PROJECT REPORT

Major Project on the topics:

1. Choose any dataset of your choice and apply a suitable CLASSIFIER/REGRESSOR .

2. Create any of the Image Processing Projects using Numpy and/or OpenCV.

Submitted by:

NIVETHA SREE M
IIIrd Year
Computer Science and Engineeering
Adhiyamaan College of Engineering, Hosur.

Date: 14-01-2023

Objective: Choose any dataset of your choice and apply a suitable CLASSIFIER/REGRESSOR .

Dataset: https://www.kaggle.com/datasets/dhamur/airtel-prepaid

```
[ ] #Major Project 1
    #Choose any dataset of your choice and apply a suitable CLASSIFIER/REGRESSOR .
    #Dataset:https://www.kaggle.com/datasets/dhamur/airtel-prepaid
```

```
[2] #1. importing library and creating dataframe
    import pandas as pd
    df = pd.read_csv('/content/Airtel Prepaid.csv')
    df
```

	Price	Data(GB) per Day	Data(GB)	Days	Additional Benefits
0	3359	2.5	0	365	Disney + Hotstar 1 year
1	2999	2.0	0	365	0
2	999	2.5	0	84	Amazon prime membership
3	839	2.0	0	84	Disney + Hotstar 3 Months
4	719	1.5	0	84	0
5	699	3.0	0	56	Amazon prime membership
6	666	1.5	0	77	0
7	599	3.0	0	28	Disney + Hotstar 1 year
8	549	2.0	0	56	0
9	499	2.0	0	28	Disney + Hotstar 1 year
10	479	1.5	0	56	0
11	455	0.0	6	84	0

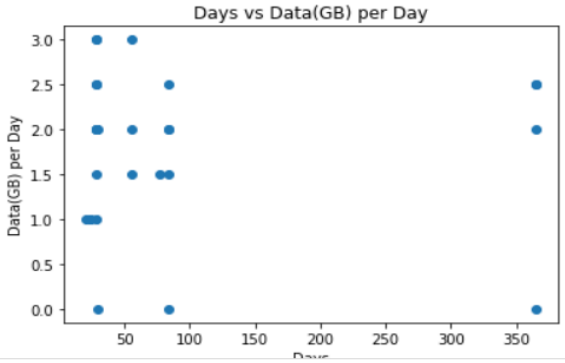
✓ 0s completed at 11:17 PM

```
[3] #2. Preprocessing – Filtering of Data ( Data Cleaning, Encoding, Dropping values, Missing values )
    x= df.iloc[:,:-1].values
```

```
[4] #Encoding
    from sklearn.preprocessing import LabelEncoder
    label_encoder_x= LabelEncoder()
    x[:, 0]= label_encoder_x.fit_transform(x[:, 0])
```

```
#3. Data Visualization
import matplotlib.pyplot as plt
plt.scatter(df['Days'],df['Data(GB) per Day'])
plt.title('Days vs Data(GB) per Day')
plt.ylabel('Data(GB) per Day')
plt.xlabel('Days')
```

```
Text(0.5, 0, 'Days')
```

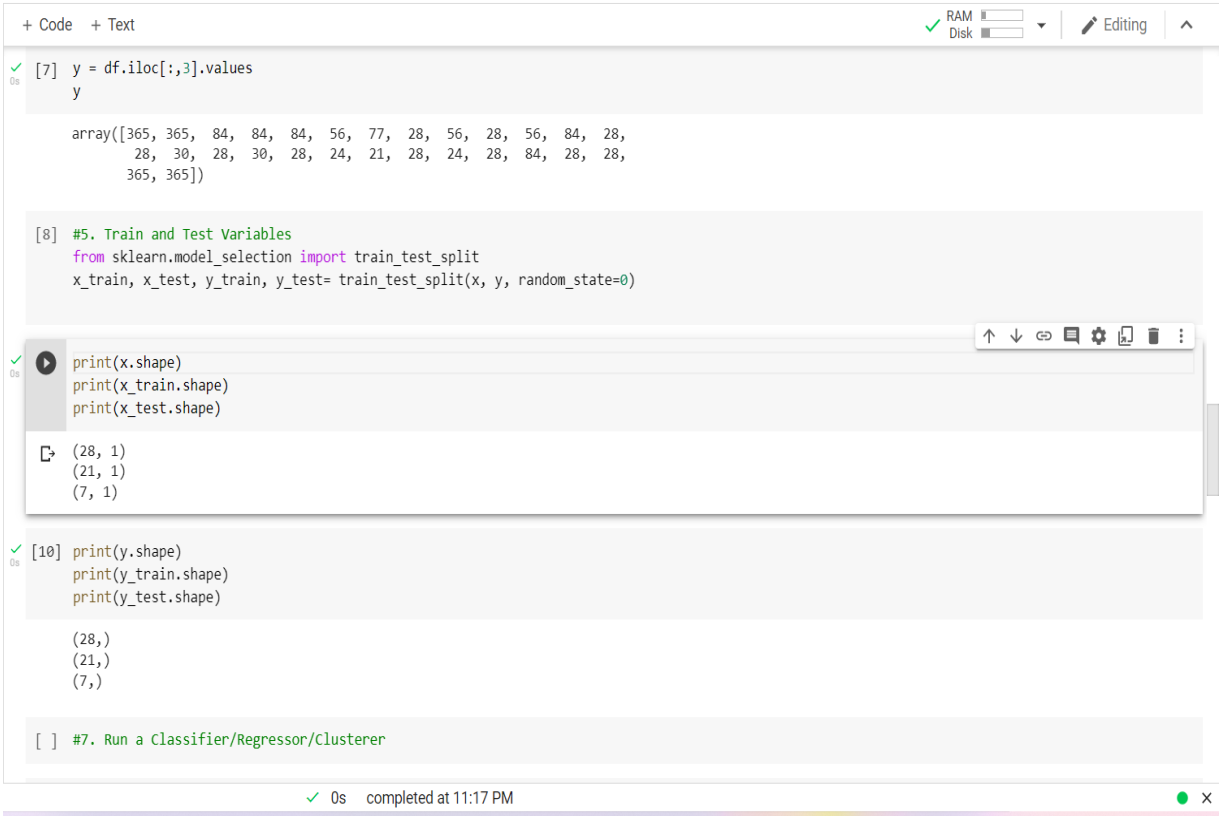


✓ 0s completed at 11:17 PM

```
#4. Divide into Input and Output (x – i/p , y-o/p)
x = df.iloc[:,3:4].values
x
```

```
array([[365],
       [365],
       [ 84],
       [ 84],
       [ 84],
       [ 56],
       [ 77],
       [ 28],
       [ 56],
       [ 28],
       [ 56],
       [ 84],
       [ 28],
       [ 28],
       [ 30],
       [ 28],
       [ 30],
       [ 28],
       [ 24],
       [ 21],
       [ 28],
       [ 24],
       [ 28],
       [ 84],
       [ 28],
       [ 28],
       [365],
       [365]])
```

✓ 0s completed at 11:17 PM



```
+ Code  + Text                                                    RAM Disk   Editing

[7]  y = df.iloc[:,3].values
     y

     array([365, 365,  84,  84,  84,  56,  77,  28,  56,  28,  56,  84,  28,
             28,  30,  28,  30,  28,  24,  21,  28,  24,  28,  84,  28,  28,
            365, 365])

[8]  #5. Train and Test Variables
     from sklearn.model_selection import train_test_split
     x_train, x_test, y_train, y_test= train_test_split(x, y, random_state=0)

     print(x.shape)
     print(x_train.shape)
     print(x_test.shape)

     (28, 1)
     (21, 1)
     (7, 1)

[10] print(y.shape)
     print(y_train.shape)
     print(y_test.shape)

     (28,)
     (21,)
     (7,)

[ ]  #7. Run a Classifier/Regressor/Clusterer

                                                    0s  completed at 11:17 PM
```

```
[ ]  #7. Run a Classifier/Regressor/Clusterer

[16] from sklearn.linear_model import LinearRegression
     model = LinearRegression()

[17] #8. Fit the model
     model.fit(x,y)

     LinearRegression()

[18] y_pred = model.predict(x) #Using the input values, predicting the output
     y_pred

     array([365., 365.,  84.,  84.,  84.,  56.,  77.,  28.,  56.,  28.,  56.,
             84.,  28.,  28.,  30.,  28.,  30.,  28.,  24.,  21.,  28.,  24.,
             28.,  84.,  28.,  28., 365., 365.])

[19] y

     array([365, 365,  84,  84,  84,  56,  77,  28,  56,  28,  56,  84,  28,
             28,  30,  28,  30,  28,  24,  21,  28,  24,  28,  84,  28,  28,
            365, 365])

[21] #Predict the output
     model.predict([[84]])

                                                    0s  completed at 11:17 PM
```

+ Code + Text

RAM Disk

Editing

```
[21] #Predict the output
     model.predict([[84]])
```

```
array([84.])
```

```
[22] m = model.coef_ #slope(m)
     m
```

```
array([1.])
```

```
[23] C = model.intercept_ # constant/y-intercept
     C
```

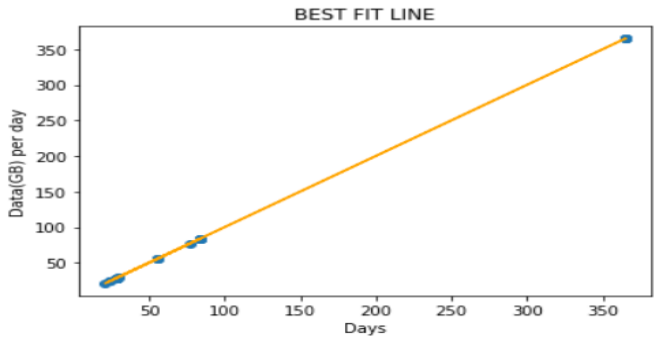
```
1.4210854715202004e-14
```

```
[24] m*2000 + C      #y = mx+c
```

```
array([2000.])
```

```
[26] #10. Visualization of the result
     plt.scatter(x,y)
     plt.plot(x,y_pred,c = 'orange')
     plt.title('BEST FIT LINE')
     plt.xlabel('Days')
     plt.ylabel('Data(GB) per day')
```

```
Text(0, 0.5, 'Data(GB) per day')
```




```
[ ]
```

```
[ ]
```

0s completed at 11:17 PM

Objective: Create any of the Image Processing Projects using Numpy and/or OpenCV.

1. Changing Colorspace

m1.py - C:\Users\admin\Documents\pro ds\m1.py (3.11.1)

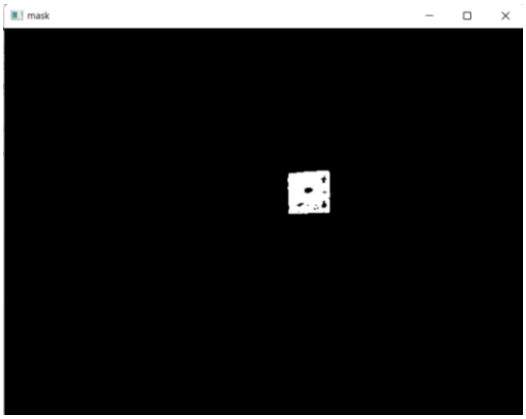
File Edit Format Run Options Window Help

```
#importing module
import cv2 as cv
import numpy as np

#capturing
cap = cv.VideoCapture(0)
while(1):
    _, frame = cap.read()  # Take each frame and perform reading

    # Convert BGR to HSV
    hsv = cv.cvtColor(frame, cv.COLOR_BGR2HSV)

    # define range of blue color in HSV
    lower_blue = np.array([110,50,50])
    upper_blue = np.array([130,255,255])

    # Threshold the HSV image to get only blue colors
    mask = cv.inRange(hsv, lower_blue, upper_blue)

    # Bitwise-AND mask and original image
    res = cv.bitwise_and(frame,frame, mask= mask)

    #display images
    cv.imshow('frame',frame)
    cv.imshow('mask',mask)
    cv.imshow('res',res)
    k = cv.waitKey(5) & 0xFF
    if k == 27:
        break
cv.waitKey()
cv.destroyAllWindows()
```

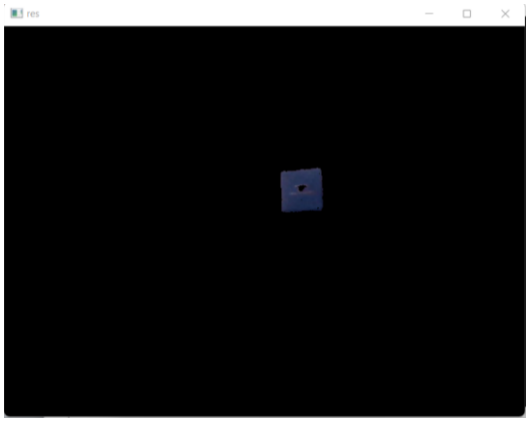
Output:

Original

Threshold



Bit-wise image



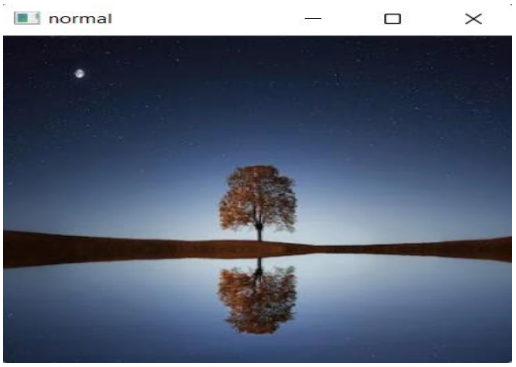
2. Bluring of the image

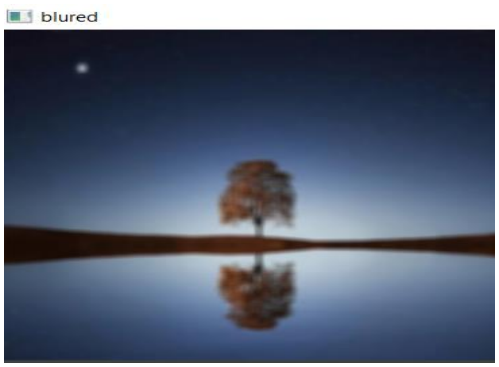
m6.py - C:\Users\admin\Documents\pro ds\m6.py (3.10.5)

File Edit Format Run Options Window Help

```
import cv2

#reading image
img = cv2.imread('image1.jpg')

#resizing
img = cv2.resize(img,(300,300))

#bluring of image
blured=cv2.blur(img,(5,5))

#displaying image
cv2.imshow('normal',img)
cv2.imshow('blured',blured)
cv2.waitKey()
cv2.destroAllWindows()
```

Output





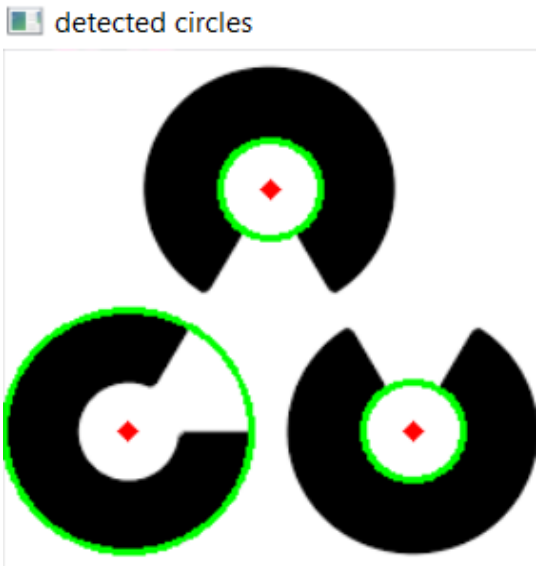
3. Hough Circle

m4.py - C:\Users\admin\Documents\pro ds\m4.py (3.10.5)

File Edit Format Run Options Window Help

```
import numpy as np
import cv2 as cv

#read image from source
img = cv.imread('image.png',0)
img = cv.medianBlur(img,5)

#grayscale
cimg = cv.cvtColor(img,cv.COLOR_GRAY2BGR)
#gradient method
circles = cv.HoughCircles(img,cv.HOUGH_GRADIENT,1,20,
                          param1=50,param2=30,minRadius=0,maxRadius=0)
circles = np.uint16(np.around(circles))
for i in circles[0,:]:

 # draw the outer circle((image, center_coordinates, color, thickness))
    cv.circle(cimg,(i[0],i[1]),i[2],(0,255,0),2)

# draw the center of the circle
#((image, center_coordinates, radius, color, thickness))

    cv.circle(cimg,(i[0],i[1]),2,(0,0,255),3)

#displaying image
cv.imshow('detected circles',cimg)
cv.waitKey(0)
cv.destroyAllWindows()
```

Output:



4. Grayscale

s4.py - C:\Users\admin\Documents\pro as\s4.py (3.10.5)

File Edit Format Run Options Window Help

```
import cv2
img = cv2.imread('image1.jpg')#reading the image
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)#conversion into grayscale
cv2.imshow('NORMAL IMAGE',img) #displays the original image
cv2.imshow('GRAY SCALE IMAGE',gray)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:

Normal

Grayscale

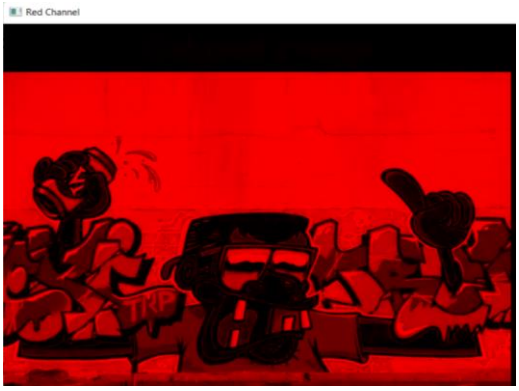
5. Splitting of image into diffrerent colour channels

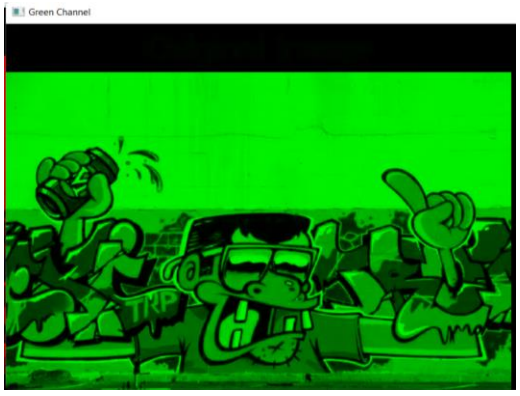
m5.py - C:\Users\admin\Documents\pro ds\m5.py (3.10.5)

File Edit Format Run Options Window Help

```
import numpy as np
import cv2 as cv

#reading image from source
input_image = cv.imread('b.jpg')
blue, green, red = cv.split(input_image)

#black background and splitting of image to different color channel
blue_channel = np.zeros(input_image.shape, input_image.dtype)
green_channel = np.zeros(input_image.shape, input_image.dtype)
red_channel = np.zeros(input_image.shape, input_image.dtype)

#matching each color channel to a 3D dimension:

cv.mixChannels([blue, green, red], [blue_channel], [0,0]) #Blue
cv.mixChannels([blue, green, red], [green_channel], [1,1]) #green
cv.mixChannels([blue, green, red], [red_channel], [2,2]) #red

# Displaying the three obtained images
cv.imshow('Blue Channel', blue_channel)
cv.imshow('Green Channel', green_channel)
cv.imshow('Red Channel', red_channel)

cv.waitKey(0)
cv.destroyAllWindows()
```

Output:





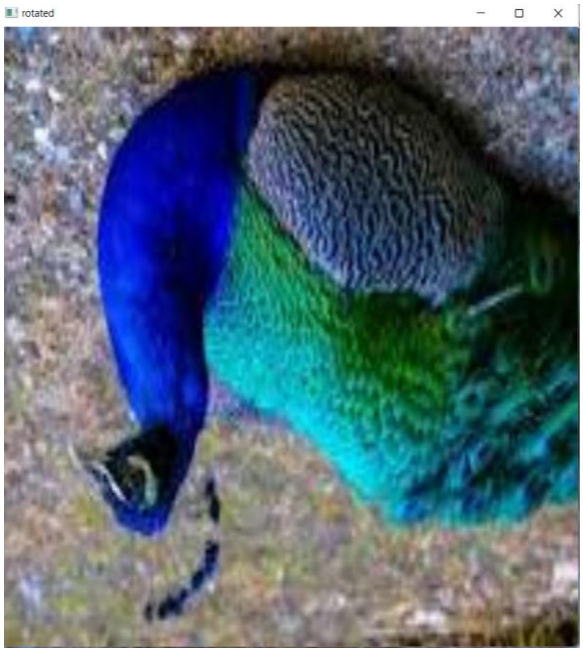
6. Flipping of the image

m7.py - C:/Users/admin/Documents/pro ds/m7.py (3.10.5)

File Edit Format Run Options Window Help

```
import cv2

#reading the image
img = cv2.imread('im.png')

#resizing image
img = cv2.resize(img,(700,700))

#rotating the image upside-down
rotate = cv2.flip(img,0)

#displaying the image
cv2.imshow('normal',img)
cv2.imshow('rotated',rotate)
cv2.waitKey()
cv2.destroyAllWindows()
```

Output:



7. Reversing the color of the image

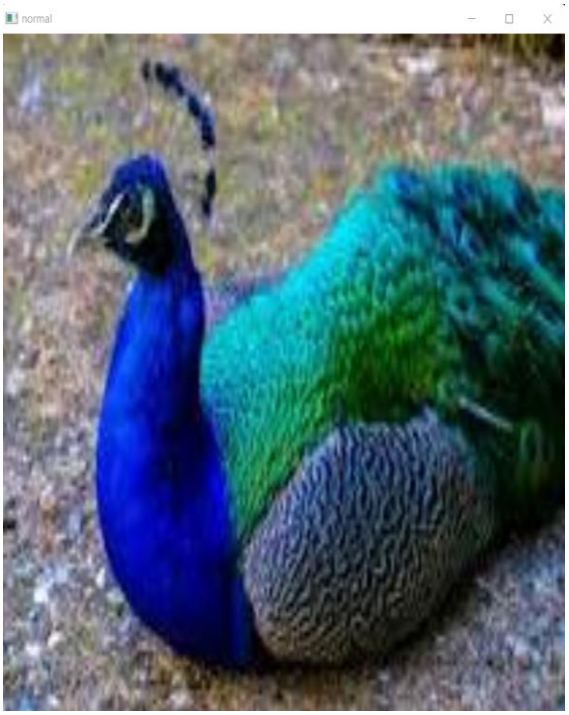
m8.py - C:/Users/admin/Documents/pro ds/m8.py (3.10.5)

File Edit Format Run Options Window Help

```
import cv2

#reading the image
img = cv2.imread('im.png')

#resizing image
img = cv2.resize(img,(700,700))

#reversing the color of the images
i_img =cv2.bitwise_not(img)

#displaying the image
cv2.imshow('normal',img)
cv2.imshow('inverted',i_img)
cv2.waitKey()
cv2.destroyAllWindows()
```

Output:

Normal



Color reversed image

8. Threshold

m9.py - C:/Users/admin/Documents/pro ds/m9.py (3.10.5)

File Edit Format Run Options Window Help

```
import cv2

#reading the image
img = cv2.imread('im.png')

#resizing image
img = cv2.resize(img,(1000,1000))

#grayscale
img = cv2.imread('im.png',0)

#thresholding
var,t = cv2.threshold(img, 100, 100, cv2.THRESH_BINARY)

#displaying the image
cv2.imshow('normal',img)
cv2.imshow('threshold',t)
cv2.waitKey()
cv2.destroyAllWindows()
```

Output:

Normal

Threshold

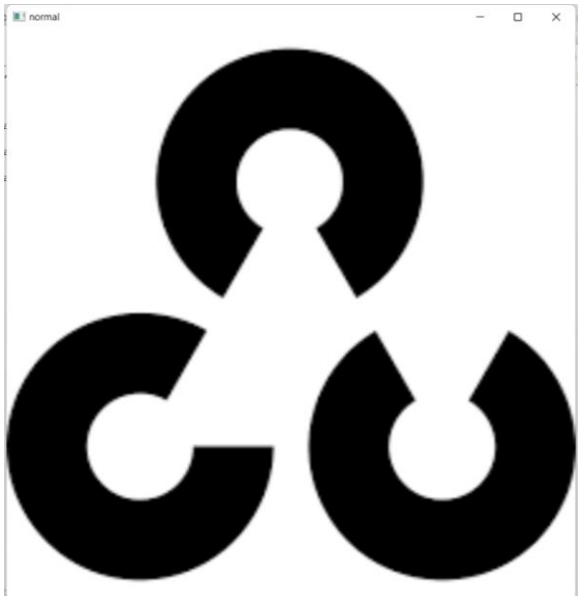
9. Detecting the edge of image

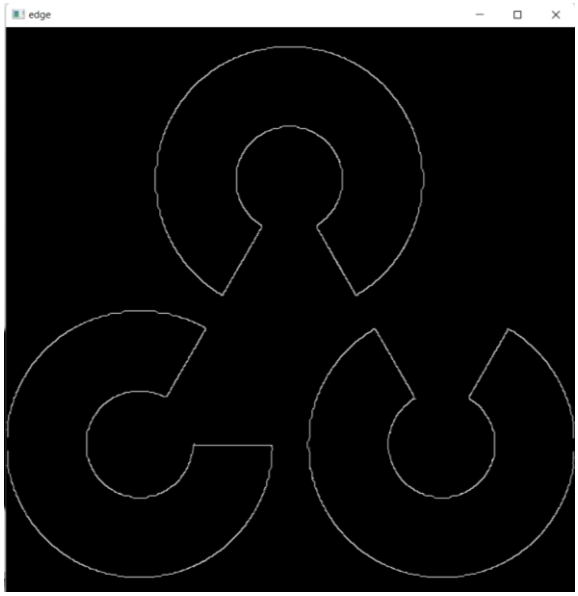
m10.py - C:/Users/admin/Documents/pro ds/m10.py (3.10.5)

File Edit Format Run Options Window Help

```
import cv2

#reading the image
img = cv2.imread('image.png')

#resizing image
img = cv2.resize(img,(700,700))

#detection of edges
i_edge = cv2.Canny(img, 100, 100)

#displaying the image
cv2.imshow('normal',img)
cv2.imshow('edge',i_edge)
cv2.waitKey()
cv2.destroyAllWindows()
```

Output:





Links:

GitHub account: https://github.com/Nivetha-Sree-M

Link to drive: https://tinyurl.com/rinexprojnivetha

Repository Link: https://github.com.Nivetha-Sree-M/Rinex-project

THANK YOU