Visualización de Datos

Ejercicio 5: Reducción Dimensionalidad

Francisco Javier Mercader Martínez Rubén Gil Martínez Francisco Barba Bernal
Guillermo López Pérez Javier Martínez Manresa

1 Objetivos

- Aplicar técnicas de reducción de dimensionalidad (PCA y t-SNE) al conjunto de datos *Iris*.
- Determinar el número óptimo de clusters usando índices de *silhouette* y dendograma
- Ejecutar clustering (K-Means) sobre las proyecciones PCA y t-SNE y comparar resultados con las etiquetas reales.

2 Materiales

- Conjunto de datos *Iris* (`sklearn.datasets`).
- Python 3.x y las librerías:
  - `pandas`
  - `scikit-learn` (PCA, t-SNE, K-Means, silhouette)
  - `scipy` (jerárquico, dendograma)
  - `matplotlib`
  - `kneed` (detección automática del "codo")

3 Fundamentos

En problemas multivariables, el elevado número de dimensiones complica la visualización y puede llevar a la "maldición de la dimensionalidad": aumento de la dispersión de los datos y pérdida de eficacia de muchos algoritmos. Para mitigar esto, usamos:

- **PCA (Análisis de Componentes Principales):** técnica lineal que maximiza la varianza explicada por componentes ortogonales.
- **t-SNE (t-Stochastic Neighbor Embedding):** método no lineal que preserva relaciones de proximidad local para visualización en 2D.

Para el agrupamiento:

- **K-Means:** particional, minimiza la inercia dentro de cada cluster.
- **Silhouette:** índice para cuantificar la calidad de separación; valores cercanos a 1 indican clusters bien definidos.
- **Clustering jerárquico:** construyen un dendograma para extraer el número de grupos óptimo.

4 Procedimiento general

1. Cargar y describir (filas, columnas) el dataset Iris.
2. Reducir a 2D con PCA y con t-SNE.
3. Determinar `k` óptimo:

- Gráfica índice silhouette para `k=2..10`.
- Dendograma de clustering jerárquico.

4. Ejecutar K-Means con `k` elegido sobre ambas proyecciones.
5. Visualizar clusters vs etiquetas reales.

5 Actividad 1: Carga de datos y reducción de dimensionalidad

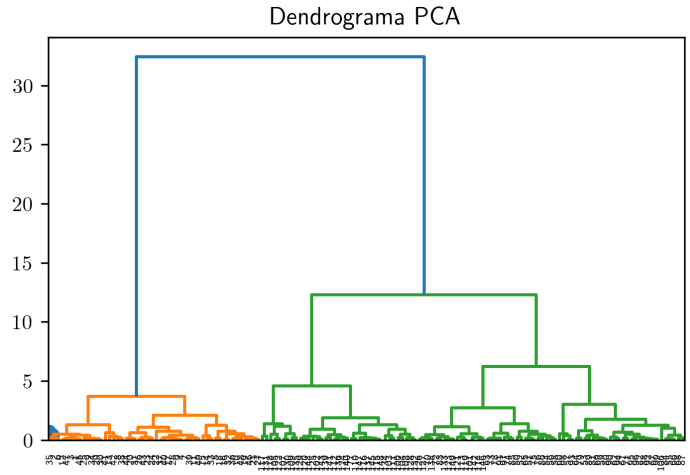
```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE

# Carga y descripción
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
print(df.describe())

# PCA a 2D
pca = PCA(n_components=2, random_state=42)
X_pca = pca.fit_transform(df)
print("Varianza PCA:", pca.explained_variance_ratio_)

# t-SNE a 2D
tsne = TSNE(n_components=2, init='random', random_state=42)
X_tsne = tsne.fit_transform(df)
```

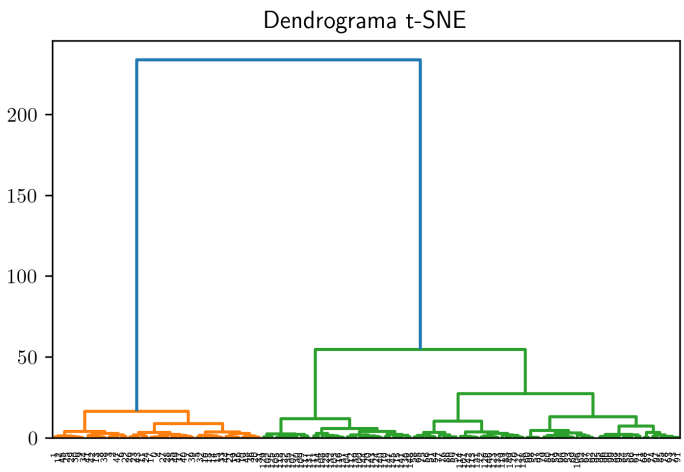
```
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000
mean            5.843333          3.057333           3.758000
std             0.828066          0.435866           1.765298
min             4.300000          2.000000           1.000000
25%             5.100000          2.800000           1.600000
50%             5.800000          3.000000           4.350000
75%             6.400000          3.300000           5.100000
max             7.900000          4.400000           6.900000

       petal width (cm)
count        150.000000
mean           1.199333
std            0.762238
min            0.100000
25%            0.300000
50%            1.300000
75%            1.800000
max            2.500000
Varianza PCA: [0.92461872 0.05306648]
```

6 Actividad 2: Selección de número de clusters

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from scipy.cluster.hierarchy import linkage, dendrogram
import matplotlib.pyplot as plt

plt.rcParams['text.usetex'] = True

# Índice silhouette PCA
sil_scores = []
ks = range(2, 11)
for k in ks:
```

```
    labels = KMeans(n_clusters=k, random_state=42).fit_predict(X_pca)
    sil_scores.append(silhouette_score(X_pca, labels))
plt.plot(ks, sil_scores, marker='o'); plt.title('Silhouette PCA')

# Dendrograma jerárquico PCA
link = linkage(X_pca, method='ward')
dendrogram(link)
plt.title('Dendrograma PCA')
plt.show()

# Dendrograma jerárquico t-SNE
link = linkage(X_tsne, method='ward')
dendrogram(link)
plt.title('Dendrograma t-SNE')
plt.show()
```





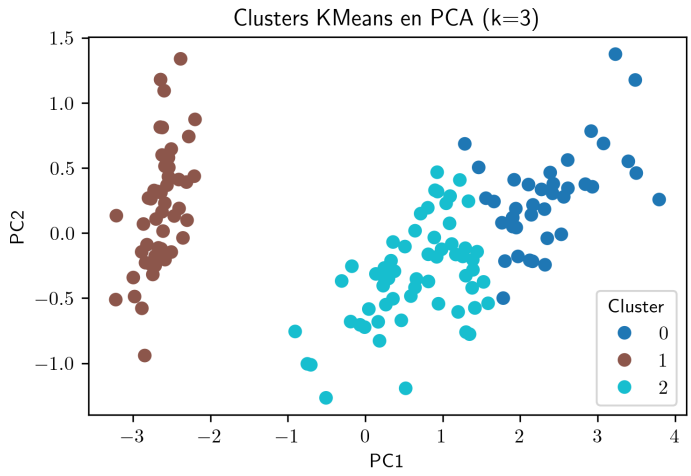
6.1 Selección automática de k_opt con kneed

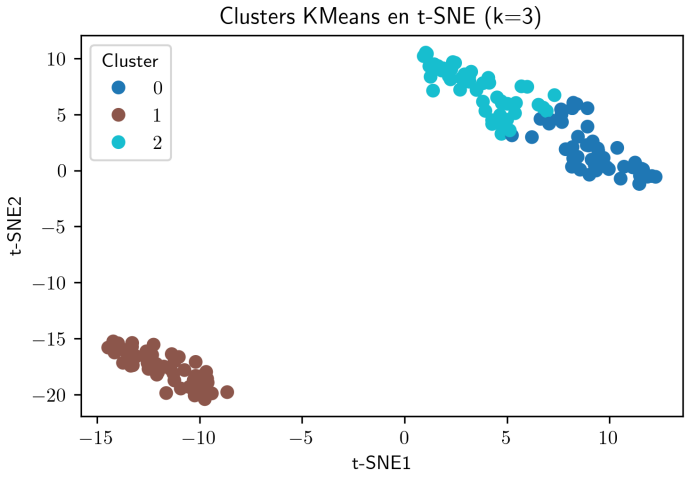
```
from kneed import KneeLocator
import numpy as np

# Usamos la proyección PCA para detectar el "codo"
ks_all = range(1, 11)

# 1) inercia para cada k
inertias = [KMeans(n_clusters=k, random_state=42).fit(X_pca).inertia_ for k in ks_all]

# 2) Silhouette para k>=2
sil_scores = [silhouette_score(X_pca, KMeans(n_clusters=k, random_state=42).fit_predict(X_pca))
    for k in range(2, 11)]

# 3) Detección automática de codo
kl = KneeLocator(ks_all , inertias, curve='convex', direction='decreasing')
k_elbow = kl.elbow

# 4) k con mejor silhouette
k_sil = np.arange(2, 11)[np.argmax(sil_scores)]

print(f"Elbow (inercia): k = {k_elbow}")
print(f"Silhouette óptimo: k = {k_sil}")

# Definimos k_opt como el valor de k_elbow
k_opt = k_elbow
```

```
Elbow (inercia): k = 3
Silhouette óptimo: k = 2
```

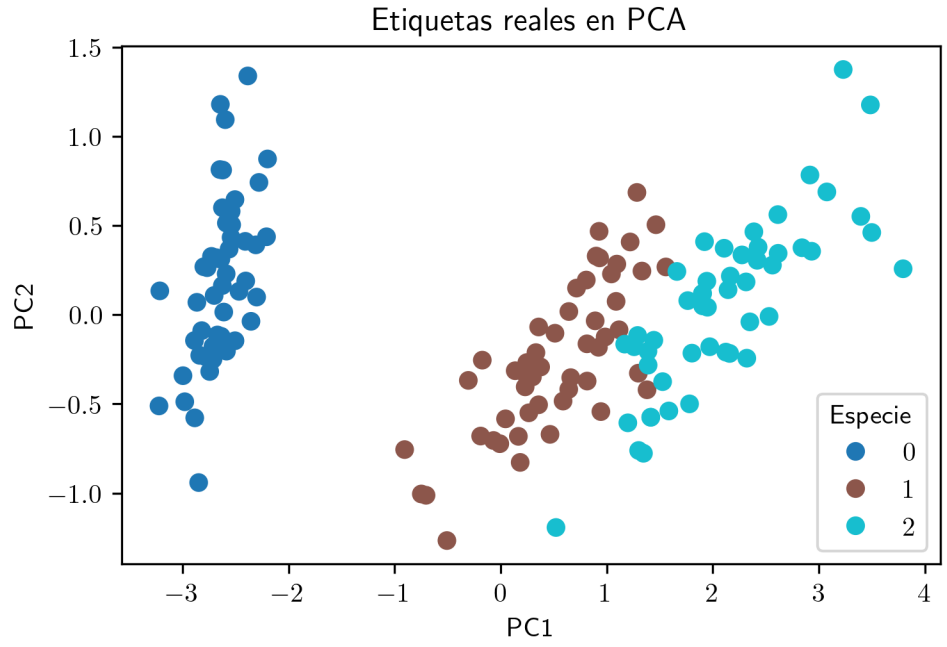
7 Actividad 3: Clustering y visualización con k_opt

```
# Clustering con k_opt sobre PCA y t-SNE
df['cluster_pca'] = KMeans(n_clusters=k_opt, random_state=42).fit_predict(X_pca)
df['cluster_tsne'] = KMeans(n_clusters=k_opt, random_state=42).fit_predict(X_tsne)

# Scatter PCA con clusters
plt.figure()
scatter_pca = plt.scatter(X_pca[:, 0], X_pca[:, 1], c=df['cluster_pca'], cmap='tab10')
plt.title(f'Clusters KMeans en PCA (k={k_opt})')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend(*scatter_pca.legend_elements(), title='Cluster')
plt.show()

# Scatter t-SNE con clusters
plt.figure()
scatter_tnse = plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=df['cluster_tsne'], cmap='tab10')
plt.title(f'Clusters KMeans en t-SNE (k={k_opt})')
plt.xlabel('t-SNE1')
plt.ylabel('t-SNE2')
plt.legend(*scatter_tnse.legend_elements(), title='Cluster')
plt.show()
```






```
# Comparación con etiquetas reales
plt.figure()
scatter_real = plt.scatter(X_pca[:, 0], X_pca[:, 1], c=iris.target, cmap='tab10')
plt.title('Etiquetas reales en PCA')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend(*scatter_real.legend_elements(), title='Especie')
plt.show()
```



8 Resultados

- **PCA**: PC1 92,46 %, PC2 5,31 % varianza explicada.
- **t-SNE**: capturó la estructura local; mayor dispersión no lineal.
- **Silhouette PCA**: máximo en k=2 (~0,71), segundo pico en k=3 (~0,56).
- **Dendrograma PCA**: corte sugerido en 3 grupos.
- **kneed**: `k_elbow`=3, `k_sil`=2 (usamos `k_opt`=3).
- **Clustering**: PCA separa bien *setosa*; *versicolor*/ *virginica* se solapan. t-SNE mejora ligeramente distinciones.

9 Discusión

- t-SNE ofrece visualización más nítida de subgrupos, pero es no lineal y menos interpretativa.
- La detección automática con kneed simplifica la elección de k.
- Validar siempre con varios índices y visualizaciones.