INSTITUTO TECNOLÓGICO SUPERIOR DE XALAPA

TÍTULO DEL PROYECTO

Diseño, implementación y evaluación de algoritmo de aprendizaje automático para control de aireadores utilizados en la crianza de tilapia para el ahorro energético

Opción de Titulación:

Tesis

Que como requisito parcial para la obtención de grado de

Maestra en Sistemas Computacionales

Presenta:

Alejandro Leyva Barrientos

No. de Control:

217O01385

Directora

Dra. Virginia Lagunes Barradas
Visto bueno_____

Co-Director

Dr. Diego Estaban Platas Rosado
Visto bueno_____

Xalapa – Enríquez, Ver., diciembre de 2023

Índice de tablas

Índice de ilustraciones

Contenido

Introducción

La producción y consumo responsables y la sostenibilidad medioambiental, son dos de los temas de investigación que se han convertido en prioridades esenciales. En este contexto, la acuicultura ha emergido como una de las industrias alimentarias de mayor crecimiento en el mundo, desempeñando un papel crucial en la producción sostenible de proteínas animales para una población global en constante expansión.

Dado lo anterior, un aspecto fundamental de la eficiencia en la acuicultura es la optimización del consumo energético, particularmente en los sistemas de aireación utilizados para mantener niveles adecuados de oxígeno disuelto en los estanques de cría. La energía utilizada para estos sistemas no solo representa un costo significativo para las operaciones acuícolas, sino que también tiene implicaciones ambientales importantes, ya que la reducción en el consumo de energía contribuye a la disminución de las emisiones de gases de efecto invernadero, apoyando así los objetivos globales de sostenibilidad.

El presente estudio se centra en la implementación y evaluación de algoritmos de optimización para mejorar la eficiencia energética de los aireadores en la cría de tilapias, una especie de pez ampliamente cultivada debido a su adaptabilidad y demanda comercial.

La investigación se basa en la metodología *CRISP-DM (Cross-Industry Standard Process for Data Mining)*, que proporciona un marco estructurado para la gestión y análisis de datos. Este enfoque metodológico asegura una comprensión integral del problema, desde la identificación de las necesidades del negocio hasta la implementación y evaluación de soluciones basadas en datos.

En la primera fase de la investigación, se realiza una comprensión detallada del negocio de la acuicultura, identificando las necesidades y desafíos específicos relacionados con el consumo energético. Esto incluye una revisión exhaustiva de la literatura existente y un análisis de los sistemas de aireación utilizados en las

granjas acuícolas en la actualidad. A partir de esta base, se procede a la recopilación de datos relevantes, que abarcan el consumo energético de los aireadores, ciertas condiciones ambientales y determinados parámetros de calidad del agua. Estos datos son sometidos a un riguroso proceso de limpieza y preprocesamiento que permita asegurar su calidad y relevancia para un análisis posterior.

La fase siguiente involucra el desarrollo de modelos de optimización mediante algoritmos genéticos, los cuales son seleccionados por su capacidad para resolver problemas complejos de optimización multiobjetivo. Estos algoritmos son diseñados para encontrar configuraciones de funcionamiento de los aireadores que minimicen el consumo energético mientras mantienen o mejoran la calidad del agua. Los modelos son evaluados mediante simulaciones que replican las condiciones operativas reales de las granjas acuícolas, y sus resultados son comparados con los métodos tradicionales de gestión de los aireadores.

Los hallazgos de este estudio tienen implicaciones prácticas significativas. En términos económicos, la optimización del consumo energético puede traducirse en una reducción considerable de los costos operativos, mejorando la rentabilidad de las granjas acuícolas. En el ámbito ambiental, la disminución en el uso de energía contribuye a la reducción de la huella de carbono de la acuicultura, alineándose con las iniciativas globales para combatir el cambio climático. Además, la mejora en la eficiencia de los sistemas de aireación puede tener un impacto positivo en la salud y el crecimiento de los peces, ya que garantiza un entorno acuático más estable y adecuado.

Finalmente, este estudio no solo busca proporcionar soluciones prácticas y aplicables para la optimización energética en la acuicultura, sino que también pretende contribuir al conocimiento científico y técnico en el campo de la gestión de recursos acuícolas. Los resultados y metodologías desarrolladas en esta investigación pueden servir de base para futuras investigaciones y aplicaciones en otras áreas de la acuicultura y en diferentes contextos geográficos. Al concluir, se presentan recomendaciones claras y concretas basadas en los resultados

obtenidos, ofreciendo una guía para la implementación de prácticas sostenibles y eficientes en las granjas acuícolas.

En resumen, esta tesis aborda la optimización energética en la acuicultura, presentando un enfoque metódico y basado en datos para desarrollar soluciones innovadoras y sostenibles que puedan ser implementadas de manera práctica, mejorando tanto la eficiencia operativa como la sostenibilidad ambiental de la industria acuícola.

Capítulo 1. Desarrollo metodológico

El desarrollo metodológico de esta investigación da a conocer la forma en cómo se va a organizar y llevar a cabo el proyecto con el fin de alcanzar el objetivo de manera satisfactoria. Este proceso sirve de guía para dar respuesta al problema consistente en optimizar el consumo de energía eléctrica derivado del uso de aireadores en un estanque piloto perteneciente a una granja acuícola dedicada a la crianza de tilapia. Lo anterior, mediante el análisis de datos y la aplicación de algoritmos que además monitorean el nivel de oxígeno disuelto permisible para la supervivencia y en el mejor de los casos, crecimiento de la especie mencionada.

El capítulo comienza con una contextualización de la acuicultura, identificando los puntos críticos y deficiencias detectadas en investigaciones previas o antecedentes. Posteriormente, se especifica la problemática a resolver junto con las preguntas de investigación correspondientes. Finalmente, se realiza la descripción de la metodología CRISP-DM utilizada para llevar a cabo la propuesta de solución.

1.1. Antecedentes

La acuicultura, fundamental en la producción alimentaria global, representa actualmente el 50% del pescado destinado al consumo alimenticio mundial (FAO, 2022). Integrando conocimientos interdisciplinarios en ecología y diversas ramas de la biología, la acuicultura incorpora ciencias aplicadas como la Bioeconomía Pesquera, que modela procesos comerciales y ecosistemas (Viamontes et al, 2019).

En México, con desarrollo formal desde 1950, la acuicultura se practica en 23 de los 32 estados, siendo Morelos el principal productor con 30 millones de peces anuales, el 70% exportado (Instituto Nacional de la Economía Social, 2018). En 2017, México generó 404 mil toneladas de pescado y mariscos (Comisión Nacional de Acuicultura y Pesca, 2018). La tilapia destaca en la piscicultura, representando el 94.3% de la pesquería nacional de la especie (Valdez Espinoza & Félix Aguilar,

2019). La crianza de tilapia, esencial para el consumo alimenticio, implica mantener niveles adecuados de oxígeno disuelto, crucial para su desarrollo (Centro de Investigación en Alimentación y Desarrollo, A.C., 2008).

Los sistemas de aireación, como turbo-máquinas, son vitales para conservar estos niveles mediante la remoción de gases del agua, dependiendo de factores como el momento de transferencia de masa y de equilibrio (Marinho-Pereira et al, 2020).

En la búsqueda de optimización de los recursos empleados, diversos investigadores han diseñado sistemas de monitoreo y control en la acuicultura. Nava (2020) remota y monitoriza niveles de oxígeno disuelto con tecnología *IoT*. Delgado & Valencia (2021) crearon un dispositivo *IoT* con *Raspberry* PI y ESP8266 para monitoreo constante de parámetros ambientales.

La inversión en costos variables, según el tipo y tamaño de empresa, denota que la energía eléctrica constituye un componente significativo (véase Tabla 1). Este panorama se vincula con la investigación de Sánchez-Ramos (2019), quien analizó la eficiencia energética en granjas acuícolas, destacando que el consumo eléctrico, principalmente en sistemas de aireación, representa un área de oportunidad para la optimización.

Tabla 1. Inversión en costos variables según el tamaño de la empresa (Torres-Tadeo el al., 2021)

COSTO VARIABLE	GRANDES EMPRESAS	MEDIANAS EMPRESAS	PEQUEÑAS EMPRESAS
ALIMENTO	65%	56%	97.2%
ENERGÍA ELÉCTRICA	28.8%	21.3%	2.2 %
ALEVINES	5.3%	22.7%	0.6%

En la tabla 1 se corrobora que el pago de energía eléctrica constituye una parte sustancial de los costos variables en la producción de tilapia. Asimismo, Sánchez-Ramos (2019) identificó deficiencias en granjas acuícolas, donde el mal dimensionamiento y operación de aireadores, junto con el monitoreo inadecuado de

estanques, son causantes comunes. Esto resalta la necesidad de abordar la optimización del consumo eléctrico en aireadores mediante técnicas de aprendizaje automático, lo que orienta las preguntas de investigación subsiguientes.

Aunado a lo anterior, se realizó un mapeo sistemático de la literatura, enfocado en el tema de investigación, le cual resaltó cuatro aspectos fundamentales:

- Sistemas de monitoreo de acuicultura.
- Procesamiento de datos en la acuicultura.
- Trabajos de internet de las cosas (*IoT*) relacionados al proceso de oxigenación de los sistemas de cultivo acuícola.
- Aprendizaje automático para el ahorro energético.

1.1.1. Sistemas de monitoreo

En este rubro existen diversos trabajos que coinciden en medir parámetros como pH, temperatura y oxígeno (Akhter, 2021; AGOSSOU, 2021; Lazo, 2021; Teixeira, 2021). Estos sistemas se concentran en analizar parámetros físico-químicos del agua, asegurando la calidad del entorno acuático para el crecimiento óptimo de los especímenes.

Sin embargo, la monitorización exclusiva de los niveles de oxigenación es un enfoque menos frecuente, observado principalmente en investigaciones de Teixeira (2021), Tawfeeq (2019), Rivera Betancur (2020) y Bryan (2021). La temperatura fue la variable más comúnmente registrada, seguida por el pH. El total de las variables mostradas en cada artículo, se muestra en la tabla 2.

Tabla 2. Variables medidas en los artículos investigador. Fuente: (Elaboración propia).

Temperatura	15	23%
PH	13	19%
Oxigen Disuelto (OD)	9	9%
Total de Solidos Disueltos (TSD)	6	9%
Turrbidez	4	5%
Salinidad	4	5%
Conductividad	3	5%
Calidad del agua	3	5%
Nitrato	2	4%
Fosfato	2	4%
Calcio	2	4%
Magnesio	2	4%
Humedad	2	4%
Nivel del agua	1	2%
Total	60	100%

Adicionalmente, el trabajo de Yao (2020) se enfocó en el monitoreo del crecimiento de peces mediante cámaras, centrándose en la evolución de los especímenes.

1.1.2. Procesamiento de datos

Aunque los sistemas de monitoreo son predominantes en muchas áreas, la acuícola no es la excepción. Algunas investigaciones implementan de manera exitosa diferentes algoritmos para el procesamiento de los datos recopilados (véase tabla 3). Los algoritmos empleados varían en los diversos proyectos de investigación, del total analizado, cinco artículos hacen uso de redes neuronales (*Neural Networks*), lo que representa alrededor del 15% de los documentos seleccionados. El agrupamiento de variables y el procesamiento para predecir el comportamiento físico-químico son las razones principales para aplicar dichos algoritmos (Akhter, 2021; AGOSSOU, 2021; Rivera Betancur, 2020).

Tabla 3. Algoritmos aplicados en los artículos investigados. Fuente: (Elaboración propia).

Neural Newtworks	5	15%
Decision Tree	4	12%
K-Means	4	12%
K-Nearest Neighbour	3	9%
Maquina de vectores (SVM)	3	9%
Naives Bayes	3	9%
ExtraTreesClassifier	2	6%
Deep Learning	2	6%
Random Forest	1	3%
Polynomial Regression	1	3%
Multiple Linear Regression	1	3%
Principal Component Analysis	1	3%
Regresión Lineal (LR)	1	3%
Support Vector Regression	1	3%
Recursive Feature Elimination	1	3%
Colonia de hormigas	1	3%
	34	

1.1.3. Trabajos de *IoT* relacionados al proceso de oxigenación de los sistemas de cultivo acuícola

Investigaciones como Hu (2022), Almalki (2021), Murali (2021), Pylianidis (2021), entre otras, resaltan el uso predominante de redes neuronales artificiales para lograr ahorro energético en diversos sistemas, desde granjas agrícolas hasta ciudades inteligentes.

Esta revisión pone en evidencia la diversidad de enfoques y aplicaciones en el campo de la acuicultura, enfatizando la necesidad de más investigaciones en áreas menos exploradas, como el procesamiento avanzado de datos y el uso de algoritmos de *IoT* para mejorar la oxigenación en sistemas acuícolas.

A partir de los antecedentes analizados, se desprende la necesidad de abordar de manera específica los desafíos vinculados al consumo de energía eléctrica en la acuicultura, centrándonos en los sistemas de aireación. Este planteamiento conduce naturalmente a la identificación puntual de los problemas que serán abordados en la siguiente sección.

1.2. Identificación del problema

El análisis de la eficiencia energética en granjas acuícolas, según el estudio de (Sánchez-Ramos, 2019), señala que la mayor parte del consumo eléctrico se atribuye a los sistemas de aireación, como se observa en la Ilustración 1. Esto coincide con las observaciones antes mencionadas, lo cual se traduce en que el gasto eléctrico representa entre el 20% y el 28% de los costos variables en la producción de tilapia, sin importar el tamaño de la explotación.

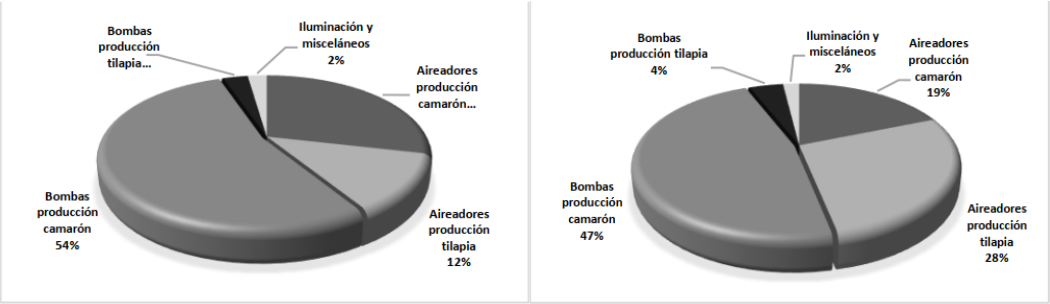

Ilustración 1. Distribución de la demanda y consumo energético. Fuente: (Sánchez- Ramos, 2019)

Al identificar las principales deficiencias en las granjas acuícolas, como se expone en la Ilustración 2, se evidencia que los problemas relacionados con los aireadores, como el mal dimensionamiento y la operación ineficiente, generan disfunciones significativas. Entre estas deficiencias se incluyen la falta de control, el monitoreo inadecuado y el dimensionamiento ineficiente, junto con la necesidad de una capacitación más completa para el personal.

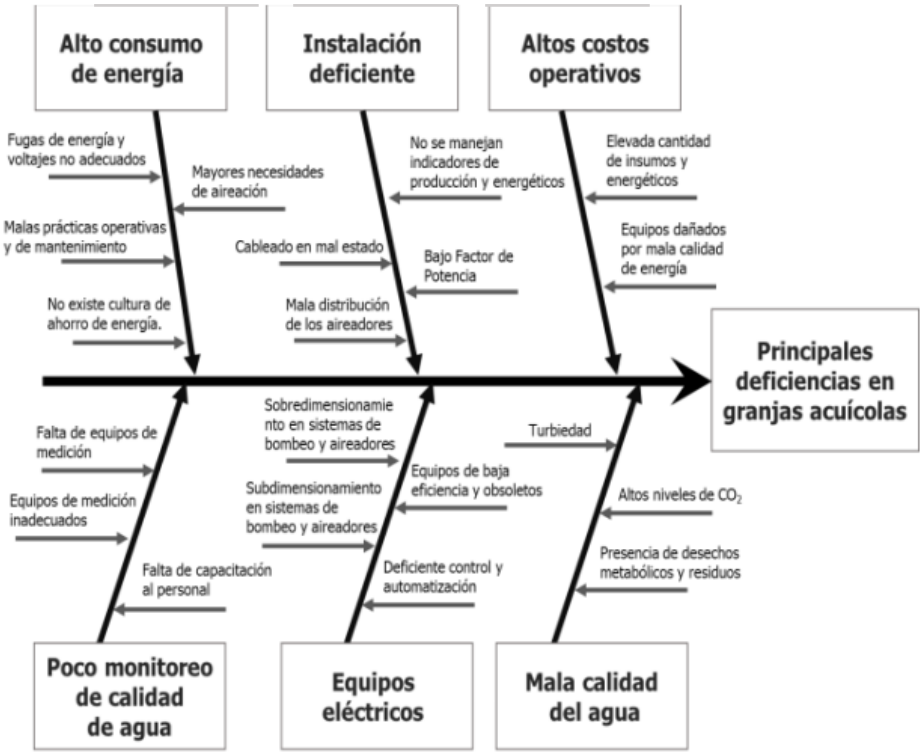

Ilustración 2. Principales deficiencias en granjas acuícolas. Fuente:(Sánchez-Ramos, 2019)

La identificación de deficiencias abre la puerta a la optimización del consumo eléctrico en los sistemas de aireación, que, como se señaló anteriormente, representan un cálculo considerable del 20% al 40% de los costos variables. La problemática se agrava debido a la carencia de control, el monitoreo ineficiente, el mal dimensionamiento y la falta de capacitación del personal. En este contexto, se plantea la oportunidad de aplicar técnicas avanzadas de aprendizaje automático para analizar los patrones de encendido y apagado de los aireadores.

En consecuencia, se propone la implementación de una propuesta de solución que permita mejorar el uso de los aireadores y, por ende, optimizar el consumo eléctrico en las granjas acuícolas sin descuidar la calidad del agua en cuanto al nivel de oxígeno disuelto presente en ella. La investigación se enfocará en responder a estas cuestiones críticas y propondrá medidas específicas para

abordar y resolver estas deficiencias en los sistemas de aireación, contribuyendo así a una gestión más sostenible y eficiente en las operaciones acuícolas.

Ante la imperante necesidad de optimizar la eficiencia energética en las granjas acuícolas, especialmente en los sistemas de aireación, surge la siguiente etapa crucial: plantear preguntas de investigación que guiarán nuestro análisis detallado.

1.2.1. Preguntas de investigación

En el contexto de la acuicultura, donde la eficiencia energética es esencial, nuestras preguntas de investigación buscan abordar los desafíos asociados con el consumo eléctrico en los sistemas de aireación. A través de la aplicación de algoritmos de aprendizaje automático, nuestro objetivo principal es determinar patrones de encendido y apagado de aireadores. Estas preguntas se centran en aspectos clave que van desde la eficacia de los algoritmos hasta la optimización de parámetros para lograr un ahorro significativo de energía.

- ¿Cómo se destacan los algoritmos de aprendizaje automático en la mejora de la eficiencia energética en la acuicultura?
- ¿Cuáles son los algoritmos disponibles y sus aplicaciones específicas en este contexto?
- ¿Cuáles son los parámetros críticos en los estanques de crianza y cómo influyen en la oxigenación?
- ¿Cómo se lleva a cabo la recolección y almacenamiento eficiente de datos procedentes de los estanques?
- ¿Qué algoritmos son efectivos para clasificar información y encontrar combinaciones óptimas de parámetros?
- ¿De qué manera los algoritmos de aprendizaje automático contribuyen a determinar parámetros óptimos con un enfoque específico en el ahorro de energía?

Estas preguntas representan la guía de la presente investigación hacía, las cuales se basan en la búsqueda de soluciones prácticas y aplicables a la necesidad de eficiencia energética en procesos acuícolas. A continuación, se presentan los objetivos correspondientes.

1.3. Objetivos

En respuesta a la problemática identificada, este estudio propone alcanzar los siguientes objetivos:

1.3.1. Objetivo general

Implementar un algoritmo que optimice el uso de aireadores en la crianza de tilapia del Nilo (*Oreochromis niloticus*), mediante el análisis de variables críticas (oxígeno disuelto y temperatura) con el objetivo de asegurar un consumo energético eficiente.

1.3.2. Objetivos específicos:

- • Realizar un mapeo sistemático para evaluar los algoritmos de aprendizaje automático utilizados en el monitoreo y control de sistemas de gestión energética en instalaciones acuícolas. Registrar y monitorear variables clave (oxígeno disuelto y temperatura) para comprender su comportamiento temporal, tanto en ausencia como en presencia de aireadores.
- Implementar un algoritmo que maximice el ahorro de energía y mejore la eficiencia del uso de aireadores en función de los períodos de encendido y apagado, mediante el análisis y procesamiento de los datos recopilados, conservando los niveles óptimos de oxígeno disuelto.

Estos objetivos específicos se enfocan en proporcionar respuestas concretas a la problemática previamente planteada.

1.3.3. Alcances

- Los resultados estimados para este proyecto contemplan la generación de un sistema para el monitoreo de únicamente dos parámetros del agua (OD, temperatura).
- Se analizan y aplican algoritmos de aprendizaje automático para identificar patrones en el comportamiento de dichas variables, los cuales permitan modelar el sistema de control para encendido y apagado de los aireadores.
- La realización de todos los experimentos se llevará a cabo en la granja acuícola "Los Chapingos", localizada en Caño Prieto, municipio de Paso de Ovejas, Veracruz (coordenadas: 19°17'36.3"N 96°22'06.5"W).

1.3.4. Limitaciones

- Debido a los altos costos de los sensores, la medición de otros elementos químicos y otras variables externas no es factible para este proyecto.
- Sólo se enfocará al monitoreo para las condiciones de crianza y engordada de tilapias del Nilo (*Oreochromis niloticus*).
- Se generará una respuesta específica para una granja de un estanque de crianza semi- intensiva con 25 metros de diámetro, 1.2 metro de profundidad en la orilla y 2 en el centro, con capacidad de 750,000 litros de agua. ubicados en la localidad de Palo Gacho, municipio de Paso de Ovejas, Veracruz.
- Se realizará trabajo de medición durante aproximadamente 90 días, tras la obtención de los dispositivos electrónicos necesarios y considerando limitantes tales como las inclemencias del tiempo y los períodos que marca el programa de maestría para la realización del proyecto.
- La propuesta no considera el uso de energías alternativas.

1.4. Justificación

La tarea de promover tecnologías y métodos eficientes y probados, parte de satisfacer una necesidad real que busca impulsar el desarrollo sostenible y la

eficiencia energética en diversos sectores, incluida la acuicultura. Para este fin, se aprovecha la capacidad del algoritmo genético para identificar patrones y tendencias en datos complejos, buscando no solo reducir el consumo eléctrico, sino también fomentar prácticas sostenibles que contribuyan a la preservación del medio ambiente y al cumplimiento de los objetivos de desarrollo sostenible (ODS) y los Programas Nacionales Estratégicos del CONAHCYT (PRONACES).

La decisión de emplear el algoritmo genético en esta investigación se basa en su probada eficacia para resolver problemas de optimización, específicamente en la determinación de patrones óptimos de encendido y apagado en sistemas de aireación acuícola. Esta elección se respalda en la revisión exhaustiva de la literatura, que destaca la capacidad de los algoritmos genéticos para optimizar procesos en diversos campos, incluida la acuicultura.

Además, el algoritmo genético ha demostrado ser una solución robusta y adaptable a las necesidades específicas de la acuicultura, especialmente en la gestión eficiente del consumo energético en los sistemas de aireación de estanques de cría. Su capacidad para manejar múltiples variables y condiciones ambientales cambiantes lo posiciona como una herramienta idónea para encontrar soluciones prácticas y eficientes.

1.4.1. Aporte a la línea de investigación

Este proyecto se inserta en el eje temático de sistemas de automatización, una categoría que, si bien suele asociarse con la robótica, abarca también el diseño y la implementación de algoritmos vinculados a dispositivos de monitoreo y control. En este contexto, se desarrollarán estrategias específicas para optimizar el consumo de energía eléctrica en aireadores, un componente crítico en sistemas acuícolas de cría de tilapias.

La recopilación y análisis de datos serán fundamentales para identificar patrones y visualizarlos, contribuyendo así al avance de esta línea de investigación.

Asimismo, se destacará el uso del algoritmo genético como una herramienta eficaz en la optimización de los patrones de encendido y apagado de los aireadores, lo que permitirá mejorar la eficiencia energética en la acuicultura. Este enfoque promueve tecnologías y métodos eficientes alineados con las directrices del PRONACES-TE, fomentando prácticas más sostenibles y rentables en la industria acuícola.

1.4.2. Alineación con PRONACES

Este proyecto se alinea de manera directa con el PRONACE "Energía y cambio climático", conforme a la temática que lo impulsa. La propuesta busca analizar el sistema de aireación en la acuicultura de tilapias, ofreciendo soluciones de largo plazo para garantizar un uso sostenible de la energía (CONACYT, 2023). La investigación se orienta a determinar la forma más eficiente de oxigenar los estanques, promoviendo el máximo ahorro de energía eléctrica y consolidando su contribución a los objetivos del PRONACE en pro del desarrollo sostenible.

1.5. Hipótesis

La implementación de un algoritmo permitirá optimizar el encendido y apagado de los aireadores utilizados en estanques dedicados a la crianza de tilapias, reduciendo así el consumo de energía eléctrica y conservando el nivel de oxígeno disuelto adecuado para su desarrollo.

Aunque en esta hipótesis se habla de un algoritmo de optimización genérico, a lo largo de la investigación se determinó la utilización de un algoritmo genético, se justificando y evaluando su uso.

1.6. Metodología

Abordar la eficiencia en la acuicultura es crucial, y para lograrlo, se recurre a la metodología CRISP-DM (*Cross Industry Standard Process for Data Mining*) propuesta por Chapman en el año 2000; esta metodología (ilustración 3) brinda una guía detallada para la realización de proyectos de minería de datos mediante el despliegue de un conjunto de acciones especificadas en diversas etapas.

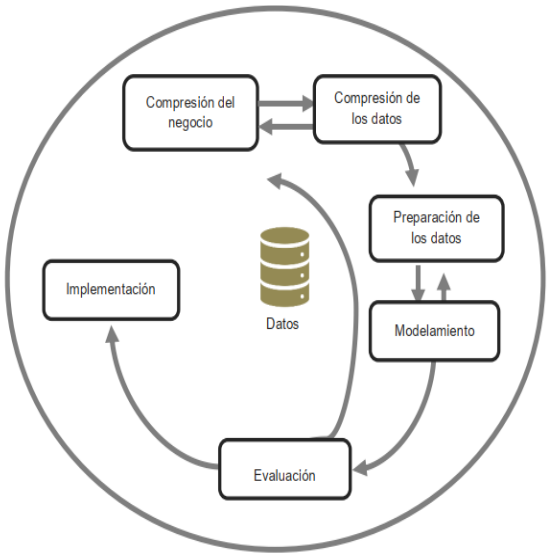

Ilustración 3. Diagrama de la Metodología CRISP-DM. Fuente: (Mancilla-Vela, 2020)

Para abordar este problema se implementará la metodología CRISP-DM (Cross *Industry Standard Process for Data Mining*) como marco de trabajo. Esta metodología, desarrollada por Chapman (2000), proporciona una guía detallada para proyectos de minería de datos.

En el proyecto, se llevarán a cabo las siguientes acciones:

1. Comprensión del Negocio: En esta fase inicial, se busca entender los objetivos y requisitos del proyecto desde una perspectiva empresarial. Luego, se traduce este conocimiento de los datos en la formulación de un problema de minería de datos y un plan preliminar para lograr los objetivos establecidos.
2. Comprensión de los Datos: La comprensión de los datos abarca desde la recopilación inicial hasta actividades que permiten familiarizarse con los datos, identificar problemas de calidad, descubrir conocimientos preliminares y/o encontrar subconjuntos interesantes para formular hipótesis. Se consideran también las fuentes de datos que no se estaban utilizando hasta ese momento, como fuentes externas.

3. Preparación de los Datos: En la fase de preparación de los datos, se llevan a cabo todas las actividades necesarias para construir el conjunto de datos final que será utilizado por las herramientas de modelado. Estas tareas incluyen la selección, limpieza, construcción de nuevas variables, integración y formateo de los datos.
4. Modelado: Durante esta fase, se aplican técnicas de minería de datos a los datos disponibles. Dichas técnicas de modelado se utilizan ajustando los parámetros hasta lograr valores óptimos. Algunas de éstas, pueden requerir especificaciones sobre el formato de los datos, lo que podría implicar regresar a la fase de preparación de los mismos.
5. Evaluación: En esta etapa, se evalúan los modelos construidos previamente para determinar su utilidad en relación con las necesidades empresariales. Los modelos ya construidos, deben cumplir con altos estándares de calidad desde una perspectiva de análisis de datos.
6. Despliegue: La fase de despliegue implica la implementación de los modelos en un entorno de producción. La creación de un modelo no marca necesariamente el final del proyecto, ya que es un proceso dinámico dentro de las decisiones de una organización. Podría ser necesario actualizar o recrear el modelo para incorporar nuevos conocimientos en el futuro.

Con estas acciones, se espera generar un modelo de optimización del uso de aireadores que pueda ser implementado de manera efectiva en entornos de acuicultura, contribuyendo así a la eficiencia energética y a la sostenibilidad en este sector.

En la fase de aplicación de algoritmos, se utilizará un Algoritmo Genético (AG) para optimizar el control de los sistemas de aireación en la acuicultura. Este enfoque de optimización se basa en principios de evolución y genética natural, siendo eficaz para resolver problemas complejos. El flujo del proceso de dicho algoritmo se explica en la Ilustración 4:

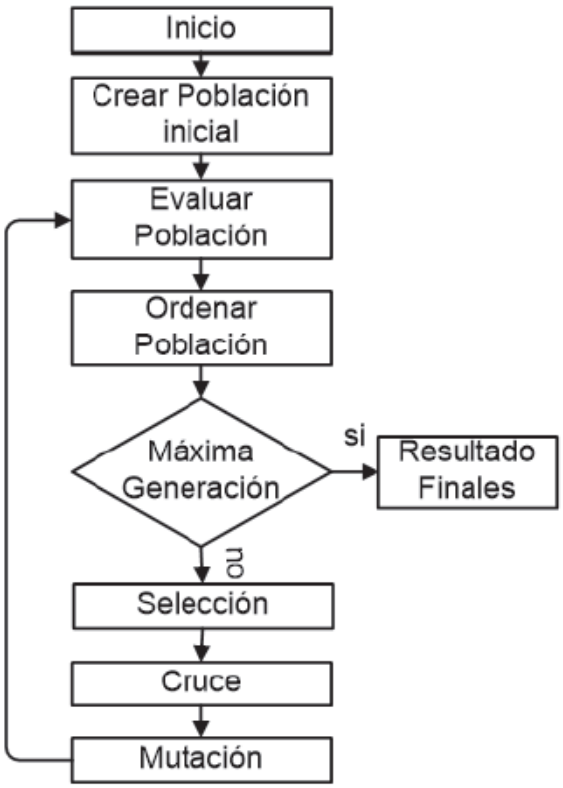

Ilustración 4. Diagrama de flujo del proceso de un algoritmo genético. Fuente:(Muñoz, 2019)

Los procesos mostrados en el diagrama de flujo anterior, se explican a continuación:

1. **Creación de la población:** Generación de soluciones iniciales que representan conjuntos de parámetros para el control de la aireación.
2. **Evaluación de la aptitud:** Evaluación de cada solución según su capacidad para optimizar el consumo eléctrico y eficiencia del sistema.
3. **Selección, cruce y mutación:** Proceso iterativo de selección de soluciones, cruce de información genética, y mutación para crear nuevas soluciones.
4. **Reemplazo de la población:** Sustitución de la población actual por la descendencia generada, repitiendo el proceso a lo largo de varias generaciones.
5. **Convergencia y optimización:** Continuación del proceso hasta alcanzar convergencia o cumplir un criterio predefinido. La solución final representa los parámetros óptimos.

Además de la metodología formal antes descrita, durante la tercera fase propuesta en CRISP-DM, dedicada a la exploración de datos, se realiza el diseño experimental respectivo, fundamentado en el uso de técnicas estadísticas para identificar y cuantificar las causas de un efecto en un estudio experimental. En este enfoque, una o más variables, relacionadas con las causas, se manipulan intencionalmente para medir su efecto en otra variable de interés. Además, el diseño experimental establece pautas sobre qué variables manipular y cómo llevarlo a cabo, todo con un nivel de confianza que permita determinar las relaciones causales (Gabriel et al., 2020).

Finalmente, los antecedentes descritos, junto con los cuestionamientos planteados, dan lugar al marco teórico, donde se explorarán conceptos, teorías y enfoques fundamentales para el análisis de la optimización energética en el caso de estudio del presente proyecto.

Capítulo 2. Marco teórico

En este segundo capítulo se aborda el marco de referencia sobre los dos tópicos principales tratados en esta investigación, la acuicultura y la ciencia de datos.

Con respecto al dominio del negocio, se explican de manera general los conceptos básicos relacionados con la acuicultura, resaltando las características de las variables que intervienen en el sistema de crianza de tilapias y el funcionamiento de los aireadores utilizados para regular los rangos de oxígeno disuelto en el agua.

Por otro lado, se definen los términos relacionados con la recopilación, análisis y visualización de los datos.

2.1. Generalidades de la acuicultura

La acuicultura implica la crianza de organismos acuáticos bajo la supervisión humana, enfocada en optimizar su producción mediante el resguardo, alimentación y protección contra depredadores (Cita). Por lo general, esta práctica se centra en peces, moluscos, crustáceos y plantas acuáticas. El cultivo no solo comprende la propiedad de los organismos, sino también la gestión y las actividades involucradas en la acuicultura, abarcando la planificación, desarrollo de sistemas, instalaciones y prácticas de producción, además del transporte (FAO, 2022).

En aras de una acuicultura sostenible y un rendimiento económico positivo, los productores deben integrar una amplia gama de sistemas y tecnologías, desde equipos de filtración, aireación, desinfección, hasta prácticas de recirculación, bombeo, alimentación y análisis. La implementación de prebióticos y probióticos en el cultivo contribuye significativamente a mejorar la salud y la productividad de los organismos (Proaqua.mx, 2022).

A diferencia de la pesca, la acuicultura permite un control más directo sobre el entorno de los seres vivos, a través de granjas de producción, fusionando

elementos de la pesca y la agricultura. Este enfoque busca equilibrar el ecosistema para garantizar una producción óptima.

La acuicultura abarca diversas ramas, como la piscicultura, la camaronicultura, la ostricultura y el cultivo de almejas, ofreciendo también la posibilidad de criar peces ornamentales, reptiles, algas marinas y plantas de agua dulce (Secretaría de Agricultura y Desarrollo Rural, 2022).

2.1.1. Clasificación de la acuicultura

La acuicultura se clasifica de acuerdo a diferentes criterios (Véase Tabla 4).

Tabla 4. Clasificación de las granjas, según su tipo (Morales, 2022).

Tipos de acuicultura
Densidad de cultivo
Intensiva: sistema que busca una más grande producción en el menor espacio y tiempo viable.
Extensiva: sistema de producción donde la mediación de las personas es mínima y se aprovechan al más alto las condiciones naturales.
Semiextensiva o semiintensiva: sistema en el que el ser humano participa en el aporte de alimento y en la adicción de alevines.
Sector de cultivo
En el océano: en viveros, jaulas o bateas.
En regiones intermareales, como por ejemplo esteros o salineras.
En estanques en tierra.
Período del cultivo
De periodo completo o integral, encierra el desarrollo de todo el periodo fundamental del desarrollo de las especies.

Periodo parcial, comprende el desarrollo de parte del periodo fundamental de las especies. (Morales, 2022).

De acuerdo con la Sociedad Nacional de Pesquería (2022), la granja objeto de estudio es de tipo semi-intensivo, con estanques abastecidos de agua mediante sistemas de bombeo.

2.1.2. Importancia de los procesos acuícolas.

La acuicultura es un sector que ha trascendido a lo largo de algunas décadas. En México el cultivo de peces está en sus fases iniciales y tiene un enorme potencial de desarrollo con amplios beneficios económicos y sociales. (Martínez, 2022).

Además de ser importante en el aporte de alimentos ricos en proteína, la acuicultura tiene una trascendencia social y económica. Su finalidad radica en promover el desarrollo sostenible, evitando la sobreexplotación pesquera y ambiental sobre los recursos acuáticos. De la misma forma, ofrece trabajo alternativo o complementario sobretodo en regiones pesqueras en crisis o rurales con alto grado de marginación, de modo que se genere arraigo en las comunidades de origen, y les permita obtener ingresos y divisas con los bienes de uso y consumo que demandan los países desarrollados (CEDRSSAR, 2015).

En lo concerniente a lo social y económico, la Organización Mundial de la ONU ha dejado de manifiesto el valor de la acuicultura. En este sentido, ha sido incluida en la meta de Desarrollo Sustentable (ODS) 14 de la agenda 2030, Vida Submarina, o conformar parte de la idea de la OMS (OMS) *One health*, la cual “busca conservar y utilizar los océanos, los mares y los recursos marinos de manera sostenible y reglamentar la explotación pesquera (Moran, 2020). Además, está incluida en un plan de apoyo a largo plazo al incremento sustentable de los sectores marino y marítimo, exitosa como Aumento Azul. Fruto de su relevancia, el 5 de diciembre de 2017, la Asamblea general, de las naciones unidas manifestó que el 2022 es El Año Universal de la Pesca y la Acuicultura Artesanales. (Ayala, 2021).

La Comisión Nacional de Acuacultura y Pesca (CONAPESCA, 2023) informó que en México existen 10,000 unidades de producción acuícola, las cuales varían desde pequeñas hasta grandes instalaciones. Según la FAO (2014), en México hay 30,753 productores dedicados a la acuicultura. A nivel mundial, se estima que existen 16,6 millones de productores acuícolas, con China representando 5 millones de estos. En América Latina, el rendimiento promedio por productor es de 7,8 toneladas por año, con una oferta de 9,9 kg por persona al año. Sin embargo, aunque la acuicultura global está en crecimiento exponencial, en México no ha alcanzado el mismo nivel de éxito. Sin embargo, México cuenta con una gran cantidad de especies endémicas con potencial productivo en las que se ha experimentado una rápida domesticación, tales como: el caimán en Tabasco, el pulpo rojo maya en Yucatán (marino), las ranas toro en Michoacán, el tegogolo en Catemaco Veracruz, pargo en Guerrero y Sinaloa (marino), camarones de agua dulce en Veracruz, Morelos y Guerrero, tortugas de tres lomos en Tabasco y Chiapas. (Platas-Rosado, 2017).

2.1.3. Proceso de cultivo de tilapias

Los sistemas de acuicultura para la crianza de tilapias varían considerablemente, abarcando desde cultivos de agua dulce hasta aquellos en ambientes marinos, involucrando prácticas que van desde la cría directa hasta condiciones de cultivo totalmente controladas. Entre los cultivos más comunes se encuentran el plancton, las macroalgas, moluscos y crustáceos (De la Olivia, 2011).

Según la SAGARPA, México fue el noveno productor de tilapia a nivel mundial en 2017, representando el 94.3% de la pesquería nacional. Las entidades productoras principales son Jalisco, Chiapas, Sinaloa, Nayarit, Michoacán, Veracruz, Tabasco, Guerrero, Hidalgo y México. En 2020, la producción nacional de mojarra-tilapia fue de 72,595.96 toneladas con un valor de 2,066.43 millones de pesos. Chiapas lidera la lista de entidades productoras con una producción de casi 31,000 toneladas.

El cultivo de tilapia abarca diversas etapas cómo se ve en la ilustración 5, las cuales dependen de los requisitos fisiológicos de estas especies, y se presentan de manera general en el siguiente diagrama.

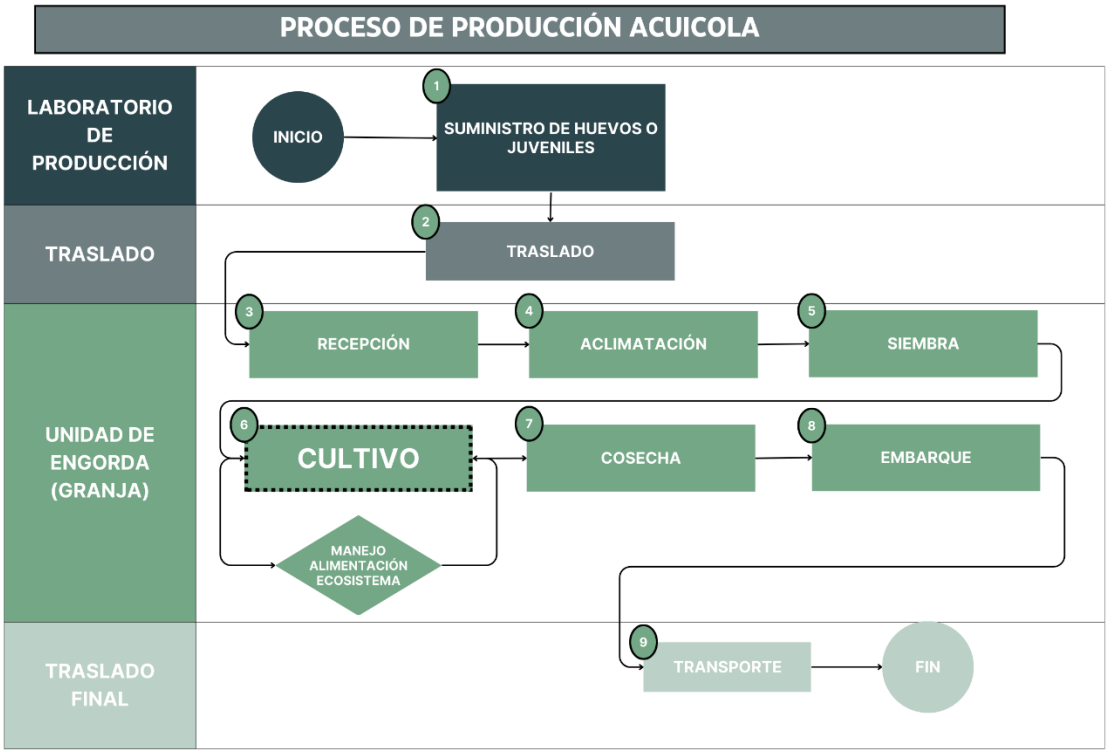

Ilustración 5. Proceso de producción en la acuicultura. Fuente: (Yumpu, 2013)

La acuicultura contemporánea se beneficia de una tecnología avanzada que facilita tanto el mantenimiento rentable como la producción comercial de diversos organismos acuáticos. Esta tecnología permite la gestión de todo el ciclo de vida de las especies en el agua, ya sea parcial o en su totalidad. Implica comprender el comportamiento, los hábitats y las necesidades ambientales, así como la biología reproductiva, los requerimientos nutricionales y la fisiología de las especies a lo largo de las fases de reproducción, preengorde y engorde para el mercado. Gestionar estos aspectos incluye el control del resguardo, la susceptibilidad a enfermedades y la implementación de estrategias que aseguren una producción efectiva y rentable. La acuicultura se ocupa de todos los aspectos de la gestión de

las especies acuáticas, desde la infraestructura necesaria hasta el manejo de todas las etapas de su ciclo de vida, incluyendo el desarrollo de alimentos nutricionalmente adecuados (Luchini, 2010).

Las tilapias del Nilo son peces tropicales que prosperan en aguas someras con temperaturas óptimas entre 31 y 36 °C. Son omnívoras, alimentándose de fitoplancton, plantas acuáticas, invertebrados y otros pequeños organismos. Su reproducción ocurre a una temperatura de 24 °C. La hembra deposita los huevos, los cuales son fertilizados por el macho, tras lo cual la hembra los incuba en su boca hasta que eclosionan, cuidando a los alevines. Estos peces pueden llegar a vivir más de 10 años y alcanzar un peso de 5 kg. (Woynarovich, 2004).

La reversión sexual es fundamental en la producción comercial de tilapias. Se utilizan hormonas para inducir que las hembras se desarrollen como machos. Una vez reversados, se crían en estanques hasta que alcanzan una talla adecuada antes de ser transferidos a instalaciones de engorda (Castillo, 2021).

El proceso de cosecha requiere drenar los estanques y recolectar a las crías, ya que tienden a depredar las crías de los siguientes desoves. La evaluación del sabor es una parte crucial del procesamiento, asegurando su calidad antes de su comercialización (Saavedra, 2022).

En resumen, las tilapias del Nilo son peces tropicales que requieren condiciones específicas para reproducirse, y la reversión sexual es fundamental en su producción comercial. La correcta manipulación de su reproducción, cría y procesamiento asegura la calidad de la tilapia en el mercado.

Comprender en profundidad el proceso de crecimiento de las tilapias y los factores que intervienen en él es esencial para comprender las variables que impactan el desarrollo de estas especies. Es fundamental tener en cuenta que cada etapa del ciclo de vida demanda diferentes niveles en múltiples parámetros, lo que influye directamente en la calidad y el éxito del cultivo.

2.1.4. Elementos del sistema de crianza de tilapias

Las instalaciones dedicadas a la cría de tilapias, conocidas como granjas, se encargan de proporcionar un espacio para resguardar, monitorear y contener los especímenes, facilitando así la crianza.

Estos sistemas se componen esencialmente de un cuerpo de agua, que en este estudio serán estanques, los cuales se oxigenan con la ayuda de algas y sistemas aireadores (Ramírez, 2019; Armijos & Carriel, 2021; Delgado, 2018; Fragoso Cervón & Auró de Ocampo, 2005; Saavedra, 2006).

Estanques y sus estructuras auxiliares:

- Toma de agua
- Canal general
- Canal de distribución
- Criadero
- Almacén

Tipos de estanques:

- Estanque en crecimiento
- Estanque de engorde
- Estanque de productores
- Incubadora
- Canaleta de alevines
- Canaleta de cría
- Estanque de segregación

Los estanques principales, como los de crecimiento, utilizan sistemas de bombeo para suministrar agua, mientras que la oxigenación se logra a través de las algas presentes en el fondo y, durante la noche, mediante aireadores artificiales, mismos que se detallan en el siguiente apartado por ser parte primordial de este estudio.

2.1.5. Sistemas de aireación

La oxigenación del agua es un pilar esencial en la acuicultura, ya que representa la transferencia de oxígeno atmosférico o puro al agua, maximizando la interacción entre el líquido y la atmósfera. Este proceso se convierte en el principal factor determinante para la calidad del agua en estos sistemas (Jiménez, 2012).

El nivel de oxígeno disuelto en los cuerpos de agua es crítico para garantizar su calidad. Mantener una concentración adecuada de oxígeno disuelto no solo previene enfermedades, parásitos y muertes, sino que también mejora la digestión del alimento por parte de los organismos acuáticos.

Suministrar oxígeno adicional para elevar la concentración requerida suele ser costoso y demanda una cantidad considerable de energía, particularmente cuando se recurre a oxígeno líquido o a concentradores de oxígeno. Existen alternativas como los aireadores mecánicos, como agitadores, mezcladores de superficie o sistemas de aireación, aunque estos métodos suelen limitar la máxima capacidad de producción de biomasa en los cultivos (Galli Merino y Miguel Sal, 2007)

a) . **Aireadores de inyección (Ilustración 6).** Se presentan como sistemas altamente efectivos para la oxigenación del agua en entornos acuícolas. Funcionan mediante una bomba de agua que, al pasar por un dispositivo Venturi, crea vacío y aspira aire, generando pequeñas burbujas que se inyectan con presión en la columna de agua. Este mecanismo garantiza una eficiente transferencia de oxígeno y homogeneización del agua, siendo especialmente apropiado para estanques de mayor profundidad (Pinagromx.com, 2023).

Ilustración. 6 Aireador de inyección Fuente: (Pinagromx.com, 2023).

A pesar de su efectividad, los aireadores de inyección pueden presentar limitaciones en determinados entornos. Su uso en estanques poco profundos, con menos de 1.5 metros de profundidad, puede ocasionar daños al fondo, lo que restringe su aplicación en tales escenarios. Asimismo, los modelos flotantes de aireadores de inyección, al generar un caudal considerable, podrían resultar perjudiciales para especies juveniles al crear una fuerza demasiado intensa.

Sin embargo, existen variantes de estos sistemas que presentan mejoras y adaptaciones a estos desafíos. Un ejemplo de ello son las bombas de inyección que succionan y reintroducen agua al estanque, con una boquilla venturi sumergida. Esta adaptación provee múltiples ventajas, como una mayor eficiencia en la oxigenación, reducción de costos operativos, minimización de ruidos y disminución del impacto visual en comparación con los aireadores flotantes convencionales (Bioaquaflco, 2019). Este sistema adaptado resulta en una alternativa más versátil y económica para entornos acuícolas diversos.

b) **Aireadores de paletas (Ilustración 7).** Es uno de los sistemas más utilizados y efectivos en la acuicultura, junto con el golpe de hélice. Su función

primaria es generar oxígeno, romper la estratificación y lograr una distribución eficiente del agua en los estanques. Por lo general, resultan más económicos que otros métodos, pero su mantenimiento puede ser costoso. Se presentan en versiones eléctricas y diésel, con una eficiencia aproximada del 60% en comparación con la electricidad, basada en la experiencia propia. La relación entre el número y tipo de hélices con la potencia del motor es fundamental para determinar su rendimiento (Ramírez, 2019)

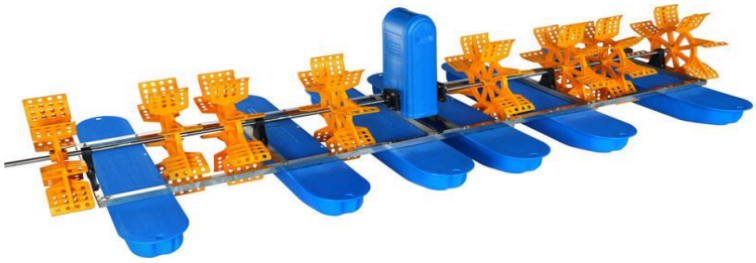

Ilustración 7. Aireador de paletas. Fuente: (TECNOACUA,2020).

El mecanismo del aireador de paletas se compone esencialmente de un motor conectado a un eje, el cual se conecta a paletas plásticas que golpean y remueven el agua. Este mecanismo tiene dos efectos: introduce aire al agua al golpear la paleta y salpica agua al aire. Estos aireadores pueden ser accionados por motores de combustión interna, con potencias entre 35 kilovatios y 75 kW, o por motores eléctricos, siendo estos últimos más económicos en términos de operación (Arias Montero, 2021).

c) **Aireador *Splash (Ilustración 8).*** Este sistema, llamado aireador de bomba vertical, emplea un motor sumergido con una hélice dirigida hacia la superficie y suspendido por un flotador. La rotación rápida de la hélice crea un flujo denso de agua que se proyecta al aire, evitando la estratificación de la columna acuática. Para su fijación en el tanque, se utiliza un sistema de cuerdas y alambres, dividido en tres secciones. Asimismo, presenta rejas en la base del motor para evitar la entrada de peces o camarones, aunque no

es recomendable para la etapa inicial de desarrollo de estas especies (Piñeros-Roldan, 2020).

Ilustración 8. Aireador de tipo fuente Splash. Fuente: (Pinagromx.com, 2023).

Las ventajas del aireador de bomba vertical radican en su sencilla instalación con cuerdas y cables, así como en su consumo eléctrico moderado. Este sistema resulta especialmente eficaz en tanques circulares, generando un movimiento radial en la columna de agua que evita la estratificación.

Sin embargo, el aireador de bomba vertical también presenta desventajas ya que el movimiento de succión puede perjudicar a los peces y camarones jóvenes, y no es apropiado para tanques de gran tamaño. En resumen, este sistema es ideal para tanques circulares con una profundidad superior a un metro, ofreciendo una eficiencia estándar aceptable y siendo una opción robusta que requiere poco mantenimiento.

2.1.6. Parámetros físico-químicos del agua

Los informes en cultivos de tilapia señalan la importancia de monitorear indicadores fisicoquímicos del agua para asegurar un cultivo exitoso. El oxígeno disuelto, la

temperatura, pH, transparencia, dureza, amoníaco y alcalinidad son esenciales para la salud de los peces. Alteraciones en estos parámetros pueden causar deficiencia de oxígeno, pérdida de apetito, vulnerabilidad a enfermedades y, en última instancia, significativas pérdidas económicas (Carrillo, 2014).

Según el Instituto Nacional de Pesca, se establecen valores óptimos y límites para estos parámetros en la crianza de tilapias. Por ejemplo, la reproducción se ve afectada por temperaturas inferiores a 20 °C y resulta letal por debajo de los 11 °C. La salinidad depende de la adaptación de la especie al entorno salino (Instituto Nacional de Pesca, 2018).

Tabla 5. Rangos de los Parámetros Físico-Químicos para la crianza de Tilapia (Instituto Nacional de Pesca, 2018).

Parámetro	Óptimo	Límites
Temperatura	24 ºC-29 ºC	>22 <32 ºC
Oxígeno disuelto	<5 mg/l	>3 mg/l
pH	7.5	>6.5 - <8.5
CO2	<30	<50
Amonio	0.1	<0.1 mg/l
Nitritos	4.6	<5 mg/l
Salinidad	<20*	<20
Turbidez	25	<30

Dentro de estos parámetros, el oxígeno disuelto es crucial para el sistema circulatorio, ya que es un factor limitante para la respiración de los peces y las bacterias nitrificantes, aspecto clave en la nitrificación. Concentraciones de oxígeno por debajo de 2 mg/l limitan esta tasa, requiriendo sistemas de aireación u oxígeno según las necesidades del cultivo (Álvarez & Quevedo, 2005).

Asimismo, el monitoreo de la temperatura del agua es fundamental, ya que puede desencadenar enfermedades o la muerte de los peces si no se controla adecuadamente. La variación entre 22 °C y 36 °C en estanques naturales influye en el crecimiento y requiere atención (García, 2018). En general, el control constante de los parámetros de calidad del agua es esencial para evitar retrasos en el crecimiento de los peces y garantizar un cultivo exitoso (González, 2019).

2.1.7. Sistemas de medición

La monitorización y el control efectivo de los parámetros del agua en entornos de acuicultura son esenciales para garantizar la salud, el crecimiento y la productividad óptimos de las especies. En este sentido, el empleo de sensores especializados ha revolucionado el monitoreo ambiental al permitir mediciones precisas y continuas de variables cruciales para mantener un ambiente acuático estable y favorable para el cultivo de organismos acuáticos. A continuación, se describen los sensores utilizados en el proyecto.

Sensor DS18B20 para Temperatura (Ilustración 9). Reconocido por su alta precisión y versatilidad, destaca por su capacidad para medir temperaturas de hasta 125°C con una exactitud de ±0.5°C y una resolución de 12 bits. Su diseño encapsulado permite su inmersión en líquidos, una característica altamente beneficiosa para entornos acuáticos. Al operar digitalmente, mantiene la integridad de la señal a lo largo de distancias significativas de cableado, y su capacidad de funcionamiento entre 3 y 5V lo hace compatible con diversos sistemas que utilizan microcontroladores, permitiendo la conexión de múltiples sensores en un solo pin a través de su ID exclusivo de 64 bits (Martínez & Bertel, 2021).

Ilustración 9. Sensor DS18B20. Fuente: (Amazon.com, 2023).

Sensor de Oxígeno Disuelto *DfRobot*. Este sensor se perfila como una herramienta integral para la medición del oxígeno disuelto en el agua, siendo valioso en diversos contextos, desde la acuicultura hasta aplicaciones en laboratorios. Utiliza una sonda de tipo galvánica con una membrana permeable al oxígeno que facilita la difusión de este gas en la muestra de agua. La reacción química resultante genera una señal eléctrica que se mide para determinar la concentración de oxígeno en el agua, con un rango de detección de 0 a 20 mg/L y una rápida respuesta en las mediciones (*DFRobots*, 2023). Además, presenta una tecnología robusta que permite una evaluación precisa y oportuna del oxígeno disuelto en el agua, un factor crítico para la salud de los organismos acuáticos, la sonda se presenta en la ilustración 10.

Ilustración 10. Sensor Oxígeno disuelto. Fuente: (Amazon.com, 2023).

Los sensores especializados como el DS18B20 y el sensor de oxígeno disuelto *DfRobot* representan avances significativos en el monitoreo preciso de los parámetros del agua en la acuicultura. Estas herramientas proporcionan datos cruciales para asegurar un ambiente acuático óptimo, permitiendo a los productores tomar decisiones informadas y precisas para el bienestar de las especies acuáticas y la eficiencia en la producción. La implementación de estos sensores ofrece una perspectiva más detallada y continua sobre las condiciones del agua, promoviendo prácticas sostenibles y la mejora constante de los entornos de cultivo acuícola.

La descripción de los objetivos y elementos de la industria acuícola, así como los sistemas de medición utilizados para el monitoreo y ahorro de energía, conlleva al logro de la sostenibilidad en dicha actividad económica. Por lo tanto, además de considerar tópicos relacionados con la tecnología, el siguiente tema trata sobre el concepto de sostenibilidad, cuya importancia radica en la crisis ambiental que se vive tanto a nivel local como mundial.

2.2. Acuicultura y sostenibilidad

El concepto de "sostenibilidad" va más allá de la simple sustentabilidad, destacándose por su enfoque a largo plazo sin agotar recursos esenciales. A diferencia del uso intercambiable con "sustentable", la sostenibilidad, según el informe Brundtland (1987), busca satisfacer las necesidades actuales sin comprometer la capacidad de las futuras generaciones para hacer lo mismo. En el contexto de la acuicultura sostenible, este enfoque implica un equilibrio entre las dimensiones económica, ambiental y social, promoviendo un crecimiento respetuoso con el medio ambiente y socialmente equitativo (Secretaría de Medio Ambiente y Recursos Naturales, 2018).

La sostenibilidad ambiental, fundamental para la acuicultura sostenible, se basa en cuatro principios clave: eficiencia energética mediante fuentes alternativas, manejo consciente del agua, reducción del uso de combustibles fósiles y fomento del reciclaje (Smith, 2015; Taylor, 2020). Estos principios buscan no solo minimizar el impacto ambiental, sino también preservar recursos y favorecer prácticas respetuosas con la naturaleza. En el ámbito empresarial acuícola, adoptar una perspectiva holística del desarrollo sostenible implica considerar todas las fases, desde la producción hasta la atención al cliente, maximizando así beneficios a largo plazo desde una perspectiva ambiental (Kumar, 2019).

La acuicultura sostenible se posiciona como un pilar fundamental para garantizar un futuro habitable y seguro, al centrarse en la sostenibilidad medioambiental, económica y social. Esta práctica busca mejorar las capacidades de desarrollo y promover el uso eficiente de la tierra en el sector acuícola (Jones, 2018). La ciencia y los algoritmos, elementos esenciales en el modelado de datos, se convierten en aliados clave para avanzar hacia una acuicultura verdaderamente sostenible.

En este contexto, la integración de la ciencia de datos emerge como un catalizador fundamental para lograr prácticas más eficientes y respetuosas con el medio ambiente. La aplicación de algoritmos en la fase de "Modelar Datos" permite analizar extensos conjuntos de información, identificar patrones y relaciones ocultas, y optimizar procesos en términos de sostenibilidad económica, social y ambiental. La minería de datos y la aplicación de algoritmos, incluyendo aquellos basados en principios genéticos, no solo mejoran la eficacia de la acuicultura, sino que también contribuyen a la toma de decisiones más informada y ética. De esta manera, la sinergia entre la ciencia de datos y la acuicultura sostenible no solo impulsa la innovación científica, sino que también promueve la armonía entre el desarrollo económico y la preservación de nuestros recursos para las generaciones futuras.

2.2.1. Consumo de energético y eficiencia energética

La evaluación del consumo de energía en la acuicultura se revela como un componente esencial para comprender y mejorar la eficiencia de las operaciones acuícolas. Al profundizar en el análisis del consumo energético, no solo se trata de cuantificar la cantidad de energía utilizada en diversas fases de producción, sino de entender la dinámica temporal de este consumo.

El consumo energético se caracteriza por la cantidad total de energía empleada por un sistema o entidad durante un período determinado. Existe una relación inversa entre el consumo de energía y la eficiencia energética, ya que un incremento en el consumo durante la provisión de servicios conlleva a una reducción en la eficiencia. La medida de electricidad utilizada en un ciclo se refleja en el costo expresado en kilovatios-hora (kWh) en la factura, representando de esta manera el desembolso efectivo por la electricidad consumida (Hernández Vaca, 2022).

La eficiencia energética se relaciona con la utilización más inteligente y efectiva de la energía, con el objetivo de reducir al mínimo tanto el consumo como el impacto ambiental. La consecución de un hogar con estas cualidades se alcanza mediante un diseño cuidadoso y la instalación de equipamiento adecuado, aprovechando al máximo la energía disponible (Martínez Pérez, 2023).

Este enfoque no solo contribuye a optimizar los costos asociados, sino que también abre la puerta a la reducción de la dependencia de fuentes no sostenibles. Al identificar patrones de consumo a lo largo de ciclos diarios y estacionales, se proporciona una base sólida para implementar estrategias específicas que mejoren la eficiencia y reduzcan la huella de carbono de la acuicultura. La comprensión detallada de cómo se distribuye y utiliza la energía en las operaciones diarias no solo beneficia la gestión económica, sino que también respalda la transición hacia prácticas más sostenibles.

2.2.2. Diferencia entre demanda y consumo energético

La distinción del contraste entre la demanda y el consumo energético en la acuicultura constituye un aspecto crucial para una gestión energética eficaz y sostenible. Al abordar la diferencia entre la cantidad real de energía utilizada y la máxima requerida en un momento específico, se obtiene una visión más completa de la eficiencia en el uso de la energía.

La demanda energética se refiere a la cantidad de potencia que un consumidor utiliza en un momento específico, lo cual puede variar a lo largo del tiempo. En términos más simples, se trata de la demanda de una instalación eléctrica expresada como un valor promedio durante un intervalo determinado conocido como intervalo de demanda. La duración de este intervalo depende del tipo de valor de demanda que se busque conocer. En resumen, la demanda de una instalación o sistema se representa como la carga promediada en los terminales de llegada durante un período de tiempo específico (Guadalupe Ayala, 2019).

En esencia, la demanda de energía representa la necesidad primordial de recursos energéticos para mantener el adecuado rendimiento de un sistema, estando su variación vinculada al grado de eficiencia en el proceso de utilización (Castellanos Ramírez, 2021).

La diferencia clave entre demanda y consumo de energía radica en el tiempo y el propósito. La demanda se centra en la máxima necesidad energética en un

momento específico, medida en kilovatios o megavatios, influyendo en la planificación de infraestructuras. En cambio, el consumo representa la cantidad total de energía utilizada durante un período definido, expresada en kilovatios-hora, proporcionando una perspectiva más amplia del uso de energía a lo largo del tiempo. Ambos conceptos son cruciales para la gestión eficiente de recursos energéticos.

Este análisis ofrece *insights* valiosos para mejorar la gestión de la demanda, reducir picos innecesarios y optimizar la infraestructura para adaptarse a fluctuaciones en la demanda energética.

La comprensión detallada de esta diferencia no solo potencia la eficiencia operativa, sino que también contribuye a la reducción de la huella ambiental, proporcionando bases sólidas para estrategias destinadas a minimizar impactos negativos y mejorar la sostenibilidad.

2.2.3. Emisión de gases de co2 por consumo energético

La investigación de las emisiones de gases de efecto invernadero vinculadas al consumo energético en la acuicultura se alza como un pilar fundamental para cuantificar y mitigar el impacto ambiental. Al comprender cómo las prácticas operativas contribuyen a las emisiones de CO2, se pueden identificar áreas específicas para implementar tecnologías más limpias, fuentes de energía renovable y prácticas más sostenibles.

El dióxido de carbono (CO2) es un gas transparente bajo condiciones normales de presión y temperatura, constituido por un átomo de carbono y dos átomos de oxígeno. Se distribuye en la Atmósfera, biósfera, hidrósfera y litósfera, desempeñando un papel crucial en el ciclo vital de la Tierra. A pesar de su importancia natural, el aumento constante de este gas está generando preocupaciones significativas relacionadas con la contaminación y el cambio climático (Sancarrano Estela, 2022).

El Factor de Emisión del Sistema Eléctrico Nacional (SEN) para el año 2022, cifrado en 0.435 tCO_2e / MWh, constituye una medida fundamental en la evaluación de las emisiones de gases de efecto invernadero vinculadas al consumo de electricidad. Este valor representa la cantidad estimada de dióxido de carbono equivalente liberado por cada megavatio-hora de electricidad utilizada (Gobierno de México, 2023).

Al estar respaldado por el Artículo 12 del Reglamento de la Ley de Transición Energética y ajustado a través de observaciones de la SEMARNAT (citar), el factor no solo sirve como un indicador ambiental clave, sino que también se convierte en una herramienta esencial para los establecimientos sujetos a reporte.

Su aplicación en el cálculo y reporte al Registro Nacional de Emisiones, considerando la generación de centrales eléctricas en la red nacional, subraya su relevancia en la medición precisa de las emisiones indirectas asociadas al consumo eléctrico en el panorama energético del año 2022.

Este enfoque permite establecer indicadores claros para el monitoreo continuo de las emisiones, fomentando la transición hacia procesos más respetuosos con el medio ambiente. La cuantificación de las emisiones de gases de efecto invernadero proporciona una base sólida para la implementación de estrategias específicas destinadas a reducir la huella de carbono de la acuicultura y avanzar hacia prácticas más sostenibles.

La acuicultura, como pilar de la seguridad alimentaria y el desarrollo sostenible, ha experimentado una transformación notable gracias a la incorporación de tecnologías innovadoras y sistemas de monitoreo avanzados. Los avances en sensores y dispositivos especializados han redefinido la capacidad de controlar y comprender el entorno acuático, permitiendo a los productores optimizar las condiciones para la crianza de organismos acuáticos. Estos desarrollos no solo han mejorado la eficiencia de la producción, sino que también han permitido un monitoreo continuo y una gestión proactiva de los sistemas acuícolas.

En la sección siguiente, se definen varios de los conceptos que se encuentran inmersos en el procesamiento y análisis de los datos generados por los dispositivos antes mencionados, de manera que su comprensión permita realizar acciones más precisas para optimizar la producción acuícola y garantizar la calidad del entorno acuático.

2. 3. Fundamentos de ciencia de datos

Esta sección resalta el papel crucial de la ciencia de datos en la minería y generación de conocimiento. Explora su evolución, la intersección con la agricultura y su vinculación con la minería de datos, destacando cómo se descubren patrones en conjuntos de datos complejos. Además, se detallan las fases clave de la metodología CRISP-DM, utilizada para optimizar el uso de aireadores en la acuicultura. Este enfoque estructurado no solo crea modelos eficientes, sino que también impulsa la comprensión profunda de los datos, abriendo puertas a una gestión más sostenible de los recursos marinos.

La ciencia de datos, actualmente, es un poderoso instrumento que engloba la minería de datos y la generación de conocimiento. Su objetivo principal es descubrir patrones y comportamientos a partir de datos para facilitar la toma de decisiones y las predicciones. Este campo ha experimentado un crecimiento significativo a medida que se ha ampliado el acceso a grandes volúmenes de información, incluso para el procesamiento en tiempo real, lo que demanda técnicas avanzadas capaces de resolver problemas complejos. Además, reúne a diversos grupos de investigación en campos como la informática, estadística, matemáticas e ingeniería, dedicados a desarrollar nuevos algoritmos, métodos computacionales e infraestructuras para la recolección, almacenamiento y procesamiento de datos (García, 2018).

La ciencia de datos se enfoca en tres áreas fundamentales. En primer lugar, se emplea el *big data* para procesar la información. En segundo término, la minería de datos tiene como objetivo identificar patrones, inclusive aquellos que no se hayan presentado anteriormente. Por último, la visualización de datos tiene como fin

simplificar la comprensión de la información y fomentar su divulgación (Lemus-Delgado, 2020).

El nexo entre la ciencia de datos y la agricultura ha evolucionado significativamente en los últimos años. Los objetivos principales de su uso en la agricultura son reducir el impacto ambiental de las prácticas agrícolas, incrementar la producción y el rendimiento de los cultivos, disminuir los costos de producción y tener un mayor control sobre la producción (Mompó Serrano, 2022).

2.3.1 Minería de datos

La minería de datos es el proceso de descubrir patrones, tendencias y conocimientos significativos en conjuntos de datos grandes y complejos. Se centra en extraer información valiosa a partir de grandes volúmenes de información, permitiendo tomar decisiones informadas. Es un campo multidisciplinario que emplea métodos estadísticos, matemáticos y computacionales para analizar datos y revelar relaciones ocultas. Como señala Michael Berthold, "La minería de datos es un proceso automatizado de descubrimiento de patrones significativos en grandes conjuntos de datos" (Berthold, 2016).

En su esencia, la minería de datos implica un proceso de varias etapas. La fase inicial, encaminada a la comprensión de los datos, involucra la exploración y comprensión de la información disponible para identificar patrones iniciales y problemas de calidad. Según NCR, "empieza con la colección inicial de datos y prosigue con ocupaciones que permiten familiarizarse con los datos, detectar los inconvenientes de calidad de los datos y encontrar los primeros conocimientos sobre los datos" (NCR, 1999). La etapa siguiente, la preparación de los datos, implica limpiar y transformar los datos para su análisis, como señala Rafael Córdova, "suele llevar el 50-70% del tiempo y esfuerzo de un plan" (Córdova, 2018).

2.3.2. Fases genéricas de la ciencia de datos

Las fases genéricas de la ciencia de datos trazan un itinerario esencial en la exploración y extracción de conocimiento a partir de conjuntos de datos. Desde la

crucial "Obtención de Datos", donde se diversifican las fuentes, hasta la fase de "Interpretar los Resultados", cada etapa contribuye a la construcción de un análisis robusto. La obra de Castaño García (2019) subraya la importancia inicial de adquirir datos de manera precisa, mientras que la fase de "Limpieza de Datos" transforma la materia prima en un formato apto para el análisis. La exploración y modelado de datos, respaldados por herramientas como Python y algoritmos de aprendizaje automático, se entrelazan en un proceso que culmina con la crucial "Interpretación de Resultados", delineando nuevas perspectivas y conclusiones valiosas para la toma de decisiones en el mundo real. A continuación, se detallan las fases antes mencionadas:

a) **Obtención de datos.** En la ciencia de datos, el primer paso es la adquisición de datos, ya sea directamente mediante equipos de medición, registros históricos almacenados previamente o mediante datos disponibles en internet. En nuestro estudio, se obtienen datos directamente a través de un sistema de sensores que registra el nivel de oxígeno disuelto y la temperatura, generando una serie de registros para su posterior procesamiento (Castaño García, 2019).

b) **Limpieza de datos.** Fase donde se transforman los datos sin procesar a un formato legible por el software. El formato CSV es comúnmente utilizado en ciencia de datos. Se realiza una organización de datos para que cada columna represente correctamente una variable y cada línea sea una observación. Acciones comunes incluyen la estandarización de formatos, corrección de errores tipográficos y ajustes en formatos numéricos. Estos procedimientos se pueden llevar a cabo mediante lenguajes de programación como Python, utilizando librerías como *Numpy* y *sklearn*.

c) **Explorar datos.** En esta etapa, se analizan los datos disponibles, formulando preguntas e hipótesis para limitar el alcance del proyecto. Se aplican medidas estadísticas y algoritmos en Python, incluyendo métodos de agrupamiento o *clustering* para una primera exploración, también utilizados en fases posteriores de modelado.

d) **Modelar datos.** El software y algoritmos de aprendizaje automático se utilizan para profundizar en la información, prever resultados y sugerir políticas de acción. Métodos como asociaciones, clasificaciones y grupos se aplican al conjunto de datos de capacitación. Se pueden probar modelos con datos de prueba predeterminados para evaluar la precisión de los resultados. Herramientas como R y Python, junto con software especializado como Orange *DataMining* y *Power BI*, facilitan la ejecución de esta labor. El modelado permite obtener salidas esperadas mediante aprendizaje para determinados valores de entrada.

e) **Interpretar los resultados.** Finalmente, tras este proceso, se obtiene nueva información o perspectivas del sistema analizado. La interpretación de resultados es esencial para generar conclusiones sobre posibles ajustes en el proceso o determinar si ha alcanzado su máxima eficiencia. La presentación clara de estos resultados es crucial para la divulgación y comprensión.

Dadas las etapas anteriores, el modelado y la evaluación son cruciales en la minería de datos. Ya que en éstas se emplean diversas técnicas y algoritmos para crear modelos predictivos y descriptivos que permitan evaluar y validar los datos. Según Blokdyk, "Los analistas de datos ejecutan diversos modelos usando las fronteras por defecto y ajustan las fronteras o vuelven a la etapa de preparación de datos para las manipulaciones primordiales por su modelo" (Blokdyk, 2018). La evaluación determina la calidad y validez de los modelos creados antes de implementarlos en la etapa de despliegue, donde los resultados se traducen en acciones prácticas en el mundo real (Galán, 2016). Este modelado se lleva a cabo mediante la implementación de diversos algoritmos de aprendizaje automático (*machine learning)*, de los cuales se detallan los utilizados en este estudio.

2.3.3. Algoritmos de aprendizaje automático

Continuando con la aplicación de algoritmos en el marco de la ciencia de datos, la fase de modelado de datos, constituye un paso clave que se sustenta en técnicas

avanzadas de aprendizaje automático. Los algoritmos de aprendizaje automático desempeñan un papel fundamental al procesar y analizar extensos conjuntos de datos con el propósito de obtener conocimiento y realizar predicciones. Esta etapa representa un salto cualitativo en la exploración de patrones y relaciones ocultas, destacando la capacidad de estos algoritmos para aprender sin requerir programación explícita (Raja et al., 2018).

La definición de los algoritmos de aprendizaje automático, según Raja et al. (2018), subraya su capacidad de mejorar el rendimiento a medida que se les proporciona más información, destacando así su adaptabilidad y capacidad de identificar patrones en los datos. Este proceso de aprendizaje se basa en el concepto de entrenamiento, donde los algoritmos ajustan sus parámetros para minimizar una función de costo, como explican Goodfellow et al. (2016). Este ajuste continuo durante el entrenamiento permite a los algoritmos mejorar su capacidad de hacer predicciones precisas.

Ejemplos de algoritmos de aprendizaje automático ilustran la diversidad de enfoques aplicados en esta fase. La Regresión Lineal modela relaciones entre variables, los Árboles de Decisión toman decisiones basadas en preguntas jerárquicas, y las Máquinas de Vectores de Soporte buscan la máxima separación entre clases (James et al., 2017; Breiman et al., 2017; Cortes et al., 2018). Estos algoritmos, respaldados por herramientas como Python, crean modelos predictivos fundamentales en la ciencia de datos.

Dentro de este panorama, las Redes Neuronales Artificiales (RNA) destacan como una técnica poderosa. Inspiradas en el sistema nervioso biológico, las RNA procesan información mediante capas interconectadas de neuronas. Este conjunto incluye las Redes Neuronales de Propagación hacia Adelante, las Redes Neuronales Convolucionales y las Redes Neuronales Recurrentes (Goodfellow et al., 2016; LeCun et al., 2015). Entre estas, las Redes Neuronales Recurrentes, y en particular las Long Short-Term Memory (LSTM), sobresalen al abordar el desafío del desvanecimiento del gradiente. Su capacidad para mantener y actualizar estados de memoria a largo plazo las convierte en herramientas cruciales para el

procesamiento de secuencias y tareas de lenguaje natural (Hochreiter & Schmidhuber, 1997).

2.3.4. Algoritmos genéticos

En la esfera de la ciencia de datos, los Algoritmos Genéticos (AG) representan una valiosa adición a la fase de modelado de datos, añadiendo una dimensión evolutiva y adaptativa al proceso. Inspirados en los principios de evolución y genética natural, los AG operan de manera autónoma y automática, imitando el proceso de selección natural para encontrar soluciones óptimas a problemas complejos. En este contexto, la inclusión de los Algoritmos Genéticos destaca su capacidad para la creación, evaluación y mejora continua de soluciones mediante la creación de una población inicial, la evaluación de la aptitud de cada solución y la aplicación de operadores genéticos como selección, cruce y mutación (Garduño Juárez, 2018)

La automatización inherente a los Algoritmos Genéticos implica que, una vez configurados inicialmente, estos pueden operar de forma autónoma, iterativa y adaptativa. La creación de nuevas soluciones, la evaluación de su rendimiento y la evolución de la población son procesos que se llevan a cabo sin intervención manual constante. Además, su capacidad para explorar el espacio de soluciones de manera exhaustiva, combinada con la adaptabilidad inherente al proceso evolutivo, hace que los Algoritmos Genéticos sean herramientas poderosas para abordar problemas complejos y dinámicos en la ciencia de datos. La conjunción de estas técnicas avanzadas, incluyendo los AG, amplía las posibilidades y la robustez de los modelos en el proceso de modelado de datos.

En la fase de modelado de datos, los algoritmos de aprendizaje automático, como la Regresión Lineal y las Redes Neuronales Recurrentes, no solo desempeñan un papel crucial en la exploración de patrones, sino que también se convierten en herramientas esenciales para abordar problemas específicos. Este enfoque adquiere mayor relevancia en el contexto actual, donde la sostenibilidad se destaca como un imperativo. Estos algoritmos, al adaptarse y aprender continuamente de los datos, no solo mejoran la comprensión de los sistemas, sino

que también contribuyen activamente a la búsqueda de soluciones eficientes y respetuosas con el medio ambiente.

Particularmente, la inclusión de Algoritmos Genéticos en este panorama refuerza la capacidad adaptativa y evolutiva de los procesos de modelado de datos. Estos algoritmos no solo ofrecen perspectivas valiosas para optimizar sistemas, sino que también se alinean con la búsqueda de prácticas sostenibles. En resumen, la aplicación de algoritmos en ciencia de datos no solo implica la generación de modelos, sino que representa un esfuerzo dirigido a solucionar problemas, contribuyendo así a un futuro más sostenible mediante la integración de enfoques analíticos avanzados y soluciones fundamentadas.

En este capítulo, hemos explorado el complejo entramado de la acuicultura, desde sus fundamentos hasta su intersección con la ciencia de datos y la sostenibilidad. Hemos desglosado las variables críticas en la acuicultura, introduciendo la influencia de los sensores y la importancia de la ciencia de datos en la toma de decisiones. La comprensión de estos elementos fundamentales sienta las bases para el diseño de un enfoque experimental que aborde no solo la eficiencia productiva, sino también la sostenibilidad. Al considerar la aplicación de algoritmos y técnicas avanzadas de aprendizaje automático, como los Algoritmos Genéticos, junto con los principios de desarrollo sostenible, se establece un puente hacia el próximo capítulo, donde exploraremos el diseño experimental que guiará nuestras investigaciones hacia soluciones innovadoras y equilibradas para la acuicultura del futuro.

Capítulo 3. Diseño experimental

El diseño estadístico del experimento es el método de prueba más eficaz para obtener resultados confiables. También conocido como diseño experimental, implica determinar las pruebas necesarias y la manera de llevarlas a cabo para recopilar datos. Estos datos, una vez analizados estadísticamente, proporcionan evidencia objetiva para responder a las preguntas planteadas, esclarecer aspectos inciertos del proceso, resolver problemas o implementar mejoras (Pulido et al., 2012).

En este capítulo se detalla el diseño experimental aplicado en la tercera fase de CRISP. Los experimentos realizados ofrecen una exploración de los datos recopilados, lo que permitirá plantear una alternativa de solución al problema, confirmando la validez de la hipótesis a través de la observación del entorno, el comportamiento de las variables implicadas y, en algunos casos, la manipulación de las mismas.

El proceso de experimentación, basado en las contribuciones de Espinoza Freire (2018), Hernández (2014), Bello Morales (2019) y Barojas Gutiérrez (2020), se desglosa en seis fases secuenciales ilustradas en la Ilustración 11.

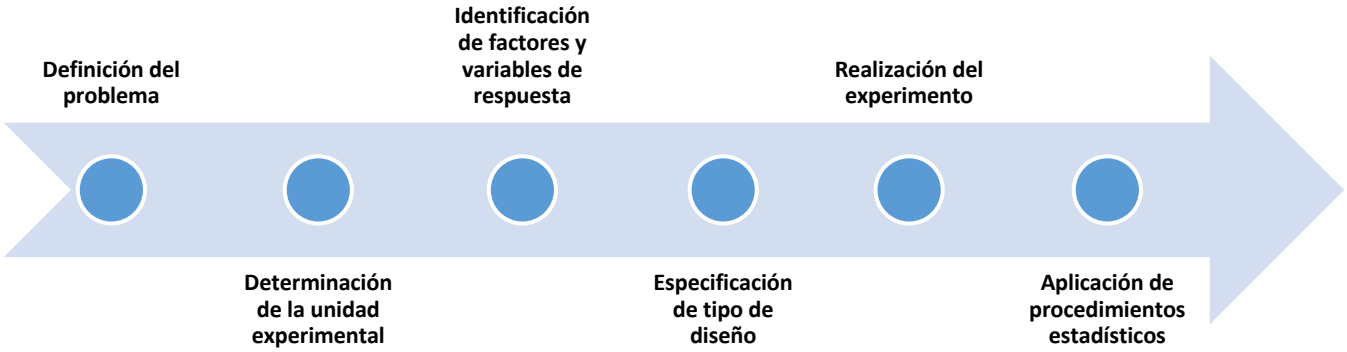

Ilustración 11. Fases secuenciales para la realización de experimentos. Fuente: (Elaboración propia).

Se inicia con la identificación de problemas prácticos, provenientes de la realidad cotidiana, como carencias, vacíos o necesidades de cambio e innovación. Es crucial definir objetivos concretos que aborden estos problemas y planteen soluciones efectivas (Espinoza Freire, 2018).

La primera etapa del proceso investigativo consiste en la identificación de problemas prácticos arraigados en la realidad cotidiana, como carencias, vacíos o la necesidad de cambio e innovación. Este paso inicial es esencial para establecer objetivos concretos que no solo aborden los problemas identificados, sino que también propongan soluciones efectivas (Espinoza Freire, 2018).

Una vez definidos los objetivos, se procede a la selección de factores y niveles. En esta fase, se eligen cuidadosamente los factores de procesamiento y los factores de interferencia (*nuisance*) que se consideran relevantes para el estudio. Los niveles específicos de cada factor se establecen con el objetivo de llevar a cabo pruebas significativas (Hernández, 2014).

La elección de la variable de respuesta es una tarea crucial y se basa en la relevancia de comprender el proceso en estudio. En la mayoría de los casos, la media o la desviación estándar son seleccionadas como variables de respuesta. En situaciones donde la eficiencia de la medición es baja, puede ser necesario realizar mediciones repetidas para garantizar resultados fiables (Bello Morales, 2019).

La siguiente fase implica la selección del diseño experimental, donde el tamaño de la muestra y los objetivos del experimento son considerados cuidadosamente. La identificación precisa de los factores y la estimación de sus efectos son elementos clave para determinar la estructura del experimento (Espinoza Freire, 2018).

La ejecución de los experimentos se lleva a cabo aplicando valores teóricos, utilizando equipos específicos y siguiendo protocolos de calibración. La precisión y el registro meticuloso de los resultados, junto con el control riguroso de cada proceso, son fundamentales durante esta fase experimental.

Finalmente, el análisis de datos se realiza utilizando métodos estadísticos. Estos métodos son esenciales para garantizar la objetividad en la interpretación de resultados y conclusiones, permitiendo una comprensión profunda de la variabilidad y el ruido en los datos obtenidos (Barojas Gutiérrez, 2020).

Cabe recordar que el proceso de realización de experimentos, corresponde a una metodología complementaria, la cual es utilizada para establecer la serie de pasos que permiten explorar los datos adquiridos, no sin antes llevar a cabo de manera detallada las dos fases previas de CRISP.

En los próximos apartados, se implementarán las fases metodológicas determinadas por CRISP, intercalando el diseño experimental correspondiente a la fase de exploración de los datos.

3.1 Definición del problema

Inspirado en el análisis de eficiencia energética realizado por Sánchez-Ramos (2019) en una granja acuícola, donde se enfocó en el consumo, destaca que una parte significativa de la demanda eléctrica corresponde a los sistemas de aireación (Ilustración 12).

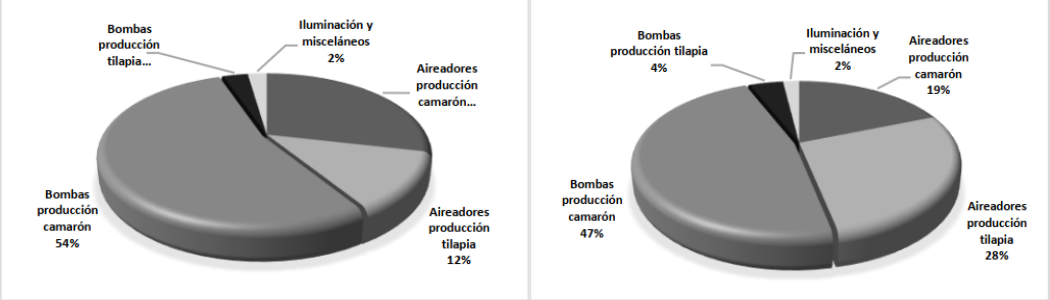

Ilustración 12. Distribución de la demanda y consumo energético. Fuente: (Sánchez- Ramos, 2019).

Además, según Torres-Tadeo et al. (2021), el gasto en energía eléctrica representa entre el 20 y el 28 por ciento de los costos variables en la producción de tilapia, sin importar el tamaño del productor. En el trabajo de Sánchez-Ramos (2019), se identificaron seis problemáticas primordiales que afectan las granjas acuícolas (ver Ilustración 13):

1. Alto consumo energético: El uso extensivo de energía para la aireación conlleva pérdidas por fugas de corriente, variaciones de voltaje y operaciones ineficientes, incrementando el consumo.
2. Instalación deficiente: La distribución de los aireadores es esencial para determinar el cableado y balance de carga en la instalación.
3. Altos costos operativos: Una instalación eléctrica inadecuada puede dañar el equipo, además del gasto en insumos para el mantenimiento y la alimentación de los peces.

4. Monitoreo deficiente de la calidad del agua: La falta de herramientas adecuadas y capacitación del personal afecta el seguimiento de los parámetros del agua.
5. No consideración de las características de los equipos eléctricos: La selección y eficiencia de los equipos utilizados impactan en el consumo y el trabajo ejecutado.
6. Baja calidad del agua: Problemas como turbulencia, altos niveles de dióxido de carbono y presencia de desechos afectan la calidad del agua en los estanques.

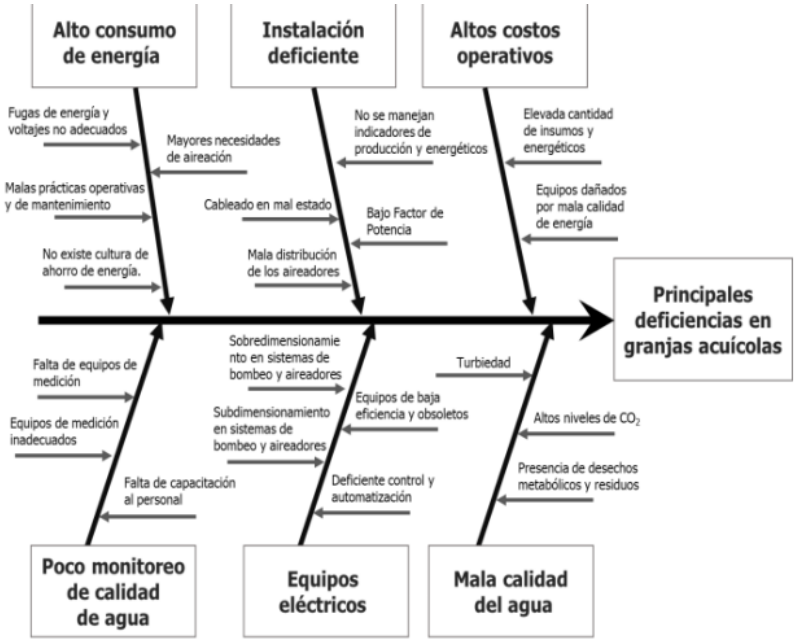

Ilustración 13. Principales deficiencias en granjas acuícolas. Fuente:(Sánchez-Ramos, 2019).

Algunos de los factores considerados como deficiencias en el proceso de crianza dentro de una granja acuícola, los cuales representan entre el 20% y el 40% de los costos variables, son precisamente el mal dimensionamiento y operación de los aireadores, así como la falta de monitoreo de los estanques. Por tal motivo, este estudio propone una oportunidad para optimizar el consumo eléctrico mejorando el

uso de dichos dispositivos. La optimización de este proceso, se enfoca en la búsqueda de un patrón de encendido y apagado.

Dado el peso de los aireadores en la acuicultura, además de examinar tanto su funcionamiento como el consumo generado por sus patrones de encendido, se busca minimizar dicho consumo, la demanda eléctrica y las pérdidas de producción, sin considerar el uso de energías renovables o adquisición de equipo de monitoreo y control. Es importante distinguir entre demanda y consumo: la demanda eléctrica se refiere a la cantidad de potencia requerida en un momento dado, mientras que el consumo eléctrico es la cantidad total de energía utilizada a lo largo de un periodo de tiempo. La implementación de sensores y relevadores en todos los estanques podría ser más efectiva, pero representa una inversión significativa.

En resumen, se busca una solución que mejore la eficiencia en el consumo eléctrico sin costos exorbitantes, siendo ésta la implementación de un algoritmo de aprendizaje automático que optimice el uso de aireadores para la cría de tilapia, reduciendo el consumo de energía eléctrica sin necesidad de grandes inversiones.

A continuación, se describen las condiciones específicas del entorno en el que se realizará el estudio.

3.1.1. Descripción del entorno

La realización de todos los experimentos se llevará a cabo en la granja acuícola “Los Chapingos”, localizada en Caño Prieto, municipio de Paso de Ovejas, Veracruz (coordenadas: 19°17’36.3”N 96°22’06.5”W). La Ilustración 14 presenta una vista aérea extraída directamente de *Google Maps*, ofreciendo una identificación clara de los tanques de la instalación. Es relevante mencionar que la nomenclatura empleada ya es reconocida por todos los usuarios de la granja, siendo esta designación reutilizada para el propósito de este proyecto.

Ilustración 14. Vista aérea de la graja "Los chapingos". Fuente: (Google Maps).

Las instalaciones se componen de 12 estanques de cosecha con forma circular y un diámetro de 10 metros (Ilustración 15). La profundidad de los estanques es de 1.2 metros en el borde y 2 metros en la parte central. Cada estanque está equipado con un sistema de aireación para oxigenar el agua durante la noche. En esta granja, se utilizan tres tipos distintos de aireadores: de paletas, de inyección y del tipo fuente.

Ilustración 15. Estanque 10, granja "Los Chapingos". Fuente: (Elaboración propia).

Este conjunto de estanques y sus respectivos sistemas de aireación se han adaptado para brindar un ambiente propicio a los experimentos (Ilustración 16), siendo esenciales para la correcta realización de las pruebas y observaciones en el entorno acuícola.

Ilustración 16. Estanque 11 de la granja "Los Chapingos". Fuente: (Elaboración propia).

Los estanques, como se observa en las imágenes, están expuestos a las condiciones atmosféricas, lo que implica que cada unidad experimental se ve influenciada por el clima regional, generalmente cálido, con temperaturas que oscilan entre 25 y 40 grados, a causa de su exposición a la luz solar. Además, a pesar de ser una instalación privada, la granja recibe un flujo significativo de personas: desde trabajadores realizando tareas diarias hasta visitas técnicas de la Universidad Autónoma Chapingo. Clientes habituales y estudiantes de posgrado también están presentes, cada uno desempeñando diversas actividades en el lugar. Durante las noches, las instalaciones están iluminadas y vigiladas por un guardia de seguridad. Se debe destacar la presencia de animales silvestres en el entorno, como aves, reptiles y especialmente perros, que se emplean para la vigilancia perimetral.

3.1.2. Especímenes utilizados

Los especímenes empleados en los experimentos son Tilapias del Nilo (*Oreochromis niloticus*) (Ilustración 17), peces comúnmente utilizados en la región para consumo, siendo el segundo espécimen acuático más producido después del camarón. Destacan por su alto contenido proteico de alto valor biológico, similar al del pollo. Por ejemplo, 100 gramos de tilapia aportan 20 gramos de proteína. Sus productos procesados, como filetes y palitos, son bien aceptados en el mercado, presentando mínimas pérdidas al ser procesados como filetes (Valdés Espinoza, 2019).

Características anatómicas como la profundidad del pedúnculo caudal, la ausencia de masa en el lado dorsal del hocico, el número de espinas y radios en las aletas, y la coloración de las aletas durante la época de reproducción, entre otros, definen las peculiaridades físicas de la tilapia (FAO, 2022). Para una representación más visual, se presenta la ilustración 17 de un espécimen real de tilapia

Ilustración 17. Tilapia del Nilo Oreochromis niloticus. Fuente: (Romana-Eguia,2020.

La tilapia, una especie de preferencia tropical, tiende a habitar aguas de poca profundidad. Sus extremos de tolerancia a la temperatura oscilan entre 11-12°C como mínimas y 42°C como máximas, encontrando su rango óptimo entre 31-36°C.

Esta especie omnívora se alimenta de una amplia gama de fuentes: desde fitoplancton, plantas acuáticas y detritos hasta pequeños invertebrados y clases de bacterias asociadas a los detritos. La tilapia del Nilo, con su capacidad de filtración, atrapa partículas en suspensión, incluyendo fitoplancton y bacterias, aprovechando también el pastoreo en la vegetación circundante para obtener nutrientes.

En condiciones de estanque, logran la madurez sexual entre los 5 y 6 meses de edad. El desove se inicia cuando la temperatura alcanza los 24°C. Los machos establecen territorio, cavando nidos en forma de cráter y defendiendo sus espacios. Para ilustrar este ciclo de crecimiento de las tilapias, se incluye la Ilustración 18 (FAO, 2022).

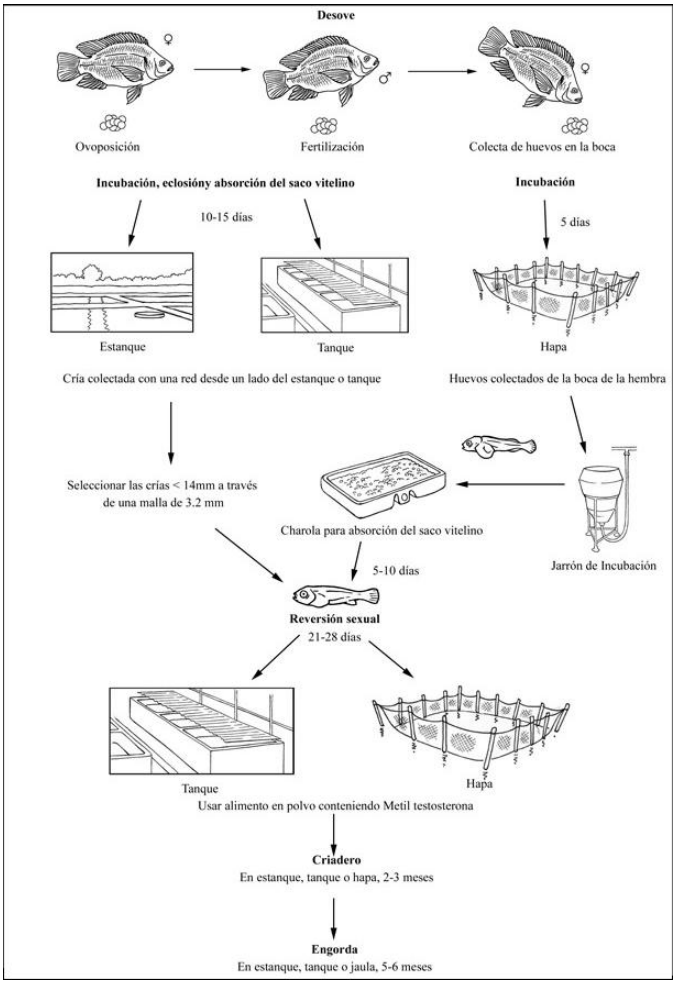

Ilustración 18. Ciclo de crianza de Tilapia. Fuente: (FAO, 2022).

3.1.3. Dispositivos de aireación

Los aireadores empleados en este proyecto, además de ser descritos en el marco teórico de manera general, pertenecen a la marca "*Team Aquaculture*", reconocida por su exportación de estos dispositivos a nivel mundial

Las especificaciones eléctricas de los tipos de aireadores utilizados se presentan en la tabla 6.

Tabla 6. Espeficaciones eléctricas de cada aireado. Fuente: (Elaboración propia)

Aireador	Voltaje (v)	Corriente (a)	Potencia(w)
Paletas	220	0.440	96.8
Fuente	220	1.32	290.4
Inyección	220	2.33	512.6

3.2. Recopilación de datos

Durante las mediciones, se delinearon tres escenarios distintos, considerando la dinámica operativa de la granja acuícola y manteniendo la integridad de los especímenes. Estos escenarios fueron:

- Estanque sin aireación: Se empleó un estanque repleto de agua y algas, pero sin presencia de tilapias para prevenir pérdidas y asegurar el bienestar de los peces. El objetivo de este escenario era observar el comportamiento natural del oxígeno, sin la influencia de elementos consumidores de este componente.
- Estanque con aireación ininterrumpida: En esta situación, se buscó comprender los máximos límites alcanzados con un aireador. Para ello, fue fundamental dejar el sistema en funcionamiento durante toda la noche, permitiendo comprender el nivel de oxígeno que podría lograrse si se

mantuviera el aireador en operación por un extenso período, simulando una jornada prolongada de trabajo.

- Estanque con aireación habitual: Esta fase se centró en medir el oxígeno disuelto durante las jornadas habituales de trabajo. El ciclo de aireación consistió en intervalos de una hora, alternando entre encenderlo y apagarlo, desde las 20:00 horas hasta las 5:00. Este escenario reflejaba el patrón regular de funcionamiento del aireador. En la ilustración 19 se muestra un ejemplo de los aireadores funcionando de noche.

Ilustración 19. Aireadores funcionando durante la noche. Fuente: (Elaboración propia).

3.2.1 Características y calidad de los datos recolectados

Para la medición de las variables, se ubicó el dispositivo a una distancia de 5 metros de los aireadores, asegurando la captura de datos sin afectar ni comprometer los

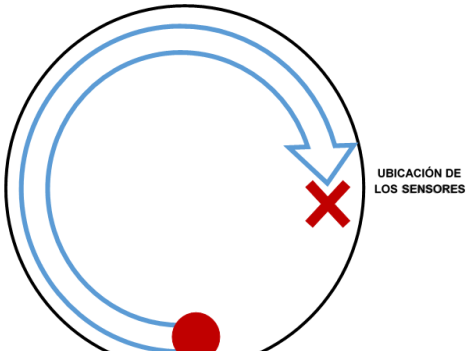

Ilustración 20. Croquis de la ubicación de los dispositivos de aireación y de medición en el estanque. Fuente: (Elaboración propia).

equipos de aireación y medición. Se consideró la dirección de rotación del agua generada por el aireador de paletas (ver Ilustración 20) para determinar el punto más alejado.

Dadas las especificaciones del sensor de oxígeno disuelto, la profundidad de medición se estableció a 30 centímetros desde la superficie del agua. La frecuencia de muestreo es de un registro por segundo, tanto para la temperatura como para el oxígeno disuelto. El conjunto de datos generado por el dispositivo consta de cuatro atributos, como se detalla en la Tabla 7.

Tabla 7. Atributos del dataset inicial. Fuente: (Elaboración propia).

DATASET	DESCRIPCIÓN
Día	Fecha de la medición en formato "DD/MM/AAAA".
Hora	Representa el momento preciso de la medición en formato "HH:MM:SS", permitiendo un análisis detallado del segundo en que se tomó la muestra.
OD	**(Oxígeno Disuelto):** Refleja el valor correspondiente al nivel de oxígeno disuelto. Es importante destacar que, aunque en este caso se presentan valores de cuatro cifras, esto se ajustó para mejorar la precisión de la medición. En el preprocesamiento, se corrige para obtener un valor más coherente con el contexto del trabajo. La unidad de esta magnitud es mg/L.
T	**(Temperatura):** Indica el valor de la temperatura del estanque, expresado en grados Celsius

Durante el proceso de recolección de datos, se observó que en algunos momentos no se registraron datos durante segundos específicos. La razón detrás de estas omisiones es actualmente desconocida, y no se identificó ningún impacto

significativo en la integridad general del conjunto de datos. A pesar de estas breves interrupciones, se implementaron medidas para garantizar la coherencia y la validez del conjunto de datos disponible. Estos lapsos, aunque presentes, no afectan de manera sustancial la interpretación y la calidad general de la información recopilada.

En el Anexo 1 puede visualizarse la consulta de los *datasets* registrados de manera individual y en el Anexo 2 se muestra el *dataset* integrado a través de comandos de *python* que destacan tanto su contenido como las características de sus atributos.

3.3. Exploración de datos

La exploración de los datos recopilados se realiza mediante la ejecución de experimentos, mismos en los que se manipula el tiempo de encendido y apagado de los aireadores, verificando en todo momento su incidencia en los niveles de OD en el agua y el nivel de consumo energético.

Un aspecto fundamental es analizar la disminución de oxígeno, lo cual proporcionará una comprensión detallada de cómo la ausencia de aireación influye en el proceso de oxigenación. Esto permitirá determinar la duración óptima del encendido del aireador para una oxigenación adecuada, sin generar pérdidas de biomasa en las especies en crianza.

Además, se considerará el tipo de aireador, anticipando que la cantidad de oxígeno variará en función del modelo empleado en las mediciones.

3.3.1. Definición del problema del experimento

Una vez recopilados los datos a través de una placa *Arduino* Uno R3 como microcontrolador, un Kit *Gravity,* sensor de oxígeno disuelto analógico/medidor, un sensor DS18B20, para el registro de la temperatura del agua y otros elementos esenciales tales como el módulo de reloj de tiempo real RTC DS3231, el módulo LCD I2C KNACRO I2C TWI 1602, el módulo lector de tarjeta micro SD para *Arduino*

y otros reguladores de voltaje, el problema consistió en monitorear el comportamiento de las variables sensadas a través de dichos dispositivos, OD y T, a través del tiempo.

Este monitoreo se realizó sin el uso de aireadores y posteriormente, con el uso por separado de cada uno de los tres tipos disponibles en la granja.

3.3.2. Determinación de la unidad experimental

El experimento implica dos factores variables (Ilustración 21): el tiempo de encendido y apagado, y es esencial considerar que el tipo de aireador influye en la producción de oxígeno y, por ende, en el consumo energético.

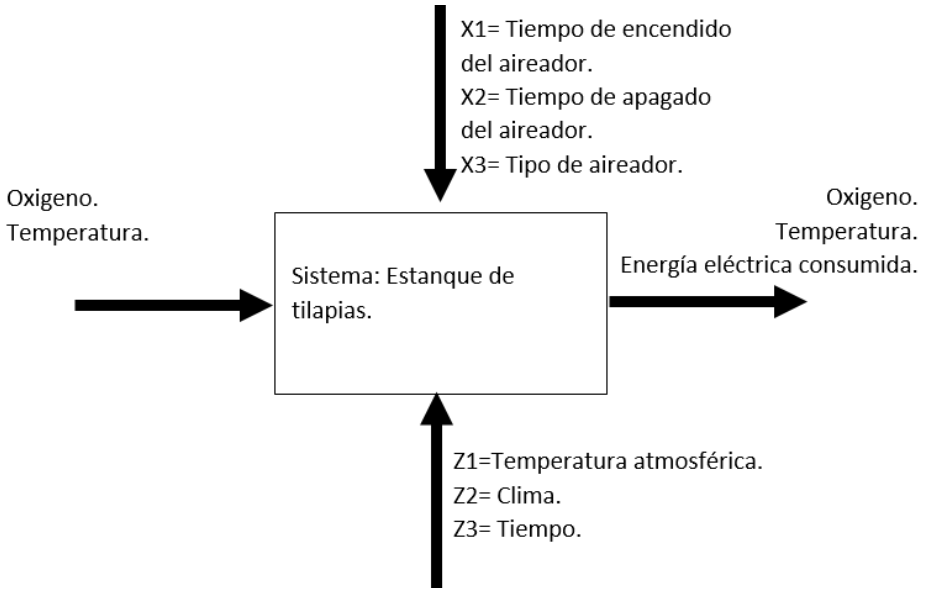

Ilustración 21. Diagrama de diseño experimental. Fuente: (Elaboración propia).

En el diseño factorial, cada réplica del experimento evalúa todas las posibles combinaciones de niveles de factores. Por ejemplo, si el factor A tiene un grado de x y B tiene un grado de y, cada réplica contendrá todas las combinaciones xy de

tratamientos. Cuando se emplean elementos en un diseño factorial, se dice que los factores "intersecan".

El diseño factorial se utiliza para analizar el efecto general de los factores en la respuesta. En un diseño factorial 2k, se consideran dos factores para cada efecto, que pueden definirse como bajo o alto, fuerte o débil, y pueden ser cuantitativos o cualitativos. Este diseño ofrece la menor cantidad de ensayos para investigar los factores k en un diseño completamente factorial, siendo útil en experimentos de cribado o selección de factores (Bello Morales, 2019).

3.3.3. Identificación de factores y variables de respuesta

Como se muestra en la Tabla 8, se presenta un ejemplo de cómo se registran las combinaciones de las variables x1, x2 y x3 en cada uno de los experimentos.

Tabla 8. Matriz de diseño factorial. Fuente: (Elaboración propia).

Estanque	ID	X1 Tiempo de encendido	X2 Tiempo de apagado.	X3 Tipo de aireador.	Oxigeno disuelto	Nivel minimo de oxgieno aceptado	Temperatura	Nivel mínimo de temperatura aceptado	Nivel máximo de temperatura aceptado	Consumo energético
10	EXP_T10_1	15 min	30 minutos	Paletas		<5 mg/l		>22 ºC	<32 ºC	
11	EXP_T11_1	15 min	30 minutos	Inyección		<5 mg/l		>22 ºC	<32 ºC	
12	EXP_T12_1	15 min	30 minutos	Fuente		<5 mg/l		>22 ºC	<32 ºC	

Para la operacionalización de las variables, se presenta en la Tabla 9 los detalles necesarios para este fin.

Tabla 9. Operacionalización de las variables. Fuente:(Elaboración propia).

Variable.	Definición.	Dimensiones	Indicador.
Consumo energético generado por el uso de los aireadores.	El ahorro o eficiencia energética consiste en utilizar la energía de mejor manera. Es decir, con la misma cantidad de energía o con menos, obtener los mismos resultados. Esto se puede lograr a través del cambio de hábitos, del uso tecnologías más eficientes, o una combinación de ambos.	Oxigeno disuelto	Nivel de oxigeno
			Mortandad
			Comportamiento de los especimenes
		Temperatura del agua	Calentamiento del agua
		Uso de airedaores	Tiempo encendido
			Tiempo apagado
			Corrientes de arranque
			Secuencias de encendido
			Voltaje
			Corriente consumida
			Kwh consumidos

En el marco de las Unidades Experimentales ya seleccionadas, se aplicarán tratamientos de manera aleatoria entre las tres. Los hallazgos de la primera ronda de mediciones orientarán las modificaciones en el análisis del sistema.

Las variables de respuesta a evaluar son:

1. Niveles de oxígeno disuelto en el agua, crucial para el desarrollo y supervivencia de las tilapias.
2. Variaciones de temperatura del agua del estanque, considerada como variable adicional para posibles correlaciones.
3. Consumo de energía de cada aireador, derivado del tiempo de funcionamiento.

3.3.4. Realización del experimento

Una vez definidas las variables, con el fin de explorar visualmente los datos, se crearon histogramas, a través de los cuales se pudo observar el comportamiento nocturno del oxígeno disuelto (OD) y la temperatura (T) a lo largo del tiempo. Aunque en primera instancia estos histogramas ofrecieron información sobre el

comportamiento individual de cada variable, se buscó una evaluación más integral que permitiera determinar la idoneidad de los niveles de OD y T para los organismos en cuestión, asignando etiquetas de calidad a partir de los parámetros establecidos por el Instituto Nacional de Pesca en 2018. Estas etiquetas se detallan en las Tablas 10, 11 y 12 proporcionando un marco de referencia esencial para la interpretación de los resultados. La tabla 10 muestra los intervalos límites y óptimos para ambas variables según el instituto antes mencionado.

Tabla 10. Niveles de T y OD establecidos por el Instituto Nacional de Pesca. Fuente:(Elaboración propia).

Parámetro	Óptimo	Límites
Temperatura	24 ºC-29 ºC	>22 <32 ºC
Oxígeno disuelto	<5 mg/l	>3 mg/l

La Tabla 11 muestra los intervalos considerados óptimos para ambos parámetros, oscilando entre los 24°C y 29°C; en el caso del OD, solo es necesario que el oxígeno supere los 5 mg/L. La Tabla 11 presenta 5 etiquetas para el nivel de OD, denominadas como "evaluaciones", donde se asigna un identificador según el nivel de OD, proporcionando datos más significativos sobre la calidad del OD. Es importante destacar que se contempla la posibilidad de fallos en la medición, para lo cual se implementó un sistema de advertencia que informa sobre cualquier error.

Tabla 11. Evaluación del nivel de OD en el estanque. Fuente:(Elaboración propia).

Evaluación	Rango (mg/l)
Óptimo	5<=Oxígeno<=15
Mortal	0<= Oxígeno <3
Tolerable	3<= Oxígeno <5
Error, dato fuera de rango	Oxígeno <0
Sobreoxigenado	15<Oxígeno

Por otro lado, en el caso de la temperatura, se generan 4 etiquetas (ver tabla 12) para calificar el nivel de esta magnitud. Se establecen intervalos donde la temperatura es ideal, así como límites donde puede ser mortal, ya sea porque esta es muy baja o muy elevada. Además, existe un segmento donde la temperatura no es letal, pero tampoco se considera ideal.

Tabla 12. Evaluación del nivel de T en el estanque. Fuente:(Elaboración propia).

Evaluación	Rango (°C)
Baja temperatura	Temperatura<22°
Temperatura ideal	24° < Temperatura < 29°
Alta temperatura	32°< Temperatura
Dentro de los limites ideales	22°= < Temperatura =< 24° 29°= < Temperatura =< 32°

Finalmente, se llevó a cabo una evaluación de la calidad del estanque, tomando como referencia los parámetros de oxígeno disuelto (OD) y temperatura (T). Esta evaluación se plasmó en la Tabla 13, que abarca todas las posibles combinaciones de los estados de estas magnitudes. Este análisis permite determinar si el estanque es adecuado para la crianza de las especies, proporcionando una visión integral de las condiciones ambientales que impactan en el cultivo acuícola.

Tabla 13. Evaluación final de calidad del estanque. Fuente:(Elaboración propia).

Estado del oxígeno	Estado de la temperatura	Estado del estanque
Sobreoxigenado	Alta temperatura	**Exceso de calor**
Sobreoxigenado	Dentro de los limites ideales	**Tolerable**
Sobreoxigenado	Baja temperatura	**Agua fría**
Sobreoxigenado	Temperatura ideal	**Estado ideal**
Óptimo	Temperatura ideal	**Estado ideal**
Óptimo	Dentro de los limites ideales	**Tolerable**
Óptimo	Baja temperatura	**Agua fría**
Óptimo	Alta temperatura	**Exceso de calor**
Tolerable	Temperatura ideal	**Tolerable**
Tolerable	Dentro de los limites ideales	**Tolerable**
Tolerable	Baja temperatura	**Agua fría**
Tolerable	Alta temperatura	**Exceso de calor**
Mortal	Temperatura ideal	**Nivel de OD mortal**
Mortal	Dentro de los limites ideales	**Nivel de OD mortal**
Mortal	Baja temperatura	**Nivel de OD mortal**
Mortal	Alta temperatura	**Nivel de OD mortal**
Error, dato fuera de rango	Temperatura ideal	**Error al registrar OD**
Error, dato fuera de rango	Dentro de los limites ideales	**Error al registrar OD**
Error, dato fuera de rango	Baja temperatura	**Error al registrar OD**
Error, dato fuera de rango	Alta temperatura	**Error al registrar OD**

3.3.5. Aplicación de procedimientos estadísticos

En esta sección, se lleva a cabo la visualización de datos, comenzando con series temporales para analizar el cambio en los niveles de oxígeno en el agua a través de distintos tipos de aireadores.

Las series temporales representan datos estadísticos recolectados a intervalos regulares a lo largo del tiempo, como ventas anuales, contratos trimestrales o valores de PIB (Valencia Granda, 2020).

Para explorar la influencia de la temperatura del agua en los niveles de oxígeno disuelto, se empleará un diagrama de caja, una representación gráfica que revela cambios en conjuntos de datos. A diferencia de los histogramas, los diagramas de caja ofrecen más detalles al permitir la comparación entre varios conjuntos de datos.

El cálculo de la potencia se realizará con la ecuación 1, utilizando datos como el voltaje (U), corriente (I) y factor de potencia (cosφ). Esta fórmula, multiplicada por el tiempo, proporciona el consumo de energía.

$$P_{electrica} = U * I * \cos\varphi \qquad [1]$$

Donde:

$$P_{electrica} = Potencial\ electrica\ consumida.$$

$$U = Voltaje.$$

$$I = Corriente\ nominal.$$

$$\cos\varphi = Factor\ de\ potencia.$$

Estos datos, obtenidos de las placas de los aireadores, contienen información necesaria para dicho cálculo (Ilustración 22)

Ilustración 22. Placa de datos del motor de un aireador de inyección. Fuente: (Elaboración propia).

Además, se analiza la factibilidad de aplicar algoritmos genéticos como una alternativa para abordar la optimización de los niveles de oxígeno en distintos intervalos de tiempo. Según John Holland, uno de los pioneros en el campo de la inteligencia artificial y los algoritmos genéticos, este enfoque imita la teoría de la evolución natural para resolver problemas complejos, entre los que incluimos la optimización de variables en procesos acuícolas.

La aplicación de este algoritmo se realiza utilizando los datos previamente analizados, como el tiempo de encendido y apagado de los aireadores, así como la influencia de la temperatura del agua en los niveles de oxígeno disuelto.

La intención es que, a través de la manipulación de estas variables con el algoritmo genético, se pueda obtener un modelo predictivo optimizado para regular el suministro de oxígeno en el entorno acuícola. Se busca encontrar un equilibrio ideal entre el consumo energético de los aireadores y la garantía de niveles adecuados de oxígeno para el desarrollo y supervivencia de las tilapias. Este enfoque, inspirado en la teoría de la evolución natural de John Holland (Holland, 1975), tiene como objetivo resolver problemas complejos en la optimización de procesos acuícolas y mejorar la eficiencia del suministro de oxígeno en el ecosistema acuático.

Capítulo 4. Resultados

Una vez definido el problema, recopilados y explorados los datos, en este capítulo se procede a determinar cuáles son los hallazgos encontrados, con el fin de proceder con la siguiente fase de CRISP, en donde se realizan tareas de limpieza y transformación de datos, para posteriormente aplicar el algoritmo de modelado correspondiente. Finalmente, se diseñan y ejecutan los casos de prueba requeridos para diversos tipos de requerimientos, con el fin de validar las preguntas de investigación.

La metodología CRISP-DM se utilizó como marco de trabajo para guiar el proceso de investigación y desarrollo de nuestro proyecto. A continuación, se detalla cómo se aplicaron sus fases en el contexto de nuestra investigación: ocio:

En esta fase, se definieron los objetivos del proyecto y se estableció el contexto general de la acuicultura, centrándonos en la optimización del consumo energético en sistemas de aireación. Se identificaron las necesidades del negocio y se definieron las preguntas de investigación que guiarían el proyecto.

1. **Comprensión de los Datos:** Se recopilaron datos relevantes relacionados con el consumo eléctrico en sistemas de aireación, así como información sobre parámetros críticos en estanques de cría de tilapias. Estos datos se analizaron para comprender su estructura, calidad y relevancia para los objetivos del proyecto.

2. **Preparación de los Datos:** Durante esta fase, se realizó un proceso de limpieza y preprocesamiento de los datos para garantizar su calidad y coherencia. Se llevaron a cabo tareas como la eliminación de valores atípicos, la imputación de datos faltantes y la transformación de variables según fuera necesario para su análisis posterior.

3. **Modelado:** En esta etapa, se desarrollaron y evaluaron varios modelos utilizando algoritmos de aprendizaje automático, centrándonos principalmente en algoritmos genéticos para optimizar los patrones de

encendido y apagado de los aireadores. Se ajustaron los modelos según los datos disponibles y se evaluaron su rendimiento utilizando métricas apropiadas.

4. **Evaluación:** Se evaluaron los modelos desarrollados para determinar su eficacia en la optimización del consumo energético y la oxigenación en los estanques de cría. Se compararon los resultados obtenidos con los objetivos del proyecto y se realizaron ajustes en los modelos según fuera necesario.

5. **Despliegue:** Una vez que se seleccionó el modelo final, se implementó en un entorno de prueba para su validación y se preparó para su despliegue en un entorno de producción. Se documentaron los resultados y se proporcionaron recomendaciones para su aplicación práctica en el contexto de la acuicultura.

La metodología CRISP-DM proporcionó un marco estructurado y sistemático para abordar los desafíos asociados con la optimización del consumo energético en sistemas de aireación en la acuicultura, permitiendo así un enfoque coherente y bien definido a lo largo de todo el proyecto.

4.1. Análisis estadístico de las variables

Durante las fases de exploración y análisis de los datos, se han observado patrones interesantes en el comportamiento de los niveles de oxígeno disuelto (OD) a lo largo del día en relación a la luz solar y, por lo tanto, a la temperatura del agua (ilustración 23). Aunque la iluminación alcanza su punto máximo al mediodía, se nota que el OD no sigue esta misma tendencia y, de hecho, alcanza su nivel máximo alrededor de las 4 p.m. Este fenómeno sugiere que hay factores acumulativos en juego, los cuales se están explorando más a fondo para comprender completamente esta dinámica.

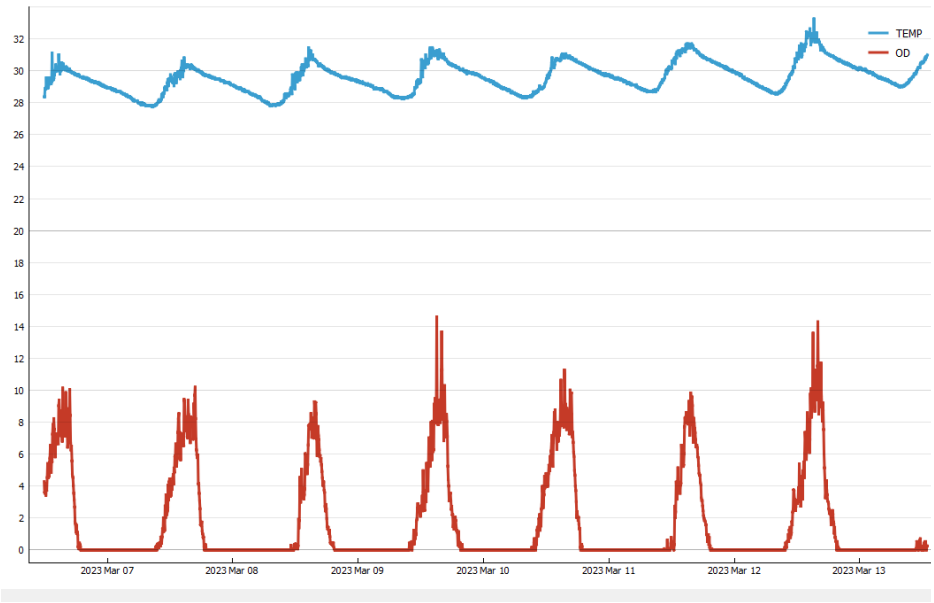

Ilustración 23. Comportamiento de la temperatura (Color azul) y el OD (Color rojo) a lo largo de una semana. Fuente: (Elaboración propia).

Al examinar estanques vacíos, se encontró una situación intrigante, ya que, a pesar de la ausencia de especímenes, los niveles de OD llegaron a cero. Se planteó la hipótesis de que las algas presentes en el estanque podrían estar absorbiendo el OD, lo que añade un elemento adicional a considerar en la gestión de estos sistemas.

Este análisis resulta indispensable, ya que se busca determinar cómo los niveles de OD influyen en la estrategia de encendido de los aireadores durante la noche. Considerar el impacto de factores biológicos, como la presencia de algas que absorben OD, añade un componente adicional a la relación entre los parámetros seleccionados y la respuesta buscada.

En la ilustración 24, se tiene el mismo comportamietno del OD de la figura 23, sin embargo, para este caso se toma en cuenta el nivel de temperatura para clasificar la calidad del estanque, el rojo indica que el estado es mortal, el verde se considera tolerable y el azul se presentó sólo en el último día por una aumento en el nivel de la temperatura.

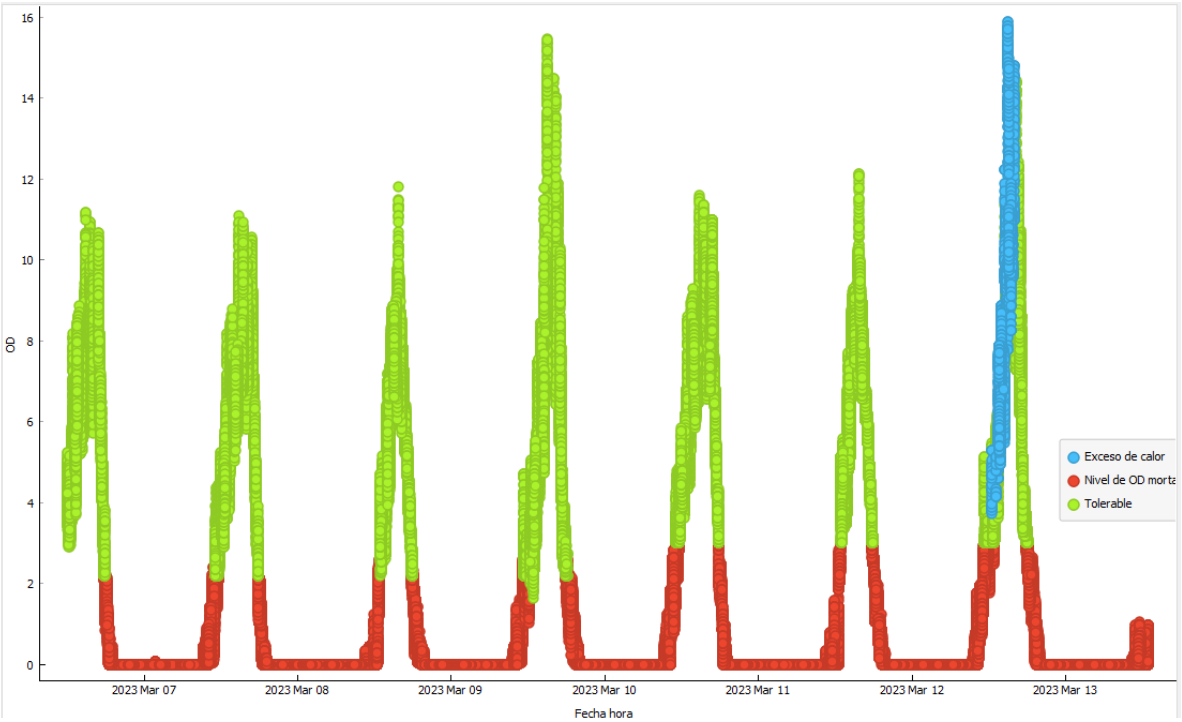

Ilustración 24. Comportamiento del OD a lo largo de una semana, con referencias de color para los niveles de temperatura. Fuente: (Elaboración propia).

Durante las horas nocturnas, se observa una variación mínima de temperatura (T), fluctuando solo uno o dos grados entre las 8 p.m. y las 5 a.m. Simultáneamente, los niveles de OD generalmente experimentaron una reducción de 3 mg/L. Estos datos nocturnos proporcionan una comprensión más completa de las condiciones ambientales en el entorno de estudio. Las ilustraciones 25, 26 y 27 muestran cómo se comporta la temperatura con diferentes aireadores, pese que existe aireación en el estanque, no existe una variación significativa por su presencia, por lo que, para fines de simplificar el estudio, se descarta esta variable para las fases venideras. Además, en el Anexo 2, se presentan los encabezados de los *dataset*.

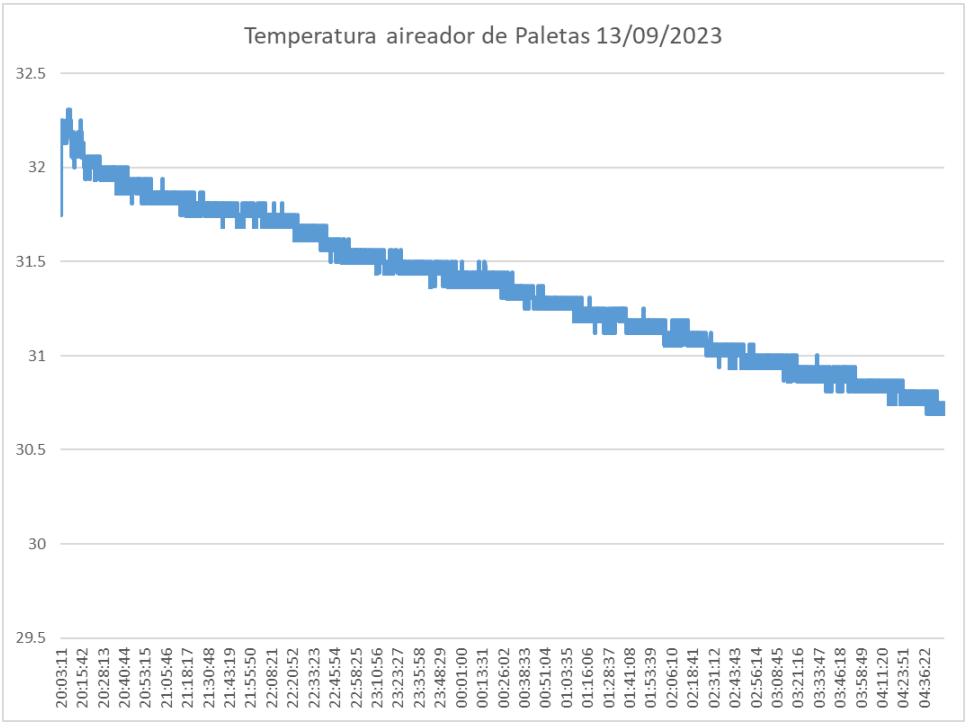

Ilustración 25. Temperatura con aireador de paletas. Fuente: (Elaboración propia).

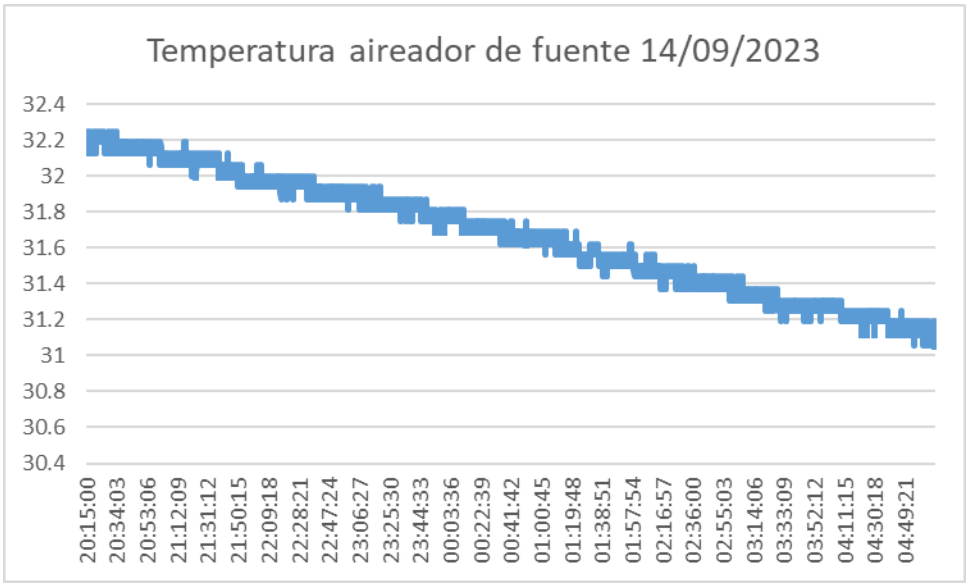

Ilustración 26. Temperatura con aireador de fuente. Fuente: (Elaboración propia).

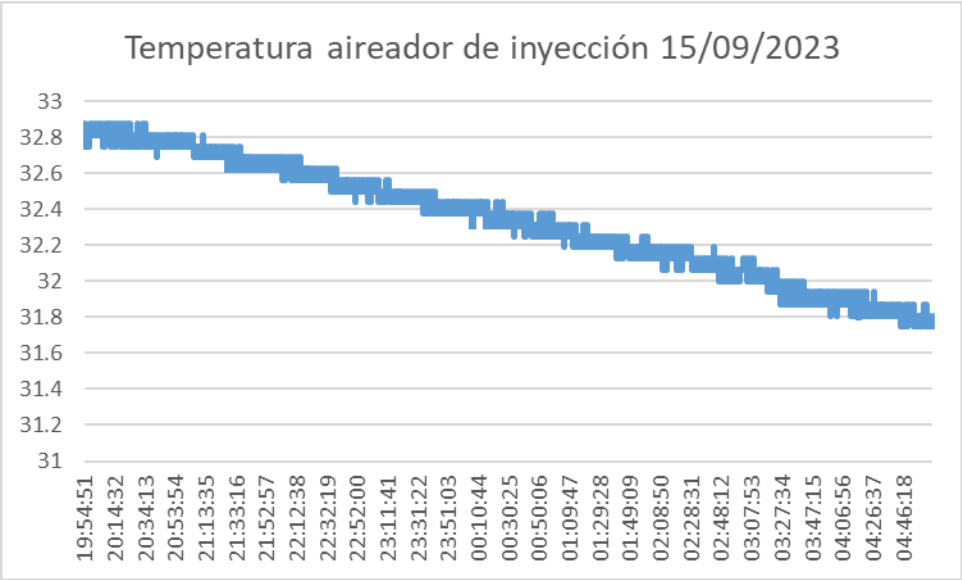

Ilustración 27 Temperatura con aireador de inyección. Fuente: (Elaboración propia).

Para el caso del oxígeno disuelto, se observa que, si bien los aireadores contribuyen a la preservación de los especímenes, en ningún momento logran levantar el oxígeno disuelto, simplemente retrasan su caída, cómo se puede apreciar en las ilustraciones 28, 29 y 30 que corresponde al comportamiento del OD con diferentes aireadores.

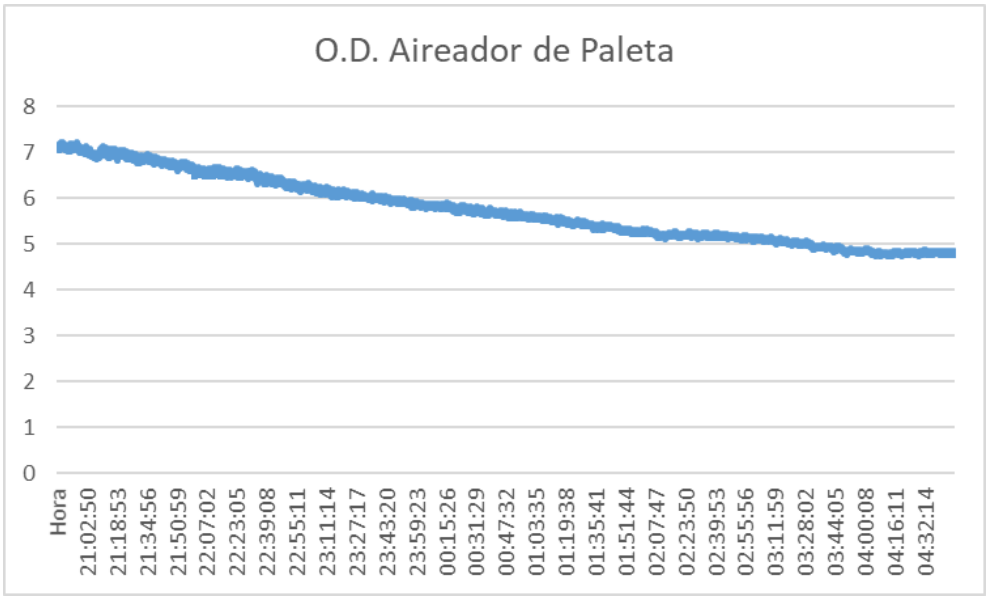

Ilustración 28. Comportamiento del O.D. con aireador de paleta. Fuente: (Elaboración propia).

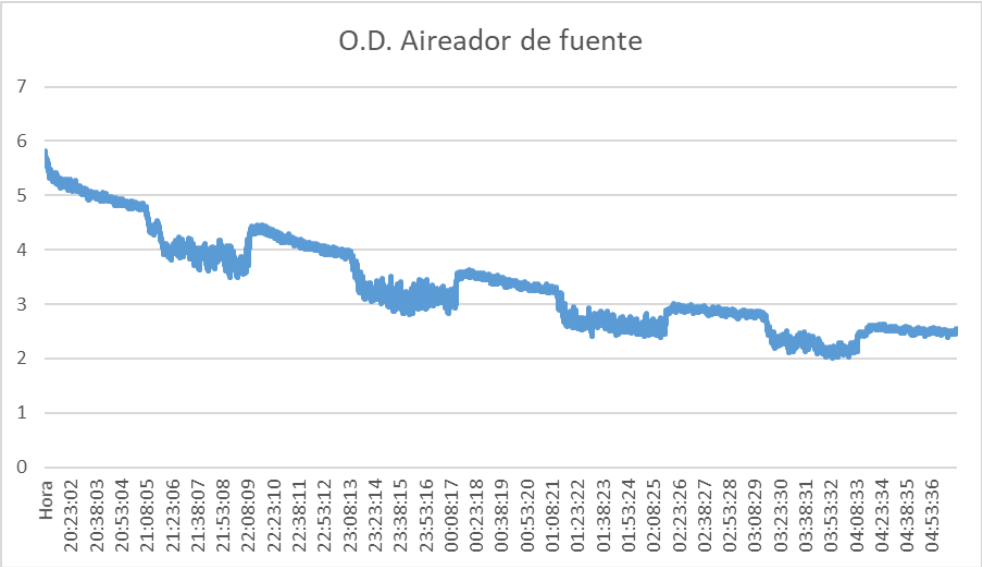

Ilustración 29. Comportamiento del O.D. con aireador de fuente. Fuente: (Elaboración propia).

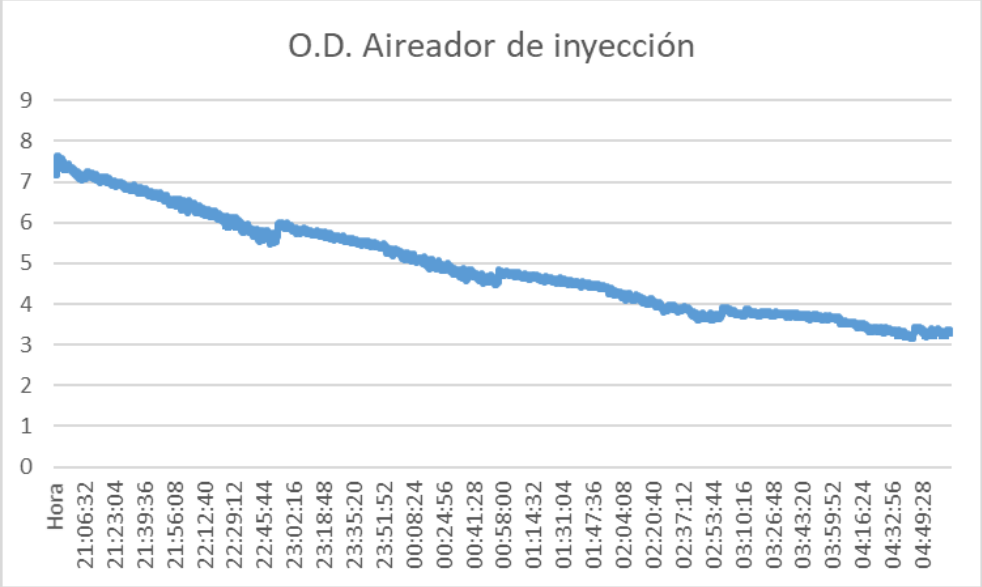

Ilustración 30. Comportamiento del O.D. con aireador de inyección. Fuente: (Elaboración propia).

Con este comportamiento detectado se plantea que para poder resolver esta incógnita será necesario determinar la variabilidad del OD, cuando el aireador está encendido y apagado, para ellos se hará el cálculo de las pendientes en las próximas secciones

4.2. Limpieza y preparación de datos

Teniendo una idea general del comportamiento de los datos, se concluyó que la mejor manera para resolver el problema era mediante la compresión cuantitativa del comportamiento de la variable OD, para ello era necesario determinar su variación, por lo que se necesitó calcular la pendiente del OD, cuando el aireador estaba encendido y cuando estaba apagado.

Sin embargo, era necesario realizar una limpieza y preparación de los datos que consiste:

- Llenar datos faltantes.
- Corregir registros negativos.
- Establecer los horarios de encendido y apagado.
- Generar un solo *dataset* (Anexo 2).

4.2.1. Llenar datos faltantes

El proceso de interpolación temporal y limpieza de datos se realiza para mejorar la integridad y consistencia del conjunto de datos original. Esto implica el ajuste de los valores faltantes en la columna 'OD' utilizando la interpolación lineal y la eliminación de registros con valores nulos.

1. Carga del Conjunto de Datos Original: En la primera etapa, se cargó el conjunto de datos original desde un archivo CSV.

```
# Ruta del dataset original
ruta_csv = "C:\\Users\\USUARIO\\Downloads\\1x1_inyC.csv"

# Cargar el dataset original en un DataFrame
df_original = pd.read_csv(ruta_csv)
```

2. Visualización del Conjunto de Datos Original: Se proporcionó una vista previa del conjunto de datos original para obtener una comprensión inicial.

```
# Mostrar el conjunto de datos original
print("Conjunto de datos original:")
print(df_original.head())
```

```
# Total de registros antes de la interpolación
print("\nTotal de registros antes de la interpolación:", len(df_original))
```

3. Procesamiento de Datos para Interpolación y Limpieza: A continuación, se llevó a cabo el procesamiento necesario para la interpolación temporal y la eliminación de registros nulos.
4. Restauración de Columnas y Reordenamiento: Se restauraron las columnas 'Dia' y 'Hora' como columnas regulares después de la interpolación, y se reordenó el DataFrame.

```
# Crear una copia del DataFrame original para la interpolación
df = df_original.copy()

# Convierte la columna "Hora" a formato datetime
df['fecha'] = pd.to_datetime(df['Dia'] + ' ' + df['Hora'], format='%d/%m/%Y %H:%M:%S')

# Establece la columna "Hora" como índice (importante para la interpolación)
df.set_index('fecha', inplace=True)

# Realiza la interpolación lineal
df = df.resample('S').interpolate(method='linear')

# Restablece la columna "Dia" como una columna regular
df['Dia'] = df.index.strftime('%d/%m/%Y')

# Restablece la columna "Hora" como una columna regular
df['Hora'] = df.index.time

# Reordena las columnas según tu preferencia
df = df[['Dia', 'Hora', 'OD', 'T']]
```

5. Visualización del Conjunto de Datos Después del Procesamiento: Se mostró una vista previa del conjunto de datos después de la interpolación y la eliminación de registros nulos.

```
# Mostrar el conjunto de datos después de la interpolación
print("\nConjunto de datos después de la interpolación:")
print(df.head())
```

```
# Total de registros después de la interpolación
print("\nTotal de registros después de la interpolación:", len(df))
```

6. Guardar el Conjunto de Datos Procesado: Finalmente, el DataFrame resultante se guardó en un nuevo archivo CSV.

```
# Guardar el DataFrame modificado en un nuevo archivo CSV
nombre_archivo_corregido =
"C:\\Users\\USUARIO\\Downloads\\1x1_inyll.csv"
df.to_csv(nombre_archivo_corregido, index=False)
print(f"\nDataFrame interpolado guardado en el archivo CSV:
{nombre_archivo_corregido}")
```

Este proceso asegura que el conjunto de datos se mantenga consistente y sea apto para análisis posteriores, proporcionando una base sólida para la investigación y la toma de decisiones. El código completo de este proceso se encuentra en el anexo 3.

La ilustración 31 muestra un ejemplo de dataset, en que se agregaron segundos faltantes que en el registro.

```
Conjunto de datos original:
          Dia      Hora     OD      T
0  14/09/2023  20:50:01  8.872  32.81
1  14/09/2023  20:50:03  8.872  32.75
2  14/09/2023  20:50:04  8.792  32.75
3  14/09/2023  20:50:06  8.792  32.81
4  14/09/2023  20:50:07  8.872  32.75

Total de registros antes de la interpolación: 18750

Conjunto de datos después de la interpolación:
                           Dia      Hora     OD      T
fecha
2023-09-14 20:50:01  14/09/2023  20:50:01  8.872  32.81
2023-09-14 20:50:02  14/09/2023  20:50:02  8.872  32.78
2023-09-14 20:50:03  14/09/2023  20:50:03  8.872  32.75
2023-09-14 20:50:04  14/09/2023  20:50:04  8.792  32.75
2023-09-14 20:50:05  14/09/2023  20:50:05  8.792  32.78

Total de registros después de la interpolación: 29749
```

Ilustración 31 Modificación del dataset para completar datos faltantes. Fuente: (Elaboración propia).

4.2.2. Eliminación de datos negativos

En esta sección, se describen los pasos realizados para el ajuste y la corrección de datos en el conjunto original. Este proceso fue fundamental para asegurar la coherencia y validez de los datos antes de cualquier análisis posterior.

1. Carga de Datos Originales:El proceso se inició cargando el conjunto de datos original desde la ruta especificada en un DataFrame utilizando la librería pandas.

```
import pandas as pd

# Ruta del dataset original
ruta_csv =

# Cargar el dataset original en un DataFrame
df = pd.read_csv(ruta_csv)
```

2. Visualización del Head del DataFrame Original: Se mostró el encabezado del DataFrame original para proporcionar una visión inicial de los datos antes de cualquier corrección.

```
# Ver el head del DataFrame original
print("Antes de la corrección:")
print(df.head())
```

3. Corrección de Valores Negativos en 'OD': En esta sección, se abordó la corrección de los valores negativos presentes en la columna 'OD', transformándolos a valores positivos mediante el uso de la función `abs` de pandas.

```
# Convertir los valores negativos de 'OD' a positivos
df['OD'] = df['OD'].abs()
```

4. Visualización del Head del DataFrame Después de la Corrección: Se presentó el encabezado del DataFrame después de realizar la corrección, permitiendo una comparación rápida con los datos originales.

```
# Ver el head del DataFrame después de la corrección
print("\nDespués de la corrección:")
```

```
print(df.head())
```

5. Guardar el DataFrame Modificado en un Nuevo Archivo CSV: Finalmente, se guardó el DataFrame modificado en un nuevo archivo CSV, preservando así los cambios realizados durante el proceso de corrección.

```
# Guardar el DataFrame modificado en un nuevo archivo CSV
nombre_archivo_corregido =
df.to_csv(nombre_archivo_corregido, index=False)

print(f"\nDataset corregido guardado en: {nombre_archivo_corregido}")
```

Este proceso fue esencial para la preparación y mejora de los datos, asegurando que estuvieran listos para análisis más avanzados y precisos. El código de este proceso se encuentra en el anexo 4.

4.2.3. Determinación de intervalos encendido y apagado

El algoritmo realiza varias operaciones en un conjunto de datos representado por un DataFrame utilizando la librería panda. A continuación, se explica paso a paso el funcionamiento del algoritmo:

1. Carga de Datos: Se carga el conjunto de datos desde un archivo CSV especificado en la ruta "ruta_csv" utilizando la función `pd.read_csv()` de pandas.

```
import pandas as pd

# Carga los datos desde el archivo CSV
ruta_csv = "ARCHIVO"
df = pd.read_csv(ruta_csv)
```

2. Conversión de la Columna "Hora" a Tipo *DateTime*: La columna "Hora" se combina con la columna "Dia" y se convierte a formato *datetime* utilizando la función `pd.to_datetime ()`.

```
# Convierte la columna "Hora" a tipo datetime
df['fecha'] = pd.to_datetime(df['Dia'] + ' ' + df['Hora'], format='%d/%m/%Y %H:%M:%S')
```

3. Establecimiento de la Columna "Hora" como Índice: La columna "Hora" se establece como índice del *DataFrame* mediante la función `set_index ()`.

```
# Establece la columna "Hora" como índice (importante para la interpolación)
df.set_index('fecha', inplace=True)
```

4. Interpolación Lineal: Se realiza la interpolación lineal utilizando el método `*resample*` de pandas con una frecuencia de muestreo de segundos ('S').

```
# Realiza la interpolación lineal
df = df.resample('S').interpolate(method='linear')
```

5. Restauración de Columnas y Reordenamiento: Se restauran las columnas "Dia" y "Hora" como columnas regulares y se reorganizan las columnas según la preferencia del usuario.

```
# Restablece la columna "Dia" como una columna regular
df['Dia'] = df.index.strftime('%d/%m/%Y')

# Restablece la columna "Hora" como una columna regular
df['Hora'] = df.index.time
```

6. Clasificación del Tiempo en "Encendido" o "Apagado": Se define una función llamada `clasificar_funcionamiento()` que clasifica la hora en "Encendido" si es par y "Apagado" si es impar.

```
# Define una función para clasificar el tiempo en "Encendido" o "Apagado" según el patrón
def clasificar_funcionamiento(hora):
    if hora.hour % 2 == 0:
        return 'Encendido'
    else:
        return 'Apagado'
```

7. La función se aplica a la columna 'Hora' utilizando `apply()` y se crea una nueva columna 'Funcionamiento'.

```
# Aplica la función de clasificación a la columna 'Hora' y crea una nueva columna 'Funcionamiento'
df['Funcionamiento'] = df.index.to_series().apply(clasificar_funcionamiento)
```

8. Impresión y Guardado de Resultados: El DataFrame resultante, ahora con la nueva columna 'Funcionamiento', se imprime en la consola. Se guarda el DataFrame en un nuevo archivo CSV especificado en la ruta "C:\\Users\\USUARIO\\Downloads\\”

```
# Imprime el DataFrame con las nuevas columnas
print(df)

# Guarda los datos en un nuevo archivo CSV
df.to_csv("C ARCHIVO ", index=False)

print("Datos con columna de destino guardados en el archivo CSV.")
```

Este algoritmo proporciona una estructura más completa y clasifica el tiempo en "Encendido" o "Apagado" según un patrón específico, enriqueciendo así la información del conjunto de datos original. La ilustración 32 muestra el resultado del dataset.

fecha	Dia	Hora	OD	T	Funcionamiento
2023-09-14 20:50:01	14/09/2023	20:50:01	7.373333	32.81	Encendido
2023-09-14 20:50:02	14/09/2023	20:50:02	7.373333	32.78	Encendido
2023-09-14 20:50:03	14/09/2023	20:50:03	7.373333	32.75	Encendido
2023-09-14 20:50:04	14/09/2023	20:50:04	7.328889	32.75	Encendido
2023-09-14 20:50:05	14/09/2023	20:50:05	7.328889	32.78	Encendido
...	...	...	...	...	...
2023-09-15 05:05:45	15/09/2023	05:05:45	3.303889	31.78	Apagado
2023-09-15 05:05:46	15/09/2023	05:05:46	3.303889	31.81	Apagado
2023-09-15 05:05:47	15/09/2023	05:05:47	3.303889	31.75	Apagado
2023-09-15 05:05:48	15/09/2023	05:05:48	3.303889	31.75	Apagado
2023-09-15 05:05:49	15/09/2023	05:05:49	3.303889	31.75	Apagado

Ilustración 32. Dataset con la columna representativa del funcionamiento del aireador. Fuente: (Elaboración propia).

El código completo de este proceso se encuentra en el anexo 5.

4.2.4. Combinación de los *dataset*

En primera instancia se intentó trabajar con los *dataset* de manera individual, pero fue una labor complicada de realizar, por lo que se optó por generar un *dataset* que contuviera todos los registros, acorde al tipo de aireador. Para ello se aplicó un algoritmo que permitiera unirlo en uno solo.

El algoritmo tiene como objetivo combinar varios conjuntos de datos CSV ubicados en una carpeta específica. A continuación, se detalla paso a paso el funcionamiento del código:

1. Especificación de la Ruta de la Carpeta de *Datasets*: Se establece la ruta de la carpeta que contiene los *datasets* en la variable "carpeta_datasets". Se debe reemplazar con la ruta correcta.

```
# Ruta de la carpeta que contiene los datasets
carpeta_datasets = CARPETA  # Reemplaza con la ruta correcta
```

2. Creación de una Lista para Almacenar *DataFrames*: Se inicializa una lista llamada "dataframes_pa2" que se utilizará para almacenar los *DataFrames* de los archivos CSV.

```
# Lista para almacenar los DataFrames de los archivos
dataframes_pa2 = []
```

3. Recorrido de los Archivos en la Carpeta: Se utiliza un bucle para recorrer todos los archivos en la carpeta especificada. Si el archivo tiene la extensión ".csv", se procede a cargarlo en un DataFrame y agregarlo a la lista "dataframes_pa2".

```
# Recorre todos los archivos en la carpeta
for archivo in os.listdir(carpeta_datasets):
    if archivo.endswith(".csv"):
        # Carga el archivo CSV en un DataFrame
        ruta_archivo = os.path.join(carpeta_datasets, archivo)
        df = pd.read_csv(ruta_archivo)
        dataframes_pa2.append(df)
```

4. Combinación de DataFrames: Se utiliza la función `pd.concat()` para combinar todos los DataFrames almacenados en la lista "dataframes_pa2" en un solo DataFrame.

```
# Combina todos los DataFrames en uno solo
df_combinado_pa2 = pd.concat(dataframes_pa2, ignore_index=True)
```

5. El parámetro "ignore_index=True" se utiliza para reindexar el DataFrame resultante.

```
# Combina todos los DataFrames en uno solo
df_combinado_pa2 = pd.concat(dataframes_pa2, ignore_index=True)
```

6. Guardado del *DataFrame* Combinado en un Nuevo Archivo CSV: Se especifica un nombre para el nuevo archivo combinado en la variable "nombre_archivo_combinado_pa2".
7. El *DataFrame* resultante se guarda en un nuevo archivo CSV en la ubicación especificada por "nombre_archivo_combinado_pa2".

```
# Guardar el DataFrame combinado en un nuevo archivo CSV
nombre_archivo_combinado_pa2 = NuevoArchivo
df_combinado_pa2.to_csv(nombre_archivo_combinado_pa2, index=False)
```

8. Mensaje de Confirmación: Se imprime en la consola un mensaje indicando que los archivos han sido combinados y guardados en un nuevo archivo CSV.

```
print("Archivos combinados y guardados en un nuevo archivo CSV.")
```

Este algoritmo es útil para combinar eficientemente varios conjuntos de datos almacenados en archivos CSV en una sola entidad, facilitando así su procesamiento y análisis posterior. El código completo de este proceso se encuentra en el anexo 6.

4.2.5. Cálculo de pendientes

El siguiente código se diseñó para analizar los datos de un *dataset* relacionado con el funcionamiento de un aireador. A continuación, se detalla cómo funcionaba el algoritmo:

1. Carga del *Dataset*: Se cargó el conjunto de datos desde el archivo especificado en la variable "archivo" utilizando la librería panda.

```
archivo = "C:\\Users\\USUARIO\\Downloads\\PA2_C.csv"
df = pd.read_csv(archivo)
```

2. Filtrado de datos cuando el aireador está encendido: Se creó un nuevo *DataFrame* llamado "df_encendido" que contenía solo los datos cuando el aireador estaba encendido.

```
df_encendido = df[df['Funcionamiento'] == 'Encendido']
```

3. Cálculo de pendientes diarias cuando el aireador está encendido: Se inicializaron listas para almacenar las pendientes por día y las fechas.
    - Se calculó la pendiente diaria cuando el aireador estaba encendido usando la fórmula de diferencias finitas (gradiente) entre los valores de oxígeno disuelto (OD) y el tiempo.

- Se evitó la división por cero al verificar que el número de puntos de datos fuera mayor que 1.
- Se calculó la pendiente promedio para cada día y se agregaron la fecha y la pendiente al resultado.

```
# Calcular la pendiente por día cuando está encendido
for fecha in df_encendido['Dia'].unique():
    df_dia = df_encendido[df_encendido['Dia'] == fecha]

    tiempo = pd.to_datetime(df_dia['Hora']).dt.second.values
    od = df_dia['OD'].values

    # Evitar divisiones por cero
    if len(tiempo) > 1:
        pendiente_od = np.gradient(od, tiempo)
        pendiente_promedio = pendiente_od.mean()

        # Agregar la fecha y la pendiente al resultado
        fechas_y_pendientes.append((fecha, pendiente_promedio))
```

La fórmula de la pendiente (m) entre dos puntos $(x_1, y_1)\ y\ (x_2, y_2)$en un gráfico cartesiano se calcula mediante la fórmula de la diferencia finita representada por la ecuación 2:

$$m = \frac{y_2 - y_1}{x_2 - x_1} \qquad [2]$$

En el contexto del código proporcionado, la pendiente se calcula con la ecuación 3 para el conjunto de datos de oxígeno disuelto (OD) en función del tiempo. La fórmula específica utilizada en el código es:

$$pendiente\_od = \frac{np.gradient(OD, tiempo)}{tiempo} \qquad [3]$$

donde:

- np.gradient: calcula la diferencia finita usando el método de gradiente de numpy.

- OD representa la columna de oxígeno disuelto en el DataFrame.
- Tiempo representa la columna de tiempo en el DataFrame.

4. Cálculo de la pendiente promedio de todas las pendientes por Día: Se creó una lista con todas las pendientes diarias calculadas. Posteriormente, se calculó la pendiente promedio de todas las pendientes por día utilizando la librería *numpy*.

```
# Calcular la pendiente promedio de todas las pendientes por día
pendientes_por_dia = [pendiente for _, pendiente in fechas_y_pendientes]
pendiente_promedio = np.mean(pendientes_por_dia)
```

La pendiente promedio se calcula mediante la ecuación 4, para cada día, y la fórmula general para el cálculo del promedio se realiza utilizando la librería *numpy*:

$$pendiente\_promedio = \frac{\sum pendiente\_por\_dia}{len(pendiente\_por\_dia)} \quad [4]$$

donde:

- pendiente_por_dia es la lista que contiene todas las pendientes diarias calculadas.

5. Impresión de resultados: Se imprimieron en la consola las fechas y las pendientes diarias. Se imprimió la pendiente promedio de todas las pendientes por día.

```
# Imprimir resultados
for fecha, pendiente in fechas_y_pendientes:
    print(f"{fecha} --- Pendiente: {pendiente:.10f}")

# Imprimir la pendiente promedio
print(f"\nPendiente promedio de todas las pendientes por día:
{pendiente_promedio:.10f}")
```

Este algoritmo permitió analizar las tendencias de cambio en el oxígeno disuelto durante los períodos en los que el aireador estaba encendido, proporcionando una visión más detallada del comportamiento del sistema. Este algoritmo, sensible a las variaciones temporales, permitirá una evaluación detallada de cómo las pendientes se comportan en presencia y ausencia de aireadores, proporcionando información crucial sobre la eficacia de los aireadores en la gestión de los niveles de OD en los estanques de cría de tilapia. La ilustración 33 muestra las pendientes promedio por día y la global. El código completo de este proceso se encuentra en el anexo 7.

```
14/09/2023 --- Pendiente: -0.0001664013
15/09/2023 --- Pendiente: -0.0001435953
01/09/2023 --- Pendiente: -0.0001414909
02/08/2023 --- Pendiente: -0.0001406061
03/07/2023 --- Pendiente: -0.0001713632
04/09/2023 --- Pendiente: -0.0001804912
05/08/2023 --- Pendiente: -0.0002069151
06/07/2023 --- Pendiente: -0.0001471131
07/09/2023 --- Pendiente: -0.0001938475
08/08/2023 --- Pendiente: -0.0001791408
09/07/2023 --- Pendiente: -0.0001005651
10/09/2023 --- Pendiente: -0.0001082465
11/08/2023 --- Pendiente: -0.0000945008
12/07/2023 --- Pendiente: -0.0001321001
14/08/2023 --- Pendiente: -0.0001707742
15/07/2023 --- Pendiente: -0.0000706618
17/08/2023 --- Pendiente: -0.0002352018
18/06/2023 --- Pendiente: -0.0001405270
18/07/2023 --- Pendiente: -0.0002034490
20/08/2023 --- Pendiente: -0.0001763638
21/06/2023 --- Pendiente: -0.0001700545
21/07/2023 --- Pendiente: -0.0002216682
23/08/2023 --- Pendiente: -0.0002079167
24/06/2023 --- Pendiente: -0.0001782560
24/07/2023 --- Pendiente: -0.0001267460
26/08/2023 --- Pendiente: -0.0001937926
27/06/2023 --- Pendiente: -0.0002007938
27/07/2023 --- Pendiente: -0.0001627401
29/08/2023 --- Pendiente: -0.0002183566
30/06/2023 --- Pendiente: -0.0001482150
30/07/2023 --- Pendiente: -0.0002025898

Pendiente promedio de todas las pendientes por día: -0.0001656285
```

Ilustración 33. Pendientes promedio por día y global. Fuente: (Elaboración propia).

4.3 Aplicación del algoritmo

Partiendo de la determinación de las pendientes diarias en el comportamiento del oxígeno disuelto (OD) cuando el aireador estaba encendido, se dio paso a la implementación de un algoritmo genético con el objetivo de optimizar dicho comportamiento. La información recopilada sobre las pendientes proporcionó la base para entender la dinámica del sistema, permitiendo la identificación de patrones y áreas de mejora. Este conocimiento previo fue esencial para la construcción del algoritmo genético, ya que sirvió como fundamento para establecer los parámetros críticos y las metas específicas.

La tabla 14 muestra las pendientes promedio obtenidas por segundo según el tipo de aireador. Sin embargo, el objetivo inicial propuesto por el propietario de la granja contemplaba jornada de encendido y apagado de media hora, por lo que las pendientes se calcularon por 1,800 para obtener las pendientes para 30 minutos, por lo que, en la tabla, se pueden ver 2 pendientes, la correspondiente a 1 según y la de 30 minutos,

Tabla 14. Pendientes de OD en aireadores. Elaboración propia.

Aireador de paletas		
Pendiente encendido	-0.0000849386	-0.15288948
Pendiente apagado	-0.0000902840	-0.1625112
Aireador de fuente		
Pendiente encendido	-0.0000059237	-0.01066266
Pendiente apagado	-0.0002089037	-0.37602666
Aireador de inyección		
Pendiente encendido	-0.0001182361	-0.21282498
Pendiente apagado	-0.0001656285	-0.2981313

4.3.1. Descripción y configuración del algoritmo

1. Configuración Inicial: Importación de librerías necesarias: `*random*`, `*numpy*`, `*matplotlib.pyplot*` y `pandas`. Definición de parámetros del algoritmo genético y del comportamiento del sistema acuático.

```
import random
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Parámetros del algoritmo genético
tamaño_población = 100
tasa_mutación = 0.1
generaciones = 262144

# Pendientes y valor inicial
pendiente_encendido = -0.15288948
pendiente_apagado = -0.1625112
valor_inicial = 8

# Duración del ciclo en horas (9 horas)
duracion_ciclo_horas = 9

# Objetivo de nivel de OD
objetivo_od = 4
```

2. Función de aptitud: La función `aptitud` evalúa la calidad de un cromosoma en términos de tiempo de cumplimiento del objetivo y eficiencia en el uso del sistema. Se aplican restricciones para mantener el nivel de OD y limitar los tiempos de encendido y apagado.

```
def aptitud(cromosoma):
    od = valor_inicial
    tiempo_encendido = 0
    tiempo_apagado = 0
    tiempo_encendido_total = 0
    tiempo_apagado_total = 0
    tiempo_cumplimiento = 0  # Tiempo para alcanzar el objetivo

    for i in range(len(cromosoma)):
        if cromosoma[i] == 1:
            tiempo_encendido += 1
            tiempo_encendido_total += 1
            tiempo_apagado = 0
            od += pendiente_encendido
        else:
            tiempo_apagado += 1
```

```
            tiempo_apagado_total += 1
            tiempo_encendido = 0
            od += pendiente_apagado

        if od < objetivo_od:
            tiempo_cumplimiento += 1

        if od < 4 or tiempo_encendido > 7 or tiempo_apagado > 7:  # Restricciones
            return 0

    # Calcula la aptitud para minimizar el tiempo de cumplimiento y la cantidad de encendidos
    aptitud = 1 / (tiempo_cumplimiento + 1) * (1 / (tiempo_encendido_total + 1))

    return aptitud
```

3. Inicialización de la población: Se genera una población inicial de cromosomas representando el encendido o apagado del sistema en intervalos de 30 minutos.

```
# Inicialización de la población (para intervalos de 30 minutos, cambiamos el tamaño del cromosoma a 18)
tamaño_cromosoma = duracion_ciclo_horas * 2  # Cada hora se divide en 2 intervalos de 30 minutos
población = [np.random.randint(2, size=tamaño_cromosoma) for _ in range(tamaño_población)]
```

4. Algoritmo genético: Se ejecutan las generaciones del algoritmo genético. Se evalúa la aptitud de cada cromosoma, seleccionan los mejores y se generan nuevos cromosomas a través de cruza y mutación.

```
# Algoritmo genético
mejores_cromosomas = []  # Almacenar los mejores cromosomas por generación

for generación in range(generaciones):
    aptitudes = [aptitud(cromosoma) for cromosoma in población]
    mejor_aptitud = max(aptitudes)
    mejor_índice = np.argmax(aptitudes)
    mejor_cromosoma = población[mejor_índice]
    mejores_cromosomas.append(mejor_cromosoma)

    nueva_población = []
    while len(nueva_población) < tamaño_población:
```

```
            padre1, padre2 = random.choices(población, k=2)
            punto_cruza = random.randint(1, len(padre1) - 1)
            hijo = np.concatenate((padre1[:punto_cruza],
    padre2[punto_cruza:]))

            if random.random() < tasa_mutación:
                punto_mutación = random.randint(0, len(hijo) - 1)
                hijo[punto_mutación] = 1 - hijo[punto_mutación]

            nueva_población.append(hijo)

        población = nueva_población
```

5. Filtrado de resultados: Se filtran los mejores cromosomas según el tiempo de encendido. Se muestran los resultados en un DataFrame.

```
# Obtener solo cromosomas con tiempos de encendido menores a 5
horas (300 minutos)
mejores_cromosomas_filtrados = [cromosoma for cromosoma in
mejores_cromosomas if sum(cromosoma) < 360]
```

6. Simulación y visualización: Se simula y visualiza el comportamiento del nivel de OD para los mejores cromosomas filtrados. Se muestran los resultados finales, incluyendo el mejor cromosoma en términos de tiempo de cumplimiento.

```
# Mostrar resultados
print("Mejores soluciones encontradas con tiempos de encendido
menores a 5 horas:")

resultados = []  # Lista para almacenar resultados
mejor_tiempo_cumplimiento = float('inf')  # Para encontrar el mejor
tiempo de cumplimiento
mejor_cromosoma_global = None  # Para almacenar el mejor cromosoma
global

for i, cromosoma in enumerate(mejores_cromosomas_filtrados):
    mejor_aptitud = aptitud(cromosoma)
    tiempo_cumplimiento = 0
    od = valor_inicial
    for j in range(len(cromosoma)):
        if cromosoma[j] == 1:
            tiempo_cumplimiento += 1
            od += pendiente_encendido
        else:
```

```
            od += pendiente_apagado
        if od >= objetivo_od:
            break

    resultados.append({
        "Solución": i + 1,
        "Cromosoma": cromosoma,
        "Aptitud": mejor_aptitud,
        "Tiempo de Cumplimiento": tiempo_cumplimiento
    })

    # Verificar si este es el mejor cromosoma en términos del
tiempo de cumplimiento
    if tiempo_cumplimiento < mejor_tiempo_cumplimiento:
        mejor_tiempo_cumplimiento = tiempo_cumplimiento
        mejor_cromosoma_global = cromosoma

# Crear un DataFrame de Pandas con los resultados filtrados
df_filtrado = pd.DataFrame(resultados)
print(df_filtrado)

# Definir la función de simulación para intervalos de 30 minutos
def simular_comportamiento(cromosoma):
    nivel_od = [valor_inicial]
    for i in range(len(cromosoma)):
        if cromosoma[i] == 1:
            nivel_od.append(nivel_od[-1] + pendiente_encendido)
        else:
            nivel_od.append(nivel_od[-1] + pendiente_apagado)
    return nivel_od

# Graficar el comportamiento de todos los cromosomas filtrados
plt.figure(figsize=(10, 6))
for cromosoma in mejores_cromosomas_filtrados:
    nivel_od_simulado = simular_comportamiento(cromosoma)
    plt.plot(range(len(cromosoma) + 1), nivel_od_simulado,
alpha=0.3)

plt.axhline(y=4, color='r', linestyle='--', label='OD objetivo
(4)')
plt.xlabel("Tiempo (intervalos de 30 minutos)")
plt.ylabel("Nivel de Oxígeno Disuelto (OD)")
plt.title("Comportamiento simulado del OD para las mejores
soluciones filtradas")
plt.legend()
plt.show()

# Graficar el comportamiento del mejor cromosoma filtrado
plt.figure(figsize=(10, 6))
nivel_od_simulado = simular_comportamiento(mejor_cromosoma_global)
plt.plot(range(len(mejor_cromosoma_global) + 1), nivel_od_simulado,
label='Mejor Cromosoma', color='b')
```

```
plt.axhline(y=objetivo_od, color='r', linestyle='--', label=f'OD
objetivo ({objetivo_od})')
plt.xlabel("Tiempo (intervalos de 30 minutos)")
plt.ylabel("Nivel de Oxígeno Disuelto (OD)")
plt.title("Comportamiento simulado del OD para la mejor solución
filtrada")
plt.legend()
plt.show()

# Mostrar el mejor cromosoma en términos del tiempo de cumplimiento
print("Mejor cromosoma en términos del tiempo de cumplimiento:")
print(mejor_cromosoma_global)
print(f"Mejor tiempo de cumplimiento: {mejor_tiempo_cumplimiento}")
print(f"Nivel de oxígeno alcanzado con la mejor solución filtrada:
{nivel_od_simulado[-1]}")
```

4.3.2. Información del algoritmo genético

El algoritmo genético fue diseñado para optimizar el comportamiento del nivel de Oxígeno Disuelto (OD) en un sistema acuático a lo largo de un ciclo de 9 horas. El OD es crucial para la salud de los ecosistemas acuáticos, y su mantenimiento en niveles adecuados es esencial para garantizar un entorno sostenible.

Parámetros del Algoritmo Genético

- **Tamaño de la Población (100 individuos):** Un tamaño de población considerable (en este caso, 100 individuos) permite una diversidad suficiente para explorar el espacio de búsqueda de soluciones. Una población más grande aumenta las posibilidades de encontrar soluciones de alta calidad y evita que el algoritmo se quede atrapado en óptimos locales.
- **Tasa de Mutación (0.1):** Una tasa de mutación del 10% equilibra la exploración y la explotación del espacio de búsqueda. Una tasa moderada de mutación permite la introducción de nuevas soluciones, promoviendo la exploración del espacio de búsqueda, mientras que evita cambios demasiado drásticos que podrían perjudicar la convergencia hacia soluciones óptimas.
- **Generaciones (262144):** El número de generaciones se estableció en 262144 para permitir un número suficiente de iteraciones a lo largo del proceso evolutivo. Esta elección se basa en la complejidad del problema y la necesidad de brindar al algoritmo la oportunidad de converger hacia

soluciones de alta calidad por lo que buscando abarcar todas las opciones se partió de combinatoria básica, donde se elevó 2 a la 18, ya que están eran todas las posibles combinaciones de encendido y apagado que se podía tener. Un mayor número de generaciones proporciona tiempo para una exploración más exhaustiva del espacio de búsqueda.

Límites del Algoritmo Genético: Para garantizar la efectividad y aplicabilidad del algoritmo genético, se establecieron ciertos límites y restricciones:

Restricciones Temporales:

- **Intervalos de Tiempo:** Los cromosomas representan el comportamiento del sistema en intervalos de 30 minutos a lo largo de un ciclo de 9 horas. Esta representación temporal tiene un límite para asegurar una modelación precisa y relevante para el problema específico.

Restricciones de Comportamiento del OD:

- **Objetivo de Nivel de OD:** Se estableció un objetivo de nivel de OD igual a 4. Este límite define el umbral que el sistema debe alcanzar y mantener. La función de aptitud penaliza cualquier desviación por debajo de este valor.
- **Restricciones de Encendido y Apagado**: Se impusieron restricciones en el tiempo de encendido y apagado para evitar soluciones no prácticas. Si el sistema está encendido o apagado durante más de 7 intervalos consecutivos, se penaliza en la función de aptitud.

Parámetros del Modelo de Comportamiento del OD:

- **Pendiente de Encendido:** Indica la tasa de cambio en el nivel de Oxígeno Disuelto (OD) cuando el sistema está encendido. Es una medida de la eficiencia en la generación de oxígeno en el cuerpo acuático.
- **Pendiente de Apagado**: Representa la tasa de cambio en el nivel de OD cuando el sistema está apagado. Refleja la velocidad de disminución del oxígeno disuelto en ausencia de la actividad del sistema.

- **Valor Inicial de OD:** Es el nivel inicial de OD en el cuerpo acuático antes de que el sistema de aireación entre en funcionamiento. Sirve como punto de partida para el análisis de cambios en los niveles de OD.

Configuración Temporal:

- **Duración del Ciclo:** Indica la longitud del ciclo de funcionamiento del sistema de aireación. Define el período durante el cual se evalúa y optimiza el comportamiento del sistema.
- **Objetivo de Nivel de OD:** Representa el nivel deseado de Oxígeno Disuelto en el agua. El algoritmo genético busca optimizar el funcionamiento del sistema para mantener el OD cerca o igual a este objetivo, asegurando condiciones adecuadas para la vida acuática.

Representación Cromosómica

Cada individuo en la población es representado por un cromosoma, donde cada gen indica si el sistema está encendido o apagado en un intervalo de tiempo específico. En este caso, la representación utiliza intervalos de 30 minutos, dividiendo cada hora en dos intervalos.

Función de Aptitud

La función de aptitud evalúa la calidad de un individuo en función de su capacidad para mantener el nivel de OD dentro de los límites deseados y cumplir con restricciones específicas. La aptitud se calcula considerando el tiempo de cumplimiento del objetivo y la eficiencia en el uso del sistema.

La función de aptitud se define como:

$$P_{electrica} = U * I * \cos\varphi \qquad [1]$$

$$Aptitud = \frac{1}{Tiempo\ de\ cumplimiento + 1}\ x \frac{1}{Tiempo\ de\ encendido\ total + 1}$$

donde:

- Tiempo de Cumplimiento: Representa el número de intervalos en los que el nivel de OD alcanza o supera el objetivo (4).
- Tiempo de Encendido Total: Es la suma de los intervalos en los que el sistema está activo (encendido).

El término $\frac{1}{Tiempo\ de\ cumplimiento+1}$ favorece soluciones que cumplen rápidamente con el objetivo de OD, ya que minimiza el valor de la aptitud cuando el tiempo de cumplimiento es alto.

El término $\frac{1}{Tiempo\ de\ encendido\ total+1}$ favorece soluciones que utilizan eficientemente el sistema, ya que minimiza el valor de la aptitud cuando el tiempo de encendido total es alto.

En resumen, la duración del ciclo y el objetivo de nivel de OD influyen directamente en la forma en que se calcula la aptitud, determinando qué soluciones son más deseables en términos de cumplimiento del objetivo y eficiencia en el uso del sistema a lo largo del tiempo.

4.3.3. Resultados del algoritmo genético en intervalos de 30 minutos

En esta sección, se presentan y analizan los resultados obtenidos mediante la ejecución del algoritmo genético diseñado para la optimización del comportamiento del nivel de Oxígeno Disuelto (OD) en intervalos de 30 minutos. El algoritmo, configurado con parámetros específicos y fundamentado en la modelización del sistema acuático, ha sido aplicado a lo largo de un ciclo de 9 horas.

A través de un exhaustivo análisis de los cromosomas generados por el algoritmo, se exploran las soluciones encontradas, enfocándose en aquellas que cumplen con restricciones temporales y de nivel de OD establecidas. Además, se examina el rendimiento de la población evolutiva en términos de tiempo de

encendido, cumplimiento de objetivos y eficiencia en la utilización del sistema de aireación.

Los resultados obtenidos proporcionarán una visión detallada de la capacidad del algoritmo para mejorar la gestión del OD, permitiendo evaluar su efectividad y su impacto en la calidad del entorno acuático. Para el caso el aireador de paletas, se tiene la representación gráfica de todos los resultados en la ilustración 34.

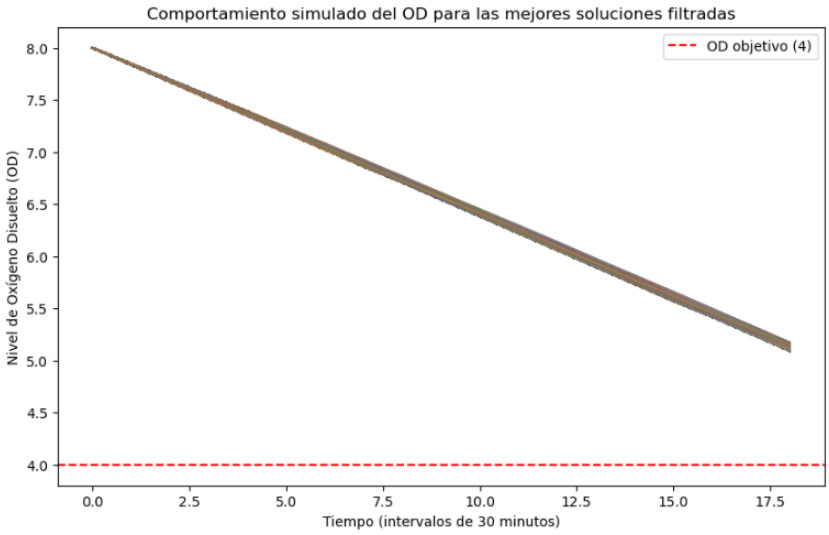

Ilustración 34. Todas las combinaciones en intervalos de 30 minutos, aireador de paletas. Fuente: (Elaboración propia).

Donde el caso más favorable se presenta en la ilustración 35 en la que se puede también notar el cromosoma que representa los intervalos de encendido y apagado, así como el nivel de OD estimado.

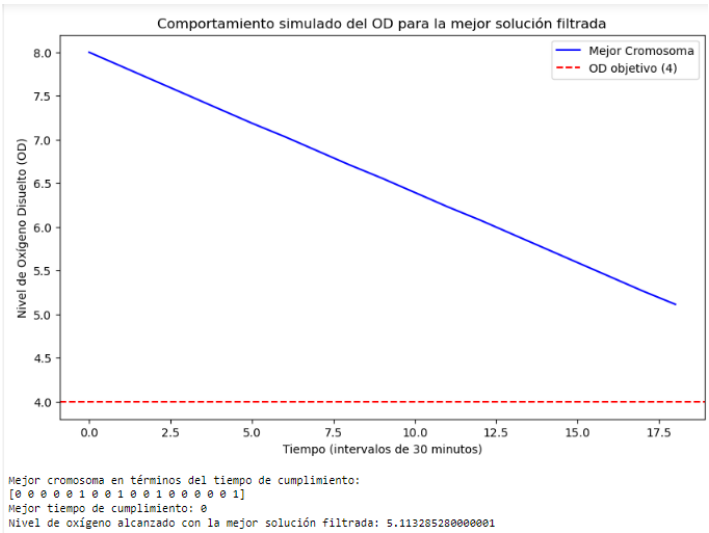

Ilustración 35. Mejor respuesta propuesta por el sistema, para aireador de paletas a 30 minutos. Fuente: (Elaboración propia).

Para el caso del aireador de fuente, se tiene la representación gráfica de todos los resultados en la ilustración 36.

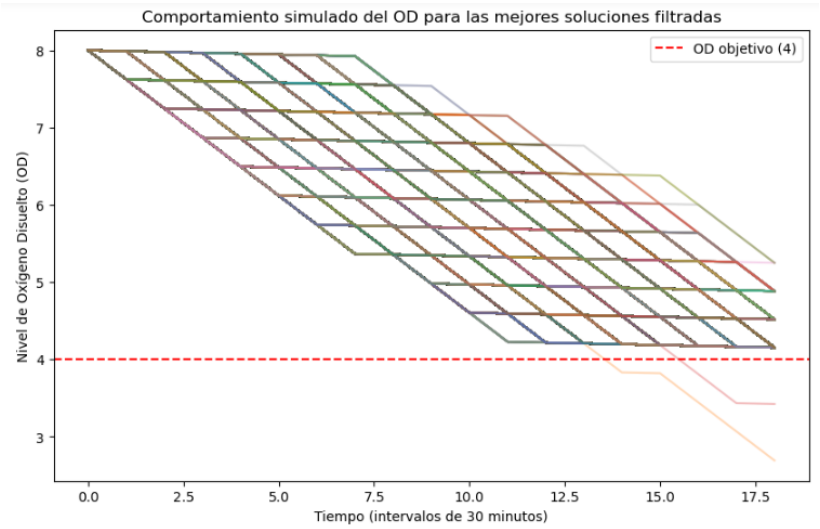

Ilustración 36 Todas las combinaciones en intervalos de 30 minutos, aireador de fuente. Fuente: (Elaboración propia).

Donde el caso más favorable se presenta en la ilustración 37, en la cual se puede también notar el cromosoma que representa los intervalos de encendido y apagado, así como el nivel de OD estimado.

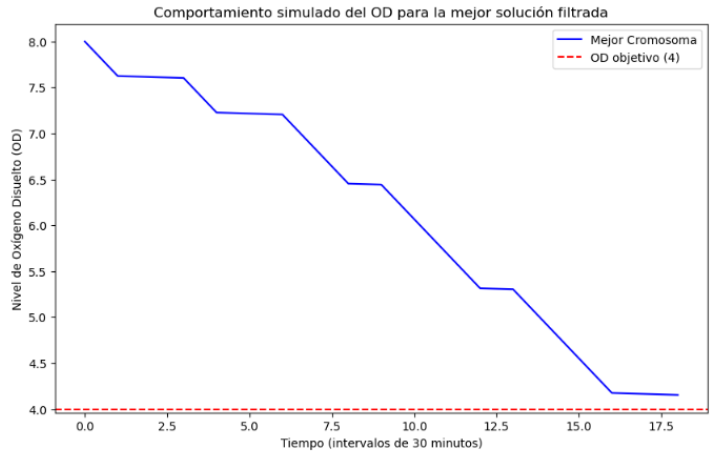

Ilustración 37. Mejor respuesta para el aireador de fuentes en intervalos de 30 minutos. Fuente: (Elaboración propia).

Para el caso el aireador de inyección, se tiene la representación gráfica de todos los resultados en la ilustración 38.

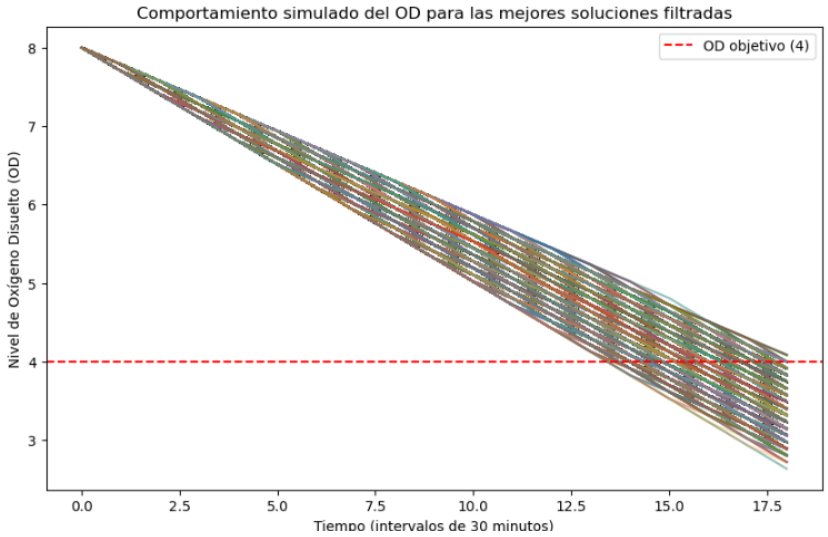

Ilustración 38. Todas las combinaciones en intervalos de 30 minutos, aireador de inyección. Fuente: (Elaboración propia).

Donde el caso más favorable se presenta en la ilustración 39 donde se puede también notar el cromosoma que representan los intervalos de encendido y apagado, así como el nivel de OD estimado.

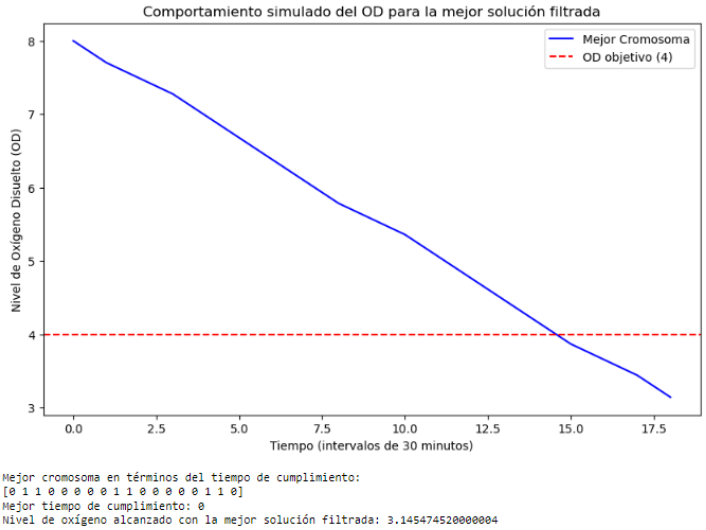

Ilustración 39. Mejor respuesta del sistema a 30 minutos, para aireador de inyección. Fuente: (Elaboración propia).

Para poder evaluar las respuestas del algoritmo, se procedió a correr el mismo algoritmo, sólo que adaptando nuevos intervalos de tiempo para 15, 20, 60 y 120 minutos. El código completo de este proceso se encuentra en el anexo 8.

4.3.4. Resultados en otros intervalos

Para tener parámetros de comparación además de la secuencia original (media hora encendido y apagado), se procedió a repetir el algoritmo, con otros intervalos de tiempo cómo 15, 20, 60 y 120 minutos los códigos correspondientes, están en los anexos 9, 10, 11 y 12

Los únicos factores que se ajustaron fueron las pendientes, las cuales se muestran en la tabla 15 que se multiplicaron por la cantidad de segundos correspondientes para caso.

Tabla 15. Pendientes del OD para otros intervalos de tiempo. Fuente:(Elaboración propia).

Aireador	Pendiente	Segundo	15 min	20 min	30 min	60 min	120 min
AP	Pendiente encendido	-0.15288948	-0.07644474	-0.10192632	-0.15288948	-0.30577896	-0.61155792
	Pendiente apagado	-0.1625112	-0.0812556	-0.1083408	-0.1625112	-0.3250224	-0.6500448
AF	Pendiente encendido	-0.01066266	-0.00533133	-0.00710844	-0.01066266	-0.02132532	-0.04265064
	Pendiente apagado	-0.37602666	-0.18801333	-0.25068444	-0.37602666	-0.75205332	-1.50410664
AI	Pendiente encendido	-0.21282498	-0.10641249	-0.14188332	-0.21282498	-0.42564996	-0.85129992
	Pendiente apagado	-0.2981313	-0.14906565	-0.1987542	-0.2981313	-0.5962626	-1.1925252

Repuestas para 15 minutos

Repuesta del aireador de paletas, con intervalos de 15 minutos, en la ilustración 40 se presentan todas las respuestas posibles en la izquierda y en la mejor respuesta en la derecha .

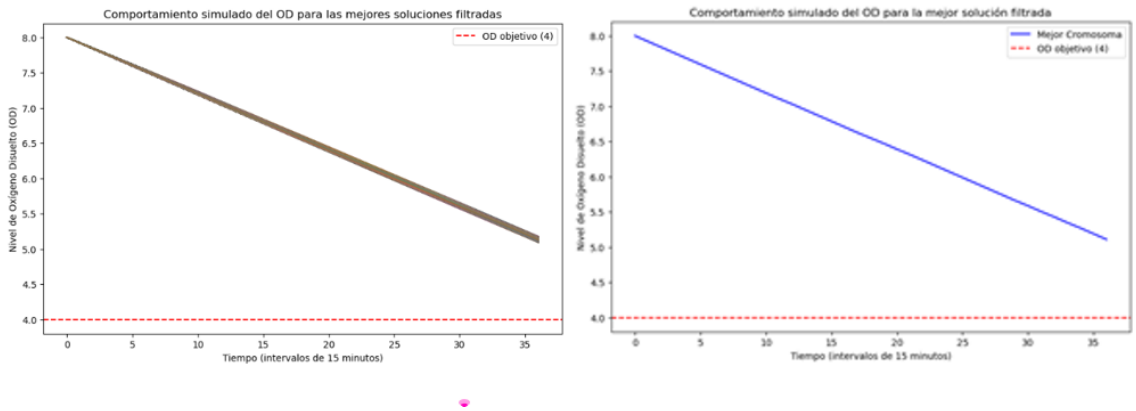

Ilustración 40. Todas las respuestas del sistema en lado izquierdo y mejor respuesta gráfica de la derecha para aireador de paletas a 15 minutos. Fuente: (Elaboración propia).

Repuesta del aireador de fuente, con intervalos de 15 minutos, en la ilustración 41 se presentan todas las respuestas posibles en la izquierda y en la mejor respuesta en la derecha.

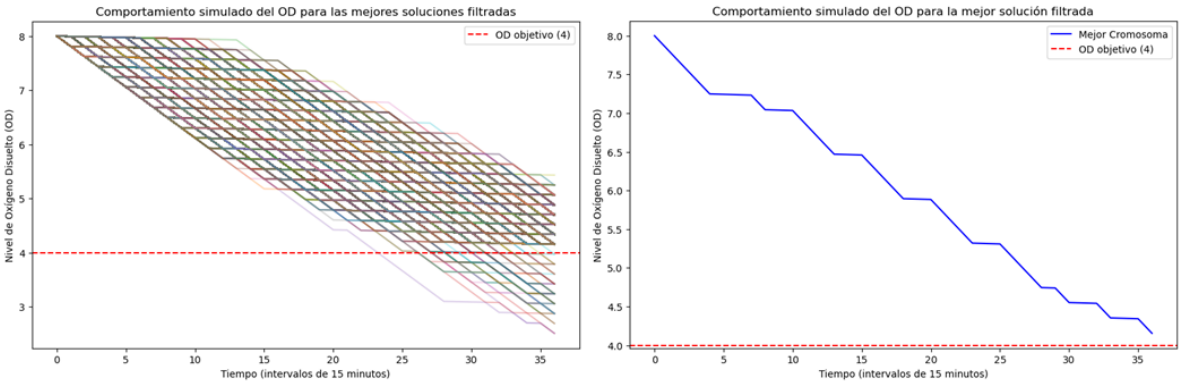

Ilustración 41. Todas las respuestas del sistema en lado izquierdo y mejor respuesta gráfica de la derecha para aireador de fuente a 15 minutos. Fuente: (Elaboración propia).

Repuesta del aireador de inyección, con intervalos de 15 minutos, en la ilustración 42 se presentan todas las respuestas posibles en la izquierda y en la mejor respuesta en la derecha .

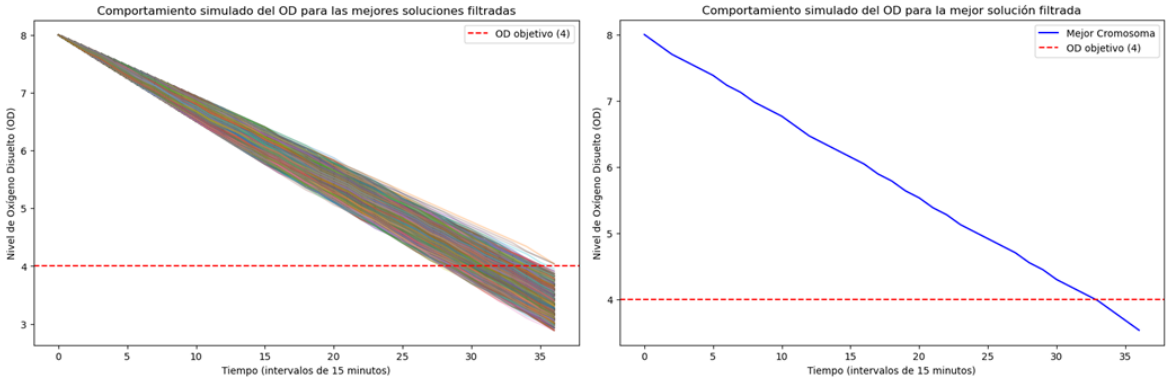

Ilustración 42. Todas las respuestas del sistema en lado izquierdo y mejor respuesta gráfica de la derecha para aireador de inyección a 15 minutos. Fuente: (Elaboración propia).

Repuestas para 20 minutos

Repuesta del aireador de paletas, con intervalos de 20 minutos, en la ilustración 43 se presentan todas las respuestas posibles en la izquierda y en la mejor respuesta en la derecha .

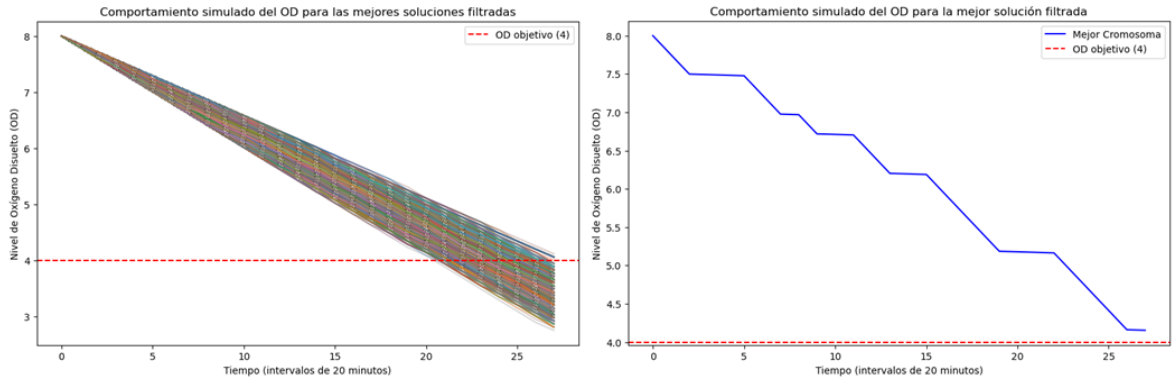

Ilustración 43. Todas las respuestas del sistema en lado izquierdo y mejor respuesta grafica de la derecha para aireador de paletas a 20 minutos. Fuente: (Elaboración propia).

Repuesta del aireador de fuente, con intervalos de 20 minutos, en la ilustración 44 se presentan todas las respuestas posibles en la izquierda y en la mejor respuesta en la derecha .

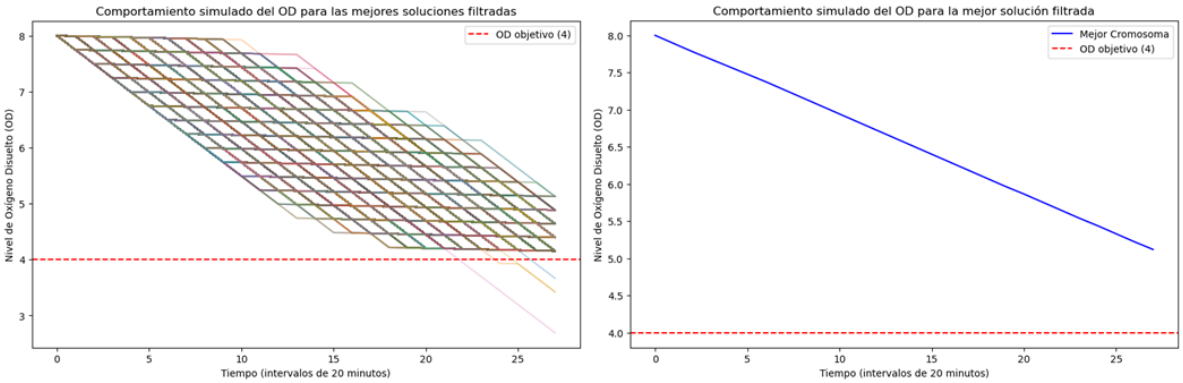

Ilustración 44. Todas las respuestas del sistema en lado izquierdo y mejor respuesta gráfica de la derecha para aireador de fuente a 20 minutos. Fuente: (Elaboración propia).

Repuesta del aireador de inyección, con intervalos de 20 minutos, en la ilustración 45 se presentan todas las respuestas posibles en la izquierda y en la mejor respuesta en la derecha .

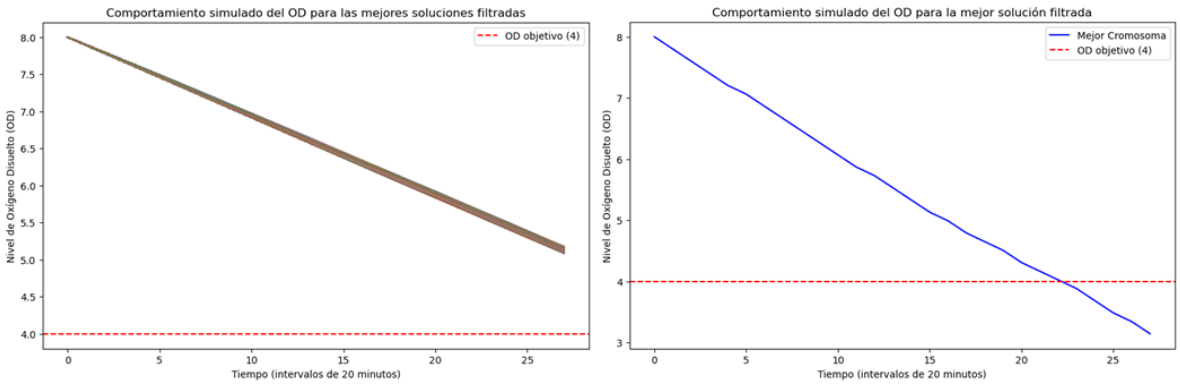

Ilustración 45. Todas las respuestas del sistema en lado izquierdo y mejor respuesta gráfica de la derecha para aireador de paletas a 20 minutos. Fuente: (Elaboración propia).

Repuestas para 60 minutos

Repuesta del aireador de paletas, con intervalos de 60 minutos, en la ilustración 46 se presentan todas las respuestas posibles en la izquierda y en la mejor respuesta en la derecha .

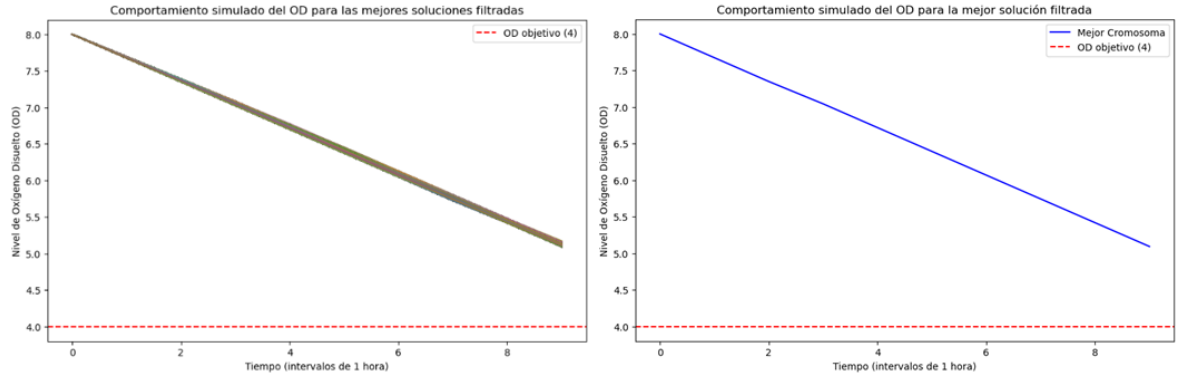

Ilustración 46. Todas las respuestas del sistema en lado izquierdo y mejor respuesta gráfica de la derecha para aireador de paletas a 60 minutos. Fuente: (Elaboración propia).

Repuesta del aireador de fuente, con intervalos de 60 minutos, en la ilustración 47 se presentan todas las respuestas posibles en la izquierda y en la mejor respuesta en la derecha .

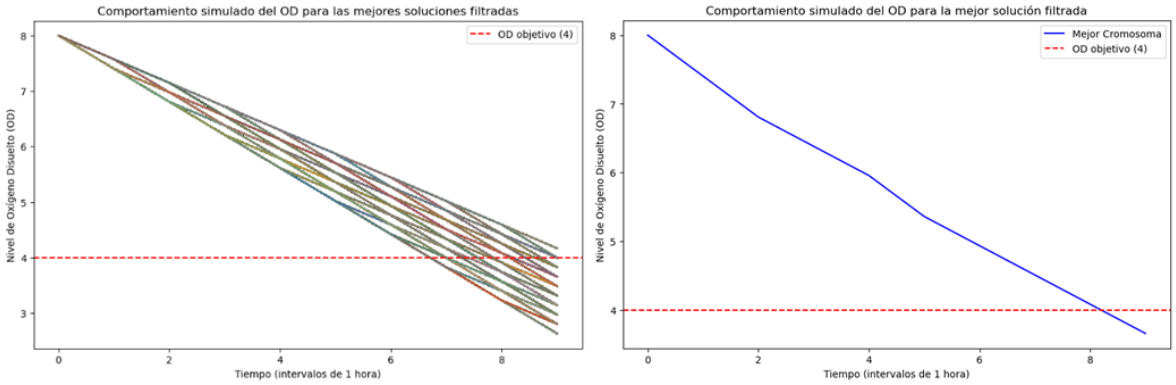

Ilustración 47. Todas las respuestas del sistema en lado izquierdo y mejor respuesta gráfica de la derecha para aireador de fuente a 60 minutos. Fuente: (Elaboración propia).

Repuesta del aireador de inyección, con intervalos de 60 minutos, en la ilustración 48 se presentan todas las respuestas posibles en la izquierda y en la mejor respuesta en la derecha .

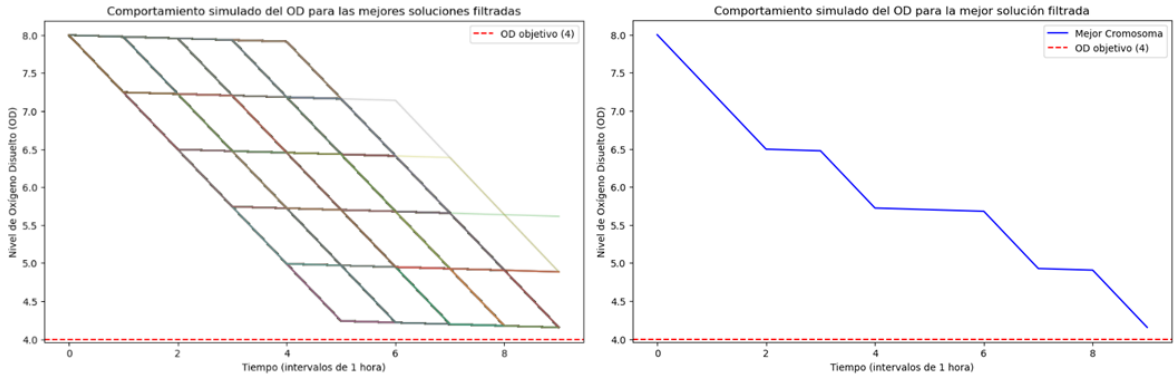

Ilustración 48. Todas las respuestas del sistema en lado izquierdo y mejor respuesta gráfica de la derecha para aireador de inyección a 60 minutos. Fuente: (Elaboración propia).

Repuestas para 120 minutos

Repuesta del aireador de paletas, con intervalos de 120 minutos, en la ilustración 49 se presentan todas las respuestas posibles en la izquierda y en la mejor respuesta en la derecha .

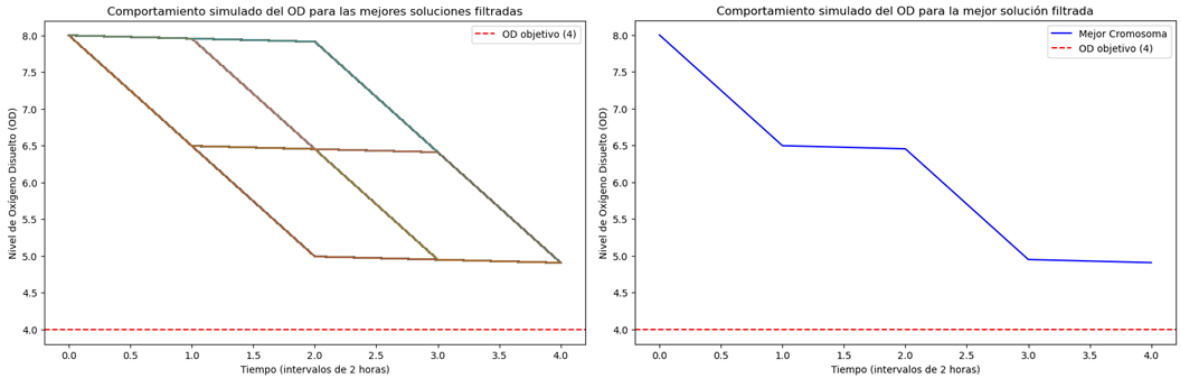

Ilustración 49. Todas las respuestas del sistema en lado izquierdo y mejor respuesta gráfica de la derecha para aireador de paletas a 120 minutos. Fuente: (Elaboración propia).

Repuesta del aireador de fuente, con intervalos de 120 minutos, en la ilustración 50 se presentan todas las respuestas posibles en la izquierda y en la mejor respuesta en la derecha .

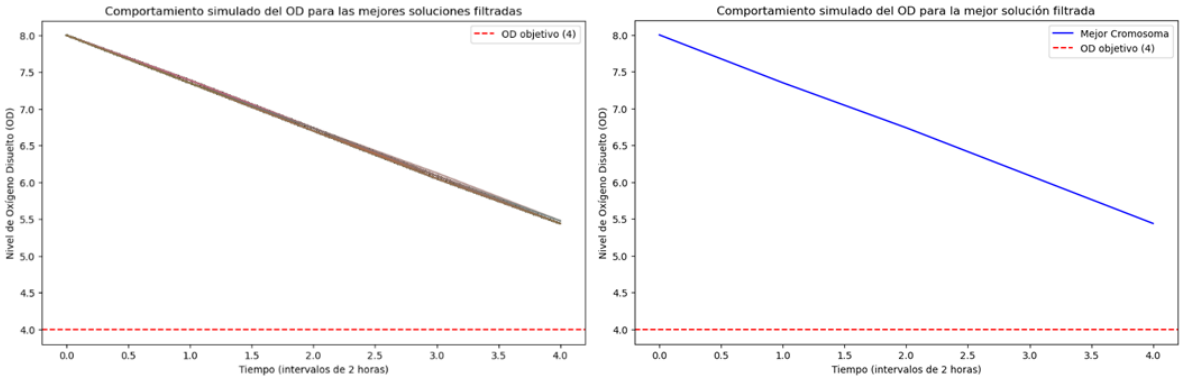

Ilustración 50. Todas las respuestas del sistema en lado izquierdo y mejor respuesta gráfica de la derecha para aireador de fuente a 120 minutos. Fuente: (Elaboración propia).

Repuesta del aireador de inyección, con intervalos de 120 minutos, en la ilustración 51 se presentan todas las respuestas posibles en la izquierda y en la mejor respuesta en la derecha .

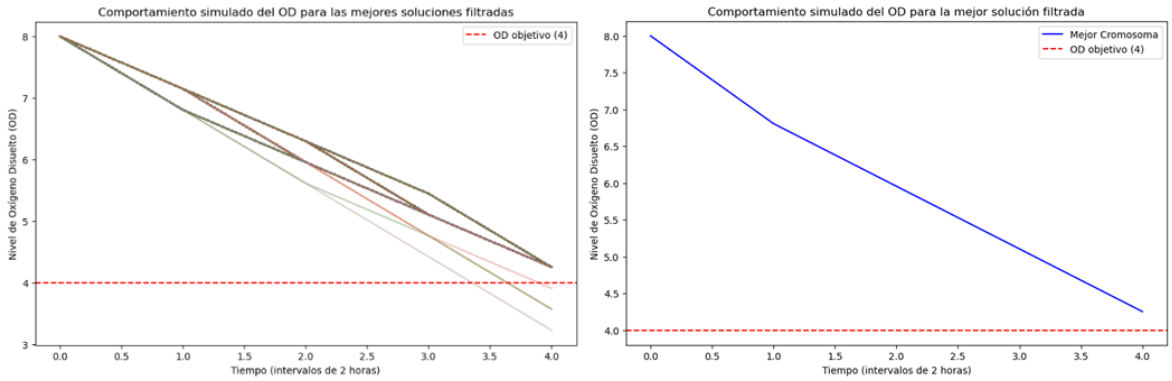

Ilustración 51. Todas las respuestas del sistema en lado izquierdo y mejor respuesta gráfica de la derecha para aireador de inyección a 120 minutos. Fuente: (Elaboración propia).

4.3.5. Resultado del algoritmo

Los resultados obtenidos para resultado se muestran en la tabla 16, donde se presenta una etiqueta para cada secuencia, el intervalo de tiempo en el que operaría, el tiempo que estaría encendido y el nivel de OD que debería alcanzar. Cabe mencionar que en los 3 primeros registros la N representa, la programación Normal, es decir, el encendido y apagado que se ha estado trabajando de manera cotidiana.

Tabla 16. Información resultante de los algoritmos. Fuente:(Elaboración propia).

Aireador	Secuencia	Intervalos de tiempo (minutos)	Nivel de OD Alcanzado	Tiempo Encendido (horas)
Paletas 60	[0,0,1,0,0,0,0,0,0]	60	5.094	1
Paletas 15	[0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,1,0,1,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,0]	15	5.1084	1.75
Paletas 120	[0,1,0,0]	120	5.4383	2
Paletas 20	[0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,1]	20	5.1196	2.3333
Inyección 20	[0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,1,0,1,1,0,1,1,1,0,0,1,0]	20	4.1544	3
Fuente 20	[0,0,1,1,1,0,0,1,0,1,1,0,0,1,1,0,0,0,0,1,1,1,0,0,0,0,1]	20	5.1132	4
Fuente 15	[0,0,0,0,1,1,1,0,1,1,0,0,0,1,1,0,0,0,1,1,0,0,0,1,1,0,0,0,1,0,1,1,0,1,1,0]	15	4.1544	4
Fuente 30	[0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1],	30	4.1544	4
Fuente 60	[0,0,1,0,1,1,0,1,0]	60	4.1544	4
Paletas 30	[0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0],	30	3.1454	4
Inyección 30	[0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0]	30	3.1454	4
Paletas N	[1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1],	30	Varia	5
Fuente N	[1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1],	30	Varia	5
Inyección N	[1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1],	30	Varia	5
Fuente 120	[0,1,0,1]	120	4.9064	5
Inyección 15	[0,0,1,1,1,0,1,0,1,1,0,0,1,1,1,1,0,1,0,1,0,1,0,1,1,1,1,0,1,0,1,1,1,0,0,0]	15	3.5293	5.25
Inyección 60	[0,0,1,1,0,1,1,1,1]	60	3.6573	6
Inyección 120	[0,1,1,1]	120	4.2535	7

A continuación, se presenta el análisis de los resultados antes mostrados.

4.4. Interpretación de los resultados

Con el fin de dar sentido a la exploración, transformación y aplicación de algoritmos a los datos recopilados, se desglosa el análisis de cada uno de los procesamientos

aplicados los códgios usados para estas respuestas se encuentran en los anexos 13 y 14.

4.4.1. Consumo diario por aireador en intervalos de 5 minutos

Para llevar a cabo un análisis exhaustivo del desempeño de las propuestas generadas por el algoritmo, se emplea una herramienta visual fundamental: una gráfica lineal que detalla el consumo energético de los aireadores a lo largo de la noche. La ilustración 52 se erige como una representación minuciosa y esclarecedora del consumo diario de energía para cada aireador, segmentado en intervalos de 5 minutos durante el transcurso de la noche.

A través de esta representación gráfica intuitiva, los usuarios son capaces de discernir patrones de consumo energético, identificar picos y valles, lo que conduce a una comprensión enriquecedora y detallada del comportamiento energético de cada aireador a lo largo del tiempo. Esta visualización proporciona una herramienta invaluable para evaluar la eficiencia energética de las propuestas y guiar la toma de decisiones estratégicas en la gestión de la energía en la granja acuícola..

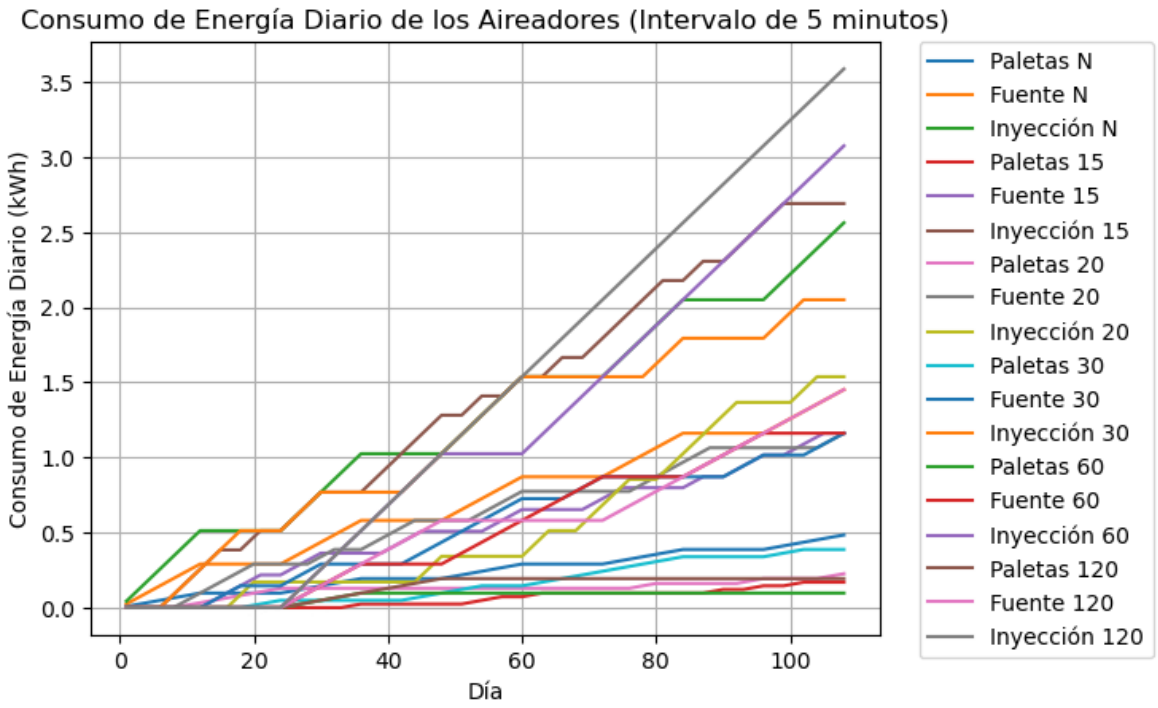

Ilustración 52. Comportamiento del consumo de energía eléctrica a lo largo de una noche con diferentes secuencias.

Fuente: (Elaboración propia).

En total son 18 líneas, 15 de las cuales coinciden con las propuestas generadas por el algoritmo, donde el diferenciador es el número que corresponde a la cantidad de minutos de intervalo, y "N" representa la programación normal, la cual se ha estado utilizando de manera cotidiana en la granja. Visualmente, la respuesta que muestra un mayor consumo es la del aireador de inyección con intervalos de 15 minutos, mientras que la menos costosa es la de aireación con paletas configurada a 60 minutos. Estos hallazgos son fundamentales para identificar las configuraciones más eficientes y económicas para cada tipo de aireador, lo que permite optimizar el consumo de energía y reducir los costos asociados en la operación de la granja.

4.4.2. Consumo mensual y anual por aireador

En las siguientes ilustraciones se muestran los resultados de los cálculos realizados para determinar y presentar el consumo diario, mensual y anual de energía para

cada aireador. Mediante gráficas de barras claramente etiquetadas, los usuarios pueden evaluar de un vistazo las variaciones estacionales en el consumo, facilitando la identificación de tendencias a largo plazo.

La ilustración 53 corresponde al consumo diario de energía, la ilustración 54 se visualiza a nivel mensual y la 55 contempla el consumo anual, para tener una perspectiva del consumo a largo plazo que representa cada aireador.

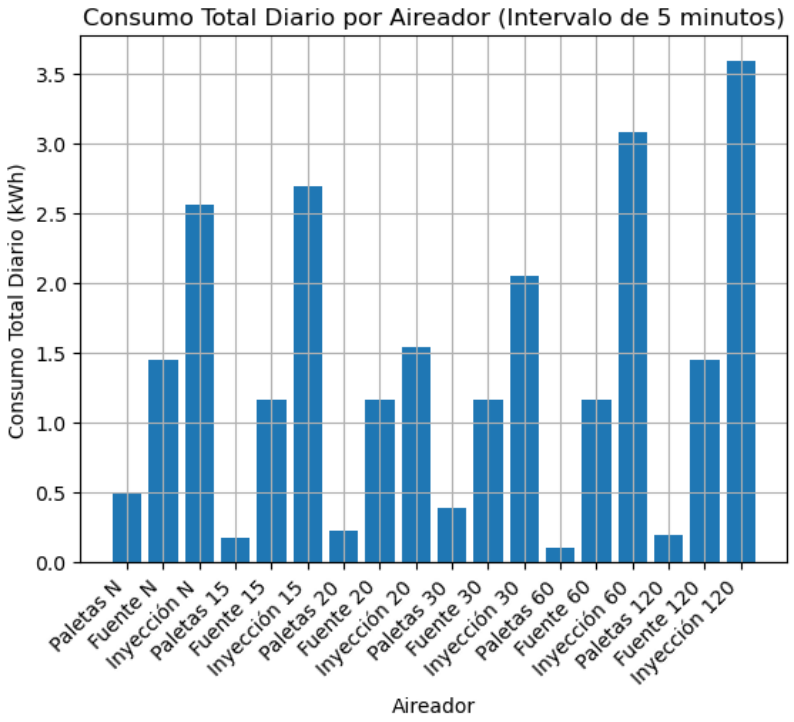

Ilustración 53. Consumo diario por secuencia y tipo de aireador. Fuente: (Elaboración propia).

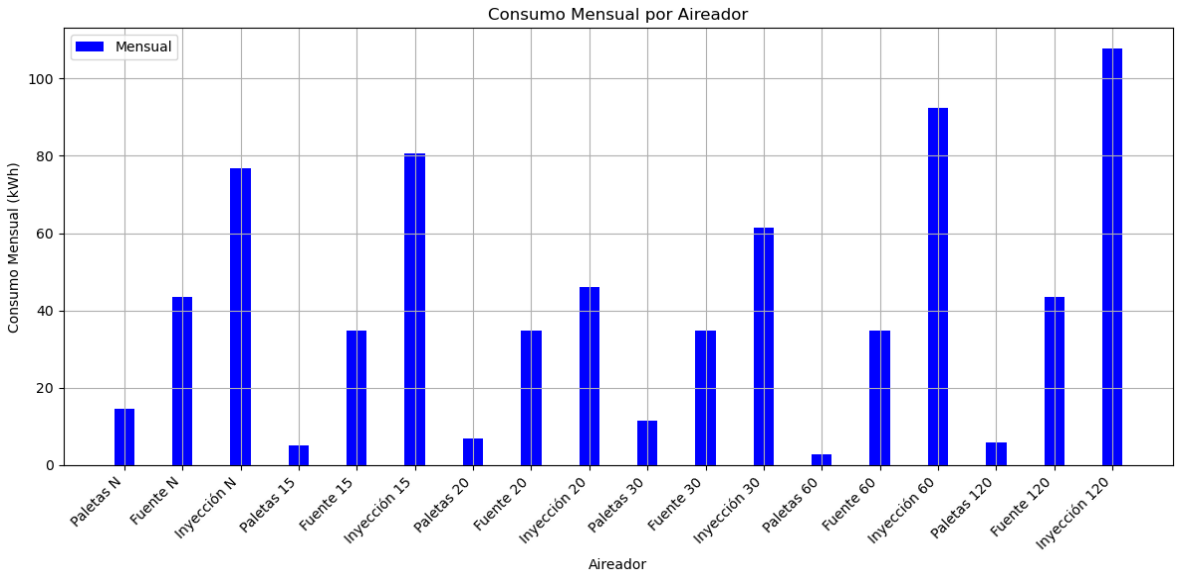

Ilustración 54. Consumo mensual por secuencia y tipo de aireador. Fuente: (Elaboración propia).

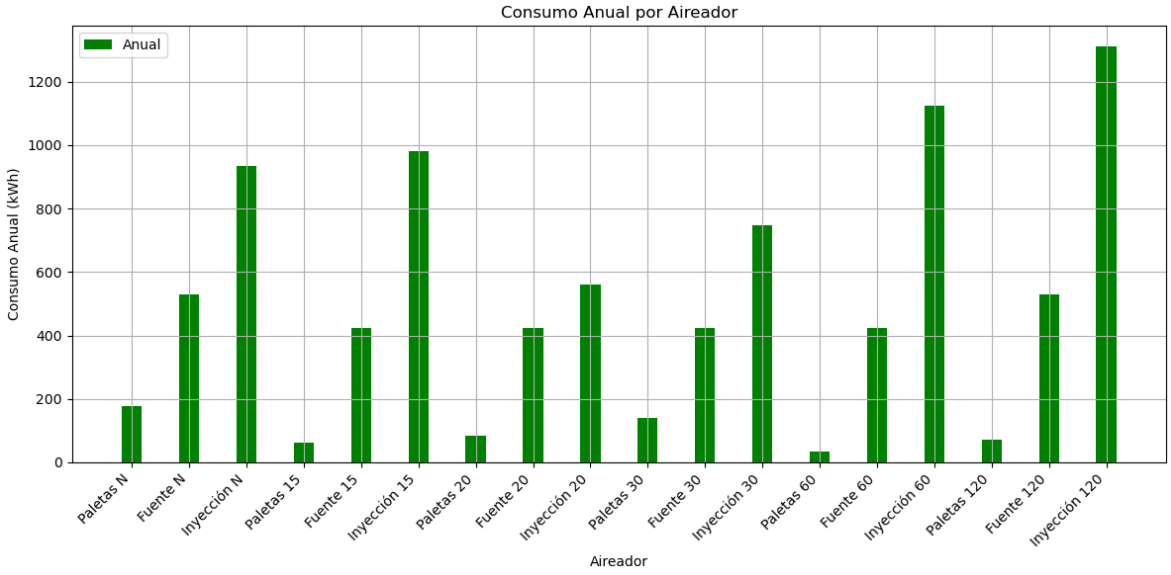

Ilustración 55. Consumo anual por secuencia y tipo de aireador. Fuente: (Elaboración propia).

4.4.3 Tiempo de encendido y consumo para cada aireador:

La tabla 17 permite visualizar un resumen diario, mensual y anual del consumo de energía de cada propuesta de los aireadores, la columna “diferencia”, corresponde a la cantidad de energía se ahorra o se gasta de más con cada propuesta, si es negativo existe un ahorro, si es positivo el aireador gasta más.

Tabla 17. Consumo de energía por cada aireador y su respectiva secuencia. Fuente:(Elaboración propia)

Nombre Aireador	Horas encendidas	Consumo Diario kWh	DIFERENCIA	Consumo Mensual kWh	DIFERENCIA	Consumo Anual kWh	DIFERENCIA
Paletas							
Paletas N	5	0.484	0	14.52	0	176.66	0
Paletas 15	1.75	0.1694	-0.3146	5.082	-9.438	61.831	-114.829
Paletas 20	2.333333	0.2259	-0.258133	6.776	-7.744	82.44133	-94.2187
Paletas 30	4	0.3872	-0.0968	11.616	-2.904	141.328	-35.332
Paletas 60	1	0.0968	-0.3872	2.904	-11.616	35.332	-141.328
Paletas 120	2	0.1936	-0.2904	5.808	-8.712	70.664	-105.996
Fuente							
Fuente N	5	1.452	0	43.56	0	529.98	0
Fuente 15	4	1.1616	-0.2904	34.848	-8.712	423.984	-105.996
Fuente 20	4	1.1616	-0.2904	34.848	-8.712	423.984	-105.996
Fuente 30	4	1.1616	-0.2904	34.848	-8.712	423.984	-105.996
Fuente 60	4	1.1616	-0.2904	34.848	-8.712	423.984	-105.996
Fuente 120	5	1.452	0	43.56	0	529.98	0
Inyección							
Inyección N	5	2.563	0	76.89	0	935.495	0
Inyección 15	5.25	2.6912	0.12815	80.7345	3.8445	982.2698	46.77475
Inyección 20	3	1.5378	-1.0252	46.134	-30.756	561.297	-374.198
Inyección 30	4	2.0504	-0.5126	61.512	-15.378	748.396	-187.099
Inyección 60	6	3.0756	0.5126	92.268	15.378	1122.594	187.099
Inyección 120	7	3.5882	1.0252	107.646	30.756	1309.693	374.198

Para el caso de los aireadores de paletas la mejor respuesta corresponde a 60 minutos encendido, ahorrando 0.3872 kWh diarios, para el caso de fuente, se tienen 4 propuestas, cada una corresponde a 4 horas encendidas, siendo las propuestas de 15, 20, 30 y 60 minutos ahorrando 0.2904 kWh y por último para el caso de inyección, la mejor respuesta corresponde a intervalos de 20 minutos ahorrando 1.0252 kWh. Entre estas tres alternativas, la que menos consume es el aireador de paletas, seguidos del aireador de fuente y por último el de inyección.

4.4.4. Resumen en forma de *dataframe*

El sistema genera un DataFrame integral que resume los aspectos clave de la información recopilada. Incluye el nombre de cada aireador, el tiempo total encendido y los consumos diario, mensual y anual de energía. Este resumen estructurado proporciona una referencia rápida y fácil para comparar y analizar el rendimiento de cada aireador ver tabla 18.

Tabla 18. Registro del consumo de energía y cumplimiento de objetivos. Fuente:(Elaboración propia).

Nombre Aireador	Nivel de OD	Horas ON	Consumo Diario (kWh)	Diferencia con N	Cumple	% con N
Paletas						
Paletas N	Varia	5	0.484	0		100%
Paletas 15	5.1084	1.75	0.1694	-0.3146	Ahorra	35%
Paletas 20	5.1196	2.3333	0.225867	-0.2581	Ahorra	47%
Paletas 30	3.1454	4	0.3872	-0.0968	Ahorra	80%
Paletas 60	5.094	1	0.0968	-0.3872	Ahorra	20%
Paletas 120	5.4383	2	0.1936	-0.2904	Ahorra	40%
Fuente						
Fuente N	Varia	5	1.452	0		100%
Fuente 15	4.1544	4	1.1616	-0.29	Ahorra	80%
Fuente 20	5.1132	4	1.1616	-0.29	Ahorra	80%
Fuente 30	4.1544	4	1.1616	-0.29	Ahorra	80%
Fuente 60	4.1544	4	1.1616	-0.29	Ahorra	80%
Fuente 120	4.9064	5	1.452	0	Ahorra	100%
Inyección						
Inyección N	Varia	5	2.563	0		100%
Inyección 15	3.5293	5.25	2.69115	0.1282	Gasta	105%
Inyección 120	4.2535	7	3.5882	1.0252	Ahorra	140%
Inyección 20	4.1544	3	1.5378	-1.025	Ahorra	60%
Inyección 30	3.1454	4	2.0504	-0.513	Ahorra	80%
Inyección 60	3.6573	6	3.0756	0.5126	Ahorra	120%

La columna “% con N” representa el porcentaje de operación de cada patrón, comparado con el patrón original “N”, donde la mejor respuesta del aireador de paletas usa sólo un 20%, 80% la de fuente y 60% en el caso de inyección.

4.4.5. Gráficas de costos

Para comprender mejor el costo de operación de cada aireador, se utilizan las gráficas de la ilustración 56 para mostrar el costo diario de operación, 57 para el mensual y 58 el anual.

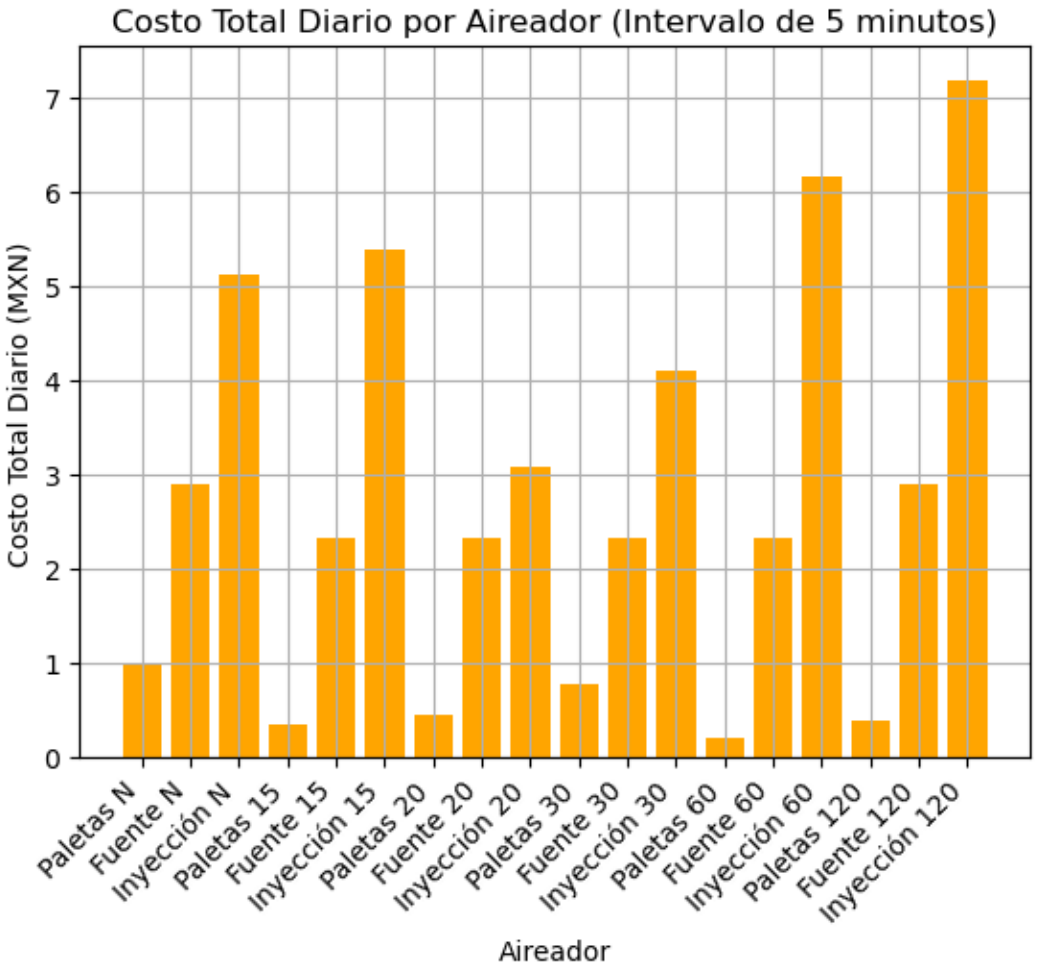

Ilustración 56. Costo diario de operación de cada aireador. Fuente: (Elaboración propia).

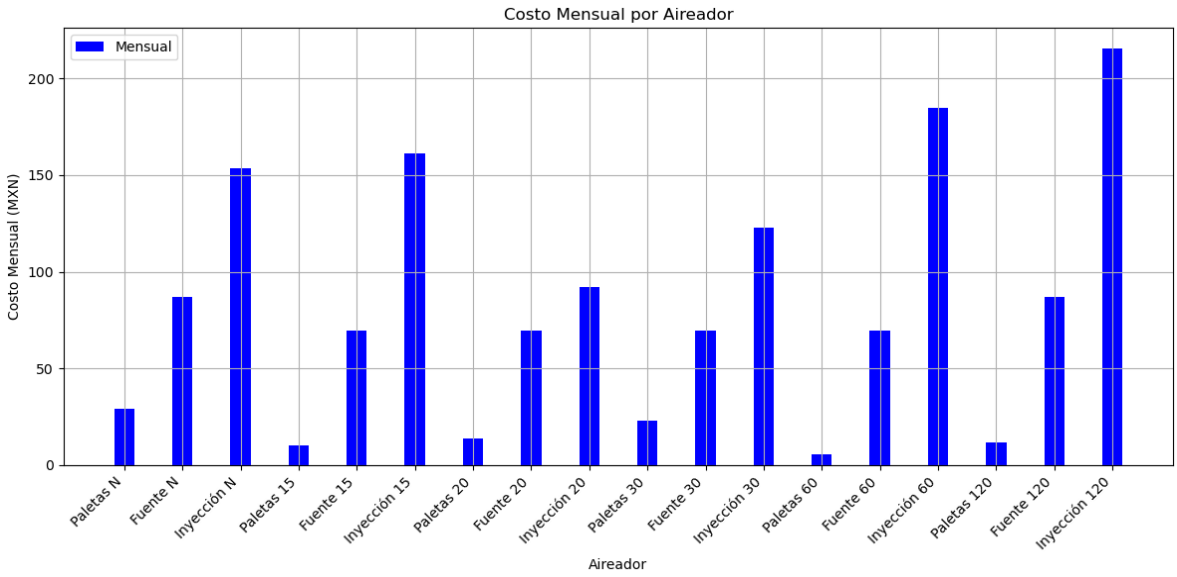

Ilustración 57. Costo mensual de operación de cada aireador. Fuente: (Elaboración propia).

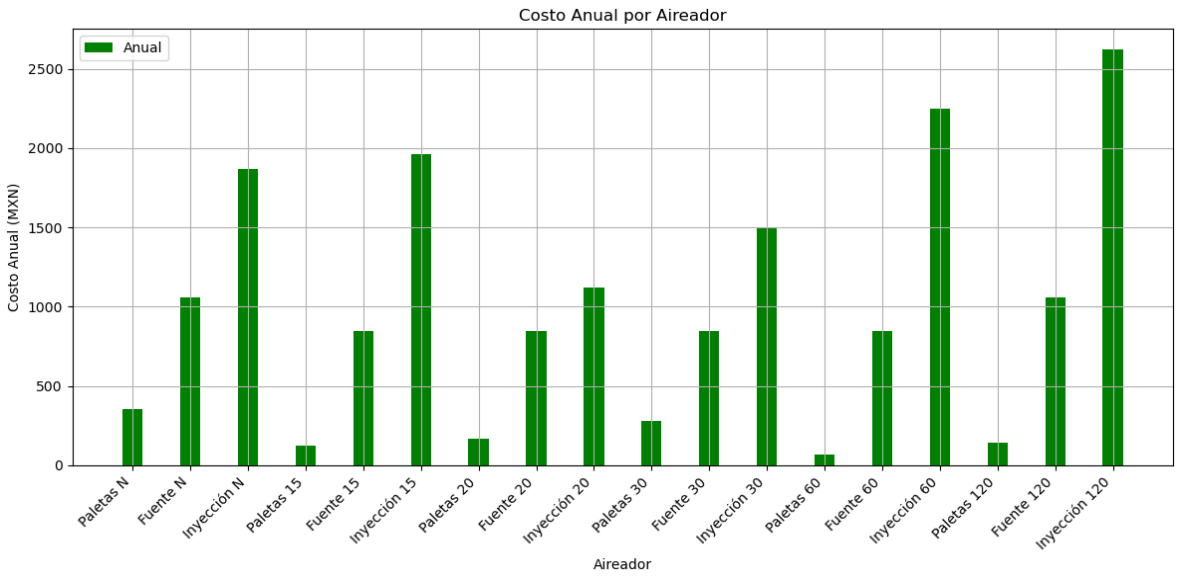

Ilustración 58. Costo anual de operación de cada aireador. Fuente: (Elaboración propia).

No existe ninguna alteración, las mejores alternativas siguen siendo las mismas, lo único que permiten estas gráficas es visualizar el costo que implicaría estas nuevas rutinas de operación y contrastarlas entre sí con la original.

4.4.6. Costos asociados

Para proporcionar una evaluación más completa, el sistema establece un costo por kilovatio-hora en pesos y calcula los costos diarios, mensuales y anuales asociados con el consumo de energía de cada aireador. Estos costos son fundamentales para tomar decisiones informadas en términos de eficiencia y rentabilidad, permitiendo una gestión óptima de los recursos financieros la tabla 19 permite un resumen de estos costos.

Tabla 19 Registro de los costos de operación diario, mensual y anual, con su respectiva comparación.

Fuente:(Elaboración propia).

Nombre Aireador	Costo diario (en pesos)	Dif	Costo mensual (en pesos)	Dif	Costo anual (en pesos)	Dif
Paletas						
Paletas N	$0.97	$0.00	$29.04	$0.00	$353.32	$ -
Paletas 15	$0.34	-$0.63	$10.16	-$18.88	$123.66	-$ 226.51
Paletas 20	$0.45	-$0.52	$13.55	-$15.49	$164.88	-$ 185.86
Paletas 30	$0.77	-$0.19	$23.23	-$5.81	$282.66	-$ 69.70
Paletas 60	$0.19	-$0.77	$5.81	-$23.23	$70.66	-$ 278.78
Paletas 120	$0.39	-$0.58	$11.62	-$17.42	$141.33	-$ 209.09
Fuente						
Fuente N	$2.90	$0.00	$87.12	$0.00	$1,059.96	$ -
Fuente 15	$2.32	-$0.58	$69.70	-$17.42	$847.97	-$ 209.09
Fuente 20	$2.32	-$0.58	$69.70	-$17.42	$847.97	-$ 209.09
Fuente 30	$2.32	-$0.58	$69.70	-$17.42	$847.97	-$ 209.09
Fuente 60	$2.32	-$0.58	$69.70	-$17.42	$847.97	-$ 209.09
Fuente 120	$2.90	$0.00	$87.12	$0.00	$1,059.96	$ -
Inyección						
Inyección N	$5.13	$0.00	$153.78	$0.00	$1,870.99	$ -
Inyección 15	$5.38	$0.26	$161.47	$7.69	$1,964.54	$ 92.27
Inyección 120	$7.18	$2.05	$215.29	$61.51	$2,619.39	$ 738.14
Inyección 20	$3.08	-$2.05	$92.27	-$61.51	$1,122.59	-$ 738.14
Inyección 30	$4.10	-$1.03	$123.02	-$30.76	$1,496.79	-$ 369.07
Inyección 60	$6.15	$1.03	$184.54	$30.76	$2,245.19	$ 369.07

Si bien los costos diarios pudieran no representar un costo significativo, es necesario evaluarlo de manera mensual y anual ya que los aireadores se encienden

a lo largo del año, para el caso de paletas corresponde a un ahorro de $278.78 anuales por aireador, $209.09 pesos para el caso de fuente y $738.14 para el caso de inyección.

4.4.7. Interpretación de los resultados

Después de un análisis exhaustivo del ahorro de energía y los beneficios económicos asociados con las propuestas generadas, es esencial profundizar en el impacto ambiental positivo que estas representan. La reducción del consumo energético conlleva una disminución significativa en la emisión de dióxido de carbono (CO2) a la atmósfera, lo que desempeña un papel crucial en la mitigación del cambio climático y sus efectos adversos. La Tabla 20 proporciona un análisis detallado de la emisión diaria, mensual y anual de CO2 para cada secuencia propuesta de aireadores.

Tabla 20. Evaluación de la emisión de gases de CO2 para cada secuencia. Fuente:(Elaboración propia).

Nombre Aireador	Consumo Diario	CO2 (Kg)	DIFERENCIA	Consumo Mensual	CO2 (Kg)	DIFERENCIA	Consumo Anual	CO2 (Kg)	DIFERENCIA
Paletas									
Paletas N	0.484	0.2217	0	14.52	6.65016	0	176.66	80.91028	0
Paletas 15	0.1694	0.0776	-0.1440868	5.082	2.327556	-4.3226	61.831	28.318598	-52.591682
Paletas 20	0.225867	0.1034	-0.1182249	6.776	3.103408	-3.54675	82.44133	37.7581305	-43.152149
Paletas 30	0.3872	0.1773	-0.0443344	11.616	5.320128	-1.33003	141.328	64.728224	-16.182056
Paletas 60	0.0968	0.0443	-0.1773376	2.904	1.330032	-5.32013	35.332	16.182056	-64.728224
Paletas 120	0.1936	0.0887	-0.1330032	5.808	2.660064	-3.9901	70.664	32.364112	-48.546168
Fuente									
Fuente N	1.452	0.665	0	43.56	19.95048	0	529.98	242.73084	0
Fuente 15	1.1616	0.532	-0.1330032	34.848	15.96038	-3.9901	423.984	194.184672	-48.546168
Fuente 20	1.1616	0.532	-0.1330032	34.848	15.96038	-3.9901	423.984	194.184672	-48.546168
Fuente 30	1.1616	0.532	-0.1330032	34.848	15.96038	-3.9901	423.984	194.184672	-48.546168
Fuente 60	1.1616	0.532	-0.1330032	34.848	15.96038	-3.9901	423.984	194.184672	-48.546168
Fuente 120	1.452	0.665	0	43.56	19.95048	0	529.98	242.73084	0
Inyección									
Inyección N	2.563	1.1739	0.952182	76.89	35.21562	0	935.495	428.45671	0
Inyección 15	2.69115	1.2325	1.0108747	80.7345	36.9764	1.760781	982.2698	449.879546	21.4228355
Inyección 20	1.5378	0.7043	0.4826404	46.134	21.12937	-14.0862	561.297	257.074026	-171.38268
Inyección 30	2.0504	0.9391	0.7174112	61.512	28.1725	-7.04312	748.396	342.765368	-85.691342
Inyección 60	3.0756	1.4086	1.1869528	92.268	42.25874	7.043124	1122.594	514.148052	85.691342
Inyección 120	3.5882	1.6434	1.4217236	107.646	49.30187	14.08625	1309.693	599.839394	171.382684

Al examinar los datos presentados en la tabla, se observan reducciones significativas en las emisiones de CO2 en comparación con el escenario base representado por los aireadores "N". Esta disminución es especialmente notable en ciertas secuencias de aireadores, donde se logra una reducción sustancial de las emisiones.

En primer lugar, se destacan los casos donde se produce una disminución significativa en la emisión de CO2 en comparación con los aireadores "N". Por ejemplo, en la secuencia "Inyección 15", se observa una reducción tanto en el consumo diario como en las emisiones de CO2 en comparación con el aireador base. Esta disminución se mantiene de manera consistente en los períodos mensuales y anuales, lo que indica una eficiencia sostenida a lo largo del tiempo.

En contraste, hay situaciones en las que las secuencias de aireadores muestran una producción de CO2 casi equivalente a la de los aireadores base. Este es el caso de las secuencias "Fuente 15", "Fuente 20", "Fuente 30", "Fuente 60" y "Fuente 120", donde las diferencias en las emisiones de CO2 son mínimas en comparación con los aireadores "N". Aunque no se observa una reducción significativa en estas situaciones, es importante tener en cuenta que el uso de estas secuencias alternativas todavía puede tener beneficios en términos de ahorro energético y económico.

Además, se identifican casos donde las secuencias de aireadores resultan en un aumento en las emisiones de CO2 en comparación con los aireadores "N". Por ejemplo, en la secuencia "Inyección 120", se observa un aumento tanto en el consumo diario como en las emisiones de CO2 en comparación con el aireador base. Esta tendencia se refleja tanto a nivel mensual como anual, lo que indica una eficiencia relativamente menor en términos de reducción de emisiones.

En resumen, la implementación de estrategias de gestión energética basadas en las secuencias de aireadores propuestas conlleva una reducción significativa en la emisión de gases de efecto invernadero, especialmente en aquellos casos donde se logra una disminución sustancial en comparación con los aireadores base. Estos

hallazgos subrayan la importancia y la eficacia de estas propuestas en la reducción del impacto ambiental en el contexto de la acuicultura moderna.

4.5. Conclusiones

Tras una exhaustiva exploración de los resultados obtenidos en el Capítulo 4, se desprenden varias conclusiones fundamentales que encapsulan la efectividad y viabilidad de las propuestas generadas por el algoritmo para la gestión energética de los aireadores en la granja. A continuación, se detallan las principales recapitulaciones:

A. Identificación de patrones de consumo: El minucioso análisis de los consumos diarios por aireador en intervalos de 5 minutos ha develado patrones distintivos que ofrecen una comprensión profunda de los comportamientos energéticos. La variación en los intervalos de tiempo ha demostrado tener un impacto directo en el consumo, evidenciado por las marcadas diferencias entre los distintos aireadores. Esta comprensión detallada es crucial para diseñar estrategias de optimización efectivas.
B. Visualización de las variaciones estacionales y tendencias a largo plazo: Las representaciones mensuales y anuales del consumo por aireador han destacado variaciones estacionales significativas, proporcionando una visión a largo plazo de la eficiencia energética. Estas gráficas de barras han servido como herramientas valiosas para identificar tendencias y tomar decisiones estratégicas informadas, permitiendo ajustes proactivos para maximizar la eficiencia operativa.
C. Optimización del tiempo encendido: La tabla detallada de tiempo encendido y consumo para cada aireador ha arrojado luz sobre las propuestas más eficientes en términos de ahorro energético. La evaluación meticulosa de estas diferencias ha destacado ciertos intervalos y patrones de funcionamiento como más rentables que otros, subrayando el considerable potencial de optimización en la gestión de la energía en la granja.
D. Integración de los datos en DataFrame: El uso unificado de un solo DataFrame ha proporcionado una herramienta efectiva para comparar y

analizar rápidamente el rendimiento de cada aireador. La inclusión de porcentajes de operación en comparación con el patrón original "N" ha facilitado una evaluación visual clara de la eficiencia de las propuestas, simplificando el proceso de toma de decisiones.

E. Evaluación de costos asociados y ahorros económicos: La evaluación exhaustiva de los costos asociados ha revelado ahorros económicos significativos asociados con las propuestas del algoritmo. Aunque los costos diarios pueden parecer inicialmente insignificantes, el análisis a lo largo del tiempo ha demostrado un ahorro acumulado considerable, brindando beneficios económicos sostenibles que mejoran la viabilidad financiera de la operación.
F. Análisis del impacto ambiental: Las propuestas no solo han generado ahorros económicos, sino que también han contribuido positivamente a la sostenibilidad ambiental al reducir las emisiones de gases de efecto invernadero. La correlación directa entre el uso eficiente de los aireadores y la disminución de emisiones subraya la importancia crítica de implementar estas propuestas desde una perspectiva ambiental, evidenciando un impacto ambiental positivo que promueve prácticas más sostenibles en la industria acuícola.

Dadas las afirmaciones anteriores, puede deducirse que la presente investigación ha proporcionado respuestas sólidas a las preguntas planteadas al inicio del proyecto, revelando importantes hallazgos sobre el papel de los algoritmos de aprendizaje automático en la mejora de la eficiencia energética en la acuicultura.

En primer lugar, la revisión sistemática de la literatura reveló una falta de enfoque específico en el ahorro de energía en los procesos de aireación en la acuicultura, utilizando algoritmos de aprendizaje automático. Sin embargo, durante la implementación de los experimentos, se constató la posibilidad de generar alternativas para eficientizar el consumo energético en dichos procesos.

En cuanto al uso de algoritmos de aprendizaje automático, se demostró que los algoritmos genéticos son herramientas poderosas para generar patrones de

encendido y apagado óptimos para los aireadores, lo que permite una gestión energética más eficiente y específica en términos de ahorro de energía.

Además, se confirmó la existencia de una amplia variedad de algoritmos en la acuicultura, aunque la mayoría se centra en la conservación y crecimiento de especímenes. Esto resalta una oportunidad de aplicación de algoritmos para el ahorro energético en este contexto.

Asimismo, se identificaron variables críticas en los estanques de crianza, como el pH, la temperatura, el oxígeno disuelto y los sólidos disueltos totales, así como factores ambientales como la radiación solar y la temperatura, que son cruciales para la oxigenación del agua y, por ende, para el bienestar de los organismos acuáticos criados en los estanques.

La recolección y almacenamiento eficiente de datos de los estanques se logró mediante el diseño de equipos que utilizan sensores de oxígeno disuelto y temperatura, permitiendo una recolección de datos con alta periodicidad y facilitando una mejor toma de decisiones en términos de gestión energética.

Aunque los algoritmos no clasifican información directamente, se utilizaron para generar etiquetas y analizar el comportamiento de las variables, con los algoritmos genéticos destacando por su capacidad para encontrar combinaciones óptimas de parámetros.

Finalmente, se evidenció que los algoritmos de aprendizaje automático permiten analizar grandes volúmenes de datos en un tiempo relativamente corto, facilitando el análisis exhaustivo de todas las posibles respuestas, especialmente en relación con el ahorro de energía.

En conclusión, esta investigación ha demostrado el potencial de los algoritmos de aprendizaje automático para mejorar la eficiencia energética en la acuicultura, ofreciendo soluciones prácticas y eficientes para la optimización del consumo energético en sistemas de aireación. Estos hallazgos tienen importantes implicaciones para la sostenibilidad y la rentabilidad de la industria acuícola, así como para la conservación del medio ambiente.

4.5.2. Trabajo futuro

Como propuestas para investigaciones futuras, se recomienda enfocarse en los siguientes aspectos:

1. Aplicación práctica y monitoreo continuo: Es fundamental implementar las nuevas estrategias de encendido y apagado de los aireadores, así como tal vez la combinación de los mismos, generadas a partir del algoritmo genético, en condiciones reales de operación. La implementación en el campo permitirá validar la eficacia de estas propuestas, proporcionando una valiosa retroalimentación que puede utilizarse para ajustar y mejorar las estrategias iniciales. Este enfoque práctico asegurará que las soluciones desarrolladas no solo sean teóricamente sólidas, sino también efectivas en un entorno real, adaptándose a las variaciones y desafíos operativos cotidianos.

2. Sistemas de monitoreo automatizado: Se sugiere la incorporación de sistemas de monitoreo automatizado que ajusten dinámicamente los patrones de funcionamiento de los aireadores en respuesta a cambios en las condiciones ambientales y operativas. Estos sistemas pueden utilizar tecnologías avanzadas como la inteligencia artificial y el aprendizaje automático para optimizar continuamente el consumo de energía. La implementación de estas tecnologías permitirá que los aireadores operen de manera eficiente bajo una variedad de condiciones, asegurando una respuesta rápida y precisa a cualquier cambio en el entorno operativo.

3. Evaluación de la adaptabilidad y sostenibilidad a largo plazo: Los futuros estudios deberían proporcionar información adicional sobre la adaptabilidad y la sostenibilidad a largo plazo de las estrategias propuestas. Es crucial asegurar una gestión eficiente de los recursos energéticos y una reducción continua del impacto ambiental. Evaluar cómo las estrategias de optimización energética se comportan a lo largo del tiempo permitirá ajustar y mejorar estas prácticas, asegurando que sean sostenibles y efectivas en el largo plazo, contribuyendo tanto a la eficiencia operativa como a la sostenibilidad ambiental.

4. Pruebas de diferentes tipos y marcas de aireadores: Es importante gestionar los recursos económicos necesarios para probar otros tipos y marcas de aireadores en diferentes condiciones operativas. Estas pruebas permitirán evaluar el rendimiento de diversos equipos en términos de eficiencia energética y calidad del agua. Comparar diferentes tecnologías y configuraciones ayudará a seleccionar las mejores opciones disponibles, adaptadas a las necesidades específicas de la granja acuícola, y permitirá tomar decisiones informadas sobre la inversión en nuevos equipos.

5. Uso de sensores avanzados: Se recomienda el uso de diferentes sensores a diversas distancias de la fuente de oxigenación y a diferentes profundidades del estanque. Estos sensores proporcionarán datos precisos y detallados que son cruciales para una evaluación completa del rendimiento de los aireadores. Realizar estas pruebas durante períodos más largos y continuos, siempre que las condiciones atmosféricas y técnicas lo permitan, garantizará que los datos recolectados sean representativos y útiles para ajustar y optimizar las estrategias de funcionamiento.

6. Desarrollo de nuevas tecnologías y configuraciones: Es esencial explorar cómo diferentes tecnologías y configuraciones pueden influir en la eficiencia energética y la calidad del agua. Investigar y desarrollar nuevas tecnologías permitirá identificar soluciones innovadoras que pueden mejorar significativamente la operación de los aireadores. Seleccionar de manera informada los equipos y estrategias de operación basados en los *insights* obtenidos de estas pruebas adicionales contribuirá a la optimización de los recursos energéticos y a la mejora general de la eficiencia operativa en la acuicultura.

Estas futuras investigaciones no solo proporcionarán información valiosa para mejorar la gestión energética en la acuicultura, sino que también contribuirán a la sostenibilidad ambiental y a la optimización de los recursos operativos en esta industria. Al continuar desarrollando y aplicando estrategias avanzadas de gestión energética, se podrá asegurar un futuro más eficiente y sostenible para la acuicultura global.

Bibliografía

1. *Acuicultura - Sociedad Nacional de Pesquería*. (2022, abril 6). Sociedad Nacional de Pesquería. https://www.snp.org.pe/acuicultura/
2. Acuicultura. (s/f). Proaqua.mx. Recuperado el 20 de abril de 2022, de https://proaqua.mx/acuicultura/
3. AGOSSOU, B. E. (2021). IoT & AI Based System to improve Fish Farming: Case study of Benin (Doctoral dissertation, Kobe Institute of Computing).
4. Akhter, F., Siddiquei, H. R., Alahi, M. E. E., & Mukhopadhyay, S. C. (2021). Recent advancement of the sensors for monitoring the water quality parameters in smart fisheries farming. Computers, 10(3), 26.
5. Almalki, F., Alsamhi, S. H., Sahal, R., Hassan, J., Hawbani, A., Rajput, N. S., ... & Breslin, J. (2021). Green IoT for eco-friendly and sustainable smart cities: future directions and opportunities. Mobile Networks and Applications, 1-25.
6. Álvarez, M., & Quevedo, J. (2005). Diseño y evaluación de un sistema de recirculación con fines didácticos y experimentales para la acuicultura. Revista de la Academia Canaria de Ciencias:= Folia Canariensis Academiae Scientiarum, 17(4), 57-72.
7. Arias Montero, D. A. (2021). Diseño, construcción, automatización y determinación de los parámetros de funcionamiento de un aireador difusor tornado para el sector camaronero.
8. Armijos Galarza, M. D. C., & Carriel Castro, A. G. (2021). Plan de exportación de aireadores con efecto Venturi al sector camaronero mexicano.
9. Asanza, W. R., & Olivo, B. M. (2018). Redes neuronales artificiales aplicadas al reconocimiento de patrones. Editorial UTMACH.
10. Ayala, I. S. D. A. (8 de octubre de 2021). *LOS PECES: DE LA ACUICULTURA A LA BIOMEDICINA*. https://www.um.es/acc/wp-content/uploads/DISCURSOS-INVESTIDURA-DE-ALFONSA.pdf
11. Barrientos Mogollon, E. S., & Mamani Mamani, S. A. (2019). Modelos de aprendizaje supervisado como apoyo a la toma de decisiones en las organizaciones basados en datos de redes sociales: Una revisión sistemática de la literatura.

12. Blancaflor, E., & Baccay, M. (2021, July). Design of a solar powered IoT (internet of things) remote water quality management system for a biofloc aquaculture technology. In 2021 3rd Blockchain and Internet of Things Conference (pp. 24-31).
13. Blokdyk, G. (2018). IBM docs: Complete self-assessment guide. Createspace Independent Publishing Platform.
14. Bolaños Castro, S. J. (2003). Captura de conocimiento a través de modelamiento. Ingeniería, 9(1), 32–36. https://doi.org/10.14483/23448393.2739
15. Bryan, C. (2021). Data Acquisition for Water Quality Control.
16. BURBANO, C. E. V., BOLAÑOS, J. M. B., & DE POPAYÁN, F. U. (2019) IMPLEMENTACIÓN Y EVALUACIÓN DE TÉCNICAS DE CLUSTERING PARA UN SISTEMA DE RECUPERACIÓN DE DOCUMENTOS JURISPRUDENCIALES.
17. Caparrini, F. S., & Windmill Web Work. (2019, noviembre 4). Algoritmos genéticos. Cs.Us.Es. http://www.cs.us.es/~fsancho/?e=65
18. Carrillo, D. A. P., Corredor, Z., & Ramírez-Iglesia, L. (2014). Características físico-químicas y morfométricas en la crianza por fases de la tilapia roja (Oreochromis spp.) en una zona cálida tropical. ZOOTECNIA TROPICAL, 30(1).
19. Castillo Valle, M. C. (2021). Manejo de tilapia (Oreochromis niloticus) en la granja demostrativa de cultivo de peces de la Universidad Nacional Agraria Septiembre, marzo 2019-2020 (Doctoral dissertation, Universidad Nacional Agraria).
20. CEDRSSAR. (junio de 2015). *La acuacultura*.
21. Centro de Investigación en Alimentación y Desarrollo, A.C. (2008). Manual de Buenas Prácticas de Producción Acuícola de Tilapia para la Inocuidad Alimentaria. Mazatlán: SAGARPA.
22. Chang, C. C., Wang, J. H., Wu, J. L., Hsieh, Y. Z., Wu, T. D., Cheng, S. C., ... & Lin, C. Y. (2021). Applying Artificial Intelligence (AI) Techniques to

Implement a Practical Smart Cage Aquaculture Management System. Journal of Medical and Biological Engineering, 41(5), 652-658.

23. Chapman, P. (2000). CRISP-DM, 1.0, Step-by-step Data Mining Guide, 2000.
24. Chiok, C. H. M., & Gallardo, J. C. (2021). Caracterización de las regiones del Perú según los indicadores de género aplicando métodos de agrupamiento. Tierra nuestra, 15(1), 84-94.
25. Coe, M., & Gutschmidt, S. (2021). Computer Vision Estimation of Physical Parameters and Its Application to Power Requirements of Natural and Artificial Swimmers. Designs, 5(4), 69.
26. Colla, P. E. (2016, November). Uso de opciones reales para evaluar la contribución de metodologías KANBAN en desarrollo de software. In Simposio Argentino de Ingeniería de Software (ASSE 2016)-JAIIO 45 (Tres de Febrero, 2016)
27. Comisión Nacional de Acuacultura y Pesca. (2023, 30 de noviembre). Coloca Gobierno de México a la acuicultura mexicana en posición prioritaria y estratégica. Gobierno de México. Recuperado de https://www.gob.mx/conapesca/prensa/coloca-gobierno-de-mexico-a-la-acuicultura-mexicana-en-posicion-prioritaria-y-estrategica?idiom=esConogasi. (2018, septiembre 21). *Algoritmos genéticos* . https://conogasi.org/articulos/algoritmos-geneticos/
28. Correa Tucumango, J. C., & Mires Briceño, L. D. (2020). Influencia de la implementación de las herramientas de minería de datos Weka y Tableau Public en la toma de decisiones en la clínica San Francisco de la ciudad de Cajamarca, 2019.
29. Cruz Bautista, M. F. (2021). Desarrollo de un prototipo de cultivo acuícola basado en el aprovechamiento del recurso hídrico con sistema de monitoreo de parámetros fisicoquímicos para la etapa de engorde de Tilapia roja (Oreochromis sp) en la Sabana de Bogotá.
30. De la Oliva, G. (2011). Manual de buenas prácticas de producción acuícola en el cultivo de trucha arco iris. Recuperado el, 20.

31. De Oliveira, E. F., de Lima Tostes, M. E., de Freitas, C. A. O., & Leite, J. C. (2017). Voltage THD analysis using knowledge discovery in databases with a decision tree classifier. Ieee Access, 6, 1177-1188.
32. Delgado Lopez, R. P. (2018). Empresas, equipos y materiales en acuicultura.
33. Delgado Tapia, C. E., & Valencia Astudillo, W. G. (2021). Diseño e implementación de prototipo IOT para el monitoreo remoto de la calidad del agua para la crianza de tilapias en estanques (Bachelor's thesis).
34. Delgado Tapia, C. E., & Valencia Astudillo, W. G. (2021). Diseño e implementación de prototipo IOT para el monitoreo remoto de la calidad del agua para la crianza de tilapias en estanques (Bachelor's thesis).
35. Dhal, S. B., Jungbluth, K., Lin, R., Sabahi, S. P., Bagavathiannan, M., Braga-Neto, U., & Kalafatis, S. (2022). A Machine-Learning-Based IoT System for Optimizing Nutrient Supply in Commercial Aquaponic Operations. Sensors, 22(9), 3510.
36. Dias, L. P. S., Barbosa, J. L. V., & Vianna, H. D. (2018). Gamification and serious games in depression care: A systematic mapping study. Telematics and Informatics, 35(1), 213-224.
37. DIFERENCIA ENTRE DATO, INFORMACIÓN Y CONOCIMIENTO. (s/f). Unam.mx. Recuperado el 23 de abril de 2022, de https://iibi.unam.mx/voutssasmt/documentos/dato%20informacion%20conocimiento.pdf
38. El Productor, (2017a, agosto 18). Etapas del cultivo de tilapia en estanques rústicos. El productor. https://elproductor.com/2017/08/etapas-del-cultivo-de-tilapia-en-estanques-rusticos/
39. *FAO fisheries & aquaculture*. (s/f). Fao.Org. Recuperado el 20 de abril de 2022, de https://www.fao.org/fishery/es/aquaculture
40. FAO. (2022). Acuicultura. Obtenido de Organización de las naciones unidas para la agricultura y la alimentación.: https://www.fao.org/aquaculture/es/
41. Félix, F., & Manuel, B. (2018). Validación interna de modelos predictivos de regresión logística. Comando Validation (Stata).

42. Fideicomiso de Riesgo Compartido. (1 de Marzo de 2017). Producción de Tilapia a través de la Acuacultura. Obtenido de Gobierno de México.: https://www.gob.mx/firco/articulos/produccion-de-tilapia-a-traves-de-la-acuacultura?idiom=es#:~:text=La%20tilapia%20o%20mojarra%20es,en%20el%20pa%C3%ADs%20desde%201964.
43. Fragoso Cervón, M., & Auró de Ocampo, A. (2005). *UNIDAD 9 ZOOTECNIA ACUÍCOLA*.https://fmvz.unam.mx/fmvz/p_estudios/apuntes_zoo/unidad_9_zootecniaacuicola.pdf
44. Galán Cortina, V. (2016). Aplicación de la metodología CRISP-DM a un proyecto de minería de datos en el entorno universitario (Bachelor's thesis).
45. Galli Merino, O., & Miguel Sal, F. (2007). Sistemas de recirculación y tratamiento de agua. Argentina: Secretaría de Agricultura, Ganadería, Pesca y Alimentos (CENADAC).
46. Garcés Toro, B. S., & Ramos Sotomayor, J. P. (2020). Prototipo de un sistema de control para optimizar el consumo eléctrico de una bomba y Blowers de una planta de tratamiento de aguas residuales (Bachelor's thesis, Quito: Universidad de las Américas, 2020).
47. García Osorio, G. E. (2018). Evaluación de las características fisicoquímicas del agua en la piscícola de Asojuncal-Huila, asociados al ciclo de producción de la Tilapia roja.
48. García, J., Molina, J., Berlanga, A., Patricio, M., Bustamante, A., & Padilla, W. (2018). Ciencia de datos. Técnicas Analíticas y Aprendizaje Estadístico. Bogotá, Colombia. Publicaciones Altaria, SL.
49. Garduño Juárez, R. (2018, 21 de septiembre). Algoritmos genéticos. Conogasi, Conocimiento para la vida. Consultado el 15 de noviembre de 2023, de URL del artículo.
50. Ghandar, A., Ahmed, A., Zulfiqar, S., Hua, Z., Hanai, M., & Theodoropoulos, G. (2021). A decision support system for urban agriculture using digital twin: A case study with aquaponics. IEEE Access, 9, 35691-35708.
51. GOMEZ FUENTES, M. D. C. (2013). Notas del curso: bases de datos.

52. González Rugel, J. C. (2019). Diseño e implementación de un sistema de monitoreo y control utilizando el controlador GSM BR160SM mediante mensajes de texto en una piscina de tilapia ubicada en la parroquia Taura, para la empresa Dipromacom (Bachelor's thesis).
53. Gonzaléz, F. (s/f). .: *PisciculturaGlobal :* Pisciculturaglobal.com., 2019, de https://www.pisciculturaglobal.com/indicadores-de-produccion-parametros-fisicos-y-quimicos-del-agua-para-tilapias/
54. Guarascio, M., Manco, G., & Ritacco, E. (2018). Knowledge discovery in databases. Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics, 336.
55. Hernandez Barraza, C. A., Aguirre Guzman, G., & Lopez Cantu, D. G. (2009). Sistemas de producción de acuiculturacon recirculación de agua para la región norte, noreste y noroeste de México. Revista mexicana de agronegocios, 25(1345-2016-104263), 117-130.
56. Hernández Ospina, J. D. (2019). Evaluación de sistemas de aireación para transferencia de oxígeno en aguas subterráneas.
57. Holland, J. H. (1975). Adaptation in Natural and Artificial System s. MIT Press
58. Hu, W. C., Chen, L. B., Huang, B. K., & Lin, H. M. (2022). A Computer Vision-Based Intelligent Fish Feeding System Using Deep Learning Techniques for Aquaculture. IEEE Sensors Journal, 22(7), 7185-7194.
59. Instituto Nacional de la Economía Social. (27 de Agosto de 2018). Acuicultura, historia y actualidad en México. Obtenido de Gobierno de México: https://www.gob.mx/inaes/es/articulos/acuicultura-historia-y-actualidad-en-mexico?idiom=es#:~:text=La%20producci%C3%B3n%20acu%C3%ADcola%20en%20M%C3%A9xico,los%20cuales%20el%2070%25%20de
60. Instituto Nacional de Pesca. (21 de marzo de 2018). AcuiculturaTilapia. Obtenido de Instituto Nacional de Pesca: https://www.gob.mx/inapesca/acciones-y-programas/acuacultura-tilapia

61. Instituto Nacional de Pesca. (s/f). AcuiculturaTilapia. gob.mx. 21 de marzo 2018, de https://www.gob.mx/inapesca/acciones-y-programas/acuacultura-tilapia
62. INTEGRATED, M. (2019). DS18B20: Programmable Resolution 1-Wire Digital Thermometer. 2014.
63. Jiménez, J. (2012). Sistemas de recirculación en acuicultura: Una visión y retos diversos para Latinoamérica. Industria Acuícola, 8(2), 6-10. Obtenido de http://www.industriaacuicola.com/PDFs/Sistemas_de_recirculacion.pdf
64. Khudoyberdiev, A., Ahmad, S., Ullah, I., & Kim, D. (2020). An optimization scheme based on fuzzy logic control for efficient energy consumption in hydroponics environment. Energies, 13(2), 289.
65. Kuang, L., Shi, P., Hua, C., Chen, B., & Zhu, H. (2020). An Enhanced Extreme Learning Machine for Dissolved Oxygen Prediction in Wireless Sensor Networks. IEEE Access, 8, 198730-198739.
66. Lazo, M. A., Geronimo, L. M. K., Comilang, L. J., Cayme, K. J. B., Ventura, J. M., & Abana, E. C. (2021). AQUACISION: a multiparameter aquaculture water quality tester and decision support system. Indonesian Journal of Electrical Engineering and Computer Science, 24(1), 530-537.
67. Lemus-Delgado, D., & Pérez Navarro, R. (2020). Ciencia de datos y estudios globales: aportaciones y desafíos metodológicos. Colombia Internacional, (102), 41-62.
68. Li, D., Sun, J., Yang, H., & Wang, X. (2020). An enhanced naive bayes model for dissolved oxygen forecasting in shellfish aquaculture. IEEE Access, 8, 217917-217927.
69. Li, D., Wang, X., Sun, J., & Feng, Y. (2021). Radial basis function neural network model for dissolved oxygen concentration prediction based on an enhanced clustering algorithm and Adam. IEEE Access, 9, 44521-44533.
70. Liu, S., Xu, L., Li, Q., Zhao, X., & Li, D. (2018). Fault diagnosis of water quality monitoring devices based on multiclass support vector machines and rule-based decision trees. IEEE Access, 6, 22184-22195.

71. Luchini, L. (2010). LA ACUICULTURA, SUS MODELOS Y EL POTENCIAL ACTUAL DEL PEJERREY COMO PEZ DE CULTIVO. Obtenido de Sitio argentino de producción animal: https://www. agroindustria. gob. ar/sitio/areas/acuicultura/publicaciones/_a rchivos//000000_Informaci% C3% B3n% 20y% 20noticias% 20vinculadas% 20al% 20sector/101118_La% 20acuicultura,% 20sus% 20modelos% 20y% 20el% 20poten cial% 20actual, 20.
72. Mancilla-Vela, G. L.-G.-O.-S. (2020). Factors associated to student success in online learning: a data mining analysis. Formacion Universitaria, 13(6),, 23-36.
73. Marinho-Pereira, T., Junior, C. H. F., Rincón, L. M. G., Britto, E. N., Cavero, B. A. S., Aride, P. H. R., & de Oliveira, A. T. (2020). Tecnología biofloc: datos, estudios y experiencias para el desarrollo de la acuicultura latinoamericana. Brazilian Journal of Development, 6(2), 7847-7862.
74. Martínez Quinchanegua, F. L., & Bertel López, Y. Y. (2021). Prototipo electrónico de medición y monitoreo remoto, de la calidad del agua en criaderos de Tilapia en estanques de tierra.
75. Martínez, G. R. S., & Mejía, J. A. S. (2011). Arboles de decisiones en el diagnóstico de enfermedades cardiovasculares. Scientia et technica, 16(49), 104-109.
76. Martínez, M. V. E. (2022). *Desarrollo y evaluación in vitro de un implante hormonal de liberación prolongada para la inducción del desove en peces*. Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California.
77. Mayorga Márquez, B. J. (2018). Pronóstico espacial de demanda eléctrica mediante la técnica de agrupamiento (clustering) de curvas S históricas–Aplicación a la Empresa Eléctrica Ambato Regional Centro Norte SA (Bachelor's thesis, Quito, 2018.).
78. Minewiskan. (s/f). Validación cruzada (Analysis Services - Minería de datos). Microsoft.com. Recuperado el 6 de junio de 2022, de

https://docs.microsoft.com/es-es/analysis-services/data-mining/cross-validation-analysis-services-data-mining?view=asallproducts-allversions.

79. Moine, J. M. (2013). Metodologías para el descubrimiento de conocimiento en bases de datos: un estudio comparativo (Doctoral dissertation, Universidad Nacional de La Plata).
80. Molina Félix, L. C. (2002). Data mining: torturando a los datos hasta que confiesen. Barcelona: UOC.
81. Mompó Serrano, A. (2022). Usos de la Ciencia de Datos aplicados al sector agrícola (Doctoral dissertation, Universitat Politècnica de València).
82. Montalvo García, J. F. (2021). CRISP-DM/SMES: una metodología de proyectos de analítica de datos para las PYME.
83. Morales, N. (s/f). La Acuicultura en el mundo. AgroClick - Blog. Recuperado el 20 de abril de 2022, de https://agroclick.org/blog/la-acuicultura-en-el-mundo
84. Moran, M. (2020, 17 junio). Océanos - Desarrollo Sostenible. Desarrollo Sostenible. https://www.un.org/sustainabledevelopment/es/oceans/
85. Moya-Rodríguez, Jorge Laureano, Becerra-Ferreiro, Ana María, & Chagoyén-Méndez, César A.. (2012). Utilización de Sistemas Basados en Reglas y en Casos para diseñar transmisiones por tornillo sinfín. Ingeniería Mecánica, 15(1), 01-09. Recuperado en 08 de marzo de 2022, de http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1815-59442012000100001&lng=es&tlng=es.
86. Muedas Alvarez, D. (2021). Efecto del tamaño de burbuja de aire, en la eficiencia de difusores para transferir oxígeno al agua.
87. Muñoz, G. T. (2019). Controlador PID con algoritmos genéticos de números reales. Industrial data, 22(2), 213-223.
88. Murali, S., Krishnan, V. S., Amulya, P. R., Alfiya, P. V., Delfiya, D. A., & Samuel, M. P. (2021). Energy and water consumption pattern in seafood processing industries and its optimization methodologies. Cleaner Engineering and Technology, 4, 100242.

89. Nava Pintor, J. A. (2020). Aplicación de Sistemas Embebidos e IoT para el Monitoreo de oxígeno disuelto en Estanques Acuícolas en Eldorado, Sinaloa.
90. Navarrete Cuellar, R. M. (2018). Evaluación del sistema de tratamiento de aguas residuales en la Unidad Lagunas del Este, Cayo Santa María (Doctoral dissertation, Universidad Central “Marta Abreu” de Las Villas. Facultad de Construcciones. Departamento de Ingeniería Hidráulica).
91. NCR, P. C., Clinton, J., NCR, R. K., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (1999). CRISP-DM 1.0.
92. Núñez, H., Vargas, R., Guerra, I., & Ramos, E. (2021). Efecto de la temperatura sobre el consumo de oxígeno en tilapias del Nilo con diferentes fenotipos de comportamiento. Centros: Revista Científica Universitaria, 10(2), 1-18.
93. Ochoa, Leticia. (2019). Evaluación de Algoritmos de Clasificación utilizando Validación Cruzada. 10.18687/LACCEI2019.1.1.471.
94. Oldan, B. R. M. (2021). A System for Water Quality Monitoring at Taal Lake with Alert Warning and Aeration System using Arduino. Review of International Geographical Education Online, 11(10), 1411-1422.
95. OVIEDO-LOPERA, J. C., OVIEDO-CARRASCAL, A. I., CARMONA-RODRIGUEZ, C. S., VELEZ-SALDARRIAGA, G. L., & REINA-ALZATE, J. (2020) Diseño de un sistema acuapónico monitoreado mediante internet de las cosas e inteligencia artificial.
96. Pablo, A. V. J., Daniel, C. R., Alfonso, Á. G. C., & Rocio, L. H. DESEMPEÑO DE LA PROGENIE DE SUPERMACHOS DE TILAPIA DEL NILO Oreochromis niloticus (L.) BAJO CONDICIONES COMERCIALES. Producción y Manejo de los Recursos Acuáticos en el Trópico, 100.
97. Palconit, M. G. B., Concepcion, R. S., Alejandrino, J. D., Fonseca, V. F., Sybingco, E., Bandala, A. A., ... & Dadios, E. P. (2021, December). IoT-based On-demand Feeding System for Nile Tilapia (Oreochromis niloticus). In TENCON 2021-2021 IEEE Region 10 Conference (TENCON) (pp. 698-702). IEEE.

98. Palconit, M. G. B., Concepcion, R. S., Tobias, R. R., Alejandrino, J., Almero, V. J. D., Bandala, A. A., ... & Dadios, E. P. (2021, November). Development of IoT-based Fish Tank Monitoring System. In 2021 IEEE 13th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM) (pp. 1-6). IEEE.
99. Paré, R. C., Costa, D. C., & Escofet, C. M. (2002). Bases de datos. UOC, la universidad virtual.
100. Pemberthy, L. A. P., & Ruiz, N. E. P. (2019). Diseño e implementación de un sistema de monitoreo de parámetros de calidad de agua en cultivo de tilapia en una granja piscícola del departamento del Cauca. Publicaciones e Investigación, 13(2), 11-22.
101. PENTAIR, (s/f). Sweetwater® regenerative blowers. Recuperado el 6 de junio de 2022, de https://pentairaes.com/regenerative-blowers-aquaculture-duty.html
102. Perdomo Carrillo, D. A., Corredor, Z., & Ramírez-Iglesia, L. (2012). Características físico-químicas y morfométricas en la crianza por fases de la tilapia roja (Oreochromis spp.) en una zona cálida tropical. *Zootecnia Tropical*, *30*(1), 099-108.
103. Pineda-Jaramillo, J. D. (2019). Una revisión de los algoritmos de Machine Learning (ML) utilizados para la modelación de la elección de modo de viaje/A review of Machine Learning (ML) algorithms used for modeling travel mode choice. Dyna, 86(211), 32.
104. Piñeros-Roldan, A. J., Gutiérrez-Espinosa, M. C., Lapa-Viana, M., & Coelho-Emerenciano, M. G. (2020). Aireación en la tecnología Biofloc (BTF): principios básicos, aplicaciones y perspectivas. Revista Politécnica, 16(31), 29-40.
105. Platas-Rosado, D. E. (2017). Importancia económico y social del sector acuícola en México. Agro Productividad, 10(2).

106. Pylianidis, C., Osinga, S., & Athanasiadis, I. N. (2021). Introducing digital twins to agriculture. Computers and Electronics in Agriculture, 184, 105942.
107. Ramírez Chalén, V. (2019). Proyecto de inversión y desarrollo para aireadores y comederos automáticos para el sector camaronero del cantón Durán (Master's thesis, Guayaquil: ULVR, 2019.).
108. Rashid, M. M., Nayan, A. A., Simi, S. A., Saha, J., Rahman, M. O., & Kibria, M. G. (2021). IoT based smart water quality prediction for biofloc aquaculture. Int. J. Adv. Comput. Sci. Appl., 12(6).
109. Rivera Betancur, L. H. Solución IoT para la optimización del proceso de piscicultura en el Centro de Desarrollo Agroalimentario El Limonal.
110. Rodríguez, M. R., Besteiro, R., Arango, T., Ortega, J., Fernández, M. D., de Compostela, S., ... & Fernández, M. D. (2020). Analyzing and Modeling Environmental and Production Variables in Weaned Piglet Farms. Prime Arch. Agric. Res, 1-28.
111. Rostam, N. A. P., Hassain Malim, N. H. A., & Abdullah, R. (2020, August). Development of a low-cost solar powered & real-time water quality monitoring system for Malaysia seawater aquaculture: Application & challenges. In Proceedings of the 2020 4th International Conference on Cloud and Big Data Computing (pp. 86-91).
112. Saavedra Martínez, M. A. (2006). Manejo del cultivo de tilapia.
113. Saavedra Torres, R. (2019). Uso de una red de sensores para el monitoreo en tiempo real de la calidad del agua en los estanques de alevinos de tilapia de la estación pesquera Ahuashiyacu-Tarapoto.
114. Saavedra Vicente, M. (2022). Crecimiento de tilapia nilótica Oreochromis niloticus (Linnaeus, 1758) a dos densidades en la etapa de pre-cría cultivado bajo el sistema de Biofloc con dos fuentes de carbono, Piura-Perú 2019.
115. Sambo, D. W. (2021). Design of a Wireless Underground Sensor Network for Precision Agriculture (Doctoral dissertation, Université de Ngaoundéré (Cameroun)).

116. SANCHEZ, C. C., GIRALDO, L. M., PIEDRAHITA, C. C., BONET, I., LOCHMULLER, C., TABARES, M. S., & PEÑA, A. (2018). Análisis comparativo entre:«el análisis exploratorio de datos» y los modelos de «árboles de decisión» y «k-means» en el diagnóstico de la malignidad en algunos exámenes de cáncer de mama. Un estudio de caso. Revista Espacios, 39(28).

117. Sánchez-Ramos, C. A., Jiménez-Rodríguez, D. J., Ruiz-Rodríguez, C. J., Márquez-Rocha, F. J., & Flores-Vela, A. I. Análisis de eficiencia energética en una granja de producción acuícola.

118. Sánchez Trujillo, M. G., & Pérez Hernández, J. Á. (2023). Metodología CRISP-DM en la gestión de proyecto de Data Mining. Caso enfermedades dermatológicas.

119. Schab, E. A., Rivera, R. A., Bracco, L. J., Coto, F., Cristaldo, P. R., Ramos, L. M. M., ... & Herrera, N. E. (2018). Minería de datos y visualización de información.

120. Secretaría de Agricultura y Desarrollo Rural, (s/f). La acuiculturay su paso por el tiempo. gob.mx. Recuperado el 20 de abril de 2022, de https://www.gob.mx/agricultura/es/articulos/descubre-desde-cuando-se-practica-la-acuacultura

121. Shi, X., An, X., Zhao, Q., Liu, H., Xia, L., Sun, X., & Guo, Y. (2019). State-of-the-art internet of things in protected agriculture. Sensors, 19(8), 1833.

122. Tariq, H. I., Sohail, A., Aslam, U., & Batcha, N. K. (2019). Loan default prediction model using sample, explore, modify, model, and assess (SEMMA). Journal of Computational and Theoretical Nanoscience, 16(8), 3489-3503.

123. Tawfeeq, A., Al Wahaibi, H. A. S., & Vijayalakshmi, K. (2019). IoT based Aquaculture system with Cloud analytics. International Journal of Applied Engineering Research, 14(22), 4136-4142.

124. TECNOACUA. (2020) Catalogo de Aireadores de Paleta Diva. Recuperado:6 de junio de 2022 de http://www.tecnoacua.com.ec/CATALOGO.pdf

125. Teixeira, R. R., Puccinelli, J. B., Poersch, L., Pias, M. R., Oliveira, V. M., Janati, A., & Paris, M. (2021). Towards Precision Aquaculture: A High Performance, Cost-effective IoT approach. arXiv preprint arXiv:2105.11493.

126. Timarán-Pereira, S. R., Hernández-Arteaga, I., Caicedo-Zambrano, S. J., Hidalgo-Troya, A. y AlvaradoPérez, J. C. (2016). El proceso de descubrimiento de conocimiento en bases de datos. En Descubrimiento de patrones de desempeño académico con árboles de decisión en las competencias genéricas de la formación profesional (pp. 63-86). Bogotá: Ediciones Universidad Cooperativa de Colombia. doi: http://dx.doi.org/10.16925/9789587600490

127. Torres-Tadeo, C. M., Platas-Rosado, D. E., & Tadeo-Castillo, C. I. (2021). Analysis of the Tilapia (Oreochromis spp.) Value Chain in the State of Veracruz Rural Aquaculture for the Small Producer. Agro Productividad, 14(4).

128. Valdés Santiago, D., Ramis Andalia, R. M., & Pría Barros, M. D. C. (2020). Métodos y desafíos en la medición de desigualdades sociales en salud de Cuba. Revista Cubana de Salud Pública, 46, e1753.

129. Valdez Espinoza, Y., & Felix Aguilar, C. D. (2019). Utilizacion del auacate regional para la modificación de la calidad del filete de Tilapia (Oreochromis niloticus).

130. Velez Zambrano, J. G. (2022). Modelo de aprendizaje para clasificación de correos electrónicos.

131. Viamontes, D. S., Batistas, A. S., & Solán, O. G. (2019). Riesgo, vulnerabilidad e incertidumbre en la acuicultura. Revista Cubana de Finanzas y Precios, 3(1), 102-113.

132. Yao, Y. C., Huang, X. R., Liu, C. Y., & Lin, P. W. (2020, December). Intelligent Controlling System in Aquaculture. In Proceedings of the 2020

ACM International Conference on Intelligent Computing and its Emerging Applications (pp. 1-2).

133. Yue, K., & Shen, Y. (2021). An overview of disruptive technologies for aquaculture. Aquaculture and Fisheries.

134. Yumpu.com. (2013). Diagrama de flujo general para los procesos de producciÃ3n acuÃ¬cola. yumpu.com. Recuperado 13 de octubre de 2022, de https://www.yumpu.com/es/document/view/14486386/diagrama-de-flujo-general-para-los-procesos-de-produccion-acuicola

135. Zhang, Z., Cao, S., & Wang, Y. (2019). A long-range 2.4 G network system and scheduling scheme for aquatic environmental monitoring. Electronics, 8(8), 909.

136. 정대현. (2020). Development of Artificial Intelligence-based Climate Control System for Smart Greenhouse (Doctoral dissertation, 서울대학교 대학원).

137. Valdez Espinoza, Y., & Felix Aguilar, C. D. (2019). Utilizacion del aguacate regional para la modificación de la calidad del filete de Tilapia (Oreochromis niloticus).

138. Woynarovich, E., & Romero, M. I. (2004). Conceptos básicos de piscicultura tropical. CEAM, Centre d'Estudis Amazònics.

139. *(FAO,2022). FAO - Oreochromis niloticus*. (s. f.). https://www.fao.org/fishery/docs/DOCUMENT/aquaculture/CulturedSpecies/file/es/es_niletilapia.htm

140. *(FAO,2022). FAO: Tilapia del Nilo - Página principal*. (s. f.). https://www.fao.org/fishery/affris/perfiles-de-las-especies/nile-tilapia/tilapia-del-nilo-pagina-principal/es/

141. Romana-Eguia, M. R. R., Eguia, R. V., & Pakingking Jr, R. (2020). *Tilapia culture: The basics*. Aquaculture Department, Southeast Asian Fisheries Development Center.

142. Pulido, H. G., De la Vara Salazar, R., González, P. G., Martínez, C. T., & Pérez, M. D. C. T. (2012). Análisis y diseño de experimentos. New York, NY, USA:: McGraw-Hill.
143. Gabriel, J., Castro, C., Valderde, A., & Indacochea, B. (2020). Diseños Experimentales.
144. Melo Martínez, O. O., López Pérez, L. A., & Melo Martínez, S. E. (2007). Diseño de experimentos: métodos y aplicaciones.
145. Hernández, I. J. G. (2014). Diseño de experimentos y su aplicación en la industria. Ingenio y Conciencia Boletín Científico de la Escuela Superior de Cd. Sahagun, 1(1).

Anexos

Datasets

Anexo 1. Daset iniciales

Aireador de paletas

```
<bound method NDFrame.describe of              Dia      Hora         OD      T
0      12/09/2023  20:46:48  13.222333  31.87
1      12/09/2023  20:46:49  13.209000  31.87
2      12/09/2023  20:46:50  13.209000  31.87
3      12/09/2023  20:46:51  13.222333  31.87
4      12/09/2023  20:46:52  13.235667  31.87
...           ...       ...        ...    ...
28885  13/09/2023  04:48:13   6.219667  30.69
28886  13/09/2023  04:48:14   6.219667  30.69
28887  13/09/2023  04:48:15   6.246333  30.69
28888  13/09/2023  04:48:16   6.246333  30.69
28889  13/09/2023  04:48:17   6.246333  30.69

[28890 rows x 4 columns]>
```

```
1 df.dtypes
```

```
Dia      object
Hora     object
OD      float64
T       float64
dtype: object
```

Aireador de fuente

```
<bound method NDFrame.describe of              Dia      Hora       OD      T
0      14/09/2023  19:54:51  10.7390  32.75
1      14/09/2023  19:54:52  10.5150  32.81
2      14/09/2023  19:54:53  10.4355  32.81
3      14/09/2023  19:54:54  10.3560  32.81
4      14/09/2023  19:54:55  10.3160  32.81
...           ...       ...      ...    ...
33054  15/09/2023  05:05:45   1.5470  31.78
33055  15/09/2023  05:05:46   1.5470  31.81
33056  15/09/2023  05:05:47   1.5470  31.75
33057  15/09/2023  05:05:48   1.5470  31.75
33058  15/09/2023  05:05:49   1.5470  31.75

[33059 rows x 4 columns]>
```

```
1 df.dtypes
```

```
Dia      object
Hora     object
OD      float64
T       float64
dtype: object
```

Aireador de inyección

```
<bound method NDFrame.describe of              Dia      Hora     OD      T
0      14/09/2023  20:50:01  8.872  32.81
1      14/09/2023  20:50:03  8.872  32.75
2      14/09/2023  20:50:04  8.792  32.75
3      14/09/2023  20:50:06  8.792  32.81
4      14/09/2023  20:50:07  8.872  32.75
...           ...       ...    ...    ...
18745  15/09/2023  05:05:42  1.547  31.75
18746  15/09/2023  05:05:44  1.547  31.75
18747  15/09/2023  05:05:46  1.547  31.81
18748  15/09/2023  05:05:47  1.547  31.75
18749  15/09/2023  05:05:49  1.547  31.75

[18750 rows x 4 columns]>
```

```
1 df.dtypes
```

```
Dia       object
Hora      object
OD       float64
T        float64
dtype: object
```

Anexo 2. Dataset combiando

Aireador paletas

```
<bound method NDFrame.describe of                Dia      Hora        OD          T Funcionamiento
0       01/08/2023  20:46:48  7.173149  32.225158      Encendido         NaN
1       01/08/2023  20:46:49  7.148418  32.200779      Encendido         NaN
2       01/08/2023  20:46:50  7.146048  32.315514      Encendido         NaN
3       01/08/2023  20:46:51  7.154250  32.165144      Encendido         NaN
4       01/08/2023  20:46:52  7.143331  32.152811      Encendido         NaN
...            ...       ...       ...        ...            ...         ...
866695  31/08/2023  04:48:13  3.915714  30.074282      Encendido         NaN
866696  31/08/2023  04:48:14  3.927257  30.085309      Encendido         NaN
866697  31/08/2023  04:48:15  3.937438  30.055138      Encendido         NaN
866698  31/08/2023  04:48:16  3.926967  30.087391      Encendido         NaN
866699  31/08/2023  04:48:17  3.902791  30.240362      Encendido         NaN

[866700 rows x 6 columns]>
```

```
1 df.dtypes
```

```
Dia               object
Hora              object
OD               float64
T                float64
Funcionamiento    object
```

Aireador fuente

```
<bound method NDFrame.describe of                Dia      Hora        OD          T Funcionamiento
0       01/07/2023  20:08:02  5.726880  32.786305      Encendido
1       01/07/2023  20:08:03  5.810512  32.537115      Encendido
2       01/07/2023  20:08:04  5.757634  32.563943      Encendido
3       01/07/2023  20:08:05  5.783522  32.487318      Encendido
4       01/07/2023  20:08:06  5.777710  32.590412      Encendido
...            ...       ...       ...        ...            ...
972445  31/07/2023  05:08:12  2.482182  31.349459        Apagado
972446  31/07/2023  05:08:13  2.482042  31.419617        Apagado
972447  31/07/2023  05:08:14  2.480568  31.506892        Apagado
972448  31/07/2023  05:08:15  2.484612  31.515514        Apagado
972449  31/07/2023  05:08:16  2.500479  31.340869        Apagado

[972450 rows x 5 columns]>
```

```
1 df.dtypes
```

```
Dia               object
Hora              object
OD               float64
T                float64
Funcionamiento    object
dtype: object
```

Aireador inyección

```
<bound method NDFrame.describe of               Dia      Hora        OD          T Funcionamiento
0       14/09/2023  20:50:01  7.373333  32.810000      Encendido
1       14/09/2023  20:50:02  7.373333  32.780000      Encendido
2       14/09/2023  20:50:03  7.373333  32.750000      Encendido
3       14/09/2023  20:50:04  7.328889  32.750000      Encendido
4       14/09/2023  20:50:05  7.328889  32.780000      Encendido
...            ...       ...       ...        ...            ...
892465  30/07/2023  05:05:45  3.335107  32.081172        Apagado
892466  30/07/2023  05:05:46  3.330524  32.113464        Apagado
892467  30/07/2023  05:05:47  3.327365  32.045022        Apagado
892468  30/07/2023  05:05:48  3.338103  32.062649        Apagado
892469  30/07/2023  05:05:49  3.344100  32.133493        Apagado

[892470 rows x 5 columns]>
```

```
1 df.dtypes
```

```
Dia               object
Hora              object
OD               float64
T                float64
Funcionamiento    object
dtype: object
```

Algoritmo para el procesamiento de datos

Anexo 3. Interpolación de registros faltantes

```
import pandas as pd

# Ruta del dataset original
ruta_csv = "C:\\Users\\USUARIO\\Downloads\\1x1_inyC.csv"

# Cargar el dataset original en un DataFrame
df_original = pd.read_csv(ruta_csv)

# Mostrar el conjunto de datos original
print("Conjunto de datos original:")
print(df_original.head())

# Total de registros antes de la interpolación
print("\nTotal de registros antes de la interpolación:",
len(df_original))

# Crear una copia del DataFrame original para la interpolación
df = df_original.copy()

# Convierte la columna "Hora" a formato datetime
df['fecha'] = pd.to_datetime(df['Dia'] + ' ' + df['Hora'],
format='%d/%m/%Y %H:%M:%S')

# Establece la columna "Hora" como índice (importante para la
interpolación)
df.set_index('fecha', inplace=True)

# Realiza la interpolación lineal
df = df.resample('S').interpolate(method='linear')

# Restablece la columna "Dia" como una columna regular
df['Dia'] = df.index.strftime('%d/%m/%Y')

# Restablece la columna "Hora" como una columna regular
df['Hora'] = df.index.time

# Reordena las columnas según tu preferencia
df = df[['Dia', 'Hora', 'OD', 'T']]

# Mostrar el conjunto de datos después de la interpolación
print("\nConjunto de datos después de la interpolación:")
print(df.head())

# Total de registros después de la interpolación
print("\nTotal de registros después de la interpolación:", len(df))

# Guardar el DataFrame modificado en un nuevo archivo CSV
nombre_archivo_corregido = "C:\\Users\\USUARIO\\Downloads\\1x1_inyll.csv"
df.to_csv(nombre_archivo_corregido, index=False)
```

```
print(f"\nDataFrame interpolado guardado en el archivo CSV:
{nombre_archivo_corregido}")
```

Anexo 4. Eliminar valores negativos

```
import pandas as pd

# Ruta del dataset original
ruta_csv =

# Cargar el dataset original en un DataFrame
df = pd.read_csv(ruta_csv)

# Ver el head del DataFrame original
print("Antes de la corrección:")
print(df.head())

# Convertir los valores negativos de 'OD' a positivos
df['OD'] = df['OD'].abs()

# Ver el head del DataFrame después de la corrección
print("\nDespués de la corrección:")
print(df.head())

# Guardar el DataFrame modificado en un nuevo archivo CSV
nombre_archivo_corregido =
df.to_csv(nombre_archivo_corregido, index=False)

print(f"\nDataset corregido guardado en: {nombre_archivo_corregido}")
```

Anexo 5. Agregar estado ON/OFF

```
import pandas as pd

# Carga los datos desde el archivo CSV
ruta_csv = "ARCHIVO"
df = pd.read_csv(ruta_csv)

# Convierte la columna "Hora" a tipo datetime
df['fecha'] = pd.to_datetime(df['Dia'] + ' ' + df['Hora'],
format='%d/%m/%Y %H:%M:%S')

# Establece la columna "Hora" como índice (importante para la
interpolación)
df.set_index('fecha', inplace=True)

# Realiza la interpolación lineal
df = df.resample('S').interpolate(method='linear')

# Restablece la columna "Dia" como una columna regular
df['Dia'] = df.index.strftime('%d/%m/%Y')

# Restablece la columna "Hora" como una columna regular
df['Hora'] = df.index.time

# Reordena las columnas según tu preferencia
df = df[['Dia', 'Hora', 'OD', 'T']]

# Define una función para clasificar el tiempo en "Encendido" o "Apagado"
según el patrón
def clasificar_funcionamiento(hora):
    if hora.hour % 2 == 0:
        return 'Encendido'
    else:
        return 'Apagado'

# Aplica la función de clasificación a la columna 'Hora' y crea una nueva
columna 'Funcionamiento'
df['Funcionamiento'] =
df.index.to_series().apply(clasificar_funcionamiento)

# Imprime el DataFrame con las nuevas columnas
print(df)

# Guarda los datos en un nuevo archivo CSV
df.to_csv("C ARCHIVO ", index=False)

print("Datos con columna de destino guardados en el archivo CSV.")
```

Anexo 6. Combinar dataset

```
import pandas as pd
import os

# Ruta de la carpeta que contiene los datasets
carpeta_datasets = CARPETA  # Reemplaza con la ruta correcta

# Lista para almacenar los DataFrames de los archivos
dataframes_pa2 = []

# Recorre todos los archivos en la carpeta
for archivo in os.listdir(carpeta_datasets):
    if archivo.endswith(".csv"):
        # Carga el archivo CSV en un DataFrame
        ruta_archivo = os.path.join(carpeta_datasets, archivo)
        df = pd.read_csv(ruta_archivo)
        dataframes_pa2.append(df)

# Combina todos los DataFrames en uno solo
df_combinado_pa2 = pd.concat(dataframes_pa2, ignore_index=True)

# Guardar el DataFrame combinado en un nuevo archivo CSV
nombre_archivo_combinado_pa2 = NuevoArchivo
df_combinado_pa2.to_csv(nombre_archivo_combinado_pa2, index=False)

print("Archivos combinados y guardados en un nuevo archivo CSV.")
```

Anexo 7. Calcular pendiente por estado

```
import pandas as pd
import numpy as np

# Cargar el dataset
archivo = "C:\\Users\\USUARIO\\Downloads\\PA2_C.csv"
df = pd.read_csv(archivo)

# Filtrar los datos cuando el aireador está encendido
df_encendido = df[df['Funcionamiento'] == 'Encendido']

# Inicializar listas para almacenar las pendientes por día y las fechas
fechas_y_pendientes = []

# Calcular la pendiente por día cuando está encendido
for fecha in df_encendido['Dia'].unique():
    df_dia = df_encendido[df_encendido['Dia'] == fecha]

    tiempo = pd.to_datetime(df_dia['Hora']).dt.second.values
    od = df_dia['OD'].values

    # Evitar divisiones por cero
    if len(tiempo) > 1:
        pendiente_od = np.gradient(od, tiempo)
        pendiente_promedio = pendiente_od.mean()

        # Agregar la fecha y la pendiente al resultado
        fechas_y_pendientes.append((fecha, pendiente_promedio))

# Calcular la pendiente promedio de todas las pendientes por día
pendientes_por_dia = [pendiente for _, pendiente in fechas_y_pendientes]
pendiente_promedio = np.mean(pendientes_por_dia)

# Imprimir resultados
for fecha, pendiente in fechas_y_pendientes:
    print(f"{fecha} --- Pendiente: {pendiente:.10f}")

# Imprimir la pendiente promedio
print(f"\nPendiente promedio de todas las pendientes por día:
{pendiente_promedio:.10f}")
```

Algoritmo genético

Anexo 8. Para 30 minutos

```
import random
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Parámetros del algoritmo genético
tamaño_población = 100
tasa_mutación = 0.1
generaciones = 262144

# Pendientes y valor inicial
pendiente_encendido = -0.15288948
pendiente_apagado = -0.1625112
valor_inicial = 8

# Duración del ciclo en horas (9 horas)
duracion_ciclo_horas = 9

# Objetivo de nivel de OD
objetivo_od = 4

def aptitud(cromosoma):
    od = valor_inicial
    tiempo_encendido = 0
    tiempo_apagado = 0
    tiempo_encendido_total = 0
    tiempo_apagado_total = 0
    tiempo_cumplimiento = 0  # Tiempo para alcanzar el objetivo

    for i in range(len(cromosoma)):
        if cromosoma[i] == 1:
            tiempo_encendido += 1
            tiempo_encendido_total += 1
            tiempo_apagado = 0
            od += pendiente_encendido
        else:
            tiempo_apagado += 1
            tiempo_apagado_total += 1
            tiempo_encendido = 0
            od += pendiente_apagado

        if od < objetivo_od:
            tiempo_cumplimiento += 1

        if od < 4 or tiempo_encendido > 7 or tiempo_apagado > 7:  #
Restricciones
            return 0
```

```
    # Calcula la aptitud para minimizar el tiempo de cumplimiento y la
cantidad de encendidos
    aptitud = 1 / (tiempo_cumplimiento + 1) * (1 /
(tiempo_encendido_total + 1))

    return aptitud

# Inicialización de la población (para intervalos de 30 minutos,
cambiamos el tamaño del cromosoma a 18)
tamaño_cromosoma = duracion_ciclo_horas * 2  # Cada hora se divide en 2
intervalos de 30 minutos
población = [np.random.randint(2, size=tamaño_cromosoma) for _ in
range(tamaño_población)]

# Algoritmo genético
mejores_cromosomas = []  # Almacenar los mejores cromosomas por
generación

for generación in range(generaciones):
    aptitudes = [aptitud(cromosoma) for cromosoma in población]
    mejor_aptitud = max(aptitudes)
    mejor_índice = np.argmax(aptitudes)
    mejor_cromosoma = población[mejor_índice]
    mejores_cromosomas.append(mejor_cromosoma)

    nueva_población = []
    while len(nueva_población) < tamaño_población:
        padre1, padre2 = random.choices(población, k=2)
        punto_cruza = random.randint(1, len(padre1) - 1)
        hijo = np.concatenate((padre1[:punto_cruza],
padre2[punto_cruza:]))

        if random.random() < tasa_mutación:
            punto_mutación = random.randint(0, len(hijo) - 1)
            hijo[punto_mutación] = 1 - hijo[punto_mutación]

        nueva_población.append(hijo)

    población = nueva_población

# Obtener solo cromosomas con tiempos de encendido menores a 5 horas (300
minutos)
mejores_cromosomas_filtrados = [cromosoma for cromosoma in
mejores_cromosomas if sum(cromosoma) < 360]

# Mostrar resultados
print("Mejores soluciones encontradas con tiempos de encendido menores a
5 horas:")

resultados = []  # Lista para almacenar resultados
mejor_tiempo_cumplimiento = float('inf')  # Para encontrar el mejor
tiempo de cumplimiento
mejor_cromosoma_global = None  # Para almacenar el mejor cromosoma global
```

```
for i, cromosoma in enumerate(mejores_cromosomas_filtrados):
    mejor_aptitud = aptitud(cromosoma)
    tiempo_cumplimiento = 0
    od = valor_inicial
    for j in range(len(cromosoma)):
        if cromosoma[j] == 1:
            tiempo_cumplimiento += 1
            od += pendiente_encendido
        else:
            od += pendiente_apagado
        if od >= objetivo_od:
            break

    resultados.append({
        "Solución": i + 1,
        "Cromosoma": cromosoma,
        "Aptitud": mejor_aptitud,
        "Tiempo de Cumplimiento": tiempo_cumplimiento
    })

    # Verificar si este es el mejor cromosoma en términos del tiempo de
cumplimiento
    if tiempo_cumplimiento < mejor_tiempo_cumplimiento:
        mejor_tiempo_cumplimiento = tiempo_cumplimiento
        mejor_cromosoma_global = cromosoma

# Crear un DataFrame de Pandas con los resultados filtrados
df_filtrado = pd.DataFrame(resultados)
print(df_filtrado)

# Definir la función de simulación para intervalos de 30 minutos
def simular_comportamiento(cromosoma):
    nivel_od = [valor_inicial]
    for i in range(len(cromosoma)):
        if cromosoma[i] == 1:
            nivel_od.append(nivel_od[-1] + pendiente_encendido)
        else:
            nivel_od.append(nivel_od[-1] + pendiente_apagado)
    return nivel_od

# Graficar el comportamiento de todos los cromosomas filtrados
plt.figure(figsize=(10, 6))
for cromosoma in mejores_cromosomas_filtrados:
    nivel_od_simulado = simular_comportamiento(cromosoma)
    plt.plot(range(len(cromosoma) + 1), nivel_od_simulado, alpha=0.3)

plt.axhline(y=4, color='r', linestyle='--', label='OD objetivo (4)')
plt.xlabel("Tiempo (intervalos de 30 minutos)")
plt.ylabel("Nivel de Oxígeno Disuelto (OD)")
plt.title("Comportamiento simulado del OD para las mejores soluciones
filtradas")
plt.legend()
```

```
plt.show()

# Graficar el comportamiento del mejor cromosoma filtrado
plt.figure(figsize=(10, 6))
nivel_od_simulado = simular_comportamiento(mejor_cromosoma_global)
plt.plot(range(len(mejor_cromosoma_global) + 1), nivel_od_simulado,
label='Mejor Cromosoma', color='b')
plt.axhline(y=objetivo_od, color='r', linestyle='--', label=f'OD objetivo
({objetivo_od})')
plt.xlabel("Tiempo (intervalos de 30 minutos)")
plt.ylabel("Nivel de Oxígeno Disuelto (OD)")
plt.title("Comportamiento simulado del OD para la mejor solución
filtrada")
plt.legend()
plt.show()

# Mostrar el mejor cromosoma en términos del tiempo de cumplimiento
print("Mejor cromosoma en términos del tiempo de cumplimiento:")
print(mejor_cromosoma_global)
print(f"Mejor tiempo de cumplimiento: {mejor_tiempo_cumplimiento}")
print(f"Nivel de oxígeno alcanzado con la mejor solución filtrada:
{nivel_od_simulado[-1]}")
```

Anexo 9. Para 15 minutos

```
import random
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Parámetros del algoritmo genético
tamaño_población = 100
tasa_mutación = 0.1
generaciones = 262144

# Pendientes y valor inicial
pendiente_encendido = -0.07644474

pendiente_apagado =-0.0812556

valor_inicial = 8

# Duración del ciclo en horas (9 horas)
duracion_ciclo_horas = 9

# Objetivo de nivel de OD
objetivo_od = 4

# Cambio a intervalos de 15 minutos
intervalo_tiempo = 15
intervalos_por_hora = 60 / intervalo_tiempo
tamaño_cromosoma = int(duracion_ciclo_horas * intervalos_por_hora)

def aptitud(cromosoma):
    od = valor_inicial
    tiempo_encendido = 0
    tiempo_apagado = 0
    tiempo_encendido_total = 0
    tiempo_apagado_total = 0
    tiempo_cumplimiento = 0  # Tiempo para alcanzar el objetivo

    for i in range(len(cromosoma)):
        if cromosoma[i] == 1:
            tiempo_encendido += 1
            tiempo_encendido_total += 1
            tiempo_apagado = 0
            od += pendiente_encendido
        else:
            tiempo_apagado += 1
            tiempo_apagado_total += 1
            tiempo_encendido = 0
            od += pendiente_apagado

        if od < objetivo_od:
            tiempo_cumplimiento += 1
```

```
        if od < 4 or tiempo_encendido > (7 * intervalos_por_hora) or
tiempo_apagado > (7 * intervalos_por_hora):
            return 0

    # Calcula la aptitud para minimizar el tiempo de cumplimiento y la
cantidad de encendidos
    aptitud = 1 / (tiempo_cumplimiento + 1) * (1 /
(tiempo_encendido_total + 1))

    return aptitud

# Inicialización de la población
población = [np.random.randint(2, size=tamaño_cromosoma) for _ in
range(tamaño_población)]

# Algoritmo genético
mejores_cromosomas = []

for generación in range(generaciones):
    aptitudes = [aptitud(cromosoma) for cromosoma in población]
    mejor_aptitud = max(aptitudes)
    mejor_índice = np.argmax(aptitudes)
    mejor_cromosoma = población[mejor_índice]
    mejores_cromosomas.append(mejor_cromosoma)

    nueva_población = []
    while len(nueva_población) < tamaño_población:
        padre1, padre2 = random.choices(población, k=2)
        punto_cruza = random.randint(1, len(padre1) - 1)
        hijo = np.concatenate((padre1[:punto_cruza],
padre2[punto_cruza:]))

        if random.random() < tasa_mutación:
            punto_mutación = random.randint(0, len(hijo) - 1)
            hijo[punto_mutación] = 1 - hijo[punto_mutación]

        nueva_población.append(hijo)

    población = nueva_población

# Obtener solo cromosomas con tiempos de encendido menores a 5 horas (300
minutos)
mejores_cromosomas_filtrados = [cromosoma for cromosoma in
mejores_cromosomas if sum(cromosoma) < (360 * intervalos_por_hora)]

# Mostrar resultados
print("Mejores soluciones encontradas con tiempos de encendido menores a
5 horas:")

resultados = []  # Lista para almacenar resultados
mejor_tiempo_cumplimiento = float('inf')  # Para encontrar el mejor
tiempo de cumplimiento
mejor_cromosoma_global = None  # Para almacenar el mejor cromosoma global
```

```
for i, cromosoma in enumerate(mejores_cromosomas_filtrados):
    mejor_aptitud = aptitud(cromosoma)
    tiempo_cumplimiento = 0
    od = valor_inicial
    for j in range(len(cromosoma)):
        if cromosoma[j] == 1:
            tiempo_cumplimiento += 1
            od += pendiente_encendido
        else:
            od += pendiente_apagado
        if od >= objetivo_od:
            break

    resultados.append({
        "Solución": i + 1,
        "Cromosoma": cromosoma,
        "Aptitud": mejor_aptitud,
        "Tiempo de Cumplimiento": tiempo_cumplimiento
    })

    # Verificar si este es el mejor cromosoma en términos del tiempo de
cumplimiento
    if tiempo_cumplimiento < mejor_tiempo_cumplimiento:
        mejor_tiempo_cumplimiento = tiempo_cumplimiento
        mejor_cromosoma_global = cromosoma

# Crear un DataFrame de Pandas con los resultados filtrados
df_filtrado = pd.DataFrame(resultados)
print(df_filtrado)

# Definir la función de simulación para intervalos de 15 minutos
def simular_comportamiento(cromosoma):
    nivel_od = [valor_inicial]
    for i in range(len(cromosoma)):
        if cromosoma[i] == 1:
            nivel_od.append(nivel_od[-1] + pendiente_encendido)
        else:
            nivel_od.append(nivel_od[-1] + pendiente_apagado)
    return nivel_od

# Graficar el comportamiento de todos los cromosomas filtrados
plt.figure(figsize=(10, 6))
for cromosoma in mejores_cromosomas_filtrados:
    nivel_od_simulado = simular_comportamiento(cromosoma)
    plt.plot(range(len(cromosoma) + 1), nivel_od_simulado, alpha=0.3)

plt.axhline(y=4, color='r', linestyle='--', label='OD objetivo (4)')
plt.xlabel("Tiempo (intervalos de 15 minutos)")
plt.ylabel("Nivel de Oxígeno Disuelto (OD)")
plt.title("Comportamiento simulado del OD para las mejores soluciones
filtradas")
plt.legend()
```

```
plt.show()

# Graficar el comportamiento del mejor cromosoma filtrado
plt.figure(figsize=(10, 6))
nivel_od_simulado = simular_comportamiento(mejor_cromosoma_global)
plt.plot(range(len(mejor_cromosoma_global) + 1), nivel_od_simulado,
label='Mejor Cromosoma', color='b')
plt.axhline(y=objetivo_od, color='r', linestyle='--', label=f'OD objetivo
({objetivo_od})')
plt.xlabel("Tiempo (intervalos de 15 minutos)")
plt.ylabel("Nivel de Oxígeno Disuelto (OD)")
plt.title("Comportamiento simulado del OD para la mejor solución
filtrada")
plt.legend()
plt.show()

# Mostrar el mejor cromosoma en términos del tiempo de cumplimiento
print("Mejor cromosoma en términos del tiempo de cumplimiento:")
print(mejor_cromosoma_global)
print(f"Mejor tiempo de cumplimiento: {mejor_tiempo_cumplimiento}")
print(f"Nivel de oxígeno alcanzado con la mejor solución filtrada:
{nivel_od_simulado[-1]}")
```

Anexo 10. Para 20 minutos

```
import random
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Parámetros del algoritmo genético
tamaño_población = 100
tasa_mutación = 0.1
generaciones = 262144

# Pendientes y valor inicial
pendiente_encendido = -0.10192632

pendiente_apagado = -0.1083408

valor_inicial = 8

# Duración del ciclo en horas (9 horas)
duracion_ciclo_horas = 9

# Objetivo de nivel de OD
objetivo_od = 4

# Cambio a intervalos de 20 minutos
intervalo_tiempo = 20
intervalos_por_hora = 60 / intervalo_tiempo
tamaño_cromosoma = int(duracion_ciclo_horas * intervalos_por_hora)

def aptitud(cromosoma):
    od = valor_inicial
    tiempo_encendido = 0
    tiempo_apagado = 0
    tiempo_encendido_total = 0
    tiempo_apagado_total = 0
    tiempo_cumplimiento = 0  # Tiempo para alcanzar el objetivo

    for i in range(len(cromosoma)):
        if cromosoma[i] == 1:
            tiempo_encendido += 1
            tiempo_encendido_total += 1
            tiempo_apagado = 0
            od += pendiente_encendido
        else:
            tiempo_apagado += 1
            tiempo_apagado_total += 1
            tiempo_encendido = 0
            od += pendiente_apagado

        if od < objetivo_od:
            tiempo_cumplimiento += 1
```

```
        if od < 4 or tiempo_encendido > (7 * intervalos_por_hora) or
tiempo_apagado > (7 * intervalos_por_hora):
            return 0

    # Calcula la aptitud para minimizar el tiempo de cumplimiento y la
cantidad de encendidos
    aptitud = 1 / (tiempo_cumplimiento + 1) * (1 /
(tiempo_encendido_total + 1))

    return aptitud

# Inicialización de la población
población = [np.random.randint(2, size=tamaño_cromosoma) for _ in
range(tamaño_población)]

# Algoritmo genético
mejores_cromosomas = []

for generación in range(generaciones):
    aptitudes = [aptitud(cromosoma) for cromosoma in población]
    mejor_aptitud = max(aptitudes)
    mejor_índice = np.argmax(aptitudes)
    mejor_cromosoma = población[mejor_índice]
    mejores_cromosomas.append(mejor_cromosoma)

    nueva_población = []
    while len(nueva_población) < tamaño_población:
        padre1, padre2 = random.choices(población, k=2)
        punto_cruza = random.randint(1, len(padre1) - 1)
        hijo = np.concatenate((padre1[:punto_cruza],
padre2[punto_cruza:]))

        if random.random() < tasa_mutación:
            punto_mutación = random.randint(0, len(hijo) - 1)
            hijo[punto_mutación] = 1 - hijo[punto_mutación]

        nueva_población.append(hijo)

    población = nueva_población

# Obtener solo cromosomas con tiempos de encendido menores a 5 horas (300
minutos)
mejores_cromosomas_filtrados = [cromosoma for cromosoma in
mejores_cromosomas if sum(cromosoma) < (360 * intervalos_por_hora)]

# Mostrar resultados
print("Mejores soluciones encontradas con tiempos de encendido menores a
5 horas:")

resultados = []  # Lista para almacenar resultados
mejor_tiempo_cumplimiento = float('inf')  # Para encontrar el mejor
tiempo de cumplimiento
mejor_cromosoma_global = None  # Para almacenar el mejor cromosoma global
```

```
for i, cromosoma in enumerate(mejores_cromosomas_filtrados):
    mejor_aptitud = aptitud(cromosoma)
    tiempo_cumplimiento = 0
    od = valor_inicial
    for j in range(len(cromosoma)):
        if cromosoma[j] == 1:
            tiempo_cumplimiento += 1
            od += pendiente_encendido
        else:
            od += pendiente_apagado
        if od >= objetivo_od:
            break

    resultados.append({
        "Solución": i + 1,
        "Cromosoma": cromosoma,
        "Aptitud": mejor_aptitud,
        "Tiempo de Cumplimiento": tiempo_cumplimiento
    })

    # Verificar si este es el mejor cromosoma en términos del tiempo de
cumplimiento
    if tiempo_cumplimiento < mejor_tiempo_cumplimiento:
        mejor_tiempo_cumplimiento = tiempo_cumplimiento
        mejor_cromosoma_global = cromosoma

# Crear un DataFrame de Pandas con los resultados filtrados
df_filtrado = pd.DataFrame(resultados)
print(df_filtrado)

# Definir la función de simulación para intervalos de 20 minutos
def simular_comportamiento(cromosoma):
    nivel_od = [valor_inicial]
    for i in range(len(cromosoma)):
        if cromosoma[i] == 1:
            nivel_od.append(nivel_od[-1] + pendiente_encendido)
        else:
            nivel_od.append(nivel_od[-1] + pendiente_apagado)
    return nivel_od

# Graficar el comportamiento de todos los cromosomas filtrados
plt.figure(figsize=(10, 6))
for cromosoma in mejores_cromosomas_filtrados:
    nivel_od_simulado = simular_comportamiento(cromosoma)
    plt.plot(range(len(cromosoma) + 1), nivel_od_simulado, alpha=0.3)

plt.axhline(y=4, color='r', linestyle='--', label='OD objetivo (4)')
plt.xlabel("Tiempo (intervalos de 20 minutos)")
plt.ylabel("Nivel de Oxígeno Disuelto (OD)")
plt.title("Comportamiento simulado del OD para las mejores soluciones
filtradas")
plt.legend()
```

```
plt.show()

# Graficar el comportamiento del mejor cromosoma filtrado
plt.figure(figsize=(10, 6))
nivel_od_simulado = simular_comportamiento(mejor_cromosoma_global)
plt.plot(range(len(mejor_cromosoma_global) + 1), nivel_od_simulado,
label='Mejor Cromosoma', color='b')
plt.axhline(y=objetivo_od, color='r', linestyle='--', label=f'OD objetivo
({objetivo_od})')
plt.xlabel("Tiempo (intervalos de 20 minutos)")
plt.ylabel("Nivel de Oxígeno Disuelto (OD)")
plt.title("Comportamiento simulado del OD para la mejor solución
filtrada")
plt.legend()
plt.show()

# Mostrar el mejor cromosoma en términos del tiempo de cumplimiento
print("Mejor cromosoma en términos del tiempo de cumplimiento:")
print(mejor_cromosoma_global)
print(f"Mejor tiempo de cumplimiento: {mejor_tiempo_cumplimiento}")
print(f"Nivel de oxígeno alcanzado con la mejor solución filtrada:
{nivel_od_simulado[-1]}")
```

Anexo 11. Para 60 minutos

```
import random
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Parámetros del algoritmo genético
tamaño_población = 100
tasa_mutación = 0.1
generaciones = 262144

# Pendientes y valor inicial
pendiente_encendido = -0.30577896

pendiente_apagado = -0.3250224

valor_inicial = 8

# Duración del ciclo en horas (9 horas)
duracion_ciclo_horas = 9

# Objetivo de nivel de OD
objetivo_od = 4

# Cambio a intervalos de 1 hora
intervalo_tiempo = 60
intervalos_por_hora = 60 / intervalo_tiempo
tamaño_cromosoma = int(duracion_ciclo_horas * intervalos_por_hora)

def aptitud(cromosoma):
    od = valor_inicial
    tiempo_encendido = 0
    tiempo_apagado = 0
    tiempo_encendido_total = 0
    tiempo_apagado_total = 0
    tiempo_cumplimiento = 0  # Tiempo para alcanzar el objetivo

    for i in range(len(cromosoma)):
        if cromosoma[i] == 1:
            tiempo_encendido += 1
            tiempo_encendido_total += 1
            tiempo_apagado = 0
            od += pendiente_encendido
        else:
            tiempo_apagado += 1
            tiempo_apagado_total += 1
            tiempo_encendido = 0
            od += pendiente_apagado

        if od < objetivo_od:
            tiempo_cumplimiento += 1
```

```
        if od < 4 or tiempo_encendido > (7 * intervalos_por_hora) or
tiempo_apagado > (7 * intervalos_por_hora):
            return 0

    # Calcula la aptitud para minimizar el tiempo de cumplimiento y la
cantidad de encendidos
    aptitud = 1 / (tiempo_cumplimiento + 1) * (1 /
(tiempo_encendido_total + 1))

    return aptitud

# Inicialización de la población
población = [np.random.randint(2, size=tamaño_cromosoma) for _ in
range(tamaño_población)]

# Algoritmo genético
mejores_cromosomas = []

for generación in range(generaciones):
    aptitudes = [aptitud(cromosoma) for cromosoma in población]
    mejor_aptitud = max(aptitudes)
    mejor_índice = np.argmax(aptitudes)
    mejor_cromosoma = población[mejor_índice]
    mejores_cromosomas.append(mejor_cromosoma)

    nueva_población = []
    while len(nueva_población) < tamaño_población:
        padre1, padre2 = random.choices(población, k=2)
        punto_cruza = random.randint(1, len(padre1) - 1)
        hijo = np.concatenate((padre1[:punto_cruza],
padre2[punto_cruza:]))

        if random.random() < tasa_mutación:
            punto_mutación = random.randint(0, len(hijo) - 1)
            hijo[punto_mutación] = 1 - hijo[punto_mutación]

        nueva_población.append(hijo)

    población = nueva_población

# Obtener solo cromosomas con tiempos de encendido menores a 5 horas (300
minutos)
mejores_cromosomas_filtrados = [cromosoma for cromosoma in
mejores_cromosomas if sum(cromosoma) < (360 * intervalos_por_hora)]

# Mostrar resultados
print("Mejores soluciones encontradas con tiempos de encendido menores a
5 horas:")

resultados = []  # Lista para almacenar resultados
mejor_tiempo_cumplimiento = float('inf')  # Para encontrar el mejor
tiempo de cumplimiento
mejor_cromosoma_global = None  # Para almacenar el mejor cromosoma global
```

```
for i, cromosoma in enumerate(mejores_cromosomas_filtrados):
    mejor_aptitud = aptitud(cromosoma)
    tiempo_cumplimiento = 0
    od = valor_inicial
    for j in range(len(cromosoma)):
        if cromosoma[j] == 1:
            tiempo_cumplimiento += 1
            od += pendiente_encendido
        else:
            od += pendiente_apagado
        if od >= objetivo_od:
            break

    resultados.append({
        "Solución": i + 1,
        "Cromosoma": cromosoma,
        "Aptitud": mejor_aptitud,
        "Tiempo de Cumplimiento": tiempo_cumplimiento
    })

    # Verificar si este es el mejor cromosoma en términos del tiempo de
cumplimiento
    if tiempo_cumplimiento < mejor_tiempo_cumplimiento:
        mejor_tiempo_cumplimiento = tiempo_cumplimiento
        mejor_cromosoma_global = cromosoma

# Crear un DataFrame de Pandas con los resultados filtrados
df_filtrado = pd.DataFrame(resultados)
print(df_filtrado)

# Definir la función de simulación para intervalos de 1 hora
def simular_comportamiento(cromosoma):
    nivel_od = [valor_inicial]
    for i in range(len(cromosoma)):
        if cromosoma[i] == 1:
            nivel_od.append(nivel_od[-1] + pendiente_encendido)
        else:
            nivel_od.append(nivel_od[-1] + pendiente_apagado)
    return nivel_od

# Graficar el comportamiento de todos los cromosomas filtrados
plt.figure(figsize=(10, 6))
for cromosoma in mejores_cromosomas_filtrados:
    nivel_od_simulado = simular_comportamiento(cromosoma)
    plt.plot(range(len(cromosoma) + 1), nivel_od_simulado, alpha=0.3)

plt.axhline(y=4, color='r', linestyle='--', label='OD objetivo (4)')
plt.xlabel("Tiempo (intervalos de 1 hora)")
plt.ylabel("Nivel de Oxígeno Disuelto (OD)")
plt.title("Comportamiento simulado del OD para las mejores soluciones
filtradas")
plt.legend()
```

```
plt.show()

# Graficar el comportamiento del mejor cromosoma filtrado
plt.figure(figsize=(10, 6))
nivel_od_simulado = simular_comportamiento(mejor_cromosoma_global)
plt.plot(range(len(mejor_cromosoma_global) + 1), nivel_od_simulado,
label='Mejor Cromosoma', color='b')
plt.axhline(y=objetivo_od, color='r', linestyle='--', label=f'OD objetivo
({objetivo_od})')
plt.xlabel("Tiempo (intervalos de 1 hora)")
plt.ylabel("Nivel de Oxígeno Disuelto (OD)")
plt.title("Comportamiento simulado del OD para la mejor solución
filtrada")
plt.legend()
plt.show()

# Mostrar el mejor cromosoma en términos del tiempo de cumplimiento
print("Mejor cromosoma en términos del tiempo de cumplimiento:")
print(mejor_cromosoma_global)
print(f"Mejor tiempo de cumplimiento: {mejor_tiempo_cumplimiento}")
print(f"Nivel de oxígeno alcanzado con la mejor solución filtrada:
{nivel_od_simulado[-1]}")
```

Anexo 12. Para 120 minutos

```
import random
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Parámetros del algoritmo genético
tamaño_población = 100
tasa_mutación = 0.1
generaciones = 262144

# Pendientes y valor inicial
pendiente_encendido = -0.61155792

pendiente_apagado = -0.6500448

valor_inicial = 8

# Duración del ciclo en horas (9 horas)
duracion_ciclo_horas = 9

# Objetivo de nivel de OD
objetivo_od = 4

# Cambio a intervalos de 2 horas
intervalo_tiempo = 120
intervalos_por_hora = 60 / intervalo_tiempo
tamaño_cromosoma = int(duracion_ciclo_horas * intervalos_por_hora)

def aptitud(cromosoma):
    od = valor_inicial
    tiempo_encendido = 0
    tiempo_apagado = 0
    tiempo_encendido_total = 0
    tiempo_apagado_total = 0
    tiempo_cumplimiento = 0  # Tiempo para alcanzar el objetivo

    for i in range(len(cromosoma)):
        if cromosoma[i] == 1:
            tiempo_encendido += 1
            tiempo_encendido_total += 1
            tiempo_apagado = 0
            od += pendiente_encendido
        else:
            tiempo_apagado += 1
            tiempo_apagado_total += 1
            tiempo_encendido = 0
            od += pendiente_apagado

        if od < objetivo_od:
            tiempo_cumplimiento += 1
```

```
        if od < 4 or tiempo_encendido > (7 * intervalos_por_hora) or
tiempo_apagado > (7 * intervalos_por_hora):
            return 0

    # Calcula la aptitud para minimizar el tiempo de cumplimiento y la
cantidad de encendidos
    aptitud = 1 / (tiempo_cumplimiento + 1) * (1 /
(tiempo_encendido_total + 1))

    return aptitud

# Inicialización de la población
población = [np.random.randint(2, size=tamaño_cromosoma) for _ in
range(tamaño_población)]

# Algoritmo genético
mejores_cromosomas = []

for generación in range(generaciones):
    aptitudes = [aptitud(cromosoma) for cromosoma in población]
    mejor_aptitud = max(aptitudes)
    mejor_índice = np.argmax(aptitudes)
    mejor_cromosoma = población[mejor_índice]
    mejores_cromosomas.append(mejor_cromosoma)

    nueva_población = []
    while len(nueva_población) < tamaño_población:
        padre1, padre2 = random.choices(población, k=2)
        punto_cruza = random.randint(1, len(padre1) - 1)
        hijo = np.concatenate((padre1[:punto_cruza],
padre2[punto_cruza:]))

        if random.random() < tasa_mutación:
            punto_mutación = random.randint(0, len(hijo) - 1)
            hijo[punto_mutación] = 1 - hijo[punto_mutación]

        nueva_población.append(hijo)

    población = nueva_población

# Obtener solo cromosomas con tiempos de encendido menores a 5 horas (300
minutos)
mejores_cromosomas_filtrados = [cromosoma for cromosoma in
mejores_cromosomas if sum(cromosoma) < (360 * intervalos_por_hora)]

# Mostrar resultados
print("Mejores soluciones encontradas con tiempos de encendido menores a
5 horas:")

resultados = []  # Lista para almacenar resultados
mejor_tiempo_cumplimiento = float('inf')  # Para encontrar el mejor
tiempo de cumplimiento
mejor_cromosoma_global = None  # Para almacenar el mejor cromosoma global
```

```
for i, cromosoma in enumerate(mejores_cromosomas_filtrados):
    mejor_aptitud = aptitud(cromosoma)
    tiempo_cumplimiento = 0
    od = valor_inicial
    for j in range(len(cromosoma)):
        if cromosoma[j] == 1:
            tiempo_cumplimiento += 1
            od += pendiente_encendido
        else:
            od += pendiente_apagado
        if od >= objetivo_od:
            break

    resultados.append({
        "Solución": i + 1,
        "Cromosoma": cromosoma,
        "Aptitud": mejor_aptitud,
        "Tiempo de Cumplimiento": tiempo_cumplimiento
    })

    # Verificar si este es el mejor cromosoma en términos del tiempo de
cumplimiento
    if tiempo_cumplimiento < mejor_tiempo_cumplimiento:
        mejor_tiempo_cumplimiento = tiempo_cumplimiento
        mejor_cromosoma_global = cromosoma

# Crear un DataFrame de Pandas con los resultados filtrados
df_filtrado = pd.DataFrame(resultados)
print(df_filtrado)

# Definir la función de simulación para intervalos de 2 horas
def simular_comportamiento(cromosoma):
    nivel_od = [valor_inicial]
    for i in range(len(cromosoma)):
        if cromosoma[i] == 1:
            nivel_od.append(nivel_od[-1] + pendiente_encendido)
        else:
            nivel_od.append(nivel_od[-1] + pendiente_apagado)
    return nivel_od

# Graficar el comportamiento de todos los cromosomas filtrados
plt.figure(figsize=(10, 6))
for cromosoma in mejores_cromosomas_filtrados:
    nivel_od_simulado = simular_comportamiento(cromosoma)
    plt.plot(range(len(cromosoma) + 1), nivel_od_simulado, alpha=0.3)

plt.axhline(y=4, color='r', linestyle='--', label='OD objetivo (4)')
plt.xlabel("Tiempo (intervalos de 2 horas)")
plt.ylabel("Nivel de Oxígeno Disuelto (OD)")
plt.title("Comportamiento simulado del OD para las mejores soluciones
filtradas")
plt.legend()
```

```
plt.show()

# Graficar el comportamiento del mejor cromosoma filtrado
plt.figure(figsize=(10, 6))
nivel_od_simulado = simular_comportamiento(mejor_cromosoma_global)
plt.plot(range(len(mejor_cromosoma_global) + 1), nivel_od_simulado,
label='Mejor Cromosoma', color='b')
plt.axhline(y=objetivo_od, color='r', linestyle='--', label=f'OD objetivo
({objetivo_od})')
plt.xlabel("Tiempo (intervalos de 2 horas)")
plt.ylabel("Nivel de Oxígeno Disuelto (OD)")
plt.title("Comportamiento simulado del OD para la mejor solución
filtrada")
plt.legend()
plt.show()

# Mostrar el mejor cromosoma en términos del tiempo de cumplimiento
print("Mejor cromosoma en términos del tiempo de cumplimiento:")
print(mejor_cromosoma_global)
print(f"Mejor tiempo de cumplimiento: {mejor_tiempo_cumplimiento}")
print(f"Nivel de oxígeno alcanzado con la mejor solución filtrada:
{nivel_od_simulado[-1]}")
```

Algoritmo para el análisis de resultados

Anexo 13. Algoritmo para calculo y graficado del consumo de energía

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from tabulate import tabulate

# Función para calcular el consumo de energía en un período de tiempo
def calcular_consumo_energia(patron, datos_aireador, tiempo, intervalo):
    consumo_total = 0.0

    for estado in patron[:tiempo + 1]:
        if estado == 1:
            consumo_total += (datos_aireador["potencia_nominal"] *
datos_aireador["tensión_nominal"]/1000)



    # Ajustar el consumo total al intervalo de tiempo especificado
    consumo_total *= intervalo / 60.0  # Convertir a kWh

    return consumo_total


# Datos eléctricos para los aireadores (ejemplo)
aireadores = [
    {
        "nombre": "Paletas N",
        "potencia_nominal": .440,  # kW
        "corriente_de_arranque": 12.0,  # amperios
        "tensión_nominal": 220,  # voltios
    },
    {
        "nombre": "Fuente N",
        "potencia_nominal": 1.32,  # kW
        "corriente_de_arranque": 36,  # amperios
        "tensión_nominal": 220,  # voltios
    },
    {
        "nombre": "Inyección N",
        "potencia_nominal":  2.33 ,  # kW
        "corriente_de_arranque": 42.0,  # amperios
        "tensión_nominal": 220,  # voltios
    },
    {
        "nombre": "Paletas 15",
        "potencia_nominal": .440,  # kW
        "corriente_de_arranque": 12.0,  # amperios
        "tensión_nominal": 220,  # voltios
    },
    {
        "nombre": "Fuente 15",
        "potencia_nominal": 1.32,  # kW
```

```
    "corriente_de_arranque": 36,  # amperios
    "tensión_nominal": 220,  # voltios
},
{
    "nombre": "Inyección 15",
    "potencia_nominal":  2.33 ,  # kW
    "corriente_de_arranque": 42.0,  # amperios
    "tensión_nominal": 220,  # voltios
},
{
    "nombre": "Paletas 20",
    "potencia_nominal": .440,  # kW
    "corriente_de_arranque": 12.0,  # amperios
    "tensión_nominal": 220,  # voltios
},
{
    "nombre": "Fuente 20",
    "potencia_nominal": 1.32,  # kW
    "corriente_de_arranque": 36,  # amperios
    "tensión_nominal": 220,  # voltios
},
{
    "nombre": "Inyección 20",
    "potencia_nominal":  2.33 ,  # kW
    "corriente_de_arranque": 42.0,  # amperios
    "tensión_nominal": 220,  # voltios
},
{
    "nombre": "Paletas 30",
    "potencia_nominal": .440,  # kW
    "corriente_de_arranque": 12.0,  # amperios
    "tensión_nominal": 220,  # voltios
},
{
    "nombre": "Fuente 30",
    "potencia_nominal": 1.32,  # kW
    "corriente_de_arranque": 36,  # amperios
    "tensión_nominal": 220,  # voltios
},
{
    "nombre": "Inyección 30",
    "potencia_nominal":  2.33 ,  # kW
    "corriente_de_arranque": 42.0,  # amperios
    "tensión_nominal": 220,  # voltios
},
{
    "nombre": "Paletas 60",
    "potencia_nominal": .440,  # kW
    "corriente_de_arranque": 12.0,  # amperios
    "tensión_nominal": 220,  # voltios
},
{
    "nombre": "Fuente 60",
```

```
        "potencia_nominal": 1.32,  # kW
        "corriente_de_arranque": 36,  # amperios
        "tensión_nominal": 220,  # voltios
    },
    {
        "nombre": "Inyección 60",
        "potencia_nominal":  2.33 ,  # kW
        "corriente_de_arranque": 42.0,  # amperios
        "tensión_nominal": 220,  # voltios
    },
    {
        "nombre": "Paletas 120",
        "potencia_nominal": .440,  # kW
        "corriente_de_arranque": 12.0,  # amperios
        "tensión_nominal": 220,  # voltios
    },
    {
        "nombre": "Fuente 120",
        "potencia_nominal": 1.32,  # kW
        "corriente_de_arranque": 36,  # amperios
        "tensión_nominal": 220,  # voltios
    },
    {
        "nombre": "Inyección 120",
        "potencia_nominal":  2.33 ,  # kW
        "corriente_de_arranque": 42.0,  # amperios
        "tensión_nominal": 220,  # voltios
    },
]

# Nuevos patrones de encendido/apagado de 5 minutos
patrones_aireadores_5min = [
    [1,1,1,1,1,1,      1,1,1,1,1,1,    0,0,0,0,0,0,    0,0,0,0,0,0,
        1,1,1,1,1,1,    1,1,1,1,1,1,    0,0,0,0,0,0,    0,0,0,0,0,0,
        1,1,1,1,1,1,    1,1,1,1,1,1,    0,0,0,0,0,0,    0,0,0,0,0,0,
        1,1,1,1,1,1,    1,1,1,1,1,1,    0,0,0,0,0,0,    0,0,0,0,0,0,
        1,1,1,1,1,1,    1,1,1,1,1,1],
[1,1,1,1,1,1,  1,1,1,1,1,1,    0,0,0,0,0,0,    0,0,0,0,0,0,    1,1,1,1,1,1,
        1,1,1,1,1,1,    0,0,0,0,0,0,    0,0,0,0,0,0,    1,1,1,1,1,1,
        1,1,1,1,1,1,    0,0,0,0,0,0,    0,0,0,0,0,0,    1,1,1,1,1,1,
        1,1,1,1,1,1,    0,0,0,0,0,0,    0,0,0,0,0,0,    1,1,1,1,1,1,
        1,1,1,1,1,1],
[1,1,1,1,1,1,  1,1,1,1,1,1,    0,0,0,0,0,0,    0,0,0,0,0,0,    1,1,1,1,1,1,
        1,1,1,1,1,1,    0,0,0,0,0,0,    0,0,0,0,0,0,    1,1,1,1,1,1,
        1,1,1,1,1,1,    0,0,0,0,0,0,    0,0,0,0,0,0,    1,1,1,1,1,1,
        1,1,1,1,1,1,    0,0,0,0,0,0,    0,0,0,0,0,0,    1,1,1,1,1,1,
        1,1,1,1,1,1],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,1,1,1,0,0,0,1,1,1,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,1,
0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,
```

```
0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,
0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,0,0,0],
[0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,1,1,1,0,0,0,1,1,1,1,1,1,0,0,0,0,0,0,
1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,1,1,1,0,0,0,1,1,1,0,0,0,1,1,1,0,0,0,1,1,1,1
,1,1,1,1,1,1,1,1,0,0,0,1,1,1,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0],
[0,0,0,0,      0,0,0,0,      1,1,1,1,      1,1,1,1,      1,1,1,1,
       1,1,1,1,      0,0,0,0,      0,0,0,0,      0,0,0,0,
       0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
       0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
       0,0,0,0,      0,0,0,0,      1,1,1,1,      0,0,0,0,
       0,0,0,0,      0,0,0,0,      1,1,1,1,      0,0,0,0,
       0,0,0,0,      1,1,1,1,],
[0,0,0,0,      0,0,0,0,      1,1,1,1,      1,1,1,1,      1,1,1,1,
       0,0,0,0,      0,0,0,0,      1,1,1,1,      0,0,0,0,
       1,1,1,1,      1,1,1,1,      0,0,0,0,      0,0,0,0,
       1,1,1,1,      1,1,1,1,      0,0,0,0,      0,0,0,0,
       0,0,0,0,      0,0,0,0,      1,1,1,1,      1,1,1,1,
       1,1,1,1,      0,0,0,0,      0,0,0,0,      0,0,0,0,
       0,0,0,0,      1,1,1,1,],
[0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,      1,1,1,1,
       0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
       0,0,0,0,      0,0,0,0,      1,1,1,1,      0,0,0,0,
       0,0,0,0,      0,0,0,0,      1,1,1,1,      0,0,0,0,
       1,1,1,1,      1,1,1,1,      0,0,0,0,      1,1,1,1,
       1,1,1,1,      1,1,1,1,      0,0,0,0,      0,0,0,0,
       1,1,1,1,      0,0,0,0,],
[0,0,0,0,0,0,  0,0,0,0,0,0,   0,0,0,0,0,0,   1,1,1,1,1,1,   0,0,0,0,0,0,
       0,0,0,0,0,0,   0,0,0,0,0,0,   1,1,1,1,1,1,   1,1,1,1,1,1,
       0,0,0,0,0,0,   1,1,1,1,1,1,   1,1,1,1,1,1,   1,1,1,1,1,1,
       1,1,1,1,1,1,   0,0,0,0,0,0,   0,0,0,0,0,0,   1,1,1,1,1,1,
       0,0,0,0,0,0],
 [0,0,0,0,0,0,  0,0,0,0,0,0,   1,1,1,1,1,1,   0,0,0,0,0,0,   1,1,1,1,1,1,
       0,0,0,0,0,0,   0,0,0,0,0,0,   1,1,1,1,1,1,   1,1,1,1,1,1,
       1,1,1,1,1,1,   0,0,0,0,0,0,   1,1,1,1,1,1,   0,0,0,0,0,0,
       0,0,0,0,0,0,   0,0,0,0,0,0,   1,1,1,1,1,1,   0,0,0,0,0,0,
       1,1,1,1,1,1],
 [0,0,0,0,0,0,  1,1,1,1,1,1,   1,1,1,1,1,1,   0,0,0,0,0,0,   1,1,1,1,1,1,
       0,0,0,0,0,0,   0,0,0,0,0,0,   1,1,1,1,1,1,   1,1,1,1,1,1,
       1,1,1,1,1,1,   0,0,0,0,0,0,   0,0,0,0,0,0,   0,0,0,0,0,0,
       1,1,1,1,1,1,   0,0,0,0,0,0,   0,0,0,0,0,0,   1,1,1,1,1,1,
       0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,
       1,1,1,1,1,1,1,1,1,1,1,1,      0,0,0,0,0,0,0,0,0,0,0,0,
       0,0,0,0,0,0,0,0,0,0,0,0,      0,0,0,0,0,0,0,0,0,0,0,0,
       0,0,0,0,0,0,0,0,0,0,0,0,      0,0,0,0,0,0,0,0,0,0,0,0,
       0,0,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,      0,0,0,0,0,0,0,0,0,0,0,0,
       1,1,1,1,1,1,1,1,1,1,1,1,      0,0,0,0,0,0,0,0,0,0,0,0,
       1,1,1,1,1,1,1,1,1,1,1,1,      1,1,1,1,1,1,1,1,1,1,1,1,
       0,0,0,0,0,0,0,0,0,0,0,0,      1,1,1,1,1,1,1,1,1,1,1,1,
       0,0,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,      0,0,0,0,0,0,0,0,0,0,0,0,
       1,1,1,1,1,1,1,1,1,1,1,1,      1,1,1,1,1,1,1,1,1,1,1,1,
```

```
        0,0,0,0,0,0,0,0,0,0,0,0,       1,1,1,1,1,1,1,1,1,1,1,1,
        1,1,1,1,1,1,1,1,1,1,1,1,       1,1,1,1,1,1,1,1,1,1,1,1,
        1,1,1,1,1,1,1,1,1,1,1,1],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
        1,1,1,1,1,1,1,1,1,1,1,1],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
        1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
        1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
        1,1,1,1,1,1,1,1,1,1,1,1]
```

```
]

# Calcular el consumo de energía por día para los 6 aireadores con
intervalo de 5 minutos
consumos_diarios_5min = []
for i, patron in enumerate(patrones_aireadores_5min):
    consumos_diarios_aireador = [calcular_consumo_energia(patron,
aireadores[i], tiempo, 5) for tiempo in range(len(patron))]

    consumos_diarios_5min.append(consumos_diarios_aireador)

# Crear gráfica que muestra el consumo de energía por día para los 6
aireadores (5 minutos)
dias = range(1, len(patrones_aireadores_5min[0]) + 1)

for i, consumos_diarios_aireador in enumerate(consumos_diarios_5min):
    plt.plot(dias, consumos_diarios_aireador,
label=aireadores[i]["nombre"])

plt.xlabel("Día")
plt.ylabel("Consumo de Energía Diario (kWh)")
plt.title("Consumo de Energía Diario de los Aireadores")
plt.grid(True)
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0.)
plt.show()

# Calcular el consumo de energía por día para los 6 aireadores con
intervalo de 5 minutos
```

```
consumos_diarios_5min = []
for i, patron in enumerate(patrones_aireadores_5min):
    consumo_diario_total = calcular_consumo_energia(patron,
aireadores[i], len(patron) - 1, 5)
    consumos_diarios_5min.append(consumo_diario_total)

# Crear gráfica de barras que muestra el consumo total por día para los 6
aireadores (5 minutos)
bar_positions = np.arange(len(aireadores))

plt.bar(bar_positions, consumos_diarios_5min, align='center')
plt.xlabel("Aireador")
plt.ylabel("Consumo Total Diario (kWh)")
plt.title("Consumo Total Diario por Aireador (Intervalo de 5 minutos)")
plt.xticks(bar_positions, [aireador["nombre"] for aireador in
aireadores], rotation=45, ha='right')
plt.grid(True)
plt.show()

# Crear un DataFrame con la información de los aireadores y los consumos
totales diarios
data = {
    "Nombre aireador": [aireador["nombre"] for aireador in aireadores],
    "Potencial nominal": [aireador["potencia_nominal"] for aireador in
aireadores],
    "Voltaje Nominal": [aireador["tensión_nominal"] for aireador in
aireadores],
    "Potencia generada (IxV)": [aireador["potencia_nominal"] *
aireador["tensión_nominal"]/1000 for aireador in aireadores],
}

data2 = {
    "Nombre aireador": [aireador["nombre"] for aireador in aireadores],
    "Consumo de energía diario": consumos_diarios_5min,
    "Consumo de energía Mensual": [consumo * 30 for consumo in
consumos_diarios_5min],  # Suponiendo 30 días por mes
    "Consumo de energía anual": [consumo * 365 for consumo in
consumos_diarios_5min],  # Suponiendo 365 días por año
}
df = pd.DataFrame(data)
df2 = pd.DataFrame(data2)
# Mostrar el DataFrame como tabla
#print(df)
#print(df2)

# Unir los DataFrames df y df2 en uno solo
df = pd.merge(df, df2, on="Nombre aireador")

# Crear gráfica de barras para el consumo mensual
plt.figure(figsize=(12, 6))
plt.bar(bar_positions, df["Consumo de energía Mensual"], width=bar_width,
label='Mensual', color='blue')
```

```
plt.xlabel("Aireador")
plt.ylabel("Consumo Mensual (kWh)")
plt.title("Consumo Mensual por Aireador")
plt.xticks(bar_positions, df["Nombre aireador"], rotation=45, ha='right')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# Crear gráfica de barras para el consumo anual
plt.figure(figsize=(12, 6))
plt.bar(bar_positions, df["Consumo de energía anual"], width=bar_width,
label='Anual', color='green')

plt.xlabel("Aireador")
plt.ylabel("Consumo Anual (kWh)")
plt.title("Consumo Anual por Aireador")
plt.xticks(bar_positions, df["Nombre aireador"], rotation=45, ha='right')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

df = pd.DataFrame(data)
df2 = pd.DataFrame(data2)
# Mostrar el DataFrame como tabla
print(df)
print(df2)
```

Anexo 14. Algoritmo para el cálculo y el graficado de costos de operación

```
# Agregar sección para calcular el costo en pesos mexicanos con un costo de kWh de 2 pesos
costo_por_kwh = 2  # Costo por kWh en pesos mexicanos

# Calcular el costo diario para los 6 aireadores con intervalo de 5 minutos
costos_diarios_5min = [consumo_diario * costo_por_kwh for consumo_diario in consumos_diarios_5min]

# Crear gráfica de barras que muestra el costo total por día para los 6 aireadores (5 minutos)
plt.bar(bar_positions, costos_diarios_5min, align='center', color='orange')
plt.xlabel("Aireador")
plt.ylabel("Costo Total Diario (MXN)")
plt.title("Costo Total Diario por Aireador (Intervalo de 5 minutos)")
plt.xticks(bar_positions, [aireador["nombre"] for aireador in aireadores], rotation=45, ha='right')
plt.grid(True)
plt.show()

# Crear DataFrame con información de costos
data3 = {
    "Nombre aireador": [aireador["nombre"] for aireador in aireadores],
    "Costo diario": costos_diarios_5min,
    "Costo mensual": [costo_diario * 30 for costo_diario in costos_diarios_5min],
    "Costo anual": [costo_diario * 365 for costo_diario in costos_diarios_5min],
}
df3 = pd.DataFrame(data3)

# Unir el DataFrame df con df2 y df3
df = pd.merge(df, df2, on="Nombre aireador")
df = pd.merge(df, df3, on="Nombre aireador")

# Mostrar el DataFrame completo

# Crear gráfica de barras para el costo mensual
plt.figure(figsize=(12, 6))
plt.bar(bar_positions, df["Costo mensual"], width=bar_width, label='Mensual', color='blue')

plt.xlabel("Aireador")
plt.ylabel("Costo Mensual (MXN)")
plt.title("Costo Mensual por Aireador")
plt.xticks(bar_positions, df["Nombre aireador"], rotation=45, ha='right')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
# Crear gráfica de barras para el costo anual
plt.figure(figsize=(12, 6))
plt.bar(bar_positions, df["Costo anual"], width=bar_width, label='Anual',
color='green')

plt.xlabel("Aireador")
plt.ylabel("Costo Anual (MXN)")
plt.title("Costo Anual por Aireador")
plt.xticks(bar_positions, df["Nombre aireador"], rotation=45, ha='right')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

print(df3)
```