1. **#Negative imgae**

```
import cv2
abc=cv2.imread("E:/SANIKA/IPMV PRAC/images/cameraman.png",0)
print("image:",abc)
cv2.imshow('image',abc)
cv2.waitKey(0)
xyz=~abc
cv2.imshow('Negative image',xyz)
print("negative image:",xyz)
```

2. **#contrast image**

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
abc=cv2.imread("E:/SANIKA/IPMV PRAC/images/cameraman.png",0)
print("image:",abc)
plt.subplot(1,2,1)
plt.title('image')
plt.imshow(abc,cmap='gray')
contrast=3.0
xyz=np.clip(contrast*abc,0,255)
plt.subplot(1,2,2)
plt.title('contrast image')
plt.imshow(xyz,cmap='gray')
plt.show()
```

3. **#Log transform**

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
abc=cv2.imread("E:/SANIKA/IPMV PRAC/images/cameraman.png",0)
print(abc)
plt.subplot(1,2,1)
plt.title('image')
plt.imshow(abc,cmap='gray')
xyz=np.zeros(abc.shape,abc.dtype)
c=255/np.log(1+np.max(abc))
for i in range(abc.shape[0]):
   for j in range(abc.shape[1]):
      xyz[i,j]=c*np.log(1+abc[i,j])
plt.subplot(1,2,2)
plt.title('log image')
plt.imshow(xyz,cmap='gray')
plt.show()
```

4. **#Thresholding**

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
abc=cv2.imread("E:/SANIKA/IPMV PRAC/images/cameraman.png",0)
print("image:",abc)
plt.subplot(1,3,1)
plt.title('image')
plt.imshow(abc,cmap='gray')
print(abc.shape,abc.dtype)
print("shape of the image:",abc.shape)
xyz=np.zeros(abc.shape,abc.dtype)
for i in range(70):
   for j in range(105):
      if(abc[i,j]<127):
         xyz[i,j]=0
      else:
         xyz[i,j]=255
plt.subplot(1,3,2)
plt.title('threshold image 1')
plt.imshow(xyz,cmap='gray')
b,a=cv2.threshold(xyz,127,255,0)
plt.subplot(1,3,3)
plt.title('threshold image 2')
plt.imshow(a,cmap='gray')
plt.show()
```

5. #Histogram and Image Equalization

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
abc=cv2.imread("E:/SANIKA/IPMV PRAC/images/cameraman.png",0)
print("image:",abc)
plt.subplot(2,2,1)
plt.title('image')
plt.imshow(abc,cmap='gray')
#for histogram
plt.subplot(2,2,2)
plt.title('histogram')
plt.hist(abc.flatten(),256,[0,255],color='r')
#for Equalization of image
img_equalized=cv2.equalizeHist(abc)
plt.subplot(2,2,3)
plt.title('Equalized image')
plt.imshow(img_equalized,cmap='gray')
#for Equalized image histogram
plt.subplot(2,2,4)
plt.title('Equalized histogram')
plt.hist(img_equalized.flatten(),256,[0,255],color='b')
plt.subplots_adjust(wspace=0.4,hspace=0.5)
plt.show()
```

6. **#Low pass filter**

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
abc=cv2.imread("E:/SANIKA/IPMV PRAC/images/cameraman.png",0)
plt.subplot(1,2,1)
plt.title('image')
plt.imshow(abc,cmap='gray')
#LPF
LPF_mask=np.ones((3,3))/9
LPF_image=cv2.filter2D(abc,-1,LPF_mask)
plt.subplot(1,2,2)
plt.title('LPF')
plt.imshow(LPF_image,cmap='gray')
plt.show()
```

7. **#High pass filter**

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
abc=cv2.imread("E:/SANIKA/IPMV PRAC/images/cameraman.png",0)
plt.subplot(1,2,1)
plt.title('image')
plt.imshow(abc,cmap='gray')
#HPF
HPF_mask=np.array(([-1,-1,-1],[-1,8,-1],[-1,-1,-1]))
HPF_image=cv2.filter2D(abc,-1,HPF_mask)
plt.subplot(1,2,2)
plt.title('HPF')
plt.imshow(HPF_image,cmap='gray')
plt.show()
```

8. **#High boost filter**

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
abc=cv2.imread("E:/SANIKA/IPMV PRAC/images/cameraman.png",0)
plt.subplot(1,2,1)
plt.title('image')
plt.imshow(abc,cmap='gray')
#HBF
HBF_mask=np.array(([-1,-1,-1],[-1,9.1,-1],[-1,-1,-1]))
HBF_image=cv2.filter2D(abc,-1,HBF_mask)
plt.subplot(1,2,2)
plt.title('HBF')
plt.imshow(HBF_image,cmap='gray')
plt.show()
```

9. **#Horizontal edge detection**

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
abc=cv2.imread("E:/SANIKA/IPMV PRAC/images/blobs.png",0)
plt.subplot(1,2,1)
plt.title('image')
plt.imshow(abc,cmap='gray')
#hed of image
hed=np.array(([-1,-1,-1],[2,2,2],[-1,-1,-1]))
HE_image=cv2.filter2D(abc,-1,hed)
plt.subplot(1,2,2)
plt.title('HED')
plt.imshow(HE_image,cmap='gray')
plt.show()
```

10. **#Vertical edge detection**

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
abc=cv2.imread("E:/SANIKA/IPMV PRAC/images/blobs.png",0)
plt.subplot(1,2,1)
plt.title('image')
plt.imshow(abc,cmap='gray')
#ved of image
ved=np.array(([-1,2,-1],[-1,2,-1],[-1,2,-1]))
VE_image=cv2.filter2D(abc,-1,ved)
plt.subplot(1,2,2)
plt.title('VED')
plt.imshow(VE_image,cmap='gray')
plt.show()
```

11. **#Diagonal edge detection**

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
abc=cv2.imread("E:/SANIKA/IPMV PRAC/images/blobs.png",0)
plt.subplot(1,2,1)
plt.title('image')
plt.imshow(abc,cmap='gray')
#ded of image
ded=np.array(([2,-1,-1],[-1,2,-1],[-1,-1,2]))
DE_image=cv2.filter2D(abc,-1,ded)
plt.subplot(1,2,2)
plt.title('DED')
plt.imshow(DE_image,cmap='gray')
plt.show()
```

12. **#image erosion and dilation**

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
abc=cv2.imread("E:/SANIKA/IPMV PRAC/images/blobs.png",0)
plt.subplot(2,2,1)
plt.title('image')
plt.imshow(abc,cmap='gray')
strct=np.array([[255,0,255],[0,255,0],[255,0,255]],np.uint8)
plt.subplot(2,2,2)
plt.title('structure image')
plt.imshow(strct,cmap='gray')
#for erosion
img_erosion=cv2.erode(abc,strct,iterations=1)
plt.subplot(2,2,3)
plt.title('eroded image')
plt.imshow(img_erosion,cmap='gray')
#for dilation
img_dilation=cv2.dilate(abc,strct,iterations=1)
plt.subplot(2,2,4)
plt.title('dilated image')
plt.imshow(img_dilation,cmap='gray')
plt.subplots_adjust(wspace=0.4,hspace=0.5)
plt.show()
```

13. **#image opening**

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
abc=cv2.imread("E:/SANIKA/IPMV PRAC/images/cameraman.png",0)
plt.subplot(2,2,1)
plt.title('image')
plt.imshow(abc,cmap='gray')
strct=np.array([[255,0,255],[0,255,0],[255,0,255]],np.uint8)
#for erosion
img_erosion=cv2.erode(abc,strct,iterations=1)
plt.subplot(2,2,2)
plt.title('erode image')
plt.imshow(img_erosion,cmap='gray')
#for open
img_open=cv2.dilate(img_erosion,strct,iterations=1)
plt.subplot(2,2,3)
plt.title('open')
plt.imshow(img_open,cmap='gray')
#for opening image
img_opening=cv2.morphologyEx(abc,cv2.MORPH_OPEN,strct,iterations=1)
plt.subplot(2,2,4)
plt.title('opening image')
plt.imshow(img_opening,cmap='gray')
plt.show()
```

14. **#image closeing**

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
abc=cv2.imread("E:\SANIKA\IPMV PRAC\images\circles.png",0)
plt.subplot(2,2,1)
plt.title('image')
plt.imshow(abc,cmap='gray')
strct=np.array([[255,0,255],[0,255,0],[255,0,255]],np.uint8)
#for dilation
img_dilation=cv2.dilate(abc,strct,iterations=1)
plt.subplot(2,2,2)
plt.title('dilated image')
plt.imshow(img_dilation,cmap='gray')
#for close
img_close=cv2.erode(img_dilation,strct,iterations=1)
plt.subplot(2,2,3)
plt.title('image close')
plt.imshow(img_close,cmap='gray')
#for closeing image
img_closeing=cv2.morphologyEx(abc,cv2.MORPH_CLOSE,strct,iterations=1)
plt.subplot(2,2,4)
plt.title('closeing image')
plt.imshow(img_closeing,cmap='gray')
plt.show()
```

15. **#edge detection**

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
abc=cv2.imread("E:/SANIKA/IPMV PRAC/images/blobs.png",0)
plt.subplot(2,2,1)
plt.title('image')
plt.imshow(abc,cmap='gray')
strct=np.array([[255,0,255],[0,255,0],[255,0,255]],np.uint8)
#edge detection of image
img_erosion=cv2.erode(abc,strct,iterations=1)
plt.subplot(2,2,2)
plt.title('Eroded image')
plt.imshow(img_erosion,cmap='gray')
xyz=abc-img_erosion
plt.subplot(2,2,3)
plt.title('edge image')
plt.imshow(xyz,cmap='gray')
plt.show()
```

16. **#K-means clustering**

```
import cv2
import numpy as np
import matplotlib.pylab as plt
abc=cv2.imread("E:/SANIKA/IPMV PRAC/images/nature.jpg",1)
print('original image:',abc)
abc=cv2.cvtColor(abc,cv2.COLOR_BGR2RGB)
plt.subplot(1,2,1)
plt.title('Original image')
plt.imshow(abc)
pixel_value=abc.reshape((-1,3))
pixel_value=np.float32(pixel_value)
k=3
criteria=(cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER,100,0.85)
retval,labels,centers=cv2.kmeans(pixel_value,k,None,criteria,10,cv2.KMEANS_RANDOM_CENTERS)
centers=np.uint8(centers)
segmented_data=centers[labels.flatten()]
segmented_image=segmented_data.reshape((abc.shape))
plt.subplot(1,2,2)
plt.title('new image')
plt.imshow(segmented_image)
plt.show()
```

17. **#verify digits data samples**

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets,svm,metrics
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
digits = datasets. load_digits ()
print ('keys of digits dataset:', digits. keys ())
print ('shape of data:' , digits.data.shape)
print ('data: ', digits.data)
print ('shape of target:', digits.target. shape)
print ('target: ' ,digits.target)
print (' shape of images: ', digits. images .shape)
for i in range (10):
   plt. subplot (2, 5, i+1)
   plt.axis ('off')
   plt. imshow (digits. images [i], cmap='gray_r')
   plt.title ('target: {}'. format (digits. target [i]))
plt. show ()
```

```
train_data, test_data, train_target, test_target=train_test_split (digits.data,
digits.target,test_size = 0.5, shuffle=True)
print ('shape of train data:', train_data.shape)
print ('shape of test data:', test_data. shape)
print ('shape of train target:', train_target. shape)
print ('shape of test target:', test_target. shape)
model_linear = svm.SVC (kernel = 'linear')
model_linear.fit (train_data, train_target)
pred=model_linear.predict (test_data)
acc = metrics.accuracy_score (test_target, pred)
print ('accuracy =', acc*100, '8')
confusion_matrics = metrics.confusion_matrix (test_target, pred)
cm_display = metrics. ConfusionMatrixDisplay (confusion_matrics)
cm_display.plot ()
plt. show ()
for i in range (4):
   plt. subplot (2,2, i+1)
   plt.axis ('off')
   abc = np. reshape (test_data [i], (8, 8))
   plt. imshow (abc, cmap = 'gray_r')
   plt.title ('Predicted output: {}'. format (pred[i]))
plt. show ()
```

18. #AND gate

```
a=1
w1=0
w2=0
b=0
print('w1={}, w2={}, b={}'.format(w1,w2,b))
x1=[1,1,-1,-1]
x2=[1,-1,1,-1]
t=[1,-1,-1,-1]
for j in range (1,3):
   print('Epoch:{}'.format(j))
   for i in range(4):
      yin=x1[i]*w1+x2[i]*w2+b
      if (yin>0):
         y=1
      elif(yin==0):
         y=0
      else:
         y=-1
```

```
        if (y==t[i]):
            print('x1={},x2={},y={},b={},w1={},w2={}'.format(x1[i],x2[i],y,b,w1,w2))
            continue
        else:
            w1=w1+a*t[i]*x1[i]
            w2=w2+a*t[i]*x2[i]
            b=b+a*t[i]
            print('x1={},x2={},y={},b={},w1={},w2={}'.format(x1[i],x2[i],y,b,w1,w2))
```

19. #OR gate

```
a=1
w1=0
w2=0
b=0
print('w1={}, w2={}, b={}'.format(w1,w2,b))
x1=[1,1,-1,-1]
x2=[1,-1,1,-1]
t=[1,1,1,-1]
for j in range (1,3):
    print('Epoch:{}'.format(j))
    for i in range(4):
        yin=x1[i]*w1+x2[i]*w2+b
        if (yin>0):
            y=1
        elif(yin==0):
            y=0
        else:
            y=-1
        if (y==t[i]):
            print('x1={},x2={},y={},b={},w1={},w2={}'.format(x1[i],x2[i],y,b,w1,w2))
            continue
        else:
            w1=w1+a*t[i]*x1[i]
            w2=w2+a*t[i]*x2[i]
            b=b+a*t[i]
            print('x1={},x2={},y={},b={},w1={},w2={}'.format(x1[i],x2[i],y,b,w1,w2))
```

#Power law

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
abc=cv2.imread("E:/SANIKA/IPMV PRAC/images/cameraman.png",0)
print(abc)
plt.subplot(1,3,1)
plt.title('image')
plt.imshow(abc,cmap='gray')
xyz=np.array(255*(abc/255)**0.2,dtype='uint8')
plt.subplot(1,3,2)
plt.title('gamma 1')
plt.imshow(xyz,cmap='gray')
pqr=np.array(255*(abc/255)**1,dtype='uint8')
plt.subplot(1,3,3)
plt.title('gamma 2')
plt.imshow(pqr,cmap='gray')
plt.show()
```