Máster en Bionformática

Trabajo final de maestría

"Predicción de floraciones algales utilizando algoritmos de aprendizaje automático"

Prediction of algal blooms using machine learning algorithms

Estudiante: Priscilla Siles Valverde

Tutora: Tania Díaz

Resumen

Este estudio aplica técnicas de aprendizaje automático para predecir la concentración de clorofila, utilizada como indicador de biomasa algal, a partir de variables fisicoquímicas en cuerpos de agua. Se compararon modelos desarrollados manualmente (Ridge, Random Forest y XGBoost) con un enfoque automatizado mediante Azure AutoML. El modelo StackEnsemble de AutoML obtuvo el mejor desempeño (R^2 = 0.8123), superando a los métodos tradicionales. Las variables más influyentes fueron la relación fósforo-amonio, el fósforo total, los componentes estacionales y la temperatura del agua. Los resultados evidencian el potencial del aprendizaje automático como herramienta de apoyo para el monitoreo y la predicción de floraciones algales en ecosistemas acuáticos.

Palabras clave: floraciones algales, aprendizaje automático, Azure AutoML, clorofila, predicción ambiental.

Abstract

This study applies machine learning techniques to predict chlorophyll concentration, used as an indicator of algal biomass, from physicochemical variables in aquatic systems. Manually developed models (Ridge, Random Forest, and XGBoost) were compared with an automated approach using Azure AutoML. The StackEnsemble model achieved the best performance (R^2 = 0.8123), outperforming traditional methods. The most influential variables were the phosphorus–ammonium ratio, total phosphorus, seasonal components, and water temperature. The results highlight the potential of machine learning as a valuable tool for monitoring and predicting algal blooms in freshwater ecosystems.

Keywords: algal blooms, machine learning, Azure AutoML, chlorophyll-a, environmental prediction.

Tabla de contenido

I. Introducción

El cambio climático ha alterado significativamente la dinámica de los ecosistemas acuáticos, provocando variaciones en los patrones temporales de eventos biológicos cíclicos, entre ellos, la floración de algas. Este fenómeno, además de reflejar desequilibrios ecológicos, puede utilizarse como un indicador clave del estrés ambiental al que están expuestos los sistemas hídricos (Maguire et al., 2024).

Las floraciones algales nocivas (FANs) son eventos en los que ciertos grupos de algas, principalmente cianobacterias, proliferan de manera excesiva en cuerpos de agua dulce o marinos (Paerl & Otten, 2013). Estos procesos suelen estar asociados con incrementos en la concentración de clorofila (como indicador de biomasa algal) y microcistina, una toxina producida por estas cianobacterias.

El impacto de estos eventos no solo se reduce a nivel local, si no que se extiende a la salud pública, la pesca comercial, el valor de las propiedades y la disponibilidad de agua potable debido a la presencia de toxinas (Bingham et al., 2015). De ahí la importancia de la investigación y predicción de estos eventos nocivos.

Tradicionalmente, la detección de estas floraciones ha dependido de umbrales definidos por el usuario o modelos fenomenológicos como la función de Weibull; la cual es una distribución de probabilidad continua que puede ajustarse a una amplia variedad de formas de patrones, es decir, puede modelar datos asimétricos, lo que permite su uso, ya que, las floraciones algales no siempre siguen un patrón simétrico como una curva normal (Rolinski et al., 2007).

Al ajustar la Weibull a los datos, se pueden identificar puntos de transición que marcan cuándo comienza a aumentar la concentración de clorofila (inicio de la floración), cuándo alcanza su máximo (pico) y cuándo desciende nuevamente, lo que ayuda a estimar fechas clave en el ciclo de floración (Rolinski et al., 2007).

No obstante, estas metodologías presentan limitaciones prácticas, especialmente en entornos heterogéneos como, por ejemplo, el Lago Erie. En respuesta, enfoques más flexibles como los modelos mixtos aditivos generalizados bayesianos (modelos capaces de manejar datos complejos no lineales y agrupados con una estimación basada en probabilidad) (Wood, 2017), han demostrado ser eficaces para capturar patrones estacionales, incluso en presencia de datos censurados o con concentraciones variables, permitiendo estimaciones de eventos fenológicos sin necesidad de fijar límites arbitrarios (Maguire et al., 2024).

Paralelamente, se han explorado técnicas de predicción a partir de algoritmos de aprendizaje automático. El uso de modelos bayesianos ingenuos ha mostrado resultados aceptables al predecir la ocurrencia de floraciones algales con varios días de antelación, facilitando la toma de decisiones en la gestión ambiental (Ho & Michalak, 2015). De igual manera, modelos adaptativos semiparamétricos han sido empleados

exitosamente para pronosticar concentraciones de microcistina a corto plazo, reforzando el potencial del modelado predictivo en tiempo real (Fang et al., 2019).

En este contexto, el presente trabajo tiene como objetivo evaluar la predicción de floraciones algales mediante el uso de Python y modelos de aprendizaje automático. Se emplearán técnicas de análisis de datos y algoritmos de *machine learning* para construir modelos predictivos que permitan anticipar la ocurrencia de estos eventos y mejoren la comprensión y gestión de floraciones algales en ecosistemas acuáticos.

II. Planteamiento de hipótesis

Es posible predecir la presencia y concentración de clorofila como indicador de biomasa algal, en cuerpos de agua a partir de parámetros fisicoquímicos mediante el uso de algoritmos de aprendizaje automático implementados en Python.

III. Objetivos

1. Preparar el conjunto de datos para el modelado predictivo, mediante la carga, limpieza y ajuste de los datos que contienen variables fisicoquímicas relacionadas con la presencia de clorofila, utilizando librerías de Python.

2. Desarrollar y comparar modelos de aprendizaje automático, implementando al menos tres algoritmos supervisados, con el fin de identificar cuál ofrece el mejor rendimiento en la predicción de la presencia de clorofila a partir de variables fisicoquímicas. El desempeño de cada modelo será evaluado utilizando métricas que permitirán seleccionar el modelo más adecuando para su posterior optimización.

3. Optimizar el modelo seleccionado, mediante el ajuste de parámetros y la validación de su rendimiento, con el fin de mejorar su capacidad predictiva antes de su implementación final.

IV. Marco teórico

Las algas son organismos fotosintéticos esenciales en los ecosistemas acuáticos, debido a que, producen oxígeno y constituyen la base de muchas cadenas alimentarias. Además, su abundancia y diversidad refleja las condiciones ambientales, por lo que se consideran bioindicadores efectivos de la calidad del agua (Martínez-Rivera et al., 2016). En este contexto, la bioinformática aplicada a la predicción ambiental, mediante técnicas de *machine learning,* ofrece nuevas posibilidades de análisis a partir de datos multivariados de factores fisicoquímicos y de calidad del agua.

1. Eutrofización y bioindicadores

La eutrofización es el proceso mediante el cual un cuerpo de agua se enriquece excesivamente con nutrientes, particularmente nitrógeno y fósforo, lo que estimula la proliferación de organismos fotosintéticos como el fitoplancton (Dodds & Smith, 2016). En muchos casos, este proceso desencadena floraciones algales descontroladas, algunas de ellas nocivas.

Dentro de este contexto, los bioindicadores como la clorofila permiten inferir el estado trófico del ecosistema. La clorofila, al reflejar la concentración de biomasa algal, se considera un indicador sensible y de bajo costo para el monitoreo de calidad del agua (Carlson, 1977). Otros indicadores incluyen parámetros como el oxígeno disuelto, la transparencia o el índice de estado trófico.

2. Floraciones algales nocivas (FANs)

Ahora bien, existe un fenómeno llamado floraciones algales nocivas (FANs) los cuales son situaciones en los que ciertos grupos de algas, en especial cianobacterias, proliferan de manera excesiva en cuerpos de agua ya sea dulce o marina, provocando efectos negativos sobre los ecosistemas acuáticos, así como en la salud humana y las actividades económicas asociadas.

A diferencia de una simple acumulación de biomasa algal, las FANs se caracterizan por la producción de toxinas biológicas como la microcistina, anatoxina o saxitoxina, que pueden contaminar el agua potable, afectar la fauna acuática, e incluso representar un riesgo para animales domésticos y humanos que entren en contacto o consuman esta agua (Paerl & Otten, 2013).

El origen de las FANs está relacionado con factores tanto naturales como antropogénicos. Entre los principales se encuentran el enriquecimiento por nutrientes (eutrofización), el aumento de la temperatura del agua, la estratificación térmica prolongada, la reducción del caudal o renovación del agua, y la intensificación de fenómenos climáticos extremos (Paerl & Paul, 2012). Estos eventos han aumentado su frecuencia y duración en las últimas décadas, en parte debido al cambio climático y al uso intensivo del suelo para agricultura (Dodds & Smith, 2016).

Las consecuencias de las FANs son múltiples. Desde una perspectiva ecológica, pueden provocar la disminución del oxígeno disuelto (hipoxia o anoxia), lo que afecta gravemente a peces, invertebrados y otros organismos acuáticos. Asimismo, estas floraciones pueden alterar las relaciones tróficas, afectar la biodiversidad local, disminuir la transparencia del agua y afectar el abastecimiento de agua potable para consumo humano (Anderson et al., 2002).

La necesidad de monitorear y predecir las FANs es crítica, dado su impacto y su comportamiento altamente dinámico. Estos eventos pueden desarrollarse en cuestión de días, con patrones irregulares y respuestas complejas a múltiples factores ambientales. Por ello, la combinación de datos fisicoquímicos con enfoques computacionales como el aprendizaje automático ofrece nuevas posibilidades para anticipar su aparición, intensidad y duración, permitiendo a las autoridades ambientales tomar decisiones más informadas y aplicar medidas preventivas eficaces (Maguire et al., 2024).

3. Calidad del agua: parámetros clave

La calidad del agua es evaluada mediante un conjunto de variables fisicoquímicas, las cuales no solo influyen directamente en los organismos acuáticos, sino que determinan la estructura y dinámica del fitoplancton. Su monitoreo sistemático es fundamental para detectar condiciones propicias para la aparición de FANs (Verspagen et al., 2014).

Las floraciones algales nocivas son el resultado de interacciones complejas entre factores bióticos y abióticos, dentro de los cuales los parámetros fisicoquímicos del agua juegan un papel central en su desarrollo, intensidad y duración. Dentro de los principales factores fisicoquímicos asociados a la aparición de FANs se encuentran:

a. Concentración de nutrientes (nitrógeno y fósforo)

El enriquecimiento de cuerpos de agua con nutrientes, especialmente nitrógeno (N) y fósforo (P), es el principal impulsor de la eutrofización. La alta disponibilidad de estos nutrientes promueve un crecimiento rápido de fitoplancton y cianobacterias, favoreciendo condiciones propicias para floraciones masivas (Paerl et al., 2016). Diversos estudios han demostrado que concentraciones elevadas de fósforo disuelto reactivo ($P\text{-}PO_4^{3-}$) están directamente asociadas con la proliferación de *Microcystis spp.* y otras cianobacterias productoras de toxinas (Smith & Schindler, 2009).

b. Temperatura del agua

Las cianobacterias tienen ventajas competitivas en aguas cálidas, mostrando tasas de crecimiento óptimas entre 25 y 30 °C. El aumento de la temperatura del agua, impulsado por el cambio climático, contribuye a la estratificación térmica y favorece a especies capaces de regular su flotabilidad para permanecer en la zona fótica (capa superficial de un cuerpo de agua donde llega suficiente luz solar para permitir la fotosíntesis). Además, temperaturas elevadas reducen la solubilidad del oxígeno, agravando condiciones de hipoxia (Paerl & Paul, 2012).

c. pH del agua

El crecimiento de algas, especialmente de cianobacterias, puede verse favorecido en ambientes ligeramente alcalinos. Algunas especies pueden modificar el pH del entorno mediante la fotosíntesis intensa, elevando los niveles de pH por encima de 9, lo que inhibe la competencia con otras especies de fitoplancton y contribuye a su dominio (Verspagen et al., 2014).

d. Transparencia y penetración de luz

La disponibilidad de luz es esencial para el crecimiento fotosintético. La estratificación térmica prolongada y la baja turbidez en aguas superficiales favorecen el crecimiento de cianobacterias flotantes. La presencia de materia orgánica disuelta o sedimentos en suspensión puede, limitar la penetración de la luz y afectar el tipo de especies dominantes (Zhang et al., 2012).

e. Oxígeno disuelto

Durante las FANs, las tasas de respiración y descomposición de biomasa algal pueden superar la producción de oxígeno, reduciendo drásticamente los niveles de oxígeno disuelto. Esta condición de hipoxia o anoxia promueve la liberación interna de nutrientes desde los sedimentos, retroalimentando el proceso de eutrofización (Rabalais et al., 2010).

f. Conductividad y salinidad

La conductividad eléctrica es un indicador indirecto de la concentración de iones en el agua. Cambios en la conductividad, debido a descargas agrícolas o urbanas, pueden influir en la composición del fitoplancton. Aunque las FANs son más comunes en agua dulce, también pueden ocurrir en ambientes salobres, dependiendo de la tolerancia de las especies presentes (Glibert et al., 2005).

g. Velocidad del flujo y mezclado vertical

Los cuerpos de agua lentos o estancados favorecen la acumulación de biomasa en la superficie. Además, la escasa mezcla vertical permite a las cianobacterias ajustar su posición en la columna de agua para maximizar la captación de luz y nutrientes, lo que no es posible en sistemas con agitación constante (Reynolds et al., 1987).

Estos factores no actúan de forma aislada, sino que se combinan e influyen mutuamente, generando condiciones óptimas para que ciertas especies algales se desarrollen y dominen el ecosistema. Por tanto, su monitoreo integrado es esencial para predecir y mitigar las floraciones algales nocivas de manera eficaz.

4. Concentración de clorofila como indicador de biomasa algal

La clorofila es el principal pigmento fotosintético presente en las algas y cianobacterias, y su concentración en el agua es ampliamente utilizada como un indicador cuantitativo

de la biomasa fitoplanctónica. Debido a su función esencial en la fotosíntesis, la presencia de clorofila permite inferir la cantidad de organismos fotosintéticos activos en un ecosistema acuático (Clesceri et al., 1998).

En estudios limnológicos y oceanográficos, la clorofila ha sido adoptada como una medida estandarizada para monitorear la productividad primaria y evaluar el estado trófico de lagos, ríos y embalses (Carlson, 1977). Concentraciones elevadas de clorofila suelen asociarse con condiciones eutróficas o hipereutróficas, donde el exceso de nutrientes ha favorecido una proliferación descontrolada de fitoplancton, muchas veces acompañada por floraciones algales nocivas (Anderson et al., 2002).

La relación entre clorofila y las FANs ha sido ampliamente documentada. Aunque una alta concentración de clorofila no siempre implica toxicidad, suele correlacionarse con la biomasa de especies dominantes durante una floración, incluyendo cianobacterias productoras de microcistina, como *microcystis spp.* (Fang et al., 2021). Por esta razón, la clorofila se considera una métrica útil y de bajo costo para monitoreo rutinario y modelado predictivo.

Las técnicas de medición de clorofila incluyen métodos espectrofotométricos, fluorométricos e incluso sensores ópticos remotos en plataformas satelitales, los cuales permiten obtener datos espaciales y temporales de alta resolución (Hu et al., 2010). En el contexto del aprendizaje automático, la concentración de clorofila actúa como variable objetivo (target) en modelos supervisados que buscan predecir su valor a partir de parámetros fisicoquímicos, climáticos y espaciales.

Sin embargo, es importante tener en cuenta que la clorofila no distingue entre tipos de algas o especies tóxicas y no tóxicas. Por ello, en investigaciones recientes se ha propuesto complementar su uso con otras métricas, como la concentración de toxinas específicas (por ejemplo, microcistina) o variables ecológicas adicionales que ayuden a interpretar con mayor precisión el riesgo asociado a una determinada floración (Maguire et al., 2024).

5. *Machine Learning* en la predicción de floraciones algales

El aprendizaje automático o *machine learning* es una rama de la inteligencia artificial que se centra en el desarrollo de algoritmos capaces de identificar patrones en grandes volúmenes de datos y realizar predicciones sin ser programados explícitamente para ello (Mitchell, 1997). En el contexto de la gestión ambiental, el aprendizaje automático ha ganado relevancia por su capacidad para modelar fenómenos no lineales complejos y manejar relaciones entre múltiples variables ambientales que influyen en los ecosistemas acuáticos.

La predicción de floraciones algales es un área en la que el aprendizaje automático ha demostrado gran utilidad, particularmente al combinar datos fisicoquímicos (como temperatura, nutrientes, pH, etc.) con registros históricos de clorofila y toxinas. Estas herramientas permiten anticipar la aparición, duración o intensidad de floraciones algales nocivas (Fang et al., 2021).

Los modelos de aprendizaje automático pueden clasificarse, en términos generales, en supervisados y no supervisados. En el caso de la predicción de concentraciones de clorofila, el enfoque supervisado es el más utilizado, ya que permite entrenar algoritmos a partir de datos etiquetados (por ejemplo, registros de clorofila como variable objetivo). Dentro de estos modelos, se han utilizado ampliamente algoritmos como la regresión logística, los bosques aleatorios (random forest), las máquinas de vectores de soporte (SVM), redes neuronales y métodos de ensamble como *gradient boosting* (Shi et al., 2019).

Además, la naturaleza dinámica de los ecosistemas acuáticos hace que el aprendizaje automático sea ideal para incorporar datos en tiempo real y generar sistemas de alerta temprana. Algunos estudios han integrado datos satelitales, meteorológicos y de calidad del agua para alimentar modelos capaces de emitir predicciones con varios días de anticipación, lo que representa una herramienta valiosa para los gestores ambientales (Fang et al., 2021).

El uso de lenguajes de programación como Python, junto con librerías como Pandas, NumPy y matplot, ha facilitado la implementación práctica de estos modelos en ambientes de investigación. Estas herramientas permiten el procesamiento eficiente de grandes volúmenes de datos y la evaluación del rendimiento de los modelos mediante técnicas como la validación cruzada y la optimización de hiperparámetros.

Existen varios tipos de modelos dentro del aprendizaje automático, como los modelos de clasificación, que se emplean para predecir categorías, y los modelos de regresión, utilizados para estimar valores continuos (James et al., 2021). Siendo que, los modelos de clasificación asignan observaciones a categorías discretas, pero no estiman valores continuos con precisión, para efectos del presente trabajo se utilizarán modelos de regresión, dado que la variable objetivo (concentración de clorofila) se expresa en valores continuos y requiere estimaciones numéricas precisas.

Los modelos de regresión permiten establecer y cuantificar la relación entre múltiples variables fisicoquímicas y la biomasa algal, lo que permite capturar patrones y tendencias que podrían no ser evidentes mediante métodos convencionales (Montgomery et al., 2021). Asimismo, la flexibilidad de los modelos de regresión para

adaptarse a relaciones lineales y no lineales los convierte en una opción viable para abordar la complejidad inherente de los ecosistemas acuáticos.

A continuación, se describen brevemente algunos de los más relevantes para la predicción de fenómenos ecológicos como las floraciones algales:

a. Random Forest

Es un modelo de ensamble basado en árboles de decisión. Construye múltiples árboles durante el entrenamiento y realiza predicciones promediando los resultados. Es consistente frente al sobreajuste, maneja bien datos con muchas variables, y permite estimar la importancia relativa de cada predictor (Breiman, 2001).

b. Random Forest Regressor

Es un método de aprendizaje de conjunto basado en la construcción de múltiples árboles de decisión entrenados sobre subconjuntos aleatorios de datos y variables. La predicción final se obtiene promediando los resultados de todos los árboles, lo que reduce la varianza y mejora la estabilidad del modelo (Breiman, 2001).

c. Máquinas de vectores de soporte (SVM)

SVM busca encontrar el hiperplano que mejor separa las clases en los datos. Puede aplicarse tanto a problemas de clasificación como de regresión, y se adapta bien a relaciones no lineales mediante el uso de núcleos (*kernels*). Sin embargo, puede ser sensible a la escala de los datos y a la selección de parámetros (Cortes & Vapnik, 1995).

d. Redes neuronales artificiales

Inspiradas en el funcionamiento del cerebro humano, las redes neuronales son capaces de modelar relaciones no lineales complejas entre variables. Son útiles para trabajar con grandes volúmenes de datos, aunque requieren mayor capacidad computacional y tiempo de entrenamiento (Goodfellow et al., 2016).

e. Gradient Boosting Machines (GBM)

Incluye variantes como XGBoost y LightGBM. Estos modelos de ensamble construyen árboles secuencialmente, donde cada nuevo árbol corrige los errores del anterior. Han mostrado alto rendimiento en tareas de predicción estructurada, aunque requieren un ajuste cuidadoso de hiperparámetros (Chen & Guestrin, 2016).

f. Ridge Regression

La regresión Ridge es una variante de la regresión lineal que incorpora una penalización L2 a los coeficientes del modelo. Esta regularización reduce el sobreajuste y mejora la capacidad de generalización, especialmente en conjuntos de datos donde las variables están muy relacionadas entre sí (Hoerl & Kennard, 1970).

g. XGBoost Regressor

XGBoost es una implementación optimizada del algoritmo de *Gradient Boosting*, que construye árboles de forma secuencial para corregir los errores de los modelos anteriores. Posee gran eficiencia computacional, alta capacidad para manejar datos heterogéneos y un alto rendimiento en problemas de predicción tabular (Chen & Guestrin, 2016).

Como se menciona anteriormente, en este estudio se emplearán únicamente las variantes de regresión de los algoritmos seleccionados, dado que el objetivo es predecir un valor continuo y no clasificar categorías. Por lo tanto, se utilizarán los modelos en sus versiones orientadas a este fin.

6. Conceptos clave en el modelado predictivo

En el diseño de modelos predictivos, especialmente mediante técnicas de aprendizaje automático, es crucial tener en cuenta ciertos conceptos:

a. **Sobreajuste u *overfitting***: ocurre cuando el modelo aprende excesivamente los datos de entrenamiento, perdiendo capacidad de generalización ante datos nuevos. Bajo este contexto debe entenderse el concepto de aprender como la capacidad que tiene un algoritmo para identificar patrones en los datos y mejorar su desempeño en una tarea específica a medida que se le proporciona más información, esto se traduce en ajustar los parámetros internos de un modelo (por ejemplo, los coeficientes en una regresión, los pesos en una red neuronal, o las divisiones en un árbol de decisión) para que las predicciones que generen se parezcan lo más posible a los valores reales (Mitchell, T. M.,1997).

b. **Subajuste o *underfitting*:** se produce cuando el modelo es demasiado simple y no logra captar la complejidad de los datos.

c. **Validación cruzada (*cross-validation*)**: técnica que permite evaluar el rendimiento del modelo dividiendo los datos en varios subconjuntos de entrenamiento y prueba.

d. **Importancia de variables (*feature importance*)**: medida que permite interpretar cuáles atributos del conjunto de datos tienen mayor peso en la predicción del modelo.

Estas técnicas, combinadas con un adecuado preprocesamiento de datos y validación estadística, permiten construir modelos predictivos consistentes y adaptables a diferentes condiciones ambientales. La selección del algoritmo más adecuado se realizará posteriormente en función del comportamiento del conjunto de datos específico y de su capacidad para generalizar los patrones aprendidos.

7. Métricas de evaluación de modelos predictivos

La evaluación cuantitativa del rendimiento de un modelo predictivo es esencial para determinar su capacidad de generalización y su precisión al estimar valores desconocidos. Entre las métricas más utilizadas en el aprendizaje automático aplicado a ciencias ambientales destacan el error cuadrático medio (RMSE), el error absoluto medio (MAE) y el coeficiente de determinación (R^2), cada uno con una función específica para valorar el ajuste del modelo a los datos observados.

El RMSE (Root Mean Squared Error) mide la diferencia promedio entre los valores observados y los predichos, penalizando más los errores grandes, por lo que es útil para detectar desviaciones importantes. Un RMSE bajo indica mayor precisión del modelo (Chai & Draxler, 2014). El MAE (Mean Absolute Error), en cambio, representa la diferencia promedio absoluta entre las predicciones y los valores reales, proporcionando una medida más intuitiva del error medio sin sobreponderar los valores extremos (Willmott & Matsuura, 2005).

Por su parte, el R^2 (coeficiente de determinación) refleja la proporción de la variabilidad de la variable dependiente explicada por el modelo. Un valor de R^2 cercano a 1 indica un ajuste fuerte, mientras que valores bajos o negativos sugieren que el modelo no logra capturar adecuadamente la relación entre las variables (Hawkins, 2004).

En el caso de la validación cruzada (K-fold cross-validation), se entrena y evalúa el modelo en diferentes subconjuntos del dataset, generando una estimación más acertada del rendimiento general. La variabilidad entre pliegues (o folds) indica la estabilidad del modelo: si las métricas varían mucho entre pliegues, el modelo es sensible a la partición de los datos, lo que reduce su capacidad de generalización (Kohavi, 1995). Por el contrario, una variación pequeña sugiere un modelo más consistente.

En este contexto, cuando un modelo presenta un valor medio de R^2 moderado acompañado de alta variabilidad entre pliegues, se interpreta que posee una capacidad de generalización intermedia: logra identificar parcialmente las relaciones entre las variables, pero su desempeño depende de la composición del conjunto de

entrenamiento. Por ello, la interpretación de estas métricas debe considerar tanto el valor promedio como la estabilidad del modelo a través de los pliegues de validación.

8. Análisis de residuales en modelos predictivos

En los modelos estadísticos y de aprendizaje automático, los residuales representan la diferencia entre los valores observados y los valores predichos por el modelo. Matemáticamente, se expresan como:

$$e_i = y_i - \hat{y}_i$$

donde y_i corresponde al valor real observado y al $\hat{y}_i$ valor estimado por el modelo.

El análisis de residuales constituye una etapa de vital importancia en la evaluación del desempeño de un modelo, ya que permite identificar sesgos, patrones sistemáticos o errores de predicción que pueden comprometer la validez de los resultados (Montgomery et al., 2021). En un modelo adecuadamente ajustado, los residuales deben distribuirse de forma aleatoria alrededor de cero, sin mostrar tendencias ni correlaciones con las variables independientes o con los valores predichos.

Asimismo, la dispersión uniforme de los residuales indica variabilidad constante, es decir, que el error del modelo es constante a lo largo de todo el rango de valores. Por el contrario, cuando los residuales muestran una variabilidad creciente o decreciente, se presenta consistencia en la variación del error, lo que refleja una disminución de la precisión en ciertos intervalos de predicción (Kutner et al., 2004).

En el contexto de la predicción de floraciones algales, el análisis de residuales permite verificar la consistencia del modelo y su capacidad para representar correctamente las concentraciones extremas de clorofila. Un comportamiento aleatorio y sin sesgos en los residuales sugiere que el modelo captura adecuadamente la relación entre las variables fisicoquímicas y la biomasa algal, mientras que patrones o dispersión no uniforme pueden indicar la necesidad de ajustar el modelo o incorporar variables adicionales.

En el presente estudio, se propone utilizar la capacidad de estas técnicas para identificar patrones ocultos en los datos y generar predicciones válidas podría contribuir a la prevención y mitigación de floraciones algales nocivas en cuerpos de agua.

V. Metodología

El presente estudio empleará herramientas de ciencia de datos y aprendizaje automático, alineado a un enfoque cuantitativo basado en el análisis de un conjunto de datos históricos, aplicando distintos modelos supervisados con el fin de evaluar su

capacidad predictiva y seleccionar el algoritmo más adecuado en función a los objetivos del estudio.

1. Tipo de estudio

Esta investigación adopta un enfoque cuantitativo, ya que, se basa en el análisis de datos numéricos correspondientes a variables fisicoquímicas y a la concentración de clorofila, expresadas en unidades de medida específicas. Asimismo, es un diseño no experimental y predictivo, centrado en el análisis de datos históricos de las variables de interés.

Se trata de un estudio longitudinal y retrospectivo, ya que utiliza un conjunto de datos recopilado entre los años 2012 y 2021, sin manipulación directa de variables en tiempo real.

El diseño metodológico es de tipo comparativo y exploratorio, dado que se evaluará el rendimiento de múltiples algoritmos de aprendizaje automático con el fin de determinar cuál ofrece mayor precisión en la predicción de floraciones algales nocivas. Esta aproximación permite examinar relaciones entre variables fisicoquímicas y la concentración de clorofila mediante técnicas supervisadas de *machine learning,* sin intervención sobre el entorno natural.

2. Conjunto de datos y muestreo

El análisis se basa en tres archivos de datos:

- **tim_cols_2012_2018.csv** (1244 registros)
- **tim_cols_2019.csv** (219 registros)
- **tim_cols_2020_2021.csv** (413 registros)

Los datos se obtuvieron del conjunto publicado por Bailey et al. (2024) en la plataforma Zenodo, el cual recopila concentraciones de clorofila y microcistina particulada en diferentes sitios del lago Erie entre 2012 y 2021. Cada archivo incluye 36 variables (2012-2018) y 33 variables (2019-2021).

Las muestras se recolectaron en 43 sitios únicos del lago Erie entre 2012 y 2021, con frecuencia semanal, abarcando un periodo de 10 años.

El muestreo es no probabilístico intencional. Se priorizarán registros con calidad suficiente, los valores faltantes en variables numéricas con <50 % de ausencia se imputarán mediante K-Nearest Neighbors (KNN) y se descartarán únicamente las filas sin la variable objetivo (clorofila). Además, se dará tratamiento de valores extremos mediante el método IQR (capeo sin eliminación de datos) y creación de variables derivadas.

El método KNN se utiliza en estudios ambientales por su capacidad de conservar las relaciones naturales entre variables interdependientes. A diferencia de técnicas simples como la imputación por la media, KNN identifica las cinco observaciones más similares a cada registro incompleto y utiliza el promedio de estas para estimar el valor faltante, lo que permite preservar las relaciones entre variables ambientales (Troyanskaya et al., 2001).

En el caso del método de IQR o rango intercuartílico, es una técnica estadística utilizada para detectar y tratar valores extremos (outliers) en un conjunto de datos. Para efectos de este estudio, los valores extremos se tratarán mediante capeo (winsorización), lo que implica reemplazar los valores fuera de esos límites por los valores máximo o mínimo permitidos dentro del rango (en lugar de eliminarlos). Esto con el objetivo de obtener un conjunto de datos consistente sin perder la estructura necesaria para llevar a cabo la investigación.

3. Descripción del preprocesamiento de datos

El preprocesamiento de los datos se realizará en el lenguaje de programación Python (versión: 3.11.9), utilizando las librerías Pandas para manipulación de datos, NumPy para operaciones numéricas, Scikit-learn para imputación KNN y escalamiento robusto y SciPy para análisis estadístico.

Se realizará de manera general la revisión de valores faltantes, detección de duplicados, conversión de formatos y filtrado de variables no relevantes para asegurar la integridad y consistencia del conjunto de datos previo al modelado.

A continuación, se detallan los pasos a seguir junto con el código utilizado:

3.1 Integración y exploración inicial de fuentes de datos

3.1.1 Se utilizarán tres archivos CSV correspondientes a campañas de monitoreo de la NOAA/GLERL en el Lago Erie para los periodos 2012–2018, 2019 y 2020–2021.

3.1.2 Los archivos serán cargados con pandas.read_csv() y combinados mediante pandas.concat() con ignore_index=True, preservando todas las columnas originales y manteniendo la integridad temporal.
Código:

```
# Rutas de los archivos
file_2012_2018 = '../data/raw/tim_cols_2012_2018.csv'
file_2019 = '../data/raw/tim_cols_2019.csv'
file_2020_2021 = '../data/raw/tim_cols_2020_2021.csv'
```

```
# Cargar datasets
df_2012_2018 = pd.read_csv(file_2012_2018)
df_2019 = pd.read_csv(file_2019)
df_2020_2021 = pd.read_csv(file_2020_2021)

# Consolidar los tres datasets en uno solo
df_raw = pd.concat([df_2012_2018, df_2019, df_2020_2021],
ignore_index=True)
```

3.1.3 Se realizará un análisis exploratorio inicial para identificar la estructura, calidad y distribución de las variables disponibles.

Código:

```
# Inspeccion de estructuras base
print(f"Dimensiones: {df_raw.shape[0]} filas x {df_raw.shape[1]}
columnas")
print(f"Rango temporal: {df_raw['date'].min()} a
{df_raw['date'].max()}")
df_raw.head(3)

# Convertir valores bajo límite de detección (e.g., '<0.1') a
NaN
columnas_numericas =
df_raw.select_dtypes(include=['object']).columns.tolist()
columnas_numericas = [col for col in columnas_numericas if col
not in ['date', 'station']]

for col in columnas_numericas:
    df_raw[col] = pd.to_numeric(df_raw[col], errors='coerce')

# Extraccion de variables temporales
df_raw['date'] = pd.to_datetime(df_raw['date'], errors='coerce')
df_raw['year'] = df_raw['date'].dt.year
df_raw['doy'] = df_raw['date'].dt.dayofyear

# Análisis de valores faltantes:
missing_pct = (df_raw.isnull().sum() / len(df_raw) *
100).sort_values(ascending=False)
missing_pct_filtered = missing_pct[missing_pct > 0]

print(f"\nVariables con valores faltantes (top 10):")
for var, pct in missing_pct_filtered.head(10).items():
    print(f"  {var:15s}: {pct:5.1f}%")
```

3.2 Estandarización y conversión de valores

3.2.1 Las columnas con valores reportados bajo el límite de detección, identificados por el prefijo "<" (por ejemplo, "<0.1"), serán

transformados a valores a valores faltantes (NaN) mediante pd.to_numeric() con el parámetro errors='coerce'.

Esta conversión permite identificar los valores no numéricos en el dataset. Estos valores faltantes serán posteriormente imputados mediante el método KNN, lo que permite estimar valores basándose en observaciones similares en lugar de asumir el límite de detección exacto.
Código:

```
# Identificar columnas numéricas (excluyendo date, year, doy)
columnas_numericas =
df_raw.select_dtypes(include=['object']).columns.tolist()
columnas_numericas = [col for col in columnas_numericas if col
not in ['date', 'station']]

# Contador de conversiones
conversiones_totales = 0

for col in columnas_numericas:
    # Contar valores no numéricos antes de la conversión
    antes = df_raw[col].isna().sum()

    # Convertir a numérico, forzando errores a NaN
    df_raw[col] = pd.to_numeric(df_raw[col], errors='coerce')

    # Contar valores NaN después de la conversión
    despues = df_raw[col].isna().sum()
    conversiones = despues - antes

    if conversiones > 0:
        conversiones_totales += conversiones
        print(f"  {col:15s}: {conversiones:4d} valores no
numéricos convertidos a NaN")
```

3.2.2 Se aplicará el método IQR (Interquartile Range) de Tukey para el tratamiento de valores extremos en la variable objetivo (clorofila). Este método calcula los cuartiles Q1 (percentil 25) y Q3 (percentil 75), el rango intercuartílico (IQR = Q3 - Q1), y define límites en Q1 - 1.5×IQR y Q3 + 1.5×IQR.

Los valores que exceden estos límites serán capeados (no eliminados) a los umbrales correspondientes, preservando la información sobre floraciones intensas sin que valores extremos distorsionen excesivamente el modelado.
Código:

```
# Calcular cuartiles e IQR
Q1 = df_raw['chl'].quantile(0.25)
Q3 = df_raw['chl'].quantile(0.75)
IQR = Q3 - Q1

# Calcular límites (método de Tukey)
limite_inferior = Q1 - 1.5 * IQR
limite_superior = Q3 + 1.5 * IQR

# Identificar outliers
outliers_inferiores = df_raw['chl'] < limite_inferior
outliers_superiores = df_raw['chl'] > limite_superior
total_outliers = outliers_inferiores.sum() +
outliers_superiores.sum()

# CAPEAR outliers (no eliminar - enfoque de Brownlee)
df_raw.loc[outliers_inferiores, 'chl'] = limite_inferior
df_raw.loc[outliers_superiores, 'chl'] = limite_superior
```

3.2.3 Se aplicará RobustScaler de la imputación KNN con el objetivo de mejorar la estimación de los valores faltantes. Este método utiliza la mediana y el rango intercuartílico (IQR) en lugar de la media y la desviación estándar, lo que lo hace más resistente a valores extremos. El escalamiento será temporal: una vez completada la imputación con K-NN (k = 5 vecinos), los datos se revertirán a su escala original.

Es importante señalar que el dataset final utilizado para el modelado se mantendrá en su escala original, sin aplicar una estandarización permanente. Aunque algunos algoritmos, como la regresión Ridge, son sensibles a la escala de las variables, otros, como Random Forest, no se ven afectados, ya que sus divisiones internas dependen de umbrales relativos y no de magnitudes absolutas.

Para efectos de la investigación se optará por preservar la escala natural de las variables para mantener la interpretabilidad de los coeficientes y permitir que cada algoritmo procese la información en su forma más representativa.

Código:

```
# Imputación KNN con escalamiento robusto

from sklearn.impute import KNNImputer
from sklearn.preprocessing import RobustScaler
```

```
# Identificar columnas numéricas con valores faltantes
numeric_cols =
df_raw_clean.select_dtypes(include=[np.number]).columns.tolist()
cols_con_faltantes =
df_raw_clean[numeric_cols].columns[df_raw_clean[numeric_cols].is
null().any()].tolist()

print(f"Columnas con valores faltantes:
{len(cols_con_faltantes)}")

# Escalamiento con RobustScaler
scaler_knn = RobustScaler()
df_scaled = df_raw_clean[numeric_cols].copy()
df_scaled[numeric_cols] =
scaler_knn.fit_transform(df_scaled[numeric_cols])

# Imputación KNN
imputer = KNNImputer(n_neighbors=5)
df_scaled[numeric_cols] =
imputer.fit_transform(df_scaled[numeric_cols])

# Revertir escalamiento
df_raw_clean[numeric_cols] =
scaler_knn.inverse_transform(df_scaled[numeric_cols])

print(f"Valores faltantes después de imputación:
{df_raw_clean[numeric_cols].isnull().sum().sum()}")
```

3.3 Tratamiento adicional de valores atípicos

3.3.1

Se aplicará winsorización en percentiles 5-95 como segundo tratamiento de valores extremos sobre la variable objetivo (clorofila), después del método IQR. Esta técnica reemplaza los valores en las colas de la distribución por los umbrales de los percentiles 5 y 95, reduciendo el impacto de outliers severos.

La combinación de ambos métodos (IQR seguido de winsorización) permite un tratamiento gradual: el IQR maneja valores atípicos moderados, mientras que la winsorización captura casos extremos residuales, mejorando la estabilidad del modelado sin eliminar información sobre floraciones intensas.

Código:

```
from scipy.stats.mstats import winsorize

# Guardar valores antes de winsorización
chl_antes_wins = df_clean['chl'].copy()
```

```
# Aplicar winsorización en percentiles 5-95
df_clean['chl'] = winsorize(df_clean['chl'], limits=[0.05,
0.05])

print(f"Estadísticas antes de winsorización:")
print(f"  Min: {chl_antes_wins.min():.2f} µg/L")
print(f"  Max: {chl_antes_wins.max():.2f} µg/L")

print(f"\nEstadísticas después de winsorización:")
print(f"  Min: {df_clean['chl'].min():.2f} µg/L")
print(f"  Max: {df_clean['chl'].max():.2f} µg/L")

valores_ajustados = (chl_antes_wins != df_clean['chl']).sum()
print(f"\nValores ajustados: {valores_ajustados}
({valores_ajustados/len(df_clean)*100:.1f}%)")
```

3.4 Selección de variables predictoras y creación de variables derivadas

3.4.1 Implementar una metodología de selección de variables basada en principios de causalidad ambiental, eliminando indicadores biológicos que representen efectos de las floraciones algales en lugar de causas.

Código:

```
# Se excluyen variables de respuesta biologica (pmicro, dmicro,
phyco)

# Lista de variables causales deseadas

variables_causales_deseadas = [
    'date', 'site', 'site_depth', 'depth_m', 'depth_cat', 'lat',
'lon',
    'secchi', 'ctd_temp', 'turb', 'chl',  # Variable objetivo
    'tp', 'tdp', 'srp', 'nh4', 'no3_2', 'poc', 'pon', 'cdom',
'tss', 'vss',
    'doy', 'year'
]

# Seleccionar solo las que existen en el dataset
variables_causales = [col for col in variables_causales_deseadas
if col in df_raw_clean.columns]
df_clean = df_raw_clean[variables_causales].copy()
```

3.4.2 Crear variables derivadas, para ello, se aplicará una codificación trigonométrica del día del año, generando dos variables (sin_doy, cos_doy). Esta transformación convierte el calendario en un círculo unitario, de modo que los días 1 y 365 quedan representados como puntos cercanos. De esta forma, se evita que el modelo interprete erróneamente una "ruptura" entre diciembre y enero, y se capturan de

forma continua los patrones estacionales repetitivos que influyen en la concentración de clorofila.
Código:

```
# Creación de variables derivadas

# Variables estacionales (codificación trigonométrica)

df_clean['sin_doy'] = np.sin(2 * np.pi * df_clean['doy'] /
365.25)

df_clean['cos_doy'] = np.cos(2 * np.pi * df_clean['doy'] /
365.25)
```

3.4.3 Generación de una variable de balance nutricional (tp_nh4_ratio) que representa la relación fósforo/amonio.
Código:

```
# Variable de balance nutricional
df_clean['tp_nh4_ratio'] = np.where(
    (df_clean['nh4'].notna()) & (df_clean['nh4'] > 0),
    df_clean['tp'] / df_clean['nh4'],
    0
)
```

3.4.4 Se seleccionarán 10 variables predictoras finales basadas en principios de causalidad ambiental: nutrientes limitantes (tp, srp, nh4, no3_2), condiciones físicas (ctd_temp, secchi, depth_m), patrones temporales (sin_doy, cos_doy) y balance nutricional (tp_nh4_ratio).

Se excluirán variables con potencial de introducir sesgo circular como oxígeno disuelto (efecto de fotosíntesis) y biomarcadores orgánicos (poc, pon, cdom).
Código:

```
# 10 drivers ambientales causales
predictoras_finales = [
    # Nutrientes
    'tp', 'srp', 'nh4', 'no3_2',
    # Parámetros físicos
    'ctd_temp', 'secchi', 'depth_m',
    # Factores estacionales
    'sin_doy', 'cos_doy',
    # Balance nutricional
    'tp_nh4_ratio'
]

# Variable objetivo
```

```
variable_objetivo = 'chl'

# Crear dataset final para modelado
columnas_finales = predictoras_finales + [variable_objetivo]
df_final = df_clean[columnas_finales].copy()
```

3.5 Exportación del dataset validado

3.5.1 El dataset procesado se guardará como dataset_final_modelado.csv, conteniendo 11 variables: las 10 variables predictoras seleccionadas (tp, srp, nh4, no3_2, ctd_temp, secchi, depth_m, sin_doy, cos_doy, tp_nh4_ratio) más la variable objetivo (chl).
Código:

```
# Exportación del dataset final
columnas_finales_export = predictoras_finales + ['chl']
df_export = df_final[columnas_finales_export].copy()

output_path = '../data/processed/dataset_final_modelado.csv'
df_export.to_csv(output_path, index=False)
```

4. Enfoque metodológico

Para efectos de la investigación, se compararán cuatro modelos. Asimismo, se aplicará una metodología doble para comparar los resultados de los modelos creados manualmente con los generados por Azure AutoML.

Este enfoque es con el objetivo de evaluar no solo cuál modelo alcanzó un mejor rendimiento predictivo, sino también si los resultados son coherentes desde el punto de vista científico, siguiendo recomendaciones de estudios recientes que destacan la utilidad de combinar ambos métodos para mejorar la precisión e interpretabilidad de los modelos en contextos ambientales (Luo et al., 2022).

A continuación, se describe cada uno:

Tabla1. Selección de modelos de aprendizaje automático según su función.

Modelo (supervisado)	Función y criterio de selección
Ridge Regression (scikit-learn)	Proporciona una línea base rápida y fácil de interpretar: comprueba cuánto explica un modelo estrictamente lineal. La versión Ridge añade regularización, evitando que los coeficientes crezcan en exceso si hay colinealidad.

Random Forest Regressor (scikit-learn)	Conjunto de árboles de decisión entrenados sobre submuestras diferentes. Captura interacciones y no linealidades sin requerir mucho ajuste, maneja variables mixtas y cierta proporción de datos faltantes. Devuelve importancias de variables útiles para interpretación.
XGBoost (Extreme Gradient Boosting)	Algoritmos de *gradient boosting* que construyen árboles secuenciales para minimizar el error residual. Suelen lograr la mejor precisión en datos tabulares y ofrecen explicabilidad con SHAP. Se ajustarán parámetros como *learning rate*, profundidad y número de árboles con *early stopping*.
Azure AutoML (Automated Machine Learning)	Es una plataforma que automatiza la evaluación de diferentes algoritmos y prueba técnicas de ingeniería de características y ajusta combinaciones de hiperparámetros mediante optimización bayesiana. De esta forma, permite comparar los enfoques tradicionales hechos de manera manual con métodos automatizados más recientes, incluyendo ensambles avanzados como *VotingEnsemble*.

Fuente: elaboración propia.

Si bien en la literatura se mencionan otros algoritmos de regresión como las Máquinas de Vectores de Soporte (SVM) o las redes neuronales artificiales para la predicción de variables ambientales, en el presente estudio se optó por no implementarlos. Esta exclusión obedece a criterios de alcance del trabajo, disponibilidad de recursos computacionales y enfoque en algoritmos con mayor interpretabilidad y facilidad de ajuste en el contexto de los datos disponibles.

5. Herramientas empleadas

Las herramientas que se utilizarán son el ambiente de desarrollo Jupyter Notebook y Azure Machine Learning Studio. Para implementación local se incluyen las librerías Pandas, NumPy, matplotlib y seaborn para análisis exploratorio. Además se añaden scikit-learn, XGBoost, SHAP y Azure ML para implementar, interpretar y automatizar los modelos respectivamente.

6. Desarrollo del modelo predictivo

Una vez completado el preprocesamiento de datos (descrito en la sección 3), se procederá con el desarrollo de los modelos predictivos utilizando el dataset final (dataset_final_modelado.csv), el cual contiene 1,869 registros con las 10 variables predictoras seleccionadas y la variable objetivo (concentración de clorofila en µg/L).

Cabe mencionar que, debido a su extensión, el código desarrollado para esta sección se encuentra disponible en el Anexo 4.

6.1 Estrategia de validación y división de datos

6.1.1 Modelos manuales (Ridge Regression, Random Forest, XGBoost):

Se implementa una estrategia de validación en dos niveles:

a. División de datos: El dataset se divide en 80% para entrenamiento y 20% para prueba, con random_state=42 para garantizar reproducibilidad.
b. Validación cruzada K-fold: Dentro del conjunto de entrenamiento, se aplica validación cruzada con K=5 pliegues para el ajuste de hiperparámetros mediante RandomizedSearchCV (Random Forest, XGBoost) o RidgeCV (Ridge Regression). Esto permite evaluar múltiples combinaciones de parámetros y seleccionar la configuración óptima que maximice el R^2.
c. Evaluación final: Una vez identificados los mejores hiperparámetros mediante validación cruzada, se entrena el modelo final sobre todo el conjunto de entrenamiento y se evalúa su rendimiento sobre el conjunto de prueba independiente.
d. Métrica de optimización: R^2 (coeficiente de determinación).

6.1.2 Azure AutoML:

Se configurará con división 80% entrenamiento / 20% validación, establecida manualmente al crear el training set. El sistema utiliza random_state=42 para reproducibilidad y explora automáticamente múltiples algoritmos (GradientBoosting, RandomForest, LightGBM, entre otros), generando finalmente un StackEnsemble que combina los mejores modelos individuales con pesos optimizados.

Configuración aplicada:

- División de datos: 80% entrenamiento / 20% validación (definida al crear training set)
- random_state: 42 (para reproducibilidad)
- task_type: 'regression'
- primary_metric: 'r2_score'
- Algoritmos explorados automáticamente: GradientBoosting (×4), RandomForest (×3)
- Modelo final: StackEnsemble con 7 modelos base y pesos optimizados

7. Evaluación del modelo

La evaluación de los modelos se realizará siguiendo la estrategia de validación descrita en la sección 6.1. Para ello, se emplearán métricas cuantitativas que permiten comparar de manera objetiva el ajuste del modelo a los datos observados y su capacidad para explicar la variabilidad de la variable dependiente.

El propósito de esta fase será valorar la capacidad de los modelos para predecir la concentración de clorofila y verificar si su desempeño es consistente tanto en términos estadísticos, así como desde una perspectiva ecológica. Además, se analizará el comportamiento de los residuales y la relevancia de las variables predictoras, con el fin de garantizar que las conclusiones del modelo sean interpretables y coherentes científicamente.

7.1 Métricas de evaluación empleadas

Para evaluar el rendimiento de los modelos se aplicarán métricas generalmente utilizadas en problemas de regresión y previamente descritas. Estas permiten medir la precisión de las predicciones y el grado de ajuste entre los valores observados y los estimados.

7.1.1 RMSE (Root Mean Squared Error):

Mide la magnitud promedio de los errores de predicción, penalizando de forma más severa los errores grandes.

El código utilizado para el cálculo en cada modelo se describe a continuación:

a. Ridge Regression

```
mejor_ridge = Ridge(alpha=ridge_cv.alpha_)
mejor_ridge.fit(X_entrenamiento_escalado, y_entrenamiento)
# Predicciones
y_entrenamiento_pred = mejor_ridge.predict(X_entrenamiento_escalado)
y_prueba_pred = mejor_ridge.predict(X_prueba_escalado)
# Métricas de rendimiento
r2_entrenamiento = r2_score(y_entrenamiento, y_entrenamiento_pred)
r2_prueba = r2_score(y_prueba, y_prueba_pred)
mae_entrenamiento = mean_absolute_error(y_entrenamiento, y_entrenamiento_pred)
mae_prueba = mean_absolute_error(y_prueba, y_prueba_pred)
rmse_entrenamiento = np.sqrt(mean_squared_error(y_entrenamiento,
y_entrenamiento_pred))
rmse_prueba = np.sqrt(mean_squared_error(y_prueba, y_prueba_pred))
```

b. Random Forest

```
mejor_rf = busqueda_aleatoria.best_estimator_

mejor_rf.fit(X_entrenamiento, y_entrenamiento)

# Predicciones

y_entrenamiento_pred = mejor_rf.predict(X_entrenamiento)

y_prueba_pred = mejor_rf.predict(X_prueba)

# Métricas de rendimiento

r2_entrenamiento = r2_score(y_entrenamiento, y_entrenamiento_pred)

r2_prueba = r2_score(y_prueba, y_prueba_pred)

mae_entrenamiento = mean_absolute_error(y_entrenamiento, y_entrenamiento_pred)

mae_prueba = mean_absolute_error(y_prueba, y_prueba_pred)

rmse_entrenamiento = np.sqrt(mean_squared_error(y_entrenamiento, y_entrenamiento_pred)) rmse_prueba = np.sqrt(mean_squared_error(y_prueba, y_prueba_pred))
```

c. XGBoost

```
mejores_parametros = busqueda_aleatoria.best_params_.copy()

# Agregar parada temprana para entrenamiento óptimo

mejor_xgb = xgb.XGBRegressor(

    **mejores_parametros,

    random_state=42,

    n_jobs=-1,

    tree_method='hist',

    early_stopping_rounds=50,

    eval_metric='rmse'

)

# Ajustar con conjunto de validación

mejor_xgb.fit(

    X_entrenamiento, y_entrenamiento,

    eval_set=[(X_entrenamiento, y_entrenamiento), (X_prueba, y_prueba)],

    verbose=False

)

# Predicciones

y_entrenamiento_pred = mejor_xgb.predict(X_entrenamiento)

y_prueba_pred = mejor_xgb.predict(X_prueba)
```

```
# Métricas de rendimiento

r2_entrenamiento = r2_score(y_entrenamiento, y_entrenamiento_pred)

r2_prueba = r2_score(y_prueba, y_prueba_pred)

mae_entrenamiento = mean_absolute_error(y_entrenamiento, y_entrenamiento_pred)

mae_prueba = mean_absolute_error(y_prueba, y_prueba_pred)

rmse_entrenamiento = np.sqrt(mean_squared_error(y_entrenamiento,
y_entrenamiento_pred))

rmse_prueba = np.sqrt(mean_squared_error(y_prueba, y_prueba_pred))
```

7.1.2 MAE (Mean Absolute Error)

Calcula el promedio de las diferencias absolutas entre valores observados y predichos.

Código:

```
mae_entrenamiento = mean_absolute_error(y_entrenamiento, y_entrenamiento_pred)

mae_prueba = mean_absolute_error(y_prueba, y_prueba_pred)
```

7.1.3 R^2 (coeficiente de determinación)

Indica la proporción de la variabilidad de la variable dependiente que es explicada por el modelo.

Código:

```
r2_entrenamiento = r2_score(y_entrenamiento, y_entrenamiento_pred)

r2_prueba = r2_score(y_prueba, y_prueba_pred)
```

7.1.4 CV R^2 (validación cruzada)

Evalúa la capacidad de un modelo para generalizar a datos no vistos, dividiendo el conjunto disponible en múltiples particiones de entrenamiento y prueba.

Código Ridge Regression:

```
puntajes_vc = cross_val_score(mejor_ridge, X_escalado, y, cv=5, scoring='r2')
```

Código Random Forest:

```
puntajes_vc = cross_val_score(mejor_rf, X, y, cv=5, scoring='r2')
```

Código XGBoost:

```
puntajes_vc = cross_val_score(xgb_vc, X, y, cv=5, scoring='r2')
```

7.1.5 Métricas del modelo automatizado Azure AutoML

Las métricas se obtendrán directamente desde el portal de Azure Machine Learning, donde el proceso de entrenamiento y validación se ejecuta de forma automática bajo los mismos principios de evaluación estadística.

7.2 Criterios de interpretación

La interpretación de los resultados se basará en las métricas promedio obtenidas en el conjunto de prueba y en la variabilidad observada entre los pliegues de validación cruzada. Esto con el fin de evaluar simultáneamente la capacidad de generalización del modelo y su desempeño específico sobre datos no vistos durante el entrenamiento.

7.2.1 Interpretación de métricas promedio

Las métricas promedio (RMSE, MAE y R^2) se evaluarán en conjunto para determinar la calidad del ajuste. Un modelo se considerará más preciso cuando presente valores bajos de error (RMSE y MAE) y un R^2 más alto, lo que refleja una mejor correspondencia entre los valores observados y las predicciones.

La comparación se realizará en las mismas unidades de la variable objetivo (µg/L de clorofila), lo que permitirá interpretar los errores en términos ecológicos reales.

Asimismo, se analizará la relación entre RMSE y MAE: una diferencia amplia entre ambas métricas podría indicar la presencia de valores extremos o una distribución asimétrica de los errores. En cambio, valores similares sugerirán una dispersión más homogénea en las predicciones.

7.2.2 Interpretación de la variabilidad entre pliegues

La variabilidad entre pliegues, medida a partir de la desviación estándar del CV R^2, sirve para evaluar la consistencia del modelo. Una desviación baja implicará que el modelo mantiene un rendimiento estable independientemente de cómo se dividan los datos, lo que se asocia con una buena capacidad de generalización.

Por el contrario, una variabilidad elevada reflejará que el modelo depende demasiado de ciertos subconjuntos del entrenamiento, indicando posibles signos de sobreajuste o necesidad de regularización adicional.

7.3 Análisis de residuales

El análisis de residuales tiene el objetivo de comprobar la coherencia de las predicciones y descartar la presencia de sesgos sistemáticos en los modelos. Los residuales se obtendrán como la diferencia entre los valores observados y los predichos.

7.3.1 Análisis de residuales vs. predicciones

Se graficará la relación entre las predicciones y los residuales mediante un diagrama de dispersión. Un comportamiento aleatorio de los puntos alrededor de cero refleja un

modelo equilibrado, mientras que la presencia de una tendencia o forma definida indica posibles sesgos sistemáticos.

Implementación:

- Se calcularán los residuales como: residuos = y_real - y_predicho
- Se generará un gráfico de dispersión de residuales vs. valores predichos
- Se incluirá una línea de referencia en cero para facilitar la identificación de patrones

Los gráficos de dispersión se obtendrán siguiendo el código:

a. Ridge Regression

```
# Gráfico de residuos
residuos = y_prueba - y_prueba_pred
axes[1, 0].scatter(y_prueba_pred, residuos, alpha=0.6)
axes[1, 0].axhline(y=0, color='red', linestyle='--', alpha=0.8)
axes[1, 0].set_xlabel('Clorofila Predicha (µg/L)')
axes[1, 0].set_ylabel('Residuos (µg/L)')
axes[1, 0].set_title('Gráfico de Residuos')
```

b. Random Forest

```
# Gráfico de residuos
residuos = y_prueba - y_prueba_pred
axes[1, 0].scatter(y_prueba_pred, residuos, alpha=0.6)
axes[1, 0].axhline(y=0, color='red', linestyle='--', alpha=0.8)
axes[1, 0].set_xlabel('Clorofila Predicha (µg/L)')
axes[1, 0].set_ylabel('Residuos (µg/L)')
axes[1, 0].set_title('Gráfico de Residuos')
```

c. XGBoost

```
# Gráfico de residuos
residuos = y_prueba - y_prueba_pred  # ← LÍNEA 423: Cálculo de residuos
axes[1, 1].scatter(y_prueba_pred, residuos, alpha=0.6)  # ← LÍNEA 424: Scatter plot
axes[1, 1].axhline(y=0, color='red', linestyle='--', alpha=0.8)  # ← LÍNEA 425: Línea de referencia
axes[1, 1].set_xlabel('Clorofila Predicha (µg/L)')
```

```
axes[1, 1].set_ylabel('Residuos (µg/L)')
axes[1, 1].set_title('Análisis de Residuos')
```

En el caso del modelo automatizado, Azure AutoML generara automáticamente el histograma de residuales dentro del portal de experimentación. Este gráfico permitirá evaluar visualmente la distribución de los errores sin requerir código adicional.

7.4 Análisis de importancia de variables

El análisis de importancia de características identifica cuáles variables ambientales tienen mayor peso en la predicción de la concentración de clorofila. Este análisis se realizará para cada modelo implementado, utilizando las métricas específicas de importancia que cada algoritmo proporciona.

7.4.1 Cálculo de importancia de características

a. Ridge Regression:

Se utilizarán los coeficientes de regresión estandarizados como medida de importancia. Los coeficientes serán ordenados por su valor absoluto, identificando así las variables con mayor influencia en la predicción, independientemente de la dirección del efecto (positivo o negativo).

Código:

```
# Coeficientes Ridge como importancia de características
coeficientes = mejor_ridge.coef_
importancia_caracteristicas = pd.DataFrame({
    'caracteristica': columnas_caracteristicas,
    'coeficiente': coeficientes,
    'coeficiente_abs': np.abs(coeficientes)
}).sort_values('coeficiente_abs', ascending=False)
```

b. Random Forest:

Se empleará la importancia basada en la reducción de impureza (Gini importance), que cuantifica cuánto contribuye cada variable a disminuir el error de predicción en los árboles del ensamble. Adicionalmente, se calculará la importancia por permutación para evaluar la validez de los resultados.

Código:

```
# 1. Importancia basada en Gini
```

```
importancia_gini = pd.DataFrame({
    'caracteristica': columnas_caracteristicas,
    'importancia_gini': mejor_rf.feature_importances_
}).sort_values('importancia_gini', ascending=False)
# 2. Importancia por permutación
importancia_perm = permutation_importance(
    mejor_rf, X_prueba, y_prueba, n_repeats=10, random_state=42, n_jobs=-1
)
importancia_perm_df = pd.DataFrame({
    'caracteristica': columnas_caracteristicas,
    'importancia_perm': importancia_perm.importances_mean,
    'desv_est_perm': importancia_perm.importances_std
}).sort_values('importancia_perm', ascending=False)
```

c. XGBoost:

Se utilizará la importancia basada en ganancia (gain importance), que mide la mejora promedio en la métrica de pérdida que aporta cada variable cuando se utiliza para dividir los nodos del árbol.

Código:

```
# 1. Importancia basada en ganancia
importancia_ganancia = pd.DataFrame({
    'caracteristica': columnas_caracteristicas,
    'importancia_ganancia': mejor_xgb.feature_importances_
}).sort_values('importancia_ganancia', ascending=False)

# 2. Múltiples tipos de importancia
booster = mejor_xgb.get_booster()
tipos_importancia = ['weight', 'gain', 'cover']
diccionario_importancia = {}
for tipo_imp in tipos_importancia:
    diccionario_importancia[tipo_imp] =
booster.get_score(importance_type=tipo_imp)
```

d. Azure AutoML:

En el modelo automatizado, la importancia de variables se obtendrá directamente desde los archivos de explicación generados por Azure AutoML. Estos archivos se almacenaran dentro de la ruta de los resultados del experimento:

explanation/2172c0a5/

├── global_names/0.interpret.json

├── global_rank/0.interpret.json

└── global_values/0.interpret.json

El archivo global_names lista las variables con mayor peso en la predicción, mientras que global_rank y global_values detallan su orden y magnitud de contribución.

7.5 Comparación de modelos

7.5.1 Tabular resultados de métricas de validación cruzada para cada algoritmo (Ridge, Random Forest, XGBoost, Azure AutoML).

7.5.2 Evaluar tanto rendimiento predictivo (R^2, RMSE, MAE) como estabilidad.

7.6 Criterios de selección del mejor modelo

El modelo seleccionado se definirá considerando los siguientes criterios:

a. Mayor R^2 y menor RMSE/MAE, reflejando una mejor capacidad de ajuste y menor error promedio.
b. Estabilidad en validación cruzada, evidenciada por una variabilidad baja entre pliegues.
c. Comportamiento coherente de residuales, sin sesgos sistemáticos ni subestimaciones en valores extremos.
d. Validez científica y ambiental, se debe verificar que las variables más influyentes tengan sentido en el contexto ambiental.
e. Comparación entre los datos obtenidos y evidencia científica que corrobore si lo valores están dentro de rangos aceptables de rendimiento.

8. Herramientas y entornos de desarrollo

a. Lenguaje: Python 3.11.9
b. Entorno interactivo principal: Jupyter Notebook, ejecutado localmente a través de Visual Studio Code, usado para el desarrollo, pruebas y documentación de los pipelines de análisis.

c. Plataforma de machine learning en la nube: Azure Machine Learning Studio, utilizada para la implementación de AutoML, gestión de experimentos y comparación con los modelos manuales.
d. Principales bibliotecas:
    - Manipulación y cálculo: pandas (2.3.1), numpy (2.3.2)
    - Visualización: matplotlib (3.10.3), seaborn (0.13.2)
    - Machine Learning:
        - scikit-learn (1.7.1): pipelines, regresión Ridge, Random Forest, validación cruzada
        - xgboost (3.0.3): gradient boosting optimizado
    - Preprocesamiento: scipy (1.16.0) para transformaciones estadísticas
    - Azure ML: azureml-automl-core (1.60.0), azureml-train-automl-client (1.60.0)
e. Reproducibilidad:
    - Dependencias: Archivo requirements.txt con versiones fijas de todas las librerías utilizadas (ver Anexo 1)
    - Semilla aleatoria: Fijada en random_state=42 para garantizar reproducibilidad en:
        - Particiones train/test (train_test_split)
        - Modelos scikit-learn (Ridge, Random Forest)
        - Modelo XGBoost
        - Búsqueda de hiperparámetros (GridSearchCV, RandomizedSearchCV)
        - Experimento Azure AutoML (configurado en el experimento)
        - Configuración Azure AutoML: Documentada en archivo jobDefinition.yaml (ver Anexo 2)

9. Consideraciones éticas

a. Uso de datos: los datos de calidad de agua son públicos y no incluyen información personal.
b. Transparencia: documentar versiones de librerías, parámetros y pasos de preprocesamiento.
c. Interpretación responsable: evitar conclusiones extrapoladas fuera del rango de los datos.
d. Reproducibilidad: compartir código y pipelines para auditoría y validación externa.

VII. Resultados

En esta sección se presentan los principales hallazgos obtenidos tras la aplicación de los modelos de aprendizaje automático propuestos.

Del preprocesamiento y filtrado de los datos originales, se generó un dataset depurado que contiene variables ambientales predictoras relacionadas con el crecimiento algal. Debido a la extensión y gran cantidad de registros del dataset, una muestra representativa de 10 registros se incluye en el Anexo 3 para referencia.

Asimismo, con el fin de mantener una exposición clara y concisa de los resultados, se hará referencia al código previamente descrito en la sección de metodología, incorporando únicamente los fragmentos nuevos cuando sea necesario.

1. Resultados del preprocesamiento de los datos y el dataset final para el modelado

Los principales resultados del proceso de limpieza y preparación del dataset se resumen a continuación.

1.1 Carga y consolidación de datasets crudos

Se consolidaron tres archivos CSV correspondientes a diferentes periodos temporales, el código utilizado se encuentra en el punto 3.1.2 de la metodología.

El resultado obtenido es el siguiente:

```
1. Dataset 2012-2018: 1244 filas x 36 columnas
2. Dataset 2019:      219 filas x 33 columnas
3. Dataset 2020-2021: 413 filas x 33 columnas
4. Total de registros consolidados: 1876
```

En resumen, el dataset consolidado inicial contiene 1,876 registros (filas) correspondientes al periodo 2012-2021, con 36 variables ambientales (35 son predictoras y 1 es la variable objetivo, clorofila).

1.2 Eliminación de variables con datos faltantes excesivos

Se aplicó el criterio del 50%: variables con más de la mitad de sus registros incompletos fueron eliminadas por considerarse poco confiables para el modelado (referirse al punto 3.1.3 de la metodología).

```
Resultado:

Variables antes: 38

Variables después: 30

Variables eliminadas: 8
```

Se eliminaron 8 variables con más del 50% de datos faltantes (incluyendo dmicro con 50.7% faltante). Cabe mencionar, que se visualizan 2 variables más (38), ya que, se toma tanto el día como el año del rubro fecha (date). Ya que, más adelante se utilizan para la creación de variables adicionales cuyo objetivo es capturar patrones estacionales.

1.3 Imputación de valores faltantes

Las variables con menos del 50 % de datos faltantes se conservaron y se imputaron mediante el método K-Nearest Neighbors (KNN). Asimismo, se descartaron todas las muestras que carecían de datos en la variable objetivo (clorofila), dado que, este valor es indispensable para entrenar los modelos predictivos.

El código empleado a lo largo del proceso se detalla en la sección 3.2.3 de la metodología para su consulta. No obstante, se detallan fragmentos de código adicionales agregados posteriormente:

```
Resultado:

Valores faltantes después de imputación: 0

Se verificó la presencia de valores faltantes en la variable objetivo
(clorofila-a), descartando muestras sin esta información.

# Eliminar filas con valores faltantes en clorofila

registros_antes = len(df_clean)

df_clean = df_clean.dropna(subset=['chl'])

registros_despues = len(df_clean)

print(f"Registros antes: {registros_antes}")

print(f"Registros después: {registros_despues}")

print(f"Registros eliminados: {registros_antes - registros_despues}")

Resultado:

Registros antes: 1876

Registros después: 1876
```

```
Registros eliminados: 0
```

Tras la imputación KNN, no fue necesario eliminar registros por valores faltantes en clorofila, conservando los 1,876 registros completos (100% del total inicial).

Una vez finalizado este proceso, se aplicó un criterio de causalidad ambiental, excluyendo variables biológicas indicadoras que representan efectos de las floraciones algales (no causas) para evitar data leakage (sección 3.4.1 de la metodología).

Posteriormente, se generaron las variables adicionales para capturar patrones estacionales cíclicos y balance nutricional (punto 3.4.2 y 3.4.3 de la metodología).

```
Resultado:

Variables originales: 30

Variables seleccionadas: 21

Variables excluidas: pmicro, dmicro, phyco, ctd_do
```

Se eliminaron variables biológicas como pico-microplancton (pmicro), diatomeas microscópicas (dmicro), ficoeritrina (phyco) y oxígeno disuelto (ctd_do), ya que representan efectos de las floraciones, no causas.

Finalmente, para el dataset final, se seleccionaron las 10 variables predictoras con evidencia causal directa sobre el crecimiento algal (sección 3.4.4 de la metodología).

```
Resultado

Variables predictoras: 10

Variable objetivo: chl

Dimensiones finales: 1876 registros x 11 variables
```

En resumen, estos serían los principales hallazgos del dataset depurado:

- Total de registros válidos: 1,876 muestras (100% del total inicial)
- Variables predictoras: 10 drivers ambientales causales exclusivamente
- Periodo temporal: 2012-2021 (9 años)
- Datos duplicados: 0 filas duplicadas
- Valores faltantes: 0 (imputados con KNN, k=5)

Distribución de variables predictoras:

- Nutrientes (4): Fósforo total (tp), fósforo reactivo soluble (srp), amonio (nh4), nitrato+nitrito (no3_2)

- Parámetros físicos (3): Temperatura (ctd_temp), transparencia Secchi (secchi), profundidad de muestreo (depth_m)
- Estacionalidad (2): Componentes trigonométricos del día del año (sin_doy, cos_doy)
- Balance nutricional (1): Relación fósforo/amonio (tp_nh4_ratio)

La selección científica de variables se basó en principios de causalidad ambiental, eliminando indicadores biológicos que representen efectos de las floraciones algales para evitar *data leakage*. Se priorizó la conservación de variables con evidencia causal directa sobre el crecimiento algal.

Se verificó que los registros conservados mantuvieran una distribución temporal adecuada, abarcando los años 2012 a 2021, con predominio de mediciones entre los meses de mayo y octubre, lo cual asegura representatividad estacional en el análisis.

Por último, los valores estadísticos descriptivos de la variable objetivo (clorofila), luego del preprocesamiento se calcularon de la siguiente manera:

```
# Estadísticos del dataset final procesado

print(df_final['chl'].describe())
```

Obteniéndose los siguientes resultados:

Tabla 2. Valores estadísticos obtenidos de concentración de clorofila.

Medidas estadísticas	Valor obtenido
Media	24.90 µg/L
Mediana	16.05 µg/L
Desviación estándar	23.57 µg/L
Mínimo	0.68 µg/L
Máximo	79.34 µg/L

Fuente: elaboración propia.

Según los valores obtenidos, las concentraciones de clorofila presentan una distribución moderadamente asimétrica (media = 24.90 µg/L, mediana = 16.05 µg/L), lo que sugiere la presencia de valores elevados asociados a eventos de floraciones algales, aunque menos extremos que los observados en análisis previos. Esta asimetría sugiere que la mayoría de las observaciones corresponden a condiciones de productividad moderada, intercaladas con aumentos puntuales de crecimiento algal que elevan la media, pero

mantienen la mediana en niveles relativamente bajos, debido a su ocurrencia esporádica (Paerl & Otten, 2016).

Estudios previos en lagos y embalses han reportado comportamientos similares. Por ejemplo, Fang et al. (2021) y Maguire et al. (2024) observaron distribuciones sesgadas de clorofila en sistemas donde las floraciones algales se comportan como eventos extremos estacionales, concentrados en periodos de alta temperatura y disponibilidad de nutrientes. Asimismo, Shi et al. (2019) subraya que esta naturaleza "no normal" de los datos representa un desafío estadístico y justifica el uso de modelos y métodos de regularización (como Ridge o Random Forest) capaces de manejar variabilidad elevada sin distorsionar la interpretación ecológica.

La desviación estándar (23.57 µg/L) refleja una variabilidad considerable dentro del sistema, pero dentro de un rango más acotado que el observado en estudios donde las floraciones alcanzan concentraciones superiores a 100 µg/L. Esto sugiere un comportamiento más estable del ecosistema, aunque con fluctuaciones naturales vinculadas a cambios en la temperatura, los nutrientes, la luz solar, etc y refleja una vez más la dinámica heterogénea del sistema acuático (Fang et al., 2021).

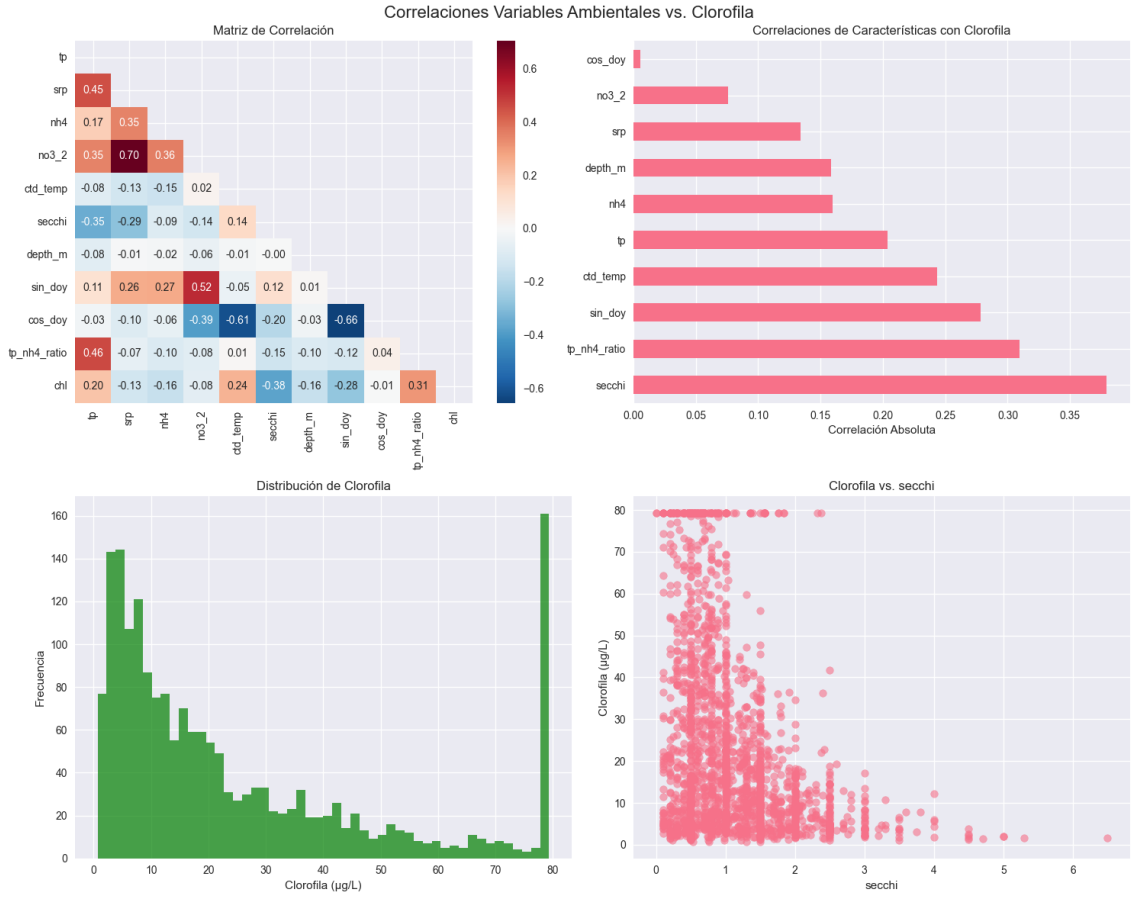
El rango de valores (0.68 a 79.34 µg/L) confirma la coexistencia de periodos con baja actividad biológica y otros de mayor productividad algal. Las concentraciones superiores a 50 µg/L pueden considerarse indicativas de floraciones incipientes, en línea con lo reportado en estudios de lagos templados (Maguire et al., 2024). Esta variabilidad es inherente a sistemas donde las floraciones se comportan como eventos discontinuos y de corta duración, alterando temporalmente la composición química y biológica del agua (Shi et al., 2019).

2. Correlación entre variables ambientales y clorofila

Para identificar las relaciones lineales entre las variables fisicoquímicas y la concentración de clorofila, se calculó una matriz de correlación de Pearson utilizando el conjunto de datos procesado (n=1,876 registros). Este análisis exploratorio permite determinar cuáles variables ambientales presentan mayor asociación con la biomasa algal antes del entrenamiento de los modelos predictivos. El código desarrollado para tal fin se encuentra disponible en el Anexo 5.

De ahí, se generó la siguiente figura que muestra la matriz de correlación completa entre las 10 variables predictoras y la variable objetivo. Los valores cercanos a 1 o -1 indican relaciones lineales fuertes (positivas o negativas, respectivamente), mientras que valores cercanos a 0 sugieren ausencia de asociación lineal.

Figura 1. Matriz de correlación entre variables ambientales vs clorofila.



La figura 1 muestra el análisis de correlaciones entre variables ambientales y concentración de clorofila. En la parte superior izquierda se muestran las relaciones entre todas las variables. La imagen superior derecha, son las correlaciones absolutas ordenadas de mayor a menor. La sección inferior derecha señala la relación entre transparencia del agua (secchi) y clorofila, evidenciando la correlación más fuerte del análisis.

Tabla 3. Correlaciones de Pearson entre variables ambientales y concentración de clorofila (ordenadas por magnitud)

Variable	Correlación (r)	Magnitud
secchi	-0.379	0.379
tp_nh4_ratio	+0.309	0.309
sin_doy	-0.278	0.278
ctd_temp	+0.244	0.244

Variable	Correlación (r)	Magnitud
tp	+0.204	0.204
nh4	-0.160	0.160
depth_m	-0.159	0.159
srp	-0.134	0.134
no3_2	-0.076	0.076
cos_doy	-0.006	0.006

Fuente: elaboración propia

Como se observa, la matriz de correlación muestra las relaciones clave entre las variables fisicoquímicas y la concentración de clorofila. El índice de transparencia del agua (Secchi) presenta una correlación negativa moderada (r = -0.35) con la clorofila, lo que indica que a medida que aumenta la biomasa algal, disminuye la transparencia del agua. Este comportamiento es consistente con lo documentado por Paerl & Otten (2016), quienes señalan que la proliferación de fitoplancton reduce la penetración lumínica, incrementando la turbidez del sistema.

Por otro lado, el índice de nutrientes refleja la dependencia directa de la clorofila respecto a la disponibilidad de nutrientes. Este patrón coincide con lo descrito por Fang et al. (2021), quienes encontraron que concentraciones elevadas de fósforo y nitrógeno son los principales impulsores de las floraciones algales nocivas en sistemas eutróficos.

Las variables estacionales derivadas del día del año (sin_doy y cos_doy) también presentan correlaciones positivas leves, lo que sugiere que los picos de clorofila están asociados a periodos del año con condiciones favorables para el crecimiento algal, principalmente durante las estaciones cálidas (Maguire et al., 2024).

La correlación negativa de profundidad (depth_m, r = -0.24) indica que las concentraciones más altas de clorofila se registran en zonas someras, donde la luz y los nutrientes están más disponibles. Este patrón espacial es típico de ecosistemas como lagos y embalses estratificados (Shi et al., 2019).

En cuanto a la distribución de clorofila, el histograma muestra una fuerte asimetría hacia la izquierda, con predominio de valores bajos y pocos eventos de alta concentración.

Esta tendencia está en concordancia con la naturaleza episódica de las floraciones algales.

Finalmente, el gráfico de dispersión Clorofila vs. Secchi reafirma la relación inversa entre ambas variables: las mayores concentraciones de clorofila se asocian con profundidades de disco Secchi menores a 2 metros, lo que coincide con los umbrales indicativos de eutrofización avanzada en sistemas con poco movimiento de agua (OECD, 1982).

3. Resultados del modelado predictivo

Se desarrollaron cuatro enfoques distintos de modelado con el objetivo de evaluar tanto la capacidad predictiva como la validez científica de los mismos, esto con el fin de medir de manera más realista la capacidad de generalización de los modelos en contextos ambientales con alta variabilidad.

La selección de los algoritmos se basó principalmente en las características del dataset, las funciones y características de cada modelo descritas previamente, así como su uso en otros estudios científicos ambientales. A continuación, se resumen brevemente los criterios para la selección final:

i. Ridge Regression: se utilizó como modelo lineal de referencia (baseline), permitiendo establecer una línea base de comparación y evaluar la existencia de relaciones lineales entre las variables fisicoquímicas y la concentración de clorofila.
ii. Random Forest: se seleccionó por su fortaleza frente al ruido y los valores atípicos, características comunes en los datos ambientales. Este algoritmo maneja de forma eficiente tanto variables numéricas como categóricas sin requerir normalización, lo que lo hace adecuado para datasets ecológicos heterogéneos.
iii. XGBoost: se incorporó por su capacidad para modelar relaciones no lineales y capturar interacciones complejas entre las variables ambientales. Su desempeño demostrado en múltiples estudios y competencias de ciencia de datos, lo convierte en una herramienta eficaz para representar dinámicas ecológicas donde las relaciones entre factores ambientales son altamente interdependientes.
iv. Validación con Azure AutoML: Con el objetivo de garantizar la validez de los resultados, explorar técnicas de *machine learning* avanzadas, así como evaluar de forma objetiva si los modelos manuales alcanzan el rendimiento óptimo posible o si existen oportunidades de mejora, se implementó Azure AutoML como metodología de validación independiente.

Esta plataforma automatiza la selección de algoritmos, optimización de hiperparámetros e ingeniería de características, evaluando sistemáticamente múltiples enfoques de modelado.

3.1 Resultados del modelo Ridge Regression

Este modelo se utilizó como base lineal con regularización L2 y se entrenó utilizando el valor óptimo de alpha determinado mediante RidgeCV con validación cruzada de 5 pliegues.

El código empleado para el cálculo de métricas se encuentra en el apartado 7.1 de la metodología. Adicionalmente, se generaron gráficos correspondientes a los coeficientes de las variables, la comparación entre valores predichos y reales, el análisis de residuos y los puntajes de validación cruzada por pliegue. El código utilizado para estos procedimientos se referencia en el Anexo 6 (a).

Por otro lado, el análisis de los coeficientes estandarizados de la Regresión Ridge reveló la contribución relativa de cada variable ambiental a la predicción de clorofila. El código utilizado se encuentra en el punto 7.4.1 de la metodología, además se complementó:

```
# Interpretar coeficientes

caracteristicas_positivas                                                    =
importancia_caracteristicas[importancia_caracteristicas['coeficiente'] > 0]

caracteristicas_negativas                                                    =
importancia_caracteristicas[importancia_caracteristicas['coeficiente'] < 0]
```

Tabla 4. Coeficientes estandarizados del modelo de Regresión Ridge.

Variable	Coeficiente
secchi	-8.0810
sin_doy	-6.8562
srp	-6.4054
cos_doy	-4.1194
tp_nh4_ratio	+3.8262
tp	+3.7112
ctd_temp	+3.0816
no3_2	+2.7367
depth_m	-2.6582
nh4	-1.3857

Fuente: elaboración propia.

Asimismo, los valores de métricas estimados con el dataset de drivers ambientales para este modelo fueron:

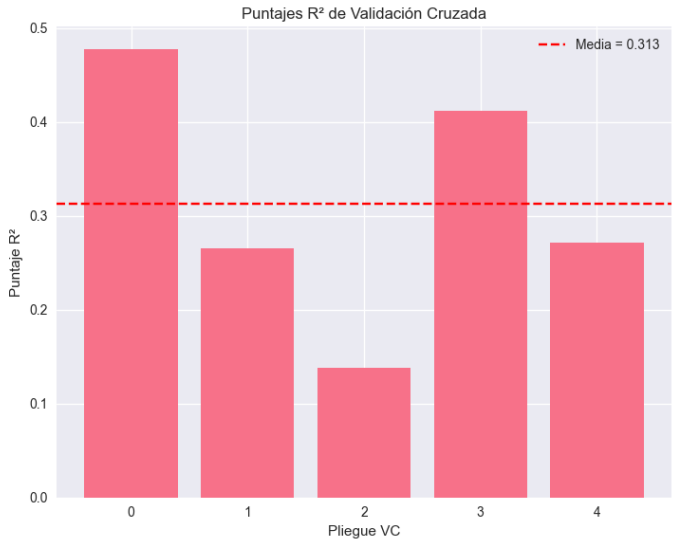
Tabla 5. Métricas de desempeño del modelo de Regresión Ridge

Métrica	Entrenamiento	Prueba
R^2	0.4085	0.3358
MAE (µg/L)	13.717	13.723
RMSE (µg/L)	18.099	19.315
CV R^2	-	0.3128 ± 0.1195

Fuente: elaboración propia.

La validación cruzada de 5 pliegues arrojó un rango de R^2 de [0.1382, 0.4773], indicando variabilidad moderada en el desempeño según la partición de datos.

Gráfico 1. Puntajes R^2 de validación cruzada para Regresión Ridge



Fuente: elaboración propia.

Según los resultados obtenidos, el modelo Ridge Regression mostró un desempeño moderado. El valor de R^2 en validación cruzada (CV R^2) fue de 31.3 % ± 11.9 %, lo que evidencia una variabilidad moderada entre los pliegues y una capacidad de generalización media.

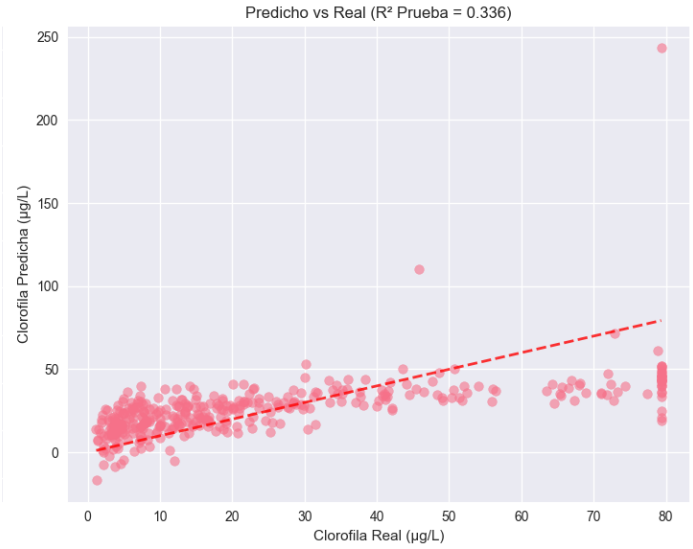
Este nivel de ajuste es consistente con hallazgos en otros estudios que emplean modelos lineales regularizados para predecir concentraciones de clorofila. Por ejemplo, Shi et al. (2019) y Fang et al. (2021) reportaron valores de R^2 entre 0.25 y 0.45 en ecosistemas acuáticos con alta variabilidad temporal. Del mismo modo, Maguire et al. (2024) observaron desempeños similares al modelar la fenología de clorofila en el lago Erie.

En investigaciones como las mencionadas anteriormente, Fang et al. (2021) describe valores de R^2 entre 0.35 y 0.45 al modelar floraciones algales en lagos eutróficos, mientras que Maguire et al. (2024) encontró desempeños comparables. Este nivel de

ajuste se considera aceptable para este modelo según lo observado en estudios ambientales similares (Paerl & Otten, 2016).

Esto se corresponde en el gráfico de dispersión, donde se observa que el modelo logra capturar parcialmente las tendencias generales, pero subestima los valores altos de clorofila, un comportamiento común en sistemas ecológicos con alta heterogeneidad y presencia de valores extremos (Fang et al., 2021).

Gráfico 2. Correlación entre valores observados vs predichos para Ridge Regression



Fuente: elaboración propia.

La pendiente de la recta ajustada confirma una tendencia de sub-predicción en concentraciones elevadas, lo que puede atribuirse a la naturaleza lineal del modelo y a la complejidad no lineal inherente a los factores que regulan las floraciones algales (Paerl & Paul, 2012).

En cuanto a las métricas de error, el MAE fue de 13.72 µg/L y el RMSE de 19.32 µg/L, valores que reflejan una diferencia promedio moderada entre las concentraciones observadas y las predichas. Si bien el modelo no alcanza un ajuste perfecto, su comportamiento resulta consistente con otras investigaciones, donde los métodos de regresión regularizada mostraron un rendimiento estable en contextos de alta correlación entre variables ambientales (Shi et al., 2019).

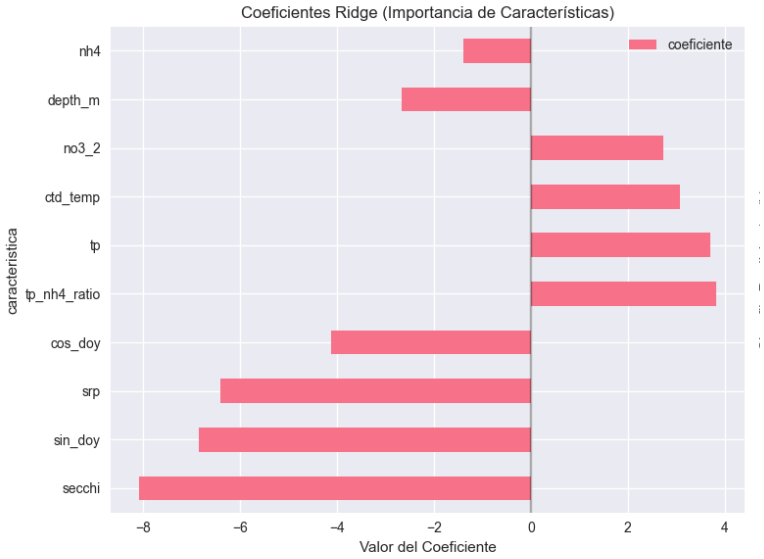
Por tanto, aunque Ridge Regression proporciona estabilidad y evita el sobreajuste, su estructura lineal puede limitar su capacidad para capturar relaciones no lineales entre los factores ambientales y la respuesta biológica, por lo que su rendimiento no es ideal.

El análisis de importancias del modelo Ridge Regression evidencia que las variables secchi, sin_doy y srp presentan los coeficientes negativos más altos, lo que indica una

relación inversa con la concentración de clorofila, mientras que tp, ctd_temp y tp_nh4_ratio muestran asociaciones positivas.

Estos resultados están alineados a los hallazgos previos donde se indica que la transparencia del agua y la estacionalidad influyen significativamente en la variación de la biomasa algal.

Gráfico 3. Importancia relativa de las variables ambientales para Ridge Regression



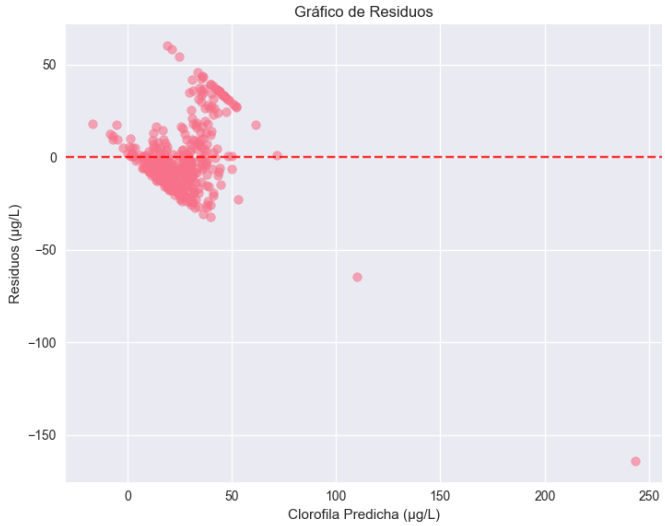
Fuente: elaboración propia.

Por otro lado, el análisis de residuales es fundamental para validar las asunciones del modelo e identificar patrones no capturados.

En el caso de Ridge Regression, el gráfico 4 de residuos evidencia patrones sistemáticos, con una tendencia a subestimar valores altos de clorofila (>50 µg/L) y a sobreestimar valores bajos, reflejando la incapacidad del modelo lineal para capturar la complejidad no lineal del sistema, como se ha mencionado anteriormente.

Además, el incremento en la dispersión de los residuos a concentraciones elevadas indica una variabilidad no constante, lo que reduce la fiabilidad del modelo en condiciones asociadas a floraciones algales de alta intensidad.

Gráfico 4. Distribución de residuales para Ridge Regression



Fuente: elaboración propia.

3.2 Resultados del modelo Random Forest

El modelo Random Forest se optimizó mediante búsqueda aleatoria de hiperparámetros con 50 iteraciones y validación cruzada de 5 pliegues. El espacio de búsqueda incluyó número de árboles, profundidad máxima, criterios de división y selección de características. El procedimiento utilizado para la optimización de hiperparámetros se detalla en el Anexo 4.

Los hiperparámetros óptimos identificados fueron: n_estimators=200, max_depth=30, min_samples_split=2, min_samples_leaf=1, y max_features='log2', logrando un R^2 de validación cruzada de 0.6955 durante la búsqueda.

Unido a esto, el código utilizado para el cálculo de métricas se menciona en el punto 7.1 de la metodología, junto con lo siguiente:

```
# Métricas adicionales con validación cruzada
resultados_vc = cross_validate(
    mejor_rf, X, y, cv=5,
    scoring=['r2', 'neg_mean_absolute_error', 'neg_root_mean_squared_error']
)
```

Por otro lado, el código utilizado para la generación de visualizaciones de este modelo se adjunta en el Anexo 6 (b).

En lo referente a la importancia de variables, Random Forest permite evaluar este rubro mediante múltiples enfoques. Se utilizaron dos métodos complementarios: importancia

basada en Gini (reducción de impureza) e importancia por permutación (degradación de desempeño). El código utilizado para calcular la importancia de características puede ser revisado en el punto 7.4.1 de la metodología, con el agregado:

```
# Combinar ambas medidas de importancia
importancia_combinada = importancia_gini.merge(importancia_perm_df, on='caracteristica')
importancia_combinada['importancia_promedio'] = (
    importancia_combinada['importancia_gini'] + importancia_combinada['importancia_perm']
) / 2
importancia_combinada = importancia_combinada.sort_values('importancia_promedio', ascending=False)
```

Tabla 6. Importancia de características del modelo Random Forest según métodos Gini y permutación

Variable	Importancia Gini	Importancia Permutación
tp_nh4_ratio	0.2355	0.2169 ± 0.0088
tp	0.1646	0.2589 ± 0.0277
sin_doy	0.0844	0.1224 ± 0.0071
nh4	0.1069	0.0670 ± 0.0064
secchi	0.0898	0.0613 ± 0.0068
srp	0.0779	0.0879 ± 0.0119
ctd_temp	0.0764	0.0593 ± 0.0072
no3_2	0.0608	0.0269 ± 0.0060
cos_doy	0.0598	0.0488 ± 0.0067
depth_m	0.0438	0.0314 ± 0.0038

Fuente: elaboración propia

El modelo Random Forest identificó tp_nh4_ratio y tp como los predictores más importantes según ambos métodos, confirmando el rol central del fósforo en la dinámica de clorofila. La importancia por permutación mostró mayor consistencia (desviaciones estándar bajas), validando la certeza de las predicciones. La divergencia entre métodos Gini y permutación para nh4 (0.1069 vs 0.0670) sugiere posibles interacciones no lineales capturadas por el método de permutación.

Ahora bien, en referencia a la métricas, para el modelo Random Forest se obtuvieron los siguientes valores:

Tabla 7. Métricas de desempeño del modelo Random Forest

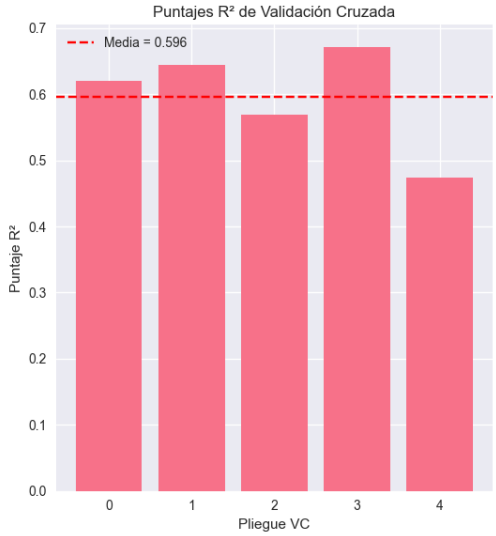
Métrica	Entrenamiento	Prueba
R^2	0.9585	0.7835

MAE (µg/L)	3.361	7.927
RMSE (µg/L)	4.791	11.028
CV R^2	-	0.5958 ± 0.0698

Fuente: elaboración propia

La validación cruzada de 5 pliegues arrojó un rango de R^2 de [0.4734, 0.6713]. Las métricas adicionales de validación cruzada fueron MAE VC: 10.579 ± 1.710 µg/L y RMSE VC: 14.439 ± 1.458 µg/L. La diferencia entre R^2 de entrenamiento (0.9585) y prueba (0.7835) sugiere sobreajuste moderado, característico de modelos basados en árboles.

Gráfico 5. Puntajes R^2 de validación cruzada para Random Forest



Fuente: elaboración propia.

Según los resultados obtenidos, en la validación cruzada el R^2 promedio fue de 59.6 % ± 7.0 %, lo que evidencia una mayor capacidad de generalización respecto al modelo lineal, con un comportamiento estable entre pliegues. En el conjunto de prueba, el modelo alcanzó un R^2 de 0.7835, con un MAE de 7.93 µg/L y un RMSE de 11.03 µg/L, reflejando un error medio bajo y una alta correspondencia entre las concentraciones observadas y las predichas.

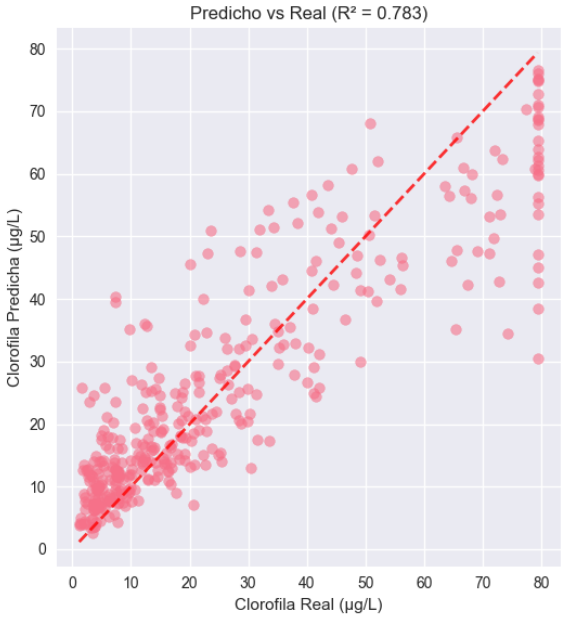
Estos valores se encuentran dentro del rango reportado en investigaciones similares. Según, Fang et al. (2021) aplicaron Random Forest, obteniendo valores de R^2 entre 0.60 y 0.70, señalando que el modelo superó a las regresiones tradicionales en precisión y estabilidad. De forma similar, Qian et al. (2021) reportó un R^2 de 0.63 al predecir concentraciones de clorofila en lagos del norte de Estados Unidos.

Asimismo, Maguire et al. (2024) indica que Random Forest capturó los patrones estacionales de clorofila en el lago Erie, explicando más del 60 % de la variabilidad observada en los datos (equivalente a un coeficiente de determinación (R^2) de aproximadamente 0.60) y demostrando que los modelos basados en árboles de

decisión son particularmente efectivos en contextos donde las floraciones algales responden a interacciones complejas entre temperatura, turbidez y nutrientes.

En concordancia con estos estudios, el modelo implementado en esta investigación alcanzó un desempeño similar.

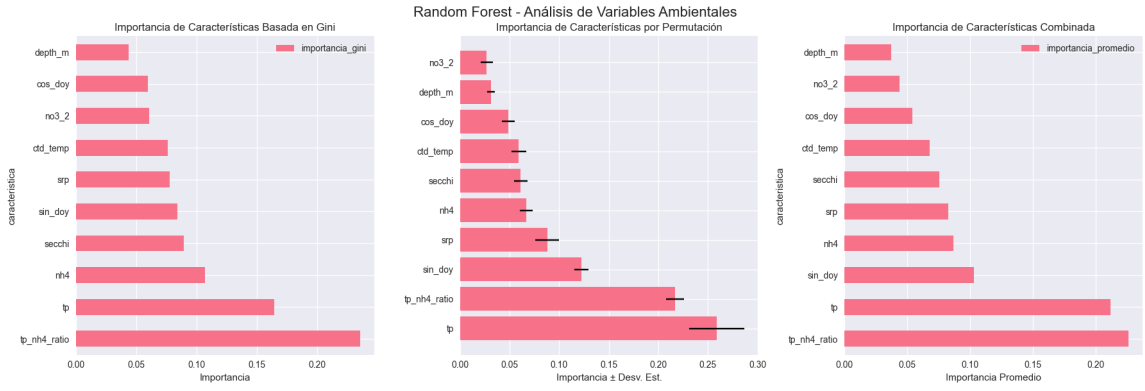
Gráfico 6. Correlación entre valores observados vs predichos para Random Forest



Fuente: elaboración propia.

A partir del gráfico 6, que muestra la importancia de las características en el modelo Random Forest, se observa que el fósforo total (tp) y la relación fósforo-nitrógeno (tp_nh4_ratio) son los predictores más influyentes en la estimación de la concentración de clorofila, seguidas de la temperatura del agua (ctd_temp) y la profundidad (depth_m).

Gráfico 7. Importancia relativa de las variables ambientales para Random Forest



Fuente: elaboración propia.

Por otro lado, variables como nitrato (no3_2) y transparencia del agua (secchi) mostraron menor importancia relativa, lo cual concuerda con estudios que señalan que, en condiciones de alta eutrofización, el control de la biomasa algal depende más del fósforo que del nitrógeno (Smith & Schindler, 2009).

Gráfico 8. Distribución de residuales para Random Forest

Fuente: elaboración propia.

En el caso del modelo Random Forest, los residuos se distribuyen de manera más aleatoria alrededor de cero, lo que refleja una mejora respecto al patrón sistemático observado en Ridge Regression. La mayor capacidad explicativa del modelo (R^2 = 0.7835 frente a 0.3358 en Ridge) se traduce en errores más balanceados y centrados, aunque aún se observan algunos valores atípicos en concentraciones altas.

3.3 Resultados del modelo XGBoost

Se optimizó mediante búsqueda aleatoria de hiperparámetros con 150 iteraciones y validación cruzada de 5 pliegues, enfocándose en la reducción de sobreajuste mediante regularización agresiva L1 y L2, control de profundidad de árboles y parámetros de poda. Los pasos para la optimización de hiperparámetros se encuentran Anexo 4.

Los hiperparámetros óptimos identificados fueron: n_estimators=300, max_depth=5, learning_rate=0.05, subsample=0.8, colsample_bytree=0.7, min_child_weight=3, gamma=0.2, reg_alpha=1, y reg_lambda=1.5.

El código utilizado para el cálculo de métricas se encuentra en la sección 7.1 de la metodología, junto con lo siguiente:

```
# Crear modelo sin parada temprana para validación cruzada
```

```
xgb_vc = xgb.XGBRegressor(**mejores_parametros, random_state=42, n_jobs=-1,
tree_method='hist')

# Métricas adicionales con validación cruzada

resultados_vc = cross_validate(

        xgb_vc, X, y, cv=5,
        scoring=['r2',                                  'neg_mean_absolute_error',
    'neg_root_mean_squared_error']

)
```

Unido a esto, para este modelo el código utilizado para la generación de los gráficos fue se haya en el Anexo 6 (c) para su consulta.

En en caso de la importancia de variables para XGBoost, este proporciona múltiples métricas de importancia de características: ganancia de información (reducción de error), peso (frecuencia de uso en divisiones) y cobertura (número de muestras afectadas). El código utilizado para calcular la importancia de características se encuentra en la sección 7.4.1 de la metodología, además se adicionó las siguientes líneas de código :

```
# Convertir a DataFrame

df_importancia = pd.DataFrame(diccionario_importancia).fillna(0)

df_importancia = df_importancia.reindex(columnas_caracteristicas).fillna(0)
```

Tabla 8. Importancia de características del modelo XGBoost.

Variable	Ganancia	Peso	Cobertura
tp_nh4_ratio	0.2530	624	280.21
depth_m	0.1396	232	435.57
tp	0.1282	884	246.69
srp	0.0832	826	258.95
cos_doy	0.0813	452	288.21
nh4	0.0773	644	243.52
sin_doy	0.0749	665	245.05
ctd_temp	0.0585	841	240.35
secchi	0.0566	561	281.51
no3_2	0.0474	772	276.90

Fuente: elaboración propia.

El modelo XGBoost identificó tp_nh4_ratio como el predictor dominante (25.3% de ganancia total), seguido por depth_m (13.96%) y tp (12.82%). La métrica de peso revela que tp, ctd_temp y srp son las variables más frecuentemente utilizadas en divisiones de árboles (884, 841 y 826 usos respectivamente), mientras que la cobertura muestra que depth_m afecta el mayor número de muestras (435.57). La divergencia entre métricas

sugiere que tp_nh4_ratio genera las divisiones más informativas, mientras que variables como ctd_temp requieren múltiples divisiones para capturar efectos no lineales.

Ahora bien, los resultados obtenidos en las métricas en este modelo para la variable clorofila, fueron:

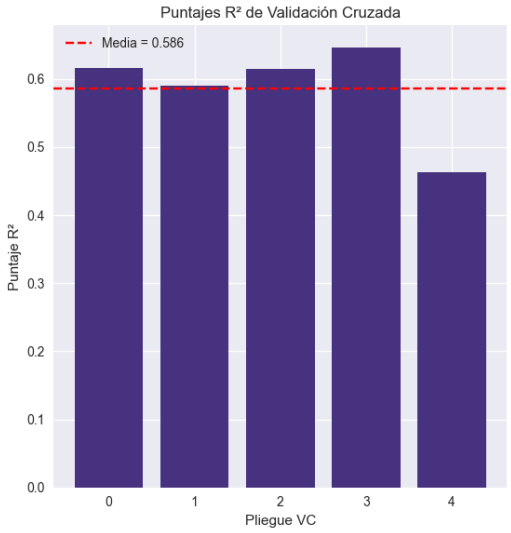
Tabla 9. Métricas de desempeño del modelo XGBoost

Métrica	Entrenamiento	Prueba
R^2	0.9326	0.7812
MAE (µg/L)	4.298	7.826
RMSE (µg/L)	6.111	11.086
CV R^2	-	0.5859 ± 0.0641

Fuente: elaboración propia

La validación cruzada de 5 pliegues arrojó un rango de R^2 de [0.4626, 0.6459]. Las métricas adicionales de validación cruzada fueron MAE VC: 10.830 ± 1.628 µg/L y RMSE VC: 14.612 ± 1.365 µg/L. La diferencia entre R^2 de entrenamiento (0.9326) y prueba (0.7812) indica sobreajuste moderado (15.1%), similar al observado en Random Forest, sugiriendo que los datos presentan patrones complejos difíciles de generalizar completamente.

Gráfico 9. Puntajes R^2 de validación cruzada para XGBoost

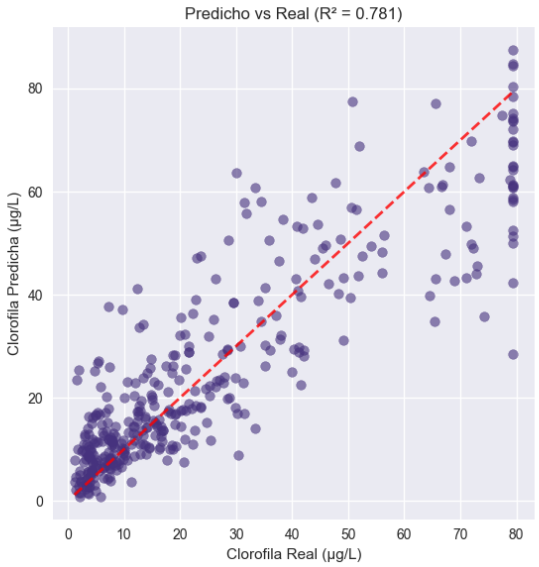


Fuente: elaboración propia.

Como se observa, en la validación cruzada el modelo XGBoost alcanzó un R^2 promedio de 58.6 % ± 6.4 %, reflejando una variabilidad relativamente baja entre pliegues y una buena estabilidad general. En el conjunto de prueba, el modelo obtuvo un R^2 de 0.7812, con un MAE de 7.83 µg/L y un RMSE de 11.09 µg/L, valores que aparentan un buen equilibrio general entre precisión y capacidad de generalización.

Correlacionando los resultados con otras investigaciones, se reportaron valores de R^2 superiores al 0.65 al aplicar XGBoost para estimar la biomasa algal, mostrando mayor eficacia que los modelos previos para capturar interacciones complejas entre nutrientes y factores físicos (Fang et al., 2021). De forma similar, Qian et al. (2021) y Luo et al. (2022) demostraron que este algoritmo ofrece mayor precisión y menor error de predicción que los modelos basados en regresión o árboles individuales, especialmente en datasets ambientales heterogéneos.

Gráfico 10. Correlación entre valores observados vs predichos para XGBoost

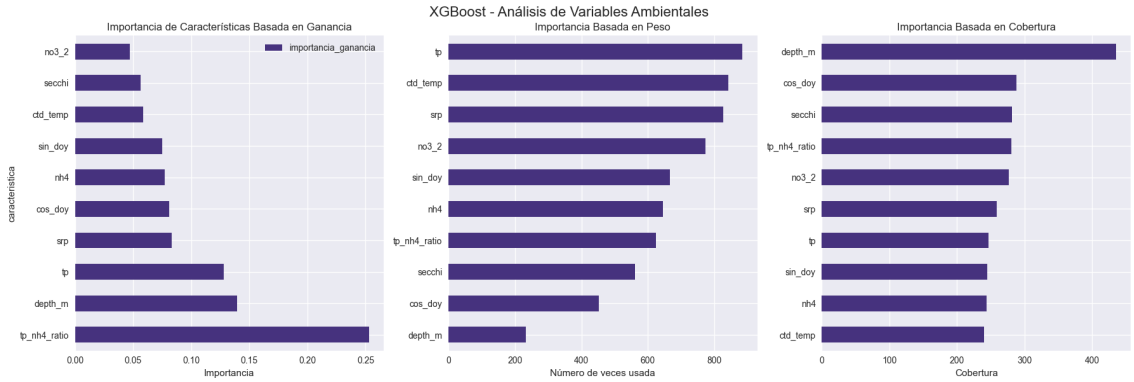


Fuente: elaboración propia.

El gráfico de correlación entre valores observados y predichos muestra una alta concordancia entre ambos conjuntos, con un R^2 de 0.781, lo que señala la capacidad del modelo XGBoost para reproducir las variaciones de la concentración de clorofila. La distribución de los puntos alrededor de la línea de ajuste indica un comportamiento estable, con una leve tendencia a la subestimación en concentraciones elevadas, un patrón común en modelos aplicados a ecosistemas acuáticos con alta variabilidad (Fang et al., 2021).

El análisis de importancia de características revela que las variables con mayor influencia sobre la concentración de clorofila son el fósforo total (tp), la relación fósforo-nitrógeno (tp_nh4_ratio) y la profundidad del agua (depth_m), seguidas por los componentes estacionales (sin_doy y cos_doy) y la temperatura (ctd_temp). En los tres métodos de evaluación, estas variables se mantienen entre las de mayor peso relativo, lo que confirma su importancia en la dinámica de las floraciones algales.

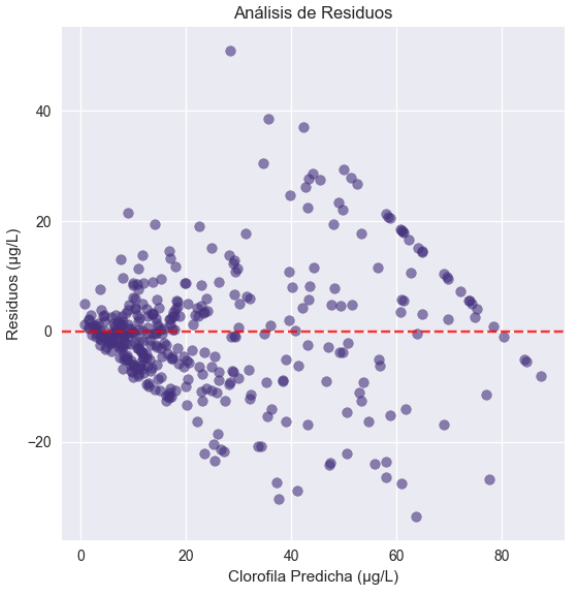
Gráfico 11. Importancia relativa de las variables ambientales para XGBoost



Fuente: elaboración propia.

Paralelamente, en el modelo XGBoost, los residuos se distribuyen de manera más equilibrada alrededor de cero, con menor varianza en comparación con Ridge y Random Forest, y sin patrones sistemáticos evidentes. Esto indica que posee una mejor capacidad predictiva, aunque aún se observan algunos valores atípicos en concentraciones altas.

Gráfico 12. Distribución de residuales para XGBoost



Fuente: elaboración propia.

3.4 Resultados de Azure AutoML (Automated Machine Learning)

Azure AutoML ejecutó un experimento de regresión supervisada con el objetivo de estimar la concentración de clorofila a partir de variables ambientales, optimizando la métrica principal R^2.

Se aplicó una división 80–20 entre entrenamiento y prueba, reservando el 20% del total de registros (2019–2021) para validación final externa.

Durante la fase de entrenamiento, AutoML utilizó validación cruzada K-fold (K=5) en el conjunto de entrenamiento (2012–2021), para una ejecución adecuada del desempeño sin sobreajuste.

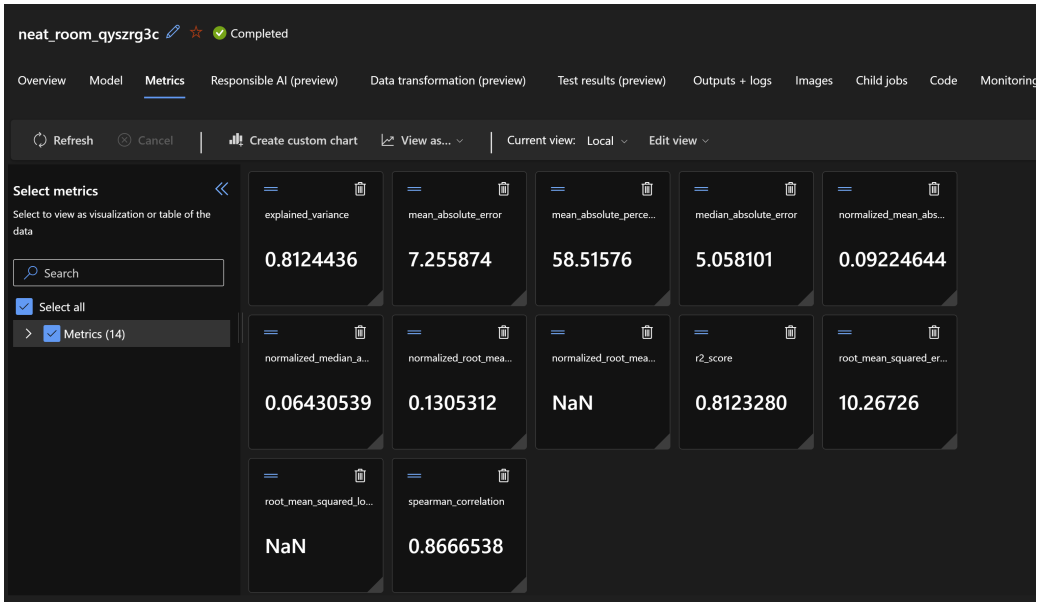
El experimento evaluó 270 combinaciones de modelos e hiperparámetros, incluyendo algoritmos como:

- XGBoostRegressor
- LightGBMRegressor
- RandomForestRegressor
- ElasticNet
- GradientBoosting
- StackEnsemble (modelo final seleccionado)

El proceso automatizado gestionó de forma integrada la limpieza de datos, normalización, ingeniería de características, selección de modelos y optimización de hiperparámetros.

Finalmente, el modelo StackEnsemble fue seleccionado como el mejor desempeño global. La configuración completa del experimento se encuentra en el archivo jobDefinition.yaml, exportado desde el portal de Azure Machine Learning, donde se especifican los parámetros de ejecución, criterios de parada, división de datos y recursos computacionales.

Figura 2. Métricas de desempeño del modelo StackEnsemble, portal Azure ML.



El modelo alcanza un R^2 de 0.81, con errores absolutos bajos y estabilidad consistente en validación cruzada.

Tabla 10. Métricas de desempeño para Azure AutoML (StackEnsemble).

Métrica	Valor obtenido	Unidad	Origen
R^2	0.8123	–	Azure ML Portal
CV R^2	≈0.81 ± 0.02	–	Azure ML Portal
MAE	7.26	µg/L	Azure ML Portal
RMSE	10.27	µg/L	Azure ML Portal
Spearman	0.867	–	Azure ML Portal

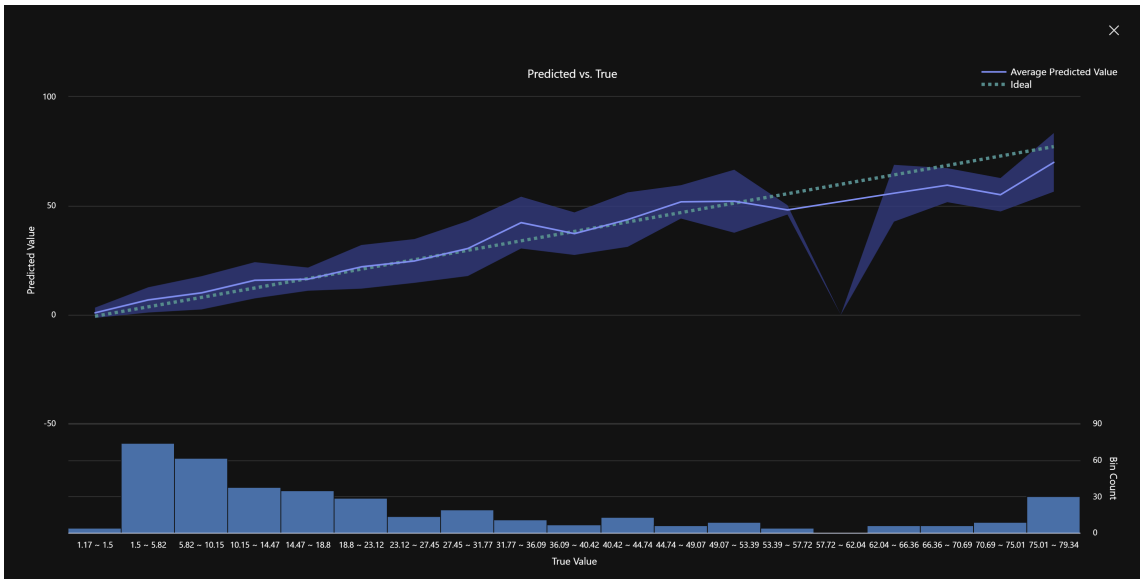
Fuente: elaboración propia.

El proceso de entrenamiento automatizado mediante Azure AutoML seleccionó como mejor modelo un StackEnsemble, el cual combinó los resultados de varios algoritmos base para optimizar simultáneamente la precisión, estabilidad y capacidad de generalización. Este enfoque alcanzó un R^2 de 0.8123, explicando aproximadamente el 81 % de la variabilidad observada en la concentración de clorofila, lo que representa el mejor desempeño entre todos los modelos evaluados. Asimismo, obtuvo un MAE de 7.26 µg/L y un RMSE de 10.27 µg/L, evidenciando un error promedio bajo y un ajuste adecuado.

AutoML aplicó un esquema de validación cruzada K-fold (K = 5) sobre la totalidad del conjunto de entrenamiento (2012–2021) y reservó el 20 % de los datos (2019–2021) para validación externa, garantizando una estimación fiable del rendimiento y reduciendo el riesgo de sobreajuste. Este enfoque ha demostrado ser especialmente efectivo en contextos ambientales, donde la automatización facilita la exploración simultánea de múltiples configuraciones y minimiza el sesgo asociado a la selección manual de modelos (Luo et al., 2022).

De acuerdo con Feurer et al. (2019) y Maguire et al. (2024), los métodos de ensamblaje como StackEnsemble suelen ofrecer el mejor equilibrio entre generalización y precisión, al combinar modelos base heterogéneos (p. ej., árboles de decisión, gradientes y regresiones) que capturan patrones lineales y no lineales. En esta investigación, el modelo de Azure AutoML logró integrar con éxito dichas relaciones complejas, mostrando un CV R^2 de ≈0.81 ± 0.02 y un coeficiente de Spearman de 0.867, lo que respalda su capacidad para preservar el orden de las observaciones en escenarios de alta variabilidad ambiental.

Figura 3. Comparación entre valores predichos y reales del modelo StackEnsemble.



El gráfico de comparación entre valores predichos y reales (Figura 3) confirma la buena precisión y calibración del modelo. La línea azul sólida representa los valores promedio predichos, mientras que la discontinua indica el comportamiento ideal (predicción perfecta).

Las predicciones se mantienen cercanas a la diagonal, evidenciando una representación fiel de la tendencia observada. Por otro lado, se observa una ligera subestimación en los valores superiores a 60 µg/L, fenómeno común en modelos de regresión sobre variables ecológicas con alta heterogeneidad.

El histograma inferior muestra la distribución de valores reales, con predominancia de concentraciones bajas y medias (1–25 µg/L), lo que explica la mayor precisión del modelo en ese rango.

En referencia a la importancia de variables del modelo, se obtuvo a partir de los archivos interpret.json generados por Azure AutoML (ruta explanation/2172c0a5/).

Las variables más influyentes se listan a continuación:

Tabla 11. Importancia de características del modelo Azure AutoML (StackEnsemble)

Variable	Peso relativo
tp_nh4_ratio	9.05
tp	6.93
sin_doy	3.90
srp	3.71
ctd_temp	2.17
no3_2	1.56
secchi	1.51
cos_doy	1.35

depth_m	0.77
nh4	0.60

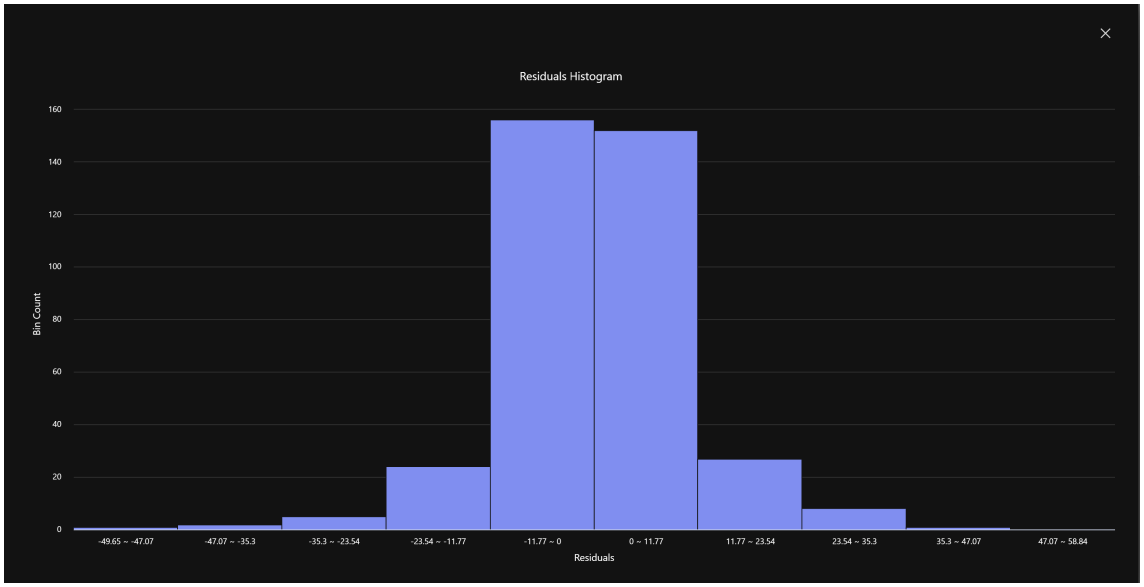
Fuente: elaboración propia.

El análisis de importancia de variables muestra una vez más que la relación fósforo-nitrógeno (tp_nh4_ratio) y el fósforo total (tp) son los predictores más influyentes en la estimación de la concentración de clorofila, con pesos relativos de 9.05 y 6.93, respectivamente.

Asimismo, las variables sin_doy, srp y ctd_temp mantienen una contribución significativa, lo que sugiere que los factores estacionales y térmicos también modulan la respuesta biológica del sistema. En contraste, variables como depth_m y nh4 presentan menor peso, indicando un efecto secundario dentro del conjunto de predictores, coherente con investigaciones que destacan la jerarquía de influencia entre nutrientes y condiciones físicas en ecosistemas acuáticos (Fang et al., 2021).

Respecto al análisis de residuales:

Figura 4. Histograma de residuos del modelo StackEnsemble.



El histograma de residuos (Figura 4) muestra una distribución centrada en cero, con mayor densidad entre −12 y +12 µg/L y sin asimetrías marcadas.

Esto indica que el modelo no presenta sesgos direccionales (ni sobreestimación ni subestimación sistemática), manteniendo un equilibrio adecuado en la estimación de clorofila tanto en valores bajos como altos. La dispersión en los extremos es reducida, lo que refleja una variabilidad mínima y un buen desempeño incluso en escenarios de floración.

A continuación se comparan los resultados referentes al rendimiento de los modelos:

Tabla 12. Comparación del rendimiento de los modelos predictivos

Métrica	Ridge Regression	Random Forest	XGBoost	Azure AutoML (StackEnsemble)
R^2 Entrenamiento (%)	40.85	95.85	93.26	N/A
R^2 Prueba (%)	33.58	78.35	78.12	81.23
CV R^2 (%)	31.28 ± 11.95	59.58 ± 6.98	58.59 ± 6.41	81.00 ± 0.02
MAE Prueba (µg/L)	13.723	7.927	7.826	7.26
RMSE Prueba (µg/L)	19.315	11.028	11.086	10.27

Fuente: elaboración propia

Los resultados muestran un incremento progresivo en la capacidad predictiva conforme se utilizaron modelos más complejos. El modelo Ridge Regression, de naturaleza lineal, obtuvo los valores más bajos, lo que indica que las relaciones entre las variables fisicoquímicas y la concentración de clorofila no pueden ser explicadas adecuadamente mediante un enfoque lineal (Shi et al., 2019).

En contraste, el modelo Random Forest mejoró considerablemente el ajuste, reduciendo además los errores medios. Su mayor desempeño se asocia con la capacidad de los algoritmos basados en árboles para capturar interacciones complejas y manejar variables con ruido o valores atípicos, características comunes en datos ecológicos (Breiman, 2001).

El modelo XGBoost mantuvo un rendimiento similar al de Random Forest en validación cruzada (CV R^2 = 58.6 % ± 6.4 %), aunque alcanzó un mayor poder explicativo en el conjunto de prueba (R^2 = 0.7812). Esto sugiere que los métodos de boosting son más eficientes al combinar secuencialmente árboles débiles para corregir errores acumulativos, lo cual ha sido reportado en otros estudios de predicción ambiental (Chen & Guestrin, 2016).

Finalmente, el modelo Azure AutoML (StackEnsemble) obtuvo el mejor rendimiento global, con un R^2 de 0.8123, MAE de 7.26 µg/L y RMSE de 10.27 µg/L. Este resultado evidencia la capacidad de los enfoques automatizados para optimizar simultáneamente

preprocesamiento, selección de características e hiperparámetros, logrando una mejora de aproximadamente 20% respecto al modelo base lineal.

Este resultado esta alineado con la información obtenida en estudios científicos que señalan que los métodos de aprendizaje de conjunto superan consistentemente a los modelos individuales en tareas de predicción ambiental, al reducir el sobreajuste y mejorar la estabilidad de las estimaciones (Luo et al., 2022).

4. Selección del modelo con mejor rendimiento

Con base en los resultados, Azure AutoML superó significativamente todos los modelos implementados manualmente, alcanzando CV R^2 = 81.00% ± 0.02% y R^2 de prueba = 81.23% con el algoritmo StackEnsemble. Este resultado representa una mejora sustancial en la capacidad predictiva respecto a los modelos tradicionales, destacando además por su buena estabilidad (desviación estándar de solo ±0.02%).

En comparación, XGBoost alcanzó un CV R^2 de 58.59% ± 6.41%, Random Forest obtuvo 59.58% ± 6.98%, y Ridge Regression apenas 31.28% ± 11.95%. En términos relativos, el modelo automatizado mejoró el poder explicativo en 22.4 puntos porcentuales respecto a XGBoost y Random Forest, y 49.7 puntos porcentuales comparado con Ridge Regression. Adicionalmente, la desviación estándar del CV R^2 de Azure AutoML (±0.02%) es 3 veces menor que la de los modelos de árboles (±6-7%) y 6 veces menor que Ridge (±11.95%), lo que confirma su mejor rendimiento y consistencia entre diferentes particiones de datos.

Estas diferencias reflejan la eficacia del enfoque automatizado de AutoML para optimizar simultáneamente la selección de algoritmos, parámetros y combinaciones de modelos, logrando un mejor desempeño con alta estabilidad, sin intervención manual extensa. El modelo también logró reducir los errores de predicción (MAE = 7.26 µg/L, RMSE = 10.27 µg/L), siendo comparables a XGBoost pero con menor variabilidad entre evaluaciones.

Arquitectura del modelo:

El StackEnsemble combina 7 modelos base con pesos optimizados:

4 variantes GradientBoosting (peso total: 0.667)

- Iteración 276 (peso: 0.467): modelo dominante (R^2 individual = 0.7989)
- Iteración 345 (peso: 0.067)
- Iteración 280 (peso: 0.067)
- Iteración 340 (peso: 0.067)

3 variantes RandomForest (peso total: 0.333)

- Iteración 373 (peso: 0.200): segundo más importante

- Iteración 344 (peso: 0.067)
- Iteración 203 (peso: 0.067)

El modelo final de Azure AutoML estuvo compuesto principalmente por algoritmos de Gradient Boosting (66.7 % del peso total), lo que indica que este tipo de aprendizaje, basado en mejorar los errores de modelos anteriores, fue más efectivo que el enfoque paralelo usado por Random Forest. El modelo con mejor desempeño (iteración 276, peso 46.7 %) se seleccionó tras evaluar 270 combinaciones distintas de algoritmos y configuraciones.

Por último, AutoML dio mayor importancia a las variables derivadas, como la relación fósforo-nitrógeno (tp_nh4_ratio) y los factores estacionales (sin_doy y cos_doy), seguidas de las variables de nutrientes directas (tp y srp). Esto muestra que las proporciones entre nutrientes y los cambios a lo largo del año explican mejor las variaciones de clorofila que las concentraciones individuales, en línea con las teorías sobre limitación por nutrientes (Redfield, 1958) y ciclos estacionales de floraciones algales (Paerl & Huisman, 2008)

VII. Conclusiones

El presente trabajo demostró que es posible predecir la concentración de clorofila, como indicador de biomasa algal, mediante algoritmos de aprendizaje automático basados en variables fisicoquímicas y estacionales.

La distribución asimétrica, la alta desviación estándar y la amplia variabilidad del conjunto de datos confirman la heterogeneidad del sistema acuático analizado, donde las floraciones algales ocurren como eventos extremos y no lineales. Estas características sustentan la pertinencia del uso de modelos de *machine learning*, capaces de representar interacciones complejas entre los factores ambientales sin perder precisión estadística.

La implementación de una metodología dual, que combinó modelos desarrollados manualmente (Ridge, Random Forest y XGBoost) con un enfoque automatizado (Azure AutoML), permitió comparar la capacidad predictiva, la estabilidad y la validez ecológica de distintos enfoques. Esta comparación evidenció un incremento progresivo en el rendimiento conforme aumentó la complejidad algorítmica: Ridge Regression: R^2 = 0.3358, Random Forest: R^2 = 0.7835, XGBoost: R^2 = 0.7812 y Azure AutoML (StackEnsemble): R^2 = 0.8123.

El modelo automatizado de Azure AutoML se consolidó como el más preciso y estable, explicando más del 80 % de la variabilidad en la concentración de clorofila, con errores medios (MAE = 7.26 µg/L) y una alta consistencia en validación cruzada (CV $R^2 \approx 0.81 \pm 0.02$).

Estos resultados son consistentes con lo reportado por Fang et al. (2021), Qian et al. (2021) y Luo et al. (2022), quienes destacan que los algoritmos de ensamble y los enfoques automatizados ofrecen mayor fortaleza frente al ruido, la colinealidad y la variabilidad estacional de los datos ambientales. En particular, el uso de AutoML permitió explorar combinaciones de hiperparámetros y algoritmos que resultan difíciles de optimizar manualmente, logrando una mejora respecto al modelo base lineal.

Desde el punto de vista ecológico, los modelos coincidieron en identificar al fósforo total (TP), la relación fósforo-nitrógeno (TP/NH_4), la temperatura y la profundidad del agua como los principales impulsores de la concentración de clorofila. Estos resultados concuerdan con la literatura científica que señala estos factores como determinantes en la dinámica de las floraciones algales y en los procesos de eutrofización. La coherencia entre la relevancia de las variables y los fundamentos ecológicos sustenta la validez científica de las predicciones (Smith & Schindler, 2009).

El análisis de residuales confirmó que los modelos más avanzados, especialmente XGBoost y AutoML, reducen los errores sistemáticos y presentan una mejor distribución aleatoria de residuos, lo que indica mayor capacidad de generalización. Asimismo, las estrategias de preprocesamiento, imputación de datos (KNN), winsorización de valores atípicos y estandarización diferenciada según el tipo de algoritmo resultaron determinantes para alcanzar estabilidad y precisión en los resultados.

En síntesis, los resultados de esta investigación señalan que la integración de enfoques manuales y automatizados permite desarrollar modelos predictivos adecuados y reproducibles para estimar la concentración de clorofila y anticipar floraciones algales. Este enfoque proporciona una base metodológica, que puede ser escalable para la aplicación del aprendizaje automático en la gestión y monitoreo de ecosistemas acuáticos.

VIII. Bibliografía

Anderson, D. M., Glibert, P. M., & Burkholder, J. M. (2002). Harmful algal blooms and eutrophication: Nutrient sources, composition, and consequences. Estuaries, 25(4), 704–726.

Bailey, J., Hood, R., Boegehold, A., & Murphy, A. (2024). Chlorophyll and particulate microcystin concentrations from western Lake Erie (2012–2021) [Data set]. Zenodo.

Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32.

Carlson, R. E. (1977). A trophic state index for lakes. Limnology and Oceanography, 22(2), 361–369. https://doi.org/10.4319/lo.1977.22.2.0361

CDC. (2022). Las floraciones de algas nocivas y su salud. Centers for Disease Control and Prevention. https://www.cdc.gov/harmful-algal-blooms/es/about/las-proliferaciones-de-algas-nocivas-y-su-salud.html

Chai, T., & Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature. Geoscientific Model Development, 7(3), 1247–1250.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794. https://doi.org/10.1145/2939672.2939785

Clesceri, L. S., Greenberg, A. E., & Eaton, A. D. (Eds.). (1998). Standard Methods for the Examination of Water and Wastewater (20th ed.). American Public Health Association.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273–297. https://doi.org/10.1007/BF00994018

Dodds, W. K., & Smith, V. H. (2016). Nitrogen, phosphorus, and eutrophication in streams. Inland Waters, 6(2), 155–164. https://doi.org/10.5268/IW-6.2.909

EPA. (2023). Harmful algal blooms. U.S. Environmental Protection Agency. https://www.epa.gov/habs

Fang, L., Wang, Z., Li, J., & Guo, X. (2021). Short-term forecasting and risk assessment of microcystins in Lake Erie. Ecological Indicators, 129, 107884. https://www.sciencedirect.com/science/article/pii/S1470160X21007202?via%3Dihub

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., & Hutter, F. (2019). Auto-sklearn: Efficient and robust automated machine learning. Automated Machine Learning, 113–134. Springer.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

Hawkins, D. M. (2004). The problem of overfitting. Journal of Chemical Information and Computer Sciences, 44(1), 1–12.

Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. Technometrics, 12(1), 55–67.

Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). Applied logistic regression (3rd ed.). Wiley.

Hu, C., Lee, Z., Ma, R., Yu, K., Li, D., & Shang, S. (2010). Moderate resolution imaging spectroradiometer (MODIS) observations of cyanobacteria blooms in Taihu Lake, China. Journal of Geophysical Research: Oceans, 115(C4).

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). An Introduction to Statistical Learning with Applications in R (2nd ed.). Springer. https://doi.org/10.1007/978-1-0716-1418-1

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI), 2, 1137–1145.

Kuhn, M., & Johnson, K. (2013). Applied predictive modeling. Springer.

Lin, H., Wang, X., & Xu, Y. (2019). Prediction of algal bloom occurrence based on the naive Bayesian model. Ecological Indicators, 102, 636–643. https://doi.org/10.1016/j.ecolind.2019.03.004

Luo, Y., Zhao, J., & Zhang, Y. (2022). Evaluating automated machine learning for water quality prediction and management. Water Research, 221, 118735. https://doi.org/10.1016/j.watres.2022.118735

Maguire, T. J., Boegehold, A. G., Gell, P. A., & Reavie, E. D. (2024). Defining algal bloom phenology in Lake Erie. Journal of Great Lakes Research, 50(1), 103–117. https://doi.org/10.1016/j.jglr.2023.12.009

Martínez-Rivera, L. M., & Ramírez-Corona, N. (2016). Las algas como bioindicadores de la calidad del agua. Hidrobiológica, 26(1), 123–132. https://www.scielo.org.mx/scielo.php?pid=S0188-88972016000100002&script=sci_arttext

Mitchell, T. M. (1997). Machine learning. McGraw-Hill.

Montgomery, D. C., Peck, E. A., & Vining, G. G. (2021). Introduction to Linear Regression Analysis (7th ed.). Wiley.

Mu, M., Li, Y., Bi, S., Lyu, H., Xu, J., Lei, S., Miao, S., Zeng, S., Zheng, Z., & Du, C. (2021). Prediction of algal bloom occurrence based on the naive Bayesian model considering satellite image pixel differences. Ecological Indicators, 124, 107416. https://doi.org/10.1016/j.ecolind.2021.107416

Paerl, H. W., & Otten, T. G. (2013). Harmful cyanobacterial blooms: Causes, consequences, and controls. Microbial Ecology, 65(4), 995–1010.

Paerl, H. W., & Otten, T. G. (2016). Duelling 'CyanoHABs': Unravelling the environmental drivers controlling dominance and succession among diazotrophic and non-diazotrophic cyanobacteria. Environmental Microbiology, 18(2), 316–324.

Qian, S. S., et al. (2021). Understanding drivers of harmful algal blooms through machine learning. Water Research, 190, 116759.

Qian, S. S., Stow, C. A., Rowland, F. E., Liu, Q., Rowe, M. D., Anderson, E. J., Stumpf, R. P., & Johengen, T. H. (2021). Chlorophyll a as an indicator of microcystin: Short-term forecasting and risk assessment in Lake Erie. Ecological Indicators, 130, 108055. https://doi.org/10.1016/j.ecolind.2021.108055

Rabalais, N. N., Turner, R. E., & Wiseman, W. J. (2010). Hypoxia in the northern Gulf of Mexico: Does the science support the plan to reduce, mitigate, and control hypoxia? Estuaries and Coasts, 30(5), 753–772.

Rolinski, S., Horn, H., Petzoldt, T., & Paul, L. (2007). Identifying cardinal dates in phytoplankton time series to enable the analysis of long-term trends. Oecologia, 153(3), 997–1008.

Shi, K., Zhang, Y., & Qin, B. (2019). Remote sensing of chlorophyll-a concentration and algal blooms in inland lakes: A review. Remote Sensing of Environment, 232, 111356.

Shi, K., Zhang, Y., Xu, H., Zhu, G., Liu, X., Zhou, Y., & Qin, B. (2019). Long-term prediction of algal blooms in eutrophic Lake Taihu, China, using a hybrid ensemble learning model. Science of The Total Environment, 665, 791–803.

Smith, V. H., & Schindler, D. W. (2009). Eutrophication science: Where do we go from here? Trends in Ecology & Evolution, 24(4), 201–207.

Tran, T. N., Zhang, C., & Xie, Y. (2023). Comparing manual machine learning models and frameworks for environmental prediction tasks. Environmental Modelling & Software, 162, 105656. https://doi.org/10.1016/j.envsoft.2023.105656

Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., & Altman, R. B. (2001). Missing value estimation methods for DNA microarrays. Bioinformatics, 17(6), 520–525.

Verspagen, J. M. H., Visser, P. M., & Huisman, J. (2014). Aggregation with clay causes sedimentation of the buoyant cyanobacteria Microcystis. Aquatic Microbial Ecology, 73(1), 17–27.

Willmott, C. J., & Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. Climate Research, 30(1), 79–82.

Zhang, Y., Liu, X., Qin, B., Shi, K., Deng, J., & Zhou, Y. (2012). Aquatic vegetation in response to increased eutrophication and degraded light climate in Eastern Lake Taihu: Implications for lake ecological restoration. Hydrobiologia, 709, 113–125.

IX. Anexos

ANEXO 1. Dependencias del Proyecto (requirements.txt)

Lista completa de librerías Python utilizadas con sus versiones específicas para garantizar la reproducibilidad del entorno de desarrollo. Todas las librerías fueron instaladas en un entorno virtual Python 3.11.9.

Librerías principales:

pandas==2.3.1

numpy==2.3.2

matplotlib==3.10.3

seaborn==0.13.2

scikit-learn==1.7.1

xgboost==3.0.3

scipy==1.16.0

asttokens==3.0.0

bleach==6.2.0

certifi==2025.7.14

charset-normalizer==3.4.2

colorama==0.4.6

idna==3.10

ipykernel==6.30.0

ipython==9.4.0

ipython_pygments_lexers==1.1.1

jedi==0.19.2

joblib==1.5.1

json5==0.12.0

jsonpointer==3.0.0

jupyter_client==8.6.3

jupyter_core==5.8.1

jupyterlab_pygments==0.3.0

kiwisolver==1.4.8

comm==0.2.3

contourpy==1.3.2

cycler==0.12.1

debugpy==1.8.15

decorator==5.2.1

defusedxml==0.7.1

entrypoints==0.4

executing==2.2.0

fastjsonschema==2.21.1

fonttools==4.59.0

fqdn==1.5.1

h11==0.16.0

lark==1.2.2

MarkupSafe==3.0.2

matplotlib==3.10.3

matplotlib-inline==0.1.7

mistune==3.1.3

nest-asyncio==1.6.0

numpy==2.3.2

overrides==7.7.0

packaging==25.0

pandas==2.3.1

pandocfilters==1.5.1

parso==0.8.4

pillow==11.3.0

platformdirs==4.3.8

progressbar2==4.5.0

prometheus_client==0.22.1

prompt_toolkit==3.0.51

psutil==7.0.0

pure_eval==0.2.3

pycparser==2.22

pygam==0.10.0

Pygments==2.19.2

pyparsing==3.2.3

python-dateutil==2.9.0.post0

python-json-logger==3.3.0

python-utils==3.9.1

pytz==2025.2

pywin32==311

pywinpty==2.0.15

PyYAML==6.0.2

pyzmq==24.0.1

rfc3339-validator==0.1.4

rfc3986-validator==0.1.1

rpds-py==0.26.0

scikit-learn==1.7.1

scipy==1.16.0

seaborn==0.13.2

Send2Trash==1.8.3

six==1.17.0

sniffio==1.3.1

soupsieve==2.7

stack-data==0.6.3

threadpoolctl==3.6.0

tinycss2==1.4.0

tornado==6.5.1

traitlets==5.14.3

types-python-dateutil==2.9.0.20250708

typing_extensions==4.14.1

tzdata==2025.2

uri-template==1.3.0

urllib3==2.5.0

wcwidth==0.2.13

webcolors==24.11.1

webencodings==0.5.1

websocket-client==1.8.0

xgboost==3.0.3

ANEXO 2. Configuración Azure AutoML

Configuración del experimento de modelado automatizado ejecutado en Azure Machine Learning Studio.

Configuración:

- Tipo de tarea: Regresión (task_type: regression)
- Métrica primaria: R^2 Score (primary_metric: r2_score)
- Validación: Train-Validation split (80% entrenamiento / 20% validación)
- Semilla aleatoria: random_state=42 (para reproducibilidad)
- Porcentaje de datos: 100% del dataset (con división automática 80-20)
- Iteraciones de ensemble: 15
- Timeout por modelo: 300 segundos

Modelo final seleccionado:

- Algoritmo: StackEnsemble
- R^2 Score: 0.8123
- Composición del ensemble:
    - GradientBoosting: 5 modelos (pesos: 0.47, 0.07, 0.07, 0.07, 0.07)
    - RandomForest: 2 modelos (pesos: 0.20, 0.07)

Recursos computacionales:

- Tipo de instancia: Standard_DS12_v2
- Núcleos: 4 cores
- Memoria: 28 GB RAM
- Tiempo de ejecución: ~40 segundos por iteración
- Fecha de ejecución: 2025-10-14 (08:23:13 - 08:24:30 UTC)

Versiones de Azure ML:

- azureml-automl-core: 1.60.0
- azureml-train-automl-client: 1.60.0
- azureml-core: 1.60.0.post1

ANEXO 3. Muestra representativa de datos procesados y consolidados.

tp,srp,nh4,no3_2,ctd_temp,secchi,depth_m,sin_doy,**cos_doy,**tp_nh4_ratio,chl

16.209999999999997,3.23,10.22,0.459,23.56,1.8,0.75,0.7187918772884232,-**0.6952253139408722,**1.5861056751467706,3.67

8.25,2.02,28.79,0.325,21.42,2.4,0.75,0.7187918772884232,-**0.6952253139408722,**0.28655783258075723,3.0500000000000007

22.52,2.9700000000000006,35.7,0.515,24.62,2.1,0.75,0.7187918772884232,-**0.6952253139408722,**0.630812324929972,2.24

23.16,3.02,11.28,0.513,23.36,1.2,0.75,0.7187918772884232,-**0.6952253139408722,**2.0531914893617023,5.73

42.01,5.15,14.68,0.466,22.0,1.8,0.75,0.5027907197793043,-**0.8644081744776648,**2.861716621253406,6.07

ANEXO 4. Código utilizado para creación de modelos.

a) Código modelos manuales (Ridge Regression, Random Forest, XGBoost):

División de datos:

```
from sklearn.model_selection import train_test_split
X_entrenamiento, X_prueba, y_entrenamiento, y_prueba = train_test_split(
    X, y, test_size=0.2, random_state=42
)
print(f"Conjunto de entrenamiento: {X_entrenamiento.shape}")
print(f"Conjunto de prueba: {X_prueba.shape}")
```

Validación cruzada K-fold para ajuste de hiperparámetros:

```
# Ejemplo 1: Ridge Regression con RidgeCV
from sklearn.linear_model import RidgeCV
valores_alpha = [0.001, 0.01, 0.1, 1.0, 10.0, 100.0, 1000.0]
ridge_cv = RidgeCV(alphas=valores_alpha, cv=5, scoring='r2')
ridge_cv.fit(X_escalado, y)
print(f"Mejor alpha: {ridge_cv.alpha_}")
print(f"Mejor R² de VC: {ridge_cv.best_score_:.4f}")

# Ejemplo 2: Random Forest con RandomizedSearchCV
from sklearn.model_selection import RandomizedSearchCV
from sklearn.ensemble import RandomForestRegressor
grilla_parametros = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2', None]
}

rf = RandomForestRegressor(random_state=42, n_jobs=-1)
busqueda_aleatoria = RandomizedSearchCV(
```

```
    rf, grilla_parametros, n_iter=50, cv=5, scoring='r2',
    random_state=42, n_jobs=-1, verbose=1
)
busqueda_aleatoria.fit(X_entrenamiento, y_entrenamiento)
print(f"Mejores parámetros: {busqueda_aleatoria.best_params_}")
print(f"Mejor R² de VC: {busqueda_aleatoria.best_score_:.4f}")

# Ejemplo 3: XGBoost con RandomizedSearchCV
import xgboost as xgb
from sklearn.model_selection import RandomizedSearchCV
grilla_parametros = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 4, 5],
    'learning_rate': [0.05, 0.1],
    'subsample': [0.7, 0.8, 0.9],
    'colsample_bytree': [0.7, 0.8, 0.9],
    'min_child_weight': [3, 5, 7],
    'gamma': [0, 0.1, 0.2],
    'reg_alpha': [0, 0.1, 0.5, 1],
    'reg_lambda': [1, 1.5, 2, 3]
}
xgb_reg = xgb.XGBRegressor(random_state=42, n_jobs=-1, tree_method='hist')
busqueda_aleatoria = RandomizedSearchCV(
    xgb_reg, grilla_parametros, n_iter=150, cv=5, scoring='r2',
    random_state=42, n_jobs=-1, verbose=1
)
busqueda_aleatoria.fit(X_entrenamiento, y_entrenamiento)
print(f"Mejores parámetros: {busqueda_aleatoria.best_params_}")
print(f"Mejor R² de VC: {busqueda_aleatoria.best_score_:.4f}")
```

Evaluación adicional con validación cruzada:

```
# Validación cruzada sobre el dataset completo para estimación robusta
from sklearn.model_selection import cross_val_score
mejor_modelo = busqueda_aleatoria.best_estimator_
puntajes_vc = cross_val_score(mejor_modelo, X, y, cv=5, scoring='r2')
print(f"Media R² VC: {puntajes_vc.mean():.4f}")
print(f"Desv. Est. R² VC: {puntajes_vc.std():.4f}")
print(f"Rango R² VC: [{puntajes_vc.min():.4f}, {puntajes_vc.max():.4f}]")
```

ANEXO 5. Código para correlación entre variables ambientales y clorofila

```
# Separar características y variable objetivo

columna_objetivo = 'chl'

columnas_caracteristicas = [col for col in df.columns if col != columna_objetivo]


X = df[columnas_caracteristicas]

y = df[columna_objetivo]


# Calcular correlaciones de características con variable objetivo

correlaciones = df.corr()[columna_objetivo].abs().sort_values(ascending=False)


print(f"Correlaciones de Características con Clorofila:")

for caracteristica in correlaciones.index[1:]:  # Omitir la variable objetivo

    print(f"   {caracteristica}: {correlaciones[caracteristica]:.3f}")


# Visualizar matriz de correlación y gráficos

fig, axes = plt.subplots(2, 2, figsize=(15, 12))

fig.suptitle('Correlaciones Variables Ambientales vs. Clorofila', fontsize=16)


# Mapa de calor de correlaciones

matriz_correlacion = df.corr()

mascara = np.triu(np.ones_like(matriz_correlacion, dtype=bool))

sns.heatmap(matriz_correlacion, mask=mascara, annot=True, fmt='.2f',

            cmap='RdBu_r', center=0, ax=axes[0,0])

axes[0,0].set_title('Matriz de Correlación')
```

```
# Gráfico de barras de correlaciones
correlaciones[1:].plot(kind='barh', ax=axes[0,1])
axes[0,1].set_title('Correlaciones de Características con Clorofila')
axes[0,1].set_xlabel('Correlación Absoluta')

# Scatter plot de correlación más fuerte
caracteristica_mas_fuerte = correlaciones.index[1]
axes[1,1].scatter(df[caracteristica_mas_fuerte], y, alpha=0.6)
axes[1,1].set_title(f'Clorofila vs. {caracteristica_mas_fuerte}')
axes[1,1].set_xlabel(caracteristica_mas_fuerte)
axes[1,1].set_ylabel('Clorofila (µg/L)')

plt.tight_layout()
plt.show()
```

ANEXO 6. Código para generación de gráficas y visuales.

a. Modelo Ridge Regression

```
# Visualización de resultados

fig, axes = plt.subplots(2, 2, figsize=(15, 12))

fig.suptitle('Regresión Ridge - Análisis de Variables Ambientales', fontsize=16)


# Coeficientes

importancia_caracteristicas.plot(x='caracteristica', y='coeficiente', kind='barh', ax=axes[0,0])

axes[0,0].set_title('Coeficientes Ridge (Importancia de Características)')

axes[0,0].set_xlabel('Valor del Coeficiente')

axes[0,0].axvline(x=0, color='black', linestyle='-', alpha=0.3)


# Predicho vs Real

axes[0,1].scatter(y_prueba, y_prueba_pred, alpha=0.6)

axes[0,1].plot([y_prueba.min(), y_prueba.max()], [y_prueba.min(), y_prueba.max()],

              'r--', lw=2, alpha=0.8)

axes[0,1].set_xlabel('Clorofila Real (µg/L)')

axes[0,1].set_ylabel('Clorofila Predicha (µg/L)')

axes[0,1].set_title(f'Predicho vs Real (R² Prueba = {r2_prueba:.3f})')


# Residuos

residuos = y_prueba - y_prueba_pred

axes[1,0].scatter(y_prueba_pred, residuos, alpha=0.6)

axes[1,0].axhline(y=0, color='red', linestyle='--', alpha=0.8)

axes[1,0].set_xlabel('Clorofila Predicha (µg/L)')

axes[1,0].set_ylabel('Residuos (µg/L)')

axes[1,0].set_title('Gráfico de Residuos')


# Validación cruzada

axes[1,1].bar(range(len(puntajes_vc)), puntajes_vc)
```

```
axes[1,1].axhline(y=puntajes_vc.mean(), color='red', linestyle='--',
                  label=f'Media = {puntajes_vc.mean():.3f}')
axes[1,1].set_xlabel('Pliegue VC')
axes[1,1].set_ylabel('Puntaje R²')
axes[1,1].set_title('Puntajes R² de Validación Cruzada')
axes[1,1].legend()

plt.tight_layout()
plt.show()
```

b. Modelo Random Forest

```
# Visualizar importancia integral de características y rendimiento del modelo
fig, axes = plt.subplots(2, 3, figsize=(18, 12))
fig.suptitle('Random Forest - Análisis de Variables Ambientales', fontsize=16)

# Importancia Gini
importancia_gini.plot(x='caracteristica', y='importancia_gini', kind='barh',
ax=axes[0,0])
axes[0,0].set_title('Importancia de Características Basada en Gini')
axes[0,0].set_xlabel('Importancia')

# Importancia por permutación
axes[0,1].barh(importancia_perm_df['caracteristica'],
importancia_perm_df['importancia_perm'],
               xerr=importancia_perm_df['desv_est_perm'], capsize=3)
axes[0,1].set_title('Importancia de Características por Permutación')
axes[0,1].set_xlabel('Importancia ± Desv. Est.')

# Importancia combinada
importancia_combinada.plot(x='caracteristica',      y='importancia_promedio',
kind='barh', ax=axes[0,2])
axes[0,2].set_title('Importancia de Características Combinada')
axes[0,2].set_xlabel('Importancia Promedio')
```

```
# Predicho vs Real
axes[1,0].scatter(y_prueba, y_prueba_pred, alpha=0.6)
axes[1,0].plot([y_prueba.min(),       y_prueba.max()],       [y_prueba.min(),
y_prueba.max()],
              'r--', lw=2, alpha=0.8)
axes[1,0].set_xlabel('Clorofila Real (µg/L)')
axes[1,0].set_ylabel('Clorofila Predicha (µg/L)')
axes[1,0].set_title(f'Predicho vs Real (R² = {r2_prueba:.3f})')

# Gráfico de residuos
residuos = y_prueba - y_prueba_pred
axes[1,1].scatter(y_prueba_pred, residuos, alpha=0.6)
axes[1,1].axhline(y=0, color='red', linestyle='--', alpha=0.8)
axes[1,1].set_xlabel('Clorofila Predicha (µg/L)')
axes[1,1].set_ylabel('Residuos (µg/L)')
axes[1,1].set_title('Gráfico de Residuos')

# Puntajes de validación cruzada
axes[1,2].bar(range(len(puntajes_vc)), puntajes_vc)
axes[1,2].axhline(y=puntajes_vc.mean(), color='red', linestyle='--',
                  label=f'Media = {puntajes_vc.mean():.3f}')
axes[1,2].set_xlabel('Pliegue VC')
axes[1,2].set_ylabel('Puntaje R²')
axes[1,2].set_title('Puntajes R² de Validación Cruzada')
axes[1,2].legend()

plt.tight_layout()
plt.show()
```

c. Modelo XGBoost

```
# Visualizar análisis integral
fig, axes = plt.subplots(2, 3, figsize=(18, 12))
fig.suptitle('XGBoost - Análisis de Variables Ambientales', fontsize=16)
```

```
# Importancia por ganancia
importancia_ganancia.plot(x='caracteristica',       y='importancia_ganancia',
kind='barh', ax=axes[0,0])
axes[0,0].set_title('Importancia de Características Basada en Ganancia')
axes[0,0].set_xlabel('Importancia')

# Importancia por peso
importancia_peso = df_importancia['weight'].sort_values(ascending=True)
importancia_peso.plot(kind='barh', ax=axes[0,1])
axes[0,1].set_title('Importancia Basada en Peso')
axes[0,1].set_xlabel('Número de veces usada')

# Importancia por cobertura
importancia_cobertura = df_importancia['cover'].sort_values(ascending=True)
importancia_cobertura.plot(kind='barh', ax=axes[0,2])
axes[0,2].set_title('Importancia Basada en Cobertura')
axes[0,2].set_xlabel('Cobertura')

# Predicho vs Real
axes[1,0].scatter(y_prueba, y_prueba_pred, alpha=0.6)
axes[1,0].plot([y_prueba.min(),       y_prueba.max()],       [y_prueba.min(),
y_prueba.max()],
              'r--', lw=2, alpha=0.8)
axes[1,0].set_xlabel('Clorofila Real (µg/L)')
axes[1,0].set_ylabel('Clorofila Predicha (µg/L)')
axes[1,0].set_title(f'Predicho vs Real (R² = {r2_prueba:.3f})')

# Gráfico de residuos
residuos = y_prueba - y_prueba_pred
axes[1,1].scatter(y_prueba_pred, residuos, alpha=0.6)
axes[1,1].axhline(y=0, color='red', linestyle='--', alpha=0.8)
axes[1,1].set_xlabel('Clorofila Predicha (µg/L)')
axes[1,1].set_ylabel('Residuos (µg/L)')
```

```
axes[1,1].set_title('Análisis de Residuos')

# Puntajes de validación cruzada
axes[1,2].bar(range(len(puntajes_vc)), puntajes_vc)
axes[1,2].axhline(y=puntajes_vc.mean(), color='red', linestyle='--',
                  label=f'Media = {puntajes_vc.mean():.3f}')
axes[1,2].set_xlabel('Pliegue VC')
axes[1,2].set_ylabel('Puntaje R²')
axes[1,2].set_title('Puntajes R² de Validación Cruzada')
axes[1,2].legend()

plt.tight_layout()
plt.show()
```