Federico Horacio Nevado

Análisis de calidad del agua a través de Técnicas de Aprendizaje Automático

2024

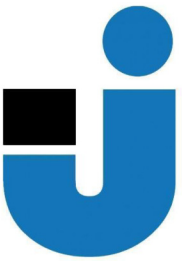
Instituto de Ingeniería y Agronomía

Carrera: Ingeniería en Informática

Universidad Nacional Arturo Jauretche

Instituto de Ingeniería y Agronomía

Carrera de Ingeniería en Informática



PRÁCTICA PROFESIONAL SUPERVISADA
Informe final

Análisis de calidad del agua a través de Técnicas de Aprendizaje Automático

Federico Horacio Nevado

Florencio Varela, abril 2024

Estudiante

Federico Horacio Nevado

federico.nevado@gmail.com

Organización donde se realiza la Práctica Profesional Supervisada

Universidad Nacional Arturo Jauretche

Av. Calchaquí N° 6200, Florencio Varela, (1888) Buenos Aires, Argentina

+54 11 4275 6100

Sector: Programa TICAPPS (Tecnologías de la Información y la Comunicación en Aplicaciones de Interés Social), Instituto de Ingeniería y Agronomía

Tutor organizacional

Dr. Ing. Marcelo Cappelletti

mcappelletti@unaj.edu.ar

Docentes Supervisores

Ing. Atía, Julissa

julissa.atia@gmail.com

Lic. Suárez, Gabriela

gabisuarez04@gmail.com

Docente tutor del Taller de Apoyo para la Producción de Textos Académicos

Lic. Carolina Kelly

kellygcarolina@gmail.com

Coordinador de la Carrera de Ingeniería en Informática

Dr. Ing. Martín Morales

martin.morales@unaj.edu.ar

1. Resumen

El presente trabajo se enfoca en el análisis de muestras de agua del Río de La Plata con el fin de determinar su calidad, centrándose específicamente en la detección de presencia de cianobacterias y su concentración a través de técnicas de Inteligencia Artificial. Las cianobacterias representan una particular amenaza para la salud humana, ya que pueden producir toxinas perjudiciales y, al mismo tiempo, afectan al ecosistema en general, degradando la biodiversidad. Durante el desarrollo de este trabajo se emplearon tres modelos de aprendizaje automático supervisado: Redes Neuronales Artificiales (RNA), Árboles de Decisión y el algoritmo de KNN (*k-Nearest Neighbors).* Cada uno de estos modelos será evaluado y comparado, utilizando diferentes métricas de efectividad de modelos de predicción, con el fin de estimar la concentración de cianobacterias de la manera más eficaz posible.

Para el entrenamiento y testeo de los modelos se utilizaron datos obtenidos desde 18 puntos de recolección ubicados sobre las costas del río, desde la desembocadura de Río Luján y Punta Piedras, zona que abarca más de 170 km de costa. Los resultados obtenidos muestran una fiabilidad medio-alta en la predicción de altas concentraciones de cianobacterias en los tres modelos, siendo el modelo de Árboles de Decisión el más efectivo con un 92% de exactitud. Con este modelo se espera poder realizar predicciones tempranas que permitan obtener medidas de manera más efectiva para mitigar los efectos nocivos que las cianobacterias tienen tanto en la calidad de vida de las personas como en el medio ambiente.

2. Introducción

El Río de la Plata es lo que se conoce como un estuario, un área donde un río se encuentra con el mar o un océano, y donde el agua dulce proveniente del río se mezcla con el agua salada del mar. Abarca 35.500 km2, de los cuales el 37% está ocupado por agua dulce, naturalmente rica en nutrientes y dominada por plancton.

Dentro de un estuario la mezcla con el ambiente marino y los intercambios entre la interfase sedimento-agua inducen variabilidad física y biogeoquímica en la columna de agua, e incluyen variaciones en la salinidad y de nutrientes, ya que el encuentro de aguas dulces del río y aguas saladas del mar crean un ambiente único con características físicas, químicas y biológicas particulares. Los estuarios son ecosistemas muy importantes y productivos que albergan una gran variedad de vida marina, incluyendo peces, aves, mamíferos, crustáceos y otras formas de vida. Estos ambientes proporcionan hábitats vitales para muchas especies, ya que sirven como áreas de reproducción, alimentación y refugio. Además tienen un papel crucial en la purificación y filtración del agua, así como en la protección de las áreas costeras contra inundaciones y tormentas al absorber y reducir la fuerza de las olas. También son importantes para la recreación y actividades económicas, como la pesca y el turismo. Es por ello que al menos la mitad de la población mundial vive en zonas estuariales, y esta proporción continúa creciendo. En el caso del Río de La Plata, más de once millones de personas viven en sus costas.

Sin embargo, los estuarios están bajo amenaza debido a la contaminación, la urbanización, la degradación del hábitat y otros impactos humanos. Así las aguas pueden ser vehículo de contaminantes físicos, químicos y biológicos, que llegan a las mismas por el vertido directo de efluentes cloacales, industriales o de la agricultura, con escaso o nulo tratamiento. La costa del Río de la Plata ha estado expuesta a estas amenazas durante décadas, debido al desarrollo urbano e industrial acelerado. La mayoría de los establecimientos industriales

correspondientes al área metropolitana de Buenos Aires y al Gran La Plata vuelcan sus efluentes directamente a los ríos, arroyos y canales receptores que finalmente desaguan en la costa. Es notorio el cambio sufrido en la línea de costas sobre la ribera, desde la zona del Delta hasta Punta Piedras, no solo por acción humana sino también por acción de fenómenos naturales como las sudestadas y la erosión causada por la acción mecánica de las mareas.

Los efectos de los diferentes aportes de contaminantes sobre los sedimentos de la Franja Costera Sur correspondientes al área de estudio tienen extrema importancia para el análisis de la calidad del agua. En tal sentido, es posible identificar a lo largo de la Franja Costera Sur varios canales, ríos y desagües cuya carga de contaminantes genera problemas importantes en la costa, entre ellas podemos mencionar el río Luján que recibe los aportes del río Reconquista, que luego de la Cuenca del Matanza-Riachuelo es la más industrializada. Justamente, en el caso de la cuenca Matanza-Riachuelo, son vertidos aproximadamente 5 m3/s de efluentes cloacales con un escaso tratamiento previo, además de efluentes industriales correspondientes a aproximadamente 800 industrias altamente contaminantes (que incluyen los del Polo Petroquímico de Dock Sud) y contaminantes derivados de la actividad portuaria (de los puertos de Buenos Aires y Dock Sud), que asimismo aportan una alta carga de metales pesados, entre los que se destaca el cromo. En el mismo sentido, el Colector Mayor de Berisso, el río Santiago (Arroyo del Gato-Zanjón y Canal Oeste) y los canales Santo Domingo y Sarandí conducen efluentes, tanto de origen industrial (como el caso del Polo Petroquímico de la Ciudad de La Plata), como cloacal y derivados de la actividad portuaria (el Puerto de La Plata) con un escaso tratamiento previo, y vierten sus aguas sobre la costa. Estas descargas, en conjunto, aportan más del 80% del total de la carga de contaminantes que ingresa a la Franja Costera Sur. Principalmente, todos estos efluentes tienen mayor incidencia entre la línea de costa y los 2.000 m, y afectan a la Franja Costera Sur desde el punto de vista ecológico, económico y paisajístico.

Otra de las amenazas a este ecosistema lo constituyen las floraciones algales nocivas, que en el Río de la Plata son cada vez más frecuentes y pueden producir impactos drásticos sobre los recursos pesqueros, la biodiversidad en general, la salud pública y, consecuentemente, sobre el uso de espacios costeros para recreación, turismo y/o como fuente de agua potable.

Con el propósito de contribuir en la evaluación de la calidad del agua en la Franja Costera Sur del Río de la Plata, en este trabajo, se presenta un análisis para la predicción de la concentración de cianobacterias potencialmente tóxicas presentes en el agua, a partir del conocimiento de datos físico-químicos y biológicos de muestras de agua. La investigación se llevó a cabo mediante la utilización y comparación de tres algoritmos de aprendizaje automático supervisado de regresión y clasificación. Los algoritmos utilizados fueron: Redes Neuronales Artificiales (RNA), Árboles de Decisión y el algoritmo de K-Vecinos más Cercanos (K-NN: K-Nearest Neighbor).

El artículo está estructurado de la siguiente manera: en la próxima sección, se presenta la problemática de las floraciones de cianobacterias en el Río de la Plata. Luego, se establece el marco teórico de los principales conceptos de las herramientas de aprendizaje automático utilizados en este estudio. A continuación, se resume la metodología de trabajo utilizada. Finalmente, se presentan los resultados y las principales conclusiones obtenidas.

3. Floraciones de cianobacterias

Las cianobacterias son uno de los organismos más antiguos de la Tierra y desempeñan un papel importante en los ecosistemas acuáticos y terrestres. Si bien no son algas propiamente dichas, las cianobacterias son una parte importante del fitoplancton, también conocidas como algas verde-azules. Su nombre deriva de la presencia de pigmentos azulados que intervienen en el proceso de captación de la luz para la fotosíntesis. A través de este proceso, en el que capturan dióxido de carbono y liberan oxígeno, es que pueden convertir la luz solar en energía química. Estas pueden observarse como una capa verdosa en la superficie del agua, y requieren especial atención, ya que algunas cianobacterias pueden producir toxinas, conocidas como cianotoxinas que son perjudiciales para los organismos acuáticos y la salud humana.

Las cianobacterias también poseen una gran capacidad de adaptación a los cambios del ambiente, y juegan un rol central en el proceso de eutrofización. La eutrofización es un proceso natural o inducido por el ser humano en el que cuerpos de agua, como lagos, ríos y estanques, acumulan una cantidad excesiva de nutrientes, especialmente nitrógeno y fósforo alterando significativamente la estructura y el funcionamiento de los ecosistemas acuáticos. Estos nutrientes provienen de diversas fuentes, como la escorrentía de fertilizantes agrícolas, las aguas residuales urbanas, los desechos industriales y la erosión del suelo. Gracias al exceso de la cantidad de nutrientes, las plantas acuáticas y el fitoplancton experimentan un crecimiento acelerado de su población y proliferaciones masivas, lo que se conoce como “bloom”, y puede generarse en períodos que van desde pocas horas a varios días, y desaparecer en un plazo similar No obstante, algunas floraciones pueden permanecer por períodos más largos, como todo el verano, o incluso establecerse de forma permanente. Como consecuencia de este crecimiento en las poblaciones de algas y cianobacterias, la turbidez del agua aumenta afectando negativamente a la vida acuática y la capacidad de penetración de la

luz solar. Finalmente, las algas mueren y con su descomposición consumen el oxígeno disuelto en el agua, lo que es perjudicial para la fauna acuática pudiendo generar lo que se conocen como "zonas muertas". Además, durante el proceso de descomposición de las cianobacterias se liberan toxinas en el agua, lo que La exposición a estas toxinas, a través del contacto directo o la ingesta directa de agua o el consumo de organismos acuáticos afectados, puede causar problemas de salud que van desde irritaciones cutáneas hasta daños en el hígado y el sistema nervioso que incluso pueden llegar a ser letales.

Para evitar los efectos negativos que estas floraciones tienen sobre la vida humana se utilizan diversos mecanismos. En primer lugar, se trabaja sobre la prevención, cuidando el uso de los suelos y la actividad humana para evitar que se generen condiciones de eutrofización, que tienen un fuerte impacto en la floración de cianobacterias. Por otro lado, es importante trabajar en la mitigación de sus efectos nocivos. Se pueden aplicar métodos de control biológico, mediante los que se emplean otros organismos que actúan como predadores de cianobacterias con el fin de reducir su población; también, tratamientos químicos como plaguicidas, que pueden ayudar al control de la proliferación, pero que también tienen efectos secundarios sobre la calidad del agua; o también, a través de la utilización de sistemas de filtrado por coagulación, filtración directa, procesos de membrana, flotación por aire disuelto, etc. Para el tratamiento de aguas contaminadas, siempre es importante la detección temprana que permita actuar de forma ágil y eficiente.

En este sentido, numerosos estudios se han llevado a cabo previamente para el análisis de las floraciones de cianobacterias sobre el Río de la Plata. En particular, en el trabajo "Empleo de descriptores fitoplanctónicos como biomonitores en la evaluación de la calidad del agua en la costa del río de la Plata (Franja Costera Sur)" de Sathicq, Maria Belén (2017), se ha llevado a cabo un estudio de los patrones y procesos que ocurren a nivel comunitario, y a distintas escalas temporales (semanal, diaria y horaria), para conocer la dinámica y los principales

factores que modulan e influencian la presencia de cianobacterias en la costa del Río de la Plata. Los resultados obtenidos indicaron que aquellas variables forzantes primarias para el desarrollo de estos organismos fueron la temperatura, el pH, la penetración de luz (turbidez y radiación fotosintéticamente activa), la conductividad y los nutrientes (particularmente NO3- y NH4+). Sumado a esto, los vientos provenientes del Sudeste y la marea alta fueron aquellas variables que empujaron la masa de agua, y con ella a las cianobacterias, hacia la costa.

Se debe destacar que hasta el momento no se han utilizado técnicas basadas en aprendizaje automático para la predicción de la concentración de cianobacterias potencialmente tóxicas sobre muestras de agua del Río de la Plata. Gracias al crecimiento de las capacidades de cálculo y al mejoramiento de los algoritmos implementados, este trabajo pretende aportar conocimiento en este campo de estudio a partir de estas herramientas, que presentan bajo costo de cómputo, facilidad de implementación y buena velocidad de convergencia.

4. Inteligencia Artificial

La inteligencia artificial (IA) es un campo de la informática que se centra en la creación de sistemas y programas de computadora capaces de imitar la inteligencia humana. Estas tareas incluyen el aprendizaje, la resolución de problemas, la toma de decisiones, el reconocimiento de patrones, el procesamiento del lenguaje natural y la percepción visual, entre otras. La IA se basa en la idea de que las máquinas pueden ser programadas para imitar la capacidad cognitiva humana y, en algunos casos, superarla en términos de velocidad y precisión en tareas específicas. Para lograr esto, se utilizan diversas técnicas y enfoques, como el aprendizaje automático, el procesamiento del lenguaje natural y la visión por computadora. El concepto de Inteligencia Artificial (IA) se aplica a cualquier técnica que permita a las computadoras imitar la inteligencia humana a través de expresiones lógicas y esquemas abstractos.

La inteligencia artificial tiene aplicaciones en una variedad de campos, como la medicina, la industria, la automatización, la ciencia, la educación y muchos otros. Su impacto en la sociedad es cada vez más relevante, y su desarrollo continúa avanzando con el tiempo.

4.1 Aprendizaje Automático

El aprendizaje automático (Machine Learning en inglés) es una subdisciplina de la inteligencia artificial que se centra en el desarrollo de algoritmos y modelos matemáticos que permiten a las computadoras aprender y mejorar su rendimiento en tareas específicas a través de la experiencia, sin necesidad de ser programadas explícitamente. En otras palabras, el aprendizaje automático permite a las máquinas mejorar su desempeño en una tarea a medida que se exponen a más datos y experiencias a través de la detección automática de patrones relevantes. En este sentido, a medida que el modelo identifica los distintos patrones a utilizar para la categorización o predicción, se dice que este “aprende”.

El proceso de aprendizaje automático implica:

1. Recopilación de datos: Se comienza por recopilar una gran cantidad de datos relevantes para la tarea que se desea automatizar o mejorar.

2. Preprocesamiento de datos: Los datos recopilados a menudo deben limpiarse y prepararse para su uso, lo que puede implicar eliminar valores atípicos, corregir datos faltantes y normalizar la información.

3. Selección de algoritmos: Se elige el algoritmo de aprendizaje automático adecuado para el problema en cuestión. Hay diversos tipos de algoritmos de aprendizaje automático, como regresión, clasificación, agrupamiento, redes neuronales, entre otros.

4. Entrenamiento del modelo: Se utiliza un conjunto de datos de entrenamiento para que el algoritmo ajuste sus parámetros y mejore su capacidad para realizar la tarea. Durante este proceso, el modelo busca patrones y relaciones en los datos.

5. Evaluación y ajuste: Se utiliza un conjunto de datos de prueba o validación para evaluar el rendimiento del modelo. Si el rendimiento no es satisfactorio, se pueden realizar ajustes en los hiper-parámetros del algoritmo o en el conjunto de datos de entrenamiento.

6. Despliegue: Una vez que el modelo está entrenado y evaluado satisfactoriamente, se puede implementar en aplicaciones o sistemas para realizar tareas específicas, como el reconocimiento de patrones, el procesamiento del lenguaje natural, la detección de fraudes, la toma de decisiones, entre otros.

El aprendizaje automático tiene aplicaciones en una amplia gama de campos, desde la medicina y la industria hasta la investigación científica y la tecnología. Se ha convertido en una herramienta poderosa para automatizar tareas, realizar predicciones basadas en datos y mejorar la eficiencia en diversas áreas. Actualmente esta tecnología se encuentra desplegada en múltiples plataformas que utilizamos de manera cotidiana, como servicios de *streaming* de

video o audio, motores de búsqueda de internet, reconocimiento de voz para escritura y reconocimiento facial en las cámaras, conducción automática de vehículos, etc.

4.2 Aprendizaje supervisado, no supervisado y por refuerzo.

Dentro del aprendizaje automático se distinguen tres tipos principales de técnicas: las de aprendizaje supervisado, aprendizaje no supervisado y aprendizaje por refuerzo. La diferencia radica principalmente en el método de entrenamiento de los modelos así como también en el tipo de problemas que abordan y pretenden resolver.

4.2.1 Aprendizaje supervisado

En las técnicas que utilizan aprendizaje supervisado se utilizan muestras previamente etiquetadas. En este caso se cuenta con un set amplio de datos en el que ya previamente fueron mapeadas las entradas del modelo con una salida correcta. Lo que intenta entonces el modelo es aprender a relacionar las entradas proporcionadas con las etiquetas que cada una de ellas posee a fin de poder realizar predicciones sobre nuevas muestras que no contengan esta salida. En este proceso se requiere que una persona asigne etiquetas (ya sea de manera manual o automática) al set de datos que va a utilizar de entrenamiento.

Para poder evaluar la efectividad del modelo a la hora de aprender estas relaciones entre entradas y salidas se utilizan las llamadas funciones de pérdida. Lo que se hace es medir con cuánta efectividad el modelo logró predecir la etiqueta a la cual corresponde. A la hora de entrenar el modelo, se divide el set de datos en dos grupos: por un lado, los datos de entrenamiento y, por otro lado, los datos de testeo. El primer grupo de datos se utiliza para que el modelo aprenda la relación entre los datos de entrada y de salida. Una vez el modelo fue entrenado, se utiliza el segundo grupo de datos. En primer lugar, se descartan de este grupo de datos la etiqueta que cada uno de ellos tiene asignada y se intenta predecir, utilizando el modelo, a qué etiqueta se corresponde cada una de las muestras; luego se

compara el resultado obtenido con las etiquetas reales para determinar la efectividad del modelo. En ese sentido se puede entrenar el modelo múltiples veces intentando disminuir la función de pérdida y logrando un modelo que se ajuste con mayor efectividad al problema.

Este enfoque de entrenamiento es utilizado en múltiples escenarios, como puede ser la clasificación de imágenes, la traducción de voz a texto, la recomendación de productos, el diagnóstico médico, etc. Es un enfoque con gran capacidad siempre que se cuente con datos previamente etiquetados.

Al mismo tiempo, dentro de esta categoría existen dos variantes dependiendo del tipo de resultado que se desea obtener:

- Técnicas de regresión: en estos modelos se obtiene un valor numérico dentro de un conjunto infinito de posibilidades. Se utiliza especialmente cuando se intenta encontrar un valor en particular.
- Técnicas de clasificación: en los modelos de clasificación el resultado es una clase o categoría a la que corresponde la muestra. Se utilizan en problemas en los que se pretende enmarcar las muestras dentro de categorías definidas. En este modelo las posibles salidas son finitas, dentro de un conjunto de posibilidades.

4.2.2 Aprendizaje no Supervisado

El aprendizaje no supervisado es un enfoque utilizado para entrenar modelos sin la necesidad de que estos se encuentren previamente etiquetados. En este enfoque no es necesario aportar información al modelo acerca de la etiqueta o categoría a la que corresponde cada una de las muestras utilizadas para su entrenamiento.

Este enfoque, a su vez, cuenta con dos posibilidades. Por un lado, está lo que se conoce como Agrupamiento o *Clustering*. En este escenario lo que se hace es agrupar las muestras basándose en sus similitudes. Es decir, se generan grupos de muestras utilizando como

criterio la cercanía de unos a otros, en tanto cada una de sus dimensiones sean similares entre sí. Por ejemplo, es de esperar que dos muestras de agua con una temperatura, ph y conductividad muy similares se encuentren dentro del mismo grupo de muestras. Por otro lado, está lo que se conoce como reducción de la dimensionalidad. Con esta técnica lo que se intenta es reducir la cantidad de dimensiones de una muestra perdiendo la menor cantidad de información posible a fin de proyectarla en un espacio de menores dimensiones y así generar grupos de esa manera.

El Aprendizaje no Supervisado es especialmente útil para tareas como exploración de datos, detección de anomalías, segmentación de muestras, comprensión de patrones, etc.

4.2.3 Aprendizaje por Refuerzo

El Aprendizaje por Refuerzo es un enfoque centrado en recompensas para un modelo de IA. Lo que se hace es asignar tareas al modelo y definir recompensas para cuando el resultado obtenido sea positivo; de esta manera el modelo irá refinando su funcionamiento a fin de obtener la mayor cantidad de recompensas posibles. Así se puede prescindir de datos de entrenamiento y el modelo "aprende" de su propio comportamiento.

Este proceso involucra fundamentalmente tres componentes:

- Agente: es el programa o modelo de IA que toma las decisiones y realiza acciones
- Entorno: es el escenario donde se desenvuelve el agente
- Recompensa: son las señales que recibe el agente de que está realizando su tarea de manera satisfactoria.

Este enfoque es ampliamente utilizado en una variedad de aplicaciones, como sistemas de control, toma de decisiones en robótica, gestión de carteras financieras, comportamiento de agentes en videojuegos, etc.

5. Metodología

La metodología empleada en este trabajo es la que, en general, se usa en el análisis mediante herramientas de Aprendizaje Automático, la cual comprende las etapas de recolección de los datos, su tratamiento y procesado, el análisis exploratorio de estos, el entrenamiento y testeo de modelos de Aprendizaje Automático y, finalmente, la interpretación de los resultados obtenidos. A continuación, se describen las etapas mencionadas previamente.

6. Recolección de datos

La fase de recolección de datos para entrenamiento de los modelos puede tener múltiples fuentes, desde sensores electrónicos o bases de datos públicas hasta páginas web libres. Para este trabajo se utilizaron muestras que fueron obtenidas directamente de 18 puntos de muestreo situados sobre la costa de la Franja Costera Sur del Río de la Plata, entre la desembocadura del río Luján y Punta Piedras.

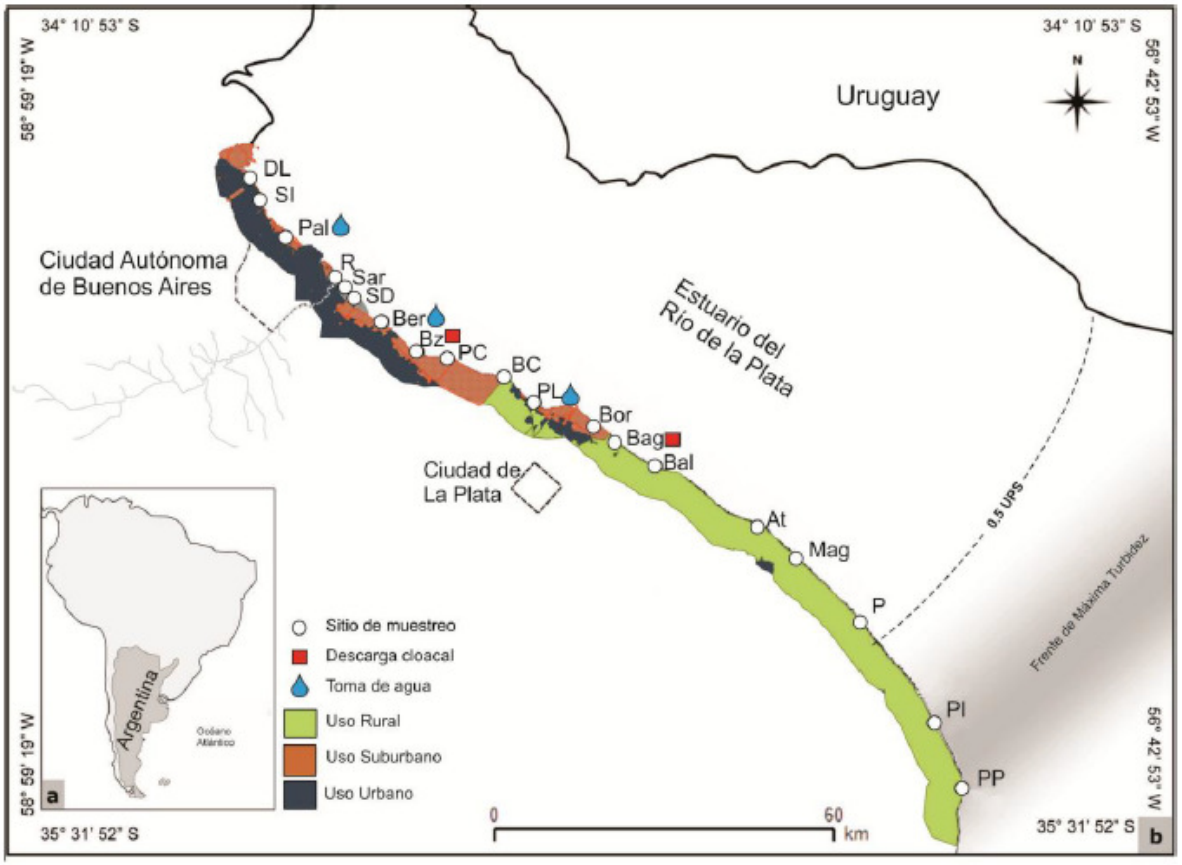


Figura 1. Los 18 puntos de muestreo de la Franja Costera del Río de La Plata. Fuente: Lic. María Belén Sathicq (2017).

Los puntos de muestreo seleccionados abarcan 170 kilómetros de costa y comprenden una variedad de distintos usos del suelo (rural, suburbano y urbano) y del agua (descargas cloacales y tomas de agua), tanto desde un punto de vista económico como recreacional y cultural. Al mismo tiempo, los puntos de muestreo localizados en el sector superior de la zona de estudio (Desembocadura del río Luján, San Isidro, Palermo, Sarandí, Santo Domingo

y Bernal) están directamente expuestos al impacto de la Ciudad de Buenos Aires donde se llevan a cabo actividades portuarias y se descargan efluentes domésticos e industriales. En el punto de recolección ubicado en Berazategui se encuentra la descarga de aguas residuales de la Ciudad de Buenos Aires (a 2500 metros de la costa), y el sitio Bagliardi es el correspondiente para la ciudad de La Plata.

También es importante destacar que dentro del área de muestreo se encuentran las tres torres de extracción que proporcionan de agua a las plantas potabilizadoras (Palermo, Bernal y Punta Lara) que proveen de aproximadamente 6.000.000 m^3/día de agua potable a más de 11 millones de personas.

Por su parte, en los sitios de muestreo localizados en el sector inferior de la zona de estudio (Balandra, Atalaya, Magdalena, Pearson, Punta Indio y Punta Piedras), dominan las actividades recreativas y la pesca a pequeña escala, aunque también se encuentran expuestos a las distintas actividades rurales que se realizan en el entorno.

Como mencionamos, todos los puntos de muestreo se encuentran expuestos a una variedad de actividades que afectan la calidad del agua de diferentes maneras. En cada uno de ellos se tomaron muestras de agua sobre la línea de la costa así como también dentro de la misma a 500, 1500 y 3000 metros. Todos los muestreos se realizaron entre los años 2008 y 2023, y se obtuvo información de las principales características físico-químicas y biológicas del agua de cada punto, tales como la conductividad, el pH, la temperatura, la turbidez, el oxígeno disuelto, la demanda química y biológica de oxígeno y nutrientes de la familia del nitrógeno y fosfato.

Toda esta información fue volcada en diversas planillas de cálculo que posteriormente fue analizada para su uso con las distintas técnicas de IA.

7. Tratamiento y procesado de datos

Como se refirió antes, los datos obtenidos de los puntos de recolección se encontraban ya disponibles en una serie de planillas de cálculo en las que se detallan distintas dimensiones de cada uno.

7.1 Unificación de fuentes de datos

Nuestro primer objetivo en este sentido fue entonces combinar las planillas de cálculo en una única matriz que contuviera todas las variables posibles, agrupándolas por fecha y lugar de la muestra, ya que en distintos archivos se encontraban disponibles datos que hacían referencia a una misma muestra (por ejemplo, los datos físico-químicos y los datos biológicos de la misma muestra se encontraban en distintos archivos). De esta manera, se reagruparon las distintas fuentes a partir de la consideración de la fecha y el sitio donde fue obtenida la muestra.

Del *dataset* obtenido tras la combinación de todas las fuentes se descartaron todas aquellas muestras para las cuales no se contaba con suficiente cantidad de datos para incluirlas en el procesamiento así como también aquellas muestras que incluían valores atípicos.

7.2 Completitud de los datos

El primer paso para poder utilizar el *dataset* en el proceso de entrenar el modelo de Inteligencia Artificial es decidir qué dimensiones del mismo vamos a utilizar. Para ello comenzamos analizando cuáles son las columnas que contienen mayor completitud de datos, es decir, en cuáles columnas contamos con mayor cantidad de filas completas así como también una mayor diversidad ya que es necesario que haya una cantidad de muestras equivalentes para los distintos valores de cianobacterias que podemos encontrar. Las columnas seleccionadas fueron: la turbidez, el ph, la conductividad, la temperatura y los caudales de los ríos Palmas y Guazú. Este *dataset* cuenta con 775 muestras.

Si bien las columnas seleccionadas eran las que mayor completitud de datos aportaban, no estaban completas al 100%. Para poder aprovechar esas muestras, sin necesidad de descartarlas por la falta de datos, es comúnmente utilizada la técnica de rellenado utilizando imputación de valores a través de una estadística descriptiva, tales como la media, promedio o moda. En este caso en particular, optamos por utilizar la media como técnica de completado de los datos, ya que es el valor más neutral a la hora de afectar el comportamiento del modelo, debido a su poca exposición a los valores extremos o atípicos.

En simultáneo también se confeccionó un segundo *dataset* que incorpora, además, las columnas de datos físico-químicos: la clorofila, el oxígeno disuelto, la demanda química de oxígeno, la demanda biológica de oxígeno, la presencia de no2, no3 y po4. Este segundo set de datos consta de 320 filas, una diferencia considerable que impacta directamente en la calidad de los modelos que pueden ser obtenidos. Dada la diferencia en la cantidad de muestras, se decidió centrar el esfuerzo en el primer *dataset* que contiene 775 muestras. El segundo *dataset*, con más columnas pero menos registros, será analizado al final del trabajo utilizando la metodología con mejores resultados para el primero.

7.3 Normalización

Otro aspecto importante a la hora de optimizar un *dataset* para ser utilizado en el proceso de entrenamiento de un modelo de IA es normalizar sus valores. Su principal función es evitar que las características o variables del *dataset,* que poseen escalas distintas, distorsionen el comportamiento del modelo. Para ello se normalizan todas las columnas para que queden expresadas en valores de una misma escala (por ejemplo entre 0 y 1 o -1 y 1), sin afectar el significado de los datos. Esta homogeneización es importante para garantizar que ninguna característica tenga un impacto desproporcionado debido a su escala original.

Gracias a este proceso evitamos que una característica de dimensiones mucho mayores en relación al resto de las características del *dataset* domine el proceso de aprendizaje. Por

ejemplo, en nuestro caso, los caudales de Río Guazú y Río Palmas son características con varios órdenes de magnitud superiores a, por ejemplo, el ph o la temperatura de la muestra. Esta diferencia de magnitudes puede generar *ruidos* al momento de entrenar el modelo y finalmente afectar al desempeño general del mismo. La normalización permite que el modelo se enfoque en las relaciones intrínsecas entre las distintas dimensiones que componen el *dataset*, promoviendo un proceso de aprendizaje más efectivo y mejorando la capacidad del modelo para generalizar y realizar predicciones precisas sobre nuevas muestras.

8. Análisis exploratorio de los datos

Como se mencionó previamente el *dataset* obtenido de las fuentes de recolección de muestras cuenta con una amplia cantidad de características, tanto biológicas como fisicoquímicas, que guardan distinta relación con el proceso de floración de cianobacterias, desde nutrientes presentes en el agua hasta la temperatura o el pH de la misma. Si bien todas ellas tienen una relación estrecha con el bloom de cianobacterias, algunas de estas características son conocidas como variables respuesta. En nuestro *dataset* algunas de ellas son, por ejemplo, la clorofila o el oxígeno disuelto. Se dice que estas son variables respuesta porque no son características que propicien la floración de cianobacterias, sino que en realidad se ven alteradas luego de que la misma se haya producido. En el caso de la clorofila esta aumenta luego de que el proceso de floración y descomposición de las cianobacterias se haya producido, y el oxígeno disuelto se ve afectado también por el proceso de fotosíntesis que llevan adelante las cianobacterias.

Ya que el principal objetivo de este trabajo es intentar generar alertas tempranas, enfocamos nuestros esfuerzos en algunas de las variables que generan ambientes propicios para que las cianobacterias se desarrollen así como también aquellas características que se ven afectadas por la floración de las cianobacterias en etapas tempranas. Las características seleccionadas fueron: pH, temperatura, turbidez, conductividad, cianobacterias y caudales de ambos ríos (Paraná de las Palmas y Paraná Guazú).

El resto de las variables fueron descartadas en esta etapa del trabajo.

Como primer paso realizamos una pequeña descripción de las características seleccionadas del *dataset*:

Parámetro	Media	Desv. Estandar	Mínimo	Máximo
pH (UpH)	7.92	0.75	6.6	12.6

Temp. (°C)	20.64	5.76	4.9	35.1
Turbidez (NTU)	140.78	128.13	18.00	999.00
Conduct. (µS/cm)	366.16	757.13	37.00	14800.00
Cianobacterias (cél./mL)	2225.72	5897.30	0.00	67325.20
Caudal R. P. Palmas (m3/seg)	5325.16	764.94	4464.00	7499.00
Caudal R. P. Guazú (m3/seg)	10627.40	3591.44	7783.00	29499.00

Así como también analizar la relación de cada una de las variables:

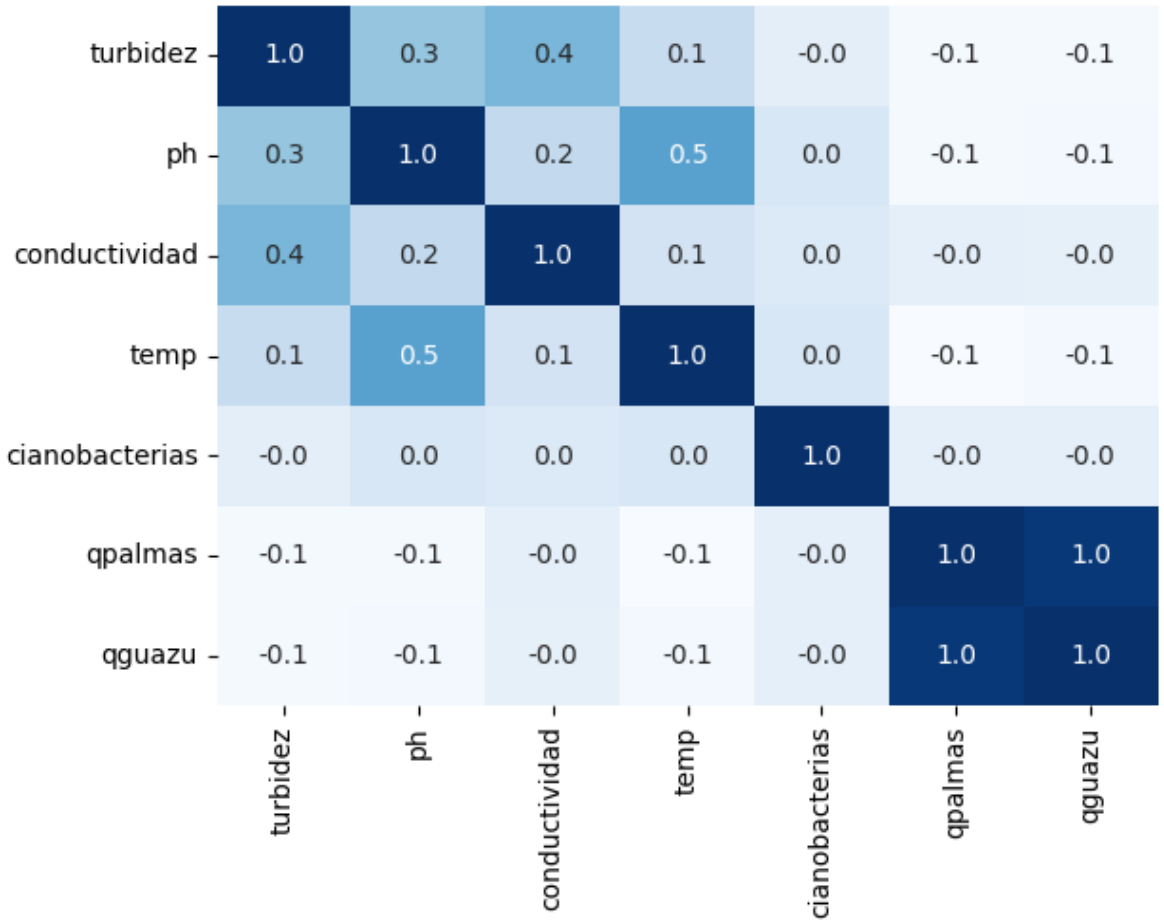


Figura 2. Mapa de calor del dataset.
Fuente: Elaboración propia utilizando la librería Seaborn de Python.

En la presente figura podemos observar a través de un mapa de calor la relación de cada una de las variables con el resto de las características que componen el *dataset*. En el gráfico un valor más cercano a 1 indica una relación directamente proporcional (cuando *x* sube, también lo hace *y*) y -1 nos indica una relación indirectamente proporcional (cuando *x* sube, *y* baja); un valor cercano a cero nos indica una baja correlación entre ambas variables.

La información que podemos obtener de este gráfico es que, más allá de los valores de caudales de los ríos Palmas y Guazú, la relación entre cada una de las características de la muestra es baja o no lineal. Solo observamos una leve relación directamente proporcional entre algunas de ellas.

Lo que este análisis nos permite es identificar qué datos son redundantes entre sí. En este caso en concreto se definió, para el caso de ambos caudales de ríos, combinar los valores ya que se veían afectados por una relación muy fuerte (valor 1.0 en matriz de confusión). Combinar estos dos datos no afectan la información presente en las muestras desde un punto de vista matemático pero permite agilizar el proceso de entrenamiento de los modelos.

Otro análisis que se puede realizar es la distribución de cada muestra respecto a su relación con otras variables:

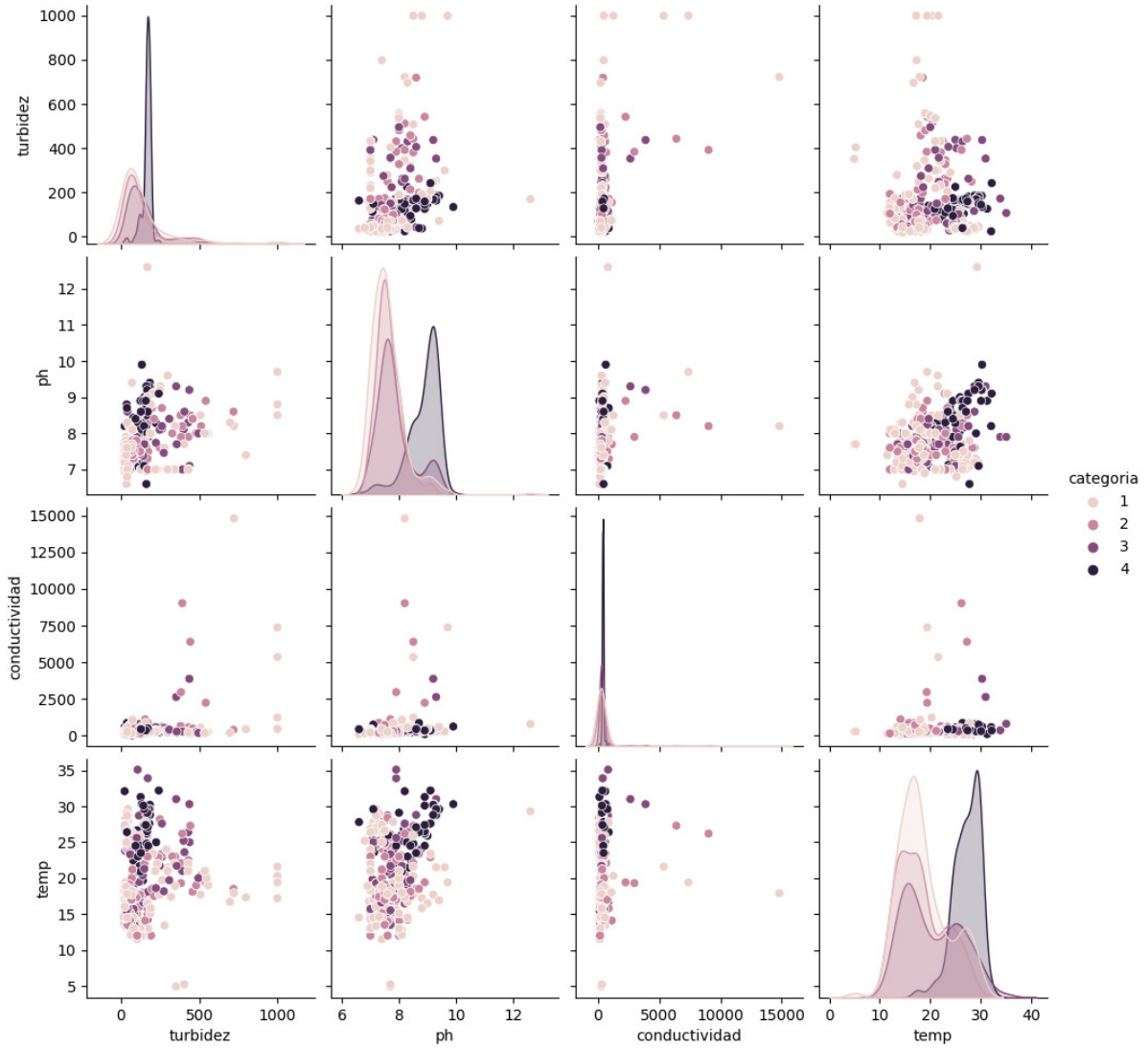


Figura 3. Matriz de gráficos de dispersión.
Fuente: Elaboración propia utilizando la librería Seaborn de Python.

Con la matriz de dispersión se intenta identificar la distribución y relación de cada una de las dimensiones del *dataset*, a fin de encontrar patrones o relaciones entre las distintas variables que componen a la muestra. En este caso en particular, no se detecta un patrón claro o una correlación fuerte entre dos o más variables. Este gráfico en conjunto con el mapa de calor nos indica que las variables que componen las muestras no parecen mostrar correlaciones fuertes más allá de los caudales de ambos ríos. Será entonces tarea de los modelos de IA encontrar los patrones que indiquen presencia o ausencia de cianobacterias en las muestras.

Si sometemos al *dataset* en el que incluímos las características físico-químicas de las muestras, nos encontramos con escenarios similares:

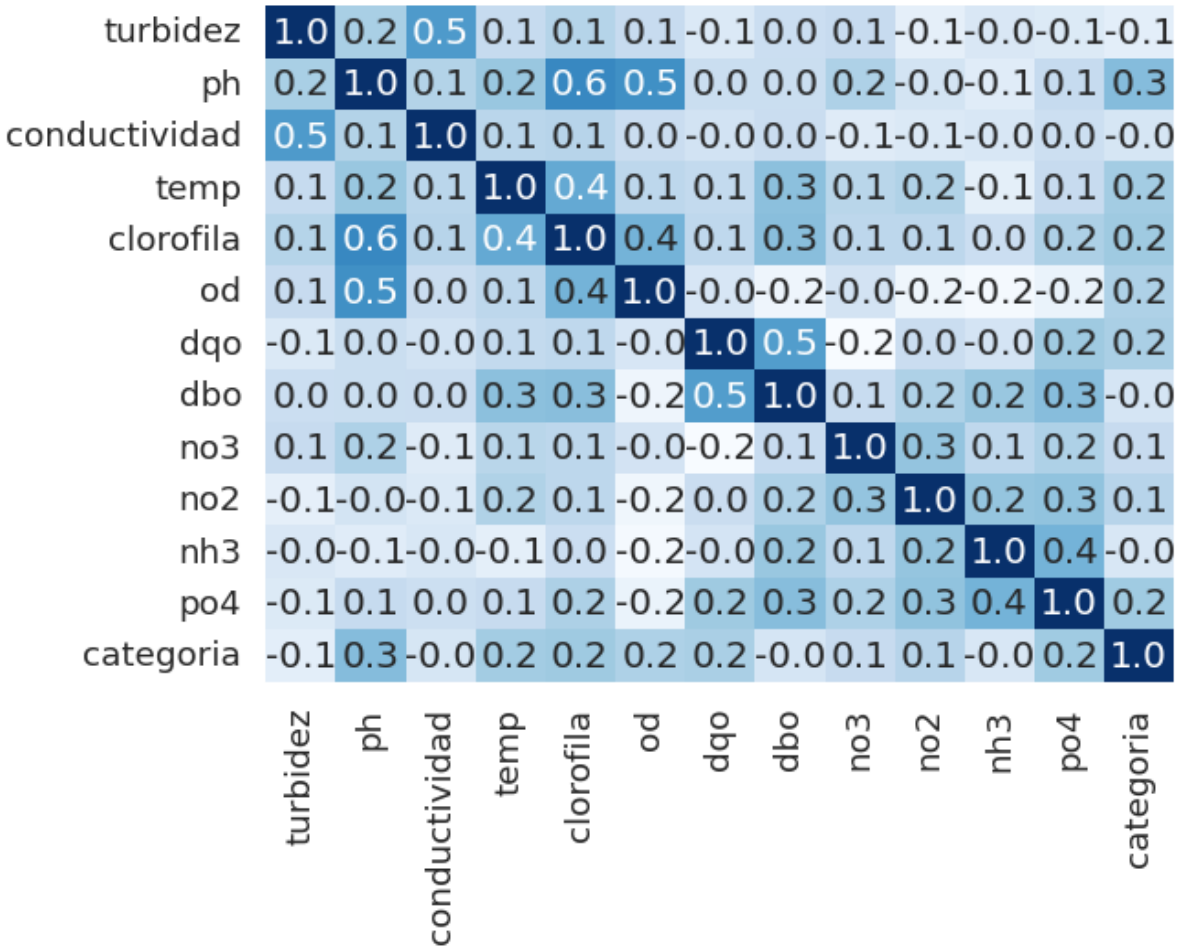


Figura 4. Mapa de calor y relaciones para el *dataset* que incluye valores físico-químicos. Fuente: Elaboración propia utilizando la librería Seaborn de Python.

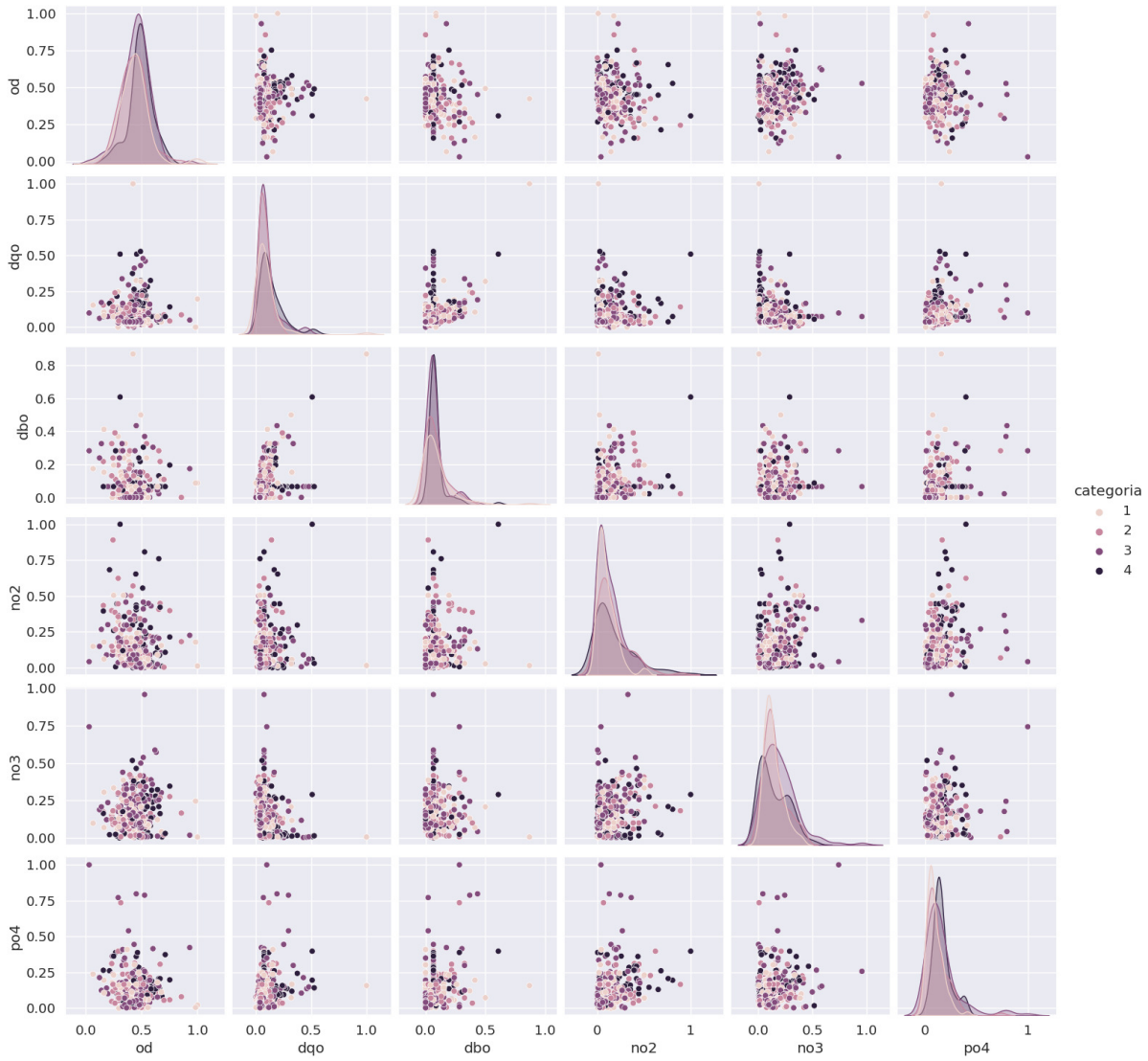


Figura 5. Matriz de dispersión y distribución para el dataset que incluye valores físico-químicos. Fuente: Elaboración propia utilizando la librería Seaborn de Python.

Aunque el mapa de calor muestra una proporción positiva entre dbo y dqo; clorofila y ph; y od y ph, los gráficos no evidencían fuertes relaciones o patrones.

9. Categorización de las muestras

Como se viene mencionando, el objetivo de este trabajo se encuentra focalizado en intentar generar alertas tempranas a la hora de detectar una cantidad determinada de cianobacterias en una muestra de agua, a fin de poder tomar los recaudos y medidas necesarias para proteger a los consumidores de la misma así como también al ecosistema en general. Para lograr este objetivo el primer paso es establecer ciertos umbrales que indicarán si la muestra se encuentra dentro de niveles peligrosos de cianobacterias.

10. Entrenamiento de modelos de Aprendizaje Automático

Para el desarrollo de este trabajo se optó por utilizar tres de los modelos más extendidos en el ámbito de la Inteligencia Artificial, a fin de comparar sus resultados y obtener el modelo que nos arroje los resultados más efectivos. Para cada uno de ellos se hará una breve explicación sobre su funcionamiento y se revisarán los resultados obtenidos para cada uno.

10.1 Redes Neuronales Artificiales (RNA)

Las Redes Neuronales Artificiales son un modelo de Inteligencia Artificial ampliamente utilizado para técnicas de aprendizaje profundo. Están inspiradas en el funcionamiento del cerebro humano y se utilizan para resolver problemas de reconocimiento de patrones, clasificación y predicción principalmente. Están compuestas por nodos que emulan el comportamiento de las neuronas organizadas en una estructura de capas y, al igual que en el cerebro humano, estas neuronas se encuentran comunicadas e intercambian mensajes a fin de ofrecer un resultado a determinada entrada de datos.

Si bien existen, dependiendo del problema que se pretende resolver, diversas arquitecturas en las que pueden ser organizadas los nodos de la RNA, una de las más utilizadas es la denominada MLP (*Multi Layer Perceptron*), en la cual el modelo se organiza con una capa de entrada de datos, *n* capas de ocultas y una capa de salida. La capa de entrada es la encargada de recibir los datos de entrada del modelo y consta de una neurona por cada característica o variable que será utilizada para entrenar el modelo. Las capas ocultas son las capas en las cuales se realizan los cálculos basados en las entradas y sus pesos asociados. Por último la capa de salida es en la que se presenta el resultado de la predicción. El número de nodos de esta capa depende del tipo de problema a resolver, tratándose de uno o más nodos.

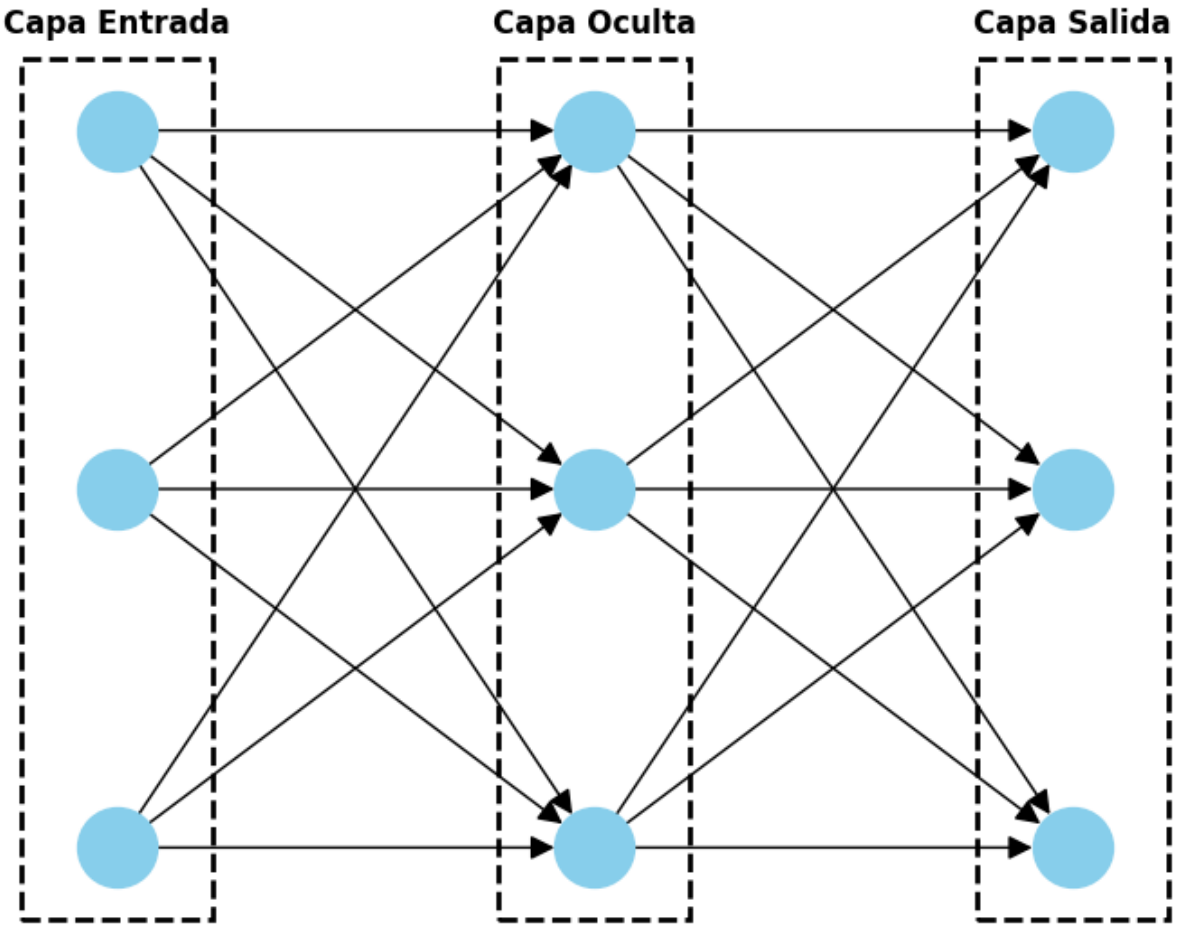


Figura 6. Modelo básico de Red Neuronal Artificial.
Fuente: Elaboración propia.

La capa de entrada del modelo recibe los datos de la fuente externa, cada una de sus neuronas analiza el dato que recibe, tal como las neuronas del cerebro humano, y envía un resultado a cada una de la primera capa oculta. Este proceso se repite a lo largo de cada una de las capas de la red hasta alcanzar la capa de salida, desde la cual se obtiene el resultado. El valor que cada neurona envía a las neuronas de la capa siguiente depende de lo que se conoce como una función de activación. Una función de activación realiza una suma ponderada de cada uno de los valores de entrada de la neurona y utiliza una función matemática no lineal que determina si la neurona debe o no debe activarse. Las funciones de activación más utilizadas son la función sigmoidal, tangente hiperbólica, lineal rectificada, etc.: todas funciones matemáticas

conocidas y ampliamente utilizadas que aportan no linealidad y desvanecimiento del gradiente, entre otras cosas.

Durante el proceso de entrenamiento los esfuerzos del modelo se centran en ajustar los pesos de cada una de las conexiones entre neuronas a fin de encontrar los valores que mejor se ajustan al problema a resolver y que entreguen los resultados más adecuados de acuerdo a los datos que se utilizan para el entrenamiento del mismo. La corrección de los pesos entre neuronas se realiza mediante un proceso denominado *backpropagation*, mediante el cual se utiliza la retropropagación para ajustar los pesos en función del error calculado entre la salida de la red y la salida deseada. Este proceso se repite hasta alcanzar un rendimiento aceptable.

Estos modelos son muy eficientes a la hora de generar predicciones ya que poseen la capacidad de generalizar y de aprender patrones de entrada produciendo valores de salida ante la recepción de estímulos similares.

Las redes neuronales cuentan con una gran capacidad para aprender representaciones complejas lo que las hace muy efectivas para problemas no lineales y pueden ser utilizadas para trabajar con una amplia cantidad de tipos de datos, que van desde texto o datos numéricos o incluso imágenes.

Por otro lado, las redes neuronales se benefician de grandes conjuntos de datos, y no suelen ser la mejor opción para trabajar con una cantidad reducida de muestras, al mismo tiempo que demandan una gran cantidad de poder de cómputo para su entrenamiento, lo que dificulta la generación de modelos en ambientes locales o equipos con características reducidas.

10.2 Árboles de Decisión

Los Árboles de Decisión son modelos de Inteligencia Artificial más simples utilizados principalmente para tareas de categorización y regresión. Su funcionamiento se basa en una serie de nodos que evalúan una característica en concreto de la muestra y, en base al resultado obtenido, toma uno u otro camino hacía otro nodo. Los nodos de decisión se organizan en una

estructura de árbol donde cada uno tiene dos posibles salidas, las cuales pueden ser otro nodo de decisión o un nodo hoja, donde se encuentra el resultado final de la categorización.

En cada uno de los nodos de decisión se evalúa una determinada característica de la muestra que permita determinar qué camino va a continuar la misma. Para definir la prueba que se realiza sobre la característica, se utilizan algunas métricas de impureza o ganancia de información. El objetivo es establecer pruebas que generen divisiones en las muestras y una correcta asignación de las mismas. Por ejemplo, una prueba podría ser evaluar la temperatura de la muestra: si la misma es mayor a 10 C°, entonces se tomará el camino de la izquierda; en caso de que sea menor o igual a 10 C°, se tomará el camino de la derecha. Cada nodo realiza pruebas similares a esta hasta llegar a un nodo hoja.

En los nodos hoja encontramos el resultado de la predicción del modelo. En el caso de un problema de clasificación, el nodo hoja contendrá la etiqueta correspondiente a la muestra evaluada; en caso de un problema de regresión, el nodo hoja contendrá un valor medio o algún otro valor estadístico de los valores del conjunto de datos.

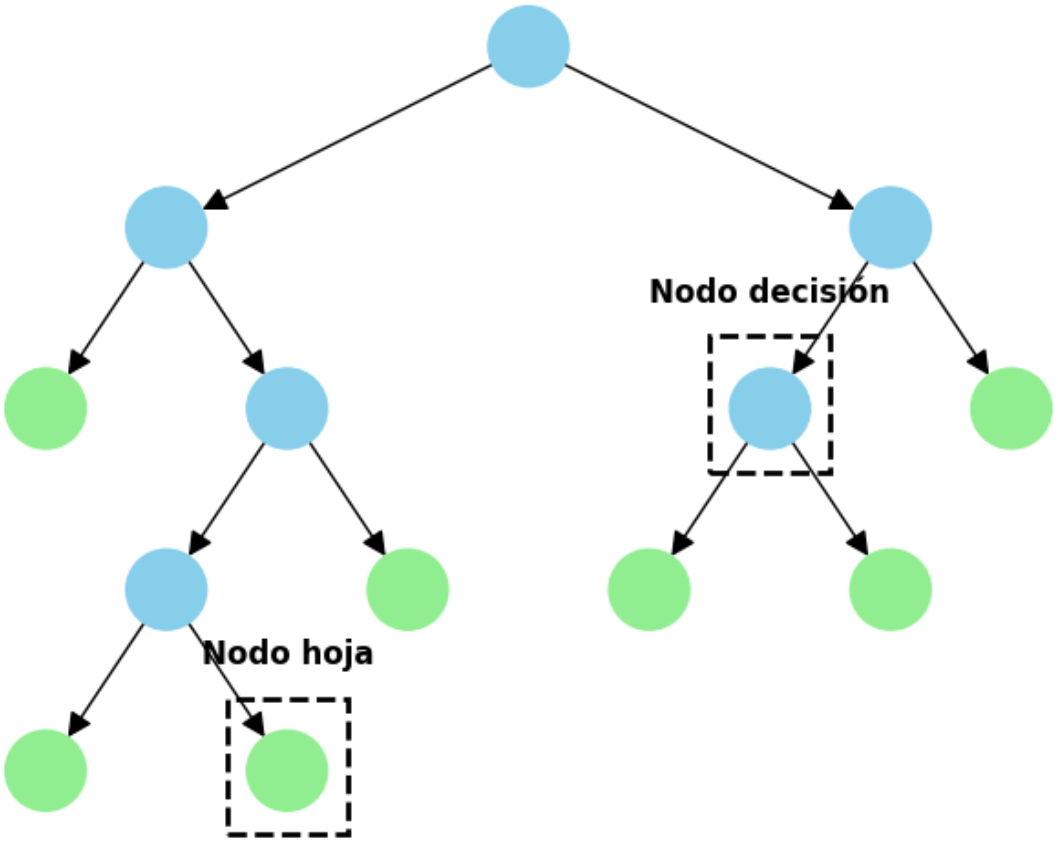


Figura 7. Representación de Árbol de Decisión
Fuente: Elaboración propia.

Este modelo presenta como principales ventajas su simplicidad y posibilidad de visualizar los nodos hoja y decisión lo que facilita la interpretación del modelo y las decisiones que toma. Puede también modelar relaciones no lineales de manera efectiva entre características de entrada y variables de salida. Además este tipo de modelos pueden trabajar con *dataset* que no estén normalizados y puede manejar fácilmente características numéricas y categóricas. Por su simplicidad es mucho más eficiente y ágil a la hora de entrenar el modelo, lo que permite generar múltiples modelos con bajo costo computacional, ofreciendo la posibilidad de compararlos y seleccionar el que obtenga mejores resultados.

10.3 Clasificador K-NN

El modelo de clasificación KNN (*K Nearest Neighbors*) es un algoritmo de Inteligencia Artificial utilizado para clasificación principalmente. Es un algoritmo simple pero efectivo que nos permite categorizar nuevas muestras analizando la similitud de sus características con muestras ya conocidas y clasificadas previamente. La principal idea de este algoritmo es situar las muestras conocidas con sus respectivas categorías en un mapa y, al momento de situar una nueva muestra en ese mismo mapa, verificar a qué categoría pertenecen los K vecinos más cercanos para asignarle la categoría correspondiente a la nueva muestra.

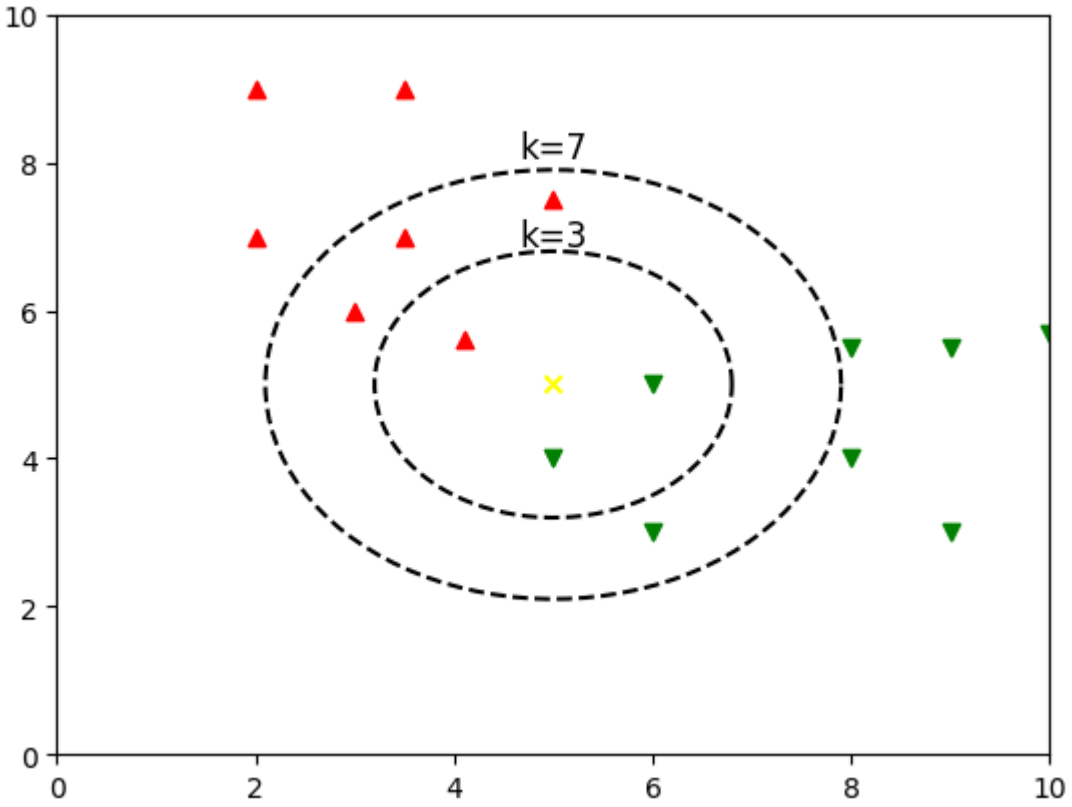


Figura 8. Representación de algoritmo de clasificación KNN.
Fuente: Elaboración propia.

Este algoritmo parte de un conjunto de datos conocido que contiene ejemplos de cada una de las posibles categorías a analizar. Utilizando las características de la muestra se sitúa la misma en el espacio de las muestras (espacio de *n* dimensiones, simplificado en la imagen

para mayor legibilidad). Cuando se sitúa en este mismo espacio una nueva muestra, se calcula la distancia con sus vecinos, ya sea de manera lineal o utilizando alguna función de ponderación por la distancia, para asignar la etiqueta correspondiente a la nueva muestra. La opción más común para medir la distancia entre vecinos es utilizar la distancia euclidiana, pero podría utilizarse cualquier otra función, como la distancia de Manhattan o la distancia de Minkowski. Los vecinos a analizar para asignar la etiqueta están dados por el valor de *k* asignado al momento de generar el modelo.

Si bien este modelo es ampliamente utilizado para clasificación, también puede ser utilizado en problemas de regresión. En ese caso, el valor asignado a la nueva muestra es calculado a través del promedio o algún valor estadístico de los *k* vecinos más cercanos.

Como principal ventaja de este modelo tenemos el muy bajo costo de entrenamiento, ya que lo único que se analiza es la posición de las muestras sobre un mapa de *n* dimensiones. Por otro lado, el costo de generar una nueva predicción suele aumentar en estos modelos ya que deben ser evaluados los puntos de entrenamiento y sus distancias para cada nueva muestra.

En resumen, KNN clasifica nuevos puntos de datos basándose en la mayoría de votos de sus vecinos más cercanos en el espacio de características. Es un algoritmo simple, pero su rendimiento puede depender de la elección adecuada de parámetros y la naturaleza de los datos.

11. Aplicación de los modelos

11.1 Análisis mediante modelos de regresión utilizando RNA

En primera instancia se optó por utilizar modelos de regresión para intentar realizar predicciones sobre el conjunto de datos. Se utilizaron distintas configuraciones de Redes Neuronales Artificiales para intentar predecir el número total de cianobacterias presente en una muestra dentro de un rango muy amplio de posibles resultados.

Si bien se comprobaron varias configuraciones, modificando la cantidad de capas ocultas, las funciones de activación, o incluso intentando utilizar distintas proporciones de las muestras para entrenamiento y testeo, el resultado no fue para nada positivo. Los porcentajes de éxito de los modelos se encontraban muy por debajo de los valores esperados, por lo que se optó descartar este primer enfoque. En su lugar se optó por continuar el trabajo utilizando modelos de clasificación, con el fin de reducir los posibles resultados que los modelos deben predecir.

11.2 Clasificación de las muestras

Para poder utilizar modelos de clasificación fue necesario, en primer lugar, generar las clases o categorías para nuestras muestras. Para ello se generaron 4 distintas categorías utilizando como parámetro la cantidad numérica de cianobacterias presentes en la muestra de acuerdo al siguiente cuadro:

Categoría	Cianobacterias (células/mL)
1	0
2	< 40
3	< 3000
4	>= 3000

Para generar las categorías mencionadas, se tuvo en cuenta la posibilidad de contar con la mayor cantidad de muestras para cada una de ellas, ya que en caso contrario se pueden sobreajustar los modelos sobre alguna categoría en específico y dejar sin cobertura otras.

Cada una de las categorías seleccionadas responde a un nivel de alerta distinto para la muestra en evaluación. Cuando se trate de una muestra de categoría cero, se dice que es segura para el consumo humano ya que no hay presencia de cianobacterias. Categoría 1 responde a un nivel de concentración bajo, que puede requerir ciertos recaudos para evitar afectar al consumo humano. Las Categoría 3 y Categoría 4 responden a niveles de cianobacterias elevados y muy elevados, que pueden requerir de la toma de acciones de mitigación para evitar los efectos adversos por la contaminación del agua. Es importante destacar que la Categoría 3 puede además de significar una alerta de prevención, darnos aviso de que la cantidad de cianobacterias puede incluso ir en aumento. Es importante en este sentido que una muestra que caiga dentro de esta categoría signifique la toma de medidas paliativas.

La siguiente figura muestra los diagramas de dispersión entre algunos de los parámetros considerados (pH, temperatura, turbidez y conductividad). Además, para cada gráfico pueden observarse las cuatro diferentes categorías utilizadas para cuantificar la concentración de cianobacterias. Por lo que se puede observar en los distintos gráficos, es difícil determinar una fuerte relación entre dos o más parámetros para el *dataset* utilizado.

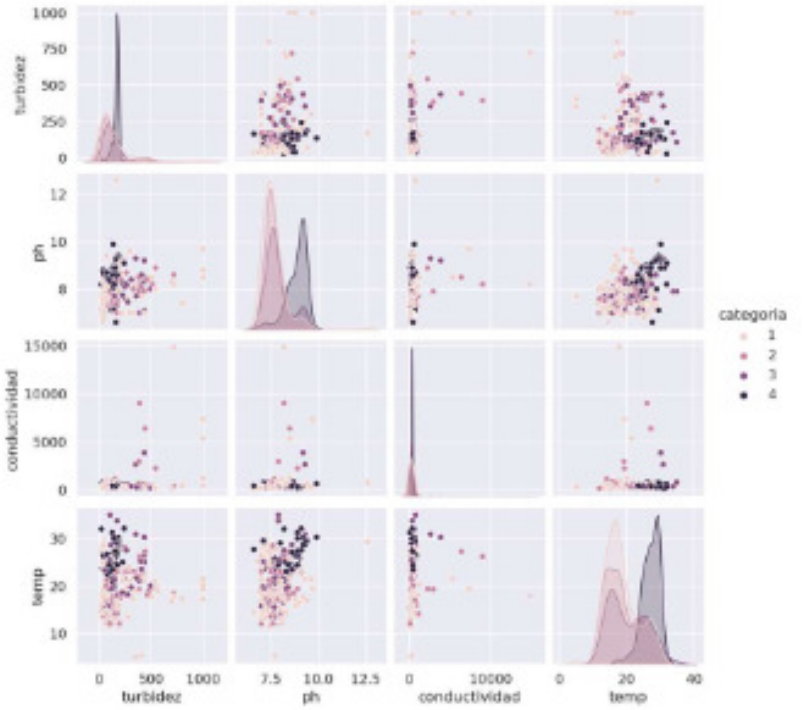


Figura 9. Diagrama de dispersión del dataset.
Fuente: Elaboración propia utilizando la librería Seaborn de Python..

11.3 Modelo de Clasificación mediante Redes Neuronales Artificiales

Con las muestras categorizadas se avanzó hacia un modelo de RNA de clasificación. La estructura de la red neuronal fue determinada exclusivamente por el *dataset* que utilizamos para el entrenamiento así como también las categorías que esperamos predecir. En este escenario se configuró una red neuronal con 5 nodos de entrada, uno para cada una de las características de la muestra que serán evaluadas por el modelo: pH, temperatura, conductividad, caudales y turbidez.

Por otro lado, en las redes neuronales de clasificación la capa de salida está determinada por la cantidad de categorías que componen el universo de posibles clases a asignar. Cada uno de los nodos o neuronas de salida se va a identificar con cada una de las clases y, al final de la predicción, arrojará la probabilidad de que la muestra corresponda a la clase. Para determinar finalmente cuál de las clases corresponde a la muestra, se toma la neurona con el porcentaje

de probabilidad más alto. En este escenario contamos con 4 posibles categorías, por lo que el modelo tendrá 4 nodos o neuronas en la capa de salida.

Finalmente, los modelos de clasificación basados en redes neuronales artificiales permiten tener una cantidad *n* de capas intermedias, cada una de ellas sin restricciones a la cantidad de neuronas. Seleccionar la cantidad de neuronas y capas ocultas para nuestro modelo es un proceso experimental en el que por medio de la prueba y el error se puede encontrar el resultado más óptimo. Sin embargo hay ciertas generalidades que se deben tener en cuenta. A menudo la complejidad del problema es proporcional a la complejidad de la red. Redes simples suelen ser capaces de resolver problemas simples. Al mismo tiempo, un factor importante a tener en cuenta es la capacidad de cómputo de la que se disponga. En nuestro caso en particular el procesamiento se hizo sobre la nube de Google Colab que impone ciertos límites. Modelos con una cantidad mayor de capas y neuronas (y por ende conexiones) son más costosos de entrenar.

También es importante en el proceso de armado del modelo comparar distintas funciones de activación que van a afectar directamente a cómo nuestras neuronas se comunican y por ende generar distintos resultados.

Entre los modelos analizados se optó por el expuesto en la siguiente figura, el cual cuenta con 10 y 8 nodos para la primera y segunda capa intermedia, respectivamente. En esta figura, los cinco nodos de entrada se corresponden con pH (ph), turbidez (turb), conductividad (cond), caudal del Río Paraná y Guazú (q) y temperatura (temp). Por último, un nodo de salida para cada una de las categorías posibles.

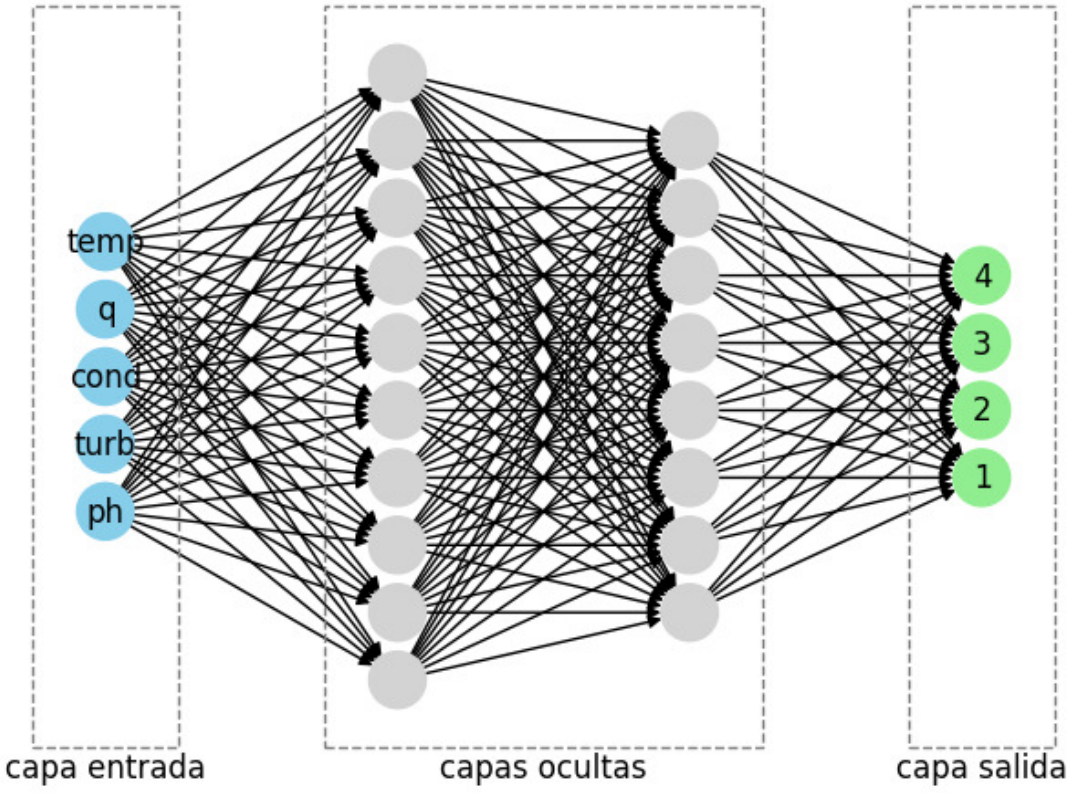


Figura 10. Representación de una Red Neuronal Artificial.
Fuente: Elaboración propia..

La función de activación utilizada fue "relu" para la capa de entrada e intermedia, y "softmax" para la capa de salida. Además, para el análisis, se utilizaron distintas proporciones para el set de datos de entrenamiento y de testeo. Considerando que el *dataset* cuenta con un total de 775 datos, el modelo óptimo fue obtenido utilizando 232 datos para testear los modelos (30% del total), y el resto para propósito de entrenamiento (70% de los datos totales). Los resultados obtenidos serán presentados en la sección Resultados y discusión.

El código del modelo descrito es el siguiente:

```
# División de los datos en test y train
X_train, X_test, y_train, y_test = train_test_split(
     muestras.drop('categoria', axis=1), # variables
independientes
```

```
      muestras['categoria'], # variable dependiente
      test_size=0.3, # porcentaje de la muestra para test
      random_state=42) # semilla para mantener siempre el mismo
resultado para la misma configuración de red.

# Usamos categorías [0, 4)
y_train = y_train - 1
y_test = y_test - 1

scaler = StandardScaler()
# ajustar el objeto de escala a los datos de entrenamiento
scaler.fit(X_train)

shape = len(X_train.keys())

# Escalado de lo dataset
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

# Construcción del modelo
model = tf.keras.models.Sequential([
      tf.keras.layers.Dense(10, activation='relu',
input_shape=[shape]),
      tf.keras.layers.Dense(8, activation='relu'),
      tf.keras.layers.Dense(4, activation='softmax')
])

# Compilación del modelo
model.compile(
      optimizer='adam',
      loss=tf.losses.SparseCategoricalCrossentropy(),
      metrics=['accuracy'])

# Entrenamiento del modelo
history = model.fit(X_train, y_train, epochs=75,
validation_split=0.15, verbose=0)

# Generación de predicciones
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
```

11.4 Análisis mediante modelos de clasificación utilizando Árboles de Decisión

Si bien las RNA pueden realizar el trabajo de predecir la categoría a las que corresponde una determinada muestra de agua, los algoritmos de Árboles de Decisión suelen ser más efectivos y eficaces en este tipo de tareas. Para la generación de estos modelos se utilizaron las librerías que provee la biblioteca de *Sklearn* para Python.

Al igual que como se realizó con el modelo de Redes Neuronales Artificiales, este algoritmo requiere dividir también el *dataset* en datos de entrenamiento y datos de testeo, con la finalidad de entrenar y validar los modelos bajo estudio. En nuestro caso en particular los porcentajes utilizados para cada uno de estos fines fueron 80% y 20% respectivamente.

La función de *Sklearn* destinada a la creación de modelos de Árboles de Decisión es *DecisionTreeClassifier*. Esta función nos permite fácilmente generar modelos de manera automática, tan solo indicando una serie de parámetros que serán utilizados para la generación del modelo, su arquitectura y comportamiento. Entre estos parámetros podemos destacar:

- El criterio que se utilizará para validar la calidad de los nodos, que puede ser '*gini*' o '*entropy*'.
- La posibilidad de indicar la altura máxima que puede tener el árbol.
- La cantidad mínima de muestras que debe acaparar un nodo para determinar si será un nodo hoja o un nodo de decisión.
- La cantidad máxima de hojas

Una técnica ampliamente utilizada para maximizar la efectividad de este tipo de modelos es la utilización de la función GridSearchCV. Esta función realiza un proceso conocido como validación cruzada en el cual se evalúan modelos de Aprendizaje Automático mediante el

entrenamiento simultáneo de varios de ellos y luego, comparando su efectividad. Dado que el entrenamiento de los Árboles de Decisión no requiere un elevado poder de cómputo gracias a su simplicidad, es relativamente sencillo y poco costoso realizar este tipo de comprobaciones a fin de encontrar el modelo óptimo para cada caso comparando distintos parámetros de generación. De esta manera se pudo realizar el entrenamiento y comparación entre varios modelos distintos que diferían en los siguientes parámetros:

- criterio de validación de nodos (criterion): 'gini' o 'entropy'
- profundidad máxima (max_depth): 2, 4, 6, 8 y 10
- Máximo de variables a considerar (max_features): sqrt, log2, 0.2, 0.4, 0.6 y 0.8
- Estrategia para división de nodos (splitter): best y random.

Las posibles combinaciones de todos esos parámetros nos arroja un total de 120 modelos que fueron generados y comparados.

Los modelos no solo fueron entrenados utilizando distinta parametrización, sino que también se utilizaron *batches* de datos de pruebas distintas para cada uno. La parametrización óptima obtenida como resultado fue:

- criterion: "gini";
- max_depth: 6;
- max_features: 0.4;
- splitter: best.

En este caso, el modelo óptimo fue obtenido utilizando 155 datos para testear los modelos (20% del total), y el resto para propósito de entrenamiento (80% de los datos totales). Los resultados obtenidos pueden observarse en la sección Resultados y discusión.

El código para este modelo es el siguiente:

```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler

# Dividimos el dataset en datos de entrada y datos de salida
X = muestras.drop('categoria', axis=1)
y = muestras['categoria']

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# ajustar el objeto de escala a los datos de entrenamiento
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

# Crear el modelo de árbol de decisión
model = DecisionTreeClassifier(
      criterion='gini',
      splitter='best',
      max_depth=None,
      min_samples_split=2,
      min_samples_leaf=1,
      min_weight_fraction_leaf=0.0,
      max_features=None,
      random_state=None,
      max_leaf_nodes=None,
      min_impurity_decrease=0.0,
      class_weight=None,
      ccp_alpha=0.0
)

# Entrenar el modelo
model.fit(X_train, y_train)

# Realizar predicciones en el conjunto de prueba
y_pred = model.predict(X_test)
```

```
# Validación cruzada para optimización del modelo
params = {
      'criterion':  ['gini', 'entropy'],
      'max_depth':  [2, 4, 6, 8, 10],
      'max_features': ['sqrt', 'log2', 0.2, 0.4, 0.6, 0.8],
      'splitter': ['best', 'random']
}

clf = GridSearchCV(
      estimator=DecisionTreeClassifier(),
      param_grid=params,
      cv=3,
      n_jobs=5,
      verbose=1,
)

# Proceso de validación
clf.fit(X_train, y_train)
print(clf.best_params_)

# Entrenar el modelo
clf.best_estimator_.fit(X_train, y_train)

# Realizar predicciones en el conjunto de prueba utilizando el
mejor modelo
y_pred = clf.best_estimator_.predict(X_test)
```

11.5 Análisis mediante modelos de clasificación utilizando K-NN

Finalmente, se llevó a cabo el análisis a partir de modelos de clasificación KNN. Una característica importante de este modelo de Aprendizaje Automático es que suele tener mejor desempeño que otros modelos en casos donde se cuenta con *dataset* de tamaño reducido. En nuestro caso en particular contamos con un *dataset* de alrededor de 800 muestras, lo que hace a este modelo un buen candidato para resolver el problema.

Al igual que el caso de Árboles de Decisión, la generación de estos modelos se realizó utilizando la biblioteca de Sklearn. La función utilizada para la generación de estos modelos fue *KNeighborsClassifier*, la cual nos permite definir una serie de hiperparámetros para definir algunas características importantes que hacen al funcionamiento del modelo. Entre esos parámetros se encuentran: la cantidad de vecinos a tener en cuenta para la categorización de la muestra, el peso que se le da a cada vecino (puede ser uniforme, ponderado por la distancia o una función propia), la función para encontrar los vecinos (provee la opción de "auto" para dejarlo a cargo del algoritmo), la métrica para medir la distancia ("euclidean", "manhatan", etc), el número de muestras en una hoja de un árbol, etc.

Otra característica importante de este modelo, compartida con el modelo de Árboles de Decisión, es su poca exigencia computacional, lo que nos permite hacer uso de la misma función GridSearchCV utilizada para la validación cruzada.

Para este modelo se compararon 112 modelos distintos con los siguientes hiperparámetros:

- cantidad de vecinos (n_neighbors): 1, 3, 5, 7, 9, 11, 13;
- peso de cada vecino (weights): uniform, distance;
- función para encontrar los vecinos más cercanos (algorithm): auto, ball_tree, kd_tree, brute;
- métrica de distancia utilizada para la búsqueda de vecinos más cercanos (p): 1 (Manhattan), 2 (Euclidiana);
- número de muestras en una hoja de un árbol (leaf_size): 5, 10, 15, 20, 25, 30.

La herramienta GridSearchCV arrojó que la parametrización más efectiva es:

- n_neighbors: 7;
- weights: uniform;
- algorithm: ball_tree;
- p: 1 (Manhattan);

- leaf_size: 20.

Nuevamente en este caso, el modelo óptimo fue obtenido utilizando 155 datos para testear los modelos (20% del total), y el resto para propósito de entrenamiento. Los resultados obtenidos pueden verse en la sección “Resultados y discusión”.

El código para la generación de este modelo es el siguiente:

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix

# Escalado de valores
columns_to_scale = ['ph', 'conductividad', 'temp', 'qpalmas']
scaler = MinMaxScaler()
muestras[columns_to_scale] =
scaler.fit_transform(muestras[columns_to_scale])

# División del dataset en prueba y test
X = muestras.drop('categoria', axis=1)
y = muestras['categoria']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Crear el modelo de KNN
clf = KNeighborsClassifier(
      n_neighbors=5,        # The number of neighbours to consider
      weights='uniform',    # How to weight distances
      algorithm='auto',     # Algorithm to compute the neighbours
      leaf_size=30,         # The leaf size to speed up searches
      p=2,                  # The power parameter for the Minkowski
metric
      metric='minkowski',   # The type of distance to use
      metric_params=None,   # Keyword arguments for the metric
function
      n_jobs=None           # How many parallel jobs to run
```

```
)

# Entrenar el modelo
clf.fit(X_train, y_train)

# Realizar predicciones en el conjunto de prueba
y_pred = clf.predict(X_test)

# Evaluar el modelo utilizando el error cuadrático medio (MSE)
accuracy = accuracy_score(y_test, y_pred)
print('Exactitud:', str(accuracy*100)[:5]+'%')

# Realizamos la validación cruzada
params = {
      'n_neighbors':  range(1, 15, 2),
      'p': [1,2],
      'weights': ['uniform', 'distance'],
      'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
      'leaf_size': range(5, 31, 5),
      'metric_params': [None],
      'n_jobs': [None]
}

clf = GridSearchCV(
      estimator=KNeighborsClassifier(),
      param_grid=params,
      cv=5,
      n_jobs=5,
      verbose=1,
)

clf.fit(X_train, y_train)

# Crear el modelo de KNN con los nuevos parámetros
clf.best_estimator_.fit(X_train, y_train)

# Realizar predicciones en el conjunto de prueba
y_pred = clf.predict(X_test)

# Evaluar el modelo utilizando el error cuadrático medio (MSE)
accuracy = accuracy_score(y_test, y_pred)
```

```
print('Exactitud:', str(accuracy*100)[:5]+'%')
```

12. Análisis del *dataset* con datos físico-químicos

Si bien el *dataset* que contiene datos físico-químicos tiene una cantidad de datos reducido, realizamos un procesamiento aplicando el modelo de Árboles de Decisión, ya que demostró tener una buena capacidad para resolver el problema con el *dataset* principal, al mismo tiempo que su demanda de poder de cómputo es muy reducida y su simplicidad lo hace más eficiente a la hora de generar los modelos.

La matriz de confusión resultante del procesamiento fue:

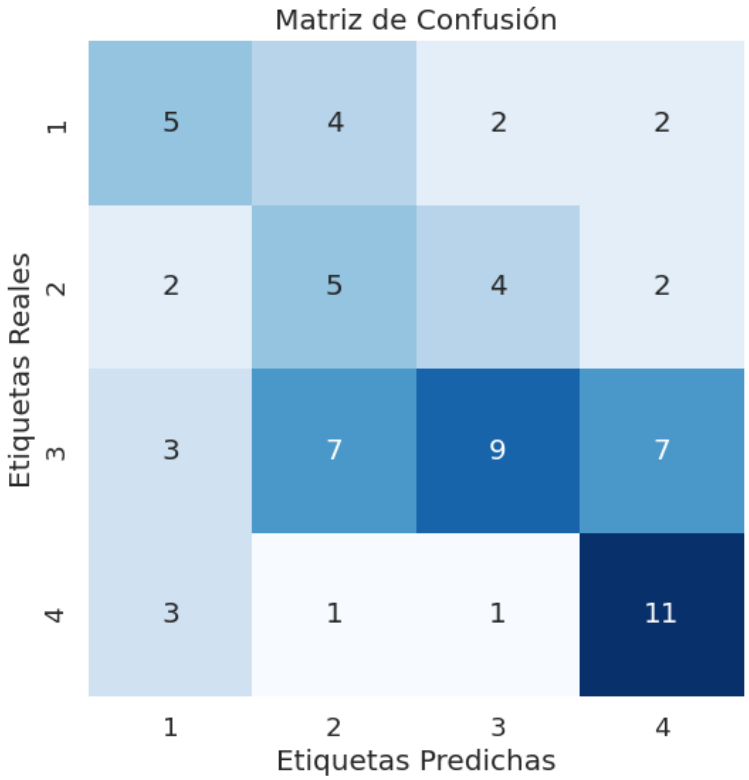


Figura 11. Matriz de confusión resultante.
Fuente: Elaboración propia utilizando la librería Seaborn de Python..

Si bien el modelo es capaz de acertar la mayoría de los casos correspondientes a la categoría número 4, el porcentaje de efectividad del modelo en general no alcanza el 50%. Los datos no son suficientes para entrenar un modelo que sea capaz de arrojar resultados fiables a la hora de predecir la concentración de cianobacterias en las muestras.

13. Resultados y discusión

Una vez finalizada la etapa de entrenamiento de cada uno de los modelos, se llevó a cabo el testeo de los mismos. En las siguientes figuras se presentan los resultados obtenidos mediante una matriz de confusión en la cual se puede verificar la precisión de cada uno de los modelos para acertar a la etiqueta correcta en cada una de las muestras:

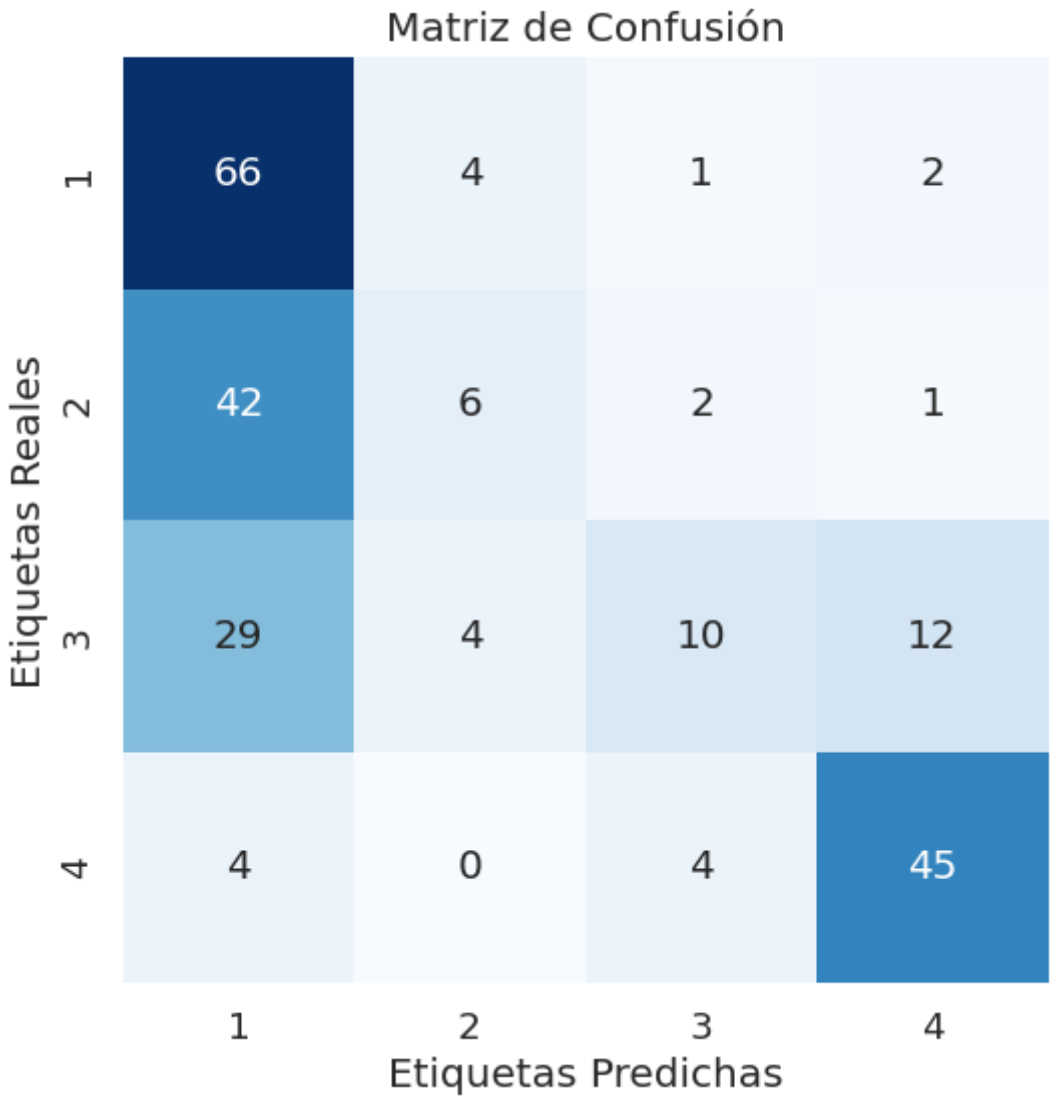


Figura 12. Matriz de confusión para resultados del procesamiento utilizando RNA. Fuente: Elaboración propia utilizando la librería Seaborn de Python..

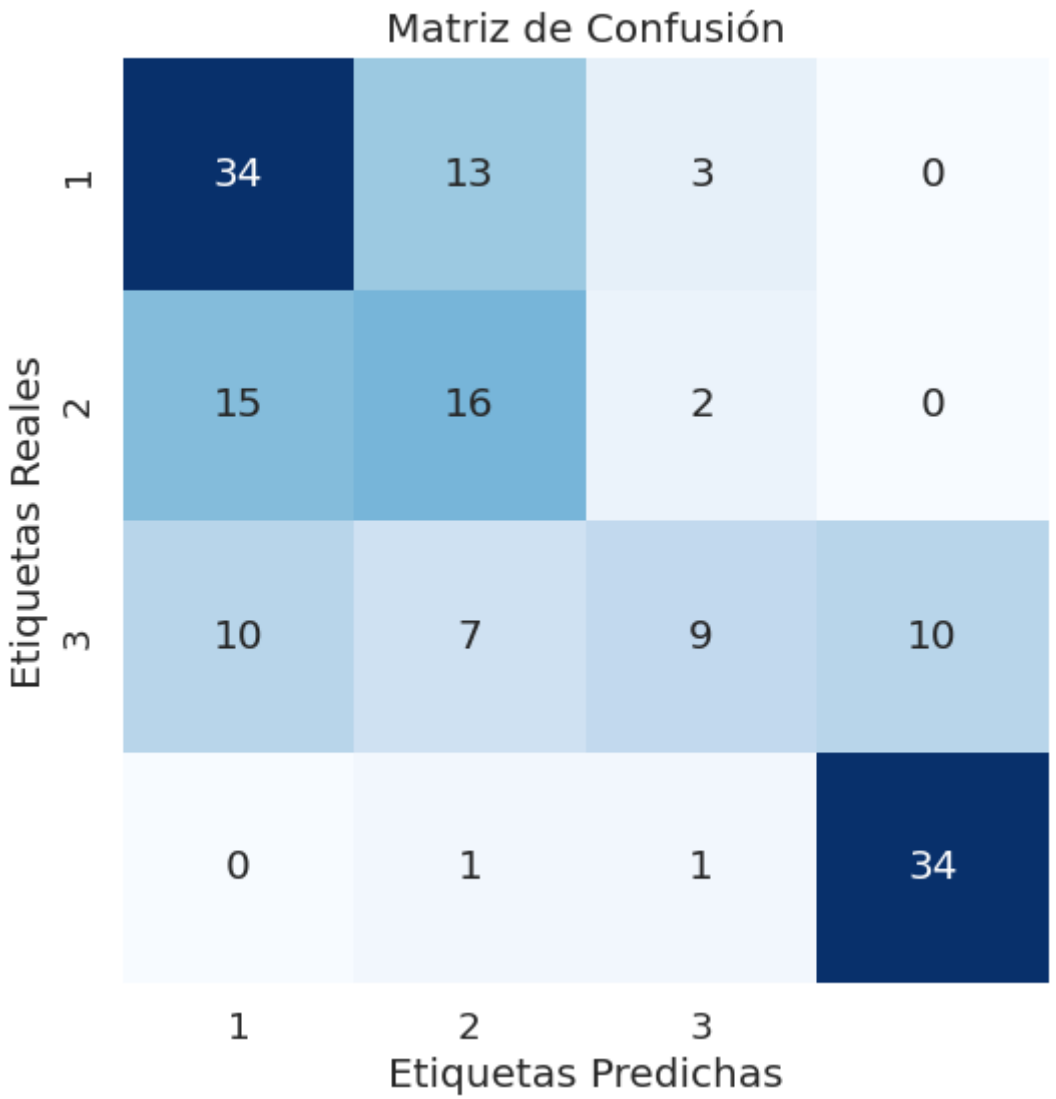


Figura 13. Matriz de confusión para resultados del procesamiento utilizando Árboles de Decisión. Fuente: Elaboración propia utilizando la librería Seaborn de Python..

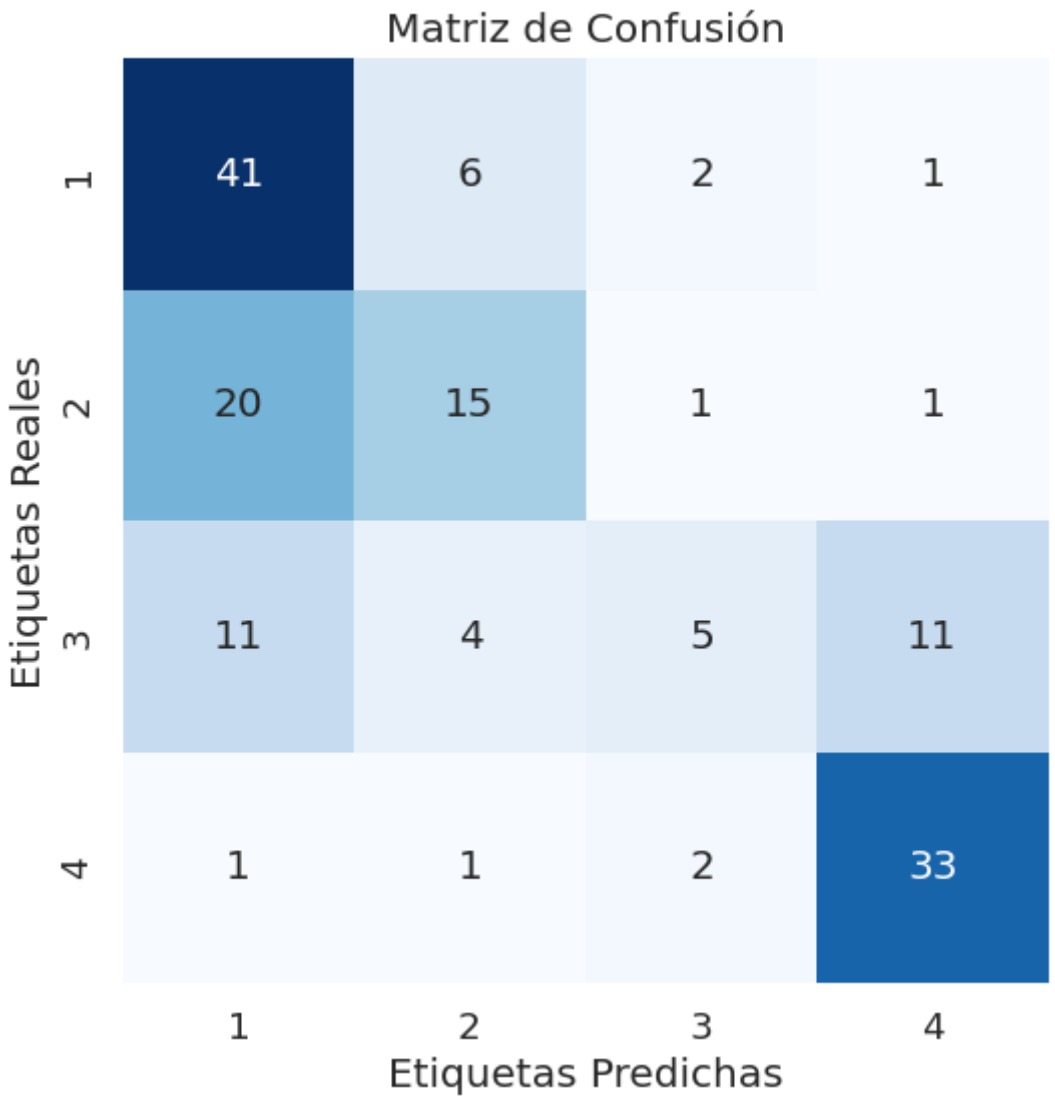


Figura 14. Matriz de confusión para resultados del procesamiento utilizando el algoritmo KNN. Fuente: Elaboración propia utilizando la librería Seaborn de Python..

Las categorías 1, 2, 3 y 4 corresponden a las cuatro diferentes condiciones de concentración de cianobacterias presentes en una muestra de agua, tal como se mencionó anteriormente en el proceso de categorización de las mismas. La matriz de confusión es una herramienta muy útil para valorar qué tan bueno es un modelo de clasificación basado en Aprendizaje Automático. Esta matriz permite evaluar y comparar la etiqueta asignada por el modelo con la etiqueta real que corresponde a la muestra de prueba evaluada.

Es importante destacar que si bien la diagonal principal de la matriz de confusión para cada uno de los modelos no está muy definida y se pueden observar resultados no tan favorables

en las primeras categorías, los modelos obtuvieron buenos resultados a la hora de detectar muestras pertenecientes a la categoría 4 y en menor proporción a la categoría 3. Al mismo tiempo, se puede observar que los cuadrantes superior izquierdo e inferior derecho son los de mayor concentración de casos. Por ejemplo, para la matriz de confusión de árboles de decisión:

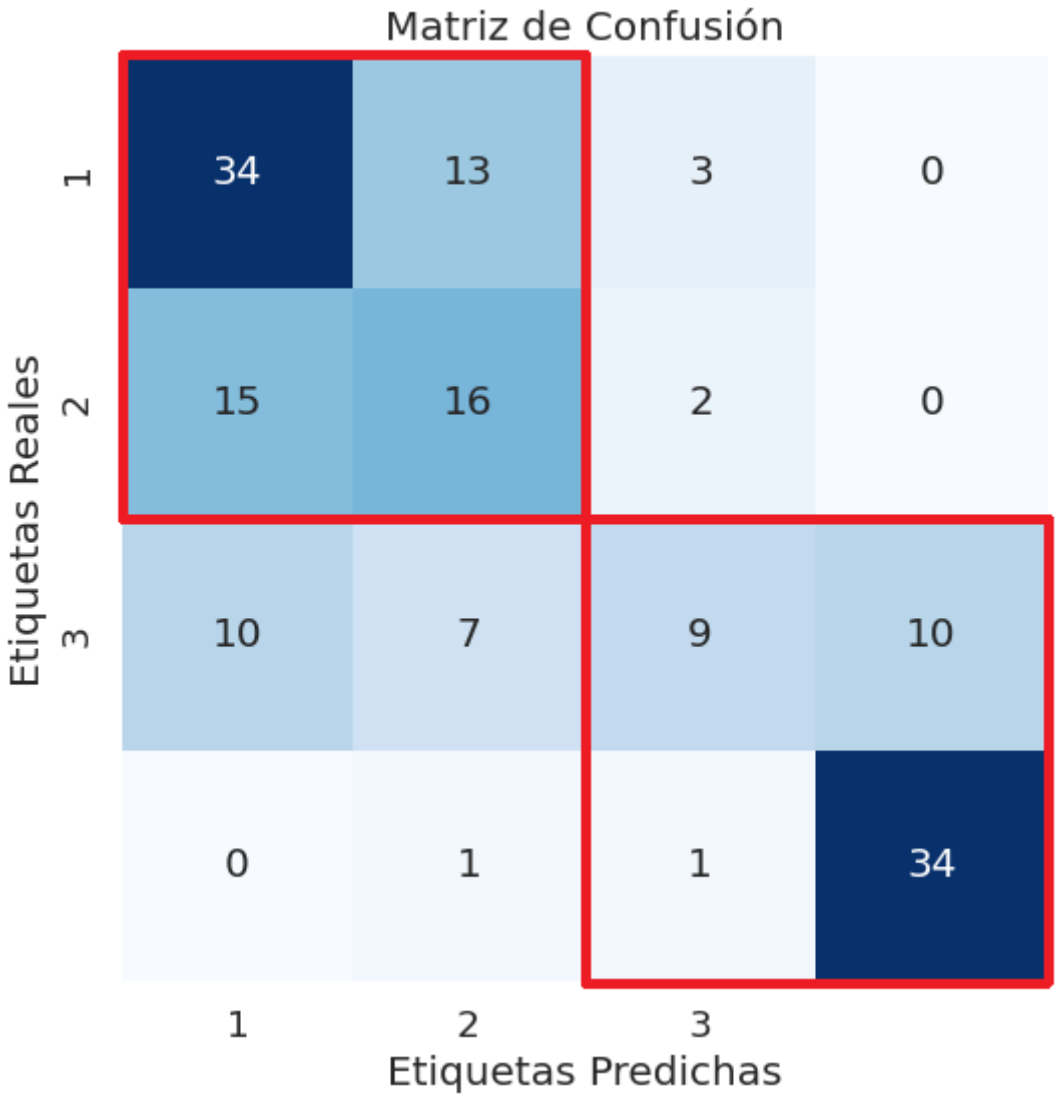


Figura 15. Áreas de interés para la matriz de confusión obtenida.
Fuente: Elaboración propia utilizando la librería Seaborn de Python..

En este caso en particular, observamos que solo el 16% de las muestras fueron clasificadas de manera que quedaron fuera de los cuadros rojos.

Esto nos muestra que si bien los modelos tienen cierta dificultad para diferenciar muestras de categoría 1 con las de categoría 2 y casos de categoría 3 con categoría 4, no son muchos los casos en que los modelos identifican muestras de baja concentración de cianobacterias cuando se trata de muestras con alta concentración o viceversa. En ese sentido, se puede decir que los modelos tienen una exactitud aún mayor en diferenciar muestras de categoría 3 o 4 respecto a muestras de categoría 1 o 2.

Además de la matriz de confusión hay otras medidas comúnmente utilizadas para medir y evaluar el desempeño de modelos de Inteligencia Artificial, tales como exactitud, precisión, *recall* y *f1_score*. Cada una de estas métricas está dada por las siguientes expresiones:

$Exactitud = \frac{(TP+TN)}{(TP+TN+FP+FN)}$
$Precisión = \frac{TP}{(TP+FP)}$
$Recall = \frac{TP}{(TP+FN)}$
$F1Score = \frac{(Precisión*Recall)}{(Precisión+Recall)}$

Donde:

TP	Verdaderos Positivos (True Positive)
FP	Falsos Positivos (False Positive)
TN	Verdaderos Negativos (True Negative)
FN	Falsos Negativos (False Negative)

Cada una de estas expresiones matemáticas nos aporta cierta información acerca del desempeño y capacidad de predicción de los modelos evaluados.

- La precisión mide la proporción de casos positivos que fueron correctamente identificados por el modelo con respecto a todas los casos que el modelo clasificó como positivos. La precisión es útil cuando el costo de los falsos positivos es alto.

- La exactitud evalúa la proporción de todas las predicciones (tanto positivas como negativas) que fueron correctas. La exactitud puede ser engañosa cuando hay un desequilibrio entre las clases, ya que un modelo puede tener una alta exactitud simplemente prediciendo siempre la clase mayoritaria.
- *Recall* mide la proporción de casos positivos que fueron correctamente identificados por el modelo con respecto a todos los casos positivos reales. El *recall* es importante cuando es crucial capturar todas las instancias positivas, incluso a costa de tener algunos falsos positivos.
- El F1-score es la media armónica entre precisión y *recall*. Proporciona un equilibrio entre ambas métricas. Es útil cuando hay un desequilibrio entre las clases y se busca un equilibrio entre la precisión y el *recall.*

En nuestro caso en particular, nuestros modelos están enfocados en categorizar las muestras de agua de acuerdo a su concentración de cianobacterias, mientras que los casos que son de mayor interés para la predicción, son aquellos de categoría 3 y 4, donde la proporción es alta y requiere de mayores recaudos para evitar problemas en el medioambiente y la salud de sus consumidores. Aunque es importante que nuestros modelos tengan una alta precisión, podemos decir que es tolerable una cierta cantidad de falsos positivos para las categorías 1 y 2, donde el riesgo es bajo. Podemos determinar los siguientes escenarios a los que puede enfrentarse nuestro modelo a la hora de detectar altas concentraciones de cianobacterias:

- Verdadero Positivo: ocurre cuando el modelo identifica correctamente la muestra como categoría 4.
- Falso Positivo: ocurre cuando el modelo identifica como categoría 4 una muestra que corresponde a la categoría 1, 2 o 3.
- Falso Negativo: ocurre cuando el modelo asigna una categoría errónea a una muestra que corresponde a la categoría 4.

De los tres casos mencionados, es deseable la mayor cantidad de verdaderos positivos, mientras que los falsos positivos no significan una amenaza crítica ya que son casos en que no habría riesgo real de contaminación y, por otro lado, los falsos negativos son el escenario más peligroso, ya que la amenaza es real y al no ser detectada pueden no realizarse los procesos de mitigación requeridos.

En este escenario, si bien la exactitud es una métrica muy importante, en este trabajo también es de especial interés que el modelo tenga buenos resultados en términos de *recall*, ya que este hace énfasis en categorías extremas (1 y 4). Debido a que la categoría 4 es esencialmente la de mayor interés para nuestro problema y dado que el valor de *recall* aumenta con la disminución del número de FN, esta métrica es considerada importante para este trabajo.

Los valores para cada uno de nuestros modelos son los siguientes:

Métricas correspondientes al modelo de RNA				
Categoría	Exactitud	Precisión	Recall	F1_Score
1	0.65	0.47	0.90	0.62
2	0.77	0.43	0.12	0.19
3	0.78	0.59	0.18	0.28
4	0.90	0.75	0.85	0.80

Métricas correspondientes al modelo de Árboles de Decisión				
Categoría	Exactitud	Precisión	Recall	F1_Score
1	0.76	0.61	0.70	0.65
2	0.79	0.56	0.49	0.52
3	0.81	0.54	0.39	0.45
4	0.92	0.77	0.92	0.84

Métricas correspondientes al modelo KNN

Categoría	Exactitud	Precisión	Recall	F1_Score
1	0.77	0.62	0.78	0.69
2	0.79	0.56	0.49	0.52
3	0.79	0.47	0.29	0.36
4	0.88	0.73	0.81	0.77

Valores de *recall* de 85%, 92% y 81% fueron obtenidos para la categoría 4 mediante la utilización de RNA, Árboles de Decisión y K-NN, respectivamente. Los valores de *recall* superiores a 90% (debido a menores probabilidades de FN), como es el caso de los Árboles de Decisión, proporcionan sistemas de clasificación eficientes para la seguridad sanitaria, debido a su gran capacidad para identificar muestras de agua con una concentración de cianobacterias superior a 3000 células/mL. Finalmente, tomando en cuenta los valores obtenidos para las métricas de exactitud y recall, el modelo generado utilizando Árboles de Decisión es el que mejores resultados ofreció.

14. Conclusiones

A lo largo de este trabajo se modelaron y compararon tres soluciones propuestas basadas en diferentes técnicas de Aprendizaje Automático, supervisado como mecanismo de clasificación, a fin de identificar muestras de agua que, dados sus niveles de concentración de cianobacterias, serían potencialmente tóxicas para el consumo humano y peligrosas para el medio ambiente en general. Los tres modelos comparados fueron entrenados y testeados a partir del mismo set de datos, que contenía datos biológicos y fisicoquímicos de una serie de muestras obtenidas de distintos puntos de recolección del Río de La Plata.

A pesar del tamaño reducido del *dataset,* los tres modelos consiguieron resultados satisfactorios para la predicción de muestras de categoría 4. Valores de exactitud y *recall* de 90% y 85% para RNA, 92% para Árboles de Decisión y 88% y 81% para K-NN fueron obtenidos respectivamente.

En el caso de la exactitud, esto implica que los modelos responden adecuadamente para clasificar como categoría 4 a aquellas muestras de agua que poseen una concentración de cianobacterias superior a 3000 células/mL y como categoría diferente de 4 en caso contrario. Y en el caso de la métrica *recall*, sus altos valores implican que los modelos tienen muy pocos fallos para clasificar una muestra que realmente pertenece a la categoría 4 como si fuese de otra categoría.

Los resultados presentados en este trabajo son de suma utilidad para poder clasificar y detectar una posible floración de cianobacterias de manera temprana que permita tomar medidas efectivas para evitar contaminación en el suministro de agua.

El modelo elegido finalmente por mejor desempeño fue el de Árboles de Decisión, que además de presentar resultados levemente superiores al de los otros modelos utilizados, tiene a su favor la ventaja de ser considerablemente menos demandante en lo que a recursos computacionales se refiere. Esta ventaja extra que aporta este modelo puede ser de utilidad al

momento de utilizar y seguir entrenando el mismo en unidades de procesamiento de bajo costo.

Por otra parte, el principal inconveniente encontrado a la hora de desarrollar los modelos de predicción expuestos en este trabajo se encuentra en los datos utilizados para el entrenamiento de los mismos. Los datos son fundamentales en el entrenamiento de modelos de Inteligencia Artificial y desempeñan un papel crucial en la calidad y el rendimiento del modelo resultante. Los modelos de Aprendizaje Automático, y en especial los modelos supervisados, demandan una gran cantidad de datos de calidad; cuanto mayor sea el *dataset* utilizado mejores resultados se obtendrán, de la misma manera que la calidad de los mismos afecta al rendimiento al momento de exponer el modelo al mundo real. Algunas de las características más importantes que nos indican que un *dataset* es de calidad son las siguientes:

- **Completitud**: que haya la menor cantidad de datos faltantes posible
- **Exactitud**: datos precisos y libres de errores
- **Consistencia**: se refiere a la uniformidad de los datos a lo largo de diferentes conjuntos y variables. Los datos deben seguir las mismas reglas y estándares en todo el conjunto para garantizar su coherencia.
- **Relevancia y Representatividad**: las características seleccionadas de las muestras deben ser relevantes para la tarea que se está abordando, así como ser representativas del dominio que se trata de modelar. Un conjunto de datos no representativo puede llevar a un modelo que no generaliza bien a nuevas instancias.
- **Diversidad**: Un conjunto de datos diverso ayuda al modelo a aprender patrones generales en lugar de depender demasiado de casos específicos.

- **Formato Consistente**: Los datos deben tener un formato consistente y estar bien estructurados. Esto facilita el procesamiento y la preparación de datos antes del entrenamiento del modelo.

En el caso puntual de este trabajo, el *dataset* incumplía varias de estas características importantes para garantizar el correcto funcionamiento y la calidad de los modelos generados. Los mismos contaban con problemas en su completitud, consistencia y formato, además de que hubiera sido deseable una mayor cantidad de muestras. Siendo los datos el recurso más importante a la hora de trabajar con estos modelos de clasificación por Aprendizaje Automático, sería de especial utilidad mejorar la calidad de los mismos.

Una alternativa que merece la pena explorar es la de automatizar el proceso de muestreo. Actualmente el proceso de obtención de muestras se hace de manera manual, lo que impide contar con un gran número de muestras, e induce a una mayor cantidad de pasos en el proceso que pueden fallar y afectar a la exactitud de las muestras. Automatizar el proceso a través de sensores y dispositivos que puedan obtener las muestras de manera periódica podría garantizar un set de datos con mayor consistencia de los mismos, reducir los riesgos de error humano y obtener una mayor cantidad de muestras al tiempo que se reducen los recursos necesarios.

Al mismo tiempo, obtener muestras de manera automatizada abre nuevas posibilidades para los modelos de detección. Con una mayor cantidad de datos se pueden analizar las variaciones diarias o incluso en rango de horas, a fin de predecir una floración de cianobacterias incluso antes de que esta suceda. Contar con una amplia cantidad de muestras incluso en el rango de horas, sería de especial utilidad en la detección de las floraciones de cianobacterias ya que estas pueden desarrollarse en periodos de tiempo muy cortos.

Finalmente, a fin de ampliar las posibilidades que nos otorgan los modelos de categorización, se podrían incluir otras fuentes de datos que están disponibles de manera libre en internet,

como pueden ser las imágenes satelitales. Hoy en día es común utilizar imágenes satelitales para la detección de floraciones de cianobacterias y otras algas, analizando el color del agua.

14. Bibliografía

Adams, S.M., "Using multiple response bioindicators to assess the health of estuarine ecosystems: An operational framework". InEstuarine indicators, ed. S.A. Bortone, Washington: CRC (2005).

Alpaydin, E., "Machine Learning: The New AI". Editorial MIT Press. (2016).

Andrinolo, D., Pereira, P., Giannuzzi, L., Aura, C., Massera, S., Caneo, M., Caixach, J., Barco, M. y Echenique, R., "Occurrence of Microcystis aeruginosa and microcystins in Río de la Plata river (Argentina)". Acta Toxicológica Argentina. Vol. 15 (1). (2007). pp. 8-14.

Arisholm, E, Briand, L, Johannessen, E. "A systematic and comprehensive investigation of methods to build and evaluate fault prediction models". Journal of Systems and Software. Vol. 83(1). (2010). pp. 2–17.

Barbier, E. B., Hacker, S. D., Kennedy, C., Koch, E. W., Stier, A. C., y Silliman, B. R., "The value of estuarine and coastal ecosystem services". Ecological monographs. Vol. 81(2) (2011). pp.169-193.

Camilloni, I., y Bidegain, M., "Escenarios climáticos para el siglo XXI". El cambio climático en el Río de la Plata. (2005) pp. 33-39.

Chorus, I., y Bartram, J., "Toxic Cyanobacteria in Water – A guide to their Public Health Consequences Monitoring and Management. London: E and FN Spon. (1999). pp. 416.

Cloern, J. E., y Dufford, R.. "Phytoplankton community ecology: principles applied in San Francisco Bay". Marine Ecology Progress Series. Vol. 285 (2005). pp. 11-28.

Cloern, J. E., "Our evolving conceptual model of the coastal eutrophication problem". Marine Ecology Progress Series. Vol. 210 (2001). pp. 223-253.

Echenique, R., Guerrero, J. M., Lamarol, A. A., "Floraciones de cianobacterias en el Río de la Plata". (2020). Informe técnico.

Fairbridge, R. W., "The estuary: its definition and geodynamic cycle". Chemistry and Biogeochemistry of Estuaries. Nova Yorque, Estados Unidos: John Wiley and Sons (1980). pp. 1-35.

Framiñan, M. B., Etala, M. P., Acha, E. M., Guerrero, R. A. Lasta, C. A., y Brown, O. B., "Physical characteristics and processes of the Río de la Plata estuary". In Estuaries of South America. Springer Berlin Heidelberg. (1999) pp. 161-194.

FREPLATA, "Análisis Diagnóstico Transfronterizo del Río de la Plata y su Frente Marítimo". Protección Ambiental del Río de la Plata y su Frente Marítimo: Prevención y Control de la Contaminación y Restauración de Hábitats. Documento Técnico. Proyecto PNUD/GEF RLA/99/G31. Montevideo, Uruguay (2005).

Giannuzzi, L., Carvajal, G., Corradini, M. G., Araujo Andrade, C., Echenique, R., y Andrinolo, D., "Occurrence of toxic cyanobacterial blooms in Rio de la Plata", Página151, Estuary, Argentina: field study and data analysis. Journal of Toxicology. (2012). doi:10.1155/2012/373618.

Gómez, N., "Phytoplankton of the Río de la Plata Estuary". In: Tell, G., Izaguirre, I., O'Farrell, I. (Eds.). Freshwater Phytoplankton of Argentina. Adv.Limnol. Vol. 65 (2014). pp. 167-181.

Haykin, S., "Neural Networks. A Comprehensive Foundation". Editorial Prentice Hall. (1999).

Jordan, M. y Mitchell, T.. "Machine learning: Trends, perspectives, and prospects", Science. Vol. 349 (6245), pp. 255-260. (2015).

López Laborde, J., Nagy, G. J., "Hydrography and sediment transport characteristics of the Río de la Plata: a review", in: Perillo, G.M.E., Piccolo, M.C., Pino-Quimira, M. (Eds.), Estuaries of South America: Their Geomorphology and Dynamics. Springer-Verlag, Berlin, Heidelberg. (1999). pp. 133-159.

Nilsson, N., "Principles of Artificial Intelligence", Springer-Verlag, New York (1980).

Nixon, S. W., "Coastal marine eutrophication: a definition, social causes, and future concerns". Ophelia. Vol. 41 (1995). pp. 199–219.

Mianzan, H., Lasta, C., Acha, E., Guerrero, R., Macchi, G., y Bremec, C., "The Río de la Plata Estuary, Argentina-Uruguay", In: Seeliger, U., y Kjerfve, B. (Eds.) Coastal Marine Ecosystems of Latin America Ecological Studies. Vol. 144 (2001). pp. 185-204.

Paerl, H. W., "Assessing and managing nutrient-enhanced eutrophication in estuarine and coastal waters: Interactive effects of human and climatic perturbations". Ecological Engineering, Vol. 26(1). (2006). pp. 40-54.

Roggiero, M. F., "Fitoplancton del Río de la Plata", I. Lilloa. Vol. 37(1). (1988). pp. 137-152.

Sathicq, M. B., "Empleo de descriptores fitoplanctónicos como biomonitores en la evaluación de la calidad del agua en la costa del río de la Plata (Franja Costera Sur)". Tesis de Doctorado. Universidad Nacional de La Plata. (2017).

Shalev-Shwartz, S. and Ben-David, S., "Understanding Machine Learning: From Theory to Algorithms". Cambridge University Press (2014).

Smayda, T. J., "Harmful algal blooms: their ecophysiology and general relevance to phytoplankton blooms in the sea". Limnology and Oceanography. Vol. 42 (5 part 2). (1997). pp. 1137-1153.

Suzuki, K., "Artificial Neural Networks – Architectures And Applications". InTech. (2013).