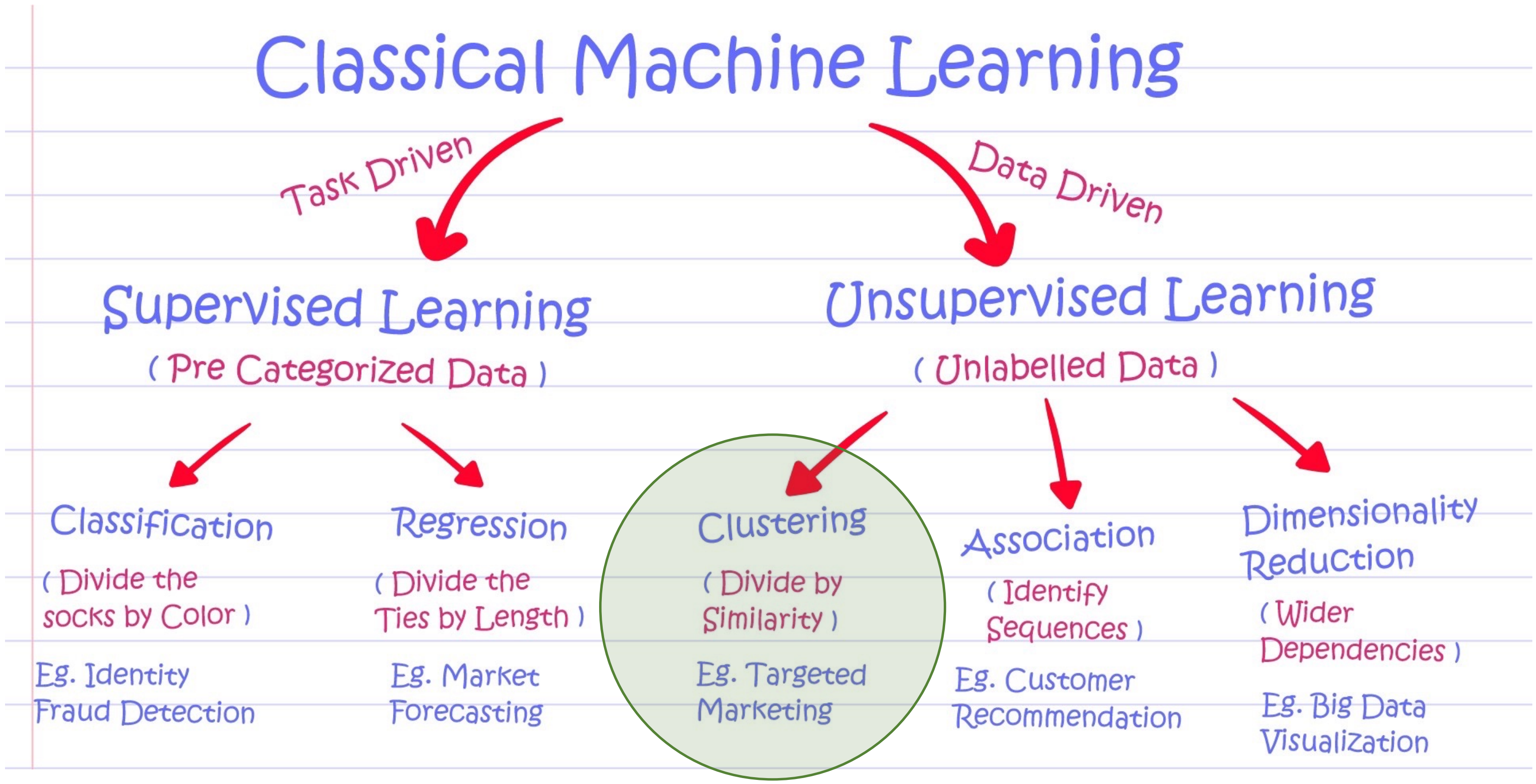
Universidad Nacional Autónoma de México
Facultad de Ingeniería

Clustering: Jerárquico y Particional
Práctica 6

Guillermo Molero-Castillo
guillermo.molero@ingenieria.unam.edu

Octubre, 2021

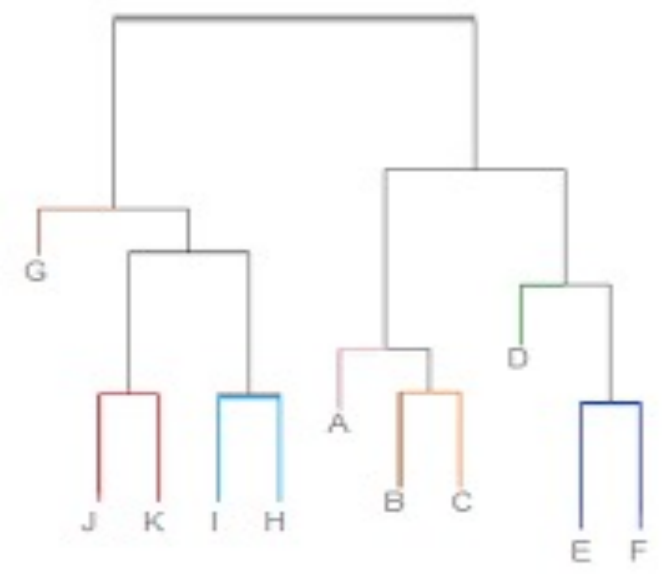
Práctica


Classical Machine Learning
Task Driven
Data Driven
Supervised Learning
(Pre Categorized Data)
Unsupervised Learning
(Unlabelled Data)
Classification
(Divide the socks by Color)
Eg. Identity Fraud Detection
Regression
(Divide the Ties by Length)
Eg. Market Forecasting
Clustering
(Divide by Similarity)
Eg. Targeted Marketing
Association
(Identify Sequences)
Eg. Customer Recommendation
Dimensionality Reduction
(Wider Dependencies)
Eg. Big Data Visualization

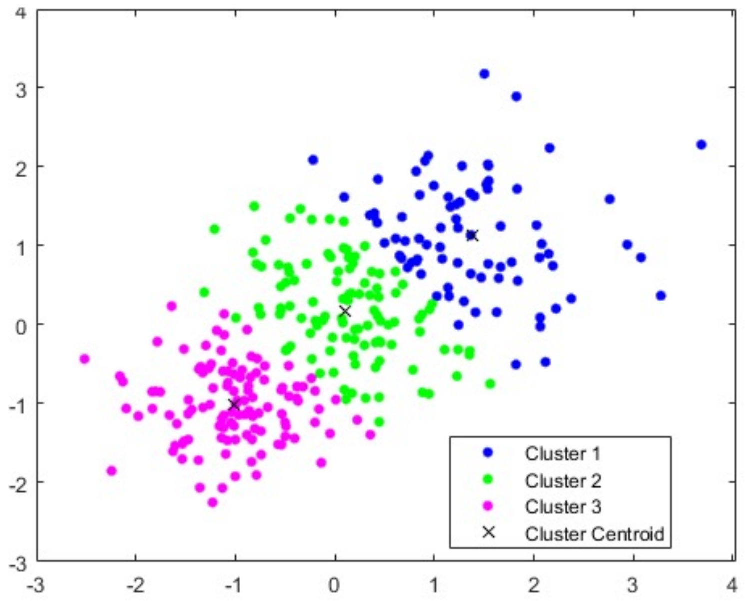
Objetivo

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

Jerárquico



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)

Práctica

1. Importar las bibliotecas y los datos

```
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
```

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

#from google.colab import drive
#drive.mount('/content/drive')
```

Práctica

1. Importar las bibliotecas y los datos

```python
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						0.1590	0.05648
567	P-927241	M	20.60	29.33	140.10						0.2397	0.07016
568	P-92751	B	7.76	24.54	47.92						0.1587	0.05884

569 rows × 12 columns

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

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

1. Importar las bibliotecas y los datos

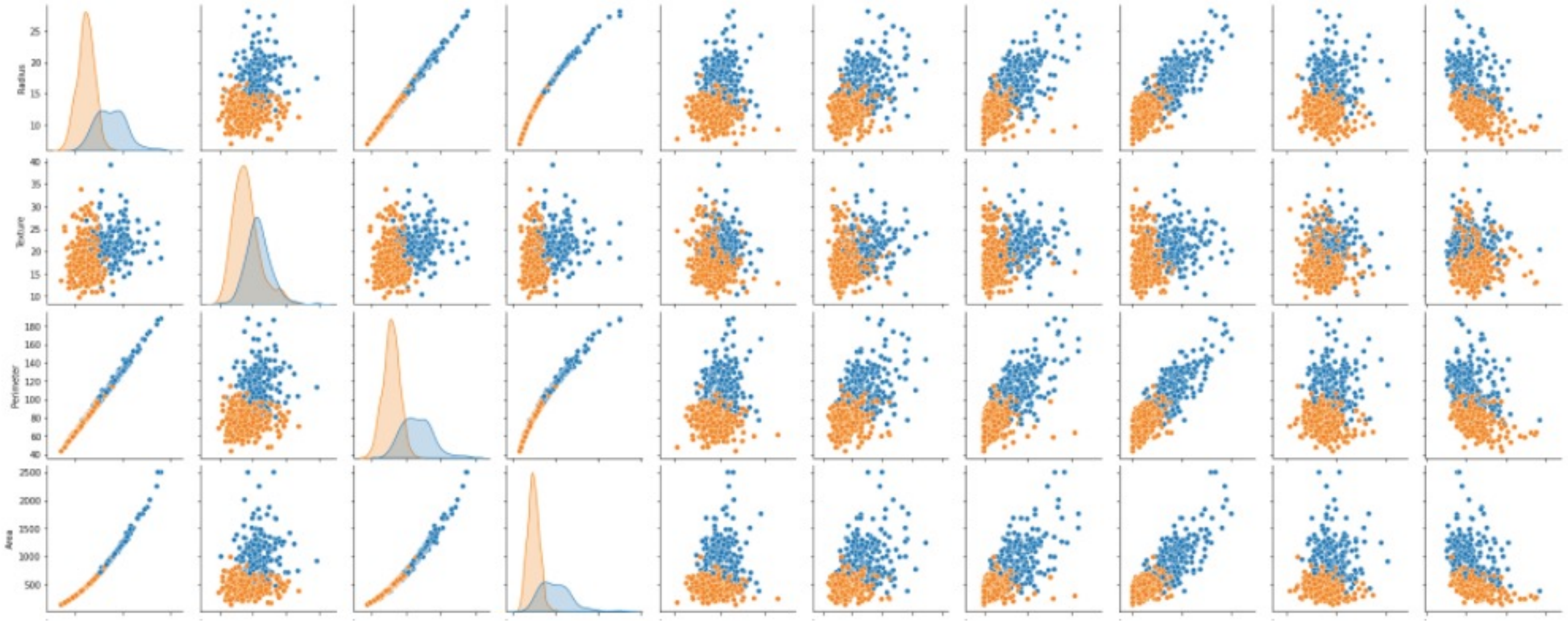
```
BCancer.info()
```

```
<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
```

2. Selección de características

Evaluación visual

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

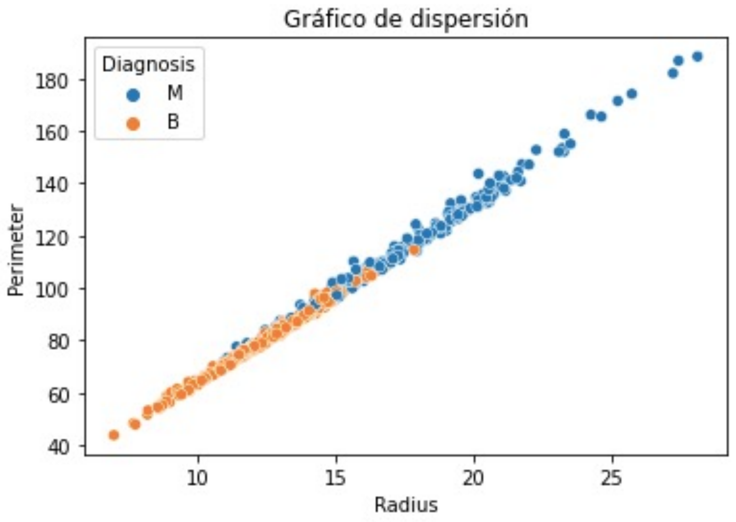


Práctica

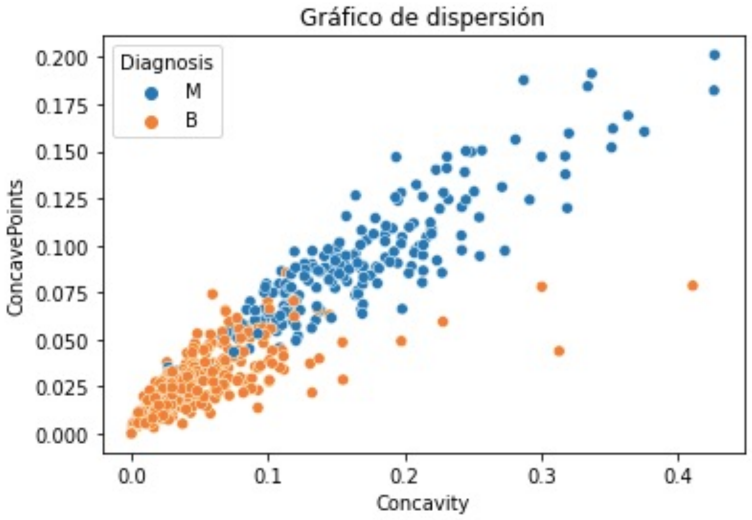
2. Selección de características

Evaluación visual

```
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()
```



2. Selección de características

Matriz de correlaciones

```
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

Práctica

2. Selección de características

Matriz de correlaciones

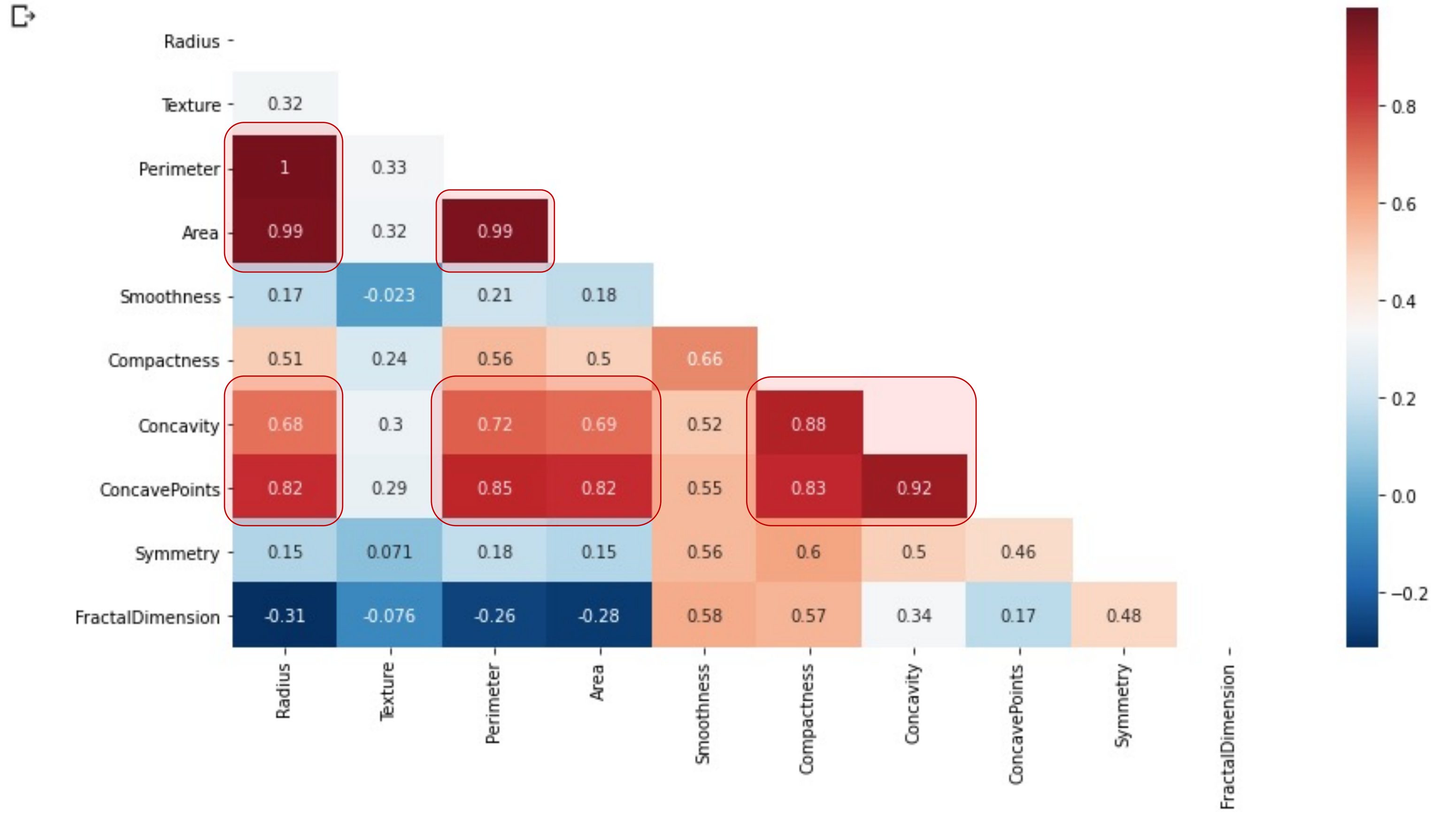
```
print(CorrBCancer['Radius'].sort_values(ascending=False)[:10], '\n')   #Top 10 valores
```

```
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
```

Práctica

2. Selección de características

```python
plt.figure(figsize=(14,7))
MatrizInf = np.triu(CorrBCancer)
sns.heatmap(CorrBCancer, cmap='RdBu_r', annot=True, mask=MatrizInf)
plt.show()
```



Varibles seleccionadas:

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

2. Selección de características

Elección de variables

```
MatrizVariables = np.array(BCancer[['Texture', 'Area', 'Smoothness', 'Compactness', 'Symmetry', 'FractalDimension']])
pd.DataFrame(MatrizVariables)
#MatrizVariables = BCancer.iloc[:, [3, 5, 6, 7, 10, 11]].values  #iloc para seleccionar filas y columnas
```

	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

3. Estandarización de datos

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
estandarizar = StandardScaler()                               # Se instancia el objeto StandardScaler o MinMaxScaler
MEstandarizada = estandarizar.fit_transform(MatrizVariables)   # Sescalan los datos
pd.DataFrame(MEstandarizada)
```

	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

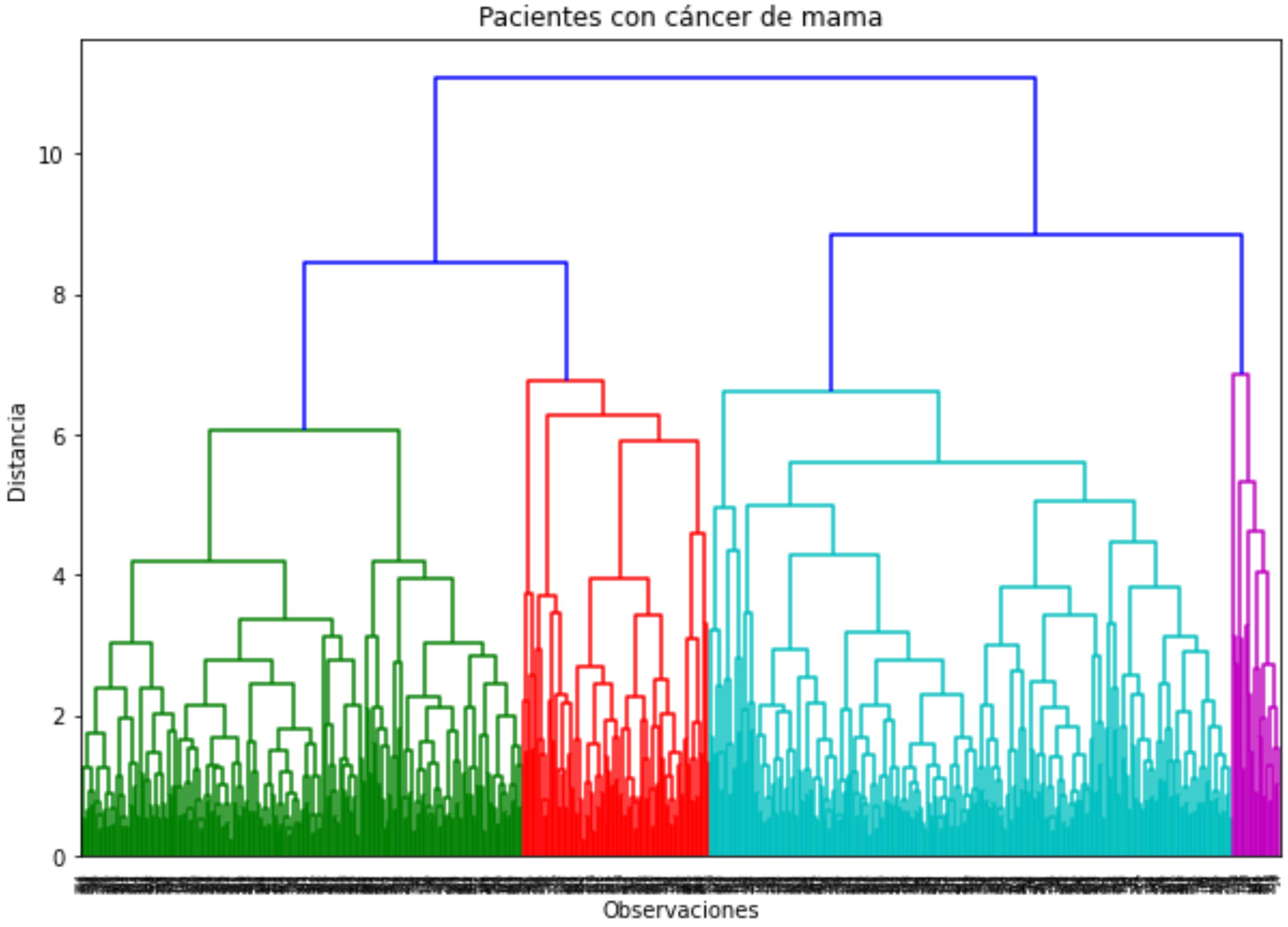
Clustering Jerárquico

Algoritmo: Ascendente Jerárquico

4. 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 medciones de distancia (euclidean, chebyshev, cityblock)
```

4. Algoritmo: Ascendente Jerárquico



Práctica

4. Algoritmo: Ascendente Jerárquico

Se crean las etiquetas en los clústeres

```python
#Se crean las etiquetas de los elementos en los clusters
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,
       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])
```

4. Algoritmo: Ascendente Jerárquico

Se crean las etiquetas en los clústeres

```
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	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.1752	0.05533	1
566	P-926954	M	16.60	28.08	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.1590	0.05648	3
567	P-927241	M	20.60	29.33	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.2397	0.07016	0
568	P-92751	B	7.76	24.54	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.1587	0.05884	3

569 rows × 13 columns

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

```
clusterH
0      23
1      88
2     248
3     210
```

Práctica

4. Algoritmo: Ascendente Jerárquico

Se crean las etiquetas en los clústeres

```
BCancer[BCancer.clusterH == 0]
```

	IDNumber	Diagnosis	Radius	Texture	Perimeter	Area	Smoothness	Compactness	Concavity	ConcavePoints	Symmetry	FractalDimension	clusterH
0	P-842302	M	17.990	10.38	122.80	1001.0	0.1184	0.2776	0.30010	0.14710	0.2419	0.07871	0
3	P-84348301	M	11.420	20.38	77.58	386.1	0.1425	0.2839	0.24140	0.10520	0.2597	0.09744	0
8	P-844981	M	13.000	21.82	87.50	519.8	0.1273	0.1932	0.18590	0.09353	0.2350	0.07389	0
9	P-84501001	M	12.460	24.04	83.97	475.9	0.1186	0.2396	0.22730	0.08543	0.2030	0.08243	0
14	P-84667401	M	13.730	22.61	93.60	578.3	0.1131	0.2293	0.21280	0.08025	0.2069	0.07682	0
22	P-8511133	M	15.340	14.26	102.50	704.4	0.1073	0.2135	0.20770	0.09756	0.2521	0.07032	0
25	P-852631	M	17.140	16.40	116.00	912.7	0.1186	0.2276	0.22290	0.14010	0.3040	0.07413	0
78	P-8610862	M	20.180	23.97	143.70	1245.0	0.1286	0.3454	0.37540	0.16040	0.2906	0.08142	0
108	P-86355	M	22.270	19.67	152.80	1509.0	0.1326	0.2768	0.42640	0.18230	0.2556	0.07039	0
122	P-865423	M	24.250	20.20	166.20	1761.0	0.1447	0.2867	0.42680	0.20120	0.2655	0.06877	0
146	P-869691	M	11.800	16.58	78.99	432.0	0.1091	0.1700	0.16590	0.07415	0.2678	0.07371	0
181	P-873593	M	21.090	26.57	142.70	1311.0	0.1141	0.2832	0.24870	0.14960	0.2395	0.07398	0

4. Algoritmo: Ascendente Jerárquico

Obtención de los centroides

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

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Indexing with multiple keys (implicitly converted to
  """Entry point for launching an IPython kernel.
```

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

4. Algoritmo: Ascendente Jerárquico

Interpretación

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

Clúster 0: Conformado por 23 pacientes con indicios de cáncer maligno por el tamaño del tumor, con un área promedio de tumor de 775 pixeles y una desviación estándar de textura de 20 pixeles. Aparentemente es un tumor compacto (0.24 pixeles), cuya suavidad alcanza 0.12 pixeles, una simetría de 0.24 y una aproximación de frontera, dimensión fractal, promedio de 0.077 pixeles.

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

```
clusterH
0     23
1     88
2    248
3    210
```

…

4. Algoritmo: Ascendente Jerárquico

Interpretación

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

Clúster 3: Es un grupo formado por 210 pacientes con el menor tamaño de tumor (posiblemente benigno), con un área promedio de tumor de 505 pixeles y una desviación estándar de textura de 19 pixeles. Es un tumor compacto (0.06 pixeles), cuya suavidad alcanza 0.08 pixeles, una simetría de 0.16 y una aproximación de frontera, dimensión fractal, promedio de 0.059 pixeles.

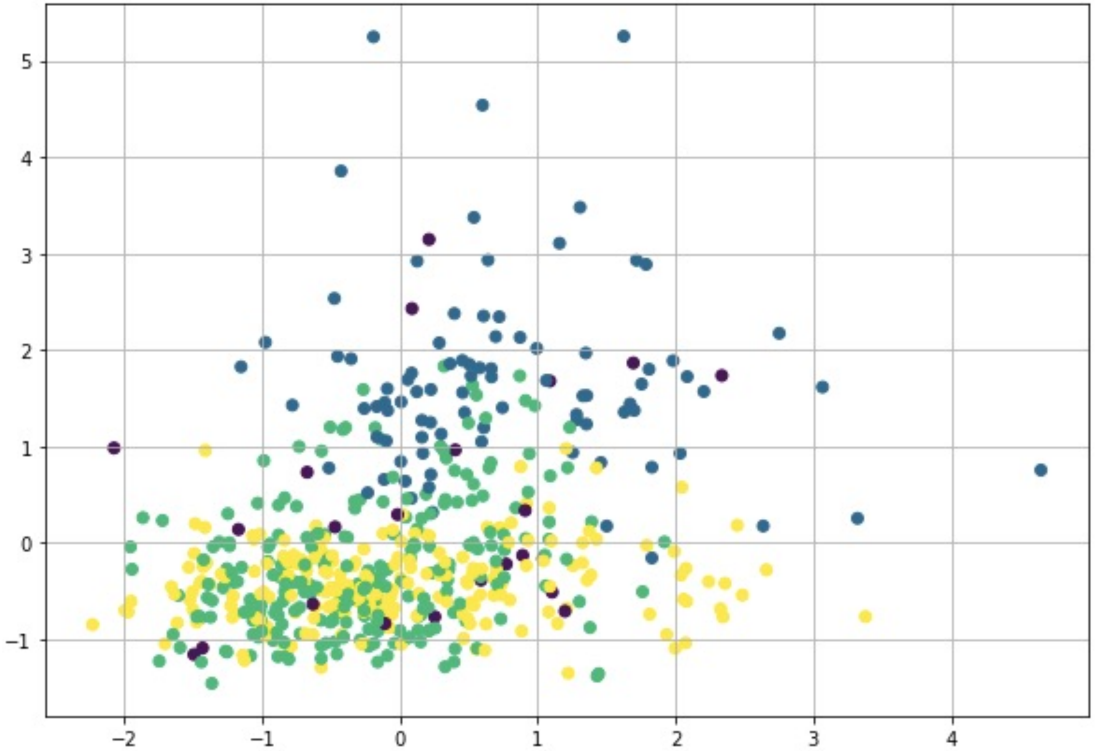
```
#Cantidad de elementos en los clusters
BCancer.groupby(['clusterH'])['clusterH'].count()
```

```
clusterH
0      23
1      88
2     248
3     210
```

…

4. Algoritmo: Ascendente Jerárquico

```
plt.figure(figsize=(10, 7))
plt.scatter(MEstandarizada[:,0], MEstandarizada[:,1], c=MJerarquico.labels_)
plt.grid()
plt.show()
```



Clustering Particional

Algoritmo: K-means

Práctica

5) Algoritmo: K-means

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

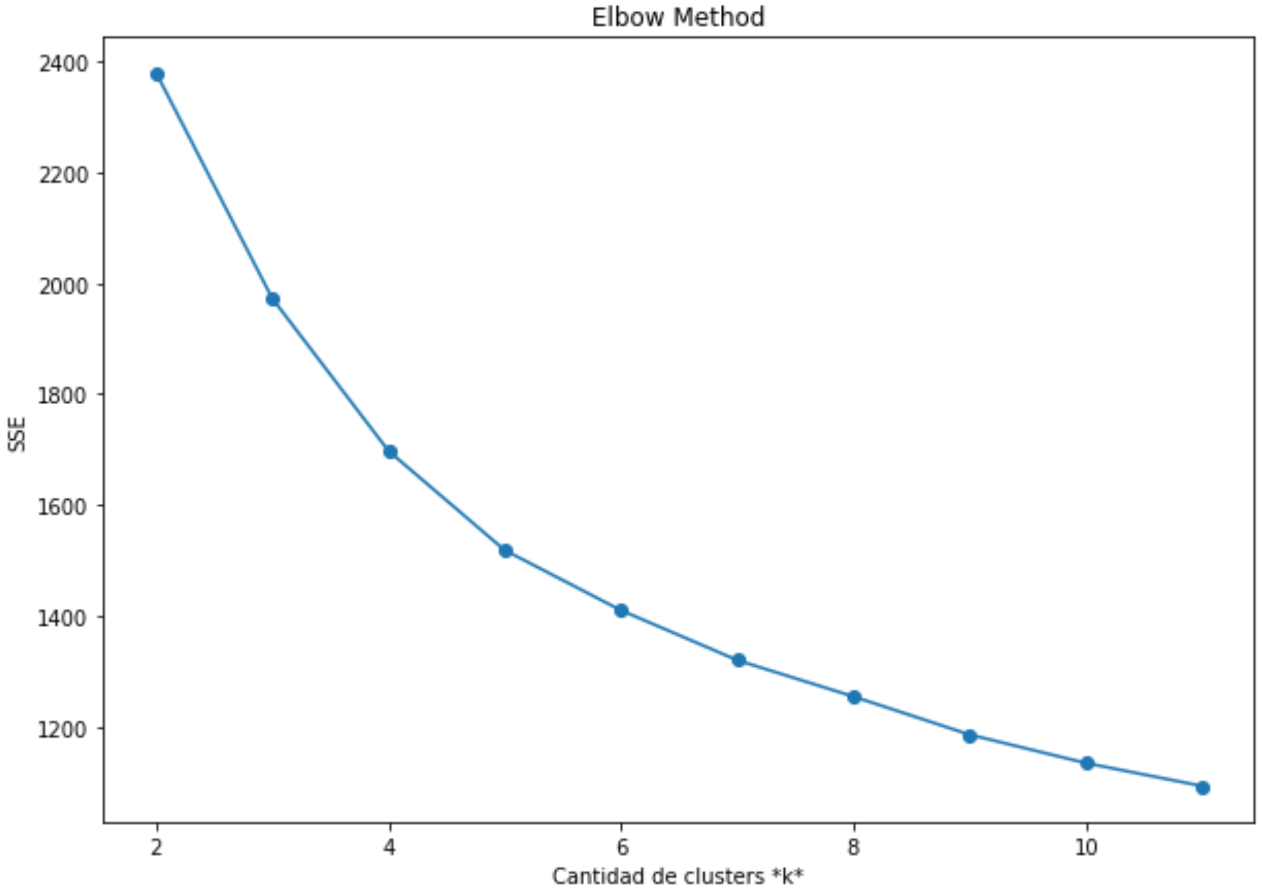
```
#Definición de k clusters para K-means
#Se utiliza random_state para inicializar el generador interno de números aleatorios
SSE = []
for i in range(2, 12):
    km = KMeans(n_clusters=i, random_state=0)
    km.fit(MEstandarizada)
    SSE.append(km.inertia_)

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

Práctica

5) Algoritmo: K-means

Método del codo



Observación:

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

5) Algoritmo: K-means

Método del codo

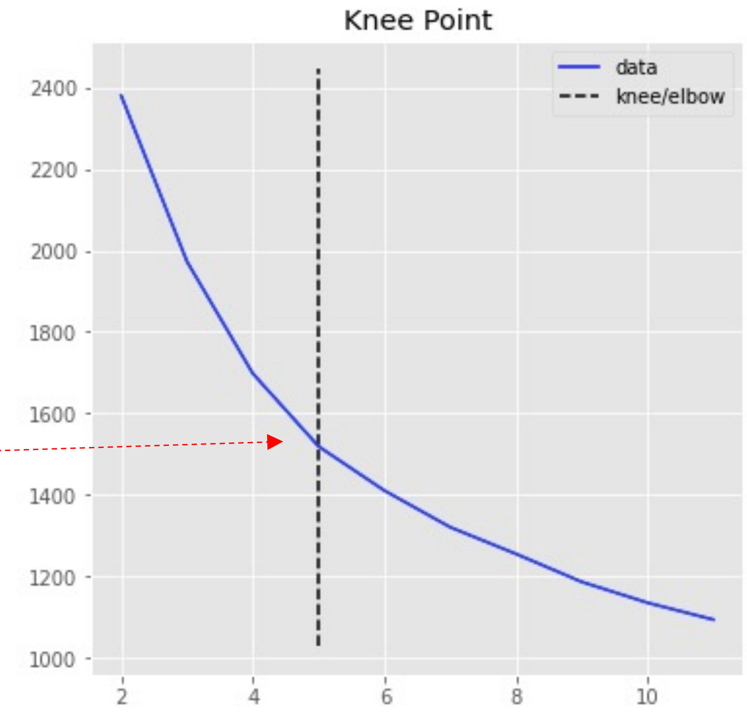
```
!pip install kneed
```

```
Collecting kneed
  Downloading kneed-0.7.0-py2.py3-none-any.whl (9.4 kB)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from kneed)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from kı
Requirement already satisfied: numpy>=1.14.2 in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/py
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from cycler>=
Installing collected packages: kneed
Successfully installed kneed-0.7.0
```

```python
from kneed import KneeLocator
kl = KneeLocator(range(2, 12), SSE, curve="convex", direction="decreasing")
kl.elbow
```

```
5
```

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



5) Algoritmo: K-means

Se crean las etiquetas en los clústeres

```
#Se crean las etiquetas de los elementos en los clusters
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)
```

5) Algoritmo: K-means

Se crean las etiquetas en los clústeres

```
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							0.1590	0.05648	3	4
567	P-927241	M	20.60	29.33							0.2397	0.07016	0	2
568	P-92751	B	7.76	24.54							0.1587	0.05884	3	4

569 rows × 14 columns

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

```
clusterP
0     172
1     100
2      56
3     156
4      85
```

Práctica

5) Algoritmo: K-means

Se crean las etiquetas en los clústeres

```
BCancer[BCancer.clusterP == 0]
```

	IDNumber	Diagnosis	Radius	Texture	Perimeter	Area	Smoothness	Compactness	Concavity	ConcavePoints	Symmetry	FractalDimension	clusterH	clusterP
16	P-848406	M	14.680	20.13	94.74	684.5	0.09867	0.07200	0.07395	0.052590	0.1586	0.05922	3	0
19	P-8510426	B	13.540	14.36	87.46	566.3	0.09779	0.08129	0.06664	0.047810	0.1885	0.05766	2	0
37	P-854941	B	13.030	18.42	82.61	523.8	0.08983	0.03766	0.02562	0.029230	0.1467	0.05863	3	0
46	P-85713702	B	8.196	16.84	51.71	201.9	0.08600	0.05943	0.01588	0.005917	0.1769	0.06503	3	0
51	P-857373	B	13.640	16.34	87.21	571.8	0.07685	0.06059	0.01857	0.017230	0.1353	0.05953	3	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
527	P-91813702	B	12.340	12.27	78.94	468.5	0.09003	0.06307	0.02958	0.026470	0.1689	0.05808	3	0
532	P-91903902	B	13.680	16.33	87.76	575.5	0.09277	0.07255	0.01752	0.018800	0.1631	0.06155	2	0
546	P-922577	B	10.320	16.35	65.31	324.9	0.09434	0.04994	0.01012	0.005495	0.1885	0.06201	3	0
547	P-922840	B	10.260	16.58	65.85	320.8	0.08877	0.08066	0.04358	0.024380	0.1669	0.06714	3	0
548	P-923169	B	9.683	19.34	61.05	285.7	0.08491	0.05030	0.02337	0.009615	0.1580	0.06235	3	0

172 rows × 14 columns

Práctica

5) Algoritmo: K-means

Obtención de los centroides

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

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Indexing with multiple keys (implicitly converted tc
  """Entry point for launching an IPython kernel.
```

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

Práctica

5) Algoritmo: K-means

Interpretación

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

Clúster 0: Conformado por 172 pacientes con alta probabilidad de tener un tumor benigno (por su tamaño), con un área promedio de tumor de 514 píxeles y una desviación estándar de textura de 16 píxeles. Aparentemente es un tumor compacto (0.06 píxeles), cuya suavidad alcanza 0.08 píxeles, una simetría de 0.16 y una aproximación de frontera, dimensión fractal, promedio de 0.059 píxeles.

…

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

```
clusterP
0    172
1    100
2     56
3    156
4     85
```

5) Algoritmo: K-means

Interpretación

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

Clúster 4: Es un grupo formado por 85 pacientes con un menor tamaño de tumor (potencialmente benigno), con un área promedio de tumor de 559 pixeles y una desviación estándar de textura de 24 pixeles. Es un tumor compacto (0.07 pixeles), cuya suavidad alcanza 0.08 pixeles, una simetría de 0.16 y una aproximación de frontera, dimensión fractal, promedio de 0.059 pixeles.

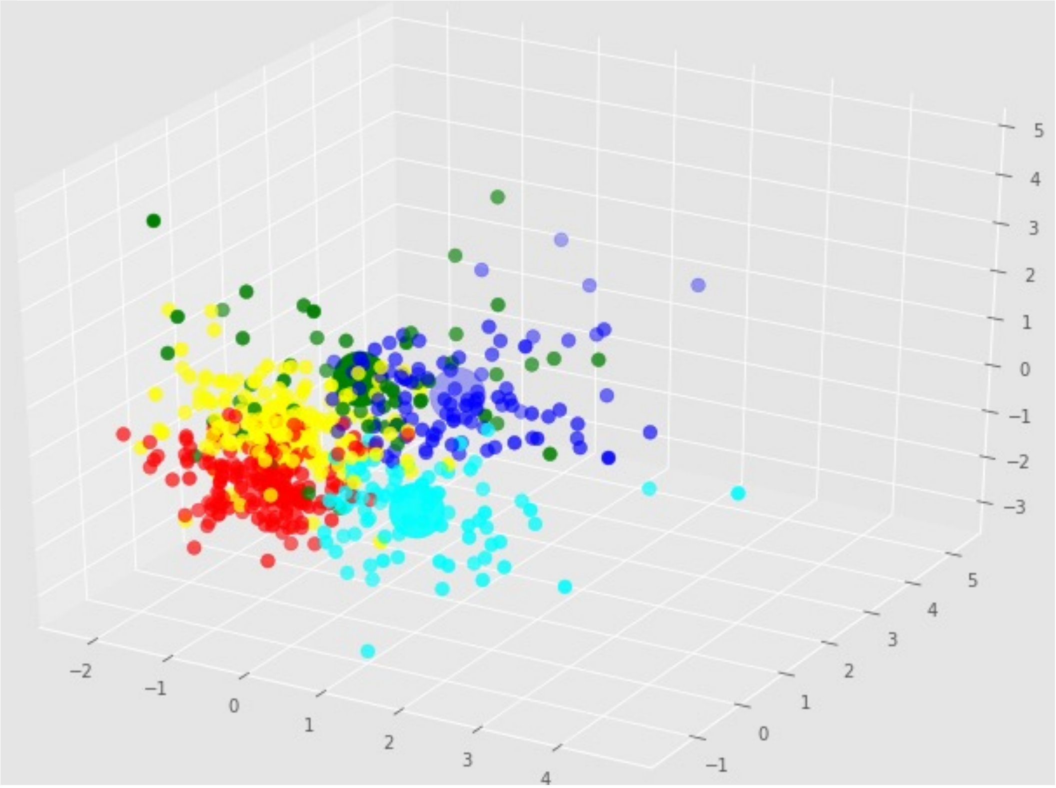
```
BCancer.groupby(['clusterP'])['clusterP'].count()
```

```
clusterP
0    172
1    100
2     56
3    156
4     85
```

5) Algoritmo: K-means

```
# 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()
```



Consideraciones finales

- Aumentar la cantidad de **clusters** mejorará naturalmente el ajuste (se hará una mejor explicación de la variación). Sin embargo, se puede caer en un sobreajuste, ya que se está dividiendo en múltiples grupos.

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