

Reporte de Selección de modelos y parámetros

Robert Lopez, Diego Monroy, Cesar Porras, Laura Becerra

✓ Cargue información y paquetes necesarios

```
#!pip install pandasql
import seaborn as sns
import pandas as pd
import numpy as np
from pandasql import sqldf
run_query = lambda query: sqldf(query, globals())
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import pairwise_distances_argmin_min, silhouette_score
from sklearn.mixture import GaussianMixture
#pip install scikit-learn-extra
from sklearn_extra.cluster import KMedoids
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import AgglomerativeClustering
import warnings
from tabulate import tabulate
warnings.filterwarnings("ignore")
from sklearn.cluster import DBSCAN
from sklearn import metrics

base_compras = pd.read_csv("/content/sample_data/BAQ_compras.csv")
sipsa = pd.read_csv("/content/sample_data/sipsa.csv")
base_productos = pd.read_csv("/content/sample_data/Products_BAQ.csv")
```

Planteamiento requerimientos de negocio

El entregable propuesto a Frubana consta de dos elementos:

- Un elemento de visualización y caracterización de cada uno de los proveedores de acuerdo a la información disponible
- Un elemento de agrupamiento y clasificación de proveedores de acuerdo a la información disponible

Para el elemento de visualización y caracterización, resulta conveniente de hacer uso de métodos descriptivos en dos escalas: Una escala absoluta y una escala temporal. La estimación de promedios, portafolios de productos y acumulado de pedidos permite realizar una caracterización de las capacidades productivas del proveedor, mientras que una evaluación a nivel temporal permite evidenciar la evolución de indicadores estratégicos que evalúen la relación comercial con los distintos proveedores.

Las métricas seleccionadas para evaluar con un corte transversal son el volumen de pedidos atendidos, el volumen de productos ofrecidos, el volumen de productos con perfiles de descomposición distintos (importante para análisis de temas de logística de almacenamiento), así como el pedido máximo y el pedido mínimo histórico que puede suplir dicho proveedor.

La métrica seleccionada para evidenciar la evolución histórica de la relación con el proveedor, así como el impacto económico de la selección de proveedores, es la diferencia porcentual del precio de compra por unidad con el precio por unidad reportado por el SIPSA (Sistema de Información de Precios y Abastecimiento del Sector Agropecuario) Dicha información se construyó realizando una estandarización de los nombres de producto y los nombres reportados en las bases de Frubana, y posteriormente realizando un cruce por fecha de compra y nombre entre ambas bases. Para esta métrica, cabe destacar que para ciertos productos nativos de las regiones (Como por ejemplo el níspero o el agraz), no se encuentra información en el SIPSA, por lo que se estimó una diferencia de 0 entre el valor de mercado y el valor del proveedor. Sin embargo, para el desarrollo final se propone una búsqueda más completa en otras bases de precios con el fin de obtener la información completa

En el caso de las métricas de agrupamiento y clasificación, al no existir una clasificación en Frubana, se hará uso de modelos predictivos de tipo clustering, con el fin de observar agrupaciones entre los datos por similitud, así como distinguir las principales características que definen a estos grupos en función de la información existente dentro de las bases de datos. Ya que corresponde a una primer aproximación de los grupos, se realizará con un corte transversal, y en el caso de la métrica de diferencia porcentual de precios se hará un promedio del corte histórico hasta la fecha. En este caso puntual debe tenerse cuidado con el análisis de dicho valor para los productos de los que no se posee información al interior del SIPSA, ya que puede sugerirse seguir con un proveedor por desinformación mas no por conveniencia para Frubana

✓ Alternativa para el problema descriptivo, visualización y caracterización de proveedores con la información disponible.

Para solucionar el problema descriptivo se propone un proyecto de BI (Business Intelligence) pequeño, el cual contiene los siguientes componentes:

- **Fuentes de información disponibles Frubana:** Archivos csv de Compras, Productos y del Tiempo promedio de vida.
- **Fuentes de información complementaria:** Archivos csv DANE, y archivos planos del SIPSA (Sistema de Información de Precios y Abastecimiento del Sector Agropecuario).
- **Herramienta de ETL:** Se realiza un proceso de extracción, transformación y carga con el fin de limpiar, transformar e integrar los archivos fuente en una bodega de datos.
- **Bodega de datos:** Con base en el diseño de un modelo relacional tipo estrella, se materializan las tablas que conforman el modelo relacional con el fin de tener la información centralizada y que sean base para cálculos y métricas.

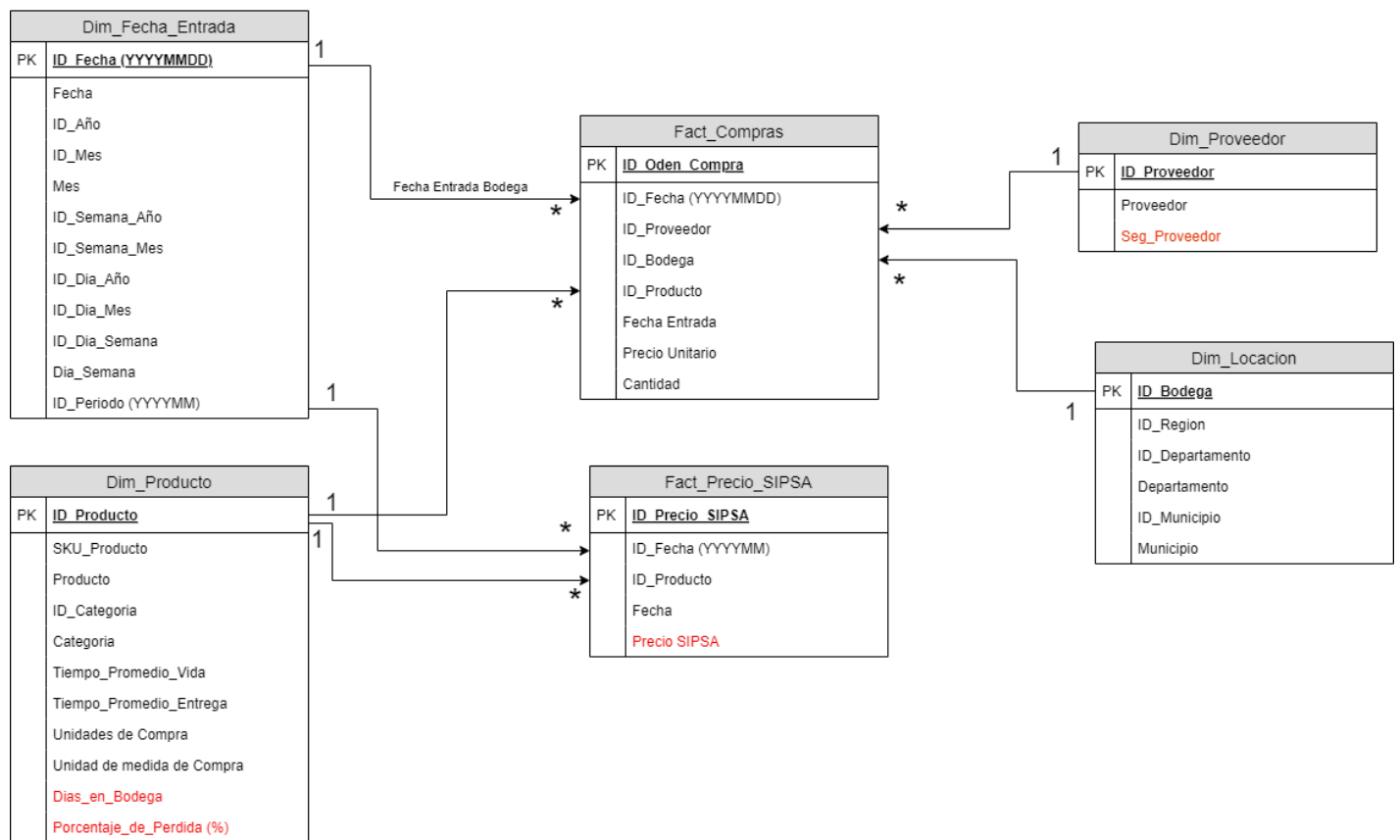
- **Reporte Inteligente:** Se generan generar cálculos, medidas, visualizaciones, comparaciones y datos descriptivos en un reporte inteligente que permite al usuario interactuar con la información.

La implementación de un proyecto de Business Intelligence (BI) conlleva una serie de ventajas significativas para Frubana. En primer lugar, proporciona una visión integral y en tiempo real de los datos empresariales, lo que permite una toma de decisiones más informada y estratégica. Al centralizar y organizar la información de diversas fuentes, el BI facilita la identificación de tendencias, patrones y oportunidades de mejora, lo que puede resultar en una mayor eficiencia operativa y una ventaja competitiva en el mercado. Además, al generar reportes inteligentes y análisis detallados, el BI ayuda a optimizar los procesos internos, a identificar áreas de inversión rentable.

✓ Modelo Relacional de Compras Fruba:

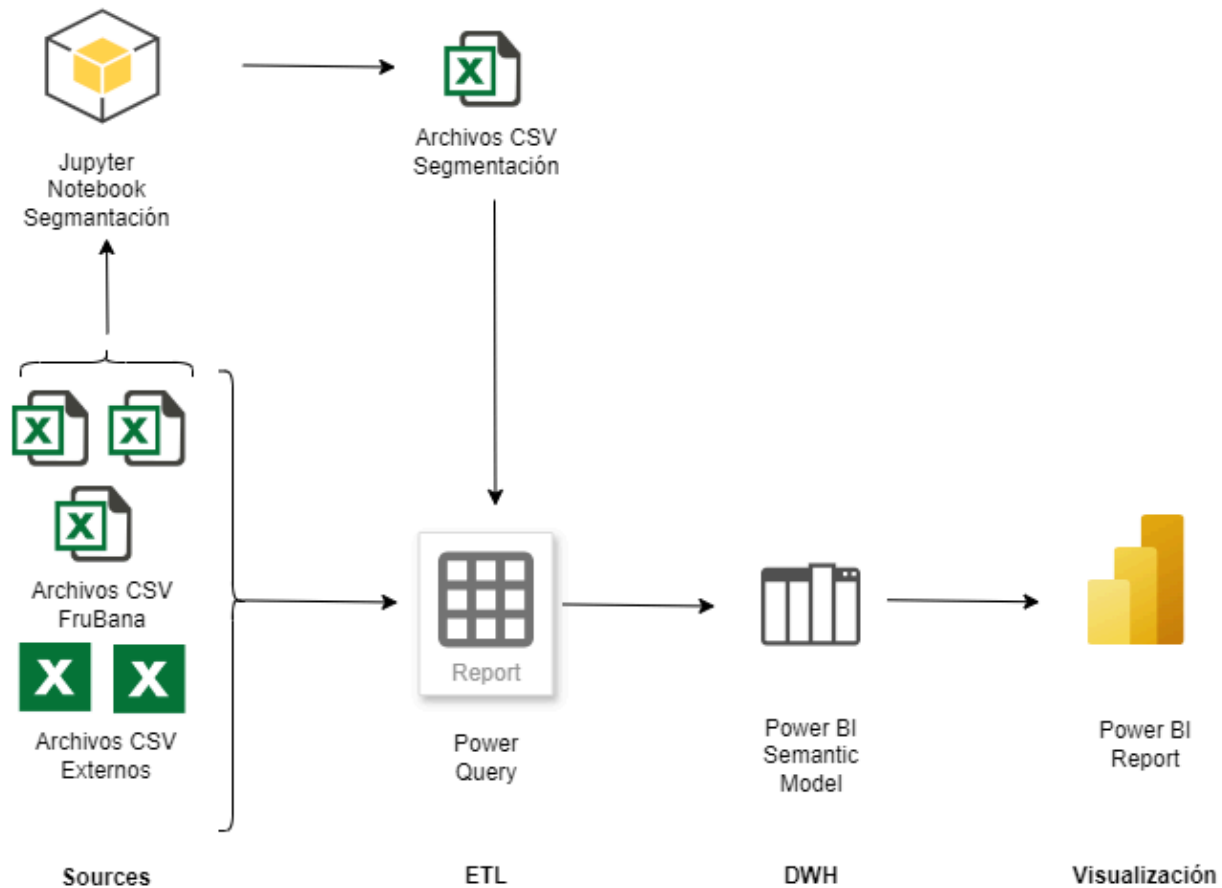
El modelo relacional de Compras de Frubana contiene una estructura organizada de datos que representa las relaciones entre diferentes entidades dentro de la empresa, como proveedores, productos, Georreferencia, Tiempo y transacciones del negocio como compras, además fue complementado con información del DANE para ayudar a describir la locación y con los precios del SIPS para poder generar las métricas de comparación de precios.

Se diseño el siguiente modelo:

MODELO RELACIONAL COMPRAS FRUBANA

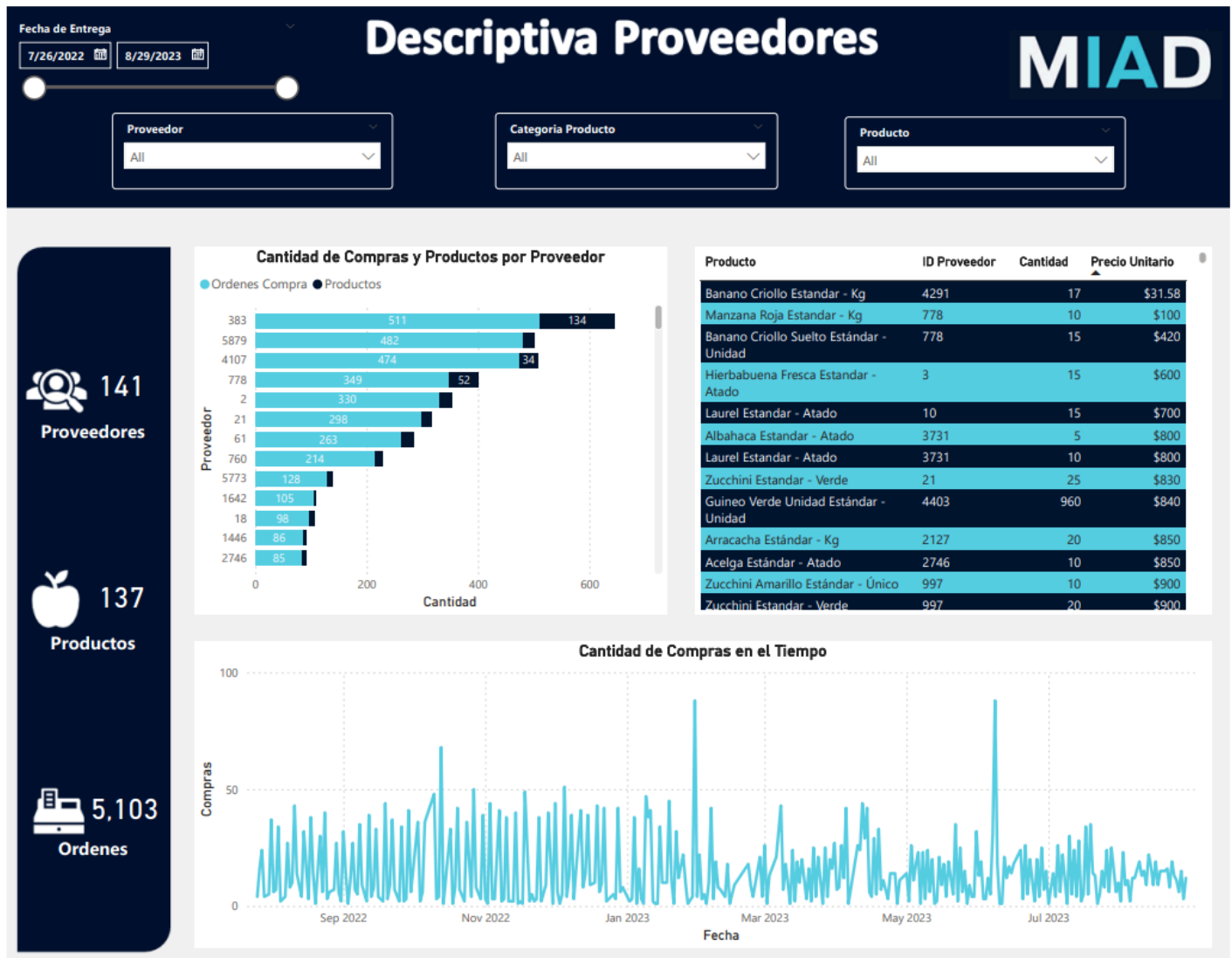
Las pruebas para de conexión a las fuentes de datos, limpieza y transformación por medio de ETL, implementación del modelo relacional y visualización se han realizado por el momento con base en la siguiente arquitectura de BI:

ARQUITECTURA PRUEBAS



Para la parte de pruebas se conecta Power BI Desktop directamente a las fuentes de Frubana, fuentes externas y al archivo csv resultante después de aplicar los modelos de segmentación utilizando Python. En este caso se utiliza Power Query como herramienta de ETL, Modelos semánticos de Power BI como herramienta de DWH o bodega, y Power BI Service como herramienta de visualización, como se puede observar en el diseño por el momento nada sale de la suite de Power BI.

Siguiendo esta arquitectura se obtuvieron las primeras visualizaciones observadas en el MockUp presentado.



Alternativas para el problema de clasificación y agrupamiento.

Para solucionar el problema de clasificación y agrupamiento, se realizará una evaluación de los métodos más comunes de clustering:

- K-medias
- Clustering jerárquico
- DBScan
- Gaussian Mixture Model (GMM)
- K-medioides

Estos métodos cuentan con la ventaja de ser de fácil implementación, además de permitir una caracterización inicial de los proveedores, el cual es el objetivo de esta primer etapa. Como desventaja, necesitan una recalibración frecuente tanto con la entrada de nuevos proveedores como con la actualización de información histórica de los proveedores actuales. Sin embargo, con esta primer aproximación se puede generar un grupo de categorías y características fijas, y posteriormente utilizar

métodos más robustos de clasificación (Como redes neuronales), que permitan clasificar con mayor detalle cada uno de los métodos planteados

También cabe destacar que en esta primera etapa, al ser una base de datos de un tamaño pequeño (141 proveedores con 6 características), no se considera necesario realizar una reducción de dimensionalidad de las características. En caso de en un futuro agregarse más características, se recomienda primero realizar un algoritmo de reducción de dimensionalidad como PCA y luego realizar dicho agrupamiento, aunque esto puede reducir la interpretabilidad de los factores que contribuyen al agrupamiento buscado

✓ Verificación de supuestos y preparación de datos para algoritmos de clustering

Los algoritmos de clustering no requieren una preparación rigurosa de la base de datos, en este caso sólo requieren que se utilicen variables numéricas (para el cálculo de distancias), así como que no se cuente con información faltante) A continuación se presenta la base de datos de los 140 proveedores disponibles, junto con las variables utilizadas para la clasificación. Como puede observarse, no se cuentan con datos faltantes o fuera de formato, por lo que puede procederse con la clasificación

```
query_1 = """
SELECT a.*,b.Diff as Prom_variacion_sipsa
from (SELECT bc.supplier_id as Proveedor,
count(distinct bc.id) as Pedidos_atendidos,
count(distinct bc.name) as Volumen_portafolio,
count(distinct bp.category) as Volumen_category_portafolio,
min(bc.quantity) as Minimo_pedido,
max(bc.quantity) as Maximo_pedido
from base_compras bc
left join base_productos bp on bc.product_id = bp.product_id
group by supplier_id) a
left join sipsa b on (a.Proveedor = b.Id)
;
"""
result_1 = run_query(query_1)
print(result_1)
```

	Proveedor	Pedidos_atendidos	Volumen_portafolio	\
0	2	330	23	
1	3	2	2	
2	9	34	4	
3	10	30	17	
4	18	98	9	
..	
136	6208	2	1	
137	6220	4	1	
138	6234	4	2	
139	6237	2	2	
140	6273	1	1	

	Volumen_category_portafolio	Minimo_pedido	Maximo_pedido	\
0	2	1.0	100.0	
1	1	15.0	20.0	
2	1	1.0	160.0	
3	1	1.0	159.0	
4	1	5.0	742.0	
..	
136	1	200.0	500.0	
137	1	760.0	2140.0	
138	2	200.0	1000.0	
139	1	12.0	43.0	
140	1	415.0	415.0	

	Prom_variacion_sipsa
0	-0.080710
1	0.000000
2	0.027052
3	-0.103380
4	0.122707
..	...
136	0.000000
137	-0.164336
138	-0.043578
139	-0.113687
140	0.000000

[141 rows x 7 columns]

✓ Calibración y selección de modelos

Para cada uno de los modelos, se realizó una evaluación en grid, la cual permite modificar los hiperparámetros del modelo para optimizar la métrica de rendimiento.

Para evaluar la calidad de los clusters obtenidos, se utilizará como criterio el coeficiente de Silhouette, el cual permite ponderar la distancia de los elementos dentro del cluster con la distancia de los elementos al cluster más próximo, dando una medida tanto de clasificación correcta como de superposición de los clusters.

En este caso particular como corresponde a una clasificación inicial de todos los proveedores, se realizará la evaluación sobre toda la muestra.

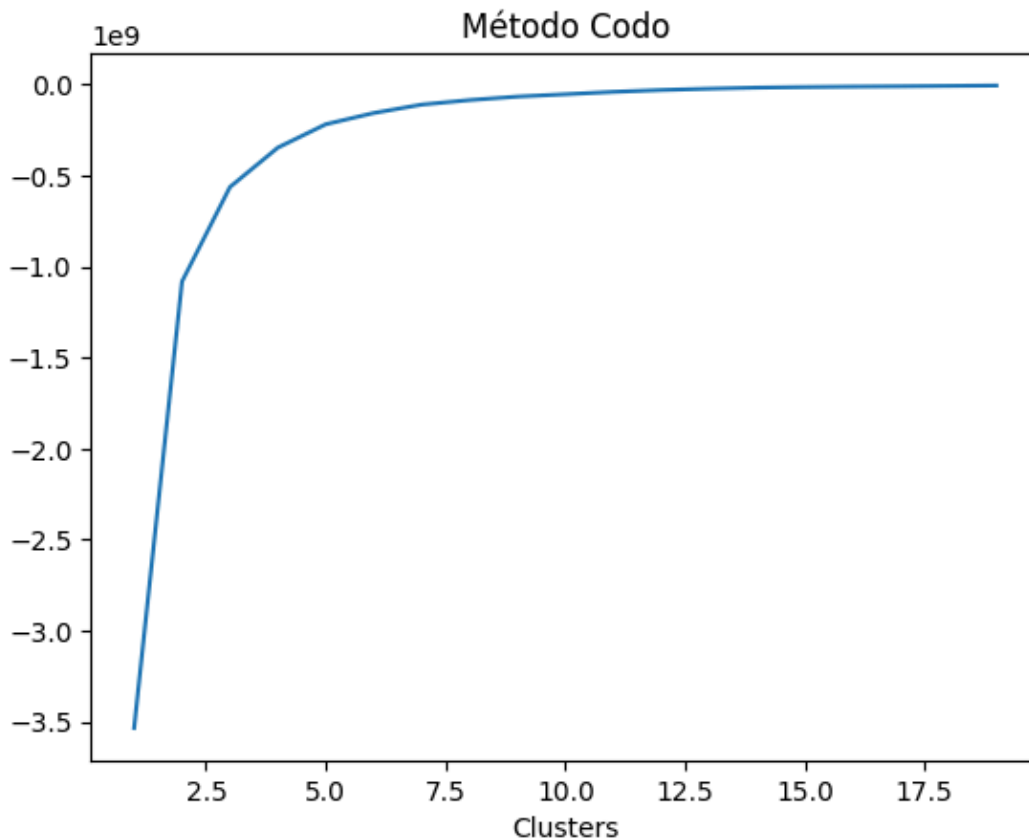
Finalmente, adicional al criterio del coeficiente de Silhouette, se realizará una evaluación de viabilidad técnica y conceptual, para determinar si cada modelo posee particularidades específicas que lo hacen más conveniente para el problema a solucionar, así posea un desempeño en clustering ligeramente inferior

Haz doble clic (o pulsa Intro) para editar

▼ K-means

En el algoritmo de k-medias, para calibrar el número óptimo de clusters, se usa el método del codo gráfico. En este caso, se observa una inflexión cercana a los 5 clusters, por lo que este será el valor utilizado para compararlo frente a los demás métodos

```
X = result_1.drop(columns = ['Proveedor'])
Nc = range(1, 20)
kmeans = [KMeans(n_clusters=i) for i in Nc]
score = [kmeans[i].fit(X).score(X) for i in range(len(kmeans))]
plt.plot(Nc,score)
plt.xlabel('Clusters')
plt.title('Método Codo')
plt.show()
```



```
kmeans = KMeans(n_clusters=5).fit(X)
sil_kmeans = silhouette_score(X, kmeans.fit_predict(X))
```

▼ Jerárquico

En el caso del algoritmo jerárquico, se evalúan todos los posibles métodos de agrupamiento (Sencillo, completo, promedio, Ward), y se selecciona el que posea un mayor desempeño. Los resultados de cada uno de los modelos se observan a continuación

```
linkage_data1 = linkage(X, method='single', metric='euclidean')
linkage_data2 = linkage(X, method='complete', metric='euclidean')
linkage_data3 = linkage(X, method='average', metric='euclidean')
linkage_data4 = linkage(X, method='ward', metric='euclidean')

hierarchical_cluster1 = AgglomerativeClustering(n_clusters=2, affinity='euclidean', link
hierarchical_cluster2 = AgglomerativeClustering(n_clusters=2, affinity='euclidean', link
hierarchical_cluster3 = AgglomerativeClustering(n_clusters=2, affinity='euclidean', link
hierarchical_cluster4 = AgglomerativeClustering(n_clusters=2, affinity='euclidean', link

sil_1 = silhouette_score(X, hierarchical_cluster1.fit_predict(X))
sil_2 = silhouette_score(X, hierarchical_cluster2.fit_predict(X))
sil_3 = silhouette_score(X, hierarchical_cluster3.fit_predict(X))
sil_4 = silhouette_score(X, hierarchical_cluster4.fit_predict(X))

data = [['Single', sil_1, 2],
['Complete', sil_2, 2],
['Average', sil_3, 2],
['Ward', sil_4, 2]]

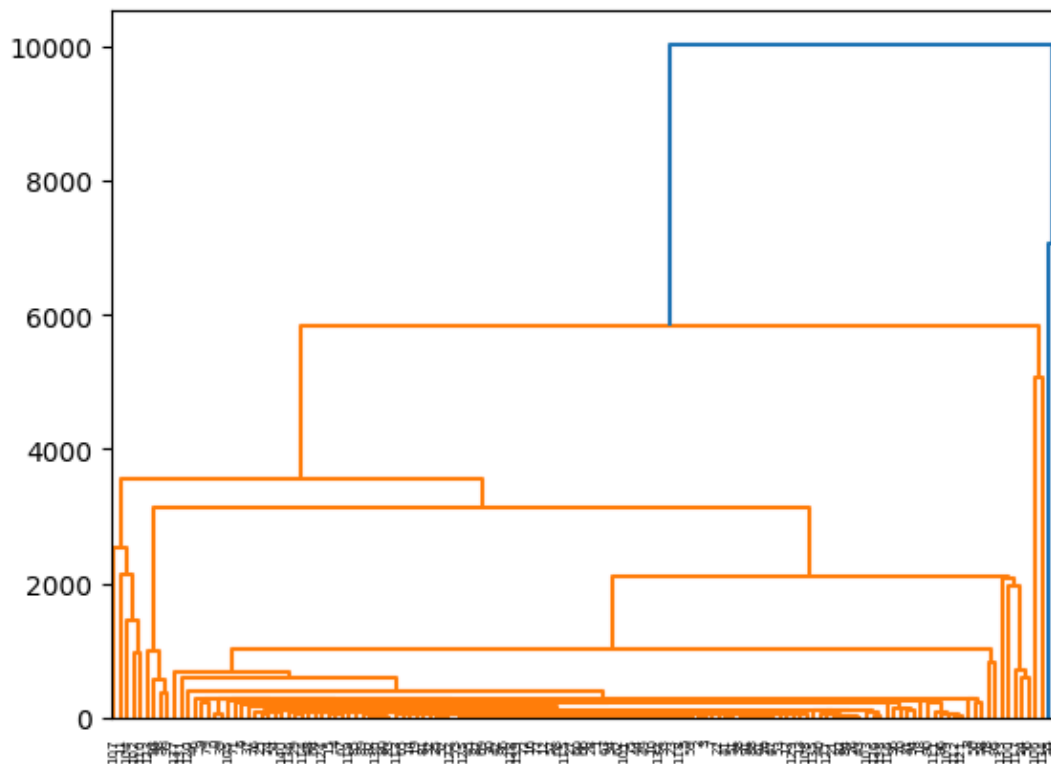
print (tabulate(data, headers=["Método", "Coef. Silhouette", "Clusters"]))
```

Método	Coef. Silhouette	Clusters
Single	0.862228	2
Complete	0.84615	2
Average	0.862228	2
Ward	0.840099	2

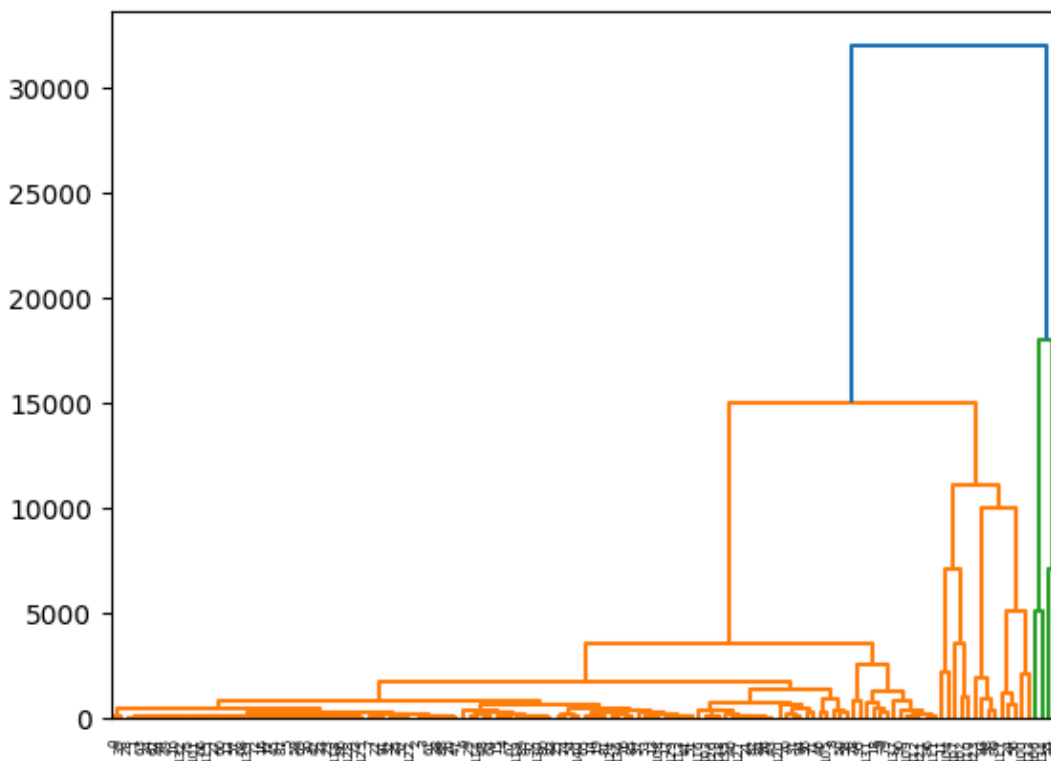
Puede observarse que el mejor rendimiento lo poseen tanto el método single como el método promedio. Sin embargo, revisando gráficamente cada uno de los dendogramas, se observa lo siguiente en la siguiente figura

```
print('Simple')
dendrogram(linkage_data1)
plt.show()
print('Complete')
dendrogram(linkage_data2)
plt.show()
print('Average')
dendrogram(linkage_data3)
plt.show()
print('Ward')
dendrogram(linkage_data4)
plt.show()
```

Simple



Complete



Average



Puede observarse que 3 de los 4 métodos evaluados crean un cluster de apenas dos o tres proveedores y agrupa a los demás en una sola categoría. Esta forma de clasificación no resulta apropiada a largo plazo, por lo que el algoritmo a utilizar, en caso de seleccionarse el método jerárquico, es el algoritmo de Ward

▼ DBScan

Para realizar la parametrización del DBScan primero se realizó un análisis exploratorio para encontrar las configuraciones de epsilon y mínimo de muestras que no clasificaran a todos los datos como outliers. Sobre dicha parametrización, se corrió un grid con las combinaciones posibles que maximizaran el coeficiente de Silhouette. Los resultados se presentan a continuación

```

epsilon = [1,2,3,4,5,6,7,8,9,10]
min_samples = [1,2,3]

sil_avg = []
max_value = [0,0,0,0]

for i in range(len(epsilon)):
    for j in range(len(min_samples)):

        db = DBSCAN(min_samples = min_samples[j], eps =epsilon[i]).fit(X)
        #cluster_labels=dbscan.fit_predict(data)
        core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
        core_samples_mask[db.core_sample_indices_] = True
        labels = db.labels_

        # Number of clusters in labels, ignoring noise if present.
        n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
        n_noise_ = list(labels).count(-1)

        silhouette_avg = metrics.silhouette_score(X, labels)
        if silhouette_avg > max_value[3]:
            max_value=(epsilon[i], min_samples[j], n_clusters_, silhouette_avg)
        sil_avg.append(silhouette_avg)

print("epsilon=", max_value[0],
      "\nmin_sample=", max_value[1],
      "\nnumber of clusters=", max_value[2],
      "\naverage silhouette score= %.4f" % max_value[3])

epsilon= 8
min_sample= 1
number of clusters= 120
average silhouette score= 0.1506

```

✓ GMM (Gaussian Mixture Model)

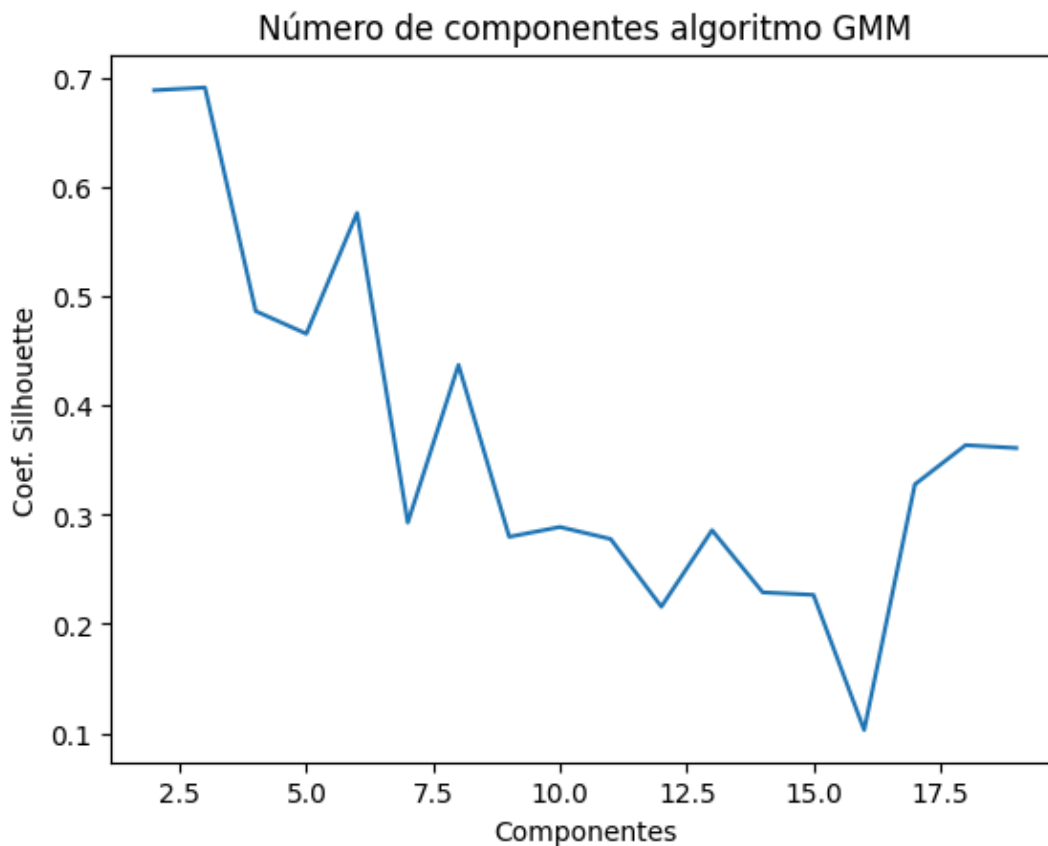
Para el modelo GMM, se parametrizó el número de componentes y se seleccionó la combinación que maximizara el coeficiente de Silhouette. Los resultados se muestran a continuación

```
componentes = range(2,20)
sil_GMM = []
for i in componentes:
    gaussian_model = GaussianMixture(n_components=i)
    gaussian_model.fit(X)
    gaussian_result = gaussian_model.predict(X)
    gaussian_clusters = np.unique(gaussian_result)
    sil_GMM.append(silhouette_score(X, gaussian_model.predict(X)))
plt.plot(componentes,sil_GMM)
plt.ylabel('Coef. Silhouette')
plt.xlabel('Componentes')
plt.title('Número de componentes algoritmo GMM')
plt.show()
```

```
# define the model
gaussian_modelF = GaussianMixture(n_components=3)

# train the model
gaussian_modelF.fit(X)

# assign each data point to a cluster
gaussian_resultF = gaussian_modelF.predict(X)
gaussian_clustersF = np.unique(gaussian_resultF)
sil_Gau = silhouette_score(X, gaussian_modelF.predict(X))
```



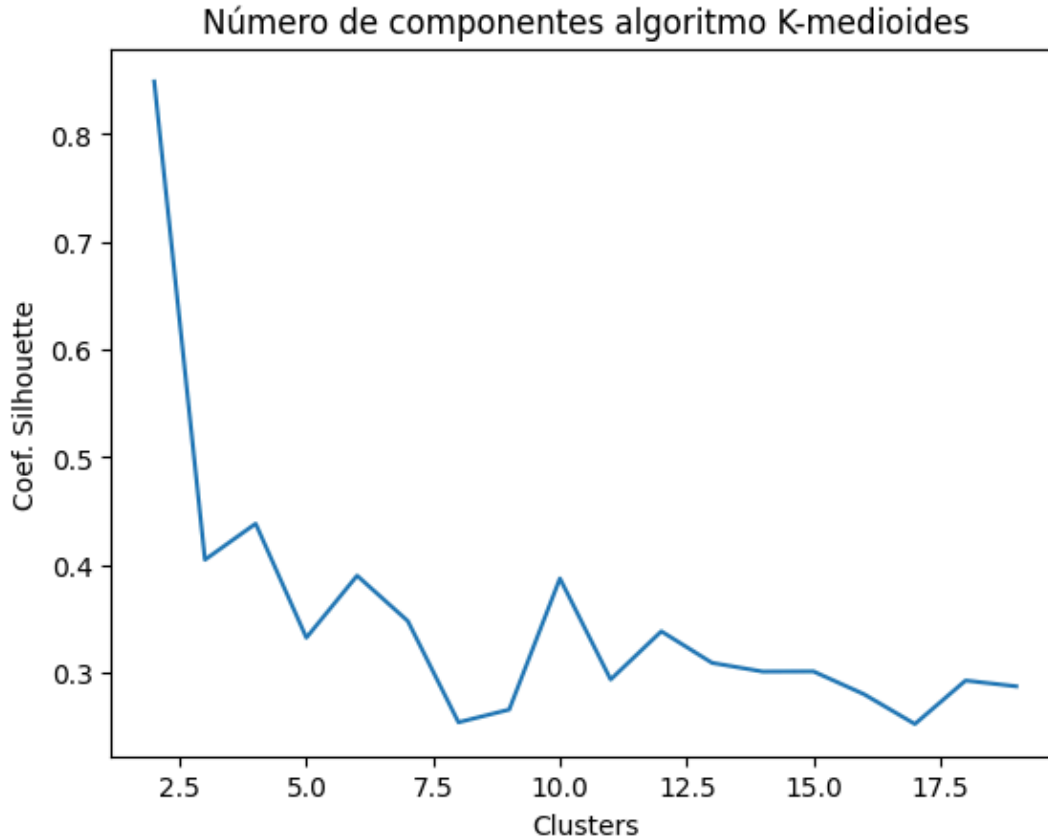
Puede observarse que el mayor valor del coeficiente de Silhouette se encuentra con tres componentes, y que el número de clusters óptimo para el método GMM es de 3

✓ K-Medoids

Finalmente, para K-medioides se realiza un proceso similar al de GMM, iterando el número de clusters y revisando el valor que optimice el coeficiente de Silhouette

```
componentes = range(2,20)
sil_KMed = []
for i in componentes:
    kmedoids = KMedoids(n_clusters=i, random_state=0).fit(X)
    sil_KMed.append(silhouette_score(X, kmedoids.predict(X)))
plt.plot(componentes,sil_KMed)
plt.ylabel('Coef. Silhouette')
plt.xlabel('Clusters')
plt.title('Número de componentes algoritmo K-medioides')
plt.show()
```

```
kmedoidsF = KMedoids(n_clusters=2, random_state=0).fit(X)
sil_KMedF = silhouette_score(X, kmedoidsF.predict(X))
```



Puede observarse que el valor de clusters que maximiza el coeficiente de Silhouette es de 2 coeficientes

✓ Análisis de alternativas y ajustes requeridos

La siguiente tabla muestra el resumen del coeficiente de Silhouette para cada una de las metodologías obtenidas, así como el número de clusters

```
data = [['K-means', sil_kmeans, 5],
        ['Jerárquico (Ward)', sil_4, 2],
        ['DBScan', max_value[3], 120],
        ['GMM', sil_Gau, 3],
        ['K-Medoids', sil_KMedF, 2]]

print (tabulate(data, headers=["Método", "Coef. Silhouette", "Clusters"]))
```

Método	Coef. Silhouette	Clusters
K-means	0.862392	5
Jerárquico (Ward)	0.840099	2
DBScan	0.150598	120
GMM	0.443299	3
K-Medoids	0.84893	2

Puede observarse que el método menos conveniente para la creación de grupos es DBScan, ya que presenta alta superposición de clusters y suele clasificar a múltiples proveedores como su propio grupo. De los demás métodos, el método que posee tanto el mayor coeficiente de Silhouette, como una clasificación más diversa de clusters es el método de K-means, por lo que será usado para la clasificación y agrupamiento de proveedores.

Para esta primera etapa, se contempla aumentar la completitud del campo de diferencia de precios frente al SIPSA, ya que la agrupación puede llegar a verse alterada por la presencia de dichos valores de cero estimados. También es importante incluir dentro del prototipo una categorización de cada uno de los clusters en función de tamaño y características, para entender con el usuario final la claridad de la clasificación y la posibilidad de evaluar la adición o remoción de variables.

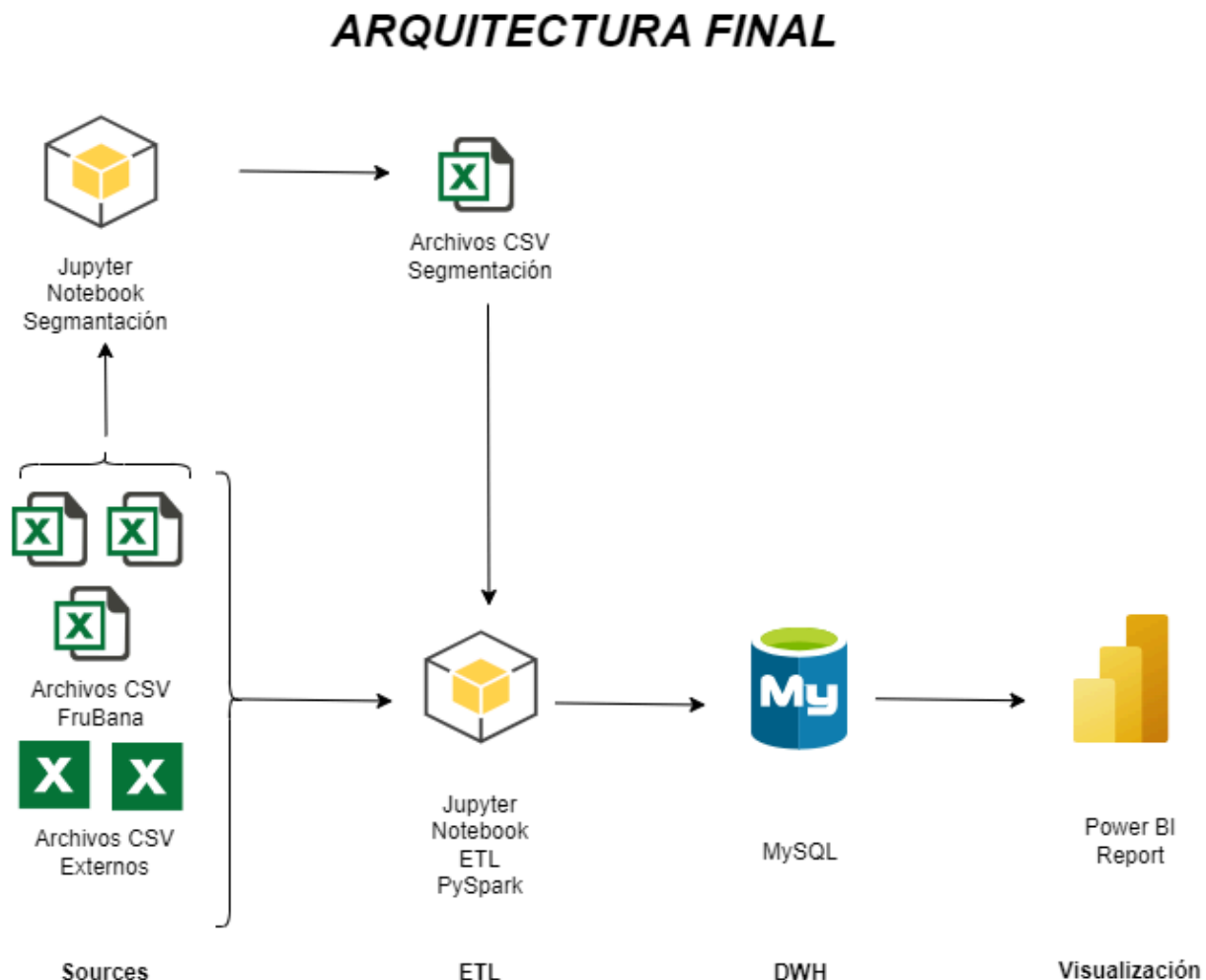
✓ Componentes y artefactos pendientes de implementación

- ✓ Alternativa para el problema descriptivo, visualización y caracterización de proveedores con la información disponible.

En este momento, el modelo relacional está conectado directamente a Power BI. Sin embargo, para la etapa final del proyecto, se planea implementar un proceso de Extracción, Transformación y Carga (ETL) utilizando PySpark en Python por medio de Jupyter Notebooks que llevará los datos de csv archivos planos a un almacén de datos en MySQL donde se materializarán las tablas que componen el modelo relacional. Este almacén contendrá datos limpios y listos para su análisis. Se utilizará este almacén de datos para generar los informes mostrados en el mockup, utilizando Power BI conectado directamente al almacén. Además, para mejorar la calidad de los datos y su capacidad de análisis, se planea realizar otra etapa de ETL utilizando también Python en donde se incorpora la categoría del proveedor resultante del modelo de segmentación en la bodega de datos y se integrará adecuadamente en la dimensión de Proveedor.

Los informes generados incluirán tanto análisis descriptivos como segmentación de datos. Estos informes se publicarán en Power BI Service para que estén disponibles para todos los usuarios relevantes.

Así el proyecto final seguirá la siguiente arquitectura de BI:



Para la parte de pruebas se conecta Power BI Desktop directamente a las fuentes de Frubana, fuentes externas y al archivo csv resultante después de aplicar los modelos de segmentación utilizando Python. En este caso se utiliza Power Query como herramienta de ETL, Modelos semánticos de Power BI como herramienta de DWH o bodega, y Power BI Service como herramienta de visualización, como se puede observar en el diseño por el momento nada sale de la suite de Power BI.

El artefacto final será un reporte de Power BI publicado en Power BI Service lo que se conoce como (Front End) el cual permite al usuario navegar por diferentes gráficos, tablas, KPI, interactuar con los filtros y poder toda la información disponible en un solo lugar, además con la integración de la categorización arrojada por el modelo de segmentación implementado también podrá interactuar con las visualizaciones creadas y priorizar los proveedores de su elección.

Se planea documentar lo realizado en el (Back End) y eso incluye, código de modelos de segmentación, código de limpieza, código de experimentos, código de ETLs, estructura de tabla en bodega de datos, y manual de uso del Tablero de control.

✓ Anexo: Análisis descriptivo de los datos

```
base_compras = pd.read_csv("/content/sample_data/BAQ_compras.csv")
base_compras.head(n=20)
```

	warehouse_code	region_code	id	delivery_date	product_id	sku
0	BAQ	BAQ	8475266	2022-09-08	660	BAQ-FRU1-CAT1-111:277:659:660
1	BAQ	BAQ	8554498	2022-09-12	660	BAQ-FRU1-CAT1-111:277:659:660
2	BAQ	BAQ	8682054	2022-09-19	660	BAQ-FRU1-CAT1-111:277:659:660
3	BAQ	BAQ	8715673	2022-09-22	660	BAQ-FRU1-CAT1-111:277:659:660
4	BAQ	BAQ	8842178	2022-09-29	660	BAQ-FRU1-CAT1-111:277:659:660
5	BAQ	BAQ	8901410	2022-10-03	660	BAQ-FRU1-CAT1-111:277:659:660
6	BAQ	BAQ	8935715	2022-10-06	660	BAQ-FRU1-CAT1-111:277:659:660
7	BAQ	BAQ	9044631	2022-10-10	660	BAQ-FRU1-CAT1-111:277:659:660
8	BAQ	BAQ	9137967	2022-10-13	660	BAQ-FRU1-CAT1-111:277:659:660
9	BAQ	BAQ	9088197	2022-10-13	660	BAQ-FRU1-CAT1-111:277:659:660
10	BAQ	BAQ	9172931	2022-10-17	660	BAQ-FRU1-CAT1-111:277:659:660
11	BAQ	BAQ	9213163	2022-10-20	660	BAQ-FRU1-CAT1-111:277:659:660
12	BAQ	BAQ	9396163	2022-10-31	660	BAQ-FRU1-CAT1-111:277:659:660
13	BAQ	BAQ	9504933	2022-11-07	660	BAQ-FRU1-CAT1-111:277:659:660
14	BAQ	BAQ	9534555	2022-11-10	660	BAQ-FRU1-CAT1-111:277:659:660

15	BAQ	BAQ	9645657	2022-11-18	660	BAQ-FRU1-CAT1-111:277:659:660
16	BAQ	BAQ	9759008	2022-11-24	660	BAQ-FRU1-CAT1-111:277:659:660
17	BAQ	BAQ	9860532	2022-11-28	660	BAQ-FRU1-CAT1-111:277:659:660
18	BAQ	BAQ	9885850	2022-12-01	660	BAQ-FRU1-CAT1-111:277:659:660
19	BAQ	BAQ	9951957	2022-12-05	660	BAQ-FRU1-CAT1-111:277:659:660

Next steps:

[Generate code with base_compras](#)[View recommended plots](#)

La base compras tiene como variables el código de bodega y la región y para este caso solo disponemos de información de una Bodega en Barranquilla, el id de compra, la fecha de entrega, el número, código y nombre del producto, el código del proveedor, el precio y la cantidad. Vemos que los cinco productos que mas se compran en esa bodega son en su orden: papa blanca, cebolla roja, pimentón verde, cebolla blanca, platano y zanahoria.

De la papa blanca el mayor proveedor es 1642 al cual se le han hecho en total 82 pedidos, de la cebolla roja el mayor proveedor es 5879 con 37 pedidos, del pimentón es el 4107 con 60 pedidos, de la cebolla blanca son el 1642 y el con 778 con 21 pedidos cada uno, el de plantano es el 297 con 30 pedidos y el de la zanahoria son el 778 y 4912 con 17 pedidos cada uno.

Analizando por proveedor los tres que más pedidos tienen son el 383 con 511 ordenes, 5879 con 482 ordenes y 4107 con 474 ordenes. También hay 3 proveedores que tienen los mejores precios en gran parte de los productos que son el 383, 778 y el 5879, que coinciden con los que tienen mayor número de ordenes.

```
base_compras.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5103 entries, 0 to 5102
Data columns (total 10 columns):
#   Column          Non-Null Count  Dtype
---  -
0   warehouse_code  5103 non-null   object
1   region_code     5103 non-null   object
2   id              5103 non-null   int64
3   delivery_date   5103 non-null   object
4   product_id      5103 non-null   int64
```

```
5    sku          5103 non-null    object
6    name         5103 non-null    object
7    supplier_id  5103 non-null    int64
8    price        5103 non-null    float64
9    quantity     5103 non-null    float64
dtypes: float64(2), int64(3), object(5)
memory usage: 398.8+ KB
```

base_compras.columns

```
Index(['warehouse_code', 'region_code', 'id', 'delivery_date', 'product_id',
      'sku', 'name', 'supplier_id', 'price', 'quantity'],
      dtype='object')
```

base_compras.describe()

	id	product_id	supplier_id	price	quantity	
count	5.103000e+03	5103.000000	5103.000000	5103.000000	5103.000000	
mean	1.059577e+07	69698.105036	2381.954537	3646.579462	553.558985	
std	1.466670e+06	149176.499020	2355.507744	7806.423940	2152.375026	
min	7.741513e+06	100.000000	2.000000	1.000000	1.000000	
25%	9.385387e+06	177.000000	297.000000	1535.000000	20.000000	
50%	1.070192e+07	1181.000000	997.000000	2500.000000	60.000000	
75%	1.188155e+07	47269.000000	4854.000000	3600.000000	300.000000	
max	1.284395e+07	641052.000000	6273.000000	95000.000000	32000.000000	

```
for column in base_compras.select_dtypes(include=['object', 'category']):
    print(base_compras[column].value_counts())
    print("\n")
```

```
warehouse_code
BAQ      5103
Name: count, dtype: int64
```

```
region_code
BAQ      5103
Name: count, dtype: int64
```

```
delivery_date
2023-06-08    88
2023-01-30    88
2022-10-13    68
2022-12-05    51
2022-10-27    50
..
2023-03-12     1
```

```

2022-11-02      1
2023-03-26      1
2023-01-05      1
2023-04-06      1
Name: count, Length: 338, dtype: int64

```

```

sku
BAQ-FRU1-CAT6-234:304:750:770      190
BAQ-FRU1-CAT2-50:75:170:171        172
BAQ-FRU1-CAT2-51:76:173:174        169
BAQ-FRU1-CAT104105-60271:510131:510132:258690  162
BAQ-FRU1-CAT1-47:67:151:152        144
...
BAQ-FRU1-CAT104105-104967:239539:239540:131193  1
BAQ-FRU1-CAT1-97:431:1111:1112                1
BAQ-FRU1-CAT2-264:351:907:908                  1
BAQ-FRU1-CAT2-31:64:145:1841                    1
BAQ-FRU1-CAT104105-73423:168376:168377:92108    1
Name: count, Length: 152, dtype: int64

```

```

name
Papa Blanca Sucia Tamaño Mixto - KG      190
Cebolla Roja Mixta Mixta - Desde 5kg     172
Pimentón Verde Mixto Estándar - Desde 1Kg  169
Cebolla Cabezona Blanca Sin Pelar Mixta - Desde 1Kg  162
Plátano Hartón Verde Estandar - Desde 2Kg  144
...
Tomate Chonto Verde Mixto - Kg            1
Ciruela Roja Estándar - Kg                1
Arveja Verde Estándar - Kg                1
Pepino Cohombro Estándar - Kg (Tamaño 🏠)  1
Tomate Chonto Extramaduro Mixto - Kg      1
Name: count, Length: 152, dtype: int64



```

```
# Base por producto y supplier
```

```

productos_mas_comprados_por_proveedor = base_compras.groupby(['supplier_id', 'product_id'])
productos_mas_comprados_por_proveedor = productos_mas_comprados_por_proveedor.sort_values(
    by='count', ascending=False)
productos_mas_comprados_por_proveedor.head(20)


```

	supplier_id	product_id	name	count	
418	1642	770	Papa Blanca Sucia Tamaño Mixto - KG	82	
474	4107	174	Pimentón Verde Mixto Estándar - Desde 1Kg	60	
487	4107	863	Yuca Tamaño Mixto - Desde 5kg	58	
101	61	401340	Tomate Chonto Maduración Mixta Semi (Mediano)	45	
			...		
17	2	18446	Jengibre Estándar - Bandeja	44	
668	5879	192	Ahuyama Estándar - Kg - 🥬 (Insuperable)	43	
476	4107	192	Ahuyama Estándar - Kg - 🥬 (Insuperable)	42	
470	4107	146	Pepino Cohombro Estándar - Kg	41	
61	21	868	Brócoli Estándar - Kg	40	
299	617	1181	Ajo Estandar - Caja	39	
665	5879	171	Cebolla Roja Mixta Mixta - Desde 5kg	37	
52	18	63788	Pimentón Rojo Nataly Estándar - Kg	37	
670	5879	196	Guayaba Maduración Mixta - Kg - 🥭 (Insuperable)	37	
102	61	563293	Tomate Chonto Maduración Mixta Estándar (Grand...	37	
334	778	171	Cebolla Roja Mixta Mixta - Desde 5kg	36	
310	760	1609	Lechuga Crespa Verde Estandar - Unidad	36	
649	5773	63788	Pimentón Rojo Nataly Estándar - Kg	35	
10	2	1878	Hierbabuena Fresca Estandar - Atado	35	
674	5879	63795	Berenjena Baby Estándar Kg	34	

Next steps:

Generate code with




productos_mas_comprados_por_proveedor



View recommended plots

```
# Filtros para saber pedidos por supplier
df_filtrado = productos_mas_comprados_por_proveedor[productos_mas_comprados_por_proveedc

# total pedidos por supplier
total_pedidos_por_supplier = productos_mas_comprados_por_proveedor.groupby('supplier_id'
total_pedidos_por_supplier = total_pedidos_por_supplier.sort_values(by='count', ascendir
total_pedidos_por_supplier
```

	supplier_id	count	
24	383	511	
120	5879	482	
79	4107	474	
39	778	349	
0	2	330	
...	
102	5283	1	
104	5296	1	
34	661	1	
35	729	1	
140	6273	1	

141 rows × 2 columns

Next steps:

[Generate code with total_pedidos_por_supplier](#)[View recommended plots](#)

```
base_compras['precio_cantidad'] = base_compras['quantity'] / base_compras['price']
```

```
# Encontrar el precio mínimo para cada combinación de 'product_name' y 'supplier_id'
precio_minimo_por_producto = base_compras.groupby(['name', 'supplier_id'])['precio_canti
```

```
# Encontrar el proveedor asociado al precio mínimo para cada producto
proveedor_menor_precio_por_producto = precio_minimo_por_producto.loc[precio_minimo_por_p
```

```
# Ordenar el resultado por 'price' (precio) de menor a mayor para encontrar el proveedor
proveedor_menor_precio_por_producto = proveedor_menor_precio_por_producto.sort_values(by
```

```
print(proveedor_menor_precio_por_producto)
```

```

           name  supplier_id \
33          Ajo  Estandar – Caja      778
159  Champiñones Estandar – Bandeja 1kg      383
248          Kiwi Estándar – Kg         63
460  Patilla/Sandía Grande – Unidad      383
200  Granadilla Estándar – Kg      383
..          ...          ...
643  Tomate Chonto Maduro  Estándar (Grande) – Kg      3840
388          Papa Blanca Segunda  Estándar – Kg      383
365          Mazorca Estándar – Unidad      658
366  Mazorca con Amero Estándar – Unidad      658
644  Tomate Chonto Maduro  Semi (Mediano) – Kg      4702

precio_cantidad
33          0.000015

```


159	0.000042
248	0.000050
460	0.000056
200	0.000083
..	...
643	0.100000
388	0.117500
365	0.142857
366	0.142857
644	0.452727

[152 rows x 3 columns]

```
# Obtener el conteo de proveedores
```

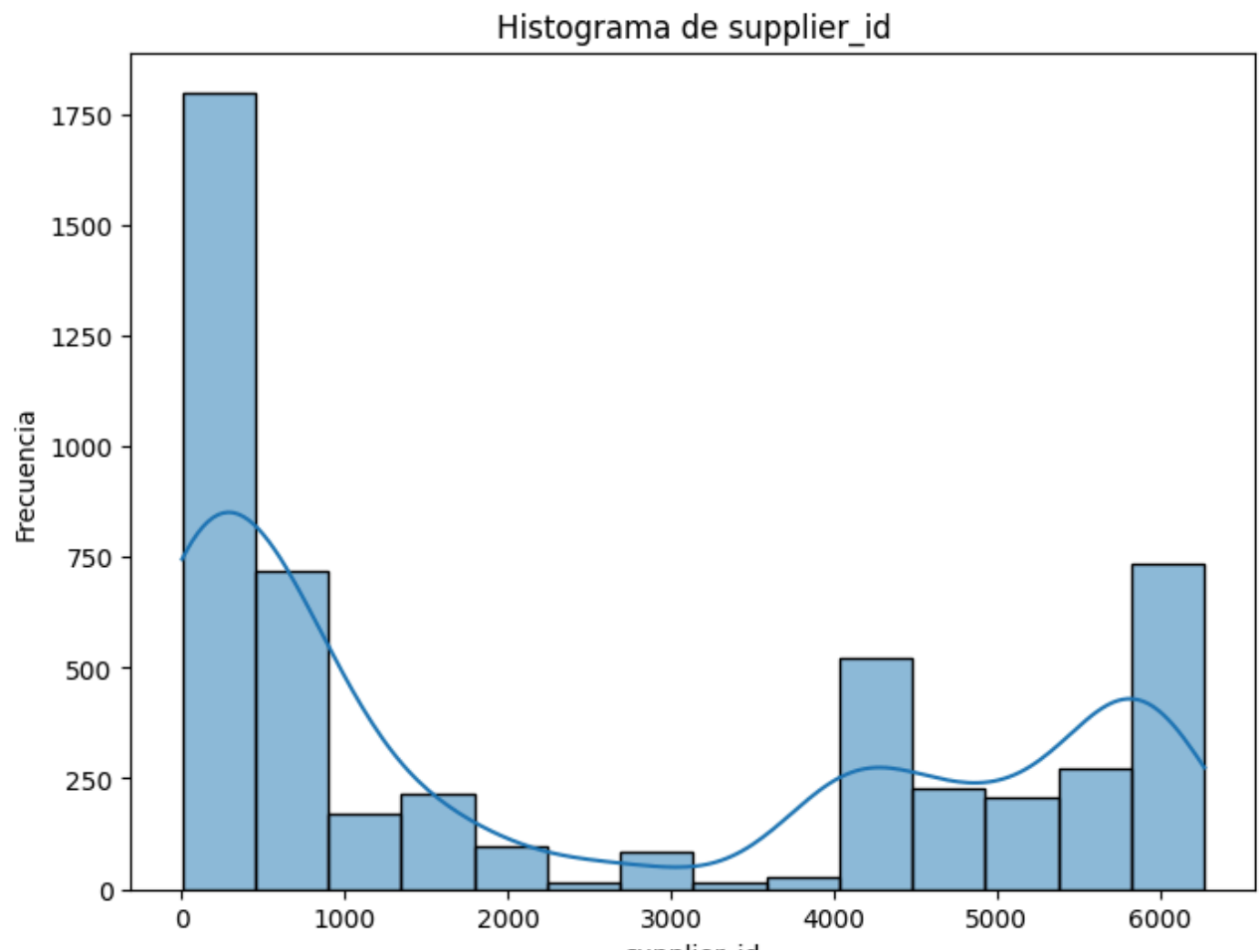
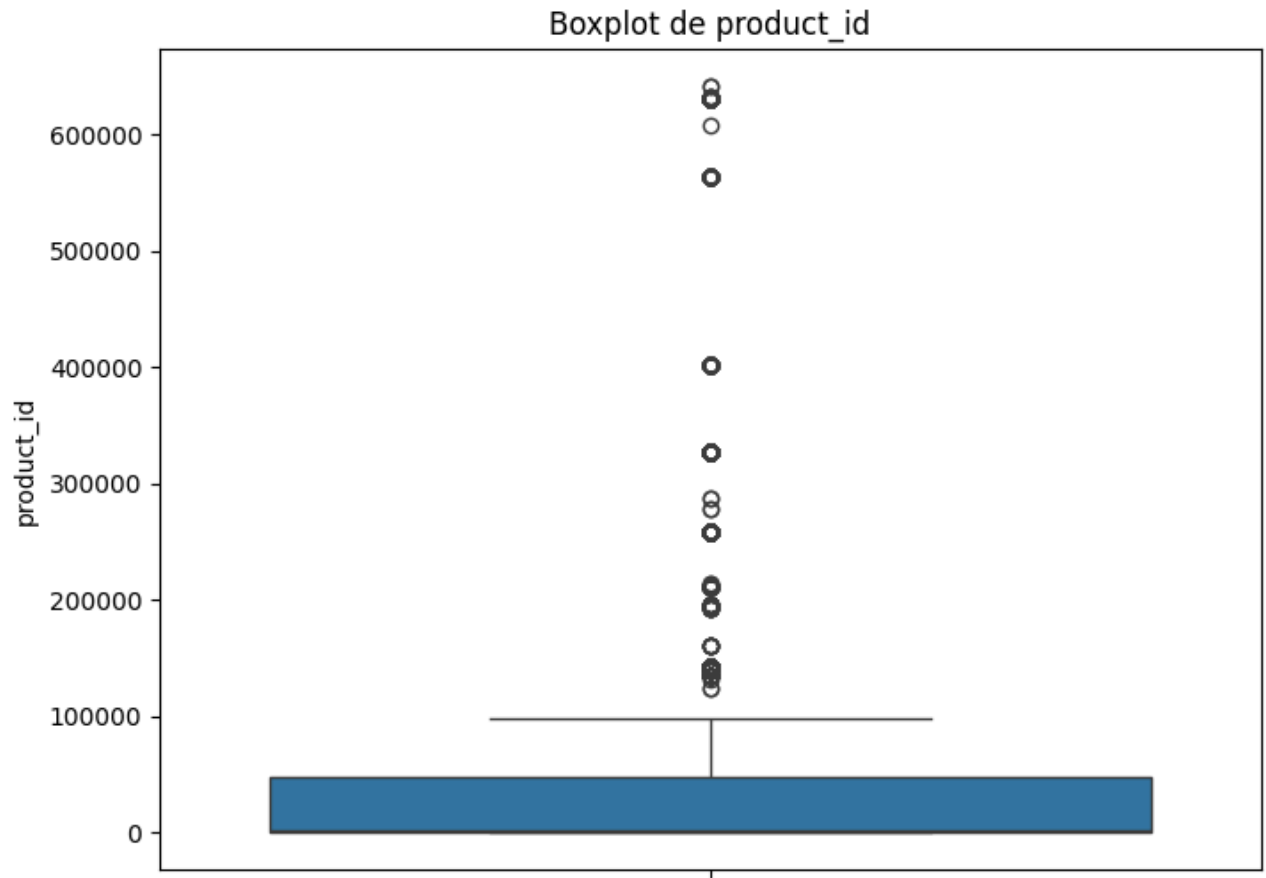
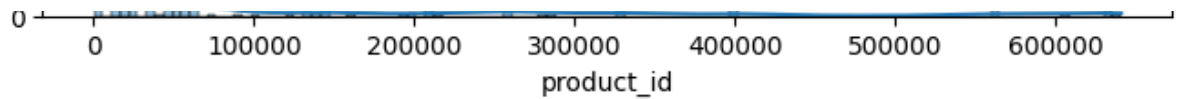
```
conteo_proveedores = proveedor_menor_precio_por_producto['supplier_id'].value_counts()
```

```
print(conteo_proveedores)
```

```
supplier_id
383      84
778      14
5879     14
4107      3
5978      3
61        2
5773      2
2         2
21        2
10        2
658       2
4623      2
4612      1
297       1
4403      1
5835      1
247       1
3840      1
4291      1
4192      1
5719      1
471       1
6237      1
1062      1
5906      1
3677      1
1646      1
202       1
5559      1
6200      1
63        1
4702      1
Name: count, dtype: int64
```

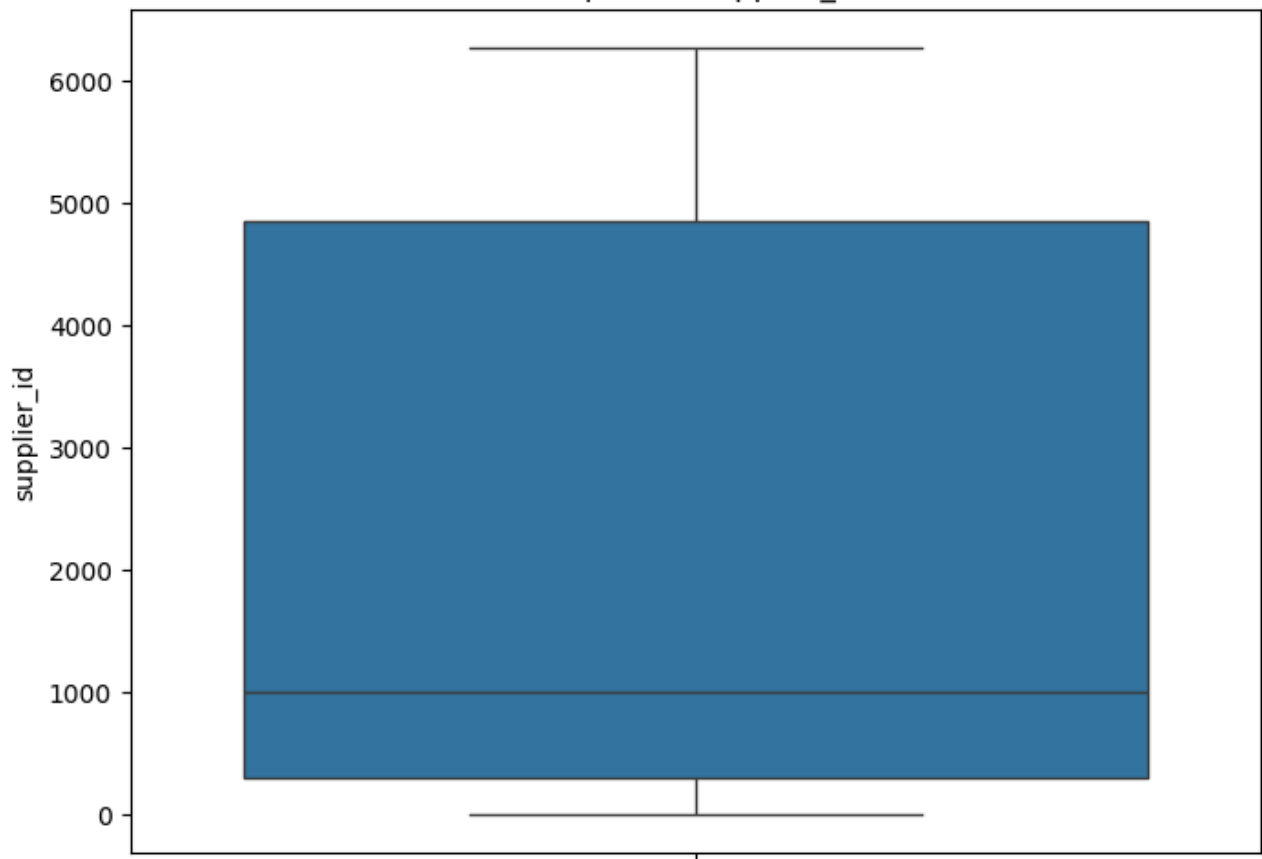
```
# Gráficos para variables numéricas
print("Gráficos para variables numéricas:")
numerical_cols = base_compras.select_dtypes(include=['int', 'float']).columns
for column in numerical_cols:
    plt.figure(figsize=(8, 6))
    sns.histplot(base_compras[column], kde=True)
    plt.title(f'Histograma de {column}')
    plt.xlabel(column)
    plt.ylabel('Frecuencia')
    plt.show()

    plt.figure(figsize=(8, 6))
    sns.boxplot(data=base_compras, y=column)
    plt.title(f'Boxplot de {column}')
    plt.ylabel(column)
    plt.show()
```



supplier_id

Boxplot de supplier_id



Histograma de price

