DETECCIÓN DE CLUSTERS SIGNIFICATIVOS EN DATOS RELACIONADOS CON FRUTAS

APRENDIZAJE AUTOMÁTICO I

PROYECTO PRÁCTICO 1

Ángel Romero Huici

Alejandro Pastor Membrado

1. Resumen

En este proyecto práctico, se nos ha planteado un problema relativo a la industria agroalimentaria. Se disponía de unos datos relativos a distintas piezas de fruta y se deseaba obtener un modelo de clustering que permitiese agrupar estas frutas.

Para ello, hemos llevado a cabo un proceso de limpieza de los datos, ya que existían errores y datos nulos, fruto del proceso de recolección de los datos. También hemos transformado todas las variables a datos numéricos, para que nuestros algoritmos fuesen capaces de trabajar con los datos de forma adecuada y hemos puesto todos los datos en una escala con media 0 y desviación típica 1.

A continuación, hemos llevado a cabo tanto un proceso de clustering jerárquico, en el que hemos probado con distintos tipos de distancias, para llegar a la conclusión de que existían tres grupos principales.

Seguidamente, hemos realizado un clustering particional mediante el método de los KMeans y, tras probar con distinto número de clusters, hemos vuelto a llegar la conclusión de que la mejor forma de agrupar los datos era en tres clusters diferentes.

Por lo tanto, tras realizar este análisis, podemos concluir que la mejor forma de agrupar estos datos sobre frutas es en tres grupos diferentes. Conocer esto puede sernos de ayuda en el futuro en caso de que deseemos predecir a cuál de los tres grupos pertenecerá una nueva fruta que llegue al almacén.

2. Introducción

Un almacén de fruta dispone de una serie de datos acerca de 180 piezas de frutas, y desean conocer la mejor forma de separar estas frutas en grupos, de acuerdo con sus características. Para ello, decidimos emplear técnicas de aprendizaje no supervisado, para resolver este problema.

El aprendizaje no supervisado es un método de aprendizaje automático donde un modelo se ajusta a las observaciones. Se distingue del aprendizaje supervisado por el hecho de que no hay un conocimiento a priori.

Concretamente, emplearemos técnicas para tratar de inferir los datos faltantes o erróneos y trataremos de crear agrupaciones o clusters con el objetivo de clasificar estas frutas. En caso de conseguirlo, esta información puede ser muy útil para el dueño del almacén, pues puede emplearla para vender cada uno de los tipos de frutas a un precio distinto, a un cliente distinto... así como entrenar a un modelo posterior que sea capaz de clasificar las nuevas piezas de fruta que lleguen al almacén.

Los datos se componen de una serie de frutas (cada una de las cuales se encuentra en una fila distinta), con 5 características para cada una (cada una de las cuales aparece en una columna diferente). Las características son el peso, la longitud, la profundidad, el coeficiente de simetría y el tamaño de una grieta presente en la fruta.

3. Metodología aplicada

Para resolver este problema, hemos aplicado una metodología que podríamos dividir en tres fases: Preprocesado, Agrupación jerárquica y Agrupación particional.

PREPROCESADO DE LOS DATOS

En primer lugar, cargamos los datos para saber sus características:

	weight	length	width	regularity	cleft
0	1.205	4.603915	2.847	5.691634	Small
1	1.726	5.978000	3.594	4.539000	Large
2	1.126	4.516534	2.710	5.965993	Average
3	1.755	5.791000	3.690	5.366000	Large
4	1.238	4.666888	2.989	6.153947	Small

	weight	length	width	regularity
count	168.000000	156.000000	163.000000	170.000000
mean	2.042863	5.382412	3.244865	3.989641
std	7.530563	0.696904	0.378125	1.629388
min	1.059000	4.236811	2.630000	0.765100
25%	1.218000	4.656544	2.911000	2.732250
50%	1.431000	5.512000	3.232000	3.973123
75%	1.682500	5.979250	3.538500	5.159713
max	99.000000	6.666000	4.032000	8.986146

Podemos observar que nuestros datos incluyen información sobre 180 frutas y 5 variables. Además, existen algunos datos nulos y las columnas son de distinto tipo (weight, length, width y regularity son float, mientras que cleft es string). El enunciado ya nos avisa de que los datos de cada una de las variables se encuentran en diferentes escalas.

Por tanto, los datos no se encuentran preparados para que podamos utilizar los algoritmos que conocemos de forma que obtengan la mayor precisión posible. Para conseguirlo, debemos procesar estos datos, tratando los datos nulos que aparecen, variables categóricas, normalizar...

Comprobamos si existe alguna fila completamente vacía, lo cual no ocurría, por lo que pasamos a comprobar el número de datos nulos que había en cada una de ellas.

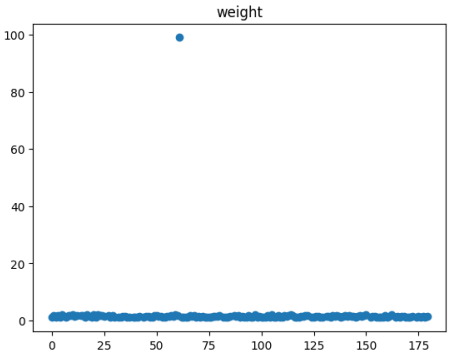
```
weight        12
length        24
width         17
regularity    10
cleft          0
dtype: int64
```

Vemos que hay algunos datos faltantes en 4 columnas, por lo que tenemos que decidir qué hacer con ellos. Como el enunciado no indica que camino seguir, vamos a probar varias maneras de resolver este problema a ver cuál es la óptima. Como en la columna con variable categórica “cleft” no aparecía ningún dato nulo, nos centramos en resolver los casos de las otras columnas con datos numéricos.

No es posible asignar un valor como 0 u otro número (pues estas características dependen de cada una de las frutas). Por tanto, tenemos que decidir si eliminamos las filas con datos nulos o estimamos los datos faltantes.

Antes de estimar, comprobamos de forma gráfica si existen outliers o datos atípicos que puedan afectar a nuestra estimación. De esta forma, conseguimos más precisión en nuestro modelo.

El resultado es que existe un valor atípico en la columna “weight”, con valor de 99. Como resulta muy complicado de creer que una fruta pese 99kg en comparación con las otras, consideramos que ha sido un error al tomar los datos y eliminamos esa fila.



Ahora que en ninguna de las columnas encontramos valores muy atípicos, procedemos a estimar los datos faltantes. Para ello probamos con diferentes opciones (eliminar las columnas en las que faltaban datos, estimarlos sustituyendo por la media, la mediana o el valor más frecuente y emplear el método del KNN Imputer).

- Eliminando las columnas con datos nulos

	weight	length	width	regularity
count	121.000000	121.000000	121.000000	121.000000
mean	1.461760	5.353520	3.231066	4.051832
std	0.279499	0.682332	0.362930	1.711822
min	1.074000	4.236811	2.641000	0.765100
25%	1.221000	4.658048	2.911000	2.704000
50%	1.433000	5.438000	3.242000	4.004000
75%	1.677000	5.877000	3.505000	5.415806
max	2.097000	6.581000	3.991000	8.986146

- Sustituyendo por media, mediana y valor más frecuente

	weight	length	width	regularity
count	179.000000	179.000000	179.000000	179.000000
mean	1.462281	5.377312	3.242358	3.998793
std	0.275277	0.647600	0.359436	1.583399
min	1.059000	4.236811	2.630000	0.765100
25%	1.221500	4.726122	2.958000	2.812500
50%	1.449000	5.377312	3.242358	3.998793
75%	1.642500	5.863500	3.495500	5.080198
max	2.097000	6.666000	4.032000	8.986146

	weight	length	width	regularity
count	179.000000	179.000000	179.000000	179.000000
mean	1.460050	5.394298	3.241327	3.997463
std	0.275404	0.649045	0.359450	1.583408
min	1.059000	4.236811	2.630000	0.765100
25%	1.221500	4.726122	2.958000	2.812500
50%	1.429000	5.504000	3.231500	3.975000
75%	1.642500	5.863500	3.495500	5.080198
max	2.097000	6.666000	4.032000	8.986146

	weight	length	width	regularity
count	179.000000	179.000000	179.000000	179.000000
mean	1.439536	5.379683	3.188475	3.894335
std	0.288128	0.647628	0.396254	1.640911
min	1.059000	4.236811	2.630000	0.765100
25%	1.183500	4.726122	2.821000	2.613500
50%	1.403000	5.395000	3.155000	3.747000
75%	1.642500	5.863500	3.495500	5.080198
max	2.097000	6.666000	4.032000	8.986146

- Aplicando el método de KNN Imputer

	weight	length	width	regularity
count	179.000000	179.000000	179.000000	179.000000
mean	1.473894	5.390063	3.247810	4.024169
std	0.289274	0.699509	0.377199	1.598814
min	1.059000	4.236811	2.630000	0.765100
25%	1.219000	4.647533	2.926000	2.800422
50%	1.437000	5.543000	3.232000	4.018000
75%	1.710000	5.985000	3.538500	5.146425
max	2.097000	6.666000	4.032000	8.986146

Consideramos que como las frutas tienden a parecerse (tiene sentido que las de mayor longitud tengan un mayor peso), finalmente, emplearemos el método de KNN Imputer para estimar los datos nulos, pues opinamos que es más preciso para resolver este problema.

A continuación, tenemos que convertir la variable categórica cleft a dato numérico para que nuestros algoritmos puedan procesarlos correctamente. Podemos utilizar tanto el método del OrdinalEncoder como el OneHotEncoder pero, en este caso, el uso del OrdinalEncoder parece más adecuado. Esto es debido a que la categoría cleft tiene valores con cierto orden (de menor a mayor prominencia) y que el uso del OneHotEncoder crearía más columnas de categorías aumentando la complejidad de cómputo.

	weight	length	width	regularity	cleft
0	1.205	4.603915	2.847	5.691634	1.0
1	1.726	5.978000	3.594	4.539000	3.0
2	1.126	4.516534	2.710	5.965993	2.0
3	1.755	5.791000	3.690	5.366000	3.0
4	1.238	4.666888	2.989	6.153947	1.0

Un breve análisis de los pros y las contras de aplicar cada uno de los métodos a esta variable sería:

ORDINAL ENCODER:

PROS

- Preserva la información ordinal.
- Reduce el número de columnas en comparación con OneHot Encoder.
- Es más eficiente en términos de memoria.

CONTRAS

- Asume que los intervalos entre los valores codificados son iguales para todos los casos, lo cual puede no ser cierto (puede haber más similitud entre “Very small” y “Small”, que entre “Small” y “Average”).

ONEHOT ENCODER:

PROS

- Preserva toda la información.
- Sin suposiciones entre intervalos.

CONTRAS

- Aumenta la dimensionalidad.
- Requiere más uso de memoria.

Finalmente, tenemos que escalar los datos. Para ello, podemos emplear StandardScaler o MinMaxScaler. En este caso, emplearemos StandardScaler. Así, los datos pasan a tener media 0 y desviación típica 1.

Para el resto del análisis hemos utilizado los conjuntos de datos con y sin escalado para comparar diferencias.

	weight	length	width	regularity	cleft
0	-0.932156	-1.127009	-1.065576	1.045865	-0.701189
1	0.873959	0.842858	0.920365	0.322912	0.717036
2	-1.206020	-1.252277	-1.429798	1.217947	0.007923
3	0.974491	0.574778	1.175587	0.841621	0.717036
4	-0.817757	-1.036732	-0.688061	1.335835	-0.701189

AGRUPACIÓN JERÁRQUICA

Una vez preprocesados los datos, ya podemos emplear nuestros algoritmos de clasificación en ellos para obtener conclusiones.

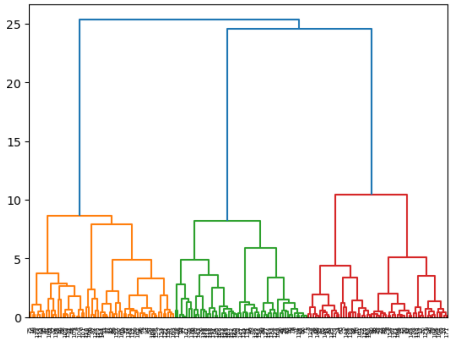
Utilizamos una estrategia de clustering de abajo hacia arriba mediante el clustering aglomerativo. Para dibujar los dendogramas, empleamos una función proporcionada en clase.

```
from scipy.cluster.hierarchy import linkage
from scipy.cluster.hierarchy import dendrogram
def plot_dendrogram(model, **kwargs):
    # Create linkage matrix and then plot the dendrogram

    # create the counts of samples under each node
    counts = np.zeros(model.children_.shape[0])
    n_samples = len(model.labels_)
    for i, merge in enumerate(model.children_):
        current_count = 0
        for child_idx in merge:
            if child_idx < n_samples:
                current_count += 1  # leaf node
            else:
                current_count += counts[child_idx - n_samples]
        counts[i] = current_count

    linkage_matrix = np.column_stack(
        [model.children_, model.distances_, counts]
    ).astype(float)

    # Plot the corresponding dendrogram
    dendrogram(linkage_matrix, **kwargs)
```

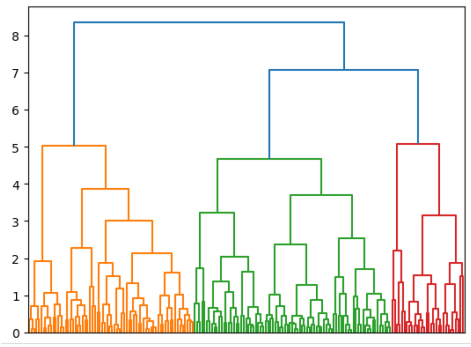
Para poder analizar el efecto que tiene emplear distintas medidas de disimilaridad, realizamos el clustering empleando distintas métricas y “linkage”:

DISTANCIA EUCLÍDEA

- LINKAGE = “WARD”

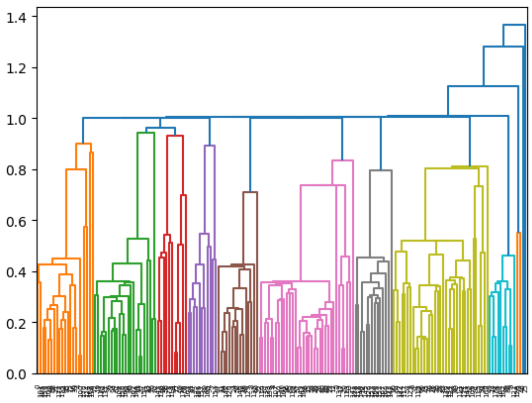


- LINKAGE = “COMPLETE”

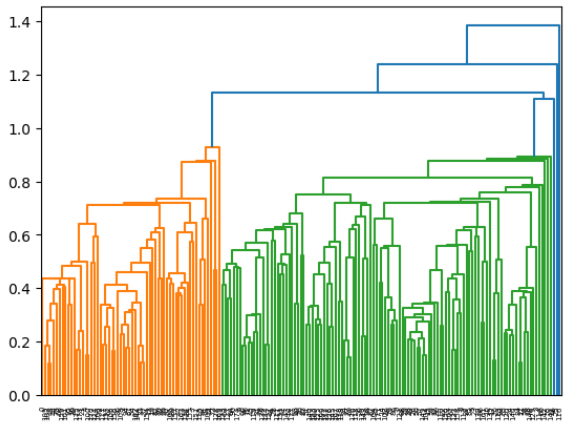


Observamos que parecen existir tres agrupaciones. Ahora comprobamos si se mantienen al variar las condiciones.

Si empleamos linkage = “single” sin escalar los datos, aparecen clusters muy disimilares entre ellos, lo cual no es ideal para obtener un modelo adecuado.

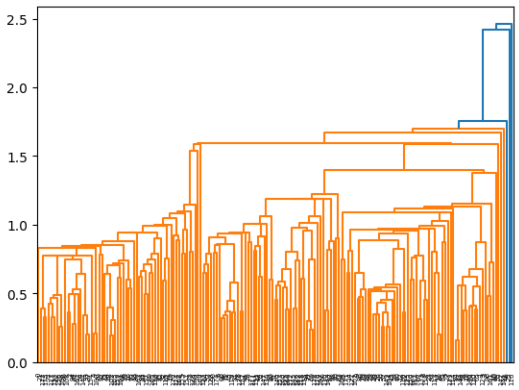


Realizando el procedimiento anterior con datos escalados, obtenemos un resultado bastante similar. Sin embargo, los grupos parecen ser más compactos.

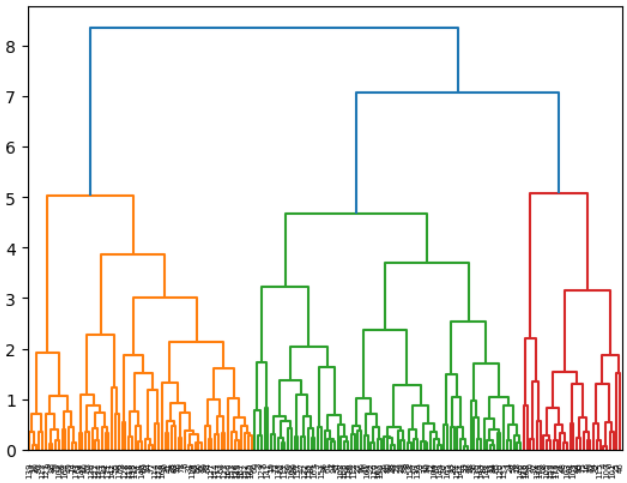


DISTANCIA MANHATTAN

Si realizamos los pasos anteriores, pero esta vez empleando la distancia manhattan, tenemos que, de nuevo, linkage = “single” nos ofrece un resultado del que podemos obtener muy poca información.



Por el contrario, si empleamos linkage = “complete”, vuelven a aparecer tres clusters bastante diferenciados.



Por lo tanto, como conclusión a este proceso de agrupación jerárquica, tenemos que el modelo ideal sería aquel en el que tengamos 3 clusters para agrupar todos los datos.

A continuación, realizaremos otro proceso de agrupación particional y compararemos los resultados obtenidos en ambos procesos para tomar una decisión.

AGRUPACIÓN PARTICIONAL

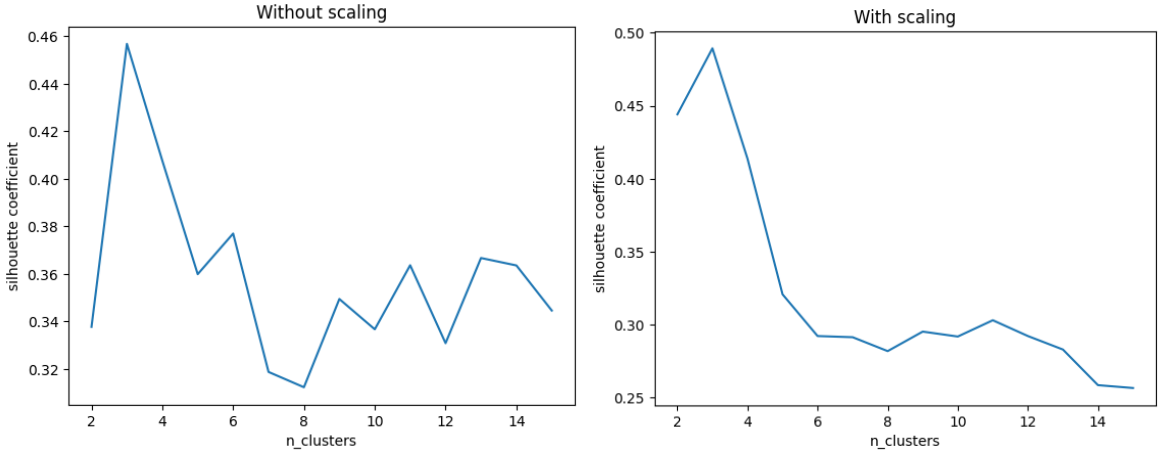
Empezamos realizando el clustering mediante Kmeans con 5 y 10 clusters sobre tanto los datos escalados como los no escalados. A partir de estas agrupaciones, obtenemos la media de su silhouette score para conocer como de acertados son estos clusters.

```
Silhouette score

      with no scale, n_clusters= 5: 0.35992393455764066

      with scale, n_clusters= 5: 0.3206976282937911
```

```
with no scale, n_clusters= 10: 0.3367286979751991

with scale, n_clusters= 10: 0.29176491071381555
```

Podemos observar que no obtenemos valores muy elevados y además que al escalar los datos se obtiene un peor resultado. Analizando independientemente los valores del agrupado con los datos escalados frente a los no escalados, vemos que al escalar los datos obtenemos más valores bajos de silhouette score.

Con estos resultados, el número de clusters elegidos no parece ser adecuado por lo que vamos a analizar la media del silhouette score de distintos clusterings para ver con cuantos clusters alcanzamos un pico.



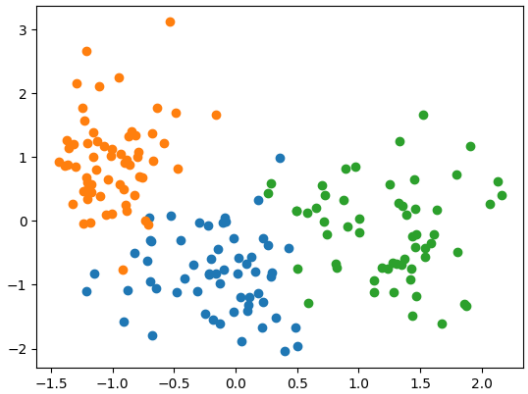
Tanto para los datos escalados como para los no escalados, alzanzamos un pico en 3 clusters. Esto apoya nuestra conclusión obtenida con la agrupación jerárquica. Además, en este caso el silhouette score medio es mayor al escalar los datos que al no hacerlo.

```
Silhouette score

    with no scale, n_clusters= 3: 0.4568315719140889

    with scale, n_clusters= 3: 0.4894016857544144
```

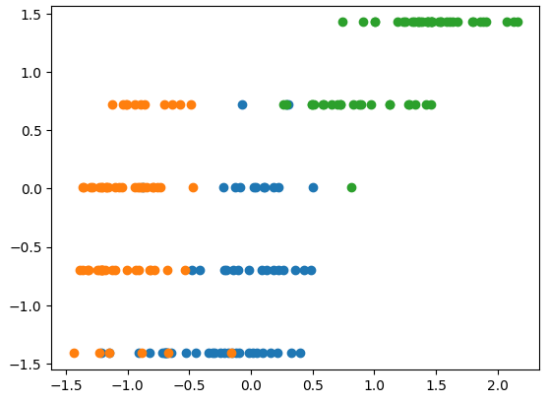
4. Resultados

Tomando el clustering con 3 clusters sobre los datos escalados como nuestro clustering final, vamos a observar cómo estamos separando el conjunto de datos inicial.

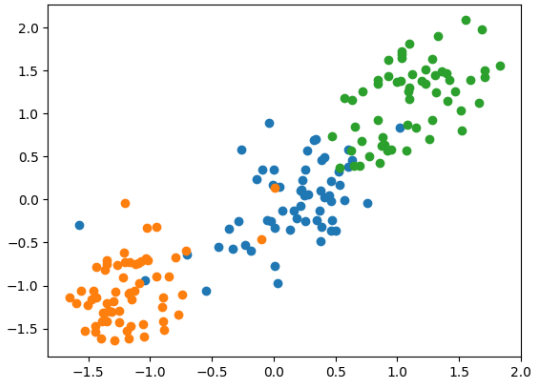
‘weight’ frente a ‘regularity’



‘weight’ frente a ‘cleft’



‘length’ frente a ‘width’



5. Análisis de los resultados

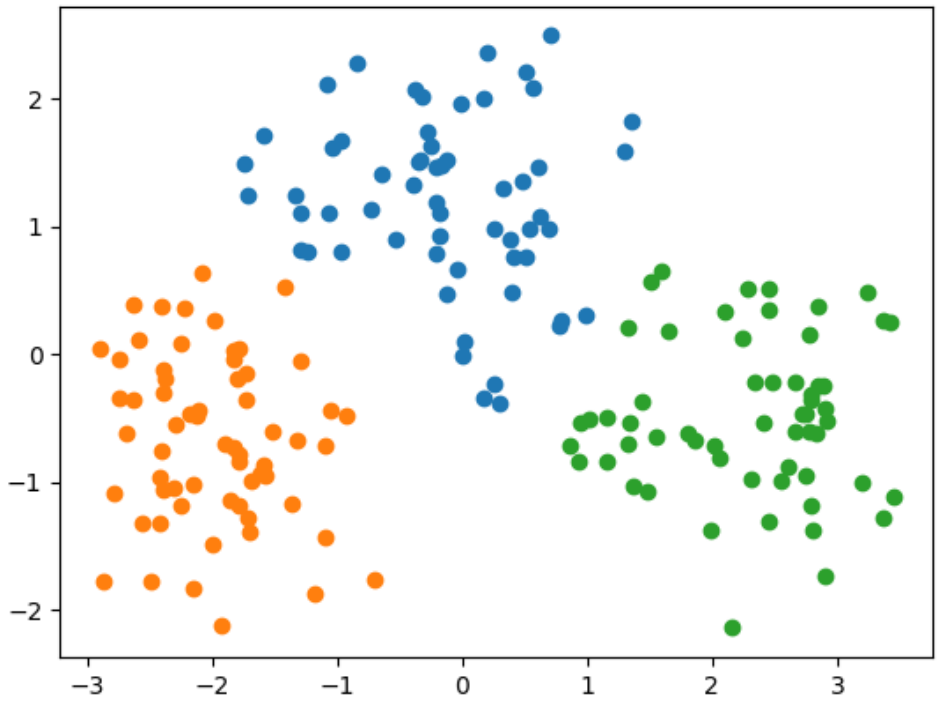
Observando las gráficas, si el clustering realizado es correcto los 3 clusters (diferenciados por sus colores) tienen características distintas entre ellos.

Por un lado, podemos ver que un cluster presenta un peso bajo, una importante regularidad y una prominencia del ‘cleft’ distinta entre sus elementos. Además, las frutas de este grupo poseen una longitud y anchura baja. (Grupo naranja en las gráficas anteriores)

En segundo lugar, tenemos un grupo de frutas con un peso medio-bajo que son bastante irregulares y tienen una prominencia del 'cleft’ por lo general inferior a la media. Este grupo de frutas son de longitud y anchura medias. (Grupo azul en las gráficas anteriores)

Por último, el tercer grupo lo conforman frutas con elevado peso y con bastante varianza en su regularidad, así como un ‘cleft’ muy prominente. Estas frutas tienen además longitud y anchura elevadas. (Grupo verde en las gráficas anteriores)

Para apoyarnos en nuestros resultados, utilizamos un análisis de componentes principales para transformar nuestro conjunto de datos a otro con dos dimensiones. A partir de este nuevo conjunto de datos, realizamos el clustering mediante KMeans con 3 clusters. Dibujamos en una gráfica este nuevo conjunto de datos utilizando las etiquietas proporcionadas mediante el clustering de KMeans como colores en los puntos de la gráfica.



Observando la gráfica, parece que nuestra elección de 3 clusters es acertada puesto que la diferenciación en 3 grupos es adecuada para la gráfica proporcionada. Esto apoya nuestra idea de diferenciar las frutas en 3 grupos con sus respectivas características:

- **Primero**: peso bajo, regular, 'cleft' dispar, longitud y anchura bajas.

- **Segundo**: peso bajo-medio, irregular, baja prominencia de 'cleft', longitud y anchura medias.

- **Tercero**: peso elevado, regularidad inestable, alta prominencia de 'cleft', longitud y anchura elevadas.

6. Conclusiones

Durante la realización de esta práctica, hemos podido emplear los métodos de preproceso y clustering de datos de todo tipo vistos en clase y en los notebooks aplicándolos a un caso concreto. En el proceso, hemos visto las distintas formas posibles de abordar los problemas presentados (datos atípicos, datos vacíos, codificación, escalado, formas de clustering) valorando los pros y los contras de cada método.

También nos hemos dado cuenta de la importancia que tiene cada conjunto de datos en específico ya que la forma de abordar los problemas puede cambiar drásticamente. Por ejemplo, si eliminásemos la columna 'cleft', el dataset sería distinto y el clustering ideal en ese caso sería de 2 clusters.

Como primera práctica de aprendizaje automático, si bien inicialmente trabajar con un conjunto de datos desconocido puede ser complicado, a medida que hemos ido avanzando y aprendiendo, también con ayuda de los gráficos, llegar a una conclusión ha sido satisfactorio.

7. Referencias

Bojan Mihaljević - pdfs con las diapositivas proporcionadas durante el transcurso de la asignatura.

Bojan Mihaljević - La función *plot_dendrogram* del notebook compartido 'Hierarchical clustering'.

https://scikit-learn.org/stable/ - documentación oficial de scikitlearn.

ENLACE A LA PRESENTACIÓN:
https://drive.google.com/file/d/1ziQRO7OsQUHepoKgDGvXW6WNvBq765jy/view?usp=drive_link