

# EXPLORACIÓN DE DATOS MEDIANTE PROYECCIONES

**Materia:** Análisis Exploratorio de Datos

**Programa:** Maestría en Ciencia de Datos e Información, INFOTEC

**Docente:** Dr. José Ortiz Béjar

**Alumno:** Rodrigo Guarneros Gutiérrez

**\*La principal idea de la reducción dimensional es mapear los datos de un espacio dimensional alto a un espacio dimensional más bajo, para los propósitos de la visualización de los mismos.\***

*B.K. Tripathy, Anvershrithaa Sundareswaran and Shruti Ghela (2022, p. xi).*

## 1. Introducción

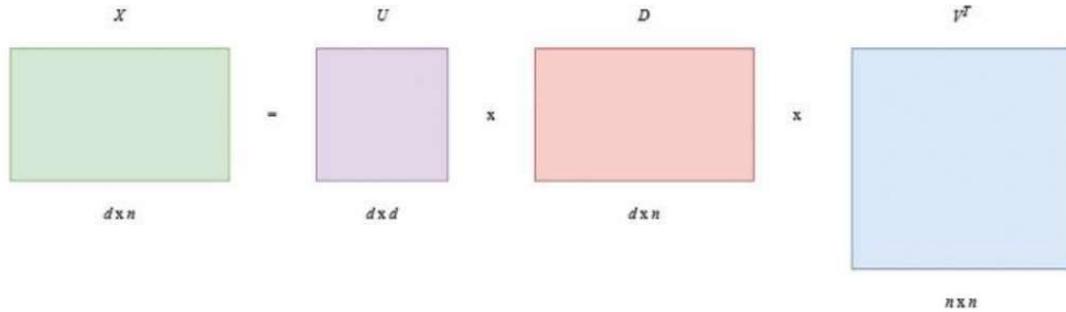
De acuerdo con B.K. Tripathy(2022), el clásico Principal Component Analysis (PCA) esta limitado para datos lineales. Mientras que los algoritmos tales como Kernel PCA, Locally Linear Embedding (LLE), Laplacian Eigenmaps (LE), Semidefinite Embedding (SE) y **t-SNE son propicios para datos no lineales.**

En cualquiera de los casos, el reto de estos algoritmos es encontrar una aproximación a los datos originales que tengan muchas menos dimensiones y, al mismo tiempo, retenga la estructura y las relaciones dentro de los datos B.K. Tripathy (2022, p. 2)

La técnica conocida como Descomposición Simple del Valor (SVD, Single Value Decomposition) se utiliza como una forma de resolver el problema de eigendecomposición de una manera eficiente. Se trata de una técnica de factorización matricial que expresa la matriz de los datos original ( $X$ ) como una combinación lineal de matrices de rango 1, haciéndola más estable que las rutinas de egenvectores [Tomado de B. K. Tripathy (2022, p. 7)]

En la siguiente imagen tomada de Tripathy, et. al. (2022, p.8), se puede ver un caso donde el número de datos es más grande que la dimensionalidad de cualquier dato puntual ( $n > d$ ). La técnica SVD produce tres matrices  $U$ ,  $D$  y  $V^T$ .  $U$  y  $V^T$  son ortogonales (su producto punto

en cada columna es cero), y cuyas columnas representan eigenvectores ortonormales de  $\mathbf{X}\mathbf{X}^T$  y  $\mathbf{X}^T\mathbf{X}$ . La matriz D es diagonal a la matriz (Valores singulares en la diagonal). Cada valor singular es a raíz cuadrada de los correspondiente eigenvalores de U o  $\mathbf{V}^T$  (ambas matrices tienen el mismo eigenvalor positivo). La matriz U representa el eigenvector de  $\mathbf{X}\mathbf{X}^T$ , el cual es la covarianza de la matriz. De aquí, se puede fácilmente encontrar el top eigenvectores  $p$  de la matriz de covarianza (la solución para PCA), usando SVD sobre la matriz X. Es importante considera que después de implementar el PCA en un conjunto de datos, los componentes principales son independientes uno del otro y no hay correlación entre ellos.



**1. Investiga sobre t-SNE y/o SVD (ver [5,6] en el notebook) su funcionamiento y sus implementaciones existentes, esto para incluirlas como una alternativa mas para proyección de datos de alta dimensión.**

## t-distribuida Incrustación de Vecino Estocástico (t-SNE)

La siguiente respuesta se elaboró con base principalmente del Capítulo 13 de Tripathy, et. al (2022).

### ¿En qué consiste?

Se trata de una técnica de reducción dimensional no lineal. Entre sus propiedades se encuentra:

- Preservan muy bien la **geometría total de los datos**.
- Retienen la estructura local o global de los datos en un simple mapeo dentro de un espacio dimensional menor.

Se trata de una variante de la técnica de Incrustación de Vecino Estocástico (SNE) que consiste en:

1. Convierte las distancias entre los puntos de datos a probabilidades condicionales. Cuál es la probabilidad de que el punto  $x_i$  elija el punto  $x_j$  como su vecino, si el criterio de selección fuera la densidad de probabilidad Gausiana centrada en  $x_i$ .
2. Como es de esperarse, en el espacio dimensional más pequeño  $p$ . Es posible computar la probabilidad similar pero de los puntos transformados  $y_i$  e  $y_j$ , con una probabilidad similar a la de sus puntos originales.
3. En virtud de que las similaridades entre los puntos de datos de la dimensión original son retenidos en el espacio menor, las probabilidades condicionales son iguales.

Basados en lo anterior, el criterio de selección del mapeo de los datos en una dimensión menor  $p$  es que se minimize la diferencia entre  $p_{i|j}$  y  $q_{i|j}$  (las funciones de densidad de probabilidad de ambos conjuntos). De tal forma que la función de costos a minimizar con este criterio está dado por  $C$  (Divergencia de Kullback-Leibler), como la medida de divergencia de las dos probabilidades condicionales:

$$C = \sum_i KL(P_i || Q_i) = \sum_{ij} p_{j|i} \log\left(\frac{p_{j|i}}{q_{j|i}}\right)$$

La minimización de esta función presenta el problema de saturación o hacinamiento, debido al hecho de que las dimensiones mayores tienen más espacio para acomodar los datos que las dimensiones menores.

La técnica t-SNE quiere resolver este problema de saturación o hacinamiento, modificando la función de costos simétrica y utilizando una distribución t en lugar de una Gausiana:

- La t-SNE utiliza probabilidades conjuntas donde

$$p_{ij} = p_{ji}$$

(en la dimensión original) y

$$q_{ij} = q_{ji}$$

(en la dimensión reducida).

- Esta técnica utiliza el método del gradiente descendente para minimizar la función de costos simétrica

## Ventajas:

- Supera a otras técnicas en el manejo eficiente de datos no lineales.
- Capta muy bien las relaciones polinómicas complejas, convirtiéndola en una técnica de reducción de dimensionalidad no lineal altamente eficiente.

- Conserva muy bien la localidad de los datos y al mismo tiempo revela algunos de sus aspectos más importantes.

## Desventajas:

- Si bien su rendimiento es eficiente en conjuntos de datos pequeños, su tiempo cuadrático y la complejidad del espacio lo hace computacionalmente complejo cuando se aplica a conjuntos de datos grandes. Su lentitud computacional es evidente en unos pocos de miles de datos.
- Además, el desempeño de t-SNE en dimensionalidad general. La reducción donde la dimensionalidad de los datos se reduce a una dimensión mayor que tres ( $p>3$ ) no está muy clara. El rendimiento de t-SNE cuando la reducción a un espacio bidimensional o tridimensional no se puede generalizar a dimensiones superiores a tres debido a las pesadas colas del estudiante distribución t que, en espacios de alta dimensión, conduce a mapeos que no preservan la estructura local de los datos ya que las colas pesadas abarcan una gran proporción de la masa de probabilidad bajo la distribución t en altos espacios dimensionales.

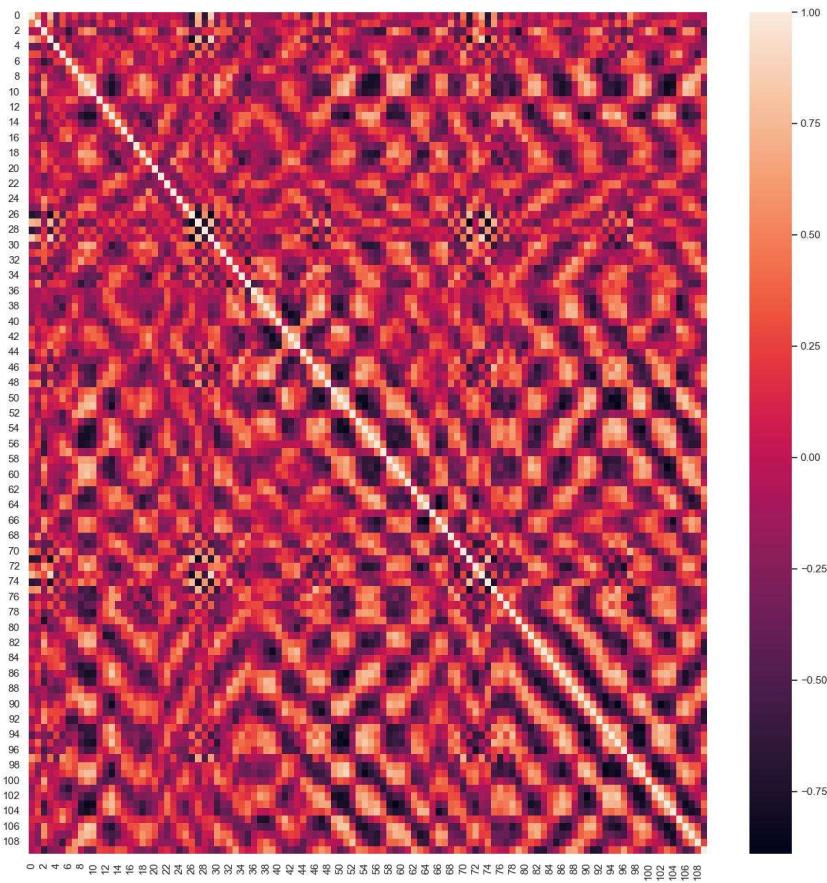
Esta técnica es utilizada en el procesamiento de imágenes, bioinformática, procesamiento de señales, procesamiento de lenguaje, NLP, y otros.

## 2. ¿Es posible encontrar una estructura en los datos2.csv?

Determina si es posible encontrar una estructura en los datos contenidos en **datos2.csv**.

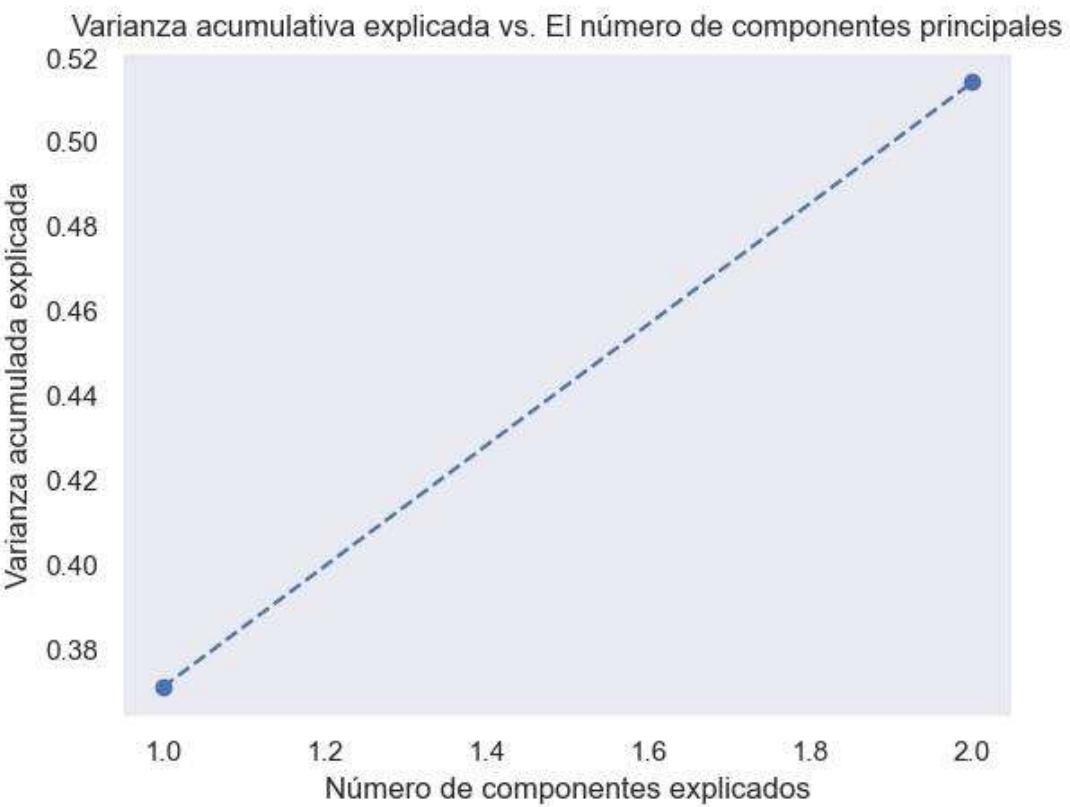
- Puedes obtener toda la matriz de datos como:  
`data=pd.read_csv('datos2.csv').values[1:,1:]` omitimos el encabezado y la primera columna que contiene el identificador de cada objeto en la colección.
- Describe los datos. Se trata de 110 variables y 94 observaciones. Los datos no contienen datos nulos o missing values. Los datos tienen un recorrido para cada columna que va del valor mínimo 0 al valor máximo 0.97 o 1. En general parecen ser muy similares. Aunque se puede advertir una seria diferencia en la variación entre cada variable, tal y como se ilustra en la matriz de covarianzas (varianzas respecto a los otros datos y ellos mismos) y la de correlación (varianza e intensidad de la relación) con una estructura y

patrones que debería reflejar una reducción de dimensiones.

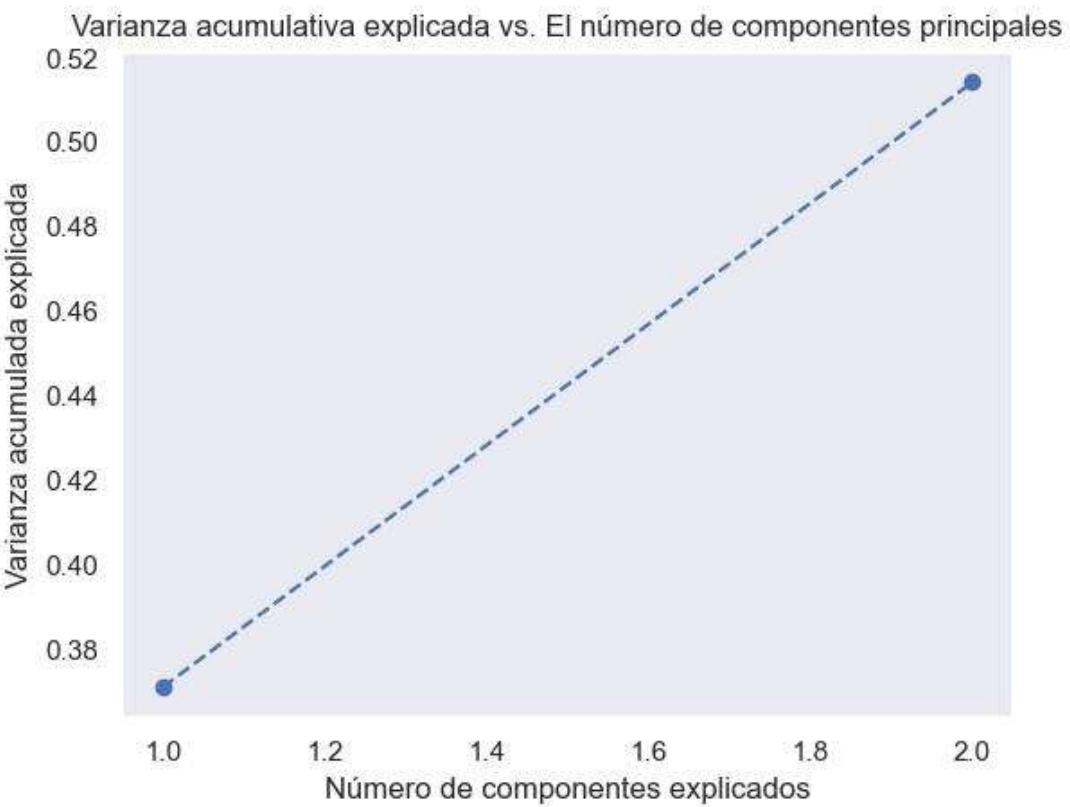


- Analiza los datos usando las diferentes estrategias/herramientas vistas hasta ahora. Deberás aplicar al menos dos alternativas de proyección para obtener una representación 2D de los datos.
  - Se utiliza el clásico Análisis de Componente Principal (PCA, por sus siglas en inglés)

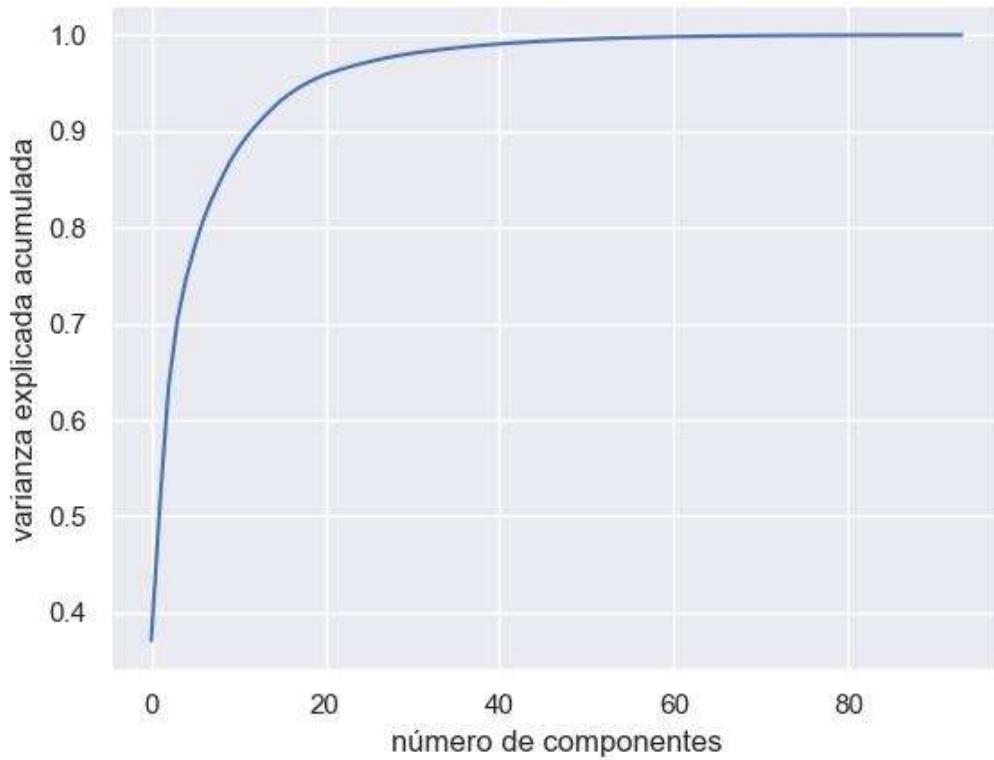
**Para este caso se reduce a dos componentes principales, los cuales tienen la capacidad de explicar una varianza acumulada de todos los datos de 51.43%, tal y como se ilustra a continuación:**



**En tanto que con tres componentes principales el nuevo espacio vectorial explica 63.60% de la varianza en la dimensión original,** tal y como se ilustra a continuación.

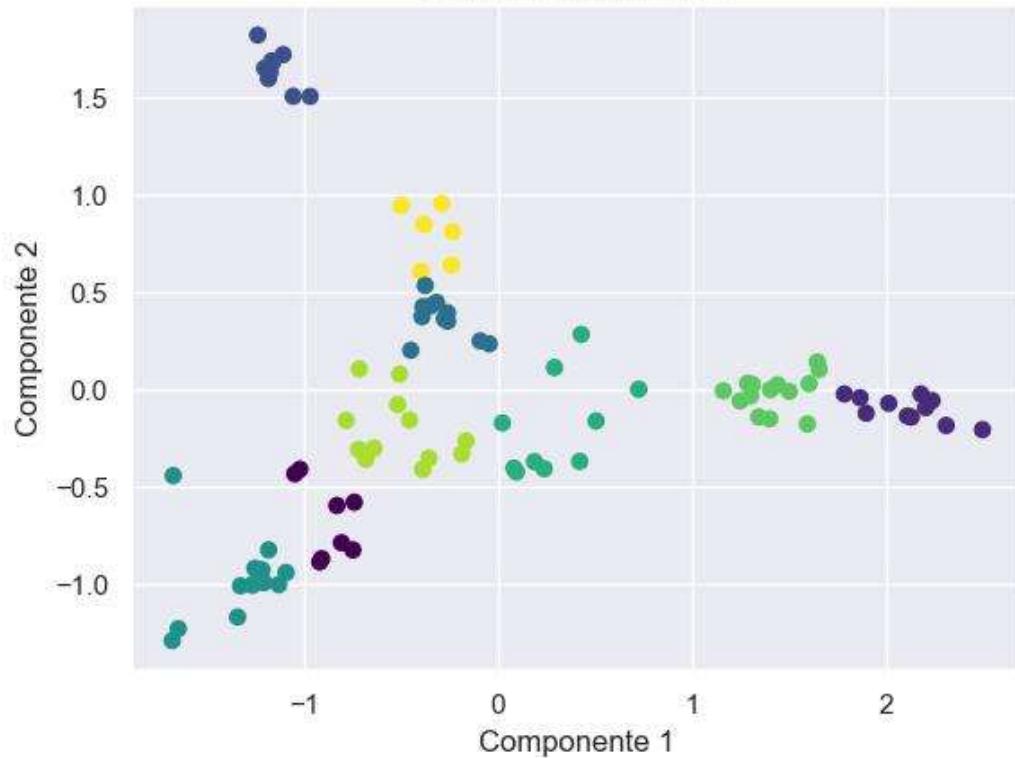


**En general se estaría considerando incluir hasta 10 componentes para capturar el mayor porcentaje de varianza del espacio vectorial original sin hacer muy complicado el modelo.**



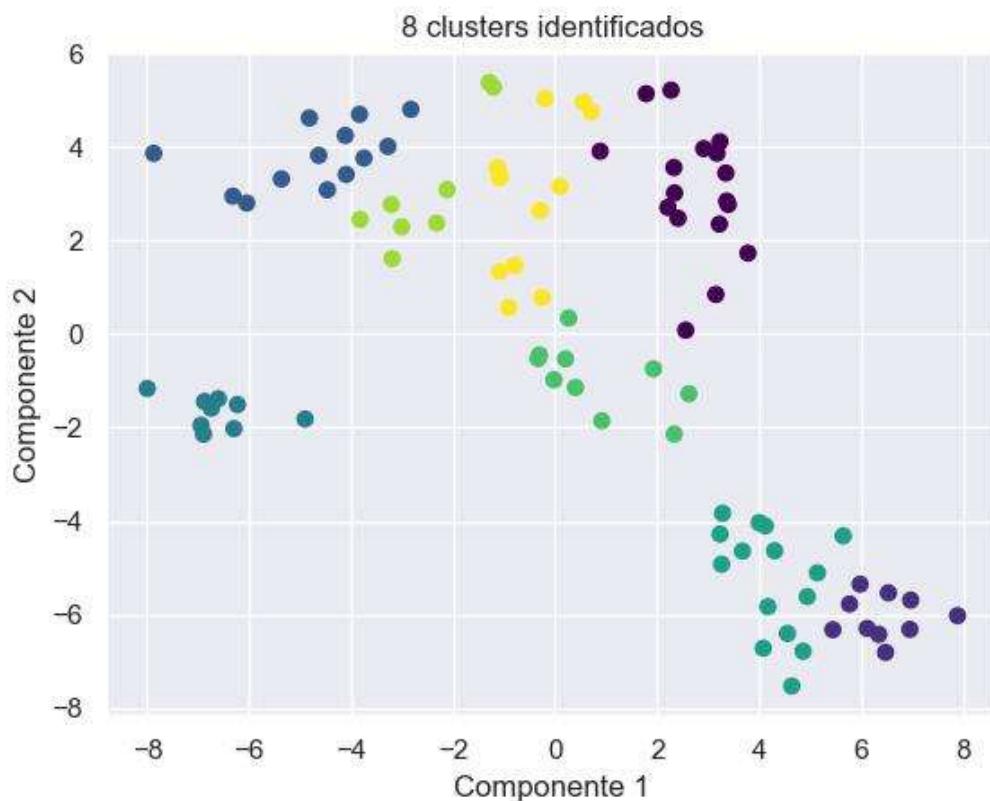
A reserva de hacer un análisis de agrupación más profundo, la cantidad de clases identificadas oscila entre 7 y 9 con esta metodología, en el espacio vectorial optimizado a 2 es el siguiente:

9 clusters identificados



- t-SNE (t distribuida Incrustación de Vecino Estocástico)

Para este caso en particular tenemos que es posible identificar hasta 14 clases:



- Compara las gráficas de las proyecciones en términos de su capacidad descriptiva y explicativa.
- Para una proyección adecuada deberían poder identificarse entre 7 y 14 clases

In [3]:

```
# Dependencias
%matplotlib inline
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from scipy import stats as st
from sklearn.datasets import make_blobs,make_circles
from sklearn import datasets
from sklearn.metrics import pairwise_distances
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA
import seaborn as sns; sns.set()
```

## Obteniendo la información

In [4]:

```
data=pd.read_csv('datos2.csv').values[1:,1:]
data.shape # 94 puntos y 110 variables, se eliminó encabezado y primera columna
```

Out[4]: (94, 110)

## Descripción de la información

```
In [14]: data = pd.DataFrame(data)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 94 entries, 0 to 93
Columns: 110 entries, 0 to 109
dtypes: object(110)
memory usage: 80.9+ KB
```

```
In [15]: data.describe()
```

```
Out[15]:
```

	0	1	2	3	4	5	6	7
<b>count</b>	94.000000	94.000000	94.000000	94.000000	94.000000	94.000000	94.000000	94.000000
<b>unique</b>	94.000000	94.000000	94.000000	94.000000	94.000000	94.000000	94.000000	94.000000
<b>top</b>	0.910831	0.760617	0.595962	0.706491	0.039998	0.391113	0.479075	0.606625
<b>freq</b>	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

4 rows × 110 columns

```
In [26]: for column in data.columns:
    data[column] = pd.to_numeric(data[column], errors='coerce')
```

```
In [37]: description = data.describe()
description
```

```
Out[37]:
```

	0	1	2	3	4	5	6	7
<b>count</b>	94.000000	94.000000	94.000000	94.000000	94.000000	94.000000	94.000000	94.000000
<b>mean</b>	0.864315	0.719927	0.354785	0.703554	0.313261	0.724301	0.584276	0.414982
<b>std</b>	0.104046	0.110338	0.216250	0.085902	0.151032	0.174446	0.188686	0.187910
<b>min</b>	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
<b>25%</b>	0.844248	0.680441	0.190346	0.684396	0.224832	0.620679	0.470157	0.269199
<b>50%</b>	0.863521	0.722430	0.285067	0.707420	0.310044	0.768337	0.543137	0.407584
<b>75%</b>	0.904964	0.769055	0.510516	0.731483	0.414598	0.840757	0.718255	0.531492
<b>max</b>	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 110 columns

```
In [33]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 94 entries, 0 to 93
Columns: 110 entries, 0 to 109
dtypes: float64(110)
memory usage: 80.9 KB
```

```
In [35]: nulls = data.isnull().sum()
print(nulls)
nulls.unique()
```

```
0      0
1      0
2      0
3      0
4      0
 ..
105    0
106    0
107    0
108    0
109    0
Length: 110, dtype: int64
```

```
Out[35]: array([0], dtype=int64)
```

```
In [46]: sns.heatmap(data.isnull(), cbar = False, cmap='Blues') #no hay valores nulos
```

```
Out[46]: <Axes: >
```



```
In [47]: data.duplicated().sum()
```

```
Out[47]: 0
```

```
In [40]: description.loc['max'].unique()
```

```
Out[40]: array([1.0, 0.97138498])
```

```
In [52]: cov_matrix = data.cov()  
cov_matrix
```

```
Out[52]:
```

	0	1	2	3	4	5	6	7	
0	0.010826	0.005511	-0.001756	0.006652	-0.005855	0.005318	-0.000733	0.003990	-0.002811
1	0.005511	0.012175	-0.000433	0.004909	-0.006934	0.005344	-0.004595	-0.000688	-0.007111
2	-0.001756	-0.000433	0.046764	0.003804	-0.002895	-0.007225	-0.026672	-0.008955	0.015801
3	0.006652	0.004909	0.003804	0.007379	-0.005627	0.002490	-0.006534	0.000194	-0.002501
4	-0.005855	-0.006934	-0.002895	-0.005627	0.022811	0.000350	0.006141	-0.007750	-0.004701
...	...	...	...	...	...	...	...	...	...
105	0.002178	0.002490	0.003422	0.000499	0.004912	0.018959	0.006154	-0.007211	0.000401
106	0.002112	-0.006288	0.030066	0.002354	0.002694	0.004145	-0.001473	-0.003132	0.023301
107	-0.000597	-0.004052	0.017172	0.001372	-0.003545	-0.006031	-0.007869	0.001496	0.018001
108	-0.000665	-0.006008	0.005047	-0.000744	-0.005876	-0.018277	0.003422	0.018908	0.022101
109	-0.002361	0.000717	-0.024635	-0.004782	-0.007170	-0.004232	0.018017	0.020335	-0.000001

110 rows × 110 columns

```
In [53]: corr_matrix = data.corr()  
corr_matrix
```

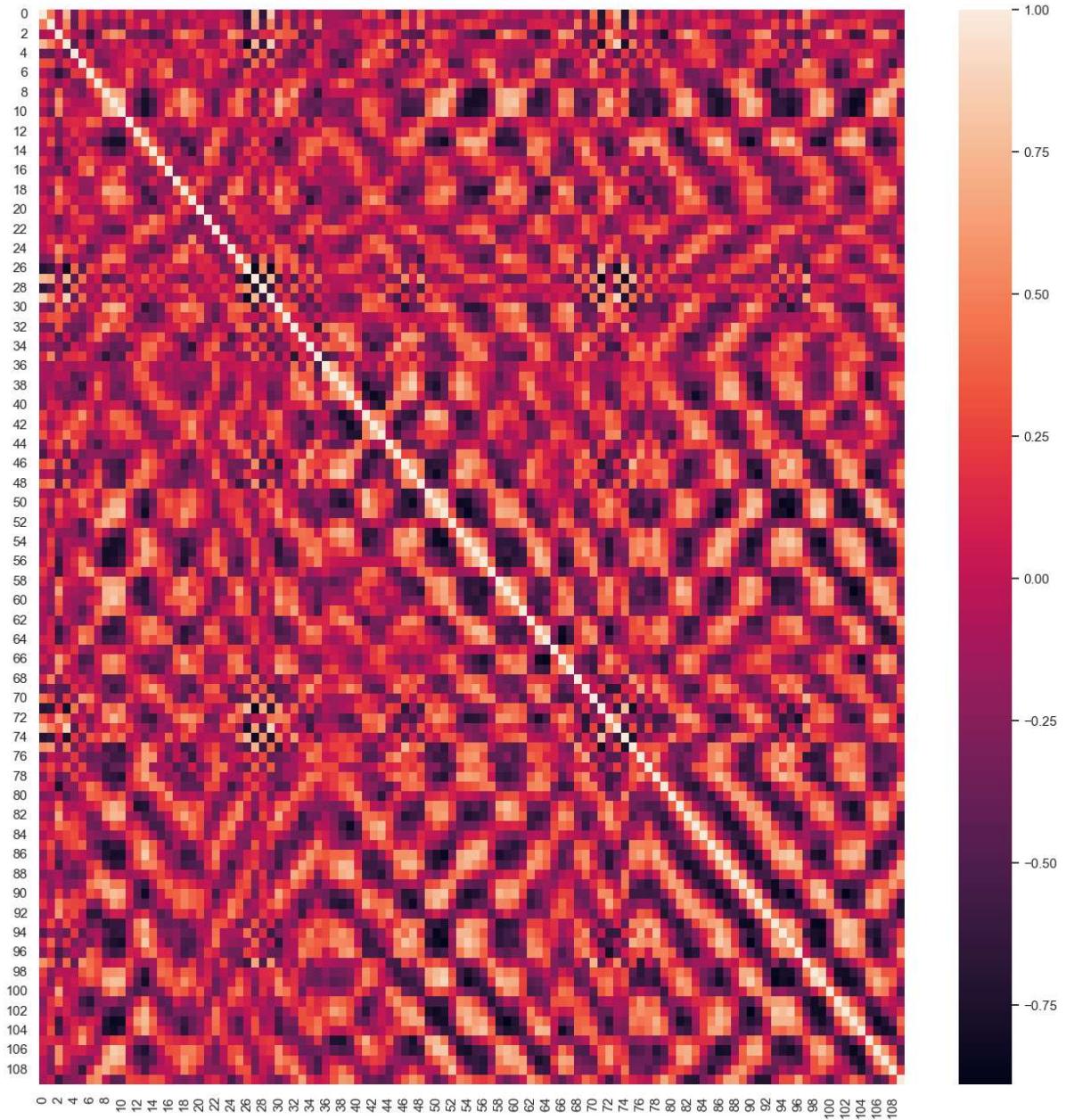
Out[53]:

	<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	
<b>0</b>	1.000000	0.480031	-0.078048	0.744300	-0.372595	0.292974	-0.037359	0.204071	-0.1474
<b>1</b>	0.480031	1.000000	-0.018135	0.517885	-0.416117	0.277620	-0.220727	-0.033206	-0.3494
<b>2</b>	-0.078048	-0.018135	1.000000	0.204797	-0.088653	-0.191512	-0.653678	-0.220363	0.3958
<b>3</b>	0.744300	0.517885	0.204797	1.000000	-0.433751	0.166142	-0.403127	0.012010	-0.1615
<b>4</b>	-0.372595	-0.416117	-0.088653	-0.433751	1.000000	0.013302	0.215481	-0.273072	-0.1718
...	...	...	...	...	...	...	...	...	...
<b>105</b>	0.124780	0.134544	0.094317	0.034655	0.193893	0.647853	0.194415	-0.228773	0.0146
<b>106</b>	0.084245	-0.236521	0.576998	0.113741	0.074038	0.098608	-0.032395	-0.069179	0.5253
<b>107</b>	-0.041337	-0.264389	0.571729	0.114969	-0.168983	-0.248939	-0.300293	0.057327	0.7039
<b>108</b>	-0.037594	-0.320439	0.137357	-0.050963	-0.228958	-0.616597	0.106744	0.592192	0.7048
<b>109</b>	-0.120136	0.034381	-0.603013	-0.294639	-0.251297	-0.128428	0.505453	0.572823	-0.0002

110 rows × 110 columns

In [56]:

```
plt.figure(figsize = (16,16))
sns.heatmap(corr_matrix, annot = False)
plt.savefig('correlation_matrix.jpg')
plt.show()
```



## Proyecciones

Para efectos de esta sección se eligieron las técnicas PAC (para relaciones lineales) y t-SNE

```
In [97]: #Calculamos V_j y \Lambda_j
pca = PCA(n_components=2)
pca.fit(data)
```

```
Out[97]: ▾      PCA
PCA(n_components=2)
```

```
In [98]: print(pca.components_)
```

```

[[ -9.69312274e-03  2.11183777e-02 -9.61009014e-02 -4.61588531e-03
-1.18674436e-02 -1.89862416e-02 -1.05499282e-02 -5.78481475e-03
-1.06391804e-01 -1.86988737e-01 -1.18127567e-01 -1.18266747e-02
6.54018285e-02  1.57549749e-01  7.28431880e-02  1.70992402e-04
-3.31892297e-02 -4.92090916e-02 -7.69086442e-02 -8.75705800e-02
-4.04424793e-02  1.17359540e-02  7.14546469e-02  3.63442178e-03
-4.55865352e-02 -6.14139073e-02  1.33978211e-03  2.55341871e-02
-1.42934789e-02 -1.65337031e-02 -1.00155969e-01 -4.76024598e-02
-1.57625197e-02  8.30871600e-02  5.76188214e-02  5.31848152e-02
3.27213673e-02  8.66108380e-02  1.35716236e-01  8.60849874e-02
-6.03307242e-03 -1.38300170e-01 -5.91609629e-02 -2.29250384e-02
-1.60307611e-02  4.79696927e-02  9.94156215e-02  1.48509873e-01
6.21197301e-02 -1.35005202e-01 -2.11213851e-01 -1.80184190e-01
-3.74090639e-02  1.12263927e-01  1.69684033e-01  1.18384042e-01
1.06056812e-01  9.16070461e-04 -1.37942604e-01 -1.46906106e-01
-1.27272255e-01 -3.75596633e-02  8.24864246e-02  6.52960057e-02
4.07614258e-02 -4.90776599e-03 -5.29779148e-02 -4.70630020e-02
7.08248151e-02  6.88809326e-02  1.20333505e-02 -3.69183316e-02
-1.02872594e-01 -2.48005956e-02 -2.10479847e-02  6.07781003e-02
1.25433622e-01  7.18306058e-02  1.31586020e-01  7.68350962e-02
-1.62277893e-02 -1.28936705e-01 -1.25246240e-01 -5.75514850e-02
7.47555632e-03  7.83625482e-02  1.75952678e-01  1.54777522e-01
3.41301768e-02 -9.54464529e-02 -1.45681563e-01 -1.71798912e-01
-1.70132845e-02  1.33079562e-01  1.40442857e-01  1.47302447e-01
9.92842235e-02 -1.52037850e-02 -1.55730024e-01 -1.95946752e-01
-6.81499860e-02  7.82328663e-02  1.63587909e-01  2.08484714e-01
8.78049483e-02 -7.28133016e-02 -1.85258451e-01 -7.84685876e-02
-4.35109429e-02  7.50671327e-02]
[-1.32414310e-02 -3.82225273e-02 -7.70725396e-02 -3.82361550e-02
-7.99058737e-02 -1.00449032e-01  1.04823411e-01  1.82971047e-01
1.34364819e-01  4.60116047e-02  3.12432792e-02  8.65103324e-03
-4.58573221e-02 -8.84492075e-02  3.40270162e-02  7.02223080e-02
5.08864561e-02  2.33900722e-03  1.18337541e-02 -2.18821778e-02
-5.12592823e-02  7.19382678e-03 -1.22281494e-02 -1.01882169e-01
-1.10606089e-01  3.19905852e-02  5.26302019e-02 -2.76980751e-02
5.27300048e-03 -4.51741867e-02  4.23841031e-03  3.23067861e-02
1.15289324e-01  1.77749278e-01  3.02491092e-02 -8.72934904e-02
2.09149948e-03  1.05214431e-01  2.49655977e-01  1.78285806e-01
2.71302594e-02 -1.46088416e-01 -7.35391292e-02 -8.81722882e-02
-1.44169325e-01  1.07414895e-02  5.21926925e-02  1.38594875e-01
1.29753580e-02 -3.21661501e-02 -7.77627349e-02  5.68512545e-02
1.37873207e-01  9.43632831e-02  5.88013508e-03 -5.19790047e-02
-8.97433241e-02  9.24332461e-03  6.10251330e-02  6.93643432e-02
1.05281693e-01  9.07879953e-02  8.20796961e-02 -9.61229701e-03
-2.07605594e-02 -5.07157271e-02  1.45683510e-03  1.65096309e-02
-3.04694750e-02  3.02566827e-02  6.59455253e-03  2.86140841e-02
-4.89607026e-02 -3.120666906e-02  7.52883786e-02 -2.58515933e-02
-7.86875218e-02 -1.18100836e-01 -4.04995685e-02  1.44353139e-01
1.05212605e-01  7.21888760e-02 -6.41378116e-02 -1.64347711e-01
-1.75939589e-01 -1.80589484e-01 -4.87567155e-02  1.43859315e-01
2.17758238e-01  2.72116958e-01  4.77793543e-02 -1.75475829e-01
-2.37226771e-01 -1.46100609e-01 -8.59420846e-04 -1.42581711e-02
8.95528560e-02  2.71174614e-02  1.18601207e-01  9.63139805e-02
-2.44852370e-02 -6.23546347e-02 -3.85144070e-02 -5.50819287e-02
-7.03712370e-02 -7.85018522e-02 -2.33243256e-02  3.71381865e-02
1.66213903e-01  1.71119057e-01]]

```

```
In [99]: # Para acceder a los valores principales utilizamos  
print(pca.explained_variance_)  
[1.36136272 0.5264207 ]
```

```
In [100... data_pca = pca.transform(data)  
  
print(f'Dimensión original: {data.shape}')  
print(f'Dimensión reducida a 2: {data_pca.shape}')  
  
Dimensión original: (94, 110)  
Dimensión reducida a 2: (94, 2)
```

```
In [101... data_pca.shape
```

```
Out[101]: (94, 2)
```

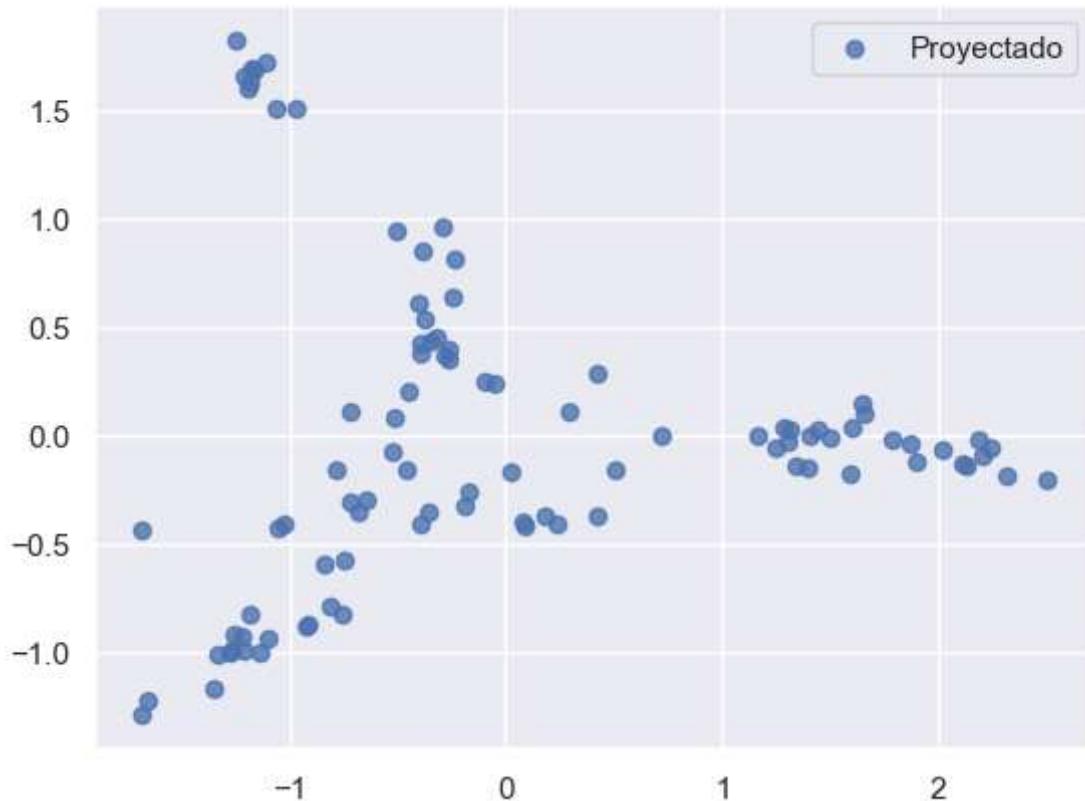
```
In [102... data.shape
```

```
Out[102]: (94, 110)
```

**Los datos transformados se han reducido a dos dimensiones. Para comprender el efecto de esta reducción de dimensionalidad, podemos graficar los datos reducidos:**

```
In [103... plt.scatter(data_pca[:, 0], data_pca[:, 1], alpha=0.8, color='b', label='Proyectado')  
plt.axis('equal');  
plt.legend()
```

```
Out[103]: <matplotlib.legend.Legend at 0x18ef9164310>
```



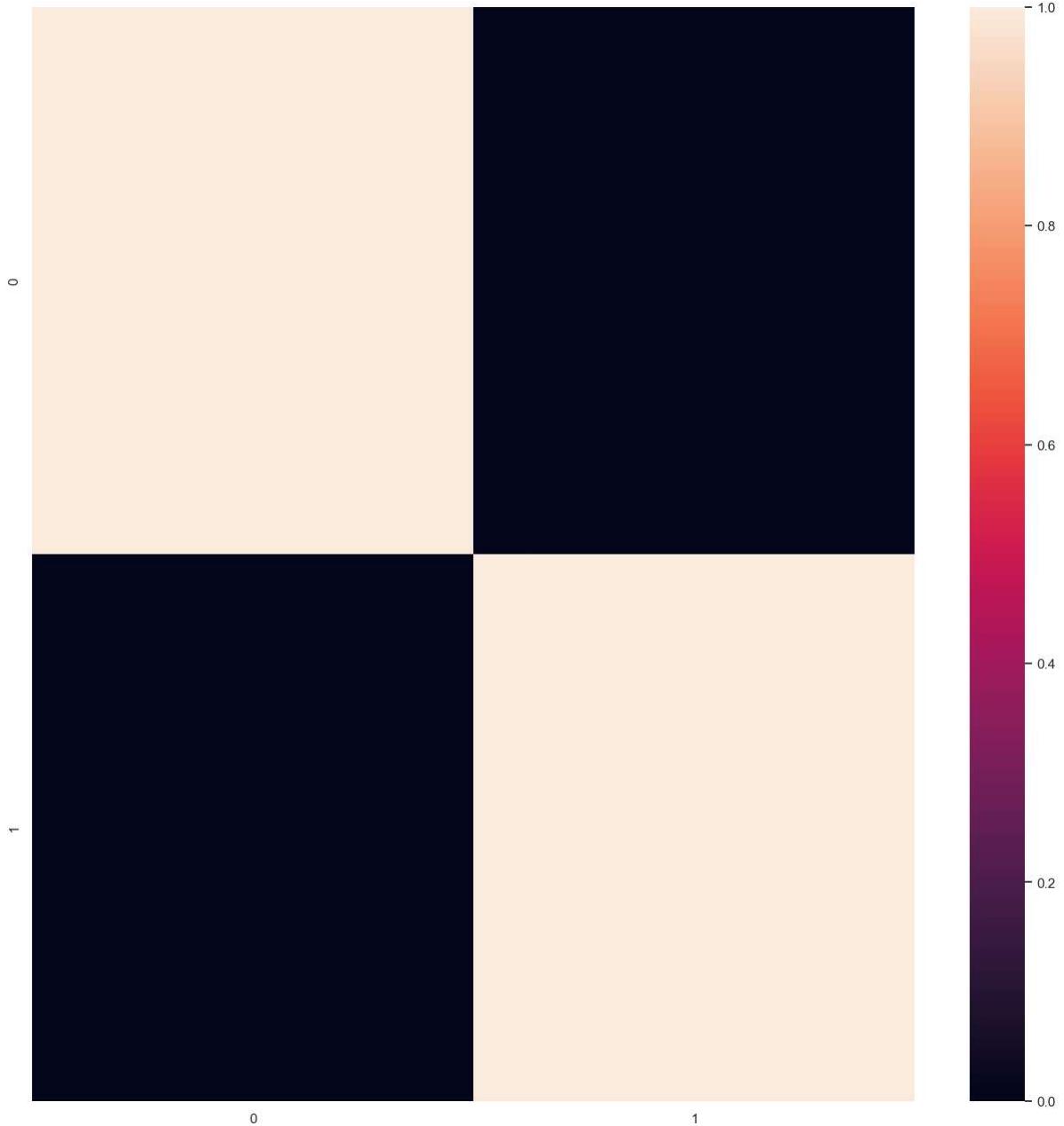
```
In [104...]: # Cómo se ve su estructura:  
data_pca_df = pd.DataFrame(data_pca)  
data_pca_df.describe()
```

```
Out[104]:
```

	0	1
<b>count</b>	9.400000e+01	9.400000e+01
<b>mean</b>	2.125959e-17	-4.842462e-17
<b>std</b>	1.166774e+00	7.255485e-01
<b>min</b>	-1.683181e+00	-1.287615e+00
<b>25%</b>	-9.593248e-01	-4.036815e-01
<b>50%</b>	-3.538034e-01	-7.109445e-02
<b>75%</b>	1.225326e+00	2.778897e-01
<b>max</b>	2.497406e+00	1.824986e+00

```
In [105...]: corr_dtapca = data_pca_df.corr()
```

```
In [106...]: plt.figure(figsize = (16,16))  
sns.heatmap(corr_dtapca, annot = False)  
plt.savefig('correlation_matriz_pca.jpg')  
plt.show()
```



```
In [107]: # qué porcentaje de la variación explican nuestros dos y tres componentes:
```

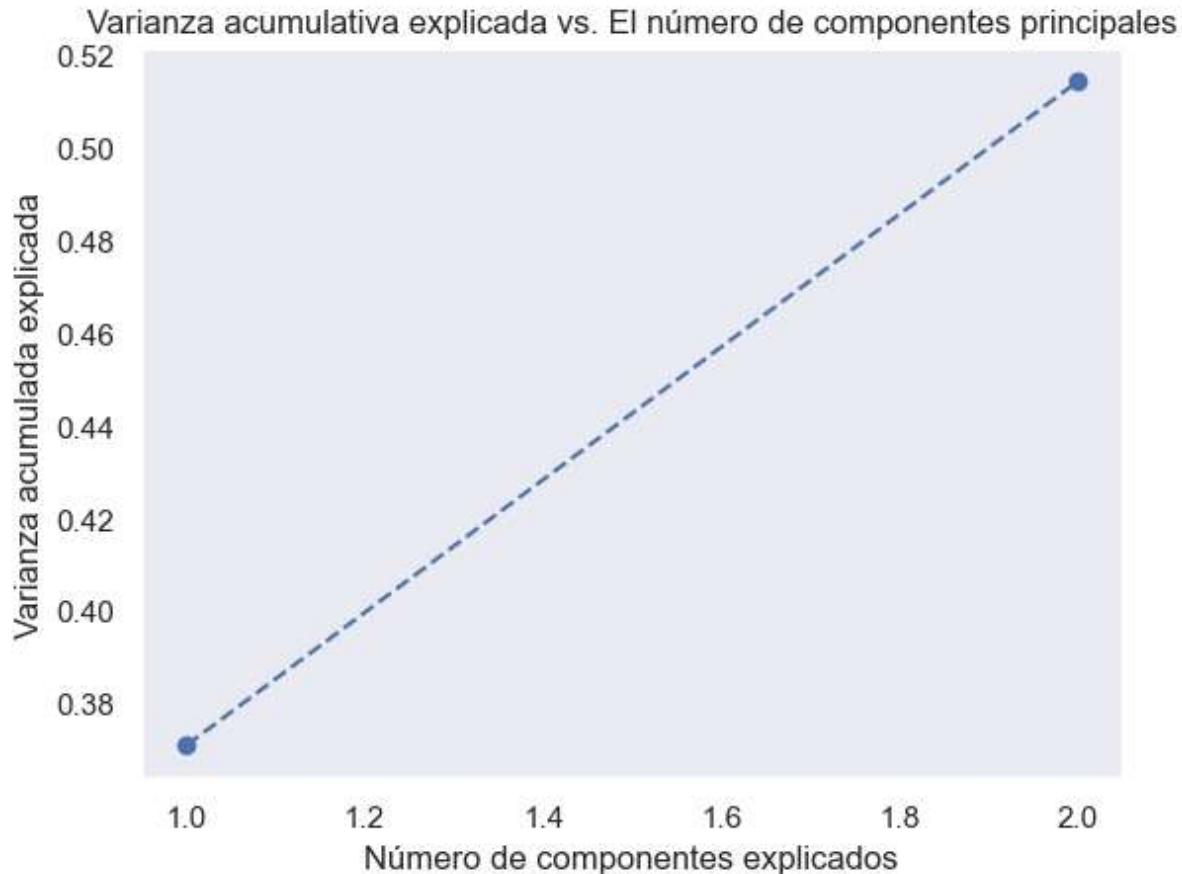
```
Tasas_variabilidad_explicados = pca.explained_variance_ratio_
Tasas_variabilidad_explicados
```

```
Out[107]: array([0.37087468, 0.14341226])
```

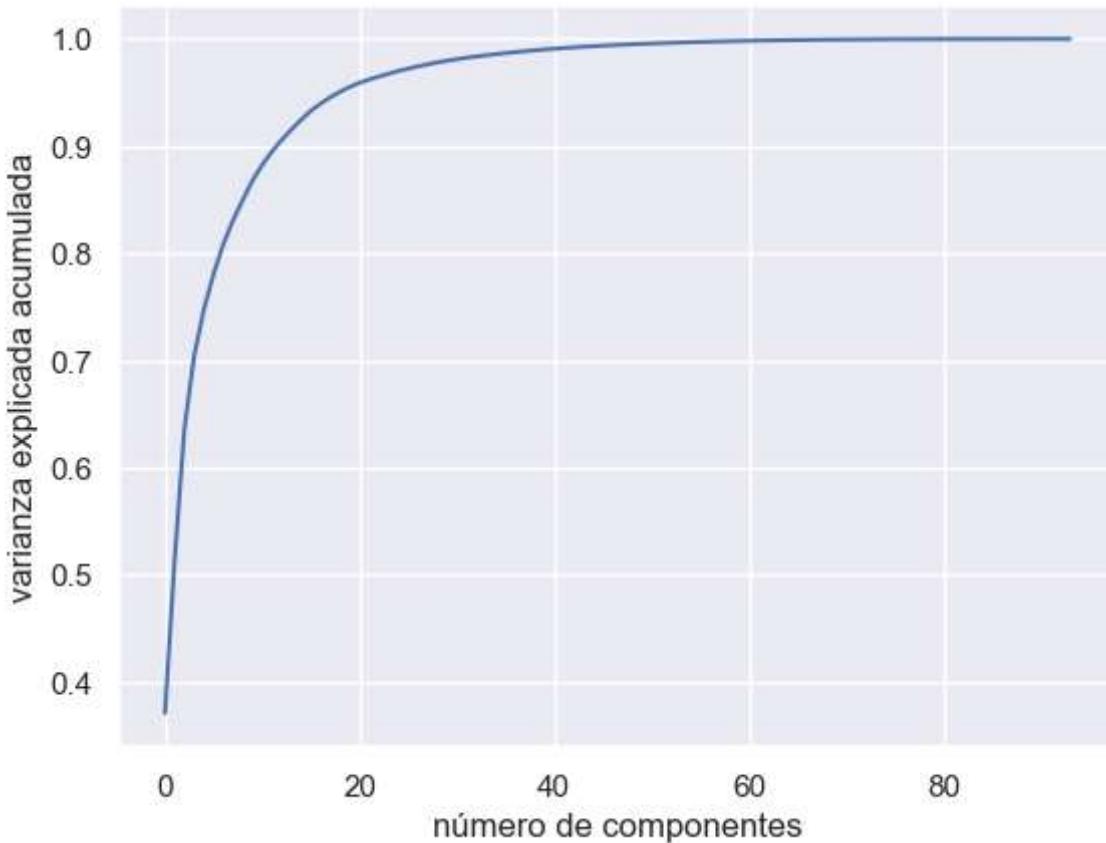
```
In [108]: cumulative_variance = Tasas_variabilidad_explicados.cumsum()
```

```
# Plot the cumulative explained variance
plt.plot(range(1, len(cumulative_variance) + 1), cumulative_variance, marker='o', l
plt.xlabel('Número de componentes explicados')
plt.ylabel('Varianza acumulada explicada')
plt.title('Varianza acumulativa explicada vs. El número de componentes principales')
plt.grid()
```

```
plt.savefig('var_explicada_3.jpg')
plt.show()
```



```
In [114]: pca = PCA().fit(data)
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('número de componentes')
plt.ylabel('varianza explicada acumulada')
plt.savefig('Numero_Componentes.jpg')
```



In [121]:

```
from sklearn.cluster import KMeans

num_clusters = 9

kmeans = KMeans(n_clusters=num_clusters)

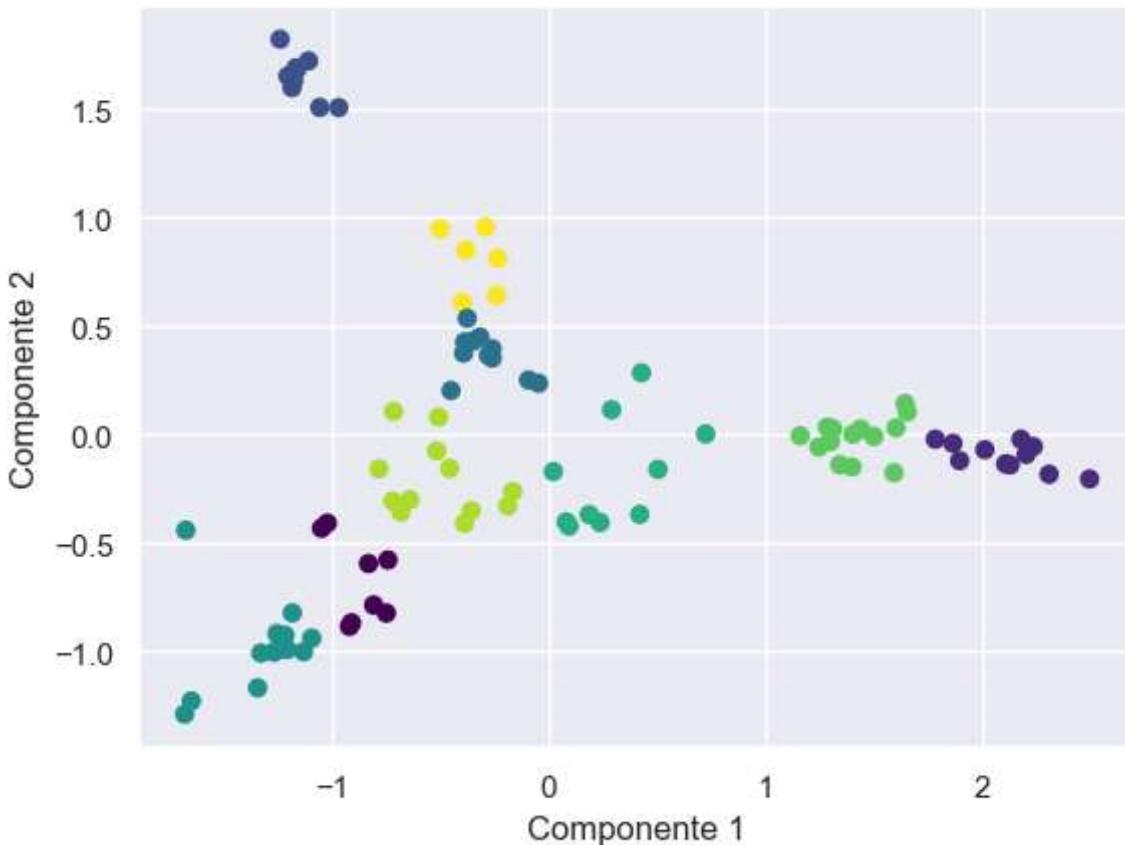
kmeans.fit(data_pca)

cluster_labels = kmeans.labels_

plt.scatter(data_pca[:, 0], data_pca[:, 1], c=cluster_labels, cmap='viridis')
plt.xlabel('Componente 1')
plt.ylabel('Componente 2')
plt.title(f'{num_clusters} clusters identificados')
plt.savefig('clases_pca.jpg')
plt.show()
```

```
C:\Users\rodri\AppData\Roaming\jupyterlab-desktop\jlab_server\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\rodri\AppData\Roaming\jupyterlab-desktop\jlab_server\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

9 clusters identificados



## t-SNE

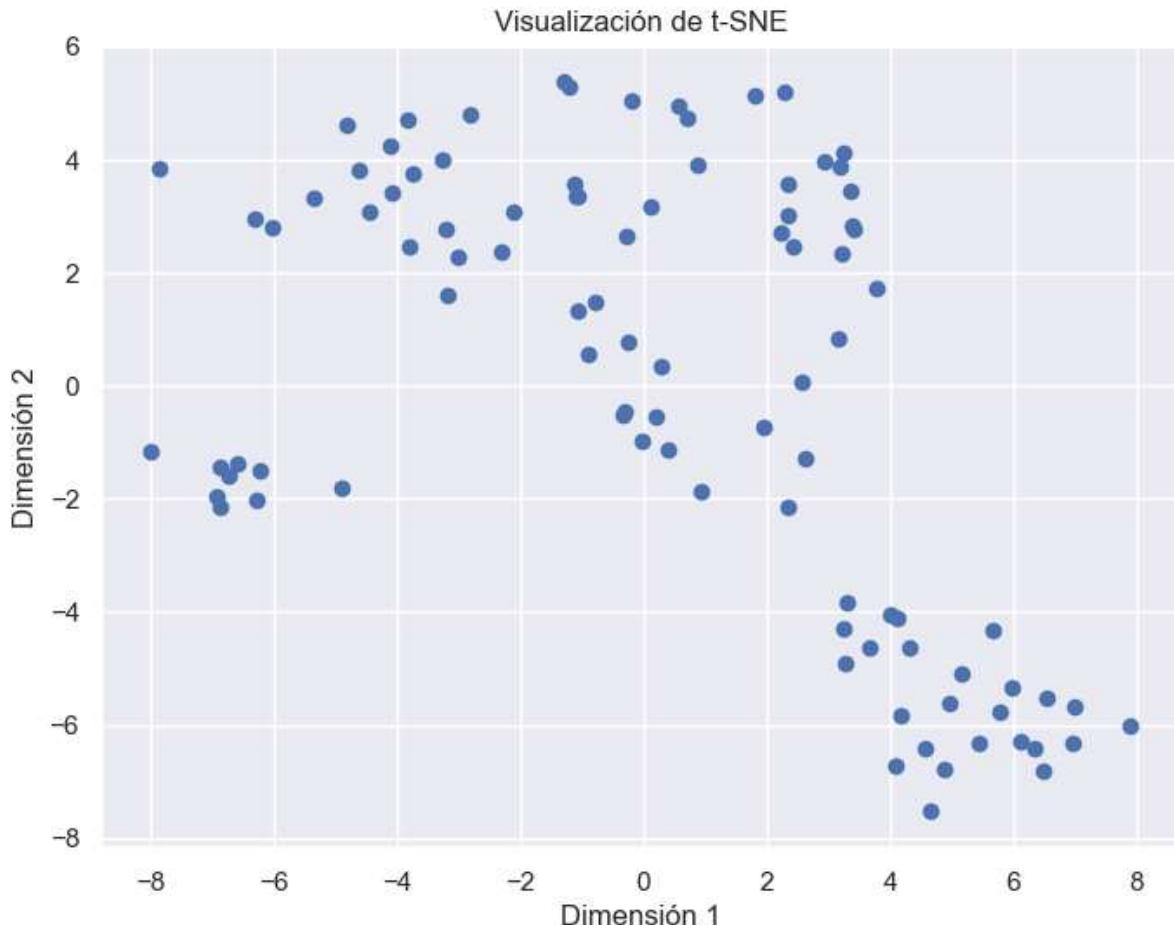
```
In [122... import numpy as np
from sklearn.manifold import TSNE

In [123... tsne = TSNE(n_components=2, perplexity=30, n_iter=300)

In [124... tsne_result = tsne.fit_transform(data)

In [127... plt.figure(figsize=(8, 6))
plt.scatter(tsne_result[:, 0], tsne_result[:, 1], cmap='rainbow')
plt.title('Visualización de t-SNE')
plt.xlabel('Dimensión 1')
plt.ylabel('Dimensión 2')
plt.show()
```

C:\Users\rodri\AppData\Local\Temp\ipykernel\_27848\1918887118.py:2: UserWarning: No data for colormapping provided via 'c'. Parameters 'cmap' will be ignored  
plt.scatter(tsne\_result[:, 0], tsne\_result[:, 1], cmap='rainbow')



```
In [132...]: num_clusters = 8

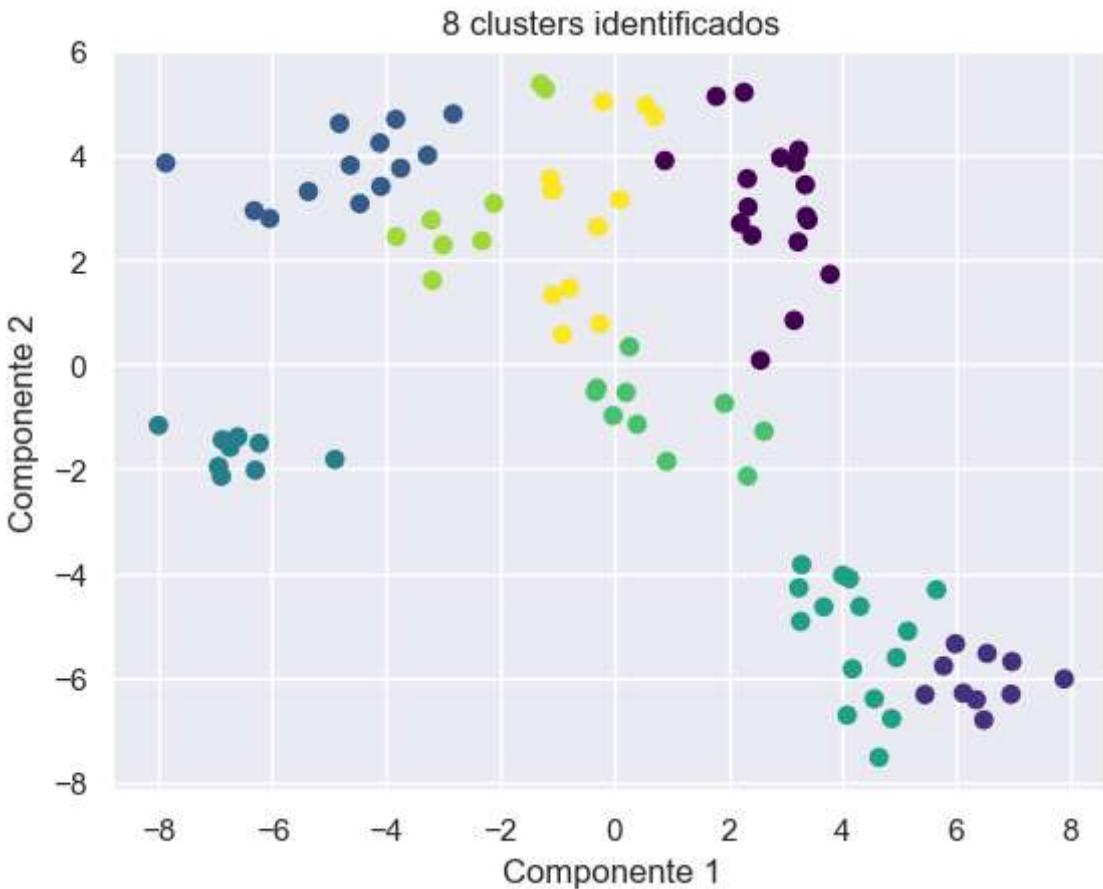
kmeans = KMeans(n_clusters=num_clusters)

kmeans.fit(data_pca)

cluster_labels = kmeans.labels_

plt.scatter(tsne_result[:, 0], tsne_result[:, 1], c=cluster_labels, cmap='viridis')
plt.xlabel('Componente 1')
plt.ylabel('Componente 2')
plt.title(f'{num_clusters} clusters identificados')
plt.savefig('clases_tsne.jpg')
plt.show()
```

```
C:\Users\rodri\AppData\Roaming\jupyterlab-desktop\jlab_server\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\rodri\AppData\Roaming\jupyterlab-desktop\jlab_server\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```



## Conclusiones

- Sin lugar a dudas cada técnica vista aquí y en la literatura nos permite observar diferentes virtudes y defectos.
- Si se tiene la certeza de que las relaciones son lineales, es mucho mejor técnica el clásico PCA.
- La reducción dimensional es un elemento fundamental para la ciencia de datos.
- Espero tener la oportunidad de platicar sobre cada unas de las técnicas y el potencial que en su experiencia tienen.

## Referencias

Dr. José Ortiz Béjar (2023), Jupyter Notebook sobre Métodos de Proyección. Disponible en:  
[https://colab.research.google.com/drive/1NVG3tMSO6VDMLKjEATI16IU6hjCMm4-I?  
usp=sharing](https://colab.research.google.com/drive/1NVG3tMSO6VDMLKjEATI16IU6hjCMm4-I?usp=sharing)

Para la actividad solicitada se utilizan las funciones implementadas en pandas  
[\(https://pandas.pydata.org/docs/user\\_guide/missing\\_data.html\)](https://pandas.pydata.org/docs/user_guide/missing_data.html)

B.K. Tripathy, Anvershirithaa Sundareswaran and Shruti Ghela (2022). Unsupervised Learning Approaches for Dimensionality Reduction and Data Visualizations. Disponible en:  
[https://aulavirtual.infotec.mx/pluginfile.php/84918/mod\\_label/intro/Unsupervised%20Learning%20Press%202021%29%281%29.pdf?time=1680398740666](https://aulavirtual.infotec.mx/pluginfile.php/84918/mod_label/intro/Unsupervised%20Learning%20Press%202021%29%281%29.pdf?time=1680398740666)