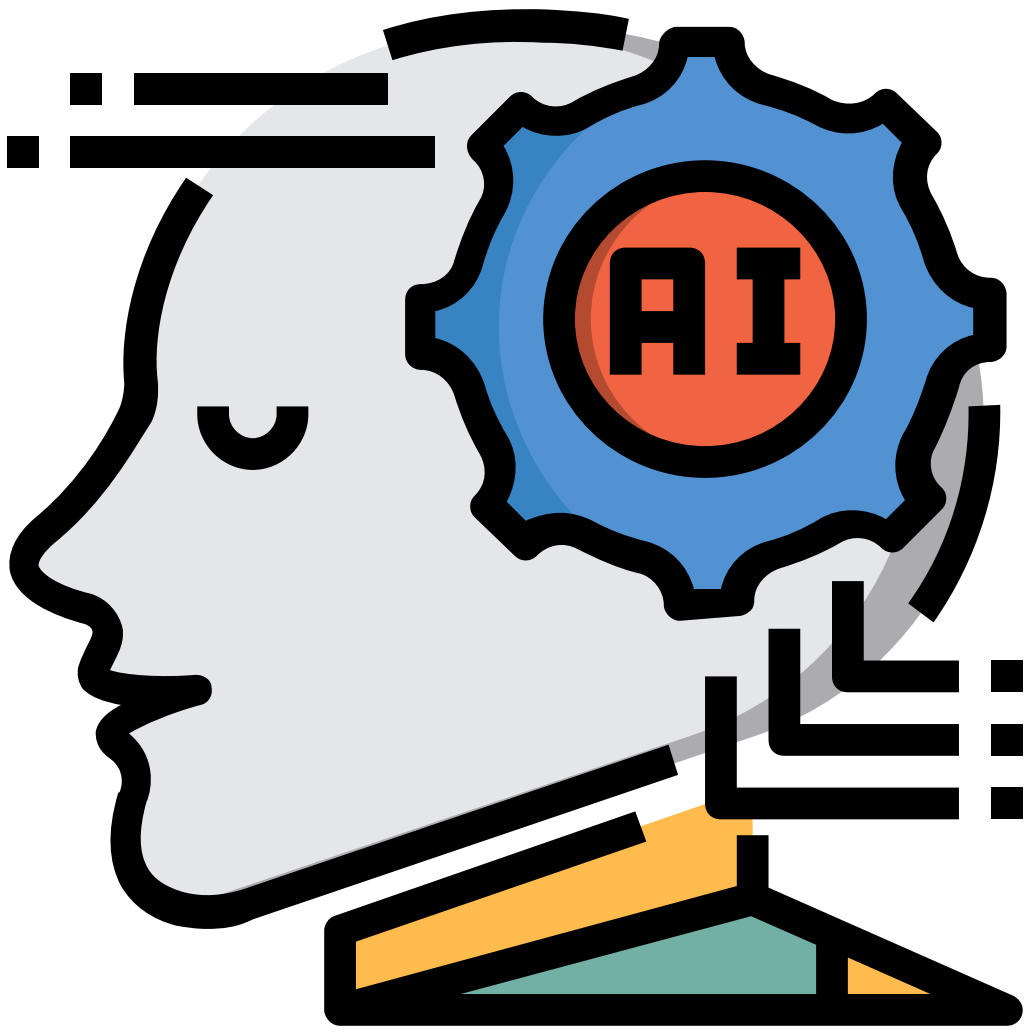
Universidad Nacional Autónoma de México
Facultad de Ingeniería

Inteligencia Artificial

PRÁCTICA 6. CLUSTERING: JERÁRQUICO Y PARTICIONAL

Casasola García Oscar
316123747
oscar.casasola.g7@gmail.com
Grupo 03

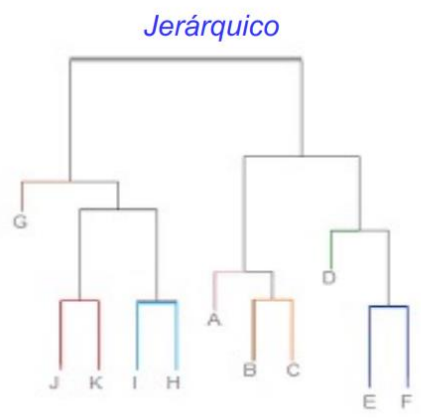


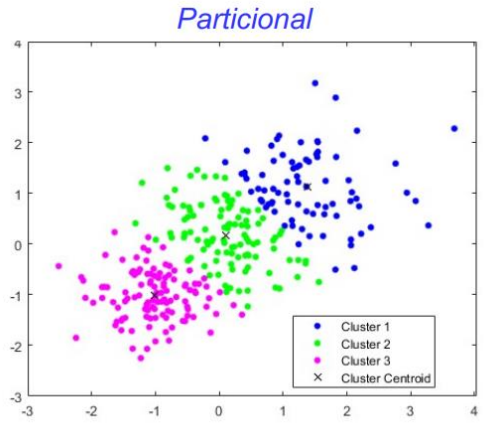
Profesor: Dr. Guillermo Gilberto Molero Castillo
Semestre 2022-1

Contenido

Contexto

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).

Variable	Descripción	Tipo
ID number	Identifica al paciente	Discreto
Diagnosis	Diagnostico (M=maligno, B=benigno)	Booleano
Radius	Media de las distancias del centro y puntos del perímetro	Continuo
Texture	Desviación estándar de la escala de grises	Continuo
Perimeter	Valor del perímetro del cáncer de mama	Continuo
Area	Valor del área del cáncer de mama	Continuo
Smoothness	Variación de la longitud del radio	Continuo
Compactness	Perímetro ^ 2 /Area - 1	Continuo
Concavity	Caída o gravedad de las curvas de nivel	Continuo
Concave points	Número de sectores de contorno cóncavo	Continuo
Symmetry	Simetría de la imagen	Continuo
Fractal dimension	"Aproximación de frontera" - 1	Continuo

Fuente: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

Preparación del entorno de ejecución

1) Importar las bibliotecas necesarias

```
import pandas as pd              # Para la manipulación y análisis de datos
import numpy as np               # Para crear vectores y matrices n dimensionales
import matplotlib.pyplot as plt  # Para la generación de gráficas a partir de los datos
import seaborn as sns            # Para la visualización de datos basado en matplotlib
%matplotlib inline
```

2) Importar los datos

Fuente de datos: WDBCOriginal.csv

```
from google.colab import files
files.upload()

# Para importar los datos desde Drive
#from google.colab import drive
#drive.mount('/content/drive')
```

```
BCancer = pd.read_csv("WDBCOriginal.csv")
BCancer
```

	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

```
BCancer.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   IDNumber         569 non-null    object 
 1   Diagnosis        569 non-null    object 
 2   Radius           569 non-null    float64
 3   Texture          569 non-null    float64
 4   Perimeter        569 non-null    float64
 5   Area             569 non-null    float64
 6   Smoothness       569 non-null    float64
 7   Compactness      569 non-null    float64
 8   Concavity        569 non-null    float64
 9   ConcavePoints    569 non-null    float64
 10  Symmetry         569 non-null    float64
 11  FractalDimension  569 non-null    float64
dtypes: float64(10), object(2)
memory usage: 53.5+ KB
```

```
print(BCancer.groupby('Diagnosis').size())
```

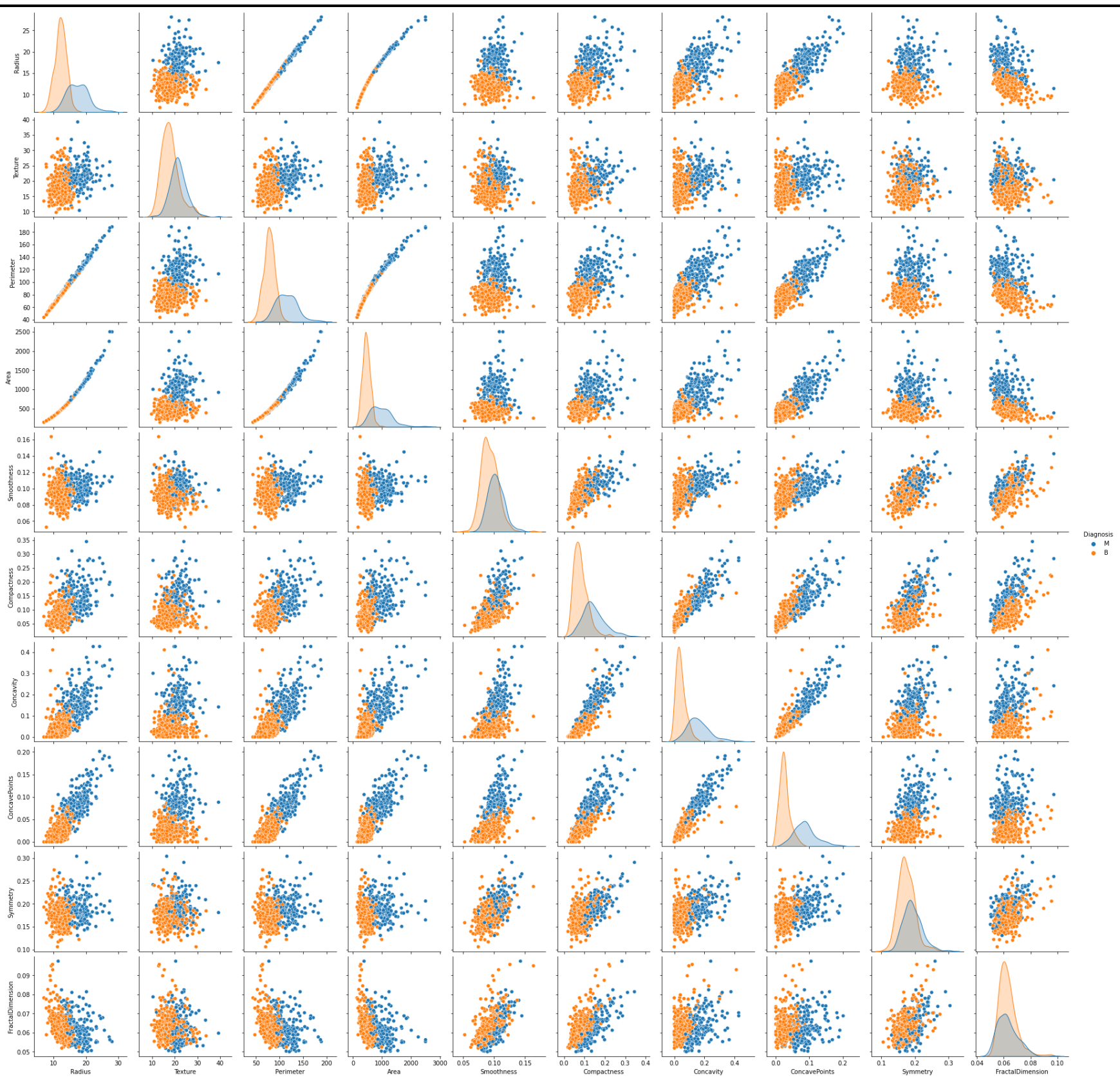
```
Diagnosis
B    357
M    212
dtype: int64
```

Se puede observar que **357 usuarios tienen un tumor benigno**, mientras que **212 usuarios tienen un tumor maligno**.

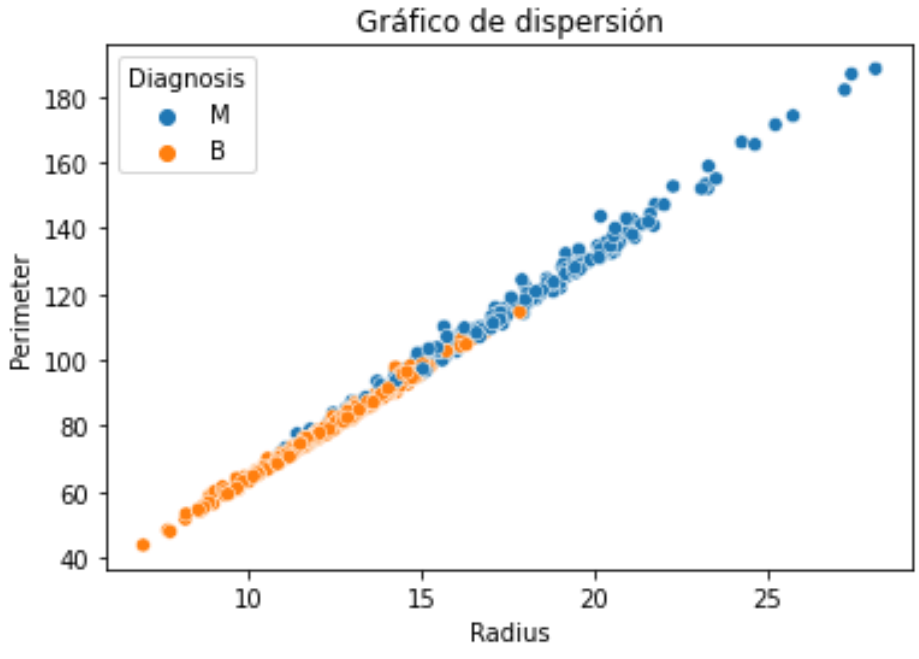
Selección de características

Evaluación visual

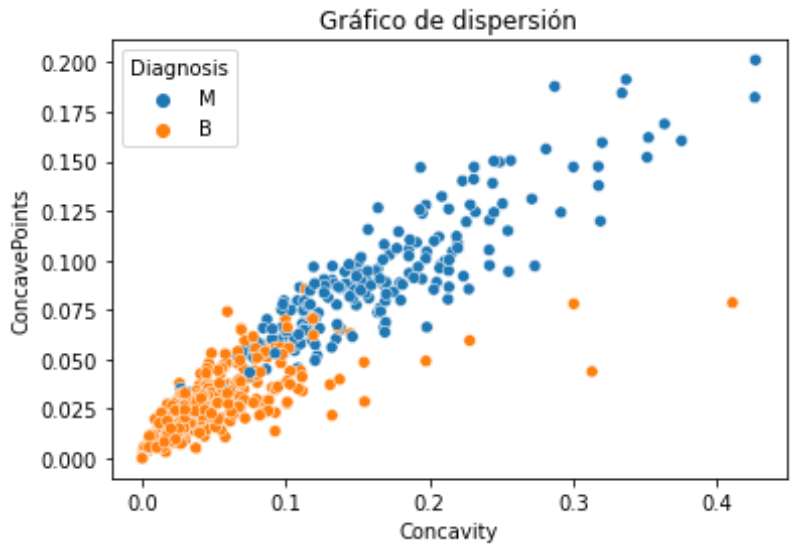
```
sns.pairplot(BCancer, hue='Diagnosis')
plt.show()
```




```
sns.scatterplot(x='Radius', y ='Perimeter', data=BCancer, hue='Diagnosis')
plt.title('Gráfico de dispersión')
plt.xlabel('Radius')
plt.ylabel('Perimeter')
plt.show()
```




```
sns.scatterplot(x='Concavity', y ='ConcavePoints', data=BCancer, hue='Diagnosis')
plt.title('Gráfico de dispersión')
plt.xlabel('Concavity')
plt.ylabel('ConcavePoints')
plt.show()
```



Matriz de correlaciones

Una matriz de correlaciones es útil para analizar la relación entre las variables numéricas. Se emplea la función corr().

```
CorrBCancer = BCancer.corr(method='pearson')
CorrBCancer
```

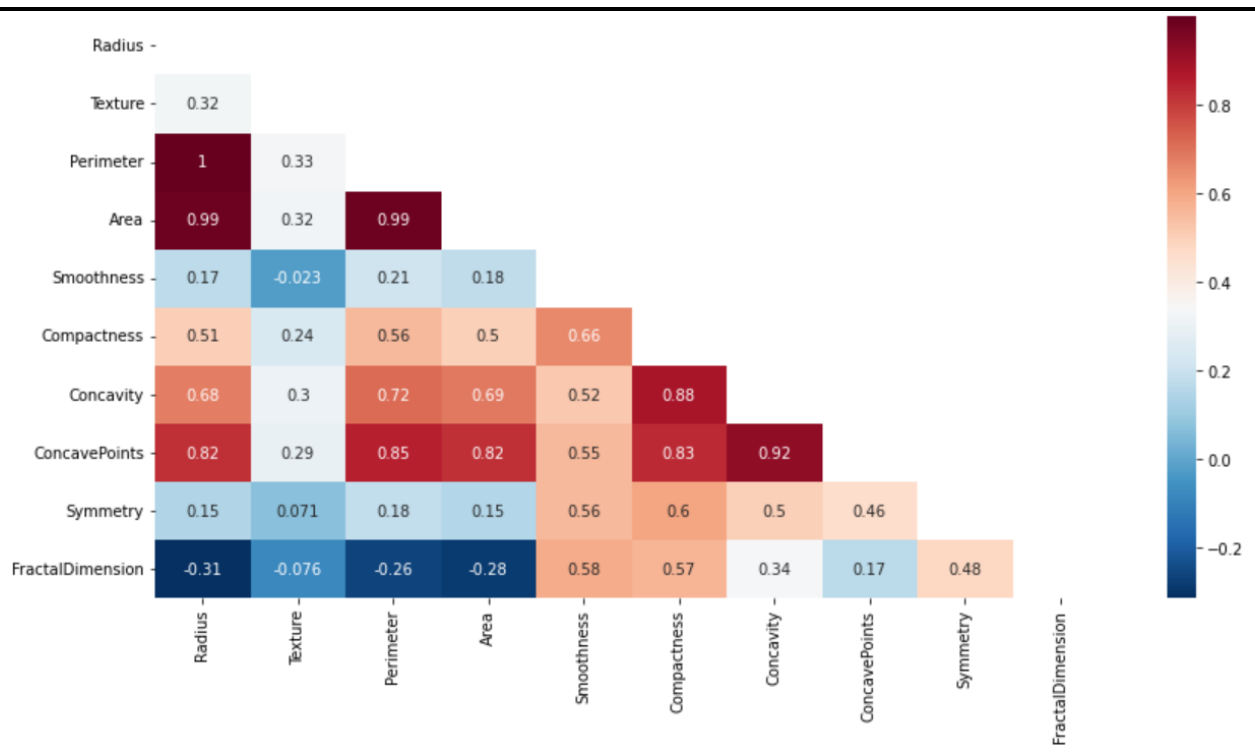
	Radius	Texture	Perimeter	Area	Smoothness	Compactness	Concavity	ConcavePoints	Symmetry	FractalDimension
Radius	1.000000	0.323782	0.997855	0.987357	0.170581	0.506124	0.676764	0.822529	0.147741	-0.311631
Texture	0.323782	1.000000	0.329533	0.321086	-0.023389	0.236702	0.302418	0.293464	0.071401	-0.076437
Perimeter	0.997855	0.329533	1.000000	0.986507	0.207278	0.556936	0.716136	0.850977	0.183027	-0.261477
Area	0.987357	0.321086	0.986507	1.000000	0.177028	0.498502	0.685983	0.823269	0.151293	-0.283110
Smoothness	0.170581	-0.023389	0.207278	0.177028	1.000000	0.659123	0.521984	0.553695	0.557775	0.584792
Compactness	0.506124	0.236702	0.556936	0.498502	0.659123	1.000000	0.883121	0.831135	0.602641	0.565369
Concavity	0.676764	0.302418	0.716136	0.685983	0.521984	0.883121	1.000000	0.921391	0.500667	0.336783
ConcavePoints	0.822529	0.293464	0.850977	0.823269	0.553695	0.831135	0.921391	1.000000	0.462497	0.166917
Symmetry	0.147741	0.071401	0.183027	0.151293	0.557775	0.602641	0.500667	0.462497	1.000000	0.479921
FractalDimension	-0.311631	-0.076437	-0.261477	-0.283110	0.584792	0.565369	0.336783	0.166917	0.479921	1.000000

```
print(CorrBCancer['Radius'].sort_values(ascending=False)[:10], '\n')   #Top 10 valores
```

```
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
```

Se muestra la correlación que tiene la variable **Radius** con las demás variables.

```
# Mapa de calor de la relación que existe entre variables
plt.figure(figsize=(14,7))
MatrizInf = np.triu(CorrBCancer)
sns.heatmap(CorrBCancer, cmap='RdBu_r', annot=True, mask=MatrizInf)
plt.show()
```



Selección de variables

1. Textura [Posición 3]
2. Área [Posición 5]
3. Smoothness [Pos. 6]
4. Compactness [Pos. 7]
5. Symmetry [Posición 10]
6. FractalDimension [Pos. 11]

```
MatrizVariables = np.array(BCancer[['Texture', 'Area', 'Smoothness', 'Compactness', 'Symmetry', 'FractalDimension']])
pd.DataFrame(MatrizVariables)
#MatrizHipoteca = BCancer.iloc[:, 0:9].values     #iloc para seleccionar filas y columnas según su posición
```

	0	1	2	3	4	5
0	10.38	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
565	28.25	1261.0	0.09780	0.10340	0.1752	0.05533
566	28.08	858.1	0.08455	0.10230	0.1590	0.05648
567	29.33	1265.0	0.11780	0.27700	0.2397	0.07016
568	24.54	181.0	0.05263	0.04362	0.1587	0.05884

569 rows × 6 columns

Aplicación de los algoritmos

Estandarización

Cuando se trabaja con clustering, dado que son algoritmos basados en distancias, es fundamental escalar los datos para que cada una de las variables contribuyan por igual en el análisis.

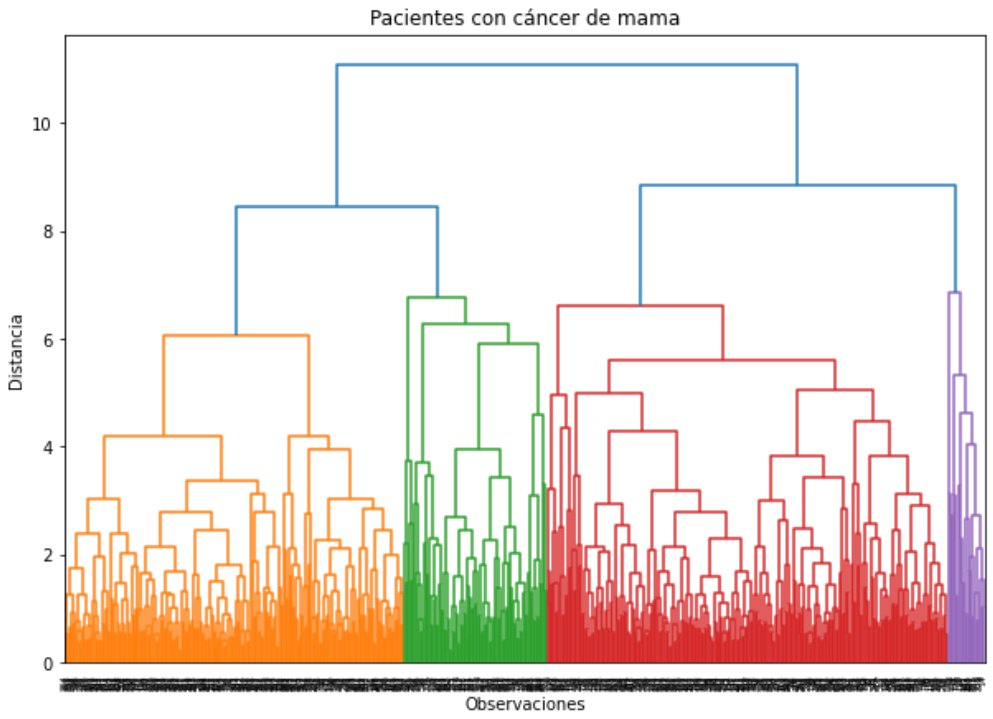
```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
estandarizar = StandardScaler()                              # Se instancia el objeto StandardScaler o MinMaxScaler
MEstandarizada = estandarizar.fit_transform(MatrizVariables)   # Se calculan la media y desviación y se escalan los datos

pd.DataFrame(MEstandarizada)  # Matriz estandarizada
```

	0	1	2	3	4	5
0	-2.073335	0.984375	1.568466	3.283515	2.217515	2.255747
1	-0.353632	1.908708	-0.826962	-0.487072	0.001392	-0.868652
2	0.456187	1.558884	0.942210	1.052926	0.939685	-0.398008
3	0.253732	-0.764464	3.283553	3.402909	2.867383	4.910919
4	-1.151816	1.826229	0.280372	0.539340	-0.009560	-0.562450
...	...	...	...	...	...	...
564	0.721473	2.343856	1.041842	0.219060	-0.312589	-0.931027
565	2.085134	1.723842	0.102458	-0.017833	-0.217664	-1.058611
566	2.045574	0.577953	-0.840484	-0.038680	-0.809117	-0.895587
567	2.336457	1.735218	1.525767	3.272144	2.137194	1.043695
568	1.221792	-1.347789	-3.112085	-1.150752	-0.820070	-0.561032

569 rows × 6 columns

Algoritmo: Ascendente Jerárquico

```
#Se importan las bibliotecas de clustering jerárquico para crear el árbol
import scipy.cluster.hierarchy as shc
from sklearn.cluster import AgglomerativeClustering
plt.figure(figsize=(10,7))
plt.title("Pacientes con cáncer de mama")
plt.xlabel('Observaciones')
plt.ylabel('Distancia')
Arbol = shc.dendrogram(shc.linkage(MEstandarizada, method='complete', metric='euclidean'))
#plt.axhline(y=7, color='orange', linestyle='--')
#Probar con otras mediciones de distancia (chebyshev, cityblock)
```



Se crean etiquetas en los clústeres

```
#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_
```

```
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,
       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,
```

```
BCancer['clusterH'] = MJerarquico.labels_
BCancer
```

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

569 rows × 13 columns

```
# Cantidad de elementos en los clústeres
```

```
BCancer.groupby(['clusterH'])['clusterH'].count()
```

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

Obtención de los centroides

```
CentroidesH = BCancer.groupby(['clusterH'])['Texture', 'Area', 'Smoothness', 'Compactness', 'Symmetry',
'FractalDimension'].mean()
CentroidesH
```

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

Interpretación de los datos obtenidos

NOTA IMPORTANTE: Para determinar si el tumor es maligno o benigno, nos basamos principalmente en el área, es decir, el tamaño del tumor, pero habría que presentarle estos datos a un especialista en el campo de la Medicina, en este caso, a un Oncólogo.

```
n = 0
colum = 0
while n < numClustersH.size:
    print("- Clúster número:", n)
    print("a) Conformado por un grupo de", numClustersH[n], "pacientes.")
    print("b) Con una desviación estándar de textura de", CentroidesH.iloc[n][colum].round(4),"píxeles")
    print("c) Con un área promedio de tumor de", CentroidesH.iloc[n][colum+1].round(4),"píxeles")
    if (CentroidesH.iloc[n][colum+1].round(4) > 600):
        print("d) El tamaño del tumor es muy grande, por ende, posiblemente este grupo de usuarios tengan un tumor maligno")
    else:
        print("d) El tamaño del tumor es relativamente pequeño, por ende, posiblemente este grupo de usuarios tengan un tumor
benigno")
    print("e) Con una suavidad promedio de", CentroidesH.iloc[n][colum+2].round(4),"píxeles")
    print("f) Con una relación compacta promedio de", CentroidesH.iloc[n][colum+3].round(4),"píxeles")
    print("g) Con una simetría promedio de", CentroidesH.iloc[n][colum+4].round(4),"píxeles")
    print("h) Y con una aproximación de frontera, dimensión fractal promedio de", CentroidesH.iloc[n][colum+5].round(4),"píxeles")
    n = n + 1
    print("-------------------------------------------------------------")
```

Clúster número: 0

a) Conformado por un grupo de 23 pacientes.
b) Con una desviación estándar de textura de 20.1335 píxeles
c) Con un área promedio de tumor de 775.5435 píxeles
d) El tamaño del tumor es muy grande, por ende, posiblemente este grupo de usuarios tengan un tumor maligno
e) Con una suavidad promedio de 0.1243 píxeles
f) Con una relación compacta promedio de 0.2422 píxeles
g) Con una simetría promedio de 0.2408 píxeles
h) Y con una aproximación de frontera, dimensión fractal promedio de 0.0778 píxeles

Clúster número: 1

a) Conformado por un grupo de 88 pacientes.
b) Con una desviación estándar de textura de 22.5406 píxeles
c) Con un área promedio de tumor de 1243.7284 píxeles
d) El tamaño del tumor es muy grande, por ende, posiblemente este grupo de usuarios tengan un tumor maligno
e) Con una suavidad promedio de 0.0984 píxeles
f) Con una relación compacta promedio de 0.1371 píxeles
g) Con una simetría promedio de 0.1826 píxeles
h) Y con una aproximación de frontera, dimensión fractal promedio de 0.0589 píxeles

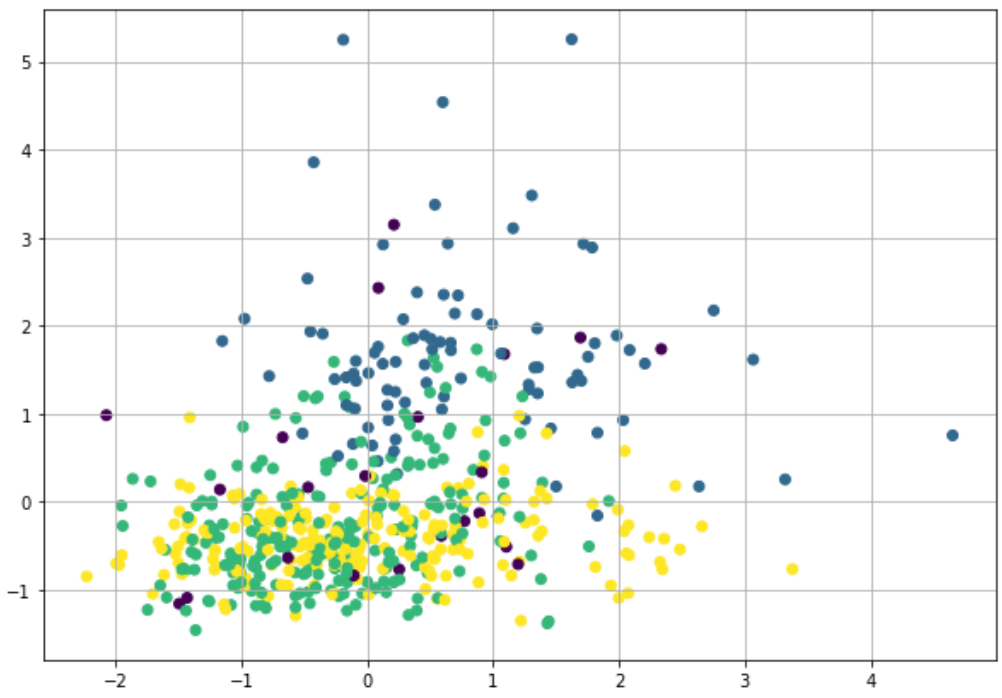
Clúster número: 2

a) Conformado por un grupo de 248 pacientes.
b) Con una desviación estándar de textura de 18.1675 píxeles
c) Con un área promedio de tumor de 561.3367 píxeles
d) El tamaño del tumor es relativamente pequeño, por ende, posiblemente este grupo de usuarios tengan un tumor benigno
e) Con una suavidad promedio de 0.1033 píxeles
f) Con una relación compacta promedio de 0.1142 píxeles
g) Con una simetría promedio de 0.1905 píxeles
h) Y con una aproximación de frontera, dimensión fractal promedio de 0.0657 píxeles

Clúster número: 3

a) Conformado por un grupo de 210 pacientes.
b) Con una desviación estándar de textura de 19.1601 píxeles
c) Con un área promedio de tumor de 505.4038 píxeles
d) El tamaño del tumor es relativamente pequeño, por ende, posiblemente este grupo de usuarios tengan un tumor benigno
e) Con una suavidad promedio de 0.0842 píxeles
f) Con una relación compacta promedio de 0.0638 píxeles
g) Con una simetría promedio de 0.163 píxeles
h) Y con una aproximación de frontera, dimensión fractal promedio de 0.0593 píxeles

Visualización gráfica

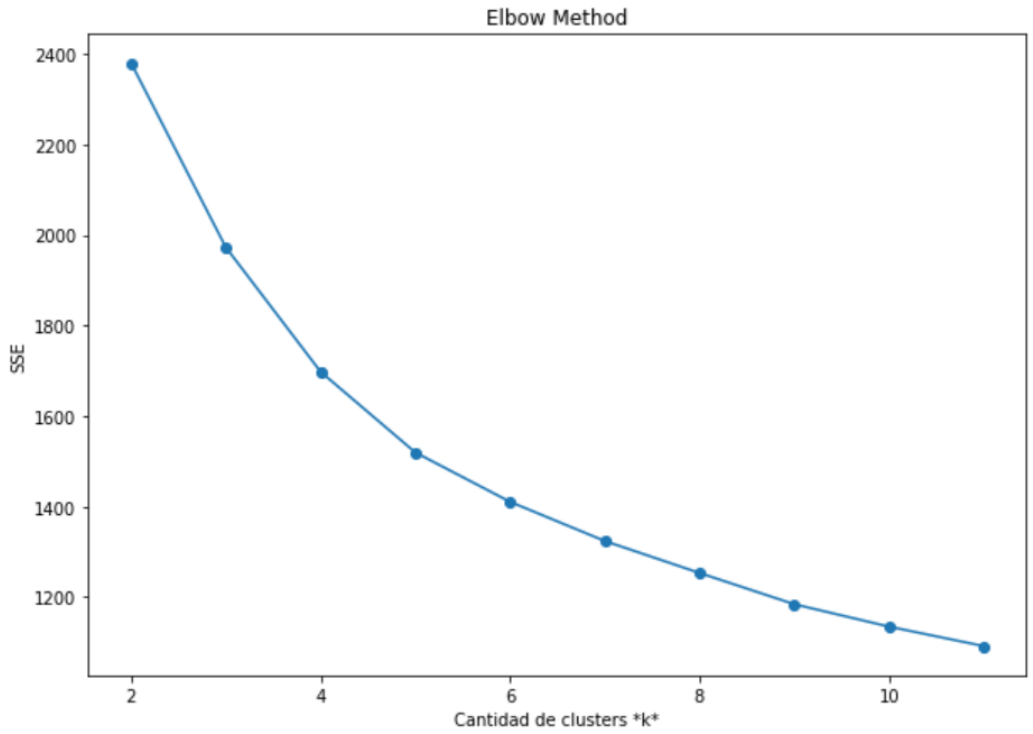


Algoritmo K-Means

Método del codo

```
# Se importan las bibliotecas
from sklearn.cluster import KMeans
from sklearn.metrics import pairwise_distances_argmin_min
```

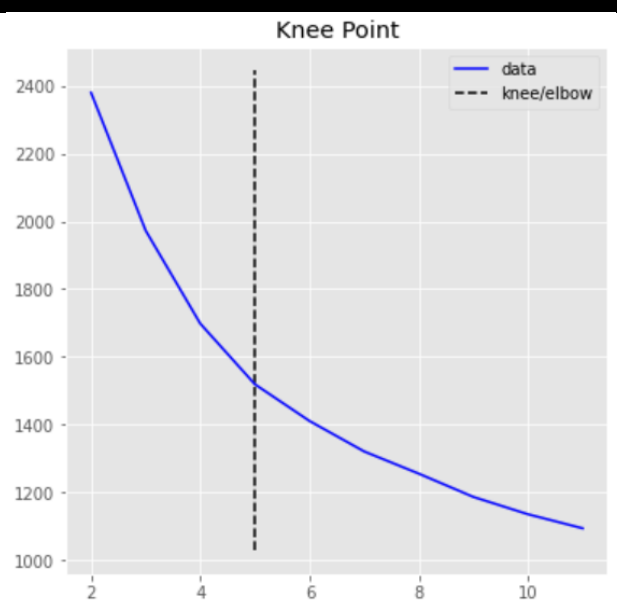
```
#Definición de k clústeres para K-Means
# Se utiliza random_state para inicializar el generador interno de números aleatorios
SSE = []
for i in range (2,12):
  km = KMeans(n_clusters=i, random_state=0)
  km.fit(MEstandarizada)
  SSE.append(km.inertia_)

#Se grafica SSE en función de k
plt.figure(figsize=(10,7))
plt.plot(range(2,12), SSE, marker='o')
plt.xlabel('Cantidad de clústeres *k*')
plt.ylabel('SSE')
plt.title('Elbow Method')
plt.show()
```



Observación. En la práctica, puede que no exista un codo afilado (codo agudo) y, como método heurístico, ese "codo" no siempre puede identificarse sin ambigüedades.

```
!pip install kneed #Función que nos permite calcular el número estimado de clústers


from kneed import KneeLocator
kl = KneeLocator(range(2,12), SSE, curve="convex", direction="decreasing")
kl.elbow #El punto de inflexión se encuentra en el cluster 5
```

```
plt.style.use('ggplot')
kl.plot_knee()
```



Creando las etiquetas en los clústeres

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

```
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)
```

```
BCancer['clusterP'] = MParticional.labels_
BCancer
```

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

569 rows × 14 columns

```
#Cantidad de elementos en los clusters
BCancer.groupby(['clusterP'])['clusterP'].count()
```

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

Obtención de los centroides

```
CentroidesP = BCancer.groupby(['clusterP'])['Texture', 'Area', 'Smoothness', 'Compactness', 'Symmetry', 'FractalDi
mension'].mean()
CentroidesP
```

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

Interpretación de los clústeres obtenidos

NOTA IMPORTANTE: Para determinar si el tumor es maligno o benigno, nos basamos principalmente en el área, es decir, el tamaño del tumor, pero habría que presentarle estos datos a un especialista en el campo de la Medicina, en este caso, a un Oncólogo.

```
n = 0
colum = 0
while n < numClustersP.size:
    print("- Clúster número:", n)
    print("a) Conformado por un grupo de", numClustersP[n], "pacientes.")
    print("b) Con una desviación estándar de textura de", CentroidesP.iloc[n][colum].round(4),"píxeles")
    print("c) Con un área promedio de tumor de", CentroidesP.iloc[n][colum+1].round(4),"píxeles")
    if (CentroidesP.iloc[n][colum+1].round(4) > 600):
        print("d) El tamaño del tumor es muy grande, por ende, posiblemente este grupo de usuarios tengan un tumor
 maligno")
    else:
        print("d) El tamaño del tumor es relativamente pequeño, por ende, posiblemente este grupo de usuarios teng
an un tumor benigno")
    print("e) Con una suavidad promedio de", CentroidesP.iloc[n][colum+2].round(4),"píxeles")
    print("f) Con una relación compacta promedio de", CentroidesP.iloc[n][colum+3].round(4),"píxeles")
    print("g) Con una simetría promedio de", CentroidesP.iloc[n][colum+4].round(4),"píxeles")
    print("h) Y con una aproximación de frontera, dimensión fractal promedio de", CentroidesP.iloc[n][colum+5].rou
nd(4),"píxeles")
    n = n + 1
    print("-----------------------------------------------------------------")
```

Clúster número: 0

a) Conformado por un grupo de 172 pacientes.
b) Con una desviación estándar de textura de 16.2974 píxeles
c) Con un área promedio de tumor de 514.2866 píxeles

d) El tamaño del tumor es relativamente pequeño, por ende, posiblemente este grupo de usuarios tengan un tumor benigno

e) Con una suavidad promedio de 0.0859 píxeles
f) Con una relación compacta promedio de 0.0627 píxeles
g) Con una simetría promedio de 0.1649 píxeles
h) Y con una aproximación de frontera, dimensión fractal promedio de 0.0591 píxeles

Clúster número: 1

a) Conformado por un grupo de 100 pacientes.
b) Con una desviación estándar de textura de 21.8375 píxeles
c) Con un área promedio de tumor de 1228.067 píxeles
d) El tamaño del tumor es muy grande, por ende, posiblemente este grupo de usuarios tengan un tumor maligno
e) Con una suavidad promedio de 0.1 píxeles
f) Con una relación compacta promedio de 0.1407 píxeles
g) Con una simetría promedio de 0.1874 píxeles
h) Y con una aproximación de frontera, dimensión fractal promedio de 0.0592 píxeles

Clúster número: 2

a) Conformado por un grupo de 56 pacientes.
b) Con una desviación estándar de textura de 20.3646 píxeles
c) Con un área promedio de tumor de 705.2839 píxeles
d) El tamaño del tumor es muy grande, por ende, posiblemente este grupo de usuarios tengan un tumor maligno
e) Con una suavidad promedio de 0.1156 píxeles
f) Con una relación compacta promedio de 0.2047 píxeles
g) Con una simetría promedio de 0.2261 píxeles
h) Y con una aproximación de frontera, dimensión fractal promedio de 0.0759 píxeles

Clúster número: 3

a) Conformado por un grupo de 156 pacientes.
b) Con una desviación estándar de textura de 17.7346 píxeles
c) Con un área promedio de tumor de 476.3372 píxeles
d) El tamaño del tumor es relativamente pequeño, por ende, posiblemente este grupo de usuarios tengan un tumor benigno
e) Con una suavidad promedio de 0.1047 píxeles
f) Con una relación compacta promedio de 0.1071 píxeles
g) Con una simetría promedio de 0.188 píxeles
h) Y con una aproximación de frontera, dimensión fractal promedio de 0.0664 píxeles

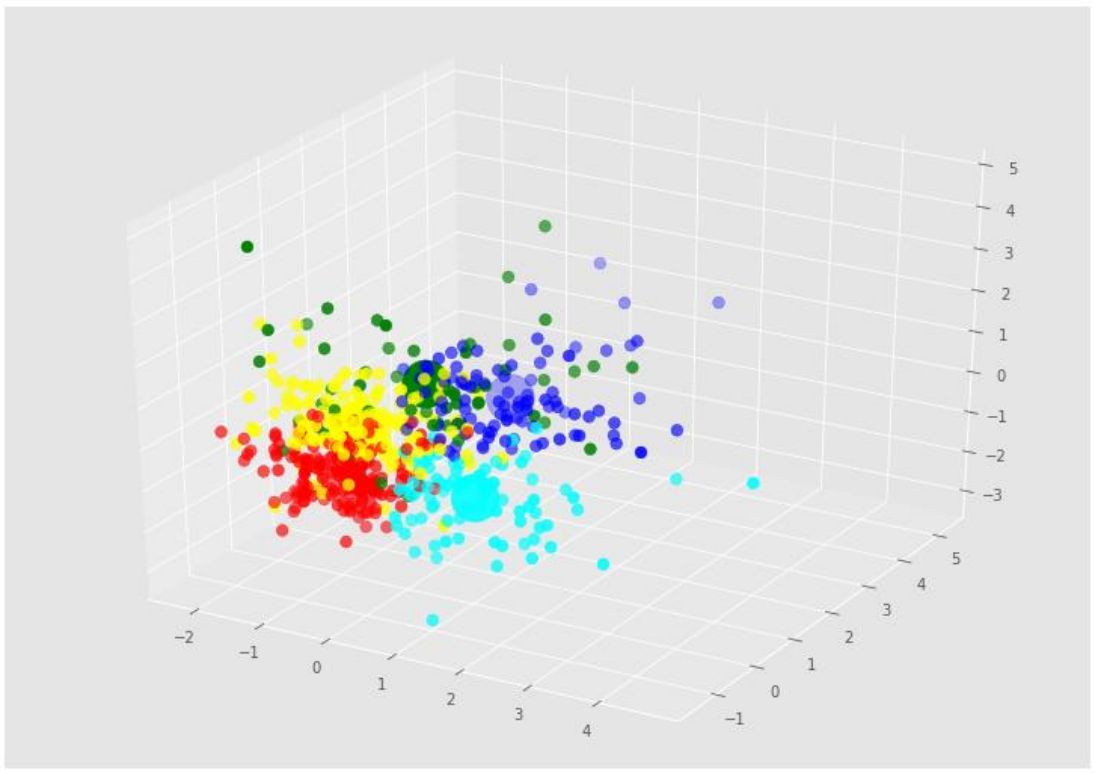
Clúster número: 4

a) Conformado por un grupo de 85 pacientes.
b) Con una desviación estándar de textura de 24.4927 píxeles
c) Con un área promedio de tumor de 559.5694 píxeles
d) El tamaño del tumor es relativamente pequeño, por ende, posiblemente este grupo de usuarios tengan un tumor benigno
e) Con una suavidad promedio de 0.085 píxeles
f) Con una relación compacta promedio de 0.0746 píxeles
g) Con una simetría promedio de 0.1645 píxeles
h) Y con una aproximación de frontera, dimensión fractal promedio de 0.0594 píxeles

Visualización gráfica

```
# Gráfica de los elementos y los centros de los clusters
from mpl_toolkits.mplot3d import Axes3D
plt.rcParams['figure.figsize'] = (10, 7)
plt.style.use('ggplot')
colores=['red', 'blue', 'green', 'yellow','cyan']
asignar=[]
for row in MParticional.labels_:
    asignar.append(colores[row])

fig = plt.figure()
ax = Axes3D(fig)
ax.scatter(MEstandarizada[:, 0],
           MEstandarizada[:, 1],
           MEstandarizada[:, 2], marker='o', c=asignar, s=60)
ax.scatter(MParticional.cluster_centers_[:, 0],
           MParticional.cluster_centers_[:, 1],
```

```
        MParticional.cluster_centers_[:, 2], marker='o', c=colores, s=1000)
plt.show()
```



Conclusiones

En esta práctica pude aprender cómo obtener grupos de pacientes con características similares, diagnosticadas con un tumor de mama, a través de clustering jerárquico y particional. Generé los clústeres a partir de una fuente de datos y en total obtuve cuatro grupos con el algoritmo de ascendente jerárquico y cinco con el algoritmo K-Means.

Como ambos algoritmos están basados en distancias, tuve que escalar los datos, de tal manera que cada variable contribuyera de igual manera en el análisis, es decir, que ninguna variable pesara más que la otra.

Finalmente, implementé ambos algoritmos, los cuales pude analizar y dar mis conclusiones sobre si cada grupo tenía potencialmente un tumor maligno o benigno, pero basándome en los criterios establecidos en clase, los cuales eran determinar si el tumor era potencialmente benigno o maligno basándonos en el tamaño de este, es decir, en el área del tumor.

En conclusión, pude aprender y visualizar de mejor manera la aplicación que tienen las distancias en el análisis de datos, de forma específica en el tema de clustering, y pude poner en práctica los conocimientos adquiridos en las prácticas pasadas.

Link de Google Colab

OCG-Práctica6-JerárquicoParticional.ipynb - Colaboratory (google.com)