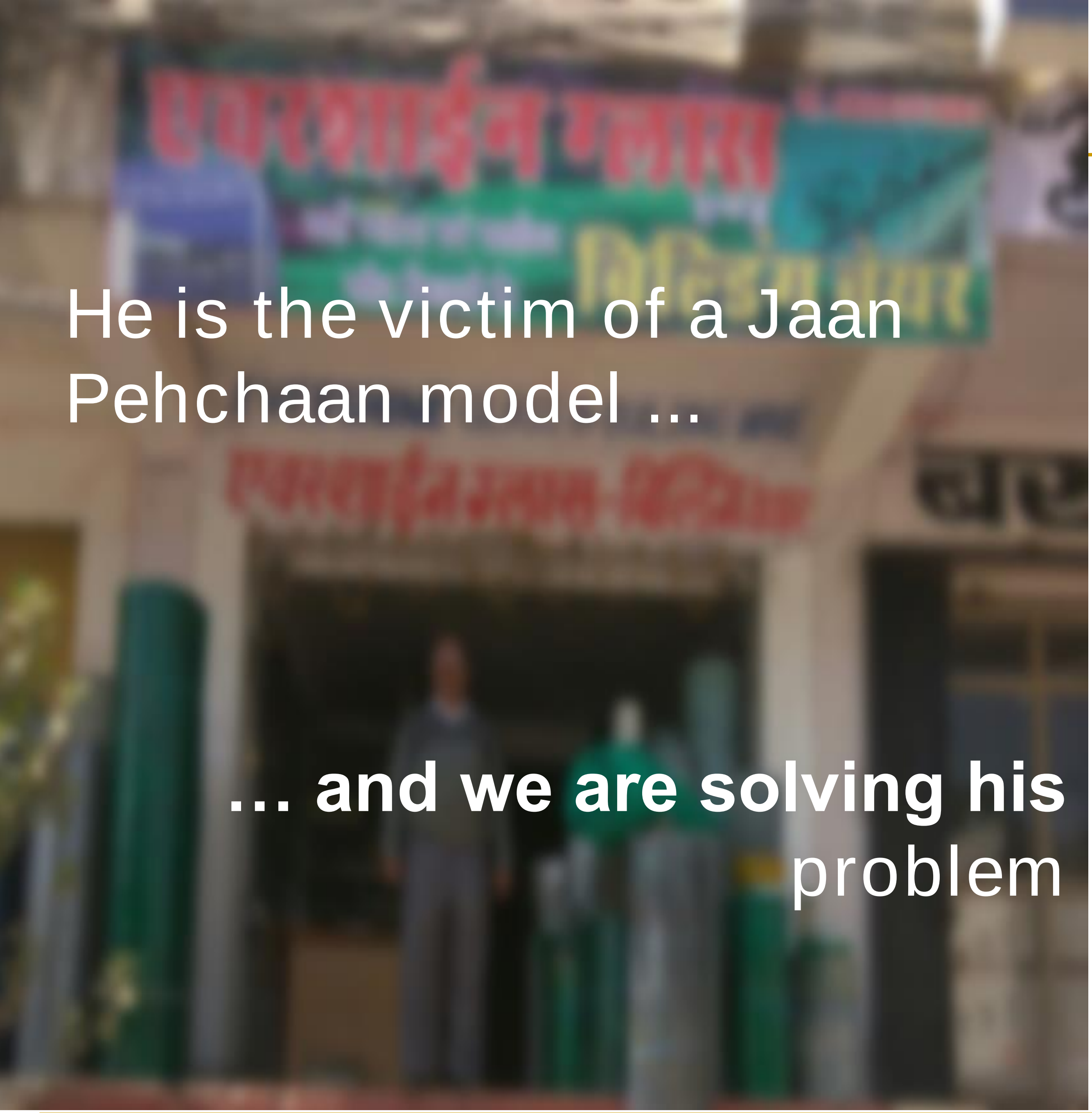
Image Processing Applications in FinTech

Bhavesh Bhatt

Agenda for the Talk

- What problems are we trying to solve?
- Introduction to OpenCV with Python
- Image processing applications in Flexiloans
- Separating foreground from background using Otsu’s Binarization

PyData MUMBAI



He is the victim of a Jaan Pehchaan model ...

… and we are solving his problem

- Runs a small business
- Based in Udaipur
- No CIBIL record, but occasionally needs financing
- **‘Phones a friend’** when he needs money because it is quick (not cheap)

Realizing Our Dream driven by Data-driven decision making

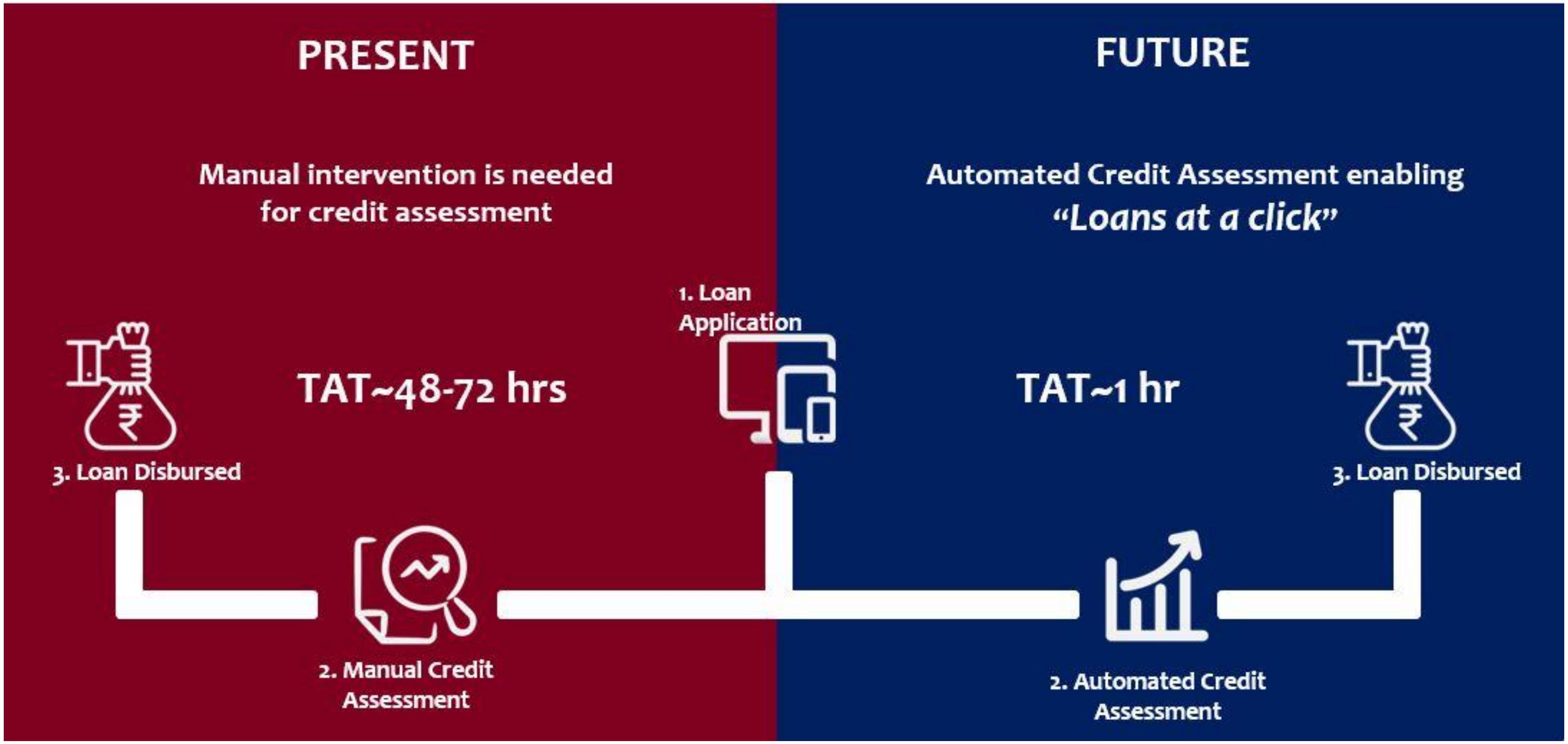

PRESENT
Manual intervention is needed for credit assessment
TAT~48-72 hrs
1. Loan Application
2. Manual Credit Assessment
3. Loan Disbursed
FUTURE
Automated Credit Assessment enabling *"Loans at a click"*
TAT~1 hr
2. Automated Credit Assessment
3. Loan Disbursed

Image Basics I

- How do we represent an image?
- Images are stored in forms of a 2-D matrix. For a colour image (RGB) we store 3 similar matrices stacked over each other.

- Why RGB - R + G + B combines to give White

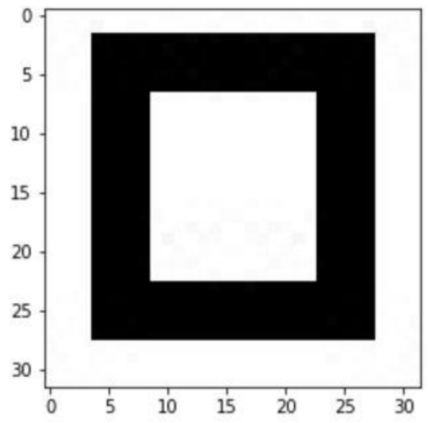
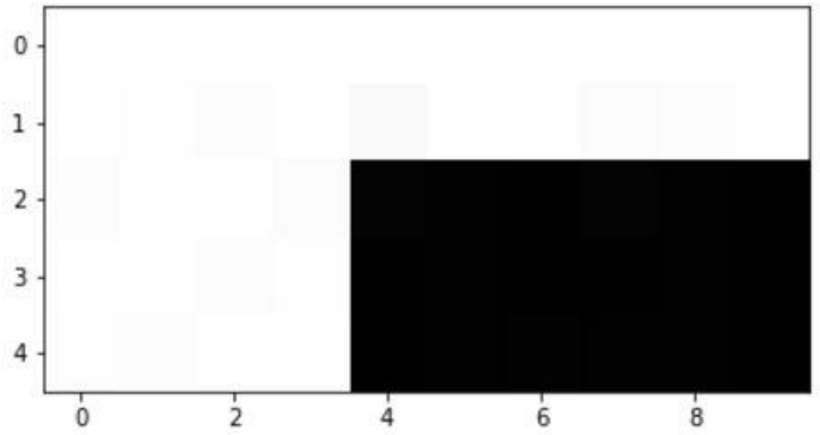


Image Basics II

```
plt.imshow(img[0:5,0:10],cmap='gray')
plt.show()
```



```
print img[0:5,0:10]
```

```
[[255 255 255 255 255 255 255 255 255 255]
 [255 254 253 255 249 255 255 252 253 255]
 [253 255 255 252   4   1   0   3   1   0]
 [255 255 253 254   0   1   0   0   1   0]
 [254 253 255 255   0   1   2   1   1   0]]
```

- Considering the top left portion of the image.
- White pixels represented by values close to 255 and black pixels represented by values close to 0.

Image Basics III

- Colour images can be represented by three matrices.
- Each matrix specifies the amount of Red, Green and Blue that makes up the image

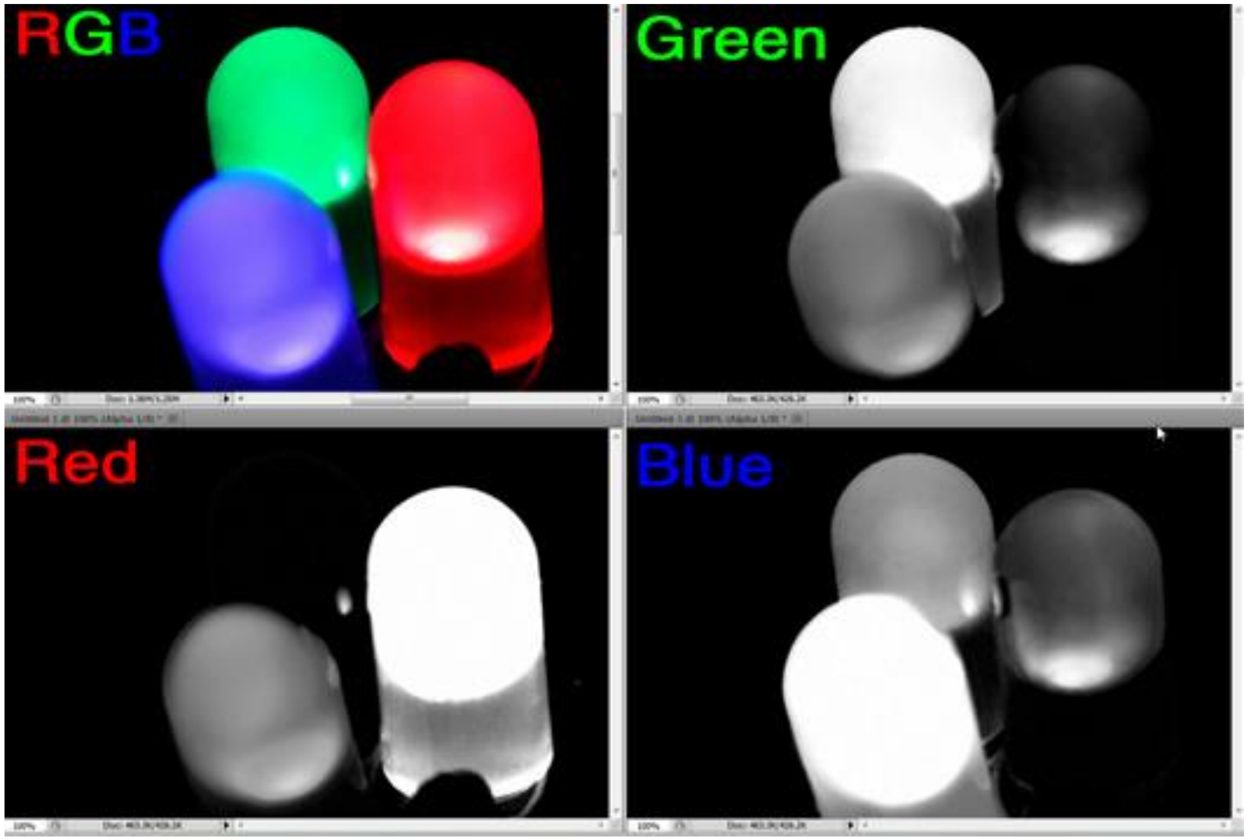
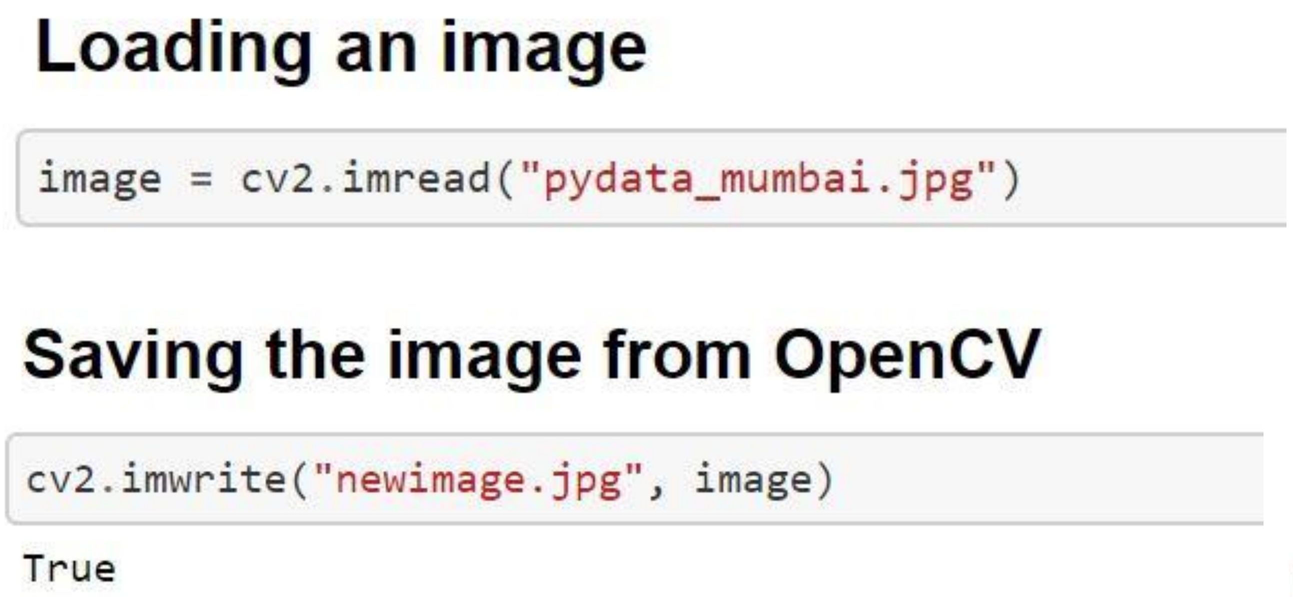


Image Processing using OpenCV I

- OpenCV is a library of programming functions mainly aimed at real-time computer vision.
- Lets learn how we can use OpenCV for Image processing.



Loading an image

```
image = cv2.imread("pydata_mumbai.jpg")
```

Saving the image from OpenCV

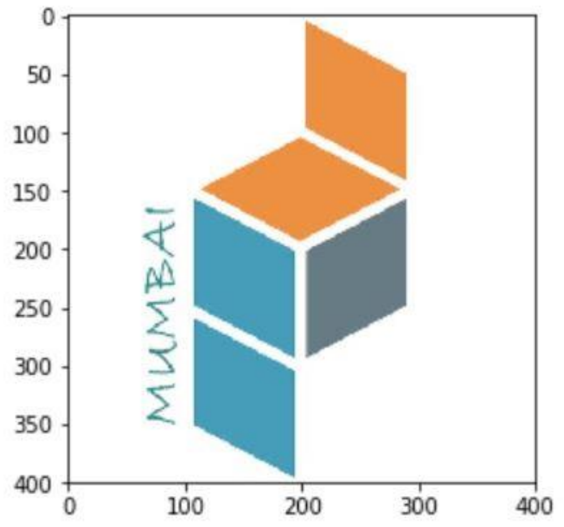
```
cv2.imwrite("newimage.jpg", image)
```

```
True
```

Image Processing using OpenCV II

Displaying the inputted image

```
show_RGBimage(image)
```



```
def show_RGBimage(image):
    '''
    OpenCV represents RGB images as
    multi-dimensional NumPy array
    but in reverse order.
    This means that images are actually
    represented in BGR order rather than RGB.
    '''
    plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    plt.axis("on")
    return plt.show()
```

```
print "width: %d pixels" % (image.shape[1])
print "height: %d pixels" % (image.shape[0])
print "channels: %d" % (image.shape[2])
```

```
width: 400 pixels
height: 400 pixels
channels: 3
```

```
type(image)
```

```
numpy.ndarray
```

Image Processing using OpenCV III

Accessing & Manipulating pixels

```
(b, g, r) = image[0, 0]
print "Pixel at (0, 0) - Red: %d, Green: %d, Blue: %d" % (r, g, b)
```

```
Pixel at (0, 0) - Red: 255, Green: 255, Blue: 255
```

Creating a black image

```
blank_image = np.zeros((400,400,3))
type(blank_image[0,0,0])
```

```
numpy.float64
```

```
black_image = np.zeros((400,400,3), np.uint8)
show_RGBimage(black_image)
```

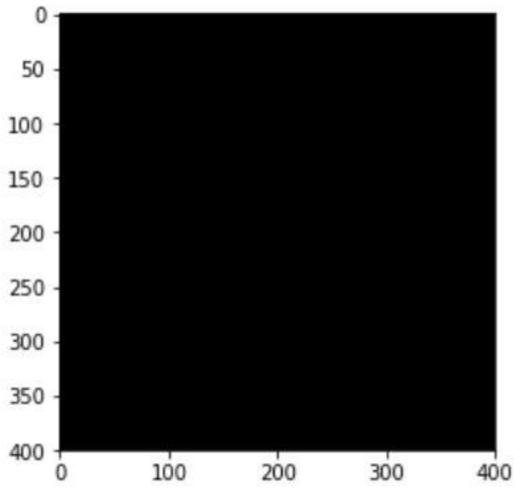
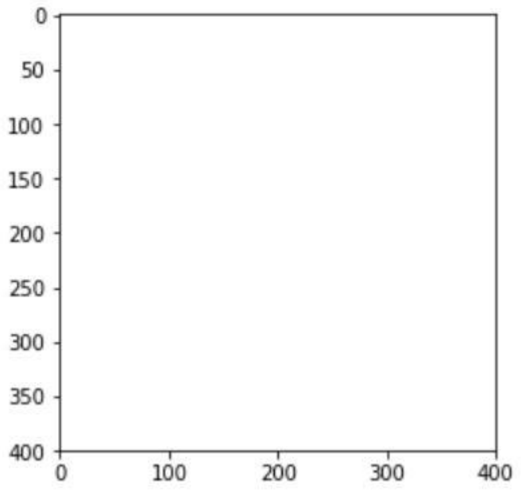


Image Processing using OpenCV IV

Creating a white image

```
white_image = 255*np.ones((400,400,3), np.uint8)
show_RGBimage(white_image)
```



```
image_ref1 = image
```

```
image_ref1[0:100, 0:100] = (0, 0, 0)
show_RGBimage(image_ref1)
```

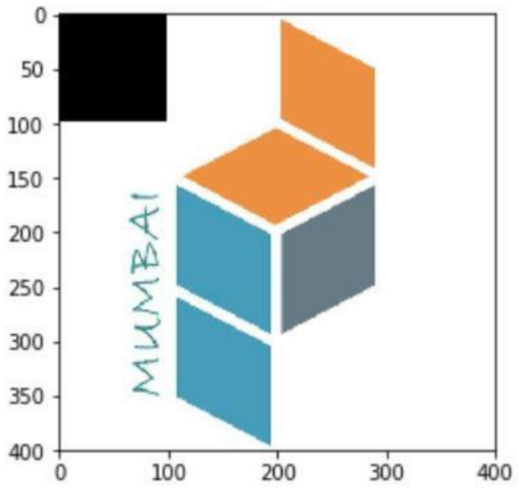
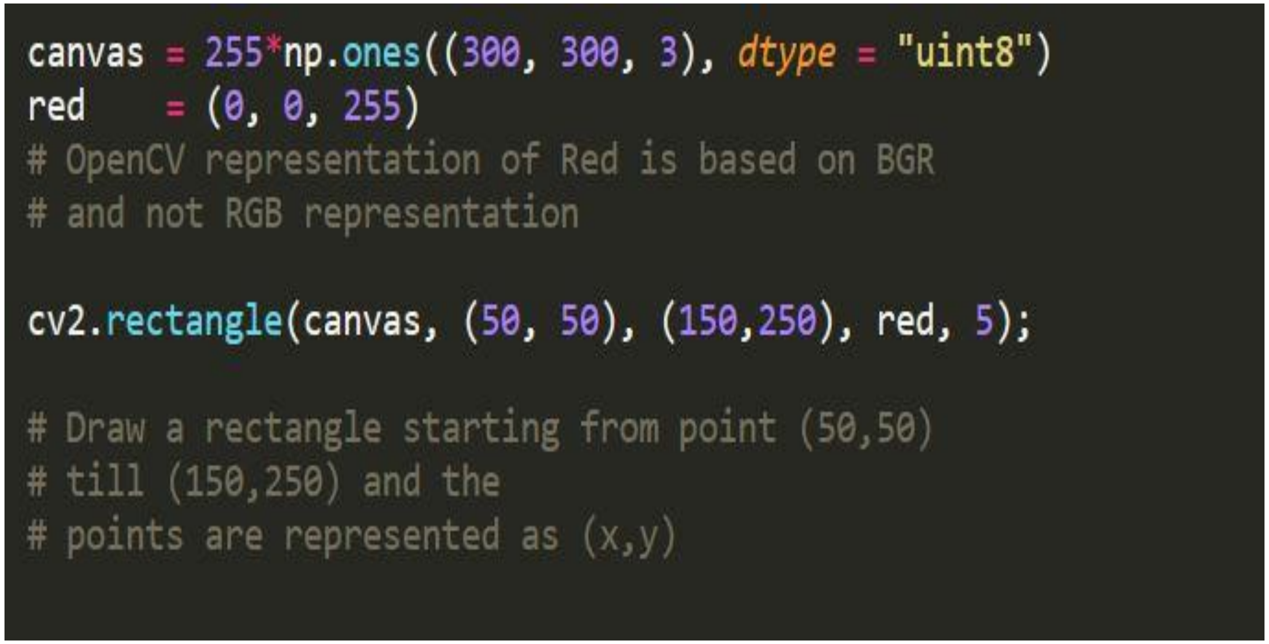


Image Processing using OpenCV V

- **Drawing shapes using OpenCV**



```
canvas = 255*np.ones((300, 300, 3), dtype = "uint8")
red    = (0, 0, 255)
# OpenCV representation of Red is based on BGR
# and not RGB representation

cv2.rectangle(canvas, (50, 50), (150,250), red, 5);

# Draw a rectangle starting from point (50,50)
# till (150,250) and the
# points are represented as (x,y)
```

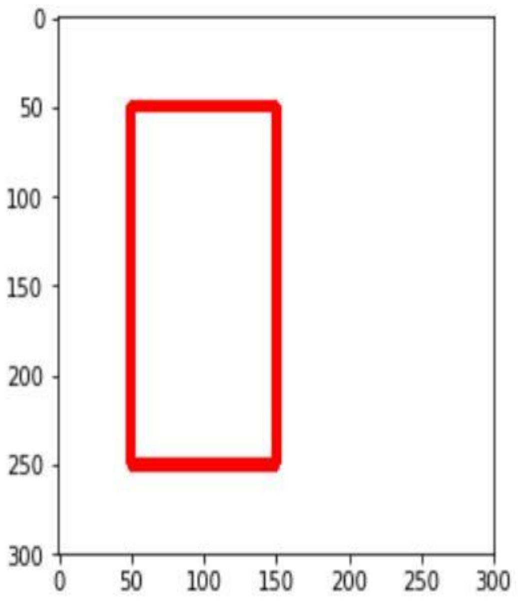


Image Processing using OpenCV VI

- **Histogram -** Displays the frequency distribution of the intensity values of an image

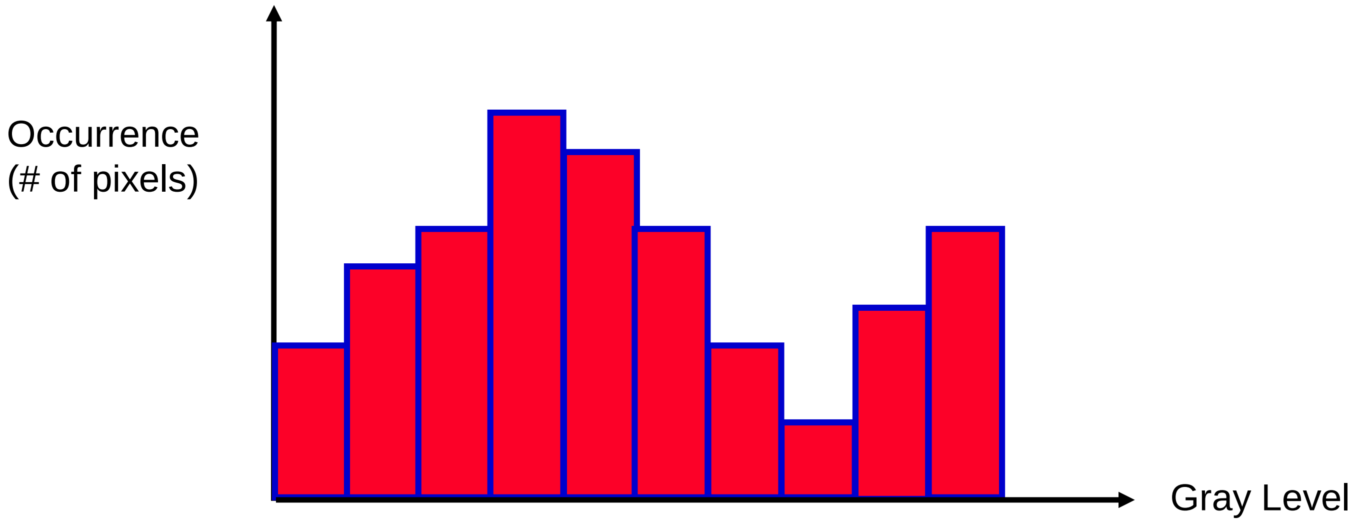


Image Processing using OpenCV VII

- **Histogram of Black Image**

```
image_black_gray = cv2.cvtColor(black_image, cv2.COLOR_BGR2GRAY)
plot_histogram(image_black_gray)
```

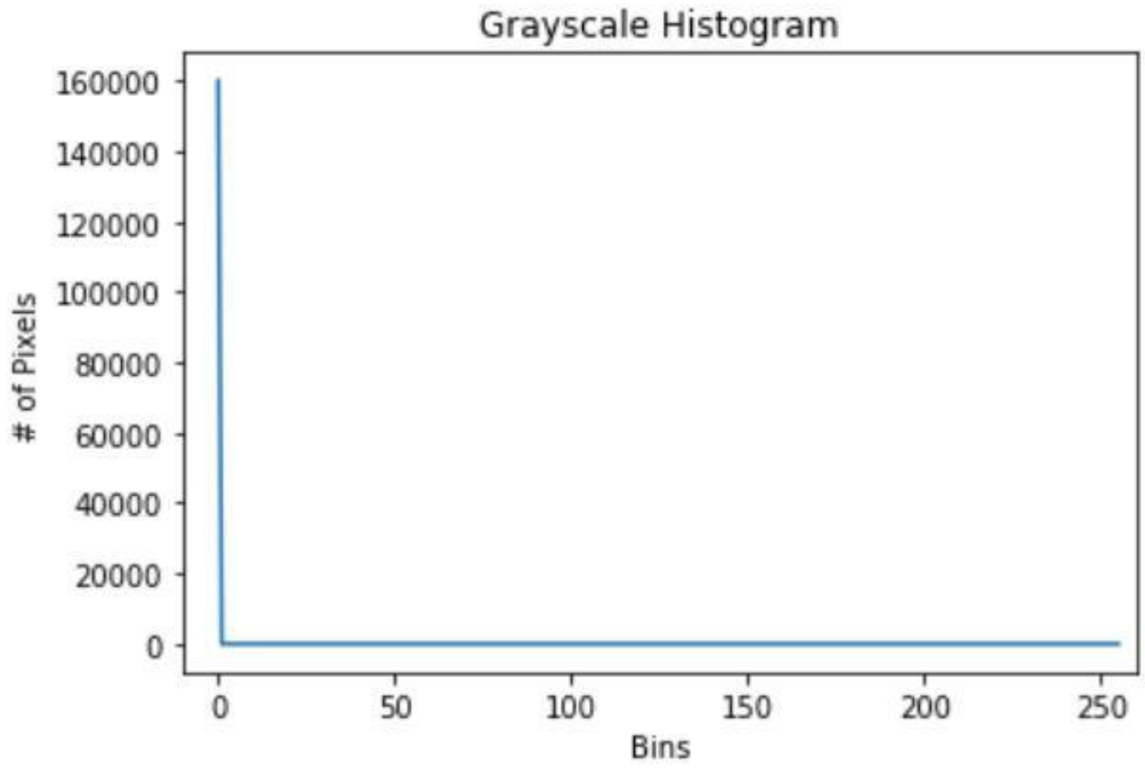


Image Processing using OpenCV VIII

- **Histogram of White Image**

```
image_white_gray = cv2.cvtColor(white_image, cv2.COLOR_BGR2GRAY)
plot_histogram(image_white_gray)
```

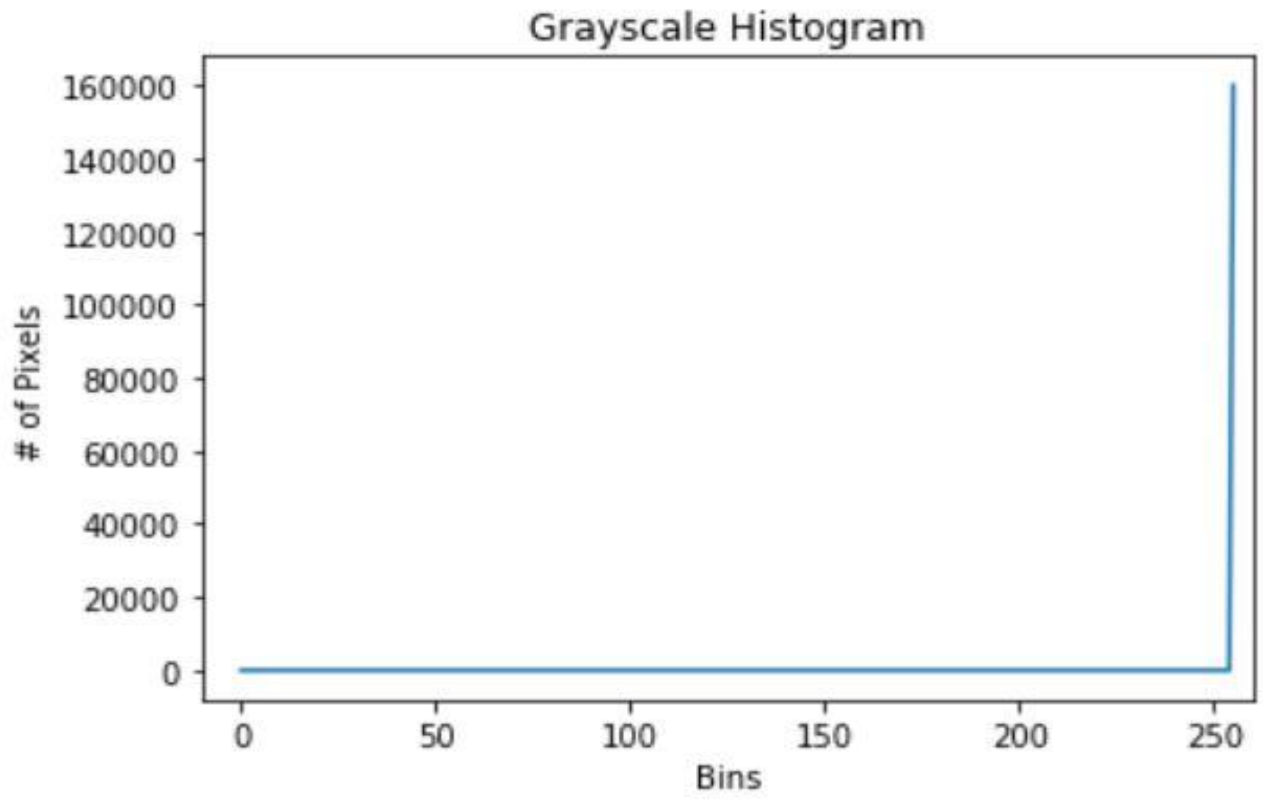


Image Processing at Flexiloans

- Business document classification
- Fraud document detection
- Automated bank cheque processing

Business Document Classification I

Business Challenge :

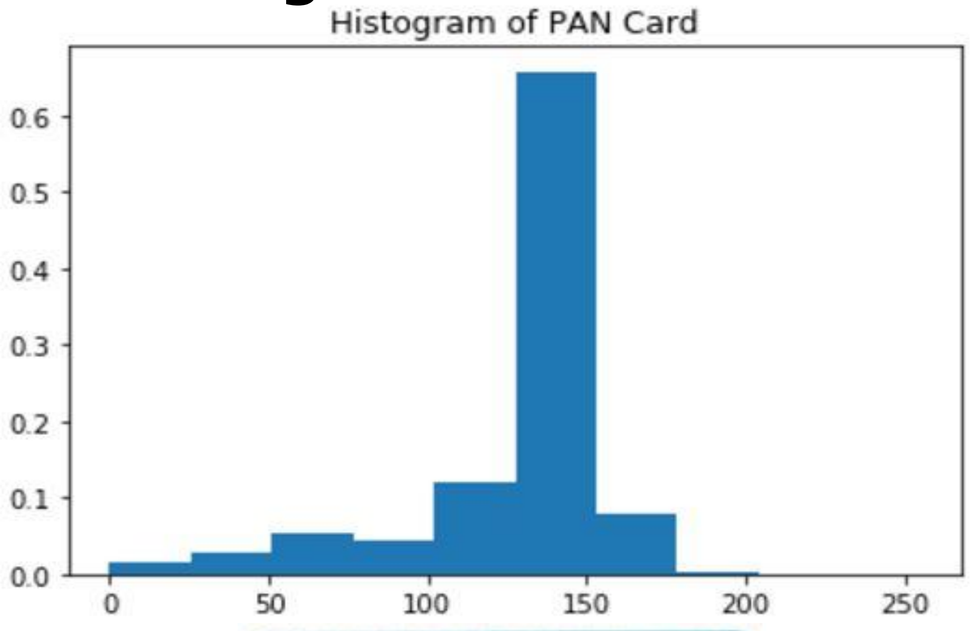
- An average human takes approximately 10-15 minutes to classify and digitize business documents.
- In a real-time lending environment, automated document classification systems should be deployed to reduce the human effort from minutes to microseconds without compromising on the accuracy

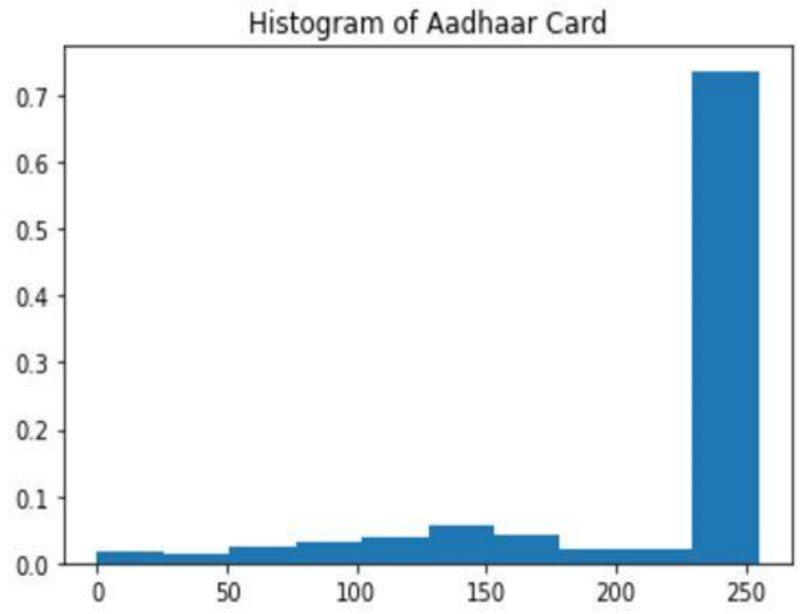
Solutions :

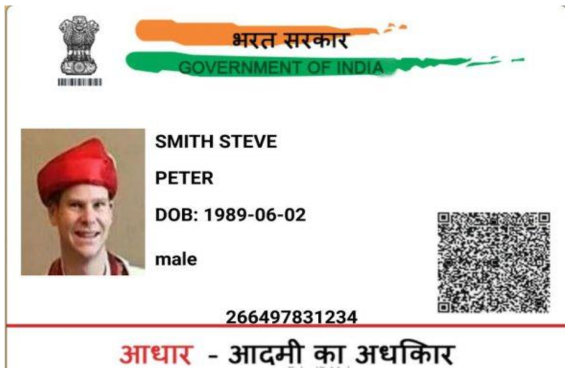
- Histogram intersection techniques
- OCR
- Pretrained models (Inception, VGG16)
- CNN (convolutional neural network)

Business Document Classification II

- **Histogram intersection techniques**





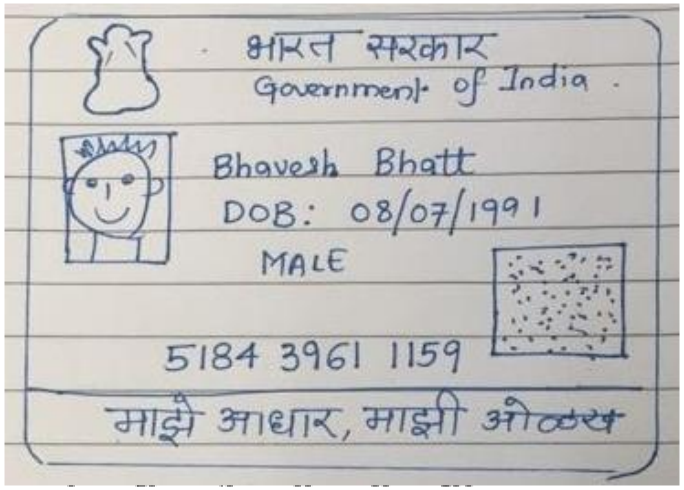

भरत सरकार
GOVERNMENT OF INDIA
SMITH STEVE
PETER
DOB: 1989-06-02
male
266497831234
आधार - आदमी का अधिकार

Business Document Classification III

- **Optical Character Recognition**

```
In [37]: img_loc = '/home/user/Desktop/Test_Aadhar/WhatsApp Image 2018-01-19 at 12.52.44 PM.jpeg'
```

```
In [38]: predict_aadhar_or_not_aadhar(img_loc)
```

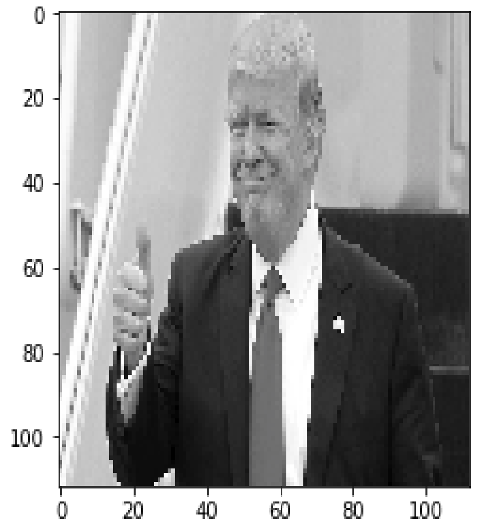

भारत सरकार
Government of India
Bhavesh Bhatt
DOB: 08/07/1991
MALE
5184 3961 1159
माझे आधार, माझी ओळख

```
INFO:tensorflow:Restoring parameters from alexnet_pan_aadhar_with_background.cpkt
The given image is not aadhaar card
```

Business Document Classification IV

```
In [33]: img_loc = '/home/user/Desktop/Test_Aadhar/Donald_Trump.jpg'
```

```
In [34]: predict_aadhar_or_not_aadhar(img_loc)
```

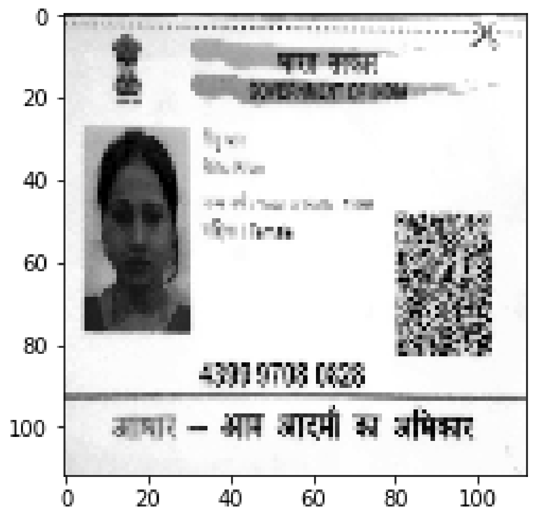


```
INFO:tensorflow:Restoring parameters from alexnet_pan_aadhar_with_background.cpkt
The given image is not aadhaar card
```

Business Document Classification V

```
In [23]: img_loc = '/home/user/Desktop/Test_Aadhar/Aadhar_Card.jpg'
```

```
In [24]: predict_aadhar_or_not_aadhar(img_loc)
```



```
INFO:tensorflow:Restoring parameters from alexnet_pan_aadhar_with_background.cpkt
The given image is aadhaar card
```

Binarizing an Image I

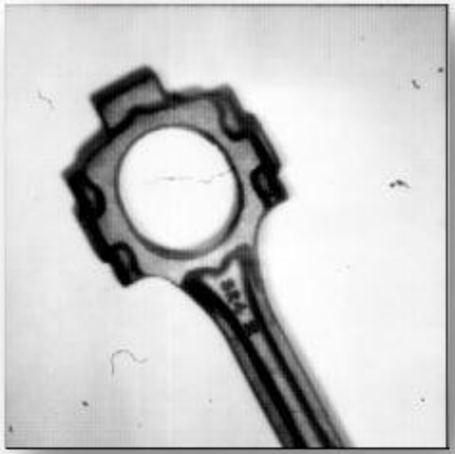
- Why do we binarize an image?
  Converting to binary is often used in order to find a Region Of Interest - a portion of the image that is of interest for further processing.

- Global Thresholding - If pixel value is greater than a threshold value, it is assigned '1' - white else it is assigned '0' - black.

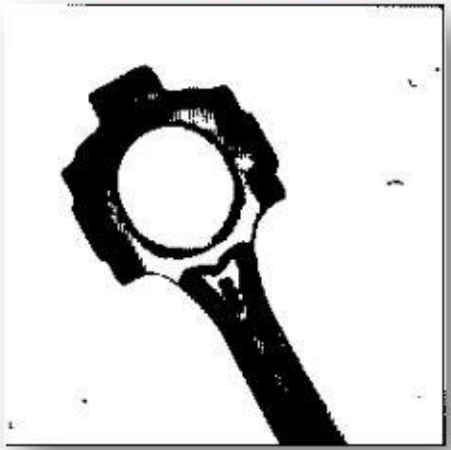
- The OpenCV function used for this task is cv2.threshold

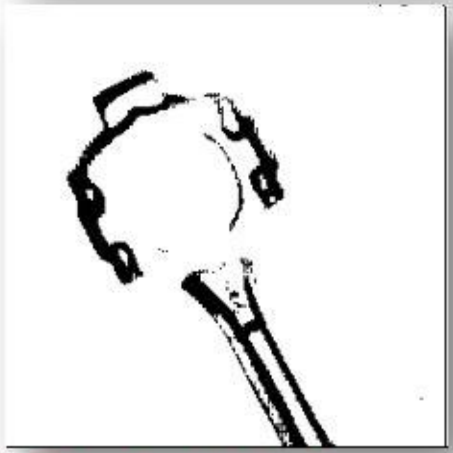
Binarizing an Image II

Original Image

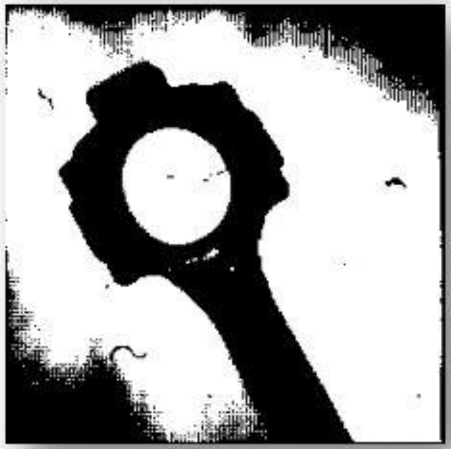


Binary Image





Threshold too low



Threshold too high

Otsu Thresholding I

- Otsu's thresholding method involves iterating through all the possible threshold values and calculating a measure of spread for the pixel levels.

- The 2 clusters should be as tight as possible for which the **within class variance** should be minimum yet the distance separating them should be maximum i.e. inter-class or **between class variance**.

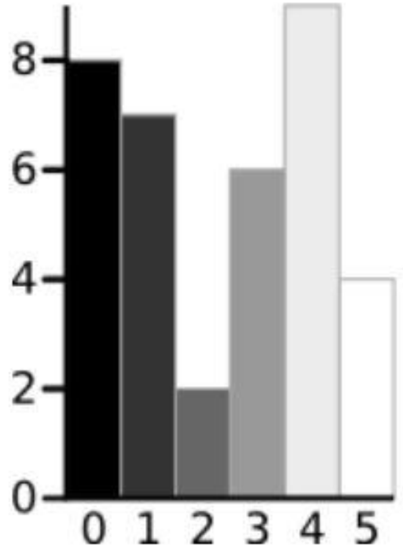
- In simple terms, I want to find a threshold value which best divides my histogram into 2 halves.

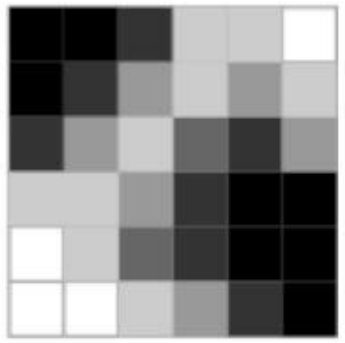
- Reference
  http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html

Otsu Thresholding II

- Consider a 6x6 image which has 6 grayscale levels.

- 0 represents black and 5 represents white.

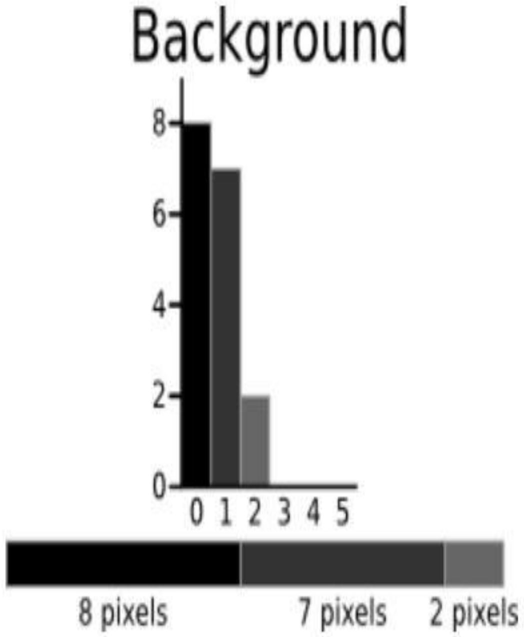




- Reference
  http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html

Otsu Thresholding III

- Assuming the threshold is 3



$$\text{Weight} \quad W_b = \frac{8+7+2}{36} = 0.4722$$

$$\text{Mean} \quad \mu_b = \frac{(0\times 8)+(1\times 7)+(2\times 2)}{17} = 0.6471$$

$$\text{Variance} \quad \sigma_b^2 = \frac{((0-0.6471)^2\times 8)+((1-0.6471)^2\times 7)+((2-0.6471)^2\times 2)}{17}$$
$$= \frac{(0.4187\times 8)+(0.1246\times 7)+(1.8304\times 2)}{17}$$
$$= 0.4637$$

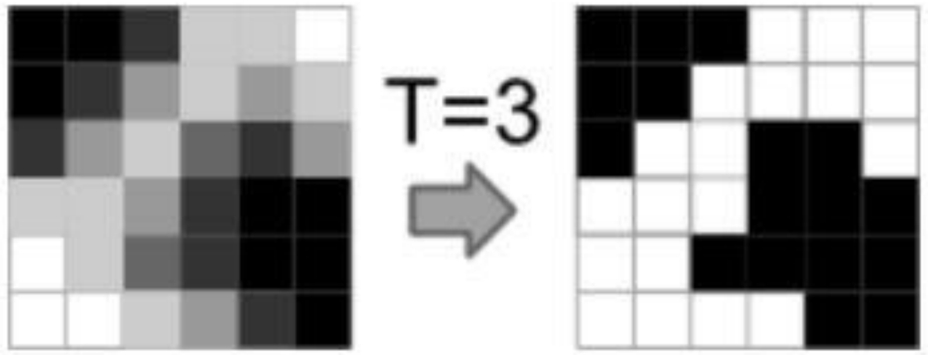
$$\text{Within Class Variance} \quad \sigma_W^2 = W_b\,\sigma_b^2 + W_f\,\sigma_f^2 = 0.4722 * 0.4637 + 0.5278 * 0.5152$$
$$= 0.4909$$

- Reference

http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html

Otsu Thresholding IV

Within Class Variance	$\sigma^2_W = 3.1196$	$\sigma^2_W = 1.5268$	$\sigma^2_W = 0.5561$	$\sigma^2_W = 0.4909$	$\sigma^2_W = 0.9779$	$\sigma^2_W = 2.2491$



- Otsu's thresholding is a simple method which works really well when the image histogram has 2 distinct peaks

- Reference
  http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html

Thank You

- Link to the code and presentation - https://github.com/bhattbhavesh91/PyDataMumbai-21April_2018