**1.**

Co-Occurrence Matrices:

Energy:

***2.***

*در این سوال، با استفاده از پایتون و کتابخانه های آن، تابع* KDE *را برای دیتا های زیر یافته و آن را رسم می کنیم.*

*برای یافتن تابع KDE از کرنل گوسی و از فرمول زیر استفاده می کنیم.*

***توضیحات کد محاسبه تابع* KDE *و رسم آن***

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

*برای حل این تمرین از کتابخانه* matplotlib *برای رسم نمودار* KDE، *از کتابخانه* numpy *برای محاسبات و از کتابخانه* math *برای محاسبات ریاضی(رادیکال و عدد پی) استفاده می کنیم.*

x = np.array([1, 1.02, 2, 2, 3, 4.2])  
y = np.array([1.02, 2.009, 3.001, 3, 3.01, 4.5])

*ابتدا مقادیر داده شده را در آرایه های* x *و* y *ذخیره می کنیم.*

def calculate\_KDE(arr):

*سپس تابع را می سازیم که یک آرایه را به عنوان ورودی دریافت می کند و در انتها* x *و* y *را به این تابع می دهیم تا مقادیر محاسبه شده توسط تابع* KDE *را دریافت کنیم.*

# Calculate Average  
average = 0  
for i in range(n):  
 average += arr[i]  
average /= 6

# Calculate Standard Deviation  
standard\_deviation = 0  
for i in range(n):  
 standard\_deviation += pow(arr[i] - average, 2)  
standard\_deviation /= 6  
standard\_deviation = math.sqrt(standard\_deviation)

*در ادامه میانگین و انحراف معیار را محاسبه می کنیم.*

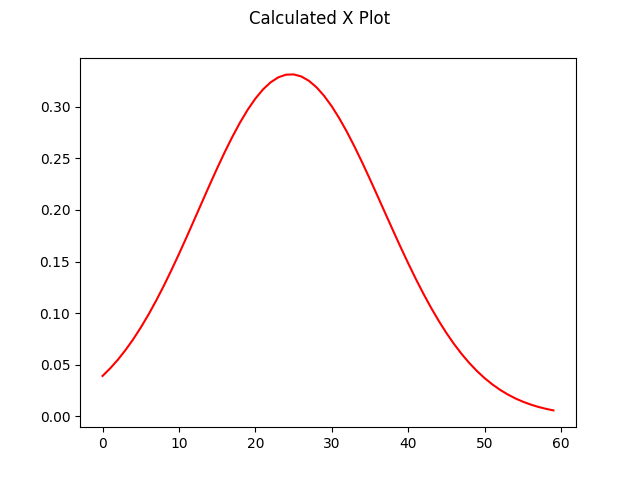
# Calculate Function  
calculated = np.array([])  
  
pre = 1 / (math.sqrt(2 \* math.pi) \* standard\_deviation \* n)  
  
for i in np.arange(-1, 5, 0.1):  
 exp\_val = 0  
 for k in range(n):  
 k\_average = 0  
 for j in range(k +1):  
 k\_average += arr[j]  
 k\_average /= (k + 1)  
  
 exp\_val += (np.exp((-1/2) \* pow((i - k\_average) / standard\_deviation, 2)))  
 value = pre \* exp\_val  
 calculated = np.append(calculated, value)

*سپس تابع را با استفاده از فرمولی که در ابتدا بیان شد، محاسبه می کنیم.*

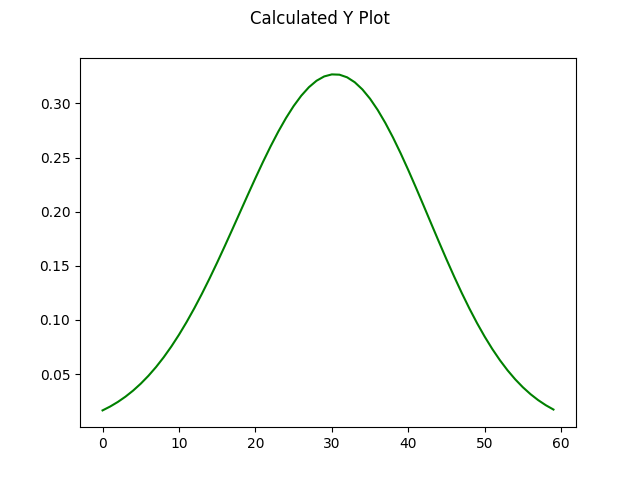
# Draw X Plot  
x\_plot\_value = calculate\_KDE(x)  
plot3 = plt.figure(3)  
plot3.suptitle('Calculated X Plot')  
plt.plot(x\_plot\_value, color="red")  
  
# Draw Y Plot  
y\_plot\_value = calculate\_KDE(y)  
plot4 = plt.figure(4)  
plot4.suptitle('Calculated Y Plot')  
plt.plot(y\_plot\_value, color="green")

*در نهایت تابع* KDE *را برای* x *و* y *رسم می کنیم.*

***نتیجه محاسبه تابع* KDE *و رسم آن***



*رسم تابع* KDE *برای مقادیر* x



*رسم تابع* KDE *برای مقادیر* y

***توضیحات کد رسم* KDE *با استفاده از کتابخانه* Seaborn**

*برای مقایسه توابع رسم شده، همین مقادیر را با استفاده از کتابخانه آماده* Seaborn *رسم می کنیم.*

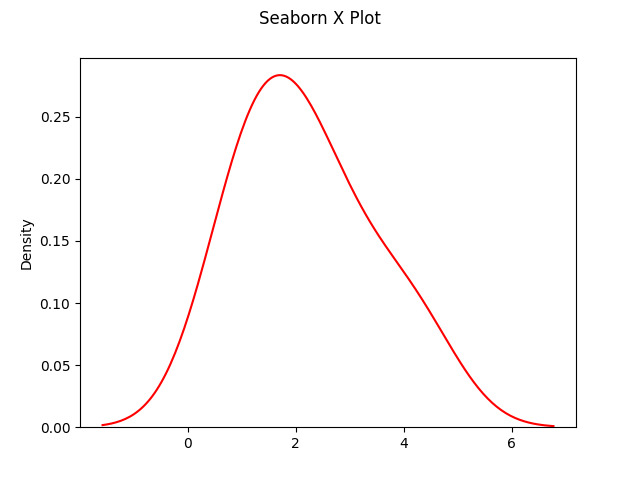
import seaborn as sns

*ابتدا این کتابخانه را* import *می کنیم.*

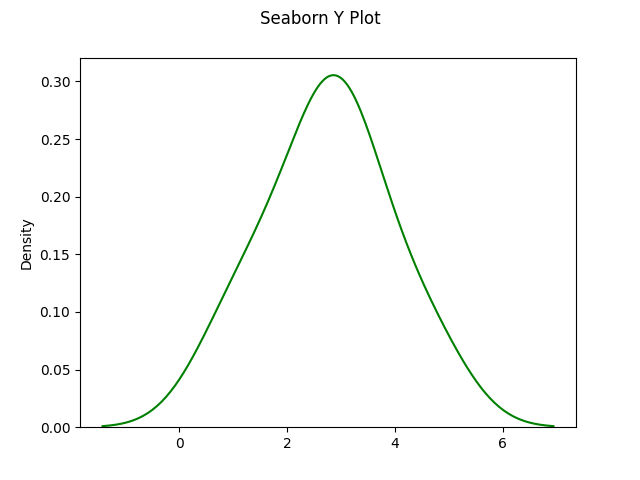
plot1 = plt.figure(1)  
plot1.suptitle('Seaborn X Plot')  
sns.kdeplot(x, color="red")  
  
plot2 = plt.figure(2)  
plot2.suptitle('Seaborn Y Plot')  
sns.kdeplot(y, color="green")

*سپس با استفاده از تابع* kdeplot *مقادیر* x *و* y *را رسم می کنیم.*

***نتیجه رسم* KDE *با استفاده از کتابخانه* Seaborn**



*رسم تابع* KDE *برای مقادیر* x *توسط کتابخانه* Seaborn



*رسم تابع* KDE *برای مقادیر* y *توسط کتابخانه* Seaborn

*پلات های رسم شده توسط تابع آماده، شباهت زیادی به پلات های محاسبه شده توسط فرمول دارد.*

***3.***

*در این سوال، با استفاده از پایتون و کتابخانه های آن، هیستوگرام تصویر خود را رسم کرده و با استفاده از آستانه محاسبه شده، تصویر را بازسازی می کنیم.*

***توضیحات کد محاسبه هیستوگرام***

from skimage import io, img\_as\_ubyte  
import numpy as np  
from matplotlib import pyplot as plt

*برای حل این تمرین از کتابخانه* skimage  *برای خواندن عکس، از* numpy *برای محاسبات و از کتابخانه* matplotlib *برای رسم نمودار هیستوگرام استفاده می کنیم.*

# Read Image as Colored and Gray  
image\_colored = img\_as\_ubyte(io.imread("KimiaMahdinejad.jpg", as\_gray=False))  
image\_gray = img\_as\_ubyte(io.imread("KimiaMahdinejad.jpg", as\_gray=True))

*ابتدا تصویر را به دو صورت رنگی و سیاه و سفید می خوانیم.*

scales\_colored = np.zeros(256)  
  
scales\_gray = np.zeros(256)

*سپس دو آرایه به تعداد 255 خانه برای ذخیره فراوانی هر کدام از اعداد می سازیم.*

# Calculate Colored Image Histogram  
for i in range(len(image\_colored)):  
 for j in range(len(image\_colored[0])):  
 for k in range(len(image\_colored[0][0])):  
 scales\_colored[image\_colored[i][j][k]] += 1

# Calculate Gray Image Histogram  
for i in range(len(image\_gray)):  
 for j in range(len(image\_gray[0])):  
 scales\_gray[image\_gray[i][j]] += 1

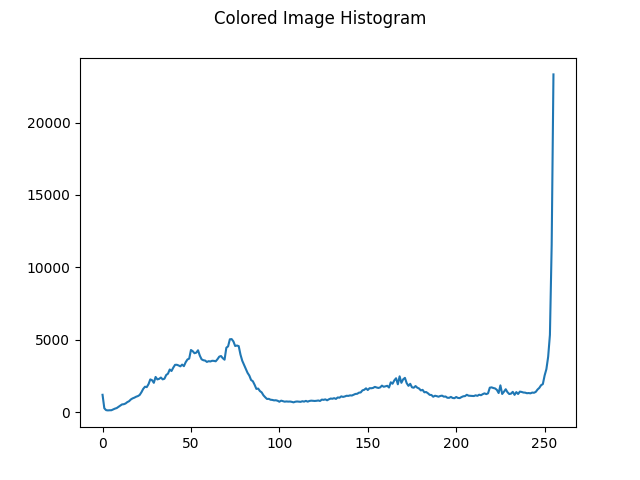
*با استفاده از حلقه های تو در تو، فراوانی هر کدام از مقادیر را در آرایه های ساخته شده ذخیره می کنیم.*

# Draw Colored Image Histogram  
plot1 = plt.figure(1)  
plot1.suptitle('Colored Image Histogram')  
plt.plot(scales\_colored)

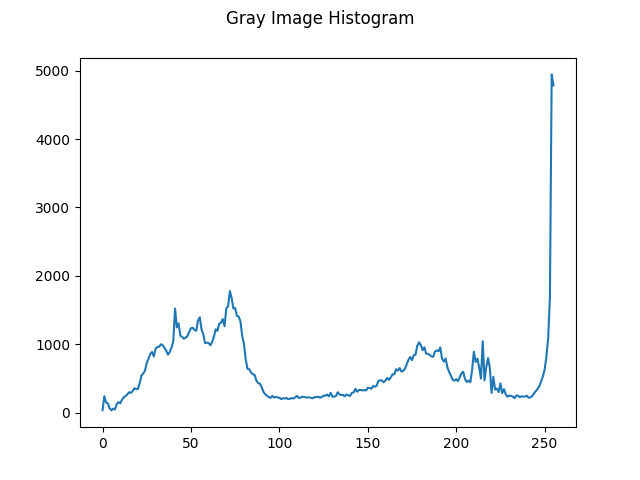
# Draw Gray Image Histogram  
plot2 = plt.figure(2)  
plot2.suptitle('Gray Image Histogram')  
plt.plot(scales\_gray)

*سپس مقادیر ذخیره شده در آرایه ها را که همان مقادیر فراوانی است را پلات می کنیم.*

***نتیجه محاسبه هیستوگرام***



*نمودار هیستوگرام رسم شده برای عکس رنگی*



*نمودار رسم شده برای عکس سیاه و سفید*

***توضیحات کد بازسازی تصویر با آستانه مناسب***

# Calculate Colored Image Threshold  
max\_val\_colored = 0  
for i in range(256):  
 if scales\_colored[i] > scales\_colored[max\_val\_colored]:  
 max\_val\_colored = i

# Calculate Gray Image Threshold  
max\_val\_gray = 0  
for i in range(256):  
 if scales\_gray[i] > scales\_gray[max\_val\_gray]:  
 max\_val\_gray = i

*با محاسبه مقدار ماکسیمم ذخیره شده در آرایه ها، آستانه هر کدام از تصاویر رنگی و سیاه و سفید را محاسبه می کنیم.*

# Rebuild Colored Image With Threshold  
new\_image\_colored = image\_colored  
  
for i in range(len(image\_colored)):  
 for j in range(len(image\_colored[0])):  
 for k in range(len(image\_colored[0][0])):  
 if new\_image\_colored[i][j][k] >= threshold\_colored:  
 new\_image\_colored[i][j][k] = 0

# Rebuild Gray Image With Threshold  
new\_image\_gray = image\_gray  
  
for i in range(len(image\_gray)):  
 for j in range(len(image\_gray[0])):  
 if new\_image\_gray[i][j] > threshold\_gray:  
 new\_image\_gray[i][j] = 0

*سپس در قسمت هایی که از آستانه مقدار بیشتری دارند را صفر می کنیم.*

# Save Colored Image With Threshold  
io.imsave("KimiaMahdinejad\_removed\_noise\_colored.jpg", new\_image\_colored.astype(np.uint8))

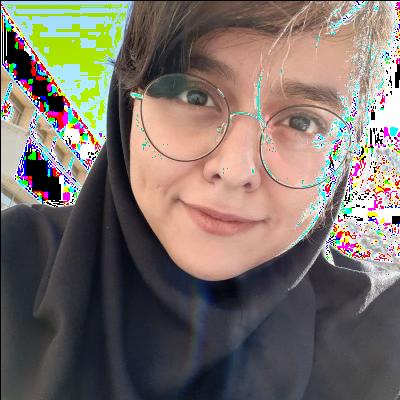
# Save Gray Image With Threshold  
io.imsave("KimiaMahdinejad\_removed\_noise\_gray.jpg", new\_image\_gray.astype(np.uint8))

*نتیجه را در قالب دو تصویر ذخیره می کنیم.*

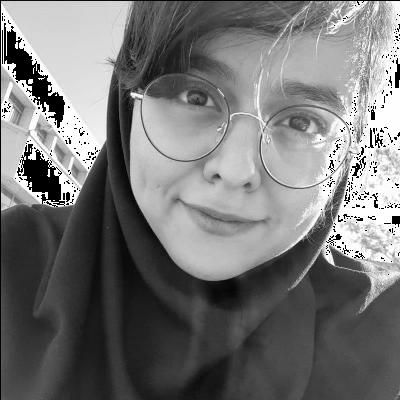
***نتیجه بازسازی تصویر با آستانه مناسب***



*تصویر اصلی*



*تصویر رنگی با آستانه 255*



*تصویر سیاه و سفید با آستانه 254*