Práctica 3:
Clustering

Curso 2020/2021

PEDRO MANUEL FLORES CRESPO

Tratamiento Inteligente de Datos

Máster en Ingeniería Informática

Índice

1. **Introducción** 2

2. ***Clustering* jerárquico** 2

3. ***Outliers*** 3

4. **K-medias** 4

5. **Reducción de la dimensionalidad** 5

6. **DBSCAN** 6

7. **Conclusiones** 6

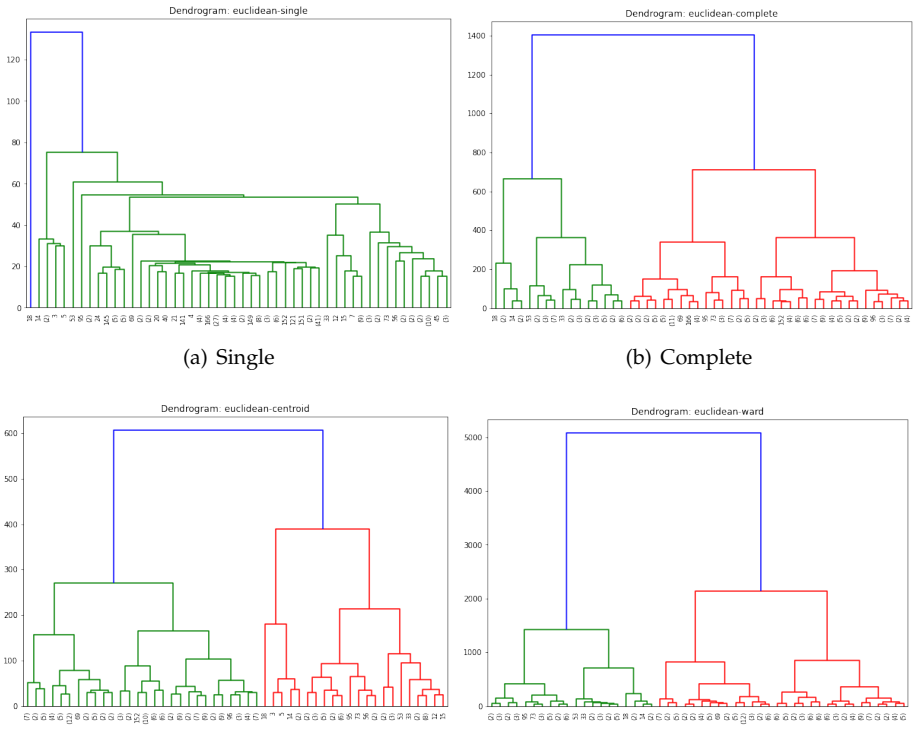
1. Introducción

El objetivo de esta práctica es aplicar una serie de algoritmos de *clustering* al conjunto de datos que se encuentran en el archivo `wine.data`. Este archivo contiene información sobre vinos procedentes de tres clases diferentes. Entre los atributos que disponemos podemos mencionar el nivel de alcohol, intensidad de color, matiz, ácido málico, etc. Tenemos un total de 13 atributos diferentes. Así durante el desarrollo de la práctica aplicaremos algoritmos de *clustering* jerárquico, detectaremos valores atípicos (*outliers*) y aplicaremos otros algoritmos como k-medias o DBSCAN y veremos si se han podido detectar correctamente las tres clases iniciales. Es claro por ello, que trabajaremos en todo momento con el conjunto de datos sin la característica `clase`.

Esta práctica se ha desarrollado en `Python`. Junto a este documento se adjunto un cuaderno de *Google Colaboratory* en el que se han ejecutado cada uno de los ejemplos.

2. *Clustering* jerárquico

Para este apartado vamos a usar la librería `scipy`. Más concretamente, la función `dendrogram` [1] que nos dibuja el dendrograma deseado.Como vemos en la documentación, a esta función le debemos de pasar un parámetro de tipo `linkage` [2]. En él, es donde indicamos tanto la métrica a usar (euclídea, minkowski, manhattan, . . .)[3] como el `linkage-method`, para medir la distancia este clústeres (*single*, *avarage*, *ward*). Veamos los distintos resultados que se han obtenido en la figura 1. Para todos ellos se ha considerado al distancia euclídea. A modo aclaratorio, como hay una gran cantidad de muestras, se ha indicado que solo se muestren lo últimos 50 clústeres unidos para que sea más fácil presentar los dendrogramas.



(a) Single (b) Complete

(c) Centroid (d) Ward

Figura 1: Dendrogramas obtenidos según el `linkage-method`. En todos ellos se ha considerado la métrica euclídea.

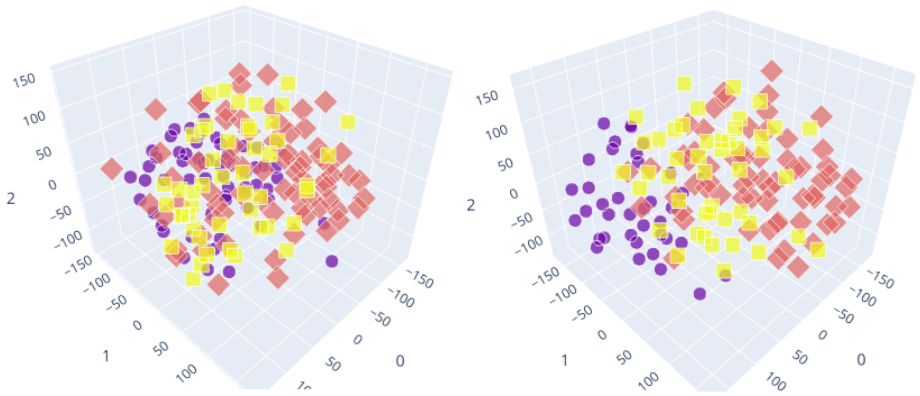
Vemos que puede haber entre dos a cuatro clústeres diferentes. El mínimo de dos clústeres bien

diferenciados es común en todos los casos pero el máximo de clústeres oscila un poco dependiendo del método elegido.

3. *Outliers*

Para detectar valores atípicos, podríamos fijarnos en el dendrograma y buscar aquellas observaciones que se hallan "juntado" más tarde. Por ejemplo, vemos sobre todo que usando `single` la instancia 18 se ha unido muy tarde al resto y sería un candidato a *outlier*. Sin embargo, se va a utilizar una estrategia diferente, más concretamente la que aparece en [13]. En este enlace se habla sobre detección de valores atípicos usando `Python` y propone diferentes estrategias. La que más nos ha llamado la atención se encuentra al final de la página donde usa la librería `PyOD` [14] que implementa varios algoritmos para *outliers*. Uno que recomienda es `k-Nearest-Neighbors` (`kNN`) que se basa en la proximidad entre instancias. Si lo aplicamos a nuestro ejemplo con un valor de *contamination* $= 0{,}2$ obtenemos que ha detectado un total de 30 valores atípicos (2, 3, 5, 10, 12, 13, 14, 17, 18, 20, 26, 28, 31, 33, 35, 40, 47, 50, 52, 53, 56, 69, 73, 78, 80, 93, 95, 96, 141, 150).

Para poder visualizar el conjunto antes y después de eliminar estos valores lo que vamos a hacer en primer lugar es una reducción de la dimensión del conjunto de características. Como tenemos 13 columnas, las vamos a dejar en 3 para poder dibujar el gráfico en 3 dimensiones. Para esta reducción se ha usado el procedimiento `t-distributed Stochastic Neighbor Embedding` o `TSNE` [11]. Por su parte, para obtener los gráficos en 3D se ha usado `Plotly Express` [12] que además nos permite obtener **gráficos interactivos** (si se ejecutan en el cuaderno proporcionado se puede cambiar la vista, acercar la imagen, seleccionar por símbolo . . .). Los gráficos obtenidos según la clase se muestran en la figura 2. Destacamos que no se ha tenido en cuenta la característica `clase` a la hora de aplicar `kNN`, solo se ha usado para dibujar el gráfico.

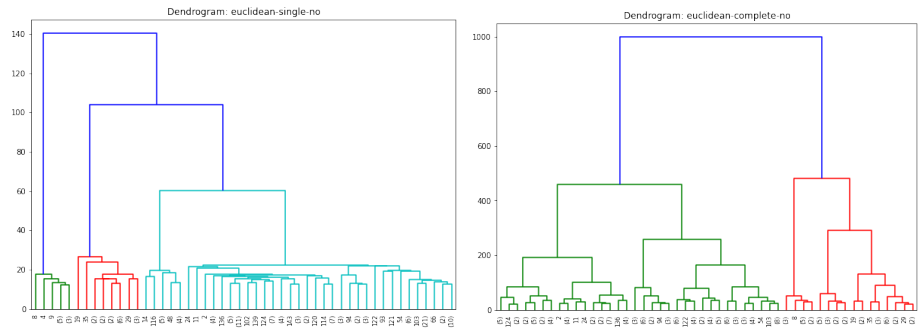


(a) Con *outliers* (b) Sin *outliers*

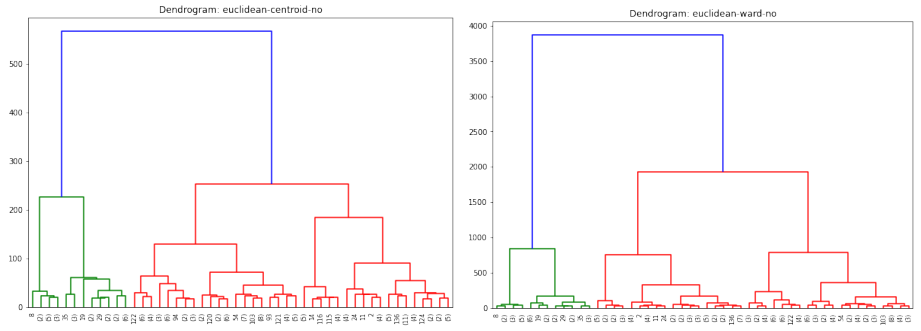
Figura 2: Conjunto de datos antes y después de eliminar valores atípicos según la clase.

Vemos cómo en el conjunto de datos original tiene las clases (cada una con color y forma diferente) más "mezcladas" mientras que si quitamos los *outliers* se pueden diferenciar mejor. Es claro que podemos obtener este gráfico porque conocemos de antemano las clases de cada una de las instancias. Así, lo que vamos a hacer ahora es volver a obtener los dendrogramas anteriores con el nuevo conjunto de instancias y veremos cuál podría ser el número de clústeres adecuado.

En la figura 3 vemos cómo sobre todo ha variado el dendrogama usando `single`. Presenta una clara diferencia para tres clústeres. En el resto se podría considerar dos o tres clústeres según el caso.



(a) Single (b) Complete

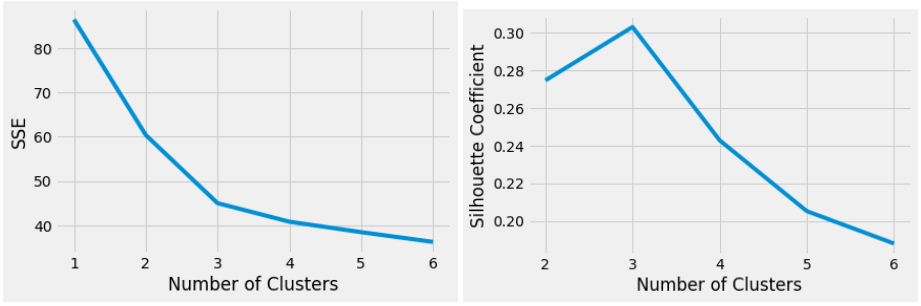


(c) Centroid (d) Ward

Figura 3: Dendrogramas obtenidos según el `linkage-method`. Se han eliminados los *outliers* En todos ellos se ha considerado la métrica euclídea.

4. K-medias

Vamos a aplicar ahora el algoritmo k-medias al conjunto de datos sin valores atípicos (previamente normalizados [10]). Se va a la implementación disponible en la biblioteca `sklearn` [7]. En la sección anterior, vimos cómo podríamos considerar entre 2 y 3 clústeres diferentes. Para poder elegir un número, utilizamos el método del codo y el coeficiente de silueta [6].

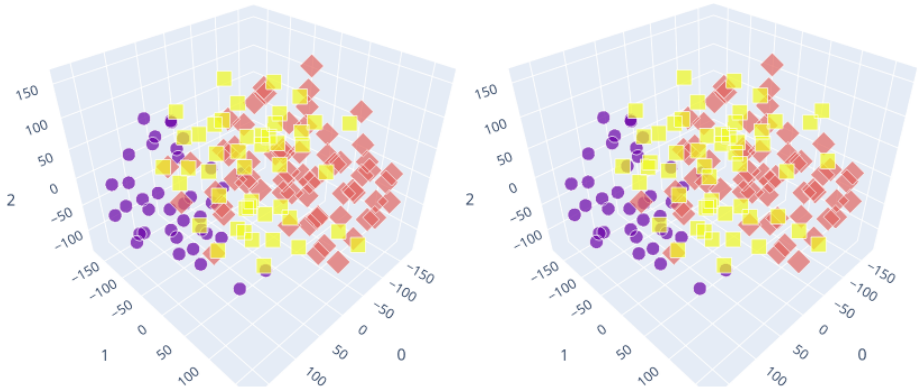


(a) Método del codo (b) Coeficiente silueta

Figura 4: Método del codo y coeficiente de silueta para obtener número de clústeres en k-medias.

Si observamos los resultados de ambos métodos en la figura 4, vemos que un número de clústeres 3 es adecuado. En el caso del método del codo que puede parecer más confuso se ha usado una detección automática mediante la librería `kneed` [15]. Aplicamos entonces el algoritmo k-medias para 3 clústeres y se han obtenido los resultados de la figura 5 y la tabla 1.

Vemos que el algoritmo ha sido capaz de identificar correctamente gran parte de las instancias obteniendo una exactitud del 95 %.



(a) Clases originales (b) Clases según k-medias

Figura 5: Comparación de las clases reales y las obtenidas mediante el algoritmo k-medias para 3 clústeres

clase/pred	1	2	3
1	38	0	0
2	1	57	6
3	0	0	46

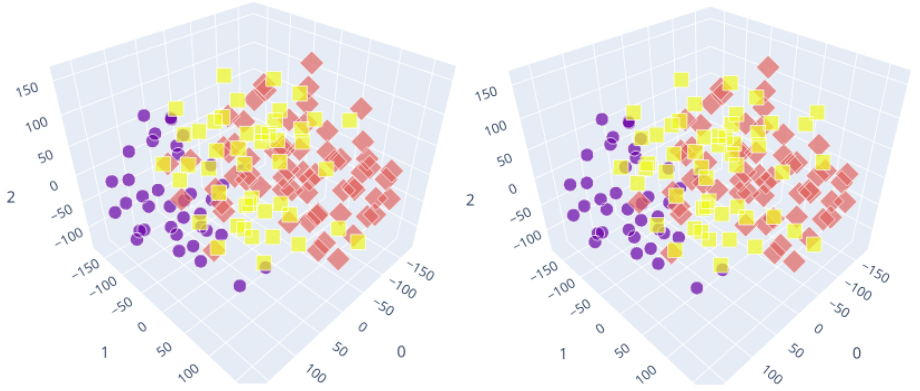
clase	*precision*	*recall*	*f1-score*
1	0.97	1.00	0.99
2	1.00	0.89	0.94
3	0.88	1.00	0.94

Exactitud	0.95

Tabla 1: Resultados del algoritmo k-medias.

5. Reducción de la dimensionalidad

Ahora, vamos a repetir el procedimiento pero aplicando una reducción de la dimensión. En nuestro caso, hemos utilizado PCA. Los resultados están la figura 6 y la tabla 2.



(a) Clases originales (b) Clases según k-medias

Figura 6: Comparación de las clases reales y las obtenidas mediante el algoritmo k-medias para 3 clústeres previa reducción de la dimensión con PCA.

Observamos que se han obtenido los mismos resultados que sin aplicar la reducción de la dimensionalidad.

clase/pred	1	2	3
1	38	0	0
2	1	57	6
3	0	0	46

clase	*precision*	*recall*	*f1-score*
1	0.97	1.00	0.99
2	1.00	0.89	0.94
3	0.88	1.00	0.94

Exactitud	0.95

Tabla 2: Resultados del algoritmo k-medias previa reducción de la dimensión con PCA.

6. DBSCAN

Finalmente, vamos a utilizar el algoritmo DBSCAN [9] para diferentes parámetros de radio y puntos mínimos. En este caso, usaremos el conjunto de datos original (con los valores atípicos incluidos). En este caso, los que considera valores atípicos los etiqueta con -1.

Uno de los valores más importantes a la hora de aplicar el algoritmo DBSCAN es el radio que consideramos. Por ello, para evitar tener que ir probando valores aleatorios se va a seguir un procedimiento parecido al del método del codo del k-medias [19]. Lo que hacemos es fijar un valor para los puntos mínimos (en nuestro caso 3) y a partir de ahí, dibujar todos los radios ordenados por distancia. De este modo, cuando los radios comienzan a aumentar de forma exponencial (el codo de la curva) significa que nos alejamos de la zona de alta densidad (valores normales) y entramos en la zona de baja densidad (valores atípicos). La curva obtenida se muestra en 7

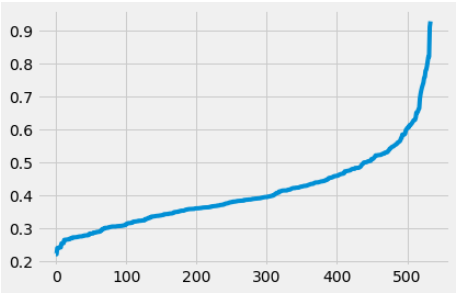


Figura 7: Curva de los radios ordenador por distancia para un número de puntos mínimos igual a 3

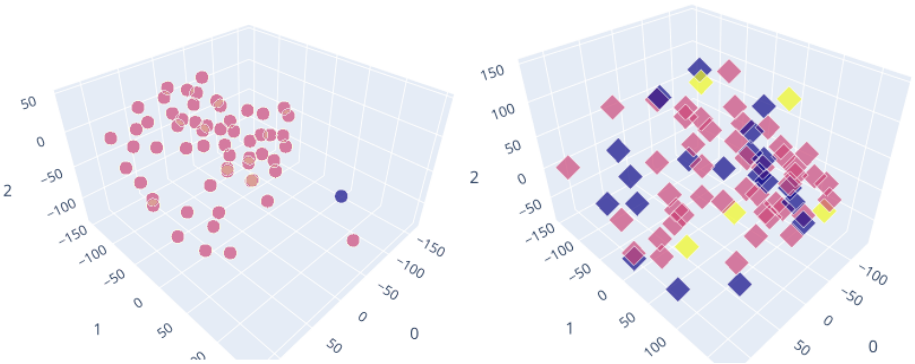
Vemos que a partir de $eps = 0{,}5$ es cuando empiezan a aumentar los radios. Aplicamos de ese modo el algoritmo DBSCAN con esos valores y obtenemos los resultados de la figura 8. Para mayor claridad se muestran los resultados para cada una de las clases reales.

Vemos que algoritmo DBSCAN solo se han detectado dos clases notadas con el color amarillo y rosa. Los valores atípicos aparecen en morado. No ha sido capaz de diferenciar correctamente la clases 1 y 2 originales, pero sí realiza un buen agrupamiento de la clase 3. En cuanto el tema de los *outliers*, los que ha detectado el algoritmo son diferentes a los que obtuvimos en la sección correspondiente 3.

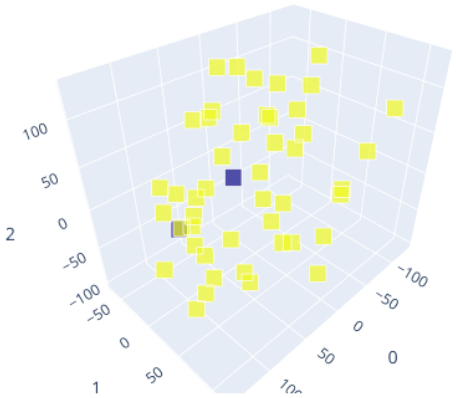
7. Conclusiones

En conclusión hemos visto cómo han variado los dendrogramas tras aplicar una eliminación de los valores atípicos. Tras este proceso ha quedado más clara de decisión del número de clústeres posibles que podremos obtener.

Posteriormente, gracias al método del codo y al coeficiente de silueta hemos definido el número de clústeres óptimo para aplicar el algoritmo k-medias y comprobado que realiza un buen agrupamiento. Estos resultados no varían aunque le apliquemos una reducción de características al conjunto (mediante



(a) Clase 1 (b) Clase 2



(c) Clase 3

Figura 8: Resultados por clases tras aplicar el algoritmo DBSCAN. Solo ha detectado dos clases, amarillo = clase 1, rosa=clase 0, morado=*outlier*.

PCA en nuestro caso).

Finalmente, hemos realizado un procedimiento parecido al del codo del k-medias para obtener un valor adecuado del radio para el algoritmo DBSCAN. Una vez aplicado hemos visto que es capaz de detectar bien la tercera clase mientras que no ha sido capaz de hacerlo con las otras dos y las ha clasificado en una sola.

Referencias

1. `scipy.cluster.hierarchy.dendrogram`: https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.dendrogram.html
2. `scipy.cluster.hierarchy.linkage` https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html
3. `scipy.spatial.distance.pdist` https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.pdist.html#scipy.spatial.distance.pdist
4. *SciPy Hierarchical Clustering and Dendrogram Tutorial*: https://joernhees.de/blog/2015/08/26/scipy-hierarchical-clustering-and-dendrogram-tutorial/
5. *Hierarchical Clustering with Python and Scikit-Learn*: https://stackabuse.com/hierarchical-clustering-with-python-and-scikit-learn/
6. *K-Means Clustering in Python: A Practical Guide:* https://realpython.com/k-means-clustering-python/
7. `sklearn.cluster.KMeans`: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
8. `sklearn.decomposition.pca`: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
9. `sklearn.cluster.DBSCAN`: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html
10. *Normalize columns of pandas data frame* https://stackoverflow.com/questions/26414913/normalize-columns-of-pandas-data-frame
11. `sklearn.manifold.TSNE` https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html
12. *3D Scatter Plots in Python*: https://plotly.com/python/3d-scatter-plots/
13. Detección de outliers en Python: https://www.aprendemachinelearning.com/deteccion-de-outliers-en-python-anomalia/
14. `PyOD` https://github.com/yzhao062/pyod
15. *Knee-point detection in Python:* https://github.com/arvkevi/kneed
16. `sklearn.metrics.silhouette_score` https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html
17. `sklearn.metrics.confusion_matrix` https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
18. `sklearn.metrics.classification_report` https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html
19. Ejemplo de uso de DBSCAN en Python para eliminación de outliers: http://exponentis.es/ejemplo-de-uso-de-dbscan-en-python-para-deteccion-de-outliers