Universidad Nacional Autónoma de México

Facultad de ingeniería

Materia: Minería de Datos

Equipo: 10

Proyecto 3

Alumnos:

- Beristain Carbajal Armando
- Barrera Peña Víctor Miguel
- Torrecilla Jimenez Aaron Israel

Fecha: 31/Marzo/2024

Proyecto 3.1 Cluster

Una empresa de distribución de bebidas de diferentes marcas realiza la distribución y venta a tiendas pequeñas con diferentes surtidos de acuerdo solicitud del dueño del negocio. El propósito que se requiere es sugerir algunas bebidas que no le solicita para conocer su comportamiento de venta, para ello se deben seleccionar marcas que sea más adecuadas al lugar y tamaño de la Tienda.

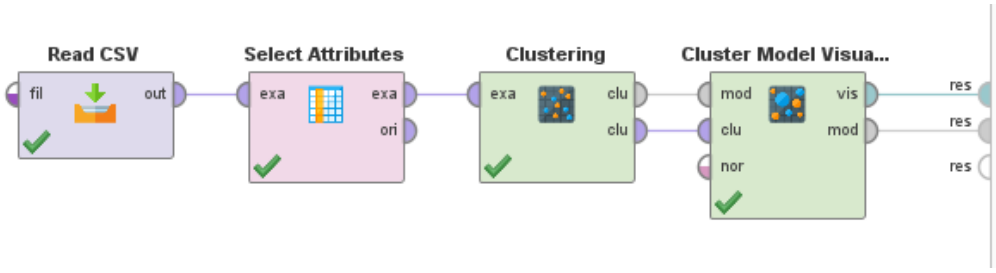
Como primera etapa se debe realizar una segmentación de tiendas en base a la Marca de bebidas vendidas.

1) Realizar los siguientes ejercicios utilizando:

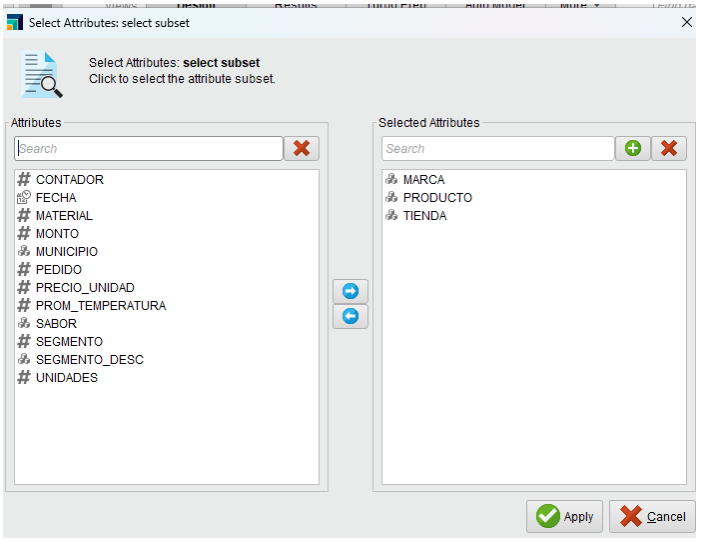
A) Con RapidMiner utilizar los algoritmos de K-means y DBScan

Algoritmo de Kmeans

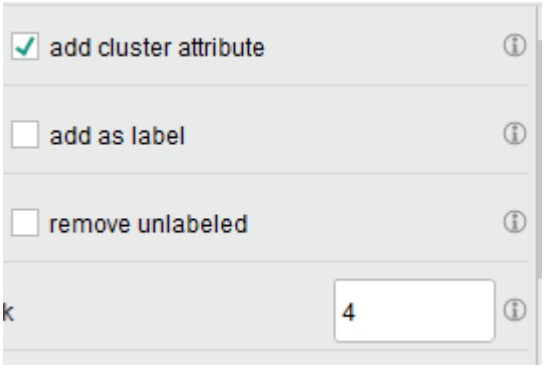
Diseño de Diagrama



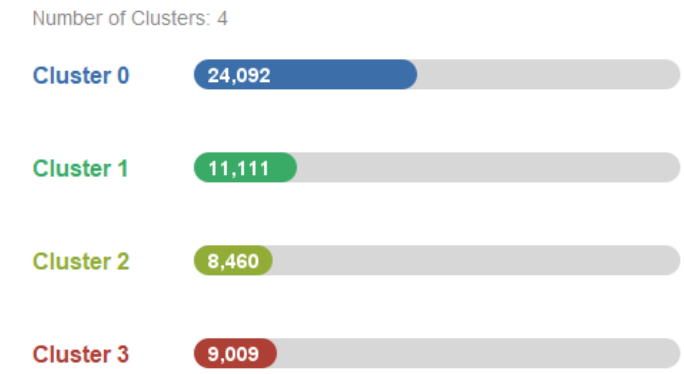
Hacemos la selección de atributos para la segmentación

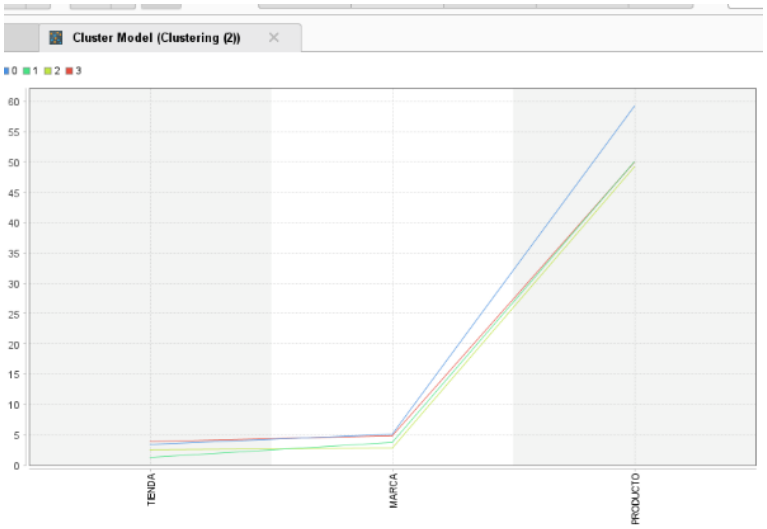


Para el algoritmo hacemos la selección de 3 clusters



Resultados del modelo

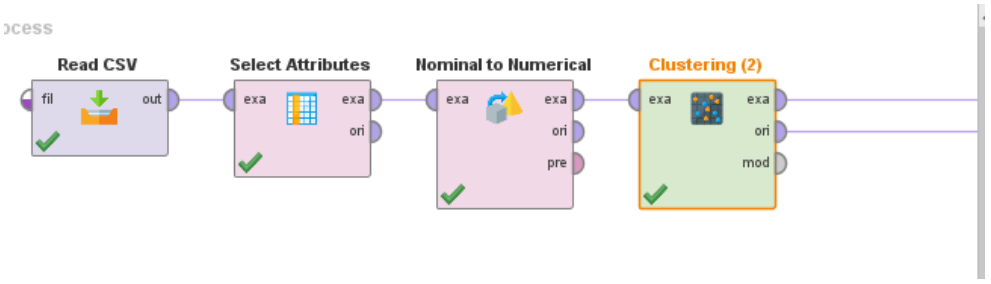




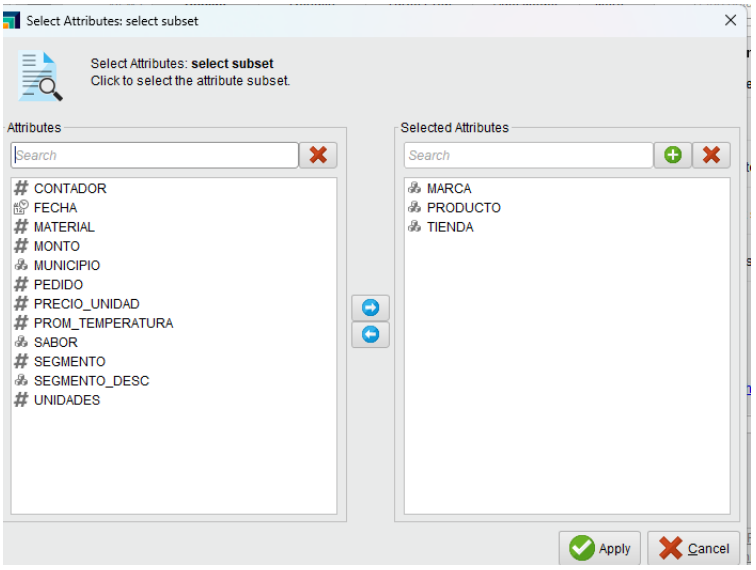
Aunque se colocó en escala automática y se moviera hasta los valores que deberían aparecer en la gráfica no se mostraba lo que se requería, y ello solo paso en la graficación, no se sabe porque, sin embargo la limpieza y acomodo de los datos es correcta.

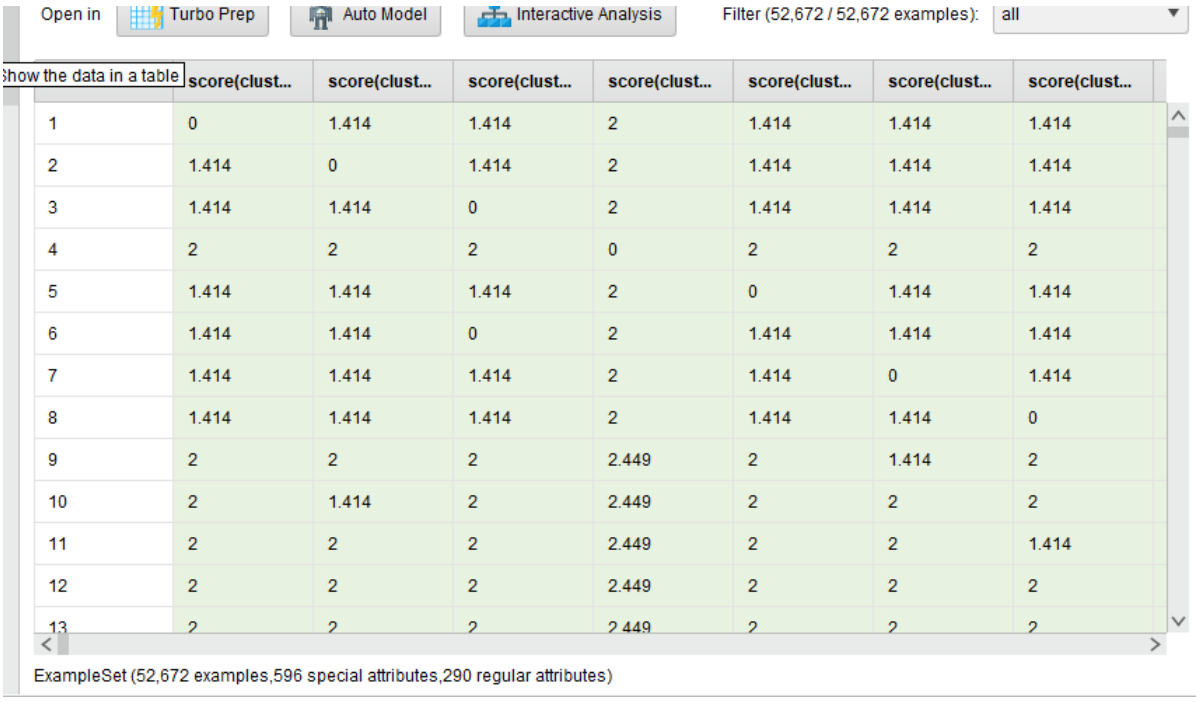
Algoritmo de DBSCAN:

Diseño del Diagrama



Se hace la selección de atributos para la segmentación





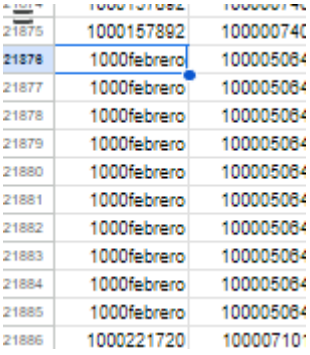
	score(clust...	score(clust...	score(clust...	score(clust...	score(clust...	score(clust...	score(clust...
1	0	1.414	1.414	2	1.414	1.414	1.414
2	1.414	0	1.414	2	1.414	1.414	1.414
3	1.414	1.414	0	2	1.414	1.414	1.414
4	2	2	2	0	2	2	2
5	1.414	1.414	1.414	2	0	1.414	1.414
6	1.414	1.414	0	2	1.414	1.414	1.414
7	1.414	1.414	1.414	2	1.414	0	1.414
8	1.414	1.414	1.414	2	1.414	1.414	0
9	2	2	2	2.449	2	1.414	2
10	2	1.414	2	2.449	2	2	2
11	2	2	2	2.449	2	2	1.414
12	2	2	2	2.449	2	2	2
13	2	2	2	2.449	2	2	2

ExampleSet (52,672 examples,596 special attributes,290 regular attributes)

PYTHON:

1. Con Python incluir el método de Elbows para determinar el número de clusters y el algoritmo de K-means
2. Con Python realizar la segmentación con DBScan
3. Se puede incluir algun algoritmo adicional que permita ver el comportamiento de los datos
4. Realizar la limpieza y Transformación que sea requerida de los datos enviados

En la columna pedido existe un error en la línea 21876, tenemos la combinación de dos valores de diferente formato



21875	1000157892	10000074(
21876	1000febrero	10000508
21877	1000febrero	10000508
21878	1000febrero	10000508
21879	1000febrero	10000508
21880	1000febrero	10000508
21881	1000febrero	10000508
21882	1000febrero	10000508
21883	1000febrero	10000508
21884	1000febrero	10000508
21885	1000febrero	10000508
21886	1000221720	10000710

Al mapearlos, se elimina esta inconsistencia y se numera como un valor más, sólo que cuando se realice un análisis es necesario tener una tabla de relación entre el n

2) Identificar el comportamiento de cada segmento y dar una explicación

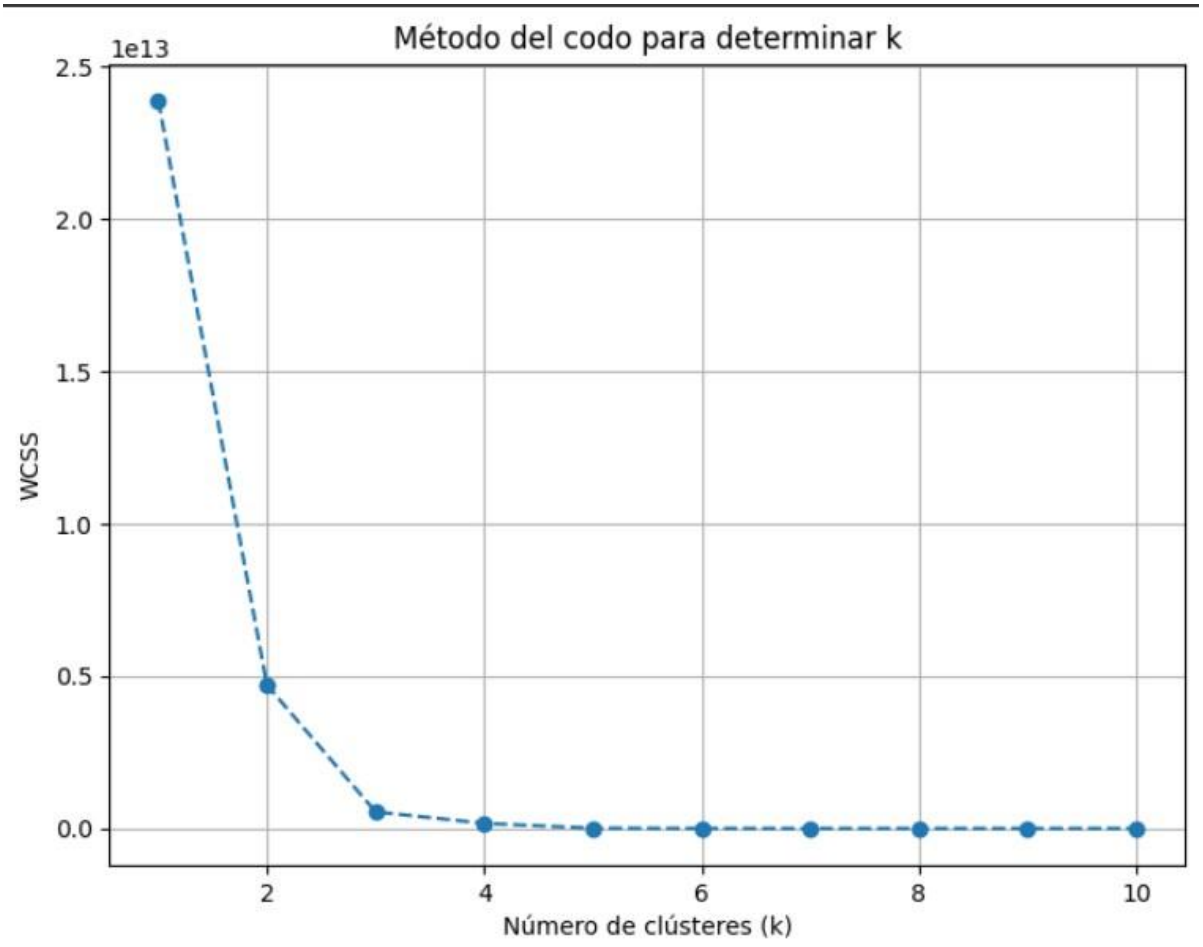
Nota: Utilizar el set de datos Venta_bebidas_Demo, se recomienda primero elegir una muestra para validar los algoritmos.

Solución

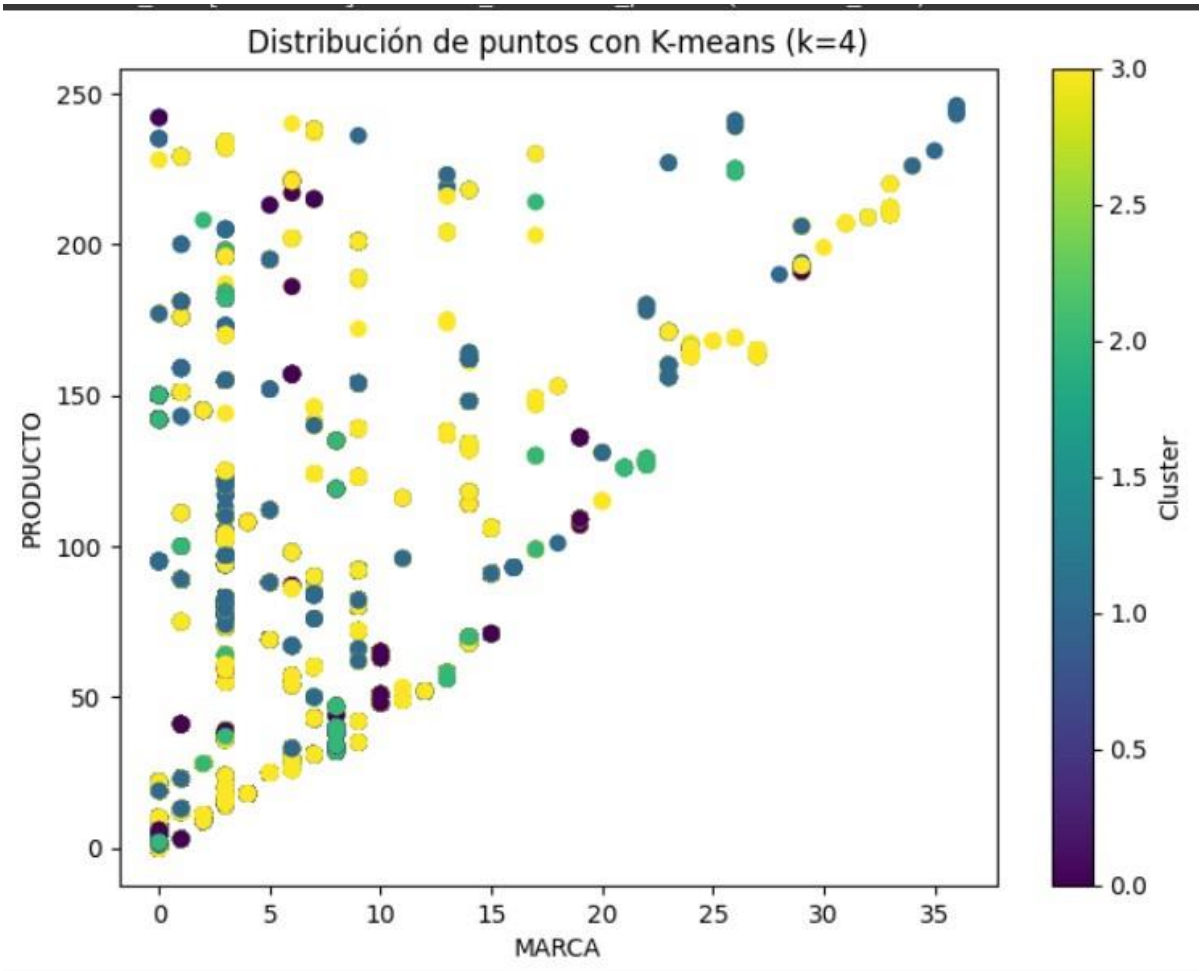
2) Con Python incluir el método de Elbows para determinar el número de clusters y el algoritmo de K-means

Elbows:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Cargar los datos desde el archivo CSV (ajusta el nombre del archivo según tu caso)
data = pd.read_csv("Ventas_Bebidas_Demo.csv")


# Crear un diccionario para mapear MARCA y PRODUCTO a valores numéricos
marca_mapping = {marca: i for i, marca in enumerate(data["MARCA"].unique())}
producto_mapping = {producto: i for i, producto in enumerate(data["PRODUCTO"].unique())}

# Reemplazar los valores categóricos con los valores numéricos
data["MARCA"] = data["MARCA"].map(marca_mapping)
data["PRODUCTO"] = data["PRODUCTO"].map(producto_mapping)

# Seleccionar las variables relevantes (MARCA, PRODUCTO, TIENDA)
selected_data = data[["MARCA", "PRODUCTO", "TIENDA"]]

# Determinar el número óptimo de clústeres utilizando la regla del codo
wcss = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, init='k-means++', random_state=42)
    kmeans.fit(selected_data)
    wcss.append(kmeans.inertia_)

# Graficar la curva del codo
plt.figure(figsize=(8, 6))
plt.plot(range(1, 11), wcss, marker='o', linestyle='--')
plt.xlabel('Número de clústeres (k)')
plt.ylabel('WCSS')
plt.title('Método del codo para determinar k')
plt.grid()
plt.show()

# Seleccionar el valor óptimo de k basado en la gráfica
optimal_k = 3  # Supongamos que el codo está en k=3
kmeans_final = KMeans(n_clusters=optimal_k, init='k-means++', random_state=42)
selected_data['cluster'] = kmeans_final.fit_predict(selected_data)
```

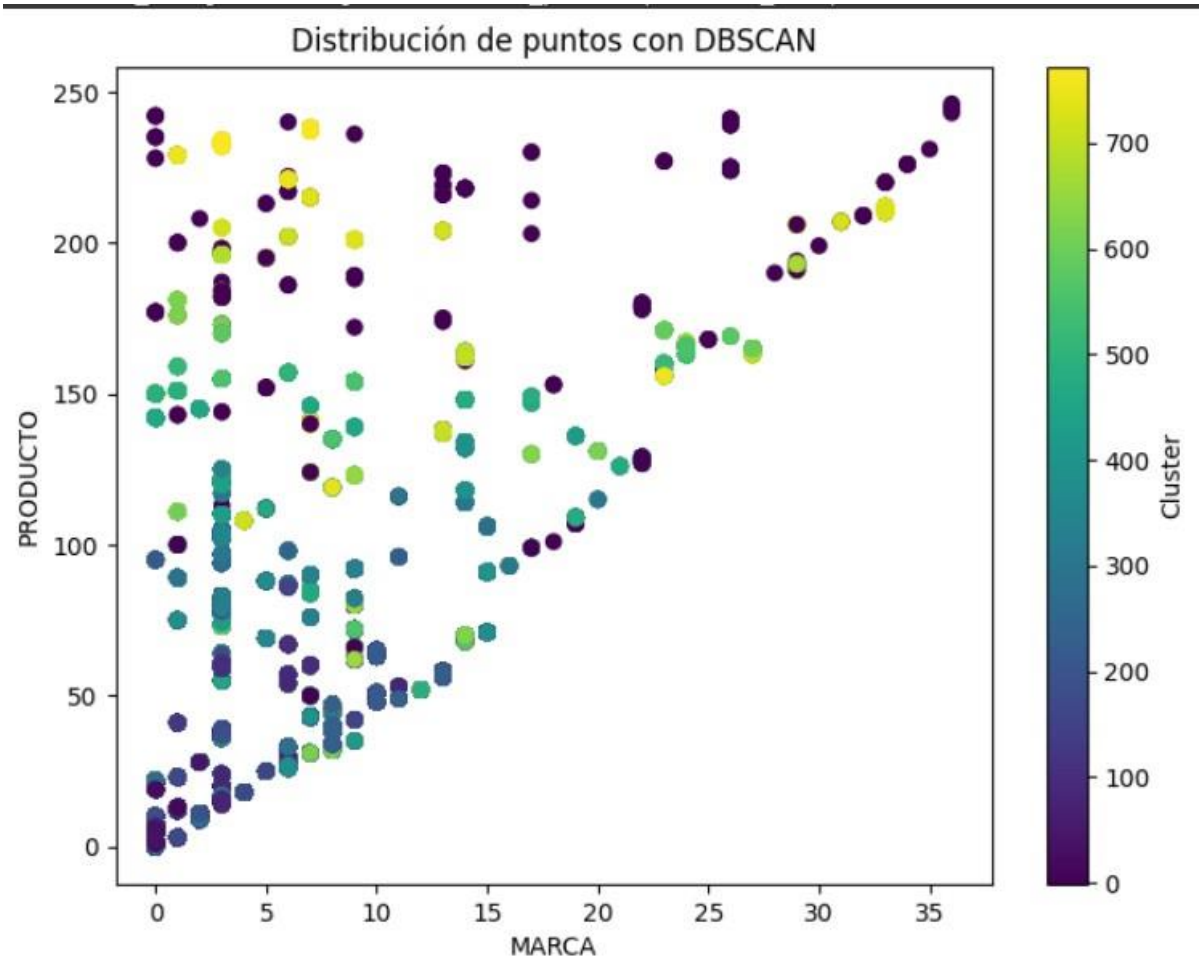


El óptimo podría ser 3 ó 4, ya que es donde se observa el codo.



DBSCAN:
La implementación se lográ mediante el siguiente código:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN

# Cargar los datos desde el archivo CSV (ajusta el nombre del archivo según tu caso)
data = pd.read_csv("Ventas_Bebidas_Demo.csv")

# Crear un diccionario para mapear MARCA y PRODUCTO a valores numéricos
marca_mapping = {marca: i for i, marca in enumerate(data["MARCA"].unique())}
producto_mapping = {producto: i for i, producto in enumerate(data["PRODUCTO"].unique())}

# Reemplazar los valores categóricos con los valores numéricos
data["MARCA"] = data["MARCA"].map(marca_mapping)
data["PRODUCTO"] = data["PRODUCTO"].map(producto_mapping)

# Seleccionar las variables relevantes (MARCA, PRODUCTO, TIENDA)
selected_data = data[["MARCA", "PRODUCTO", "TIENDA"]]

# Aplicar DBSCAN con parámetros eps=0.5 y min_samples=5
dbscan = DBSCAN(eps=0.5, min_samples=5)
selected_data['cluster'] = dbscan.fit_predict(selected_data)

# Visualizar la distribución de puntos
plt.figure(figsize=(8, 6))
plt.scatter(selected_data["MARCA"], selected_data["PRODUCTO"], c=selected_data["cluster"], cmap='viridis')
plt.xlabel('MARCA')
plt.ylabel('PRODUCTO')
plt.title('Distribución de puntos con DBSCAN')
plt.colorbar(label='Cluster')
plt.show()

```



El resultado es similar a DBSCAN, por ello se considera correcto, que signa las mismas tendencias en ambos algoritmos.

Algoritmo de Agrupamiento Jerárquico (Hierarchical Agglomerative Clustering - HAC)

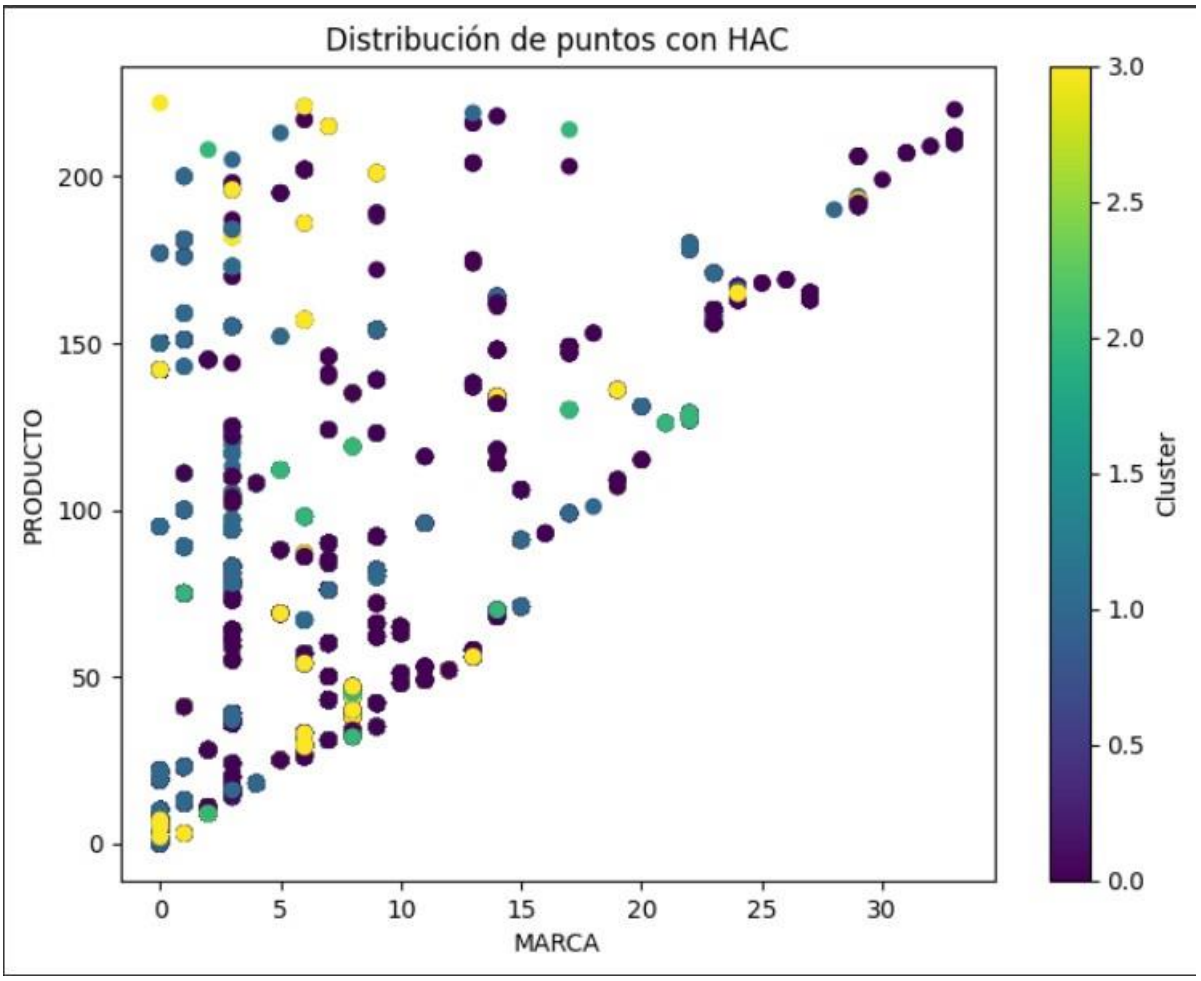
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import AgglomerativeClustering

# Cargar los datos desde el archivo CSV (ajusta el nombre del archivo según tu caso)
data = pd.read_csv("Ventas_Bebidas_Demo.csv")

# Crear un diccionario para mapear MARCA y PRODUCTO a valores numéricos
marca_mapping = {marca: i for i, marca in enumerate(data["MARCA"].unique())}
producto_mapping = {producto: i for i, producto in enumerate(data["PRODUCTO"].unique())}

# Reemplazar los valores categóricos con los valores numéricos
data["MARCA"] = data["MARCA"].map(marca_mapping)
data["PRODUCTO"] = data["PRODUCTO"].map(producto_mapping)

# Seleccionar las variables relevantes (MARCA, PRODUCTO, TIENDA)
selected_data = data[["MARCA", "PRODUCTO", "TIENDA"]]

# Aplicar HAC con parámetros n_clusters=4 y linkage='ward'
hac = AgglomerativeClustering(n_clusters=4, linkage='ward')
selected_data['cluster'] = hac.fit_predict(selected_data)

```

Muestra un poco más de dispersión este último algoritmo, sin embargo, se mantiene la misma tendencia, esto fue retomado con la implementación de la biblioteca, por ello lo único que se tuvo que arreglar, fue arreglar los datos para que funcionen los algoritmos, en este caso funciona igual que K-means. Las transformaciones realizadas están vinculadas a conversión numérica de Marca y Producto haciendo un mapeo, para todos los algoritmos era necesario de otra manera se ocasionaba un error.

Limpiezas realizadas, ninguna, transformaciones realzadas:

- Mapeo Marca a número entero.
- Producto mapeado a número entero.

```python
# Visualizar la distribución de puntos
plt.figure(figsize=(8, 6))
plt.scatter(selected_data["MARCA"], selected_data["PRODUCTO"],
c=selected_data["cluster"], cmap='viridis')
plt.xlabel('MARCA')
plt.ylabel('PRODUCTO')
plt.title('Distribución de puntos con HAC')
plt.colorbar(label='Cluster')
plt.show()

```



El gráfico sigue la tendencia de los anteriores gráficas usando algoritmos ya conocidos, por lo cual este también se considera que esta correcto, la diferencia notable de este algoritmo es que no se le tiene que especificar el valor de K, ya que no lo necesita.

Conclusiones

- **Barrera Peña Víctor Miguel:** Se cumplió las actividades encomendadas, no se tuvo algún error al realizarla, las comprobaciones se lograron al determinar que mediante las implementaciones de los algoritmos el resultado era idéntico, sólo se tuvo que investigar cuando se obtenía alguna clase de error, la búsqueda de
- **Torrecilla Jimenez Aaron Israel:** Las actividades realizadas en el proyecto de Minería de Datos cumplen con los objetivos establecidos. Se implementaron con éxito los algoritmos de K-means y DBScan, tanto en RapidMiner como en Python, con resultados consistentes entre ambos enfoques. La determinación del número óptimo de clusters se realizó mediante el método del codo, sugiriendo 3 o 4 clusters para el conjunto de datos. Se compararon los resultados de DBSCAN y K-means, encontrando similitudes en las tendencias identificadas. Se exploró también el algoritmo de Agrupamiento Jerárquico (HAC), observando resultados similares a K-means, aunque con una ligera dispersión mayor. Se llevaron a cabo transformaciones de datos, como el mapeo de marcas y productos a números enteros, para garantizar la compatibilidad con los algoritmos utilizados. El proyecto proporciona una sólida base para tomar decisiones en la distribución y venta de bebidas por parte de la empresa.
- **Beristain Carbajal Armando:** Mediante el caso practico de la venta de bebidas de un conjunto de tiendas logramos aplicar con claridad los diferentes algoritmos abordados en las horas teóricas , mediante el uso de K-Means y DBScan segmentamos el tipo de tiendas con ciertas necesidades y demandas deacuerdo a sus condiciones y características donde comercian , tras explorar los algoritmos en Python y Rapidminer , llegamos a las mismas conclusiones y resultados