



EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO
NACIONAL DE MÉXICO®

INSTITUTO TECNOLÓGICO DE CULIACÁN



Optimización de sitios de muestreo de Sinaloa en aguas superficiales implementando aprendizaje automático

TESIS

OBTENER EL GRADO DE
INGENIERO EN SISTEMAS COMPUTACIONALES

POR:

Edgar Michel Pérez Ramírez

DIRECTOR DE TESIS:
Dr. Héctor Rodríguez Rangel

CULIACÁN, SINALOA

13/10/2025

AUTORIZACIÓN DE IMPRESIÓN



Educación
Secretaría de Educación Pública



TECNOLÓGICO
NACIONAL DE MÉXICO



Instituto Tecnológico de Culiacán

DIVISION DE ESTUDIOS PROFESIONALES
ASUNTO: **Autorización de Impresión**

Culiacán, Sinaloa, 23/09/2025 19:23

C. EDGAR MICHEL PEREZ RAMIREZ
PASANTE(S) DE LA CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES
P R E S E N T E.

De acuerdo con fallo emitido por la Comisión Revisora Integrada por:

Cédula Profesional

Presidente:	Maestría en Gestión de Tecnologías de Información Marisol Manjarrez Beltrán	6920340
Secretario:	Maestría en Ciencias en Ciencias Computacionales Arcelia Judith Bustillos Martínez	13118410
Vocal:	Maestría en Ciencias de la Computación José Luis Medina Jiménez	13778869

Y considerando que reúne los requisitos establecidos en la Opción: **TITULACIÓN INTEGRAL: TESIS PROFESIONAL: OPTIMIZACIÓN DE SITIOS DE MUESTREO DE SINALOA EN AGUAS SUPERFICIALES IMPLEMENTANDO APRENDIZAJE AUTOMÁTICO**", damos a usted nuestra autorización para que proceda a imprimir su trabajo profesional.

Sin otro particular, quedo de usted.

ATENTAMENTE
Excelencia en Educación Tecnológica®



Educación
Secretaría de Educación Pública



TECNOLÓGICO
NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CULIACÁN

LIC. CONCEPCION MENDOZA ROSALES

DEPARTAMENTO DE

DIVISIÓN DE ESTUDIOS PROFESIONALES

JEFA DE LA DIVISIÓN DE ESTUDIOS PROFESIONALES



2025
Año de
La Mujer
Indígena

Juan de Dios Batic 310 Pte. Col. Guadalupe C.P. 80220
Culiacán, Sinaloa. Tel. 667-454-0100
tecnm.mx www.culiacan.tecnm.mx



I. Dedicatoria

Esta tesis está dedicada a todas las personas que han tenido un impacto significativo en el desarrollo académico y personal que he tenido al haber tomado la decisión de estudiar la carrera de Ingeniería de Sistemas Computacionales en el Instituto Tecnológico de Culiacán.

Siendo más específico, a los docentes que imparten clases dentro de la institución, entre los que se encuentran: el Dr. Héctor Rodríguez Rangel, quién me brindó la oportunidad de desarrollar el proyecto para la cual esta tesis se basa fundamentalmente. Además, el doctor me brindó sus destacados conocimientos en el área de la inteligencia artificial durante la realización de OSMIAA, siendo pieza fundamental para el desarrollo de este mismo.

Al docente M.C. José Luis Medina Jiménez, quien me brindó un apoyo constante para el desarrollo de OSMIAA, proporcionándome información puntual y significativa durante el periodo establecido. A la docente MGTI. Marisol Beltrán Manjarrez, quién además de también brindarme sus conocimientos durante mi desarrollo académico, formó parte en la realización de OSMIAA al haber sido designada como uno de mis asesores.

De la misma manera, agradezco a todas y cada una de las personas que han formado parte en mi desarrollo personal, a mis padres Edgar Adalberto Pérez Zazueta y María Ofelia Ramírez Zavala, quienes siempre han velado por mi bienestar para siempre dar todo lo mejor de mí, ensañándome los valores que se deben de tener en cuenta durante toda la vida y apoyándome en todos los sentidos para haber llegado al lugar en el que estoy actualmente. A mis abuelos Hermelinda Zazueta y Jesús Pérez, que durante parte de mi infancia siempre velaron por mi salud en caso de que mis padres no estuvieran disponibles por cuestiones laborales, y hoy en día les guardo un enorme cariño y respeto por todo lo que han sacrificado en favor de nuestra familia.

Por último, quiero agradecer a todos mis amigos y compañeros que conocí durante mi estancia en el Instituto Tecnológico de Culiacán, quienes me ayudaron a desarrollarme en el ámbito social, pero sobre todo, me brindaron experiencias inolvidables que siempre atesoraré a lo largo de mi vida.



II. Agradecimientos

Agradezco a los docentes y trabajadores del Instituto Tecnológico de Culiacán por su constante esfuerzo en impartir clases y compartir su conocimiento sobre la materia; así como por trabajar en mantener una calidad reconocible dentro de la institución en cuanto a instalaciones se refiere.

Además, agradezco a la Comisión Nacional del Agua (CONAGUA) por haber elaborado y proporcionado la información contenida en la Red Nacional de Medición de Calidad del Agua (RENAMECA), vista como una acción fundamental más en su arduo trabajo en mantener el agua en las mejores condiciones posibles, al llevar a cabo el monitoreo sistemático y permanente de la calidad del agua de los principales cuerpos de agua del país.



III. Declaración de autenticidad

Por la presente declaro que, salvo cuando se haga referencia específica al trabajo de otras personas, el contenido de esta tesis es original y no se ha presentado total o parcialmente para su consideración para cualquier otro título o grado en esta o cualquier otra Universidad. Esta tesis es resultado de mi propio trabajo y no incluye nada que sea resultado de algún trabajo realizado en colaboración, salvo que se indique específicamente en el texto.

Edgar Michel Pérez Ramírez

Culiacán, Sinaloa, México, 2025



IV. Resumen

El monitoreo de la calidad del agua en México enfrenta desafíos significativos debido a limitaciones económicas, tecnológicas y geográficas. Este proceso es esencial para garantizar la salud pública y la preservación de ecosistemas ya que permite identificar y prevenir problemas de contaminación de manera oportuna.

La obtención de resultados al realizar un monitoreo sobre la calidad del agua requiere la recolección de datos en campo, su procesamiento, análisis e interpretación, lo cual requiere tiempo, recursos y herramientas especializadas. Esto quiere decir que las metodologías tradicionales suelen ser costosas e ineficientes, lo que limita su aplicación en ciertas regiones.

En respuesta a esta problemática, se propone OSMIAA (Optimización de Sitios de Muestreo de Sinaloa en aguas superficiales Implementando Aprendizaje Automático), una aplicación web que permite obtener y visualizar de forma más accesible los resultados obtenidos tras haber aplicado algoritmos de aprendizaje no supervisado, específicamente técnicas de agrupamiento de datos (clustering). Esto se logró con la clasificación de los sitios de muestreo con base en el nivel de calidad del agua que poseen para facilitar su posterior análisis y comparación.

OSMIAA partió de una base de datos proporcionada por la Comisión Nacional del Agua (CONAGUA), la cual contiene información sobre parámetros físicos, químicos y biológicos recolectados en diversos puntos geográficos del estado de Sinaloa. El proceso comenzó con la organización y limpieza de los datos, seguido de la creación de una matriz numérica, donde cada valor correspondía al promedio de los registros obtenidos en cada sitio de muestreo.

A lo largo del estudio, se realizaron múltiples pruebas, desde análisis centrados en un solo municipio (Ahome), hasta la integración de todos los sitios del estado. En este contexto, se implementaron ciertos algoritmos de agrupamiento con el fin de entender su funcionamiento general y comparar los resultados generados.

Como resultado final, se desarrolló una aplicación web interactiva que permite al usuario seleccionar los parámetros a considerar, elegir el algoritmo de agrupamiento a utilizar, ajustar sus configuraciones, visualizar los resultados directamente en un mapa del estado, entre otras funcionalidades. OSMIAA representó un avance importante en el uso de inteligencia artificial aplicada al análisis ambiental, con un alto potencial para apoyar en futuras decisiones relacionadas con el monitoreo del agua.



Por último, el trabajo realizado en OSMIAA ayudará a explorar nuevas formas de representar datos, así como la obtención de inferencias útiles a partir de la identificación de puntos estratégicos en cada clúster generado.



V. Palabras clave

OSMIAA: Optimización de sitios de muestreo de Sinaloa en aguas superficiales implementando aprendizaje automático

CONAGUA: Comisión Nacional del Agua

RENAMECA: Red Nacional de la Medición de la Calidad del Agua

IA: Inteligencia Artificial

Machine Learning: Aprendizaje automático

Clustering: Agrupación de datos similares

HCA: Hierarchical Cluster Analysis, agrupamiento jerárquico

PCA: Análisis de Componentes Principales

CSV: Comma-Separated Values, Valores Separados por Comas



VI. Índice general

1.	Introducción	1
1.1.	Planteamiento del problema	2
1.2.	Hipótesis	3
1.3.	Objetivos	3
1.3.1.	Objetivo General	3
1.3.2.	Objetivos Específicos	3
1.4.	Justificación	3
1.5.	Estructura de la tesis	4
2.	Marco teórico	6
2.1.	Inteligencia artificial	6
2.1.1.	Aprendizaje automático (Machine Learning)	6
2.1.1.1.	Aprendizaje supervisado	8
2.1.1.2.	Aprendizaje no supervisado	8
2.2.	Enfoques del aprendizaje no supervisado	9
2.2.1.	Agrupación de datos (clustering)	9
2.2.1.1.	Algoritmos de Agrupación de datos	10
2.2.1.1.1.	Propagación por Afinidad (Affinity Propagation)	11
2.2.1.1.2.	BIRCH	12
2.2.1.1.3.	DBSCAN	14
2.2.1.1.4.	K-Means	16
2.2.1.1.5.	Mini-Batch K-Means	18
2.2.1.2.	Métricas de distancia	20
2.2.1.2.1.	Métrica de distancia euclidiana	20
2.2.1.2.2.	Métrica de distancia Manhattan	23
2.2.1.3.	Agrupación Jerárquica de Clústeres (HCA)	24
2.2.2.	Reducción de dimensionalidad	25
2.2.2.1.	Análisis de Componentes Principales (PCA)	25
2.3.	Herramientas y recursos	26
2.3.1.	Python	26
2.3.1.1.	Librerías	27
2.3.1.1.1.	NumPy	27
2.3.1.1.2.	Pandas	28
2.3.1.1.3.	Scikit-learn	28
2.3.1.1.4.	SciPy	29



2.3.1.1.5.	Matplotlib	30
2.3.1.1.6.	Plotly	32
2.3.1.1.7.	Dash	33
2.3.2.	Render	35
3.	Estado del arte	37
3.1.	Antecedentes	37
3.2.	Tendencias actuales	39
3.2.1.	Sensores IoT (Internet de las Cosas) para monitoreo ambiental en tiempo real.....	39
3.2.2.	Aplicación de inteligencia artificial y aprendizaje automático	40
3.2.3.	Integración de GIS (Sistemas de Información Geográfica) y modelos hidrológicos.....	42
4.	Desarrollo del proyecto.....	46
4.1.	Tratamiento de datos	46
4.2.	Implementación de algoritmos de agrupamiento y reducción de dimensionalidad	47
4.3.	Desarrollo de aplicación web interactiva y visualización geográfica del agrupamiento.....	48
5.	Pruebas y Resultados	51
5.1.	Resultados del tratamiento de datos	51
5.2.	Resultados de la aplicación de algoritmos de agrupamiento y reducción de dimensionalidad	53
5.3.	Resultados de la aplicación web interactiva y visualización geográfica del agrupamiento	56
6.	Conclusiones y trabajo futuro	62
6.1.	Conclusiones	62
6.2.	Trabajo a futuro	63
7.	Bibliografía.....	64
8.	Anexos.....	68
A.	Código de la aplicación web	68
B.	Carta de no inconveniente.....	74



VII. Índice de figuras

Figura 2-1. Mapa de la clasificación general de Machine Learning (Nota. tomado de (Contreras et al., 2024, pág. 20))	7
Figura 2-2. Conjunto de datos numéricos MNIST (Nota. tomado de (ChrisAlcid, 2018))	8
Figura 2-3. Ejemplo básico de Agrupación de datos (Nota. tomado de (Bobadilla Sancho, 2020, pág. 6))	10
Figura 2-4. Propagación por Afinidad en Python (Nota. elaboración propia)	12
Figura 2-5. Visualización gráfica de Propagación por Afinidad (Nota. elaboración propia)	12
Figura 2-6. BIRCH en Python (Nota. elaboración propia)	14
Figura 2-7. Visualización gráfica de BIRCH (Nota. elaboración propia)	14
Figura 2-8. DBSCAN en Python (Nota. elaboración propia)	16
Figura 2-9. Visualización gráfica de DBSCAN (Nota. elaboración propia)	16
Figura 2-10. K-Means en Python (Nota. elaboración propia)	17
Figura 2-11. Visualización gráfica de K-Means (Nota. elaboración propia)	18
Figura 2-12. Mini-Batch K-Means en Python (Nota. elaboración propia)	19
Figura 2-13. Visualización gráfica de Mini-Batch K-Means (Nota. elaboración propia)	19
Figura 2-14. Distancia Interclúster vs Distancia Intracúster (Nota. (López Ollarbez & Gómez Abreu, 2021, pág. 10))	20
Figura 2-15. Teorema de Pitágoras (Nota. tomado de (Neurochispas, s.f.))	21
Figura 2-16. Representación visual de la distancia euclidiana en 2D (Nota. tomado de (Chugani, 2024))	21
Figura 2-17. Representación visual de la distancia euclidiana en 3D (Nota. tomado de (Chugani, 2024))	22
Figura 2-18. Representación visual de la distancia Manhattan en 2D (Nota. tomado de (Chugani, 2024))	23
Figura 2-19. Ejemplo dendograma de algoritmo jerárquico (Nota. tomado de (Noble, 2024))	24
Figura 2-20. Equipo de desarrollo y mantenimiento de Scikit-learn (Nota. tomado de (Scikit-learn Development Team, s.f.))	29
Figura 2-21. Parte del equipo de desarrollo y mantenimiento de SciPy (Nota. tomado de (SciPy Development Team, s.f.))	30
Figura 2-22. Código implementando la librería Matplotlib (Nota. elaboración propia)	31
Figura 2-23. Resultado de la ejecución del código: gráfico de dispersión (Nota. elaboración propia)	31
Figura 2-24. Código implementando la librería Plotly (Nota. elaboración propia)	32
Figura 2-25. Resultado de la ejecución del código: diagrama de burbujas (Nota. elaboración propia)	33
Figura 2-26. Código implementando Dash de Python (Nota. elaboración propia)	34
Figura 2-27. Resultado de la ejecución del código: gráficos de barras para tres columnas contenidas en el dataset (Nota. elaboración propia)	35
Figura 2-28. Render GUI al inicializar un proyecto (Nota. elaboración propia)	36
Figura 3-1. Red de monitoreo optimizada en la cuenca del Río Culiacán para temporada lluviosa y seca (Nota. tomado de (Mendivil-García et al., 2024))	38
Figura 3-2. Comparación entre datos obtenidos por sensores IoT y análisis de laboratorio (Nota. tomado de (Agudelo Méndez & Lozano Bernal, 2025, pág. 137))	40
Figura 3-3. Comparación entre PCA y t-SNE en el conjunto de datos "transecto de Oslo" (Nota. tomado de (Liu et al., 2021, pág. 6))	41
Figura 3-4. Comparación entre PCA y t-SNE en el conjunto de datos "agua kárstica Taiyuana" (Nota. tomado de (Liu et al., 2021, pág. 8))	42
Figura 3-5. Comparación entre PCA y t-SNE en el conjunto de datos "agua subterránea de la Llanura Jiangnan" (Nota. tomado de (Liu et al., 2021, pág. 9))	42
Figura 3-6. Nivel medio de aguas subterráneas para el escenario actual y futuro (Nota. tomado de (Tsegaye et al., 2024, pág. 10))	43
Figura 3-7. Variación del nivel de agua en estanques (a, b) y humedales (c, d) en escenarios de 2024 y 2100 (Nota. tomado de (Tsegaye et al., 2024, pág. 10))	44



Figura 3-8. Mapas de inundaciones en la actualidad (2024) y futuro (2100) considerando el uso existente del suelo (Nota. tomado de (Tsegaye et al., 2024, pág. 12))	45
Figura 4-1. Diagrama del proceso de tratamiento de datos aplicado al conjunto de datos de RENAMECA (Nota. elaboración propia)	47
Figura 4-2. Diagrama del proceso de implementación de algoritmos de agrupamiento y reducción de dimensionalidad (Nota. elaboración propia)	48
Figura 4-3. Interfaz gráfica de la aplicación web con todas las funciones implementadas (Nota. elaboración propia) ..	49
Figura 4-4. Ejemplo de implementación (Nota. tomado de (Plotly Development Team, s.f.))	50
Figura 5-1. Selección de parámetros: gráfico de barras Comparación columnas (Nota. elaboración propia)	52
Figura 5-2. Extracción de datos relacionados con Sinaloa: gráfico de barras Comparación filas (Nota. elaboración propia)	53
Figura 5-3. Agrupamiento generado: Propagación por Afinidad (Nota. elaboración propia)	53
Figura 5-4. Agrupamiento generado con BIRCH (Nota. elaboración propia)	54
Figura 5-5. Agrupamiento generado: DBSCAN (Nota. elaboración propia)	54
Figura 5-6. Agrupamiento generado: K-Means (Nota. elaboración propia)	55
Figura 5-7. Agrupamiento generado: Mini-Batch K-Means (Nota. elaboración propia)	55
Figura 5-8. Visualización geográfica de clústeres generados: Propagación por Afinidad (Nota. elaboración propia)	57
Figura 5-9. Visualización geográfica de clústeres generados: BIRCH (Nota. elaboración propia)	58
Figura 5-10. Visualización geográfica de clústeres generados: DBSCAN (Nota. elaboración propia)	59
Figura 5-11. Visualización geográfica de clústeres generados: K-Means (Nota. elaboración propia)	60
Figura 5-12. Visualización geográfica de clústeres generados: Mini-Batch K-Means (Nota. elaboración propia)	61



Capítulo 1

1. Introducción

El acceso al agua de calidad es esencial para la salud pública, el equilibrio de los ecosistemas y el mantenimiento de un desarrollo sustentable. Por lo tanto, el agua es un recurso vital aplicado en distintas funciones tales como: cocinar, regar, generar energía, transportar bienes, regular el clima, reducir enfermedades, etc. Este elemento no solo se resume en la vida cotidiana, ya que también es utilizado en muchos otros sectores.

De acuerdo con el (FIDA, 2021), las actividades de cultivos bajo riego, ganadería, pesca, acuicultura y silvicultura concentran cerca del 70% de las extracciones de agua a escala mundial y superan el 90% en la mayoría de los países menos adelantados; dentro de ese uso, casi el 67% se destina al riego.

En México, la calidad del agua se regula a través de la Norma Oficial Mexicana NOM-127-SSA1-2021, administrada por la Secretaría de Salud Federal y la Comisión Federal para la Protección contra Riesgos Sanitarios (COFEPRIS). Esta norma establece los límites permisibles de calidad y tratamientos necesarios para garantizar que el agua sea segura y adecuada para el consumo humano, lo cual resulta fundamental para un país en vías de desarrollo como México (CSR Staff, 2025).

Por otra parte, la Comisión Nacional del Agua (CONAGUA) es el organismo federal responsable de la gestión integral del agua en México, operando bajo la Secretaría de Medio Ambiente y Recursos Naturales (SEMARNAT), aun así, poseyendo autonomía técnica administrativa (Gobierno de México, s.f.). Entre sus objetivos principales, se pueden destacar los siguientes:

- Regular el otorgamiento de concesiones para el uso y aprovechamiento del agua
- Supervisar la distribución del agua para asegurar un acceso equitativo y sostenible
- Planificar y mantener infraestructura hidráulica (presas, canales y acueductos)
- Gestionar riesgos ante sequías e inundaciones mediante alertas tempranas
- Coordinar el Servicio Meteorológico Nacional (SMN).

Debido a su destacada producción agrícola, la calidad del agua es de suma importancia en Sinaloa (Zamudio et al., 2024). Estas actividades contribuyen a la contaminación de cuerpos de agua superficiales y



subterráneos, afectando tanto al suministro destinado para el consumo personal como el usado por los ecosistemas locales.

En este contexto, un importante problema que enfrenta la región es la dificultad para monitorear de manera continua y eficiente la calidad del agua, lo que limita la detección temprana de contaminantes y retrasa la implementación de medidas correctivas. La información existente suele obtenerse mediante muestreos puntuales y procesos manuales, lo que genera retrasos en la toma de decisiones y reduce la capacidad de respuesta ante riesgos sanitarios y ambientales.

Para afrontar esta situación, se desarrolló una aplicación web basada en técnicas de aprendizaje automático que permita evaluar, clasificar y visualizar registros relacionados con la calidad del agua generados a partir de mediciones físicas, químicas y biológicas. Ésta integra fuentes de datos locales, aplica modelos supervisados de clasificación y genera gráficos que facilitan la interpretación para el usuario.

1.1. Planteamiento del problema

El constante monitoreo de la calidad del agua representa un desafío importante para países en vías de desarrollo como México. Factores como la falta de infraestructura, la dispersión geográfica de los cuerpos de agua y la disponibilidad limitada de recursos económicos y tecnológicos dificultan la implementación de sistemas de monitoreo eficientes.

En consecuencia, muchas veces estas mediciones resultan ser insuficientes y no se realizan en todos los cuerpos de agua (Gutiérrez Jaber, 2024), lo que incrementa el riesgo de contaminación y amplifica los impactos en la salud pública y el medio ambiente.

Asimismo, las metodologías tradicionales de monitoreo se fundamentan en procedimientos de muestreo establecidos, los cuales exigen el desplazamiento hacia sitios de interés para la toma de muestras simples o compuestas, su preservación inmediata, etiquetado y traslado posterior a laboratorios acreditados por CONAGUA para un eventual análisis (Comisión Nacional del Agua, s.f.). Este proceso, a pesar de asegurar la obtención de buenos resultados, implican mayores tiempos de operación y uso de recursos logísticos.

Por ejemplo, según datos del Registro Público de Derechos del Agua (REPDa) la información proporcionada frecuentemente resulta poco confiable, las series de datos son incompletas y hay una escasa continuidad temporal, lo que dificulta analizar procesos contaminantes o detectar tendencias significativas (Perevochtchikova, 2009).



1.2. Hipótesis

La aplicación de métodos de aprendizaje automático no supervisado, como clustering (agrupación de datos similares) y reducción de dimensionalidad, permitirá identificar patrones en los puntos de muestreo de aguas superficiales de Sinaloa, optimizando su distribución y mejorando la representación de la calidad del agua a través de un modelo computacional integrado en una aplicación web interactiva.

1.3. Objetivos

1.3.1. Objetivo General

Diseñar e implementar un modelo computacional capaz de correlacionar diferentes puntos de muestreo en aguas superficiales de Sinaloa, aplicando métodos de aprendizaje no supervisado como: clustering (agrupamiento de datos similares) y reducción de dimensionalidad. Esto con el fin de mejorar la distribución de los puntos de muestreo y mostrar los resultados obtenidos en una aplicación web.

1.3.2. Objetivos Específicos

- Preprocesar la información que se encuentra en el conjunto de datos obtenido a partir de la Red Nacional de la Medición de la Calidad del Agua (RENAMECA).
- Desarrollar un módulo inteligente que permita regresar la correlación con distintos modelos de agrupamiento para la reducción de sitios de muestreo (BIRCH, DBSCAN, K-Means, etc.).
- Evaluar los resultados obtenidos mediante la cohesión y separación de clústeres, implementando métricas como: agrupamiento jerárquico y análisis de varianza explicada.
- Desarrollar una aplicación web que muestre visualmente el desarrollo de este módulo mediante la generación de mapas de agrupamiento y sus ajustes correspondientes.

1.4. Justificación

El agua es un recurso esencial para el desarrollo humano, económico y ambiental. Según la (ONU, s.f.): “El agua está en el epicentro del desarrollo sostenible y es fundamental para el desarrollo socioeconómico, la energía, la producción de alimentos, los ecosistemas y para la supervivencia de los seres humanos”. (párr.1)



Este recurso se considera indispensable en todos los sentidos, ya que es utilizada para actividades antropogénicas, agropecuarias o agrícolas que están en constante desarrollo. Por ejemplo, en México la actividad agrícola es de aproximadamente 21.6 millones de hectáreas, siendo uno de los más destacados el estado de Sinaloa, generando alrededor de 12,170 toneladas de cultivos como el tomate o el maíz (Zamudio et al., 2024, pág. 5).

Siguiendo con el tema, datos del Registro Público de Derechos de Agua (REPDa) compilados por la Comisión Nacional del Agua (CONAGUA) arrojan que las cinco entidades mexicanas con mayor volumen concesionado (cantidad de agua específica) para uso agrícola son Sinaloa: 9,005.4 hm³/año, Sonora: 6,389.9 hm³/año, Michoacán: 4,807.9 hm³/año, Chihuahua: 4,723.6 hm³/año y Tamaulipas 3,756.3 hm³/año (CEFP, 2019).

Debido a esto, la calidad del agua se ha convertido en un tema de suma importancia, ya que es necesario saber si se encuentra en buenas condiciones para su uso. La Comisión Nacional del Agua (CONAGUA) contiene una red de monitoreo llamada Red Nacional de Medición de la Calidad del Agua (RENAMECA), en la cual se lleva cabo el monitoreo sistemático y permanente de la calidad del agua de los principales cuerpos de agua del país (Comisión Nacional del Agua, 2025).

Es aquí en donde entra la gran importancia del desarrollo de OSMIAA, ya que es necesario optimizar los puntos de muestreo distribuidos en Sinaloa; y con esto, mejorar su distribución geográfica. Esto es muy valioso, ya que mientras más eficiente sea la toma de puntos de muestreo, se requerirá menores costos y tiempo a la hora de determinar la calidad del agua en los cuerpos de agua a trabajar.

1.5. Estructura de la tesis

Antes que nada, el siguiente documento está dividido en siete capítulos y está presentado de la siguiente manera:

- **Capítulo 1 – Introducción:** Se explica el planteamiento del problema, la hipótesis, los objetivos generales y específicos, y la justificación.
- **Capítulo 2 – Marco teórico:** Se describen los conceptos más importantes para la elaboración de OSMIAA, los cuales fueron: inteligencia artificial, aprendizaje automático, algoritmos de agrupamiento (clustering), métricas de distancia, reducción de dimensionalidad y las herramientas utilizadas para el desarrollo.



- **Capítulo 3 – Estado del arte:** Se describen las investigaciones y técnicas que se han desarrollado hasta la fecha y que están relacionados con los temas abordados en la tesis.
- **Capítulo 4 – Desarrollo:** Se detalla el proceso técnico que permitió transformar el conjunto de datos original en una base depurada y lista para su posterior implementación.
- **Capítulo 5 – Pruebas y Resultados:** Se presentan los resultados obtenidos tras aplicar los algoritmos de agrupamiento, iniciando con pruebas de datos sintéticos, la implementación de PCA para mejorar la visualización de resultados, hasta el desarrollo de una aplicación web interactiva que permita generar y analizar los clústeres generados de forma dinámica.
- **Capítulo 6 – Conclusiones y trabajo futuro:** Se reflexiona sobre los logros alcanzados por haber desarrollado a OSMIAA y se plantea una línea de trabajo a futuro.
- **Capítulo 7 – Bibliografía:** Se presentan todas las fuentes de información en la cual se basa esta investigación, respetando de forma íntegra el formato APA como lo fue estipulado con anterioridad.



Capítulo 2

2. Marco teórico

2.1. Inteligencia artificial

La inteligencia artificial (IA) representa un concepto que ha cobrado relevancia en años recientes. Sin embargo, su definición no es tan sencilla como pudiera parecer, ya que abarca comportamientos y metodologías que, en un principio, no se evidencian a primera vista.

Según el autor (Nilsson, 1998), establece que la inteligencia artificial, en términos generales y circulares, está relacionada con el comportamiento inteligente de artefactos. En consecuencia, el comportamiento inteligente involucra percepción, razonamiento, aprendizaje, comunicación y actuación en ambientes complejos. Una de las metas a largo plazo de la inteligencia artificial es el desarrollo de máquinas que puedan realizar estas acciones del mismo modo (o hasta mejor) que los humanos pueden.

Por otra parte, (Ríos, 2023) define la inteligencia artificial como el campo de la ciencia informática dedicado a dotar al software de capacidades que simulan los procesos cognitivos del ser humano. Estas simulaciones se aplican a la resolución de problemas, y sus resultados pueden traducirse en lenguaje escrito o hablado, acciones en movimiento, representaciones gráficas, etc.

Conviene subrayar que la inteligencia artificial cuenta con aplicaciones prácticas cada vez más presentes en nuestra vida cotidiana. Por ejemplo, el reconocimiento facial, las búsquedas en internet, la traducción automática de texto o la conducción automática de ciertos vehículos, todos estos son respaldados por técnicas basadas en dos conceptos ampliamente difundidos en la actualidad: aprendizaje automático y aprendizaje profundo.

2.1.1. Aprendizaje automático (Machine Learning)

De acuerdo con (Bobadilla Sancho, 2020), se define al aprendizaje automático como la ciencia que hace que las computadoras “aprendan” a partir de los datos. En vez de programar paso a paso cada solución específica para cada necesidad planteada, como lo estipula la programación convencional, el área del aprendizaje



automático está dedicada al desarrollo de algoritmos genéricos que puedan extraer patrones de diferentes tipos de datos.

En otras palabras, el aprendizaje automático proporciona soluciones generales que se adaptan a problemas particulares, reduciendo así el esfuerzo y tiempo invertido en tareas repetitivas. Por tanto, se facilita la dedicación a otras actividades de mayor peso estratégico.

Por otro lado, el aprendizaje ayuda a los modelos de Machine Learning a identificar patrones y tendencias durante el procesamiento de los datos. Dentro de este concepto, se distinguen dos enfoques principales

- Aprendizaje supervisado
- Aprendizaje no supervisado

Como se muestra en la Figura 2-1, ambos enfoques tienen diversas aplicaciones, según el tipo de problema que se busque resolver.

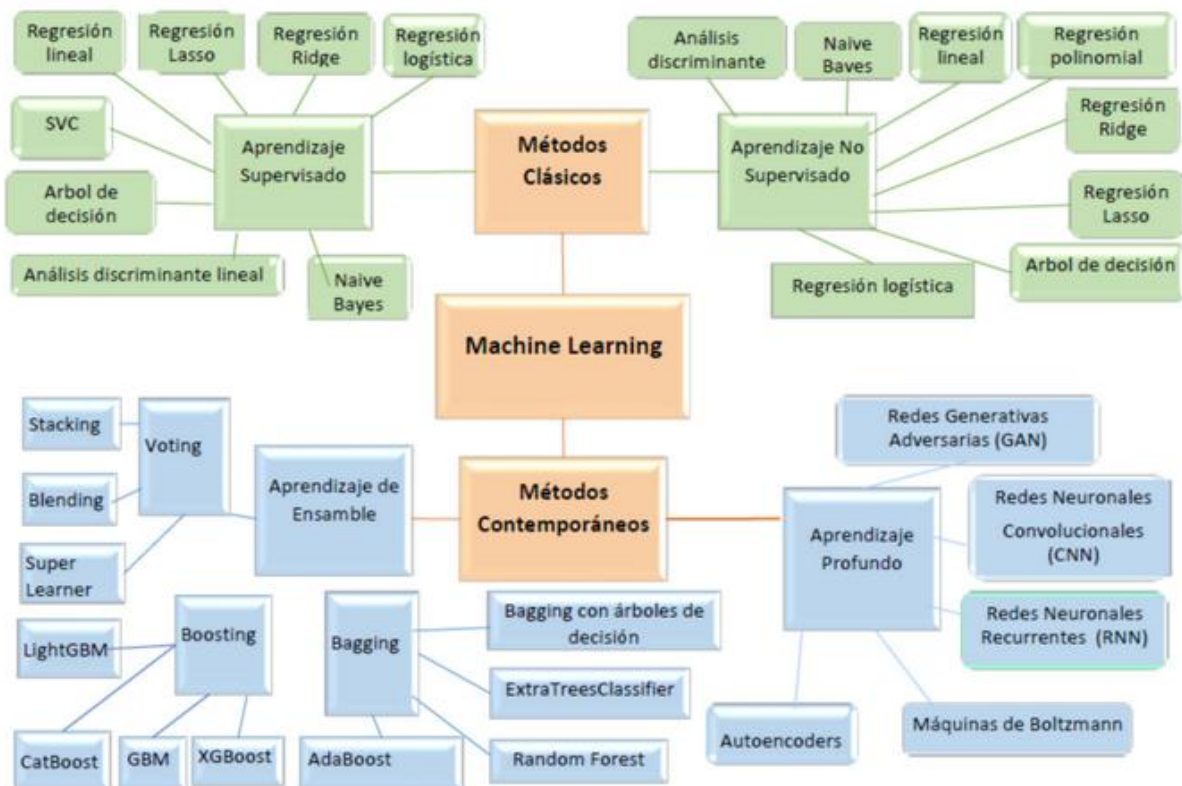


Figura 2-1. Mapa de la clasificación general de Machine Learning (Nota. tomado de Contreras et al., 2024, pág. 20)

2.1.1.1. Aprendizaje supervisado

En cuanto a los autores (Contreras et al., 2024), estos establecen que dentro del aprendizaje supervisado se encuentran algoritmos que son usados cuando los datos de entrada tienen una etiqueta o resultado conocido. Por tanto, el procedimiento consiste en entrenar un modelo para posteriormente hacer predicciones del resultado y realizar correcciones cuando las predicciones sean erróneas.

Asimismo, (Bobadilla Sancho, 2020) indica que el aprendizaje supervisado funciona cuando cada muestra o dato entrada se asocia a una etiqueta establecida. Esto permite entrenar modelos para realizar tareas de clasificación o regresión.

Por ejemplo, si se dispone de un conjunto de datos con imágenes de números escritos a mano (como en la Figura 2-2), se puede entrenar un modelo para predecir qué número representa una nueva imagen que no haya sido incluida en el conjunto de entrenamiento.

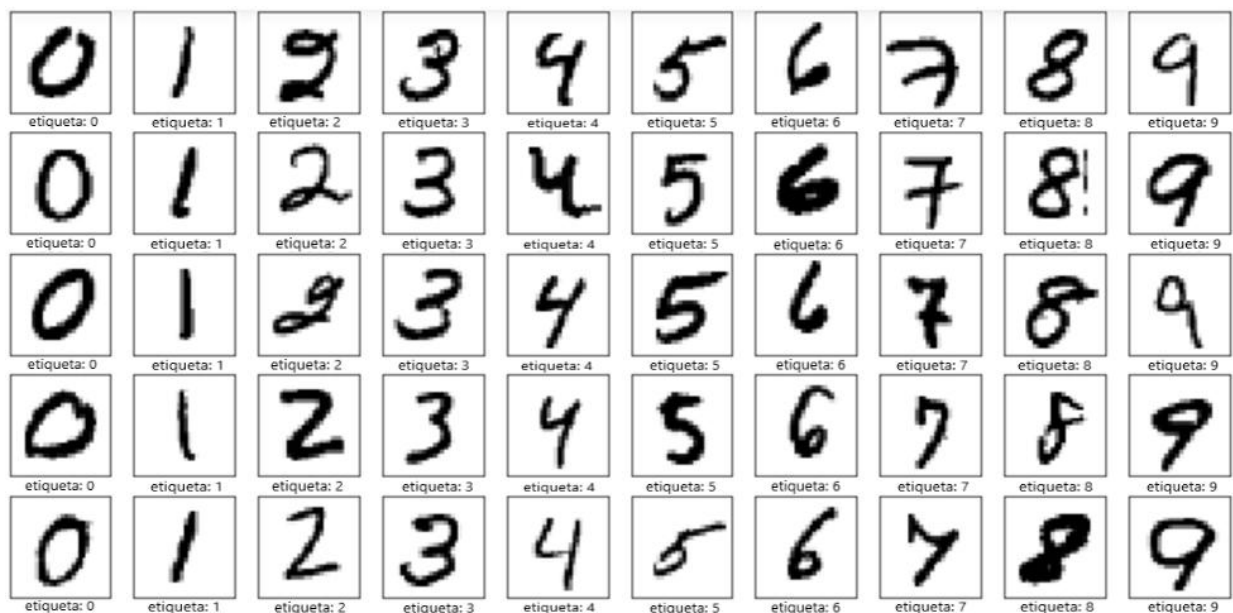


Figura 2-2. Conjunto de datos numéricos MNIST (Nota. tomado de ChrisAllcid, 2018)

2.1.1.2. Aprendizaje no supervisado

Por otro lado, (Murphy, 2012):habla sobre el aprendizaje no supervisado, en donde solo se proporcionan datos de salida y ningún dato de entrada. La meta de este método es descubrir un “descubrimiento de

conocimiento”, el cual consiste en determinar la existencia de una estructura interesante en los datos; y a diferencia del aprendizaje supervisado, aquí se desconoce la salida deseada para cada entrada.

Así pues, (IBM Corporation, 2021) describe que los algoritmos no supervisados permiten analizar y agrupar conjuntos de datos no etiquetados, identificando patrones ocultos sin intervención humana.

2.2. Enfoques del aprendizaje no supervisado

De acuerdo con (IBM Corporation, 2021), el aprendizaje no supervisado se clasifica en tres enfoques principales, según el objetivo del análisis:

- Agrupación de datos (clustering)
- Reglas de asociación
- Reducción de dimensionalidad

En este caso, se consideran únicamente el primer y tercer enfoque, al mismo tiempo que otros conceptos complementarios que se integran en la elaboración de OSMIAA.

2.2.1. Agrupación de datos (clustering)

Retomando lo divulgado por (IBM Corporation, 2021), la agrupación de datos se define como una técnica de minería de datos que agrupa datos no etiquetados, según sus similitudes o diferencias. En este enfoque, los algoritmos de agrupación procesan datos brutos sin clasificar y los organizan en grupos o estructuras que representen patrones subyacentes.

(Bobadilla Sancho, 2020) lo describe como una de las aplicaciones más conocidas del aprendizaje no supervisado. Su objetivo principal consiste en agrupar muestras con características similares, ya sea para identificar patrones de comportamiento en conductores, obtener los tipos de clientes en un servicio online, categorizar productos en un comercio electrónico, etc.

Por consiguiente, esta técnica resulta especialmente útil en problemas donde sea necesario detectar tendencias ocultas en los datos, facilitando la toma de decisiones en entornos complejos.

Por ejemplo, en la Figura 2-3 se ilustra una representación gráfica típica de agrupamiento de datos. En esta, se visualizan tres clústeres distintos correspondientes a diferentes tipos de lirios. Se observa que la especie



setosa es más fácil de identificar, mientras que versicolor y virginica presentan dificultades en su representación visual.

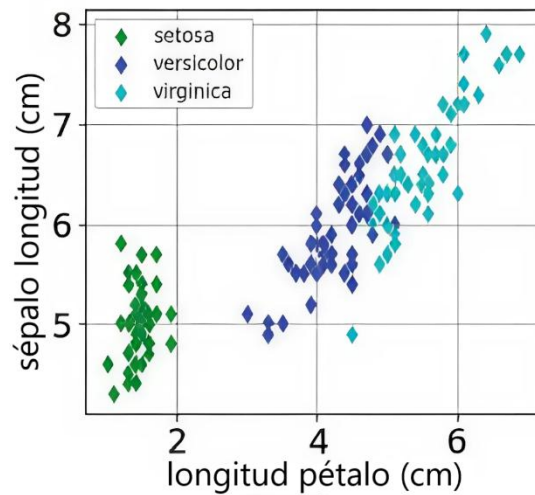


Figura 2-3. Ejemplo básico de Agrupación de datos (Nota. tomado de Bobadilla Sancho, 2020, pág. 6)

2.2.1.1. Algoritmos de Agrupación de datos

(Ahmad, 2023) señala que una de las técnicas más potentes del aprendizaje no supervisado consiste en aplicar algoritmos de agrupamiento que detectan patrones similares entre los datos. Generalmente, estos métodos se utilizan para entender aspectos específicos relacionados con el problema a resolver.

Al mismo tiempo, (Ahmad, 2023) añade la siguiente metáfora: “Pensemos, por ejemplo, en una gran biblioteca llena de libros. Cada libro representa un punto de datos, que contiene multitud de atributos como género, autor, año de publicación, etc. Ahora imaginemos a un bibliotecario (algoritmo de agrupamiento) encargado de organizar estos libros. Sin categorías ni instrucciones preexistentes, el bibliotecario empieza a clasificar los libros en función de sus atributos; todos los de misterio juntos, los clásicos juntos, los libros del mismo autor juntos, etc.” (pág. 56)

Con esto quiere decir que todos los algoritmos de agrupamiento comparten un principio común: agrupar datos con características similares, con el fin de detectar patrones ocultos en los datos que contribuyan a una toma de decisiones informada.

2.2.1.1.1. Propagación por Afinidad (Affinity Propagation)

El algoritmo de agrupación Propagación por Afinidad es una técnica propuesta por (Frey & Dueck, 2007), la cual, a diferencia de otros métodos, no requiere predefinir el número de clústeres. En cambio, este algoritmo selecciona automáticamente un conjunto de puntos representativos denominados “ejemplares” y asigna los datos restantes a estos con base en su similitud.

Asimismo, los autores establecen que el método se fundamenta en el intercambio iterativo de mensajes entre puntos de datos dentro de una matriz de similitud. En este proceso, cada punto puede convertirse en un potencial ejemplar. Los mensajes se actualizan continuamente hasta que el algoritmo converge en una estructura estable de clústeres.

Los dos tipos principales de mensajes utilizados son:

- **Responsabilidad $r(i, k)$:** representa qué tan adecuado es el punto k para ser el ejemplar del punto i , considerando su similitud relativa con otros posibles ejemplares.
- **Disponibilidad $a(i, k)$:** representa qué tan apropiado es para el punto i considerar al punto k como su ejemplar, teniendo en cuenta el apoyo de otros puntos.

Además, (Frey & Dueck, 2007) complementan que el método Propagación por Afinidad toma como entrada medidas de similitud entre pares de puntos de datos. Posteriormente, se intercambian mensajes con valores reales con dichos puntos de datos, hasta que emerge gradualmente en un conjunto de ejemplares de alta calidad y los clústeres correspondientes.

Por otro lado, (Brownlee, 2020) explica que este algoritmo se implementa a través de la clase AffinityPropagation y cuenta con dos hiperparámetros:

- **Atenuación (damping):** acepta valores entre 0.5 y 1.
- **Preferencia (preference):** permite valores enteros positivos y negativos.



En la Figura 2-4, se muestra un ejemplo básico del algoritmo, desarrollado en Python, mientras que la Figura 2-5 presenta los resultados de dicha implementación.

```
1 # affinity propagation clustering |
2 from numpy import unique
3 from numpy import where
4 from sklearn.datasets import make_classification
5 from sklearn.cluster import AffinityPropagation
6 from matplotlib import pyplot
7 # define dataset
8 X, _ = make_classification(n_samples=1000, n_features=2, n_informative=2, n_redundant=0, n_clusters_per_class=1, random_state=4)
9 # define the model
10 model = AffinityPropagation(damping=0.9)
11 # fit the model
12 model.fit(X)
13 # assign a cluster to each example
14 yhat = model.predict(X)
15 # retrieve unique clusters
16 clusters = unique(yhat)
17 # create scatter plot for samples from each cluster
18 for cluster in clusters:
19     # get row indexes for samples with this cluster
20     row_ix = where(yhat == cluster)
21     # create scatter of these samples
22     pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
23 # show the plot
24 pyplot.show()
```

Figura 2-4. Propagación por Afinidad en Python (Nota. elaboración propia)

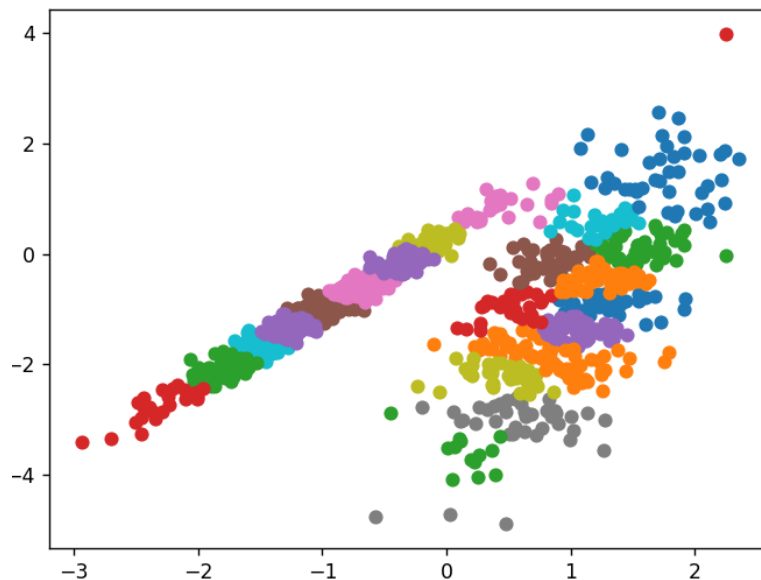


Figura 2-5. Visualización gráfica de Propagación por Afinidad (Nota. elaboración propia)

2.2.1.1.2. BIRCH

El algoritmo BIRCH (Reducción y Agrupación Balanceada Iterativa usando Jerarquías) fue introducido por (Zhang et al., 1998) y está diseñado específicamente para realizar agrupaciones eficientes sobre grandes



volúmenes de datos. Su principal característica radica en el uso de una estructura jerárquica compacta que reduce la carga computacional, sin arruinar la calidad de los resultados.

Igualmente, los autores proponen que BIRCH introduce la estructura denominada CF-Tree (Árbol de Agrupación de Características), la cual almacena y actualiza información relevante de los datos en un espacio de memoria limitado. Cada nodo en el árbol contiene un conjunto de características de agrupación CF (Clustering Features), lo que permite representar los datos y actualizar los clústeres de manera incremental, sin necesidad procesar nuevamente todo el conjunto de datos.

Asimismo, los autores (Zhang et al., 1998) añaden que al haber evaluado la eficiencia espacio/temporal (sensibilidad en el orden de los datos de entrada) y comparándolo con otros algoritmos existentes a través de experimentos, afirman que BIRCH es el mejor método de agrupamiento disponible para bases de datos de gran volumen.

Por tanto, una de sus ventajas más notables es su capacidad para procesar grandes volúmenes de datos haciendo uso eficiente de la memoria y tiempo de cómputo, lo cual lo hace adecuado para entornos con recursos limitados.

(Brownlee, 2020) señala que este algoritmo se implementa mediante la clase Birch, y sus hiperparámetros son:

- **Umbral (threshold):** admite valores positivos.
- **Número de clústeres (n_clusters):** permite valores enteros positivos.



En la Figura 2-6, se muestra un ejemplo básico del algoritmo implementado en Python, mientras que la Figura 2-7 presenta su salida gráfica.

```

1  # birch clustering
2  from numpy import unique
3  from numpy import where
4  from sklearn.datasets import make_classification
5  from sklearn.cluster import Birch
6  from matplotlib import pyplot
7  # define dataset
8  X, _ = make_classification(n_samples=1000, n_features=2, n_informative=2, n_redundant=0, n_clusters_per_class=1, random_state=4)
9  # define the model
10 model = Birch(threshold=0.01, n_clusters=2)
11 # fit the model
12 model.fit(X)
13 # assign a cluster to each example
14 yhat = model.predict(X)
15 # retrieve unique clusters
16 clusters = unique(yhat)
17 # create scatter plot for samples from each cluster
18 for cluster in clusters:
19     # get row indexes for samples with this cluster
20     row_ix = where(yhat == cluster)
21     # create scatter of these samples
22     pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
23 # show the plot
24 pyplot.show()

```

Figura 2-6. BIRCH en Python (Nota. elaboración propia)

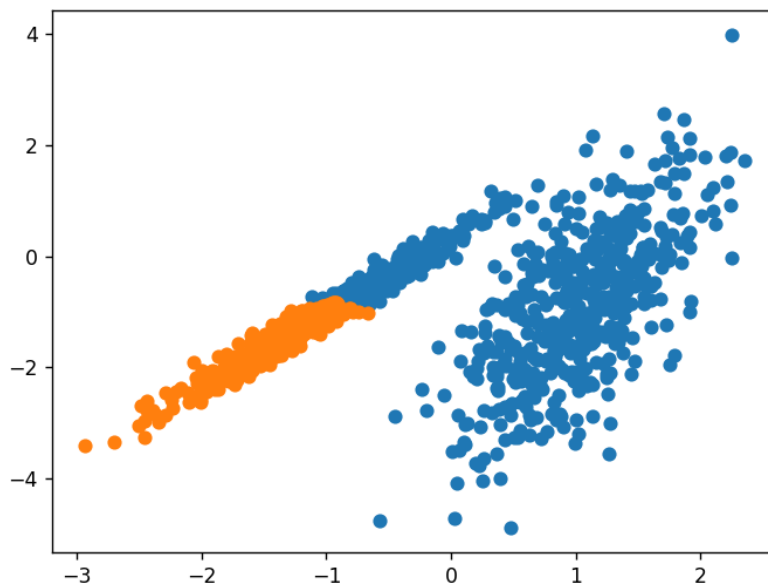


Figura 2-7. Visualización gráfica de BIRCH (Nota. elaboración propia)

2.2.1.1.3. DBSCAN

El algoritmo DBSCAN (Agrupación Espacial Basado en Densidad de Aplicaciones con Ruido) es una técnica de agrupamiento propuesta por (Ester et al., 1996) cuyo rasgo distintivo consiste en que no requiere



especificar el número de clústeres con antelación. En cambio, identifica grupos de forma arbitraria y distingue el ruido presente entre los datos.

También, (Ester et al., 1996) agregan que: “La idea clave de nuestro enfoque es que, por cada punto de un clúster, el vecindario entre un radio establecido debe de contener al menos un número mínimo de puntos”. (pág. 227)

En términos operativos, este algoritmo se basa en dos parámetros fundamentales:

- **Épsilon:** determina el radio de vecindad para considerar un punto como parte de un clúster.
- **Mínimo de puntos:** especifica el número mínimo de puntos necesarios dentro del radio (épsilon) para que un punto sea considerado núcleo de un clúster.

Sin embargo, el rendimiento de DBSCAN puede verse afectado por la elección de estos valores, ya que una mala configuración puede distorsionar la agrupación generada, afectando la calidad de los resultados obtenidos.

De acuerdo con (Brownlee, 2020), este algoritmo se implementa a través de la clase DBSCAN, y sus hiperparámetros son:

- **Épsilon (eps):** admite valores positivos.
- **Mínimo de muestras (min_samples):** acepta únicamente enteros positivos.

En la Figura 2-8, se muestra un ejemplo básico de su implementación en Python, mientras que la Figura 2-9 muestra los resultados visuales obtenidos.



```

1  # dbscan clustering
2  from numpy import unique
3  from numpy import where
4  from sklearn.datasets import make_classification
5  from sklearn.cluster import DBSCAN
6  from matplotlib import pyplot
7  # define dataset
8  X, _ = make_classification(n_samples=1000, n_features=2, n_informative=2, n_redundant=0, n_clusters_per_class=1, random_state=4)
9  # define the model
10 model = DBSCAN(eps=0.30, min_samples=9)
11 # fit model and predict clusters
12 yhat = model.fit_predict(X)
13 # retrieve unique clusters
14 clusters = unique(yhat)
15 # create scatter plot for samples from each cluster
16 for cluster in clusters:
17     # get row indexes for samples with this cluster
18     row_ix = where(yhat == cluster)
19     # create scatter of these samples
20     pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
21 # show the plot
22 pyplot.show()

```

Figura 2-8. DBSCAN en Python (Nota. elaboración propia)

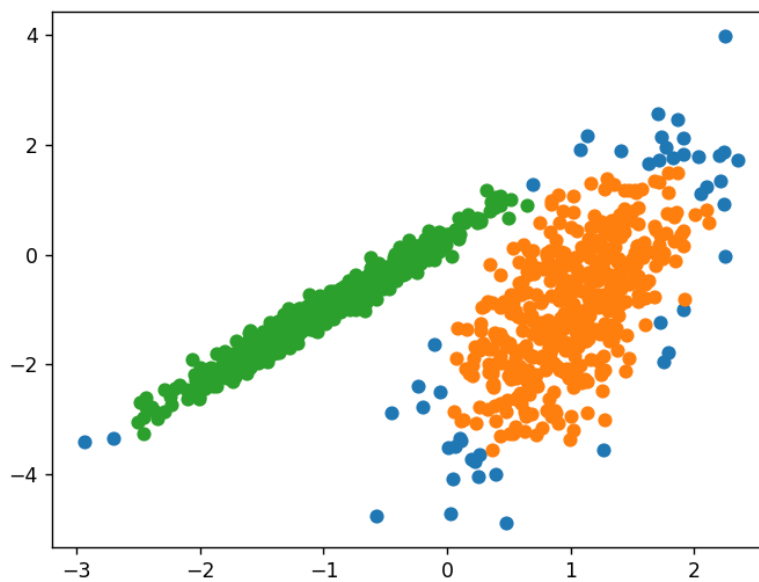


Figura 2-9. Visualización gráfica de DBSCAN (Nota. elaboración propia)

2.2.1.1.4. K-Means

El algoritmo K-Means es una de las técnicas de agrupamiento más utilizadas en la materia, y fue introducido por (MacQueen, 1967). Su objetivo principal es particionar un conjunto de datos en k grupos, asignando cada punto al clúster con el centroide más cercano.

De forma informal, (MacQueen, 1967) también complementa que el procedimiento K-Means consiste en simplemente empezar con grupos k (cada uno siendo un punto aleatorio); posteriormente, se va añadiendo



un punto nuevo al grupo cuyo promedio es el más cercano a éste mismo. Tras haber sido agregado, el promedio de ese grupo es ajustado para tomar en consideración al nuevo punto.

En consecuencia, este método recibe su nombre de la operación reiterada de calcular el promedio (mean) de cada grupo (k).

No obstante, K-Means presenta ciertas limitantes. Por ejemplo, requiere definir el número de clústeres previamente, además de ser sensible a la elección de los centroides iniciales, lo que puede influir en la calidad del resultado final. A pesar de ello, su eficiencia y facilidad de implementación, lo convierten en una herramienta fundamental para el análisis de datos.

Según (Brownlee, 2020), este algoritmo se implementa mediante la clase KMeans y posee un solo hiperparámetro:

- **Número de clústeres (n_clusters):** acepta únicamente valores enteros positivos.

En la Figura 2-10, se muestra un ejemplo básico del algoritmo implementado en Python, mientras que la Figura 2-11 presenta su resultado gráfico.

```
1  # k-means clustering
2  from numpy import unique
3  from numpy import where
4  from sklearn.datasets import make_classification
5  from sklearn.cluster import KMeans
6  from matplotlib import pyplot
7  # define dataset
8  X, _ = make_classification(n_samples=1000, n_features=2, n_informative=2, n_redundant=0, n_clusters_per_class=1, random_state=4)
9  # define the model
10 model = KMeans(n_clusters=2)
11 # fit model and predict clusters
12 yhat = model.fit_predict(X)
13 # retrieve unique clusters
14 clusters = unique(yhat)
15 # create scatter plot for samples from each cluster
16 for cluster in clusters:
17     # get row indexes for samples with this cluster
18     row_ix = where(yhat == cluster)
19     # create scatter of these samples
20     pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
21 # show the plot
22 pyplot.show()
```

Figura 2-10. K-Means en Python (Nota. elaboración propia)



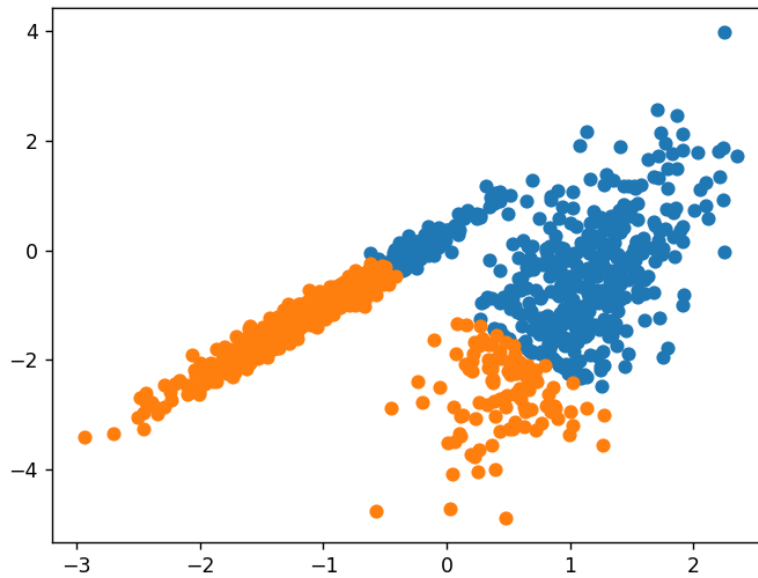


Figura 2-11. Visualización gráfica de K-Means (*Nota. elaboración propia*)

2.2.1.1.5. Mini-Batch K-Means

El algoritmo Mini-Batch K-Means fue introducido por (Sculley, 2010) como una variante optimizada de K-Means, diseñada específicamente para mejorar el rendimiento en grandes conjuntos de datos y reducir el uso de memoria y costo computacional.

A diferencia del algoritmo K-Means convencional, que recalcula los centroides utilizando todo el conjunto de datos en cada iteración, la variante emplea pequeños subconjuntos aleatorios de datos (mini-batches) para actualizar los centroides. Esta estrategia permite acelerar significativamente el proceso de agrupamiento, manteniendo una calidad comparable a la del algoritmo original.

Siguiendo con el autor (Sculley, 2010), se propone el uso de optimización de subconjuntos aleatorios de datos para la agrupación de datos K-Means. La motivación para haber desarrollado este método es que dichos subconjuntos tienden a tener un ruido estocástico más bajo que ejemplos individuales en SGD (permitiendo converger a mejores soluciones), pero no sufren de un costo computacional elevado cuando los grupos de datos aumentan de tamaño con ejemplos redundantes.

Como resultado, este algoritmo se convierte en una opción adecuada para aplicaciones que manejan grandes volúmenes de datos, tales como el análisis de texto, clasificación de imágenes, entre otros; a causa de la rapidez y escalabilidad que ofrece.

De acuerdo con (Brownlee, 2020), este algoritmo se implementa mediante la clase MiniBatchKMeans y requiere un único hiperparámetro:

- **Número de clústeres (n_clusters):** solo acepta valores enteros positivos.

En la Figura 2-12, se presenta un ejemplo básico del algoritmo en Python, mientras que en la Figura 2-13 se muestra el resultado visual correspondiente.

```
1 # mini-batch k-means clustering
2 from numpy import unique
3 from numpy import where
4 from sklearn.datasets import make_classification
5 from sklearn.cluster import MiniBatchKMeans
6 from matplotlib import pyplot
7 # define dataset
8 X, _ = make_classification(n_samples=1000, n_features=2, n_informative=2, n_redundant=0, n_clusters_per_class=1, random_state=4)
9 # define the model
10 model = MiniBatchKMeans(n_clusters=2)
11 # fit the model
12 model.fit(X)
13 # assign a cluster to each example
14 yhat = model.predict(X)
15 # retrieve unique clusters
16 clusters = unique(yhat)
17 # create scatter plot for samples from each cluster
18 for cluster in clusters:
19     # get row indexes for samples with this cluster
20     row_ix = where(yhat == cluster)
21     # create scatter of these samples
22     pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
23 # show the plot
24 pyplot.show()
```

Figura 2-12. Mini-Batch K-Means en Python (Nota. elaboración propia)

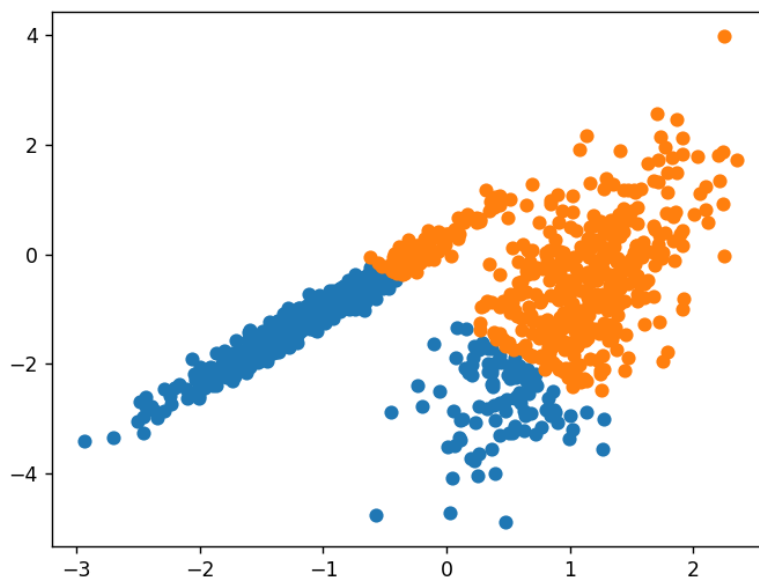


Figura 2-13. Visualización gráfica de Mini-Batch K-Means (Nota. elaboración propia)

2.2.1.2. Métricas de distancia

(Ahmad, 2023) establece que la eficacia de los algoritmos de agrupamiento depende directamente de la capacidad para medir con precisión las similitudes entre los datos. A este proceso se le denomina métrica de distancia, debido a que determina qué tan cercanos o diferentes son los puntos de datos entre sí.

Por otra parte, (Ahmad, 2023) complementa que una métrica de distancia es una fórmula o método matemático que calcula la distancia o similitud entre dos puntos de datos. En este contexto, es fundamental entender que el término “distancia” no se refiere a la distancia física, sino a la similitud o disimilitud entre los puntos de datos en función de sus rasgos o características.

En este sentido, en la agrupación de datos se distinguen dos tipos principales de distancias:

- **Interclúster:** distancia entre diferentes grupos de datos (clúster).
- **Intraclúster:** distancia entre puntos pertenecientes al mismo grupo de datos.

En otras palabras, el objetivo ideal de un algoritmo de agrupamiento es maximizar la distancia interclúster (grupos bien definidos) y minimizar la distancia intraclúster (grupos homogéneos). Esta lógica se representa gráficamente en la Figura 2-14.

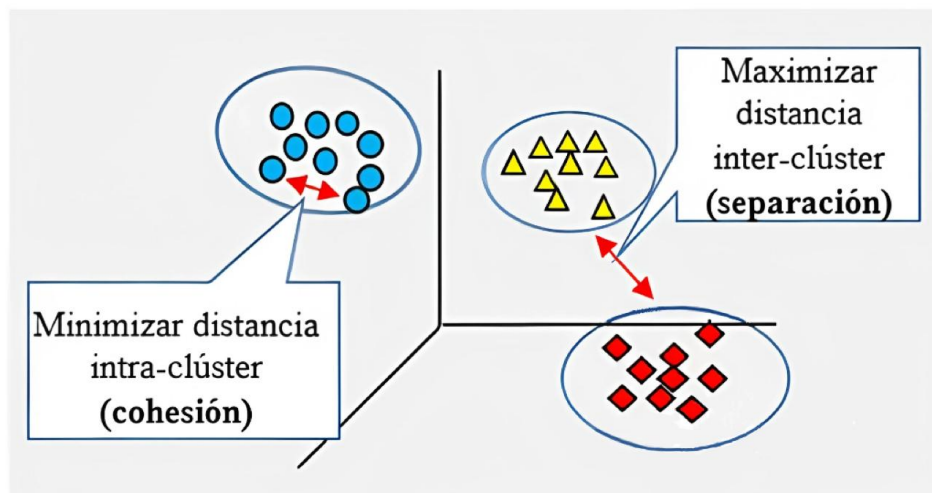


Figura 2-14. Distancia Interclúster vs Distancia Intraclúster (Nota. López Ollarbez & Gómez Abreu, 2021, pág. 10)

2.2.1.2.1. Métrica de distancia euclidiana

(Chugani, 2024) describe la distancia euclidiana como la representación del camino más corto entre dos puntos en un espacio euclidiano. Conviene subrayar que esta distancia se calcula con el teorema de

Pitágoras, el cual establece que, en un triángulo rectángulo, el cuadrado de la longitud de la hipotenusa es igual a la suma de los cuadrados de los catetos. Dicho concepto se representa en la Figura 2-15.

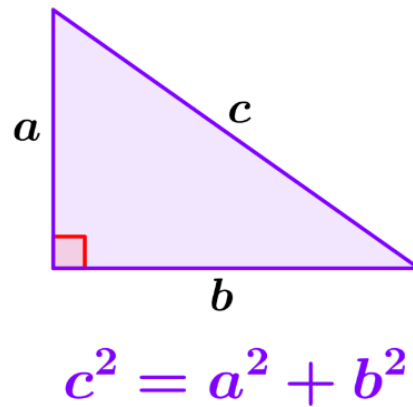


Figura 2-15. Teorema de Pitágoras (Nota. tomado de Neurochispas, s.f.)

En un plano bidimensional, la distancia euclidiana se calcula aplicando la fórmula observada en la Ecuación 1:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Ecuación 1. Fórmula de la distancia euclidiana en 2D (Nota. tomado de Chugani, 2024)

Como se muestra en la Figura 2-16, la línea roja representa la distancia directa entre puntos, mientras que las líneas punteadas forman el triángulo rectángulo que ilustra dicha relación.

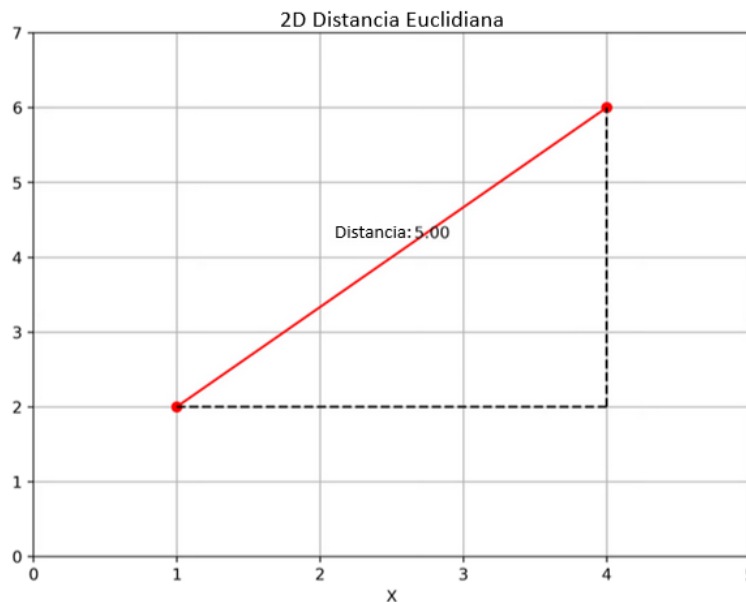


Figura 2-16. Representación visual de la distancia euclidiana en 2D (Nota. tomado de Chugani, 2024)



En un plano tridimensional, la fórmula se adapta como se muestra en la Ecuación 2:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Ecuación 2. Fórmula de la distancia euclidiana en 3D (Nota. tomado de Chugani, 2024)

Según la representación gráfica de la Figura 2-17, la línea verde y punteada representa la distancia directa entre puntos.

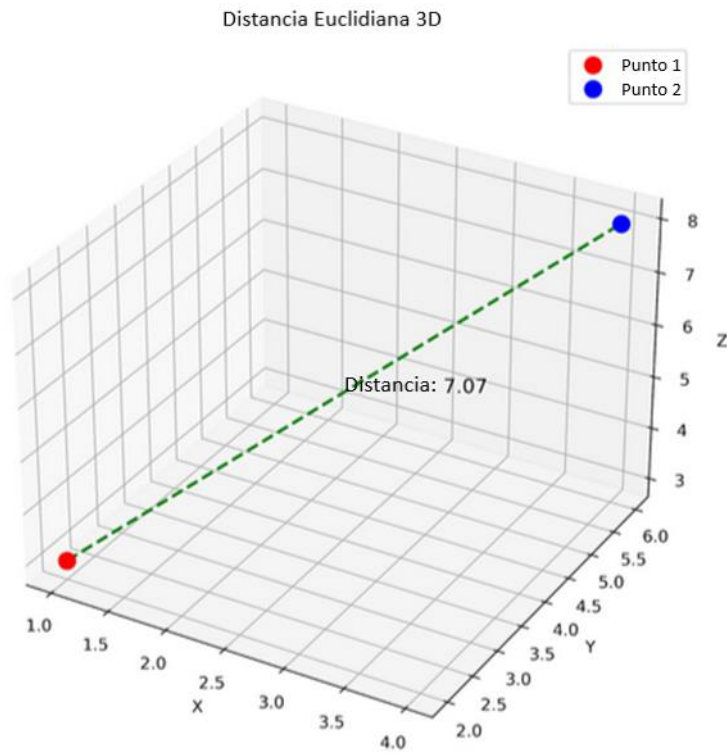


Figura 2-17. Representación visual de la distancia euclidiana en 3D (Nota. tomado de Chugani, 2024)

Por último, en un espacio de n dimensiones, la fórmula se generaliza como puede apreciarse en la Ecuación 3:

$$d = \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2 + \dots + (b_n - a_n)^2}$$

Ecuación 3. Fórmula de la distancia euclidiana en n dimensiones (Nota. tomado de Chugani, 2024)

2.2.1.2.2. Métrica de distancia Manhattan

Por otro lado, (Chugani, 2024) define que la distancia Manhattan mide la suma de las diferencias absolutas entre las coordenadas cartesianas de dos puntos. A diferencia de la distancia euclidiana, esta métrica simula un recorrido en forma de cuadrícula, como si se tratara de un desplazamiento por calles rectangulares.

En un plano bidimensional, la distancia Manhattan se calcula aplicando la fórmula presente en la Ecuación 4:

$$\text{Distancia Manhattan} = |x_1 - x_2| + |y_1 - y_2|$$

Ecuación 4. Fórmula de la distancia Manhattan en 2D (Nota. tomado de Chugani, 2024)

La Figura 2-18 ilustra la trayectoria en forma de cuadrícula, en donde la línea azul representa la distancia entre dos puntos.

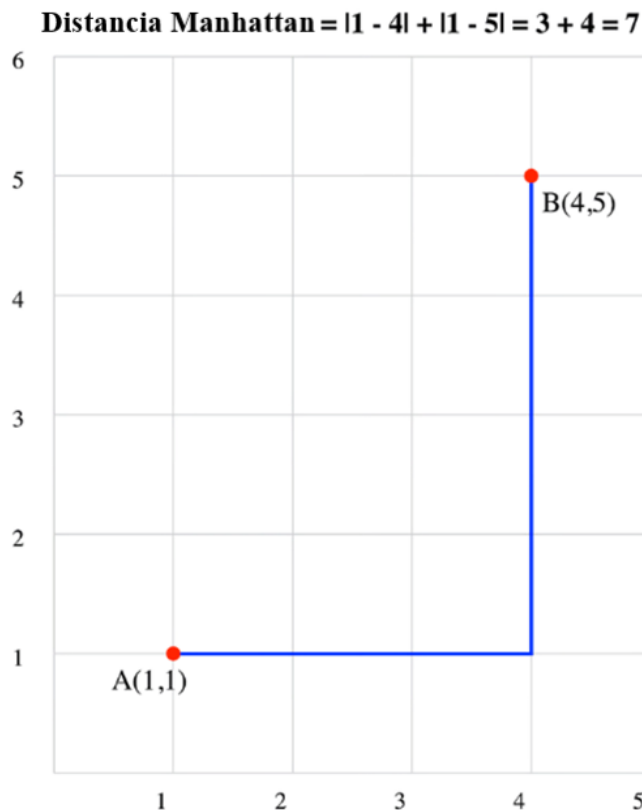


Figura 2-18. Representación visual de la distancia Manhattan en 2D (Nota. tomado de Chugani, 2024)

En un plano tridimensional, la fórmula se extiende como se muestra en la Ecuación 5:

$$\text{Distancia Manhattan} = |x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2|$$

Ecuación 5. Fórmula de la distancia Manhattan en 3D (Nota. tomado de Chugani, 2024)



Para finalizar, en un espacio de n dimensiones, la fórmula se generaliza conforme a la Ecuación 6:

$$\text{Distancia Manhattan} = |x_1 - x_2| + |y_1 - y_2| + \dots + |v_{n1} - v_{n2}|$$

Ecuación 6. Fórmula de la distancia Manhattan en n dimensiones (Nota. tomado de Chugani, 2024)

2.2.1.3. Agrupación Jerárquica de Clústeres (HCA)

Según (Noble, 2024), la agrupación jerárquica de clústeres es un algoritmo de aprendizaje no supervisado, cuyo objetivo consiste en agrupar datos en forma de un árbol de clústeres anidados. Esta técnica permite identificar patrones y conexiones dentro de los conjuntos de datos con los que se trabaja. Una vez implementado, sus resultados se representan mediante un diagrama llamado dendograma, que muestra diferentes grados de granularidad y visualiza de forma clara las distancias entre los clústeres generados. (Figura 2-19)

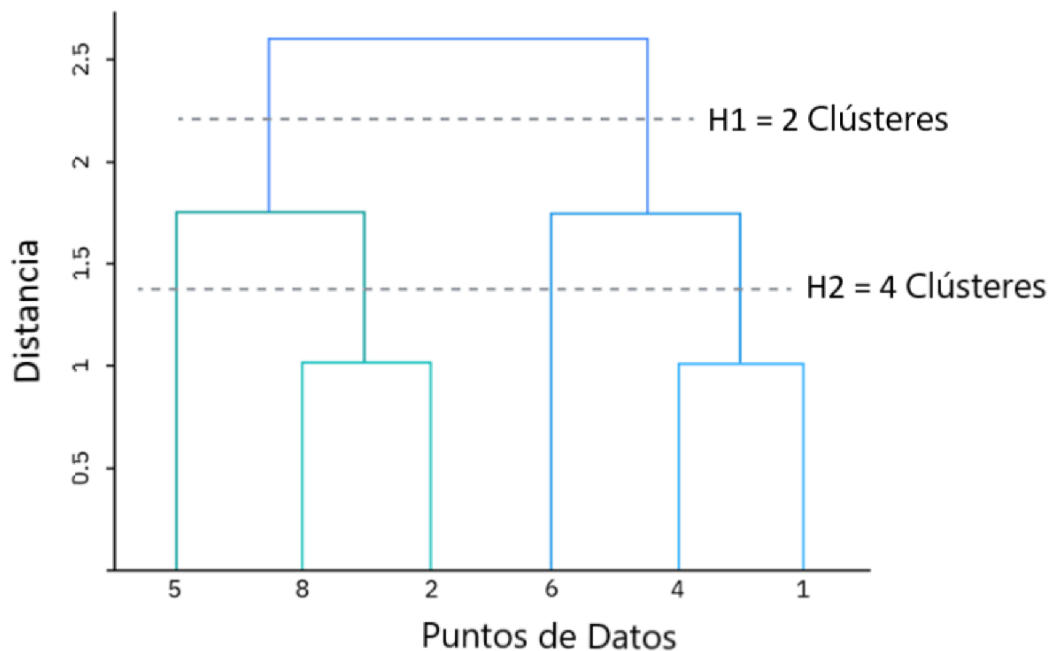


Figura 2-19. Ejemplo dendograma de algoritmo jerárquico (Nota. tomado de Noble, 2024)

De acuerdo con (Noble, 2024), existen dos enfoques principales en el HCA:

- **Enfoque aglomerativo:** fusiona repetidamente los grupos de datos en otros más grandes hasta formar un único clúster.
- **Enfoque divisivo:** inicia con todos los datos en un único grupo y los divide sucesivamente hasta que cada uno queda en su propio clúster.



Igualmente, (Maimon & Rokach, 2010) destacan: “Los algoritmos jerárquicos tienen dos ventajas básicas. Primero, el número de clases no necesita ser especificado de antemano, y segundo, son independientes de condiciones iniciales”. (pág. 479)

Es decir, una de sus principales fortalezas radica en que, a diferencia de algoritmos como K-Means, no requiere definir previamente la cantidad de clústeres.

No obstante, la elección de la métrica de distancia (euclidiana, Manhattan, etc.) y otros parámetros puede influir considerablemente en los resultados. Por tanto, es importante seleccionar estas opciones según la naturaleza del conjunto de datos y los objetivos del análisis.

2.2.2. Reducción de dimensionalidad

Actualmente, los conjuntos de datos de alto volumen se han vuelto comunes, lo que dificulta su interpretación y análisis. Ante este escenario, la reducción de dimensionalidad emerge como una técnica esencial.

Según (Murel & Kavlakoglu, 2024), esta técnica busca representar un conjunto de datos utilizando un menor número de características (dimensiones), sin perder información esencial. Para ello, se eliminan variables irrelevantes o redundantes, construyendo un nuevo modelo más eficiente. Asimismo, los autores señalan que existen diversos métodos según el tipo de problema presente, entre los cuales destaca el Análisis de Componentes Principales (PCA) por su eficacia y amplia adopción.

2.2.2.1. Análisis de Componentes Principales (PCA)

El Análisis de Componentes Principales es una técnica estadística que transforma variables posiblemente correlacionadas en un nuevo conjunto de variables no correlacionadas, dando como resultado a los componentes principales.

Cada componente sucesivo maximiza la varianza restante del conjunto original, tomando como base la fórmula de la Ecuación 7. Esta transformación facilita la interpretación de los datos más sencilla, además que minimiza la pérdida de información.

Como afirman (Cadima & Jolliffe, 2016): “PCA es una técnica para reducir la dimensionalidad de un conjunto de datos, aumentando la interpretabilidad, pero al mismo tiempo, minimizando la pérdida de información”. (pág. 2)



$$\pi_j = \frac{\lambda_j}{\sum_{j=1}^p \lambda_j} = \frac{\lambda_j}{tr(S)'}$$

Ecuación 7. Fórmula varianza total para Análisis de Componentes Principales (Nota. tomado de Cadima & Jolliffe, 2016, pág. 4)

Los símbolos de dicha fórmula representan lo siguiente:

- π_j : proporción de la varianza explicada por el componente principal j
- λ_j : j-ésimo valor propio de la matriz de covarianza $\mathbf{S}$
- $tr(\mathbf{S})'$: traza de $\mathbf{S}$, suma de todos los valores propios (varianza total)

Según (Cadima & Jolliffe, 2016), aplicar esta técnica implica resolver un problema con autovalores y autovectores. En consecuencia, permite generar los componentes principales directamente de los datos, sin involucrar supuestos previos, lo que convierte a PCA en una técnica adaptativa que se ajusta a distintos conjuntos de datos.

2.3. Herramientas y recursos

2.3.1. Python

Python es un lenguaje de programación interpretado, de alto nivel y propósito general, que ha ganado una amplia aceptación en la comunidad científica debido a su sintaxis clara, una filosofía orientada a la legibilidad y un robusto ecosistema de bibliotecas.

Según (Oliphant, 2007), Python presenta diversas características que lo hacen especialmente valioso en el área de la computación científica:

- El hecho de que Python funcione en tantas plataformas significa que no hay que preocuparse por realizar una aplicación con portabilidad limitada, lo que también ayuda a evitar la dependencia de un solo proveedor (vendor lock-in).
- La sintaxis limpia y sofisticada del lenguaje permite programar de manera procedimental o completamente orientada a objetos, dependiendo de la situación presentada.



- Tener un potente compilador interactivo permite el desarrollo de código en tiempo real y la experimentación en vivo, eliminando así el paso de compilación que consume tiempo y reduce la productividad en el proceso de desarrollo de código.
- La capacidad de extender Python con tu propio código compilado significa que este lenguaje puede ser utilizado para hacer cualquier cosa tan rápido como lo permita el hardware que se tiene a la mano.
- Su gran número de librerías permite la construcción rápida de programas sofisticados, como un servidor web que resuelve ecuaciones diferenciales parciales, una librería distribuida de procesamiento de imágenes con balanceo automático de carga o una aplicación que recopila datos de Internet, publica resultados en una base de datos y te envía correos electrónicos con reportes de progreso.
- La existencia de enlaces (bindings) de Python relacionados a todos los principales conjuntos de herramientas de interfaces gráficas (GUI toolkits) significa que se pueden aplicar técnicas de desarrollo rápido al construir una interfaz de usuario.

Como resultado, uno de los principales atractivos de Python es su versatilidad, especialmente en tareas de análisis de datos, visualización, modelado estadístico y desarrollo de aplicaciones. Por otra parte, su sintaxis accesible lo convierte en una opción ideal para investigadores y analistas con poca experiencia en programación.

2.3.1.1. Librerías

2.3.1.1.1. NumPy

NumPy es una librería esencial para el desarrollo científico en Python. (VanderPlas, 2016) establece que su principal fortaleza se encuentra en el objeto ndarray, ya que permite trabajar de manera eficiente con arreglos multidimensionales, utilizando una sintaxis clara y operaciones vectorizadas. En concreto, este objeto reemplaza a las listas nativas de Python cuando se requieren operaciones numéricas de alto rendimiento.

Además, proporciona una gama extensa de funciones matemáticas (como operaciones estadísticas, generación de números aleatorios, funciones de algebra lineal, etc.) sin necesidad de bucles explícitos.

Otra ventaja clave de NumPy es su interoperabilidad con otras librerías del ecosistema científico, como Pandas, SciPy, Scikit-learn, etc., muchas de las cuales dependen de sus estructuras internas para almacenar



datos (NumPy Development Team, s.f.). También incluye funciones para entrada y salida de archivos, lo que facilita el manejo de grandes volúmenes de datos.

2.3.1.1.2. Pandas

Pandas es una librería de código abierto orientada al análisis y manipulación de datos en Python. Está diseñada para trabajar de manera eficiente con estructuras de datos como:

- **Series:** vectores unidimensionales etiquetados.
- **DataFrames:** tablas bidimensionales con filas y columnas etiquetadas.

Según la documentación oficial (Pandas Development Team, s.f.), esta herramienta ofrece funcionalidades clave como lectura y escritura de datos en diferentes formatos (CSV, Excel, SQL), filtrado, agrupamiento y transformación de datos, así como la limpieza de registros vacíos o erróneos. Todo esto se realiza con una sintaxis clara y legible, facilitando su comprensión incluso para usuarios principiantes en el tema.

Por otro lado, Pandas es altamente compatible con otras librerías del ambiente Python (NumPy, Scikit-learn, Matplotlib, etc), lo que permite su integración en flujos de trabajo más complejos, como en proyectos sobre ciencia de datos o aprendizaje automático.

2.3.1.1.3. Scikit-learn

Scikit-learn es una librería de código abierto ampliamente utilizada para el desarrollo de aprendizaje automático en Python. Fue construida sobre librerías fundamentales como NumPy, Matplotlib y SciPy, y ofrece una interfaz coherente para una gran variedad de algoritmos (Scikit-learn Development Team, s.f.)

Siguiendo con la documentación oficial, este proyecto fue iniciado por David Cournapeau en 2007 durante el Google Summer of Code (GSoC). Posteriormente, Matthieu Brucher comenzó a trabajar en el proyecto como parte de su tesis.

En 2010, investigadores del Instituto Nacional de Búsqueda de Información en Automático (INRIA) como Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort y Vincent Michel tomaron las riendas del proyecto y lanzaron la primera versión estable el 1 de febrero del 2010. En la actualidad, una gran comunidad internacional ha estado liderando el desarrollo, dichos miembros que la integran aparecen en la Figura 2-20.



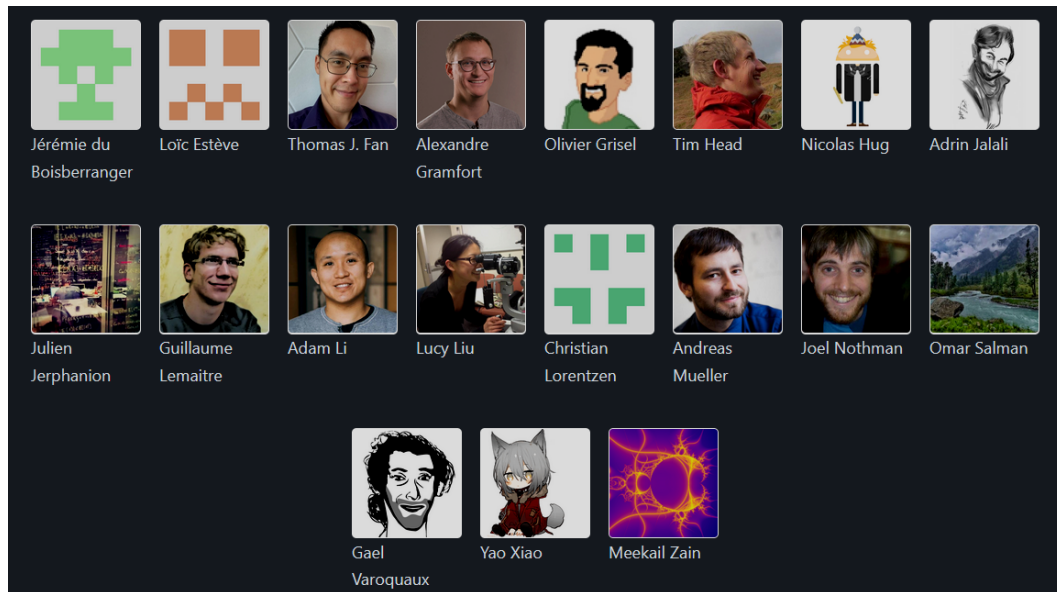


Figura 2-20. Equipo de desarrollo y mantenimiento de Scikit-learn (Nota. tomado de Scikit-learn Development Team, s.f.)

Entre sus principales funcionalidades se encuentran:

- Algoritmos de clasificación, regresión y agrupamiento
- Reducción de dimensionalidad (PCA)
- Selección de características y herramientas para el preprocesamiento de datos
- Evaluación de modelos, ajuste de hiperparámetros y validación cruzada
- Canalizaciones (pipelines) para flujos de trabajo más complejos

Gracias a su versatilidad y la comunidad activa que la respalda, Scikit-learn se ha consolidado como una herramienta indispensable para el desarrollo de proyectos sobre análisis predictivo y ciencia de datos.

2.3.1.1.4. SciPy

SciPy es una librería de código abierto en Python que proporciona herramientas y algoritmos avanzados para los campos de matemáticas, ingeniería y ciencia (SciPy Development Team, s.f.). Está construida sobre NumPy y extiende sus capacidades mediante módulos especializados en tareas de:

- Optimización
- Integración numérica
- Álgebra lineal
- Ecuaciones diferenciales



- Procesamiento de señales
- Estadística

Continuando con su documentación oficial, una de las principales ventajas de esta librería radica en su capacidad para resolver problemas científicos complejos mediante una sintaxis clara y modular. Por ejemplo, los submódulos `scipy.optimize`, `scipy.integrate`, `scipy.linalg`, `scipy.signal` y **`scipy.cluster`** permiten importar únicamente las funciones necesarias, lo que contribuye a la eficiencia computacional.

Igualmente, esta librería es ampliamente utilizada en proyectos de investigación académica, científica e industrial, y cuenta con una comunidad internacional activa que garantiza su mantenimiento y evolución constante, algunos miembros que la conforman se pueden visualizar en la Figura 2-21.

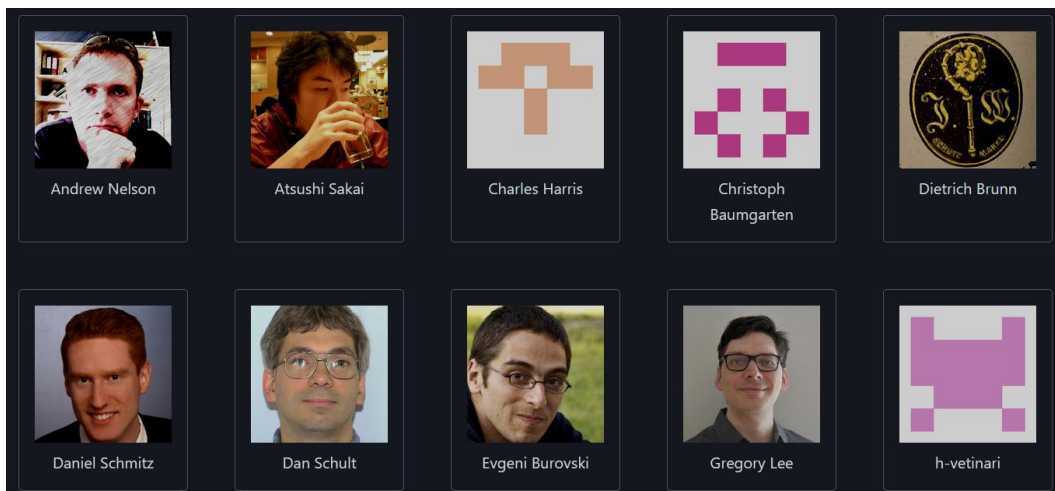


Figura 2-21. Parte del equipo de desarrollo y mantenimiento de SciPy (*Nota. tomado de SciPy Development Team, s.f.*)

2.3.1.1.5. Matplotlib

Matplotlib es una de las librerías más utilizadas para la visualización de datos en Python. Permite generar gráficos de alta calidad, tanto estáticos, como animados e interactivos, facilitando la interpretación visual de datos complejos.

Según el (Matplotlib Development Team, s.f.), esta biblioteca admite la creación de:

- Diagramas de dispersión
- Gráficos de barras y líneas
- Histogramas
- Subgráficos personalizados con múltiples ejes y leyendas

En la Figura 2-22 se muestra un ejemplo de código implementando la librería, acompañado por la Figura 2-23 mostrando su ejecución (Matplotlib Development Team, s.f.).

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 plt.style.use('_mpl-gallery')
5
6 # make the data
7 np.random.seed(3)
8 x = 4 + np.random.normal(0, 2, 24)
9 y = 4 + np.random.normal(0, 2, len(x))
10 # size and color:
11 sizes = np.random.uniform(15, 80, len(x))
12 colors = np.random.uniform(15, 80, len(x))
13
14 # plot
15 fig, ax = plt.subplots()
16
17 ax.scatter(x, y, s=sizes, c=colors, vmin=0, vmax=100)
18
19 ax.set(xlim=(0, 8), xticks=np.arange(1, 8),
20       | ylim=(0, 8), yticks=np.arange(1, 8))
21
22 plt.show()
```

Figura 2-22. Código implementando la librería Matplotlib (Nota. elaboración propia)

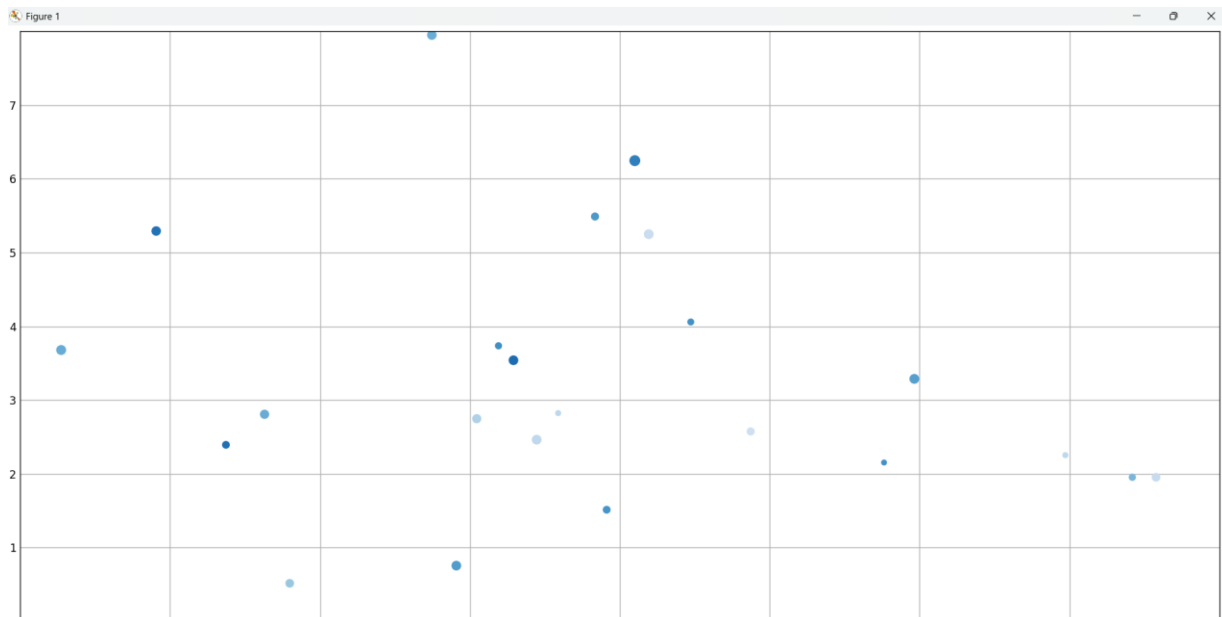


Figura 2-23. Resultado de la ejecución del código: gráfico de dispersión (Nota. elaboración propia)

Además, esta herramienta soporta diversos formatos de salida (PNG, JPG, TIFF, PDF, SVG), lo cual resulta ideal para publicaciones académicas o profesionales. También es compatible con estructuras de datos como arreglos o DataFrames, lo cual favorece su integración con otras librerías del ambiente Python.



Retomando lo expuesto por (VanderPlas, 2016), Matplotlib destaca por su alto nivel de personalización y control sobre los elementos gráficos, lo cual la convierte en una herramienta clave en entornos de análisis de datos.

2.3.1.1.6. Plotly

Plotly es una librería especializada en la creación de gráficos interactivos y visualmente atractivos en Python. A diferencia de otras herramientas de visualización, Plotly se destaca por su capacidad de ofrecer interactividad de alto nivel, permitiendo al usuario ampliar, desplazar y explorar datos dinámicamente (Plotly Development Team, s.f.). Esta característica facilita la comprensión de relaciones y patrones complejos conjuntos de datos voluminosos, mejorando la experiencia del usuario.

Asimismo, entre los tipos de gráficos soportados se encuentran:

- Gráficos de líneas, barras y dispersión
- Histogramas
- Gráficos en 3D
- Diagramas choropleth (mapas geográficos)
- Análisis de series temporales

En la Figura 2-24 se muestra un ejemplo de código con Plotly, mientras que en la Figura 2-25 se encuentra su resultado visual (Ter Sahakyan, 2019).

```
1 import plotly.express as px
2
3 gapminder = px.data.gapminder()
4 fig = px.scatter(gapminder.query("year==2007"), x="gdpPercap", y="lifeExp", size="pop", color="continent",
5 | | | hover_name="country", log_x=True, size_max=60)
6
7 fig.show()
```

Figura 2-24. Código implementando la librería Plotly (**Nota. elaboración propia**)



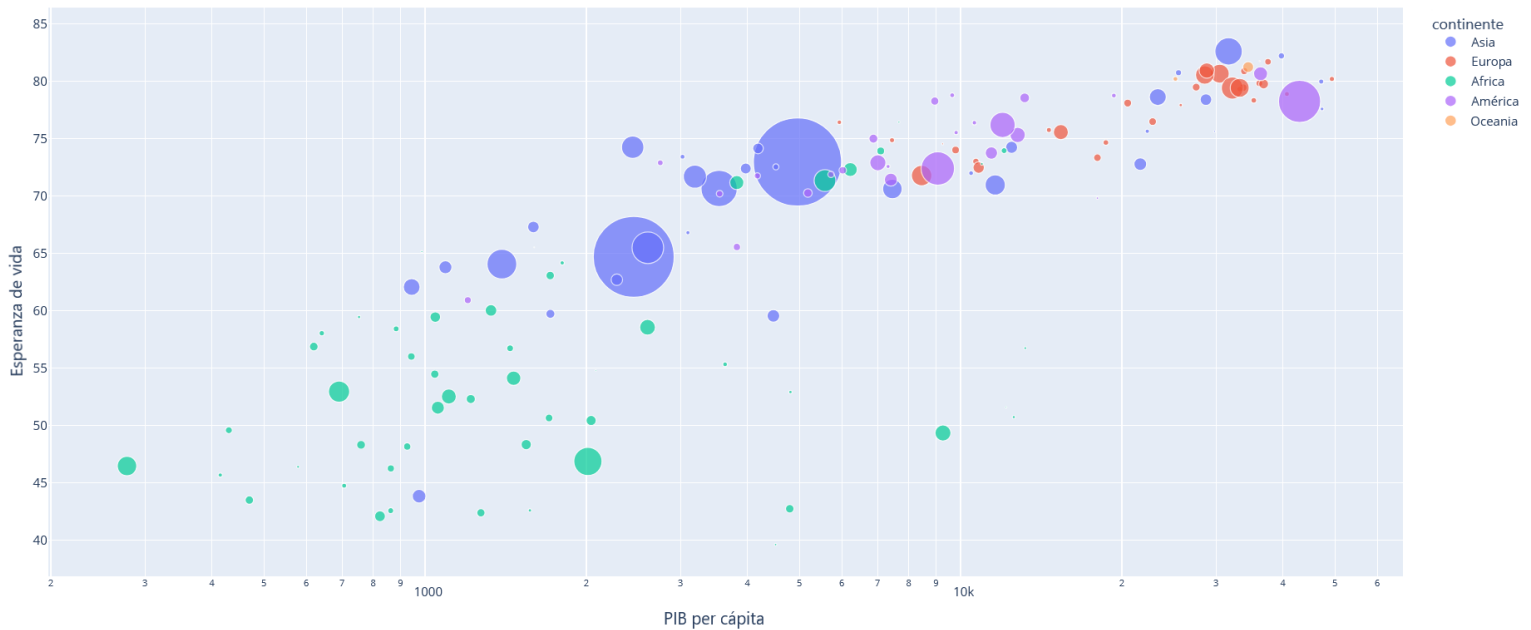


Figura 2-25. Resultado de la ejecución del código: diagrama de burbujas (*Nota. elaboración propia*)

Gracias a su versatilidad, Plotly es comúnmente integrado en entornos analíticos avanzados, así como en aplicaciones web a través de estructuras de trabajo (frameworks) como Dash.

2.3.1.1.7. Dash

Dash es una estructura de trabajo (framework) de código abierto creado por los desarrolladores de Plotly, orientado al desarrollo de aplicaciones web interactivas con Python. Su principal ventaja consiste en que permite construir tableros de control (dashboards) analíticos sin necesidad de utilizar otros lenguajes de programación dedicados a la programación web, como JavaScript o React.

Según la documentación oficial (Plotly Development Team, s.f.), su arquitectura se basa en tres componentes principales:

- **Componentes de interfaz de usuario:** botones, gráficos, controles deslizantes, menús desplegables, cuadros de texto, etc.
- **Callbacks:** gestionan la interactividad entre los componentes de forma dinámica y reactiva.
- **Gráficos generados por Plotly:** que pueden representar datos geográficos, series temporales o visualizaciones más complejas.

Debido a su diseño modular, Dash es una herramienta versátil en proyectos donde se requiera desarrollar interfaces dinámicas para análisis de datos. Por ejemplo, la Figura 2-26 muestra un ejemplo de código implementando la estructura, mostrando el resultado de su ejecución en la Figura 2-27 (Plotly Development Team, s.f.)

```

1  from dash import Dash, dash_table, dcc, callback, Output, Input
2  import pandas as pd
3  import plotly.express as px
4  import dash_mantine_components as dmc
5
6  df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/gapminder2007.csv')
7
8  app = Dash()
9
10 app.layout = dmc.Container([
11     dmc.Title('My First App with Data, Graph, and Controls', c="blue", size="h3"),
12     dmc.RadioGroup(
13         [dmc.Radio(i, value=i) for i in ['pop', 'lifeExp', 'gdpPercap']],
14         id='my-dmc-radio-item',
15         value='lifeExp',
16         size="sm"
17     ),
18     dmc.Grid([
19         dmc.Col([
20             dash_table.DataTable(data=df.to_dict('records'), page_size=12, style_table={'overflowX': 'auto'})
21         ], span=6),
22         dmc.Col([
23             dcc.Graph(figure={}, id='graph-placeholder')
24         ], span=6),
25     ]),
26 ], fluid=True)
27
28
29 @callback(
30     Output(component_id='graph-placeholder', component_property='figure'),
31     Input(component_id='my-dmc-radio-item', component_property='value')
32 )
33 def update_graph(col_chosen):
34     fig = px.histogram(df, x='continent', y=col_chosen, histfunc='avg')
35     return fig
36
37 if __name__ == '__main__':
38     app.run(debug=True)

```

Figura 2-26. Código implementando Dash de Python (Nota. elaboración propia)



Mi Primera Aplicación con Datos, Gráficas y Controles

☐ población
 ☒ Esperanza de vida
 ☐ PIB per cápita

país	población	continente	Esperanza de vida	PIB per cápita
Afghanistan	31889923	Asia	43.828	974.5803384
Albania	3600523	Europe	76.423	5937.029525999999
Algeria	33333216	Africa	72.301	6223.367465
Angola	12420476	Africa	42.731	4797.231267
Argentina	40301927	Americas	75.32	12779.37964
Australia	20434176	Oceania	81.235	34435.367439999995
Austria	8199783	Europe	79.829	36126.4927
Bahrain	708573	Asia	75.635	29796.04834
Bangladesh	150448339	Asia	64.062	1391.253792
Belgium	10392226	Europe	79.441	33692.60508
Benin	8078314	Africa	56.728	1441.284873
Bolivia	9119152	Americas	65.554	3822.137084

<< < 1 / 12 > >>

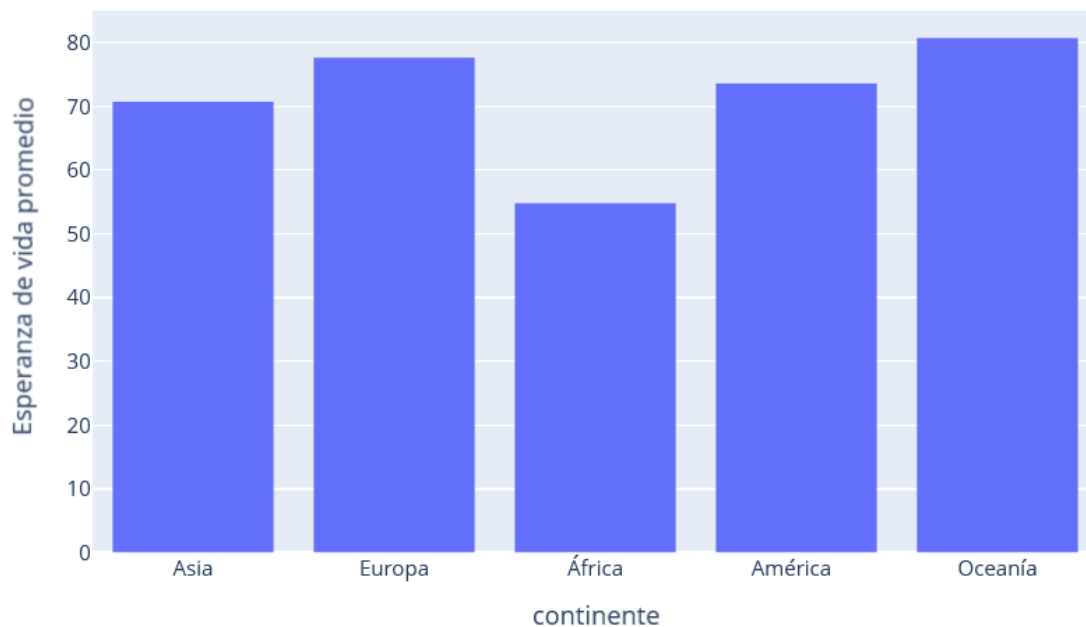


Figura 2-27. Resultado de la ejecución del código: gráficos de barras para tres columnas contenidas en el dataset (*Nota. elaboración propia*)

2.3.2. Render

Render es una plataforma de Infraestructura como Servicio (IaaS) que permite desplegar aplicaciones web, interfaz de programación de aplicaciones (APIs) y microservicios de forma sencilla. Ofrece una alternativa

moderna al alojamiento tradicional, facilitando un entorno de despliegue continuo con integración directa desde repositorios de código como GitHub y GitLab (Render Development Team, s.f.).

Una de sus principales fortalezas es la automatización del ciclo de despliegue, ya que una vez configurado el repositorio y las variables del entorno, Render detecta cambios en el código y reconstruye automáticamente la aplicación. Como resultado, los desarrolladores pueden concentrarse en el código sin preocuparse por la configuración del servidor. La interfaz que proporciona al inicializar un proyecto se puede apreciar en la Figura 2-28.

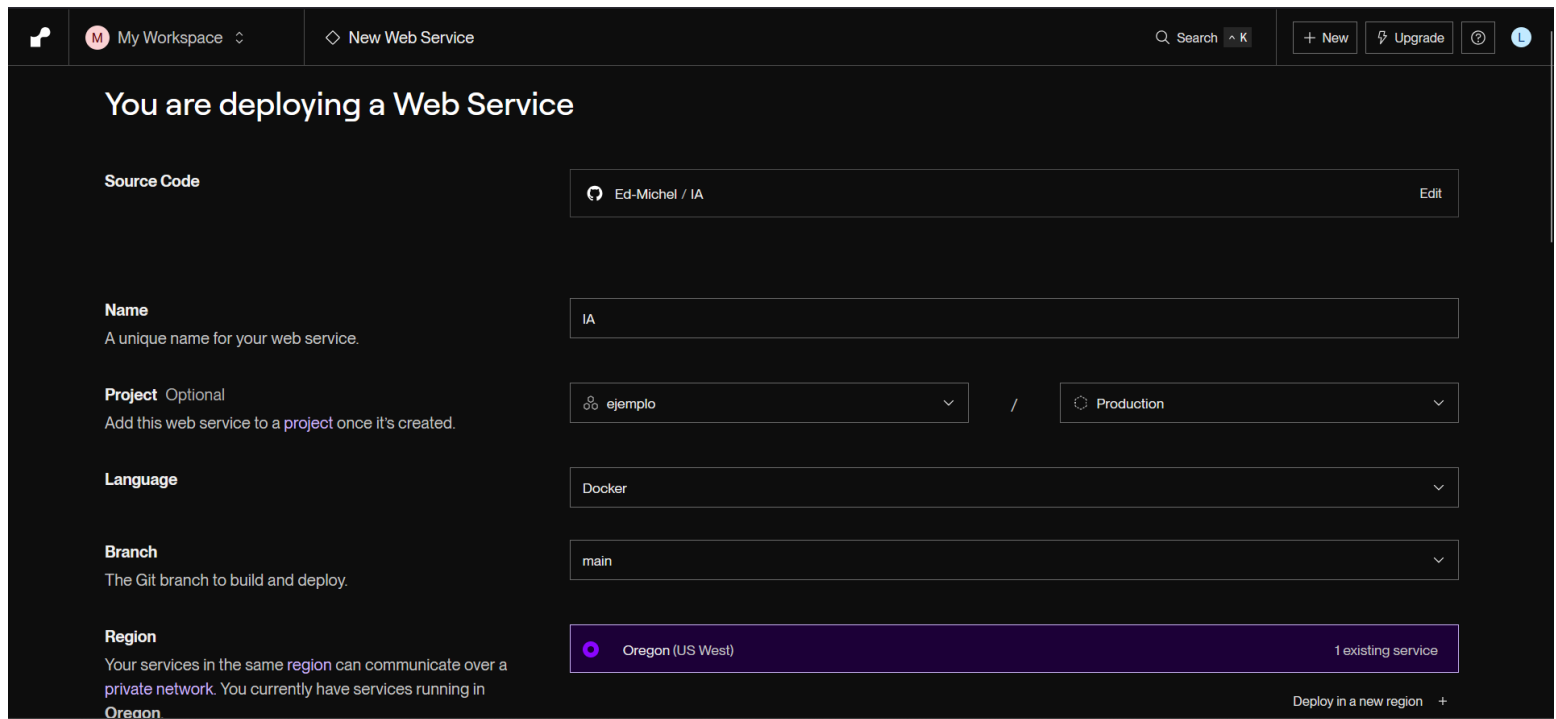
The image shows the 'New Web Service' form in the Render GUI. The form is titled 'You are deploying a Web Service'. It has several sections: 'Source Code' with a dropdown for 'Ed-Michel / IA' and an 'Edit' button; 'Name' with a text input 'IA' and a description 'A unique name for your web service.'; 'Project' with a dropdown 'ejemplo' and a description 'Add this web service to a project once it's created.'; 'Language' with a dropdown 'Docker'; 'Branch' with a dropdown 'main' and a description 'The Git branch to build and deploy.'; and 'Region' with a dropdown 'Oregon (US West)' and a description 'Your services in the same region can communicate over a private network. You currently have services running in Oregon'. There is also a 'Deploy in a new region' link.

Figura 2-28. Render GUI al inicializar un proyecto (*Nota. elaboración propia*)

En el caso de OSMIAA, la aplicación desarrollada en Dash fue montada en Render, aprovechando su compatibilidad con aplicaciones web dinámicas en Python.

Al mismo tiempo, Render ofrece otras características como escalado automático, balanceo de carga y monitoreo en tiempo real, de modo que la convierte en una opción sólida para proyectos académicos o profesionales que requieran confiabilidad sin complejidades técnicas.



Capítulo 3

3. Estado del arte

La calidad del agua es un recurso esencial para la salud humana, el desarrollo socioeconómico y un equilibrio ecológico, por lo que su monitoreo efectivo resulta fundamental, especialmente en regiones con alta actividad agrícola como el estado de Sinaloa.

No obstante, los sistemas tradicionales de monitoreo enfrentan retos como la dispersión geográfica de los cuerpos de agua, la gran cantidad de parámetros que deben evaluarse y limitaciones económicas/técnicas para el análisis de datos en tiempo y forma. Asimismo, el desarrollo de tecnologías basadas en inteligencia artificial y aprendizaje automático ha permitido mejorar la eficiencia y precisión en la evaluación de la calidad del agua, permitiendo la optimización de redes de monitoreo y facilitando la toma de decisiones ambientales.

En este capítulo se ofrece una revisión crítica de los antecedentes y tendencias actuales en el monitoreo de calidad del agua, enfocándose en estudios relevantes y aplicaciones recientes en determinadas regiones.

3.1. Antecedentes

Con el propósito de establecer un conocimiento profundo y actualizado sobre el estado actual de la tecnología, así como de los métodos utilizados para la medición de la calidad del agua en los cuerpos hídricos del estado de Sinaloa, se llevó a cabo una investigación exhaustiva que sirvió como base teórica para el desarrollo de OSMIAA.

Uno de los trabajos más relevantes que sirvió de referencia para el desarrollo de OSMIAA fue elaborado por (Mendivil-García et al., 2024), quienes propusieron una metodología basada en inteligencia artificial para optimizar la red de monitoreo sobre la calidad del agua en la cuenca del río Culiacán, una región caracterizada por su intensa actividad agrícola. En dicho estudio, se utilizó una base de datos proporcionada por CONAGUA, con información recolectada entre 2013 y 2020, la cual comprendía de 23 parámetros físicos, químicos y biológicos distribuidos en 23 sitios de muestreo.

La metodología implementada combinó técnicas de reducción de dimensionalidad, como el Análisis de Componentes Principales (PCA), con algoritmos de Agrupación Jerárquica de Clústeres (HCA), utilizando



como criterios la distancia euclidiana y el método de Ward. Esta combinación permitió identificar los parámetros más influyentes en la calidad del agua, entre los cuales destacaron:

- Nutrientes (TN y TP)
- Compuestos orgánicos disueltos ($\text{NH}_3\text{-N}$ y TOC)
- Coliformes fecales

Igualmente, los resultados del estudio demostraron que fue posible agrupar sitios de muestreo con características similares, lo cual permitió reducir en un 50% la red de monitoreo sin comprometer la representación original de la información recolectada, como se puede observar en la Figura 3-1.

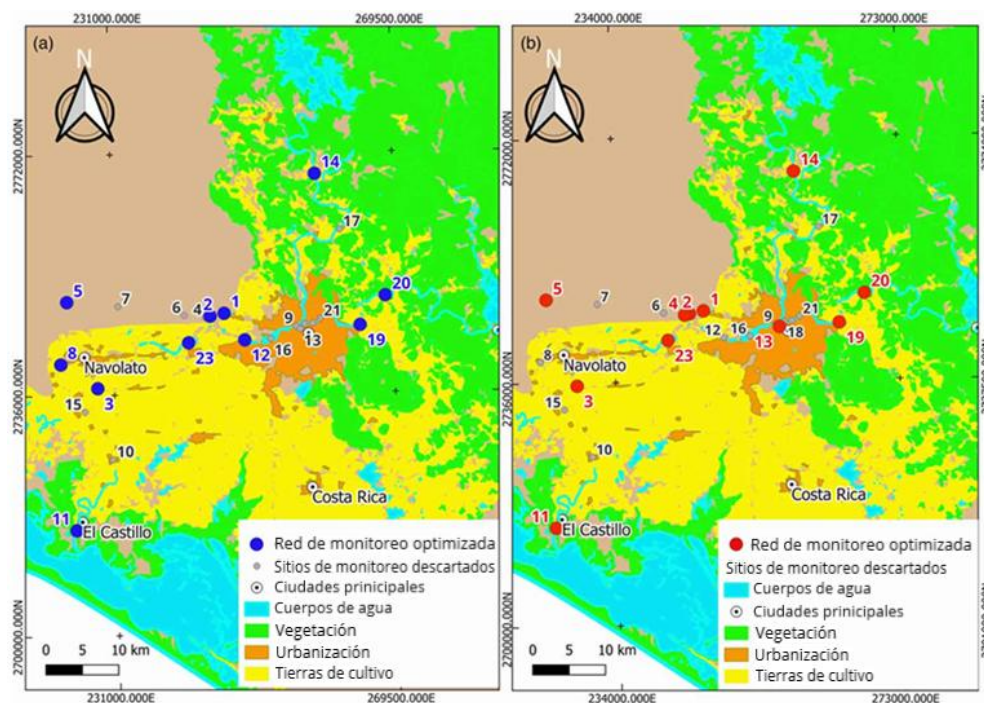


Figura 3-1. Red de monitoreo optimizada en la cuenca del Río Culiacán para temporada lluviosa y seca (Nota. tomado de *Mendivil-García et al., 2024*)

Por consiguiente, este enfoque demostró la utilidad del aprendizaje automático no supervisado para la gestión de recursos hídricos, particularmente en regiones con limitaciones económicas y grandes volúmenes de datos disponibles.

3.2. Tendencias actuales

3.2.1. Sensores IoT (Internet de las Cosas) para monitoreo ambiental en tiempo real

En un estudio realizado por (Agudelo Méndez & Lozano Bernal, 2025), se presentó el diseño e implementación de un sistema de monitoreo ambiental basado en sensores IoT para la vigilancia continua de parámetros relacionados con la calidad del agua. El sistema incorporó nodos sensores conectados mediante comunicación inalámbrica a una plataforma web, la cual es capaz de registrar y mostrar en tiempo real datos como:

- pH
- Turbidez
- Temperatura
- Conductividad

Por otro lado, las pruebas de validación en campo mostraron una alta correlación entre las mediciones de los sensores y las obtenidas mediante métodos de laboratorio, con desviaciones mínimas. Esto confirmó la capacidad del sistema para detectar variaciones rápidas en la calidad del agua, incluso en intervalos de minutos.

En la Figura 3-2 se muestra la comparación de resultados entre el sistema IoT (Internet de las Cosas) y los análisis de laboratorio, evidenciando su precisión y fiabilidad para aplicaciones de monitoreo ambiental.



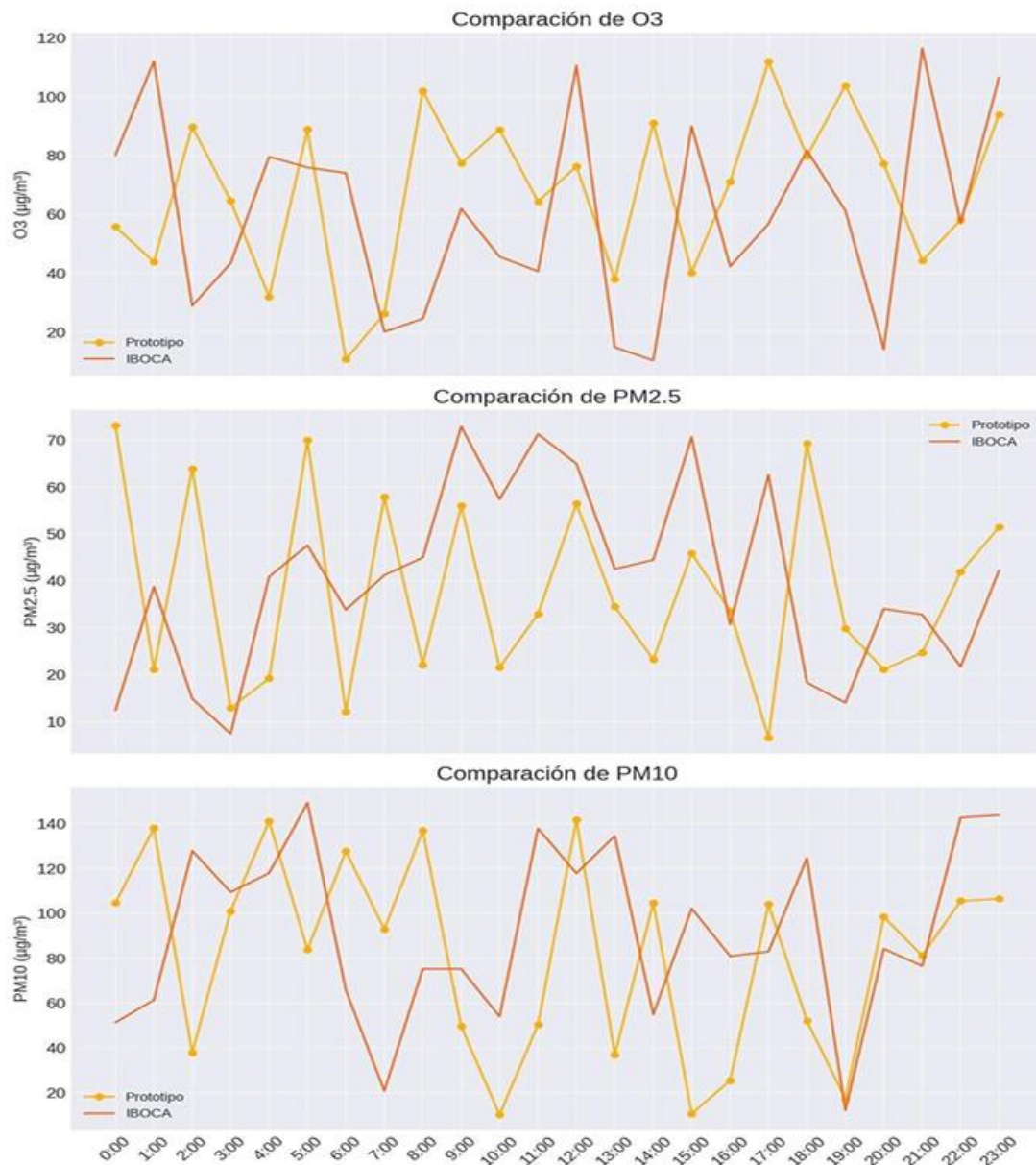


Figura 3-2. Comparación entre datos obtenidos por sensores IoT y análisis de laboratorio (Nota. tomado de Agudelo Méndez & Lozano Bernal, 2025, pág. 137)

3.2.2. Aplicación de inteligencia artificial y aprendizaje automático

La investigación realizada por (Liu et al., 2021) analizó el uso conjunto de técnicas de reducción de dimensionalidad y algoritmos de agrupamiento (HCA) para la caracterización de zonas con distintas propiedades hidroquímicas.

En primera instancia, en el conjunto de datos "Oslo transect" (transecto de Oslo) se hizo uso de la técnica "Incrustación de Vecinos Estocástica distribuida en t" (t-SNE), logrando separar de manera nítida los nueve



grupos de muestras correspondientes a diferentes materiales vegetales, superando visualmente la representación generada por PCA.

Además, en el conjunto de datos "Taiyuan karst water" (agua kárstica Taiyuana) la visualización con t-SNE coincidió con las divisiones hidrogeológicas conocidas, habiendo mínimas excepciones ocasionadas por anomalías químicas. Finalmente, se trabajó con el conjunto de datos "Jiangnan Plain groundwater" (agua subterránea de la Llanura Jiangnan), en donde t-SNE redujo los solapamientos presentes en PCA, facilitando la identificación de subzonas con características propias.

En las siguientes ilustraciones se observa el contraste entre las visualizaciones obtenidas con PCA y t-SNE para los tres conjuntos de datos analizados por el autor (Liu et al., 2021) en su trabajo: transecto de Oslo (Figura 3-3), agua kárstica Taiyuana (Figura 3-4) y agua subterránea de la Llanura Jiangnan (Figura 3-5). En todos estos escenarios, t-SNE ofreció una separación más clara y consistente entre los grupos en comparación con PCA, lo que facilitó la interpretación de patrones espaciales en datos ambientales.

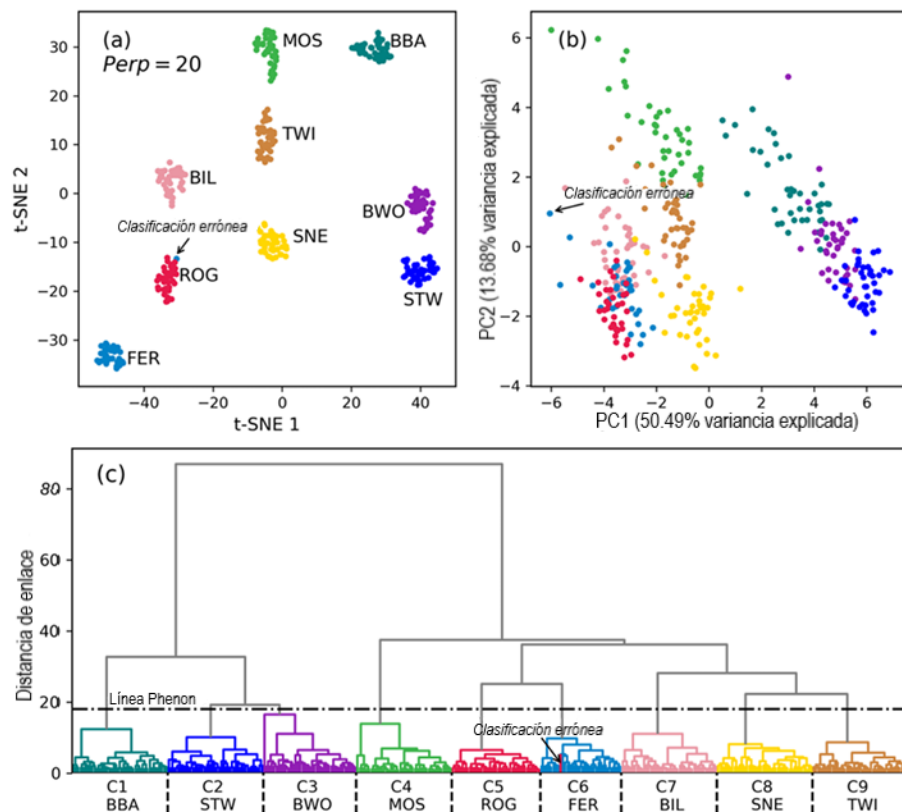


Figura 3-3. Comparación entre PCA y t-SNE en el conjunto de datos "transecto de Oslo" (Nota. tomado de Liu et al., 2021, pág. 6)

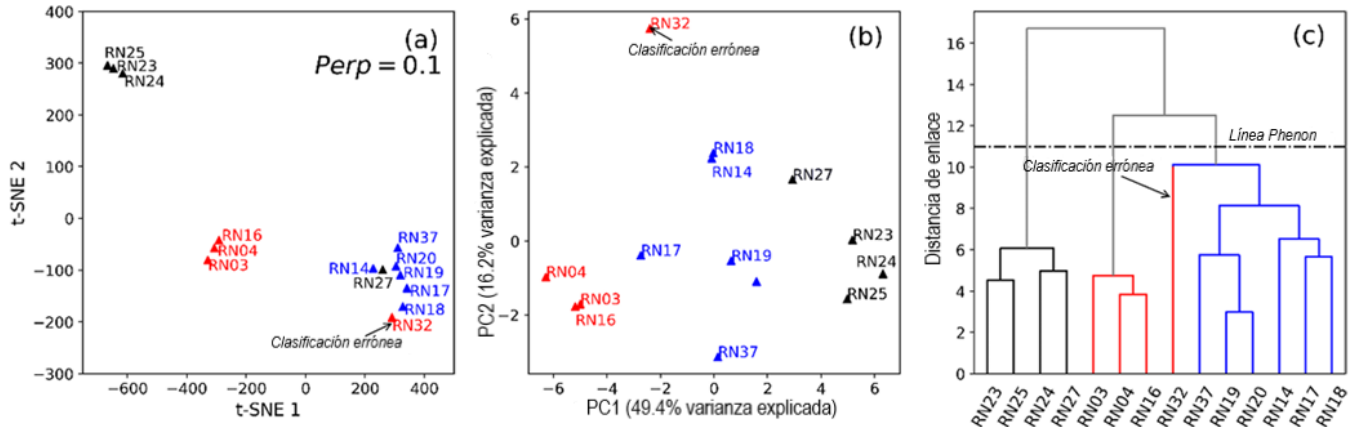


Figura 3-4. Comparación entre PCA y t-SNE en el conjunto de datos “agua kárstica Taiyuana” (Nota. tomado de Liu et al., 2021, pág. 8)

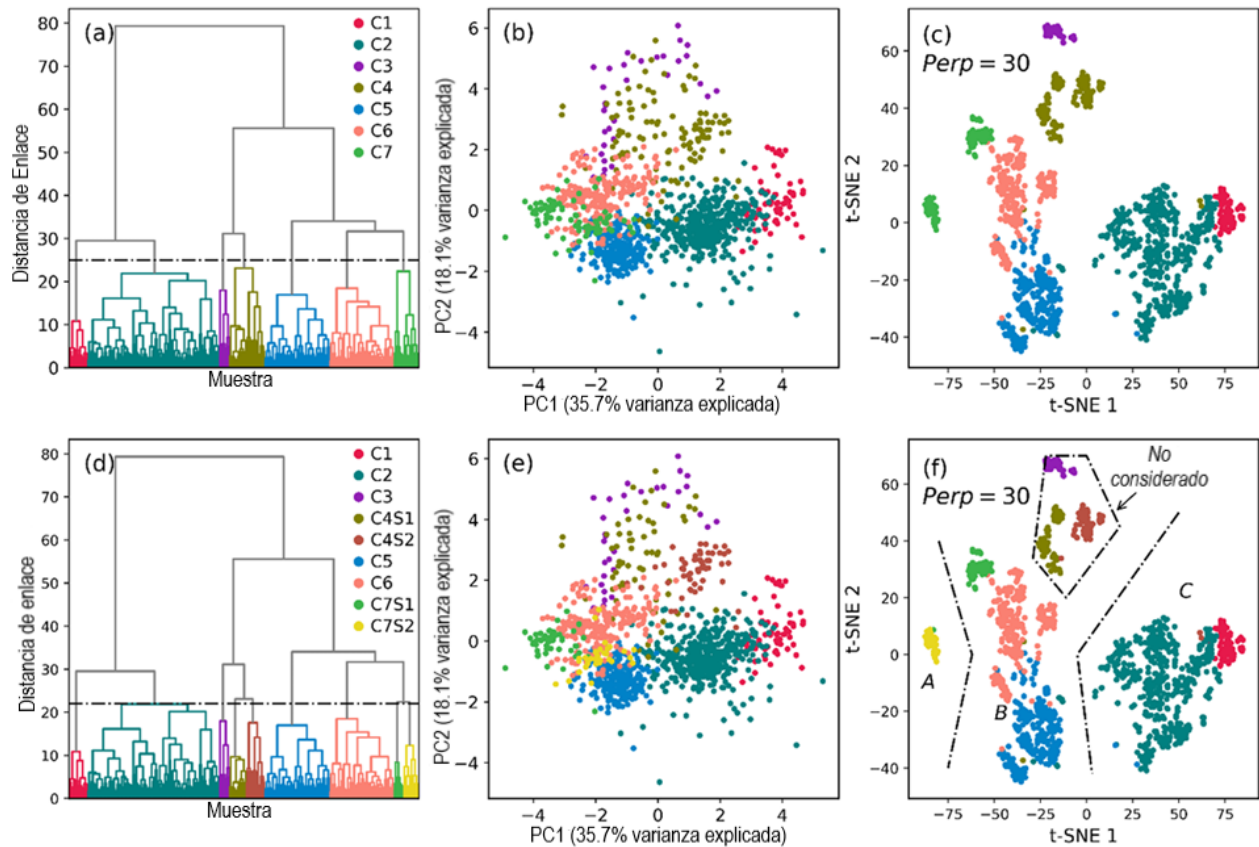


Figura 3-5. Comparación entre PCA y t-SNE en el conjunto de datos “agua subterránea de la Llanura Jiangnan” (Nota. tomado de Liu et al., 2021, pág. 9)

3.2.3. Integración de GIS (Sistemas de Información Geográfica) y modelos hidrológicos

El trabajo de investigación realizado por (Tsegaye et al., 2024) demostró el valor de integrar Sistemas de Información Geográfica (GIS) con modelos hidrológicos e hidráulicos para evaluar los siguientes elementos:



- Impacto combinado del aumento del nivel del mar (SLR)
- Cambios en el uso del suelo
- Fluctuaciones acuíferas
- Intensidad de tormentas sobre la capacidad de almacenamiento e inundación en zonas urbanas

Como se puede observar en la Figura 3-6, las simulaciones para la comunidad costera de Bonita Bay, en Florida, revelaron que el nivel del mar proyectado para 2100 podría elevar el nivel acuífero hasta 0.91 m respecto a las condiciones actuales, provocando que la mayoría de los estanques superen su nivel superficial durante eventos de tormenta.

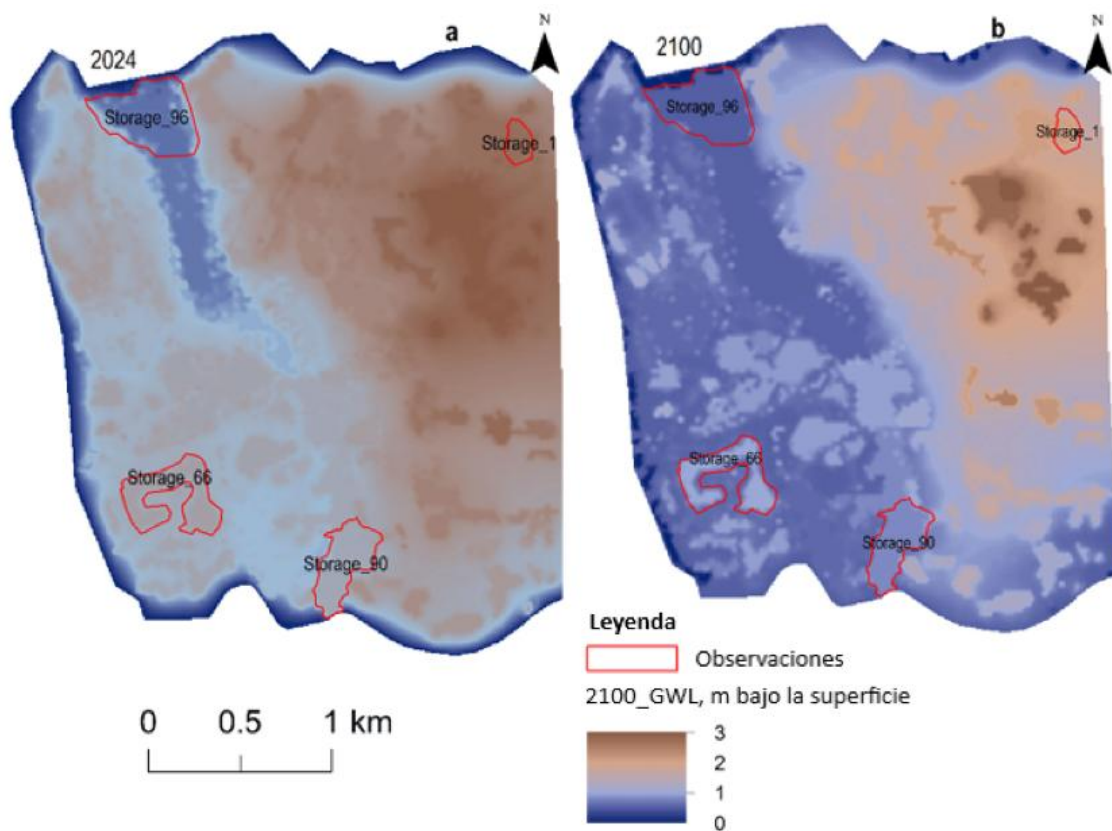


Figura 3-6. Nivel medio de aguas subterráneas para el escenario actual y futuro (Nota. tomado de Tsegaye et al., 2024, pág. 10)

Por otro lado, la Figura 3-7 presenta un gráfico sobre los humedales cercanos a la costa, en donde hubo un incremento entre 0.61 m y 0.76 m, alterando su papel y convirtiéndolos en almacenamiento superficial, siendo esta el agua retenida en el suelo tras haber pasado una tormenta.

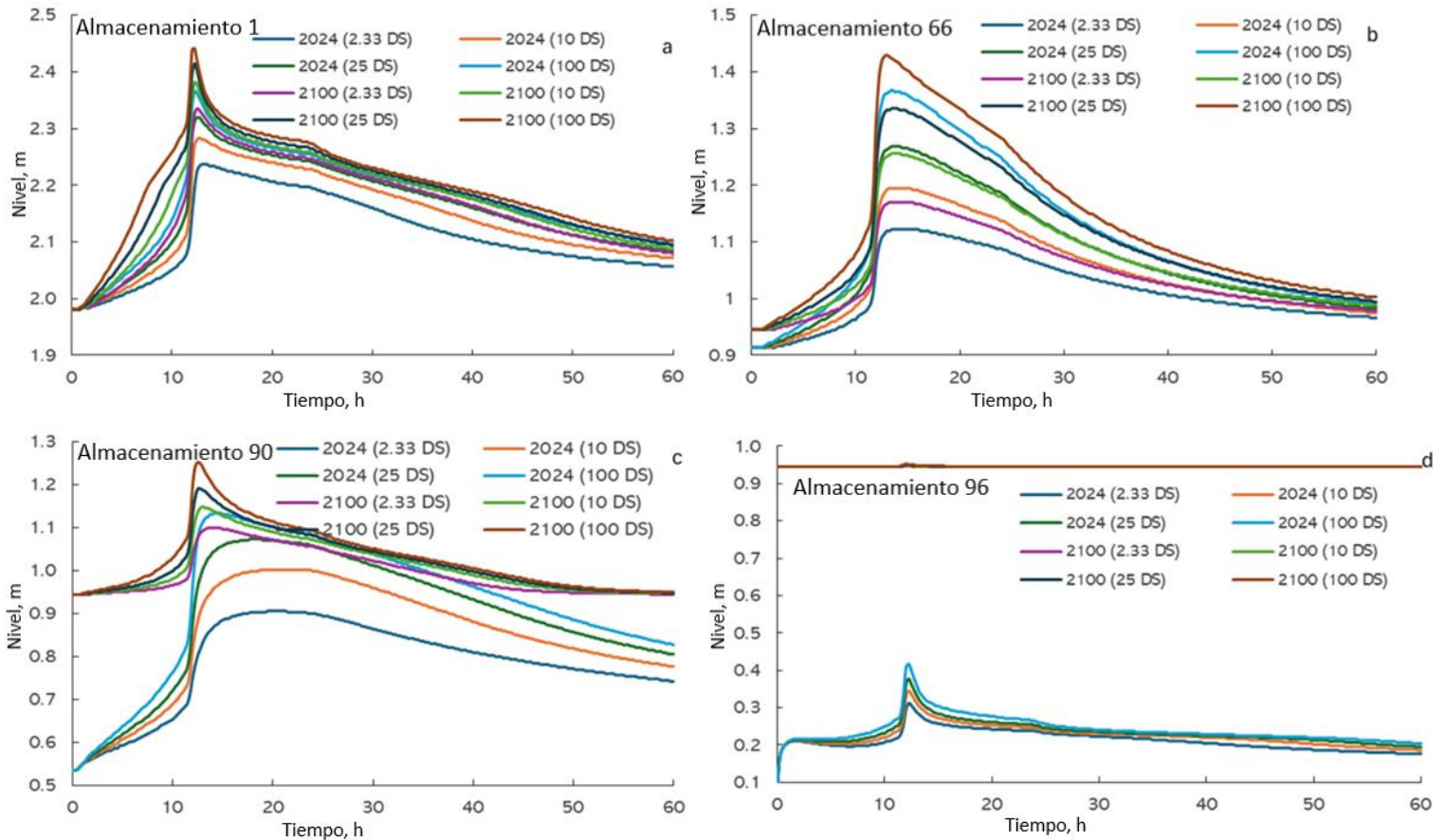


Figura 3-7. Variación del nivel de agua en estanques (a, b) y humedales (c, d) en escenarios de 2024 y 2100 (Nota. tomado de Tsegaye et al., 2024, pág. 10)

Por último, la extensión de las zonas inundadas aumentó del 12.3% en tormentas promedio al 18.4% en tormentas extremas. Al llegar al 2100, el incremento adicional llegaría a un 5%, alcanzando hasta el 22.5% del área. En la Figura 3-8 se presenta la comparación espacial de las zonas afectadas bajo los escenarios actuales y futuros, mostrando el agravamiento en aumento de las inundaciones.

Tormenta de diseño anual promedio



Tormenta de diseño en 24 horas, 10 años



Tormenta de diseño en 24 horas, 25 años



Tormenta de diseño en 24 horas, 100 años

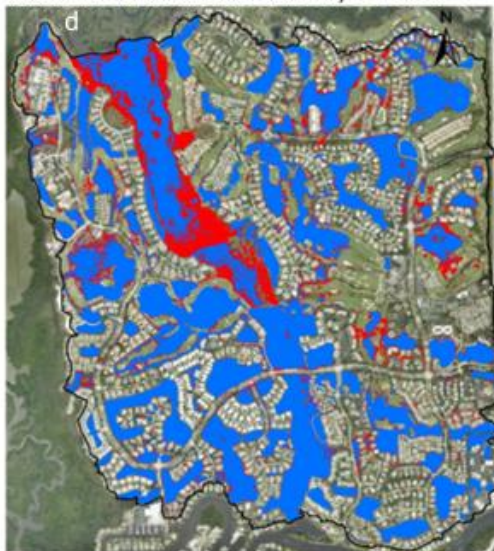


Figura 3-8. Mapas de inundaciones en la actualidad (2024) y futuro (2100) considerando el uso existente del suelo (Nota. tomado de Tsegaye et al., 2024, pág. 12)

Capítulo 4

4. Desarrollo del proyecto

En este capítulo se describe a detalle todo el proceso seguido para desarrollar el proyecto OSMIAA, desde la preparación inicial de los datos hasta la implementación de un sistema interactivo que integra el procesamiento, análisis, y visualización de los resultados del clustering (agrupación de datos).

El desarrollo de la aplicación web se realizó completamente en un único script (código) de Python, el cual concentra la lógica de tratamiento de datos, ejecución de algoritmos, reducción de dimensionalidad y despliegue visual en un entorno web interactivo.

4.1. Tratamiento de datos

El punto de partida fue la obtención de los datos proporcionados por la Comisión Nacional del Agua (CONAGUA) a través de su red de monitoreo RENAMECA, en el que se incluyen diversos parámetros físicos, químicos y biológicos de diferentes sitios de muestreo establecidos por toda la república mexicana.

Este conjunto de datos requería un proceso de depuración previo a su uso, el cual incluyó lo siguiente:

- Filtrado geográfico para delimitar la base de datos a sitios que se encontrarán en la región de estudio, el cual vendría siendo el estado de Sinaloa.
- Filtrado por tipo de cuerpo de agua con el que se va a trabajar, en donde solo los de carácter superficial terminarían siendo excluidos,
- Eliminación de símbolos no numéricos para evitar interferencias en el procesamiento automático y evitar problemas en la generación de clústeres.
- Eliminación de columnas que contengan valores redundantes o en donde más del 35% solo fueran datos vacíos.
- Eliminación de registros incompletos (filas) para evitar la generación de sesgos en análisis posteriores.
- Generación de un archivo CSV conteniendo todo el proceso aplicado, siendo un formato más adecuado para el análisis automatizado.

Todo este proceso puede ser apreciado en síntesis en la Figura 4-1.



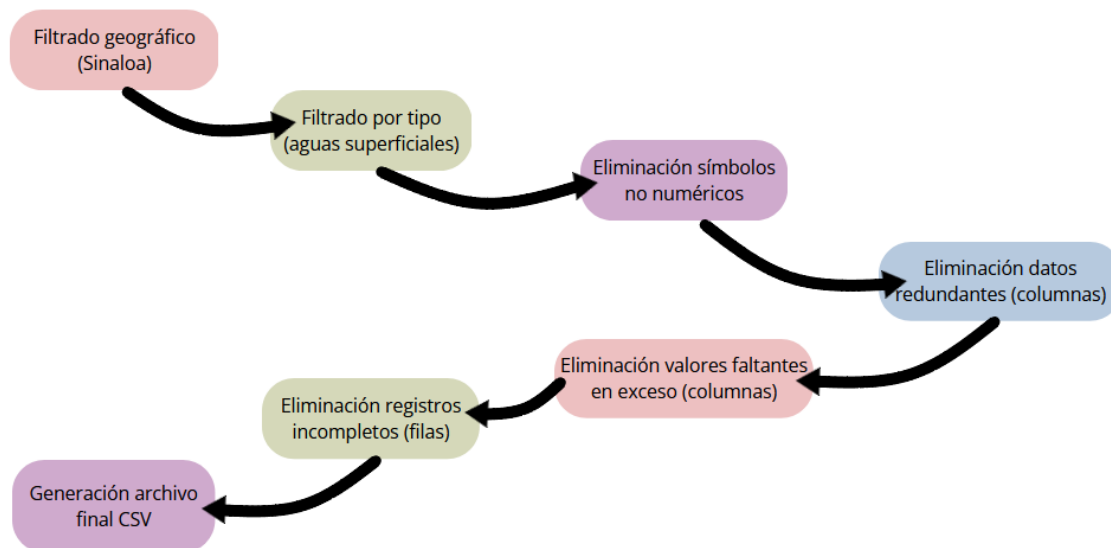


Figura 4-1. Diagrama del proceso de tratamiento de datos aplicado al conjunto de datos de RENAMECA (Nota. elaboración propia)

4.2. Implementación de algoritmos de agrupamiento y reducción de dimensionalidad

Antes de la aplicación de los algoritmos, se realizó una normalización de datos mediante el método **StandardScaler** para que todas las variables tuvieran la misma escala y evitar problemas con valores de gran tamaño decimal.

Posteriormente, se aplicó un Análisis de Componentes Principales (PCA) para reducir la dimensionalidad a siete componentes principales, lo que optimizó el rendimiento y llevó a una mejor interpretación de los resultados.

El siguiente paso consistió en aplicar diferentes algoritmos de aprendizaje no supervisado, todos implementados usando la librería Scikit-learn:

- Propagación por Afinidad (Affinity Propagation)
- BIRCH (Reducción y Agrupación Balanceada Iterativa usando Jerarquías)
- DBSCAN (Agrupación Espacial Basado en Densidad de Aplicaciones con Ruido)
- K-Means
- Mini-Batch K-Means

Para una mejor aclaración, todo el proceso seguido en esta etapa puede ser visto en la Figura 4-2.

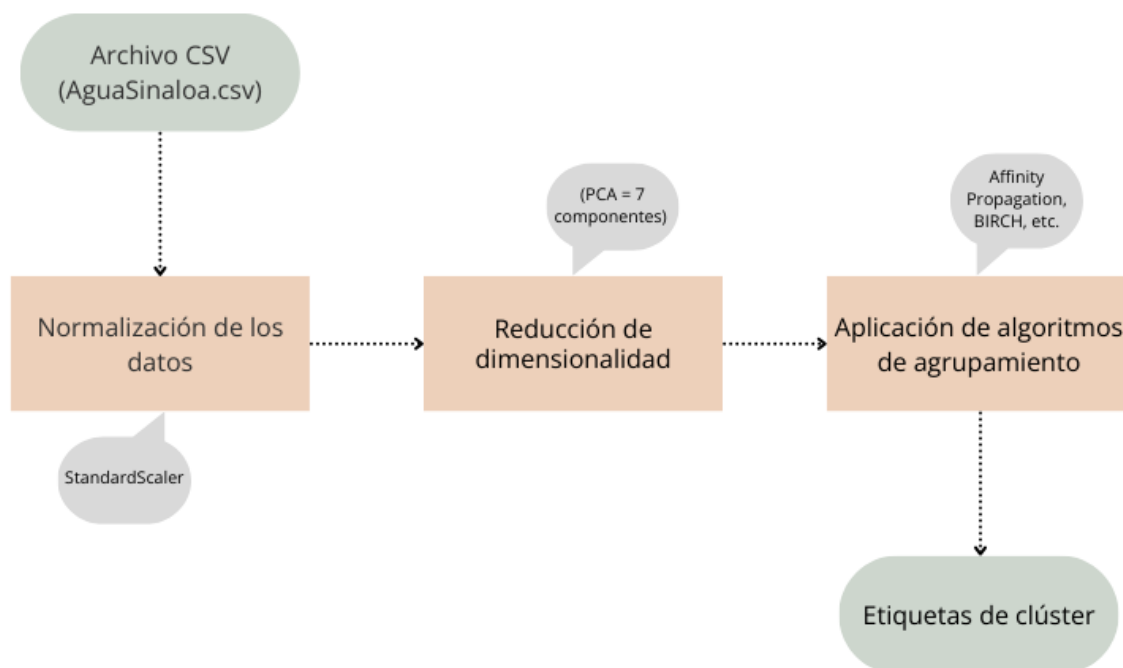


Figura 4-2. Diagrama del proceso de implementación de algoritmos de agrupamiento y reducción de dimensionalidad (*Nota. elaboración propia*)

4.3. Desarrollo de aplicación web interactiva y visualización geográfica del agrupamiento

Toda la lógica se desarrolló en un único script de Python (Anexo a) utilizando la librería **Dash** como núcleo para interfaz interactiva. Desde una sola ejecución, este diseño permitió que el usuario pudiera acceder a las siguientes funcionalidades (Figura 4-3):

- **Selección de parámetros:** se puede escoger qué variables (del conjunto de datos) incluir en el análisis.
- **Selección de algoritmo de agrupamiento:** se puede elegir entre los cinco algoritmos que se acotaron.
- **Configuración de hiperparámetros:** se puede definir manualmente los valores de los hiperparámetros que conforman a los algoritmos de agrupamiento, teniendo validaciones dinámicas que previenen errores en la aplicación.
- **Filtrado interactivo por clúster:** los clústeres detectados pueden ser activados o desactivados en el mapa según las preferencias del usuario.
- **Generación del agrupamiento:** se visualiza el resultado obtenido directamente sobre un mapa interactivo del territorio sinaloense.
- **Exportación a CSV:** se pueden descargar los datos del agrupamiento generado en formato CSV, facilitando su análisis posterior.

- **Reiniciar aplicación:** se incluye un botón para reiniciar la aplicación si ocurre algún error inesperado o si se desea comenzar desde cero.

Clustering - Sitios de muestreo Sinaloa

Selección de parámetros

☒ COLI_FEC ☒ COT ☒ COT_SOL ☒ N_NH3 ☒ N_NO2 ☒ N_NO3 ☒ N_ORG ☒ N_TOT ☒ N_TOTK ☒ P_TOT ☒ ORTO_PO4 ☒ COLOR_VER
☒ ABS_UV ☒ SDT ☒ SST ☒ TURBIEDAD ☒ AS_TOT ☒ CD_TOT ☒ CR_TOT ☒ HG_TOT ☒ NI_TOT ☒ PB_TOT ☒ TEMP_AMB

Selecciona un algoritmo

Reiniciar

Generar clusters Descargar CSV

Filtrar clusters generados

Figura 4-3. Interfaz gráfica de la aplicación web con todas las funciones implementadas (*Nota. elaboración propia*)

La representación visual de las agrupaciones de datos obtenidas se realizó utilizando librería **Plotly Express** mediante el componente **scatter_mapbox**, el cual permitió:

- Asignar un color distinto a cada clúster detectado.
- Mantener interactividad permitiendo ampliar la imagen, desplazar el mapa y filtrar clústeres visibles.
- Mostrar etiquetas emergentes con información detallada de cada sitio.

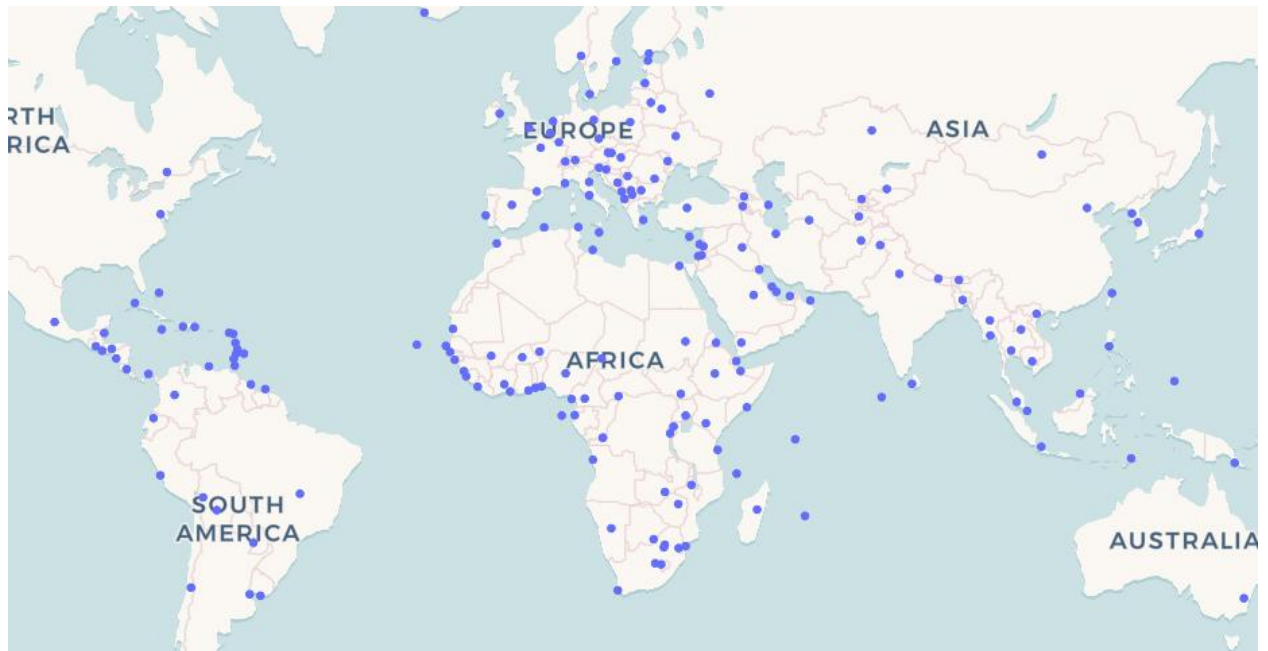


Figura 4-4. Ejemplo de implementación (Nota. tomado de Plotly Development Team, s.f.)

Capítulo 5

5. Pruebas y Resultados

En este capítulo se presentan las pruebas realizadas y los resultados obtenidos en cada etapa del desarrollo de OSMIAA, siguiendo el flujo metodológico descrito en el capítulo anterior.

5.1. Resultados del tratamiento de datos

El proceso de filtrado y depuración redujo el volumen inicial de información, pasando de cientos de columnas a una matriz limpia con 25 parámetros clave:

Columnas de identificación

- CLAVE SITIO
- NOMBRE DEL SITIO

Parámetros numéricos

- **COLI_FEC:** coliformes fecales
- **COT:** carbono orgánico total
- **COT_SOL:** carbono orgánico soluble
- **N_NH3:** nitrógeno amoniacal
- **N_NO2:** nitrógeno de nitritos
- **N_NO3:** nitrógeno de nitratos
- **N_ORG:** nitrógeno orgánico
- **N_TOT:** nitrógeno total (cálculo)
- **N_TOTK:** nitrógeno Kjeldahl
- **P_TOT:** fósforo total
- **ORTO_PO4:** fósforo reactivo total (o-fosfatos)
- **COLOR_VER:** color verdadero
- **ABS_UV:** absorción UV
- **SDT:** sólidos disueltos totales (cálculo)



- **SST:** sólidos suspendidos totales
- **TURBIEDAD:** turbiedad
- **AS_TOT:** arsénico total
- **CD_TOT:** cadmio total
- **CR_TOT:** cromo total
- **HG_TOT:** mercurio total
- **NI_TOT:** níquel total
- **PB_TOT:** plomo total
- **TEMP_AMB:** temperatura ambiente

Todo este proceso mejoró la consistencia y utilidad de los datos para el posterior agrupamiento, eliminando ruido y valores atípicos que podían afectar el rendimiento de los algoritmos. En las siguientes figuras se presentan gráficas que ofrecen una serie de comparaciones entre las características que originalmente poseía el conjunto de datos proporcionado y las que cuenta actualmente tras haber superado esta etapa.

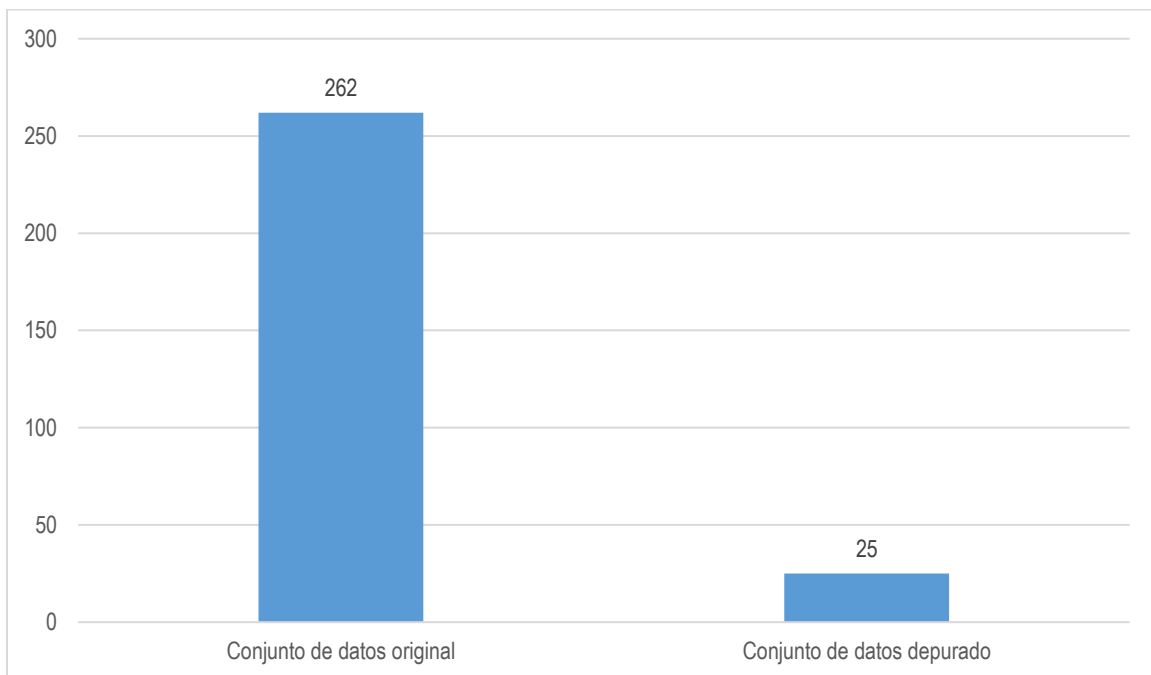


Figura 5-1. Selección de parámetros: gráfico de barras Comparación columnas (Nota. elaboración propia)

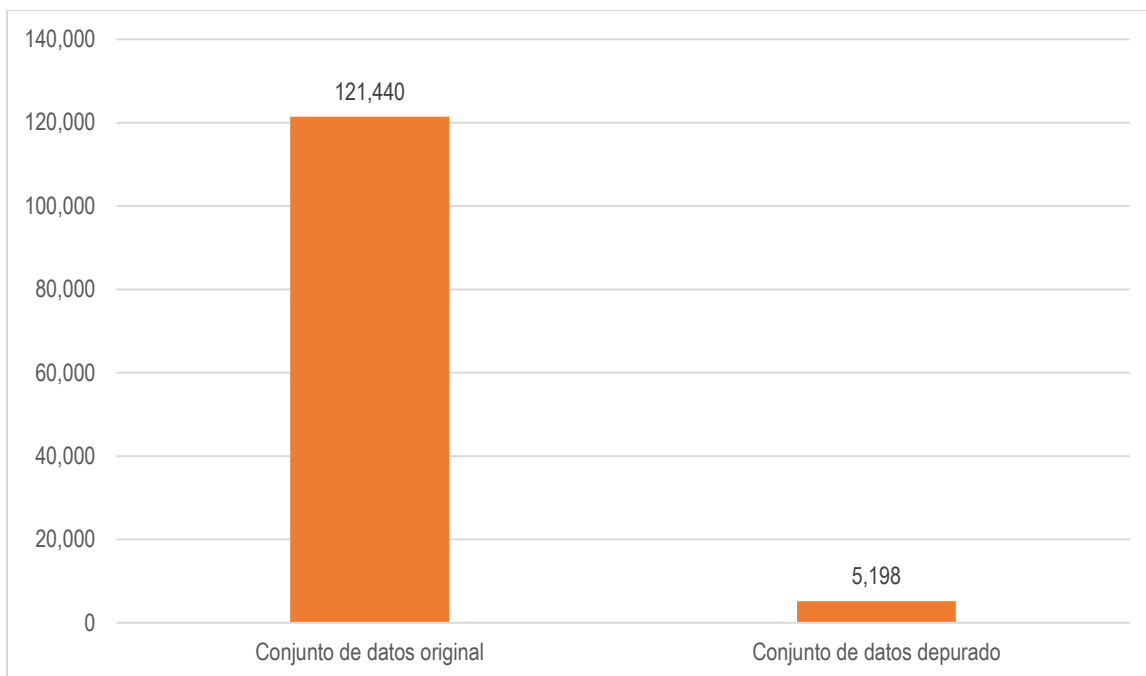


Figura 5-2. Extracción de datos relacionados con Sinaloa: gráfico de barras Comparación filas (Nota. elaboración propia)

5.2. Resultados de la aplicación de algoritmos de agrupamiento y reducción de dimensionalidad

Cada algoritmo produjo agrupaciones con características diferentes:

- **Propagación por Afinidad (Affinity Propagation):** Identificó automáticamente el número de clústeres, aunque fue el más difícil de manipular a conveniencia del usuario. (Figura 5-3)

Affinity Propagation

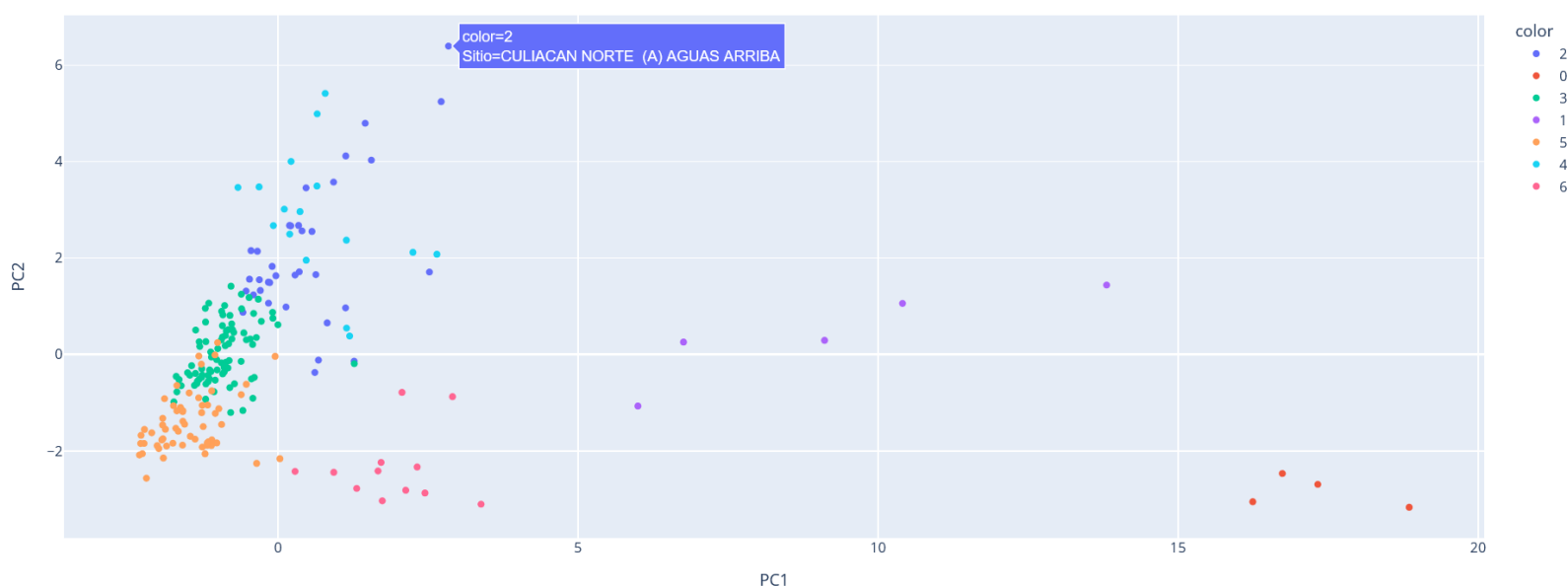


Figura 5-3. Agrupamiento generado: Propagación por Afinidad (Nota. elaboración propia)



- **BIRCH:** Alta eficiencia, pero sensible a la configuración de su hiperparámetro umbral. (Figura 5-4)

BIRCH

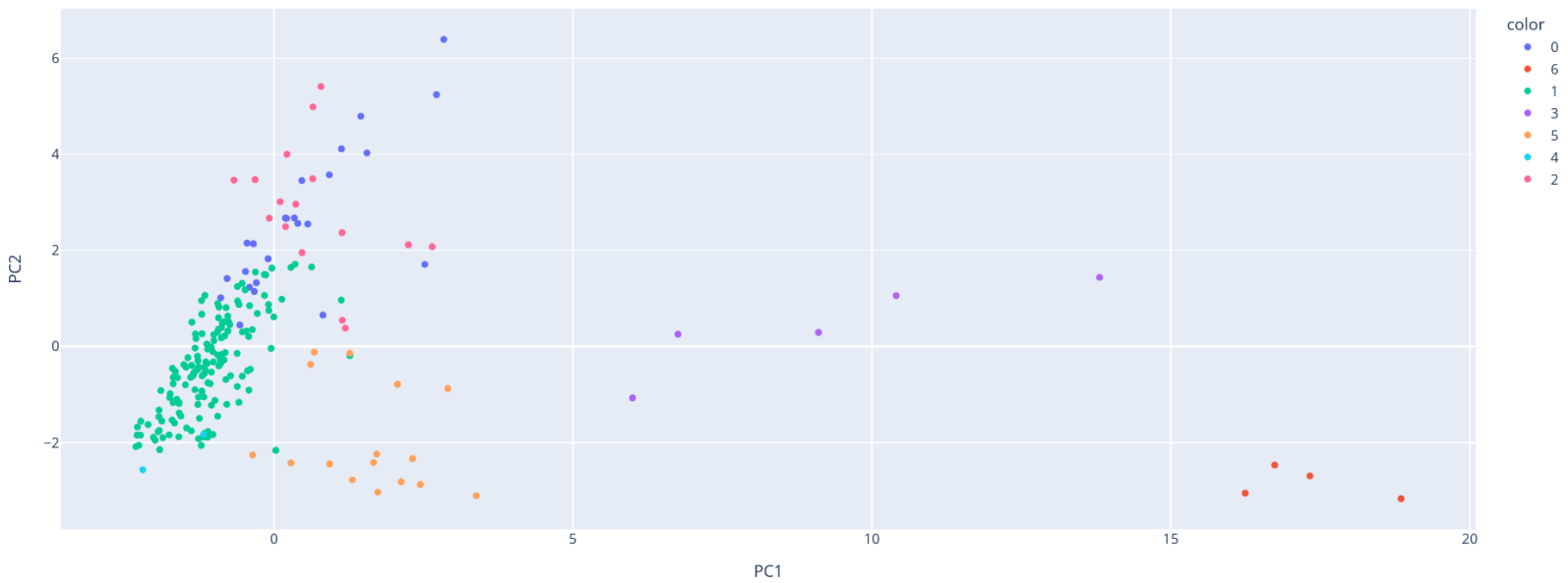


Figura 5-4. Agrupamiento generado con BIRCH (Nota. elaboración propia)

- **DBSCAN:** Detectó clústeres densos, aisló ruido y fue útil para datos con patrones irregulares; a pesar de haber entregado una distribución de clústeres no tan favorable. (Figura 5-5)

DBSCAN

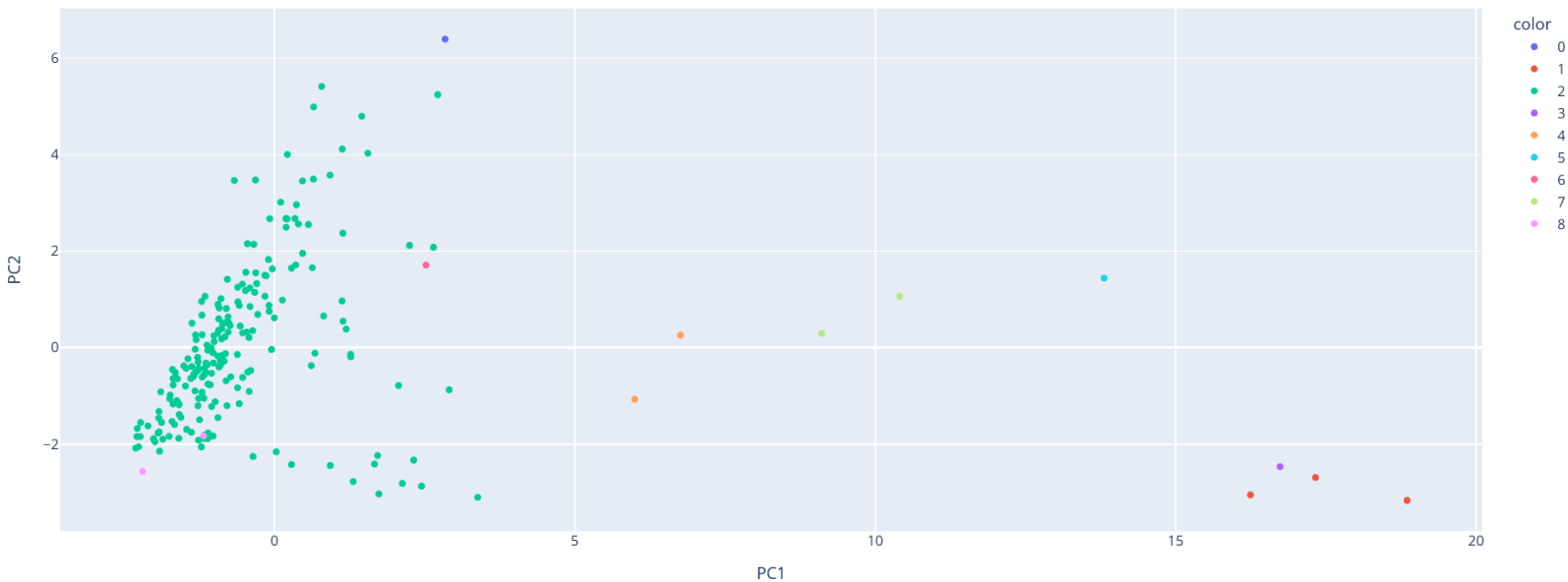


Figura 5-5. Agrupamiento generado: DBSCAN (Nota. elaboración propia)



- **K-Means y Mini-Batch K-Means:** Resultados rápidos y consistentes, adecuados para conjuntos de datos más homogéneos. Aunque se debe de saber la cantidad de clústeres a generar de antemano, cada uno entregó resultados diferentes y exitosos a su manera.

K-Means

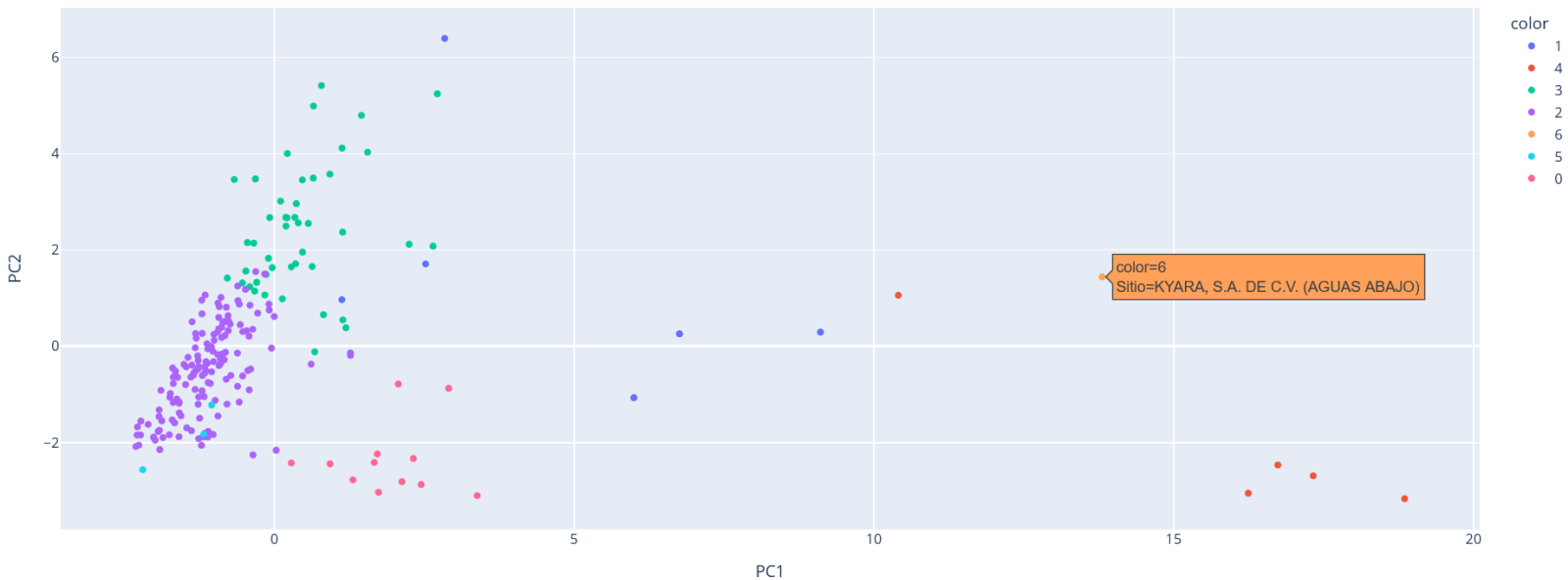


Figura 5-6. Agrupamiento generado: K-Means (Nota. elaboración propia)

Mini-Batch K-Means

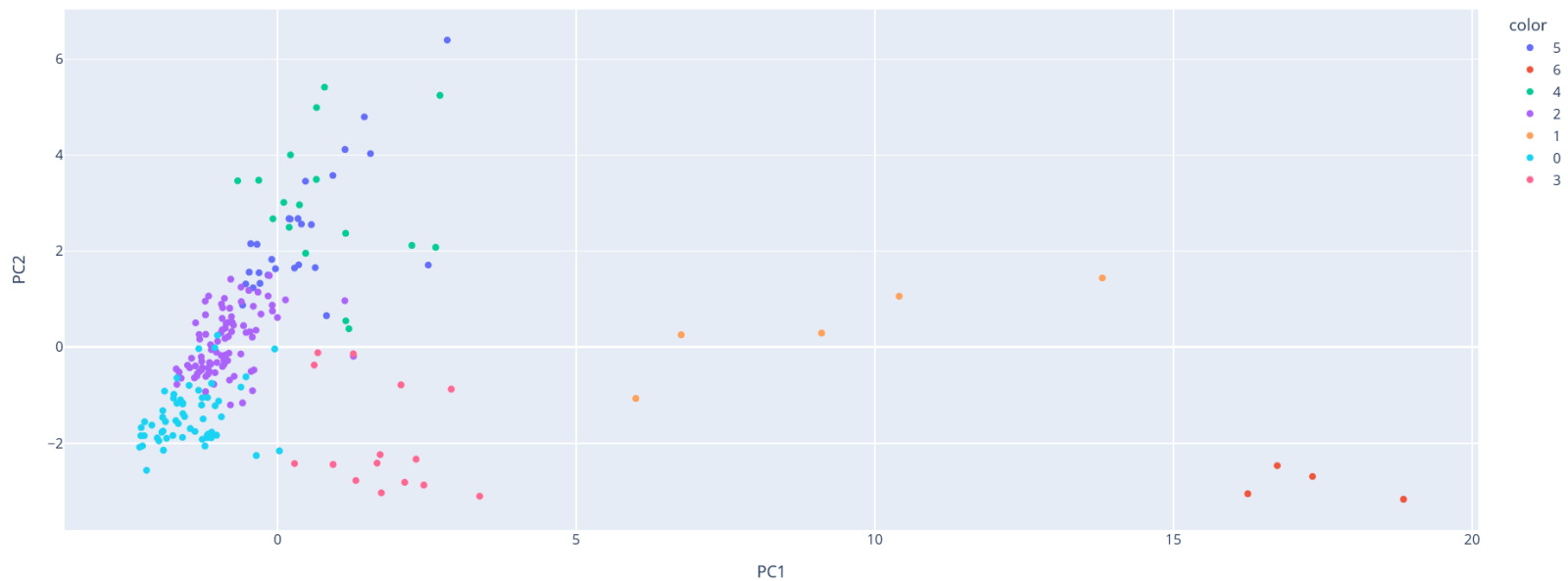


Figura 5-7. Agrupamiento generado: Mini-Batch K-Means (Nota. elaboración propia)



5.3. Resultados de la aplicación web interactiva y visualización geográfica del agrupamiento

La interfaz de Dash demostró ser efectiva para que el usuario final pudiera explorar los mapas de agrupamiento generados sin tener conocimientos técnicos avanzados, basta con saber el rango de valores y funcionamiento que ejercen los hiperparámetros en los algoritmos para poder interactuar con la aplicación sin ningún problema.

Por otro lado, los mapas interactivos permitieron identificar patrones espaciales significativos. En particular, se observó que ciertos clústeres se concentraban en zonas específicas, lo que puede servir como ayuda para estrategias de monitoreo y toma de decisiones a futuro.

Para ilustrar el funcionamiento de la aplicación web desarrollada, en las siguientes figuras se presentan ejemplos de resultados obtenidos al seleccionar los distintos algoritmos de agrupamiento que la herramienta ofrece. Estas visualizaciones permiten observar cómo los sitios de muestreo se organizan en diversos clústeres sobre el mapa geográfico de Sinaloa, lo que facilita al usuario comparar las salidas de cada método y analizar la distribución espacial de la calidad del agua bajo diferentes escenarios de prueba.



Clustering - Sitios de muestreo Sinaloa

Selección de parámetros

☒ COLI_FEC

☒ COT

☒ COT_SOL

☒ N_NH3

☒ N_NO2

☒ N_NO3

☒ N_ORG

☒ N_TOT

☒ N_TOTK

☒ P_TOT

☒ ORTO_PO4

☒ COLOR_VER

☒ ABS_UV

☒ SDT

☒ SST

☒ TURBIEDAD

☒ AS_TOT

☒ CD_TOT

☒ CR_TOT

☒ HG_TOT

☒ NI_TOT

☒ PB_TOT

☒ TEMP_AMB

Affinity Propagation

damping0.7

preference-8.4

Reiniciar

Generar clusters

Descargar CSV

Filtrar clusters generados

☒ 1

☒ 2

☒ 3

☒ 4

☒ 5

☒ 6

☒ 7

☒ 8

☒ 9

☒ 10

☒ 11

☒ 12

☒ 13

☒ 14

☒ 15

☒ 16

☒ 17

☒ 18

☒ 19

☒ 20

☒ 21

☒ 22

☒ 23

☒ 24

☒ 25

☒ 26

☒ 27

☒ 28

☒ 29

☒ 30

Affinity Propagation

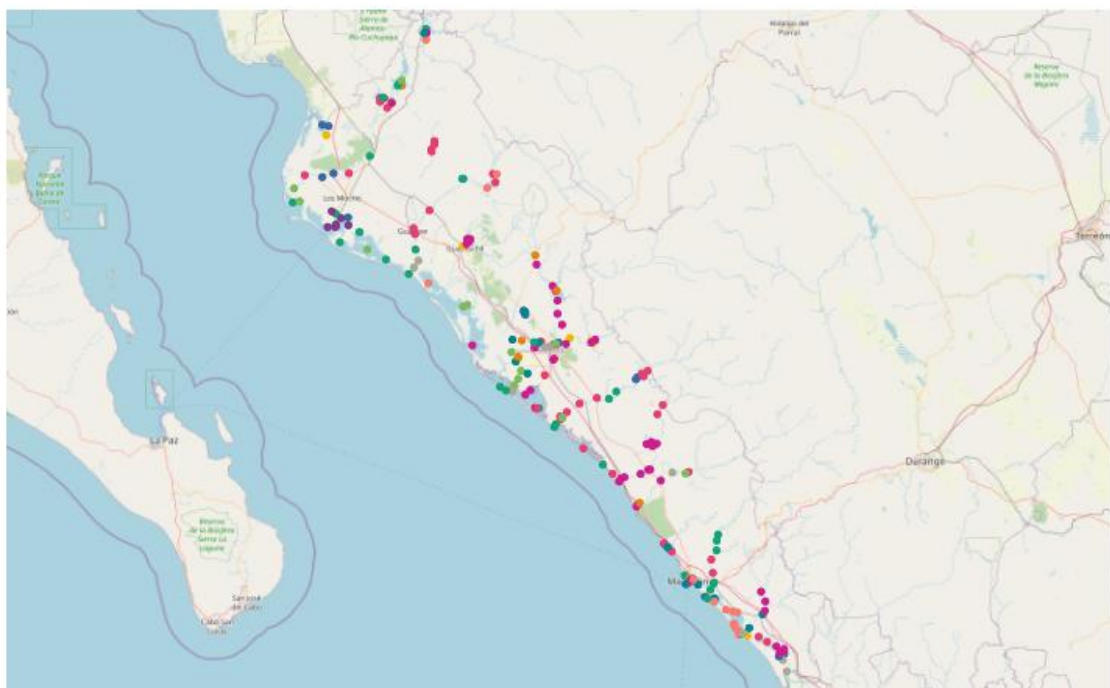


Figura 5-8. Visualización geográfica de clústeres generados: Propagación por Afinidad (*Nota. elaboración propia*)

Clustering - Sitios de muestreo Sinaloa

Selección de parámetros

☒ COLIFEC

☒ COT

☒ COT_SOL

☒ N_NH3

☒ N_NO2

☒ N_NO3

☒ N_ORG

☒ N_TOT

☒ N_TOTK

☒ P_TOT

☒ ORTO_PO4

☒ COLOR_VER

☒ ABS_UV

☒ SDT

☒ SST

☒ TURBIEDAD

☒ AS_TOT

☒ CD_TOT

☒ CR_TOT

☒ HG_TOT

☒ NI_TOT

☒ PB_TOT

☒ TEMP_AMB

BIRCH

threshold

0.7

n clusters

30

Reiniciar

Generar clusters

Descargar CSV

Filtrar clusters generados

☒ 1

☒ 2

☒ 3

☒ 4

☒ 5

☒ 6

☒ 7

☒ 8

☒ 9

☒ 10

☒ 11

☒ 12

☒ 13

☒ 14

☒ 15

☒ 16

☒ 17

☒ 18

☒ 19

☒ 20

☒ 21

☒ 22

☒ 23

☒ 24

☒ 25

☒ 26

☒ 27

☒ 28

☒ 29

☒ 30

BIRCH

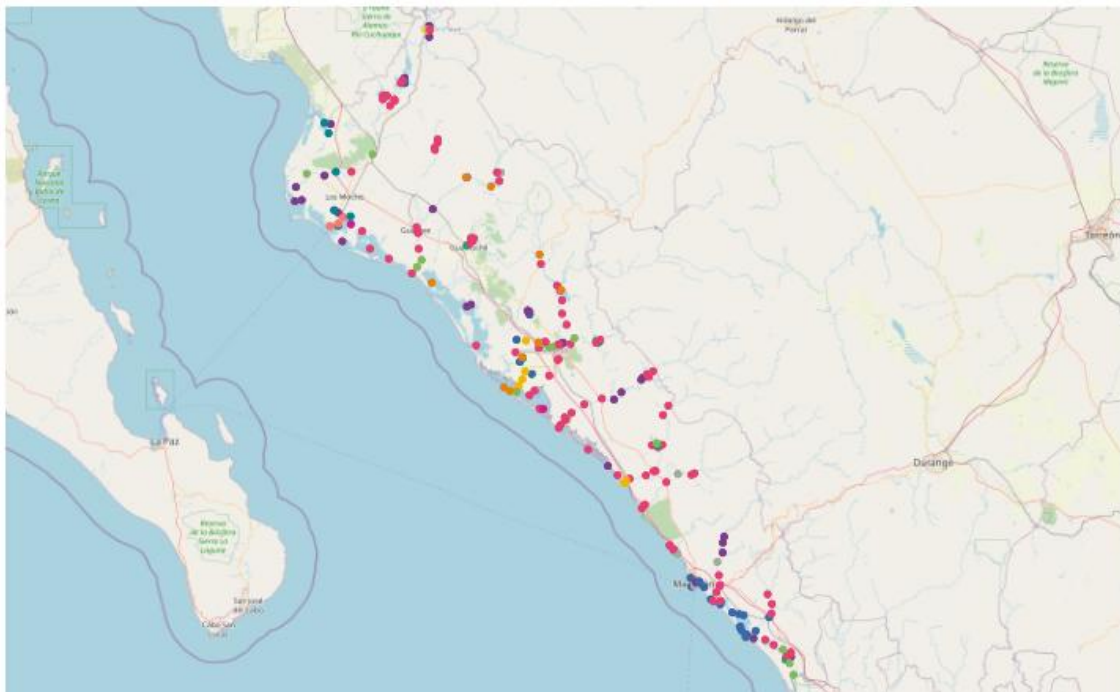


Figura 5-9. Visualización geográfica de clusters generados: BIRCH (*Nota. elaboración propia*)

Clustering - Sitios de muestreo Sinaloa

Selección de parámetros

☒ COLI_FEC
 ☒ COT
 ☒ COT_SOL
 ☒ N_NH3
 ☒ N_NO2
 ☒ N_NO3
 ☒ N_ORG
 ☒ N_TOT
 ☒ N_TOTK
 ☒ P_TOT
 ☒ ORTO_PO4
 ☒ COLOR_VER
 ☒ ABS_UV
 ☒ SDT
 ☒ SST
 ☒ TURBIEDAD
 ☒ AS_TOT
 ☒ CD_TOT
 ☒ CR_TOT
 ☒ HG_TOT
 ☒ NI_TOT
 ☒ PB_TOT
 ☒ TEMP_AMB

DBSCAN

epsilon 1.82
 min samples 1

Reiniciar

Generar clusters
 Descargar CSV

Filtrar clusters generados

☒ 1
 ☒ 2
 ☒ 3
 ☒ 4
 ☒ 5
 ☒ 6
 ☒ 7
 ☒ 8
 ☒ 9
 ☒ 10
 ☒ 11
 ☒ 12
 ☒ 13
 ☒ 14
 ☒ 15
 ☒ 16
 ☒ 17
 ☒ 18
 ☒ 19
 ☒ 20
 ☒ 21
 ☒ 22
 ☒ 23
 ☒ 24
 ☒ 25
 ☒ 26
 ☒ 27
 ☒ 28
 ☒ 29
 ☒ 30

DBSCAN

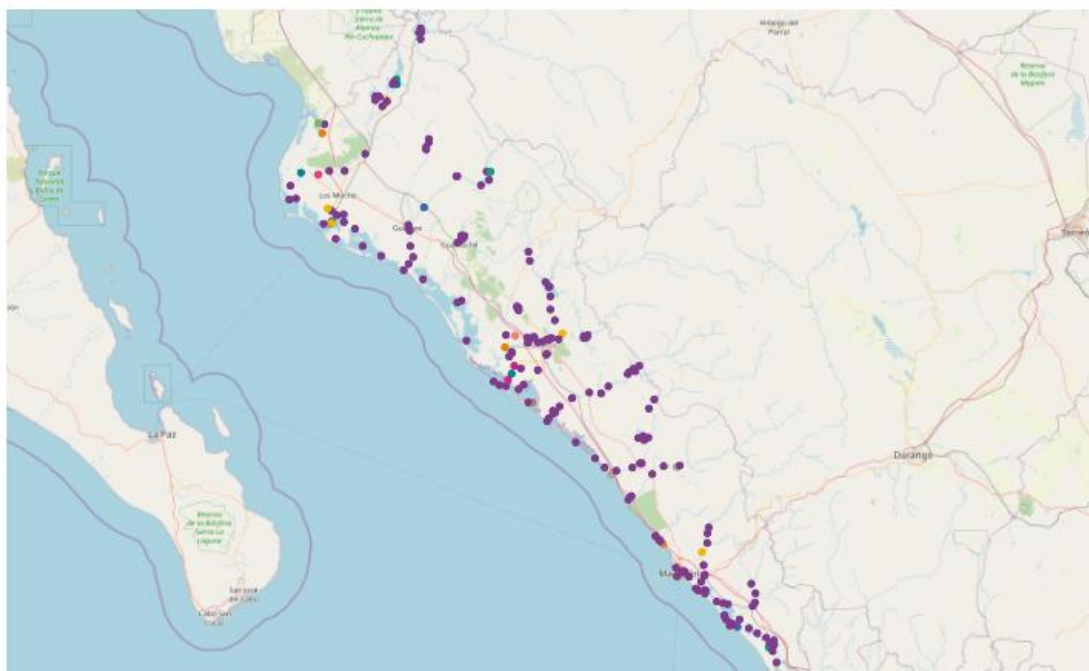


Figura 5-10. Visualización geográfica de clústeres generados: DBSCAN (*Nota. elaboración propia*)

Clustering - Sitios de muestreo Sinaloa

Selección de parámetros

☒ COLI_FEC
 ☒ COT
 ☒ COT_SOL
 ☒ N_NH3
 ☒ N_NO2
 ☒ N_NO3
 ☒ N_ORG
 ☒ N_TOT
 ☒ N_TOTK
 ☒ P_TOT
 ☒ ORTO_PO4
 ☒ COLOR_VER
 ☒ ABS_UV
 ☒ SDT
 ☒ SST
 ☒ TURBIDAD
 ☒ AS_TOT
 ☒ CO_TOT
 ☒ CR_TOT
 ☒ HG_TOT
 ☒ NIL_TOT
 ☒ PB_TOT
 ☒ TEMP_AMB

K-Means

n clusters 30

Reiniciar

Generar clusters
 Descargar CSV

Filtrar clusters generados

☒ 1
 ☒ 2
 ☒ 3
 ☒ 4
 ☒ 5
 ☒ 6
 ☒ 7
 ☒ 8
 ☒ 9
 ☒ 10
 ☒ 11
 ☒ 12
 ☒ 13
 ☒ 14
 ☒ 15
 ☒ 16
 ☒ 17
 ☒ 18
 ☒ 19
 ☒ 20
 ☒ 21
 ☒ 22
 ☒ 23
 ☒ 24
 ☒ 25
 ☒ 26
 ☒ 27
 ☒ 28
 ☒ 29
 ☒ 30

K-Means

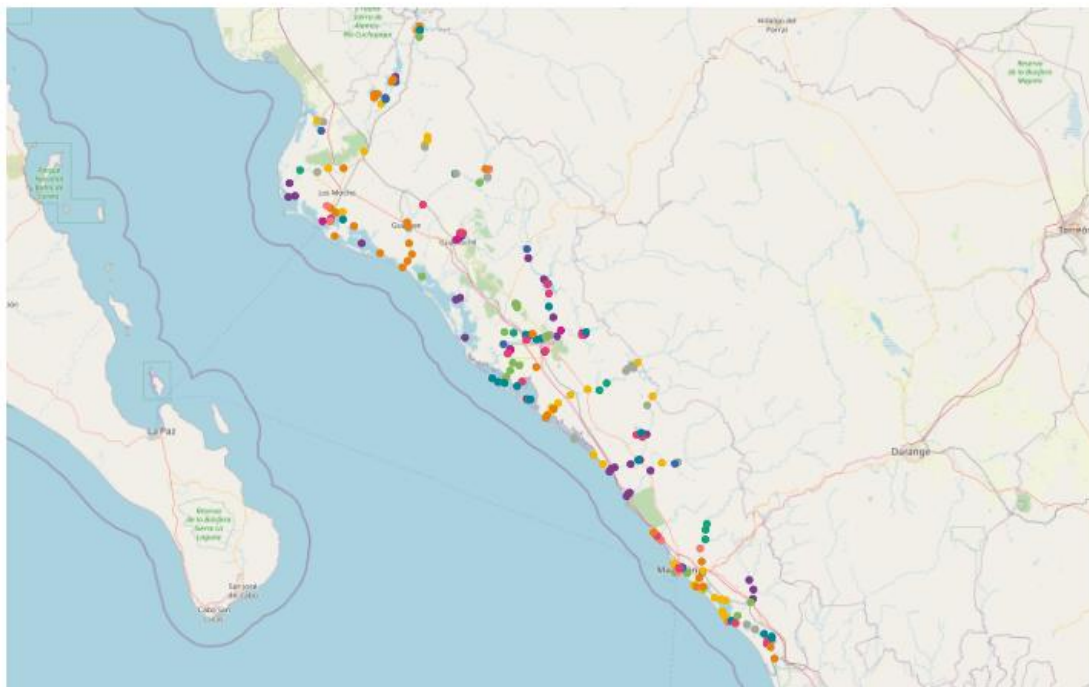


Figura 5-11. Visualización geográfica de clústeres generados: K-Means (*Nota. elaboración propia*)

Clustering - Sitios de muestreo Sinaloa

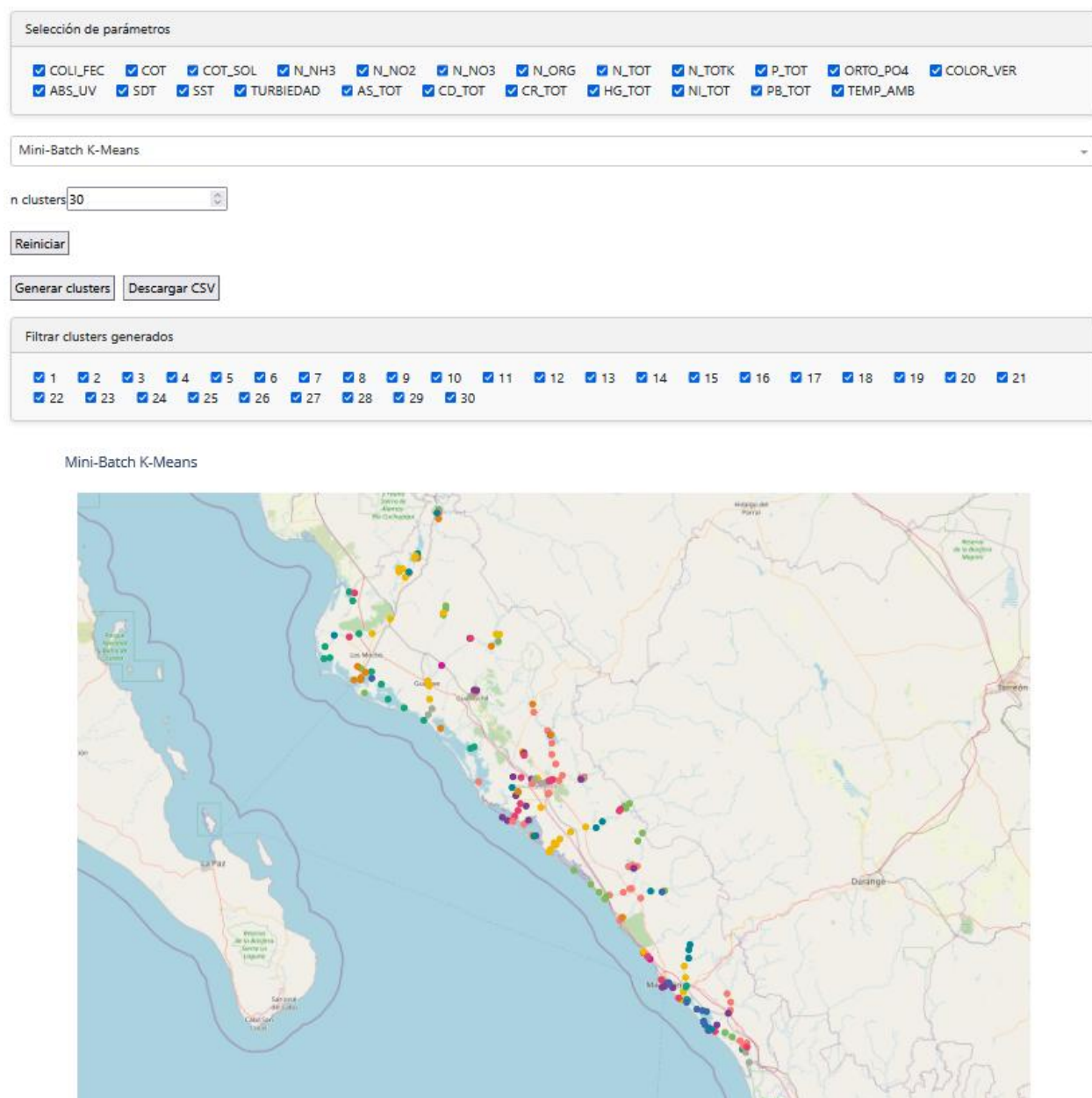


Figura 5-12. Visualización geográfica de clústeres generados: Mini-Batch K-Means (Nota. elaboración propia)

Capítulo 6

6. Conclusiones y trabajo futuro

6.1. Conclusiones

OSMIAA tuvo como objetivo principal la optimización de la red de sitios de muestreo en cuerpos de agua superficiales del estado de Sinaloa, mediante la aplicación de técnicas de aprendizaje automático no supervisado, enfocándose particularmente en algoritmos de agrupamiento y reducción de dimensionalidad (PCA). Desde su planteamiento inicial, se consideró que la implementación de estas técnicas podría resultar en una mejor distribución de los puntos de muestro, sin afectar la representatividad y precisión de los datos.

Luego de un proceso riguroso de limpieza, filtrado y preprocesamiento de los datos proporcionados por CONAGUA, se logró delimitar el análisis exclusivamente a cuerpos de agua superficiales en Sinaloa. Después, se evaluaron múltiples algoritmos de agrupamiento, integrando PCA para mejorar la visualización y desempeño de los modelos.

Los resultados obtenidos fueron satisfactorios, ya que cada algoritmo permitió identificar agrupamientos coherentes entre los sitios analizados. Esto facilitó la detección de patrones territoriales que ayudarán a la identificación de redundancias y oportunidades de optimización en la selección de puntos clave para el monitoreo. Asimismo, la representación visual de los resultados en mapas aportó una dimensión analítica adicional, lo que permitió interpretar la distribución y coherencia de los clústeres generados.

Entre los aportes más relevantes de esta investigación destacó el desarrollo de una aplicación web interactiva, diseñada para facilitar el análisis exploratorio. Esta herramienta ofreció al usuario una serie de funcionalidades que no solo facilitaron la comprensión del comportamiento de los algoritmos, sino que también ayudaron a experimentar y tomar decisiones fundamentadas en tiempo real con los resultados obtenidos en mapas del estado sinaloense.

En consecuencia, se puede afirmar que los objetivos planteados fueron cumplidos satisfactoriamente. La hipótesis inicial, en donde se propuso que el aprendizaje automático no supervisado permitiría optimizar la selección y distribución de sitios de muestreo en cuerpos de agua sinaloenses, fue validada al haber obtenido



resultados positivos. OSMIAA representa una alternativa flexible y de bajo costo para apoyar en la toma de decisiones sobre el monitoreo de la calidad del agua, especialmente útil en instituciones que cuentan con recursos limitados.

6.2. Trabajo a futuro

A partir de los resultados alcanzados con OSMIAA, se identificaron diversas líneas de trabajo que podrán ser abordadas en investigaciones posteriores. En particular, se propone la continuación de OSMIAA como base para un trabajo de investigación a nivel maestría, que podrá presentar un enfoque complementario al establecido por el sistema.

Este nuevo planteamiento buscará representar los datos de una manera diferente, alejándose del promedio simple por sitio de muestreo. Se explorarán nuevas técnicas de transformación de los datos que puedan enriquecer la interpretación de los patrones detectados. Además, se contempla el desarrollo de métodos para la selección automatizada de sitios clave dentro de cada clúster generado, con el fin de identificar puntos óptimos que permitan inferir con precisión los valores del resto de sitios agrupados.

Esta propuesta abre la puerta a estudiar técnicas de inferencia de datos que permitan estimar los valores de la calidad del agua en sitios no muestreados originalmente, siendo esto a partir del comportamiento encontrado en sitios seleccionados estratégicamente. Esto no solo incrementaría la eficiencia del monitoreo ambiental, sino que también haría viable la realización de proyecciones más robustas para escenarios futuros o situaciones de emergencia donde los recursos para el muestreo directo sean aún más limitados.



7. Bibliografía

- Ahmad, I. (2024). 50 algoritmos que todo programador debe conocer. Marcombo. https://www.google.com.mx/books/edition/50_algoritmos_que_todo_programador_debe/bxEIEQAAQBAJ
- Agudelo, O. H. & Lozano, J. D. (2025). Diseño de un prototipo modular diferenciado para monitoreo de niveles contaminantes medioambientales bajo la implementación de sensores inalámbricos y tecnología IoT. [Proyecto aplicado]. Repositorio Institucional UNAD. <https://repository.unad.edu.co/handle/10596/70037>
- Bobadilla-Sancho, J. (2020). Machine learning y deep learning: Usando Python, Scikit y Keras. Ra-Ma. https://www.google.com.mx/books/edition/Machine_Learning_y_Deep_Learning/2c-4EAAAQBAJ?hl=es&gbpv=1&printsec=frontcover
- Brownlee, J. (2020). 10 Clustering Algorithms With Python. Machine Learning Mastery, 02/05/2025, <https://machinelearningmastery.com/clustering-algorithms-with-python/>
- Centro de Estudios de las Finanzas Públicas (CEFP). (2019). Cambios en materia de Derechos por la extracción de agua para uso agrícola y pecuario. Palacio Legislativo de San Lázaro. <https://www.cefp.gob.mx/publicaciones/nota/2019/notacefp0602019.pdf>
- ChrisAllcid. (2018). Tutorial PyTorch: Aprendizaje profundo (Deep learning) en Python. IntelDig, 02/05/2025, <https://www.inteldig.com/2018/05/tutorial-pytorch-aprendizaje-profundo-deep-learning-en-python/>
- Chugani, V. (2024). Comprender la distancia euclidiana: De la teoría a la práctica. 02/05/2025, <https://www.datacamp.com/es/tutorial/euclidean-distance>
- Chugani, V. (2024). ¿Qué es la distancia Manhattan? 02/05/2025, <https://www.datacamp.com/es/tutorial/manhattan-distance>
- Comisión Nacional del Agua. (s.f.). Procedimiento de muestreo, análisis y reporte de calidad de las aguas residuales. Gobierno de México. https://www.conagua.gob.mx/conagua07/Declaragua/Procedimiento_Muestreo.pdf
- Comisión Nacional del Agua. (2025). Resultados de la Red Nacional de Medición de Calidad del Agua (RENAMECA). Gobierno de México. <https://www.gob.mx/conagua/es/articulos/resultados-de-la-red-nacional-de-medicion-de-calidad-del-agua-renameca>
- Contreras-Bravo, L. E., Fuentes-López, H. J., & Rodríguez-Molano, J. I. (2024). Algoritmos supervisados y de ensamble con Python: Implementación y estrategia de optimización. Ediciones de la U. https://www.google.com.mx/books/edition/Algoritmos_supervisados_y_de_ensamble_co/dCweEQAAQBAJ?hl=es&gbpv=1&q=lenguaje+supervisado&pg=PA93&printsec=frontcover
- CSR Staff. (2025). Norma Oficial Mexicana NOM-127-SSA1-2021, Agua para uso y consumo humano. CSR Consulting. 02/05/2025, <https://www.csrconsulting.com.mx/2025/01/14/norma-oficial-mexicana-nom-127-ssa1-2021-agua-para-uso-y-consumo-humano/>



- Ester, M., Kriegel, H., Sander, J., & Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. KDD-96 Proceedings (págs. 226–231). AAAI Press. <https://file.biolab.si/papers/1996-DBSCAN-KDD.pdf>
- FIDA. (2021). ¿Por qué el agua es esencial para los sistemas alimentarios sostenibles? FIDA, 08/07/2025, <https://www.ifad.org/es/w/explicadores/por-que-el-agua-es-esencial-para-los-sistemas-alimentarios-sostenibles>
- Frey, B. J., & Dueck, D. (2007). Clustering by Passing Messages Between Data Points. Science (págs. 972–976). <https://utstat.toronto.edu/reid/sta414/frey-affinity.pdf>
- IBM Corporation. (2021). ¿Qué es el aprendizaje no supervisado? IBM, 02/05/2025, <https://www.ibm.com/mx-es/topics/unsupervised-learning>
- Gobierno de México. (s.f.). Comisión Nacional del Agua (CONAGUA): Gestión Integral de los Recursos Hídricos en México. Gobierno de México, 08/07/2025, <https://gobierno.com.mx/conagua-recursos-hidricos.html>
- Gutiérrez Jaber, I. (2024). Internet de las cosas para monitorear la calidad del agua. TecScience, 02/05/2025, <https://tecscience.tec.mx/es/tecnologia/internet-de-las-cosas-y-calidad-del-agua/>
- Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: a review and recent developments. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences (págs. 2–16). <https://royalsocietypublishing.org/doi/pdf/10.1098/rsta.2015.0202>
- Kaufman, L., & Rousseeuw, P. J. (2005). Finding groups in data: An introduction to cluster analysis. John Wiley & Sons, Inc. https://books.google.com.mx/books?id=YeFQHiikNo0C&printsec=frontcover&hl=es&source=gbs_ge_summary_r#v=onepage&q&f=false
- López-Ollarbez, A., & Gómez-Abreu, E. (2021). Metodología para selección de portafolios de inversión a través del análisis de clúster y programación lineal entera binaria. Ingeniería Industrial: Actualidad y Nuevas Tendencias, (Vol. 7, págs. 7–26). <https://servicio.bc.uc.edu.ve/ingenieria/revista/Inge-Industrial/vol7-n26/art01.pdf>
- Liu, H., Yang, J., Ye, M., James, S. C., Tang, Z., Dong, J., & Xing, T. (2021). Using t-distributed Stochastic Neighbor Embedding (t-SNE) for cluster analysis and spatial zone delineation of groundwater geochemistry data. Journal of Hydrology, (Vol. 597, págs. 1–13). <https://www.sciencedirect.com/science/article/abs/pii/S0022169421001931>
- MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, (Vol. 1, págs. 281–297). University of California Press. <https://scispace.com/pdf/some-methods-for-classification-and-analysis-of-multivariate-4pswti19oz.pdf>
- Maimon, O., & Rokach, L. (Eds.). (2010). Data Mining and Knowledge Discovery Handbook (2.^a ed.). Springer. https://tanthiamhuat.wordpress.com/wp-content/uploads/2015/04/data_mining_and_knowledge_discovery_handbook.pdf
- Matplotlib Development Team. (s.f.). scatter(x, y). Matplotlib documentation, 02/05/2025, https://matplotlib.org/stable/plot_types/basic/scatter_plot.html#sphx-glr-plot-types-basic-scatter-plot-py



- Mendívil-García, K., Medina, J. L., Rodríguez-Rangel, H., Roé-Sosa, A., & Amábilis-Sosa, L. E. (2024). Optimization of the water quality monitoring network in a basin with intensive agriculture using artificial intelligence algorithms. *Water Supply*, 24(1), págs. 205–222. <https://iwaponline.com/ws/article/24/1/204/99523/Optimization-of-the-water-quality-monitoring>
- Murel, J., & Kavlakoglu E. (2024). ¿Qué es la reducción de la dimensionalidad? IBM Think, 02/05/2025, <https://www.ibm.com/mx-es/think/topics/dimensionality-reduction>
- Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press. <https://raw.githubusercontent.com/kerasking/book-1/master/ML%20Machine%20Learning-A%20Probabilistic%20Perspective.pdf>
- ONU. (s.f.). Agua. Naciones Unidas, 02/05/2025, <https://www.un.org/es/global-issues/water>
- Neurochispas. (s.f.). Teorema de Pitágoras – Demostración e Historia. 02/05/2025, <https://www.neurochispas.com/matematicas/teorema-de-pitagoras-historia-demostracion-y-ejemplos/>
- Nilsson, N. J. (1998). *Artificial Intelligence: a new synthesis*. Morgan Kaufmann Publishers, Inc. <https://books.google.com.mx/books?id=GYOFsdfETgC&printsec=frontcover#v=onepage&q&f=false>
- Noble, J. (2024). ¿Qué es el clustering jerárquico? IBM Think, 02/05/2025, <https://www.ibm.com/es-es/think/topics/hierarchical-clustering>
- NumPy Development Team. (s.f.). NumPy. NumPy documentation, 02/05/2025, <https://numpy.org/>
- Oliphant, T. E. (2007). *Python for Scientific Computing*. Computing in Science & Engineering, 9(3), págs. 10–20. https://www.researchgate.net/publication/3422935_Python_for_Scientific_Computing
- Pandas Development Team. (s.f.). Package overview. Pandas documentation, 02/05/2025, https://pandas.pydata.org/docs/getting_started/overview.html#development-team
- Perevochtchikova, M. (2009). La situación actual del sistema de monitoreo ambiental en la Zona Metropolitana de la Ciudad de México. Estudios demográficos y urbanos, (Vol. 24, págs. 513–547). <https://estudiosdemograficosyurbanos.colmex.mx/index.php/edu/article/view/1327>
- Plotly Development Team. (s.f.). Dash in 20 minutes Tutorial. Dash documentation, 02/05/2025, <https://dash.plotly.com/tutorial>
- Plotly Development Team. (s.f.). Dash Python User Guide. Dash documentation, 02/05/2025, <https://dash.plotly.com/>
- Plotly Development Team. (s.f.). Plotly Open Source Graphing Library for Python. Plotly documentation, 02/05/2025, <https://plotly.com/python/>
- Plotly Development Team. (s.f.). Scatter Plots on Tile Maps in Python. Plotly documentation. 15/08/2025, Plotly documentation, <https://plotly.com/python/tile-scatter-maps/>
- Render Development Team. (s.f.). Render Documentation. Render documentation, 19/05/2025, <https://render.com/docs>



- Ríos, M. D. (2023). Inteligencia artificial: Cuando la tecnología es el menor de los paradigmas (Artificial Intelligence: When Technology Is the Smallest of the Paradigms). SSRN. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4521736
- Scikit-learn Development Team. (s.f.). Scikit-learn: Machine learning in Python. Scikit-learn documentation, 02/05/2025, <https://scikit-learn.org/stable/>
- SciPy Development Team. (s.f.). SciPy: Fundamental algorithms for scientific computing in Python. SciPy documentation, 02/05/2025, <https://scipy.org/>
- Sculley, D. (2010). Web-Scale K-Means Clustering. Proceedings of the 19th International Conference on World Wide Web (WWW '10), (págs. 1177–1178). ACM. <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=b452a856a3e3d4d37b1de837996aa6813bedfdcf>
- Ter Sahakyan, E. (2019). How to create a Plotly visualization and embed it on websites. Medium, 02/05/2025, <https://medium.com/data-science/how-to-create-a-plotly-visualization-and-embed-it-on-websites-517c1a78568b>
- Tsegaye, S., Kebedew, M. G., Albrecht, K. K., Missimer, T. M., Thomas, S., & Elshall, A. S. (2024). Integrated GIS-hydrologic-hydraulic modeling to assess combined flood drivers in coastal regions: a case study of Bonita Bay, Florida. *Frontiers in Water*, (Vol. 6, págs. 1–14). [Frontiers | Integrated GIS-hydrologic-hydraulic modeling to assess combined flood drivers in coastal regions: a case study of Bonita Bay, Florida](#)
- VanderPlas, J. (2016). Python Data Science Handbook: Essential Tools for Working with Data. O'Reilly Media, Inc.
- Zamudio-Trujillo, D.A., Mendivil-García K, Amábilis-Sosa L.E., Medina-Jimenez J.L., Rodríguez-Rangel H. (2024). Inteligencia Artificial como alternativa para la calidad del agua. *Revista Ciencia y Naturaleza* (1108).
- Zhang, T., Ramakrishnan, R., & Livny, M. (1998). BIRCH: An Efficient Data Clustering Method for Very Large Databases. Proceedings of the ACM SIGMOD International Conference on Management of Data, (págs. 103–114). ACM. <https://dl.acm.org/doi/pdf/10.1145/235968.233324>



8. Anexos

A. Código de la aplicación web

```
import os
import pandas as pd
from sklearn.cluster import AffinityPropagation, Birch, DBSCAN, KMeans, MiniBatchKMeans
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import dash as dash
from dash_extensions import EventListener
import dash_bootstrap_components as dbc
import plotly.express as px

# Cargar datos
coordenadas = pd.read_csv("datasets/sitios_sinaloa_coordenadas.csv")
X = pd.read_csv('datasets/AguaSinaloa_2.csv')

# excluir las dos primeras columnas de identificación,
# se guardan las columnas numericas con las se van a generar los clusters en columnas_numericas
columnas_numericas = X.columns[2:]

# nombre completo de todos los parametros de la base de datos
traducciones_parametros_bd = {
    "COLI_FEC": "Coliformes Fecales",
    "COT": "Carbono Orgánico Total",
    "COT_SOL": "Carbono Orgánico Soluble",
    "N_NH3": "Nitrógeno Amoniacal",
    "N_NO2": "Nitrógeno de Nitritos",
    "N_NO3": "Nitrógeno de Nitratos",
    "N_ORG": "Nitrógeno Orgánico",
    "N_TOT": "Nitrógeno Total (Cálculo)",
    "N_TOTK": "Nitrógeno Kjeldahl",
    "P_TOT": "Fósforo Total",
    "ORTO_PO4": "Fósforo Reactivo total (o-fosfatos)",
    "COLOR_VER": "Color Verdadero",
    "ABS_UV": "Absorción UV",
    "SDT": "Sólidos Disueltos Totales (Cálculo)",
    "SST": "Sólidos Suspendidos Totales",
    "TURBIEDAD": "Turbiedad",
    "AS_TOT": "Arsénico Total",
    "CD_TOT": "Cadmio Total",
    "CR_TOT": "Cromo Total",
    "HG_TOT": "Mercurio Total",
    "NI_TOT": "Níquel Total",
    "PB_TOT": "Plomo Total",
    "TEMP_AMB": "Temperatura Ambiente"
}

# inicializar la app Dash
app = dash.Dash(__name__, suppress_callback_exceptions=True, external_stylesheets=[dbc.themes.BOOTSTRAP])

# establecer etiquetas y datos iniciales para los algoritmos
app.layout = dbc.Container([
    dash.html.H1("Clustering - Sitios de muestro Sinaloa"),

    dash.html.Br(),
    dbc.Card([
        dbc.CardHeader("Selección de parámetros"),
        dbc.CardBody([
            dash.dcc.Checklist(
                id="columnas-agrupacion",
```



```

        options = [
            {
                "label": dash.html.Span(col, title=desc),
                "value": col
            }
            for col, desc in traducciones_parametros_bd.items()
        ],
        value=list(traducciones_parametros_bd.keys()),
        inline=True,
        inputStyle={"margin-right": "5px", "margin-left": "10px"},
        labelStyle={"display": "inline-block", "margin-right": "15px"}
    )
    ],
    style={
        "margin-top": "20px",
        "background-color": "#f9f9f9",
        "box-shadow": "0 2px 5px rgba(0,0,0,0.1)"
    },
    dash.html.Br(),
    dash.dcc.Dropdown(
        id="algoritmo-dropdown",
        options=[
            {"label": "Affinity Propagation", "value": "Affinity Propagation"},
            {"label": "BIRCH", "value": "BIRCH"},
            {"label": "DBSCAN", "value": "DBSCAN"},
            {"label": "K-Means", "value": "K-Means"},
            {"label": "Mini-Batch K-Means", "value": "Mini-Batch K-Means"}
        ],
        placeholder="Selecciona un algoritmo",
        searchable=False,
        clearable=False
    ),
    dash.html.Br(),
    dash.html.Div([
        dash.html.Div([dash.html.Label(dash.html.Span("damping"), title="Atenuación"), dash.dcc.Input(id="damping",
type="number", value=0.7, step=0.1)], id="damping-div"),
        dash.html.Div([dash.html.Label(dash.html.Span("preference"), title="Preferencia"), dash.dcc.Input(id="preference",
type="number", value=-8.8, step=0.01)], id="preference-div"),
        dash.html.Div([dash.html.Label(dash.html.Span("threshold"), title="Umbral"), dash.dcc.Input(id="threshold",
type="number", value=0.7, step=0.01)], id="threshold-div"),
        dash.html.Div([dash.html.Label(dash.html.Span("n clusters"), title="Número de clusters"),
dash.dcc.Input(id="n_clusters", type="number", value=30, step=1)], id="n_clusters-div"),
        dash.html.Div([dash.html.Label(dash.html.Span("epsilon"), title="Epsilon"), dash.dcc.Input(id="eps",
type="number", value=1.82, step=0.01)], id="eps-div"),
        dash.html.Div([dash.html.Label(dash.html.Span("min samples"), title="Mínimo de muestras"),
dash.dcc.Input(id="min_samples", type="number", value=1, step=1)], id="min_samples-div"),
    ], id="parametros-container"),
    dash.html.Br(),
    dash.html.Div([
        EventListener(
            dash.html.Button("Reiniciar", id="reload-button"),
            events=[{"event": "click", "props": {"href": "/", "target": "_self"}}]
        ),
        dash.dcc.Location(id='url', refresh=True)
    ]),
    dash.html.Br(),
    dash.html.Button("Generar clusters", id="run-button", n_clicks=0, style={'margin-right': '10px'}, disabled=True),
    dash.html.Button("Descargar CSV", id="download-button", n_clicks=0, disabled=True),
    dash.dcc.Download(id="descarga-csv"),
    dash.html.Br(),
    dash.dcc.Store(id="clustering-data"),
    dbc.Card([
        dbc.CardHeader("Filtrar clusters generados"),
        dbc.CardBody([

```



```

dash.dcc.Checklist(
    id="cluster-checklist",
    options=[],
    value=[],
    inline=True,
    inputStyle={"margin-right": "5px", "margin-left": "10px"},
    labelStyle={"display": "inline-block", "margin-right": "15px"}
)
])
], style={
    "margin-top": "20px",
    "background-color": "#f9f9f9",
    "box-shadow": "0 2px 5px rgba(0,0,0,0.1)"
}),
dash.dcc.ConfirmDialog(
    id='mensaje-error',
    message="Cantidad de parámetros no valida",
    displayed=False
),
dash.html.Br(),
dash.dcc.Graph(id="mapa-clustering")
])

# callback y método para el botón de Reiniciar
@app.callback(
    dash.Output("url", "href"),
    [dash.Input("reload-button", "n_clicks")]
)
def reiniciar(n_clicks):
    if n_clicks:
        return "/"
    return None

# callback para mostrar y ocultar los parámetros según el algoritmo seleccionado
@app.callback(
    [dash.Output("damping-div", "style"),
     dash.Output("preference-div", "style"),
     dash.Output("threshold-div", "style"),
     dash.Output("n_clusters-div", "style"),
     dash.Output("eps-div", "style"),
     dash.Output("min_samples-div", "style")],
    dash.Input("algoritmo-dropdown", "value")
)

# se actualizan los mapas al cambiar los valores de los parámetros
def actualizar_parametros(algoritmo):
    # se oculta todo al inicio
    oculto = {"display": "none"}
    visible = {}

    return (
        visible if algoritmo == "Affinity Propagation" else oculto, # damping
        visible if algoritmo == "Affinity Propagation" else oculto, # preference
        visible if algoritmo == "BIRCH" else oculto, # threshold
        visible if algoritmo in ["BIRCH", "K-Means", "Mini-Batch K-Means"] else oculto, # n_clusters
        visible if algoritmo == "DBSCAN" else oculto, # eps
        visible if algoritmo == "DBSCAN" else oculto, # min_samples
    )

# callback y método para habilitar el boton Generar Clustering al seleccionar un algoritmo
@app.callback(
    dash.Output("run-button", "disabled"),
    dash.Input("algoritmo-dropdown", "value")
)

def habilitar_boton(algoritmo):
    return algoritmo is None

# callback y método para generar el mapa de clustering

```



```

@app.callback(
    dash.Output("mensaje-error", "message"),
    dash.Output("mensaje-error", "displayed"),
    dash.Output("clustering-data", "data"),
    dash.Input("run-button", "n_clicks"),
    [
        dash.State("algoritmo-dropdown", "value"),
        dash.State("damping", "value"),
        dash.State("preference", "value"),
        dash.State("threshold", "value"),
        dash.State("n_clusters", "value"),
        dash.State("eps", "value"),
        dash.State("min_samples", "value"),
        dash.State("columnas-agrupacion", "value")
    ]
)
def generar_clustering(n_clicks, metodo, damping, preference, threshold, n_clusters, eps, min_samples,
    columnas_seleccionadas):
    # evitar ejecucion al iniciar la aplicacion
    if n_clicks is None or n_clicks == 0:
        return "", False, dash.no_update

    # agrupar por "CLAVE SITIO" y calcular el promedio de cada columna numérica seleccionada
    datos_filtrados = X.groupby("CLAVE SITIO")[columnas_seleccionadas].mean()

    # validacion adicional para evitar problemas en la generacion de clusters
    if not metodo or not columnas_seleccionadas or datos_filtrados.shape[1] < 7:
        return "Debe seleccionar un algoritmo y al menos 7 columnas (PCA) para el clustering.", True, dash.no_update

    # validaciones adicionales para los parámetros de entrada
    if metodo == "Affinity Propagation":
        if damping is None or not (0.5 <= damping < 1.0):
            return "Introduzca un valor entre 0.5 y 1.0", True, dash.no_update
        try:
            float(preference)
        except (ValueError, TypeError):
            return "Introduzca un valor numérico", True, dash.no_update

    elif metodo == "BIRCH":
        if threshold is None or threshold <= 0:
            return "Introduzca un valor mayor a 0.0", True, dash.no_update
        if n_clusters is None or not (1 <= n_clusters <= 217):
            return "Introduzca un valor entre 1 y 217", True, dash.no_update

    elif metodo == "DBSCAN":
        if eps is None or eps <= 0:
            return "Introduzca un valor mayor a 0.0", True, dash.no_update
        if min_samples is None or min_samples < 1:
            return "Introduzca un valor mayor o igual a 1", True, dash.no_update

    elif metodo in ["K-Means", "Mini-Batch K-Means"]:
        if n_clusters is None or not (1 <= n_clusters <= 217):
            return "Introduzca un valor entre 1 y 217", True, dash.no_update

    try:
        datos_filtrados = X.groupby("CLAVE SITIO")[columnas_seleccionadas].mean()
        datos_filtrados = StandardScaler().fit_transform(datos_filtrados)
        X_pca = PCA(n_components=7).fit_transform(datos_filtrados)

        if metodo == "Affinity Propagation":
            model = AffinityPropagation(damping=damping, preference=preference)
        elif metodo == "BIRCH":
            model = Birch(threshold=threshold, n_clusters=n_clusters)
        elif metodo == "DBSCAN":
            model = DBSCAN(eps=eps, min_samples=min_samples)
        elif metodo == "K-Means":
            model = KMeans(n_clusters=n_clusters)
        elif metodo == "Mini-Batch K-Means":
            model = MiniBatchKMeans(n_clusters=n_clusters)

```



```

model.fit(X_pca)
yhat = model.labels_
coordenadas["Cluster"] = (yhat + 1).astype(str)

# validación adicional para el algoritmo BIRCH
if metodo == "BIRCH":
    cantidad_clusters_reales = len(set(yhat)) - (1 if -1 in yhat else 0)
    if cantidad_clusters_reales < n_clusters:
        return (
            f"Se generaron solo {cantidad_clusters_reales} clusters de los {n_clusters} solicitados. Disminuya
el valor de 'threshold'.",
            True,
            dash.no_update
        )

fig = px.scatter_mapbox(
    coordenadas, lat="LATITUD", lon="LONGITUD",
    color="Cluster", hover_name="NOMBRE DEL SITIO",
    category_orders={"Cluster": sorted(coordenadas["Cluster"].unique())},
    zoom=6.6, height=900, color_discrete_sequence=px.colors.qualitative.Bold
)

fig.update_traces(marker=dict(size=10))
fig.update_layout(
    showlegend=False,
    mapbox_style="open-street-map",
    mapbox_center={
        "lat": coordenadas["LATITUD"].mean(),
        "lon": coordenadas["LONGITUD"].mean()
    },
    title=f"Clustering {metodo}"
)

coordenadas["Cluster"] = coordenadas["Cluster"].astype(str)

return "", False, {
    "datos": coordenadas.to_dict("records"),
    "metodo": metodo
}

except Exception as e:
    return f"Ocurrió un error inesperado: {str(e)}", True, dash.no_update

# callback y método para actualizar el checklist con los clusters detectados
@app.callback(
    [dash.Output("cluster-checklist", "options"),
    dash.Output("cluster-checklist", "value")],
    dash.Input("clustering-data", "data")
)
def actualizar_checklist(data):
    if not data or "datos" not in data:
        raise dash.exceptions.PreventUpdate

    df = pd.DataFrame(data["datos"])
    df["Cluster"] = df["Cluster"].astype(str)

    # ordenar los clusters en el checklist
    opciones = [{"label": str(c), "value": str(c)} for c in sorted(df["Cluster"].astype(int).unique())]
    seleccionados = [op["value"] for op in opciones]

    return opciones, seleccionados

# callback y método para actualizar el mapa al seleccionar los clusters detectados en el checklist
@app.callback(
    dash.Output("mapa-clustering", "figure"),
    [dash.Input("clustering-data", "data"),
    dash.Input("cluster-checklist", "value")]
)

```



```

def actualizar_mapa(data, clusters_filtrados):
    if not data or "datos" not in data or not clusters_filtrados:
        raise dash.exceptions.PreventUpdate

    df = pd.DataFrame(data["datos"])
    metodo = data.get("metodo", "desconocido")
    df["Cluster"] = df["Cluster"].astype(str)
    df_filtrado = df[df["Cluster"].isin(clusters_filtrados)]

    fig = px.scatter_mapbox(
        df_filtrado, lat="LATITUD", lon="LONGITUD",
        color="Cluster", hover_name="NOMBRE DEL SITIO",
        category_orders={"Cluster": sorted(df["Cluster"].unique())},
        zoom=6.6, height=900, color_discrete_sequence=px.colors.qualitative.Bold
    )
    fig.update_traces(marker=dict(size=10))
    fig.update_layout(
        showlegend=False, mapbox_style="open-street-map",
        mapbox_center={"lat": df_filtrado["LATITUD"].mean(), "lon": df_filtrado["LONGITUD"].mean()},
        title=f"{metodo}"
    )

    return fig

# callback y método para habilitar el botón de descarga
@app.callback(
    dash.Output("download-button", "disabled"),
    dash.Input("run-button", "n_clicks")
)
def habilitar_descarga(n_clicks):
    # habilitar el botón de descarga si se ha generado un clustering
    return n_clicks is None or n_clicks == 0

# callback y método para descargar el CSV del clustering generado, según lo que se haya seleccionado en el checklist
@app.callback(
    dash.Output("descarga-csv", "data"),
    [dash.Input("download-button", "n_clicks"),
     dash.State("cluster-checklist", "value")],
    prevent_initial_call=True
)
def descargar_csv(n_clicks, checklist_value):
    try:
        # filtrar según las opciones seleccionadas en el checklist
        if checklist_value:
            df_filtrado = coordenadas[coordenadas["Cluster"].isin(checklist_value)]
        else:
            df_filtrado = coordenadas

        # se guardan los clusters generados y se ordenan por la columna "Cluster"
        df_filtrado = df_filtrado[["NOMBRE DEL SITIO", "Cluster"]]
        df_filtrado["Cluster"] = df_filtrado["Cluster"].astype(int)
        df_filtrado = df_filtrado.sort_values("Cluster")

        # devolver el archivo CSV filtrado
        return dash.dcc.send_data_frame(df_filtrado.to_csv, "clusters_asignados_sitios.csv", index=False)

    except Exception as e:
        print("Error al generar el archivo a descargar")
        print(str(e))
        return None

if __name__ == "__main__":
    # se establece el puerto para correr la app Dash en Render
    port = int(os.environ.get("PORT", 8050))
    app.run(debug=False, host="0.0.0.0", port=port)

```



B. Carta de no inconveniente

Hoja membretada de la empresa:

(con RFC y dirección fiscal)

RFC: TNM140723GFA

JUAN DE DIOS BÁTIZ #310 PTE GUADALUPE 80220 CULIACÁN SINALOA

Culiacán, Sin., a 20 de junio de 2025.

Asunto: Carta de no inconveniencia.

MARISOL MANJARREZ BELTRÁN

Asesor interno del Instituto tecnológico de Culiacán

Por medio de la presente, se hace constar que el(la) C. **EDGAR MICHEL PEREZ RAMIREZ**, alumno(a) del Instituto Tecnológico de Culiacán de la carrera de **ING. EN SISTEMAS COMPUTACIONALES** y con número de control **20170778**, ha concluido satisfactoriamente su Proyecto de Residencia Profesional y aspira a obtener el título profesional con el informe del proyecto denominado **OPTIMIZACIÓN DE SITIOS DE MUESTRO DE SINALOA EN AGUAS SUPERFICIALES IMPLEMENTANDO APRENDIZAJE AUTOMÁTICO**, que realizó en esta empresa, un período **Enero-Junio del 2025**.

Por lo que **no se tiene ningún inconveniente en el uso de la información contenida** en dicho informe para que se utilice únicamente para los fines antes mencionados y se concede la autorización para su uso ante el Instituto Tecnológico de Culiacán, con el fin de que [el/la] interesado(a) alcance su objetivo.

Atentamente

TECNOLÓGICO NACIONAL DE MÉXICO / INSTITUTO TECNOLÓGICO DE CULIACÁN

TNM140723GFA



DR. HÉCTOR RODRÍGUEZ RANGEL
PROFESOR TITULAR C

[Sello de la empresa]

Nota: si no se tiene sello u hoja membretada con RFC, poner al final del escrito el RFC de la empresa y dirección fiscal.

