# Trabajo Especial: Ejercicios con iris

Reconocimiento de Patrones - 0757

*Facultad de Ingeniería*
*Universidad Nacional Autónoma de México*

Ceballos Equihua C. N.
*Ingeniería en Computación*
*Facultad de Ingeniería, UNAM*
Ciudad de México, México
ceballos.equihua@gmail.com

Muñoz Marbán J.
*Ingeniería en Mecatrónica*
*Facultad de Ingeniería, UNAM*
Ciudad de México, México
juanmzmb@gmail.com

Murrieta Villegas A.
*Ingeniería en Computación*
*Facultad de Ingeniería, UNAM*
Ciudad de México, México
alfonsomvmx@gmail.com

Solano González F. J.
*Ingeniería en Mecatrónica*
*Facultad de Ingeniería, UNAM*
Ciudad de México, México
felipe.solano.gos@gmail.com

**Profesores:**
Dr. Boris Escalante Ramírez
Dra. Olveres Montiel Jimena
I.I.M.A.S. - UNAM

*Resumen*— **Este ejercicio consiste en analizar un conjunto de datos sobre flores de la especie iris y mostrar gráficas con información relevante sobre las características diferenciables de estas especies de flores (la anchura y longitud del sépalo y del pétalo). Además, se realizó un modelo simple de regresión logística para clasificar los datos.**

## I. Introducción

A lo largo de esta práctica, se obtendrá la información sobre tres especies de plantas a partir de archivos proporcionados y se desplegará de forma gráfica, utilizando las bibliotecas Matplotlib y Seaborn de Python. Se utilizará la biblioteca Pandas para cargar la información y obtener sus estadísticas básicas, como la media y la desviación estándar. Se mostrarán gráficas para poder visualizar la información relevante de forma sencilla y las relaciones entre las características medidas de las especies de plantas; esto es, la distribución de las medidas en relación con la especie de planta y en comparación con las otras especies.

El dataset de las mediciones de las plantas contiene 50 muestras por especie, por lo que, al ser 3 especies, en total contiene 150 muestras. Como se comprobará posteriormente, no existen muestras nulas.

Las bibliotecas de Python que se utilizarán para manipular las imágenes en la presente práctica son:

**Pandas:** Biblioteca orientada a herramientas de manipulación y análisis de datos.

**Numpy:** Biblioteca con herramientas para crear y manipular arreglos y matrices, junto con funciones para operar en dichas estructuras.

**Matplotlib:** Biblioteca para crear imágenes y gráficas para visualizar información.

**Seaborn:** Biblioteca para visualizar datos basada en Matplotlib; proporciona herramientas para desplegar gráficas estadísticas.

**SciPy:** Biblioteca orientada a usos científicos, ingenieriles, matemáticos y de cómputo técnico.

**Sklearn:** Biblioteca orientada a *Machine Learning*, provee herramientas con algoritmos de clasificación, regresión y agrupación.

## II. Desarrollo

La práctica consiste en 3 ejercicios, cada uno de los cuales consiste en varias actividades. El primer ejercicio consta de actividades básicas (cargar los datos y mostrarlos de diferentes formas con Pandas), el segundo se trata de actividades de visualización (despliegue de gráficas con Matplotlib y Seaborn) y el último ejercicio es sobre regresión logística (construcción y prueba del modelo).

### *A. Ejercicios básicos*

Para comenzar con la actividad de ejercicios básicos, los datos del dataset de iris se cargaron en un dataframe de Pandas, utilizando esta biblioteca, también se obtuvieron la forma de los datos, el tipo y las primeras 10 filas.

```
df_iris.dtypes
```

```
0      float64
1      float64
2      float64
3      float64
4       object
dtype: object
```

```
df_iris.head(10)
```

| | 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 |

```
type(df_iris)
```

```
pandas.core.frame.DataFrame
```

Fig. 1 Información del Dataframe

Después de obtener la información del Dataframe, se utiliza el dataset que contiene los datos del header para imprimir las llaves, el número de filas y de columnas.

```
df_irisH.axes
```

```
[RangeIndex(start=0, stop=150, step=1),
 Index(['SepalLength', 'SepalWidth', 'PetalLength', 'PetalWidth', 'Class'], dtype='object')]
```

Fig. 2 Llaves

```
df_irisH.shape
```

```
(150, 5)
```

Fig. 2 Forma del dataset

Como punto adicional se obtiene información acerca de la existencia de valores nulos en el dataset.

```
df_iris.isnull().sum()
```

```
0    0
1    0
2    0
3    0
4    0
dtype: int64
```

Fig. 3 Cantidad de valores nulos por columnas

En el siguiente ejercicio, se crea una matriz identidad de tamaño 5x5 y se obtiene una matriz dispersa en formato CRS.

```
matrix=np.identity(5)
matrix
```

```
array([[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.]])
```

Fig. 4 Creación de la matriz identidad

Se obtiene la matriz dispersa en formato CRS.

```
matrix_csr=csr_matrix(matrix)
print(matrix_csr)
```

```
  (0, 0)        1.0
  (1, 1)        1.0
  (2, 2)        1.0
  (3, 3)        1.0
  (4, 4)        1.0
```

Fig. 5 Valores nulos por columna del dataset

Utilizando las propiedades de un dataframe, se obtienen las mediciones estadísticas básicas por columnas.

```
df_iris.describe()
```

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| **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 |

Fig. 6 Datos estadísticos del dataset.

De manera similar, se obtiene ahora sólo el valor de la media y la desviación estándar por columna.

```
df_iris.mean()
```

```
0    5.843333
1    3.054000
2    3.758667
3    1.198667
dtype: float64
```

```
df_iris.std()
```

```
0    0.828066
1    0.433594
2    1.764420
3    0.763161
dtype: float64
```

Fig. 7 Datos por columna de la media y desviación estándar.

De manera similar, se obtiene el número de muestras para cada clase en el dataset.

```
dataClass = df_irisH["Class"].value_counts()
dataClass
```

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

Fig. 8 Numero de muestras de cada clase.

Dada la naturaleza del dataset, se agregan los encabezados de las columnas.

```
headers = ['sepal length' , 'sepal width','petal length', 'petal width', 'class']
df_iris.columns=headers
df_iris.head()
```

| | 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 |

Fig. 9 Numero de muestras de cada clase.

Con los encabezados agregados, el dataframe ya lleva los encabezados y es más fácil de poder observar la información.

```
df_iris.iloc[[0,1,2,3,4,5,6,7,8,9,10],[0,1]]
```

| | sepal length | sepal width |
|---|---|---|
| 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 |
| 10 | 5.4 | 3.7 |

Fig. 10 Primeras diez columnas con encabezados.

## *B. Ejercicios de visualización*

En esta actividad, se utilizan bibliotecas como Matplotlib y Seaborn, para esta primera actividad, se crean gráficos de barras que guardan o exponen medidas estadísticas para cada columna de características del dataset.

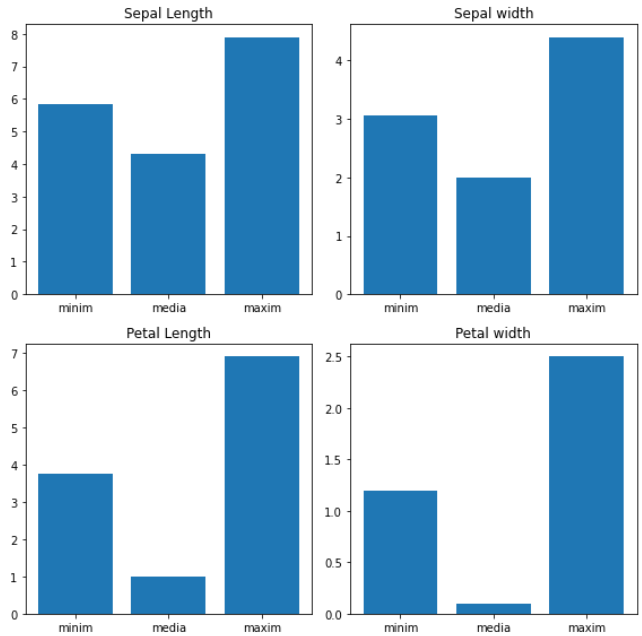


Fig. 11 Medidas estadísticas para cada columna

De manera similar que los ejercicios anteriores, se observa mediante una gráfica de pastel.

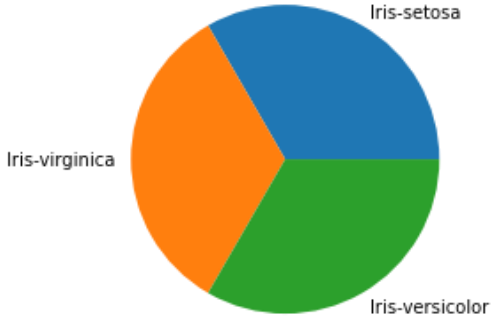


Fig. 12 Distribución de clases del dataset.

Mediante el uso de Matplotlib y Seaborn, se puede observar la relación que existe entre características, específicamente, en el ejercicio se generan las gráficas que muestran la relación que existe entre el ancho y longitud de sépalo.

Para la primera gráfica podemos observar como las características del sépalo evolucionan conforme se recorren los datos.

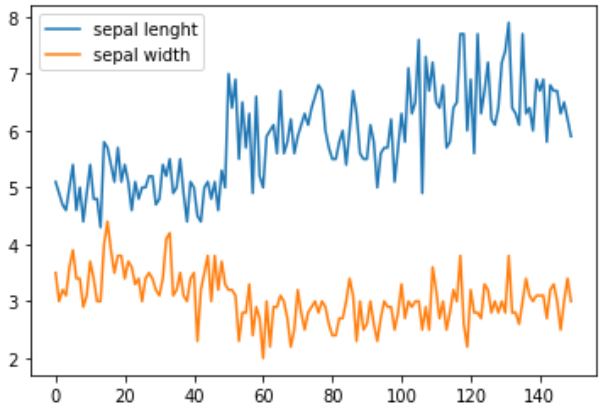


Fig. 13 Evolución de las características del sépalo

Ahora, se relacionan estas variables directamente, tomando en cuenta la categorización para cada muestra.

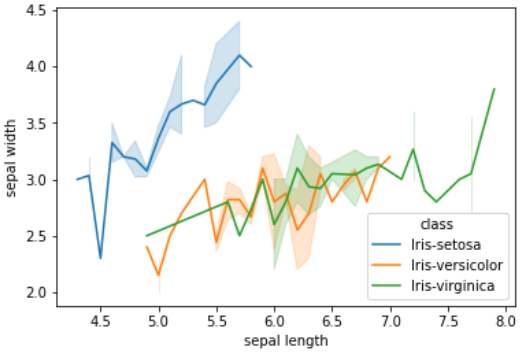


Fig. 14 Relación entre el ancho y largo del sépalo con categorización.

De manera similar que la gráfica anterior, ahora se muestra la relación, pero sin tomar en cuenta la categorización, esta gráfica nos da otra perspectiva de la relación que existe entre estas variables.

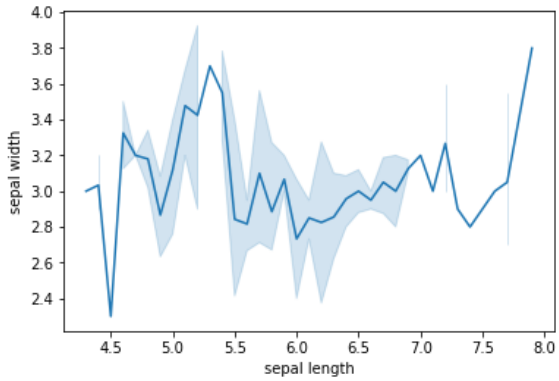


Fig. 15 Relación entre el ancho y largo del sépalo.

Para poder observar la distribución de cada característica del dataset se generan sus respectivos histogramas.

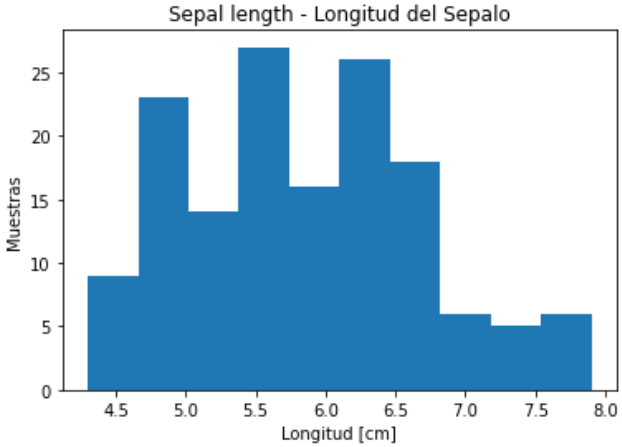


Fig. 16 Histograma de la longitud del sépalo.

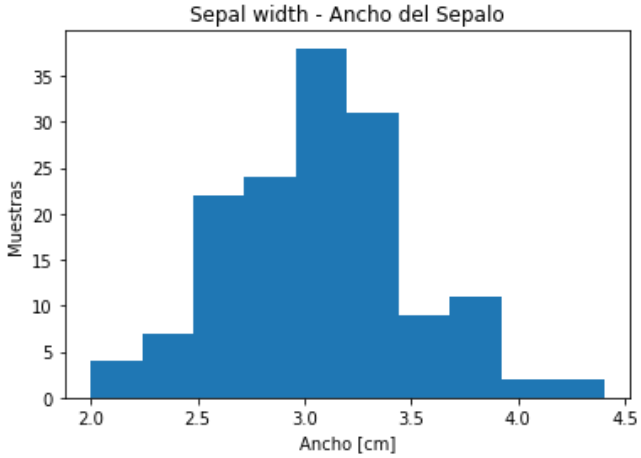


Fig. 17 Histograma del ancho de sépalo.

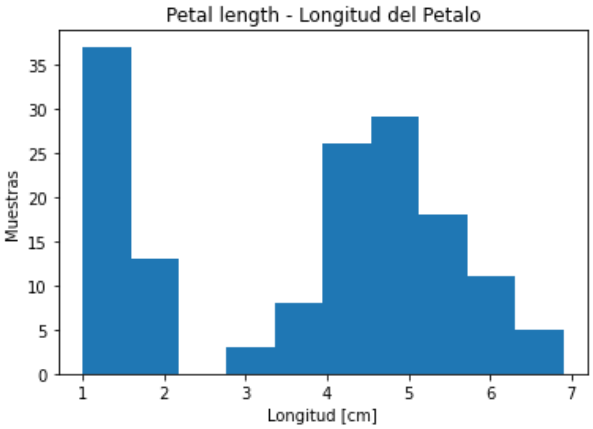


Fig. 18 Histograma de la longitud del pétalo.

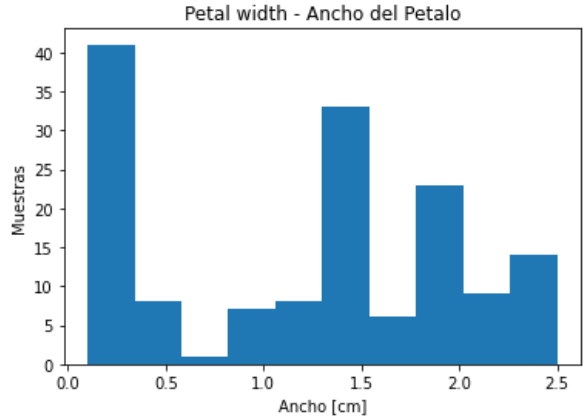


Fig. 19 Histograma del ancho del pétalo.

Una de las gráficas más útiles es pairplot de Seaborn, que genera todas las gráficas de relaciones entre las características e incluyendo su distribución. Se puede observar que los sépalos de las especies de iris versicolor y virginica son muy similares, por lo que, para realizar la categorización, en general sería mejor tomar los datos de los pétalos.

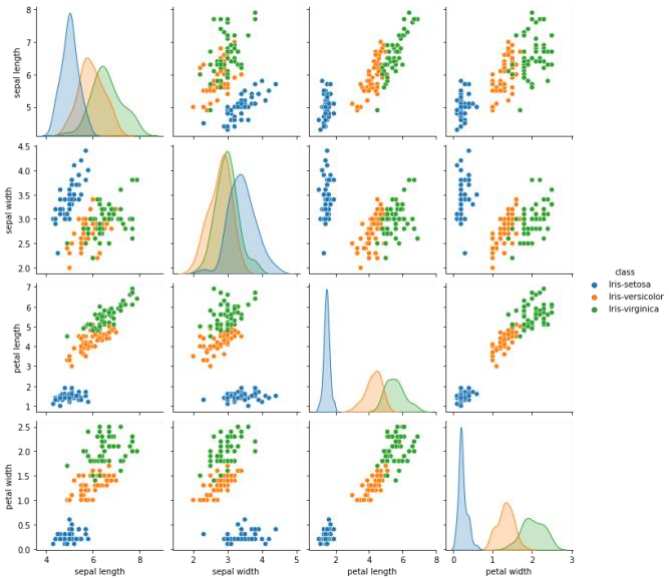

Fig. 20 Pairplot del dataset.

Se crea una gráfica usando joinplot para poder observar la dispersión entre la longitud y ancho del sépalo, así como sus distribuciones.

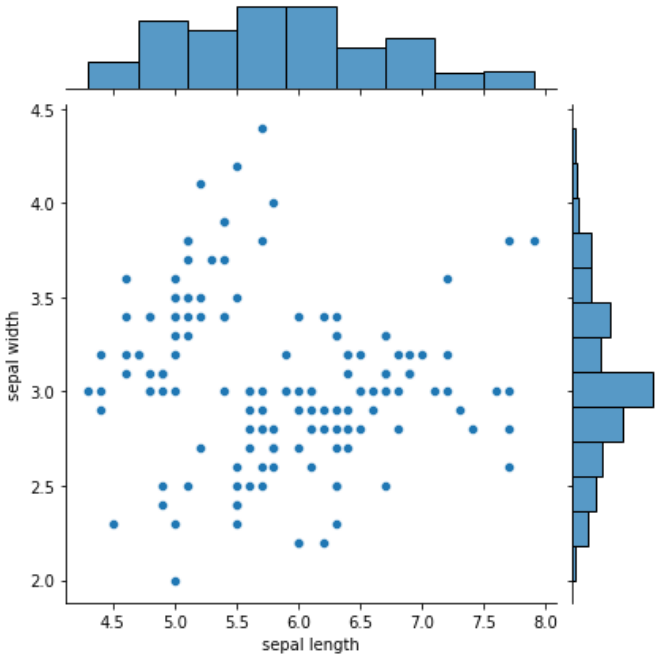


Fig. 21 Joinplot de la longitud y ancho de sépalo

Se genera la misma gráfica joinplot pero utilizando otro tipo de plot.

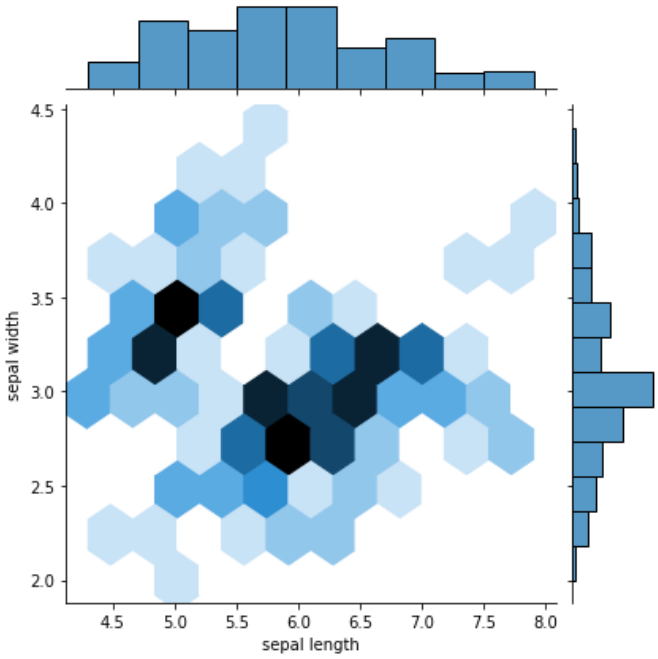


Fig. 22 Joinplot de la longitud y ancho de sépalo con diferente tipo de plot.

## *C. Ejercicios de regresión logística*

Para esta actividad, se comienza observando los datos estadísticos para cada clase.

Iris setosa

| | sepal length | sepal width | petal length | petal width |
|---|---|---|---|---|
| count | 50.00000 | 50.000000 | 50.000000 | 50.00000 |
| mean | 5.00600 | 3.418000 | 1.464000 | 0.24400 |
| std | 0.35249 | 0.381024 | 0.173511 | 0.10721 |
| min | 4.30000 | 2.300000 | 1.000000 | 0.10000 |
| 25% | 4.80000 | 3.125000 | 1.400000 | 0.20000 |
| 50% | 5.00000 | 3.400000 | 1.500000 | 0.20000 |
| 75% | 5.20000 | 3.675000 | 1.575000 | 0.30000 |
| max | 5.80000 | 4.400000 | 1.900000 | 0.60000 |

Fig. 23 Datos estadísticos para iris setosa.

Iris versicolor

| | sepal length | sepal width | petal length | petal width |
|---|---|---|---|---|
| count | 50.000000 | 50.000000 | 50.000000 | 50.000000 |
| mean | 5.936000 | 2.770000 | 4.260000 | 1.326000 |
| std | 0.516171 | 0.313798 | 0.469911 | 0.197753 |
| min | 4.900000 | 2.000000 | 3.000000 | 1.000000 |
| 25% | 5.600000 | 2.525000 | 4.000000 | 1.200000 |
| 50% | 5.900000 | 2.800000 | 4.350000 | 1.300000 |
| 75% | 6.300000 | 3.000000 | 4.600000 | 1.500000 |
| max | 7.000000 | 3.400000 | 5.100000 | 1.800000 |

Fig. 24 Datos estadísticos para iris versicolor.

Iris virginica

| | sepal length | sepal width | petal length | petal width |
|---|---|---|---|---|
| count | 50.00000 | 50.000000 | 50.000000 | 50.00000 |
| mean | 6.58800 | 2.974000 | 5.552000 | 2.02600 |
| std | 0.63588 | 0.322497 | 0.551895 | 0.27465 |
| min | 4.90000 | 2.200000 | 4.500000 | 1.40000 |
| 25% | 6.22500 | 2.800000 | 5.100000 | 1.80000 |
| 50% | 6.50000 | 3.000000 | 5.550000 | 2.00000 |
| 75% | 6.90000 | 3.175000 | 5.875000 | 2.30000 |
| max | 7.90000 | 3.800000 | 6.900000 | 2.50000 |

Fig. 25 Datos estadísticos para iris virginica.

Se genera una gráfica de dispersión para observar los conjuntos que se forman relacionando el ancho pétalo y longitud de sépalo.

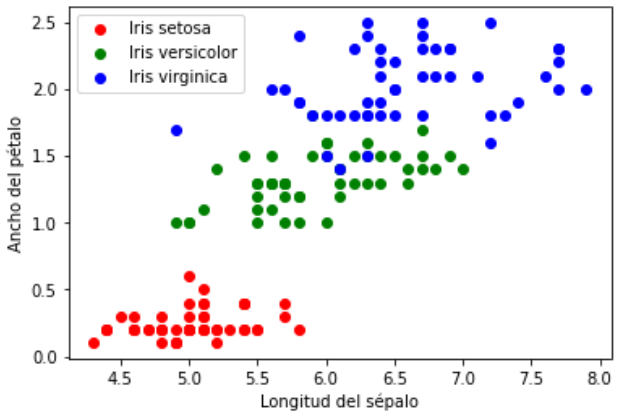


Fig. 26 Gráfica de dispersión

Como último punto de los ejercicios, se realiza una clasificación utilizando una regresión logística. Para esto, se hace uso de las librerías de sklearn.

Primero, se hace la división de los datos entre conjunto de entrenamiento y conjunto de prueba, utilizando el 75% de estos como conjunto de entrenamiento. Se crea el modelo logístico y se llena con los datos de entramiento.

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.75)
lr_model = LogisticRegression()
lr_model.fit(x_train, y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

Fig. 27 Primera parte de la regresión logística.

Después de haber creado el modelo, se utilizan los datos de prueba y se obtienen la precisión de este.

```
Y_pred = lr_model.predict(x_test)
print('Precisión: ',lr_model.score(x_test, y_test))
```

```
Precisión:  0.9734513274336283
```

Fig. 28 Evaluando el modelo obtenido.

Por último, se genera la matriz de confusión y se utiliza un mapa de calor para representar la distribución de nuestros datos además de mostrar los casos falsos positivos.

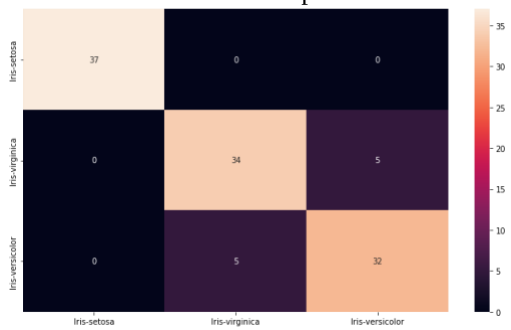


Fig. 29 Matriz de confusión.

## III. Conclusiones

En la presenta práctica aprendimos a emplear bibliotecas de Python bastante comunes en la analítica y búsqueda de patrones de dataset como fue el caso de ***pandas*** y ***numpy*** para el tratamiento de datos y, por otro lado, ***matplotlib*** y ***seaborn*** para la graficación y contraste de comportamientos en nuestros datos. Por otro lado, a través de estadística descriptiva y medidas de tendencia central es como encontramos patrones y relación entre variables en nuestros datos, dándonos de esa forma un primer acercamiento a nuestras futuras ideas y propuestas para el diseño de una clasificación mediante una inteligencia artificial basada en una regresión logística.

Por último, a través del uso de ***sklearn*** entrenamos un modelo de inteligencia artificial para la clasificación de cada tipo de flor, además, no solamente validamos la veracidad o precisión de nuestro modelo mediante el resultado promedial sino que validamos cada clase de nuestro conjunto de datos a través de una matriz de confusión.

## Referencias


[1] H “Matplotlib: Visualization with Python”, matplotlib, [En línea]. Disponible en: https://matplotlib.org/ [Consultado el 26 de Septiembre de 2021].

[2] “scipy.sparse.csr_matrix”, SciPy.org, [En línea]. Disponible en: https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csr_mat rix.html?highlight=csr#scipy.sparse.csr_matrix [Consultado el 26 de Septiembre de 2021].

[3] H “Isklearn.linear_model.LogisticRegression ”, scikit-learn. [En línea]. Disponible en: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html [Consultado el 26 de Septiembre de 2021].