Práctica 1: Visualización Iris Setosa

Serapio Hernández Alexis Arturo
Facultad de Ingeniería
UNAM
Ciudad de México, México
alexis.serapio@ingenieria.unam.

García López Erik
Facultad de Ingeniería
UNAM
Ciudad de México, México
erikpumas999@gmail.com

Reyes Herrera Rogelio
Facultad de Ingeniería
UNAM
Ciudad de México, México
masterfive5of@gmail.com

***Abstract*—This article is about the making of a practical document at the Faculty of Engineering at UNAM, it is about the Visualization of the "Iris Setosa" that its a kind of flower that can be found in the nature but it is because its petal characteristics that it can be computable and with data managing determine via pictures if a flower its an "Iris Setosa" or not. Here we managed the info that the professor has proportioned us, deploying and explaining it.**

Keywords—Basic Images Processing, Python, Pattern, Pattern Recognition, flowers, Iris Setosa.

I. Introducción

El manejo básico de imágenes en el ámbito de la informática, algunas veces puede enfocarse en la extracción de información significativa de imágenes digitales. Este proceso involucra varios pasos desde el preprocesamiento de la imagen hasta la clasificación de patrones y objetos.

El reconocimiento de patrones, en particular, es una técnica utilizada para identificar entonces estructuras y regularidades dentro de los datos, que pueden ser imágenes, señales o datos numéricos.[1]

Algunos conceptos generales para abordar la práctica son:

Obtención de información en imágenes: Cuando se trabaja con imágenes, el primer paso es la adquisición de datos. Las imágenes son generalmente matrices de pixeles donde cada pixel tiene un valor de intensidad o valores RGB. Para esto se necesita que a las imágenes se les aplique un preprocesamiento y una extracción de características

Clasificación de Objetos: La clasificación de objetos es el proceso de asignar una etiqueta o clase a un objeto dentro de una imagen basada en sus características. Esto se puede lograr con los modelos de clasificación que ya habíamos visto anteriormente y junto con estos modelos procesarlos en un Entrenamiento y prueba ya que el modelo se entrena con un conjunto de datos etiquetados para evaluar su precisión.

Iris Setosa

La clasificación de la Iris Setosa se hace a partir de características morfológicas que de igual manera podemos encontrar dentro de las imágenes. Si queremos aplicar la clasificación de imágenes de flores para identificar si una flor es la especie Iris Setosa, para eso haríamos lo siguiente:

a) **Recopilación de Imágenes:** Se necesitan imágenes de distintas especies de iris para que el modelo conozca y clasifique de manera correcta a la Iris Setosa y a sus otras variantes

b) **Extracción de Imágenes:** A partir de las imágenes, se podrían extraer caracteristicas morfologicas especificas como la forma y el tamaño de los petalos, los patrones de color y otras caracteristicas visuales.

c) **Entrenamiento de un modelo:** Con estas características, se entrena un modelo de clasificación para aprender a distinguir entre las diferentes especies de Iris.

d) **Clasificación:** Al presentarle al modelo una nueva imagen de una flor, este puede predecir si la flor es una Iris Setosa basándose en las características que ha aprendido.

Es por esto que el análisis de imágenes y el reconocimiento de patrones son fundamentales en tareas como la clasificación de objetos, en el caso de la Iris Setosa, podemos aplicar estas técnicas de procesamiento de imágenes y clasificación para identificar esta especie en imágenes.

a) Características que puede encontrar un modelo entrenado.

II. DESARROLLO PRÁCTICA 1

3.1.- Cargue los datos iris en un data frame (pandas) e imprima la descripción de los datos (columnas y renglones), tipo y las 10 primeras filas de los datos. Fuente de datos: https://archive.ics.uci.edu/ml/datasets/Iris.

Para iniciar, como se nos comenta en la actividad necesitamos utilizar pandas, por lo que se realiza la instalación de dicha librería.

Posteriormente, importamos las librerías y descargamos los datos desde las URL.
Una vez con los datos, tenemos datos en un DataFrame de Pandas e imprimimos la descripción de los datos, así como las 10 primeras filas.

Al imprimir tenemos la siguiente información:

```
Tipos de datos:
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
class            object
dtype: object

Las 10 primeras filas:
   sepal_length  sepal_width  petal_length  petal_width        class
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
5           5.4          3.9           1.7          0.4  Iris-setosa
6           4.6          3.4           1.4          0.3  Iris-setosa
7           5.0          3.4           1.5          0.2  Iris-setosa
8           4.4          2.9           1.4          0.2  Iris-setosa
9           4.9          3.1           1.5          0.1  Iris-setosa
```

De igual manera, desplegamos la información contenida en el zip proporcionado por la profesora:

```
Descripción de los datos:
Número de filas: 150
Número de columnas: 5

Tipos de datos:
0    float64
1    float64
2    float64
3    float64
4     object
dtype: object

Las 10 primeras filas:
     0    1    2    3            4
0  5.1  3.5  1.4  0.2  Iris-setosa
1  4.9  3.0  1.4  0.2  Iris-setosa
2  4.7  3.2  1.3  0.2  Iris-setosa
3  4.6  3.1  1.5  0.2  Iris-setosa
4  5.0  3.6  1.4  0.2  Iris-setosa
5  5.4  3.9  1.7  0.4  Iris-setosa
6  4.6  3.4  1.4  0.3  Iris-setosa
7  5.0  3.4  1.5  0.2  Iris-setosa
8  4.4  2.9  1.4  0.2  Iris-setosa
9  4.9  3.1  1.5  0.1  Iris-setosa
```

```
9  4.9  3.1  1.5  0.1  Iris-setosa
Descripción de los datos:
Número de filas: 151
Número de columnas: 5

Tipos de datos:
0    object
1    object
2    object
3    object
4    object
dtype: object

Las 10 primeras filas:
             0            1            2            3            4
0  SepalLength  SepalWidth  PetalLength  PetalWidth        Class
1          NaN          3.5          1.4          0.2  Iris-setosa
2          4.9          3.0          1.4          0.2  Iris-setosa
3          4.7          3.2          1.3          0.2  Iris-setosa
4          4.6          3.1          1.5          0.2  Iris-setosa
5          5.0          3.6          1.4          0.2  Iris-setosa
6          5.4          3.9          1.7          0.4  Iris-setosa
7          4.6          3.4          1.4          0.3  Iris-setosa
8          5.0          3.4          1.5          0.2  Iris-setosa
9          4.4          2.9          1.4          0.2  Iris-setosa
```

Para observar el código del programa, ver en el apéndice I.

3.2.- Imprima las llaves y el número de filas y de las columnas

Del mismo sitio, utilizando las funciones df.keys() y tolist() obtenemos el nombre de las llaves.
De igual manera, con df.shape[í] obtenemos el número de filas y columnas, cada arreglo 0 y 1 obteniendo uno de estos datos.

```
Llaves (nombres de las columnas):
['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']

Número de filas: 150
Número de columnas: 5
```

```
Llaves (nombres de las columnas):
[0, 1, 2, 3, 4]

Número de filas: 150
Número de columnas: 5

Llaves (nombres de las columnas):
[0, 1, 2, 3, 4]

Número de filas: 151
Número de columnas: 5
```

Para los archivos de la profesora pudimos obtener las llaves de las columnas y su número, dado que en el código no agregamos su nombre solo nos aparecen sus llaves.

Para observar el código del programa, ver en el apéndice II.

3.3.- Obtenga el número de muestras faltantes o Nan.

En este punto, para obtener el número de muestras faltantes ya sea por columna o en Total, lo podemos hacer con df.isnull() y sum() para obtener el número total de muestras.

```
Número de muestras faltantes o NaN por columna:
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
class           0
dtype: int64

Número total de muestras faltantes o NaN:
0
```

Posteriormente, haciendo uso del archivo .zip proporcionado por la profesora, desplegamos el mismo ejercicio para observar si se tiene una cantidad diferente de muestras faltantes o NaN.

```
      0    1    2    3           4
0  5.1  3.5  1.4  0.2  Iris-setosa
1  4.9  3.0  1.4  0.2  Iris-setosa
2  4.7  3.2  1.3  0.2  Iris-setosa
3  4.6  3.1  1.5  0.2  Iris-setosa
4  5.0  3.6  1.4  0.2  Iris-setosa

Número de muestras faltantes o NaN por columna:
0    3
1    0
2    0
3    1
4    0
dtype: int64

Número total de muestras faltantes o NaN:
4
            0            1            2            3            4
0  SepalLength  SepalWidth  PetalLength  PetalWidth        Class
1          NaN          3.5          1.4          0.2  Iris-setosa
2          4.9          3.0          1.4          0.2  Iris-setosa
3          4.7          3.2          1.3          0.2  Iris-setosa
4          4.6          3.1          1.5          0.2  Iris-setosa

Número de muestras faltantes o NaN por columna:
0    1
1    1
2    0
3    1
4    0
dtype: int64

Número total de muestras faltantes o NaN:
3
```

Dado que encontramos una gran cantidad de número de filas erroneas, decidimos continuar la práctica con un número

3.4.- 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

```
Matriz NumPy original:
[[1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]]

Matriz dispersa de SciPy (formato CRS):
  (0, 0)        1.0
  (1, 1)        1.0
  (2, 2)        1.0
  (3, 3)        1.0
  (4, 4)        1.0

Forma de la matriz dispersa: (5, 5)
Número de elementos no cero: 5
Densidad de la matriz: 0.2

¿La matriz es considerada dispersa? Sí
```

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

Recordando lo que significan las estadísticas básicas:

- **Percentil**: Valor que indica el porcentaje de datos por debajo de él en una distribución.
- **Media**: El promedio aritmético de un conjunto de valores.
- **Mínimo**: El valor más bajo en un conjunto de datos.
- **Máximo**: El valor más alto en un conjunto de datos.
- **Desviación estándar**: Medida de la dispersión o variabilidad de los datos respecto a la media.

```
Estadísticas básicas:
       sepal_length  sepal_width  petal_length  petal_width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000

Media y desviación estándar:
      sepal_length  sepal_width  petal_length  petal_width
mean      5.843333     3.054000      3.758667     1.198667
std       0.828066     0.433594      1.764420     0.763161
```

Con .describe se muestran las estadísticas básicas del dataframe; después con .loc elegimos solo la fila mean y std para mostrar únicamente la media y la desviación.

3.6.- Obtenga el número de muestras para cada clase

```
Número de muestras por clase:
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

Para obtener esta salida, seleccionamos la columna class y hacemos uso de la función count sobre dicha columna del dataframe.

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

Este paso ya se había hecho previamente al cargar los datos al dataframe, como se puede observar en el ejercicio 3.1.

```
column_names = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']
```

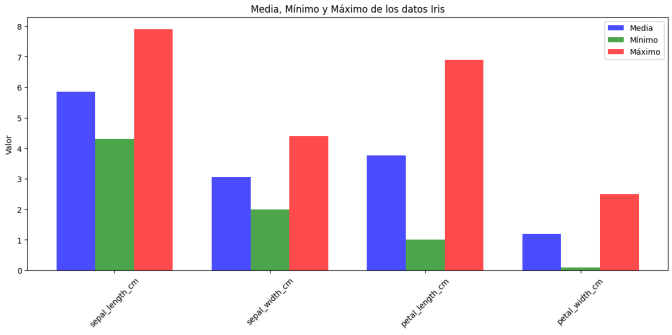
3.8 Imprima las diez primeras filas y las dos primeras columnas del data frame usando los índices de las columnas.

```
Las diez primeras filas y las dos primeras columnas:
   sepal_length_cm  sepal_width_cm
0              5.1             3.5
1              4.9             3.0
2              4.7             3.2
3              4.6             3.1
4              5.0             3.6
5              5.4             3.9
6              4.6             3.4
7              5.0             3.4
8              4.4             2.9
9              4.9             3.1

Nombres de las columnas seleccionadas:
['sepal_length_cm', 'sepal_width_cm']
```

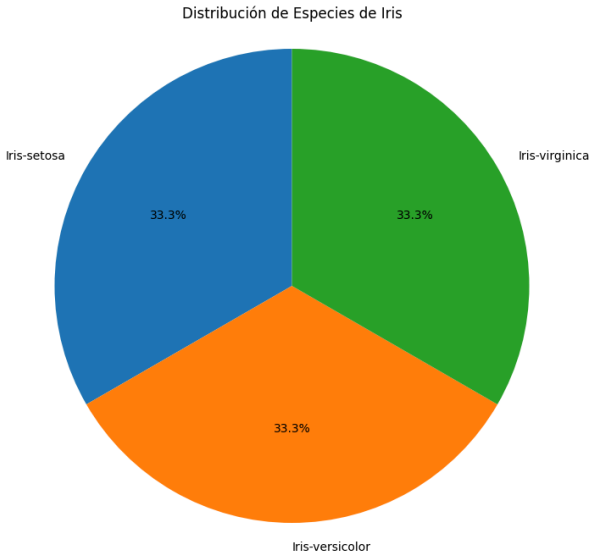
En esta ocasión en vez de usar loc se usa iloc, ya que este método va enfocado a cuando quieres seleccionar datos basándote en la posición (índice) de filas y columnas.

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



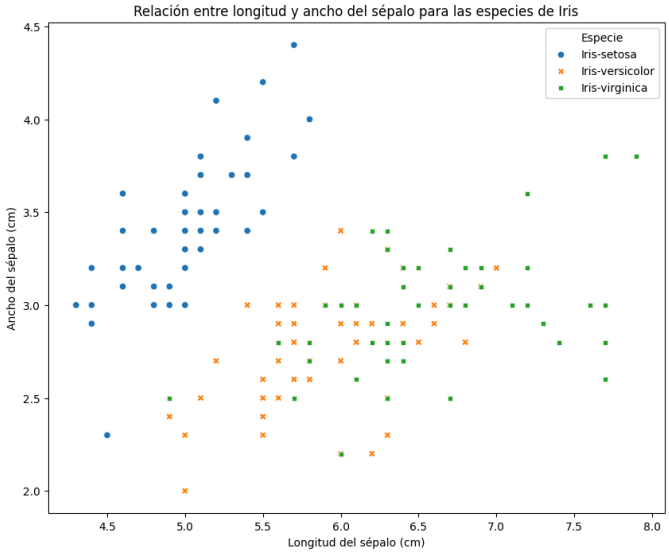
Se crea la gráfica y se asigna color azul para el valor medio, el verde para el mínimo y el rojo para el máximo, refiriéndonos a estos valores en cada una de las columnas numéricas del dataframe. Nos ayudamos de la librería numpy para realizar las gráficas.

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



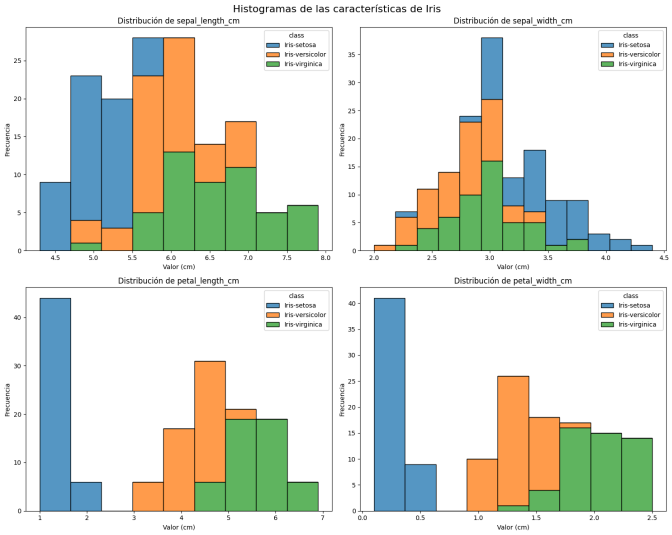
En esta parte, usando el método count que habíamos usado previamente, creamos una forma visual de representarlo con `plt.pie`.

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



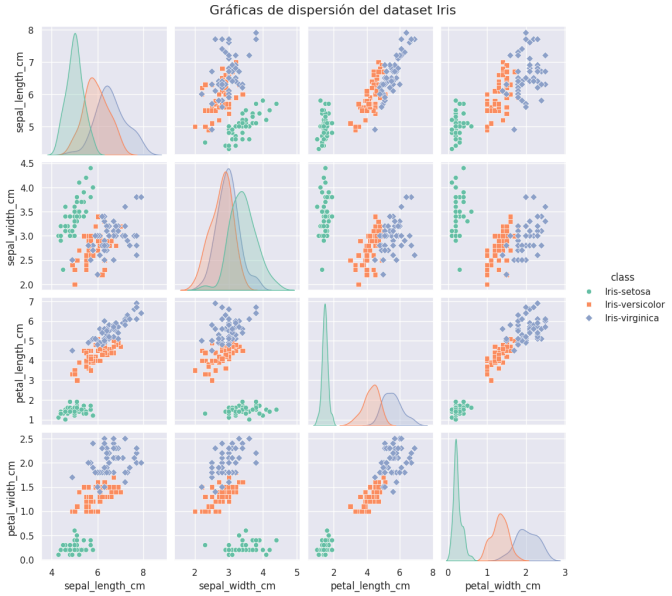
Se propuso un diagrama de dispersión muy comúnmente usado en tareas de ML. Aquí se aprecia como los datos se separan en especies dependiendo del largo y ancho del sépalo, aunque no estén separados del todo ya que no estamos tomando en cuenta el pétalo, se aprecia como se discriminan los datos por zona.

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



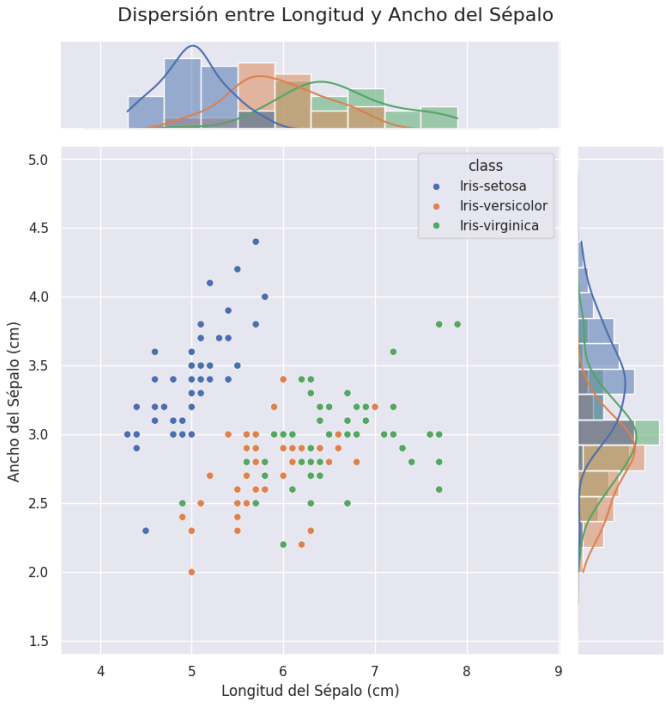
Cada subplot mostrará el histograma de una de las variables mencionadas. La función sns.histplot de Seaborn se encarga de crear los histogramas, apilando las barras según la clase de la flor (hue='class'). Finalmente, se ajusta el diseño de la figura para evitar solapamientos y se muestra la figura completa con los histogramas.

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



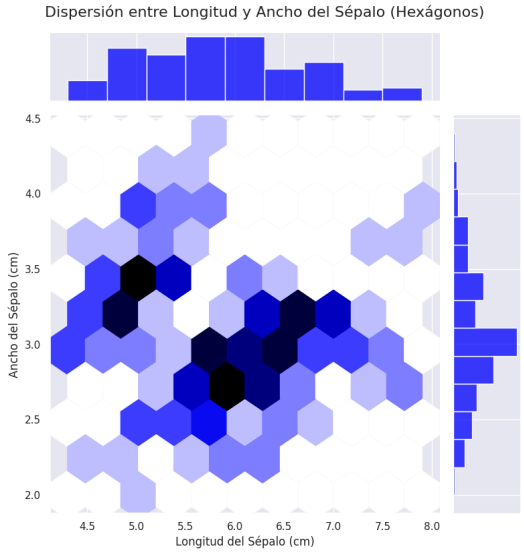
Con esta función se forma una cuadricula de 4x4 ya que son cuatro características y cada una se compara con las características restantes; en la diagonal principal tenemos curvas y no puntos debido a que en la diagonal principal no compara una característica contra otra, sino que muestra la distribución de cada característica por sí misma.

3.14 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.



Este punto se hizo como se menciona con joinplot, que es una función de Seaborn que crea un gráfico conjunto para mostrar la relación entre dos variables junto con sus distribuciones; es cómo conjuntar los diagramas de dispersión y los histogramas para un análisis más profundo de resultados, o quizá más visual.

3.14 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.



Con kind se especifica el tipo de gráfico principal que se utilizará, en este caso cambiamos los puntos por hexágonos, otras opciones son regresión y densidad de kernel.

Para observar el código del programa, ver en el apéndice VIII.

III. CONCLUSIONES

En esta práctica, hemos explorado el análisis de imágenes y el reconocimiento de patrones para la clasificación de flores, centrándonos en la identificación de la Iris setosa. A lo largo del proceso, utilizamos técnicas de procesamiento de imágenes para extraer características relevantes de las flores, como la forma, el tamaño y los patrones de color de los pétalos y sépalos. Estas características fueron esenciales para entrenar un modelo de clasificación capaz de diferenciar entre las distintas especies de Iris.

Además, esta práctica subraya la importancia del preprocesamiento de las imágenes y la selección adecuada de características, ya que estos factores influyen directamente en la precisión del modelo de clasificación.

Es por esto que la aplicación del análisis de imágenes y el reconocimiento de patrones en la clasificación de flores no solo proporciona una metodología robusta para la identificación de especies, sino que también abre la puerta a aplicaciones más avanzadas en la botánica y otras disciplinas científicas. La experiencia adquirida en este ejercicio refuerza la comprensión de cómo las técnicas de visión por computadora y el aprendizaje automático pueden integrarse para resolver problemas complejos en el mundo real.

REFERENCES

[1] Kaggle, "Machine Learning with Iris dataset," Kaggle.com, 7 de agosto de 2017. [En línea]. Disponible: https://www.kaggle.com/code/jchen2186/machine-learning-with-iris-dataset. [Accedido: 2 de septiembre de 2024].

[2] . Waskom, "seaborn: statistical data visualization," seaborn: statistical data visualization — seaborn 0.13.2 documentation, 2 de septiembre de 2024. [En línea]. Disponible: https://seaborn.pydata.org/ . [Accedido: 2 de septiembre de 2024].

[3] PANDAS, "What is Python for Data Analysis," Pydata.org, 10 de abril de 2024. [En línea]. Disponible: https://pandas.pydata.org /. [Accedido: 2 de septiembre de 2024].

[4] R. A. Fisher, "Iris," UCI Machine Learning Repository, 1988. [En línea]. Disponible: https://doi.org/10.24432/C56C76 . [Accedido: 2 de septiembre de 2024].

IV. Apendice

En este apéndice, se anexará el contenido de los códigos de los ejercicios desarrollados en la presente.

I. *Cargue los datos iris en un data frame (pandas) e imprima la descripción de los datos (columnas y renglones), tipo y las 10 primeras filas de los datos. Fuente de datos: https://archive.ics.uci.edu/ml/datasets/Iris.*

```
import pandas as pd
import requests
from io import StringIO

# URL de los datos Iris
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

# Descargar los datos
response = requests.get(url)
data = StringIO(response.text)

# Cargar los datos en un DataFrame de pandas
column_names = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']
df = pd.read_csv(data, names=column_names)

# Imprimir la descripción de los datos
print("Descripción de los datos:")
print(f"Número de filas: {df.shape[0]}")
print(f"Número de columnas: {df.shape[1]}")
print("\nTipos de datos:")
print(df.dtypes)

# Imprimir las 10 primeras filas
print("\nLas 10 primeras filas:")
print(df.head(10))
```

```
import pandas as pd
import requests
from io import StringIO
# Leer el archivo .data usando pandas
data_file = '/content/iris.data'
data = pd.read_csv(data_file, header=None)

# Leer el archivo .data usando pandas
data_file2 = '/content/irisH.data'
data2 = pd.read_csv(data_file2, header=None)

# Imprimir la descripción de los datos
print("Descripción de los datos:")
print(f"Número de filas: {data.shape[0]}")
print(f"Número de columnas: {data.shape[1]}")
print("\nTipos de datos:")
print(data.dtypes)

# Imprimir las 10 primeras filas
print("\nLas 10 primeras filas:")
print(data.head(10))

# Imprimir la descripción de los datos
print("Descripción de los datos:")
print(f"Número de filas: {data2.shape[0]}")
print(f"Número de columnas: {data2.shape[1]}")
print("\nTipos de datos:")
print(data2.dtypes)

# Imprimir las 10 primeras filas
print("\nLas 10 primeras filas:")
print(data2.head(10))
```

II. *Imprima las llaves y el número de filas y de columnas.*

```
print("\nLlaves (nombres de las columnas):")
print(df.keys().tolist())

print(f"\nNúmero de filas: {df.shape[0]}")
print(f"Número de columnas: {df.shape[1]}")

# Imprimir las llaves y el número de filas y columnas
```

```
print("\nLlaves (nombres de las columnas):")
print(data.keys().tolist())

print(f"\nNúmero de filas: {data.shape[0]}")
print(f"Número de columnas: {data.shape[1]}")

# Imprimir las llaves y el número de filas y columnas
print("\nLlaves (nombres de las columnas):")
print(data2.keys().tolist())

print(f"\nNúmero de filas: {data2.shape[0]}")
print(f"Número de columnas: {data2.shape[1]}")
```

III. Obtenga el número de muestras faltantes o Nan.

```
# Obtener el número de muestras faltantes o NaN
print("\nNúmero de muestras faltantes o NaN por columna:")
print(df.isnull().sum())

print("\nNúmero total de muestras faltantes o NaN:")
print(df.isnull().sum().sum())
import pandas as pd
# Cargar los datos en un DataFrame de pandas

# Leer el archivo .data usando pandas
data_file = '/content/iris.data'
data = pd.read_csv(data_file, header=None)

# Mostrar las primeras filas del conjunto de datos
print(data.head())
```

```
print("\nNúmero de muestras faltantes o NaN por columna:")
print(data.isnull().sum())

print("\nNúmero total de muestras faltantes o NaN:")
print(data.isnull().sum().sum())

data_file2 = '/content/irisH.data'
data2 = pd.read_csv(data_file2, header=None)

# Mostrar las primeras filas del conjunto de datos
print(data2.head())

print("\nNúmero de muestras faltantes o NaN por columna:")
print(data2.isnull().sum())

print("\nNúmero total de muestras faltantes o NaN:")
print(data2.isnull().sum().sum())
```

IV. 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.

```
import numpy as np
from scipy import sparse

# Crear un arreglo 2-D de tamaño 5x5 con unos en la diagonal y ceros en el resto
matriz = np.eye(5)

print("Matriz NumPy original:")
print(matriz)

# Convertir la matriz NumPy a una matriz dispersa de SciPy en formato CRS
matriz_dispersa = sparse.csr_matrix(matriz)
```

```
print("\nMatriz dispersa de SciPy (formato CRS):")
print(matriz_dispersa)

# Información adicional
print("\nForma de la matriz dispersa:", matriz_dispersa.shape)
print("Número de elementos no cero:", matriz_dispersa.nnz)
print("Densidad de la matriz:", matriz_dispersa.nnz / (matriz_dispersa.shape[0] * matriz_dispersa.shape[1]))

# Verificar si la matriz es considerada dispersa
es_dispersa = matriz_dispersa.nnz / (matriz_dispersa.shape[0] * matriz_dispersa.shape[1]) < 0.5
print("\n¿La matriz es considerada dispersa?", "Sí" if es_dispersa else "No")
```

V. *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.*

```
# Mostrar estadísticas básicas
print("\nEstadísticas básicas:")
descripcion = df.describe()
print(descripcion)

# Imprimir sólo la media y la desviación estándar
print("\nMedia y desviación estándar:")
print(descripcion.loc[['mean', 'std']])
```

VI. *Obtenga el número de muestras para cada clase.*

```
# Obtener el número de muestras para cada clase
print("\nNúmero de muestras por clase:")
print(df['class'].value_counts())
```

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

```
# Definir los nombres de las columnas según iris.names
column_names = [
                    'sepal_length_cm', 'sepal_width_cm',
                    'petal_length_cm', 'petal_width_cm',
    'class'
]
```

```
# Reemplazar los nombres de las columnas en el DataFrame
df.columns = column_names

# Verificar los nuevos nombres de las columnas
print("Nuevos nombres de las columnas:")
print(df.columns)

# Repetir el conteo de muestras por clase
print("\nNúmero de muestras por clase (con nuevos encabezados):")
print(df['class'].value_counts())
```

VIII. *Imprima las diez primeras filas y las dos primeras columnas del data frame usando los índices de las columnas.*

```
# Imprimir las diez primeras filas y las dos primeras columnas usando índices
print("Las diez primeras filas y las dos primeras columnas:")
print(df.iloc[:10, :2])

# Opcional: Mostrar los nombres de las columnas seleccionadas
print("\nNombres de las columnas seleccionadas:")
print(df.columns[:2].tolist())
```

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

```
import matplotlib.pyplot as plt

import numpy as np



# Seleccionar solo las columnas numéricas

numeric_columns = df.select_dtypes(include=[np.number]).columns



# Calcular media, mínimo y máximo

mean_values = df[numeric_columns].mean()
```

```
min_values = df[numeric_columns].min()

max_values = df[numeric_columns].max()



# Crear la gráfica de barras

fig, ax = plt.subplots(figsize=(12, 6))



# Posiciones de las barras

x = np.arange(len(numeric_columns))

width = 0.25



# Crear las barras

ax.bar(x - width, mean_values, width, label='Media', color='b', alpha=0.7)

ax.bar(x, min_values, width, label='Mínimo', color='g', alpha=0.7)

ax.bar(x + width, max_values, width, label='Máximo', color='r', alpha=0.7)



# Personalizar la gráfica

ax.set_ylabel('Valor')

ax.set_title('Media, Mínimo y Máximo de los datos Iris')

ax.set_xticks(x)

ax.set_xticklabels(numeric_columns, rotation=45)

ax.legend()



# Ajustar el diseño y mostrar la gráfica

plt.tight_layout()

plt.show()
```

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

```
import matplotlib.pyplot as plt



# Contar la frecuencia de cada especie

species_counts = df['class'].value_counts()



# Crear la gráfica de pastel

plt.figure(figsize=(10, 8))

plt.pie(species_counts.values, labels=species_counts.index, autopct='%1.1f%%', startangle=90)

plt.title('Distribución de Especies de Iris')



# Asegurar que el círculo se vea como un círculo

plt.axis('equal')



# Mostrar la gráfica

plt.show()



# Opcional: Mostrar los valores numéricos

print("\nDistribución numérica de especies:")

print(species_counts)
```

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

```
import matplotlib.pyplot as plt

import seaborn as sns



# Crear la figura y los ejes

plt.figure(figsize=(10, 8))



# Usar seaborn para crear un gráfico de dispersión

sns.scatterplot(data=df,
x='sepal_length_cm',
y='sepal_width_cm',     hue='class',
style='class')



# Personalizar la gráfica

plt.title('Relación entre longitud y ancho del sépalo para las especies de Iris')

plt.xlabel('Longitud    del    sépalo (cm)')

plt.ylabel('Ancho del sépalo (cm)')



# Añadir una leyenda

plt.legend(title='Especie')



# Mostrar la gráfica

plt.show()
```

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

```
import matplotlib.pyplot as plt

import seaborn as sns



# Definir las variables para los histogramas

variables    =    ['sepal_length_cm', 'sepal_width_cm', 'petal_length_cm', 'petal_width_cm']



# Crear una figura con subplots

fig,   axs   =   plt.subplots(2,   2, figsize=(15, 12))

fig.suptitle('Histogramas   de   las características      de      Iris', fontsize=16)



# Aplanar la matriz de ejes para facilitar la iteración

axs = axs.flatten()



# Crear un histograma para cada variable

for i, var in enumerate(variables):

      sns.histplot(data=df,   x=var, hue='class',       multiple="stack", ax=axs[i])

     axs[i].set_title(f'Distribución de {var}')

   axs[i].set_xlabel('Valor (cm)')

   axs[i].set_ylabel('Frecuencia')



# Ajustar el diseño

plt.tight_layout()



# Mostrar la figura

plt.show()
```

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

```
import seaborn as sns

import matplotlib.pyplot as plt



# Crear gráficas de dispersión usando pairplot

sns.pairplot(df,        hue="class", markers=["o",       "s",       "D"], palette="Set2")



# Añadir título a la figura

plt.suptitle("Gráficas de dispersión del dataset Iris",    y=1.02, fontsize=16)



# Mostrar las gráficas

plt.show()
```

XIV. *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.*

```
import seaborn as sns

import matplotlib.pyplot as plt



# Crear el joinplot con los nombres de columnas correctos

g = sns.jointplot(

   data=df,

        x="sepal_length_cm",      # Reemplazar con el nombre correcto de la columna

        y="sepal_width_cm",       # Reemplazar con el nombre correcto de la columna

   hue="class",

   kind="scatter",

   height=8

)



# Personalizar los histogramas marginales

g.plot_marginals(sns.histplot, kde=True)



# Añadir título y etiquetas

g.fig.suptitle("Dispersión entre Longitud y Ancho del Sépalo", fontsize=16, y=1.03)

g.set_axis_labels("Longitud del Sépalo (cm)", "Ancho del Sépalo (cm)")



# Mostrar la gráfica

plt.show()
```

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

```
import seaborn as sns

import matplotlib.pyplot as plt



# Crear el joinplot con kind="hex" para mostrar la dispersión con hexágonos

g = sns.jointplot(

   data=df,

        x="sepal_length_cm",      # Reemplazar con el nombre correcto de la columna
```

```
        y="sepal_width_cm",         # Reemplazar con el nombre correcto de la columna
    kind="hex",
    height=8,
    color="blue"
)


# Añadir título y etiquetas
g.fig.suptitle("Dispersión entre Longitud y Ancho del Sépalo (Hexágonos)", fontsize=16, y=1.03)
g.set_axis_labels("Longitud del Sépalo (cm)", "Ancho del Sépalo (cm)")


# Mostrar la gráfica
plt.show()
```