



22 DE NOVIEMBRE DE 2022

PRÁCTICA 8

Clustering - Jerárquico - Particional

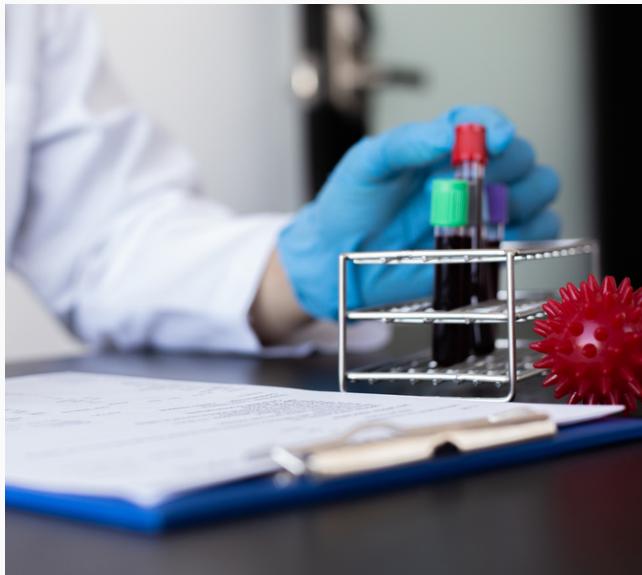


Objetivo de la práctica

Por Angel Damian Monroy Mendoza

No. de Cuenta: 316040707

Obtener grupos de pacientes con características similares, diagnosticadas con un tumor de mama, a través de clustering jerárquico y particional.



Características:

- El significado de cada variable se muestra a continuación:
- ID number: Identifica al paciente.
- Diagnosis: Diagnóstico (M=maligno, B=benigno).
- Radius: Media de las distancias del centro y puntos del perímetro.
- Texture: Desviación estándar de la escala de grises.
- Perimeter: Valor del perímetro del cáncer de mama.
- Area: Valor del área del cáncer de mama.
- Smoothness: Variación de la longitud del radio.
- Compactness: $\text{Perímetro}^2 / \text{Area}$.
- Concavity: Caída o gravedad de las curvas de nivel.
- Concave points: Número de sectores de contorno cóncavo.
- Symmetry: Simetría de la imagen.
- Fractal dimension: “Aproximación de frontera”.

Desarrollo

Para el desarrollo de esta práctica dividimos el proceso en cinco secciones:

1. Importación de las bibliotecas necesarias y datos.
2. Selección de características.
3. Estandarización de los datos.
4. Clustering Jerárquico.
5. Clustering Particional.



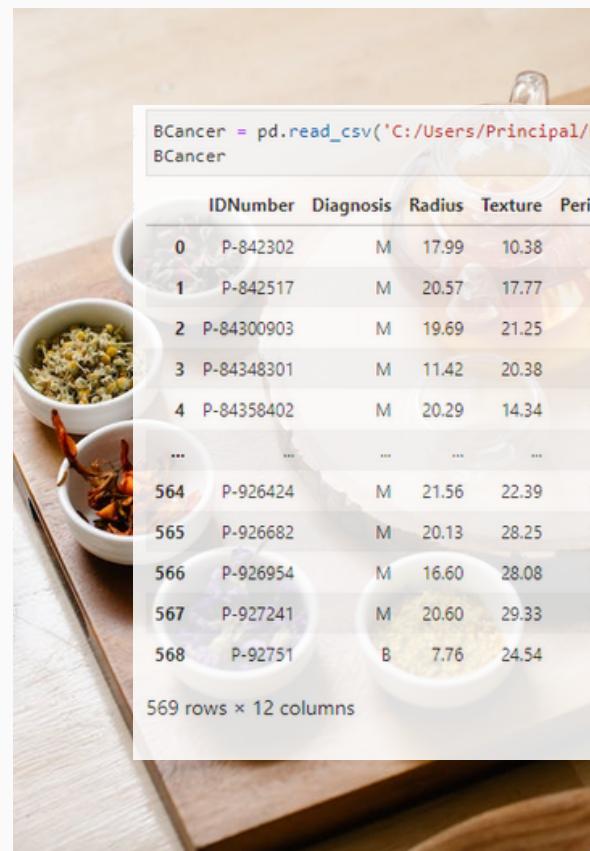
1. Bibliotecas y datos

En este apartado se instalaron las bibliotecas necesarias para la manipulación de los datos, creación de vectores y matrices, generación de gráficas y posteriormente, en la sección de *Clustering Jerárquico* y *Clustering Particional*, se instalaron las bibliotecas para K-Means y Ascendente Jerárquico con *sklearn.cluster*, *sklearn.metrics* y *scipy.cluster.hierarchy*.

Después, se cargaron los datos de forma tradicional en donde, se realizó un llamado al método de *.info()* para observar de una forma más condensada las características del conjunto de datos, así como su tipo de variables que almacenaba.

Para este punto, observamos que la variable *Diagnosis* era de tipo *object*, lo cual tuvo sentido pues guardaba la clasificación del tipo de tumor (M=maligno, B=benigno) de las observaciones, por lo que decidimos imprimir el agrupamiento de dicha variable, junto con el método de *.size()* para saber cuántas observaciones se clasificaron como Benignos y cuántos como Malignos

```
print(BCancer.groupby('Diagnosis').size())
Diagnosis
B    357
M    212
dtype: int64
```



	IDNumber	Diagnosis	Radius	Texture	Perimeter	Area	Symmetry	Fractal Dimension	Concavity	Concave Points	Thickness
0	P-842302	M	17.99	10.38	30.57	187.0	15.99	0.392	0.485	0.005	0.001
1	P-842517	M	20.57	17.77	34.61	205.8	19.70	0.446	0.479	0.005	0.001
2	P-84300903	M	19.69	21.25	34.14	201.0	20.57	0.432	0.453	0.005	0.001
3	P-84348301	M	11.42	20.38	24.58	105.1	12.99	0.281	0.291	0.005	0.001
4	P-84358402	M	20.29	14.34	34.98	202.2	20.88	0.432	0.453	0.005	0.001
...
564	P-926424	M	21.56	22.39	35.73	205.8	20.57	0.446	0.479	0.005	0.001
565	P-926682	M	20.13	28.25	34.54	195.0	19.92	0.429	0.453	0.005	0.001
566	P-926954	M	16.60	28.08	34.54	195.0	19.92	0.429	0.453	0.005	0.001
567	P-927241	M	20.60	29.33	35.73	205.8	20.57	0.446	0.479	0.005	0.001
568	P-92751	B	7.76	24.54	19.86	95.9	11.42	0.281	0.291	0.005	0.001

Desarrollo

2. Selección de Características

Primeramente se realizó una inspección visual muy general con la ayuda de la biblioteca de *seaborn* con el método *.pairplot* y con un matiz basado en la variable *Diagnóstico* (*hue = 'Diagnosis'*). Esto nos permitió observar el tipo de relación con la que contaban cada una de las variables entre ellas. Sin embargo, se comentó en clase que si se quiere expandir la documentación, se pueden graficar las variables de interés de forma individual.

Posteriormente, se realizó la matriz de correlaciones para realizar el análisis de la relación entre variables de forma numérica. No obstante, aquí se presenta un problema, pues al contar con 10 características se vuelve complicado observar cada celda para ver la relación y se vuelve más fácil equivocarse.

Existen dos formas de resolver esto; la primera, consiste en imprimir la clasificación de los valores de cada variable de forma ascendente o descendente de los primeros 10 valores, pero el problema es que se vuelve poco eficiente realizar esto mismo para cada característica.

Por esta razón, optamos por la segunda forma, la cual consiste en realizar la matriz de correlación con la función *.corr()* pero apoyándonos de *seaborn* para realizar un mapa de calor con el método *.heatmap*, en donde le decimos que nos grafique la parte inferior de la matriz y nos muestre los valores correspondientes a la correlación.

Posteriormente, se seleccionaron las variables con las que se trabajó mediante el criterio del tercio medio superior/inferior, que dice lo siguiente:

- Coeficiente de correlación:
 - $|+1.00 \text{ a } +0.67|$ --> Fuerte o Alta
 - $|+0.66 \text{ a } +0.34|$ --> Moderada o Media
 - $|+0.33 \text{ a } +0.00|$ --> Débil o Baja

Desarrollo

2. Selección de Características

Dichas variables fueron las siguientes:

Variables seleccionadas:

- | | |
|----------------------------|-----------------------------------|
| 1. Textura [Posición 3] | 4. Compactness [Posición 7] |
| 2. Area [Posición 5] | 5. Symmetry [Posición 10] |
| 3. Smoothness [Posición 6] | 6. FractalDimension [Posición 11] |

Por último, se realizó un nuevo conjunto de datos donde sólo se tuvieran las variables seleccionadas, ya sea extrayendo las el nombre de las columnas del conjunto de datos original o seleccionando las filas y columnas con el método de `iloc`.

	0	1	2	3	4	5
0	1038	1001.0	0.11840	0.27760	0.2419	0.07871
1	17.77	1326.0	0.08474	0.07864	0.1812	0.05667
2	21.25	1203.0	0.10960	0.15990	0.2069	0.05999
3	20.38	386.1	0.14250	0.28390	0.2597	0.09744
4	14.34	1297.0	0.10030	0.13280	0.1809	0.05883
...
564	22.39	1479.0	0.11100	0.11590	0.1726	0.05623

3. Estandarización de los datos.

Para esta parte se utilizó la biblioteca `sklearn.preprocessing` para importar el método de `normalizar` y `escalar`, pues al estar trabajando con datos numéricos que difieren mucho entre ellos, es preciso realizar la estandarización para que cada uno de los valores tenga el mismo peso y aporte de forma equilibrada a los resultados.

Lo que se realizó en este apartado fue escoger el método de estandarización para los datos, el cual fue el escalado y, posteriormente, se instanció una nueva matriz estandarizada "`MEstandarizada`" ajustando los datos de la matriz obtenida de la selección de características, al escalado.

	0	1	2	3	4	5
0	-2.073335	0.984375	1.568466	3.283515	2.217515	2.217515
1	-0.353632	1.908708	-0.826962	-0.487072	0.001392	-0.826962
2	0.456187	1.558884	0.942210	1.052926	0.939685	-0.353632
3	0.253732	-0.764464	3.283553	3.402909	2.867383	4.900000
4	-1.151816	1.826229	0.280372	0.539340	-0.009560	-0.539340
...
564	0.721473	2.343856	1.041842	0.219060	-0.312589	-0.900000
565	2.085134	1.723842	0.102458	-0.017833	-0.217664	-1.000000
566	2.045574	0.577953	-0.840484	-0.038680	-0.809117	-0.809117
567	2.336457	1.735218	1.525767	3.272144	2.137194	1.000000
568	1.221792	-1.347789	-3.112085	-1.150752	-0.820070	-0.539340
569	1.010101	0.101010	0.101010	0.101010	0.101010	0.101010

Desarrollo

4. Clustering Jerárquico

En este apartado se realiza una figura (`figsize = (10,7)`), se le coloca un título y las etiquetas a x & y, para graficar ahí el *dendrograma* que surge de vincular todos los elementos (`method = 'complete'`) de la matriz MEstandarizada y con la métrica *euclidiana* (`metric='euclidean'`).

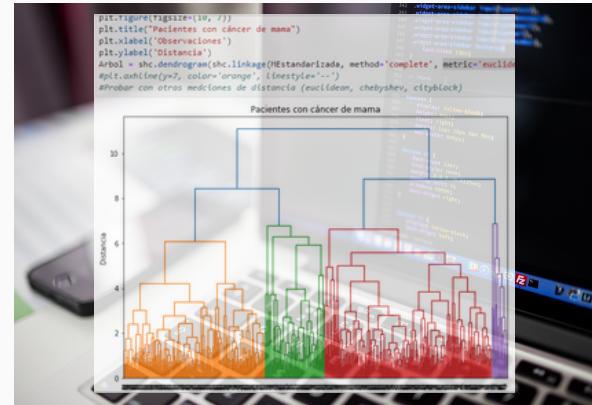
Posteriormente, se crean las etiquetas de los elementos en los clústeres instanciando un objeto de nombre MJerárquico que recibirá el algoritmo de Ascendente Jerárquico ('*AgglomerativeClustering*') con un número de clusters igual a 4 (visualizado en el dendrograma). Después, se ajustan las predicciones del algoritmo en la matriz MEstandarizada y, por último, se imprimen las etiquetas obtenidas.

En seguida, se agrega una columna más a la matriz original de nuestros datos en donde guardamos los valores de las etiquetas obtenidas por el algoritmo de Ascendente Jerárquico e imprimimos la nueva matriz original.

Luego se hace un conteo (`.count()`) de la nueva columna de etiquetas y se agrupan los valores diferentes de la misma columna de etiquetas para obtener la cantidad de elementos en cada uno de los 4 clusters.

Después se imprime la matriz generada anteriormente, pero sólo donde el valor de la etiqueta sea '0', es decir, que pertenezca al cluster 0 con la finalidad de hacer breves observaciones acerca de ese cluster.

Por último, se obtienen los centroides instanciando un nuevo objeto de nombre *CentroidesH*, en donde se guarda el promedio (`método .mean()`) de los valores de las características seleccionadas, agrupadas por la columna de etiquetas de los clusters; pues imprimiendo este objeto ya se puede realizar el análisis de cada cluster.



*Nota: no se presentaron los análisis de las configuraciones debido a que éstas se encuentran en el cuaderno.



Desarrollo

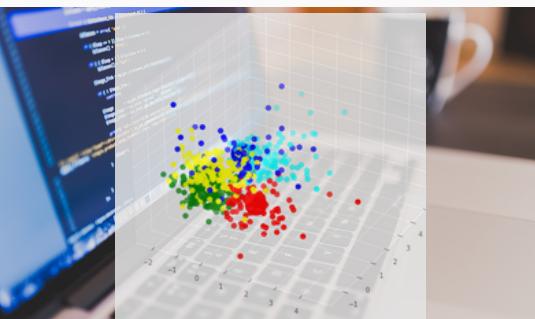
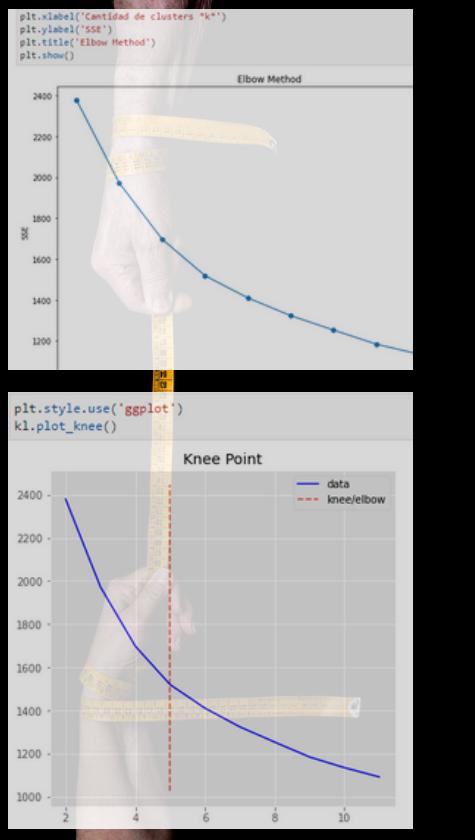
5. Clustering Particional

Para realizar el clustering particional, debemos recordar que nosotros tenemos que proponer el número de clusters que tendrá nuestro conjunto de datos. Por esta razón, utilizamos el "método del codo" en donde se utiliza la suma total de la distancia al cuadrado dentro de cada grupo (SSE) y se realiza un recorrido entre 2 y 12 (rango recomendado de número de clusters), instanciamos el algoritmo KMeans, ajustamos este algoritmo a la matriz MEstandarizada y realizamos una lista sobre SSE con elementos de km con su atributo `.inertia_`.

Luego, graficamos el método del codo obtenido de lo anterior y se observó que no hay un codo afilado (codo agudo), es decir, un número claro de clusters, por lo que se optó por realizar el complemento de este método, el cual es llamado *método de la rodilla (knee method)*.

Para utilizar el *método de la rodilla* se instaló `kneed` y, después, se importó la biblioteca de `KneeLocator`. En seguida, se instanció un localizador en el mismo rango de la Suma de la Distancias al Cuadrado de los de la curva convexa y decreciente que se vio en el *método del codo* y, a continuación, se imprimió y graficó el valor de cinco clusters que arrojó el localizador.

Luego, al igual que en el apartado de *Clustering Jerárquico*, se crean las etiquetas de los elementos en los clústeres y se imprimen. Se agrega otra columna al conjunto de datos en donde se guardan los valores de las etiquetas para el clustering particional. Se hace una agrupación y conteo de los elementos en cada uno de los cluster. Se imprime la matriz de datos donde el cluster es 0 para una inspección visual. Y, por último, se obtienen los centroides para hacer el análisis de cada clúster.





Conclusiones

De esta práctica podemos concluir que las métricas de distancia siguen siendo una parte importante cuando se están analizando los datos y cuando se obtiene un modelo pues, en especial con este tipo de aprendizaje basado en métricas, los resultados que puedes obtener varian considerablemente.

Consideremos otro tipo de métrica para el dendrograma; en lugar de utilizar la métrica euclidiana, utilizaremos la métrica de Manhattan y la de Chebyshev para observar qué tanto difiere uno del otro.

Como podemos observar, la distribución en los clusters de la métrica de Manhattan es diferente a la Euclíadiana a pesar de tener el mismo número de clusters (4) y, por su parte, la métrica de Chebyshev además de diferir en la distribución en los clústers, sí aumento el número de ellos, pues obtuvo un total de 6.

Pasando a otro tema, se apreció la importancia de una buena ingeniería de características, sin embargo, cuando se presentan muchas variables y muchas relaciones estrechas entre variables, lo más conveniente es realizar varios modelos para presentarlos ante el especialista ya que, tal y como vimos en clase, se requerirá ayuda del experto para que él nos diga qué variables es imprescindible conservar y cuáles no, basándose en nuestro modelo.

Por último, apreciamos la importancia de contar con el complemento del método del codo, el *método de la rodilla*, debido a que en la práctica generalmente no se contará con algún codo afilado que proporcione una información que no sea ambigua para poder elegir el número de clusters que mejor se ajuste a nuestro modelo.

