TÍTULO

CLASIFICACIÓN DE MUESTRAS DE AGUA PARA DETERMINAR SU POTABILIDAD MEDIANTE EL USO DE ALGORITMOS DE APRENDIZAJE AUTOMÁTICO

AUTOR

Daniel González Andújar

Esta edición electrónica ha sido realizada en 2024

Tutores	Dr. D. Diego Marín Santos ; Dr. D. Manuel Emilio Gegúndez Arias
Institución	Universidad Internacional de Andalucía
Curso	*Máster de Formación Permanente en Big Data (2022/23)*
©	Daniel González Andújar
©	De esta edición: Universidad Internacional de Andalucía
Fecha documento	2023

Trabajo Fin de Máster

Clasificación de muestras de agua para determinar su potabilidad mediante el uso de algoritmos de aprendizaje automático

Daniel González Andújar

Septiembre de 2.023

Tutores: Diego Marín Santos

Manuel Emilio Gegúndez Arias

Máster propio en Big Data

Universidad Internacional de Andalucía

Universidad Internacional de Andalucía, 2024

Agradecimientos.

“A mi familia y tutores por el apoyo en este largo camino”

Abstract.

The main porpuse of this work is to analyze the results obtained in the classification of different water samples through the use of algorithms and machine learning methods. Access to safe drinking water services is still a problem for approximately 2,000 million people around the world, which makes it even more necessary to study and predict the potability of water samples, this analysis is a tool for the prevention of disease and even death. A dataset with water samples has been extracted from the Kaggle website, specifically, the database is made up of a total of 3,276 instances of water samples, where 59.67% correspond to non-potable water samples. Different supervised learning methods such as K-NN, Random Forest or SVM have been used to classify the water samples, and they have been evaluated with different methodologies. The results obtained with the different classifiers, depending on the methodology, vary significantly.

The algorithm which we have obtained the best results has been SVM, which is capable of working with an AUC of 0.75, Hit Rate of 0.72, Sensitivity of 0.71 and Specificity of 0.73. Although these results are susceptible of improvement, they indicate that the application of algorithms based on machine learning can constitute an important tool to predict non-potable water samples.

Keywords.

Machine learning, supervised learning, binary classification, public heath, water potability.

Resumen.

El objetivo principal de este trabajo es analizar los resultados obtenidos en la clasificación de diferentes muestras de agua mediante el uso de algoritmos y métodos de aprendizaje automático. El acceso a servicios de agua potable segura es todavía un problema para 2.000 millones de personas aproximadamente en todo el mundo, lo que hace aún más necesario el estudio y predicción de la potabilidad de las muestras de agua, siendo este análisis una herramienta de prevención de enfermedades e incluso de la muerte. Se ha hecho uso de un conjunto de datos con muestras de agua extraído de la web *Kaggle*. Concretamente, la base de datos se compone de un total de 3.276 instancias de muestras de agua, donde 59,67% se corresponden con muestras de agua no potable. Para la clasificación de las muestras de agua se han empleado diferentes técnicas de aprendizaje supervisado como K-NN, Random Forest o SVM, y se han evaluado con diferentes metodologías. Los resultados obtenidos con los diferentes clasificadores, dependiendo de la metodología, varían significativamente.

El algoritmo con el que mejores resultados hemos obtenido ha sido el SVM, que es capaz de trabajar con un AUC de 0,75, Tasa de Acierto de 0,72, Sensibilidad de 0,71 y Especificidad de 0,73. Aunque estos resultados son susceptibles de mejora, indican que la aplicación de algoritmos basados en aprendizaje automático puede constituir una importante herramienta para predecir muestras de agua no potable.

Palabras clave.

Aprendizaje automático, aprendizaje supervisado, clasificación binaria, potabilidad del agua, salud pública.

Índice general.

1. Propuesta de Trabajo Fin de Máster. 5

1.1. Motivación. 5
1.2. Objetivos. 6
1.3. Organización del documento. 7

2. Introducción y estado del conocimiento. 8

3. Marco teórico. 10

3.1. Técnicas de clasificación seleccionadas. 10
3.1.1. K-NN (K-Nearest Neighbors). 10
3.1.2. Naive-Bayes. 12
3.1.3. Random Forest. 14
3.1.4. Máquina Vector Soporte (SVM). 16
3.2. Metodologías y métricas de evaluación. 19
3.2.1. Hold-Out. 19
3.2.2. Validación cruzada. 19
3.2.3. Leave-One-Out. 19
3.2.4. Matriz de confusión. 20
3.2.5. Métricas. 20

4. Materiales. 23

4.1. Descripción de la base de datos. 23
4.2. Análisis de datos. 26
4.3. Estandarización de datos. 31
4.4. Importancia de predictores. 31

5. Experimentación. 33

5.1. Descripción de metodología. 33
5.2. Conjunto de datos. 34
5.3. Selección y ajuste de modelos. 35
5.4. Aplicación y generación de resultados. 37
5.5. Análisis de resultados 42

6. Conclusiones. 43

7. Bibliografía. 44

Anexos. 45

1. Propuesta de Trabajo Fin de Máster.

1.1. Motivación.

En la actualidad el acceso a agua potable todavía es limitado para una parte de la población mundial. Este recurso es esencial para el desarrollo socioeconómico, energético y fundamental para la supervivencia del ser humano. El acceso a agua potable y a un sistema de saneamiento es vital para reducir el número de enfermedades, así como para mejorar la salud, educación y productividad económica de la población.

Debemos tener en cuenta que 2.000 millones de personas aproximadamente carecen de acceso a agua potable en todo el mundo y que alrededor de 247.000 niños de entre 0 y 5 años mueren anualmente por enfermades causadas por las malas condiciones del agua o ingesta de agua no potable.

Además, las aguas contaminadas dificultan la erradicación de la pobreza extrema y de las enfermedades en los países más pobres.

En este contexto, encontramos en el análisis de datos, más concretamente en el aprendizaje automático, una herramienta de predicción y prevención, de tal forma que sirva de ayuda en la reducción de enfermedades.

Nuestro objetivo con el desarrollo de este proyecto se basa en la generación de un modelo mediante el uso de algoritmos de aprendizaje supervisado que nos permita predecir y clasificar nuevas muestras de agua (potable o no potable). Para poder generar nuestro modelo se hace uso de una base de datos con muestras de agua de la plataforma *Kaggle*. A partir de la información recogida tratamos de clasificar nuevas muestras de agua en base a diferentes variables como pH, solubilidad, turbidez, sulfatos o cloraminas, entre otros.

1.2. Objetivos.

Los objetivos específicos de nuestro proyecto se centran en tres bloques que indicamos a continuación:

1. Desarrollar varios modelos de *machine learning* para la predicción de la potabilidad del agua. Concretamente, se han aplicado K-NN, Naive-Bayes, Random Forest y SVM.
2. Entrenar, evaluar y optimizar los diferentes modelos indicados anteriormente. Para ello se han empleado estrategias de validación cruzada, concretamente 10-fold cross-validation y leave-one-out. La evaluación se ha llevado a cabo mediante análisis de curva ROC y métricas de clasificación binaria en el punto óptimo de la curva (Tasa de Aciertos, Sensibilidad y Especificidad),
3. Comparar y valorar los resultados obtenidos con cada uno de los modelos, de tal forma que estemos en disposición de hacer uso de un modelo de *machine learning* capaz de clasificar muestras de agua.

El objetivo global de nuestro proyecto es encontrar un clasificador con el mayor rendimiento capaz de predecir si una muestra de agua es potable o no. Para llegar a este objetivo será necesario cumplir cada uno de los objetivos específicos indicados anteriormente.

1.3. Organización del documento.

La estructura de nuestra memoria se divide en seis apartados. El primer y segundo apartado tratan sobre el objetivo del proyecto e introducción de la materia. El tercer apartado contiene información teórica acerca de los diferentes modelos y metodologías de evaluación empleadas. El apartado cuarto describe y prepara el conjunto de datos para el próximo apartado, el quinto, donde se desarrolla la experimentación y se analizan los resultados obtenidos. Por último, el apartado sexto, contiene las conclusiones del proyecto.

2. Introducción y estado del conocimiento.

El aprendizaje automático trata de extraer conocimiento a partir de los datos y es un campo que engloba parte de estadística, inteligencia artificial y ciencias de la computación.

La aplicación de métodos o técnicas de aprendizaje automático se ha convertido en los últimos años en algo imprescindible en nuestra vida diaria, que va desde recomendaciones de qué película ver hasta qué productos comprar.

En el inicio de las aplicaciones "inteligentes", muchas de ellas usaban reglas *if* y *else* para el procesamiento de datos, un ejemplo sencillo es, por ejemplo, el del filtro antispam de nuestro correo electrónico que irá moviendo a una carpeta los correos *no deseados*. Este es un claro ejemplo del uso de reglas para la toma de decisiones, pero hay casos en los que el uso de estas reglas no es válido. Un ejemplo de ello puede ser el de reconocimiento de imágenes, que fue un problema que no se resolvió hasta el año 2.001 aproximadamente, debido a cómo eran percibidos los píxeles por el ordenador y cómo los percibía el ser humano. Hoy en día, con el uso de técnicas de aprendizaje automático solo necesitamos un programa con un número elevado de imágenes para que el algoritmo determine qué características necesita para identificar la cara de un ser humano.

Aquellos algoritmos de aprendizaje automático que son capaces de generar una respuesta en base a conocimientos previos adquiridos los conocemos como algoritmos de *aprendizaje supervisado*, en los cuales se les da una información de entrada esperando de ellos una respuesta. El algoritmo es capaz de generar una respuesta a partir de una entrada que no ha visto anteriormente sin la ayuda de un ser humano, solo en base a ese conocimiento previamente adquirido. Algún ejemplo de este tipo de aprendizaje puede ser la identificación de códigos postales de cartas, determinación de si un tumor es benigno o no en base a una imagen o detección de actividades fraudulentas en tarjetas de crédito y transferencias bancarias. Dentro del *aprendizaje supervisado* nos encontramos con problemas de clasificación y regresión. En el primero de ellos el objetivo es conocer la clase a la que pertenece nuestra instancia mientras que en el segundo de ellos el objetivo es conocer un número.

Por otro lado, tenemos los algoritmos de *aprendizaje no supervisado*, en los cuales solo se conocen los datos de entrada al algoritmo y ningún dato de salida es conocido. Habitualmente son algoritmos más complicados de comprender y evaluar que los algoritmos de aprendizaje supervisado. Algunos ejemplos de problemas que pueden resolverse con este tipo de algoritmos son la segmentación de clientes en función de las

preferencias o identificación de tendencias en *blogs* online. El análisis de agrupaciones (clustering) es uno de los problemas que se pueden resolver con algoritmos de *aprendizaje no supervisado*, tratando de descubrir una estructura interna y agrupándolos en grupos con características similares.

Principalmente nuestro problema se basa en la predicción de la potabilidad de muestras de agua, determinando si es apta para el consumo humano o no. Este recurso es fundamental para el desarrollo de los seres humanos, por lo que es estrictamente necesario conocer sus características, sobre todo en regiones pobres donde el acceso a agua potable es un grave problema. Es aquí donde el *aprendizaje automático*, y en particular el *aprendizaje supervisado (clasificación),* permite predecir salidas y obtener respuestas. El aprendizaje supervisado es utilizado en el ámbito educativo, financiero, marketing, sanidad, etc, en definitiva, está presente en todos los ámbitos de nuestra vida.

Sanaa Kaddoura [10] hace uso de diferentes herramientas de *aprendizaje automático* para clasificar muestras de agua. Amir Hamzeh Haghiabi, Ali Heidar Nasrolahi and Abbas Parsaie [11], en su artículo también hacen uso de diferentes técnicas de *aprendizaje automático* para predecir la calidad de diferentes muestras de agua de Tireh River (Iran), obteniendo buenos resultados con SVM. Jinal Patel [14], en su artículo emplea diferentes algoritmos de *aprendizaje automático* para comparar el rendimiento en la predicción de la calidad de muestras de agua.

En nuestro trabajo nos basaremos en algoritmos de *aprendizaje supervisado* para la resolución de un problema de clasificación, y más concretamente en un problema de clasificación binaria

3. Marco teórico.

En este apartado haremos una descripción teórica las técnicas de clasificación empleadas en nuestro proyecto y la metodología y métricas de evaluación.

3.1. Técnicas de clasificación seleccionadas.

3.1.1. K-NN (K-Nearest Neighbors).

Podemos decir que el algoritmo K-NN es uno de los clasificadores más simples que existe. Es llamado *lazy learner* y se le designa de esta forma no por su aparente simplicidad sino por su funcionamiento, ya que el algoritmo no genera un modelo durante el entrenamiento, sino que memoriza los datos en su lugar. La clasificación se hace cuando llega la instancia de test.

Basándonos en la métrica de distancia elegida, el algoritmo K-NN determina el número de muestras (K) en el conjunto de entrenamiento más cercanas a la instancia que queremos clasificar. La clase de la instancia que se está clasificando viene determinada por la clase mayoritaria entre los K vecinos.

Cada instancia en el espacio *d-dimensional* puede expresarse como un *d-vector* de coordenadas:

$$p = (p_1, p_2, p_3 \dots p_n)$$

La instancia objeto de estudio para clasificar puede expresarse:

$$q = (q_1, q_2, q_3 \dots q_n)$$

La métrica de distancia elegida entre dos puntos en el espacio *d-dimensional* puede calcularse de muchas formas. Alguna de las más usuales son la distancia Euclídea o la distancia de Manhattan.

La distancia Euclídea la definimos como:

$$dist\ (p, q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

La distancia de Manhattan la definimos como:

$$dist'(p,q) = \sum_{i=1}^{n} | p_i - q_i |$$

Podemos resumir el funcionamiento del algoritmo K-NN en los siguientes pasos:

1.- Elegir el número K y la métrica de distancia (Euclidea, Manhatan, Minkoski, etc).

2.- Encontrar los K-vecinos más cercanos de la instancia objeto de estudio que queremos clasificar.

3.- Asignar la clase mayoritaria entre los K-vecinos más cercanos.

El proceso de clasificación que acabamos de describir se puede apreciar en la siguiente figura, *Figura 1*.

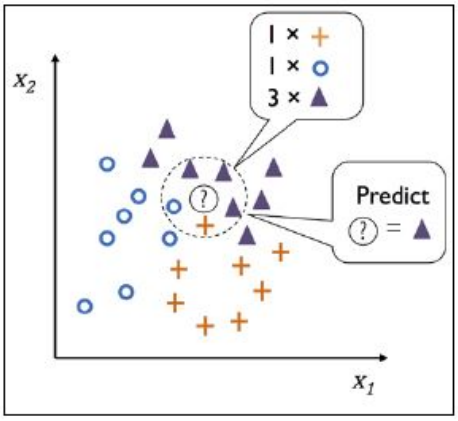

Figura 1. Clasificador K-NN.

Principalmente, el clasificador K-NN tiene dos importantes parámetros, el número de vecinos (K) y la métrica de distancia elegida.

Uno de los puntos fuerte de este algoritmo es su fácil comprensión y la rapidez para la construcción de modelos. En cambio, con conjuntos de entrenamiento grandes es un algoritmo lento.

3.1.2. Naive-Bayes.

El método de clasificación Naive-Bayes es un modelo de predicción basado en el teorema de Bayes. Se le asigna una clase a la instancia que tenga mayor probabilidad de pertenencia a dicha clase.

El teorema de Bayes se expresa con la siguiente ecuación:

$$P\ (A_i\ |B) = \frac{P(B|A_i)P(A_i)}{P(B)}$$

Sea $A = A_1, A_2, A_3 \ldots A_i$ un conjunto de sucesos mutuamente excluyentes tales que la probabilidad de cada uno de ellos es distinta de 0 y sea B un suceso cualquiera, donde:

$P(A_i)$ es la probabilidad a priori.

$P(B \mid A_i)$ es la probabilidad de B dado el suceso A_i.

$P(A_i \mid B)$ es la probabilidad a posteriori.

Aplicando el modelo de probabilidad para un clasificador:

$$P\ (y\ |X) = \frac{P(X|y)P(y)}{P(X)}$$

Del modelo anterior, tenemos:

y es la clase de salida.

X es la instancia para clasificar.

$P(y|X)$ es la probabilidad a posteriori de pertenencia de una instancia/observación a la clase y.

$P(X|y)$ es la probabilidad a posteriori de la instancia X en las muestras de la clase y.

$P(y)$ es la probabilidad a priori de pertenencia de una instancia/observación a la clase y.

Podemos encontrarnos con varios tipos de clasificadores Naive-Bayes:

1.- Clasificador multinomial: este clasificador se usa generalmente en la clasificación de textos.

2.- Clasificador Bernoulli: el clasificador de Bernoulli se implementa cuando los valores de la instancia son binarios.

3.- Clasificador Gaussiano: este clasificador es usado cuando los valores de la instancia son continuos, asumiendo que los valores siguen una distribución normal, $N(\mu,\sigma)$

En nuestro proyecto se implementará un clasificador Gaussiano. El clasificador Naive-Bayes, por lo general, es rápido, pero merma la precisión del modelo tendiendo a la generalización. Este clasificador es usado con grandes volúmenes de datos.

3.1.3. Random Forest.

El clasificador Random Forest se compone de un conjunto de árboles de decisión. En cada árbol de decisión, las observaciones se van distribuyendo por nodos, generando la estructura de árbol hasta un nodo terminal. La predicción de una nueva instancia se obtiene agregando las predicciones de todos los árboles que forman el clasificador y la premisa es predecir la clase de la observación en estudio de la clase predominante de una región determinada.

Resumiremos, a continuación, la fase de diseño para la construcción de un clasificador basado en árboles:

1.- Dividir el conjunto de valores en regiones, diferentes y no superpuestas.

2.- A cada una de estas regiones generadas se le asigna un valor de predicción definido por la clase más frecuente de las observaciones que componen la región en cuestión.

3.- Ya en la fase de test, para cada observación se predice la salida como la clase asociada a la región a la que pertenece por los valores de sus predictores.

Para dividir nuestro conjunto de datos en regiones necesitamos alguna métrica/técnica que cuantifique el error cometido en la clasificación de las instancias. Entre estas métricas se encuentran el *índice de Gini* y *entropía*, que cuantifican la variabilidad de las clases en una región determinada.

El índice de Gini:

$$G = \sum_{k=1}^{K} \hat{p}_{mk} \, (1 - \hat{p}_{mk})$$

$\hat{p}_{mk}$, representa la proporción de observaciones de entrenamiento de la región m que son de la clase k. El índice de Gini es referido como una medida de pureza del nodo, un valor bajo del índice de Gini representa que el nodo contiene observaciones de una clase de manera predominante.

Como alternativa tenemos la entropía:

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \, log \, \hat{p}_{mk}$$

Al igual que el índice de Gini, si la entropía tiene un valor bajo indica pureza del nodo.

El proceso de clasificación que hemos descrito se puede apreciar en la siguiente figura, *Figura 2*.

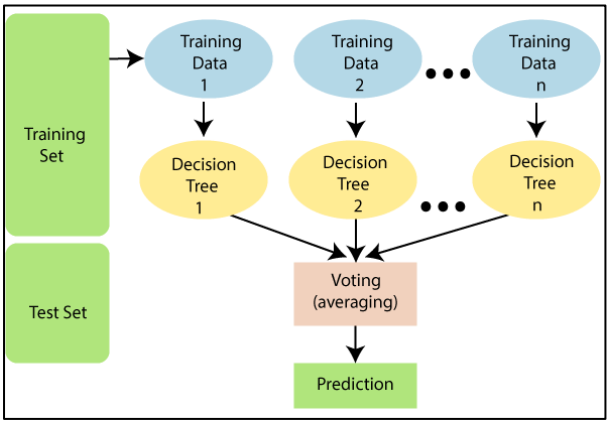

Figura 2. Funcionamiento de algoritmo Random Forest.

En cuanto a sus ventajas, son clasificadores simples y no se ven muy influenciados por valores *outlier*, tienen buena escalabilidad, pueden aplicarse a grandes conjuntos de datos. Por otro lado, al combinar varios árboles, se pierde la interpretabilidad que tienen los modelos basados en un único árbol.

3.1.4. Máquina Vector Soporte (SVM).

Las máquinas de vector soporte se basan en el *Maximal Margin Classifier,* que al mismo tiempo se basan en el concepto de hiperplano.

Un hiperplano lo definimos como un subespacio plano de dimensiones *p-1* (en un espacio *p-dimensional*), es decir, en un espacio de dos dimensiones nuestro hiperplano será una recta, y en un espacio de tres dimensiones nuestro hiperplano será un plano.

La definición matemática para un hiperplano en dos dimensiones:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$

Para el caso *p-dimensional*, esto se puede extender:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 \ldots . \beta_p X_p = 0$$

En definitiva, el hiperplano, divide el espacio *p-dimensional* en dos mitades. Podemos observar en la siguiente figura, *Figura 3,* un hiperplano en 2D y 3D tal y como hemos descrito.

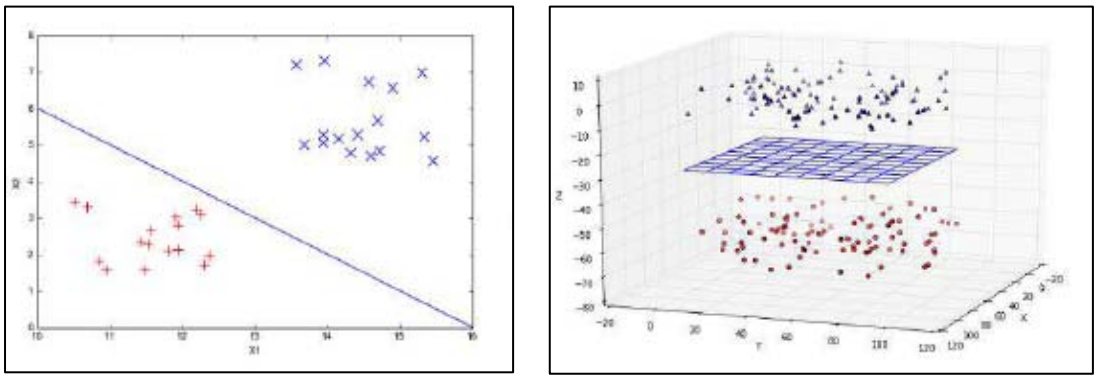

Figura 3. Hiperplano de separación de conjunto de datos en 2D y 3D.

Si disponemos de un conjunto de datos separable mediante un hiperplano habrá un número infinito de posibilidades de encontrar un hiperplano que separe las observaciones. Por tanto, es necesario encontrar un hiperplano que se encuentre lo más alejado de todas las observaciones, al que llamamos *maximal margin hyperplane,* tal como muestra la *Figura 4*. Debemos calcular la distancia perpendicular de cada observación a un hiperplano, la menor de las distancias calculadas es conocida como margen.

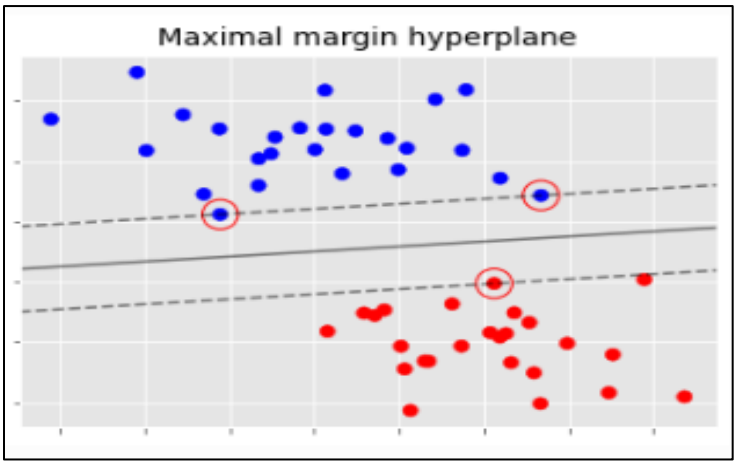

Figura 4. Hiperplano de margen máximo.

Si β_0, β_1, β_2 …. β_p son los coeficientes del hiperplano de margen máximo, entonces el clasificador de margen máximo clasifica las instancias de test x^*, basado en el signo:

$$f(x^*) = \beta_0, + \beta_1 x_1^* + \beta_2 x_2^* + \ldots. + \beta_p x_p^*$$

Este método es válido para casos linealmente separables, por lo que es limitante ya que rara vez encontraremos conjuntos de datos perfectamente separables.

Para asegurar que cada observación se ubica en el lado correcto del hiperplano, debemos cumplir:

$$\sum_{i=1}^{p} \beta_j^2 = 1$$

y

$$y_i \, (\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots. + \beta_p x_p) \geq M \; \forall \; i = 1, \ldots . . n.$$

M representa el margen del hiperplano y la optimización elige β_0, β_1, β_2 …. β_p de forma que maximice M.

En casos en los que nos encontremos con conjunto de datos separables cuasi-linealmente, se empleará el *clasificador de margen suave o clasificador de vector soporte,* permitiendo situar a determinadas instancias en un lugar incorrecto del margen e incluso del hiperplano, de manera que se cometan pocos errores, creando así un método más robusto ante nuevas instancias. El clasificador tiene un hiperparámetro, C, que controla la violación del margen.

Para asegurar que cada observación se ubica en el lado correcto del hiperplano, pero clasificando alguna instancia en el lugar incorrecto, debemos cumplir:

$$\sum_{i=1}^{p} \beta_j^2 = 1$$

y

$$y_i\,(\beta_0 + \ \beta_1\,x_1 + \beta_2\,x_2 + \ldots. + \beta_p\,x_p)\ \geq\ M\,(1 - \epsilon_i), \quad \epsilon_i \geq 0, \ \ \sum_{i=1}^{n} \epsilon_i \leq C$$

M representa el margen del hiperplano y C el hiperparámetro que controla la violación del margen, $\epsilon_1, \epsilon_2, \ldots. \epsilon_n$ son *slack variables* que permiten a las instancias estar en el lugar incorrecto del margen o del hiperplano. Si $\epsilon_i = 0$, la observación está en el lugar correcto del margen, si $\epsilon_i > 0$ la observación está en el lugar incorrecto del margen y decimos que se ha violado el margen, si $\epsilon_i > 1$ entonces la observación está en el lugar incorrecto del hiperplano. En el caso de C, si C = 0 no hay violaciones del margen, si C > 0 no puede haber más *C-observaciones* en el lado incorrecto del hiperplano. Conforme aumenta el valor de C somos más tolerantes con las violaciones del margen.

Cuando no podemos separar ni lineal ni cuasi-linealmente, se empleará la máquina de vector soporte, técnica que produce contornos de decisión no lineales. Se introduce el concepto de *kernel*, para ampliar el espacio de características y generar fronteras no lineales. Un *kernel* es una función que devuelve el resultado del producto escalar de dos vectores realizado en un nuevo espacio dimensional diferente al espacio original en el que se encuentran los vectores. Podemos encontrar diferentes *kernels*: lineal, radial, polinomial, etc.

3.2. Metodologías y métricas de evaluación.

Una vez entrenado el modelo en cuestión, el siguiente paso será evaluar la precisión con datos diferentes a los utilizados durante el entrenamiento.

A continuación, se resumen las distintas metodologías de evaluación aplicables en problemas de aprendizaje automático.

3.2.1. Hold-Out.

Como se muestra en la *Figura 5,* consiste en dividir el conjunto de datos en dos, uno para entrenar el modelo y otro para evaluar el rendimiento de este.

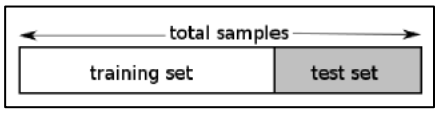

Figura 5. División de conjunto de datos.

3.2.2. Validación cruzada.

Este método consiste en dividir el conjunto de entrenamiento en K partes iguales, conocido como *K-fold cross validation*, el valor de K suele ser 5 ó 10. En una evaluación con K = 5, en el primer modelo generado, el Fold 1 será el de evaluación mientras que los otros cuatro (Fold 2-5) serán los de entrenamiento. En el segundo modelo, el Fold 2 será el de evaluación mientras que los otros cuatro (Fold 1, 3-5) serán los de entrenamiento. Y así hasta generar los cinco modelos. Una vez generados, una forma común de obtener la precisión del modelo es realizar la media de la precisión de cada uno de los cinco modelos. En la siguiente figura, *Figura 6,* se muestra el principio de funcionamiento de este método.

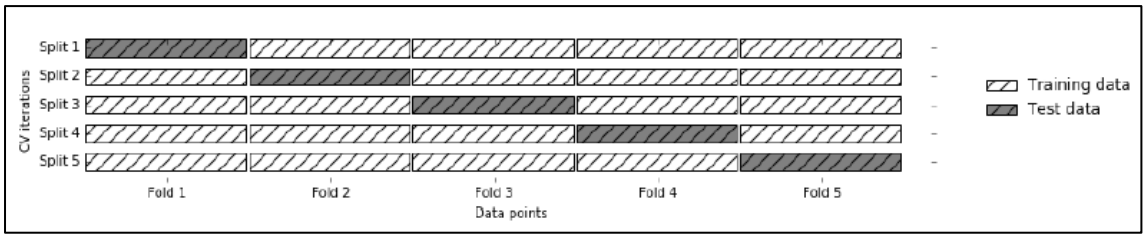

Figura 6. Partición de datos para validación cruzada.

3.2.3. Leave-One-Out.

Este método es una variante de la validación cruzada, en el cual el número de particiones (K) es igual al número de observaciones de nuestro conjunto de datos. Es

muy costoso computacionalmente por lo que solo es recomendable utilizarlo con conjuntos de datos pequeños.

Respecto a las métricas de clasificación, a continuación, se relacionan las más utilizadas en problemas de clasificación binaria, como es el que nos ocupa en este trabajo.

3.2.4. Matriz de confusión.

Se trata de una tabla cruzada que muestra el recuento de aciertos y errores del clasificador de cada una de las clases (predicciones) en relación con la clase real de las observaciones. Como se muestra en la *Figura 7,* tenemos cuatro tipos de predicciones.

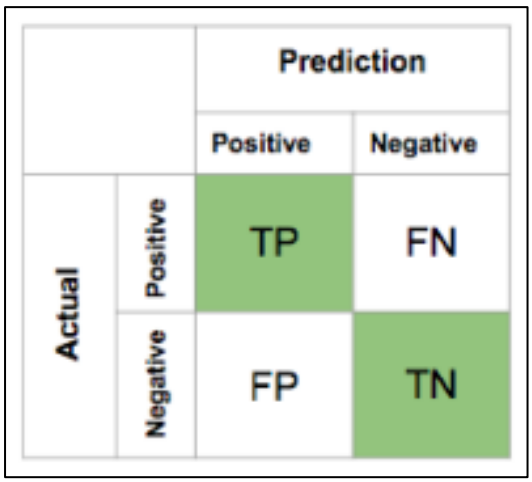

Figura 7. Matriz de confusión.

True Positive (TP): observaciones positivas, predichas positivas.

False Negative (FN): observaciones positivas, predichas negativas.

False Positive (FP): observaciones negativas, predichas positivas.

True Negative (TN): observaciones negativas, predichas negativas.

3.2.5. Métricas.

Accuracy (Acc): definimos la tasa de aciertos como la tasa de observaciones clasificadas correctamente por el clasificador.

$$Acc = \frac{Nª\ de\ aciertos\ clasificación}{Nº\ de\ observaciones} = \frac{TP + TN}{TP + FN + FP + TN}$$

Precisión (P): definimos la precisión como la tasa de observaciones predichas positivas clasificadas correctamente por el clasificador.

$$P = \frac{Nª\ de\ aciertos\ clase\ positiva}{Nº\ de\ observaciones\ predichas\ positivas} = \frac{TP}{TP + FP}$$

Sensibilidad o Recall (Se = R): definimos la sensibilidad como la tasa de observaciones positivas clasificadas correctamente por el clasificador.

$$Se\ = R = \frac{Nª\ de\ aciertos\ clase\ positiva}{Nº\ de\ observaciones\ positivas} = \frac{TP}{TP + FN}$$

Especificidad (Sp): definimos la especificidad como la tasa de observaciones negativas clasificadas correctamente por el clasificador.

$$Sp = \frac{Nª\ de\ aciertos\ clase\ negativa}{Nº\ de\ observaciones\ negativas} = \frac{TN}{FP + TN}$$

3.2.6. Curva ROC y AUC.

La curva ROC nos indica el rendimiento del clasificador, representa la variación de la sensibilidad o recall frente a la tasa de falsos positivos. Lo ideal para obtener un clasificador excelente sería que la curva estuviese lo más cerca posible de la esquina superior izquierda, es decir, que Se = R = 1 y FPR = 0, como se indica en la *Figura 8.*

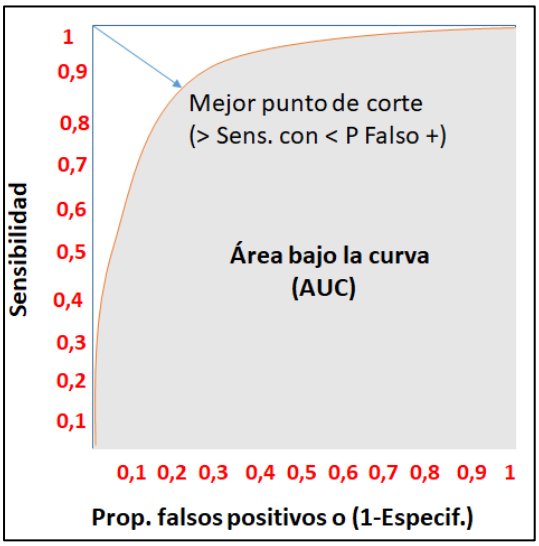

Figura 8. Curva ROC y AUC.

En el caso de AUC (Area Under Curve), nos permite medir el rendimiento global del clasificador resumido en todos sus puntos posibles de operación. El clasificador tendrá un rendimiento mayor cuanto más cerca de 1 esté.

4. Materiales.

En este apartado introduciremos nuestro conjunto de datos, describiendo las variables que lo componen y haremos un preprocesado de datos previo al entrenamiento de los diferentes modelos.

4.1. Descripción de la base de datos.

Los datos de nuestro proyecto han sido obtenidos de la web Kaggle (*www.kaggle.com*). El conjunto de datos se compone de 3.276 mediciones de muestras de agua, cada una de ellas se compone de nueve variables, las cuales determinarán la potabilidad o no de la muestra.

Las variables que componen el conjunto de datos son:

1. pH (potencial de hidrógeno):

El pH es un parámetro importante en la evaluación del equilibrio ácido-base del agua. También es el indicador de la condición ácida o alcalina del estado del agua. La Organización Mundial de la Salud recomienda un pH de entre 6,5 y 8,5.

Unidad de medida: N/A.

2. Dureza:

La dureza es causada principalmente por sales de calcio y magnesio. Estas sales se disuelven a partir de depósitos geológicos, este contenido depende de la geología del terreno por donde circula el agua. Originalmente, la dureza del agua se definía como una medida de la capacidad de una muestra de agua para precipitar jabón[1]. Hoy en día, la dureza se define como la suma de las concentraciones de calcio y magnesio.

Unidad de medida: mg/l.

3. Sólidos (Total dissolved solids - TDS):

El agua tiene la capacidad de disolver una amplia gama de minerales o sales, como potasio, calcio, sodio, bicarbonatos, cloruros, magnesio, sulfatos, etc, provocando un sabor no deseado. Este es uno de los parámetros más importantes para el uso del agua

[1] Cuando hay mucha cal (cuando el agua es muy dura), las moléculas de jabón empiezan a asociarse con estos iones y las micelas *(esferas en las que todas las cabezas están en contacto con el agua y todas las colas orientadas hacia el interior (las moléculas de jabón tienen una cola que repele el agua (hidrófoba) y una cabeza que es soluble en ella (hidrófila))* se rompen o no llegan a formarse. Por lo tanto, en zonas con aguas muy duras, los jabones formarán menos espuma.

(potable o no). Un valor alto de TDS indica que el agua está altamente mineralizada. El límite deseable de TDS es de 500 mg/l y el límite máximo es de 1000 mg/l.

Unidad de medida: ppm.

4. Cloraminas:

El cloro y la cloramina son los principales desinfectantes. Las cloraminas se forman con mayor frecuencia cuando se agrega amoníaco al cloro para tratar el agua. Los niveles de cloro de hasta 4 mg/l o 4 partes por millón (ppm) se consideran seguros en el agua potable.

Unidad de medida: ppm.

5. Sulfatos:

Los sulfatos son sustancias naturales que se encuentran en los minerales, también están presentes en el aire, aguas subterráneas, las plantas y los alimentos. La concentración de sulfatos en el agua de mar es de aproximadamente 2.700 mg/l. Varía de 3 a 30 mg/l en la mayoría de los suministros de agua dulce, aunque se encuentran concentraciones mucho más altas (1000 mg/l) en algunas zonas geográficas.

Unidad de medida: mg/l.

6. Conductividad:

El agua no es un buen conductor de la corriente eléctrica, más bien es un buen aislante. El aumento de la concentración de iones mejora la conductividad eléctrica del agua. Generalmente, la cantidad de sólidos disueltos en el agua determina la conductividad eléctrica. Según los estándares de la Organización Mundial de la Salud, el valor de la conductividad eléctrica no debe exceder los 400 µS/cm.

Unidad de medida: µS/cm (S = Siemens).

7. Carbono:

El carbono orgánico total (TOC, Total Organic Carbon del inglés) en el agua procede de la materia orgánica natural en descomposición, así como de fuentes sintéticas. TOC, es una medida de la cantidad total de carbono de compuestos orgánicos en agua. Según la Agencia de Protección Ambiental de EE. UU. los valores deben ser <2 mg/l en agua tratada/potable.

Unidad de medida: ppm.

8. Trihalometanos:

Los trihalometanos (THM) son sustancias químicas que se pueden encontrar en el agua tratada con cloro. La concentración de THM en el agua potable varía según el nivel de material orgánico presente en el agua, la cantidad de cloro necesaria para tratar el agua y la temperatura del agua que se está tratando. Niveles de THM de hasta 80 ppm se consideran seguros.

Unidad de medida: µg/l.

9. Turbidez:

La turbidez del agua depende de la cantidad de material sólido presente en suspensión. El valor recomendado por la OMS es inferior a 5,00 NTU (Unidades Nefelométricas de Turbidez). El instrumento usado para su medida es el nefelómetro o turbidímetro, que mide la intensidad de la luz dispersada a 90º cuando un rayo de luz pasa a través de una muestra de agua.

Unidad de medida: NTU.

10. Potabilidad:

Indica si el agua es segura para el consumo humano. El valor 1 indica que el agua es potable y 0 que no lo es.

4.2. Análisis de datos.

Se presentará a continuación mediante un análisis global de los predictores las características principales, en nuestro caso todos los predictores son numéricos. En primer lugar, para el desarrollo de nuestro trabajo se obviaron las muestras que presentaban, en alguna de las nueve variables, valores faltantes. Esta técnica es conocida como *listwise deletion,* tras aplicar esta técnica nuestro conjunto de datos pasa a tener 2.011 muestras de agua (se dividirá el conjunto de datos en train y test, 80% y 20% respectivamente).

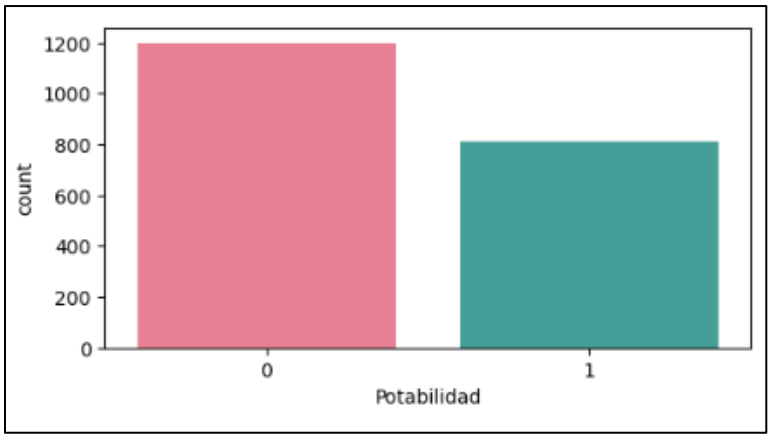

Figura 9. Muestras por clase.

En la *Figura 9*, se muestra la cantidad de muestras por clase. En la siguiente figura, *Figura 10*, mostramos la distribución de clases.

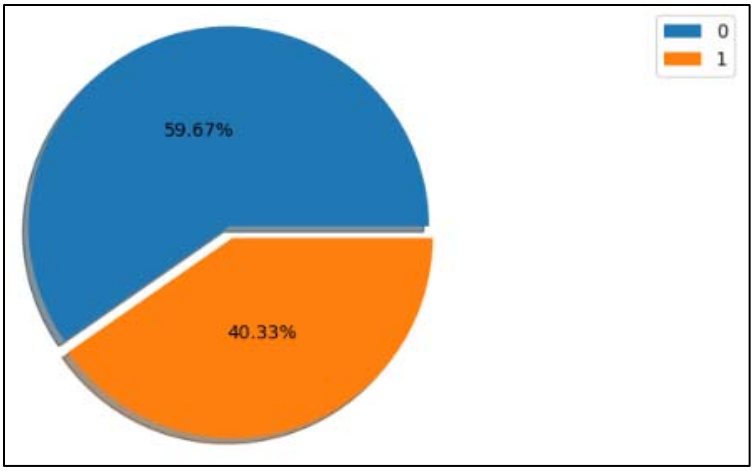

Figura 10. Porcentaje de muestras por clase.

A continuación, en la *Tabla 1,* se muestran algunos datos estadísticos de los predictores:

	pH	Dureza	TDS	Cloraminas	Sulfatos	Conductividad	TOC	THM	Turbidez	Potabilidad
count	1608.00	1608.00	1608.00	1608.00	1608.00	1608.00	1608.00	1608.00	1608.00	1608.00
mean	7.08	195.82	21731.84	7.10	333.82	425.17	14.32	66.63	3.99	0.40
std	1.57	32.75	8591.18	1.56	40.71	81.05	3.31	16.06	0.77	0.49
min	0.23	73.49	320.94	1.92	129.00	201.62	2.20	8.58	1.50	0.00
25%	6.10	176.65	15416.92	6.12	308.11	364.83	12.13	56.35	3.49	0.00
50%	7.04	197.10	20541.32	7.13	332.75	422.00	14.30	66.63	3.99	0.00
75%	8.04	216.12	26958.36	8.09	359.84	479.94	16.65	77.32	4.53	1.00
max	14.00	317.34	56488.67	13.04	481.03	753.34	27.01	124.00	6.49	1.00

Tabla 1. Descripción de los datos.

De los datos descritos, se observa como los datos de las variables *pH, Cloraminas y Turbidez* están agrupados en torno a su media (valores en un rango de datos pequeño). También observamos que en gran parte de las variables los valores de la media y mediana coinciden, lo que nos hace concluir que gran parte de los datos siguen una distribución simétrica.

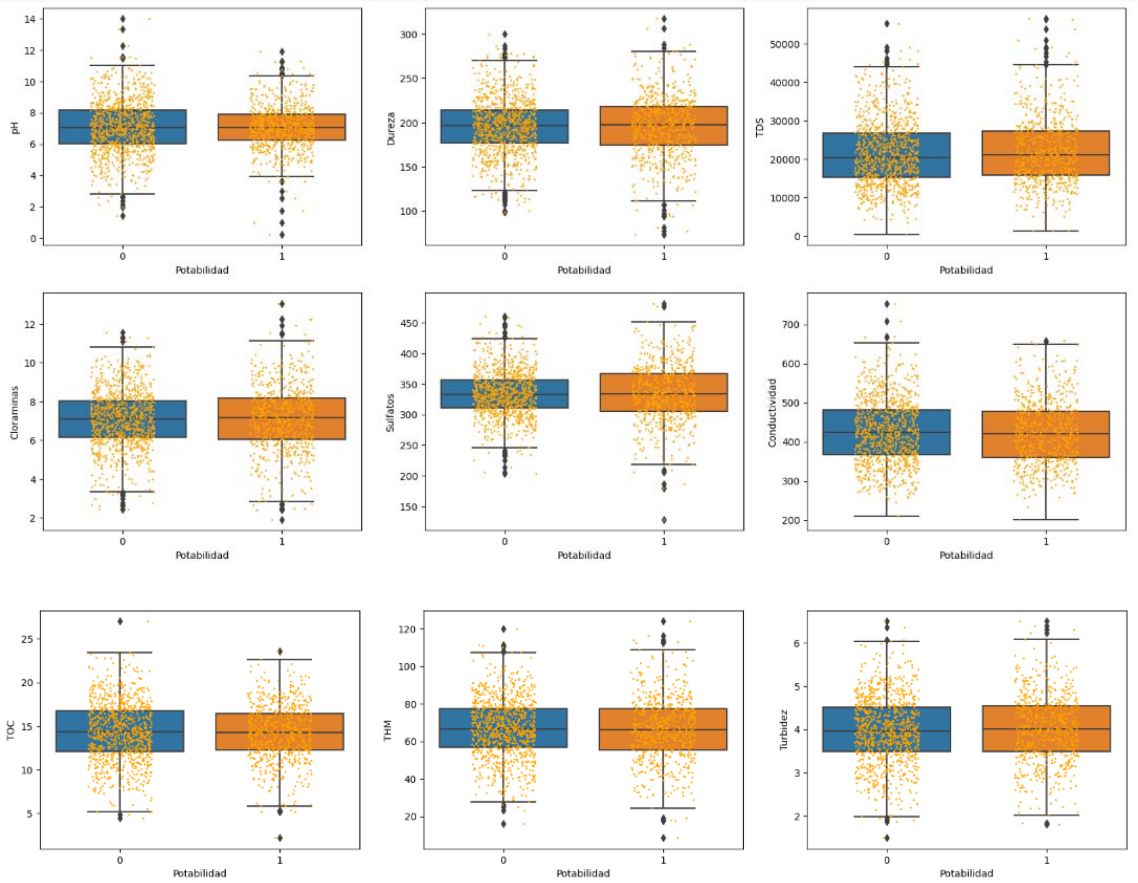

Figura 11. Diagrama de cajas por clase.

En la *Figura 11*, se representan los diagramas de caja de cada una de las variables por clase. Como se observa en los diagramas, indicado en el párrafo anterior, todas las variables se distribuyen de forma normal simétrica, sin observar sesgos, a excepción de la variable TDS que presenta un ligero sesgo, podemos comprobar esto en la T*abla 1,* ya que el valor de la media y mediana no son coincidentes, en este caso tenemos asimetría positiva. Observamos también, en cada una de las variables, homogeneidad en los datos. Además, no se observan cantidad elevada de valores atípicos (valores fuera de los límites inferior y superior).

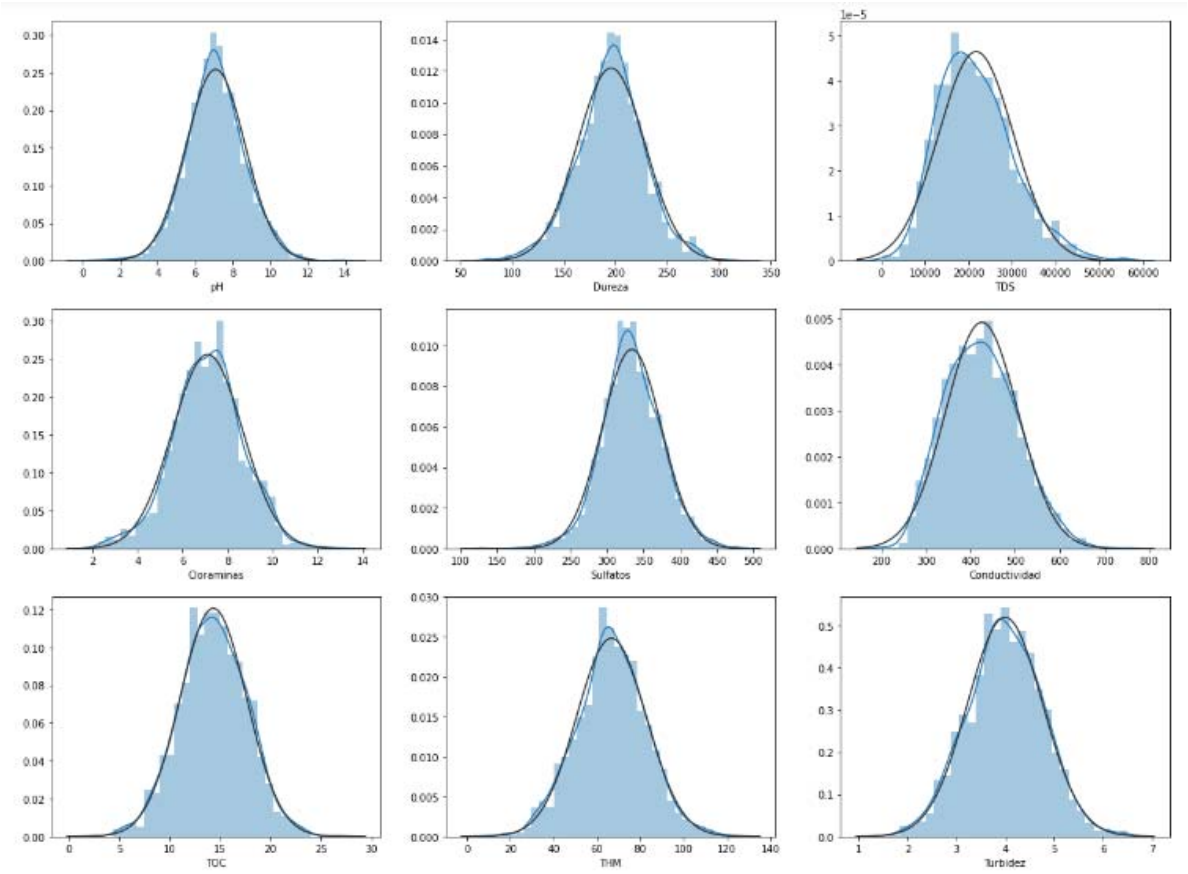

Figura 12. Representación de distribución normal.

En la *Figura 12*, vemos la distribución de cada una de las variables, así como la distribución normal. Observando el caso de la variable TDS se comprueba el sesgo que detectamos anteriormente en los diagramas de cajas.

Comprobaremos si existe correlación entre pares de variables, para ello representaremos la matriz de correlaciones con objeto de conocer el grado de correlación de las variables disponibles y la influencia que pudieran tener estas en nuestro trabajo.

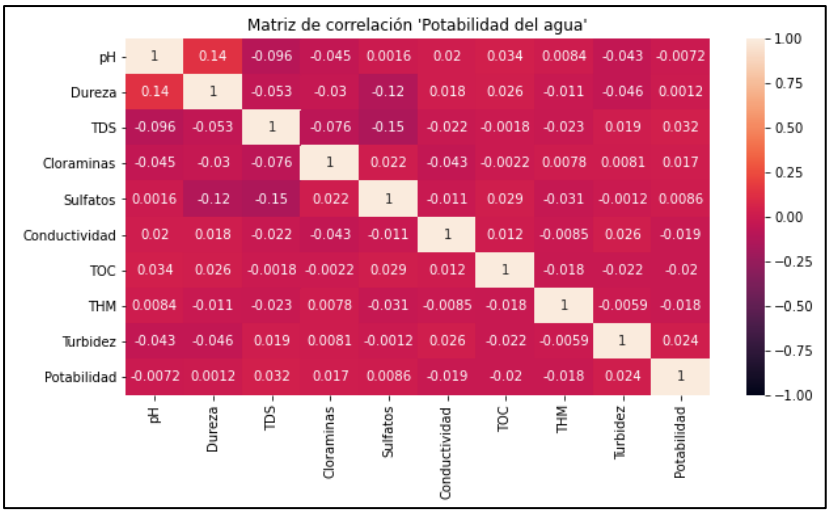

Figura 13. Matriz de correlación de variables.

De la matriz anterior, *Figura 13,* podemos vemos que existe una correlación mayor entre variables Sulfatos-Dureza, Sulfatos-TDS y Dureza-pH, aunque la correlación es baja. En el caso de Sulfatos-Dureza y Sulfatos-TDS la correlación es negativa, y en el caso Dureza-pH es positiva. No se observa correlación entre las variables de entrada y salida.

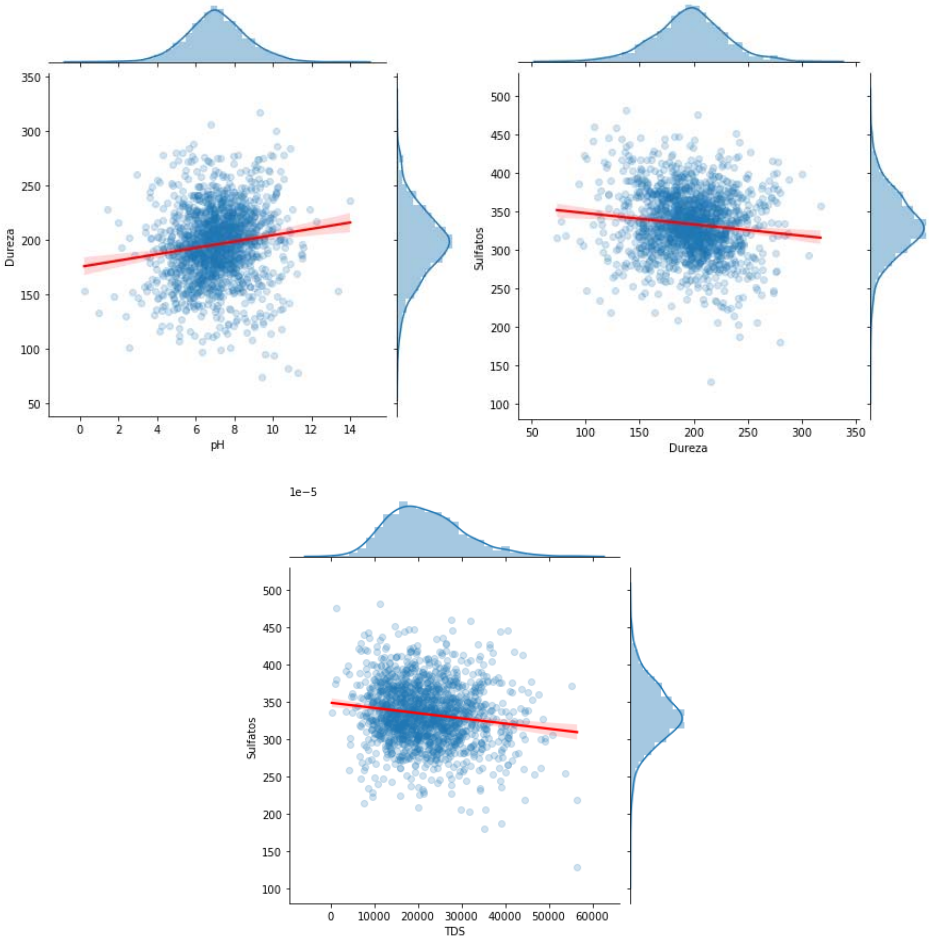

Figura 14. Correlación de variables.

En la *Figura 14,* hemos representados los pares de variables que tenían mayor correlación según la matriz de correlación. La correlación de los dos pares de variables Sulfatos-Dureza y Sulfatos-TDS es lineal, baja y negativa, de lo cual se puede observar que al aumentar el valor de Sulfatos disminuye la Dureza y TDS. En el caso de las variables Dureza-pH la correlación es positiva.

4.3. Estandarización de datos.

Analizadas las características principales de los predictores, es recomendable normalizar los datos con el objetivo de evitar que las variables que toman valores grandes dominen al resto. En nuestro caso, se han normalizado los datos mediante el método *Z-Score*, consistente en la transformación de las variables en otras que poseen media aritmética 0 y desviación típica 1, mediante la siguiente expresión:

$$z_i = \frac{x_i - \bar{x}}{S} \qquad i = 1, \ldots, n$$

Donde x y S representan, respectivamente, la media y desviación típica. También es conocida como estandarización, y es usada cuando hay valores *outlier*, de forma que evite una distorsión de los resultados.

La estandarización del conjunto de datos se hace teniendo en cuenta la división del conjunto de entrenamiento y test.

4.4. Importancia de predictores.

Previamente a la construcción de los modelos de evaluación se estudiará la importancia de las variables del conjunto de datos. Para ello, se ha hecho uso del método Random Forest, de tal manera que podamos cuantificar la importancia de las variables y como método de validación se ha aplicado *Out-Of-Bag Error*. *Bagging* hace referencia al empleo de muestro repetido con reposición con el objetivo de reducir la varianza de los modelos. De forma general, se generan *pseudo-muestras* con las que ajustar el modelo y posteriormente se hace la media de las predicciones. En el proceso de *bagging* el ajuste del modelo se hace solo con aproximadamente dos tercios de las observaciones, al tercio restante se le conoce como *Out-Of-Bag.* Los valores numéricos obtenidos son:

pH	0,140
Dureza	0,114
TDS	0,112
Cloraminas	0,123
Sulfatos	0,139
Conductividad	0,091
TOC	0,093
THM	0,096
Turbidez	0,092

Tabla 2. Valores de importancia de predictores.

Representando estos valores, se obtiene:

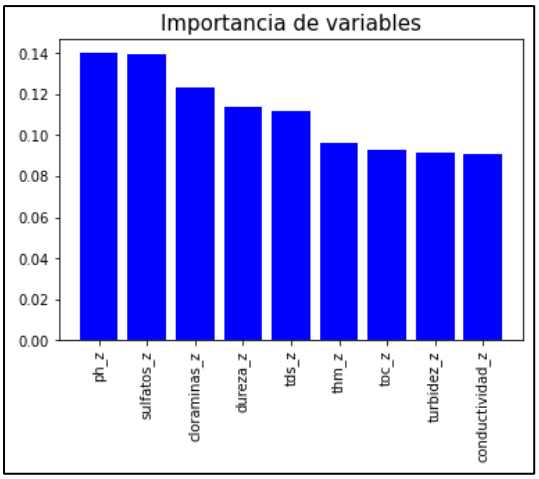

Figura 15. Importancia de variables en el conjunto de datos.

De la *Figura 15*, observamos que no hay una variable destacada en cuanto a importancia, tenemos una diferencia entre la variable de mayor y menor importancia del 4% aproximadamente, por lo que *a priori* no podemos afirmar que haya una influencia predominante de alguna de los predictores en la variable respuesta. En nuestro trabajo se considerarán todas las variables.

5. Experimentación.

Tras el análisis y preprocesado de nuestro conjunto de datos, a continuación, entrenaremos y evaluaremos los diferentes modelos.

5.1. Descripción de metodología.

En este apartado describiremos la metodología seguida en nuestra experimentación. Para el desarrollo de nuestro proyecto se han usado diferentes modelos de clasificación, preparados para trabajar con variables numéricas. Los modelos son:

1.- K-NN (K-Nearest Neighbors).

2.- Naive-Bayes.

3.- Random Forest.

4.- SVM (Support Vector Machine)

Generaremos los modelos y los resultados siguiendo los pasos descritos a continuación:

1.- Aplicación del modelo entrenado en el conjunto de test. El modelo da la probabilidad de que la instancia bajo consideración sea de la clase positiva (agua no potable).

2.- Generación de curva ROC, mediante un barrido de umbrales y cálculo del área bajo la curva AUC.

3.- Para el umbral óptimo de la curva (punto más cercano al punto (0,1)), cálculo de la matriz de confusión, tasa de acierto, sensibilidad y especificidad.

Las metodologías de evaluación empleadas han sido: validación cruzada (Cross-Validation) y su caso particular, *Leave-One-Out*.

5.2. Conjunto de datos.

Nuestro conjunto de datos disponible se compone de 2.011 muestras, de las cuales el 80% (1.608) corresponderán al conjunto de entrenamiento y el 20% (403) restante al conjunto de test, la metodología de evaluación principal ha sido mediante *Hold-Out.*

El conjunto de entrenamiento se compone de un 59,67% de muestras “no potable” y un 40,33% de muestras “potable”. Como se indicó en apartados anteriores, el conjunto de datos disponible se modificó, eliminando aquellas muestras que presentaban valores faltantes en algunas de las muestras, para nuestro conjunto de datos se ha mantenido la proporción de clases del conjunto de datos original.

En nuestro proyecto, la clase positiva se corresponde con las muestras “no potable”, mientras que la clase negativa se corresponde con las muestras “potable”.

5.3. Selección y ajuste de modelos.

En este apartado describiremos el ajuste realizado a cada uno de los modelos empleados, con objeto de optimizar los hiperparámetros de cada uno de los modelos (siempre y cuando tenga parámetros a optimizar). Tras la optimización de los hiperparámetros, se evalúa el modelo mediante validación cruzada (*K-fold cross validation*) y su variante, *Leave-One-Out*.

Para la evaluación de los modelos se ha empleado la función *GridSearchCV* de *Python*, incluida en la librería *Scikit-learn.*

Los modelos que necesitan optimización de parámetros son:

K-NN (K-Nearest Neighbors).

Los hiperparámetros a ajustar son el número de vecinos (K) y la métrica de distancia.

- K: número de vecinos elegidos para la evaluación del modelo.

Para el caso de *K-fold cross validation* el k óptimo es 19 y para *Leave-One-Out* es 28.

- Métrica de distancia: para determinar qué muestras están más cerca de la instancia que se está evaluando será necesario medir la distancia entre ellas. Esta métrica ayuda a formar los límites de decisión. Podemos elegir entre Euclidiana, Minkowski o Manhattan, entre otras.

En el caso de la métrica de distancia, tanto para *K-fold cross validation* como para *Leave-One-Out* la métrica es Manhattan.

Random Forest.

Los hiperparámetros a ajustar en este caso son el número de árboles del bosque, número mínimo de muestras que debe haber en un nodo y *ccp_alpha* (se trata de un parámetro de complejidad usado para el *Minimal Cost-Complexitiy Pruning*)

- Número de árboles: número de árboles que contendrá el modelo.

Para el caso de *K-fold cross validation* el número de árboles óptimo es 88 y para *Leave-One-Out* es 600.

- Número mínimo de muestras en un nodo:

Para el caso de *K-fold cross validation* y *Leave-One-Out* el valor óptimo es 0.05 (expresado en porcentaje).

- Parámetro de complejidad, ccp_alpha.

En ambos casos, *K-fold cross validation* y *Leave-One-Out,* el valor optimo es 0.

En ambos métodos de evaluación, la métrica/criterio empleado para encontrar las regiones es el *índice de Gini,* parámetro por defecto de *Python*. Este índice cuantifica la variabilidad de las clases de las observaciones en una misma región.

SVM (Support Vector Machine)

En el caso de la máquina de vector soporte, se ha empleado el kernel lineal y radial.

- Parámetro de ajuste C: se trata de un parámetro para cuantificar el grado de violación del margen que le permitimos a las observaciones. Un valor elevado de C implica un margen mayor y por tanto una mayor tolerancia a que las observaciones estén clasificadas incorrectamente, en cambio, un valor bajo de C reduce el margen, y la tolerancia a que las observaciones se clasifiquen en un lugar incorrecto también disminuye. Este parámetro es utilizado en los kernel indicados anteriormente.

En el caso del kernel lineal, tanto en *K-fold cross validation* como en *Leave-One-Out* el valor óptimo de C es 25. Para el kernel radial, en *K-fold cross validation* es 7,14 y *Leave-One-Out* 16,67.

- Parámetro Gamma: coeficiente del kernel, usado en el kernel radial. Cuantifica la influencia de la muestra.

Para el kernel radial, en *K-fold cross validation* es 0,071 y *Leave-One-Out* 0,16.

5.4. Aplicación y generación de resultados.

En el apartado anterior hemos indicado los valores óptimos de los hiperparámetros de cada uno de los modelos que se han aplicado en nuestro proyecto. A continuación, generaremos y mostraremos los resultados de la evaluación.

K-NN (K-Nearest Neighbors).

Representamos la curva ROC, esta representa la variación de la sensibilidad (recall) frente a la tasa de falsos positivos, para la evaluación mediante *K-fold cross-validation* y *Leave-One-Out.*

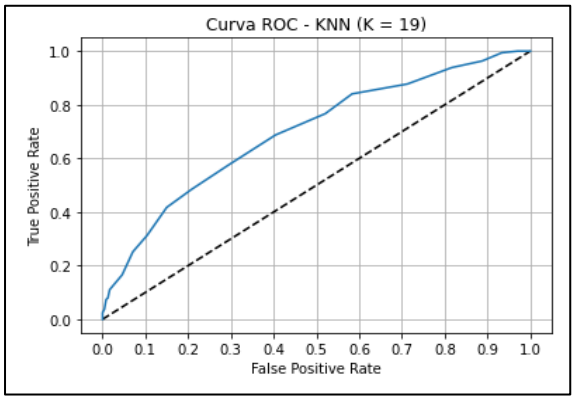

Figura 16. Curva ROC (validación cruzada).

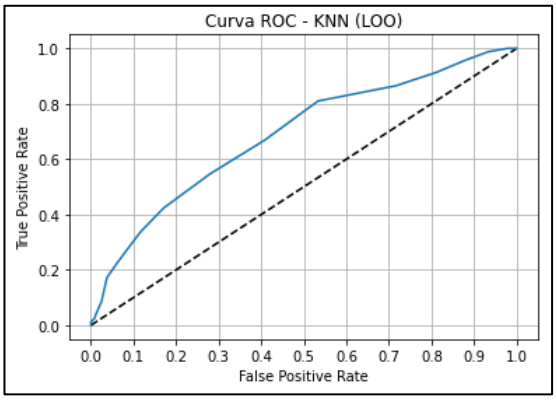

Figura 17. Curva ROC (Leave-One-Out).

A continuación, mostramos las matrices de confusión obtenidas en el punto óptimo de las curvas.

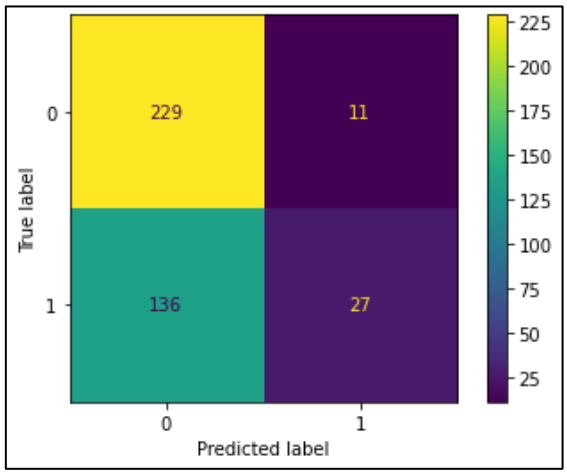

Figura 18. Matriz de confusión (validación cruzada).

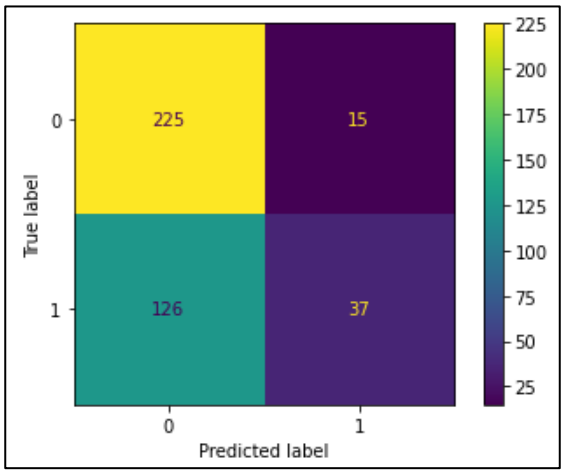

Figura 19. Matriz de confusión (Leave-One-Out).

En la siguiente tabla, *Tabla 3*, exponemos los resultados de las métricas obtenidas y los resultados de AUC.

Método de evaluación	AUC	Tasa de acierto (Acc)	Sensibilidad	Especificidad
Cross-Validation	0,70	0,64	0,63	0,71
Leave-One-Out	0,69	0,65	0,64	0,71

Tabla 3. Resultados de evaluación para K-NN

Naive-Bayes.

La evaluación del modelo Naive-Bayes es probable que sea la más sencilla si la comparamos con el resto de los clasificadores empleados en nuestro proyecto.

Representamos la curva ROC, esta representa la variación de la sensibilidad (recall) frente a la tasa de falsos positivos.

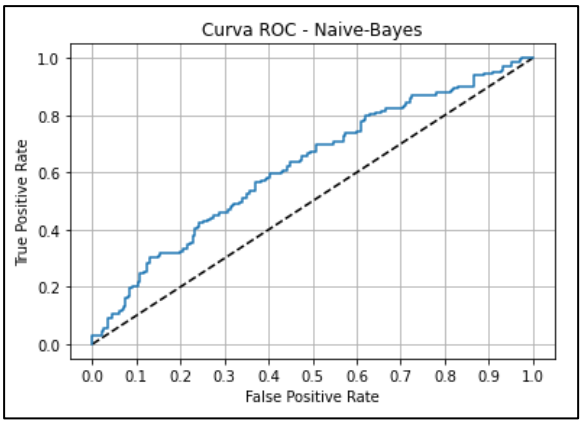

Figura 20. Curva ROC.

A continuación, mostramos la matriz de confusión obtenida en el punto óptimo de la curva.

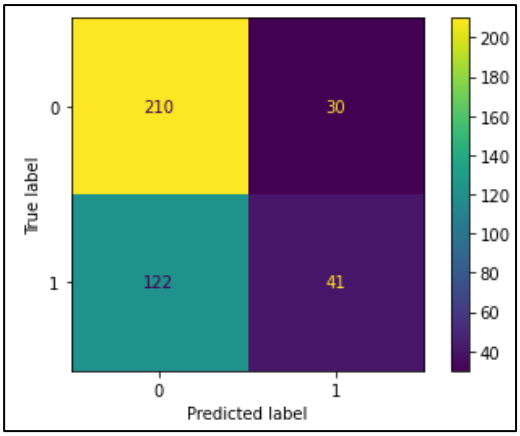

Figura 21. Matriz de confusión.

En la siguiente tabla, *Tabla 4*, exponemos los resultados de las métricas obtenidas y los resultados de AUC.

Naive-Bayes	AUC	Tasa de acierto (Acc)	Sensibilidad	Especificidad
Resultados	0,62	0,62	0,63	0,58

Tabla 4. Resultados de evaluación para Naive-Bayes.

Random Forest.

Representamos la curva ROC, esta representa la variación de la sensibilidad (recall) frente a la tasa de falsos positivos, para la evaluación mediante *K-fold cross-validation* y *Leave-One-Out.*

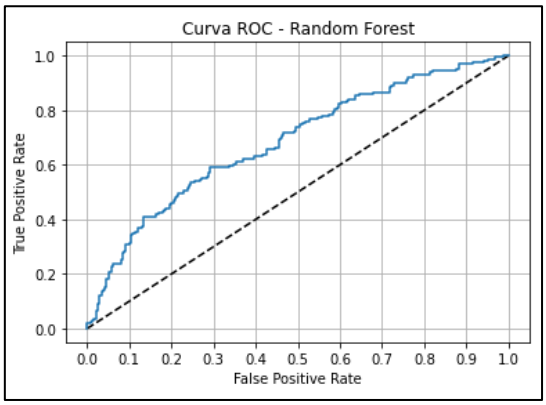

Figura 22. Curva ROC (validación cruzada).

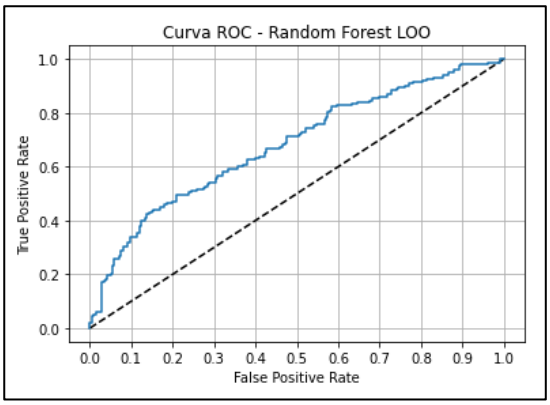

Figura 23. Curva ROC (Leave-One-Out).

A continuación, mostramos las matrices de confusión obtenidas en el punto óptimo de las curvas.

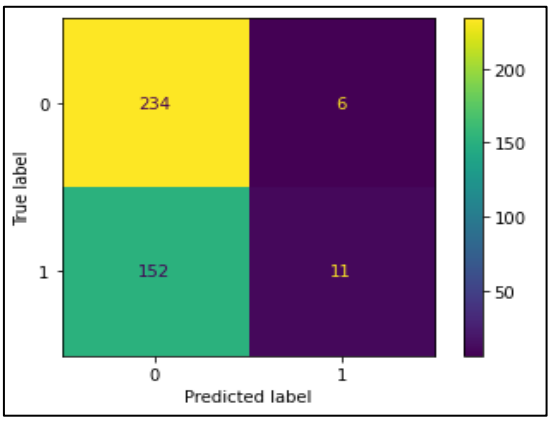

Figura 24. Matriz de confusión (validación cruzada).

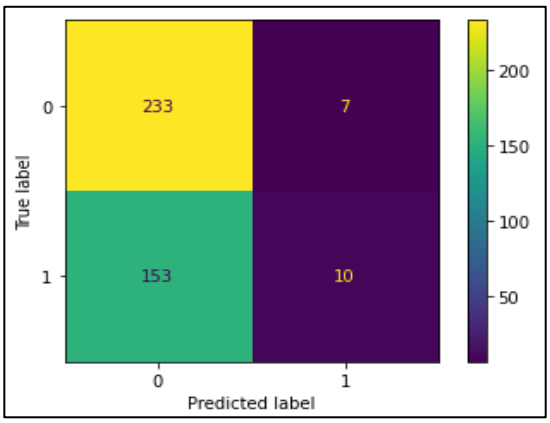

Figura 25. Matriz de confusión (Leave-One-Out).

En la siguiente tabla, *Tabla 5*, exponemos los resultados de las métricas obtenidas y los resultados de AUC.

Método de evaluación	AUC	Tasa de acierto (Acc)	Sensibilidad	Especificidad
Cross-Validation	0,68	0,61	0,61	0,65
Leave-One-Out	0,68	0,60	0,60	0,59

Tabla 5. Resultados de evaluación para Random Forest.

SVM (Support Vector Machine) – Kernel radial.

Representamos la curva ROC, esta representa la variación de la sensibilidad (recall) frente a la tasa de falsos positivos, para la evaluación mediante *K-fold cross-validation* y *Leave-One-Out.*

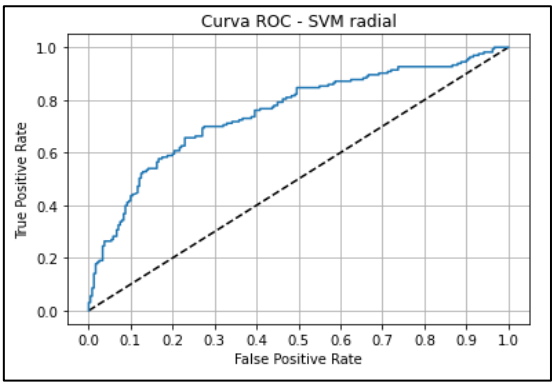

Figura 26. Curva ROC (validación cruzada).

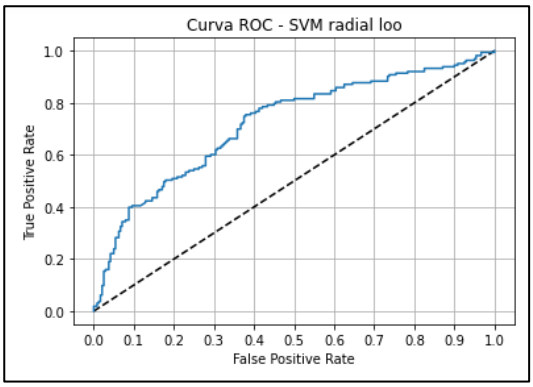

Figura 27. Curva ROC (Leave-One-Out).

A continuación, mostramos las matrices de confusión obtenidas en el punto óptimo de las curvas.

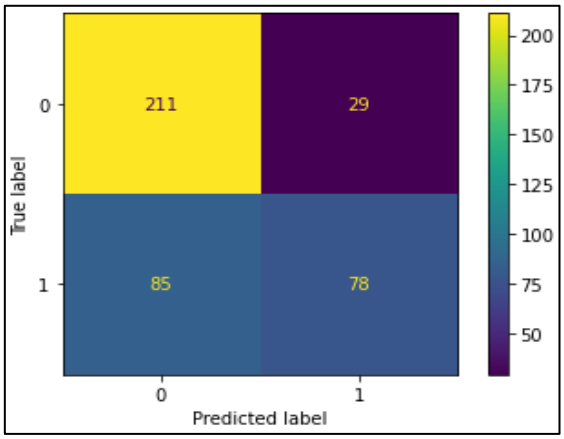

Figura 28. Matriz de confusión (validación cruzada).

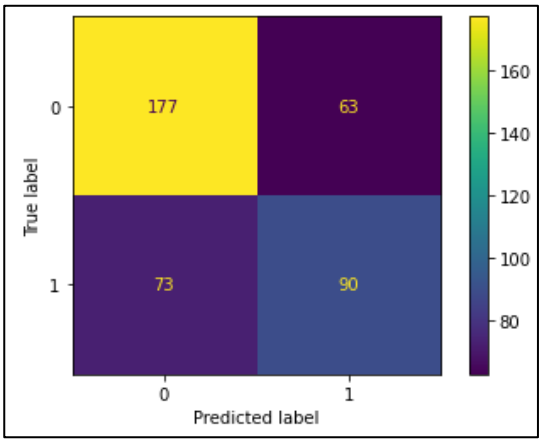

Figura 29. Matriz de confusión (Leave-One-Out).

En la siguiente tabla, *Tabla 6*, exponemos los resultados de las métricas obtenidas y los resultados de AUC.

Método de evaluación	AUC	Tasa de acierto (Acc)	Sensibilidad	Especificidad
Cross-Validation	0,75	0,72	0,71	0,73
Leave-One-Out	0,72	0,66	0,71	0,59

Tabla 6. Resultados de evaluación para SVM – Kernel radial.

5.5. Análisis de resultados.

En la siguiente tabla, *Tabla 7,* se resumen los resultados obtenidos de cada clasificador con la mejor técnica de evaluación.

Clasificador	Tasa de acierto (Acc)	AUC	Sensibilidad	Especificidad
K-NN (CV)	0,64	0,70	0,63	0,71
Naive-Bayes	0,62	0,62	0,63	0,58
Random Forest (CV)	0,61	0,68	0,61	0,65
SVM – Kernel radial (CV)	0,72	0,75	0,71	0,73

Tabla 7. Comparativa de resultados de clasificadores.

Se observa de la *Tabla 7* que el clasificador que mayor rendimiento proporciona es el SVM con Kernel radial y evaluación mediante validación cruzada. Este clasificador es el que mejor AUC, tasa de acierto, sensibilidad y especificidad proporciona. De todos los modelos estudiados, es que menor error comete al clasificar las clases positiva y negativa (mayores valores de sensibilidad y especificidad obtenemos).

-Tasa de aciertos es del 72%, esto es, del total de instancias, el clasificador predice correctamente el 72%.

- Sensibilidad del 71%: del total de instancias de agua no potable, el clasificador predice correctamente el 71%.

- Especificidad del 73%: del total de instancias de agua potable, el clasificador predice correctamente el 73%.

Estos datos han sido obtenidos en el punto óptimo de la curva ROC. Es posible trabajar en otros puntos de operación de interés, seleccionando otros umbrales, para minimizar los falsos negativos, especialmente peligroso (con instancias clasificadas como de agua potable, cuando en realidad no lo son). El coste de intentar minimizar estos falsos negativos es trabajar con una especificidad demasiado baja: el porcentaje de instancias clasificados como de agua no potable disminuiría, aumentando de esta forma los falsos positivos.

6. Conclusiones.

En nuestro proyecto hemos revisado, estudiado y comparado diferentes métodos de aprendizaje supervisado para la clasificación de instancias de un conjunto de datos, hemos desarrollado y optimizado los parámetros de funcionamiento de los diferentes métodos para lograr obtener un clasificador óptimo y hemos realizado el preprocesado de datos, clave en la obtención de buenos resultados e importante a la hora de construir modelos e interpretar características del conjunto de datos.

Tras el entrenamiento y evaluación de cada uno de los modelos, hemos obtenido un clasificador visiblemente mejor que el resto, SVM con Kernel radial. Los resultados conseguidos con el clasificador SVM con Kernel radial y comparado con alguno de los modelos evaluados varían de manera significativa.

Para futuras líneas de investigación, podemos distinguir dos vías para la mejora y desarrollo de nuestro trabajo. En primer lugar, imputar valores NaN, lo que resultaría en un estudio de un caso real y, en segundo lugar, hacer uso de otros métodos de clasificación, como redes neuronales.

Con todo ello, el aprendizaje automático y en concreto el aprendizaje supervisado están muy presentes en nuestra vida cotidiana y son cada vez más necesarios, ya que ayudan a mejorar el desarrollo socioeconómico y energético de la sociedad en general.

Personalmente, el desarrollo de este trabajo supone dar mayor importancia, más de la que ya tenía para mí, a la investigación con herramientas de *machine learning*, de manera que no solo se contribuya al desarrollo económico de la sociedad sino al desarrollo social de la misma, llegando a la obtención de resultados que, sin el uso de dichas herramientas y en determinados sectores, podrían llegar a ser críticos para las personas (principalmente en el sector sanitario).

7. Bibliografía.

[1] https://www.bancomundial.org/es/topic/water/overview

[2] https://www.un.org/es/global-issues/water

[3] Andreas C. Müller and Sarah Guido (2016), *Introduction to Machine Learning with Python,* First Edition, O'Reilly Media.

[4] John Paul Mueller and Luca Massaron (2016), *Machine Learning for Dummies,* John Wiley & Sons.

[5] Chris Albon (2018), *Machine Learning with Python Cookbook*, O'Reilly Media.

[6] Sebastian Raschka, Vahid Mirjalili (2017), *Python Machine Learning*, Second Edition, Packt Publishing.

[7] José Luis Ruiz Reina, *Evaluación de modelos,* Dpto. Ciencias de la Computación e Inteligencia Artificial, Universidad de Sevilla.

[8] https://www.cienciadedatos.net

[9] Diego Marín Santos, Manuel E. Gegúndez Arias, *Fundamentos de Aprendizaje Automático,* Universidad de Huelva.

[10] Sanaa Kaddoura (2022), *Evaluation of Machine Learning Algorithm on DrinkingWater Quality for Better Sustainability,* MDPI.

[11] Amir Hamzeh Haghiabi, Ali Heidar Nasrolahi and Abbas Parsaie (2018), *Water quality prediction using machine learning methods*, Water Quality Research Journal.

[12] Jinal Patel, Charmi Amipara, Tariq Ahamed Ahanger , Komal Ladhva, Rajeev Kumar Gupta, Hashem O. Alsaab , Yusuf S. Althobaiti and Rajnish Ratna, *A Machine Learning-Based Water Potability Prediction Model by Using Synthetic Minority Oversampling Technique and Explainable AI,* Hindawi.

[13] Quansheng Kuang, and Lei Zhao, *A Practical GPU Based KNN Algorithm,* School of Computer Science and Technology, Soochow University, China. Proceedings of the Second Symposium International Computer Science and Computational Technology.

[14] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. *An Introduction to Statistical Learning with Applications in R.* Springer.

Anexos.

Código generado para determinación de importancia de predictores:

```
from sklearn.ensemble import RandomForestClassifier
X = z_train # definimos X como conjunto de atributos.
Y = df_train['Potabilidad'] # definimos Y como la variable de
salida.
forest = RandomForestClassifier(n_estimators=500,random_state=1)
forest.fit(X, Y)
print("pH:", forest.feature_importances_.round(3)[0],
     "\nDureza:", forest.feature_importances_.round(3)[1],
     "\nTDS:", forest.feature_importances_.round(3)[2],
     "\nCloraminas:", forest.feature_importances_.round(3)[3],
     "\nSulfatos:", forest.feature_importances_.round(3)[4],
     "\nConductividad:", forest.feature_importances_.round(3)[5],
     "\nTOC:", forest.feature_importances_.round(3)[6],
     "\nTHM:", forest.feature_importances_.round(3)[7],
     "\nTurbidez:", forest.feature_importances_.round(3)[8])
```

Códigos generados de clasificadores:

K-NN Cross-Validation:

```
knn_cv = GridSearchCV(estimator = KNeighborsClassifier(),
                      param_grid = parameters,
                      cv = 10,
                      scoring = 'accuracy',
                      n_jobs = -1,
                      refit = True)

knn_cv.fit(X = X_train, y = Y_train)

print (knn_cv.best_score_, knn_cv.best_params_)

knn = knn_cv.best_estimator_
knn.fit(X_train, Y_train)
Acc_training_knn = knn.score(X_train, Y_train)
Acc_test_knn = knn.score(X_test, Y_test)
print('\nTraining CV:{0:0.4f}'.format(Acc_training_knn),'\nTest
CV:{0:0.4f}\n'.format(Acc_test_knn))
```

K-NN Leave-One-Out:

```
knn_cv_LOO = GridSearchCV(estimator = KNeighborsClassifier(),
                          param_grid = parameters,
                          cv = LeaveOneOut(),
                          scoring = 'accuracy',
                          n_jobs = -1,
                          refit = True)

knn_cv_LOO.fit(X = X_train, y = Y_train)
print (knn_cv_LOO.best_score_, knn_cv_LOO.best_params_)
knn_LOO = knn_cv_LOO.best_estimator_
knn_LOO.fit(X_train, Y_train)
Acc_training_knn_LOO = knn_LOO.score(X_train, Y_train)
Acc_test_knn_LOO = knn_LOO.score(X_test, Y_test)
print('\nTraining
LOO:{0:0.4f}'.format(Acc_training_knn_LOO),'\nTest
LOO:{0:0.4f}'.format(Acc_test_knn_LOO))
```

Naive-Bayes:

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, Y_train)
Acc_training_gnb = gnb.score(X_train, Y_train)
Acc_test_gnb = gnb.score(X_test, Y_test)
print('\nTraining:{0:0.4f}'.format(Acc_training_gnb),'\nTest:{0:0.4
f}'.format(Acc_test_gnb))
```

Random Forest Cross-Validation:

```
parameters = {'n_estimators':    [int(x) for x in
np.linspace(start=30, stop=600, num=30)],
              'max_depth'   :     [10, 20, 30],
              'min_samples_leaf': [0.05, 0.1],
              'max_features':     ['sqrt', 'log2'],
              'ccp_alpha':        [0, 0.001, 0.009]
             }

rf_cv1 = GridSearchCV(estimator =
RandomForestClassifier(random_state = 42),
                      param_grid = parameters,
                      scoring="accuracy",
                      cv=5,
                      n_jobs = -1,
                      refit = True)
```

```
rf_cv1.fit(X = X_train, y = Y_train)
print (rf_cv1.best_score_, rf_cv1.best_params_)
rf1 = rf_cv1.best_estimator_

rf1.fit(X_train, Y_train)
Acc_training_rf = rf1.score(X_train, Y_train)
Acc_test_rf = rf1.score(X_test, Y_test)
print('\nTraining:{0:0.4f}'.format(Acc_training_rf),'\nTest:{0:0.4f
}'.format(Acc_test_rf))
```

Random Forest Leave-One-Out:

```
parameters = {'n_estimators':    [int(x) for x in
np.linspace(start=30, stop=600, num=2)],
              'max_depth'   :     [20],
              'min_samples_leaf': [0.05],
              'max_features':     ['sqrt'],
              'ccp_alpha':        [0]
             }

rf_loo = GridSearchCV(estimator =
RandomForestClassifier(random_state = 42),
                      param_grid = parameters,
                      scoring = "accuracy",
                      cv = LeaveOneOut(),
                      n_jobs = -1,
                      refit = True)

rf_loo.fit(X = X_train, y = Y_train)
print (rf_loo.best_score_, rf_loo.best_params_)
rf2 = rf_loo.best_estimator_

rf2.fit(X_train, Y_train)
Acc_training_rf_loo = rf2.score(X_train, Y_train)
Acc_test_rf_loo = rf2.score(X_test, Y_test)
print('\nTraining:{0:0.4f}'.format(Acc_training_rf_loo),'\nTest:{0:
0.4f}'.format(Acc_test_rf_loo))
```

SVM lineal Cross-Validation:

```
parameters = {'C': np.linspace(0, 50, 3)}


svm_lineal_cv = GridSearchCV(estimator = SVC(kernel =
'linear',random_state = 301, probability = True),
                             param_grid = parameters,
                             cv = 10,
                             scoring = 'accuracy',
                             n_jobs = -1,
                             refit = True)

svm_lineal_cv.fit(X = X_train, y = Y_train)
print (svm_lineal_cv.best_score_, svm_lineal_cv.best_params_)

lineal = svm_lineal_cv.best_estimator_

lineal.fit(X_train, Y_train)
Acc_training_lineal = lineal.score(X_train, Y_train)
Acc_test_lineal = lineal.score(X_test, Y_test)
print('\nTraining:{0:0.4f}'.format(Acc_training_lineal),'\nTest:{0:
0.4f}'.format(Acc_test_lineal))
```

SVM lineal Leave-One-Out:

```
parameters = {'C': np.linspace(0, 50, 3)}

svm_lineal_cv_loo = GridSearchCV(estimator = SVC(kernel =
'linear',random_state = 302, probability = True),
                                 param_grid = parameters,
                                 cv = LeaveOneOut(),
                                 scoring = 'accuracy',
                                 n_jobs = -1,
                                 refit = True)

svm_lineal_cv_loo.fit(X = X_train, y = Y_train)
print (svm_lineal_cv_loo.best_score_,
svm_lineal_cv_loo.best_params_)

lineal_loo = svm_lineal_cv_loo.best_estimator_

lineal_loo.fit(X_train, Y_train)
Acc_training_lineal_loo = lineal_loo.score(X_train, Y_train)
Acc_test_lineal_loo = lineal_loo.score(X_test, Y_test)
print('\nTraining:{0:0.4f}'.format(Acc_training_lineal_loo),'\nTest
:{0:0.4f}'.format(Acc_test_lineal_loo))
```

SVM radial Cross-Validation:

```
parameters = {'C': np.linspace(0, 50, 8),
              'gamma': np.linspace(0, 0.5, 8)}

svm_radial_cv = GridSearchCV(estimator = SVC(kernel = 'rbf',
random_state = 310, probability = True),
                                    param_grid = parameters,
                                    cv = 10,
                                    scoring = 'accuracy',
                                    n_jobs = -1,
                                    refit = True
                             )

svm_radial_cv.fit(X = X_train, y = Y_train)
print (svm_radial_cv.best_score_, svm_radial_cv.best_params_)

radial = svm_radial_cv.best_estimator_

radial.fit(X_train, Y_train)
Acc_training_radial = radial.score(X_train, Y_train)
Acc_test_radial = radial.score(X_test, Y_test)
print('\nTraining:{0:0.4f}'.format(Acc_training_radial),'\nTest:{0:0.4
f}'.format(Acc_test_radial))
```

SVM radial Leave-One-Out:

```
parameters = {'C': np.linspace(0, 50, 4),
              'gamma': np.linspace(0, 0.5, 4)}

svm_radial_cv_loo = GridSearchCV(estimator = SVC(kernel = 'rbf',
random_state = 300, probability = True),
                                    param_grid = parameters,
                                    cv = LeaveOneOut(),
                                    scoring = 'accuracy',
                                    n_jobs = -1,
                                    refit = True
                                 )

svm_radial_cv_loo.fit(X = X_train, y = Y_train)
print (svm_radial_cv_loo.best_score_, svm_radial_cv_loo.best_params_)

radial_loo = svm_radial_cv_loo.best_estimator_

radial_loo.fit(X_train, Y_train)
Acc_training_radial_loo = radial_loo.score(X_train, Y_train)
Acc_test_radial_loo = radial_loo.score(X_test, Y_test)
print('\nTraining:{0:0.4f}'.format(Acc_training_radial_loo),'\nTest:{0
:0.4f}'.format(Acc_test_radial_loo))
```