Universidad de Costa Rica
Escuela de Ingeniería eléctrica

IE0435
Inteligencia Artificial Aplicada a la Ingeniería Eléctrica

Proyecto 1:
Agrupamiento de circuitos de distribución eléctrica

Estudiante:

Yehohnathan Miranda Vigil C04879

Profesor:
PhD. Marvin Coto Jiménez

II - 2024

10 de septiembre del 2024

Resumen

Este proyecto se enfoca en el análisis del Informe de Calidad del Suministro Eléctrico de la Autoridad Reguladora de Servicios Públicos (ARESEP) del año 2022 en Costa Rica, utilizando algoritmos de agrupamiento KMeans y DBSCAN para identificar patrones en los circuitos de distribución eléctrica, y consecuentemente formar grupos de datos que proporcionen información valiosa respecto del servicio eléctrico costarricense. El procedimiento incluyó la extracción de datos del informe, su conversión de PDF a formato Markdown y posteriormente a CSV para su limpieza y análisis, y la creación de clases mediante la herramienta Python para conversión y uso de los algoritmos de agrupamiento. Los datos se normalizaron para observar la precisión de los algoritmos de agrupamiento en distintos casos.

El algoritmo KMeans se aplicó a los datos brutos y normalizados del Anexo A del Informe, generando diferentes agrupaciones de circuitos en función del número de abonados, la duración promedio de interrupciones (DPIR) y la frecuencia promedio de interrupciones (FPI). Con datos normalizados, KMeans reduce el impacto del número de abonados, permitiendo una mejor identificación de circuitos con características extremas en DPIR y FPI. Por otro lado, el algoritmo DBSCAN identificó tanto los mejores como los peores casos de desempeño en los circuitos eléctricos, clasificando algunos casos como ruido, sin encontrar variaciones significativas entre los resultados con datos normalizados y no normalizados.

KMeans resultó útil para clasificar los circuitos en relación de su desempeño general respecto DPI y FPI con los datos normalizados. DBSCAN permitió detectar casos atípicos con un rendimiento inusual, como el mejor caso y los peores casos entre los datos en desempeño. Permitiendo un análisis de la calidad del suministro eléctrico con el objetivo a futuro de mejorar la gestión y la regulación de los servicios de distribución en Costa Rica.

Palabras clave: Informe de Calidad del Suministro Eléctrico, ARESEP, Algoritmo de Agrupamiento, KMeans, DBSCAN, Normalización de Datos, Circuitos de Distribución Eléctrica, Duración Promedio de, Interrupciones (DPIR), Frecuencia Promedio de Interrupciones (FPI), Clusters

Índice

Lista de figuras

Lista de tablas

1. Introducción

La presencia de una entidad encargada de evaluar la calidad del servicio eléctrico del país, como lo es la Autoridad Reguladora de los Servicios Públicos (ARESEP) es la entidad encargada de monitorear, asegurando que se cumplan con los estándares establecidos, es una medida de seguridad de que cada cuidado de Costa Rica de entra un servicio digno, seún lo estipulado por el artículo 42 de la constitución política. En su Informe de Calidad del Suministro Eléctrico de 2022, ARESEP presenta datos sobre la frecuencia y duración de las interrupciones del servicio en diferentes circuitos, con un número de abonados, de distribución eléctrica por empresa.

Analizar estos datos permite identificar patrones de comportamiento en la red de distribución, para mejorar la eficiencia de este servicio público fundamental año tras año. Para poder analizar todos los datos y la relación entre las métricas dadas por la ARESEP, es útil el uso los algoritmos de agrupamiento, como KMeans y DBSCAN. Ofreciendo una opción robusta para identificar grupos o patrones dentro de los datos.

Este proyecto aplica los algoritmos de KMeans y DBSCAN a los datos proporcionados en el informe de ARESEP, principalmente al Anexo A, con el objetivo de identificar agrupaciones significativas basadas en tres métricas clave: el número de abonados, la duración promedio de interrupciones (DPIR), y la frecuencia promedio de interrupciones (FPI). Estas métricas proporcionan una visión clara desempeño de los circuitos eléctricos por empresa y así evaluar su calidad.

Demostrando cómo los algoritmos de agrupamiento, aplicados a los datos de calidad del suministro eléctrico, pueden proporcionar datos valiosos para la optimización de la red de distribución eléctrica en Costa Rica a futuro, ayudando a la ARESEP y a las empresas eléctricas a identificar y corregir problemas de calidad en el servicio.

2. Procedimiento

2.1. Selección del informe de la ARESEP

El presente proyecto se enfoca en el análisis de la calidad de la energía mediante el agrupamiento de circuitos de distribución eléctrica. La Autoridad Reguladora de Servicios Públicos (ARESEP) es la entidad encargada de evaluar esta calidad, dado que la electricidad es un servicio público en Costa Rica.

Para realizar este análisis de calidad, se selecciona el Informe de Calidad del Suministro de Electricidad del 2022 [1]. Debido a que estos datos tienen dos años de antigüedad, ya han sido procesados y actualizados, obteniendo así la versión más reciente disponible.

A partir de este informe, se presta especial atención a las tablas de los anexos A y B, que contienen los datos de los circuitos con frecuencia promedio por abonado y duración promedio de las interrupciones superior a los límites normativos, y los circuitos con frecuencia promedio por abonado y duración promedio de las interrupciones igual o inferior a los valores normativos, respectivamente.

2.2. Extracción de datos del informe

2.2.1. PDF a Markdown

En las páginas 69 y 70 del informe de la ARESEP [1], se encuentran las tablas mencionadas en la sección anterior. La idea es extraer estas tablas y convertirlas a formato Markdown en dos archivos separados: `AnexoA.md` y `AnexoB.md`.

Dado que el proyecto está diseñado para ser mostrado en GitHub, Markdown presenta el texto de forma minimalista y estética, permitiendo que cualquier persona que acceda al repositorio pueda ver estos datos de manera sencilla. La herramienta utilizada para convertir estos datos de un documento PDF a Markdown fue ChatGPT modelo 4. El procedimiento utilizado para solicitarle este trabajo a ChatGPT fue el siguiente:

1. Utilizando la herramienta de impresión a PDF, se seleccionan únicamente las páginas 69 a 76 del informe de la ARESEP [1]. Estas se dividen en dos archivos PDF: `AnexoA.pdf` y `AnexoB.pdf`.

2. Los archivos PDF se envían a ChatGPT con el siguiente prompt: *Los PDF contienen el Informe de la calidad del suministro de electricidad. Quiero que extraigas esta información en una tabla en formato texto de Markdown, sin omitir ningún dato.*

3. La información proporcionada por ChatGPT se guarda en los archivos `AnexoA.md` y `AnexoB.md`, respectivamente.

2.2.2. Markdown a CSV

La herramienta para aplicar los algoritmos de agrupamiento será Python. Por ello, es necesario convertir la información de Markdown a CSV para poder utilizar herramientas como `Pandas`. Para lograr esto, se solicitó a Matlab un código en Python, específicamente una clase (más información en 2), que transforme estos archivos Markdown a CSV.

Ejecutando el siguiente código de python, en `main.py`, se realiza la conversión a .csv:

```
# ---------------------- # Convierte de Markdown a CSV # ---------------------- #
# Determinar la ruta absoluta del archivo markdown
name_md = os.path.join(current_dir, "..", "Datos", "AnexoA.md")

# Convertir de Markdown a CSV
md_to_csv = MarkdownToCSV(name_md)
name_csv = md_to_csv.convert()
```

Listing 1: Convierte de Markdown a CSV

El funcionamiento consiste en extraer la ruta donde se encuentra el archivo Markdown (que contiene el título y la tabla). Luego, se crea un objeto que recibe la dirección del archivo Markdown y, utilizando el método `md_to_csv.convert()`, se genera un archivo CSV en la misma ubicación donde se encuentra el archivo Markdown.

2.2.3. Limpieza del CSV al pasar a Dataframe

Para manipular un archivo CSV en Python, una opción es utilizar la librería Pandas y transformar el CSV en un dataframe. Uno de los problemas en la evaluación son los datos imprecisos, lo cual se refleja en el informe de la ARESEP al tener filas con datos nulos. Por ello, se utiliza el método `dropna()` de Pandas para eliminar estos casos y trabajar cómodamente posteriormente.

2.3. Interpretación general de los datos

En el Cuadro 1 se presentan las primeras cinco filas del archivo AnexoA.csv. En él se puede observar el nombre de la empresa, el circuito donde se realizó la evaluación, y los conceptos de *Abonados*, *DPIR* y *FPI*. Esto se repite en la AnexoB.csv.

Antes de seleccionar y aplicar un algoritmo de agrupamiento a estos archivos .csv, es necesario comprender el significado de cada columna. Tras una investigación proactiva, se encontraron las definiciones de los conceptos en el Informe sobre calidad del suministro eléctrico de 2013 de la ARESEP:

1. **Abonados:** se refiere al número total de usuarios que están conectados al sistema eléctrico y reciben el servicio de electricidad [2].

Empresa	Circuito	Abonados	DPIR	FPI
CNFL	BRASIL-CIUDAD COLON	8430	17.04	19.85
CNFL	CAJA-INDUSTRIAS	2189	16.36	11.93
CNFL	ELECTRIONA-MONTANA	821	15.90	10.15
CNFL	BARVA-CIPRESAL	3677	15.76	7.54

Cuadro 1: Primeras 5 filas de los Datos de Circuito de AnexoA.csv

2. **DPI (Duración Promedio de Interrupción):** mide el tiempo promedio que los usuarios experimentan interrupciones en el suministro eléctrico durante un período específico. *Se calcula sumando la duración total de todas las interrupciones y dividiéndola entre el número total de usuarios afectados* [2].

$$DPI = \frac{duracion\_total\_interrupciones}{\#total_usarios\_afectados}$$

3. **FPI (Frecuencia Promedio de Interrupción):** indica la cantidad promedio de interrupciones que experimentan los usuarios en un período determinado. *Se calcula dividiendo el número total de interrupciones entre el número total de usuarios* [2].

$$FPI = \frac{\#total\_interrupciones}{\#total\_usarios\_afectados}$$

Con una comprensión clara de la información que proporciona cada columna, se procede a analizar la primera fila del Cuadro 1. De esta se extrae la siguiente información:

- El circuito Brasil-Ciudad Colon, gestionado por la empresa Compañia Nacional de Fuerza y Luz (CNFL), cuenta con 8430 usuarios. Cada usuario experimento 19.85 interrupciones que, en promedio, duraron en conjunto 17.04 minutos.

2.4. Aplicación de algoritmos de agrupamiento

2.4.1. ¿Por qué se eligió Kmeans y DBSCAN?

La elección entre K-means y DBSCAN, en lugar de otros algoritmos disponibles, fue en parte aleatoria y también para responder a una pregunta personal sobre cuándo es conveniente seleccionar uno sobre el otro.

Ambos algoritmos se enfocan en dividir el conjunto de datos en clusters, maximizando la similitud dentro de los clusters y minimizando la similitud entre ellos. Además, ambos utilizan la distancia euclidiana para determinar la cercanía entre los datos y asignarlos a clusters. La diferencia principal es que K-means usa esta distancia para medirla hacia un centroide, mientras que DBSCAN la utiliza para medir la distancia entre puntos y evaluar la densidad local [3].

Por lo tanto, este proyecto no solo proporcionará un análisis de los datos del informe de la ARESEP [1], sino que también permitirá distinguir entre K-means y DBSCAN como herramientas de agrupamiento similares pero con enfoques diferentes, como una hipótesis. Esto se puede ver en la investigación *Comparison of Dbscan and K-means clustering methods in the selection of representative clients for a vehicle routing model*, donde Villalba & Rotta (2020), que demostraron que el el método de agrupamiento Dbscan selecciona eficazmente clientes más representativos para un modelo de enrutamiento de vehículos, mientras que el método de agrupamiento K-means es más eficaz para seleccionar clientes en función de la densidad [3]; donde se deja claro que no se arrojan los mismos resultados.

2.4.2. Implementación en código Kmeans y DBSCAN

El enunciado del proyecto incluye código para el uso de distintos algoritmos de agrupamiento [4], creado por diversos autores. Utilizando estos códigos de ejemplo y algunos desarrollados previamente por mí, se procedió a crear dos clases en Python: `KMedias` 3 y `DBSCAN_3D` 4. El usar Programación Orientada Objetos (POO) permite versatilidad, flexibilidad, mejor visibilidad y reutilización del código, por ello la decisión de utilizar, a primera vista, una implementación más compleja de estos algoritmos

Explicación de los métodos de cada clase:

- `setDataFrame(dataframe)`: sirve para ingresar los datos del CSV, previamente transformados en un dataframe utilizando la librería Pandas de Python. La información de este dataframe se guarda en un atributo privado llamado `dataframe`, que será utilizado en otros métodos.
- `__verificador_columnas(columnas)`: encargado de verificar que las columnas ingresadas, al momento de utilizar otro método graficador o sintonizador, existan dentro del dataframe proporcionado.
- `mostrar_clusters(colum1, colum2, colum3, ...)`: varia la métricas a ingresar dependiendo de si se usa DBSCAN o KMeans, pero su objetivo es el de mostrar la ubicación y la cantidad de elementos de cada cluster.

- **KMedias**:
  - `metodo_elbow(colum1, colum2, colum3)`: permite ingresar hasta un máximo de tres columnas y emplea el método del codo para determinar manualmente el número óptimo de clusters que se utilizarán en otros métodos de graficación.
  - `grafica_KMeans(colum1, colum2, k_clusters, xlabel=, ylabel=)`: dadas las columnas, el número de clusters y las etiquetas para los ejes x e y, grafica los clusters en un plano bidimensional, diferenciándolos por color.
  - `grafica_KMeans_3D(colum1, colum2, colum3, k_clusters, xlabel=, ylabel=, zlabel=)`: funciona de manera similar al método anterior, pero grafica los clusters en un espacio tridimensional.
- **DBSCAN_3D**:
  - `__realizar_DBSCAN(columnas, eps=0.5, min_samples=5)`: dado un valor de `eps` y `min_samples`, ejecuta el algoritmo DBSCAN de Python en las columnas seleccionadas del dataframe para graficar.
  - `estimar_eps(columnas, min_samples=5)`: dado un valor de `min_samples` y las columnas de un dataframe, muestra un gráfico de la distancia KNN. Con la ayuda de esta gráfica, y posicionando el cursor en el punto de inflexión de la curva, se selecciona un valor para `eps`.
  - `grafica_DBSCAN_3D(colum1, colum2, colum3, xlabel=, ylabel=, zlabel=, eps=0.5, min_samples=5)`: dado los parámetros de `eps`, `min_samples`, las columnas del dataframe y los nombres de los ejes, genera una gráfica tridimensional con los clusters encontrados.

El parámetro `eps` define el radio máximo de búsqueda alrededor de cada punto para considerar a sus vecinos. Por su parte, `min_samples` establece el número mínimo de puntos que deben estar dentro del radio `eps` para formar un cluster (un valor que es el doble de los parámetros a ingresar).

Para más información del código, puede ir a la sección de Anexos.

3. Resultados

Todas las secciones a continuación, incluyendo la aplicación de algoritmos y la identificación de métricas de cada algoritmo, se basan en el análisis de la tabla del Anexo A. Por lo tanto, se utilizó únicamente el archivo AnexoA.csv. La razón para no usar la tabla del Anexo B fue una indicación del profesor del curso, Marvin Coto.

Se informa que el archivo AnexoA.csv ha sido limpiado utilizando el método `dropna()` después de convertirlo a un dataframe. Además, se ha creado un nuevo archivo denominado AnexoA_normalize.csv, que también ha sido limpiado al pasarlo a un dataframe, para comparar los datos normalizados con los datos originales. A partir de estos dataframes es que se aplicarán los algoritmos de agrupamiento.

3.1. Agrupamiento KMeans

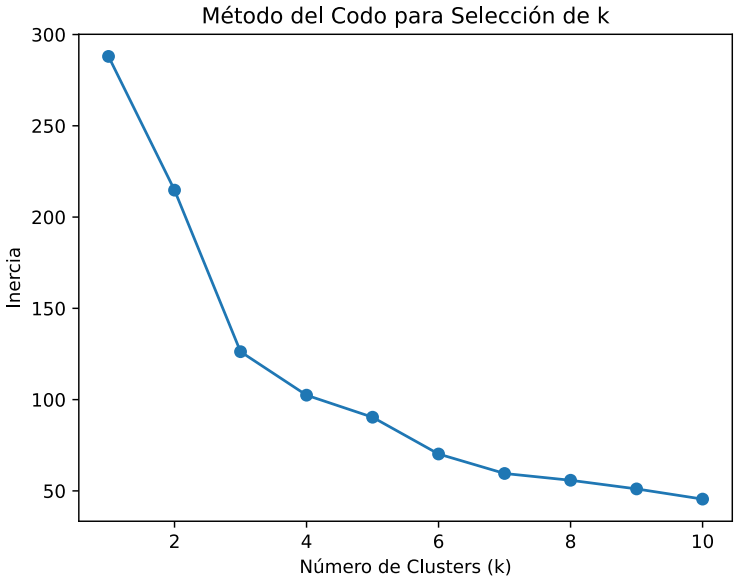
Se crean dos objetos con la clase KMedias, se ingresa el dataframe con el método setDataFrame() y se procede a trabajar con el.

De adelanto, se puede observar como hay diferencias en lso clusters comparando las Gráficas 3 y 7, incluso hay un cambio en la cantidad de elementos que tiene cada cluster al comparar los Cuadros 2 y 3.

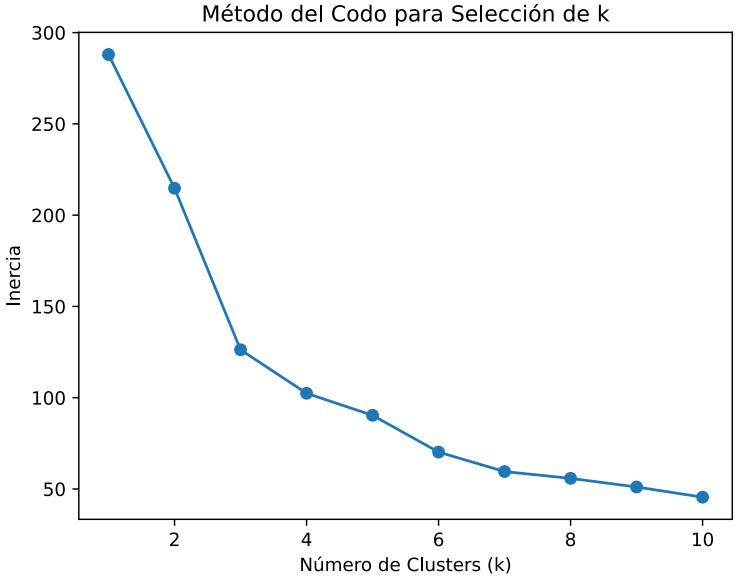
3.1.1. Método del códo

Uno de los parámetros más importantes del algoritmo KMeans es el número de clusters. Sin este parámetro, se podrían tener clusters faltantes o sobrantes, lo que afectaría el análisis correcto. Para determinar el número adecuado de clusters, se utiliza el método del codo, que, como se muestra en la figura 1, consiste en seleccionar el punto donde la gráfica empieza a desacelerar, es decir, el codo.

En esta ocasión, se ha escogido un valor de clusters de $k = 3$ para datos brutos y normalizados. No hay diferencias, en esta ocasión, al momento de definir el número de clusters.



(a) AnexoA datos brutos



(b) AnexoA datos normalizados

Figura 1: Método del codo aplicado para calcular el número de clusters con el AnexoA.

3.1.2. KMeans 3D

Utilizando un número de clusters (K = 3), se obtienen dos resultados diferentes. Además, se proporciona la ubicación del centroide de cada cluster y el número de elementos que lo conforman, tanto para los datos brutos como para los datos normalizados del Anexo A.

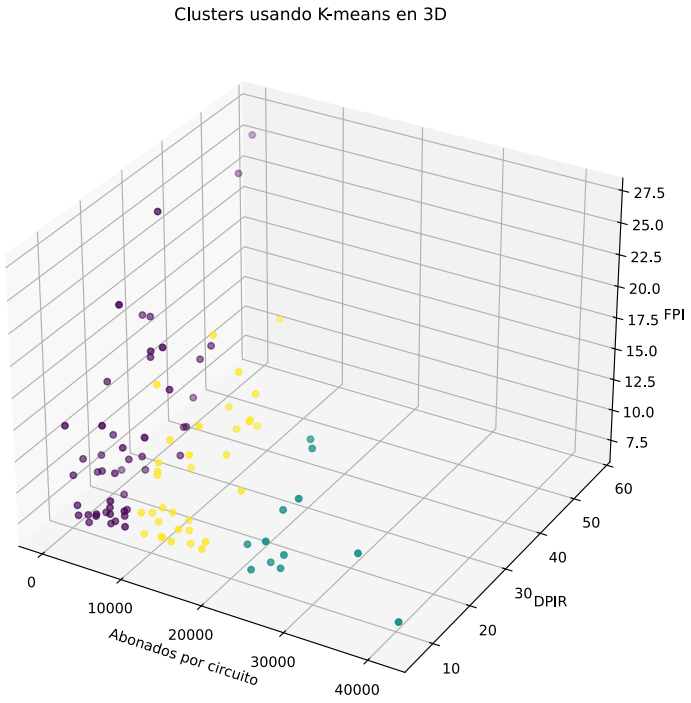


Figura 2: Agrupamiento KMeans utilizando los datos brutos de Anexo A.

Información de la ubicación de los centroides y cantidad de elementos de la imagen 2, se evidencia en el cuadro 2:

Cluster	Centroide	Número de elementos
1	[4317.16, 16.8514, 12.3182]	50
2	[2.69594167e+04, 12.3625, 10.0317]	12
3	[1.3025e+04, 15.4774, 11.4706]	34

Cuadro 2: Datos de los clusters de la Gráfica 2, algoritmo KMeans aplicado con datos brutos de Anexo A..

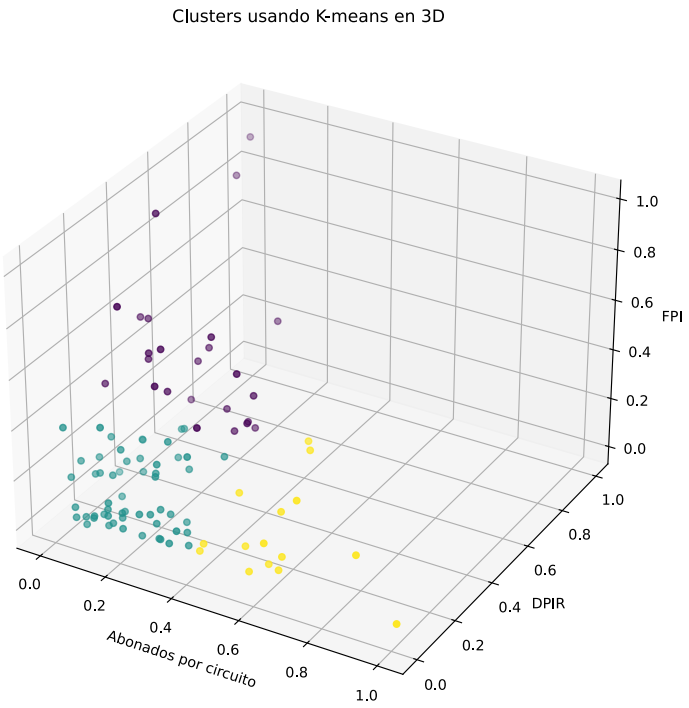


Figura 3: Agrupamiento KMeans utilizando los datos normalizados de Anexo A

Información de la ubicación de los centroides y cantidad de elementos de la imagen 3, se evidencia en el cuadro :

Cluster	**Centroide**	**Número de elementos**
1	[0.20448241, 0.38399686, 0.52217523]	25
2	[0.16951409, 0.11740703, 0.12623184]	56
3	[0.60899353, 0.11067923, 0.13830144]	15

Cuadro 3: Datos de los clusters de la Gráfica 3, algoritmo KMeans aplicado con datos normalizados de Anexo A.

3.2. Agrupamiento DBSCAN

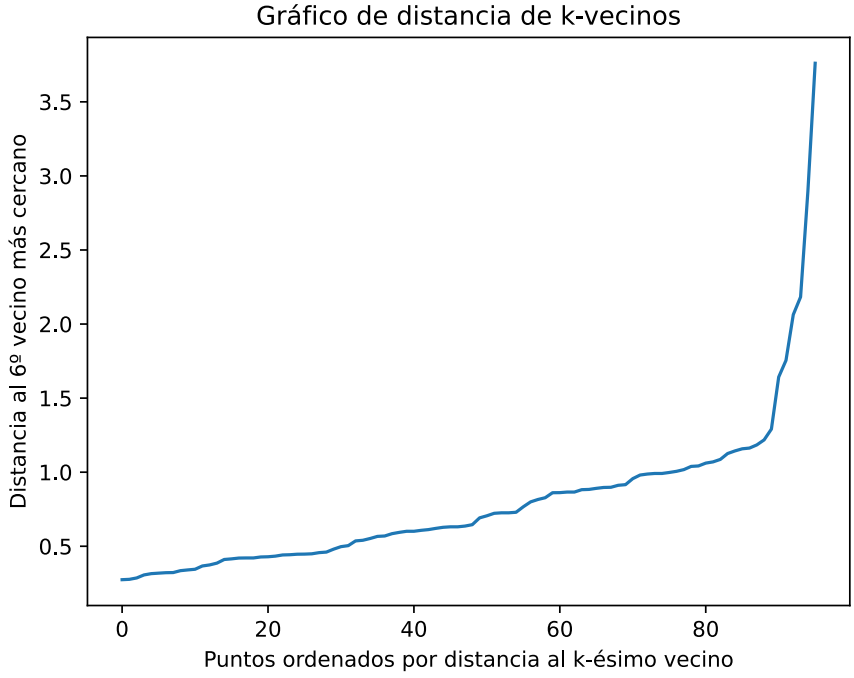
De la misma forma que en la sección anterior, se crean dos objetos con la clase DCSVCAN_3D, se ingresa el dataframe con el método setDataFrame() y se procede a trabajar con el.

En este sistema de agrupamiento no hubo cambios al realizar el algoritmo DBSCAN con datos normalizados y brutos para este caso de estudio.

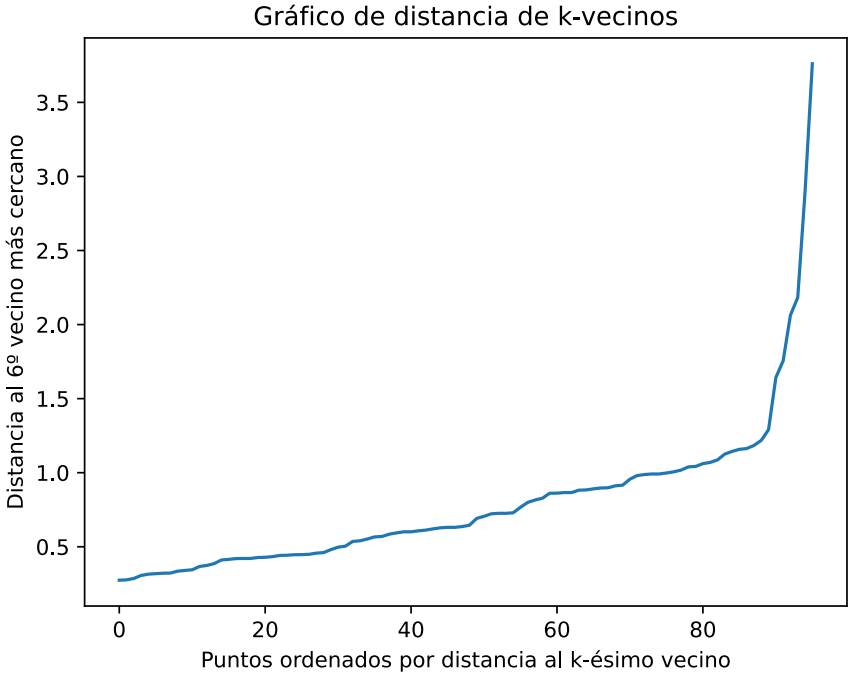
3.2.1. Método de identificación eps con KNN

Para un uso correcto del algoritmo DBSCAN, es crucial seleccionar importante los parámetros eps y min_samples, como se explicó en la sección de Procedimiento. Utilizando el método de KNN, con un min_samples = 6 (el doble del número de características ingresadas), se obtiene la figura 5.

Para determinar el valor correcto de eps, se selecciona un punto en la curvatura final de la gráfica, antes de que el comportamiento se torne asintótico en el eje y. El valor escogido para eps fue 1.23, para ambos casos brutos y normalizados. Tampoco hay una diferencia al momento de calcular el epsilon.



(a) AnexoA datos brutos



(a) AnexoA datos normalizados

Figura 5: Método KNNN aplicado el valor de EPS con el AnexoA.

3.2.2. DBSCAN 3D

Habiendo definido los parámetros eps = 1,23 y min_samples = 6, utilizando como características Abonados, DPI y FPI; se procede a realizar las gráficas de haber aplicado el algoritmo DBSCAN con las variables con datos brutos y normalizados del Anexo A.

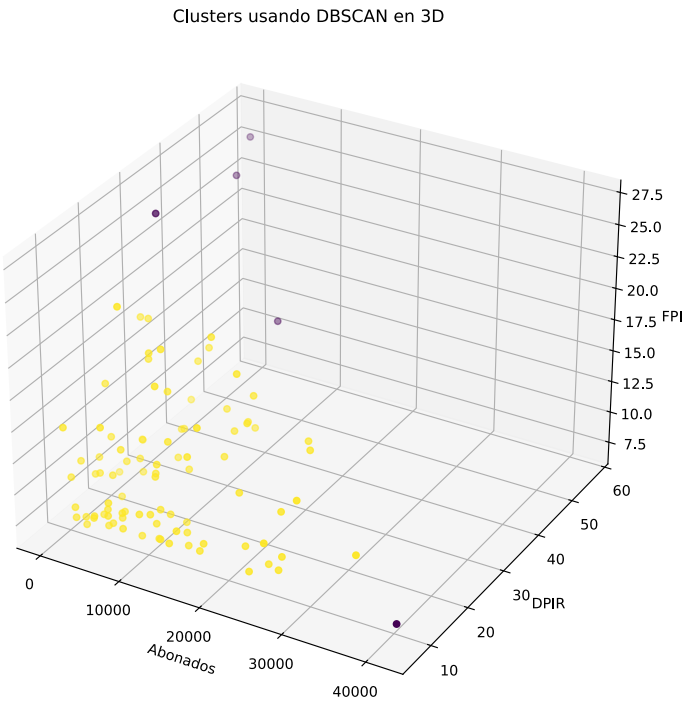


Figura 6: Agrupamiento DBSCAN utilizando los datos brutos de Anexo A.

Información de la ubicación de los centroides y cantidad de elementos de la imagen 6, se evidencia en el cuadro 4:

Cluster	**Centroide**	**Número de elementos**
1	[-0.00874908, -0.1247225, -0.10588123]	91
Ruido	-	5

Cuadro 4: Datos de los clusters y ruido extraigos de la Gráfica 6, algoritmo DBSCAN aplicado con datos brutos de Anexo A.

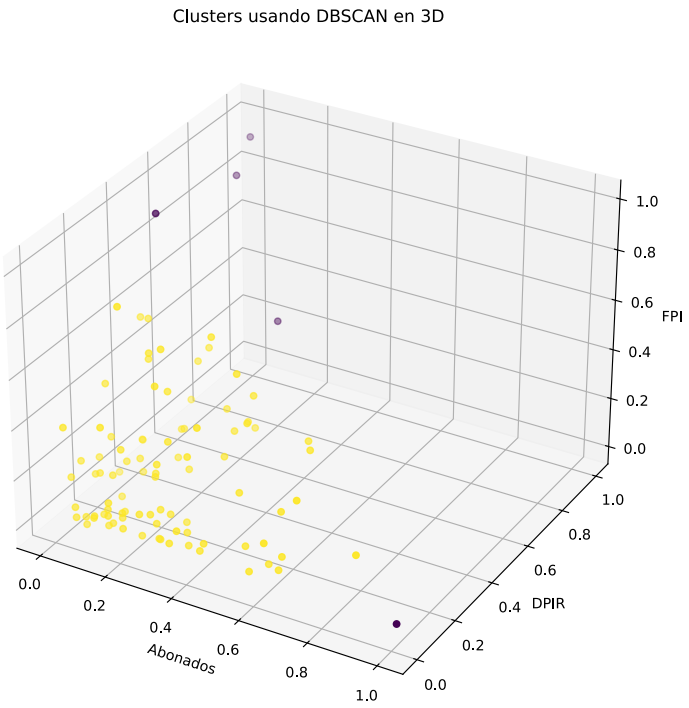


Figura 7: Agrupamiento DBSCAN utilizando los datos normalizados de Anexo A.

Información de la ubicación de los centroides y cantidad de elementos de la imagen 7, se evidencia en el cuadro 5:

Cluster	**Centroide**	**Número de elementos**
1	[-0.00874908, -0.1247225, -0.10588123]	91
Ruido	-	5

Cuadro 5: Datos de los clusters y ruido extraigos de la Gráfica 7, algoritmo DBSCAN aplicado con datos normalizados de Anexo A.

4. Análisis de resultados

4.1. Informe de calidad del suministro de electricidad de la ARESEP

Es común escuchar en la mayoría de los costarricense, incluyéndome, decir que el servicio brindado por una empresa es ineficientes y tiene mucho margen de mejora. Eso toma más fuerza con las situaciones que ha presentado el ICE durante este gobierno, y la tentativa mensual que a causa de las lluvias el suministro eléctrico se verá pausado; y qué podría hacer la ARESEP ante un ajento externo como lo son las lluvias.

Lo cierto es que si nos remontamos a los inicios del siglo XX, antes de la existencia del ICE, el suministro eléctrico era dado a Costa Rica mediante empresas de Estados Unidos, siendo participe Minor Cooper Keith [5]. Este era un servicio, que a palabras de quienes tuvieron familiares que vivieron esa experiencia, horrible y hostil hacia el consumidor, que pasado una cierta cantidad de amperaje una cuchilla bajaba y dejaba sin luz las casas. A partir de estas situaciones, el país en un movimiento de proteccionismo funda el Servicio Nacional de Electricidad, que posteriormente pasa a ser la Autoridad Reguladora de Servicios Públicos (ARESEP) posterior a la creación del ICE.

El informe de calidad del suministro de electricidad, independientemente del año, por parte de la ARESEP, de muestra que el paso del tiempo ha sido agradecido con Costa Rica y sus ciudadanos. Que mediante la constante aplicación del artículo 46 de la constitución política [6], cada persona puede acceder a electricidad en el país, exceptuando un porcentaje un pequeño de la población, y verificar mediante informes y métricas la calidad de su servicio.

Mediante este proyecto, se ha podido observar que ante todo aquellos prejuicios que se puede tener sobre el suministro eléctrico, existe una acción ordenada y constante para velar sobre la calidad del servicio, siendo este evaluado desde distintas perspectivas y dando resultados sobre lo obtenido[1]. Siendo ahora una obligación de cada compañía el aporte de estas métricas. Si bien hay detalles a mejorar, que la misma ARESEP reconoce, como que el 30.7 % de las interrupciones no tuvieron identificación [1]; es un proceso en mejora. Aclarando que este proyecto funciona como un análisis queja para a futuro exista un nuevo sesgo del costarricense, que a partir de un mejor servicio, se olvide del pasado trajico que se tuvo.

4.2. Agrupamiento obtenido a partir del algoritmo KMeans

Aunque el algoritmo KMeans es de aprendizaje no supervisado, la selección de las variables Abonados, DPI y FPI tenía como objetivo encontrar un punto de partida para identificar aquellos casos con una cantidad y duración de interrupciones más relevantes. Esto permitiría posteriormente seleccionar

un grupo (cluster) específico como caso de estudio.

4.2.1. Datos brutos

Como se puede observar en la Figura 2, que muestra tres grupos en una gráfica tridimensional según el parámetro K establecido, los clusters parecen depender en mayor medida de la métrica de Abonados por circuito. Tomando la afirmación, como punto de partida, se podría decir que:

- **Cluster 1 (morado)**:
  - Presenta la mayor cantidad de elementos 2 y tiene la menor presencia de abonados. Esto puede indicar que agrupa en su mayoría a aquellas pequeñas y medias empresas y sus indices de DPI y FPI obtenidos en el año 2022.
  - El punto anterior, a parecer de ser una afirmación valida, puede resultar falsa. Utilizando la herramienta de Excel para ordenar de menor a mayor por abonado las empresas, resulta que muchas son pertenecientes al ICE y CNFL, ambas siendo las más grandes del país.
  - Un mejor razonamiento, es que, este cluster verdaderamente contiene los resultados de DPI y FPI de los circuitos más pequeños en cuanto a cantidad de abonados.
- **Cluster 2 (turquesa)**:
  - Presenta la menor cantidad de elementos 2 y tiene la mayor presencia de abonados, siendo el contrario del cluster 1. Utilizando la misma analogía, este cluster engloba a los circuitos con mayor cantidad de abonados.
  - Algo a considerar de este cluster, es que la mayoría de datos presentarían un DPI y FPI bajo (no el mejor).
  - Se podría afirmar que este cluster sería el menos prioritario de los 3, pero hay más variables externas que no estan dentro de la tabla que se puede considerar Una hipotesis, es que al tener los circuitos con mayor cantidad de abonados, es que que más reparación y atención tiene durante el año.
- **Cluster 3 (amarillo)**:
  - Presenta la cantidad media de abonados (Cuadro 2). Cuenta con los circuitos con una población medía, y es el segundo grupo más grande.

A partir de la información de estos clusters, se puede determinar que el algoritmo KMeans, aplicado a los datos en bruto del Anexo A, decidió agrupar principalmente según la cantidad de abonados por circuito.

4.2.2. Datos normalizados y comparativa con datos brutos

Los tres grupos formados con los datos normalizados son diferentes a los de la sección anterior, como se observa en la Figura 3. Esto era de esperarse debido a la ubicación de los centroides, ya que los datos están normalizados entre 0 y 1. Sin embargo, lo relevante es que la cantidad de elementos y la forma de los clusters se han modificado. Se procede a analizar cada cluster:

- **Cluster 1 (morado)**:
  - A comparación de los resultados KMeans pasado, este cluster ya no cuenta con la mayor cantidad de elementos 3. Además que no presenta el mismo comportamiento de agrupación por la cantidad de abonados, por lo que esta métrica ha perdido peso en el algoritmo. Haciendo una comparación con las siguientes métricas, no hay ninguna que destaque de sobre otra.
  - No teniendo una métrica predominante, este cluster ahora parece englobar a todos aquellos casos cuya cantidad de DPI y FPI sea mayor, en en rango de 0 a 0.4 de abonados.
  - La información proporcionada por este cluster destaca datos más relevantes que su contraparte morada en los datos brutos, que solo mostraba los circuitos con menor cantidad de abonados. Este cluster engloba los peores casos de desempeño.
- **Cluster 2 (turquesa)**:
  - La comparación con su contraparte, no tiene sentido, ya que ahora muestran información totalmente diferente. Con una ubicación y cantidad de elementos muy diferente, siendo esta la que más elementos tiene ahora 3.
  - Este cluster se puede ver más, como la contraparte del cluster morado de la misma gráfica. Mostrando los mejores datos de DPI y FPI, con un rango de abonados de 0 a 0.4 abonados.
- **Cluster 2 (amarillo)**:
  - Mismo análisis que el cluster turquesa de la misma gráfica. Mostrando información muy similar también, pero en este caso los mejores resultados de DPI y FPI para una cantidad de abonados mayor.

Obteniendo varias diferencias en cuanto al enfoque de la información, a pesar de no tener un resultado diferente en la estimación del número de clusters.

4.2.3. Aporte de la normalización a los datos en KMeans

La decisión de normalizar los datos, al principio, fue una recomendación del profesor del curso, que posterior a los resultados obtenidos ha sido una desición que cambio el enfoque de la información que

proporcionó el algoritmo KMeans.

Se podría decir que la normalización redujo la influencia del vector de la métrica de abonados, permitiendo así la formación de clusters con un aporte más significativo de métricas como DPI y FPI, que indican la cantidad y duración de las interrupciones en conjunto. Esto se pudo deber a que el valor numérico de los abonados era considerablemente más alto que el valor de DPI y FPI.

Ahora los clusters parece ser divida por la cantidad de abonados con peores métricas de DPI y FPI, y lo contrario.

4.3. Agrupamiento obtenido a partir del algoritmo DBSCAN

Al analizar los resultados obtenidos con el algoritmo DBSCAN, utilizando una densidad de clústeres de $eps = 1{,}23$ y una cantidad mínima de elementos para formar grupos $min\_samples = 6$, se puede apreciar cómo este algoritmo hace honor a su nombre al español: Clustering Espacial Basado en Densidad de Aplicaciones con Ruido. Dado que en la Figura 6, se identifica un **Cluster (amarillo)** que agrupa todos los casos "normales" según el algoritmo, y se detectan **5 datos como Ruido**.

Dado que cada elemento de Ruido no rerpesenta lo mismo, es necesario hacer un análisis utilizando el Informe de Calidad del Suministro Eléctrico del 2022 [1]. Los 5 datos fueron clasificados, mediante la figura 6, como:

- **Mejor relación entre abonados, DPI y FPI:** circuito LEESVILLE-RIO FRIO, empresa ICE, con 41,338 abonados, DPI = 7.64 y FPI = 7.47.
- **Peor relación entre abonados, DPI y FPI:** circuito LIBERIA-URBANO, empresa ICE, con 11,245 abonados, DPI = 41.08 y FPI = 16.46.
- **Peor caso en relación a DPI y FPI:** circuito PAILAS-HOSPITAL, empresa ICE, con 12 abonados, DPI = 57.28 y FPI = 25.50.
- **Segundo peor caso en relación a DPI y FPI:** circuito GARITA-PARRITA, empresa ICE, con 1,912 abonados, DPI = 49.67 y FPI = 24.47.
- **Peor caso de FPI:** circuito RIO CLARO-CHACARITA, empresa ICE, con 3,088 abonados, DPI = 26.87 y FPI = 27.00.

El ruido puede tener tanto aspectos positivos como negativos, por lo que no se puede hacer una afirmación generalizada sobre lo que representa. Por ejemplo, no se puede concluir que la mejor o peor empresa es el ICE, solo observando el Ruido. Además, estos casos no son necesariamente los más

relevantes, ya que, aunque incluyen los peores casos, son una minoría. Esto plantea interrogantes como: ¿por qué el circuito con mayor número de abonados tiene un desempeño tan bueno en comparación con el circuito con menor cantidad de abonados que tiene un desempeño muy malo?

4.3.1. Aporte de la normalización a los datos en DBSCAN

Los datos normalizados no fueron utilizados para el análisis, debido a que los resultados, al compararse con los datos brutos, fueron los mismos. Eso se puede observar comparando las Figuras 6 y 7, y los Cuadros 4 y 5. Tampoco hubo variación en el cálculo del epsilon.

Un aspecto interesante, es que no hubo una variación en la ubicación del centrode. No se pudo encontrar una justificación clara para esto, pero se aclara que no fue un error de código debido a que la Figura 7 normalizada si tuvo cambios en el valor de sus ejes.

4.4. Comparación entre KMeans y DBSCAN

De manera personal, se cumplió el objetivo de observar las diferencias entre KMeans y DBSCAN. KMeans resultó ser una herramienta útil para formar clústeres basados en el desempeño del DPI según la cantidad de abonados. Por otro lado, DBSCAN permitió identificar los mejores y peores casos que se salían de lo normal, clasificándolos como ruido.

Aunque se reconoce que el resultado podría mejorar, sería más efectivo **utilizar primero DBSCAN** para separar los datos de ruido, **como un detector de anomalias**, y luego aplicar KMeans a los datos ya procesados. De esta manera, se hubieran podido formar más o menos clusteres, con información más detallada en relación a DPI y FPI con KMeans.

4.5. Comparación de los resultados con otro estudio

Bule y Nair (2023) realizaron el estudio titulado *Smart Meter Customer Data Synchronization Technique by KMean Clustering* [7], en el cual revisaron métodos de clasificación estocástica (algoritmos de agrupamiento) aplicados al consumo de energía de los clientes mediante medidores inteligentes, utilizando técnicas de KMeans. El objetivo era proporcionar información, como la demanda de carga de los clientes, a los operadores, utilizando grandes conjuntos de datos en escalas de tiempo pequeñas, para ofrecer un servicio más adecuado a la demanda eléctrica en Australia.

Esta investigación está relacionada con este proyecto en términos de la evaluación de la calidad del suministro eléctrico, aunque con un enfoque diferente. Se centra en medir el desempeño de cada compañía en relación con la cantidad de interrupciones y su duración por cliente en una determinada compañía y circuito.

5. Conclusiones

- La aplicación del algoritmo KMeans al análisis de la calidad del suministro eléctrico permitió identificar tres grupos principales de circuitos de distribución eléctrica, al usar datos normalizados, basado en número de abonados, DPI y FPI. Mostrando que los circuitos se puede separar entre cantidad de abonados, y luego separarse por el valor de DPI y FPI. Así visualizando aquellos circuitos que con cantidad de abonados diferentes, tienen desempeños contrarios..
- La normalización de los datos modificó significativamente la composición de los clusters obtenidos mediante KMeans, reduciendo la influencia de la variable .ªbonadosz destacando circuitos con características extremas en términos de DPIR y FPI. Esto resalta la importancia de la normalización para una evaluación más equilibrada de la calidad del servicio eléctrico, en casos que hay mucha predominancia en una métrica.
- La aplicación del algoritmo DBSCAN permitió detectar de manera eficaz tanto el mejor como los peores casos en términos de calidad de suministro eléctrico, clasificando estos casos como ruido". Identificando aquellos circuitos con comportamientos fuera de la normal, que no se agrupan con otros circuitos.
- No hubo diferencias significativas en los resultados del agrupamiento al aplicar DBSCAN a los datos normalizados en comparación con los datos originales. Este caso fue más importamte realizar el análisis con los datos brutos del Anexo A.
- La comparación entre los algoritmos KMeans y DBSCAN mostró que: KMeans es más efectivo para clasificar grupos basados en características generales del desempeño del circuito por abonados, mientras que DBSCAN es más adecuado para identificar casos inusuales (datos anómalos).
- Una combinación de ambos algoritmos podría proporcionar una estrategia más completa para analizar la calidad del suministro eléctrico. Utilizar primero DBSCAN para identificar y excluir datos de ruido, seguido de KMeans para clasificar los datos restantes, podría mejorar la precisión de los agrupamientos.
- El uso de técnicas de agrupamiento aplicado al Informe de Calidad del Suministro Eléctrico de ARESEP permite una mejor identificación de problemas de desempeño, facilitando una toma de decisiones a futuro más informada para el mejoramiento de la calidad de servicio de cada empresa.

6. Anexos

6.1. Enlace al repositorio:

El proyecto fue creado para ser publicado en GitHub, es por ello que se habilita el link para quien deseé visitarlo https://github.com/yehohnathan/ie0435/tree/main/Proyecto1. **El código fue testeado en Windows.**

Las instrucciones de ejecución son las siguientes:

1. Realizar un `git clone` al repositorio en la dirección deseada por el usuario.
2. Posicionarse en la dirección `C:\..\ie0435\Proyecto1>`.
3. Ejecutar `make` en la terminal:

```
C:\..\ie0435\Proyecto1> make
```

4. Si no tiene `make` instalado o tiene algún problema, puede ejecutar el `main`:

```
C:\..\ie0435\Proyecto1\scr> py main.py
```

6.2. Clase MarkdownToCSV

Esta clase fue solicitada a ChatGPT modelo 4 y personalizada por mí para prevenir errores. Sus métodos se encargan de procesar tablas en archivos Markdown con el formato (Título, Tabla).

```python
# ---------------------- # Clase solicitada a ChatGPT # ---------------------- #
# ---------------- # Se importan las librer as necesarias # ----------------- #
import pandas as pd
from sklearn.preprocessing import MinMaxScaler


# ------------ # Clase para convertir archivos markdown a CSV # ------------- #
class MarkdownToCSV:

    def __init__(self, file_path):
        self.file_path = file_path  # Almacena la ruta del archivo
        self.df = None  # Inicializa el DataFrame como None

    # -------------------- # Leer el archivo markdown # --------------------- #
```

```
    def read_markdown(self):
        # Imprime un mensaje indicando que intentar  abrir el archivo
        print(f"Intentando abrir markdown: {self.file_path}")
        try:
            # Abre el archivo en modo de lectura con codificaci n UTF-8
            with open(self.file_path, 'r', encoding='utf-8') as file:
                # Lee todas las l neas del archivo y las almacena en una lista
                lines = file.readlines()
        except UnicodeDecodeError:  # Captura cualquier error de codificaci n
            print("Failed to read the file due to encoding issues." +
                  " Please check the file encoding.")
            return None  # Devuelve None si ocurre un error

        return lines  # Devuelve las l neas le das del archivo

    # ------------------ # Extraer la tabla del markdown # ------------------ #
    def extract_table(self, lines):
        if lines is None:  # Si no se leyeron l neas, devuelve None
            print("No lines to process, returning None.")
            return None

        # Filtra las l neas que contienen '|' (partes de la tabla) y
        # elimina los espacios adicionales
        table_lines = [line.strip() for line in lines if '|' in line]
        # Extrae los datos de la tabla, eliminando los espacios adicionales
        # y reemplazando comas por puntos
        table_data = []
        for line in table_lines[2:]:
            cells = line.split('|')[1:-1]
            clean_cells = [cell.strip().replace(',', '.') for cell in cells]
            table_data.append(clean_cells)

        # Extrae y limpia los nombres de las columnas desde la primera l nea
        # de la tabla
        columns = [col.strip() for col in table_lines[0].split('|')[1:-1]]
        # Crea un DataFrame con los datos extra dos
        self.df = pd.DataFrame(table_data, columns=columns)
        return self.df  # Devuelve el DataFrame creado

    # ------------ # Guardar el DataFrame como un archivo CSV # ------------- #
    def save_csv(self):
        if self.df is None:  # Verifica si el DataFrame es None
            print("DataFrame is None, cannot save to CSV.")
            return None  # Devuelve None si no hay DataFrame para guardar

        # Reemplaza la extensi n del archivo de .md a .csv
```

```
        output_csv = self.file_path.replace(".md", ".csv")
        # Guarda el DataFrame en un archivo CSV
        self.df.to_csv(output_csv, index=False)
        # Imprime la ubicaci n donde se guard  el CSV
        print(f"CSV saved to: {output_csv}")
        return output_csv  # Devuelve la ruta del archivo CSV generado

    # ----------------- # Proceso completo de conversi n # ------------------ #
    def convert(self):
        # Lee el archivo markdown
        lines = self.read_markdown()
        # Extrae la tabla desde las l neas le das
        self.extract_table(lines)
        # Guarda el contenido como CSV y devuelve la ruta del
        # archivo CSV generado
        return self.save_csv()

    # ------- # M todo para normalizar las columnas seleccionadas # --------- #
    def normalize_columns(self, csv_path):
        # Verifica si el archivo proporcionado es un CSV
        if not csv_path.endswith('.csv'):
            print("El archivo no es un CSV. Proporcione un archivo CSV para"+
                  " la normalizaci n.")
            return None

        # Carga el archivo CSV
        self.df = pd.read_csv(csv_path)

        # Convertir las columnas a tipo num rico
        self.df['Abonados'] = pd.to_numeric(self.df['Abonados'],
                                            errors='coerce')
        self.df['DPIR'] = pd.to_numeric(self.df['DPIR'], errors='coerce')
        self.df['FPI'] = pd.to_numeric(self.df['FPI'], errors='coerce')

        # Normalizar las columnas usando MinMaxScaler para el rango [0, 1]
        scaler = MinMaxScaler()
        self.df[['Abonados', 'DPIR', 'FPI']] = scaler.fit_transform(self.df[['
    Abonados', 'DPIR', 'FPI']])

        # Guardar el DataFrame normalizado con el nuevo nombre
        output_csv = csv_path.replace(".csv", "_normalize.csv")
        self.df.to_csv(output_csv, index=False)
        print(f"CSV normalizado guardado en: {output_csv}")
        return output_csv
```

Listing 2: Código MarkdownToCSV en Python

6.3. Clase KMedias

La decisión de crear una clase para KMedias se tomó con el objetivo de facilitar el uso del sistema de agrupamiento KMeans, mejorar la claridad del código principal y permitir su reutilización en diversas pruebas. La descripción del uso y funcionamiento de cada atributo y método se encuentra dentro del código. A continuación, se presenta el código de `KMedias.py` utilizado para implementar KMeans:

```python
# ---------------- # Se importan las librer as necesarias # ----------------- #
import pandas as pd                          # Manejo del CSV
import matplotlib.pyplot as plt              # Graficas
from sklearn.cluster import KMeans           # Agrupamiento K-Means
from sklearn.preprocessing import StandardScaler    # Para elbow method


# --------------- # Clase para los Sistemas de Agrupamiento # --------------- #
class KMedias:
    def __init__(self) -> None:
        self.__dataframe = pd.DataFrame()

    def setDataFrame(self, dataframe):
        # Verifica si dataframe es algo diferente a lo esperado
        if not isinstance(dataframe, pd.core.frame.DataFrame):
            raise ValueError("\nNo se ingres  un DataFrame.")
        self.__dataframe = dataframe

    def __verificador_columnas(self, columnas):
        # Verifica que columnas sea una lista y contenga m s de 2 elementos
        if not isinstance(columnas, list) or len(columnas) < 2:
            raise ValueError("Debe ingresar una lista con al menos dos" +
                             " nombres de columnas.")

        # Verifica que todas las columnas en la lista sean cadenas de texto
        for col in columnas:
            if not isinstance(col, str):
                raise ValueError("Cada nombre de columna debe ser una" +
                                 "cadena de texto.")

        # Verificar que todas las columnas existan dentro del DataFrame
        for col in columnas:
            if col not in self.__dataframe.columns:
                raise ValueError(f"La columna {col} no existe dentro del" +
                                 "DataFrame.")

    def metodo_elbow(self, colum1, colum2, colum3):
        columnas = [colum1, colum2, colum3]
        self.__verificador_columnas(columnas)
```

```
40
41          # Se seleccionan las columnas que se quieren utilizar
42          data = self.__dataframe[columnas]
43
44          # Procesamiento de los datos, para crear un ajuste multidimensional
45          scaler = StandardScaler()                    # Crea un objeto
46          scaler_data = scaler.fit_transform(data)     # Entrena los datos
47
48          # Selecci n del n mero de clusters usando el m todo Elbow
49          inercia = []
50          for k in range(1, 11):
51              kmeans = KMeans(n_clusters=k, random_state=42)
52              kmeans.fit(scaler_data)
53              inercia.append(kmeans.inertia_)
54
55          # Graficar el M todo del Codo
56          plt.plot(range(1, 11), inercia, marker='o')
57          plt.title('M todo del Codo para Selecci n de k')
58          plt.xlabel('N mero de Clusters (k)')
59          plt.ylabel('Inercia')
60          plt.show()
61
62      def grafica_KMeans(self, colum1, colum2, k_clusters,
63                         xlabel="", ylabel=""):
64          columnas = [colum1, colum2]
65          self.__verificador_columnas(columnas)
66
67          # Si los labels no se proporcionan, usar los nombres de las columnas
68          xlabel = xlabel if xlabel else colum1
69          ylabel = ylabel if ylabel else colum2
70
71          # Verifica que el n mero de clusters sea permitido
72          if not isinstance(k_clusters, int):
73              raise ValueError("El n mero de clusters debe ser entero.")
74          if k_clusters < 1:
75              raise ValueError("El n mero de clusters debe ser mayor a 1.")
76
77          # Se selecciona las dos columas que se quieren utilizar.
78          data = self.__dataframe[[colum1, colum2]]
79
80          # n_init inicializa 10 veces Kmeans con semillas aleatorias
81          # nuevo de clusters decidido por k_clusters
82          kmeans = KMeans(n_clusters=k_clusters, random_state=42, n_init=10)
83
84          # Se crea una nueva columna con el nombre de "clusters_kmeans" con los
85          # datos del entrenamiento de predicci n del kmeans (configurado)
```

```
        self.__dataframe['cluster_kmeans'] = kmeans.fit_predict(data)

        # Graficar el KMeans
        plt.scatter(self.__dataframe[colum1], self.__dataframe[colum2],
                    c=self.__dataframe['cluster_kmeans'],
                    cmap='viridis', marker='.')
        plt.title('Clusters usando K-means')
        plt.xlabel(str(xlabel))
        plt.ylabel(str(ylabel))
        plt.show()

    def grafica_KMeans_3D(self, colum1, colum2, colum3, k_clusters,
                          xlabel="", ylabel="", zlabel=""):

        columnas = [colum1, colum2, colum3]
        self.__verificador_columnas(columnas)

        # Si los labels no se proporcionan, usar los nombres de las columnas
        xlabel = xlabel if xlabel else colum1
        ylabel = ylabel if ylabel else colum2
        zlabel = zlabel if zlabel else colum3

        if not isinstance(k_clusters, int):
            raise ValueError("El n mero de clusters debe ser entero.")
        if k_clusters < 1:
            raise ValueError("El n mero de clusters debe ser mayor a 1.")

        # Se seleccionan las tres columnas que se quieren utilizar.
        data = self.__dataframe[[colum1, colum2, colum3]]

        # Realiza KMeans
        kmeans = KMeans(n_clusters=k_clusters, random_state=42, n_init=10)
        self.__dataframe['cluster_kmeans'] = kmeans.fit_predict(data)

        # Graficar en 3D
        fig = plt.figure(figsize=(10, 7))
        ax = fig.add_subplot(111, projection='3d')
        ax.scatter(self.__dataframe[colum1], self.__dataframe[colum2],
                   self.__dataframe[colum3],
                   c=self.__dataframe['cluster_kmeans'],
                   cmap='viridis', marker='o')

        ax.set_title('Clusters usando K-means en 3D')
        ax.set_xlabel(str(xlabel))
        ax.set_ylabel(str(ylabel))
        ax.set_zlabel(str(zlabel))
```

```
        plt.show()

    def mostrar_clusters(self, colum1, colum2, colum3, k_clusters):
        columnas = [colum1, colum2, colum3]
        self.__verificador_columnas(columnas)

        data = self.__dataframe[columnas]
        kmeans = KMeans(n_clusters=k_clusters, random_state=42, n_init=10)
        labels = kmeans.fit_predict(data)

        # Calcular los centroides
        centroids = kmeans.cluster_centers_

        # Mostrar cada cluster con su centroide y cantidad de elementos
        for i in range(k_clusters):
            cluster_points = data[labels == i]
            print(f"Cluster {i+1}: Centroide = {centroids[i]} con {len(
    cluster_points)} elementos.")
```

Listing 3: Código KMedias en Python

6.4. Clase DBSCAN_3D

Utilizando el mismo argumento que en la sección de la clase KMedias. A continuación, se presenta el código de `DBSCAN_3D` utilizado para implementar el algoritmo DBSCAN únicamente de 3 elementos:

```
# ---------------- # Se importan las librer as necesarias # ----------------- #
import pandas as pd                          # Manejo del CSV
import numpy as np
import matplotlib.pyplot as plt              # Gr ficas
from sklearn.cluster import DBSCAN           # Agrupamiento DBSCAN
from sklearn.preprocessing import StandardScaler    # Escalado de datos
from sklearn.neighbors import NearestNeighbors      # Calcular distancias KNN


# ----------- # Clase para los Sistemas de Agrupamiento DBSCAN # ------------ #
class DBSCAN_3D:
    def __init__(self) -> None:
        self.__dataframe = pd.DataFrame()

    def setDataFrame(self, dataframe):
        # Verifica si dataframe es algo diferente a lo esperado
        if not isinstance(dataframe, pd.core.frame.DataFrame):
            raise ValueError("\nNo se ingres  un DataFrame.")
        self.__dataframe = dataframe
```

```
20
21      def __verificador_columnas(self, columnas):
22          # Verifica que columnas sea una lista y contenga 3 elementos
23          if not isinstance(columnas, list) or len(columnas) != 3:
24              raise ValueError("Debe ingresar una lista con tres" +
25                               "nombres de columnas.")
26
27          # Verifica que todas las columnas en la lista sean cadenas de texto
28          for col in columnas:
29              if not isinstance(col, str):
30                  raise ValueError("Cada nombre de columna debe ser" +
31                                   " una cadena de texto.")
32
33          # Verificar que todas las columnas existan dentro del DataFrame
34          for col in columnas:
35              if col not in self.__dataframe.columns:
36                  raise ValueError(f"La columna {col} no existe dentro" +
37                                   " del DataFrame.")
38
39      def __realizar_DBSCAN(self, columnas, eps=0.5, min_samples=5):
40          # Escalado de los datos
41          data = self.__dataframe[columnas]
42          scaler = StandardScaler()
43          scaler_data = scaler.fit_transform(data)
44
45          # Realizar DBSCAN
46          dbscan = DBSCAN(eps=eps, min_samples=min_samples)
47          clusters = dbscan.fit_predict(scaler_data)
48
49          # Agregar los resultados de los clusters al DataFrame
50          self.__dataframe['cluster_dbscan'] = clusters
51
52      def estimar_eps(self, columnas, min_samples=5):
53          # Verificar las columnas
54          self.__verificador_columnas(columnas)
55
56          # Escalado de los datos
57          data = self.__dataframe[columnas]
58          scaler = StandardScaler()
59          scaler_data = scaler.fit_transform(data)
60
61          # Calcular distancias de k-vecinos
62          neigh = NearestNeighbors(n_neighbors=min_samples)
63          nbrs = neigh.fit(scaler_data)
64          distances, indices = nbrs.kneighbors(scaler_data)
65
```

```
        # Ordenar distancias de k-vecinos
        distances = np.sort(distances[:, min_samples - 1], axis=0)

        # Graficar distancias de k-vecinos
        plt.plot(distances)
        plt.title('Gr fico de distancia de k-vecinos')
        plt.xlabel('Puntos ordenados por distancia al k- simo  vecino')
        plt.ylabel(f'Distancia al {min_samples}  vecino m s cercano')
        plt.show()

    def grafica_DBSCAN_3D(self, colum1, colum2, colum3,
                          xlabel="", ylabel="", zlabel="",
                          eps=0.5, min_samples=5):
        # Verificar las columnas antes de proceder
        columnas = [colum1, colum2, colum3]
        self.__verificador_columnas(columnas)

        # Realizar DBSCAN (llamando al m todo privado)
        self.__realizar_DBSCAN(columnas, eps, min_samples)

        # Si los labels no se proporcionan, usar los nombres de las columnas
        xlabel = xlabel if xlabel else colum1
        ylabel = ylabel if ylabel else colum2
        zlabel = zlabel if zlabel else colum3

        # Graficar en 3D
        fig = plt.figure(figsize=(10, 7))
        ax = fig.add_subplot(111, projection='3d')
        ax.scatter(self.__dataframe[colum1], self.__dataframe[colum2],
                   self.__dataframe[colum3],
                   c=self.__dataframe['cluster_dbscan'],
                   cmap='viridis', marker='o')

        ax.set_title('Clusters usando DBSCAN en 3D')
        ax.set_xlabel(xlabel)
        ax.set_ylabel(ylabel)
        ax.set_zlabel(zlabel)
        plt.show()

    def mostrar_clusters(self, colum1, colum2, colum3, eps=0.5, min_samples=5):
        columnas = [colum1, colum2, colum3]
        self.__verificador_columnas(columnas)

        data = self.__dataframe[columnas]
        scaler = StandardScaler()
        scaler_data = scaler.fit_transform(data)
```

```
112
113         dbscan = DBSCAN(eps=eps, min_samples=min_samples)
114         labels = dbscan.fit_predict(scaler_data)
115
116         unique_labels = set(labels)
117         for label in unique_labels:
118             if label == -1:
119                 # -1 es considerado ruido
120                 print(f"Ruido: {len(data[labels == label])} elementos.")
121             else:
122                 cluster_points = scaler_data[labels == label]
123                 centroid = cluster_points.mean(axis=0)
124                 print(f"Cluster {label + 1}: Centroide = {centroid} con {len(
    cluster_points)} elementos.")
```

Listing 4: Código DBSCAN_3D en Python

6.5. Codigo main.py

Código principal final empleado para los resultados del proyecto:

```
# ---------------- # Se importan las librer as necesarias # ----------------- #
import pandas as pd                      # Manejo del CSV
from KMedias import KMedias
from DBSCAN_3D import DBSCAN_3D
from MarkdownToCSV import MarkdownToCSV
import matplotlib.pyplot as plt

# Configurar backend interactivo para matplotlib
plt.switch_backend('TkAgg')

# Para evitar el error de detecci n de nucleos de mi computadora
import os   # Evita la detecci n de nucleos
os.environ['LOKY_MAX_CPU_COUNT'] = '2'

# --------------------- # Convierte de Markdown a CSV # --------------------- #
# Determinar la ruta absoluta del archivo markdown
current_dir = os.path.dirname(os.path.abspath(__file__))
name_md = os.path.join(current_dir, "..", "Datos", "AnexoA.md")

# Convertir de Markdown a CSV
md_to_csv = MarkdownToCSV(name_md)
name_csv = md_to_csv.convert()

# Dataframe normalizado
name_csv_norm = md_to_csv.normalize_columns(name_csv)
# ---------------------- # An lisis y limpiza del CSV # --------------------- #
# Se carga el CSV en un DataFrame:
aresep_df = pd.read_csv(name_csv)
aresep_df_norm = pd.read_csv(name_csv_norm)
aresep_df = aresep_df.loc[:, ['DPIR', 'FPI', 'Abonados']]
aresep_df_norm = aresep_df_norm.loc[:, ['DPIR', 'FPI', 'Abonados']]

# Eliminar filas con valores NaN
aresep_df = aresep_df.dropna()
aresep_df_norm = aresep_df_norm.dropna()

# Se muestra la informaci n del DataFrame al usuario:
print("\n================= Datos originales del DataFrame: ==================")
print(aresep_df)

# An lisis exploratorio de datos:
print("\n==================== Informaci n del DataFrame: ====================")
aresep_df.info()
```

```
44
45 # --- # Prueba de K-Means con los dos csv# --- #
46 kmedias = KMedias()
47 kmedias_norm = KMedias()
48 kmedias.setDataFrame(aresep_df)
49 kmedias_norm.setDataFrame(aresep_df_norm)
50 # Estimar un valor adecuado de clusters usando el m todo del codo
51 kmedias.metodo_elbow('Abonados', 'DPIR', 'FPI')
52 kmedias_norm.metodo_elbow('Abonados', 'DPIR', 'FPI')
53 # Gr fica de los crusters creados por el algortimos
54 print("\nPrueba: KMeans con datos brutos de Anexo A.")
55 kmedias.grafica_KMeans_3D('Abonados', 'DPIR', 'FPI', k_clusters=3,
56                           xlabel='Abonados por circuito')
57 kmedias.mostrar_clusters('Abonados', 'DPIR', 'FPI', k_clusters=3)
58 print("\nPrueba: KMeans con datos normalizado de Anexo A.")
59 kmedias_norm.grafica_KMeans_3D('Abonados', 'DPIR', 'FPI', k_clusters=3,
60                           xlabel='Abonados por circuito')
61 kmedias_norm.mostrar_clusters('Abonados', 'DPIR', 'FPI', k_clusters=3)
62
63 # --- # Prueba de DBSCAN terna con los dos csv # --- #
64 dbscan = DBSCAN_3D()            # Brutos
65 dbscan_norm = DBSCAN_3D()       # Normalizados
66 dbscan.setDataFrame(aresep_df)
67 dbscan_norm.setDataFrame(aresep_df_norm)
68 # Estimar un valor adecuado para eps para ambos casos
69 dbscan.estimar_eps(['Abonados', 'DPIR', 'FPI'], min_samples=6)
70 dbscan_norm.estimar_eps(['Abonados', 'DPIR', 'FPI'], min_samples=6)
71 # Gr fica de los crusters creados por el algortimos
72 print("\nPrueba: DBSCAN con datos brutos de Anexo A.")
73 dbscan.grafica_DBSCAN_3D('Abonados', 'DPIR', 'FPI', eps = 1.23, min_samples=6)
74 dbscan.mostrar_clusters('Abonados', 'DPIR', 'FPI', eps = 1.23, min_samples=6)
75 print("\nPrueba: DBSCAN con datos normalizado de Anexo A.")
76 dbscan_norm.grafica_DBSCAN_3D('Abonados', 'DPIR', 'FPI', eps = 1.23, min_samples=6)
77 dbscan_norm.mostrar_clusters('Abonados', 'DPIR', 'FPI', eps = 1.23, min_samples=6)
```

Listing 5: Código DBSCAN_3D en Python

Referencias

[1] A. R. de los Servicios Públicos (ARESEP), "Informe de calidad del suministro de electricidad 2022," Jun 2022, páginas 69-76. [Online]. Available: https://aresep-my.sharepoint.com/:b:/g/personal/multimedia_aresep_go_cr/EXCrW3PWJh5MpoNaCEyxfGYBc6j2hykDzEFWqmpYjG-S4Q?e=a8tENW&download=1

[2] ——, "Informe sobre calidad del suministro eléctrico 2013," May 2014, página 6. [Online]. Available: https://aresep.go.cr/wp-content/uploads/2014/07/Informe_de_calidad-2013.pdf

[3] A. F. L. Villalba and E. C. G. L. Rotta, "Comparison of dbscan and k-means clustering methods in the selection of representative clients for a vehicle routing model," 2020 Congreso Internacional de Innovación y Tendencias en Ingeniería (CONIITI), pp. 1–6, 2020.

[4] M. C. Jiménez, "Proyecto 1: Agrupamiento de circuitos de distribución eléctrica," 2024, curso de Inteligencia Artificial Aplicada a la Ingeniería Eléctrica, Universidad de Costa Rica.

[5] Instituto Costarricense de Electricidad, "Historia del ice," 2021, accedido: 2024-09-11. [Online]. Available: https://www.grupoice.com/wps/portal/ICE/quienessomos/quienes-somos/historia#:~:text=Como%20resultado%2C%20el%2031%20de%20julio%20de%201928,fueron%20la%20antesala%20del%20ICE%2C%20creado%20en%201949.

[6] ARESEP, "Derechos de los usuarios de servicios públicos," 2024, accedido: 2024-09-11. [Online]. Available: https://aresep.go.cr/gestion-usuarios/derechos-usuarios/#:~:text=La%20Constituci%C3%B3n%20Pol%C3%ADtica%20de%20Costa%20Rica%20en%20el,libertad%20de%20elecci%C3%B3n%2C%20y%20a%20un%20trato%20equitativo.

[7] L. Bule and N. Nair, "Smart meter customer data synchronization technique by kmean clustering," 2023 IEEE PES GTD International Conference and Exposition (GTD), pp. 409–413, 2023.