Curso: Inteligencia Artificial

Unidad 2: Aprendizaje automático

Sesión 15: *Clustering*

Docente: Carlos R. P. Tovar

Universidad Tecnológica del Perú

INICIO

¿Tienen dudas o consultas sobre la clase previa?

OBJETIVO
Objetivos de la sesión

Al finalizar la sesión, los alumnos serán capaces de:

- Comprender los fundamentos del aprendizaje no supervisado y *clustering*
- Implementar algoritmos de clustering como K-Means y DBSCAN
- Evaluar la calidad de los clusters generados
- Visualizar e interpretar resultados de clustering
- Aplicar técnicas de clustering a problemas reales

UTILIDAD
Ejemplos del mundo real:

Marketing:

- Segmentacion de clientes
- Analisis de comportamiento de compra

Medicina:

- Agrupamiento de pacientes por sintomas
- Clasificacion de celulas y tejidos

Tecnologia:

- Deteccion de anomalias en redes
- Organizacion de documentos
- Sistemas de recomendacion

Ciencia de datos:

- Preprocesamiento de datos
- Reduccion de dimensionalidad

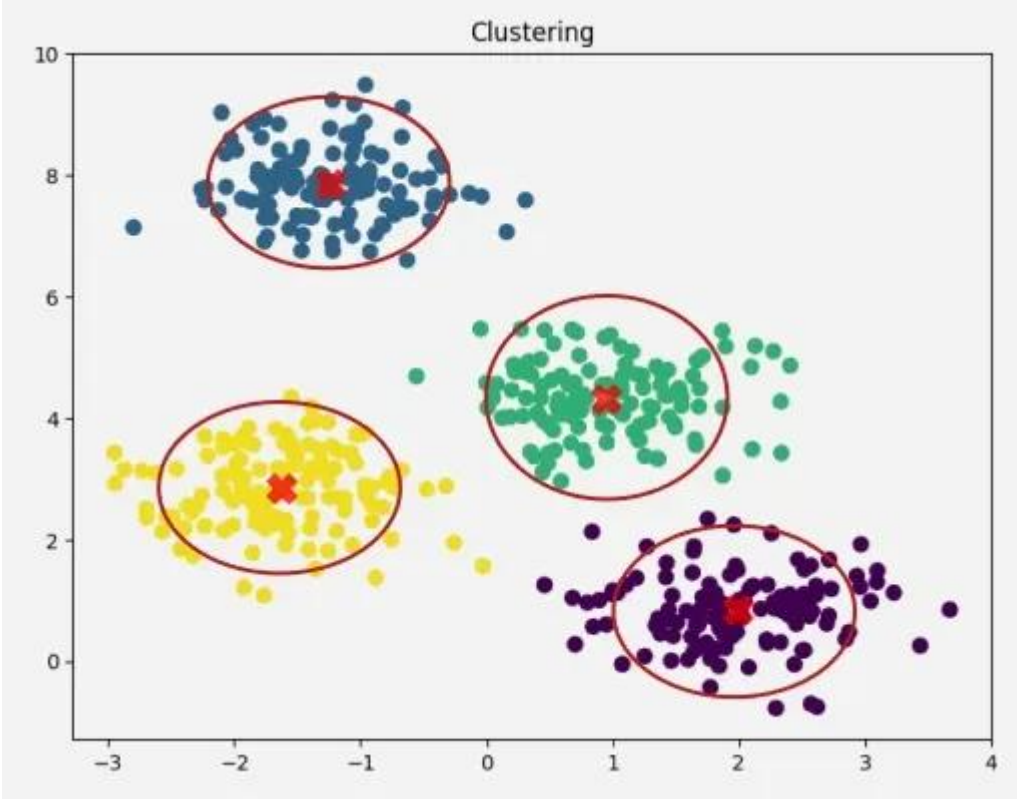
TRANSFORMACIÓN
¿Qué es el Clustering?

Definición:

- Técnica de aprendizaje no supervisado para agrupar datos similares
- Los objetos en un cluster son mas similares entre si que con objetos de otros clusters
- No se usan etiquetas predefinidas

Objetivos principales:

- Descubrir estructura en los datos
- Reducir la complejidad
- Identificar patrones ocultos



https://www.geeksforgeeks.org/machine-learning/clustering-in-machine-learning/

Tipos de clustering:

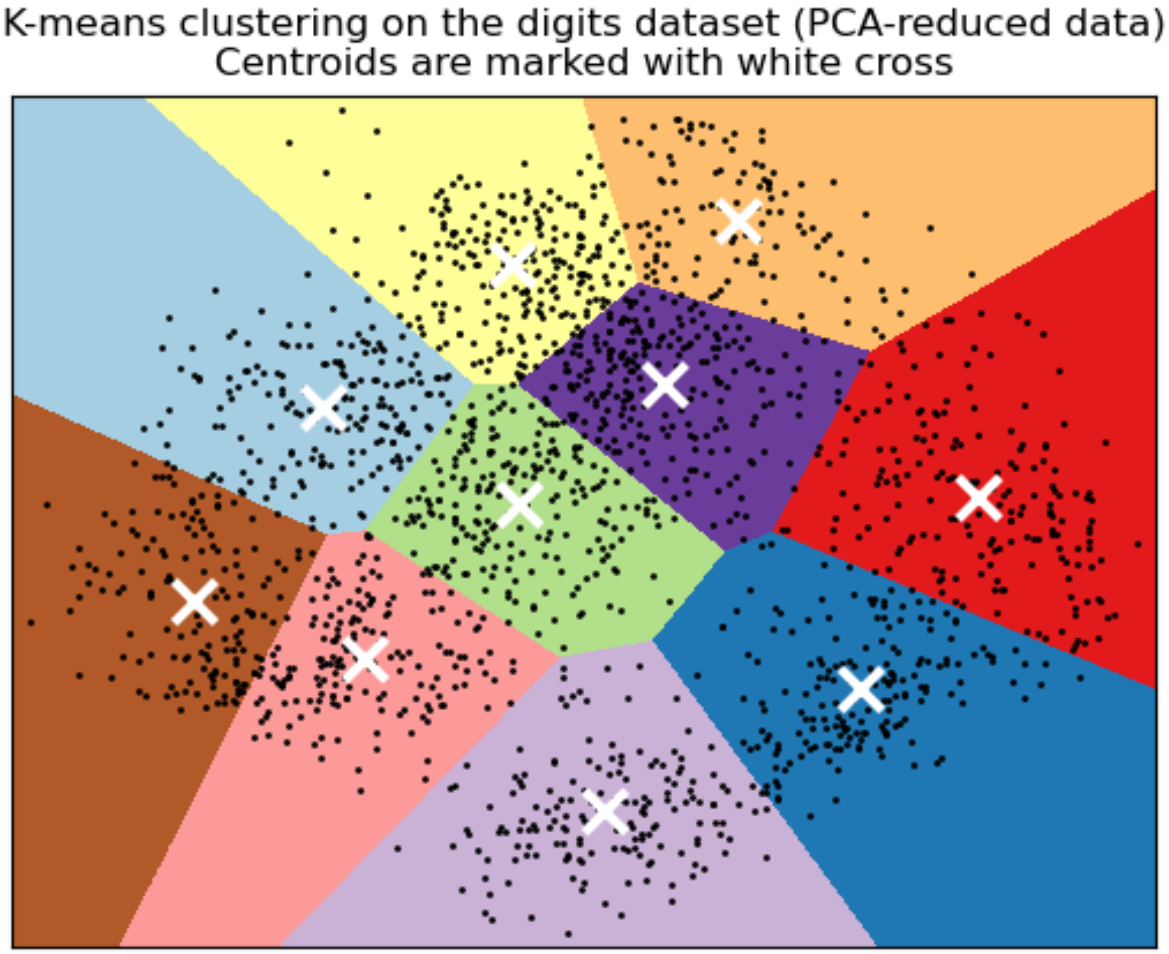
- Particional (K-Means)
- Jerarquico
- Basado en densidad (DBSCAN)

Algoritmo K-Means

Como funciona K-Means:

- Seleccionar K puntos aleatorios como centroides iniciales
- Asignar cada punto al centroide mas cercano
- Recalcular los centroides como promedios de los puntos asignados
- Repetir pasos 2-3 hasta convergencia

Algoritmo K-Means



https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_digits.html

Algoritmo K-Means

Ventajas:

- Simple y rapido
- Escalable a grandes datasets

Desventajas:

- Sensible a inicializacion
- Requiere especificar K
- Sensible a outliers

Algoritmo DBSCAN

DBSCAN (Density-Based Spatial Clustering)

Conceptos clave:

- Punto core: Tiene min_samples puntos en su radio epsilon
- Punto frontera: Esta en el radio de un punto core pero no es core
- Ruido: No es core ni frontera

Algoritmo DBSCAN

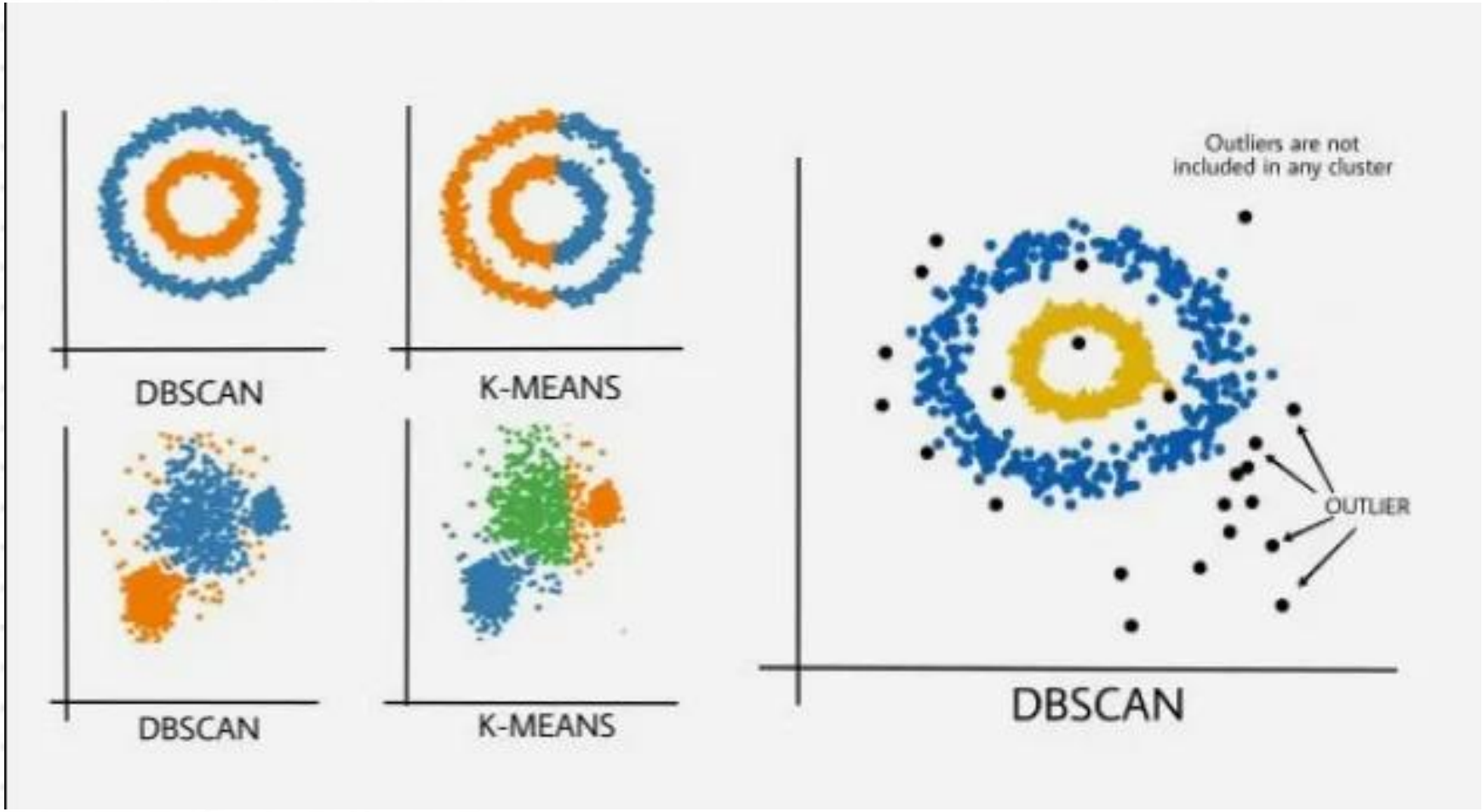
Ventajas:

- No requiere especificar numero de clusters
- Encuentra clusters de formas arbitrarias
- Robusto a outliers

Desventajas:

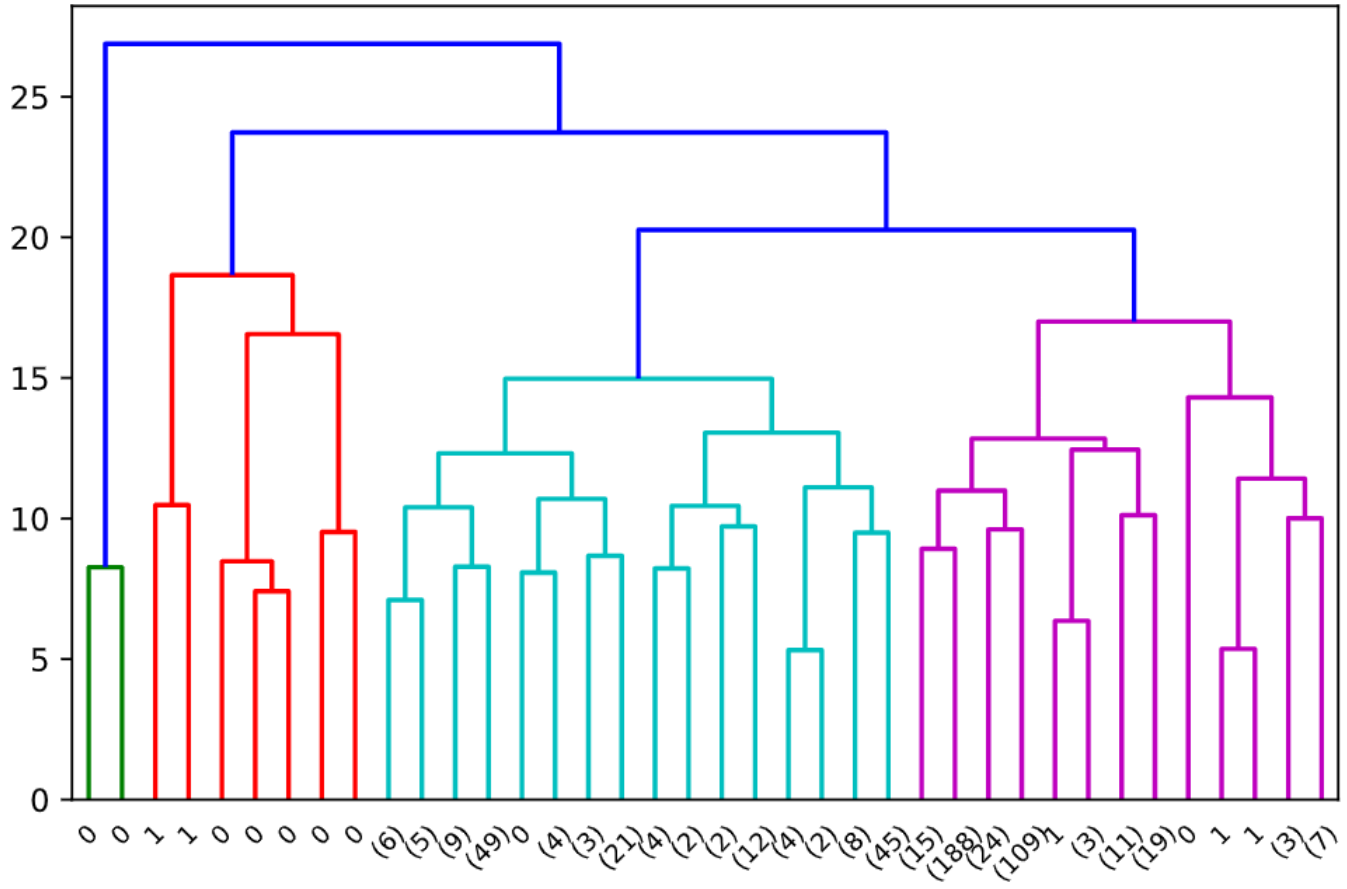
- Sensible a parametros epsilon y min_samples
- Dificil con clusters de densidad variable

Algoritmo DBSCAN



https://www.geeksforgeeks.org/machine-learning/dbscan-clustering-in-ml-density-based-clustering/

Algoritmo DBSCAN



https://healthdataminer.com/analitica-en-accion/modelos-no-supervisados-en-salud-clusterizando-celulas/

Métricas de Evaluación

Metricas internas:

- Silhouette Score: Mide cohesion y separacion
- Davies-Bouldin Index: Ratio de dispersion intra-cluster vs inter-cluster
- Calinski-Harabasz Index: Ratio de dispersion entre clusters vs dentro de clusters

Metricas externas (cuando hay etiquetas reales):

- Adjusted Rand Index
- Normalized Mutual Information
- Homogeneity, Completeness, V-score

Implementación en Python - K-Means

```
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt

X, y_true = make_blobs(n_samples=300, centers=4, random_state=42)

kmeans = KMeans(n_clusters=4, random_state=42)
y_pred = kmeans.fit_predict(X)

plt.scatter(X[:, 0], X[:, 1], c=y_pred, cmap='viridis')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
marker='x', s=200, linewidths=3, color='red')
plt.title('K-Means Clustering')
plt.show()
```

Implementación en Python - DBSCAN

```
from sklearn.cluster import DBSCAN
from sklearn.datasets import make_moons
import matplotlib.pyplot as plt

X, y_true = make_moons(n_samples=300, noise=0.05, random_state=42)

dbscan = DBSCAN(eps=0.3, min_samples=5)
y_pred = dbscan.fit_predict(X)

plt.scatter(X[:, 0], X[:, 1], c=y_pred, cmap='viridis')
plt.title('DBSCAN Clustering')
plt.show()

print(f"Numero de clusters: {len(set(y_pred)) - (1 if -1 in y_pred else 0)}")
print(f"Puntos considerados ruido: {list(y_pred).count(-1)}")
```

Determinando el Número Óptimo de Clusters

```
from sklearn.metrics import silhouette_score
import numpy as np
def find_optimal_clusters(X, max_k=10):
wcss = []
silhouette_scores = []
for k in range(2, max_k + 1):
    kmeans = KMeans(n_clusters=k, random_state=42)
    labels = kmeans.fit_predict(X)
    wcss.append(kmeans.inertia_)
    silhouette_scores.append(silhouette_score(X, labels))
```

Determinando el Número Óptimo de Clusters

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
ax1.plot(range(2, max_k + 1), wcss, 'bo-')
ax1.set_title('Metodo del Codo')
ax1.set_xlabel('Numero de Clusters')
ax1.set_ylabel('WCSS')
```

```
ax2.plot(range(2, max_k + 1), silhouette_scores, 'ro-')
ax2.set_title('Puntuacion Silhouette')
ax2.set_xlabel('Numero de Clusters')
ax2.set_ylabel('Silhouette Score')

plt.tight_layout()
plt.show()

find_optimal_clusters(X)
```

Comparación de Algoritmos

```
from sklearn.cluster import
AgglomerativeClustering
import pandas as pd

def
compare_clustering_algorithms(X):
algorithms = {
'K-Means': KMeans(n_clusters=3,
random_state=42),
'DBSCAN': DBSCAN(eps=0.5,
min_samples=5),
'Agglomerative':
AgglomerativeClustering(n_clusters=
3)
}
```

```
results = []

fig, axes = plt.subplots(1, 3, figsize=(1
5, 4))


for idx, (name, algorithm) in enumerat
e(algorithms.items()):
    y_pred = algorithm.fit_predict(X)


    if hasattr(algorithm, 'labels_'):
        labels = algorithm.labels_
    else:
        labels = y_pred
```

Comparación de Algoritmos

```
    if len(set(labels)) > 1:
        silhouette = silhouette_score(X, labels)
    else:
        silhouette = -1


    axes[idx].scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis')
    axes[idx].set_title(f'{name}\nSilhouette: {silhouette:.3f}')
```

```
results.append({
        'Algorithm': name,
        'Silhouette_Score': silhouette,
        'N_Clusters': len(set(labels))
    })


plt.tight_layout()
plt.show()
return pd.DataFrame(results)
results_df = compare_clustering_algorithms(X)
print(results_df)
```

PRACTICA
Ejercicio Práctico - Segmentación de Clientes

```
import pandas as pd
from sklearn.preprocessing import StandardScaler

def customer_segmentation_exercise():
data = {
'Age': [25, 45, 35, 50, 23, 40, 60, 48, 33, 55],
'Annual_Income': [15, 80, 45, 25, 10, 90, 35, 70, 50, 30],
'Spending_Score': [39, 77, 55, 40, 25, 85, 30, 75, 60, 35]
}
```

```
df = pd.DataFrame(data)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df)


kmeans = KMeans(n_clusters=3, random_state=42)
df['Cluster'] = kmeans.fit_predict(X_scaled)
```

PRACTICA
Ejercicio Práctico - Segmentación de Clientes

```
plt.figure(figsize=(10, 6))
scatter = plt.scatter(df['Annual_Income'], df['Spending_Score'],
                      c=df['Cluster'], cmap='viridis', s=100)
plt.xlabel('Ingreso Anual (miles)')
plt.ylabel('Puntuacion de Gasto')
plt.title('Segmentacion de Clientes')
plt.colorbar(scatter)
plt.grid(True, alpha=0.3)
plt.show()

cluster_analysis = df.groupby('Cluster').mean()
print("Caracteristicas por cluster:")
print(cluster_analysis)
```

CIERRE
Consejos Prácticos

Seleccion de algoritmo:

- K-Means: Cuando se conoce K y clusters esfericos
- DBSCAN: Para datos con ruido y clusters de densidad variable
- Jerarquico: Cuando se necesita analisis multinivel

Preprocesamiento:

- Estandarizar siempre los datos
- Considerar reduccion de dimensionalidad (PCA) si hay muchas caracteristicas

Validacion:

- Usar multiples metricas de evaluacion
- Visualizar los clusters para interpretabilidad
- Considerar el contexto del problema

Preguntas y Próximos Pasos

Preguntas comunes:

- ¿Como elijo entre K-Means y DBSCAN?
- ¿Que hacer si los clusters no se visualizan bien?
- ¿Como interpretar los resultados de clustering?

Proximos pasos:

- Practicar con datasets reales
- Explorar clustering jerarquico
- Aprender sobre clustering espectral

Recursos:

- Documentacion de scikit-learn
- Dataset Iris para practica
- Dataset Mall_Customers para segmentacion

Resumen:

- Clustering es fundamental en aprendizaje no supervisado
- K-Means y DBSCAN son algoritmos populares con diferentes fortalezas
- La evaluación y visualización son clave para interpretar resultados

Universidad
Tecnológica
del Perú