วิชา: Digital Image Processing

เสนอโดย:

ผศ.ดร.โอฬาริก สุรินต๊ะ

จัดทำโดย:

นายศักดิ์สิทธิ์ รัตนชวานนท์ รหัส: 58011221034

นายเจตวรรษ ธำรงทรัพย์ รหัส: 58011211040

Mr. Sovan Pen รหัส: 58011211166

Mr. Dara Phon รหัส: 58011211167

สาขาเทคโนโลยีสารสนเทศ คณะวิทยาการสารสนเทศ

มหาวิทยาลัยมหาสารคาม

License Plate Recognition

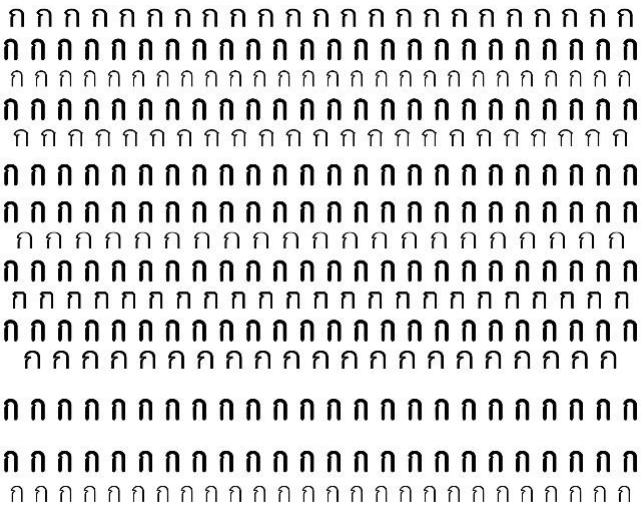
License Plate Recognition เป็นโปรแกรมอ่านป้ายทะเบียนรถยนต์อัตโนมัติ คือ ระบบจะทำการครอปป้ายทะเบียนรถออกจากรถมาก่อนแล้วจะทำการวิเคราะห์ และถอดเป็นตัวหนังสือเพื่อให้ง่ายต่อการบริหารจัดการร่วมกับระบบอื่นๆ

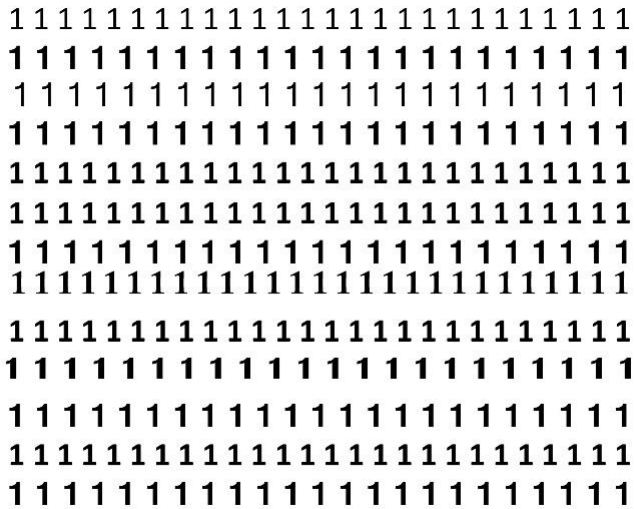
Dataset

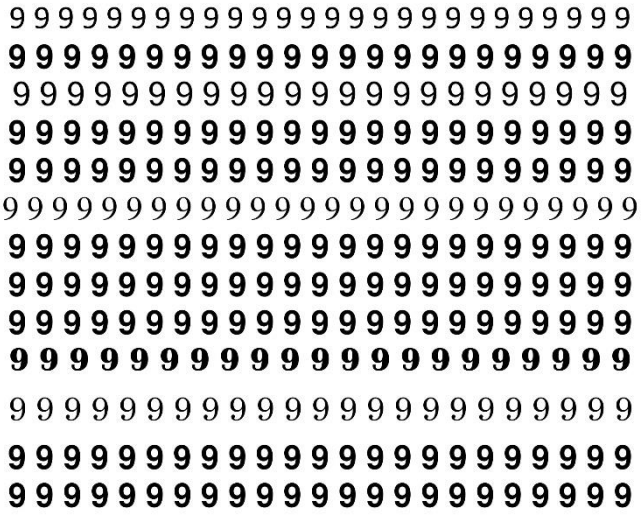
Dataset เป็นชุดข้อมูลที่เราจะเตรียมไว้เพื่อจะเอามาใช้ในกระบวนการต่างๆ ซึ่งจะเป็นดังต่อไปนี้

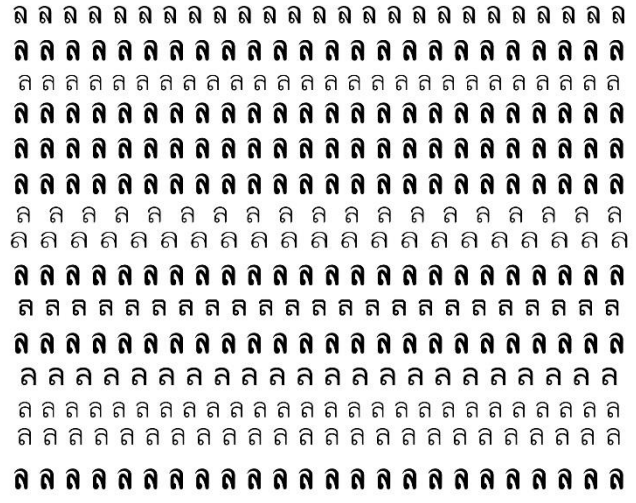
1.รูปรถที่มีป้ายทะเบียน: ถ่ายมาจากสถานที่ต่างๆ

2. ข้อมูลตัวอักษร: นำตัวพยัญชนะไทย และตัวเลขมาทำเป็นรูปภาพ ซึ่งมีทั้งหมดดจำนวณ 58 รูป ในหนึ่งรูปภาพก็จะมีตัวอักษรหลายร้อยตัวเพื่อจะนำมาสร้างเป็นโมเดลดังรูปด้านล่าง









Preprocessing

1.การครอปป้ายทะเบียน

1.1 กระบวนการรู้จำป้ายทะเบียน

โหลด directory ที่ชื่อว่า dlib เพื่อใช้ tool ที่ชื่อว่า imlab ในการสร้าง file “.xml”

วิธีการติดตั้ง directory dlib

$ git clone https://github.com/davisking/dlib.git

วิธีการติดตั้ง imlab tool

$ mkdir build

$ cd build

$ cmake ..

$ cmake --build . --config Release

$ sudo make install

วิธีการสร้าง file “.xml”

$ imglab -c mydataset100.xml /home/dara/Desktop/project/carTrain100

หลังจากสร้าง file เราจะได้ file ที่ชื่อว่า “mydataset100.xml” ที่มีข้อมูลรูปภาพจากโฟลเดอร์ carTrain100 ซึ่งมีรูปทั้งหมด 100 รูป

ตัวอย่าง file “.xml”

mydataset100.xml

เปิด ไฟล์ “.xml” ด้วย imglab tool ขึ้นมาเพื่อทำการสร้าง bounding boxes ในแต่ละภาพ

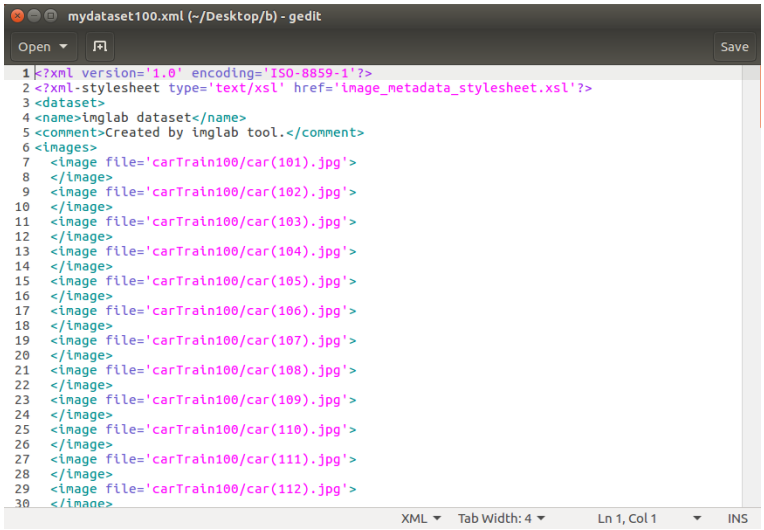
วิธีเปิดไฟล์ “.xml” ด้วย imglab tool

$ imglab mydataset100.xml

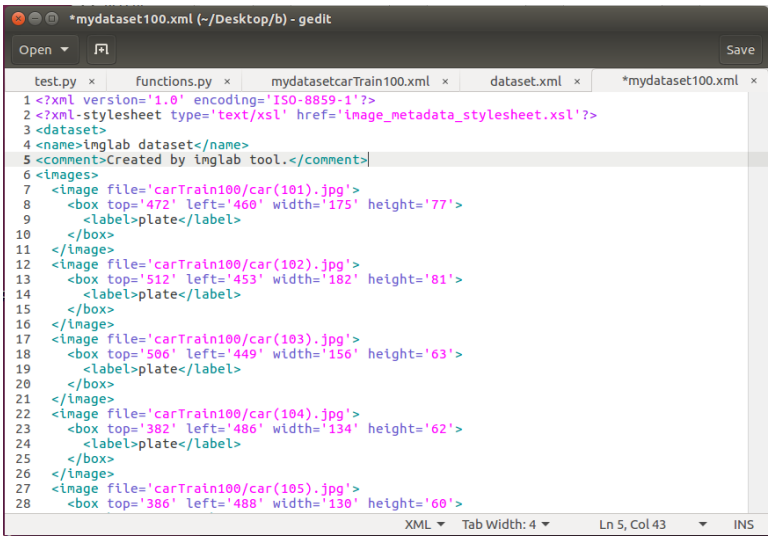
ทำการสร้าง bounding box ดังรูป

เมื่อสร้าง bounding boxes ครบทุกรูปก็ทำการ save

ข้อมูลในไฟล์ “.xml” ก่อนและหลังการสร้าง bounding box



ก่อน



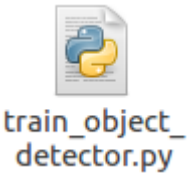
หลัง

1.2 กระบวนการ Train เพื่อสร้าง Model ป้ายทะเบียน

วิธีการสร้าง Model ป้ายทะเบียน แบบ SVM

ใช้โค้ดจากไฟล์ “train_object_detector.py” เพื่อทำการสร้าง Model ป้ายทะเบียน แบบ SVM โดยทำการ run ไฟล์ ดังนี้

```
$ python ./train_object_detector.py ../project/
```



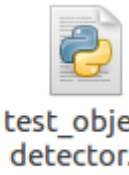
ไฟล์ “train_object_detector.py”

ไฟล์ “.svm” ที่ได้จากการรัน

1.3 กระบวนการ Test การครอปป้ายทะเบียนรถ

ใช้โค้ดจากไฟล์ “test_object_detector.py” เพื่อทำการ test การครอปป้ายทะเบียนจากโฟลเดอร์ที่มีรูปจำนวน 100 รูป โดยทำการ run ไฟล์ ดังนี้

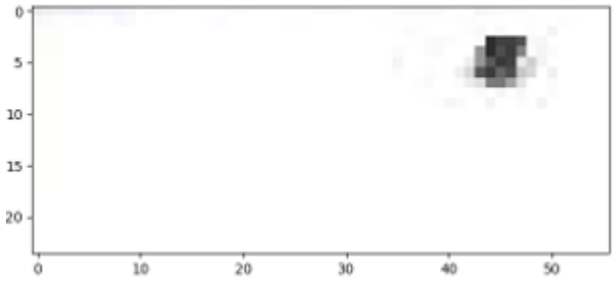
python ./test_object_detector.py ../project/CarTest100/



ไฟล์ “test_object_detector.py”

ผลจากการTest

Test ทั้งหมด 100 รูป หาไม่เจอ 12 รูป ผิดพลาด 1 รูป ครอปได้ปกติ 87 รูป



รูปที่ผิดพลาด

รูปที่ครอปได้

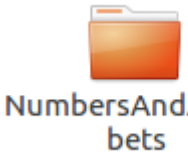
รูปที่ครอปได้

รูปที่ครอปได้

2.การสร้างโมเดลของตัวอักษร

โมเดลของตัวอักษรเป็นข้อมูลที่สำคัญที่เราจะใช้ในขั้นตอนPredict ตอนท้ายสุดเพื่อทำการเดาว่าตัวอักษรที่อยู่ในป้ายทะเบียนเป็นตัวอะไรบ้าง

รูปภาพของตัวอักษรทั้ง 58ตัวได้เก็บอยู่ใน folder ชื่อ NumbersAndAlphabets



เพื่อที่จะทำการสร้างโมเดล เราต้องทำการ run ไฟล์ที่มีชื่อว่า train.py

วิธีการ run ไฟล์ train.py

$ python ../project/train.py

แล้วขั้นตอนของการสร้างโมเดลจะมีดังต่อไปนี้

-ทำการเก็บตำแหน่งของแต่ละรูปภาพอยู่ในตัวแปร matches ซึ่งเป็นประเภท list โดยเรียกฟังก์ชั้น dir2filename() ที่อยู่ในไฟล์ function.py

```
 9 dirName = '/home/dara/Desktop/project/NumbersAndAlphabets'
10 matches = func.dir2filename(dirName)
```

train.py

```
19 def dir2filename(dirName):
20     # read files from directory
21     matches = []
22     for root, dirnames, filenames in os.walk(dirName):
23         for filename in fnmatch.filter(filenames, '*.[jp]*'):
24             matches.append(os.path.join(root,filename))
25             print('({}/{})'.format(root,filename))
26 
27 
28     print("Number of image files", len(matches))
29 
30     return matches
```

function.py

ค่าของตัวแปร matches ในแต่ละ Index ก็จะมีลักษณะประมาณนี้ ซึ่งมีทั้งหมดจำนวณ 58 ค่า โดยเริ่มตั้งแต่ index ที่ 0 ถึง 57

'/home/dara/Desktop/project/NumbersAndAlphabets/train-0.jpg'

-ทำการวนลูปโดยจำนวณรอบเท่ากับจำนวณรูปทั้งหมด และ ทำการส่งตำแหน่งของแต่ละรูปเข้าไปในฟังก์ชัน imgTrain2fv ที่อยู่ในไฟล์ function.py เพื่อทำการเรียกรูปภาพมาใช้

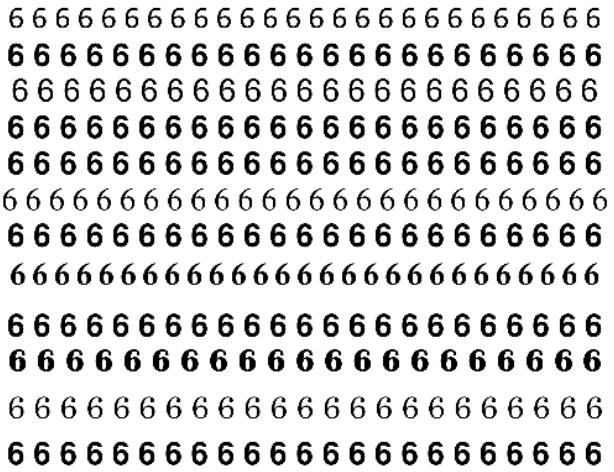
```
13 for i in(range(0,len(matches))):
14     print("#", i+1)
15     # create feature vector
16     digit_fv = func.imgTrain2fv(matches[i], img_row, img_col)
17     feature = np.vstack((feature, digit_fv))
```

train.py

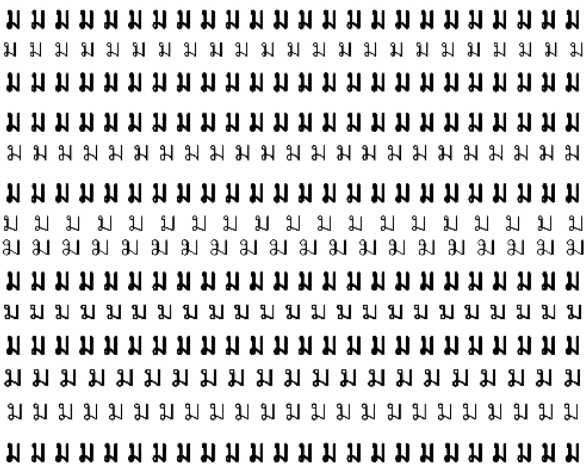
-ในฟังก์ชัน imgTrain2fv ของไฟล์ function.py โปรแกรมจะทำการเรียกรูปขึ้นมา และทำให้รูปเป็นสีเทาก่อน

```
33 def imgTrain2fv(fileName, img_row, img_col):
34
35     digit_y = fileName.split('-')[-1].split('.')[0]
36
37     digit = io.imread(fileName)
38     gray_image = color.rgb2gray(digit)
```

function.py



รูปสีเทา

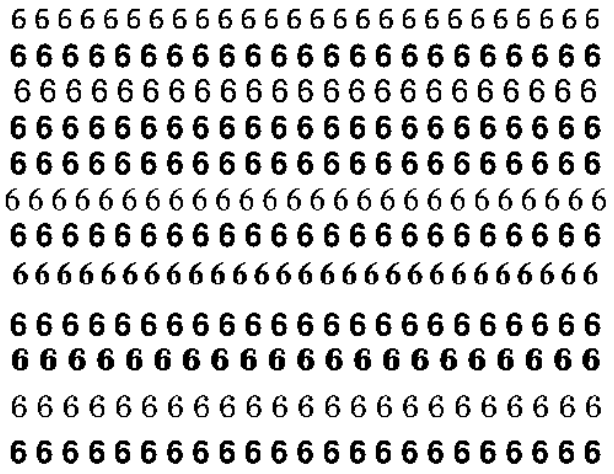


รูปสีเทา

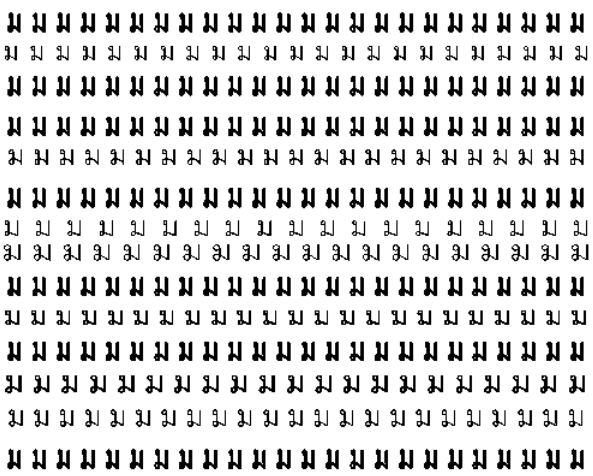
-ต่อไปเราต้องทำให้รูปเป็นสีเทาเป็นสีขาว-ดำโดยใช้ threshold_mean

```
41     thresh = filters.threshold_mean(gray_image)
42     binary_image = gray_image > thresh
```

function.py



รูปขาว-ดำ

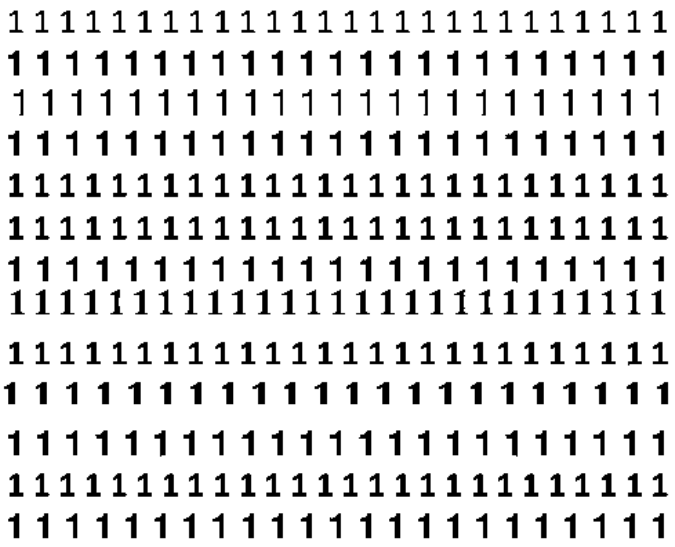


รูปขาว-ดำ

-ใชฟังก์ชัน median() เพื่อทำการ reduce noise เพราะว่าถ้าเราไม่ได้นำฟังก์ชันนั้นมาช่วยเวลาทำการค้นหา object ของบางรูป โปรแกรมก็จะค้นหาเจอ object ที่เป็นจุดเล็กๆที่ไม่สามารถมองด้วยตาได้ ซื่อจะทำให้โมเดลของเราเกิดข้อผิดพลาดเวลาเรานำค่าของ object นั้นมาสร้างเป็นโมเดลด้วย

```
50          r = 1
51      binary_image = median(binary_image, disk(r))
```

function.py



เลข 1 มีจำนวณ 317 ตัว

ไม่ได้ใช้ median()

Object ที่เจอทั้งหมดมี 334 object

```
('digit_y is ', '1')
('Number of objects = ', 334)
```

ใช้ median()

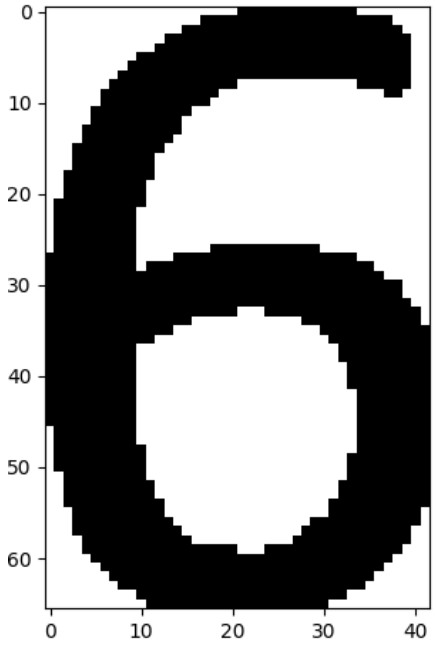
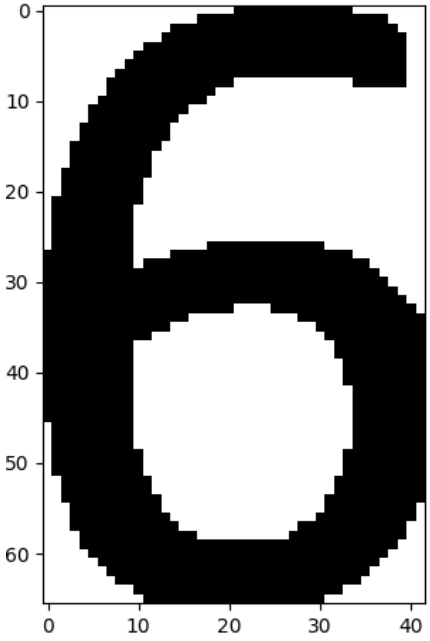
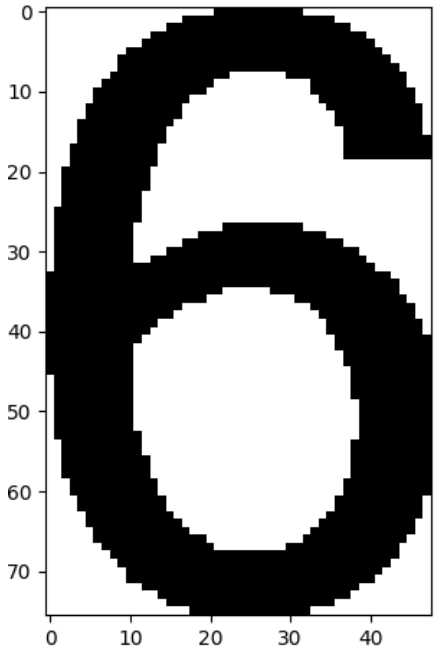
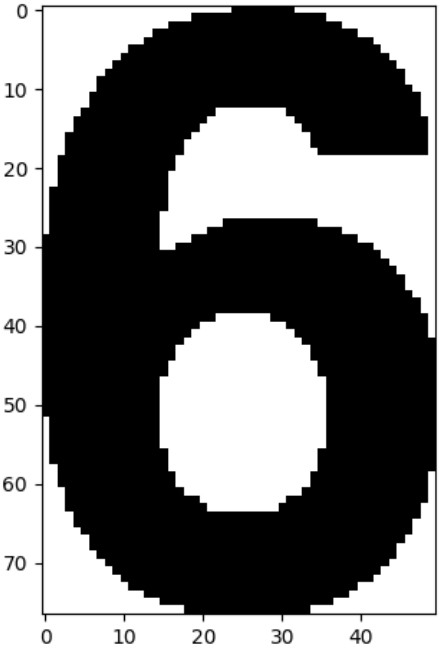
Object ที่เจอทั้งหมดมี 317 object

```
('digit_y is ', '1')
('Number of objects = ', 317)
```

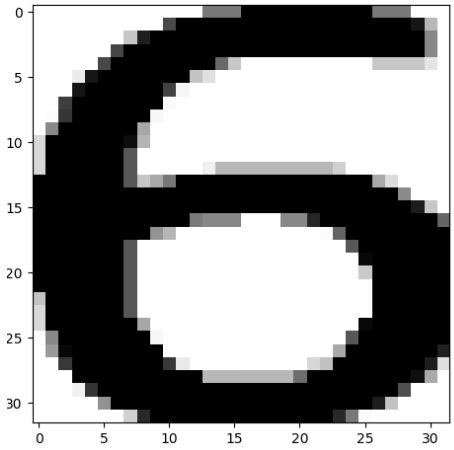
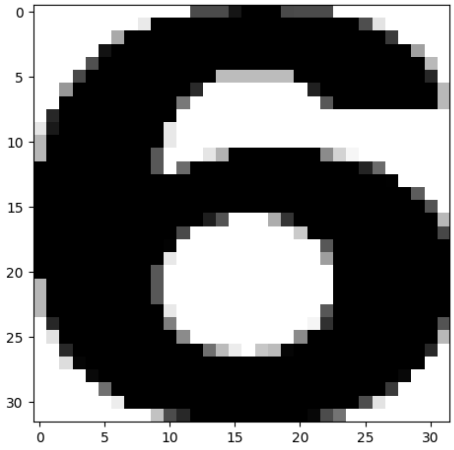
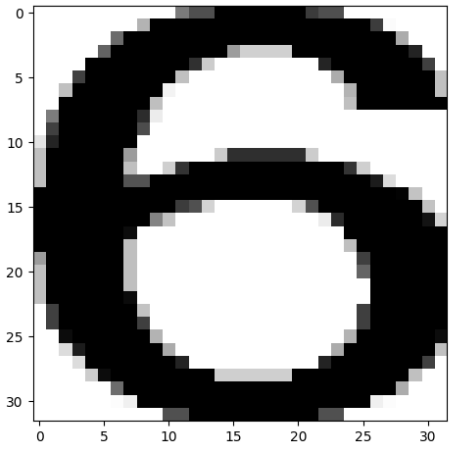
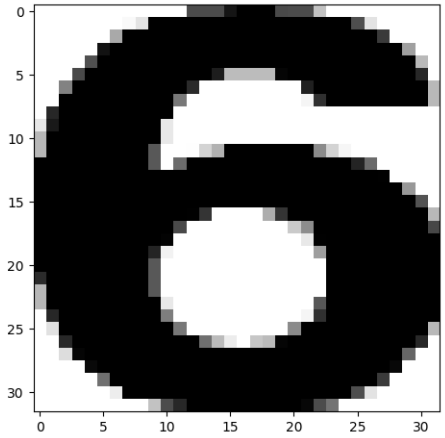
-ทำการค้นหา object โดย label_objects เก็บค่า object ทั้งหมด, nb_labels เป็นจำนวณ object ทั้งหมด

```
52    label_objects, nb_labels = ndi.label(np.invert(binary_image))
```

-วนลูปโดยจำนวณรอบเท่ากับจำนวณ object ที่เจอเพื่อจะจัดการกับแต่ละ object และครอบ object นั้มมา

เนื่องจากแต่ละ object ที่ครอบมามีจำนวณ pixel ของความสูง ความกว้างและจำนวณ pixel ทั้งหมดไม่เท่ากัน ดังนั้นเราจึงต้องทำการ resize ขนาด object ทั้งหมดให้มีขนาดเท่ากัน ในที่นี้ผมกำหนดเป็น 32*32 pixel

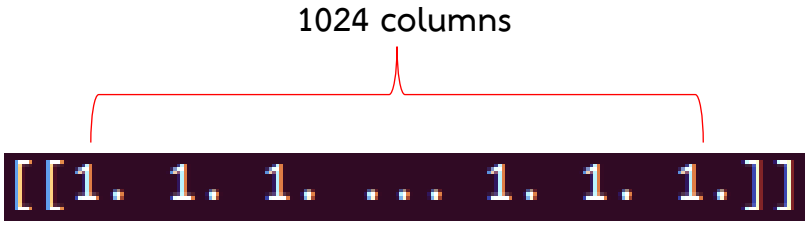
ค่าของpixel ของแต่ละ object ที่ครอบมาหลังจากทำการ resize เสร็จเรียบร้อยแล้วจะมีค่าเป็นประมาณนี้

32 columns

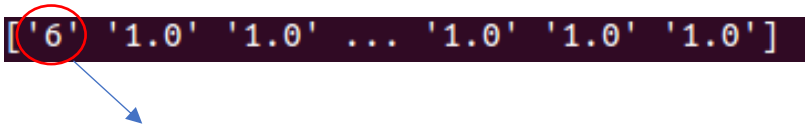
32 rows

```
[[1.         1.         1.         ... 1.         1.         1.        ]
 [1.         1.         1.         ... 0.08886719 0.72167969 1.        ]
 [1.         1.         1.         ... 0.         0.53125    1.        ]
 ...
 [1.         1.         1.         ... 1.         1.         1.        ]
 [1.         1.         1.         ... 1.         1.         1.        ]
 [1.         1.         1.         ... 1.         1.         1.        ]]
```

เพื่อที่จะนำค่า pixel มาสร้างเป็นค่า Feature Vector เราต้องทำการ reshape ค่า pixel โดยนำค่า pixel ของแต่ละแถวมาต่อกันให้เป็นแถวเดียวซึ่งจะทำให้ขนาดของคอลัมน์มีทั้งหมดจำนวน 32x32=1024 ดังรูปด้านล่าง

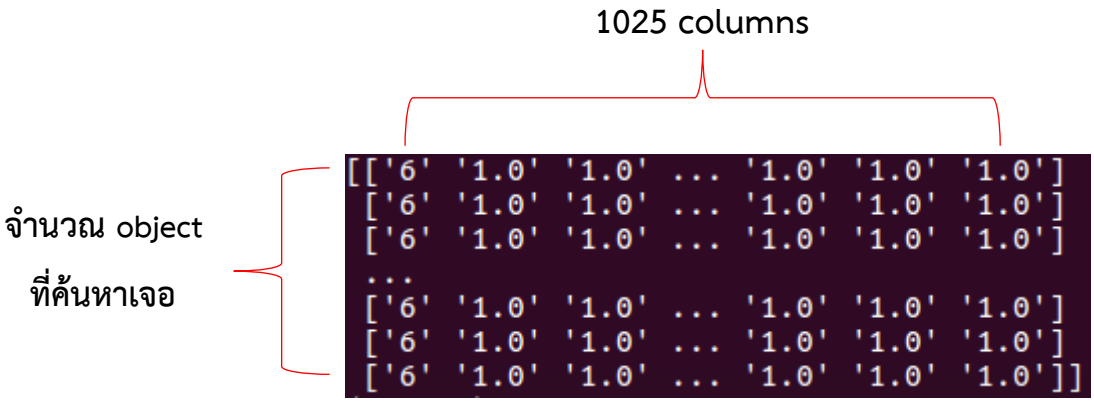


ต่อไปเราก็เอาค่าของ pixel นี้ไปต่อหลัง(hstack)ค่าที่แสดงถึง label ของ object นั้น ที่นี้ตัวแปรที่เป็น label คือ digit_y ซึ่งเราก็ได้ Feature Vector ของ object แล้ว



ค่า digit_y

ต่อไปเราก็เอาค่า Feature Vector ของแต่ละ object มาต่อกันเป็นแถว(vstack) โดยใส่ในตัวแปรเดียว เราก็ได้ Feature Vector ของแต่ละประเภทตัวอักษรแล้ว



Feature Vector ของแต่ละประเภทตัวอักษร

โคดที่ใช้สำหรับการสร้างและเก็บค่า Feature Vector ของแต่ละ object มีดังนี้

```
56      #create feature vector
57      digit_x = np.zeros((1,(img_row*img_col)), dtype='float64')
58      digit_fv = np.zeros((1,(img_row*img_col)+1), dtype='float64')
59
60      for i in(range(1,nb_labels+1)):
61          tmp = label_objects == i
62          r, = np.where(tmp.sum(axis=1) > 1)
63          c, = np.where(tmp.sum(axis=0) > 1)
64
65          # crop
66          tmp_img = binary_image[(r.min()):(r.max()), (c.min()):(c.max())]
67
68          digit_x = transform.resize(tmp_img, (img_row,img_col), mode='reflect')
69          digit_x = digit_x.reshape((1,(img_row*img_col)))
70          tmp_digit_data = np.hstack((digit_y, digit_x[0,:]))
71          digit_fv = np.vstack((digit_fv, tmp_digit_data))
72
73      # remove first row
74      digit_fv = np.delete(digit_fv, 0, 0)
75      print digit_fv
76
77      return digit_fv
```

function.py

สำหรับการสร้าง Feature Vector ของประเภทตัวอักษรอื่นๆ เราก็ทำในทำนองนี้เดียวกัน แล้วก็จะทำการเก็บในตัวแปรเดียว ที่นี้ผมตั้งชื่อตัวแปรว่า feature

-การนำ Feature Vector ทั้งหมดมาสร้างเป็น model

สร้างตัวแปร list ใหม่สองอันชื่อ x กับ y โดย y จะเก็บค่าคลาสหรือ labelทั้งหมดและนอกจากนี้จะเก็บไว้ในตัวแปร x ดังรูปด้านล่าง

```
23 x = feature[:,1:]
24 y = feature[:,0]
```

train.py

ทำการสร้าง model ในรูปแบบ svm ที่มีชื่อว่า ‘Modeltrain-Binaryimage.pkl’ ดังโคดด้านล่าง

```
26 clf_svm=svm.SVC()
27 clf_svm.fit(x,y)
28 joblib.dump(clf_svm,'Modeltrain-Binaryimage.pkl')
```

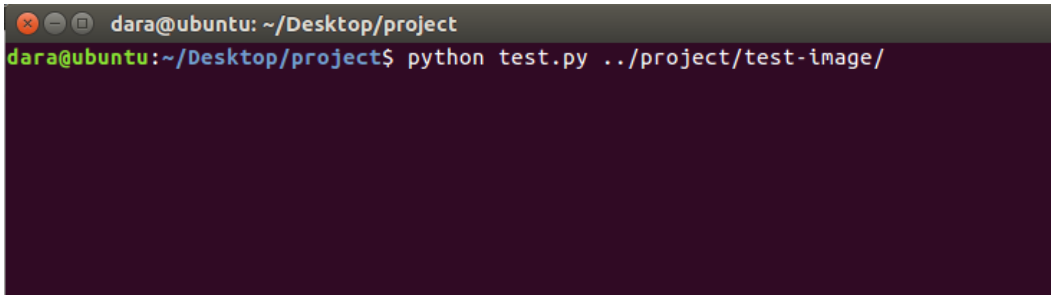
train.py

หลังจากนี้ไฟล์ Modeltrain-Binaryimage ก็ถูกสร้างขึ้นมา

การทำงานของโปรแกรม และ Prediction

ในไดเรกทอรีของโปรแกรมจะมีโฟลเดอร์หนึ่งชื่อว่า test-image ซึ่งเป็นโฟลเดอร์ที่เราจะต้องใส่รูปของรถที่มีป้ายทะเบียนที่เราอยากจะทำการ predict โดยสามารถใส่ได้หลายรูป

หลังจากที่เรามีรูปรถ แล้วใส่เข้าไปในโฟล์เดอร์ test-image แล้ว สิ่งที่เราต้องทำคือเปิด Terminalขึ้นมา แล้วต้องมั่นใจว่าใน Terminal เราได้เข้าไปที่ไดเรกทอรีชื่อ project แล้ว ต่อไปเราต้องทำการ run ไฟล์ python ดังรูปด้านล่างนี้



โปรแกรมก็จะทำการเรียกไฟล์ detector100.svm กับ mydataset100.xml เพื่อทำการครอบป้ายทะเบียน และทำการบันทึกไว้ในโฟลเดอร์ Save-LP-From-Car

ต่อไปจะทำการส่งป้ายทะเบียนเข้าในฟังก์ชัน imgTest2fv() ของไฟล์ function.py เพื่อทำการครอบเอาตัวอักษร แล้วสร้างเป็น Feature Vector เพื่อนำมาทำการ predict ซึ่งจะมีกระบวนการทำดังต่อไปนี้

-ทำรูปป้ายทะเบียนให้เป็นสีเทาก่อน

```
76 def imgTest2fv(fileName, img_row, img_col):
77
78     digit_y = fileName.split('-')[-1].split('.')[0]
79     digit = io.imread(fileName)
80     gray_image = color.rgb2gray(digit)
```

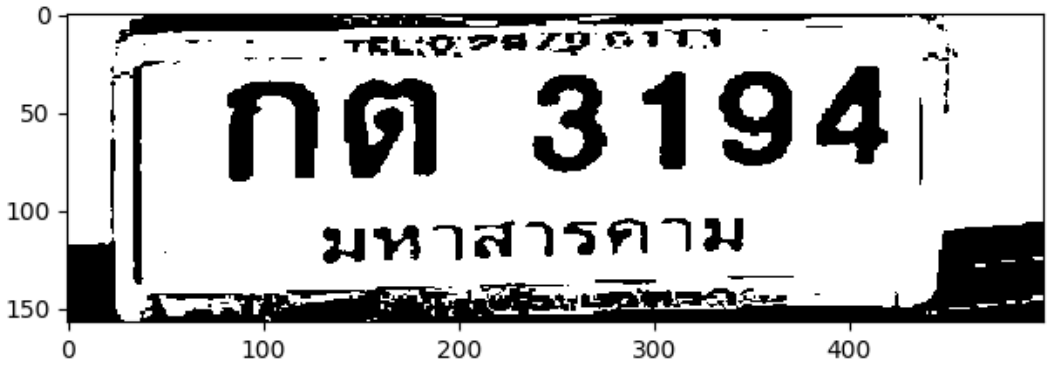
function.py

รูปสีเทา

-ทำรูปสีเทาให้เป็นรูปขาว-ดำโดยใช้ Adaptive threshold

```
81    # Using adaptive threshold
82    block_size = 305
83    adaptive_thresh = filters.threshold_local(gray_image, block_size, offset=0.1)
84    binary_image = gray_image > adaptive_thresh
```

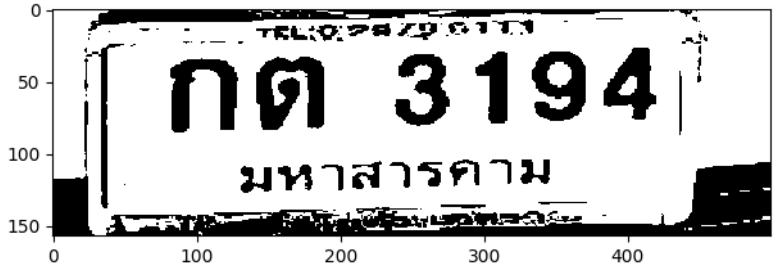
function.py



รูปขาว-ดำ

-ใช้ฟังก์ชัน median() เพื่อทำการ reduce noise โดยทำให้จำนวณ object เหลือน้อยลง

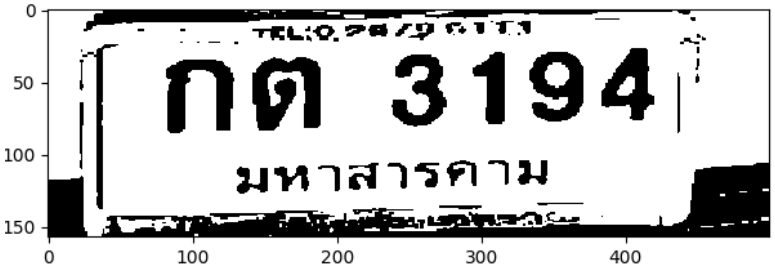
```
89    r = 1
90    binary_image = median(binary_image, disk(r))
```




```
('nb_label before using median is :', 103)
```

ไม่ได้ใช้ median()

Object ที่เจอทั้งหมดมี 103 objects

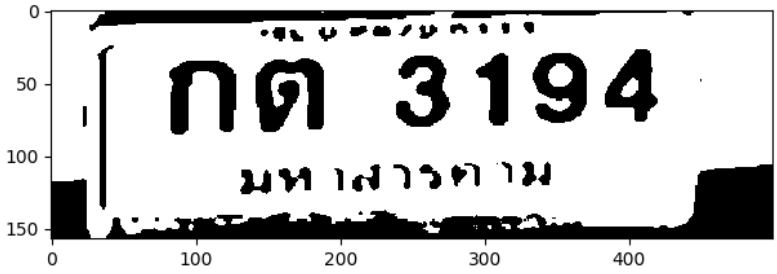



```
('nb_label after using median is :', 72)
```

ใช้ median()

Object ที่เจอทั้งหมดมี 72 objects

ถ้าเรายิ่งใช้ค่า r เยอะเท่าไร จำนวณ object ยิ่งเหลือน้อยลงแต่มันจะทำให้ตัวอักษรที่เราต้องการขาดได้ เหมือนรูปด้านล่าง




```
('nb_label after using median is :', 44)
```

ใช้ median() โดยกำหนดให้ r=3
Object ที่เจอก็จะเหลือ 44 objects
แต่คำว่า 'มหาสารคาม' จะขาด

-ทำการค้นหา object โดย label_objects เก็บค่า object ทั้งหมด, nb_labels เป็นจำนวณ object ทั้งหมด

```
52    label_objects, nb_labels = ndi.label(np.invert(binary_image))
```

-วนลูปโดยให้จำนวณรอบเท่ากับจำนวณ object ที่เจอเพื่อจะทำการตรวจเอาแต่ object ที่ต้องการเท่านั้น ซึ่งจะมีกระบวนการดังต่อไปนี้

หาค่าความสูงและความกว้างของobject และเก็บในตัวแปร r1_object และ c1_object

```
107    #collect all c1_min of the objects which are the character
108    for j in(range(1,nb_labels+1)):
109        tmp1 = label_objects == j
110        r1, = np.where(tmp1.sum(axis=1) > 1)
111        c1, = np.where(tmp1.sum(axis=0) > 1)
112        if (len(r1)==0 or len(c1)==0):
113            continue
114        r1_object=r1.max()-r1.min()
115        c1_object=c1.max()-c1.min()
```

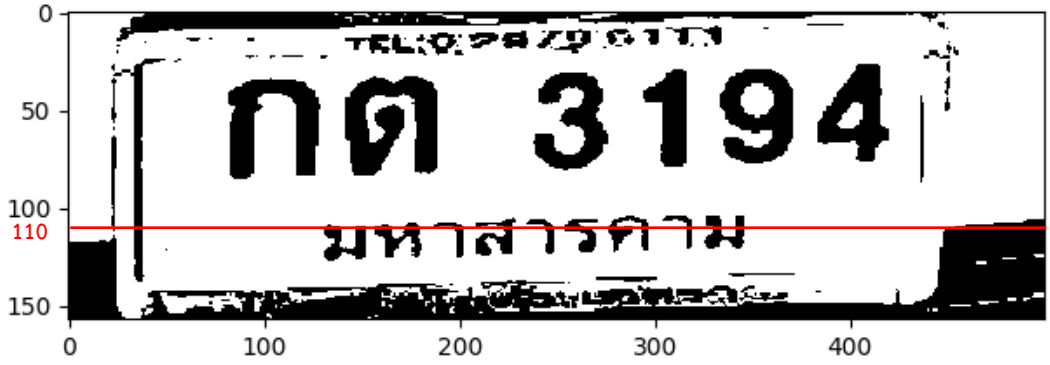
function.py

แล้วนำค่าในตัวแปรนี้ไปตรวจสอบเงื่อนไขว่าเป็นตัวอักษรตัวใหญ่ที่เป็นหมายเลขทะเบียนไหม

```
117        if (35<=r1_object<=80) and (18<=c1_object<=80) and (r1.max()<110):
118            print ('r1_object == ',r1_object)
119            print ('c1_object',c1_object)
120            c1_max_all_large = np.hstack((c1_max_all_large, c1.max()))
121            c1_min_all_large = np.hstack((c1_min_all_large, c1.min()))# collect all c1_min of large img
```

function.py

*โดยผ่านการทดลองtestรูปประมาณ 40รูป เห็นได้ว่าไม่มี object ที่เราต้องการตัวไหนมีค่า r1.max() มากกว่าตำแหน่งที่ pixel 110 เพราะฉะนั้นถ้าเจอ object ที่มีค่าความสูงและความกว้างอยู่ระหว่างเงื่อนไขของเรา แต่มันมีตำแหน่ง pixel ด้านล่างสุดของ object คือ r1.max() มีค่ามากกว่าหรือเท่ากับ 110 แสดงว่ามันไม่ใช่ object ที่เราต้องการ



r1.max() >=110 (ไม่เอา)

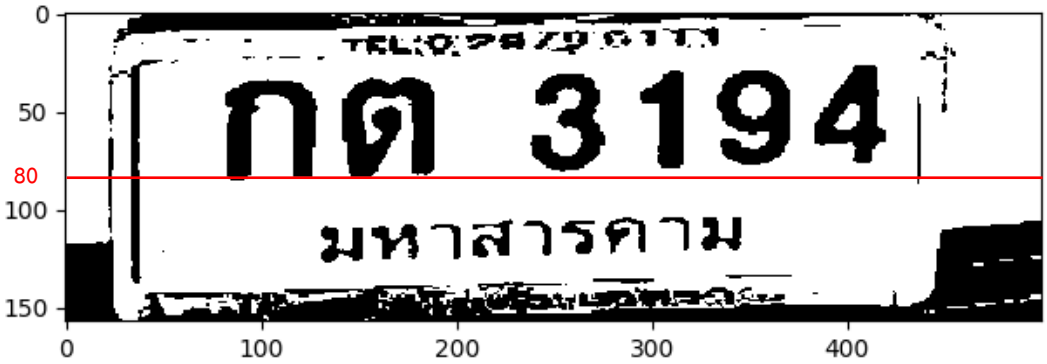
ที่นี้ถ้าเงื่อไขเป็นจริง ให้ทำการเก็บค่าตำแหน่ง pixel ซ้ายสุด(เพื่อใช้ในการเรียงตัวอักษรจากซ้ายไปขวา)และขวาสุด(เพื่อใช้เป็นเงื่อนไขในการครอบตัวอักษรที่เป็นชื่อจังหวัด)ของทุก object

แต่ถ้าไม่จริง ให้นำค่าความสูงและความกว้างของobject นั้นไปตรวจสอบเงื่อนไขว่าเป็นตัวอักษรตัวเล็กที่เป็นชื่อจังหวัดหรือไหม

```
122         elif ((15<=r1_object<=25) and (5<=c1_object<=35)) and (r1.min() > 80):
123             c1_min_all_small = np.hstack((c1_min_all_small, c1.min()))# collect all c1_min of small img
```

function.py

*โดยผ่านการทดลองtestรูปประมาณ 40รูป เห็นได้ว่าไม่มี object ที่เราต้องการตัวไหนมีค่า r1.min() น้อยกว่าตำแหน่งที่ pixel 80 เพราะฉะนั้นถ้าเจอ object ที่มีค่าความสูงและความกว้างอยู่ระหว่างเงื่อนไขของเรา แต่มันมีตำแหน่ง pixel ด้านบนสุดของ object คือ r1.min() มีค่าน้อยกว่าหรือเท่ากับ 80 แสดงว่ามันไม่ใช่ object ที่เราต้องการ



r1.min() <=80 (ไม่เอา)

ที่นี้ถ้าเงื่อไขเป็นจริง ให้ทำการเก็บค่าตำแหน่ง pixel ซ้ายสุด(เพื่อใช้ในการเรียงตัวอักษรจากซ้ายไปขวา) ของของทุก object

เมื่อวนลูปเสร็จเรียบร้อยแล้ว เราต้องใช้ฟังก์ชัน .sort() เพื่อทำให้ ตัวแปร c1_min_all_large , c1_max_all_large และ c1_min_all_small เรียงค่าจากน้อยไปมากเพื่อใช้ในการเรียงลำดับ object จากซ้ายไปขวา

```
125     c1_min_all_large.sort()
126     c1_max_all_large.sort()
127     if(len(c1_min_all_small)>1):
128         c1_min_all_small.sort()
```

function.py

* บรรทัดที่ 127 เช็คเพื่อป้องกัน error เพราะถ้าป้ายทีเรานำมาทำการ predict เป็นป้ายที่ไม่ค่อยชัดหรือไม่ค่อยดี แล้วทำให้detect object ที่เป็นชื่อของจังหวัดไม่เจอสักตัว ก็จะทำให้ตัวแปร c1_min_all_small เก็บค่าแค่ตัวเดียวคือ 0 ที่ติดมาตั้งแต่ตอนประกาศตัวแปรไว้ ดังนั้นถ้าเราเรียกใช้ฟังก์ชัน .sort() ให้กับตัวแปรนั้น จะทำให้เกิด error ขึ้น

```
('c1_min_all_large ', array([  0, 383, 294, 333, 238, 142,  82]))
('c1_max_all_large ', array([  0, 419, 315, 370, 275, 188, 128]))
('c1_min_all_small ', array([  0, 276, 305, 325, 209, 235, 254, 157, 187, 131]))
('c1_min_all_large after sort', array([  0,  82, 142, 238, 294, 333, 383]))
('c1_max_all_large after sort', array([  0, 128, 188, 275, 315, 370, 419]))
('c1_min_all_small after sort', array([  0, 131, 157, 187, 209, 235, 254, 276, 305, 325]))
```

ก่อน sort และหลัง sort

ลบ index ที่ 0 มีค่าเท่ากับ 0 ซึ่งติดมาตั้งแต่ตองประกาศตัวแปร

```
135     c1_min_all_large = np.delete(c1_min_all_large,0)# Delete index = 0
136     c1_max_all_large = np.delete(c1_max_all_large,0)# Delete index = 0
137     c1_min_all_small = np.delete(c1_min_all_small,0)# Delete index = 0
```

Function.py

```
('c1_min_all_large after sort', array([ 82, 142, 238, 294, 333, 383]))
('c1_max_all_large after sort', array([128, 188, 275, 315, 370, 419]))
('c1_min_all_small after sort', array([131, 157, 187, 209, 235, 254, 276, 305, 325]))
```

หลังจากลบค่า index ที่ 0

เพื่อให้การครอบ object ที่เป็นชื่อจังหวัดมีความแม่นยำมากขึ้น เราต้องครอบเอาแต่ object ที่มี pixel ซ้ายสุดอยู่ระหว่างหมายเลขทะเบียนป้ายเท่านั้น(ระหว่างบรรทัดสีแดง) ถ้าตำแหน่ง pixel ซ้ายสุดไม่ได้อยู่ในครอบบรรทัดสีแดง ต่อให้มันมีขนาดเหมือนกันกับ object ที่เป็นชื่อจังหวัด ก็เราไม่ได้สนใจทำการครอบออกมา เหมือนรูปด้านล่างนี้

โคดที่ใช้เพื่อตรวจสอบเอาแต่ object ที่อยู่ในครอบสีแดงเท่านั้น

```
136     # check for small characters which are between large characters from left to right
137     a=0
138     while a < len(c1_min_all_small):
139         if (c1_min_all_small[a] <= c1_min_all_large.min()) or (c1_min_all_small[a] >= c1_max_all_large.max()):
140             c1_min_all_small = np.delete(c1_min_all_small,a)
141         else:
142             a = a + 1
```

Function.py

-ทำการวนลูปใหม่โดยจำนวณรอบเท่ากับจำนวณ object ที่ detect เจอในป้ายทะเบียนหลังผ่านกระบวนการ reduce noise เสร็จเรียบร้อย

ในแต่ละรอบโปรแกรมจะทำการตรวจสอบว่าความสูงและความกว้างของ object อยู่ระหว่างเงื่อนไขของตัวอักษรตัวใหญ่หรือไม่ ถ้าอยู่ก็ให้ตรวจต่อไปว่า ตำแหน่ง pixelด้านซ้ายสุดมีค่า

เท่ากับตัวแปร c1_min_all_large โดยเริ่มจาก index ที่ 0 หรือไม่ ถ้าเท่ากัน ก็ให้ครอบ object นั้นมาเพื่อสร้างเป็น Feature Vector ซึ่งจะวนลูปจนมีค่าตำแหน่ง pixel ของ object มีค่าเท่ากับ c1_min_all_large ของ indexสุดท้าย

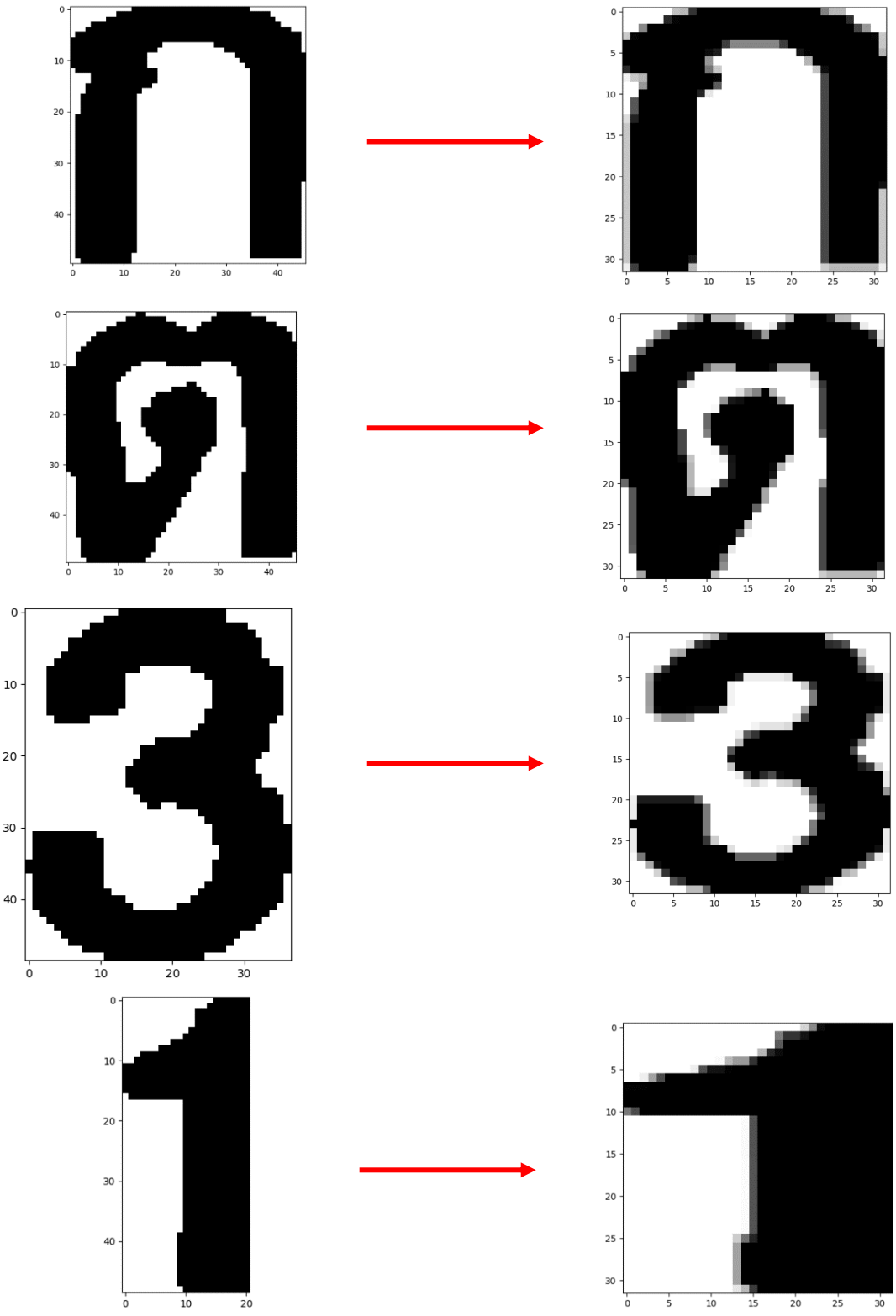
สำหรับตัวอักษรที่เป็นชื่อจังหวัดก็ทำในลักษณะเดียวกันจนกว่ามีค่าตำแหน่ง pixel ของ object มีค่าเท่ากับ c1_min_all_small ของ indexสุดท้าย

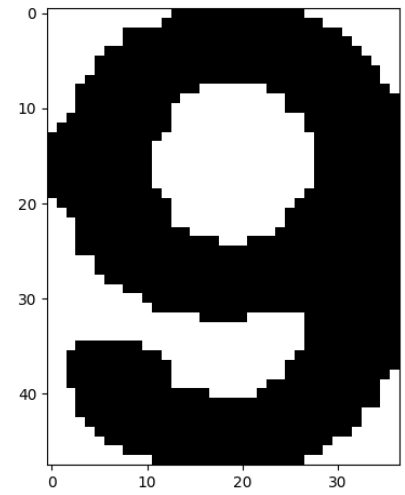
```
    Index_c1_min_all_large=0
    Index_c1_min_all_small=0
    n=1
    while n < nb_labels+1:
        tmp = label_objects == n
        r, = np.where(tmp.sum(axis=1) > 1)
        c, = np.where(tmp.sum(axis=0) > 1)
        n=n+1
        if (len(r)==0 or len(c)==0):
            continue
        r_object=r.max()-r.min()
        c_object=c.max()-c.min()

        if (Index_c1_min_all_large < len(c1_min_all_large)):
            if (35<=r_object<=80) and (15<=c_object<=80) and (r.max()<110):
                if (c1_min_all_large[Index_c1_min_all_large] == c.min()):
                    #for Show white image with characters
                    print ('r_object == ',r_object)
                    print ('c_object',c_object)
                    for a in(range(0,157)):
                        for b in(range(0,500)):
                            if(tmp[a,b]==True):
                                white_img[a,b]=not white_img[a,b]
                    # crop character
                    tmp_img = binary_image[(r.min()):(r.max()), (c.min()):(c.max())]
                    digit_x = transform.resize(tmp_img, (img_row,img_col), mode='reflect')
                    digit_x = digit_x.reshape((1,(img_row*img_col)))

                    tmp_digit_data = np.hstack((digit_y, digit_x[0,:]))
                    digit_fv_large = np.vstack((digit_fv_large, tmp_digit_data))

                    Index_c1_min_all_large = Index_c1_min_all_large + 1
                    n=1

        elif (Index_c1_min_all_small < len(c1_min_all_small)):
            if (15<=r_object<=30) and (5<=c_object<=35):
                if (c1_min_all_small[Index_c1_min_all_small] == c.min()):
                    #for Show white image with characters
                    print ('r_object ==== ',r_object)
                    print ('c_object',c_object)
                    for a in(range(0,157)):
                        for b in(range(0,500)):
                            if(tmp[a,b]==True):
                                white_img[a,b]=not white_img[a,b]
                    # crop character
                    tmp_img = binary_image[(r.min()):(r.max()), (c.min()):(c.max())]
                    digit_x = transform.resize(tmp_img, (img_row,img_col), mode='reflect')
                    digit_x = digit_x.reshape((1,(img_row*img_col)))

                    tmp_digit_data = np.hstack((digit_y, digit_x[0,:]))
                    digit_fv_small = np.vstack((digit_fv_small, tmp_digit_data))

                    Index_c1_min_all_small = Index_c1_min_all_small + 1
                    n=1
        else:
            n=nb_labels+1

    #io.imshow(white_img)
    #io.show()
    # remove first row
    digit_fv_large = np.delete(digit_fv_large, 0, 0)
    digit_fv_small = np.delete(digit_fv_small, 0, 0)

    return digit_fv_large, digit_fv_small, len(c1_min_all_large), len(c1_min_all_small)
```

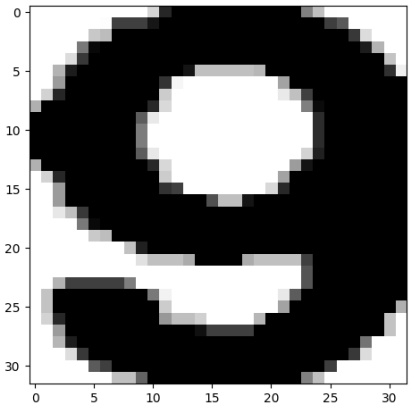
Function.py

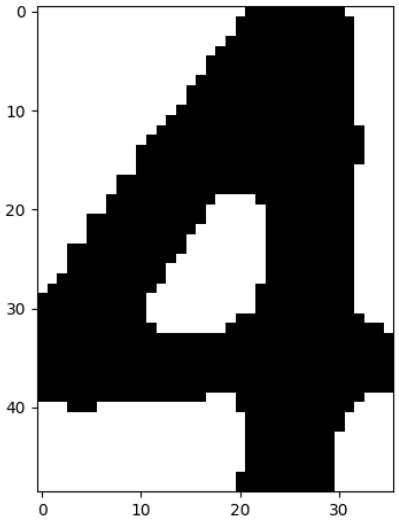
การสร้าง Feature Vector ของตัวอักษรในป้ายทะเบียนจะใช้กระบวนการเดียวกันกับการสร้าง Feature Vector ของ Dataset โดยเราต้องนำ object ที่ครอบมามาทำการ resize ก่อนโดยให้มีขนาดเป็น 32*32 pixel

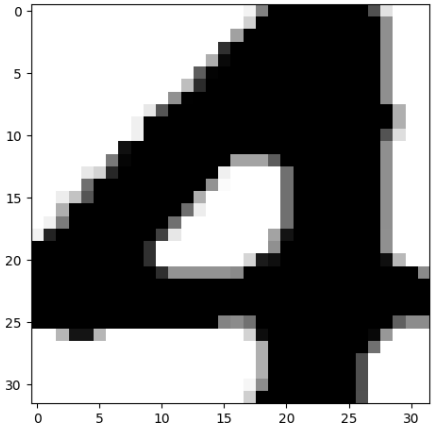
รูปที่ครอบมา **รูปที่ปรับขนาดเป็น 32*32px**

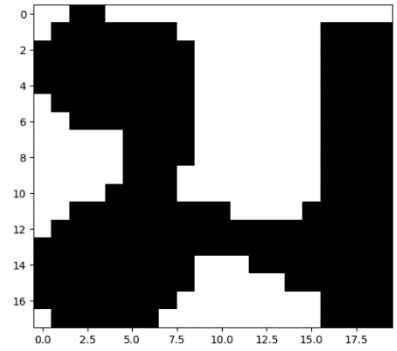


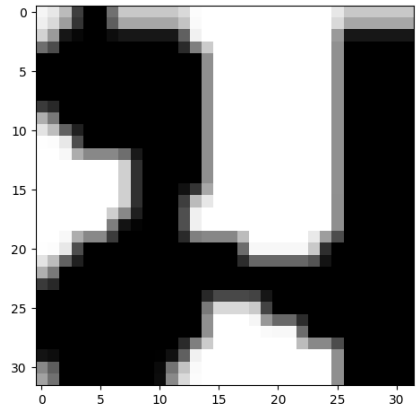


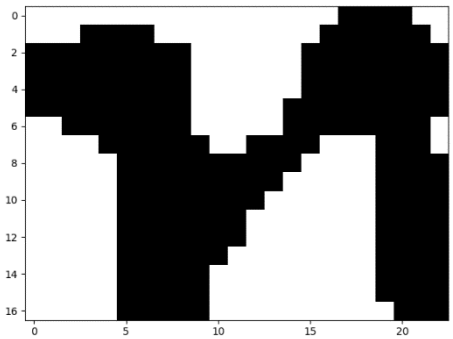


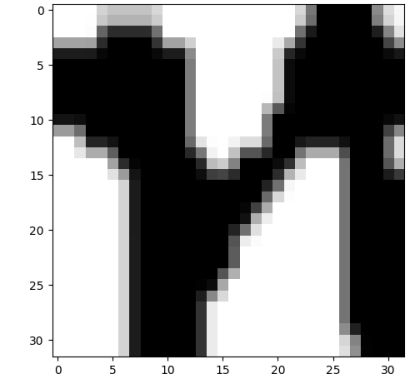


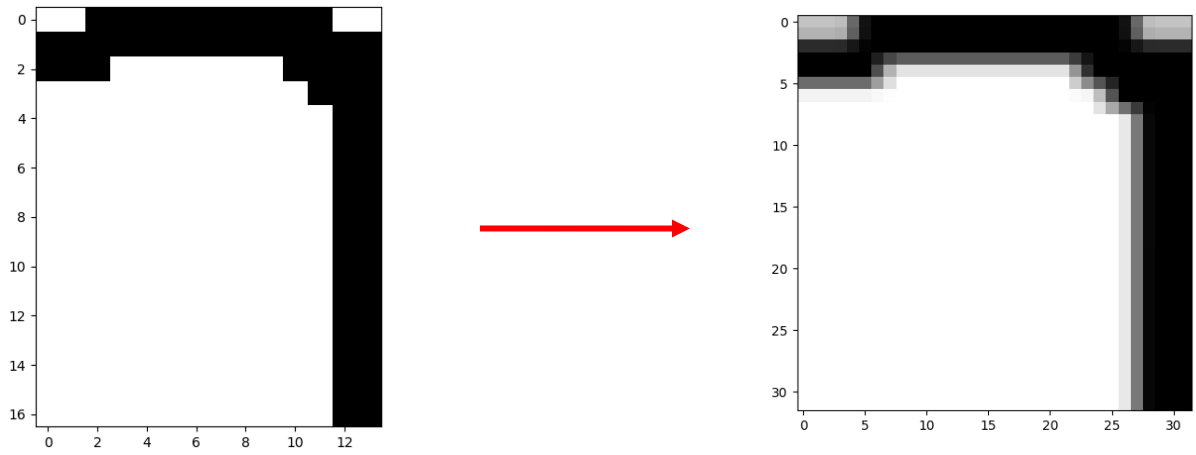


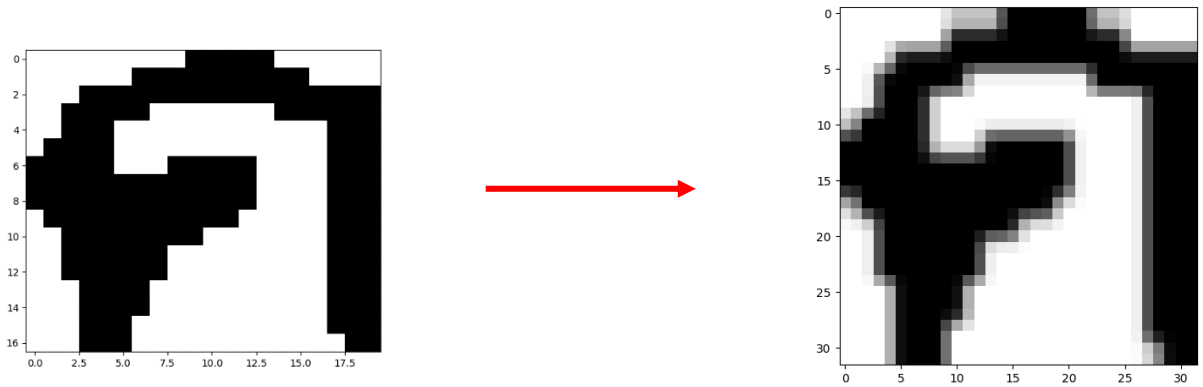


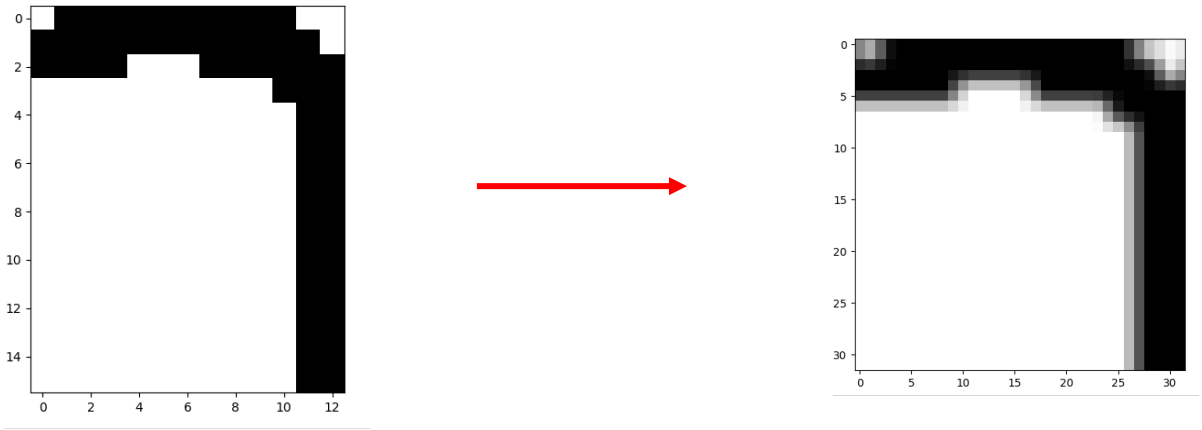


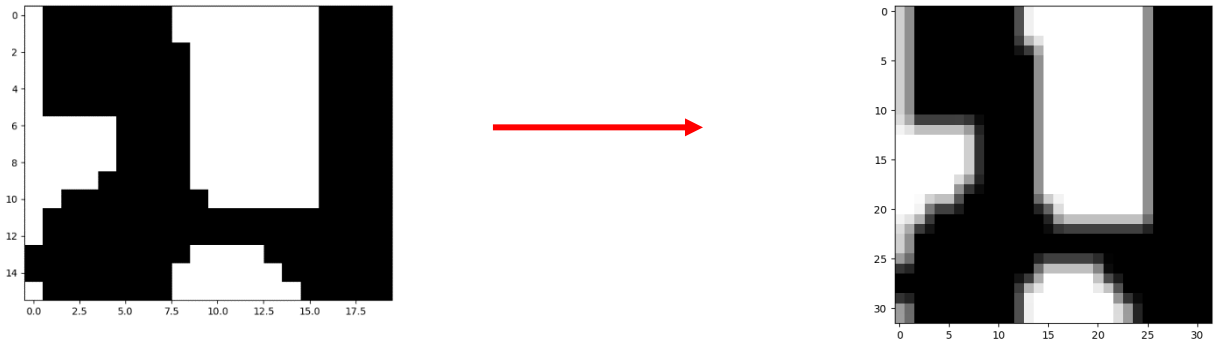












ค่าของpixel ของแต่ละ object ที่ครอบมาหลังจากทำการ resize เสร็จเรียบร้อยแล้วจะมีค่าเป็นประมาณนี้

32 columns

32 rows

```
[[1.         1.         1.         ... 1.         1.         1.        ]
 [1.         1.         1.         ... 0.92089844 1.         1.        ]
 [1.         1.         1.         ... 0.         0.20410156 0.91113281]
 ...
 [1.         1.         0.         ... 0.         0.         0.        ]
 [1.         1.         0.         ... 0.         0.         0.12207031]
 [1.         1.         0.65136719 ... 0.71875    0.71875    0.93847656]]
```

เพื่อที่จะนำค่า pixel มาสร้างเป็นค่า Feature Vector เราต้องทำการ reshape ค่า pixel โดยนำค่า pixel ของแต่ละแถวมาต่อกันให้เป็นแถวเดียวซึ่งจะทำให้ขนาดของคอลัมน์มีทั้งหมดจำนวน 32x32=1024 ดังรูปด้านล่าง

1024 columns

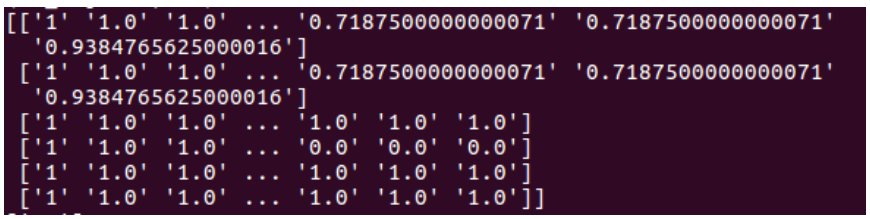
ต่อไปเราก็เอาค่าของ pixel นี้ไปต่อหลัง(hstack)ค่าที่แสดงถึง label ของ object นั้น ที่นี้ตัวแปรที่เป็น label คือ digit_y ซึ่งเราก็ได้ Feature Vector ของ object แล้ว

ค่า digit_y

ต่อไปเราก็เอาค่า Feature Vector ของแต่ละ object มาต่อกันเป็นแถว(vstack) โดยใส่ในตัวแปรเดียว เราก็ได้ Feature Vector ของแต่ละประเภทตัวอักษรแล้ว

1025 columns

จำนวน object ที่ค้นหาเจอ(ตัวใหญ่)



Feature Vector ของตัวอักษรตัวใหญ่

สำหรับการสร้าง Feature Vector ของตัวอักษรที่เป็นชื่อจังหวัดเราก็ทำในทำนองนี้เดียวกัน แล้วก็จะทำการเก็บในตัวแปรเดียว

หลังจากนั้นเราก็ได้ค่า Feature Vector ของหมายเลขป้ายทะเบียน และข้อความที่เป็นชื่อของจังหวัดที่เก็บอยู่ในตัวแปร digit_fv_large, digit_fv_small

-ขั้นตอนการ Predict

สร้างตัวแปร list ใหม่สองอันชื่อ X_large กับ X_small โดยX_large จะเก็บค่าของ digit_fv_large และ X_small จะเก็บค่าของ digit_fv_small ดังโคดด้านล่าง

```
95      X_large = digit_fv_large[:,1:]
96      X_small = digit_fv_small[:,1:]
```

test.py

-การ predict หมายเลขป้ายทะเบียน(ตัวอักษรใหญ่)

ทำการวนลูปจำนวณรอบเท่ากับจำนวณ object ของตัวอักษรใหญ่ ในแต่ละรอบโปรแกรมจะทำการดึงค่าจากตัวแปร X_large ในรูปแบบเป็นแถวเพื่อทำการ predict(บรรทัด 109 ในรูปภาพ) หลังจากนั้นก็ได้ค่า pre_out ซึ่งเป็นค่า labelของDataset เราก็จะใช้ค่า pre_outเพื่อทำการระบุถึง index ของตัวแปร Text เพื่อให้ predict ออกมาเป็นตัวเลขหรือตัวอักษรไทยตามรูปในป้ายทะเบียน

เมื่อทำการวนลูปเสร็จ เราก็จะได้ค่าของการ predict ของหมายเลขป้ายทะเบียน โดยเก็บอยู่ในตัวแปร tmp_title_large

```
103     text=
    [0,1,2,3,4,5,6,7,8,9,'ก','ข','ฃ','ค','ฅ','ฆ','ง','จ','ฉ','ช','ซ','ฌ','ญ','ฎ','ฏ','ฐ','ฑ','ฒ','ณ',
104
105
106     for j in(range(0, num_large)):
107         if cnt_large==len(X_large): #len(X) is the number of character(large) in an test-image
108             break
109         else:
110             pre_out = clf_svm.predict([X_large[cnt_large]])
111             if (int(pre_out) == 54):# Yor Ying
112                 pre_out = 22
113             elif (int(pre_out) == 55):# Thor Than
114                 pre_out = 25
115             #tmp_title_large = tmp_title_large + str(int(pre_out)) + ' '
116             tmp_title_large = tmp_title_large + str(text[int(pre_out)]) + ' '
117
118             cnt_large = cnt_large+1
```

test.py

-การ predict ตัวอักษรที่เป็นชื่อจังหวัด(ตัวอักษรเล็ก)

กระบวนการ predict ของตัวอักษรเล็กมีลักษณะเหมือนกันกับการ predict ตัวอักษรตัวใหญ่ ซึ่งหลังจาก predictเสร็จเราก็ได้ค่าของการ predict ของตัวอักษรช่อของจังหวัดโดยเก็บอยู่ในตัวแปร tmp_title_small

หลังจากนั้นเราจะเอาค่าของทั้งสองตัวแปร tmp_title_large และ tmp_title_small ไปใส่ขางบนภาพป้ายทะเบียนของเรา แล้วก้ทำการบันทึกรูปภาพในโฟลเดอร์ชื่อ Save-Image-Predict

กต 3194
มหาสารคาม

ผลลัพธ์สุดท้าย

-การ Test

ใช้รูปรถที่มีป้ายทั้งหมด 20 รูป

ครอบป้ายออกมาถูก 20 ป้าย

หมายเลขของป้ายทะเบียน(ตัวใหญ่) 120 ตัว predict ผิด 2 ตัว

ตัวอักษรของจังหวัด(ตัวเล็ก) 154 ตัว predict ผิด 25 ตัว

ป้ายที่detect เจอทุกตัวอักษร และ predict ถูกทั้งหมดมี 7 ป้าย

<u>ตัวอย่างรูปที่detect เจอทุกตัวอักษร และ predict ถูกทั้งหมด</u>

กล3216
ขอมแกน

บม1482
มหาสารคาม

กท8453
รอยเอด

บต8912
มหาสารคาม

ตัวอย่างรูปที่detect ไม่เจอทุกตัวอักษร และ predict มีผิดพลาดบาง

ก ว 2 5 1 7
อ ด า บ

บ ว 2 3 7 1
ถ ย เ อ ด

บ บ 2 1 9 2
ม ห ส 1 ษ า 1 1

2 ก ต 4 2 7 1
ก ร ง เ ท พ ม ห า น ค ร