**Basile Álvarez Andrés José**

No. Cuenta: 316617187

Email: andresbasile123@gmail.com

Fecha: 26/10/21

**Inteligencia Artificial**

**Grupo III**

Semestre 2022-1

# Reporte Práctica 6: Clustering Jerárquico y Particional (Segmentación de Clientes)

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

**Fuente de datos:** Estudios clínicos a partir de imágenes digitalizadas de pacientes con cáncer de mama de Wisconsin (WDBC, Wisconsin Diagnostic Breast Cancer), donde:

- ID number: Identifica al paciente (valor discreto).
- Diagnosis: Diagnóstico (M=maligno, B=benigno).
- Radius: Media de las distancias del centro y puntos del perímetro.
- Texture: Desvaición 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: Perímetro ^2 / Area -1
- Concavity: Caída o gravedad de las curvas de nivel.
- Concave Points: Número de sectores de contorno cóncavo.
- Symmetry: Simetría de la imagen
- Fractal dimensión: Aproximación de frontera – 1.

**Características Generales:**

En esta práctica, utilizaremos el clustering jerárquico y el clústering particional. Ambos son un tipo de aprendizaje no supervisado en donde, a partir de una fuente de datos, se organizan los elementos dentro de clústeres, en los cuales los elementos son unidos por características que éstos tienen en común. El objetivo de este tipo de aprendizaje es dividir una población heterogénea de elementos en un número de grupos homogéneos, de acuerdo con sus similitudes.

Dentro de este tipo de aprendizaje, utilizaremos otros algoritmos de IA vistos previamente, como las métricas de distancia, las cuales nos serán útiles para conocer las similitudes entre objetos.

El clustering jerárquico organiza elementos de forma recursiva en una estructura de árbol y consiste en cuatro pasos fundamentales: medir la similitud de los elementos (utilizando métricas

de distancia), agrupar los elementos similares, decidir la cantidad adecuada de grupos, interpretar los grupos obtenidos.

En el clustering particional, se establecen *k* centroides para la formación de *k* grupos. Estos centroides se eligen de forma aleatoria. Cada elemento de la fuente de datos es asignado al centroide más cercano. Luego, se actualiza la posición del centroide con base en la media de los elementos asignados en el clúster. Se repiten los dos pasos anteriores de manera iterativa hasta que los centroides no cambien de lugar.

**Desarrollo**

Primeramente, tenemos que definir aquellas bibliotecas de Python que nos serán útiles para importar, limpiar y analizar los datos contenidos en el archivo separado por comas *WDBCOriginal.csv*. Estas serán: *pandas* (manipulación y análisis de datos), *matplotlib* (para la creación de gráficas y visualización de los datos), *numpy* para utilizar vectores y matrices de *n* dimensiones y *seaborn*, para la visualización de datos basado en *matplotlib*.

Más tarde, importamos el archivo *WDBCOriginal.csv* y lo primero que hacemos es guardarlo en un DataFrame. La importación del archivo se realizó a partir del explorador de archivos que abrimos utilizando el comando *files.upload()*. Una vez importados los datos, mostramos el DataFrame que los contiene:

| | IDNumber | Diagnosis | Radius | Texture | Perimeter | Area | Smoothness | Compactness | Concavity | ConcavePoints | Symmetry | FractalDimension |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | P-842302 | M | 17.99 | 10.38 | 122.80 | 1001.0 | 0.11840 | 0.27760 | 0.30010 | 0.14710 | 0.2419 | 0.07871 |
| 1 | P-842517 | M | 20.57 | 17.77 | 132.90 | 1326.0 | 0.08474 | 0.07864 | 0.08690 | 0.07017 | 0.1812 | 0.05667 |
| 2 | P-84300903 | M | 19.69 | 21.25 | 130.00 | 1203.0 | 0.10960 | 0.15990 | 0.19740 | 0.12790 | 0.2069 | 0.05999 |
| 3 | P-84348301 | M | 11.42 | 20.38 | 77.58 | 386.1 | 0.14250 | 0.28390 | 0.24140 | 0.10520 | 0.2597 | 0.09744 |
| 4 | P-84358402 | M | 20.29 | 14.34 | 135.10 | 1297.0 | 0.10030 | 0.13280 | 0.19800 | 0.10430 | 0.1809 | 0.05883 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 564 | P-926424 | M | 21.56 | 22.39 | 142.00 | 1479.0 | 0.11100 | 0.11590 | 0.24390 | 0.13890 | 0.1726 | 0.05623 |
| 565 | P-926682 | M | 20.13 | 28.25 | 131.20 | 1261.0 | 0.09780 | 0.10340 | 0.14400 | 0.09791 | 0.1752 | 0.05533 |
| 566 | P-926954 | M | 16.60 | 28.08 | 108.30 | 858.1 | 0.08455 | 0.10230 | 0.09251 | 0.05302 | 0.1590 | 0.05648 |
| 567 | P-927241 | M | 20.60 | 29.33 | 140.10 | 1265.0 | 0.11780 | 0.27700 | 0.35140 | 0.15200 | 0.2397 | 0.07016 |
| 568 | P-92751 | B | 7.76 | 24.54 | 47.92 | 181.0 | 0.05263 | 0.04362 | 0.00000 | 0.00000 | 0.1587 | 0.05884 |

569 rows × 12 columns

Figura 1: Data Frame con los datos de pacientes.

Como los datos ya se encuentran preparados, comenzaremos mostrando información sobre ellos, para después poder hacer una selección de variables para evitar la alta dimensionalidad de los datos.

Luego, utilizamos la biblioteca *seaborn* para mostrar un *pairplot* (el cual nos permitirá visualizar rápidamente gráficas de dispersión que muestran la relación entre las variables de nuestro *dataframe*). Dentro de este *pairplot*, utilizamos la variable *diagnosis* como *hue*, destacando su valor con respecto de las demás variables que se están graficando.

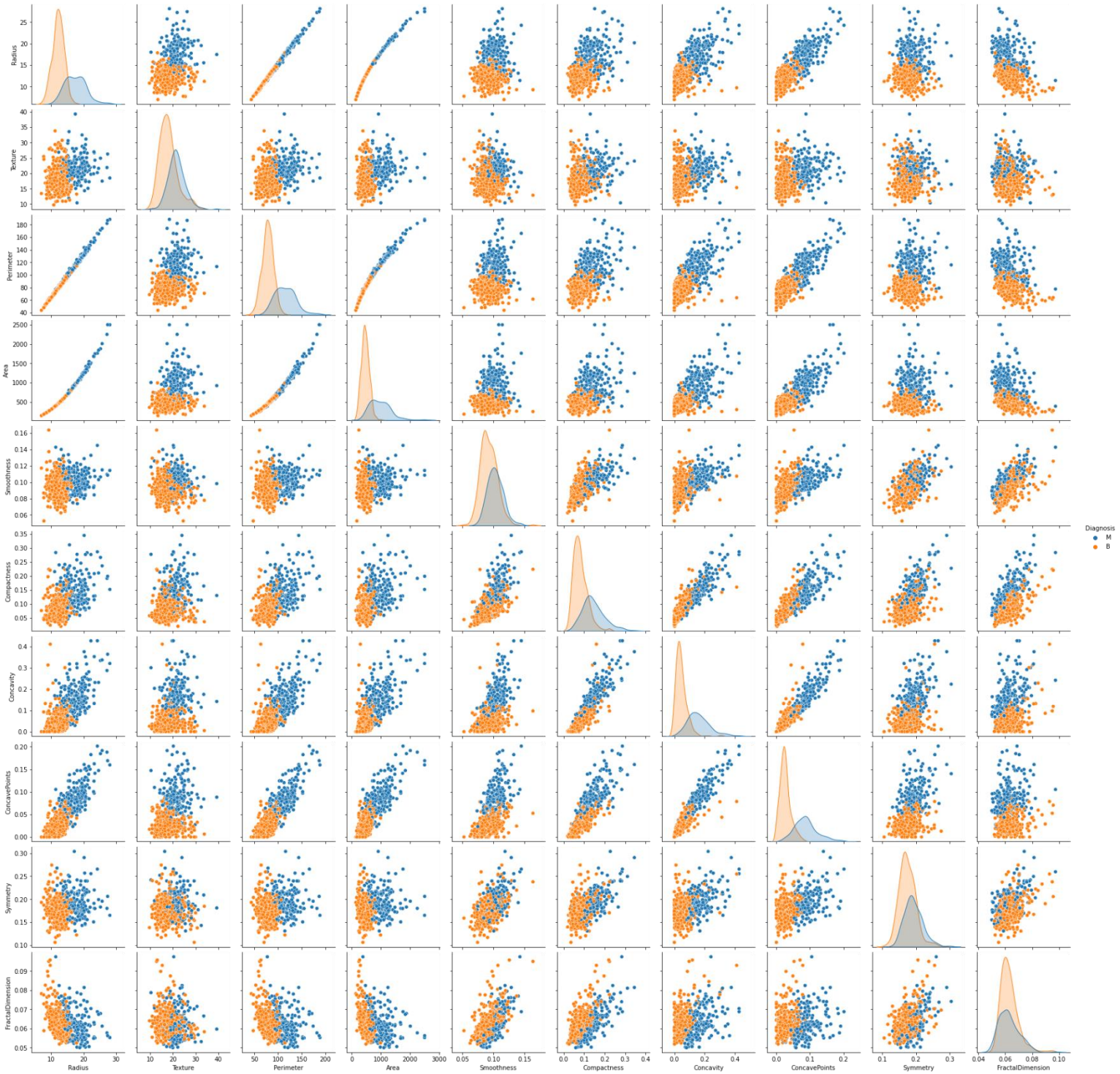


Figura 2: *Pairplot* con los datos de pacientes.

Las gráficas de tipo *pairplot* son una gran manera de identificar tendencias para un análisis posterior de los datos.

A manera de pormenorizar el análisis, obtenemos nada más algunas gráficas (para un par de variables) utilizando la función *scatterplot* de *seaborn*. A continuación, se muestra un ejemplo con las variables *Radius* y *Perimeter*, utilizando como *hue* la variable *Diagnosis*:

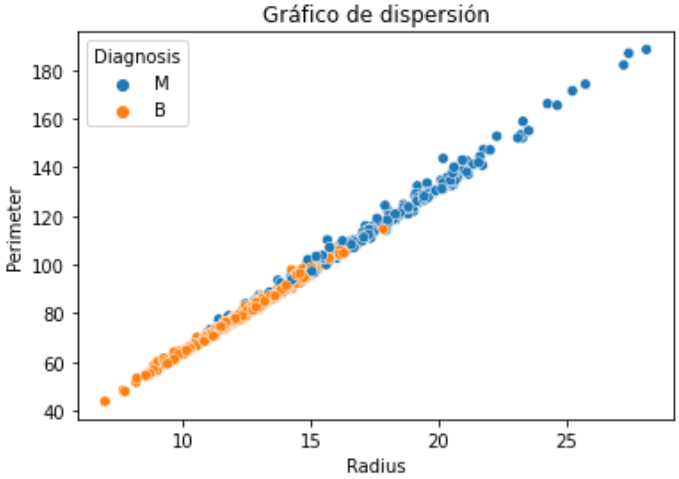


Figura 3: Gráfico de dispersión *Perimeter* y *Radius*, con una dispersión moderada. Cabe destacar que, alcanzamos a observar cómo los pacientes con diagnóstico de tumor benigno tienen tumores con radio y perímetro más chicos, mientras que los pacientes con diagnóstico de tumor maligno tienen tumores con radio y perímetro mayores. Por lo que alcanzamos a ver, las variables graficadas podrán ser seleccionadas para eliminarse y reducir la dimensionalidad del análisis. Más adelante, con la matriz de correlaciones, podremos concluir si vale la pena hacer esto o no.

Mostramos también un ejemplo con las variables *Concavity* y *ConcavePoints*, utilizando como *hue* la variable *Diagnosis:*

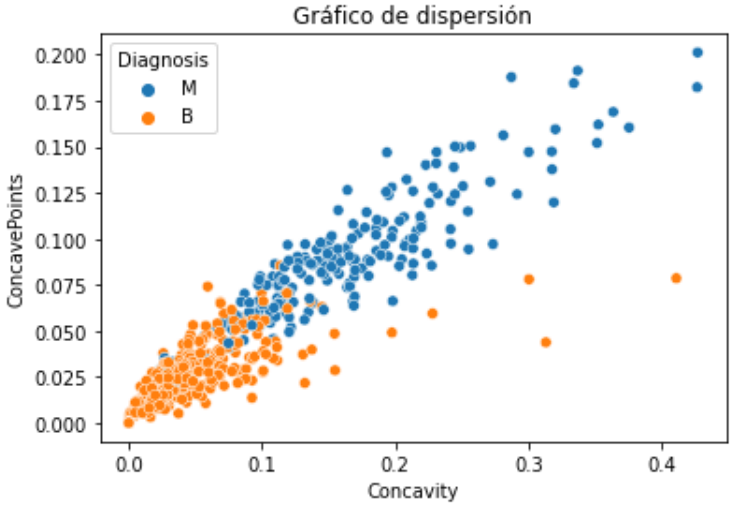


Figura 4: Gráfico de dispersión *Concavity* y *ConcavePoints*, con una dispersión moderada. Por lo que alcanzamos a ver, las variables graficadas podrán ser seleccionadas para eliminarse y reducir la dimensionalidad del análisis. Más adelante, con la matriz de correlaciones, podremos concluir si vale la pena hacer esto o no, al analizar la relación entre estas variables y las demás.

Posteriormente, utilizamos el método de Pearson para obtener la matriz de correlaciones de los datos. Imprimimos el top 10 de valores para obtener lo siguiente:

```
Radius              1.000000
Perimeter           0.997855
Area                0.987357
ConcavePoints       0.822529
Concavity           0.676764
Compactness         0.506124
Texture             0.323782
Smoothness          0.170581
Symmetry            0.147741
FractalDimension   -0.311631
Name: Radius, dtype: float64
```

Figura 5: Top 10 de valores correlacionados.

Además, utilizando la biblioteca *seaborn*, imprimimos el *heatmap* de la matriz de correlaciones para obtener lo siguiente:

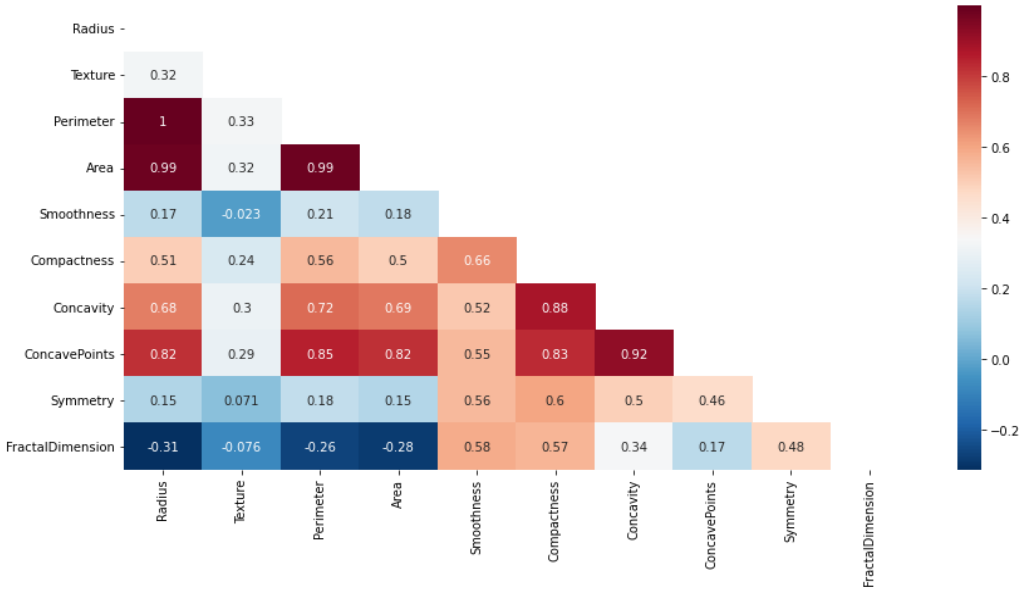


Figura 6: *Heatmap* de los valores correlacionados.

A partir de un análisis del *heatmap* anterior, podemos observar que existe una alta correlación entre las variables *Area*, *Perimeter* y *Radius*, por lo que nos conviene discriminar estas variables. Nos quedaremos con la variable *Area* porque es la que tiene menor cantidad de dependencia con las demás variables.También eliminaremos la variable *Concavity* y la variable *ConcavePoints* debido a que tienen alta dependencia con otras variables.

Para poder trabajar con los datos y aplicarles un algoritmo de clustering, es necesario escalarlos para que cada variable tenga un peso igual sobre el conjunto de datos. En este caso, utilizaremos el objeto *StandardScaler* de la biblioteca *sklearn.preprocessing* para realizar este proceso. Podríamos utilizar también el método *MinMaxScaler*, el cual puede ser útil cuando se identifica que ninguna variable tiene un dato uera de rango (por ejemplo, una edad de 150 años).

Luego, importamos la biblioteca *cluster.hierarchy* de *scipy* y el método *AgglomerativeClustering* de *sklearn.cluster* para realizar el árbol de clustering jerárquico. Creamos un *dendogram* con la matriz estandarizada previamente y la métrica de distancia euclidiana. El *dendogram* permitirá visualizar los grupos que se generan en el clúster jerárquico. Al utilizar la métrica de distancia euclidiana, obtenemos 4 clústeres diferentes, como se muestra a continuación:

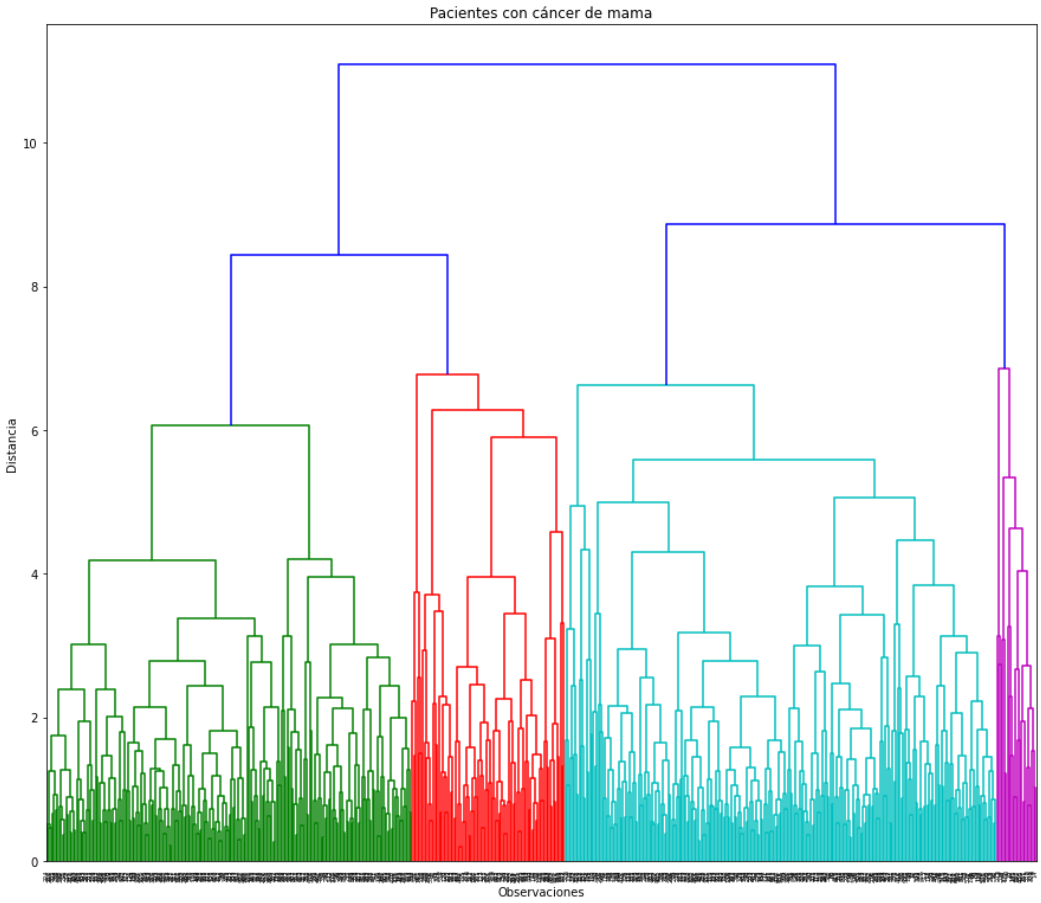


Figura 7: Árbol jerárquico para pacientes con cáncer de mama. Vemos que se produjeron 4 clústeres.

Para cada uno de los registros, le asignamos una etiqueta que indique a qué clúster pertenece, como podemos ver a continuación:

```
#Se crean las etiquetas de los elementos en los clústeres
MJerarquico = AgglomerativeClustering(n_clusters=4, linkage='complete', affinity='euclidean')
MJerarquico.fit_predict(MEstandarizada)
MJerarquico.labels_   #poner etiquetas a cada cluster. #el array indica qué cluster va a tomar cada uno
```

```
array([0, 1, 1, 0, 1, 2, 1, 2, 0, 0, 3, 2, 1, 3, 0, 2, 3, 2, 1, 2, 2, 2,
       0, 1, 2, 0, 2, 1, 2, 2, 1, 2, 2, 1, 2, 2, 2, 3, 3, 2, 3, 2, 1, 2,
       2, 2, 3, 2, 2, 3, 3, 3, 3, 2, 3, 3, 1, 2, 3, 2, 2, 2, 2, 2, 2, 2,
       2, 3, 2, 2, 1, 2, 2, 2, 2, 1, 2, 2, 0, 2, 2, 2, 1, 1, 2, 2, 2, 2,
       3, 2, 3, 3, 3, 2, 2, 2, 2, 2, 3, 2, 3, 2, 3, 2, 2, 2, 2, 3, 0, 3,
       2, 2, 2, 2, 2, 3, 2, 2, 2, 1, 3, 2, 0, 2, 3, 3, 3, 1, 2, 1, 2, 2,
       2, 2, 1, 3, 3, 2, 2, 2, 3, 2, 2, 2, 3, 2, 0, 3, 2, 3, 2, 2, 2, 2,
       2, 3, 2, 1, 3, 3, 2, 1, 2, 3, 1, 3, 3, 1, 1, 2, 2, 3, 2, 2, 3, 3,
       2, 2, 3, 3, 1, 0, 1, 3, 3, 3, 1, 2, 2, 3, 0, 3, 3, 2, 2, 3, 2, 1,
       1, 2, 2, 1, 1, 0, 3, 3, 2, 1, 2, 3, 1, 3, 1, 1, 2, 2, 3, 3, 2, 1,
       3, 2, 2, 2, 3, 2, 2, 2, 3, 2, 2, 3, 3, 1, 3, 3, 1, 1, 3, 1, 2, 3,
       2, 3, 2, 2, 3, 2, 2, 2, 1, 3, 1, 1, 1, 2, 1, 0, 0, 1, 1, 3, 2, 3,
       2, 1, 3, 3, 2, 2, 3, 2, 1, 3, 3, 2, 3, 1, 3, 2, 1, 3, 1, 2, 3, 3,
       3, 3, 2, 3, 2, 2, 2, 3, 2, 3, 3, 2, 3, 3, 1, 3, 1, 2, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 2, 3, 3, 1, 2, 3, 2, 1, 2, 1, 3, 2, 3, 3, 2, 2,
       2, 2, 2, 3, 3, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 2, 3, 0,
       1, 1, 3, 3, 2, 3, 2, 2, 3, 3, 2, 1, 3, 1, 1, 2, 1, 1, 2, 3, 1, 1,
       3, 2, 2, 3, 3, 0, 2, 3, 3, 2, 3, 3, 3, 3, 2, 2, 2, 2, 2, 1, 2, 3,
       2, 3, 3, 3, 0, 3, 3, 2, 3, 2, 2, 3, 2, 3, 3, 2, 3, 2, 3, 2, 2, 2,
       3, 3, 3, 2, 2, 3, 2, 3, 2, 3, 3, 3, 2, 2, 1, 2, 3, 2, 3, 3, 3, 3,
       2, 3, 3, 2, 2, 2, 1, 2, 3, 1, 3, 1, 3, 2, 3, 3, 3, 3, 3, 3, 1, 1,
       3, 3, 3, 3, 3, 3, 2, 2, 2, 3, 3, 3, 2, 2, 3, 3, 2, 2, 3, 3, 2, 2,
       2, 2, 3, 1, 2, 1, 3, 3, 2, 3, 3, 3, 2, 3, 2, 1, 2, 2, 2, 1, 0, 0,
       2, 2, 2, 2, 2, 3, 2, 2, 3, 2, 1, 1, 2, 2, 2, 1, 3, 2, 2, 2, 2, 3,
       2, 2, 2, 2, 2, 2, 2, 1, 2, 0, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 2, 3, 3, 3, 3, 2, 3, 3, 3, 1, 3, 1, 1, 1, 1, 3, 0, 3])
```

Figura 8: Etiquetas de los elementos en los clústeres producidos por el algoritmo jerárquico.

```
clusterH
0     23
1     88
2    248
3    210
Name: clusterH, dtype: int64
```

Figura 9: Cantidad de elementos en cada clúster del algoritmo jerárquico.

Imprimimos los centroides de cada uno de los clústeres para hacer una interpretación de ellos:

| clusterH | Texture | Area | Smoothness | Compactness | Symmetry | FractalDimension |
|---|---|---|---|---|---|---|
| **0** | 20.133478 | 775.543478 | 0.124274 | 0.242200 | 0.240830 | 0.077839 |
| **1** | 22.540568 | 1243.728409 | 0.098441 | 0.137140 | 0.182560 | 0.058889 |
| **2** | 18.167540 | 561.336694 | 0.103316 | 0.114235 | 0.190486 | 0.065737 |
| **3** | 19.160095 | 505.403810 | 0.084217 | 0.063813 | 0.163030 | 0.059317 |

Figura 10: Centroides de cada clúster generado por el algoritmo jerárquico.

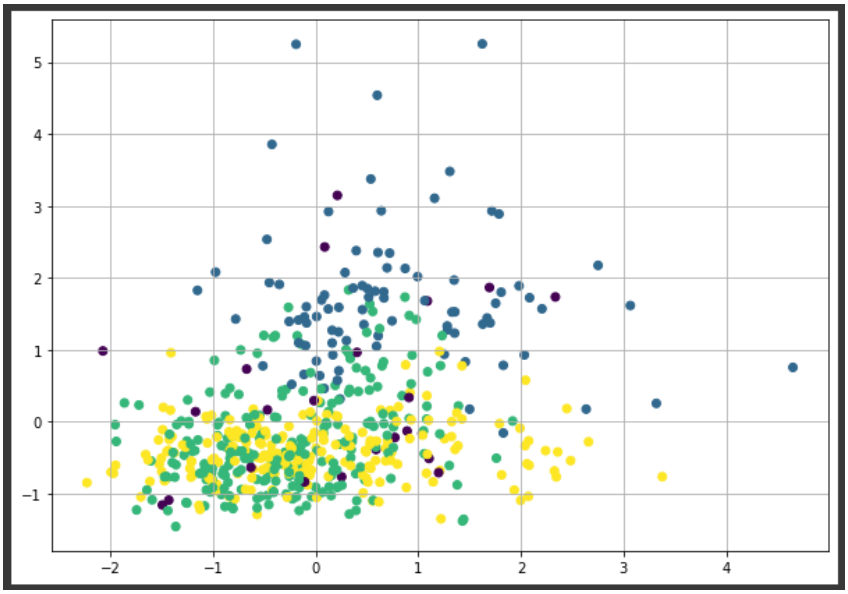

Figura 11: Clústeres generados por el algoritmo jerárquico.

**Interpretación de los resultados obtenidos con el algoritmo jerárquico:**

**Clúster 0:** En este cluster se encuentran 23 pacientes cuyos tumores tienen una textura de 20 y un área de 775 unidades. La variación de la longitud del radio fue de 0.12, con un *compactness* (relación del perímetro al cuadrado entre el área menos uno) de 0.24, una simetría de 0.24 y una *fractal dimension* o aproximación de frontera de 0.07. Se podría interpretar que en este grupo están los pacientes con probabilidades de desarrollar un tumor maligno; el tumor tiene un tamaño promedio intermedio entre lo que podría considerar maligno y lo que se considera benigno.

**Clúster 1:** En este cluster se encuentran 88 pacientes cuyos tumores tienen una textura de 22 y un área de 1243 unidades. La variación de la longitud del radio fue de 0.09, con un *compactness* (relación del perímetro al cuadrado entre el área menos uno) de 0.13, una simetría de 0.18 y una *fractal dimension* o aproximación de frontera de 0.058. El tamaño del tumor, en promedio, es considerablemente mayor que el de los demás grupos. Se podría interpretar que en este grupo se encuentran los pacientes con más probabilidad de tener un tumor maligno.

**Clúster 2:** En este cluster se encuentran 248 pacientes cuyos tumores tienen una textura de 18 y un área de 561 unidades. La variación de la longitud del radio fue de 0.10, con un *compactness* (relación del perímetro al cuadrado entre el área menos uno) de 0.11, una simetría de 0.19 y una *fractal dimension* o aproximación de frontera de 0.065. El tamaño del tumor, en promedio, es "bajo" en relación con los clústeres 0 y 1. Se podría interpretar que los pacientes en este grupo tienen mayor probabilidad de tener un tumor benigno, al igual que los pacientes del siguiente clúster.

**Clúster 3:** En este cluster se encuentran 210 pacientes cuyos tumores tienen una textura de 19 y un área de 505 unidades. La variación de la longitud del radio fue de 0.08, con un *compactness* (relación del perímetro al cuadrado entre el área menos uno) de 0.06, una simetría de 0.16 y una *fractal dimension* o aproximación de frontera de 0.059. El tamaño del tumor, en promedio, es "bajo" en relación con los clústeres 0 y 1. Se podría interpretar que los pacientes en este grupo tienen mayor probabilidad de tener un tumor benigno, al igual que los pacientes del cluster anterior.

Ahora, para el **algoritmo *K-Means* particional**, importamos *KMeans* de *sklearn.cluster* y *pairwise_distances_argmin_min* de *sklearn.metrics*. Luego, utilizamos el método del codo para definir la cantidad adecuada de clústeres a utilizar. En Python, podemos utilizar la biblioteca *kneed* para usar el método *KneeLocator,* el cual facilita la tarea de encontrar el "codo" de la curva pasando como parámetro los valores de SSE, el tipo de curva (en este caso convexa) y la dirección (en este caso decreciente).

El método *KneeLocator* nos muestra lo siguiente de forma gráfica:

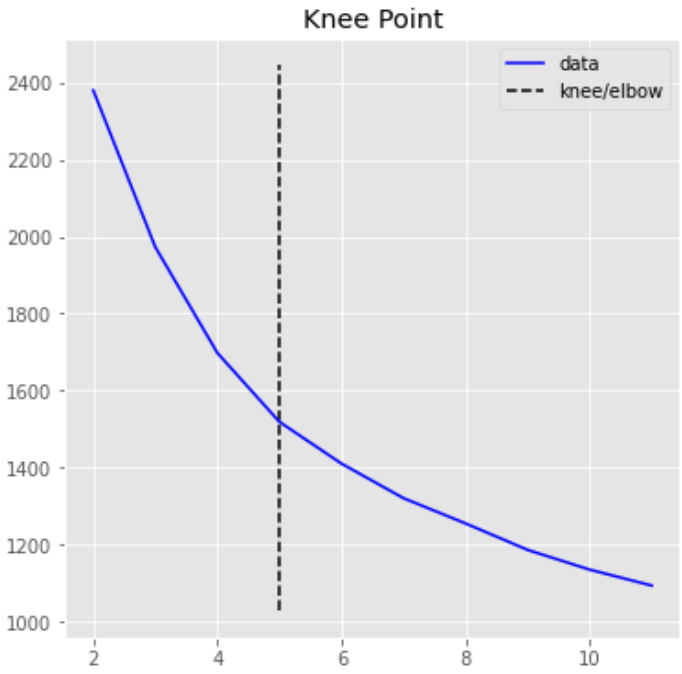


Figura 12: Knee Point devuelto por el método *KneeLocator*.

A partir de lo devuelto por el método, utilizaremos 5 clústeres para el algoritmo particional. Al aplicar el algoritmo, obtenemos la siguiente división de los elementos en cada cluster:

```
MParticional = KMeans(n_clusters=5, random_state=0).fit(MEstandarizada)
MParticional.predict(MEstandarizada)
MParticional.labels_
```

```
array([2, 1, 1, 2, 1, 2, 1, 2, 2, 2, 4, 3, 2, 4, 2, 2, 0, 2, 1, 0, 3, 3,
       2, 1, 1, 2, 2, 1, 1, 3, 1, 2, 2, 1, 3, 1, 3, 0, 4, 3, 4, 3, 1, 3,
       4, 1, 0, 3, 3, 4, 4, 0, 0, 1, 4, 0, 1, 3, 0, 3, 3, 3, 2, 3, 3, 3,
       3, 0, 2, 0, 1, 2, 1, 3, 0, 0, 3, 2, 2, 3, 3, 3, 1, 1, 3, 1, 4, 1,
       4, 3, 4, 4, 0, 0, 3, 1, 3, 3, 0, 3, 4, 3, 0, 3, 3, 2, 3, 0, 2, 4,
       3, 3, 2, 3, 3, 4, 3, 2, 2, 1, 0, 1, 2, 3, 0, 0, 4, 1, 3, 1, 3, 3,
       1, 0, 1, 4, 0, 0, 3, 3, 0, 3, 3, 0, 0, 3, 2, 0, 3, 0, 3, 2, 2, 0,
       0, 0, 1, 0, 0, 0, 3, 1, 1, 3, 1, 0, 0, 0, 1, 0, 3, 0, 3, 0, 0, 0,
       3, 1, 4, 0, 1, 2, 4, 0, 4, 0, 0, 0, 0, 0, 2, 4, 0, 3, 3, 0, 3, 4,
       1, 3, 3, 1, 1, 2, 3, 0, 3, 1, 3, 0, 1, 0, 1, 4, 3, 3, 3, 0, 1, 4,
       0, 3, 3, 3, 0, 0, 3, 0, 4, 2, 1, 4, 4, 1, 0, 4, 1, 1, 4, 4, 0, 0,
       3, 4, 1, 3, 0, 0, 4, 3, 1, 0, 1, 1, 1, 3, 1, 2, 2, 1, 1, 4, 1, 0,
       1, 1, 3, 4, 0, 3, 0, 0, 1, 0, 4, 3, 0, 0, 0, 3, 1, 0, 1, 3, 0, 0,
       4, 0, 3, 0, 3, 0, 3, 0, 0, 0, 0, 0, 0, 4, 1, 0, 2, 3, 0, 4, 0, 0,
       0, 0, 0, 0, 0, 0, 3, 0, 0, 1, 2, 0, 3, 1, 3, 2, 0, 3, 0, 0, 3, 3,
       3, 3, 3, 0, 0, 1, 3, 1, 3, 1, 3, 3, 3, 1, 3, 3, 0, 0, 0, 3, 0, 2,
       1, 4, 0, 0, 3, 0, 0, 3, 0, 4, 3, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1,
       0, 3, 2, 4, 0, 2, 3, 0, 4, 3, 0, 4, 0, 0, 3, 1, 3, 3, 3, 1, 3, 0,
       3, 0, 0, 0, 2, 0, 0, 3, 0, 3, 0, 4, 1, 0, 0, 3, 4, 4, 4, 3, 3, 2,
       0, 4, 0, 2, 3, 3, 3, 4, 3, 4, 0, 0, 2, 3, 1, 1, 0, 3, 3, 0, 0, 0,
       0, 4, 0, 0, 1, 3, 1, 0, 0, 1, 4, 1, 4, 3, 0, 4, 4, 4, 4, 4, 1, 1,
       4, 0, 0, 4, 4, 0, 1, 3, 3, 4, 0, 4, 3, 0, 4, 0, 3, 2, 0, 0, 3, 0,
       3, 3, 0, 1, 3, 4, 4, 0, 1, 0, 4, 0, 3, 0, 1, 1, 3, 2, 3, 1, 2, 2,
       3, 3, 0, 2, 0, 0, 2, 0, 0, 3, 1, 1, 3, 3, 2, 1, 0, 3, 0, 3, 3, 0,
       3, 3, 3, 3, 0, 1, 3, 1, 3, 2, 4, 3, 3, 4, 4, 4, 3, 4, 0, 0, 0, 4,
       4, 3, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 2, 2, 1, 1, 4, 2, 4],
      dtype=int32)
```

Figura 13: División de los elementos en cada cluster del algoritmo particional.

```
clusterP
0    172
1    100
2     56
3    156
4     85
Name: clusterP, dtype: int64
```

Figura 14: Cantidad de los elementos en cada cluster del algoritmo particional.

| clusterP | Texture | Area | Smoothness | Compactness | Symmetry | FractalDimension |
|---|---|---|---|---|---|---|
| **0** | 16.297442 | 514.286628 | 0.085941 | 0.062736 | 0.164908 | 0.059056 |
| **1** | 21.837500 | 1228.067000 | 0.100036 | 0.140695 | 0.187407 | 0.059186 |
| **2** | 20.364643 | 705.283929 | 0.115617 | 0.204721 | 0.226070 | 0.075936 |
| **3** | 17.734615 | 476.337179 | 0.104744 | 0.107066 | 0.188042 | 0.066356 |
| **4** | 24.492706 | 559.569412 | 0.085045 | 0.074626 | 0.164491 | 0.059430 |

Figura 15: Centroides para cada cluster del algoritmo particional.

Haciendo una gráfica tridimensional de los elementos y los centros de cada cluster, vemos lo siguiente:

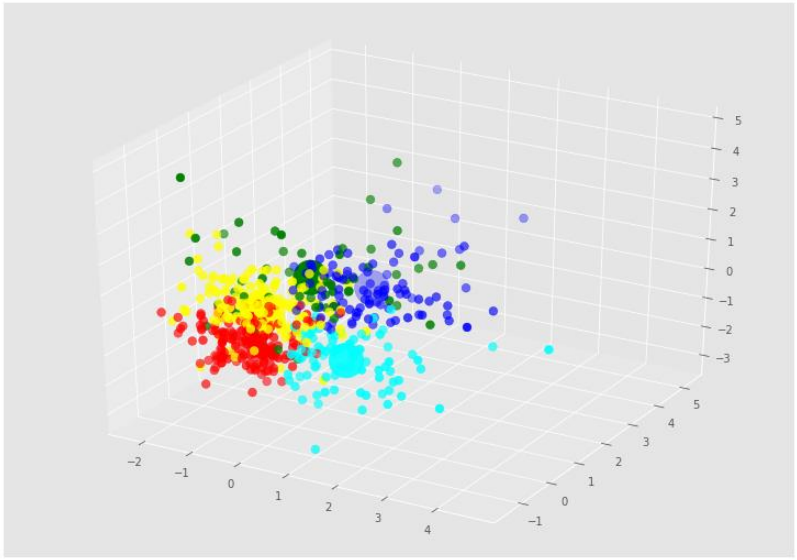

Figura 16: Visualización tridimensional de los elementos y centros de cada cluster con el algoritmo particional.

**Interpretación de los resultados obtenidos con el algoritmo particional:**

A diferencia de lo obtenido con el algoritmo jerárquico, el algoritmo particional, gracias al uso del método del codo, dividió los elementos del conjunto de datos en 5 clústeres. A continuación, definimos las características de cada uno de ellos:

**Clúster 0:** En este cluster se encuentran 172 pacientes cuyos tumores tienen una textura de 16 y un área de 514 unidades. La variación de la longitud del radio fue de 0.085, con un *compactness* (relación del perímetro al cuadrado entre el área menos uno) de 0.062, una simetría de 0.16 y una *fractal dimension* o aproximación de frontera de 0.059. El tamaño del tumor se encuentra en valores que nos harían pensar que se trata de un tumor benigno.

**Clúster 1:** En este cluster se encuentran 100 pacientes cuyos tumores tienen una textura de 21 y un área de 1228 unidades. La variación de la longitud del radio fue de 0.1, con un *compactness* (relación del perímetro al cuadrado entre el área menos uno) de 0.14, una simetría de 0.187 y una *fractal dimension* o aproximación de frontera de 0.059. El tamaño del tumor, en promedio, nos haría pensar que es un grupo donde hay altas probabilidades de que sean tumores malignos.

**Clúster 2:** En este cluster se encuentran 56 pacientes cuyos tumores tienen una textura de 20 y un área de 705 unidades. La variación de la longitud del radio fue de 0.115, con un *compactness* (relación del perímetro al cuadrado entre el área menos uno) de 0.2, una simetría de 0.22 y una *fractal dimension* o aproximación de frontera de 0.075. El tamaño del tumor, en promedio, nos haría pensar que los integrantes de este clúster tienen probabilidad de desarrollar un tumor maligno (se encuentran en valores intermedios entre los que tienen mayor probabilidad de tener un tumor maligno y los que tienen menor probabilidad de tener un tumor maligno).

**Clúster 3:** En este cluster se encuentran 156 pacientes cuyos tumores tienen una textura de 17 y un área de 476 unidades. La variación de la longitud del radio fue de 0.10, con un *compactness* (relación del perímetro al cuadrado entre el área menos uno) de 0.107, una simetría de 0.18 y una *fractal dimension* o aproximación de frontera de 0.066. El tamaño del tumor se encuentra en valores que nos harían pensar que se trata de un tumor benigno.

**Clúster 4:** En este cluster se encuentran 85 pacientes cuyos tumores tienen una textura de 24 y un área de 559 unidades. La variación de la longitud del radio fue de 0.085, con un *compactness* (relación del perímetro al cuadrado entre el área menos uno) de 0.074, una simetría de 0.164 y una *fractal dimension* o aproximación de frontera de 0.059. El tamaño del tumor se encuentra en valores que nos harían pensar que se trata de un tumor benigno, aunque, entre los grupos con área de tumor en valores bajos, es el grupo que tiene un tamaño de tumor promedio más alto (559 unidades).

## Conclusiones

A lo largo de esta práctica, tuvimos como principal objetivo el obtener grupos de pacientes con características similares, diagnosticadas con un tumor de mama, a través de clustering jerárquico y particional.

Para la primera parte de esta práctica, utilizamos algunas gráficas de la biblioteca *seaborn* para poder visualizar las variables a utilizar. Con el método de Pearson, obtuvimos la matriz de correlaciones que nos permitió discriminar entre las variables y determinar la dependencia de cada una con las demás. Habiendo hecho lo anterior, fue sencillo reducir la dimensionalidad de los datos para aplicar los algoritmos de clustering.

Primeramente, utilizamos el algoritmo jerárquico para obtener clústeres relacionados de pacientes. Al utilizar la métrica de distancia euclidiana, obtuvimos 4 clústeres diferentes que, al analizarlos, nos percatamos las similitudes entre cada elemento de ese cluster y el porqué de la división en dichos grupos.

Al utilizar el algoritmo particional, obtuvimos 5 clústeres diferentes a partir del resultado del método del codo. De igual manera que con el algoritmo jerárquico, se nota una distinción clara entre cada uno de los grupos creados.

Se realizó un análisis a mayor profundidad de cada uno de los clústeres creados con el algoritmo jerárquico y con el algoritmo particional en el desarrollo de la práctica. Para esta fuente de datos en particular, creo que no hace mucha diferencia el utilizar un algoritmo u otro. Si tuviéramos una fuente de datos con una cantidad mucho mayor de datos, sería más conveniente optar por utilizar el algoritmo particional.

En general, puedo concluir que se cumplieron los objetivos de la práctica y que ésta fue útil tanto para comprender de mejor manera los algoritmos de clustering jerárquico y de clustering particional, como para entender las diferencias y beneficios entre utilizar uno u otro.