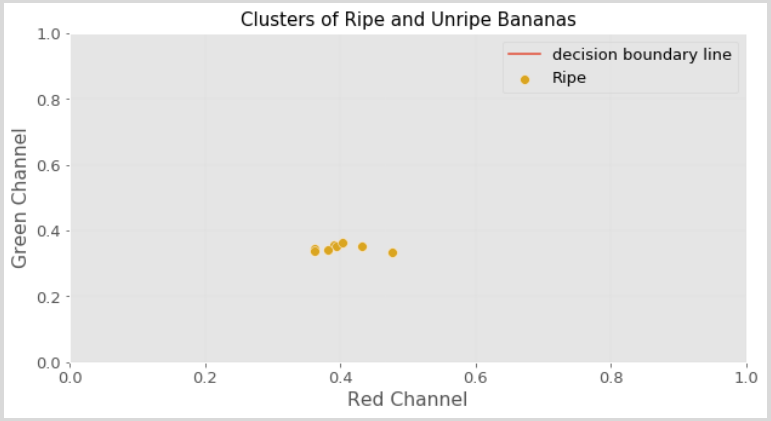


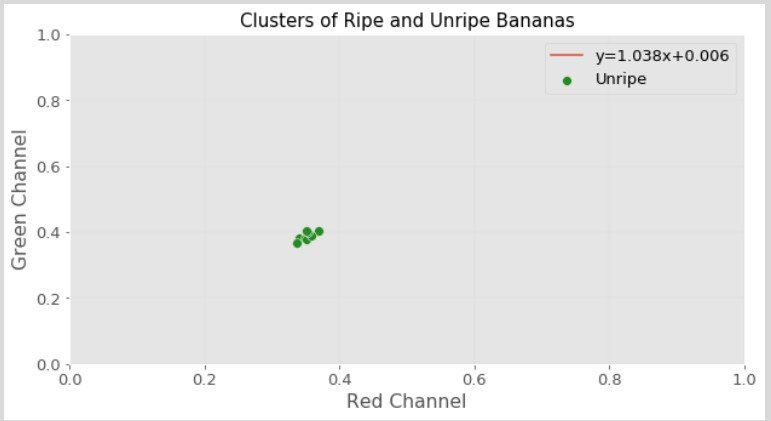


For some apparent reason, my code doesn't work for small dataset. I had 15 ripe and 15 unripe bananas which will lead to a series of errors.

Mary Chris Go / 2014 11122

Test






```
[0.9999626400297533, 0.9999716362042899, 0.9999693863572039, 0.9999680607145639, 0.9999592956251065, 0.9999630745274062, 0.9999704056664867, 0.999971959235563, 0.9851517695252546, 0.9866926114216733, 0.9830868929748109, 0.9792854465851869, 0.9704838868608351, 0.9683656358188235, 0.9667339411561624, 0.9645620215615867, 0.9998709247200426, 0.9998868584016485, 0.9998859451574921, 0.9998814826890247, 0.9998615181137961, 0.9998687295165699, 0.9998756427034015, 0.9998920654862348, 0.950498442368622, 0.9492184892831561, 0.9404345238275545, 0.9284897893830587, 0.9083794327531882, 0.8992409795281012, 0.8793718773142573, 0.8830825399720702, 0.9995863455740098, 0.9995850578380449, 0.9996077578686317, 0.9995940126091697, 0.9995628547119133, 0.9995682180394052, 0.9995214968116993, 0.9996175700913862, 0.8569095588418462, 0.8380156960980905, 0.8260275933053609, 0.7997961572150091, 0.7708998561621097, 0.7484157240228095, 0.6834346193393052, 0.7143031620364338]
z =  0.945339241376003
Red, Green : y =  0.468654240868982 x + -1.3220588919990255
```

```
[0.9993895546132961, 0.999453711869632, 0.9994081183652175, 0.9994086647464419, 0.9994299227259582, 0.9994258502742863, 0.9994246449748124, 0.9993781761538548, 0.8024156842437368, 0.7985936070072541, 0.7628491393843133, 0.7402127356790297, 0.7213848737140367, 0.6938609595131633, 0.6669843020822003, 0.636904376247212, 0.9984524008276864, 0.9985185476792544, 0.9984817449735499, 0.9984876475978337, 0.9985058219100189, 0.9984873288028757, 0.9984801647358786, 0.9984651302376565, 0.615061106898019, 0.6006432282819174, 0.570964713050884, 0.5491913234358653, 0.5275636085958064, 0.5012547625173844, 0.4779069716978181, 0.46579375041214494, 0.9969428478030267, 0.9969264870531218, 0.99697868573876, 0.99699886375877, 0.9969827588477359, 0.9969357580021124, 0.9969173397572708, 0.9970403251195246, 0.44547374573546633, 0.42528057793111257, 0.412223156396733, 0.397435815504345, 0.37879910787696675, 0.3592916262813012, 0.34308655405907557, 0.34568072757000945]
z =  0.7749536864725911
Red, Green : y =  0.10450975393728158 x + 2.8999228253164264
```

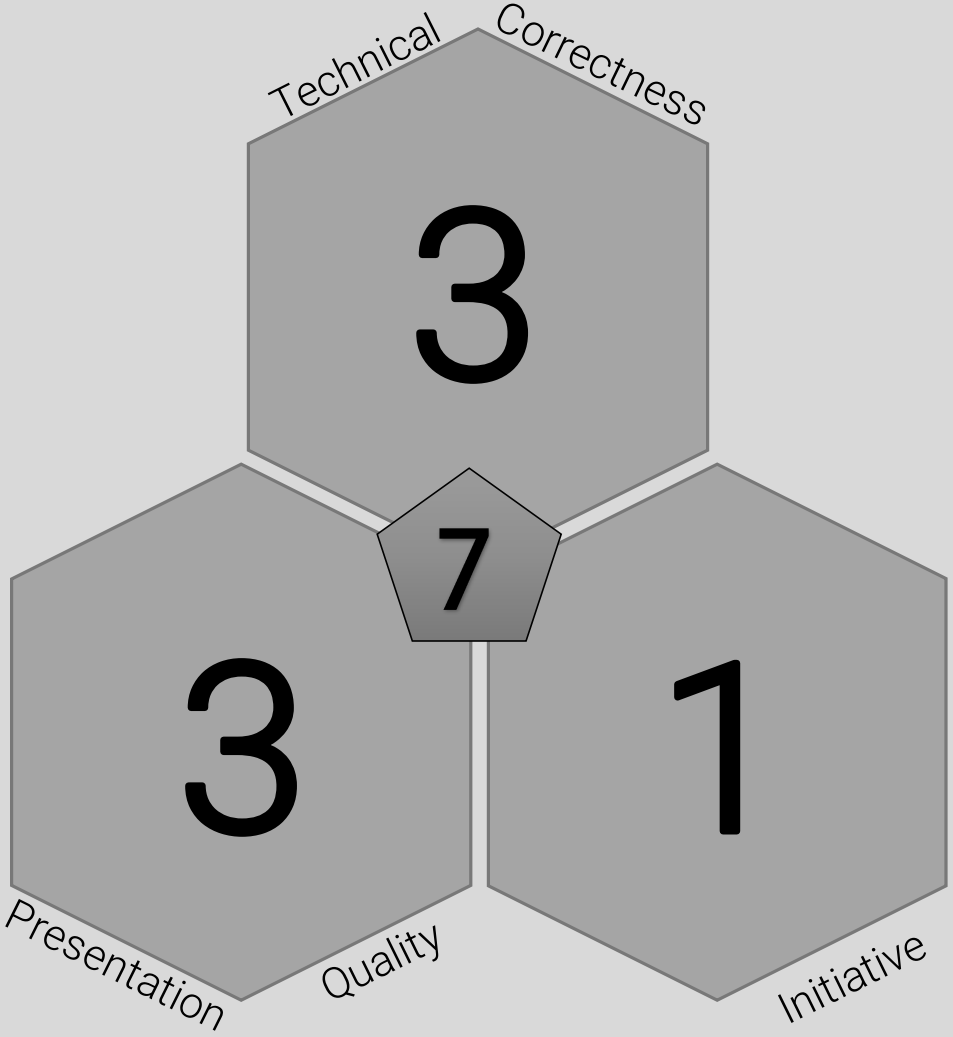
Now we test. Images were taken from the internet. These images were not yet seen by the neuron. Surprisingly, if we check the previous graphs, the location of the points where along the expected location.

Summary

The hardest of all. I'm not really sure if I was able to make it right. Though, I kind of made it possible to cluster them by testing them with new photos.

All in all, it was hard and confusing.

Self-Evaluation



References

- Soriano, M., “Logistic Regression”. 2019