

APRENDIZAJE NO SUPERVISADO

PRESENTADO POR MyA

Asignatura: Inteligencia Artificial

Docente: Hugo Espetia Huamanga

MyA:

- Jean Carlos Huanca Pilares
- Jamil Zamata Sotec
- Del Castillo Molina Gino Galois
- Sebastian Rodrigo Anaya Rodriguez
- Zavaleta Handa Fernando Hiroshi



CLUSTERING

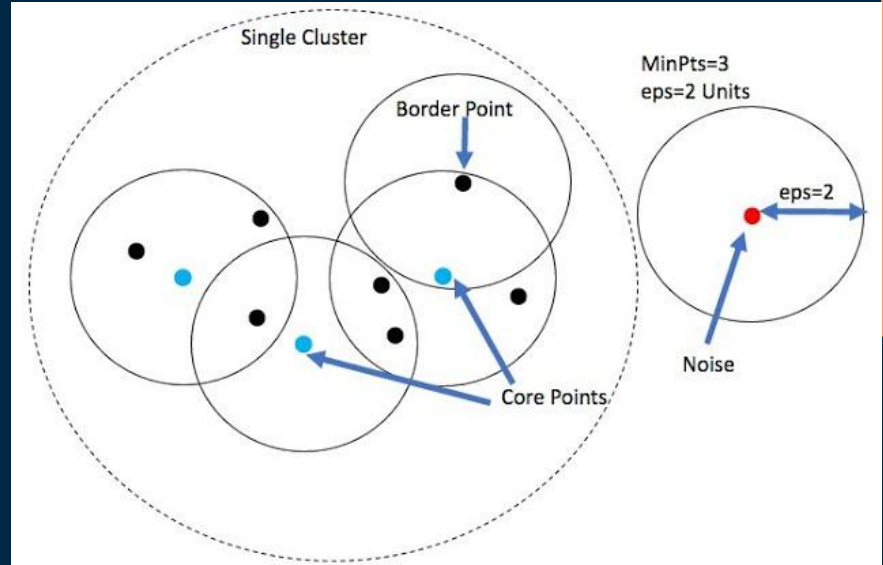
- DBScan -

DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

DBSCAN es un algoritmo de agrupamiento basado en densidad, que identifica clusters en grandes conjuntos de datos según la densidad de puntos en el espacio. A diferencia de otros métodos de Clustering (como K-means), no necesita que se especifique el número de clusters con anticipación.



- Epsilon (ϵ): Radio que define una región alrededor de cada punto. Si otros puntos están dentro de este radio, se consideran vecinos.
- MinPts: Número mínimo de puntos que debe tener una región para considerarse un cluster.
- Core Points: Puntos que tienen al menos MinPts vecinos dentro de su radio ϵ .
- Border Points: Puntos que están dentro del radio ϵ de un punto core pero que tienen menos de MinPts vecinos.
- Noise Points: Puntos que no pertenecen a ningún cluster (no son puntos core ni border).



Características clave:

- Identificación de ruido: DBSCAN marca los puntos que no encajan bien en ningún cluster como ruido, permitiendo una mejor identificación de patrones.
- Clusters de forma arbitraria: Es eficaz para detectar clusters de cualquier forma, ya que se basa en la densidad de puntos.
- No requiere especificar el número de clusters. A diferencia de K-means, el número de clusters se deriva del parámetro ϵ .

Ventajas

- Flexibilidad en formas de clusters: DBSCAN detecta clusters de formas arbitrarias, como clusters no esféricos o con distribuciones complejas.
- Detección automática de ruido.
- No requiere predefinir el número de clusters.

Desventajas

- Elección sensible de parámetros: Elegir un valor inapropiado para ϵ o MinPts puede afectar significativamente el rendimiento del algoritmo.
- Problemas en datos de alta dimensión: En espacios de alta dimensión, puede ser difícil
- encontrar una buena distancia ϵ debido a la "maldición de la dimensionalidad."

IMPLEMENTACION

- Base de datos de departamentos en cusco.csv
- Importar pandas y librerías de clustering
- Cargar archivos

Precio	Locacion	Arean2	NroDormitorios	NroBanos	NroPisos	Zona
289311	San Sebastian	125	2	1	6	Urbanizacion
529492	San Jeronimo	150	2	1	6	Urbanizacion
529820	Saylla	111	2	1	1	Institucion

```
# Instalar scikit-learn si no lo tienes instalado
!pip install -U scikit-learn

# Importar las librerías necesarias
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import DBSCAN
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df = pd.read_csv("datos_casas_cusco.csv")
print(df.head())
```

	Precio	Locacion	Arean2	NroDormitorios	NroBanos	NroPisos	\
0	289311	San Sebastian	125	2	1	6	
1	529492	San Jeronimo	150	2	1	6	
2	529820	Saylla	111	2	1	1	
3	612894	Santiago	249	3	3	8	
4	778810	Wanchaq	242	5	5	3	

	Zona
0	Urbanizacion
1	Urbanizacion
2	Institucion

- Normalización de datos

```
# Seleccionar las columnas relevantes para el clustering
columnas_relevantes = ['Precio', 'Aream2', 'NroDormitorios', 'NroBanos', 'NroPisos']
datos = df[columnas_relevantes]

# Normalizar los datos para que tengan media 0 y varianza 1
scaler = StandardScaler()
datos_normalizados = scaler.fit_transform(datos)

# Mostrar un resumen de los datos normalizados
print("Datos normalizados:\n", datos_normalizados[:5])
```

Datos normalizados:

[[-1.34612217	-0.39489029	-1.34232	-1.43231906	0.36498958]
[-0.65314851	-0.01691804	-1.34232	-1.43231906	0.36498958]
[-0.65220216	-0.60655476	-1.34232	-1.43231906	-1.53106019]
[-0.41251587	-1.47985209	-0.45248241	-0.01942726	1.12340949]

- Aplicar DBSCAN para la generación de clusters

```
# Configurar y aplicar DBSCAN
dbscan = DBSCAN(eps=0.85, min_samples=5) # Ajusta estos parámetros según sea necesario
etiquetas = dbscan.fit_predict(datos_normalizados)

# Agregar las etiquetas de los clusters al DataFrame original
df['Cluster'] = etiquetas

# Ver cuántos clusters se han encontrado (ignora el ruido, que es -1)
n_clusters = len(set(etiquetas)) - (1 if -1 in etiquetas else 0)
print(f"Número de clusters encontrados: {n_clusters}")
print(df['Cluster'].value_counts())
```

Número de clusters encontrados: 6

Cluster	
2	506
0	486
3	482
1	469
-1	51
5	3
4	3

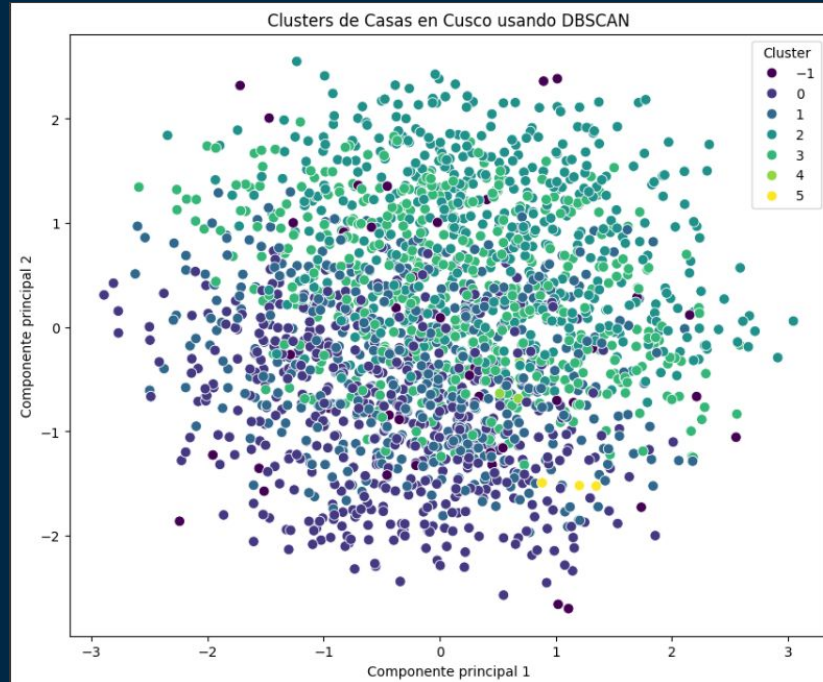
Name: count, dtype: int64

- Generacion de graficos

```
from sklearn.decomposition import PCA

# Reducir la dimensionalidad a 2D usando PCA para visualización
pca = PCA(n_components=2)
datos_reducidos = pca.fit_transform(datos_normalizados)

# Crear un scatter plot de los clusters
plt.figure(figsize=(10, 8))
sns.scatterplot(x=datos_reducidos[:, 0], y=datos_reducidos[:, 1], hue=df['Cluster'], palette='viridis', s=60)
plt.title('Clusters de Casas en Cusco usando DBSCAN')
plt.xlabel('Componente principal 1')
plt.ylabel('Componente principal 2')
plt.legend(title='Cluster')
plt.show()
```



- Estadísticas descriptivas por cluster



```
# Estadísticas descriptivas por cluster
estadisticas_clusters = df.groupby('Cluster')[columnas_relevantes].mean()
print("Estadísticas descriptivas por cluster:\n", estadisticas_clusters)
```



Estadísticas descriptivas por cluster:

	Precio	Arean2	NroDormitorios	NroBanos	NroPisos
Cluster					
-1	7.776413e+05	174.843137	3.058824	2.941176	5.529412
0	7.522917e+05	151.473251	2.000000	2.960905	5.043210
1	7.326469e+05	143.978678	3.000000	3.095949	4.735608
2	7.622094e+05	153.345850	5.000000	3.059289	5.164032
3	7.743269e+05	153.109959	4.000000	2.979253	5.188797
4	1.140024e+06	52.333333	4.000000	5.000000	1.333333
5	1.770490e+05	210.000000	4.000000	5.000000	1.000000

● Estadísticas descriptivas por cluster

1. Cluster -1 (Ruido) Precio promedio: 777,641.3 Área promedio: 174.8 m² Número de dormitorios: 3.06 Número de baños: 2.94 Número de pisos: 5.53 Este cluster representa el "ruido" o los puntos que no pudieron ser agrupados en clusters principales. Estas propiedades tienden a tener un precio alto, un área grande y un número elevado de pisos. Podrían ser casas que no se ajustan a un patrón común en los otros clusters.
2. Cluster 0 Precio promedio: 752,291.7 Área promedio: 151.5 m² Número de dormitorios: 2 Número de baños: 2.96 Número de pisos: 5.04 Las casas en este cluster tienen un precio intermedio, con un área relativamente grande y suelen tener 2 dormitorios. El número de baños y pisos es alto, lo que podría indicar que estas casas están diseñadas para aprovechar mejor el espacio vertical.
3. Cluster 1 Precio promedio: 732,646.9 Área promedio: 144 m² Número de dormitorios: 3 Número de baños: 3.1 Número de pisos: 4.74 Aquí, el precio y el área son un poco más bajos que en el cluster 0, pero estas casas tienen un promedio de 3 dormitorios y 3 baños. El número de pisos es ligeramente menor.
4. Cluster 2 Precio promedio: 762,209.4 Área promedio: 153.3 m² Número de dormitorios: 5 Número de baños: 3.06 Número de pisos: 5.16 Las casas en este cluster tienden a ser más grandes, con un promedio de 5 dormitorios, lo que podría indicar propiedades más adecuadas para familias grandes. El precio también es más alto, y el número de pisos es elevado.
5. Cluster 3 Precio promedio: 774,326.9 Área promedio: 153.1 m² Número de dormitorios: 4 Número de baños: 2.98 Número de pisos: 5.19 Las casas en el cluster 3 tienen características intermedias con un promedio de 4 dormitorios y 3 baños, similar al cluster 2 en cuanto a área, pero con un precio ligeramente superior.
6. Cluster 4 Precio promedio: 1,140,024 Área promedio: 52.3 m² Número de dormitorios: 4 Número de baños: 5 Número de pisos: 1.33 Estas casas son más caras, pero tienen un área mucho menor en comparación con otros clusters. Sin embargo, tienen muchos baños (5 en promedio), lo que sugiere que están diseñadas con un enfoque especial, posiblemente viviendas lujosas o con un propósito específico, como oficinas o alojamientos.
7. Cluster 5 Precio promedio: 177,049 Área promedio: 210 m² Número de dormitorios: 4 Número de baños: 5 Número de pisos: 1 Las propiedades en este cluster son notablemente más baratas y tienen un área grande, lo que podría indicar que son viviendas más sencillas o en ubicaciones menos cotizadas, pero con características adecuadas para familias grandes (4 dormitorios, 5 baños).

Conclusiones Cluster -1 representa el ruido, con propiedades que no encajan bien en otros clusters. Clusters 0, 1, 2 y 3 tienen características similares en cuanto a precio, área y número de habitaciones, con algunas diferencias sutiles. Clusters 4 y 5 parecen ser casos extremos con características muy específicas: uno con precios muy altos y un área pequeña (posiblemente propiedades de lujo), y el otro con precios bajos pero un área grande (posiblemente viviendas más económicas).

https://colab.research.google.com/drive/1klu7o5uJOm11qG_HoLU88HRK3J3BySgw?usp=sharing#scrollTo

The background is a dark blue field decorated with various geometric elements. There are numerous small squares in white, orange, and teal, some of which are solid and others are outlines. Thin white vertical lines of varying lengths are scattered across the composition, creating a modern, minimalist aesthetic.

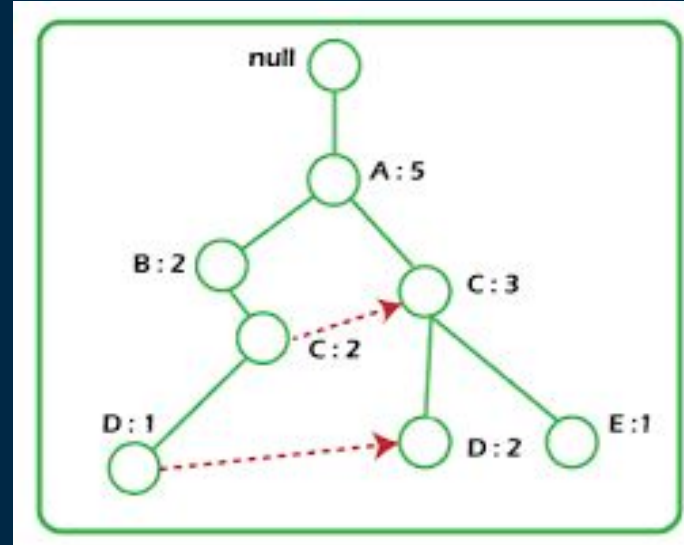
ASOCIACIÓN -FP-

FP-Growth

Los algoritmos de reglas de asociación, como FP-Growth, son técnicas que buscan descubrir relaciones interesantes entre variables en grandes bases de datos. Son especialmente útiles para análisis de transacciones, como en supermercados, donde se busca encontrar qué productos suelen comprarse juntos



FP-Growth Construye una estructura de árbol compacta (FP-tree) para evitar el recuento reiterado de itemsets frecuentes, mejorando la eficiencia en grandes bases de datos.



Características clave:

- Análisis exploratorio de grandes bases de datos: Proporciona reglas que pueden ayudar a comprender mejor los patrones ocultos.
- Eficiencia en bases de datos grandes: FP-Growth es más eficiente manejando grandes bases de datos.

Ventajas

- Mejora la eficiencia en bases de datos grandes, ya que evita la generación innecesaria de conjuntos de ítems

Desventajas

- Aunque más rápido, su implementación es más compleja y requiere más memoria.

IMPLEMENTACION

- Base de datos de departamentos en cusco.csv
- Importar pandas y mixtend
- Cargar archivos

Precio	Locacion	Aream2	NroDormitorios	NroBanos	NroPisos	Zona
289311	San Sebastian	125	2	1	6	Urbanizacion
529492	San Jeronimo	150	2	1	6	Urbanizacion
529820	Saylla	111	2	1	1	Institucion

```
[2] import pandas as pd
     from mixtend.frequent_patterns import fpgrowth, association_rules
```

```
# Cargar el archivo al DataFrame
df = pd.read_csv('/content/datos_casas_cusco.csv')

# Evaluar la estructura del DataFrame: mostrar primeras 5 filas
print(df.head())
```

```
Datos  datos_casas_cusco.csv  sample_data
   Precio  Locacion  Aream2  NroDormitorios  NroBanos  NroPisos  \
0  289311  San Sebastian    125             2         1         6
1  529492  San Jeronimo    150             2         1         6
2  529820  Saylla         111             2         1         1
```

- Convertir los datos a formato binario

```
[6] # Convertir columnas categóricas (Locacion y Zona) a formato binario
df_binarizado = pd.get_dummies(df[['Locacion', 'Zona']])

# Opcional: Binarizar los datos numéricos en rangos específicos
df_binarizado['Area_Grande'] = (df['Arean2'] > 150).astype(int)
df_binarizado['Muchos_Dormitorios'] = (df['NroDormitorios'] >= 3).astype(int)

# Visualizamos los primeros datos binarizados
print(df_binarizado.head())
```

	Locacion_Cusco	Locacion_San Jeronimo	Locacion_San Sebastian	\
0	False	False	True	
1	False	True	False	
2	False	False	False	
3	False	False	False	
4	False	False	False	

	Locacion_Santiago	Locacion_Saylla	Locacion_Wanchaq	Zona_Avenida	\
0	False	False	False	False	
1	False	False	False	False	

- Aplicar FP-Growth

```
[7] from mlxtend.frequent_patterns import fpgrowth, association_rules

# Ejecutar FP-Growth con soporte mínimo del 20% (0.2)
patrones_frecuentes = fpgrowth(df_binarizado, min_support=0.2, use_colnames=True)

# Mostrar los patrones frecuentes encontrados
print(patrones_frecuentes)

# Generar reglas de asociación con un umbral de confianza del 50%
reglas = association_rules(patrones_frecuentes, metric="confidence", min_threshold=

# Mostrar las reglas de asociación
print(reglas)
```

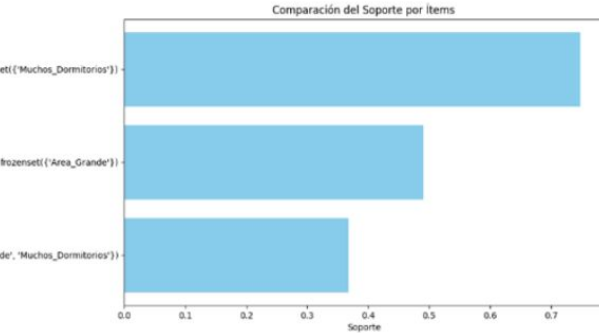
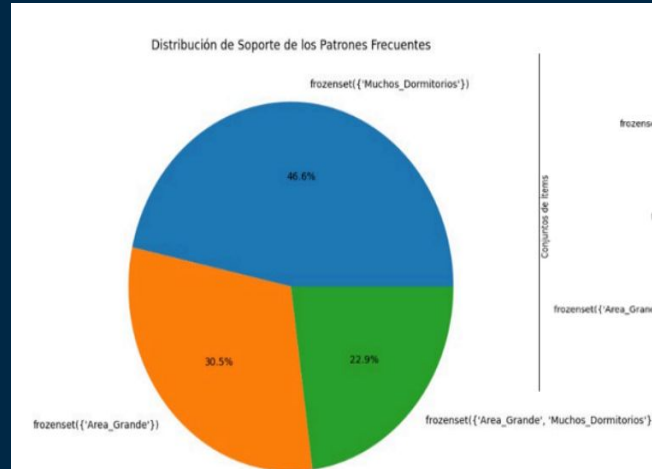
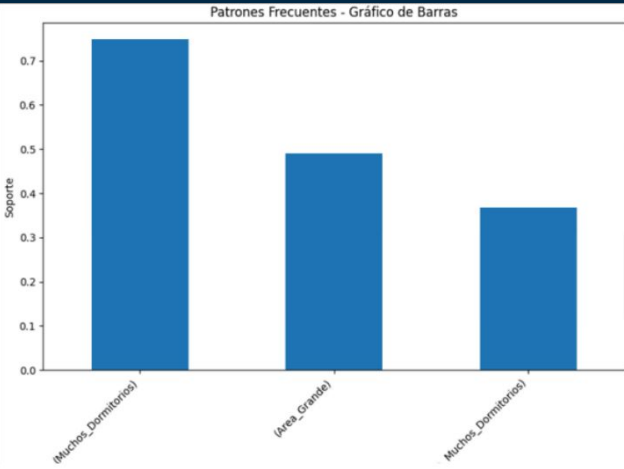
	support	itemsets
0	0.7485	(Muchos_Dormitorios)
1	0.4905	(Area_Grande)
2	0.3675	(Area_Grande, Muchos_Dormitorios)

	antecedents	consequents	antecedent support	\
0	(Area_Grande)	(Muchos_Dormitorios)	0.4905	

	consequent	support	support	confidence	lift	leverage	conviction	\
0		0.7485	0.3675	0.749235	1.000983	0.000361	1.002933	

- Generación de graficos

```
# Generar un gráfico de barras de los soportes de los patrones frecuentes
plt.figure(figsize=(10, 6))
patrones_frecuentes.plot(kind='bar', x='itemsets', y='support', legend=False, ax=plt)
plt.title('Patrones Frecuentes - Gráfico de Barras')
plt.xlabel('Conjuntos de Ítems')
plt.ylabel('Soporte')
plt.xticks(rotation=45, ha='right')
plt.show()
```



https://colab.research.google.com/drive/17wtMBh9FEaEoRRDy-50Fo7_oBXM8E__S?authu%20ser=1#scrollTo=6A3bUzwNF9a6

The background is a dark blue gradient. It is decorated with several vertical white lines of varying lengths. Scattered throughout are small squares in pink, teal, and orange. Some squares are solid, while others are outlined.

THANKS

A solid white horizontal rectangular bar.

Team : MyA