Aprendizaje no supervisado

Lección 4: Agrupar datos con distintos algoritmos de agrupamiento

Introducción

En una empresa inmobiliaria de la cual eres el científico de datos, se pueden recopilar datos de inmuebles relacionados con el precio de cada vivienda, año de construcción, tamaño, estado del inmueble, entre otras informaciones. Estos datos por sí solos no generan información valiosa más que la propia descripción de esos indicadores. Sin embargo, pueden ser usados para segmentar el mercado inmobiliario.

Este ejemplo de información es muy importante para que tu empresa impulse su transformación, realizando acciones adecuadas derivadas de este tipo de análisis. Como ya conoces, a través de algoritmos de agrupamiento puedes agrupar los inmuebles según sus características, de manera que se obtengan grupos de inmuebles similares.

En este tema lograrás usar algoritmos del paquete *Scikit-learn* para agrupar datos. Ya estás al tanto de los conceptos básicos del agrupamiento de datos, ahora los pondrás en práctica. También conoces como transformar los datos si es necesario hacerlo, así que estás listo para comenzar a agrupar datos.

Definición del algoritmo de agrupamiento *K-Means*

Imagina que en tu labor como científico de datos se requiere que apoyes a la administración de una reserva de animales, en donde se han recopilado información sobre los animales. Ayudemos al director para agruparlos en grupos.

¡Comencemos!

Implementación

Se requiere que apoyes a la administración de una reserva de animales, el director de la reserva ha recopilado información sobre todos los animales y necesita agruparlos en grupos similares para reestructurar el hábitat de cada uno de ellos. Un resumen de los datos se muestra en la tabla:

Nombre	Acuático	Patas
León marino	Verdadero	2
Elefante	Falso	4
Leopardo	Falso	4
Cangrejo	Verdadero	8
Medusa	Verdadero	0
Gusano	Falso	0

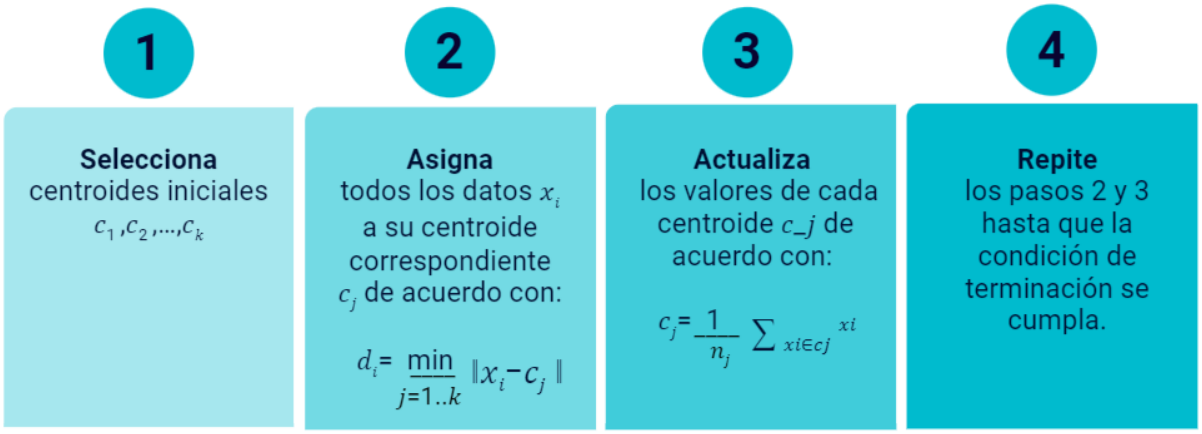
En el Tema 1 de este curso mencionamos una estrategia para agrupar datos similares basada en encontrar centroides o datos representantes que permiten resumir la información. Veamos a continuación como implementa esa estrategia, el que es quizás el algoritmo de agrupamiento más conocido: el *K-means* (Sinaga et al., 2020).

- **Objetivo**

El objetivo principal de este algoritmo es agrupar datos de manera que los datos en un mismo grupo sean similares entre sí y diferentes a los datos de otros grupos.

- **Pasos**

Los pasos principales del algoritmo *K-means* son:



Revisa con cuidado estos pasos. No pierdas de vista que la idea del algoritmo es encontrar datos similares a partir de seleccionar datos representantes por grupo.

¿Cómo podemos encontrar esos centroides?

La idea general del algoritmo es seleccionar aleatoriamente datos como centroides y luego mejorar esos centroides iterativamente. Para ello, particiona, es decir, segmenta, agrupa, los datos asignando cada dato al centroide más cercano.

¿Qué significa que un dato sea cercano a otro?

Este es un concepto importante. Si revisas los datos de la tabla notarás que los animales **Elefante** y **Leopardo** tienen exactamente los mismos valores en las variables, por lo tanto, son datos cercanos, y en este caso similares. ¿Cómo compararías la cercanía entre todos los datos?

Para ello se utiliza una función de similaridad. El K-means en particular utiliza la **Distancia Euclidiana** (Patel et al., 2020), aunque hay modificaciones para otras funciones.

La distancia euclidiana seguramente la conoces, es la manera en que se mide la distancia entre dos puntos en el plano cartesiano, y está definida como:

$$d(p,q) = \sqrt{\sum_{i=1}^{m} (p_i - q_i)^2}$$

Lo que hace esta función es restar los valores de dos datos en cada variable, elevar ese resultado al cuadrado, y finalmente sumar todas esas diferencias y calcular la raíz cuadrada. No te preocupes si leyendo la ecuación no comprendes del todo, a continuación, veremos un ejemplo.

Antes del ejemplo, revisemos las dos ecuaciones que aparecen en los pasos 2 y 3 de *K-means*. La ecuación del paso 2 lo que plantea es que cada dato será asociado con su centroide más cercano utilizando esta fórmula de la distancia euclidiana, solo que en la definición aparece de otra manera para dar espacio a que se usen otras funciones de comparación. El paso 3 plantea que una vez que se han asignado los datos a los centroides, se recalcula el centroide de cada grupo calculando el promedio en cada variable.

Ejemplo de una iteración de *K-means*

Analiza los datos de la Tabla 1. La columna **Acuático** tiene valores categóricos. ¿Cómo los transformas a numéricos para poder calcular esas distancias y promedios del algoritmo *K-means*? Si tu respuesta es usando *One Hot Encoder* tu respuesta es correcta. Pero por simplicidad usaremos *Ordinal Encoder* esta vez. Por lo tanto, la Tabla 1 transformada quedaría así:

Nombre	Acuático	Patas
León marino	1	2
Elefante	0	4
Leopardo	0	4
Cangrejo	1	8
Medusa	1	0
Gusano	0	0

- **¿Cómo aplicar el algoritmo *K-means*?**

Paso 1

- **Selecciona** centroides iniciales.
- Asumamos que inicialmente se seleccionan los datos 2 y 5 como centroides el **Elefante** y la **Medusa**, ya que de momento queremos obtener solo dos grupos.

Paso 2

Seguidamente se debe asignar cada dato a su centroide mas cercano, para ello se calculan las distancias euclidianas de todos los datos a los centroides seleccionados. Vamos a agregar esos valores de distancia a la tabla para comprender este paso, aunque realmente la tabla no se modifica. ¿Al calcular los valores de distancia por tu cuenta obtuviste los mismos resultados?

Si analizas esta tabla con las distancias seguramente notarás que los animales **Elefante**, **Leopardo** y **Cangrejo** son más cercanos a **Elefante** que a **Medusa**. Por lo tanto, estos tres animales irían en un grupo y los otros tres en el otro grupo, como muestra la tabla siguiente:

Nombre	Acuático	Patas
León marino	1	2
Elefante	0	4
Leopardo	0	4
Cangrejo	1	8
Medusa	1	0
Gusano	0	0

Paso 3

Ya se puede ejecutar el tercer paso. ¿Cómo calculas los nuevos centroides? Efectivamente, calculando el promedio de las variables por grupo. Quedarían así:

- Centroide 1
  Acuático = (0+0+1) / 3 => 0.3
  Patas = (4+4+8) / 3 => 5.3
- Centroide 2
  Acuático = (1+1+0) / 3 => 0.7
  Patas = (2+0+0) / 3 => 0.7

Veamos esos valores como datos de la misma tabla:

Nombre	Acuático	Patas
Centroide 1	0.3	5.3
Centroide 2	0.7	0.7

Si te fijas, los datos de los centroides no se corresponden con los de algún animal. De hecho, ningún animal tiene, por ejemplo, 5.3 patas. ¿Qué pasó entonces? Que precisamente los centroides no son datos iguales a los originales, sino que son unos representantes de éstos.

Paso 4

Aquí termina una iteración de *K-means*.

En el algoritmo, los pasos 2 y 3 se repiten un número determinado de veces, o hasta que ya los centroides no cambien sus valores de una iteración a otra. Por lo tanto, ya estás listo para agrupar datos. Puedes implementar todo este proceso por tu cuenta, o mejor, en el siguiente subtema aprenderás a usar la implementación que de este algoritmo aparece en *Scikit-learn*.

Ejemplo de agrupamiento usando el algoritmo *K-Means*

¿Recuerdas la situación del tema anterior donde el director nacional de ventas de una empresa de biotecnología te pedía agrupar flores Iris? En esa ocasión solo transformaste los datos, pero ya estás listo para utilizar el algortimo *K-Means* para agruparlos.

¡Comencemos!

Transformar datos

Es momento de agruparlos los datos de las flores Iris de la empresa de biotecnología. Veamos cómo lo puedes realizar apoyándote en la librería *Scikit-learn*. Primeramente, vamos a cargar los datos de la base de datos.

```
# Importando librerias
from sklearn import datasets
import numpy as np

# Cargando iris
iris = datasets.load_iris()

# Cargando los datos sin la clase
X = np.array(iris.data)
```

A continuación, agruparás los datos usando *K-means*. El algoritmo aparece en el paquete ***cluster*** y el nombre del objeto es ***Kmeans***. Su principal parámetro es el número de grupos en los que quieres agrupar, que por defecto es **8**. Como esta base de datos tiene tres clases, vamos a intentar 'adivinarlas', así que agrupemos en tres grupos.

Los dos principales métodos del objeto ***Kmeans*** son ***fit( )*** y ***predict ( )***. El primero, entrena el algoritmo sobre los datos, mientras que el segundo realiza el agrupamiento. Ese entrenamiento lo que hace es ejecutar el algoritmo ***K-means*** y almacenar la información de los centroides finales

más otros detalles. El segundo método realmente no hace una predicción, sino que asigna a los datos el número de grupo que le corresponde en el agrupamiento, devolviendo precisamente esas etiquetas.

Analiza el siguiente código donde aparece todo este proceso:

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=3)

kmeans.fit(X)

labels = kmeans.predict(X)
```

Código

Ahora, en la variable ***labels***, aparece la asignación de cada dato a su grupo correspondiente. ¿Cómo puedes saber si este agrupamiento coincide o adivinó las especies? Antes de responder, ploteemos como se relacionan las dos últimas variables en un *scatter*.

El código sería el siguiente:

```
import matplotlib.pyplot as plt

# Ploteamos las últimas dos columnas
plt.scatter(X[:, 2], X[:, 3])
plt.xlabel('Petal length')
plt.ylabel('Petal width')
plt.show()
```

- Resultado del *scatter*

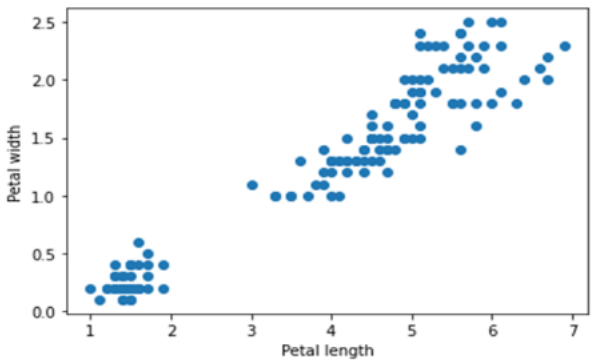


Figura 3. Relación entre las dos últimas variables de la base de datos Iris.

¿Cómo agruparías esos puntos en el plano? ¿Cómo comparas tu agrupamiento con el resultado del algoritmo? Responder estas preguntas tiene un gran componente subjetivo.

Por ejemplo, ¿por qué agrupar en tres grupos si a simple vista se 'nota' que hay dos grupos bien definidos? Estos detalles serán estudiados con mayor profundidad en el siguiente tema del curso.

De momento, una mejor validación puede ser comparar los resultados con las etiquetas originales. Aunque el objetivo de agrupar no es adivinar etiquetas previas, aquí solo se plantea para validar los resultados. Por lo tanto, el código siguiente plotea las dos últimas columnas, de manera que los colores de esos puntos estarán asociados a las etiquetas originales en un caso, y a los grupos obtenidos en el otro.

Veamos ese código:

```
# Aquí necesitamos las especies
# para ver cómo de distribuyen
y = np.array(iris.target)

# Ploteamos las últimas dos columnas en cada caso
fig, (ax0, ax1) = plt.subplots(ncols=2, figsize=(9,4))

# Se asocia el color con las etiquetas originales
ax0.scatter(X[:, 2], X[:, 3], c=y)
ax0.set_title('Clases originales')
ax0.set_xlabel(iris.feature_names[2])
ax0.set_ylabel(iris.feature_names[3])

# Se asocia el color con los grupos obtenidos
ax1.scatter(X[:, 2], X[:, 3], c=labels)
ax1.set_title('Grupos obtenidos')
ax1.set_xlabel(iris.feature_names[2])
ax1.set_ylabel(iris.feature_names[3])

plt.show()
```

El resultado que muestra es:

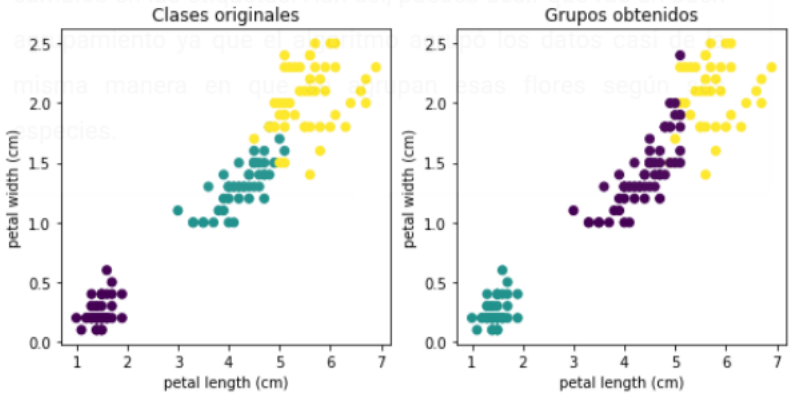


Figura 4. Base de datos Iris. A la izquierda aparecen las clases originales. A la derecha los grupos obtenidos

Como puedes ver en la imagen, los grupos obtenidos por *K-means* son muy parecidos a las etiquetas o clases originales de la base de datos, a pesar de que hubo algunos errores o cambios en las etiquetas. Aun así, puedes decir que fue un buen agrupamiento ya que el algoritmo agrupó los datos casi de la misma manera en que se agrupan esas flores según sus especies.

Probablemente te preguntas, ¿qué habría pasado si agrupamos en dos grupos? ¿Eres capaz de modificar los códigos anteriores para ello? Para hacerlo, el código siguiente te puede servir:

```
y2 = KMeans(n_clusters=2).fit_predict(X)

plt.scatter(X[:, 2], X[:, 3], c=y2)
plt.title('Dos grupos')
plt.xlabel('Petal length')
plt.ylabel('Petal width')
plt.show()
```

Si ahora coloreamos los puntos con estos nuevos dos grupos, el resultado sería:

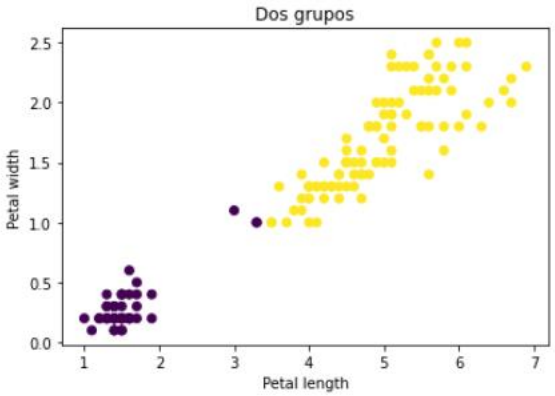


Figura 5. Dos grupos en Iris.

¿Esperabas estos resultados? Hay dos puntos morados mezclados con los puntos amarillos. ¿Por qué crees que *K-means* no agrupó esos puntos con los amarillos? Porque aquí estamos ploteando solo dos variables y, en términos de las cuatro variables, esos puntos están más cerca del grupo morado. A pesar de eso, y obviando las etiquetas originales (o sea, las especies de las flores), este agrupamiento en dos grupos parece tener sentido, de manera que puedes decir al jefe de ventas que alrededor de dos tercios de las flores se parecen entre ellas y son diferentes al tercio restante.

De momento, una mejor validación puede ser comparar los resultados con las etiquetas originales. Aunque el objetivo de agrupar no es adivinar

etiquetas previas, aquí solo se plantea para validar los resultados. Por lo tanto, el código siguiente plotea las dos últimas columnas, de manera que los colores de esos puntos estarán asociados a las etiquetas originales en un caso, y a los grupos obtenidos en el otro.

Efectos de transformar datos

Hasta el momento has agrupados los datos numéricos de las flores sin transformarlos.

¿Qué crees que sucedería si los transformas?

Utilizaremos *StandardScaler* para comparar los resultados.

```
from sklearn.preprocessing import StandardScaler
```

```
X2 = StandardScaler().fit_transform(X)
y2 = KMeans(n_clusters=2).fit_predict(X2)
y3 = KMeans(n_clusters=3).fit_predict(X2)

fig, (ax0, ax1, ax2) = plt.subplots(ncols=3, figsize=(16,4))
ax0.scatter(X[:, 2], X[:, 3], c=y)
ax0.set_title('Clases originales')
ax0.set_xlabel(iris.feature_names[2])
ax0.set_ylabel(iris.feature_names[3])
ax1.scatter(X2[:, 2], X2[:, 3], c=y2)
ax1.set_title('Dos grupos')
ax1.set_xlabel(iris.feature_names[2])
ax1.set_ylabel(iris.feature_names[3])
ax2.scatter(X2[:, 2], X2[:, 3], c=y3)
ax2.set_title('Tres grupos')
ax2.set_xlabel(iris.feature_names[2])
ax2.set_ylabel(iris.feature_names[3])
```

En el código anterior agrupamos los datos en dos y tres grupos, pero primero transformamos los valores numéricos. El resultado se muestra en la imagen siguiente:

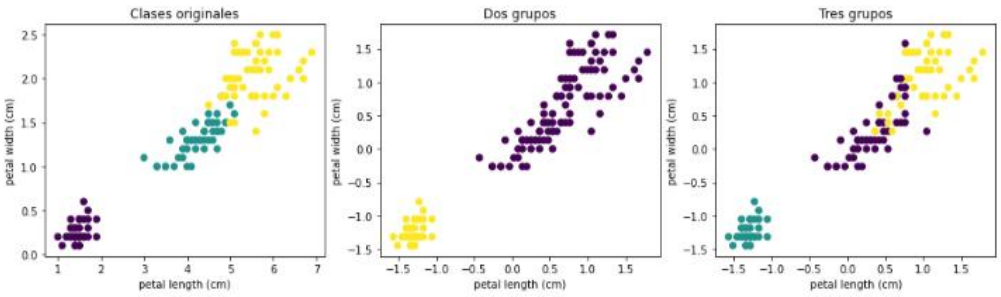


Figura 6. Agrupamientos en Iris después de estandarizar.

En las tres imágenes, la distribución de colores es diferente. La razón es porque el número que se le asigna a cada grupo es aleatorio, por lo tanto, varía de ejecución a ejecución. Teniendo en cuenta eso, compara los resultados con los anteriores.

¿A cuáles conclusiones llegas?

Una posible conclusión es que estandarizar los datos ayuda a obtener dos grupos mejor definidos, pero cuando se agrupa en tres los datos se 'solapan' más.

Este proceso, junto con las interpretaciones que puedes hacer del mismo, será una herramienta muy útil en tu labor como científico de datos. Como ya hemos dicho, agrupar datos te permitirá verlos desde otra perspectiva, simplificarlos, y apoyar la toma de decisiones en tu organización basado en las interpretaciones y conclusiones que obtengas.

Agrupar datos utilizando otros algoritmos de agrupamiento

En el laboratorio químico especializado donde analizaste la calidad de distintas variedades de vino, los líderes necesitan obtener conclusiones distintas a las obtenidas cuando agrupaste los datos usando *K-means*. La razón principal es que tus líderes no están conformes con los aspectos encontrados en los datos.

¡Comencemos!

Introducción

Debido a que este proceso de agrupamiento es no supervisado, no es simple conocer cuándo los resultados obtenidos son adecuados. Para ello, los expertos del tema pueden analizar los resultados, o un resumen de estos, y opinar acerca de las conclusiones obtenidas.

Analiza la siguiente frase de Ulrike von Luxburg et al., 2012: **"En el análisis exploratorio de los datos, los algoritmos de agrupamiento son útiles para descubrir aspectos completamente nuevos de los datos, aspectos que ya se sospecha que existen, o aspectos que no se espera que existan".**

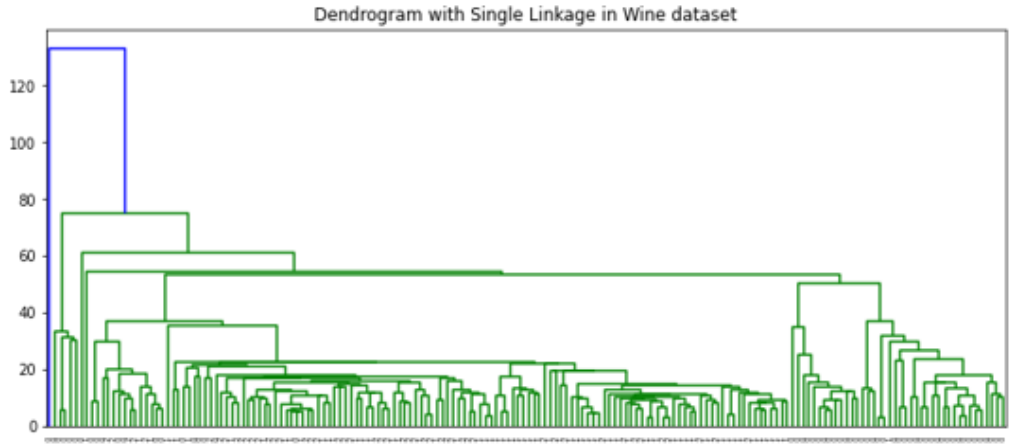
¿Qué opinas de la frase anterior? ¿Consideres que tus líderes exageran al pedir más información, o es un requerimiento 'normal' en determinadas ocasiones? ¿Cómo puedes generar nuevas conclusiones? Si no se está conforme con los aspectos encontrados en los datos, entonces se puede ampliar la exploración usando otros algoritmos de agrupamiento e, incluso, se puede modificar la modelación del problema.

En este subtema mostraremos dos nuevos algoritmos de agrupamiento con definiciones y supuestos distintos a los de *K-means*, y que pueden ser utilizados para encontrar soluciones bastante diferentes a las que encuentra *K-means*.

La motivación es ampliar tu 'set de herramientas' de algoritmos de agrupamiento para que puedas analizar un mismo problema desde

diferentes perspectivas y encuentres conclusiones realmente útiles al finalizar este análisis. Recuerda siempre que el propósito es mejorar la toma de decisiones en tu organización, en tu rol como científico de datos.

Single Linkage (Ros et al., 2019)
Es uno de varios métodos de agrupación jerárquica. Se basa en agrupar grupos de forma ascendente, aglomerativa, de manera que en cada paso va combinando los grupos que contienen pares de elementos muy cercanos, es decir, con distancia mínima. Los resultados pueden representarse con una estructura de árbol o dendograma, para mostrar cómo se fueron asociando los datos en el proceso de agrupamiento.



Un inconveniente de este método es que tiende a producir grupos largos y delgados en los que los elementos cercanos del mismo grupo tienen distancias pequeñas, pero los elementos en los extremos opuestos del mismo grupo pueden tener distancias muy grandes. Esto puede dar lugar a dificultades a la hora de agrupar datos similares. Sin embargo, cuando se requiere datos **conectados** como en redes de amistad en las redes sociales, este método puede ser relevante y encontrar aspectos interesantes en los datos.

- **Aplicación**

A continuación, te presentamos cómo usar este método para agrupar los distintos tipos de vinos. Analiza el código siguiente:

```
from sklearn.cluster import AgglomerativeClustering

wine = datasets.load_wine()
X = np.array(wine.data)
y1 = np.array(wine.target)

linkage = AgglomerativeClustering(linkage="single", n_clusters=3)
y_pred = linkage.fit_predict(X)
```

En el código anterior, lo primero es importar la familia de algoritmos de agrupamiento aglomerativo que están implementados en *Scikit-learn*. Ya conoces como cargar los datos de la base de vinos, por lo tanto, lo siguiente es agruparlos, en este caso en tres grupos.

Nota que para ejecutar *Single-linkage* debes pasar como parámetro en el campo *linkage* el texto **'single'**. Con otros valores obtendríamos otros algoritmos de la familia de los aglomerativos. Para más información al respecto puedes apoyarte en la documentación de *Scikit-learn*.

En este punto es natural que te preguntes ¿si realmente se obtienen resultados interesantes con este algoritmo?

Sobre todo después de leer que es un método que solo funciona de manera adecuada para casos específicos. Veamos un *scatter* de dos de las variables de los datos para crear una idea visual de los resultados del agrupamiento.

```
from sklearn.cluster import AgglomerativeClustering

wine = datasets.load_wine()
X = np.array(wine.data)
y1 = np.array(wine.target)

linkage = AgglomerativeClustering(linkage="single", n_clusters=3
y_pred = linkage.fit_predict(X)

fig, (ax0, ax1) = plt.subplots(ncols=2, figsize=(10,4))
ax0.scatter(X[:, 0], X[:, 1], c=y1)
ax0.set_title('Clases originales')
ax0.set_xlabel('Alcohol')
ax0.set_ylabel('Malic acid')
ax1.scatter(X[:, 0], X[:, 1], c=y_pred)
ax1.set_title('Tres grupos')
ax1.set_xlabel('Alcohol')
ax1.set_ylabel('Malic acid')
plt.show()
```

Después de analizar el código genera la siguiente imagen:

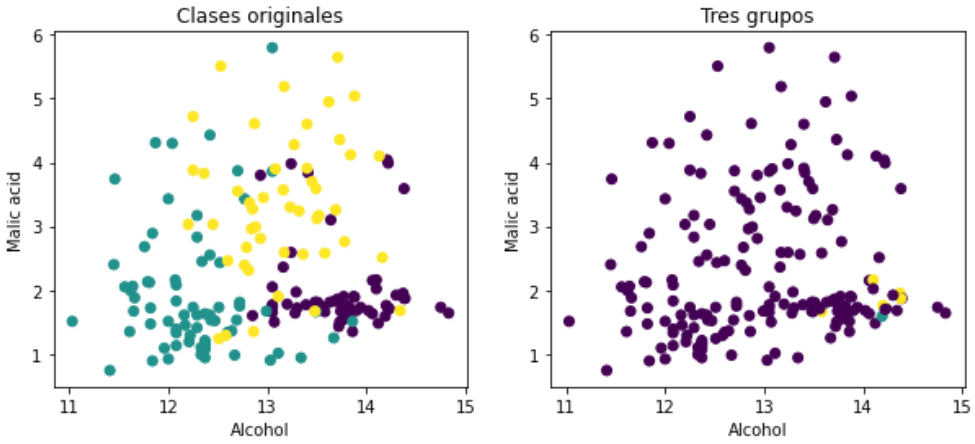


Figura 8. Agrupamiento obtenido por Single-Linkage en Wine.

¿Consideras que es un buen resultado?

Es difícil decir que el resultado obtenido es bueno, en particular porque prácticamente se creó un solo grupo donde están la gran mayoría de los datos. ¿Por qué pasó algo así con este algoritmo? Una razón puede ser que al crear grupos de datos conectados el algoritmo encontró este gran grupo donde cada dato está muy cerca de al menos otro dato y de ahí que podamos decir que todos están conectados.

- Endograma del proceso

Además de lo anterior, puedes mostrar el dendograma del proceso, por si ayuda visualizar la jerarquía a encontrar otras conclusiones. El código siguiente es el indicado para generar el dendograma anterior.

```
import scipy.cluster.hierarchy as shc


plt.figure(figsize=(12,5))
plt.title("Dendrogram with Single Linkage in Wine dataset")
dend = shc.dendrogram(shc.linkage(X, method='single'), labels=y1)
```

Para este ejemplo de situación, el algoritmo *Single-linkage* no parece que pueda ayudar a obtener conclusiones relevantes. Por lo tanto, analicemos

otro algoritmo de agrupamiento, que es más parecido al algoritmo *K-means*.

Spectral Clustering (Huang et al., 2019)
Utiliza los valores propios de la matriz de similitud de los datos para realizar una reducción de dimensionalidad antes de agrupar en menos dimensiones. La matriz de similitud se proporciona como entrada y consiste en una evaluación cuantitativa de la similitud relativa de cada par de datos. Como ya conoces, una función que te puede ayudar a calcular esa similitud entre los datos es la **Distancia Euclidiana**.

A continuación, un ejemplo de cómo usar este algoritmo, en particular en la base de datos *Wine*.

```
from sklearn.cluster import SpectralClustering

wine = datasets.load_wine()
X = np.array(wine.data)
y1 = np.array(wine.target)

spectral = SpectralClustering(n_clusters=3, affinity="nearest_neighbors")
y_pred = spectral.fit_predict(X)
```

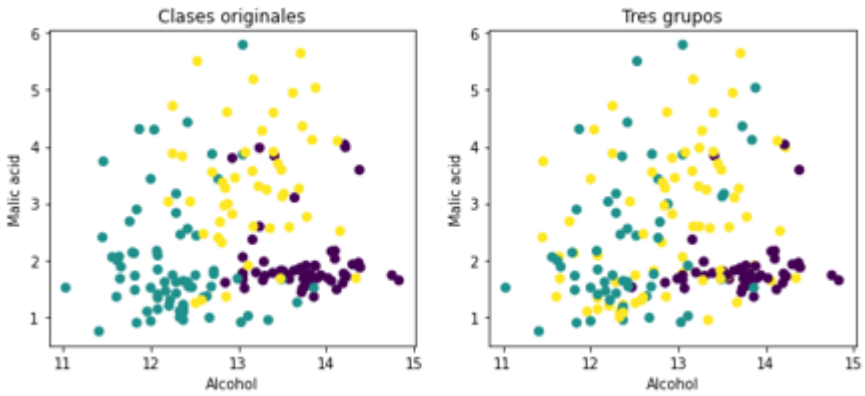


Figura 9. Agrupamiento usando Spectral Clustering.

Para plotear los resultados utilizamos el mismo código que el anterior. Como lo analizaste en la imagen, los resultados del agrupamiento se ven bastante bien en comparación con las clases originales. Sin embargo, en la práctica no contamos con esa referencia, por lo que la interpretación de los

resultados es más difícil. ¿Cómo sabes entonces que obtuviste un buen resultado? ¿Por qué te quedarías con este resultado en lugar del obtenido por *K-means*?

Responder las preguntas anteriores no es fácil. Una manera de intentar responderlas es a través de usar **índices de validación de agrupamientos**. Estos índices los estudiaremos en el próximo tema.

Ideas para llevar

Agrupar es muy importante para la toma de decisiones. Sin embargo, escoger un algoritmo de agrupamiento en particular puede ser una tarea difícil. Es por eso por lo que recomendamos probar y evaluar varios algoritmos de agrupamiento para una sola tarea, de manera que puedas analizar los resultados desde distintas perspectivas.

Otras recomendaciones se enuncian a continuación:

- Cuando agrupes usando distintos algoritmos para obtener resultados diversos, asegúrate de escoger los resultados que más se acercan a los requerimientos del problema.
- No pases por alto que plotear algunas variables ayudan considerablemente a escoger unos resultados sobre otros. No obstante, otros métodos de visualización deben siempre considerarse.
- Antes de agrupar, intenta conocer más acerca del problema que se requiere resolver. Los expertos pueden ayudarte en esta interpretación. Una vez que conozcas los detalles, piensa en cuál estrategia de agrupamiento es la adecuada. Finalmente, intenta encontrar algunas características en los datos que te ayuden a validar que el proceso realizado fue correcto.

Material de apoyo

Bibliografía

Los contenidos de esta lección están basados en la siguiente bibliografía:

- Huang, D., Wang, C. D., Wu, J. S., Lai, J. H., & Kwoh, C. K. (2019). Ultra-scalable spectral clustering and ensemble clustering. *IEEE Transactions on Knowledge and Data Engineering*, *32*(6), 1212-1226.
- Patel, S. P., & Upadhyay, S. H. (2020). Euclidean distance based feature ranking and subset selection for bearing fault diagnosis. *Expert Systems with Applications*, *154*, 113400.
- Ros, F., & Guillaume, S. (2019). A hierarchical clustering algorithm and an improvement of the single linkage criterion to deal with noise. *Expert Systems with Applications*, *128*, 96-108.
- Sinaga, K. P., & Yang, M. S. (2020). Unsupervised K-means clustering algorithm. *IEEE Access*, *8*, 80716-80727.
- Von Luxburg, U., Williamson, R. C., & Guyon, I. (2012, June). Clustering: Science or art?. In *Proceedings of ICML workshop on unsupervised and transfer learning* (pp. 65-79). JMLR Workshop and Conference Proceedings.