Práctica 1: Manejo básico de imágenes y

visualización Iris Setosa

Ricardo Carrillo Sánchez, Juan Pablo Álvarez Loran, Alejandro Romo González,

Mariana Martínez Soto

*Facultad de Ingeniería, UNAM. Reconocimiento de Patrones*

[[1]](#footnote-2) ***Abstract—******Estos ejercicios ponen en práctica las funciones básicas para el manejo de imágenes, desde conocer los módulos necesarios para su control, los comandos hasta el uso y visualización del dataset Iris.***

# Objetivo

Al término de esta práctica, el alumno conocerá las distintas formas de desplegar una imagen en distintos formatos, así como algunas manipulaciones básicas, para el procesamiento de imágenes.

# Introducción

Una imagen es la representación en dos dimensiones, que contiene la información de color (intensidad) así como el diseño, esta información es guardada en los elementos más simples llamados pixeles.

Una imagen puede ser juzgada por la calidad que posee al momento de ser obtenida. La resolución de la fuente de imágenes es definida por medio de tres cantidades:

* Resolución Espacial: cantidad de pixeles en columnas (C) y renglones (R) para representar el espacio visual capturado en una imagen.
* Resolución Temporal: determina la cantidad de cuadros por una cantidad de tiempo.
* Resolución de bit o profundidad de bit: Define el número posible de valores de intensidad/color que un

píxel puede tener y se relaciona con la cuantización de la información de una imagen.

# Manipulación de Imágenes con Python

En la figura 1 se muestra la importación de los módulos necesarios para el manejo de imágenes en Python.

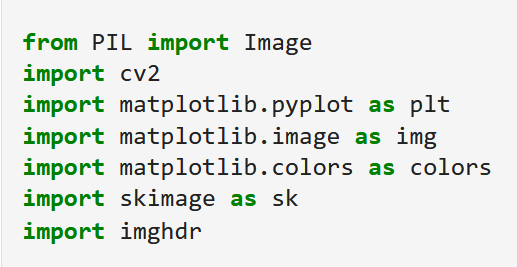


Figura 1 Importación de módulos

# Ejercicios parte a: imágenes

* 1. Desarrolla un script para leer y desplegar cada imagen con los paquetes de Matplotlib, OpenCV, Scikit-Image y PIL.

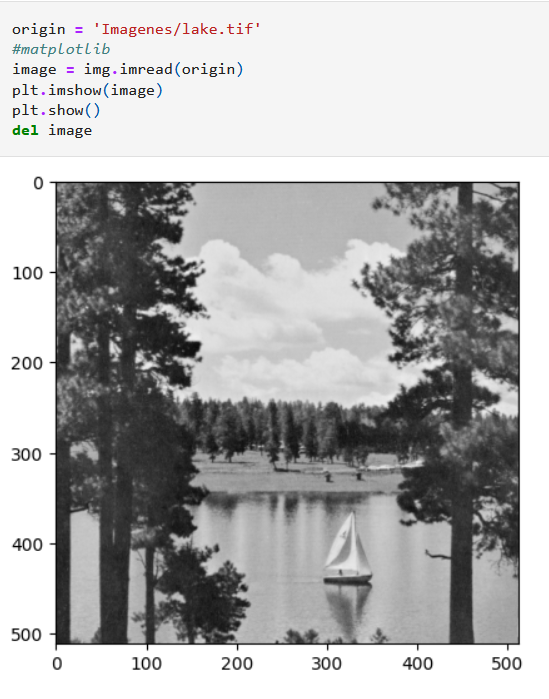
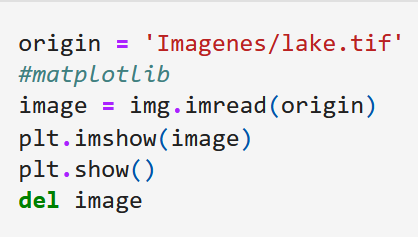


Figura 2. Despliegue de la imagen con Matplotlib

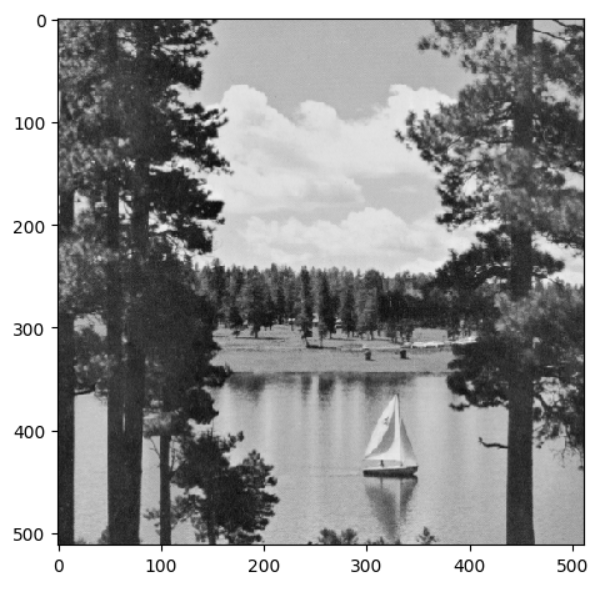


Figura 3. Imagen desplegada primero con PIL y después con el paquete skimage

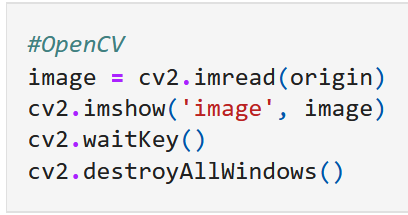


Figura 4 Despliegue de imagen con OpenCV

* 1. Imprimir el tipo de imagen, el tamaño y el tipo de dato

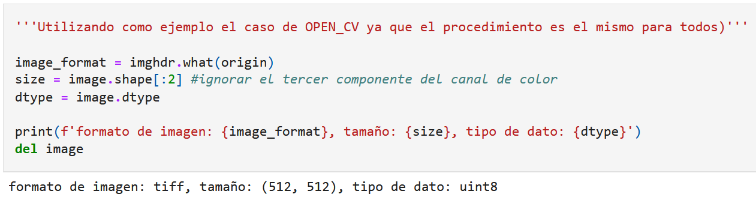


Figura 5 Comandos que muestran los atributos de tipo, tamaño y tipo de dato

La figura 5 muestran la impresión de los atributos de la variable image.

* 1. De las imágenes “lena\_color\_512.tif”, “peppers\_color.tif”. Desarrolla un script con OpenCV y Scikit- Image para cambiar el espacio de color de:
* RGB a Escala de grises
* RGB a YUV
* RGB a HSV
* Despliega la paleta de colores de RGB por separado
* Despliega la paleta de colores HSV por separado

import numpy as np

def use\_opencv(filename):

dict = {}

image = cv2.imread(filename)

dict['ORIGINAL'] = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

dict['RGB2GRAY'] = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

dict['RGB2YUV'] = cv2.cvtColor(image, cv2.COLOR\_BGR2YUV)

dict['RGB2HSV'] = cv2.cvtColor(image, cv2.COLOR\_BGR2HSV)

del image

return dict

def use\_skimage(filename):

dict = {}

image = sk.io.imread(filename, plugin='pil')[:,:,:3]

dict['ORIGINAL'] = image

dict['RGB2GRAY'] = sk.color.rgb2gray(image)

dict['RGB2YUV'] = sk.color.rgb2yuv(image)

dict['RGB2HSV'] = sk.color.rgb2hsv(image)

del image

return dict

def show(dict):

fig, axes = plt.subplots(1, 4, figsize=(8, 4))

ax = axes.ravel()

ax[0].imshow(dict['ORIGINAL'])

ax[0].set\_title("Original")

ax[1].imshow(dict['RGB2GRAY'], cmap=plt.cm.gray)

ax[1].set\_title("Grayscale")

ax[2].imshow(dict['RGB2YUV'])

ax[2].set\_title("YUV")

ax[3].imshow(dict['RGB2HSV'], cmap=plt.cm.hsv)

ax[3].set\_title("HSV")

fig.tight\_layout()

plt.show()

origin1 = 'Imagenes/lena\_color\_512.tif'

origin2 = 'Imagenes/peppers\_color.tif'

a = use\_opencv(origin1)

b = use\_skimage(origin1)

c = use\_opencv(origin2)

d = use\_skimage(origin2)

show(a)

show(b)

show(c)

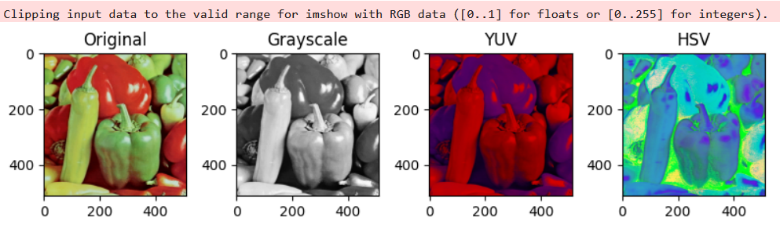
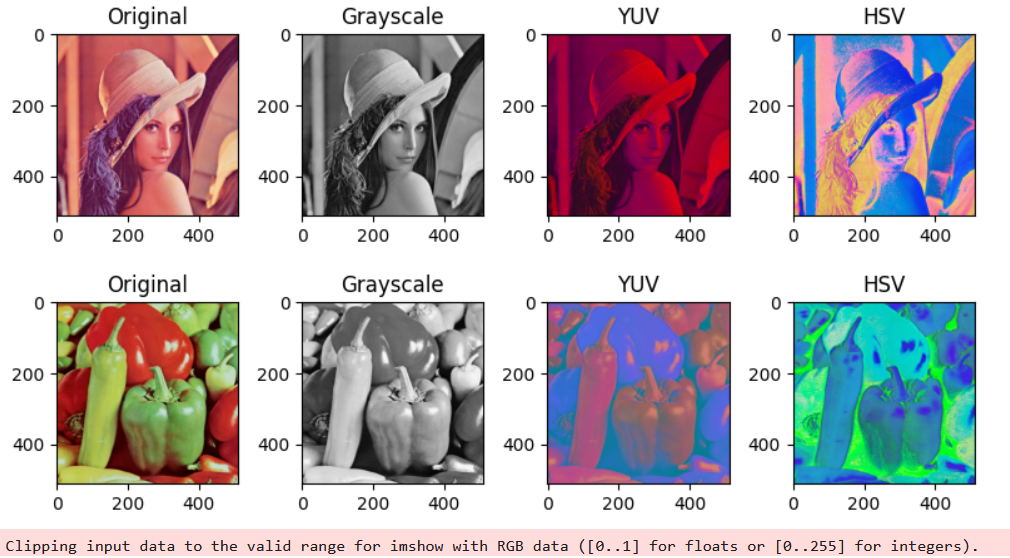
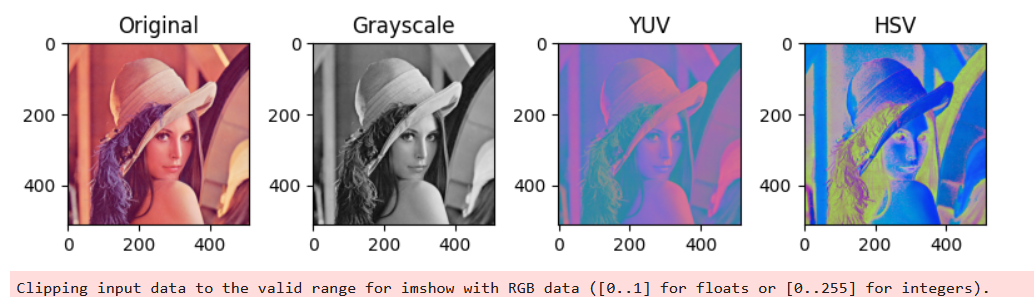


Figura 6. Cambios de color

La figura anterior (6) despliega la transformación de dos imágenes RGB (primera columna de izquierda a derecha) a otros espacios de color (subsecuentes columnas).

Las dos filas de en medio representan la paleta de colores RGB que se descompone. Y la última, la HSV.

* 1. Despliega la paleta de colores de RGB por separado, ver figura siguiente, la barra de la derecha con valores es la paleta de colores.
  2. De una imagen que usted escoja, dejarla en escala de grises y procure que sea igual en renglones y en columnas. Programe una función que realice decimación de una imagen, reduciendola a la mitad de su tamaño original. Y promediando en grupos de 4 pixeles. Pruebe con su imagen.

1. from statistics import mean
2. from skimage.util import view\_as\_blocks

5. '''El programa redimenciona en menores tamanos con potencias de base 2. Se puede modificar cambiando el factor'''
6. def decimate(image, factor):
7. resized\_image = []
8. new\_size = len(image)//factor
10. blocks = view\_as\_blocks(image,(factor,factor)) #tomar una submatriz de manera consecutiva (horizontal y vertical)
12. for a in blocks.reshape(-1, factor\*\*2):
13. b = a.reshape(-1) # "aplanar" el vector a analizar
14. c = np.around(mean(b)) #promedio de los pixels de la region seleccionada
15. resized\_image.append(c)
17. resized\_image=np.array(resized\_image)
18. return resized\_image.reshape(new\_size,new\_size)
20. image = cv2.imread('Imagenes/apa.jpg')
21. image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)
22. resized = decimate(image,4)
23. print(resized.shape)

26. fig, axes = plt.subplots(1, 2, figsize=(8, 4))
27. x = axes.ravel()
28. x[0].imshow(image, cmap=plt.cm.gray)
29. x[0].set\_title("ORIGINAL")
30. x[1].imshow(resized, cmap = plt.cm.gray)
31. x[1].set\_title("RESIZED")
32. plt.show()

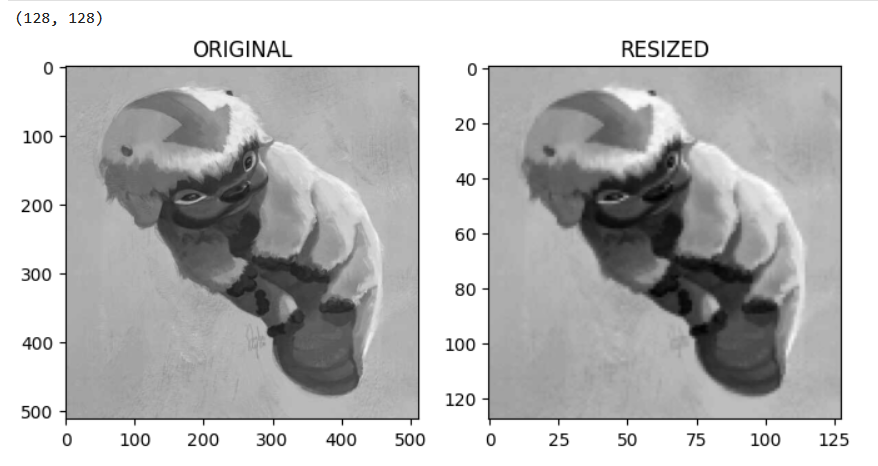


Figura 7 A la izquierda la imagen original, a la derecha la imagen con decimación aplicada

4.6 Convierte la imagen peppers\_color.tif a escala de grises,

* Recortela de manera que solo quede uno de los pimientos verdes en ese recorte
* Guárdela en formato .jpg.

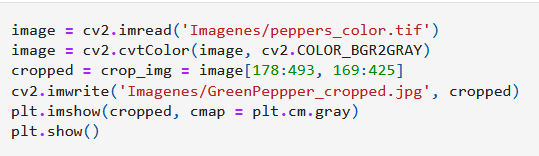


Figura 8. Uso de un rango de pixeles para recortar la imagen, y cambio de formato

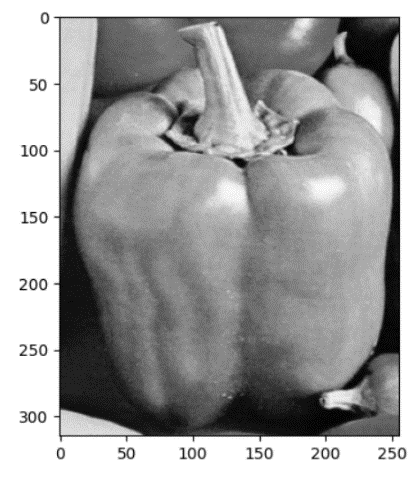


Figura 9. Imagen recortada

En la figura 9 se puede observar la imagen anterior de los pimientos ahora recortada y en otro formato.

* 1. Un formato de imágenes sin ningún tipo de codificación se conoce como formato crudo (RAW). De la imagen “rosa800x600.raw” lea y despliegue la imagen. Tome en cuenta que esta imagen maneja la precisión de integer8 y el tamaño es de 600x800 pixeles.

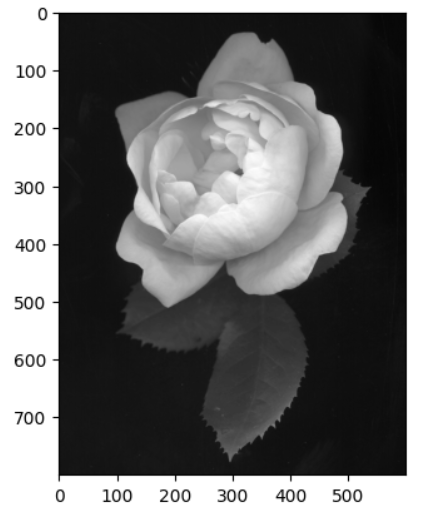


Figura 10 Imagen sin codificación

# Ejercicios parte b: iris setosa

5.13 Cree gráficas de dispersión usando pairplot de seaborn y muestre con distintos colores las tres especies en las gráficas de dispersión.



5.2 Imprima las llaves y el número de filas y de columnas.

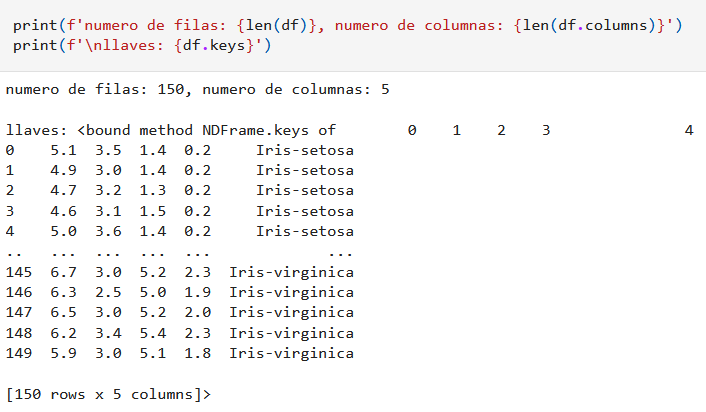


Figura 11. Atributos del conjunto de datos

En la figura 11 desplegamos el número total de datos.

5.3 Obtenga el número de muestras faltantes o Nan.

#NO APLICA PARA ESTE EJERCICIO YA QUE NO HAY COLUMNAS CON NaN

5.4 Cree un arreglo 2-D de tamaño 5x5 con unos en la diagonal y ceros en el resto. Convierta el arreglo NumPy a una matriz dispersa de ScyPy en formato CRS. Nota: una matriz se considera dispersa cuando el porcentaje de ceros es mayor a 0.5.

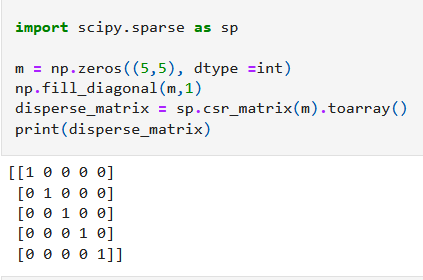


Figura 12 Uso de scipy para crear una matriz dispersa

5.5 Muestre estadísticas básicas como percentil, media, mínimo, máximo y desviación estándar de los datos. Use describe para ello. Imprima sólo la media y la desviación estándar.

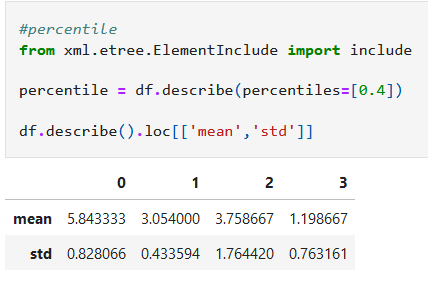


Figura 13 Uso de describe para conocer estadísticas

5.6 Obtenga el número de muestras para cada clase.

El atributo counts() nos permite contabilizar la cantidad de datos que existe de las clases que existen en el conjunto de datos.

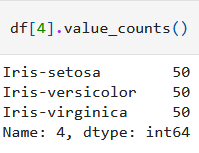


Figura 14 Uso de value\_counts()

5.7 Añada un encabezado a los datos usando los nombres en iris.names y repita el ejercicio anterior.

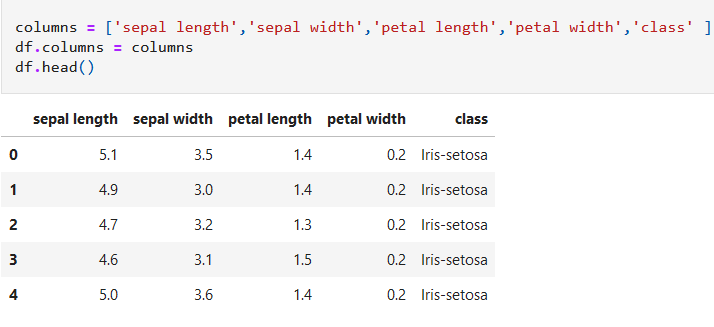


Figura 15 Encabezado añadido

5.8 Imprima las diez primeras filas y las dos primeras columnas del data frame usando los índices de las

columnas.

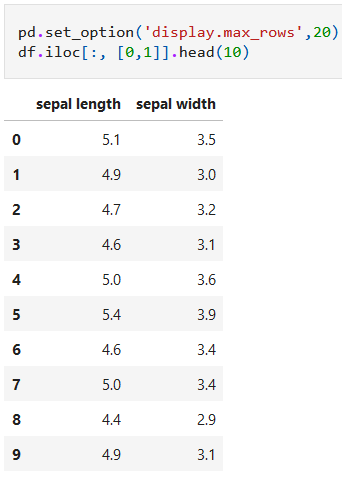


Figura 16 Muestra limitada del conjunto completo

5.9. Cree una gráfica de barras que muestre la media, mínimo y máximo de todos los datos.

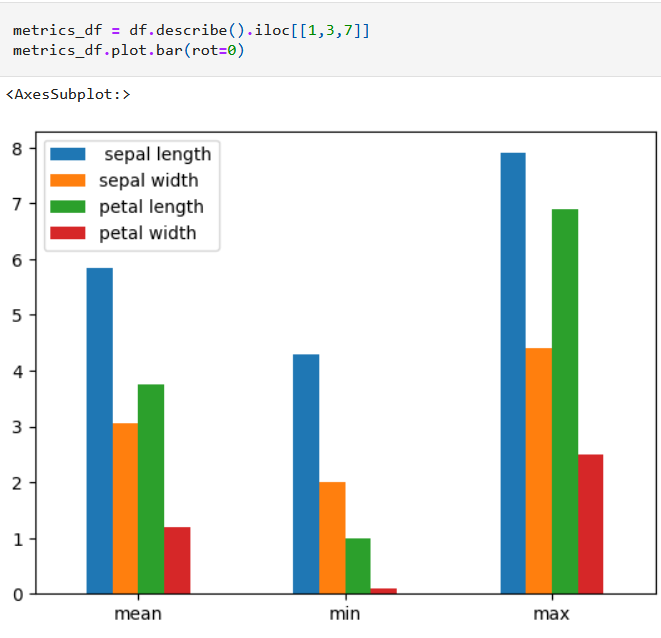


Figura 17 Gráfica de barras de conjunto

5.10. Muestre la frecuencia de las tres especies como una gráfica de pastel.

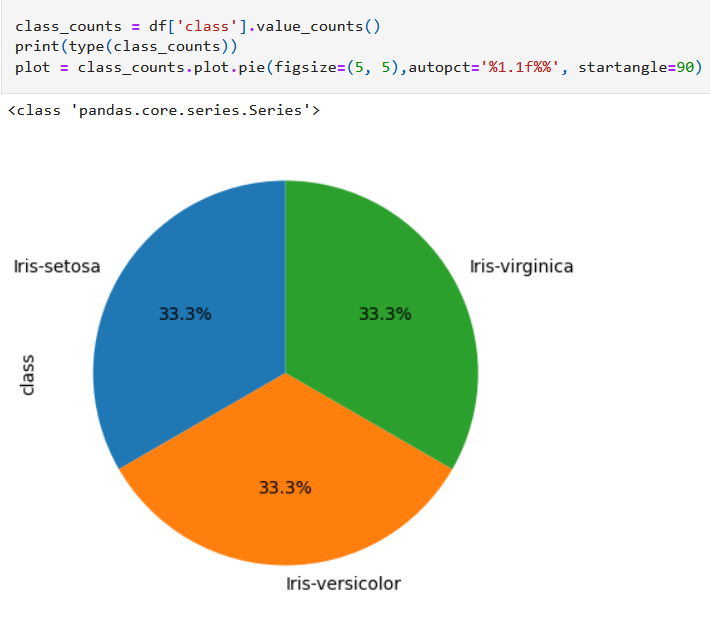


Figura 18 Gráfica de pastel del conjunto de datos

5.11. Cree una gráfica que muestre la relación entre la longitud y ancho del sépalo de las tres especies conjuntamente.

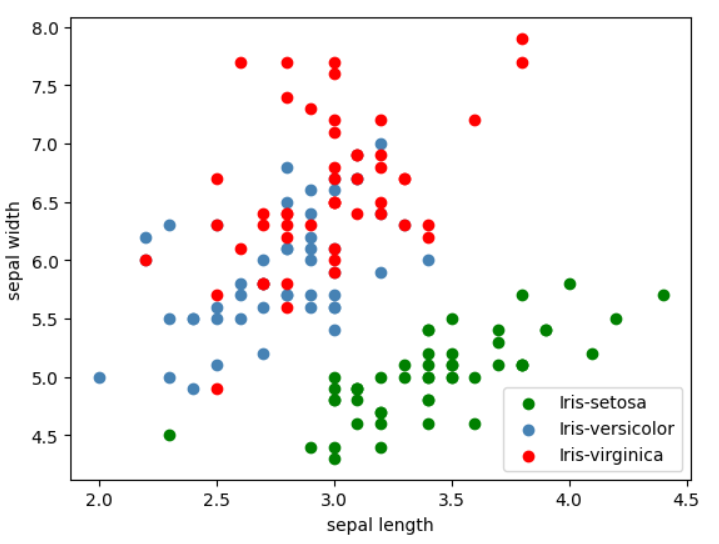


Figura 19 Gráfica que muestra la varianza entre las distancias de cada clase

5.12. Obtenga los histogramas de las variables SepalLength, SepalWidth, PetalLength y PetalWidth.

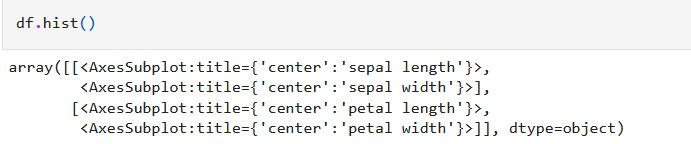


Figura 20 La instrucción .hist()

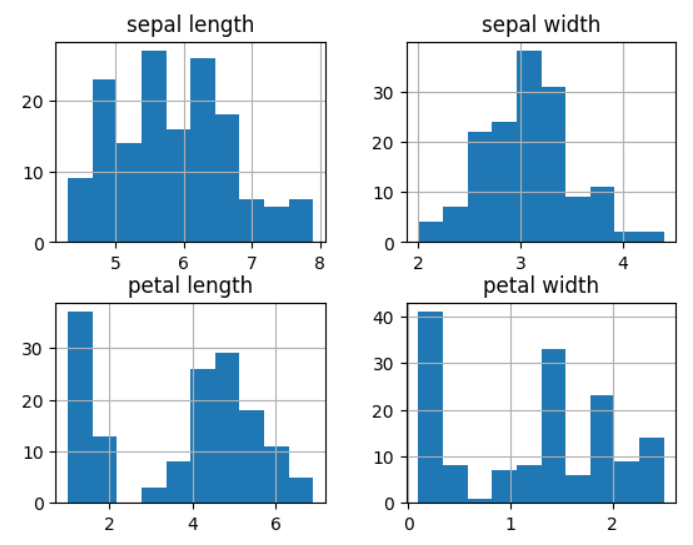
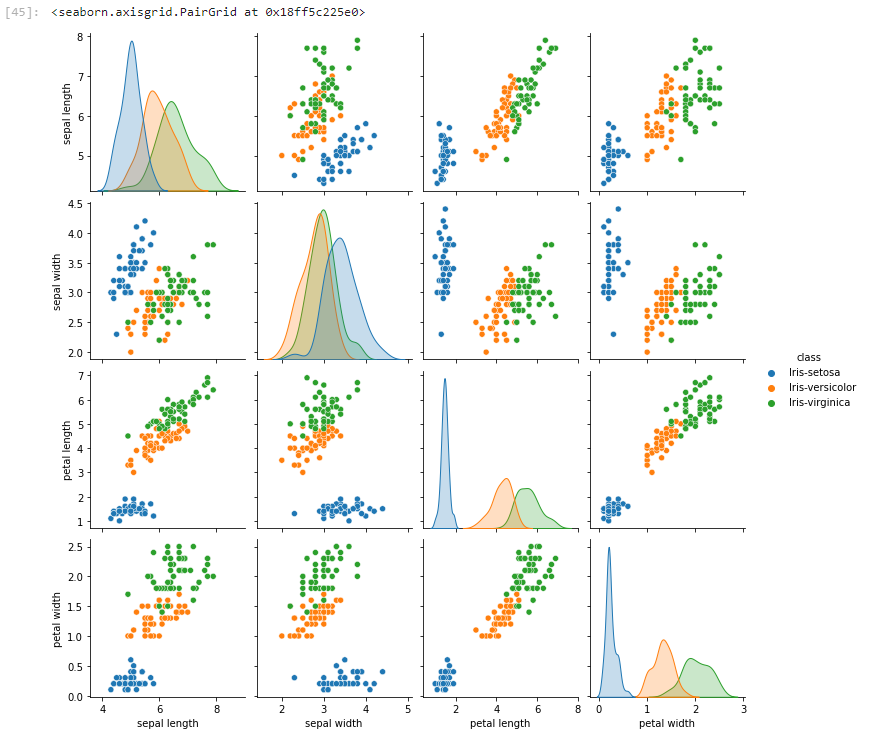
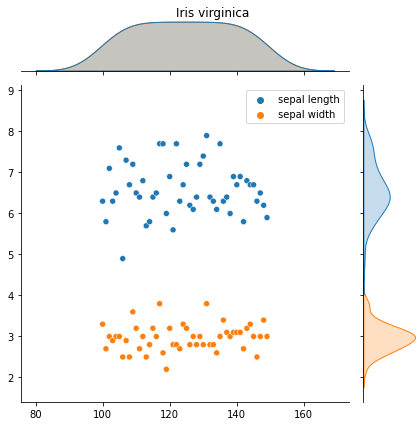
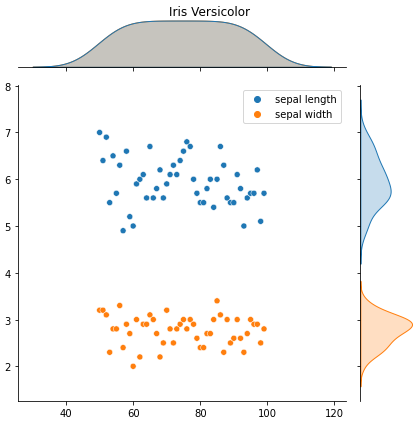
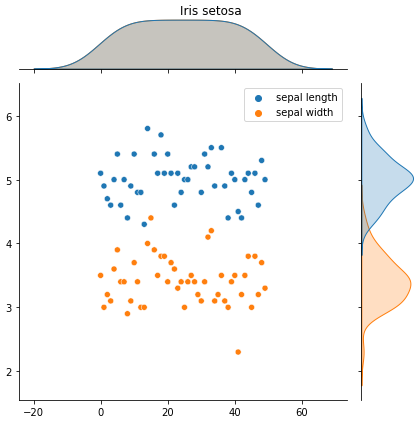
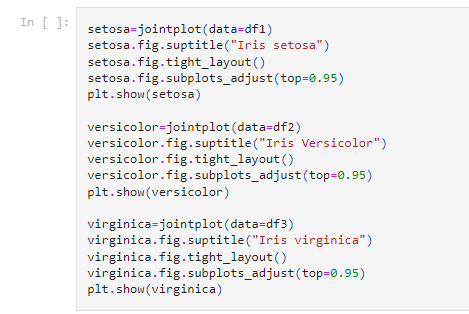


Figura 21 Histogramas de cuatro atributos del conjunto

5.13. Cree gráficas de dispersión usando pairplot de seaborn y muestre con distintos colores las tres especies en las gráficas de dispersión.

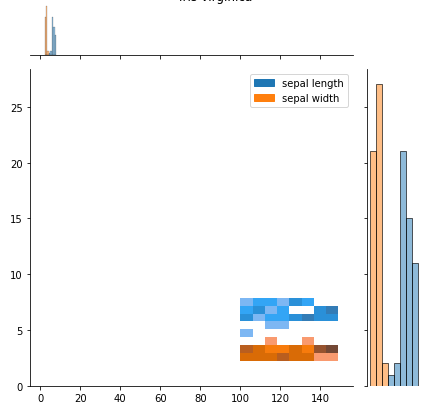
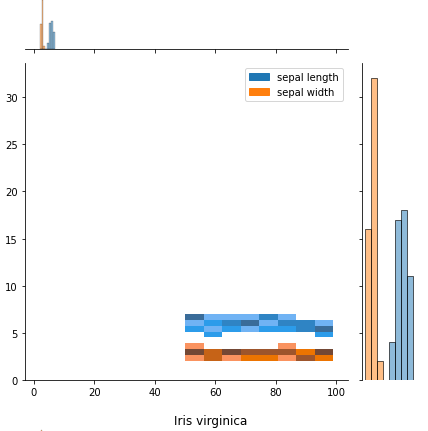
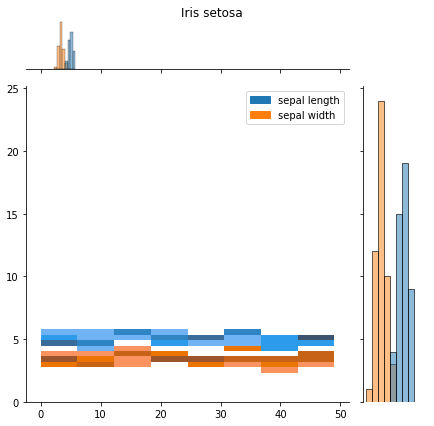


5.14. Cree una gráfica usando joinplot de seaborn para mostrar la dispersión entre la longitud y ancho del sépalo y las distribuciones de estas dos variables.



5.15. Repita el ejercicio anterior, pero esta vez usando joinplot con kind="hex".

Al utilizar la opción hex, jointplot tiene un bug que genera un error por lo que se utilizó la opcion hist en su lugar.



# Conclusiones

Los atributos que rodean a una imagen son de gran importancia, y son tomadas en cuenta para su procesamiento. Los diferentes paquetes que Python nos permite utilizar de forma fácil y cómoda permiten crear una base sólida para su futuro análisis. Como se leyó en el artículo previamente, sabemos que también es de importancia trabajar con conjuntos de datos establecidos para ciertos puntos del reconocimiento de patrones.

# Enlace al repositorio

1. https://github.com/ricardo-casa/ReconocimientoPatrones/tree/main/Practicas/Practica1

# Referencias

1. S. Dey, Hands-On Image Processing with Python: Expert techniques for advanced image analysis and efective interpretaction of image data, Birmingham, UK: Packt Publishing, 2018.

1. \*Reconocimiento de patrones. Facultad de Ingeniería, 2022. [↑](#footnote-ref-2)