Manejo Visualización Iris Setosa

Julio Pérez
Estudiante, Facultad de Ingeniería
Ciudad Universitaria, Ciudad de México, México

Mauricio Bautista
Estudiante, Facultad de Ingeniería
Ciudad Universitaria Ciudad de México, México

***Resumen*—Esta práctica se usarán las bases de datos proporcionadas con el fin de que los alumnos aprendan a desplegar y analizar la información de manera que se aprecie la información y que además se despliegue de forma gráfica.**

Palabras clave—pandas, numpy, scipy, seaborn, matplotlib

I. Introducción

En esta práctica usaremos 5 bibliotecas para poder leer nuestra base de datos:

A. Pandas

Pandas es una biblioteca que se encarga de leer archivos separados por comas, en este caso nuestra base de datos sobre iris setosa que ya está clasificado según sus características.

B. Numpy

La biblioteca se encarga de ayudarnos a generar un arreglo 2d al ya tener una función para poder generarlo con las características que solicita el ejercicio 3.4.

C. Scipy

Esta Biblioteca es una extensión de Numpy la cual sirve generar y trabajar con librerías dispersas, con lo que nos apoyaremos en el ejercicio 3.4 para poder generar la matriz con las características que pide.

D. Matplothlib

Para la generación de la parte grafica nos apoyaremos de matplothlib que sirve para poder desplegar datos de maneras diferentes, en este caso lo usaremos para generar histogramas y diferentes graficas para comparar los tipos de Iris setosa.

A. Seaborn

Seaborn servirá como una extensión para poder generar graficas mas complejas, en este caso lo usaremos para generar graficas de dispersión para poder comparar los diferentes tipos de Iris setosa.

```python
import pandas as pd
import numpy as np
from scipy.sparse import csr_matrix
import seaborn as sns
import matplotlib.pyplot as plt
```

Fig.1 Bibliotecas de la practica

Definimos las rutas de las bases para poder ver las que tienen NaN o no, apoyándonos de pandas para leer y extraer los datos de ambos y generar el dataframe

```python
# Define la ruta del archivo iris.data en la ubicación actual del script
#ruta_actual = 'EjerciciosIris/iris.data'  # Ruta del archivo en la carpeta EjerciciosIris
ruta_actual = 'iris.data'  # Ruta del archivo en la carpeta EjerciciosIris

# Usa una de las siguientes líneas, comentando o descomentando según sea necesario
iris_df = pd.read_csv(ruta_actual, header=None, names=['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species'])

# Extraer características y etiquetas directamente del DataFrame
X = iris_df[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]
y = iris_df['species']

# Crear un DataFrame de pandas
iris_data = pd.DataFrame(data=np.c_[X, y], columns=["SepalLength", "SepalWidth", "PetalLength", "PetalWidth", "Class"])
```

Fig.2 Generación del dataframe y distinción de rutas

II. Ejercicio 1

El primer ejercicio es sencillo al ya tener definido el dataframe solo se imprimen los primeros 10 del encabezado usando la función head con parámetro 10 para que imprima los primeros 10.

```python
print("Ejercicio 3.1")
# Mostrar la descripción de los datos
print("Descripción de los datos:")
print(iris_data.head(10))
```

Fig.3 Código ejercicio 1

```
Ejercicio 3.1
Descripción de los datos:
  SepalLength SepalWidth PetalLength PetalWidth        Class
0         5.1        3.5         1.4        0.2  Iris-setosa
1         4.9          3         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        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        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
```

Fig.4 resultados del ejercicio 1

III. Ejercicio 2

Para el segundo ejercicio igual al ya tener el dataframe solo usamos la función keys para obtener las llaves y la función shape para obtener sus filas y columnas que son de pandas.

```
print("Ejercicio 3.2")
# Imprimir las llaves y el número de filas y columnas
print("\nLlaves:")
print(iris_data.keys())
print("\nNúmero de filas y columnas:")
print(iris_data.shape)
```

Fig.5 Código ejercicio 2

```
Ejercicio 3.2

Llaves:
Index(['SepalLength', 'SepalWidth', 'PetalLength', 'PetalWidth', 'Class'], dtype='object')

Número de filas y columnas:
(150, 5)
Ejercicio 3.3
```

Fig.6 resultado ejercicio 2

IV. EJERCICIO 3

Para obtener los valores nulos que nos pide el ejercicio nos apoyamos del dataframe y una función isnull y sum para que obtenga los valores nulos de cada característica de la base además de que tipo de datos se hizo el conteo.

```
print("Ejercicio 3.3")
# Obtener el número de muestras faltantes o NaN
print("\nNúmero de muestras faltantes:")
print(iris_data.isnull().sum())
```

Fig.7 Código de ejercicio3

```
Número de muestras faltantes:
SepalLength    3
SepalWidth     0
PetalLength    0
PetalWidth     1
Class          0
dtype: int64
```

Fig.8 Resultados ejercicio 3

V. EJERCICIO 4

El ejercicio 4 pide crear un arreglo con numpy con una diagonal de 1 con 0 en el resto y luego dispersarla con scipy, entonces usamos la función eye de numpy con parámetro 5 que crea una matriz diagonal cuadrada y el parámetro define el porque en este caso queremos una 5x5, la imprimimos para corroborar y luego con csr_matrix de scipy la convertimos en dispersa para imprimirla.

```
print("Ejercicio 3.4")
# Paso 3.4: Crear un arreglo 2-D de tamaño 5x5
con unos en la diagonal y ceros en el resto
array_2d = np.eye(5)
print(array_2d)
# Convertir el arreglo a una matriz dispersa de
Scipy en formato CRS
sparse_matrix = csr_matrix(array_2d)
# Mostrar la matriz dispersa
print("\nMatriz dispersa:")
print(sparse_matrix)
```

Fig.9 Código ejercicio 4

```
Ejercicio 3.4
[[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:
  (0, 0)        1.0
  (1, 1)        1.0
  (2, 2)        1.0
  (3, 3)        1.0
  (4, 4)        1.0
```

Fig.10 Resultados ejercicio 4

VI. EJERCICIO 5

En este ejercicio nos pide usar describe para desplegar el promedio y la desviación estándar por lo que nos apoyaremos de loc que nos permite obtener solo una fracción de los datos de describe en este caso mean para el promedio y std para la desviación estándar.

```
# Paso 3.5: Mostrar estadísticas básicas
utilizando describe
print("\nEjercicio 3.5")
print(iris_df.describe().loc[['mean', 'std']])
```

Fig.11 Código ejercicio 5

```
Ejercicio 3.5
      sepal_length  sepal_width  petal_length  petal_width
mean      5.857823     3.054000      3.758667     1.196644
std       0.828013     0.433594      1.764420     0.765331
Ejercicio 3.6
```

Fig.12 Resultado ejercicio 5

VII. Ejercicio 6

Para el ejercicio 6 nos pide obtener cuantas muestras hay de cada clase, por lo que usamos nuestro dataframe de solo class y usamos la función value_counts y la guardamos en una variable para poder imprimirlo.

```
print("Ejercicio 3.6")
# Paso 3.6: Obtener el número de muestras para cada clase
samples_per_class = iris_data['Class'].value_counts()
print("\nNúmero de muestras por clase:")
print(samples_per_class)
```

Fig.13 Código ejercicio 6

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

Fig.14 Resultado ejercicio 6

VIII. Ejercicio 7

Para el ejercicio 7 solo buscamos en el documento iris.data los nombres para ponerlos en una variable header e igualamos las columnas a las del dataframe e imprimimos con value_counts como en el ejercicio anterior.

```
print("Ejercicio 3.7")
# Paso 3.7: Añadir un encabezado a los datos usando los nombres en iris.names
header = ['sepal length in cm', 'sepal width in cm', 'petal length in cm', 'petal width in cm', 'Class']
iris_data.columns = header
print(iris_data['Class'].value_counts())
```

Fig.15 Código ejercicio 7

```
Ejercicio 3.7
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Class, dtype: int64
```

Fig.16 Resultados ejercicio 7

IX. Ejercicio 8

El ejercicio 8 pide solo imprimir las primeras 10 filas y las 2 primeras columnas por lo que nos apoyaremos de la función iloc de pandas que con los parámetros [menor:filas,menor:columnas] le pondermos [:10,:2] para cumplir las características.

```
print("Ejercicio 3.8")
# Paso 3.8: Imprimir las diez primeras filas y las dos primeras columnas del data frame
print("\nLas diez primeras filas y las dos primeras columnas:")
print(iris_data.iloc[:10, :2])
```

Fig.17 Código ejercicio 8

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

Fig.18 Resultados ejercicio 8

X. Ejercicio 9

El ejercicio 9 pide crear una grafica de barras por lo que usaremos de nuevo la función describe con loc para obtener el promedio, mínimo y máximo y con plot de matplotlib lo añadimos a la gráfica de barras con sus títulos.

```
# 3.9 Crear una gráfica de barras para la media, mínimo y máximo de todos los datos
iris_describe = iris_df.describe()
iris_describe.loc[['mean', 'min', 'max']].plot(kind='bar', figsize=(10, 6))
plt.title('Media, Mínimo y Máximo de características del Iris')
plt.ylabel('Valores')
plt.xlabel('Estadísticas')
plt.show()
```

Fig.19 Código ejercicio 9

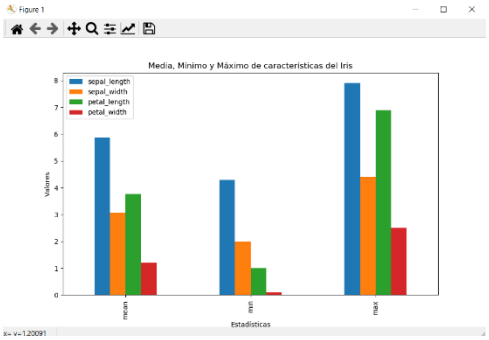

Fig.20 Resultados ejercicio 9

XI. EJERCICIO 10

El ejercicio 10 pide hacer una grafica pastel con la frecuencia de las especies de la Iris setosa, por lo que nos apoyaremos de la función value_counts de pandas y lo metemos a la grafica con plot y pie para definirlo como pastel y le añadimos el título.

```
# 3.10 Mostrar la frecuencia de las tres especies como una gráfica de pastel
iris_df['species'].value_counts().plot.pie(autopct='%1.1f%%', figsize=(8, 8))
plt.title('Distribución de especies en el dataset Iris')
plt.ylabel('')
plt.show()
```

Fig.21 Código ejercicio 10

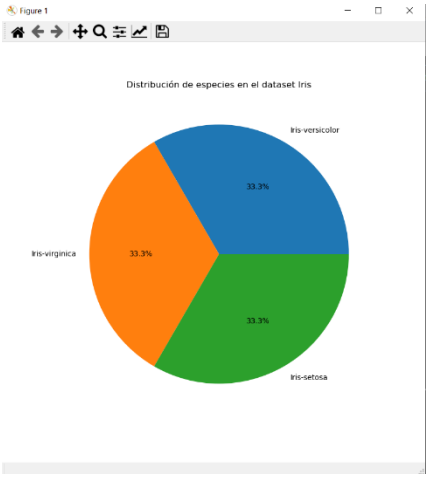

Fig.22 Resultados ejercicio 10

XII. EJERCICIO 11

Este ejercicio pide que usemos una gráfica para mostrar la relación entre la longitud y ancho del sépalo del dataframe a lo que usaremos seaborn con la función scatterplot para que directamente del dataframe de pandas los compare según su especie y le agregamos sus títulos.

```
# 3.11 Gráfica de la relación entre la longitud y ancho del sépalo para las tres especies
sns.scatterplot(x='sepal_length', y='sepal_width', hue='species', data=iris_df)
plt.title('Relación entre longitud y ancho del sépalo por especie')
plt.show()
```

Fig.23 Código ejercicio 11

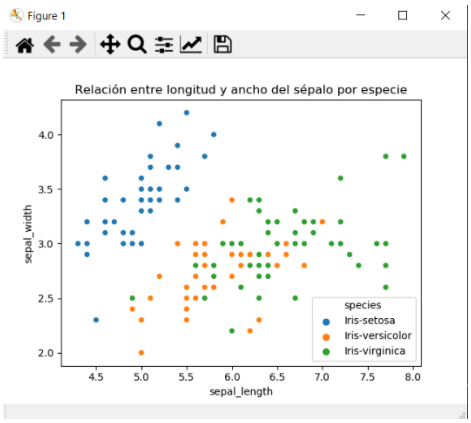

Fig.24 Resultados ejercicio11

XIII. EJERCICIO 12

Este ejercicio pide hacer histogramas de los datos por lo que para hacerlo usaremos la función drop de pandas para quitar las categorías de las Iris y la función hist de matplotlib para graficar el resto de los datos contables.

```
# 3.12 Histogramas de las variables SepalLength, SepalWidth, PetalLength y PetalWidth
iris_df.drop('species', axis=1).hist(bins=15, figsize=(10, 7))
plt.suptitle('Histogramas de características del Iris')
plt.show()
```

Fig.25 Código ejercicio12

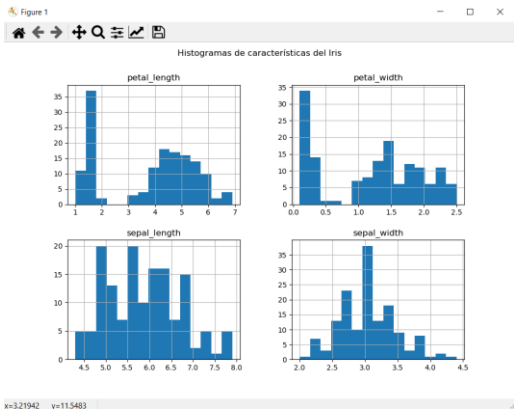

Fig.26 Resultados ejercicio12

XIV. EJERCICIO 13

Para este ejercicio usaremos pairplot de seabron para generar las gráficas que comparan los datos numéricos y los distinguiremos por especies ya directamente del dataframe.

```python
# 3.13 Gráficas de dispersión con pairplot de seaborn
sns.pairplot(iris_df, hue='species')
plt.show()
```

Fig.27 Código ejercicio 13

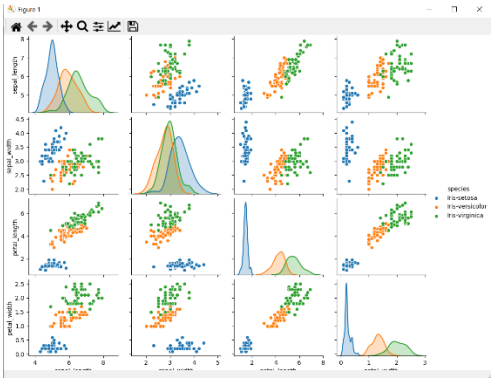

Fig.28 Resultados ejercicio 13

XV. Ejercicio 14

Para este ejercicio nos pide usar joinplot de seaborn para comparar la longitud y el ancho del sépalo a lo que pasaremos esas características como x, y respectivamente en data pasaremos el dataframe y de en kind scatter para que sea de dispersión.

```python
# 3.14 Gráfica de dispersión entre longitud y ancho del sépalo con jointplot
sns.jointplot(x='sepal_length', y='sepal_width', data=iris_df, kind='scatter', color='green')
plt.suptitle('Dispersión entre longitud y ancho del sépalo')
plt.show()
```

Fig.29 Código ejercicio 14

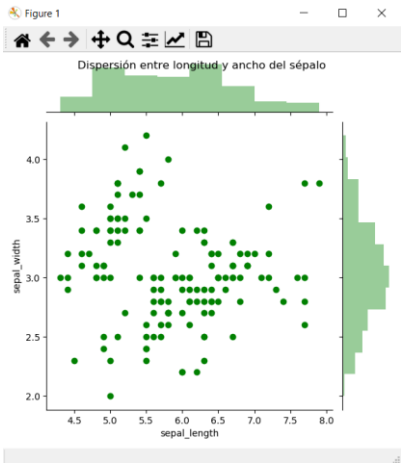

Fig.30 Resultados ejercicio 14

XVI. Ejercicio 15

En este ejercicio repetimos el uso de joinplot y sus parámetros con la excepción de poner hex en kind por lo que solo cambia el como se representa la gráfica.

```python
# 3.15 Repetir el ejercicio anterior usando kind="hex"
sns.jointplot(x='sepal_length', y='sepal_width', data=iris_df, kind='hex', color='green')
plt.suptitle('Hexbin de longitud y ancho del sépalo')
plt.show()
```

Fig.31 Código ejercicio 15

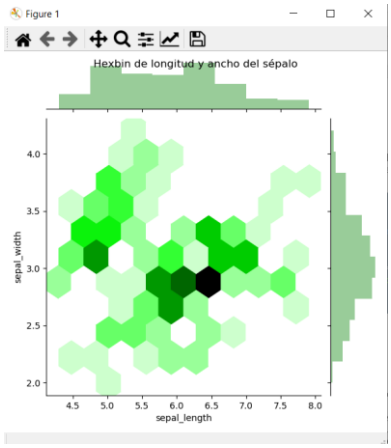

Fig.32 Resultados ejercicio 15

XVII. Conclusión

La practica nos ayudo a ver lo que podemos hacer con esta clase de bases de datos al orientarnos en como usarlas en código y el como desplegar los datos según las necesidades, ya sea por características, tipos, datos desconocidos o directamente compararlos de manera grafica.

XVIII. Referencias

[1] [2] A. S. Alberca, “La librería Matplotlib”, Aprende con Alf. [En línea]. Disponible en: https://aprendeconalf.es/docencia/python/manual/matplotlib/. [Consultado: 04-mar-2024].

[2] “Seaborn”, W3schools.com. [En línea]. Disponible en: https://www.w3schools.com/python/numpy/numpy_random_seaborn.asp. [Consultado: 04-mar-2024].

[3] [3] “Pandas DataFrames”, W3schools.com. [En línea]. Disponible en: https://www.w3schools.com/python/pandas/pandas_dataframes.asp. [Consultado: 04-mar-2024].