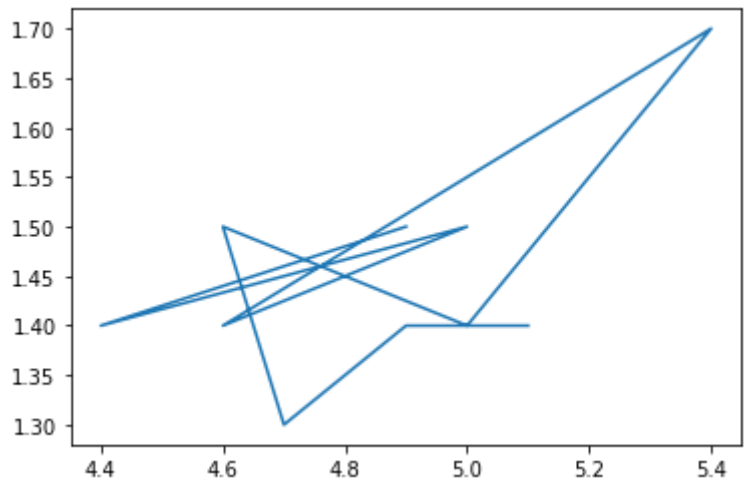
```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
import pandas as pd
from google.colab.patches import cv2_imshow as cv2


df = pd.read_csv('iris.csv')


df.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
x = df['sepal_length'][:10]
y = df['petal_length'][:10]

plt.plot(x,y)
```

```
[<matplotlib.lines.Line2D at 0x7f8d79347310>]
```




```
frame = cv.imread('sample.jfif')
cv2(frame)
cv.waitKey(0)
cv.destroyAllWindows()
```

```
some = cv.VideoCapture('sample2.gif')
while True:
  isTrue, frame = some.read()
  if isTrue:
    cv2(frame)
    if cv.waitKey(20) & 0xFF==ord('q'):
      break
some.release()
some.destroyAllWindows()
```

Digital Image Processing

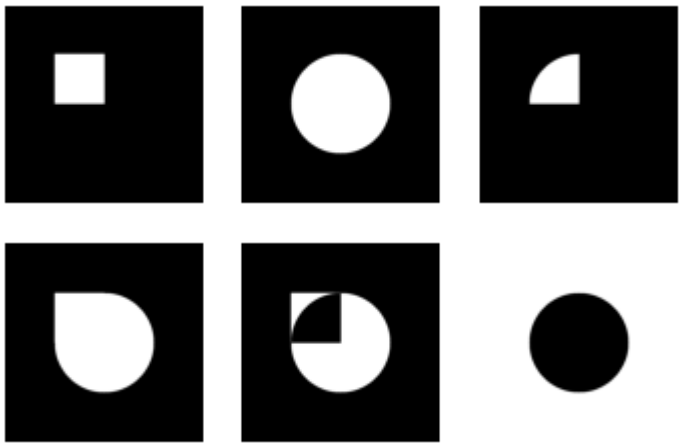
1) Swap red and blue planes using matplotlib

```
from matplotlib import pyplot as plt
import numpy as np

img1 = plt.imread('lake.jpg')
figure, plots = plt.subplots(ncols=3, nrows=1)
for i, subplot in zip(range(3), plots[::-1]):
    temp = np.zeros(img1.shape,dtype='uint8')
    temp[:,:,i] = img1[:,:,i]
    subplot.imshow(temp)
    subplot.set_axis_off()
plt.show()
```

2) Bitwise Operations

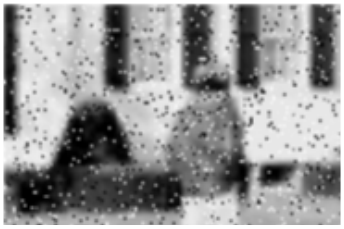
```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

spare = np.zeros((500,500),dtype='uint8')
rec = cv.rectangle(spare.copy(),(125,125),(250,250),255,thickness=-1)
```

```
cir = cv.circle(spare.copy(),(250,250),125,255,thickness=-1) # -1 fills the image

And = cv.bitwise_and(rec,cir)
Or = cv.bitwise_or(rec,cir)
Xor = cv.bitwise_xor(rec,cir)
Not = cv.bitwise_not(cir)
figure, plots = plt.subplots(2,3)

images = [rec,cir,And,Or,Xor,Not]
for i in range(2):
  for j in range(3):
    plots[i,j].imshow(images[0],'gray')
    plots[i,j].axis('off')
    images.pop(0)
```



3) Mean,Median,Gaussian filters

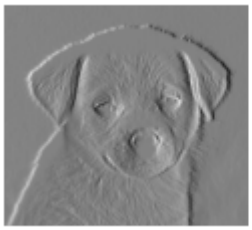
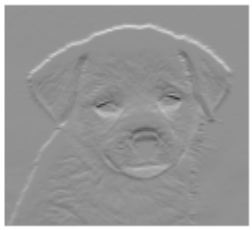
```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

image = cv.imread('gray_noise.png')

mean_gray = cv.blur(image,(3,3))
median_gray = cv.medianBlur(image,5)
gauss_gray = cv.GaussianBlur(image,(7,7),0)

figure,plots = plt.subplots(1,4,figsize=(15,15))
images = [image,mean_gray,median_gray,gauss_gray]
for i in range(4):
  plots[i].imshow(images[0])
  plots[i].axis('off')
  images.pop(0)
```

4) Canny and Sobel Filter

```
import cv2 as cv
import matplotlib.pyplot as plt

image = cv.imread('dog.jpg',0)
can_fil = cv.Canny(image,100,200,3)
sobelx = cv.Sobel(image,cv.CV_64F,1,0,ksize=5)
sobely = cv.Sobel(image,cv.CV_64F,0,1,ksize=5)
images = [image,can_fil,sobelx,sobely]
figure,plots = plt.subplots(1,4,figsize=(10,10))
for i in range(4):
  plots[i].imshow(images[0],'gray')
  plots[i].axis('off')
  images.pop(0)
```



5) Morphological operations

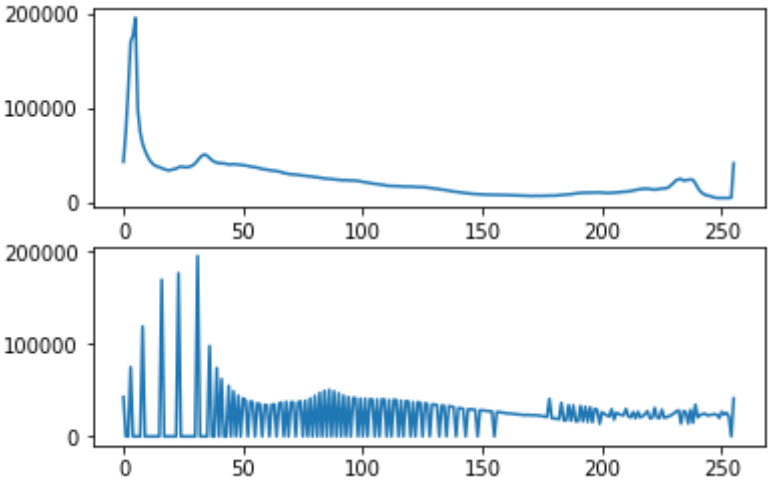
```
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np

image = cv.imread('sample.png')
kernel = np.ones((5,5),np.uint8)

erosion  = cv.erode(image,kernel,iterations=1) # Take a kernel of 1' and put it on the ima
dilate  = cv.dilate(image,kernel,iterations=1) # reverse of erosion
# open  = cv.morphologyEx(image,cv.MORPH_OPEN,kernel) # erosion followed by dilation
# close = cv.morphologyEx(image,cv.MORPH_CLOSE,kernel) # reverse of open
# grad  = cv.morphologyEx(image,cv.MORPH_GRADIENT,kernel) # diff between dilation and eros
# tophat  = cv.morphologyEx(image,cv.MORPH_TOPHAT,kernel) # diff between input image and c
# blackhat  = cv.morphologyEx(image,cv.MORPH_BLACKHAT,kernel) # diff between closing of im

images = [image,erosion,dilate]
figure,plots = plt.subplots(1,3)
```

```
for i in range(3):
  plots[i].imshow(images[0])
  plots[i].axis('off')
  images.pop(0)
```

6) Histogram Equilization

```
import cv2 as cv
import matplotlib.pyplot as plt

image = cv.imread('car_gray.jpg')

hist = cv.calcHist([image],[0],None,[256],[0,256])
img = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
eq_image = cv.equalizeHist(img)
new_hist = cv.calcHist([eq_image],[0],None,[256],[0,256])

# Plotting images
figure,plots = plt.subplots(2,1,figsize=(5,5))
images = [image,eq_image]
for i in range(2):
  plots[i].imshow(images[0],'gray')
  images.pop(0)
  plots[i].axis('off')

#Plotting Histograms
figure,plots = plt.subplots(2,1)
hists = [hist,new_hist]
for i in range(2):
  plots[i].plot(hists[0])
  hists.pop(0)


figure,plots = plt.subplots(2,1,figsize=(5,5))
images = [image,eq_image]
for i in range(2):
  plots[i].imshow(images[0],'gray')
  images.pop(0)
  plots[i].axis('off')
```

```
figure,plots = plt.subplots(2,1)
hists = [hist,new_hist]
for i in range(2):
  plots[i].plot(hists[0])
  hists.pop(0)
```

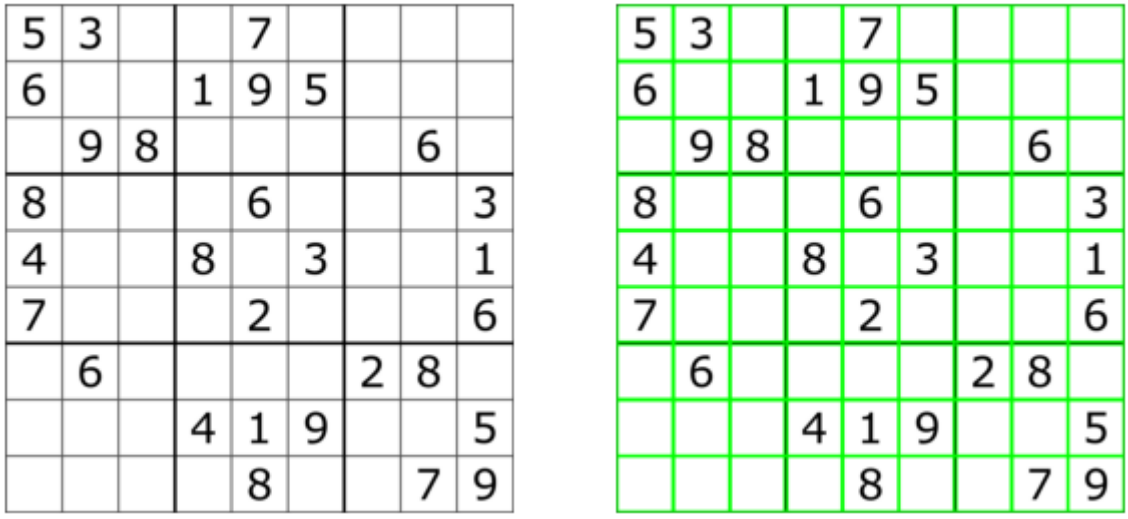


7) HPF and LPF Filters

```
import cv2
import matplotlib.pyplot as plt

image = cv2.imread('gray.png')
lpf = cv2.GaussianBlur(image,(7,7),0)
hpf = image-lpf
images = [image,lpf,hpf]
figure,plots = plt.subplots(1,3)
for i in range(3):
  plots[i].imshow(images[0])
  plots[i].axis('off')
  images.pop(0)
```

8) Hough Transformation

```
import cv2
import matplotlib.pyplot as plt

image = cv2.imread('sudoku.png')
a_image = image.copy()
#image = image.astype('float')
can_fil = cv2.Canny(image,100,200,3)
lines_list =[]
lines = cv2.HoughLinesP(
            can_fil, # Input edge image
            1, # Distance resolution in pixels
            np.pi/180, # Angle resolution in radians
            threshold=100, # Min number of votes for valid line
            minLineLength=5, # Min allowed length of line
            maxLineGap=10 # Max allowed gap between line for joining them
            )

for points in lines:
    x1,y1,x2,y2=points[0]
    cv2.line(image,(x1,y1),(x2,y2),(0,255,0),2)
    lines_list.append([(x1,y1),(x2,y2)])

images = [a_image,image]
figure,plots = plt.subplots(1,2,figsize=(10,10))
for i in range(2):
  plots[i].imshow(images[0])
  plots[i].axis('off')
  images.pop(0)
```

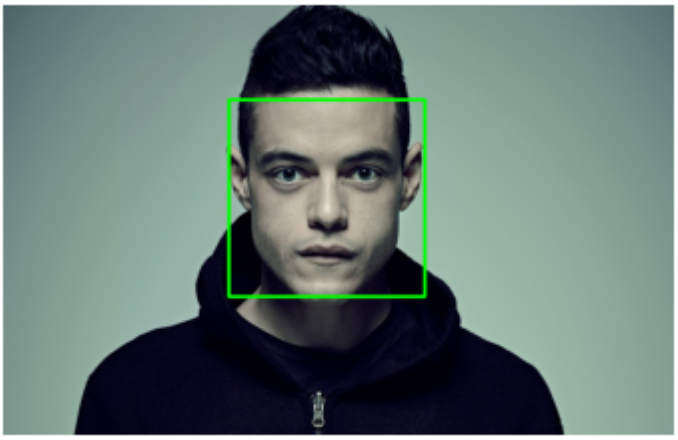


9) Face Detection using OpenCV

```
import cv2
```

```
import matplotlib.pyplot as plt

image = cv2.imread("sample_face.jpg")
gray = cv2.cvtColor(image,cv2.COLOR_RGB2GRAY)
detect = cv2.CascadeClassifier('haar_face_detect.xml')
faces = detect.detectMultiScale(gray,scaleFactor=1.1,minNeighbors=3)
x,y,w,h = faces[0]
cv2.rectangle(image,(x,y),(x+w,y+h),(0,255,0),thickness=5)

plt.imshow(cv2.cvtColor(image,cv2.COLOR_RGB2BGR))
plt.axis('off')
```

```
(-0.5, 1331.5, 849.5, -0.5)
```



Machine Learning

A* Algorithm

```
heuristic = {0:5,1:1,2:4,3:2,4:0}
cost = [[0,5,1,0,0],
        [5,0,0,0,1],
        [1,0,0,2,0],
        [0,0,2,0,2],
        [0,0,0,0,0],]
a = ['0']
c = [0]
def ways():
  while True:
    g = [c[i] + heuristic[int(a[i][-1])] for i in range(len(c))]
    o = a[g.index(min(g))]
    if o[-1] == '4':
      print(o)
      break
    k = int(o[-1])
    for j in range(k+1,5):
      if cost[k][j]!=0:
        a.append(str(o)+str(j))
```

```
        c.append(c[a.index(o)]+cost[k][j])
    a.remove(o)
    c.pop(g.index(min(g)))


ways()
```

```
024
```

E-mail Spam Detection

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score

df = pd.read_csv('spam_ham_dataset.csv')
X = df['text']
y = df['label_num']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

vectorizer = TfidfVectorizer(stop_words='english')
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)

model = MultinomialNB()
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
accuracy_score(y_pred,y_test)
```

```
0.9244288224956063
```

```
a = "Subject: fyi"


test = vectorizer.transform([a])
model.predict(test)
```

```
array([0])
```

Credit Card Fraud Detection

```
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.metrics import accuracy_score

df = pd.read_csv('creditcard.csv')
ex = df[df['Class']==0]
e = ex.sample(492,random_state=42)
a = df[df['Class']==1]
df = pd.concat([e,a],axis=0)

X = df.drop('Class',axis=1)
y = df['Class']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

model = KNeighborsClassifier(n_neighbors=5)
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
accuracy_score(y_pred,y_test)
```

```
0.916923076923077
```

Stock Prices Prediction

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM,Dense

df = pd.read_csv('tesla.csv')
X = df['Open']
y = df['Close']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

scaler  = MinMaxScaler()
X_train = scaler.fit_transform(np.array(X_train).reshape(-1,1))
X_test = scaler.transform(np.array(X_test).reshape(-1,1))

scaler_y = MinMaxScaler()
y_train = scaler_y.fit_transform(np.array(y_train).reshape(-1,1))
y_test = scaler_y.transform(np.array(y_test).reshape(-1,1)

model = Sequential()
model.add(LSTM(128,return_sequences=True,input_shape=(X_train.shape[1],1)))
model.add(LSTM(64, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))

model.compile(optimizer='adam',loss='mean_squared_error',metrics=['mse'])
model.fit(X_train,y_train,epochs=10)
```

```
pred = scaler.transform([[21]])
model.predict(pred)
scaler_y.inverse_transform([[0.0232087]])
```

```
scaler_y.inverse_transform([[0.0232087]])
```

```
array([[22.02735827]])
```