

## Actividad Análisis de Clustering con Python: Pasos básicos

Para poder realizar la actividad, sigue los siguientes pasos:

### 1. Definir el problema

Primero, debemos identificar el problema que deseamos resolver y plantear nuestros objetivos. En este caso, vamos a desarrollar un clustering a partir de una base de datos pública muy popular: “Iris”. “Iris” fue creada por el estadístico británico Ronald Fisher en 1936, y se utiliza comúnmente como ejemplo en la literatura sobre el machine learning, el clustering y la visualización.

Esta base de datos contiene información sobre tres especies de flores iris: setosa, versicolor y virginica. Para cada especie, se midieron cuatro variables: longitud del sépalo, ancho del sépalo, longitud del pétalo y ancho del pétalo. En total, este conjunto de datos recoge 150 observaciones.



#### ¿Cuál es el objetivo del Clustering?

Encontrar patrones y relaciones entre las características de las flores, y determinar si dichas características pueden utilizarse para distinguir entre las tres especies de iris (setosa, versicolor y virginica). Cabe mencionar que esta base de datos es adecuada para desarrollar un análisis no supervisado porque no contiene información previa sobre la especie de la flor para cada observación.

### 2. Preparar los datos

En segundo lugar, procedemos a recopilar los datos necesarios y prepararlos para el análisis. Esto puede incluir la limpieza y la normalización de los datos, así como la selección de las variables relevantes. Para este ejercicio, vamos a importar el conjunto de datos “Iris” al entorno Spyder, que ya no necesita depuración.

Antes de nada, crearemos un nuevo archivo de Python llamado “clustering”.

Después, descargaremos la base de datos “Iris” pinchando en el siguiente enlace: <https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data>. Guárdala en la misma carpeta que el archivo de Python que has creado previamente. (No le cambies el nombre. Se descarga por defecto como iris.data).

#### 2.1. Importar las librerías en Spyder

A continuación, en el entorno Spyder, debemos importar las bibliotecas o paquetes necesarios.

## ¡IMPORTANTE!

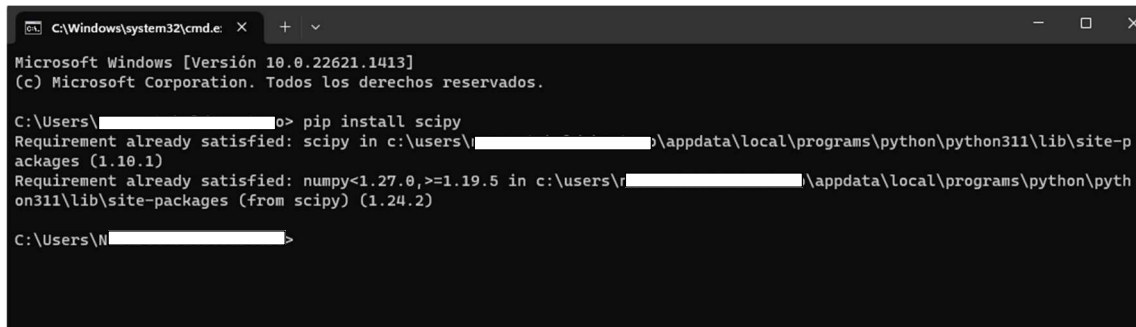
Para desarrollar el clustering, vamos a utilizar una biblioteca específica llamada `sciPy`. Para poder importarla a Spyder, primero debemos instalar el paquete en nuestro sistema a través de la terminal de Windows, introduciendo el código “**pip install scipy**”.

Para abrir la terminal:

**Si estás en Windows** → tecla Windows + tecla “R”. Después, escribe “cmd” en el cuadro de diálogo y acepta.

**Si estás en Mac** → cmd + tecla “espacio”.

En la terminal, escribe el código **pip install scipy** y pulsa “intro”.



```
C:\Windows\system32\cmd.e. X + v
Microsoft Windows [Versión 10.0.22621.1413]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\N[redacted]> pip install scipy
Requirement already satisfied: scipy in c:\users\N[redacted]\appdata\local\programs\python\python311\lib\site-packages (1.10.1)
Requirement already satisfied: numpy<1.27.0,>=1.19.5 in c:\users\N[redacted]\appdata\local\programs\python\python311\lib\site-packages (from scipy) (1.24.2)

C:\Users\N[redacted]>
```

Después, en el entorno Spyder, introduce los siguientes códigos para importar las librerías necesarias:

```
import pandas as pd
import numpy as np
import scipy
import matplotlib as mpl
import matplotlib.pyplot as plt
```

Verifica en Spyder que todos los paquetes se han importado correctamente. Si no tienes instalado uno de los paquetes en tu sistema, en la consola inferior derecha aparecerá este mensaje: **ModuleNotFoundError: No module named 'xxxxxx'**. En ese caso, tendrías que instalar la biblioteca faltante en el sistema tal y como se ha hecho antes.

## 2.2. Importar la base de datos

Seguidamente, introduce el siguiente código para abrir la base de datos “iris.data” y ejecuta.

```
dataframe = pd.read_csv('iris.data', header=None, names=['longitud_sepalo', 'ancho_sepalo', 'longitud_petal', 'ancho_petal', 'especies'])
x = dataframe
y = dataframe['especies']
target_names = np.array(['Iris-senotsa', 'Iris-versicolor', 'Iris-virginica'])
```

En la consola superior derecha, en la pestaña “Variable Explorer”, podrás visualizar la base de datos si haces doble clic sobre la fila “dataframe”. **Pega una imagen de la tabla en tu actividad para verificar que la has cargado adecuadamente.**

### 2.3. Explorar los datos mediante una tabla de frecuencias y una tabla cruzada.

```
# Explorar los datos mediante tablas de frecuencias y de contingencia

# tabla de frecuencias de la variable 'especies'

print(dataframe.groupby('especies').size())

# tabla cruzada

cols = ['longitud_sepalo', 'ancho_sepalo', 'longitud_petal', 'ancho_petal', 'especies']
dataframe = dataframe[cols]

cross_tab = pd.pivot_table(dataframe, index='especies', aggfunc='mean')
print(cross_tab)
```

Podrás visualizar la tabla de frecuencias y la tabla de contingencia en la consola inferior derecha. **Pega una imagen de las tablas en tu documento de actividad.**

### 3. Seleccionar el algoritmo de clustering

Existen diversos algoritmos de clustering, y la elección del adecuado dependerá del problema específico y de los datos disponibles.

Para la base de datos Iris, utilizaremos el algoritmo **K-means**, muy popular para el clustering debido a su simplicidad y velocidad. K-means agrupa los datos en K grupos, donde K es un número predefinido de grupos. Este algoritmo funciona minimizando la distancia entre los puntos de los datos y los centroides de los grupos.

Para nuestra actividad, usaremos el algoritmo K-means con 3 clusters. En este caso, podemos definir el número de clusters porque sabemos de antemano que la base de datos contiene tres tipos diferentes de especies de iris. En otras situaciones, donde no hay información previa sobre el número de grupos, será necesario utilizar técnicas de selección del número óptimo de clusters, como la técnica del codo.

Introduce el siguiente código y ejecuta el algoritmo:

```
# Calcular el K-Means

from scipy import stats
from scipy.stats import mode
from scipy.cluster.vq import kmeans, vq

x = dataframe.iloc[:, :-1].values
centroids, labels = kmeans(x, 3)

centers = centroids
labels, _ = vq(x, centroids)
```

En la consola superior derecha, verás, entre otras tablas, una llamada “centroids”. Los centroides son puntos que representan el centro de masa de un cluster. En un modelo como K-means, los centroides se utilizan para representar el centro de cada cluster formado por los datos. **Pega esta tabla en tu actividad.**

#### 4. Visualiza los clusters

```
# Graficamos los puntos y los clusters

plt.scatter(x[:,0], x[:,1], c=labels)
plt.scatter(centers[:,0], centers[:,1], marker='*', s=200, c='#050505')
plt.show()
```

En la consola superior derecha, en la pestaña “plots”, visualizarás un gráfico con los clústers y sus centroides. **Pega el gráfico en tu actividad.**

#### 5. Evaluar los resultados del clustering.

Finalmente, debes evaluar los resultados del clustering. Para evaluar un modelo de clustering K-means, vamos a utilizar dos métricas de evaluación comunes:

- Suma de cuadrados intra-cluster (SSW): es la suma de las distancias cuadradas de cada punto de datos al centroide de su cluster correspondiente. SSW es una medida de la cohesión dentro de los clusters y se desea que sea lo más pequeño posible.
- Suma de cuadrados inter-cluster (SSB): es la suma de las distancias cuadradas entre los centroides de los clusters. SSB es una medida de la separación entre los clusters y se busca que sea lo más grande posible.

Introduce y ejecuta este código:

```
# Evaluar el modelo

# Suma de cuadrados intra-cluster (SSW)

def ssw(x, labels, centroids):
    ssw = 0
    for i in range(len(centroids)):
        cluster = x[labels == i]
        ssw += np.sum((cluster - centroids[i])**2)
    return ssw
print('SSW:', ssw(x, labels, centroids))

# Suma de cuadrados inter-cluster (SSB)

def ssb(x, labels, centroids):
    ssb = 0
    mean = np.mean(x, axis=0)
    for i in range(len(centroids)):
        cluster = x[labels == i]
        ssb += len(cluster) * np.sum((centroids[i] - mean)**2)
    return ssb
print('SSB',ssb(x, labels, centroids))
```

En la consola inferior derecha, podrás ver el resultado del SSW y del SSB. **Pega los resultados en el documento de tu actividad.**

**Envía la actividad a tu tutor en un documento de pdf. En dicho documento, deberás incluir todos los elementos que estimes oportunos: texto explicativo, imágenes, volcados de pantalla, código de Python, etc., para demostrar que has resuelto las cuestiones solicitadas.**

## **ANEXOS DE LA ACTIVIDAD**

**ANEXO 1.** Se adjunta el código completo del ejercicio en un documento de texto.

### **ANEXO 2.** Clustering con Scikit-learn (sklearn)

Como habéis podido comprobar, para desarrollar el Clustering, hemos utilizado la biblioteca SciPy. No obstante, la más común es Scikit-learn (sklearn), puesto que ofrece algoritmos y herramientas de análisis adicionales, y es más completa que SciPy.

Entonces, ¿por qué nos hemos decantado por esta última? Porque al utilizar Spyder como una aplicación de escritorio, pueden surgir conflictos e incompatibilidades a la hora de importar el paquete sklearn. Así pues, hemos querido evitarlos y asegurarnos de que pudieseis desarrollar el clustering sin obstáculos.

Ahora bien, si queréis seguir estudiando la técnica del Clustering a un nivel avanzado, os proporcionamos un documento que recoge el código Python para desarrollar este análisis con la base de datos “Iris”.