Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
Prof. Marvin Coto Jimenez
IE-0435
Inteligencia Artificial Aplicada a la Ingeniería Eléctrica

Proyecto 1

II Semestre de 2024

Agrupamiento de circuitos de distribución eléctrica

—

Entregado por:

Paulette Pérez
B95916

Issue date: 10 de septiembre

Abstract

En el siguiente reporte se utilizan los algoritmos de DBSCAN y K-means para agrupar circuitos de las redes eléctricas cuya duración promedio de las interrupciones en el servicio eléctrico mayores a 5 minutos (DPIR) y el promedio de interrupciones en el servicio eléctrico mayores a 5 minutos (FPI) sea mayor alos límites establecidos con el fin de encontrar grupos donde los transformadores de distribución necesiten una intervención. Se encuentra que el algoritmo K-means presentó los datos de la mejor forma. Los datos de DBSCAN tuvieron un cirterio correcto pero una segmentación poco práctica.

Contents

List of Figures

List of Tables

1 Introducción

La Autoridad Reguladora de Servicios Públicos es una institución pública encargada de fiscalizar la prestación de servicios públicos, así como su precio [1]. Ésta realiza, anualmente, un informe de la calidad de la energía suministrada por las empresas distribuidoras del país para asegurarse de que cumplan con las normativas establecidad. Para esto, la ARESEP usa dos indicadores, los cuales serían la duración promedio de las interrupciones en el servicio eléctrico mayores a 5 minutos (DPIR) que no debe pasar de 6 horas y el promedio de interrupciones en el servicio eléctrico mayores a 5 minutos (FPI), que debe ser igual o menor a 7 [2]. Estos datos se presentan al final del informe en forma de tabla, desglosados por circuito correspondiente y con la información de la empresa distribuidora. En el informe también hay una sección que muestra los resultados del "Programa de Intervención de Transformadores de Distribución", el cal debe ser ejecutado por las empresas distribuidoras para revisar que estos entreguen la tensión suficiente para abastecer a los usuarios [2], que afectaría directamente las métricas anteriores.
El siguiente proyecto tiene por hipótesis que, si se segmentan los circuitos en grupos con la mayor similitud, podría encontrarse áreas que requieran una atención prioritaria, particularmente si no están cumpliendo los estándares. Se buscarán grupos entre los circuitos con DPIR y FPIs por encima de los límites normativos para así sacar segmentos de población, ponderando entre la duración y cantidad de interrupciones al servicio con la cantidad de abonados, que podrían ayudar a las empresas que realizen Programa de Intervención de Transformadores a enfocar sus esfuerzos en las áreas que más lo requieran. Para esto se utilizarán principalmente dos algoritmos de agrupamiento: **K-means y DBSCAN**.

Algoritmos de agrupamiento

Hay diferentes tipos de conjuntos, pero una de las formas de clasificarlos es en si son **convexos** o no. Un conjunto convexo es uno en el cual, si se traza una línea recta para unir dos puntos, esta nunca sale del cluster [3]. Tomando en cuentas esta diferenciación, el análisis se hizo con los siguientes algoritmos de la librería *sklearn*:

K-means

Es un algoritmo que agrupa datos en k grupos de igual varianza. En su primera iteración designa puntos llamados centroides al azar. Asigna cada elemento a su centroide más cercano para posteriormente crear nuevos con el promedio de todas las distancias hacia los elementos asignados a los centroides anteriores. Este proceso se repite hasta llegar a

un cierto threshold, intentando obtener la menor inercia/ suma de cuadrados intra-cluster (WCSS: *within-cluster sum of squares*(ecuación 1)).

$$\sum_{i=0}^{n} \min_{\mu_j \in C}(||x_i - \mu_j||^2) \tag{1}$$

Donde μ_j es cada punto individual y x_i los centroides.
El K-means tiene desventajas como:

- Se **supone que los grupos son convexos** e isotrópicos, cuando no necesariamente sea el caso.
- K-means siempre convergerá eventualmente, aunque sea a un mínimo local, lo cual depende en gran medida de la inicialización de los centroides.
- En espacios de muy alta dimensión, las distancias euclidianas, y por ende el WCSS tienden a inflarse.
- La cantidad de grupos k debe elegirse a priopi.

[7]
Existen varios métodos de elección de k. En este proyecto se usarán los siguientes:

Método del Codo: Se hacen varias iteraciones de Kmeans con un k distinto, calculando en cada una el WCSS total entre todos los datos. Se grafican los pares (k, WCSS) esperando obtener una gráfica decreciente similar a un $y = 1/x$, en la que la variable dependiente desciende abruptamente para que en cieto punto los cambios sucesivos sean cada vez menores. Este sería el "codo" de la gráfica y el valor de k apropiado, aunque, al ser estrictemente visual, puede inducir a errores [10].

Coeficiente silueta: El coeficiente de silueta mide qué tan bien encaja cada punto de datos en su cluster, combinando información sobre la qué tan cerca está un de otros puntos en su propio cluster (cohesión) y la qué tan lejos está de los puntos en otros clusters (separación). De este modo se hace un promedio del coeficiente para todos los puntos. Un valor de 1 significa una clasificacón correcta, mientras que un -1 todo lo contrario [11]. Para escoger un valor de k, se harían varias iteraciones de Kmeans con un k distinto y se eligiría la que tenga el coeficiente silueta más alto.

DBSCAN

Es un algoritmo que considera los clústeres como áreas de alta densidad separadas por áreas de baja densidad, y que, a diferencia de K-means, **puede reconocer cuando un cluster no es convexo** [7].

Un cluster estaría formado de *core points*, puntos rodeados de una cantidad mínima m de otros puntos en un radio ϵ, y otros puntos que estén cercanos a los core points. Para designarlos, primero reconoce los core points, forma clústers juntando los que tengan otros core points dentro de su radio para luego añadirle los puntos que no sean core, pero que se encuentren dentro del radio de los primeros. Los puntos que sobren serían considerados outliers sin un grupo asignado [9][7].

Aunque no existe una métrica ideal para obtener m, aparte de que se recomienda que sea mayor a la cantidad de variables de los datos, exsiste una forma de obtener ϵ muy parecida al método del codo, calculando la distancia de cada punto hacia su *m-nearest-neighbor* y ordenándolas de menor a mayor en el eje y del plano junto al índice de su respectivo punto en el eje x. El punto que dictará el valor de ϵ será el que marque un aumento exponencial en el gráfico, a lo cual tomaremos el valor en y [5].

2 Procedimiento

2.1 Preprocesamiento de datos

Para obtener los anexos A y B del informe se procedió a usar Microsoft Excel, importando datos de un pdf. Como se puede ver en la figura 1, el software extrajo una gran cantidad de tablas, pues las separa si están en diferentes páginas. De estas se eligen las que están dentro del rango de páginas de los anexos.

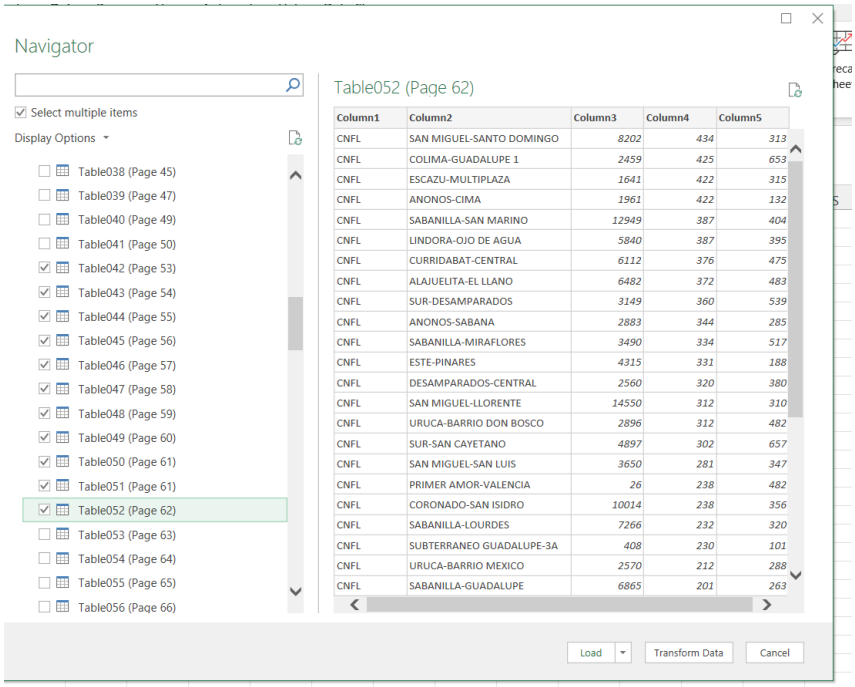


Figure 1: Selección de tablas con Excel

Posteriormente, usando Power Query, se modificó el nombre de las columnas de todas las tablas, apretándolas con click derecho, abriendo el editor avanzado y utilizando el comando resaltado en la figura 2.

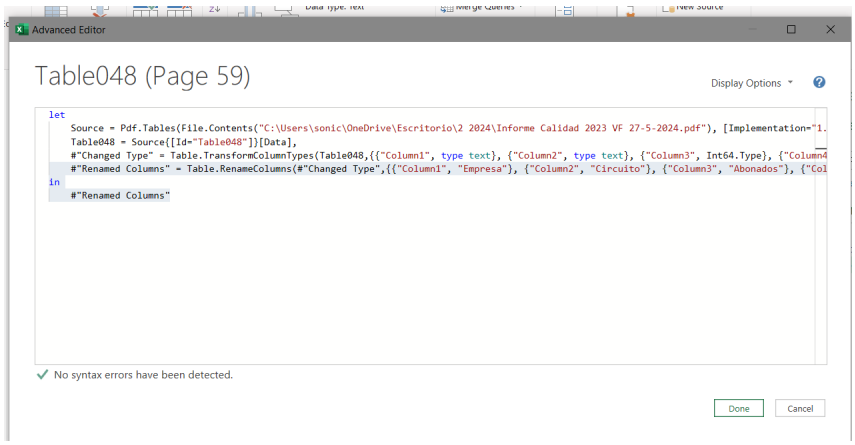


Figure 2: Código para cambio de nombre de las columnas en Power Query

Este paso fue necesario para combinar adecuadamente las tablas separadas en una por cada anexo (figura 3).

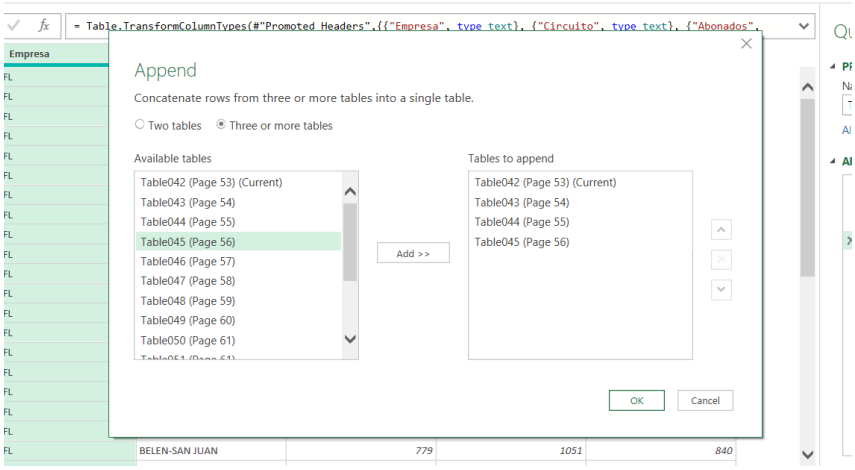
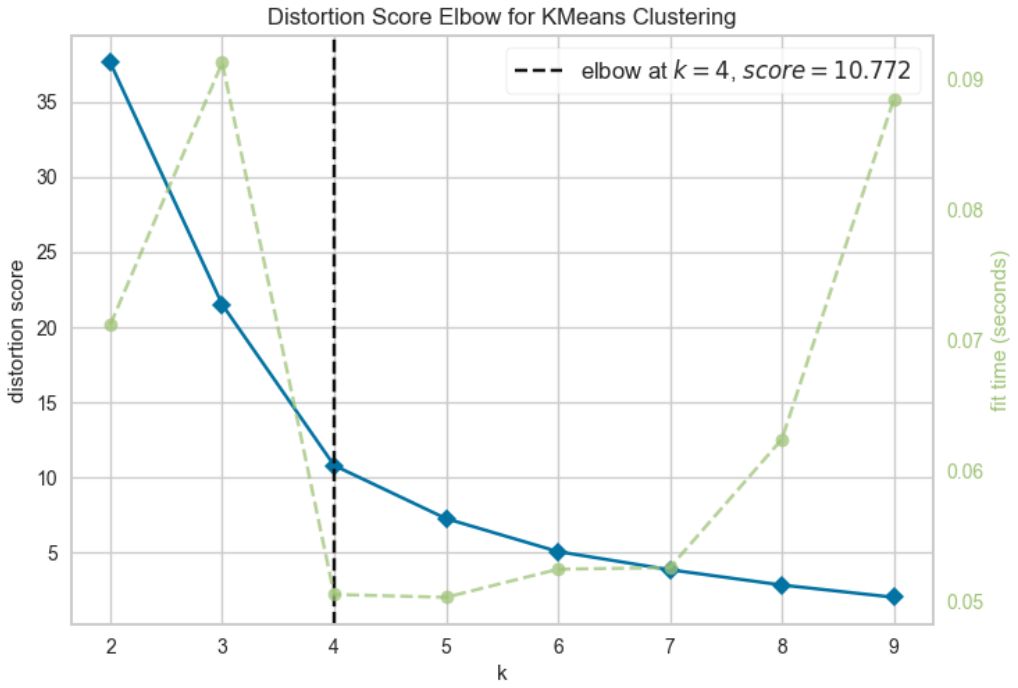


Figure 3: Combinación de tablas en Power Query

Ya que las tablas fueron acomodadas en Excel, se dividieron DPIR y FPI entre 100 porque al principio no se detectó el punto decimal y se ajustó a mano, luego, se exportaron a formato CSV UTF-8, el cual es el aceptado por herramientas como Pandas.

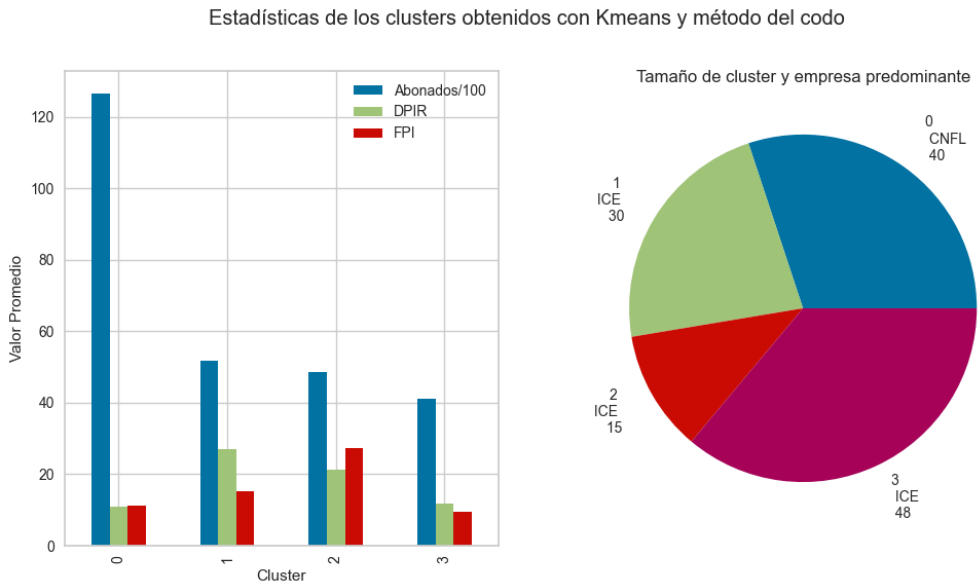
3 Resultados

Los resultados utiliza el Anexo A del informe y pueden ser generados con el código del anexo 4 o en el repositorio de git, donde se encuentran tablas detalladas.



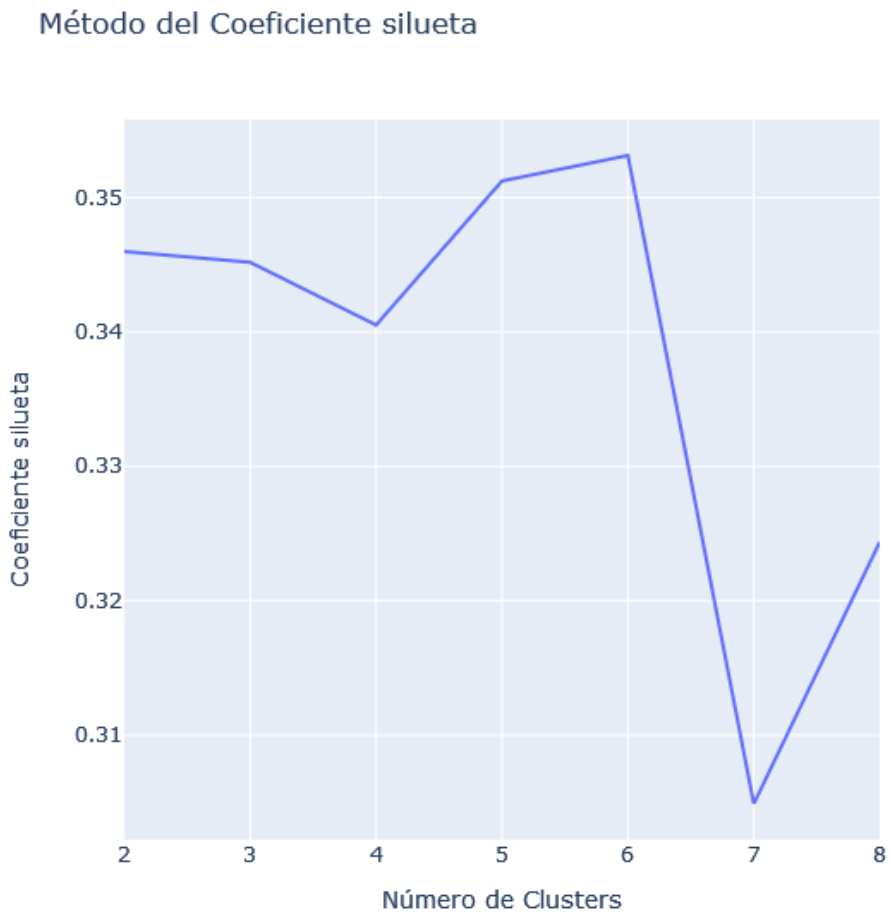
(a) Método del codo

(b) Clústers

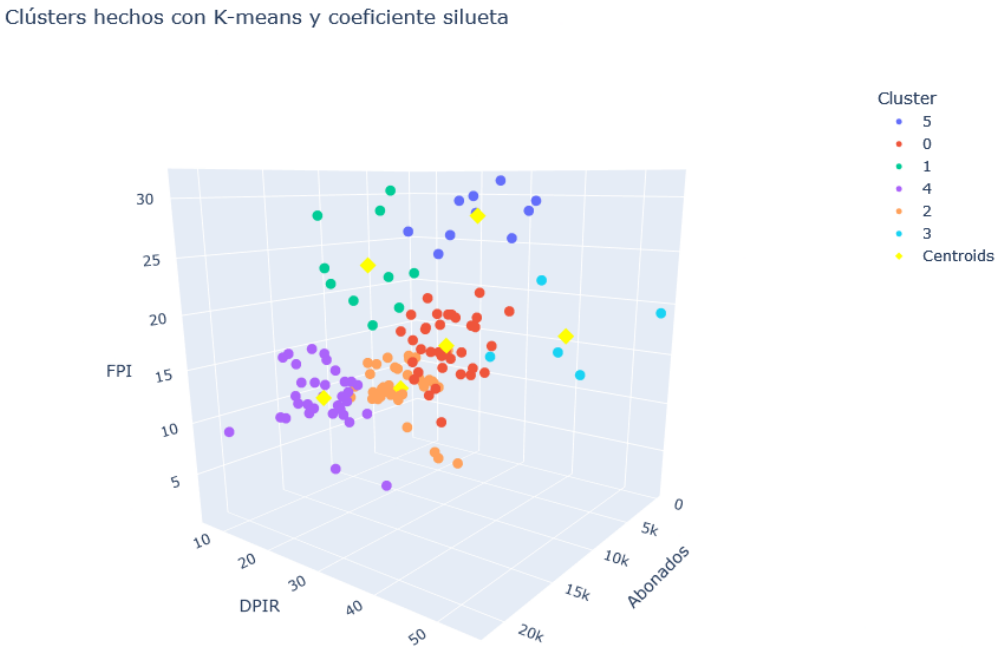


(c) Estadísticas de los clusters

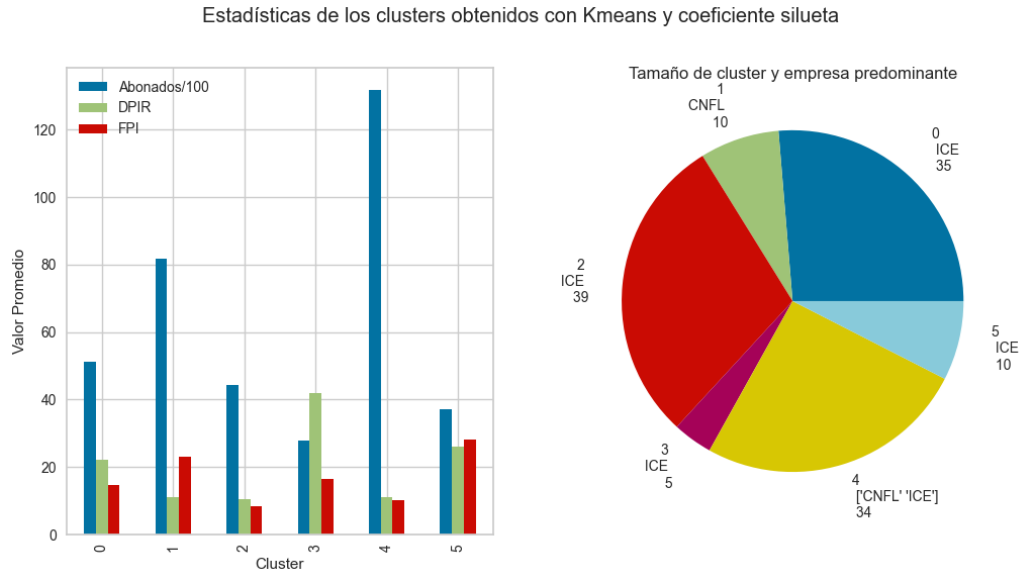
Figure 4: Resultados obtenidos con Kmeans y método del codo



(a) Coeficiente silueta

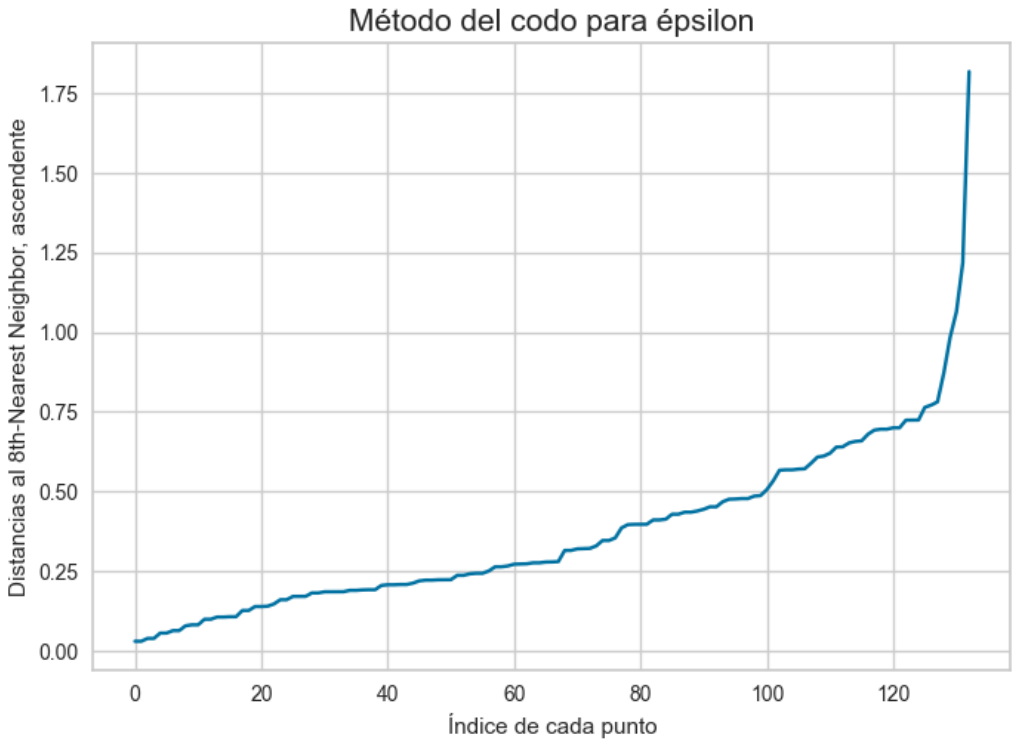


(b) Clústers

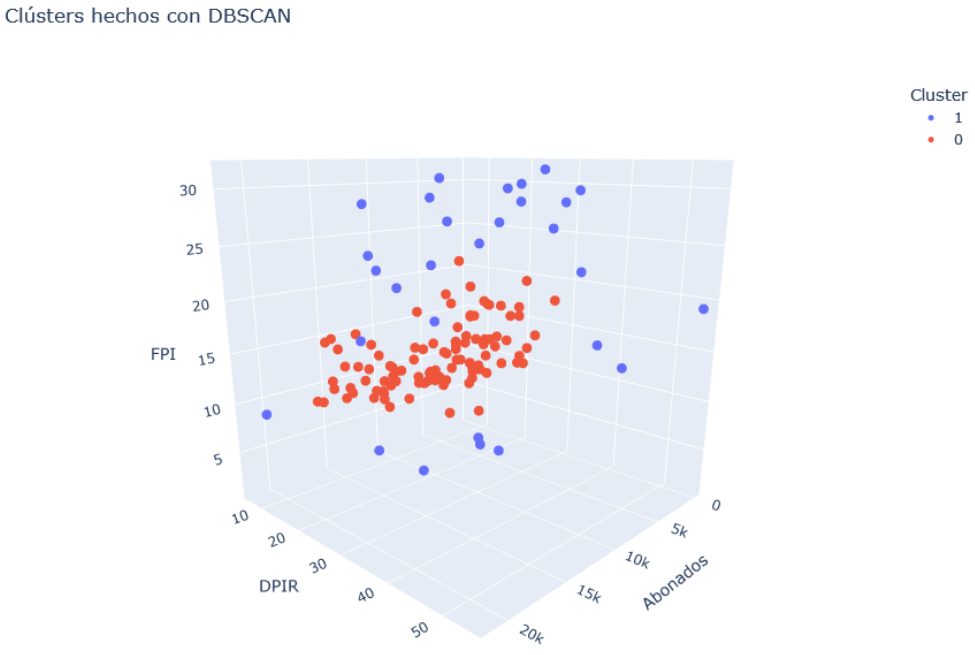


(c) Estadísticas de los clusters

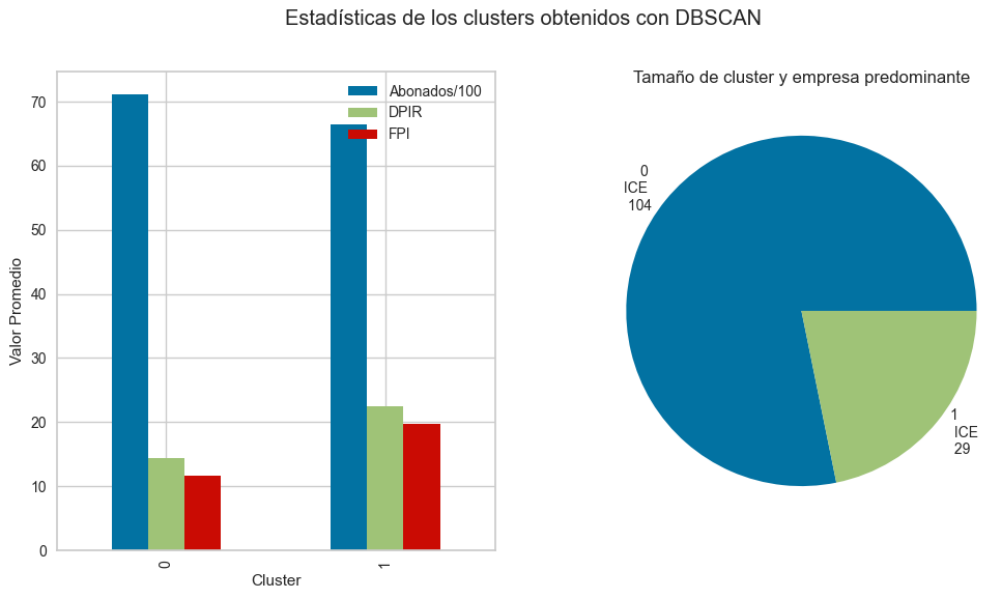
Figure 5: Resultados obtenidos con Kmeans y coeficeinte silueta



(a) Método del codo para épsilon



(b) Clústers



(c) Estadísticas de los clusters

Figure 6: Resultados obtenidos con DBSCAN

3.1 Análisis

Primero es necesario comparar los clústers reunidos por cada algoritmo. La figura 4b muestra 4 distintos después de utilizar el método del codo y la librería *yellowbrick* para elegir el valor. Comparándola con la figura 5b, que genera 6, su segmentación es más fácil de analiza, sin embargo, varios circuitos del clúster 1, los cuales están alejados del resto, son colocados en el clúster 3 (cyan) por el método de coeficiente silueta. Podría argumentarse que esto es correcto por su distancia a los demás, pues es más fácil verlos como outliers que como miembros del conjunto. Por su parte, en la figura 6b, DBSCAN reconoce un cúmulo de circuitos cercanos al origen en el clúster 0, con el clúster 1 rodeándolo. Se considera que esta segmentación es la más correcta, pues agrupa los circuitos de atributos que en promedio, son más bajos y por tanto, menos urgentes de revisar, juntos, mientras que hace otro grupo con de los circuitos de atributos numéricamente más áltos y con más necesidad de atención, sin embargo, ésta segmentación puede no ser la más útil, ya que podemos querer grupos más específicos. La métrica utilizada para dar atención a un grupo de circuitos por sobre otro es la que dictaminará qué agrupamiento es mejor. La frecuencia de cortes puede esar más relacionada con el bienestar de los transformadores que su duración, que puede tener más que ver con la respuesta de la empresa encargada ante estos. Si el clúster es pequeño pero la cantidad de abonados es grande, hay que revisar si sus equipos cuentan con la infraestructura necesaria para soportarlos.
En cuanto a las características de los clústers, la cantidad de abonados a cada circuito se dividió entre 100 para que fuera acorde a la escala del gráfico.

En la figura 4c se muestran los valores promedio de los 4 clústers obtenidos con K-means y método del codo. La mayoría de circuitos distintos están repartidos en el 0, 1 y 3. El primero tiene una gran cantidad de abonados con DPIR y FPIs relativamente bajos con respecto a los otros, principalmente la duración, lo que podemos relacionarlo a que, por haber tantas personas utilizando ésta red, se les da prioridad en el arreglo. El cluster 1 tiene menos abonados pero un DPIR más grande y FPI ligeramente mayor, siendo mayor la duración que la frecuencia. El número 3 tiene una cantidad de abonados semejante al anterior con una cantidad de cortes mayor a la duración. Por último, el clúster número 2, que es el más pequeño, tiene una baja cantidad de abonados, de cortes, y de duración en éstos; ya sea que la prioridad del circuito dependa de la cantidad de abonados o las otras dos métricas, este tiene valores más bajos que los anteriores, por lo que la atención hacia este sería de la más baja prioridad.
En la figura 5c se muestran los valores promedio de los 6 clústers obtenidos con K-means y el coeficiente silueta. La mayoría de circuitos distintos están repartidos en el 0, 2 y 4. El primero tiene ununos atributos intermedios con respecto a los otros. El cluster 2 tiene un poco menos de abonados pero un DPIR y FPI mucho menor. El número 4 presenta una situación similar al número 0 del caso anterior. Finamente hay tres clústers minoritarios, los 1, 3 y 5, que entre todos cuentan con la mayor cantidad y duración de cortes de luz. Con éste método podemos deducir por ejemplo, que el clúster 1 necesita atención, pues una gran catidad de abonados están experimentando muchas interrupciones en el servicio y que la mayor cantidad de fallos están en pocos circuitos.
Por último, DBSCAN genera dos conjuntos, pero uno de estos abarca a más de la mitad de los circuitos registrados, el que tiene más abonados pero menos fallas. Aunque respalda la teoría de que la duración de los cortes es inversamente proporcional al número de abonados, no sirve de mucho para segmentar poblaciones, porque no nos da información útil.
Podemos ver que el método K-means da unos valores más fáciles de analizar y se les puede sacar más provecho. Ya que, tanto en con el coeficiente silueta como con el método del codo los clusters son distintos entre sí se puede usar el que mejor sirva para la ocasión: aunque el coeficiente silueta busque la mejor pertenencia al clúster de cada elemento, es cierto que 6 son difíciles de observar.
Nótese que, además de la CNFL y el ICE, las demás empresas distribuidoras no figuran, lo que puede explicarse con su baja cuota de mercado, pero también indica que sus circuitos no tienen un comportamiento particular que no pueda ser equiparado al de las otras compañías mayoritarias. También, preprocesar los datos para obtener información de la zona geográfica de los circuitos no daría información con la que podríamos conocer si un lugar en particular está siendo poco atendido y buscar por qué.

3.2 Preguntas

1. ¿Por qué se requiere una autoridad como ARESEP para evaluar el sistema de generación, transmisión, distribución y la calidad de la energía eléctrica nacional?

La ARESEP tiene varias funciones importantes, entre ellas están velar porque las empresas con consesiones de servicios públicos mantengan un estandar de calidad, pues no podemos garantizar que lo sigan si no son fiscalizados, y menos si son los únicos a cargo de este.
Otra es mantener las tarifas de los servicios públicos en un valor razonable y acorde con las condiciones tanto del usuario como de la empresa, pues, igual que en el caso anterior, no se puede garantizar que esta las mantenga en un precio razonable si tiene el completo control sobre su sector.
Una función más es recoger datos como los analizados en este proyecto. En caso contrario, puede que no haya un órgano con incentivos de capturar indicadores que puedan darnos una idea sobre el verdadero estado de los servicios que consumimos.
Para que una entidad como esta cumpla su cometido, es necesario asegurar que usa las metodologías más eficientes, en pro del beneficio de la mayoría.

2. ¿Cuál es el propósito de utilizar varios métodos de agrupamiento para este problema?

Esto es debido a que cada uno tiene sus ventajas y desventajas. Como se mencionó anteriormente, K-means es sensible a la escala de los datos y no reconoce conjuntos convexos, mientras que DBSCAN sí lo hace. Al usar ámbos podemos ver cuál de los dos se ajusta mejor a los datos y a nuestras necesidades. Como lo vimos en este proyecto, k-means genera conjuntos más útiles para el análisis, aunque los conjuntos de DBSCAN sean más fáciles de explicar.

3. Mencione y describa un caso de métodos de agrupamiento que se encuentre en la literatura científica reciente relacionada con el sector eléctrico

En [6] se utilizan algoritmos de agrupamiento para conocer patrónes de consumo de electricidad de clientes de Irlanda mediante sus medidores inteligentes. Algunos de los algoritmos utilizados fueron K-means, Fuzzy C-means (FCM), clustering jerárquico, mapas auto-organizados (SOM), y Gaussian Mixture Models (GMM). Se logra observar picos en ciertas horas del día.

4. ¿Existe alguna anomalía en estos conjuntos de datos? ¿Cómo pueden detectarse anomalías con otros métodos además de agrupamiento?

En cuanto a los datos preprocesados, no se detectaron anomalías aademás de que no se

hubiera reconocido el punto decimal de éstos al inicio, lo cual fue corregido a mano. En los clusters no se presentaron outliers, mas aumentar la cantidad de éstos logra que datos más dispersos se separen en su propio clúster.
El algoritmo Isolation Forest basado en árboles de decisión, está especialmente diseñado para encontrar anomalías en datos [4].

5. ¿Qué papel juega la normalización de los datos en este problema? ¿Existen otras formas de realizar normalización de datos? ¿Cambian los resultados si no se aplica la normalización?

La función de normalización *StandardScaler()* de sklearn modificó los datos de modo que tubieran una media de 0 y una desviación estandar de 1 [8]. Como experimento, el el código se realizó una simulación de cómo se verían los clusters de k-means con método del codo con los datos sin escalar, de lo que, si bien se obtuvieron la misma cantidad de clusters, su forma cambió bastante. Obsérvese que la cantidad de abonados al un circuito tiene una excala mucho mayor al DPIR y al FPI, por lo que, si no estuvieran escalados, las distancias se inflarían y sus variaciones se haría menos significativas, por lo que podemos decir que los datos escalares fueron más convenientes, especialmente en K-means. Si se observa el gráfico sin escalar dentro del código en el repositorio, se nota que los grupos están divididos estrictamente a lo largo del eje de los abonados, lo que refleja lo anteriormente comentado.
La librería sklearn también cuenta con las funciones *MinMaxScaler()* o *MaxAbsScaler()*, las cuales escalan los valores por dentro de cierto rango, generalmente entre 0 y 1 [8].

3.3 Uso de LLM

Para resolver lo solicitado, se utilizó la versión de paga de chat gpt y se le proporcionó el informe, junto con el siguiente prompt:

> "hi chat gpt, convert the table inside this pdf file that spreads across page 53 to 56, called anexo A, into a CSV file. Do the same with the table anexo B, which spreads across page 57 to 66, put this one in another file"

Seguido de lo cual la herramienta generó estos resultados:

	A	B	C	D	E	F	G	H
1	Empresa	Circuito	Abonados	DPIR	FPI			
2	CNFL	BARVA-CIF	3756	32.73	25.97			
3	COOPELES	MONTERR	4820	28.91	23.48			
4	JASEC	PARAISO	1234	19.83	16.22			
5	ICE	ANGOSTU	7896	22.45	20.58			
6	COOPEGU	LA TABLA	2345	25.97	18.76			
7								
8								
9								
10								
11								
12								
13								
14								
15								
16								

(a) Anexo A

	A	B	C	D	E	F	G	H
1	Empresa	Circuito	Abonados	DPIR	FPI			
2	COOPEALF	SAN JOSE	1200	5.76	6.89			
3	ESPH	HEREDIA	3400	4.32	5.67			
4	CNFL	SAN PEDR	2750	3.98	4.32			
5	COOPESAI	ALAJUELA	1900	2.45	3.21			
6	ICE	LIMON	4500	1.89	2.98			
7								
8								
9								
10								
11								
12								
13								
14								

(b) Anexo B

Figure 7: Resultados de ChatGPT ante el primer prompt

Como puede observarse, estos no tienen la información completaque se extrajo manualmente con PowerQuery. Seguido de esto, se le envió el siguiente promp, intentando que la IA pudiera corregir su curso, con más indicaciones:

> "the tables are separated across many pages, because they don't fit in juts one, so i expect you to be able to recognize this and join the corresponding sections into their respective tables: one for anexo A and another for anexo B. What you did was select a few values from the tables and putting them in a file, but i want you to put all"

A lo que ChatGPT generó archivos csv sin nada de contenido. Después de muchos intentos, añadiendo cada vez más detalles sobre el objetivo de la conversación, **no se logró** de ninguna forma que este generara las tablas solicitadas. La interacción completa puede observarse en el chat archivado "Convert Tables to CSV (Paulette)" dentro de la cuenta comunitaria de ChatGPT.

4 Conclusiones

- Las empresas distribuidoras que no son el ICE ni la CNFL no presentan un comportamiento particular en cuanto a las caracterísiteicas de los circuitos con baja calidad de suministro.
- K-means fue el algoritmo más útil para segmentar poblaciones, mientras que DBSCAN fue el más exacto.
- Los circuitos con más abonados presentan menos fallas en el sevicio, tanto en duración como en frecuencia.
- Una segmentación por zona geográfica de los circuitos podría brindar más información.

References

[1] ARESEP. *¿Quiénes somos?* 2023. URL: https://aresep.go.cr/quienes-somos/.

[2] ARESEP. *Informe de la calidad del suministro de electricidad Sistema de Distribución 2023 Intendencia de Energía*. Tech. rep. ARESEP, 2023. URL: https://aresep.go.cr/electricidad-calidad/informe-de-calidad-electrica-del-servicio-de-distribucion-2023/.

[3] Geoff Gordon. *Lecture 3: September 4*. 2012. URL: https://www.cs.cmu.edu/~ggordon/10725-F12/scribes/10725_Lecture3.pdf.

[4] Dhiraj K and James Skelton. *Anomaly Detection Using Isolation Forest in Python*. 2024. URL: https://www.digitalocean.com/community/tutorials/anomaly-detection-isolation-forest.

[5] Tara Mullin. *DBSCAN Parameter Estimation Using Python*. 2020. URL: https://medium.com/@tarammullin/dbscan-parameter-estimation-ff8330e3a3bd.

[6] Amin Rajabi et al. "A comparative study of clustering techniques for electrical load pattern segmentation". In: *Renewable and Sustainable Energy Reviews* 120 (2020), p. 109628. URL: https://www.sciencedirect.com/science/article/pii/S1364032119308354.

[7] scikit-learn.org. *2.3. Clustering*. URL: https://scikit-learn.org/stable/modules/clustering.html.

[8] scikit-learn.org. *6.3. Preprocessing data*. URL: https://scikit-learn.org/stable/modules/preprocessing.html#preprocessing-scaler.

[9] Josh Starmer. *Clustering with DBSCAN, Clearly Explained!!!* 2022. URL: https://www.youtube.com/watch?v=RDZUdRSDOok.

[10] Anmol Tomar. *Stop Using Elbow Method in K-Means Clustering*. 2023. URL: https://builtin.com/data-science/elbow-method.

[11] Suraj Yadav. *Silhouette Coefficient Explained with a Practical Example: Assessing Cluster Fit*. 2023. URL: https://medium.com/@Suraj_Yadav/silhouette-coefficient-explained-with-a-practical-example-assessing-cluster-fit-c0bb3fdef719.

Appendix

Apendice 1: Código de Python

```
  #!/usr/bin/env python
# coding: utf-8

# # Proyecto 1: Inteligencia Artificial Aplicada a la Ingeniería Eléctrica

# ## Algoritmo 1: K-means

# In[1]:


# Importamos las librerías
import pandas as pd
import numpy as np
import plotly.express as px
import plotly.graph_objects as go
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans, DBSCAN
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import NearestNeighbors
from yellowbrick.cluster import KElbowVisualizer, SilhouetteVisualizer


# In[2]:


# Importamos los datos del anexo A
csv_file = 'AnexoA.csv'
datosA = pd.read_csv(csv_file)
print(datosA.info())
print(datosA.describe())


# In[3]:


# Importamos los datos del anexo b
csv_file = 'AnexoB.csv'
datosB = pd.read_csv(csv_file)
print(datosB.info())
print(datosB.describe())


# In[4]:
```

```
# Análisis exploratorio gráfica de los datos


fig = px.scatter_3d(datosA,
                    x="Abonados",
                    y="DPIR",
                    z="FPI",
                    title="Datos Anexo A")
fig.update_traces(
    textfont=dict(
        family="Arial",
        size=18,
        color='red'
    ),
    marker=dict(
        size=5
    )
)
fig.show()


# In[5]:


# Análisis exploratorio gráfica de los datos
fig = px.scatter_3d(datosB,
                    x="Abonados",
                    y="DPIR",
                    z="FPI",
                    title="Datos Anexo B")
fig.update_traces(
    textfont=dict(
        family="Arial",
        size=18,
        color="crimson"
    ),
    marker=dict(
        size=5
    )
)
fig.show()


# ***Escalamos los datos***

# In[6]:
```

```
data = datosA.select_dtypes(include=[int,float])
scaler = StandardScaler() #investigar qué hace
data = scaler.fit_transform(data)


# ### Método del codo

# In[7]:


km = KMeans(random_state=42)
visualizer = KElbowVisualizer(km, k=(2,10))

visualizer.fit(data)
visualizer.show()


# Visualizamos que el gráfico no converge, sino que sigue bajando indefinidamente. Se
↪  elige k = 4 porque la librería yellowbrick proporciona un gráfico que sugiere un
↪  valor de k adecuado.

# In[8]:


# Elegimos k = 4 a partir de la gráfica
kmeans = KMeans(n_clusters=4, random_state=42)
clusters = kmeans.fit(data)
y_means = kmeans.predict(data).astype(str)
y_means = y_means.reshape(133,1)
# quitamos escalamientos
centers = scaler.inverse_transform(kmeans.cluster_centers_)

data_kmeans_elbow =  datosA.copy()
data_kmeans_elbow["Cluster"]= y_means


# In[9]:


fig = px.scatter_3d(data_kmeans_elbow,
                    x="Abonados",
                    y="DPIR", z="FPI",
                    color="Cluster",
                    title='Clústers hechos con K-means y método del codo')
fig.add_trace(go.Scatter3d(
    x=centers[:, 0],
```

```
    y=centers[:, 1],
    z=centers[:, 2],
    mode='markers',
    marker=dict(size=20,
                color='yellow',
                symbol='diamond'),
    name='Centroids'
))

fig.update_traces(
    textfont=dict(
        family="Arial",
        size=15,
        color="crimson"
    ),
    marker=dict(
        size=5,
    )
)
fig.show()


# #### Estadísticas

# In[10]:


data_kmeans_grouped = data_kmeans_elbow.groupby("Cluster").agg({"Abonados":['mean'],
                                    "DPIR":['mean'],
                                    "FPI":['mean'],
                                  "Empresa":[pd.Series.mode],
                                    "Cluster":'count'})
print(data_kmeans_grouped)


# In[11]:


# generamos gráficos más comprensibles
data_kmeans_grouped1 = data_kmeans_elbow.groupby("Cluster").agg({"Abonados":['mean'],
                                    "DPIR":['mean'],
                                    "FPI":['mean'],
                                  "Empresa":[pd.Series.mode]
                                    })

data_kmeans_grouped1['Abonados'] = data_kmeans_grouped1['Abonados']/100
fig, ax = plt.subplots(1, 2, figsize=(12, 6))
data_kmeans_grouped1.plot(kind='bar',
```

```
                            ax=ax[0],
                            xlabel='Cluster',
                            ylabel='Valor Promedio')
ax[0].legend(['Abonados/100', 'DPIR', 'FPI'])

empresas=[data_kmeans_grouped1.Empresa.values[i][0] \
for i in range(len(data_kmeans_grouped1.Empresa.values))]
data_kmeans_grouped2 = data_kmeans_elbow.groupby("Cluster").agg({"Cluster":'count'})


formatter = [f"{i} \n {j} \n {l}" for i,j,l in zip(list(data_kmeans_grouped1.index),
                                                   empresas,
                                                   list(data_kmeans_grouped2.Cluster))]
ax[1].pie(data_kmeans_grouped2.Cluster,
          labels=formatter,
          labeldistance=1.2)
ax[1].set(title="Tamaño de cluster y empresa predominante")
fig.suptitle('Estadísticas de los clusters obtenidos con Kmeans y método del codo')
plt.show()


# ### Coeficiente silueta

# In[12]:


sc = {"Número de Clusters":[],
      "Coeficiente silueta":[]}

for k in range(2,9):
      kmeans = KMeans(n_clusters=k, random_state=42).fit(data)
      label = kmeans.labels_
      sil_coeff = silhouette_score(data,label,metric = 'euclidean')
      sc["Número de Clusters"].append(k)
      sc["Coeficiente silueta"].append(sil_coeff)

fig = px.line(sc, x="Número de Clusters", y="Coeficiente silueta",
              title="Método del Coeficiente silueta",
               width=600,
               height=600)

fig.show()


# In[13]:
```

```
# Elegimos k = 2 a partir de la gráfica
kmeans = KMeans(n_clusters=6, random_state=42)
clusters = kmeans.fit(data)
y_means = kmeans.predict(data).astype(str)
y_means = y_means.reshape(133,1)
centers = scaler.inverse_transform(kmeans.cluster_centers_) #quitamos escalamientos

data_kmeans_sil =  datosA.copy()
data_kmeans_sil["Cluster"]= y_means


# In[ ]:




# In[14]:


fig = px.scatter_3d(data_kmeans_sil,
                    x="Abonados",
                    y="DPIR", z="FPI",
                    color="Cluster",
                    title='Clústers hechos con K-means y coeficiente silueta')
fig.add_trace(go.Scatter3d(
    x=centers[:, 0],
    y=centers[:, 1],
    z=centers[:, 2],
    mode='markers',
    marker=dict(size=20,
                color='yellow',
                symbol='diamond'),
    name='Centroids'
))

fig.update_traces(
    textfont=dict(
        family="Arial",
        size=15,
        color="crimson"
    ),
    marker=dict(
        size=5,
    )
)
fig.show()
```

```
# #### Estadísticas

# In[15]:


data_kmeans_grouped = data_kmeans_sil.groupby("Cluster").agg({"Abonados":['mean'],
                                    "DPIR":['mean'],
                                    "FPI":['mean'],
                                  "Empresa":[pd.Series.mode],
                                    "Cluster":'count'})
print(data_kmeans_grouped)


# In[16]:


# generamos gráficos más comprensibles
data_kmeans_grouped1 = data_kmeans_sil.groupby("Cluster").agg({"Abonados":['mean'],
                                    "DPIR":['mean'],
                                    "FPI":['mean'],
                                  "Empresa":[pd.Series.mode]
                                    })

data_kmeans_grouped1['Abonados'] = data_kmeans_grouped1['Abonados']/100
fig, ax = plt.subplots(1, 2, figsize=(12, 6))
data_kmeans_grouped1.plot(kind='bar',
                          ax=ax[0],
                          xlabel='Cluster',
                          ylabel='Valor Promedio')
ax[0].legend(['Abonados/100', 'DPIR', 'FPI'])

empresas=[data_kmeans_grouped1.Empresa.values[i][0] \
for i in range(len(data_kmeans_grouped1.Empresa.values))]
data_kmeans_grouped2 = data_kmeans_sil.groupby("Cluster").agg({"Cluster":'count'})


formatter = [f"{i} \n {j} \n {l}" for i,j,l in zip(list(data_kmeans_grouped1.index),
                                                   empresas,
                                                   list(data_kmeans_grouped2.Cluster))]
ax[1].pie(data_kmeans_grouped2.Cluster, labels=formatter, labeldistance=1.2)
ax[1].set(title="Tamaño de cluster y empresa predominante")
fig.suptitle('Estadísticas de los clusters obtenidos con Kmeans y coeficiente
silueta')
plt.show()
```

```
# ## Algoritmo 2: DBSCAN

# Mediante prueba y error se concluyó que los clusters del algoritmo DBSCAN convergen
↪  en 2, y la cantidad mínima de puntos que rodean a un core point necesaria para
↪  conseguirlo fue 8

# ### Obtención de eps/epsilon

# In[17]:


neighbors = NearestNeighbors(n_neighbors=4)
neighbors_fit = neighbors.fit(data)
distances, indices = neighbors_fit.kneighbors(data)
distances = np.sort(distances, axis=0)
distances = distances[:,1]
plt.plot(distances)
plt.title("Método del codo para épsilon", fontsize=15)
plt.ylabel("Distancias al 8th-Nearest Neighbor, ascendente")
plt.xlabel("Índice de cada punto")
plt.show()


# In[18]:


dbscan = DBSCAN(eps=0.75, min_samples=4)
dbscan.fit(data)
data_DBSCAN = pd.DataFrame(datosA.copy()) #datos sin escalar
data_DBSCAN["Cluster"] = dbscan.labels_.astype(str).tolist()
data_DBSCAN["Cluster"].replace({'-1':'1'}, inplace=True)
fig = px.scatter_3d(data_DBSCAN, x="Abonados", y="DPIR", z="FPI",
                    color="Cluster", title='Clústers hechos con DBSCAN')

fig.update_traces(
    textfont=dict(
        family="Arial",
        size=18,
        color="crimson"
    ),
    marker=dict(
        size=5
    )
)
fig.show()


# ### Estadísticas
```

```
# In[19]:


data_DBSCAN_grouped = data_DBSCAN.groupby("Cluster").agg({"Abonados":['mean'],
                                    "DPIR":['mean'],
                                    "FPI":['mean'],
                                   "Empresa":[pd.Series.mode],
                                    "Cluster":'count'})
print(data_DBSCAN_grouped)


# In[20]:


# generamos gráficos para entender mejor la data
data_DBSCAN_grouped1 = data_DBSCAN.groupby("Cluster").agg({"Abonados":['mean'],
                                    "DPIR":['mean'],
                                    "FPI":['mean'],
                                   "Empresa":[pd.Series.mode]
                                    })
data_DBSCAN_grouped1['Abonados'] = data_DBSCAN_grouped1['Abonados']/100

fig, ax = plt.subplots(1, 2, figsize=(12, 6))
data_DBSCAN_grouped1.plot(kind='bar',
                          ax=ax[0],
                          xlabel='Cluster',
                          ylabel='Valor Promedio')
ax[0].legend(['Abonados/100', 'DPIR', 'FPI'])


empresas=[data_DBSCAN_grouped1.Empresa.values[i][0] \
for i in range(len(data_DBSCAN_grouped1.Empresa.values))]
data_DBSCAN_grouped2 = data_DBSCAN.groupby("Cluster").agg({"Cluster":'count'})

formatter = [f"{i} \n {j} \n {l}" for i,j,l in zip(list(data_DBSCAN_grouped1.index),
                                                   empresas,
                                                   ↪ list(data_DBSCAN_grouped2.Cluster))]
ax[1].pie(data_DBSCAN_grouped2.Cluster, labels=formatter)
ax[1].set(title="Tamaño de cluster y empresa predominante")
fig.suptitle('Estadísticas de los clusters obtenidos con DBSCAN')
plt.show()


# # Extra: Datos sin escalar

# ## K- means: Método del codo
```

```
# In[21]:


sin_escalar = datosA.select_dtypes(include=[int,float])
km = KMeans(random_state=42)
visualizer = KElbowVisualizer(km, k=(2,10))

visualizer.fit(sin_escalar)
visualizer.show()


# In[22]:


kmeans = KMeans(n_clusters=4, random_state=42)
clusters = kmeans.fit(sin_escalar)
y_means = kmeans.predict(sin_escalar).astype(str)
y_means = y_means.reshape(133,1)
centers = kmeans.cluster_centers_
data_kmeans_elbow_sin_escalar =  datosA.copy()
data_kmeans_elbow_sin_escalar["Cluster"]= y_means


# In[23]:


data_kmeans_grouped1 =
↪  data_kmeans_elbow_sin_escalar.groupby("Cluster").agg({"Abonados":['mean'],
                                    "DPIR":['mean'],
                                    "FPI":['mean'],
                                  "Empresa":[pd.Series.mode]
                                    })

data_kmeans_grouped1['Abonados'] = data_kmeans_grouped1['Abonados']/100
fig, ax = plt.subplots(1, 2, figsize=(12, 6))
data_kmeans_grouped1.plot(kind='bar',
                          ax=ax[0],
                          xlabel='Cluster',
                          ylabel='Valor Promedio')
ax[0].legend(['Abonados/100', 'DPIR', 'FPI'])

empresas=[data_kmeans_grouped1.Empresa.values[i][0] \
for i in range(len(data_kmeans_grouped1.Empresa.values))]
data_kmeans_grouped2 = data_kmeans_elbow.groupby("Cluster").agg({"Cluster":'count'})


formatter = [f"{i} \n {j} \n {l}" for i,j,l in zip(list(data_kmeans_grouped1.index),
```

```
                                                    empresas,
                                                    ↪ list(data_kmeans_grouped2.Cluster))]
ax[1].pie(data_kmeans_grouped2.Cluster, labels=formatter, labeldistance=1.2)
ax[1].set(title="Tamaño de cluster y empresa predominante")
fig.suptitle('Estadísticas de los clusters obtenidos con Kmeans y método del codo con
↪ datos sin escalar')
plt.show()


# In[24]:


fig = px.scatter_3d(data_kmeans_elbow_sin_escalar,
                    x="Abonados",
                    y="DPIR",
                    z="FPI",
                    color="Cluster",
                    title='Clústers sin escalar hechos con K-means y método del codo')
fig.add_trace(go.Scatter3d(
    x=centers[:, 0],
    y=centers[:, 1],
    z=centers[:, 2],
    mode='markers',
    marker=dict(size=20,
                color='yellow',
                symbol='diamond'),
    name='Centroids'
))

fig.update_traces(
    textfont=dict(
        family="Arial",
        size=15,
        color="crimson"
    ),
    marker=dict(
        size=5,
    )
)
fig.show()
```