UNIVERSIDAD TÉCNICA DEL NORTE

Facultad de Ingeniería en Ciencias Aplicadas

Carrera de Ingeniería en Mecatrónica

Modelo de riego inteligente determinado a través de aprendizaje de máquina

Trabajo de grado previo a la obtención del título de Ingeniero en Mecatrónica

Autor:

Sara Betsabe Sandoval Santander

Director:

Ing. Carlos Xavier Rosero Chandi PhD.

Ibarra - Ecuador

2024

UNIVERSIDAD TÉCNICA DEL NORTE
BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	100481374-5		
APELLIDOS Y NOMBRES:	Sandoval Santander Sara Betsabe		
DIRECCIÓN:	Tobías Mena 20-15 y Avenida Eugenio Espejo		
EMAIL:	sbsandovals@utn.edu.ec		
TELÉFONO FIJO:	062585545	**TELÉFONO MÓVIL:**	0967863450

DATOS DE LA OBRA	
TÍTULO:	Modelo de riego inteligente determinado a través de aprendizaje de máquina
AUTOR (ES):	Sandoval Santander Sara Betsabe
FECHA DE APROBACIÓN: DD/MM/AAAA	6/05/2024
PROGRAMA:	■ **PREGRADO** ☐ **POSGRADO**
TITULO POR EL QUE OPTA:	INGENIERO EN MECATRÓNICA
ASESOR /DIRECTOR:	Ing.Miltón Gavilánez Villalobos, MSc Ing. Xavier Rosero Chandi PhD.

2. CONSTANCIAS

La autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 6 días del mes de mayo de 2024

EL AUTOR:

(Firma).....
Nombre: Sandoval Santander Sara Betsabe

Universidad Técnica del Norte

Facultad de Ingeniería en Ciencias Aplicadas

CERTIFICACIÓN DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

En mi calidad de director del trabajo de grado "MODELO DE RIEGO INTELIGENTE DETERMINADO A TRAVÉS DE APRENDIZAJE DE MÁQUINA", presentado por la egresada Sara Betsabe Sandoval Santander, que opta por el título de Ingeniero en Mecatrónica, certifico que el mencionado proyecto fue realizado bajo mi dirección.

Ibarra, 6 de mayo de 2024

Ing. Carlos Xavier Rosero Chandi PhD.

Director de Tesis

Universidad Técnica del Norte

Facultad de Ingeniería en Ciencias Aplicadas

APROBACIÓN DEL COMITÉ CALIFICADOR

El Tribunal Examinador del trabajo de titulación. Modelo de riego inteligente determinado a través de aprendizaje de máquina. Elaborado por Sara Betsabe Sandoval Santander, previo a la obtención del título de Ingeniero en Mecatrónica, aprueba elpresente informe de investigación en nombre de la Universidad Técnica del Norte:

Ing. Carlos Xavier Rosero Chandi PhD.

Director de Tesis

Ing. Alejandro Miltón Gavilánez Villalobos MSc

Asesor de Tesis

Dedicatorias

A mis amados padres, Byron y Albina, les dedico estas palabras con todo mi corazón. Su presencia ha sido el pilar fundamental de mi vida, guiándome con amor, paciencia y sabiduría en cada paso que doy. A mis queridas hermanas, Gabriela y Saskia, les agradezco por ser mis compañeras de vida, mis primeras maestras y mis confidentes más fieles. Su amor incondicional y constante han sido mi mayor fortaleza.

A mis adorados sobrinos, Natan e Ismael, quienes han iluminado mi vida con su alegría y amor incondicional, les dedico cada palabra de este mensaje. Su curiosidad infinita y su entusiasmo por descubrir el mundo me inspiran a ser mejor cada día.

Y a ti, mi amado Josué, mi ángel en el cielo, aunque la distancia nos separe, tu amor y recuerdo están siempre presentes en mi corazón. Eres mi fuerza y mi refugio, mi compañero en la alegría y en la adversidad. Gracias por enseñarme que el amor verdadero no conoce fronteras ni límites.

Agradecimientos

Quiero expresar mi profunda gratitud hacia mis padres por su infinito amor. Cada sacrificio que han hecho por mí hoy culmina en este logro, que siento más suyo que mío. Su constante dedicación y sacrificio son la razón por la que hoy puedo celebrar este éxito.

A mi adorada hermana Gabriela, gracias por ser mi fuente inagotable de amor y por brindarme tu cariño y apoyo. Tus sabios consejos y tu forma de alentarme a enfrentar desafíos me han dado fuerza para encarar el mundo con valentía.

Querida compañera de vida, Saskia, no encuentro palabras suficientes para expresar mi gratitud por tu constante presencia en mi camino. Gracias por escucharme sin juzgarme, por permitirme tropezar y aprender, y por ser mi apoyo. Tu amor es un tesoro que valoro más de lo que puedo decir.

Agradezco también a mi querido tutor, el Ingeniero Xavier Rosero, quien no solo compartió conmigo su conocimiento, sino que también me brindó su amistad invaluable. Su guía y apoyo fueron fundamentales en este camino.

Y no puedo dejar de mencionar a mi querida amiga, Dayana Cabezas, quien ha sido como una hermana para mí. Gracias por estar siempre a mi lado.

Índice general

Índice de figuras

Índice de tablas

Resumen

La práctica de cultivos urbanos es cada vez más común en las ciudades, pero el uso de métodos de riego imprecisos a menudo resulta en un consumo excesivo de agua y el deterioro de las plantas.Para abordar la problemática del riego en los cultivos urbanos, se propone un modelo inteligente específico para el cultivo de rábanos. Inicialmente, se recopila una base de datos detallada sobre esta hortaliza y se lleva a cabo un proceso exhaustivo de normalización y análisis de datos para determinar las condiciones óptimas de temperatura y humedad. Estos datos serán fundamentales para implementar el algoritmo de etiquetado, el cual proporcionará las etiquetas necesarias para indicar cuándo es necesario regar y cuándo no. Posteriormente, se analizan seis algoritmos de clasificación dentro del campo del aprendizaje automático, con un enfoque en el aprendizaje supervisado para la clasificación de datos. centrándose en la clasificación de datos. Estos algoritmos se evalúan utilizando métricas estándar y se comparan para seleccionar la opción que mejor se adapta a las necesidades de riego del rábano, asegurando una precisión y robustez óptimas. La propuesta busca proporcionar a los cultivadores urbanos una herramienta efectiva para optimizar el riego de sus cultivos. El algoritmo seleccionado debe destacarse por su precisión y robustez, asegurando una gestión eficiente del agua. Este enfoque integrado de aprendizaje de máquina y gestión del agua tiene el potencial de mejorar significativamente la

eficiencia de los cultivos urbanos y promover un desarrollo agrícola más sostenible en entornos urbanos.

Palabras clave : Cultivos urbanos,Aprendizaje de automático ,Algoritmos de clasificación.

Abstract

The practice of urban cultivation is increasingly common in cities, but the use of inaccurate irrigation methods often results in excessive water consumption and plant deterioration.To address the irrigation problem in urban crops, an intelligent model specific to radish cultivation is proposed. Initially, a detailed database on this vegetable is compiled and a thorough data normalization and analysis process is carried out to determine the optimal temperature and humidity conditions. These data will be fundamental to implement the labeling algorithm, which will provide the necessary labels to indicate when it is necessary to irrigate and when it is not.Subsequently, six classification algorithms within the field of machine learning are analyzed, with a focus on supervised learning for data classification. focusing on data classification. These algorithms are evaluated using standard metrics and compared to select the option that best suits radish irrigation needs, ensuring optimal accuracy and robustness. The proposal seeks to provide urban growers with an effective tool to optimize the irrigation of their crops. The selected algorithm should stand out for its accuracy and robustness, ensuring efficient water management. This integrated machine learning and water management approach has the potential to significantly improve the efficiency of urban crops and promote a more sustainable agricultural development in urban environments.

Keywords: Urban Crops,Machine Learning,Classification Algorithms.

Capítulo I

Introducción

1.1. Planteamiento del problema

Los cultivos urbanos son una práctica cada vez más popular en las ciudades ya que, ofrecen una serie de beneficios como la producción de alimentos saludables, la reducción de la huella de carbono y la promoción de la biodiversidad. Sin embargo, existen desventajas a considerar como la falta de capacitación en técnicas de cultivo que puede resultar en prácticas ineficientes, la contaminación del suelo, el consumo ineficiente de agua, el alto costo inicial, el tiempo y dedicación que requiere el mantenimiento [1]. Los métodos de riego tradicionales, como el riego por aspersión o el riego manual, a menudo conducen a un desperdicio de agua debido a la falta de monitoreo y regulación.

Para abordar este problema, el desarrollo de sistemas de riego inteligentes y eficientes, basados en tecnologías de aprendizaje de máquina, ofrecen una alternativa que permite una gestión precisa del agua y una adaptación dinámica a las necesidades específicas de los cultivos urba-

nos [1]. Esto no solo contribuye a la sostenibilidad de la agricultura urbana, sino también a la conservación de un recurso indispensable como el agua en entornos cada vez más poblados y urbanizados.

El uso de algoritmos de aprendizaje de máquina permite una toma de decisiones más precisa y adaptativa en el riego de los cultivos [2]. Estos algoritmos podrían analizar grandes cantidades de datos recopilados de sensores ubicados en el suelo, como la humedad, la temperatura y otros parámetros relevantes para determinar las necesidades hídricas específicas de cada planta, evitando el riesgo de proporcionar un riego insuficiente o excesivo, optimizando el uso del agua y promoviendo el crecimiento del cultivo.

El trabajo planteado, los agricultores urbanos pueden lograr una mayor productividad. Es importante tener en cuenta que, para la realización de un modelo, se requiere de una recopilación precisa de los datos de mayor relevancia del caso y así garantizar su funcionalidad en cualquier sistema automático.

1.2. Objetivos

1.2.1. Objetivo general

Desarrollar un modelo inteligente con base en aprendizaje de máquina para riego de cultivos urbanos.

1.2.2. Objetivos específicos

Analizar las técnicas de aprendizaje de máquina que se aplican a problemas de riego en cultivos urbanos.

Proponer un algoritmo para la determinación del modelo de riego inteligente a partir de una base de datos.

Validar el funcionamiento del modelo sobre la base de datos.

1.3. Justificación

La agricultura es uno de los sectores que más agua consume en el mundo, y la escasez de agua es un problema cada vez más acuciante. Por lo tanto, es necesario desarrollar modelos de riego más eficientes y sostenibles que permitan reducir el consumo de agua y mejorar la productividad de los cultivos [3]. El modelo de riego aplicado será una herramienta inteligente, siendo una alternativa tecnológica que facilitará el trabajo de los agricultores.

La tecnología de aprendizaje de máquina es una herramienta para el análisis de datos y la toma de decisiones en tiempo real. Al aplicar esta tecnología al riego de cultivos, se pueden desarrollar modelos de riego automatizados, que tienen un impacto significativo en la sostenibilidad de la agricultura, al reducir el consumo de agua y mejorar la productividad de los cultivos [4].

El uso de tecnologías de aprendizaje de máquina en la ingeniería mecatrónica puede permitir reducir el consumo de agua y mejorar la productividad de los cultivos. La investigación en este campo puede contribuir al avance de la ingeniería, al desarrollar nuevos modelos, técnicas para

el riego investigación y sus posibles impactos [5].

La implementación de un sistema de adquisición de datos en cultivos urbanos nos permite acceder a información importante y en base a esto generar soluciones que beneficien a la agricultura como: detección de plagas, enfermedades, predicción de rendimientos de cultivos, la optimización de la gestión del agua a través del análisis de factores como la calidad del suelo, la cantidad de agua y la luz solar. Investigar y desarrollar modelos de riego inteligente para la agricultura urbana puede ser crucial para su viabilidad.

1.4. Alcance

El presente proyecto consiste en desarrollar un modelo inteligente, que brinde a los agricultores urbanos la posibilidad de automatizar el riego en sus cultivos, aplicando técnicas de aprendizaje de máquina sobre información recopilada previamente a través de un sistema de adquisición de datos.

Teniendo en cuenta que, en los huertos urbanos se cultivan una variedad de productos con características diferentes, se presenta un escenario en el que cada uno de estos requiere de un modelo específico. En base a lo anterior se elige al rábano para este caso de estudio [6].

La información recopilada pertenece a una base de datos con parámetros importantes para su análisis como son la humedad del suelo, la temperatura ambiente y la humedad relativa en un cultivo de rábanos, durante todo su ciclo de desarrollo desde la siembra hasta la cosecha.

El modelo se realizará en software matemático libre que permite acceder a herramientas y librerías especializadas en análisis de datos y aprendizaje de máquina.

Capítulo II

Revisión literaria

2.1. Estado del arte

Se muestra como el lenguaje de máquina ayuda en diferentes ámbitos como lo es el riego en la agricultura al igual que en la fumigación lo que es relevante, por medio de una base de datos con variables como la temperatura, humedad del suelo y del ambiente se puede lograr que por medio de un modelo algebraico logre aprender para poder optimizar un proceso o hacerlo de manera automática como muestran los siguientes trabajos:

Según María Pousada, en el contexto de la aplicación de modelos predictivos en el ámbito del riego inteligente, se pueden identificar varios tipos de enfoques. Estos se dividen en dos categorías principales: supervisado y no supervisado. En el enfoque supervisado, se realizaron a cabo entrenamientos utilizando cuatro modelos de regresión y seis modelos de clasificación. Los resultados revelaron que todos los clasificadores enfrentaron desafíos significativos al predecir las decisiones relacionadas con el riego.

Además, es importante destacar la relevancia de las herramientas de software disponibles para la implementación de estos modelos predictivos. En este contexto, se mencionan dos lenguajes de programación ampliamente utilizados: Python y Lenguaje R se destaca por su abundancia de bibliotecas, como Scikit Learn, que facilitan el desarrollo de modelos de código abierto. Esto aporta ventajas significativas al proceso, ya que elimina la necesidad de adquirir costosas licencias de software [7].

El aprendizaje supervisado, específicamente el enfoque de clasificación, se revela como la elección más eficiente en el contexto de las aplicaciones de riego inteligente, según la investigación de Daniel Jiménez. En el marco de este proyecto, los resultados validan esta afirmación, ya que el error de predicción se mantiene por debajo del 20 % durante la primera y última fase de validación, llegando incluso al 0 %.

El estudio incluyó la evaluación de cuatro modelos distintos: Adaboost, Gradientboosting, SGDClassifier y MPLClassifier. Destaca que, al analizar su desempeño mediante la puntuación F1-Score, Adaboost sobresalió con una impresionante calificación del 84.45 %, mientras que el MPLClassifier obtuvo la puntuación más baja, registrando un 66.05 % [8].

En la investigación titulada "SISTEMA DE RIEGO DE PLANTAS INTELIGENTE BASADO EN IOT CON APRENDIZAJE MEJORADO", se propone la creación de un sistema IoT que se apoya en un modelo de riego inteligente. Este modelo aprende progresivamente las necesidades de riego de las plantas sin requerir datos preexistentes. A modo de prueba de concepto, se desarrolla un prototipo de aplicación que se ajusta automáticamente a las condiciones óptimas de riego después de un par de riegos manuales iniciales.

Para evaluar el desempeño de este sistema, se llevan a cabo pruebas tanto en el contexto de

riego manual como en el de riego automático, empleando algoritmos de aprendizaje automático. Los resultados obtenidos indican que el modelo exhibe una notable precisión en la toma de decisiones relacionadas con el riego [9].

El proyecto titulado "MODELO CLASIFICATORIO PARA RESIDUOS DE PLAGUICIDAS EN LOS ALIMENTOS BASADO EN MÉTODOS DE MACHINE LEARNING", desarrollado por Ludwin Flores, se enfoca en la aplicación de modelos de clasificación y regresión utilizando técnicas algebraicas como Redes Neuronales, Arboles de Decisión y Regresión. Para llevar a cabo este estudio, se emplea una base de datos que consta de 96638 registros, destinando el 80 % de los datos para el entrenamiento de los modelos, y el 20 % restante para la validación y cálculo del error de cada modelo.

Los resultados revelan que el modelo alcanza un error mínimo del 1,03 % en la clasificación de residuos orgánicos de plaguicidas. Esta eficacia en la clasificación se logra gracias al uso de algoritmos de aprendizaje automático, lo que demuestra cómo la implementación de lenguaje de máquina contribuye significativamente a mejorar la eficiencia de este proceso [10].

2.2. Riego de cultivos

La utilización del riego es una práctica fundamental en el cultivo de hortalizas, ya que estas plantas necesitan una cantidad adecuada de agua para crecer y producir frutos de calidad.

Existen diferentes sistemas de riego que se pueden utilizar. Uno de los más eficientes es el riego por goteo, que permite suministrar agua directamente a las raíces de las plantas, reduciendo el desperdicio de agua y evitando la aparición de enfermedades [11]. Además, este sistema

permite controlar la cantidad de agua que se suministra a cada planta, lo que es especialmente útil en cultivos de alta densidad.

Por otro lado, el riego por aspersión es un sistema que se basa en la dispersión de agua a través de un conjunto de tuberías y boquillas que rociarán el agua en forma de lluvia sobre el suelo y las plantas. Esta técnica se adapta especialmente bien a cultivos de gran extensión y baja densidad de plantación, ya que permite la distribución uniforme del agua en una amplia área. Sin embargo, es importante destacar que este método puede ser menos eficiente que el riego por goteo, debido a la posible pérdida de agua por evaporación y escurrimiento.

Además, el riego por aspersión conlleva el riesgo potencial de aumentar la incidencia de enfermedades fúngicas en las plantas, ya que las hojas y los tallos permanecen húmedos durante períodos más prolongados en comparación con otros sistemas de riego [12]. Esta consideración es crucial al evaluar la idoneidad de este método para un cultivo en particular y destaca la importancia de sopesar los pros y contras antes de su implementación.

Por último, el riego por inundación es un sistema de riego que se caracteriza por la aplicación de agua en una superficie nivelada que está rodeada por un dique. En este método, se utiliza una cantidad considerable de agua que se permite infiltrarse gradualmente en el suelo. Este enfoque de riego es particularmente apropiado para cultivos como el arroz, que demandan grandes volúmenes de agua y se desarrollan en suelos con una tasa de infiltración baja.

Es esencial destacar que el riego por inundación puede aumentar el riesgo de enfermedades fúngicas en las plantas. Esto se debe a que las hojas y los tallos permanecen en contacto prolongado con la humedad, lo que favorece el desarrollo de microorganismos patógenos. Por lo tanto, al considerar la implementación de este sistema, se debe evaluar cuidadosamente su

idoneidad en función de los cultivos específicos [13].

2.3. Cultivos en Imbabura

El cultivo de hortalizas en Imbabura, Ecuador, es un aspecto importante de la agricultura del país en esta región andina, ubicado en el norte del Ecuador el cual cuenta con una variedad de climas y altitudes que son aptos para el cultivo de una amplia gama de hortalizas [14].

Llevar los cultivos urbanos a convertirse en una práctica cada vez más importante en las áreas metropolitanas de la provincia es una respuesta a la creciente preocupación por la seguridad alimentaria, la sostenibilidad ambiental y la promoción de la agricultura local en las ciudades.

En las áreas urbanas de Imbabura, como en muchas ciudades de todo el mundo, los cultivos de hortalizas se realizan en una variedad de espacios disponibles, como patios traseros, terrazas, balcones, jardines comunitarios y parcelas pequeñas, donde los habitantes urbanos aprovechan cualquier espacio disponible para cultivar hortalizas frescas de manera saludable [15].

Imbabura cuenta con una amplia gama de elevaciones, desde áreas altas hasta áreas bajas, por lo que se puede cultivar diferentes tipos de hortalizas dependiendo de las condiciones del suelo y del clima debido a la variación de altitud. Por ejemplo, es posible cultivar tomates y pimientos en zonas bajas , mientras que papas y otros tubérculos se pueden cultivar en las zonas montañosas [16].

2.4. Cultivo del rábano

El rábano es una hortaliza que se desarrolla en el piso medio de la región interandina, que se situa entre los 3.000 – 3.500 metros sobre el nivel del mar, pertenece a la familia de las crucíferas, la cual es una planta que florece cada año en la misma temporada de siembra. Forma un tubérculo comestible, es un engrosamiento de la raíz en el que se acumulan las reservas [17].

Es una hortaliza que crece rápidamente, la cual es una buena opción para cultivar en espacios reducidos, por lo general, las condiciones climáticas ideales para cultivar rábanos de manera urbana en Imbabura pueden variar según la ubicación geográfica, pero en general se necesitan humedad relativa del aire, temperaturas frescas, suelo adecuado, buena cantidad de luz solar y materia orgánica.

2.4.1. Variables para el crecimiento del rábano

El cultivo de rábanos en entornos urbanos presenta desafíos específicos que requieren consideraciones especiales. Aunque varios factores que influyen en el crecimiento de los rábanos son similares a los aplicables al cultivo en general, pueden verse afectados por las condiciones urbanas. Por lo tanto, es esencial tener en cuenta estas variables y realizar un seguimiento constante y control de ellas para satisfacer las necesidades de las plantas.

Una estrategia efectiva en este contexto es la implementación de sistemas automatizados de monitoreo y control de variables ambientales, como la temperatura, iluminación y riego. Estas tecnologías pueden mejorar significativamente el desarrollo de las plantas y garantizar condiciones óptimas de crecimiento. Además, la utilización de tecnologías de control y automatización

de variables ambientales ofrece la oportunidad de optimizar el rendimiento de los cultivos [18].

Para garantizar el desarrollo adecuado del rábano, es esencial tener en cuenta varias variables, como la temperatura ambiente y la humedad del aire y del suelo. Las temperaturas óptimas para el cultivo de rábanos se sitúan entre 15 y 18 grados Celsius, con mínimas de 4 grados Celsius y máximas de 21 grados Celsius. Exponer las plantas a temperaturas por debajo de 7 grados Celsius durante períodos prolongados podría desencadenar el desarrollo prematuro del tallo floral [19]. En lo que respecta a la humedad del suelo, el porcentaje adecuado puede variar según diversos factores, como el tipo de suelo, la temperatura y la disponibilidad de agua. Generalmente, los rábanos necesitan un suelo húmedo pero bien drenado para crecer de manera saludable. Por tanto, es crucial mantener la humedad apropiada según el estado de maduración de la planta. Se pueden seguir las recomendaciones proporcionadas en la tabla 2.1 para asegurar una humedad del suelo adecuada durante las diferentes fases de desarrollo del rábano.

Fase	**Humedad del Suelo**	**Recomendaciones**
Germinación	50 % - 75 %	Mantener el suelo constantemente húmedo, pero no empapado.
Crecimiento	50 % - 80 %	Mantener el suelo húmedo, pero no saturado, a medida que las plántulas crecen.
Madurez	40 % - 70 %	Conservar el suelo ligeramente húmedo a medida que las raíces se desarrollan y el cultivo madura.

Tabla 2.1: Recomendaciones de humedad del suelo para diferentes fases del desarrollo del rábano.

2.5. Aprendizaje de máquina

El aprendizaje de máquina se enfoca en capacitar a los dispositivos para aprender de manera automática basándose en patrones identificados en los datos. Esto se logra mediante el desarrollo de algoritmos que permiten a las computadoras mejorar su desempeño en tareas específicas sin requerir una programación explícita para cada tarea [20]. Python se destaca como una herramienta esencial para la implementación de estos algoritmos debido a su amplia gama de bibliotecas, como NumPy y Pandas, que facilitan la manipulación de datos y el álgebra lineal. Especialmente, la librería Scikit-Learn proporciona una colección de algoritmos de aprendizaje supervisado y no supervisado ya definidos [21]. Durante la fase de aprendizaje de máquina, se realiza un análisis exhaustivo de los datos disponibles, y se aplican técnicas de aprendizaje específicas para resolver el problema en cuestión. Una vez entrenado el modelo con los datos disponibles, se prueba su rendimiento utilizando nuevos datos para evaluar su precisión y generalización. La Figura 2.1 ilustra los pasos típicos en el proceso de aprendizaje de máquina.

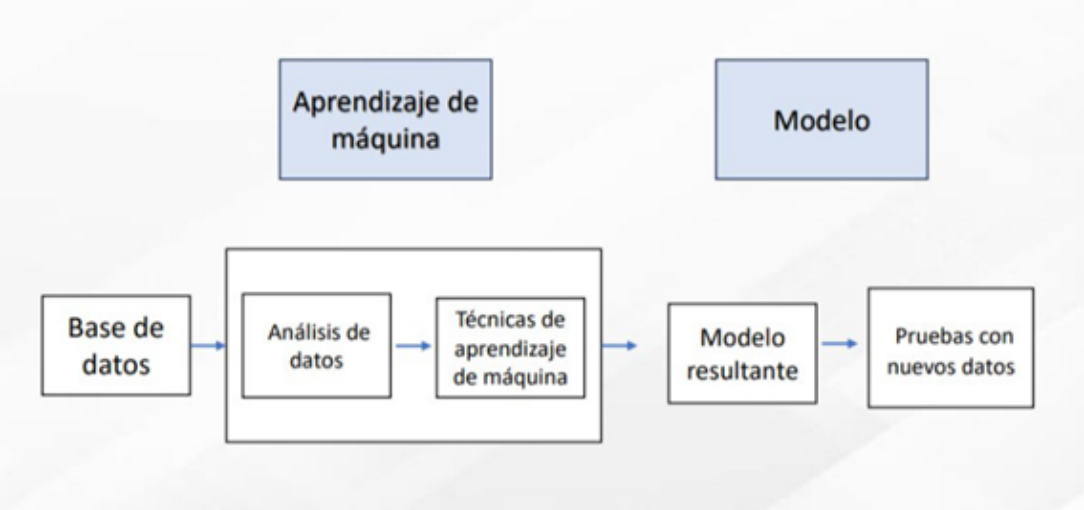

Figura 2.1: Pasos del aprendizaje de máquina.

El aprendizaje de máquina desempeña un papel esencial en el ámbito agrícola, donde se emplea en diversas aplicaciones clave. Por ejemplo, se utiliza para monitorear constantemente la temperatura y la humedad en invernaderos, así como para analizar el comportamiento de los índices de consumo de combustible durante las labores agrícolas. Además, se aplica en la producción de hortalizas orgánicas en atmósferas controladas, donde ayuda a predecir las condiciones de temperatura y humedad relativa del aire en los invernaderos, y a evaluar cómo las variables climáticas afectan el rendimiento de cultivos específicos.

En resumen, el aprendizaje de máquina se ha convertido en una herramienta valiosa en la agricultura, ya que permite optimizar el crecimiento y el rendimiento de los cultivos mediante una comprensión y gestión más precisa de las condiciones ambientales y otros factores relevantes.

Existen tres tipos principales de aprendizaje en el aprendizaje de máquina: supervisado, no supervisado y por refuerzo. En el aprendizaje supervisado, se utilizan datos etiquetados para entrenar al modelo y predecir resultados futuros. En el aprendizaje no supervisado, el modelo busca patrones y relaciones en los datos sin la necesidad de etiquetas previas. Por último, en el aprendizaje por refuerzo, el modelo aprende a través de la retroalimentación que recibe al interactuar con su entorno y tomar decisiones.

2.5.1. Aprendizaje supervisado

Es un tipo de aprendizaje automático en el que se proporciona al modelo de aprendizaje de máquina un conjunto de datos etiquetados, es decir, datos que ya tienen una respuesta conocida, para que el modelo pueda aprender a predecir la respuesta correcta para nuevos datos . En este

tipo de aprendizaje, el modelo se entrena con un conjunto de datos de entrenamiento y se evalúa con un conjunto de datos de prueba para medir su precisión [22].

Se utiliza en una variedad de aplicaciones, como medicina predictiva, cultivos inteligentes, riego en la agronomia entre otros.

2.5.2. Aprendizaje no supervisado

Es un tipo de aprendizaje automático en el que el modelo de aprendizaje de máquina se entrena con un conjunto de datos no etiquetados, es decir, datos que no tienen una respuesta conocida. Su objetivo de aprendizaje es poder encontrar patrones y estructuras en los datos sin la ayuda de etiquetas o respuestas previas [23]. Se utiliza en una variedad de aplicaciones, como agrupamiento, reducción de dimensionalidad y detección de anomalías.

2.5.3. Aprendizaje por refuerzo

Es una técnica de aprendizaje automático que se utiliza para entrenar a un agente en un entorno dinámico. En este enfoque, el agente aprende a tomar decisiones para maximizar una recompensa a largo plazo. El agente interactúa con el entorno y recibe una señal de recompensa o castigo en función de sus acciones [24].

El objetivo del agente es aprender una política que maximice la recompensa total a lo largo del tiempo, donde se lo ha utilizado en una variedad de aplicaciones, como juegos, robótica, control de procesos y publicidad en línea [25].

2.6. Técnicas de aprendizaje aplicadas al riego de cultivos urbanos

En el ámbito del riego de cultivos, se utilizan diversas técnicas, como la clasificación y la regresión, dentro del aprendizaje supervisado. Este campo de la inteligencia artificial se enfoca en desarrollar métodos para que las computadoras aprendan de datos disponibles. Estas técnicas se aplican con el fin de mejorar la eficiencia en el riego y fumigación de los cultivos de hortalizas, lo que contribuye a una producción agrícola más eficiente y sostenible.

Un ejemplo destacado de técnica aplicable es el de los Bosques Aleatorios (Random Forest), un algoritmo de aprendizaje automático que se utiliza tanto para clasificación como para regresión. Este método se basa en el ensamblaje, combinando múltiples árboles de decisión para mejorar la precisión y evitar el sobreajuste. A diferencia de usar un solo árbol de decisión, el Random Forest opera con una colección de árboles, cada uno entrenado con un subconjunto aleatorio de características y datos [26].

2.6.1. Clasificación

En el contexto del riego de cultivos, la clasificación es una técnica de aprendizaje supervisado utilizada para predecir la clase o categoría a la que pertenece un objeto o dato [27]. Esta técnica permite determinar, por ejemplo, si un cultivo requiere más o menos agua según sus características y condiciones ambientales.

Entre los algoritmos de clasificación más destacables se encuentran los árboles de decisión , k-nearest neighbors , el modelo de regresión logística y Naive Bayes . Los árboles de decisión

son métodos que dividen el conjunto de datos en subconjuntos más pequeños y homogéneos, facilitando la toma de decisiones [28]. Por otro lado, k-nearest neighbors es un método simple pero efectivo que se utiliza en una variedad de aplicaciones, incluyendo el reconocimiento de patrones y la minería de datos [29]. La regresión logística, por su parte, es un algoritmo de clasificación binaria que estima la probabilidad de que un objeto pertenezca a una de las dos clases posibles [30]. Naive Bayes se basa en el teorema de Bayes para predecir la probabilidad de que un nuevo dato pertenezca a una clase específica, asumiendo independencia entre las características [31]. Estos algoritmos son fundamentales en el análisis de datos agrícolas para tomar decisiones informadas sobre el riego y otros aspectos de la gestión de cultivos.

2.6.2. Regresión

La regresión es una técnica de aprendizaje supervisado que se emplea para predecir un valor numérico basado en un conjunto de variables de entrada [32]. En el contexto del riego de cultivos, los algoritmos de regresión son útiles para estimar la cantidad de agua necesaria por un cultivo en función de sus características y las condiciones ambientales.

Uno de los algoritmos más utilizados en regresión es la regresión lineal. Este algoritmo busca establecer una relación lineal entre las variables de entrada y la variable objetivo, con el objetivo de predecir valores numéricos. En particular, la regresión lineal busca encontrar la línea recta que mejor se ajuste a los datos disponibles y utiliza esta línea para realizar predicciones futuras [33].

2.7. Justificación sobre la técnica a usar en este trabajo

El modelo de clasificación en aprendizaje automático resulta más apropiado que el de regresión para un sistema de riego inteligente en agricultura. Esto se debe a que el modelo de clasificación es idóneo para predecir la pertenencia a una categoría, como determinar si un cultivo requiere riego o no, aspecto fundamental en un sistema de riego inteligente. Por otro lado, el modelo de regresión se destina más a la predicción de valores numéricos, como la cantidad exacta de agua a aplicar, lo cual podría no ser tan relevante en este contexto. Además, el modelo de clasificación puede facilitar la toma de decisiones binarias, como encender o apagar un sistema de riego, según las condiciones del cultivo y del suelo [34].

El empleo de algoritmos de aprendizaje de máquina para la gestión de la irrigación inteligente en la agricultura de precisión resulta crucial, ya que permite analizar datos en tiempo real y anticipar recomendaciones de riego. Un estudio sobre un "Modelo de aprendizaje de máquina en Tiempo Real para Agricultura de Precisión"subraya la importancia de estos modelos para la gestión del agua en el cultivo de lúpulo [35]. Además, el modelo de clasificación se utiliza ampliamente en diversos ámbitos, incluida la agricultura, para tomar decisiones basadas en la clasificación de datos.

En resumen, el modelo de clasificación en aprendizaje de máquina resulta más adecuado que el de regresión para un sistema de riego inteligente en agricultura, ya que permite tomar decisiones binarias fundamentadas en la clasificación de datos, aspecto esencial en la gestión de la irrigación inteligente.

Este capítulo demuestra que, a partir del apartado 2.6, se cumple con el primer objetivo

secundario, que se centra en el análisis de las técnicas de aprendizaje de máquina aplicadas a problemáticas de riego en cultivos urbanos. Este análisis abarca un estudio exhaustivo de las técnicas supervisadas, no supervisadas y por esfuerzo. Además, se detalla la aplicabilidad de cada técnica en el contexto del sistema de riego inteligente de hortalizas. Esta fase resulta fundamental para la concepción y desarrollo del proyecto, ya que permite seleccionar de manera eficiente la técnica más adecuada para alcanzar los objetivos planteados.

Capítulo III

Desarrollo

En este capítulo, se aborda el desarrollo del estudio, que implica la implementación y evaluación de seis algoritmos de clasificación utilizando una base de datos específicamente diseñada para el cultivo de rábanos. El objetivo principal es identificar el algoritmo más eficaz para predecir cuándo activar el sistema de riego. Se detalla el proceso de preparación de datos, que incluye la limpieza, normalización y estandarización de variables climáticas, condiciones del suelo y niveles de humedad. Además, se describe el desarrollo del algoritmo de etiquetado, diseñado para asignar etiquetas a los datos en función de criterios predefinidos para clasificar los períodos de activación del sistema de riego. La metodología de implementación abarca la selección y configuración de los algoritmos, así como su ajuste al conjunto de datos de rábanos.

3.1. Base de datos

La información para el modelo de riego inteligente se obtiene de la tesis sistema de adquisición de datos en cultivos de huertos urbanos en [6]. Esta fuente proporciona los datos fundamentales para el desarrollo del modelo de riego inteligente. La base de datos está compuesta por mediciones recopiladas históricamente, centrándose en tres parámetros principales la humedad del suelo, humedad del aire y temperatura del aire.

3.1.1. Preparación de datos

Los datos recopilados se preparan para el análisis de aprendizaje de máquina. La preparación de datos incluye los siguientes pasos:

- Limpieza de datos: Se eliminan datos erróneos o incompletos.
- Transformación de datos: Se transforman los datos para que estén en el formato adecuado para el análisis de aprendizaje de máquina.

Para el desarrollo del modelo de riego inteligente, se utilizan únicamente los parámetros, mostrados en la Figura 3.1. Nótese que se tiene las dimensiones humedad del suelo (HumSuelo), humendad del aire (HumAire), temperatura (Temperatura) y la salida riego (Riego). Tanto estos parámetros como el proceso de preparación de datos desempeñan un papel crucial en la creación y validación del modelo de riego inteligente.

	HumSuelo	HumAire	Temperatura	Riego
0	20.00	74.94	10.23	No
1	21.36	75.00	9.93	No
2	22.73	75.00	8.90	No
3	24.09	75.00	8.73	No
4	25.45	75.00	8.89	Yes
5	26.82	75.00	9.49	Yes
6	28.18	75.00	9.41	Yes
7	29.55	75.00	9.17	Yes
8	30.91	75.00	10.03	Yes
9	32.27	75.00	10.43	Yes

Figura 3.1: Parámetros de la base de datos.

3.1.2. Algoritmo de etiquetado

Se ilustra el funcionamiento de la asignación de etiquetas de entrenamiento en el modelo, como se muestra en la Figura 3.2, un paso fundamental en el aprendizaje supervisado. Este proceso es esencial para la implementación de los algoritmos mencionados en el marco teórico. A través de la asignación de etiquetas, el modelo aprende a clasificar nuevos datos según las características presentes en los datos de entrenamiento.

Para entender el algoritmo, observamos su diagrama de flujo y la tabla 3.1, que describe el significado de las variables utilizadas. El diagrama muestra visualmente el proceso de determinación de la necesidad de riego, mientras que la tabla proporciona detalles sobre las variables clave en el algoritmo.

Variable Calculada	**Significado**
totalHumSuelo	Serie temporal de datos de humedad del suelo.
stdDevHumSuelo	Desviación estándar de los datos de humedad del suelo.
Der[i]	Primera derivada de la serie temporal de datos de humedad del suelo en el punto i.

Tabla 3.1: Variables calculadas.

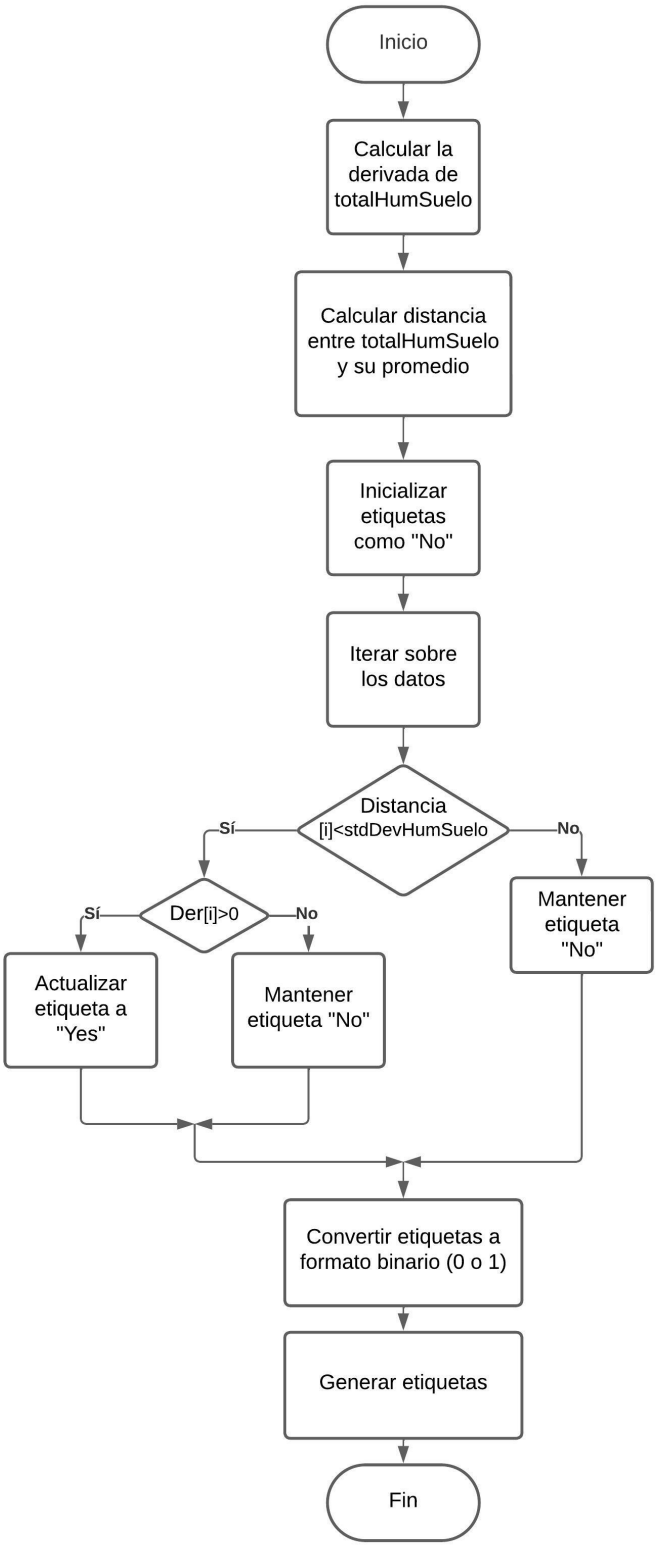

Figura 3.2: Diagrama de flujo del algoritmo de etiquetado para el modelo de riego.

Este diagrama proporciona una representación visual del proceso que sigue el algoritmo para determinar la necesidad de riego en función de los datos de humedad del suelo.

Como se observa en la Figura 3.2, el algoritmo comienza calculando el promedio y la desviación estándar de los datos de humedad del suelo. Luego, utiliza una serie de condiciones lógicas para evaluar si la humedad del suelo está por encima o por debajo del promedio, y si la tasa de cambio indica una disminución en la humedad. Basándose en estos criterios, el algoritmo determina si se recomienda o no el riego.

El algoritmo implementado genera una gráfica inicial que muestra la humedad total del suelo, así como la relación entre esta y la derivada de los datos de humedad del suelo.

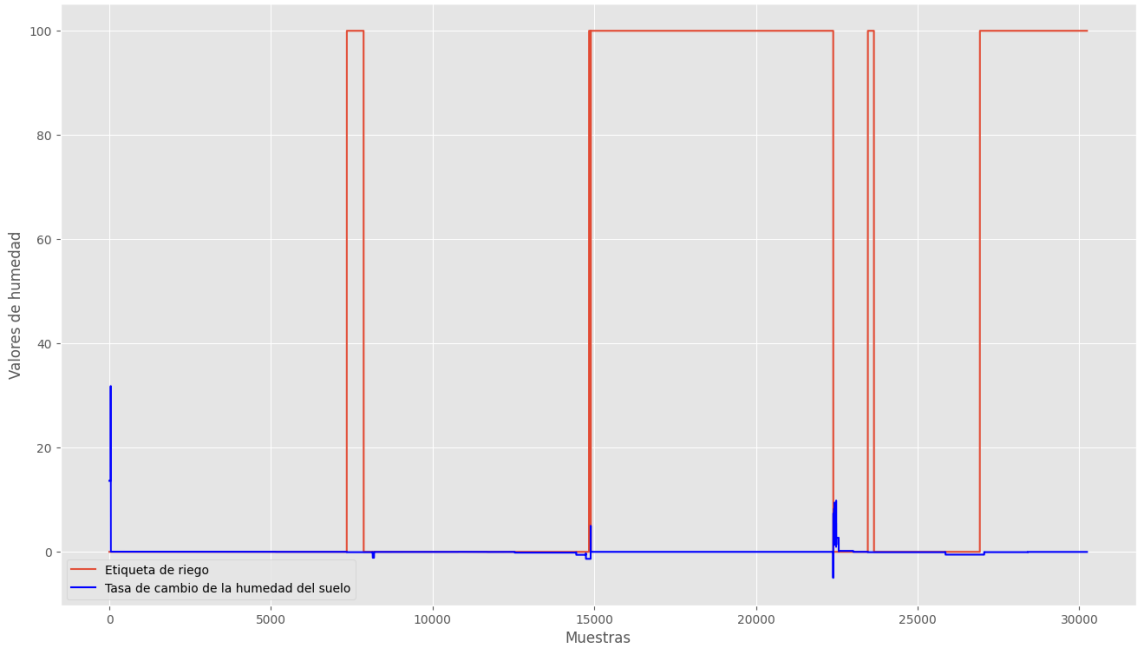

Figura 3.3: Visualización de la humedad del suelo y su derivada.

La Figura 3.3 muestra la humedad total del suelo representada por la línea principal y la tasa de cambio de la humedad del suelo representada por la línea secundaria. La humedad del suelo se expresa en términos porcentuales en el eje vertical, mientras que en el eje horizontal se

encuentran las muestras o puntos de datos. La tasa de cambio de la humedad del suelo, mostrada por la línea secundaria, indica cómo varía la humedad en relación con las muestras. La relación entre estas dos representaciones visuales ofrece información sobre cómo cambia la humedad del suelo con respecto a los patrones observados en los datos.

A partir de los resultados obtenidos, donde el promedio de la humedad del suelo es de 60.97 y la desviación estándar es de 35.77, se procede a generar las etiquetas. En este proceso, se establece un umbral superior de 96.74. El algoritmo compara los valores de la humedad del suelo con el promedio más la desviación estándar para determinar la etiqueta correspondiente.

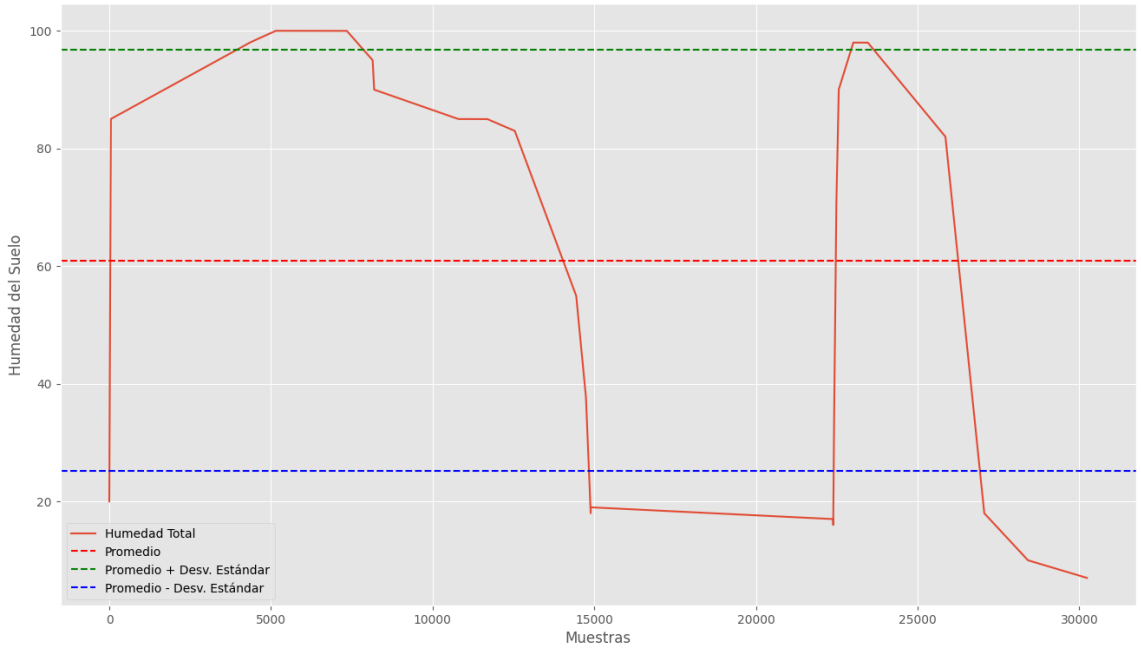

Figura 3.4: Visualización del promedio de humedad del suelo y desviación estándar

En la Figura 3.4, se visualiza el promedio de humedad del suelo y la desviación estándar. Si el valor de la humedad del suelo está por debajo de este umbral y su derivada es negativa, se etiqueta como “Yes” (indicando que se debe regar). De lo contrario, se etiqueta como “No”. Este enfoque permite clasificar eficazmente los momentos en los que es necesario realizar riegos

adicionales en función de los datos disponibles.

Con la inclusión de las etiquetas “Yes” y “No”, se proporciona información valiosa que puede ser aprovechada en el desarrollo de los algoritmos de clasificación mencionados en el marco teórico.

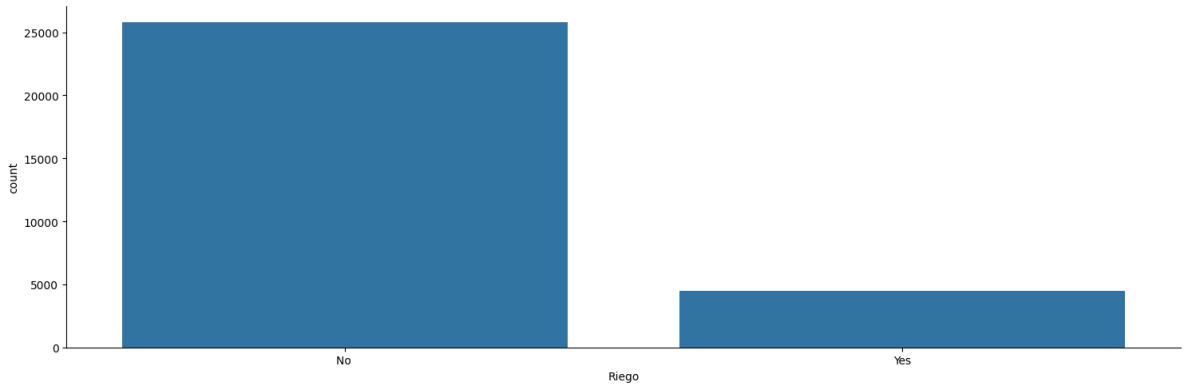

Figura 3.5: Visualización de etiquetas (No-Yes).

En la Figura 3.5, se presenta una visualización clara de la distribución de los datos, lo que la convierte en una herramienta valiosa para el análisis y la comprensión de los patrones presentes en los datos.

3.2. Metodología de implementación

Se detalla de la metodología utilizada para implementar y operar los algoritmos empleados en el proyecto. Estos algoritmos están diseñados específicamente para trabajar con la base de datos normalizada y las etiquetas correspondientes, lo que garantiza un procesamiento eficiente y preciso de los datos.

Los algoritmos seleccionados en el marco teórico son desarrollados siguiendo la metodología CRISP-DM (Cross-Industry Standard Process for Data Mining), la cual se aplica al campo

del Aprendizaje de Máquina. CRISP-DM es reconocida por ser una metodología ampliamente utilizada y bien establecida para guiar proyectos de minería de datos [36]. Además, se destaca por su flexibilidad, lo que le permite adaptarse a los cambios que puedan surgir durante el proceso.

Figura 3.6: Visualización de las fases de CRISP-DM.

En la Figura 3.6 se visualizan las fases de CRISP-DM, donde la primera fase se enfoca en

definir los objetivos y requisitos del proyecto desde una perspectiva empresarial. La segunda fase, comprensión de datos, se encarga de recopilar y explorar los datos. La tercera fase aborda la preparación de los datos para su análisis. La cuarta fase, modelado, implica la aplicación de diversas técnicas de modelado, ya sean predictivas o descriptivas. En la quinta fase, evaluación, se determina el rendimiento del modelo aplicado. Finalmente, en la fase de despliegue, se integran los modelos en sistemas existentes.

3.3. Implementación de los algoritmos de aprendizaje

Resulta fundamental para comprender la aplicación práctica de los algoritmos en el proyecto y su impacto en la resolución de los desafíos planteados.

Se adaptó las fases de la metodología CRISP-DM para ajustarlas al enfoque específico del proyecto, centrándonos en la selección del mejor algoritmo. De esta manera, se adaptó una metodología estructurada y organizada que se alinea perfectamente con las necesidades y objetivos.

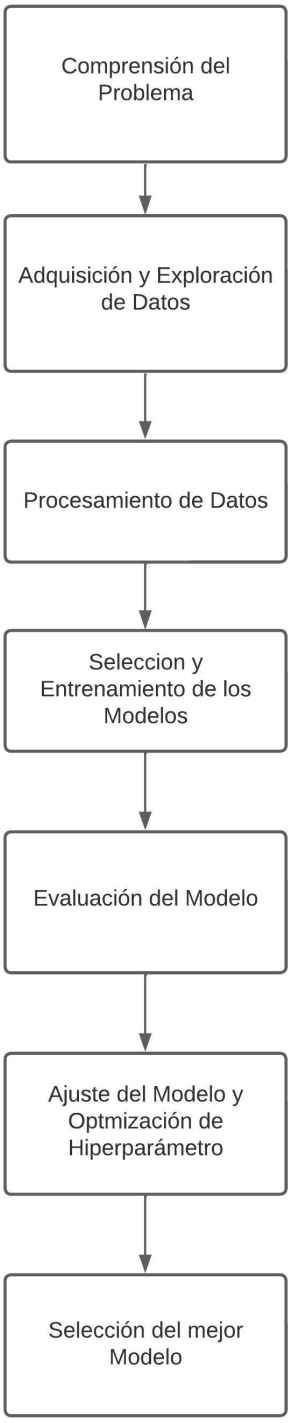

Figura 3.7: Visualización de las fases para el desarrollo de modelos de aprendizaje de máquina.

En la 3.7 se presenta la metodología adaptada a las necesidades del proyecto, la cual consta

de varias fases:

- En la primera fase, se identifica el problema de predecir cuándo regar y cuándo no hacerlo. Esto implica análizar los requisitos del proyecto y de los objetivos a alcanzar.

- En la segunda fase, se adquiere la base de datos según se describe en el apartado 3.1. Se realiza una cuidadosa selección de la fuente de datos para garantizar la calidad y relevancia de los mismos para el análisis posterior.

- En la tercera fase del proceso, se lleva a cabo la normalización de los datos y la selección de los parámetros a utilizar. Además, se genera una nueva salida llamada "RiegoCod", donde se asigna el valor de 1 a la etiqueta "Yes" y el valor de 0 a la etiqueta "No". Esta adición permite una mejor preparación de los datos para su análisis y modelado subsiguiente, como se ilustra en la Figura 3.8.

	HumSuelo	HumAire	Temperatura	Riego	RiegoCod
0	20.00	74.94	10.23	No	0
1	21.36	75.00	9.93	No	0
2	22.73	75.00	8.90	No	0
3	24.09	75.00	8.73	No	0
4	25.45	75.00	8.89	Yes	1
5	26.82	75.00	9.49	Yes	1
6	28.18	75.00	9.41	Yes	1
7	29.55	75.00	9.17	Yes	1
8	30.91	75.00	10.03	Yes	1
9	32.27	75.00	10.43	Yes	1

Figura 3.8: Implementación de la nueva salida "RiegoCod" en la base de datos.

- La cuarta fase implica la implementación de los algoritmos seleccionados: Árbol de Decisión (AD), K-Nearest Neighbors (k-NN), Regresión Logística (RL), Naive Bayes (NB),

Máquinas de Vectores de Soporte (SVM) y Bosques Aleatorios (RF). Todos los modelos se entrenan con el 70 % de los datos y se validan con el 30 %.

- En la quinta fase, se evalúan los algoritmos utilizando métricas de evaluación estándar como precisión, recall, f1-score y la matriz de confusión y eficiencia computacional, para medir su rendimiento en la clasificación de nuevas instancias.

- En la sexta fase, se aplica la validación cruzada a todos los algoritmos seleccionados para controlar el sobreajuste y obtener una estimación más precisa de su rendimiento.

- Finalmente, en la última fase, se comparan las métricas de evaluación de cada algoritmo y se selecciona el óptimo para resolver el problema.

3.4. Descripción de los algoritmos

3.4.1. Árbol de Decisión

El algoritmo de Árbol de Decisión opera mediante una estructura jerárquica de tipo árbol, donde cada nodo representa una característica del conjunto de datos y cada rama una decisión basada en esa característica. Para encontrar los puntos de división óptimos, el algoritmo utiliza la estrategia de "divide y vencerás", evaluando todas las posibles divisiones en todas las características y seleccionando aquella que maximice algún criterio de impureza. La selección de los hiperparámetros óptimos, como la profundidad máxima del árbol, se realiza mediante la búsqueda en cuadrícula, probando diferentes combinaciones de valores y evaluando el rendimiento del modelo en un conjunto de validación. Esta combinación de enfoques permite construir un

modelo predictivo eficaz y interpretable para tomar decisiones basadas en las características más relevantes del conjunto de datos.

3.4.2. K-Nearest Neighbors

El algoritmo de K-Nearest Neighbors clasifica nuevas instancias basándose en su proximidad con las instancias de entrenamiento más cercanas. Para implementarlo en Python, se comienza preparando los datos, dividiéndolos en conjuntos de entrenamiento y validación mediante la función train_test_split de scikit-learn. Luego, se realiza el preprocesamiento, escalando las características con técnicas como MinMaxScaler para mantener su uniformidad. Posteriormente, se crea y entrena el modelo utilizando la clase KNeighborsClassifier de scikit-learn, calculando las distancias entre los puntos de datos durante el entrenamiento. La selección del valor óptimo de 'k' se realiza mediante técnicas como la validación cruzada o la búsqueda en cuadrícula, maximizando la precisión en datos no vistos. Una vez entrenado, se evalúa el modelo con datos de validación, calculando métricas como precisión, recall y F1-score. Además, se evalúa la eficiencia computacional, incluyendo tiempos de entrenamiento, predicción y visualización de resultados.

3.4.3. Regresión Logística

La Regresión Logística es un modelo de aprendizaje supervisado que predice la probabilidad de una variable categórica binaria basada en variables predictoras. Su implementación en Python es útil para clasificación binaria y sirve como línea base sólida en Aprendizaje de Máquina. Los pasos para implementarla implican cargar datos desde un archivo CSV, dividirlos

en conjuntos de entrenamiento y validación con train_test_split de scikit-learn, crear y entrenar el modelo LogisticRegression ajustando sus parámetros internos. Durante el entrenamiento, se ajustan los parámetros utilizando técnicas de optimización como el descenso del gradiente para maximizar la verosimilitud de los datos de entrenamiento. Cuando se manejan características categóricas, es esencial usar técnicas de codificación como la codificación one-hot para convertirlas en vectores binarios, preservando su información única. Al evaluar el modelo con datos de validación, se generan métricas como precisión, recall y F1-score para medir su desempeño. Finalmente, se evalúa la eficiencia computacional del algoritmo, incluyendo tiempos de entrenamiento y predicción. Este enfoque sistemático garantiza una implementación efectiva de la Regresión Logística en Python.

3.4.4. Naive Bayes

El algoritmo de Naive Bayes es una técnica simple pero efectiva en el modelado predictivo dentro del aprendizaje automático, especialmente en tareas de clasificación como la predicción de riesgo en el presente proyecto. Se basa en el Teorema de Bayes y asume independencia condicional entre las características. Primero, se carga los datos desde un archivo CSV y se dividen en conjuntos de entrenamiento y validación. Luego, se calculan las probabilidades a priori de las clases y se entrena el clasificador Naive Bayes Gaussiano, donde se modelan las distribuciones de probabilidad condicional de las características para cada clase. La elección del enfoque Gaussiano se debe a que asume que las características siguen una distribución normal, lo cual puede afectar el rendimiento del modelo en función de la naturaleza de los datos. Posteriormente, se evalúa el modelo mediante validación cruzada para calcular la precisión

media. Las predicciones del modelo en el conjunto de prueba se utilizan para construir la matriz de confusión y generar un informe detallado de métricas como precisión, recall y F1-score. Finalmente, se visualiza la matriz de confusión para evaluar intuitivamente el rendimiento del modelo, y se calcula el tiempo total de eficiencia computacional, que incluye el tiempo de entrenamiento y visualización.

3.4.5. Máquinas de Vectores de Soporte

Las Máquinas de Vectores de Soporte (SVM) son un algoritmo de aprendizaje supervisado comúnmente utilizado en problemas de clasificación. Se basan en el teorema de Bayes y pueden calcular la probabilidad de un evento utilizando información previa sobre condiciones relacionadas. En la implementación, los datos se dividen en conjuntos de entrenamiento y validación, y se entrena el clasificador SVM utilizando el conjunto de entrenamiento. Se evalúa el modelo mediante validación cruzada y se generan métricas de rendimiento como precisión, recall y F1-score. Las predicciones se realizan en el conjunto de prueba, se construye una matriz de confusión y se calcula el tiempo total de eficiencia computacional, que incluye el tiempo de entrenamiento y predicción. Para encontrar el hiperplano de separación óptimo, SVM maximiza el margen entre clases. Si los datos no son linealmente separables, SVM utiliza el truco del kernel para mapearlos a un espacio de mayor dimensión donde sí lo sean. La selección de hiperparámetros, como el tipo de kernel y el parámetro de regularización, se realiza para optimizar el rendimiento del modelo.

3.4.6. Bosques Aleatorios

El algoritmo de Bosques Aleatorios es reconocido en el ámbito del aprendizaje automático por su capacidad para construir múltiples árboles de decisión y combinar sus predicciones, ofreciendo resultados precisos y robustos. En la implementación, luego de cargar y dividir los datos, se define una cuadrícula de hiperparámetros para la búsqueda en cuadrícula. Se ajustan parámetros como el número de árboles en el bosque y la profundidad máxima de cada árbol. Luego, se realiza una búsqueda exhaustiva de los mejores hiperparámetros mediante GridSearchCV de scikit-learn. Se utilizan los datos de prueba para hacer predicciones, y se genera un informe detallado con métricas de rendimiento. Además, se construye y visualiza la matriz de confusión para una evaluación intuitiva del modelo. Finalmente, se calcula el tiempo total de eficiencia computacional, incluyendo el tiempo de entrenamiento, predicción y visualización de la matriz de confusión.

Capítulo IV

Pruebas de funcionamiento

En este capítulo, se ha llevado a cabo un analisis de diversas métricas de rendimiento en el conjunto de pruebas para cada algoritmo, con el fin de validar su eficacia en relacion con la base de datos utilizada. Este análisis ha permitido la identificación y selección del algoritmo mas adecuado para abordar el problema específico planteado en este estudio. Al evaluar las métricas de rendimiento, se busca asegurar la idoneidad y efectividad del algoritmo seleccionado en la resolución del problema en cuestión.

4.1. Descripción de las pruebas

Se detallan las pruebas específicas para evaluar el rendimiento de los algoritmos de clasificación considerados en el contexto del sistema de riego automatizado para cultivos urbanos. Se describen los criterios de evaluación y las métricas utilizadas para medir la eficacia de cada algoritmo en la clasificación de los datos relacionados con el riego. Los conjuntos de datos de

entrenamiento se compondrán del 70 % , mientras que el conjunto de prueba representará el 30 % restante. Además, se explica la técnica de validación cruzada utilizada para garantizar la robustez de los resultados.

4.1.1. Métricas de validación

Los principios de evaluación consideran el empleo de diversas métricas para evaluar la eficacia de los algoritmos de clasificación [37]. Entre estas métricas se seleccionan las siguientes para evaluar a los algoritmos elegidos, como se muestra en la tabla 4.1.

Métrica	**Significado**
Exactitud (Accuracy)	Proporción de instancias clasificadas correctamente por el modelo.
Recuperación (Recall)	Proporción de valores positivos correctamente clasificados.
Puntuaje (F1-score)	Media armónica entre precisión y recuperación.
Matriz de Confusión (MC)	Permite comparar las etiquetas predichas con las reales y calcular diversas métricas de rendimiento.
Precisión (Precision)	Proporción de predicciones positivas correctas.
Eficiencia computacional	Tiempo necesario para entrenar, predecir y visualizar resultados del modelo.

Tabla 4.1: Descripción de Métricas de Evaluación de Modelos.

Estas métricas son fundamentales en la evaluación de algoritmos de clasificación, ya que ofrecen una visión del rendimiento del modelo en términos de su capacidad para clasificar correctamente los datos y encontrar casos relevantes dentro de ellos.

4.1.2. Validación cruzada

La validación cruzada es una técnica robusta para evaluar la generalización del análisis estadístico a conjuntos de datos independientes. Esta estrategia implica entrenar múltiples modelos en subconjuntos de los datos de entrada disponibles y evaluar su desempeño en el subconjunto complementario de datos [38]. Además, la validación cruzada facilita la detección del sobreajuste de manera eficiente, lo que resulta especialmente útil en la validación del algoritmo seleccionado.

Los algoritmos seleccionados se validan mediante el uso de la validación cruzada k-fold estratificada, lo que garantiza una distribución equitativa de las clases en cada fold o pliegue. Este conjunto se utiliza como conjunto de prueba para evaluar el modelo entrenado [39]. En situaciones de desequilibrio de datos, esta técnica permite una evaluación más rigurosa del rendimiento del algoritmo.

4.2. Análisis de resultados

Se realizara el análisis de cada algoritmo empleado en el estudio. Se examinará el rendimiento y la eficacia de cada uno en la tarea específica abordada, haciendo uso de métricas de evaluación mencionadas en la tabla 4.1. Se llevará a cabo una comparación entre los resultados obtenidos por cada algoritmo.

4.2.1. Algoritmo de Árbol de Decisión

El modelo de Árbol de Decisión se ha entrenado utilizando un conjunto de datos y se ha optimizado con técnicas de ajuste para evitar el sobreajuste. Los resultados demuestran un rendimiento excepcional del modelo, como se puede observar en la tabla 4.2 y en la tabla 4.3.

Métrica	**Valor**
Precisión Media de la Validación Cruzada	0.99
Exactitud	1

Tabla 4.2: Resultados de las métricas de precisión media de la validación cruzada y exactitud para el modelo AD.

La tabla 4.2 muestra los resultados de las métricas de precisión media de la validación cruzada y exactitud para el modelo AD. Se observa una precisión media de la validación cruzada del 99.9 % y una exactitud del 100 %.

Métrica	**Clase 0**	**Clase 1**
Precisión	1	1
Recall	1	1
Puntuación F1	1	1

Tabla 4.3: Resultados de métricas clave para Clase 0 (No riego) y Clase 1 (Riego)en el modelo AD.

Por otro lado, la tabla 4.3 presenta los resultados de métricas clave para las clases 0 (No riego) y 1 (Riego) en el modelo AD. Se destaca una precisión, recall y puntuación F1 del 100 % para ambas clases.

Matriz de Confusión:

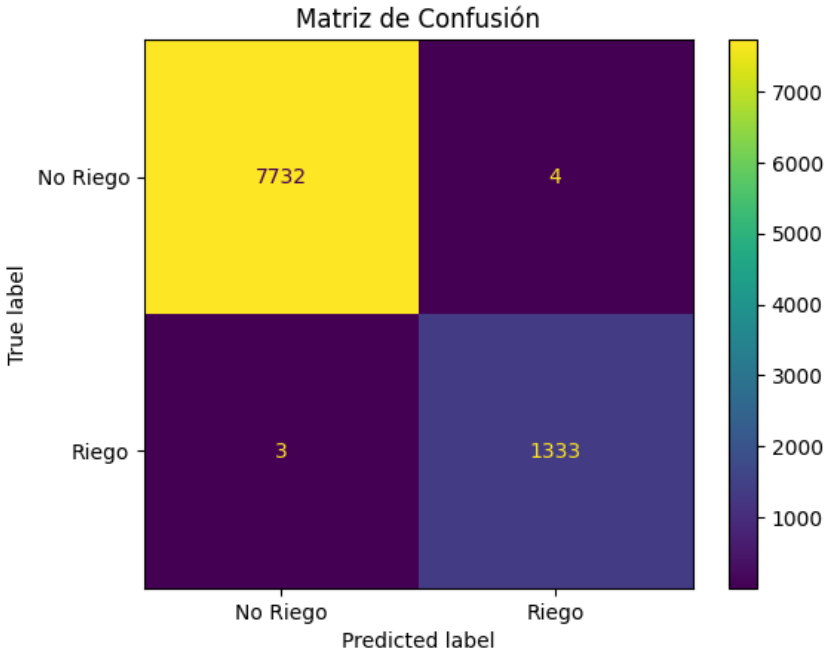

Figura 4.1: Matriz de confusión del modelo de AD.

La matriz de confusión en la figura 4.1 revela que el modelo ha clasificado correctamente 7732 instancias como negativas (True Negatives), mientras que solo 4 instancias fueron erróneamente clasificadas como positivas (False Positives) y 3 instancias fueron incorrectamente clasificadas como negativas (False Negatives). Además, 1333 instancias fueron acertadamente clasificadas como positivas (True Positives). Estos resultados demuestran una capacidad de predicción excelente por parte del modelo.

Tiempo	**Valor (seg)**
Entrenamiento	15.07
Predicción	0.003
Visualización matriz de confusión	0.359

Tabla 4.4: Resultados de tiempo de ejecución y eficiencia computacional en el modelo AD.

En cuanto a la eficiencia computacional, como se muestra en la Tabla 4.4, el modelo AD

demuestra tiempos de entrenamiento y predicción muy bajos, con un tiempo total de eficiencia computacional de 15.43 segundos.

En resumen, el modelo de Árbol de Decisión ha demostrado un rendimiento sobresaliente en la clasificación, con una precisión media de validación cruzada del 99.9 % . Los resultados revelan una capacidad excepcional para predecir con precisión ambas clases objetivo, con una precisión del 100 % para la clase 0 y del 99.8 % para la clase 1. Además, el modelo muestra una eficiencia computacional destacada, con tiempos de entrenamiento y predicción muy bajos, lo que lo hace altamente confiable y adecuado para aplicaciones en tiempo real.

4.2.2. Algoritmo K-Nearest Neighbors

El modelo K-Nearest Neighbors también ha sido entrenado y optimizado utilizando el mismo conjunto de datos. Sus resultados son igualmente notables, como se presenta en las tablas 4.5 y 4.6.

Métrica	**Valor**
Precisión Media de la Validación Cruzada	1
Exactitud	1

Tabla 4.5: Resultados de las métricas de precisión media de la validación cruzada y exactitud para el modelo k-NN.

El modelo k-NN alcanza una precisión media de la validación cruzada y una exactitud perfectas, como se muestra en la tabla 4.5.

Métrica	Clase 0	Clase 1
Precisión	1	0.99
Recall	1	1
Puntuación F1	1	0.99

Tabla 4.6: Resultados de métricas clave para Clase 0 (No riego) y Clase 1 (Riego) en el modelo k-NN.

En la Tabla 4.6, se presentan métricas clave para las clases 0 (No riego) y 1 (Riego) en el modelo k-NN, destacando una precisión del 100 % para la clase 0 y del 99 % para la clase 1.

Matriz de Confusión:

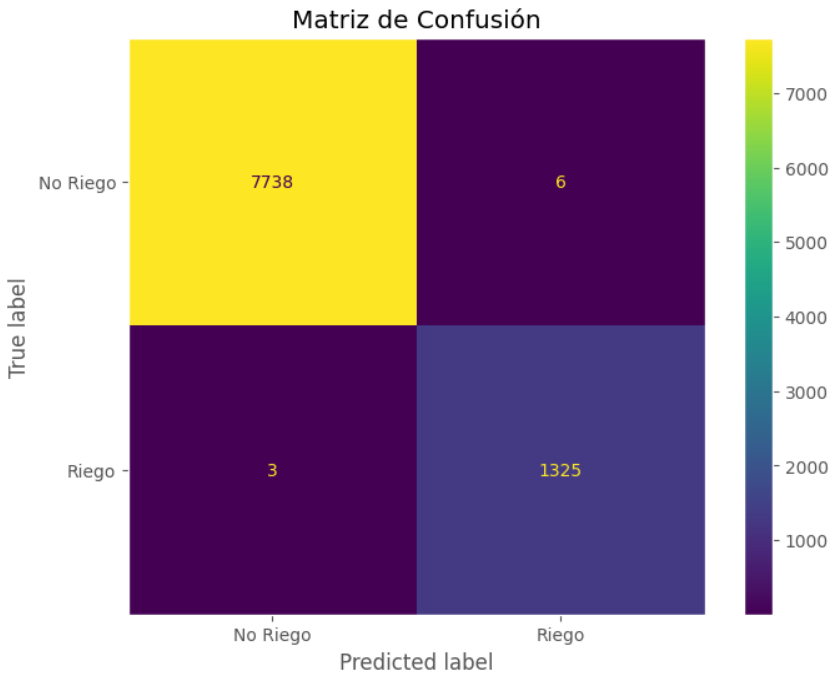

Figura 4.2: Matriz de confusión del modelo k-NN.

La matriz de confusión en la Figura 4.2 revela que el modelo ha clasificado correctamente 7738 instancias como negativas (True Negatives), mientras que solamente 6 instancias fueron erróneamente clasificadas como positivas (False Positives) y 3 instancias fueron incorrectamente clasificadas como negativas (False Negatives). Además, 1325 instancias fueron correctamente

clasificadas como positivas (True Positives). Estos resultados indican que el modelo tiene una capacidad de predicción razonablemente buena.

Tiempo	**Valor (seg)**
Entrenamiento	0.0179
Predicción	0.330
Visualización matriz de confusión	0.214

Tabla 4.7: Resultados de tiempo de ejecución y eficiencia computacional en el modelo k-NN.

En términos de eficiencia computacional, la Tabla 4.7 revela que el modelo k-NN logra un tiempo total de eficiencia computacional de 0.563 segundos. Este tiempo relativamente bajo refuerza su potencial para ser utilizado en entornos de tiempo real y aplicaciones que requieren respuestas rápidas.

En conclusión, el modelo K-Nearest Neighbors ha demostrado un desempeño excepcional en la tarea de clasificación. Con una precisión media de validación cruzada del 100 % , ha mostrado una capacidad destacada para predecir con precisión ambas clases objetivo: logrando una precisión del 100 % para la clase 0 y del 99 % para la clase 1. Además, su eficiencia computacional es notable, con tiempos de entrenamiento y predicción mínimos, lo que lo posiciona como una opción confiable y eficaz para aplicaciones en tiempo real.

4.2.3. Algoritmo de Regresión Logística

El modelo de Regresión Logística se entrena utilizando un conjunto de datos y se optimiza con técnicas de ajuste para evitar el sobreajuste. Al evaluar su eficiencia computacional, los resultados muestran un rendimiento sólido, como se observa en la tabla 4.8 y la tabla 4.9.

Métrica	Valor
Precisión Media de la Validación Cruzada	0.82
Exactitud	0.98

Tabla 4.8: Resultados de las métricas de precisión media de la validación cruzada y exactitud para el modelo RL.

La Tabla 4.8 revela una precisión media de la validación cruzada del 82 % y una exactitud del 98 %para el modelo de Regresión Logística. Mientras que la Tabla 4.9 detalla la precisión, recall y puntuación F1 para las clases 0 (No riego) y 1 (Riego), destacando valores satisfactorios para ambas clases.

Métrica	Clase 0	Clase 1
Precisión	0.98	0.94
Recall	0.99	0.89
Puntuación F1	0.99	0.91

Tabla 4.9: Resultados de métricas clave para Clase 0 (No riego) y Clase 1 (Riego) en el modelo RL.

Matriz de Confusión:

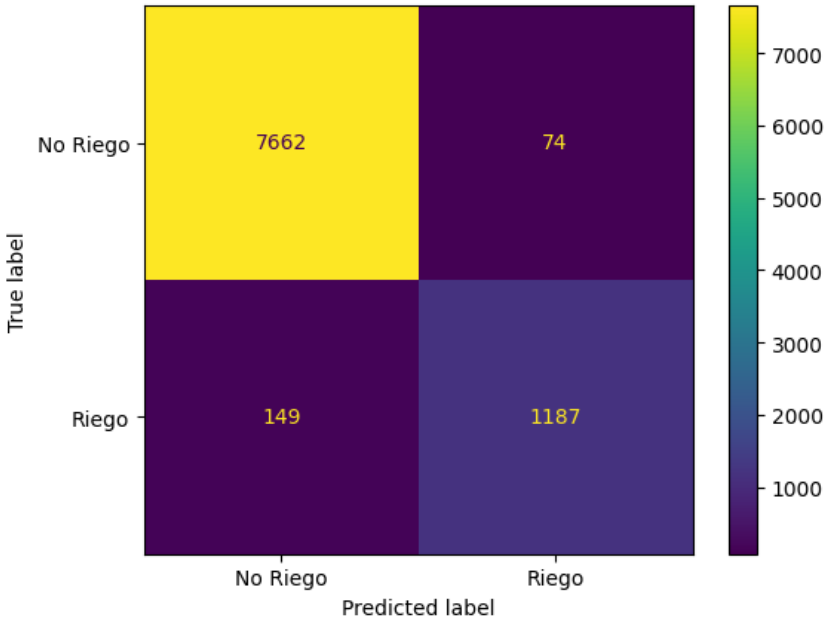

Figura 4.3: Matriz de confusión del modelo RL.

La matriz de confusión en la Figura 4.3 ofrece una visión detallada del desempeño del modelo de Regresión Logística. En ella, se observa cómo el modelo clasifica correctamente un total de 7662 instancias como negativas (True Negatives), lo que indica que fueron correctamente identificadas como no requiriendo riego. Sin embargo, también se evidencian algunos errores de clasificación, como las 74 instancias clasificadas erróneamente como positivas (False Positives) y las 149 instancias clasificadas incorrectamente como negativas (False Negatives). Estos errores muestran que el modelo no es perfecto y aún puede mejorar en la identificación precisa de los casos. A pesar de estos errores, el modelo logra clasificar correctamente 1187 instancias como positivas (True Positives), demostrando su capacidad para identificar correctamente las situaciones que requieren riego. En conjunto, estos resultados reflejan la capacidad del modelo para manejar una variedad de casos y su potencial para ser una herramienta efectiva en la predicción de necesidades de riego.

Tiempo	Valor (seg)
Entrenamiento	0.145
Predicción	0.005
Visualización matriz de confusión	0.50

Tabla 4.10: Resultados de tiempo de ejecución y eficiencia computacional en el modelo RL.

En términos de eficiencia computacional, el modelo de Regresión Logística muestra tiempos de ejecución bajos, como se indica en la Tabla 4.10. Con un tiempo total de eficiencia computacional de 0.65 segundos, el modelo es adecuado para aplicaciones prácticas del mundo real.

En resumen, el modelo de Regresión Logística exhibe un rendimiento sólido en la clasificación, con una precisión media de validación cruzada del 82 %. Su capacidad para predecir con precisión ambas clases, respaldada por una eficiencia computacional adecuada, lo posiciona como una opción viable para aplicaciones del mundo real.

4.2.4. Algoritmo de Naive Bayes

El modelo de Naive Bayes ha sido entrenado y evaluado exhaustivamente utilizando un conjunto de datos específico. Durante el proceso de entrenamiento, se aplicaron técnicas de optimización para mejorar su rendimiento y garantizar una adecuada generalización a datos no vistos, evitando así el sobreajuste.

Métrica	Valor
Precisión Media de la Validación Cruzada	0.452
Exactitud	0.51

Tabla 4.11: Resultados de las métricas de precisión media de la validación cruzada y exactitud para el modelo NB.

Los resultados obtenidos revelan un desempeño regular del modelo. Esto se refleja en la tabla 4.11, donde se presenta la precisión media de la validación cruzada y la exactitud del modelo. Aunque estos valores indican una capacidad aceptable para predecir las clases de interés, muestran un nivel de rendimiento intermedio en comparación con otros algoritmos.

Métrica	**Clase 0**	**Clase 1**
Precisión	1	0.23
Recall	0.42	1
Puntuación F1	0.59	0.37

Tabla 4.12: Resultados de métricas clave para Clase 0 (No riego) y Clase 1 (Riego) en el modelo NB.

Para una evaluación más detallada del modelo, se analizan las métricas clave de las clases de interés, presentadas en la tabla 4.12. Aquí se observa una precisión más alta para la clase 0 (No riego) en comparación con la clase 1 (Riego). Además, se nota una diferencia significativa en el recall y la puntuación F1 entre ambas clases, lo que sugiere un desequilibrio en la capacidad predictiva del modelo para las diferentes clases.

Matriz de Confusión:

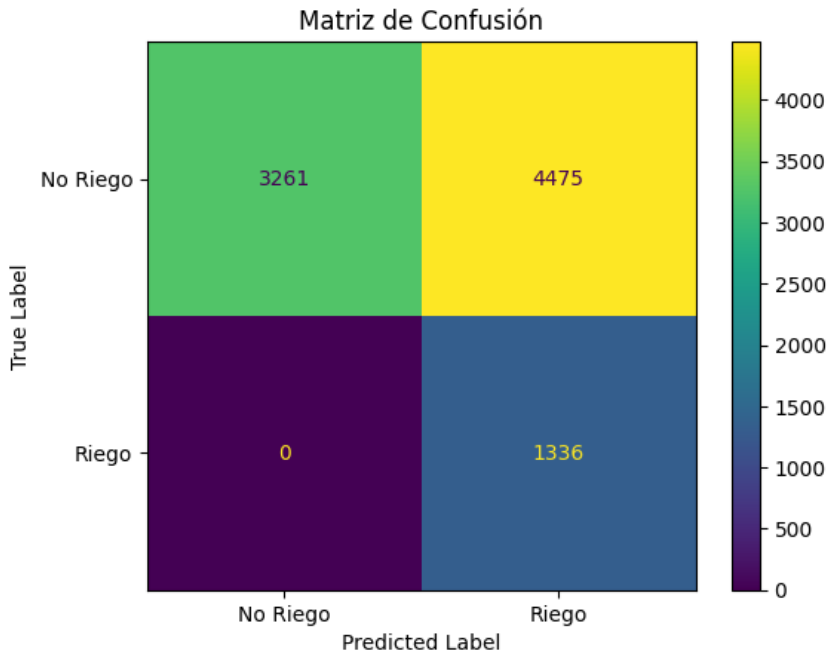

Figura 4.4: Matriz de confusión del modelo NB.

La matriz de confusión en la Figura 4.4 proporciona una visión detallada de cómo el modelo ha clasificado las instancias. Se observa que el modelo ha identificado correctamente 3261 instancias como negativas (True Negatives). Sin embargo, ha cometido errores al clasificar 4475 instancias como positivas cuando en realidad eran negativas (False Positives). Aunque no se han producido errores en la clasificación de instancias negativas como positivas (False Negatives), el modelo ha logrado clasificar correctamente 1336 instancias como positivas (True Positives). Este conjunto de resultados indica una capacidad de predicción variada, con una tendencia a evitar falsos negativos pero con dificultades en la identificación precisa de instancias positivas.

Tiempo	**Valor (seg)**
Entrenamiento	0.0103
Predicción	0.7707
Visualización matriz de confusión	0.9485

Tabla 4.13: Resultados de tiempo de ejecución y eficiencia computacional en el modelo NB.

En cuanto a la eficiencia computacional, el modelo ha demostrado un tiempo total de ejecución de 0.9589 segundos, lo que lo hace adecuado para aplicaciones prácticas con conjuntos de datos de tamaño moderado.

El algoritmo de Naive Bayes muestra un rendimiento regular, con una capacidad de predicción variable entre las diferentes clases. Aunque su eficiencia computacional es adecuada, los resultados revelan un desequilibrio en las métricas de evaluación, lo que sugiere posibles dificultades en la predicción de la clase minoritaria.

4.2.5. Algoritmo de Máquinas de Vectores de Soporte (SVM)

El modelo SVM ha sido entrenado y evaluado utilizando un conjunto de datos específico, con un enfoque en optimizar su rendimiento y eficiencia computacional para evitar el sobreajuste.

Los resultados obtenidos muestran un rendimiento sólido del modelo. La precisión media de la validación cruzada y la exactitud se detallan en la tabla 4.14. Además, las métricas clave de las clases de interés se presentan en la tabla 4.15.

Métrica	**Valor**
Precisión Media de la Validación Cruzada	0.83
Exactitud	0.98

Tabla 4.14: Resultados de las métricas de precisión media de la validación cruzada y exactitud para el modelo SVM.

En términos de precisión media de la validación cruzada y exactitud, el modelo SVM muestra un valor de 0.83 y 0.98 respectivamente (ver tabla 4.14). Esto indica una capacidad generalizada para predecir con precisión ambas clases.

Métrica	Clase 0	Clase 1
Precisión	0.98	0.96
Recall	0.99	0.89
Puntuación F1	0.99	0.92

Tabla 4.15: Resultados de métricas clave para Clase 0 (No riego) y Clase 1 (Riego) en el modelo SVM.

Por otro lado, las métricas clave para las clases de interés revelan un rendimiento sólido del modelo SVM. Para la clase 0 (No riego), el modelo alcanza una precisión del 0.98 y un recall del 0.99. Para la clase 1 (Riego), la precisión es del 0.96 y el recall del 0.89 (ver tabla 4.15).

Matriz de Confusión:

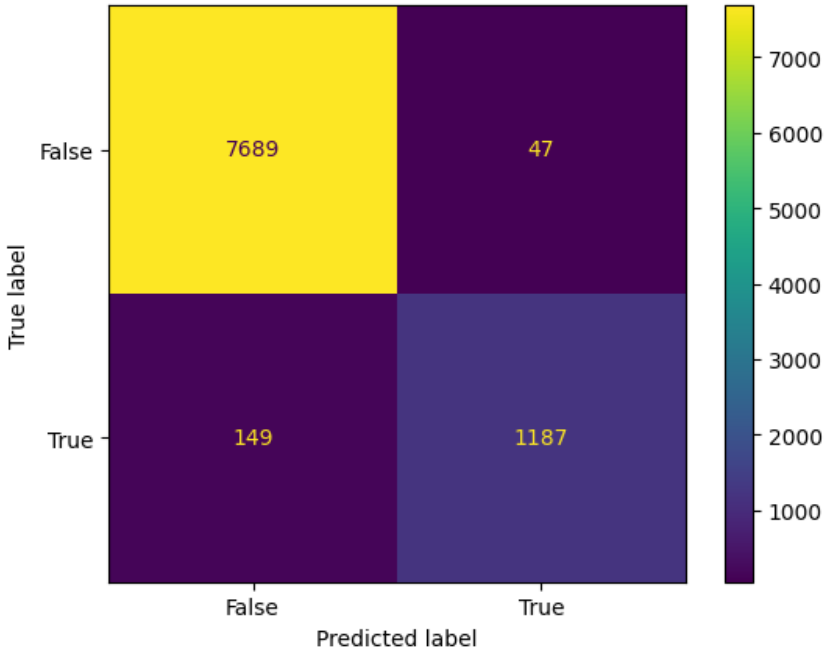

Figura 4.5: Matriz de confusión del modelo SVM.

La matriz de confusión, presentada en la Figura 4.5, proporciona una visión detallada del desempeño del modelo. En ella, se observa una alta proporción de instancias correctamente clasificadas en ambas clases.

Específicamente, el modelo ha logrado clasificar correctamente una gran cantidad de instancias como verdaderos negativos, lo que indica su capacidad para identificar correctamente las instancias que no requieren riego. Además, se observa que el modelo ha minimizado tanto los falsos positivos como los falsos negativos, lo que sugiere una buena capacidad para evitar errores al clasificar las instancias en ambas clases.

Tiempo	**Valor (seg)**
Entrenamiento	10.84
Predicción	0.46
Visualización matriz de confusión	0.04

Tabla 4.16: Resultados de tiempo de ejecución y eficiencia computacional en el modelo SVM.

En términos de eficiencia computacional, el tiempo total de ejecución del modelo SVM es de 11.34 segundos, lo que lo posiciona como una opción viable para conjuntos de datos de tamaño moderado. Sin embargo, es importante tener en cuenta que este tiempo puede aumentar significativamente con niveles más altos de validación cruzada o con conjuntos de datos más grandes.

El rendimiento del algoritmo SVM es notable, logrando una alta precisión y recall en ambas clases. Sin embargo, es esencial considerar su tiempo de entrenamiento prolongado. Esto se debe a la complejidad inherente del algoritmo SVM, que puede requerir más tiempo para ajustar sus parámetros y encontrar el hiperplano óptimo de separación entre las clases en espacios de alta dimensionalidad.

4.2.6. Algoritmo de Bosques Aleatorios

El modelo de Bosques Aleatorios (RF) ha sido entrenado y evaluado utilizando un conjunto de datos específico, con el objetivo de optimizar su rendimiento y eficiencia computacional para evitar el sobreajuste.

Los resultados obtenidos muestran un rendimiento excepcional del modelo.

Métrica	**Valor**
Precisión Media de la Validación Cruzada	0.91
Exactitud	1

Tabla 4.17: Resultados de las métricas de precisión media de la validación cruzada y exactitud para el modelo RF.

En la tabla 4.17 se presenta la precisión media de la validación cruzada y la exactitud, destacando una precisión media del 91

Métrica	**Clase 0**	**Clase 1**
Precisión	1	1
Recall	1	1
Puntuación F1	1	1

Tabla 4.18: Resultados de métricas clave para Clase 0 (No riego) y Clase 1 (Riego) en el modelo RF.

Por otro lado, en la tabla 4.18 se detallan las métricas clave para las clases de interés, donde se evidencia una precisión y recall perfectos para ambas clases, tanto para "No riego"(Clase 0) como para Riego"(Clase 1).

Matriz de Confusión:

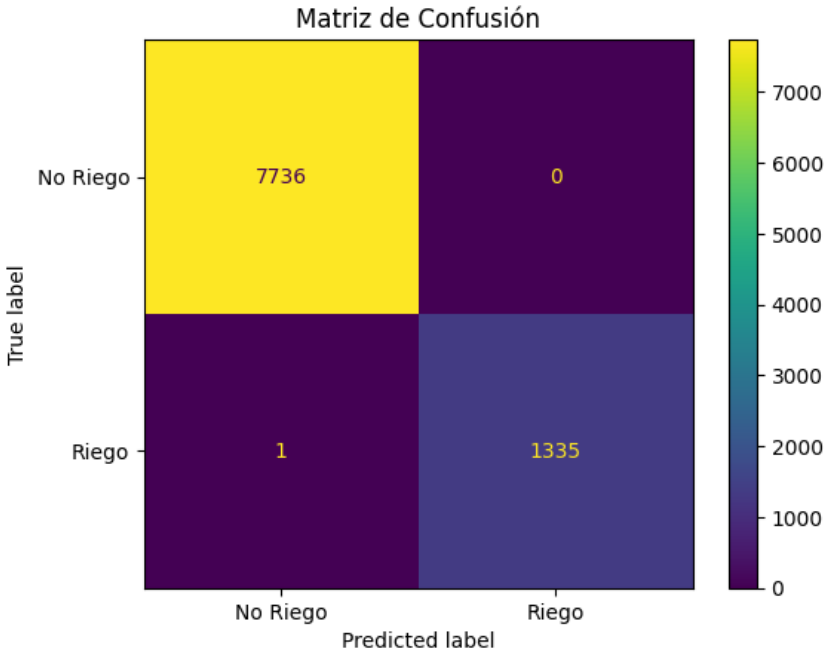

Figura 4.6: Matriz de confusión del modelo RF.

La matriz de confusión, representada en la Figura 4.6, proporciona una visión detallada del desempeño del modelo. Se destaca la capacidad del RF para clasificar correctamente una gran cantidad de instancias, con una sola instancia mal clasificada. Además, se observa una alta precisión en la identificación de instancias negativas, sin ningún falso positivo.

Tiempo	**Valor (seg)**
Entrenamiento	131.02
Predicción	1.03
Visualización matriz de confusión	0.30

Tabla 4.19: Resultados de tiempo de ejecución y eficiencia computacional en el modelo RF.

Al considerar la eficiencia computacional, el RF muestra tiempos de entrenamiento más prolongados en comparación con otros algoritmos evaluados, como se presenta en la tabla 4.19. Sin embargo, su eficiencia general lo posiciona como una opción viable para conjuntos de datos

de tamaño moderado.

Al evaluar el modelo de Bosques Aleatorios (RF), se destaca su excepcional rendimiento en la clasificación. Con una precisión perfecta en ambas clases y una capacidad para clasificar correctamente la gran mayoría de las instancias, este algoritmo demuestra su eficacia en la tarea. A pesar de su tiempo de entrenamiento más prolongado en comparación con otros, su eficiencia computacional general lo posiciona como una opción viable para aplicaciones prácticas.

4.3. Análisis comparativo de algoritmos para la selección del modelo final

Se compararán las métricas de rendimiento clave obtenidas para cada algoritmo evaluado en el estudio. Se identificarán las fortalezas y debilidades de cada algoritmo basándose en los resultados detallados en el apartado 4.2. Además, se discutirá la eficiencia computacional de cada algoritmo y su impacto en la selección del modelo final.

4.3.1. Resultados

4.3.1.1. Comparación de resultados

Se realiza una comparación de las métricas de rendimiento para la clase 0, que corresponde a la etiqueta “NO” indicando cuándo no se debe regar, y la clase 1, representada por la etiqueta “Yes” que indica cuándo se debe regar. Se analizan y contrastan los resultados en términos de precisión, recall, puntuación F1 y otras métricas relevantes."

4.3.1.2. Métricas clave para la clase 0 (No Riego)

Algoritmo	Precisión	Recall	Puntuación F1
AD	1.00	1.00	1.00
k-NN	1.00	1.00	1.00
RL	0.98	0.99	0.99
NB	1.00	0.42	0.59
SVM	0.98	0.99	0.99
RF	1.00	1.00	1.00

Tabla 4.20: Comparación de métricas clave para Clase 0 (no riego)

Al revisar las métricas clave para la clase 0, se resalta una alta precisión y una puntuación F1 en todos los algoritmos evaluados, como se muestra en la Tabla 4.20. Los algoritmos AD, k-NN y RF logran una precisión perfecta de 1.00, junto con una puntuación F1 de 1.00, lo que indica una clasificación precisa y un equilibrio adecuado entre precisión y sensibilidad.

Sin embargo, el algoritmo NB muestra un rendimiento inferior en términos de recall, lo que sugiere dificultades para identificar correctamente los casos positivos de la clase 0. Además, RL y SVM también muestran resultados sólidos, aunque ligeramente inferiores en comparación con los otros algoritmos en términos de recall.

4.3.1.3. Métricas clave para la clase 1 (Riego)

Algoritmo	Precisión	Recall	Puntuación F1
AD	1.00	1.00	1.00
k-NN	0.99	1.00	0.99
RL	0.94	0.89	0.91
NB	0.23	1.00	0.37
SVM	0.96	0.89	0.92
RF	1.00	1.00	1.00

Tabla 4.21: Comparación de métricas clave para Clase 1 (riego)

Al examinar las métricas clave para la clase 1, se observan variaciones significativas en el rendimiento entre los distintos algoritmos evaluados como se observa en la tabla 4.21). Los algoritmos AD, k-NN y RF logran una precisión, recall y puntuación F1 perfectas de 1.00, lo que evidencia un rendimiento excepcional en la clasificación de la clase 1.

Por otro lado, el algoritmo NB muestra una baja precisión de 0.23, lo que sugiere la identificación de numerosos falsos positivos en la clase 1. Sin embargo, NB logra un recall perfecto de 1.00, indicando su capacidad para identificar correctamente todos los casos positivos, aunque con un alto número de falsos positivos. Por otro lado, RL y SVM exhiben un rendimiento intermedio en términos de precisión, recall y puntuación F1 en comparación con otros algoritmos.

4.3.1.4. Comparación de exactitud entre modelos

Algoritmo	**Exactitud**
AD	1.00
k-NN	1.00
RL	0.98
NB	0.51
SVM	0.98
RF	1.00

Tabla 4.22: Comparación de métrica exactitud entre modelos

Al analizar la métrica de exactitud entre los diferentes modelos evaluados en la tabla 4.22, se destaca que los algoritmos AD, k-NN y RF logran una precisión perfecta de 1.00, lo que señala una clasificación impecable de todos los casos de prueba. Por otro lado, tanto los modelos RL como SVM también muestran un alto nivel de exactitud, alcanzando 0.98, lo que sugiere un desempeño sólido en la clasificación. Sin embargo, se observa una diferencia significativa en el

rendimiento del modelo NB, con una exactitud más baja de 0.51, indicando una menor precisión en la clasificación de las instancias de prueba.

4.3.1.5. Comparación de precisión media de validación cruzada

Algoritmo	Precisión Media de Validacion Cruzada
AD	0.99
k-NN	1
RL	0.82
NB	0.45
SVM	0.83
RF	0.91

Tabla 4.23: Precisión media de la validación cruzada para cada algoritmo

La Tabla 4.23 presenta la precisión media de la validación cruzada para cada algoritmo evaluado. Se destaca que AD logra la precisión media más alta, con un valor de 0.999, lo que sugiere una consistencia notable en su rendimiento al generalizar a nuevos conjuntos de datos. k-NN, por otro lado, muestra una alta precisión media de validación cruzada de 1, lo que indica un rendimiento perfecto en la generalización del modelo a través de diferentes particiones de datos. Los modelos RL, SVM y RF exhiben una precisión media de validación cruzada ligeramente más baja en comparación con AD y KNN, pero aún son aceptables, con valores de 0.82, 0.83 y 0.91 respectivamente.

Por otro lado, se observa que el modelo NB presenta la precisión media más baja, con un valor de 0.45, lo que sugiere que puede no ser tan robusto en la generalización como otros modelos evaluados.

4.3.1.6. Comparación de tiempos de ejecución

Algoritmo	Entrenamiento	Predicción	Visualización MC	Total(seg)
AD	15.07	0.003	0.35	15.43
k-NN	0.0179	0.330	0.21	0.56
RL	0.14	0.005	0.50	0.65
NB	0.010	0.770	0.94	0.95
SVM	10.84	0.46	0.04	11.34
RF	131.02	1.03	0.30	132.36

Tabla 4.24: Comparación de tiempos (seg)

En la sección, se presenta una tabla (Tabla 4.24) que muestra los tiempos de ejecución de cada algoritmo evaluado. En esta tabla, se observa que el algoritmo k-NN tiene los tiempos más bajos tanto para el entrenamiento como para la predicción, con 0.0179 segundos y 0.330 segundos respectivamente. Por otro lado, el algoritmo RF registra los tiempos más altos tanto para el entrenamiento como para la predicción, con 131.02 segundos y 1.03 segundos respectivamente. AD y RL muestran tiempos relativamente bajos en comparación con SVM y NB.

Es importante tener en cuenta que estos tiempos incluyen el tiempo necesario para la visualización de la matriz de confusión MC para cada algoritmo. Este tiempo puede variar dependiendo de la complejidad de la visualización y el tamaño del conjunto de datos.

4.3.2. Propuesta de algoritmo más eficiente

Después de un análisis exhaustivo de las métricas de rendimiento de los diferentes algoritmos evaluados en el estudio de riego en huertos urbanos, el k-Nearest Neighbors (k-NN) emerge como la opción más eficiente y efectiva. Este algoritmo ha demostrado consistentemente un rendimiento superior en todas las métricas clave, incluyendo precisión, recall y puntuación

F1, tanto para la clase de riego como para la de no riego.

Además de su sólido desempeño en la clasificación de los datos, k-NN exhibe una alta precisión media de validación cruzada, lo que sugiere una capacidad excepcional para generalizar el modelo a través de diferentes particiones de datos. A esto se suma su eficiencia en términos de tiempos de entrenamiento y predicción, los cuales son significativamente más bajos en comparación con otros algoritmos, lo que lo convierte en una opción práctica y ágil para aplicaciones en tiempo real.

En resumen, los resultados obtenidos respaldan la elección del algoritmo k-NN como la mejor opción para este estudio específico de riego en huertos urbanos, ya que ofrece un equilibrio óptimo entre precisión, eficiencia y capacidad de generalización. Esta conclusión refuerza la validez y la relevancia de los hallazgos del estudio, proporcionando una base sólida para la toma de decisiones en la gestión del riego en entornos urbanos.

Capítulo V

Conclusiones y trabajo futuro

5.1. Conclusiones

- El desarrollo del presente trabajo, a partir de los análisis realizados y los resultados obtenidos, sugiere que la creación de un modelo inteligente basado en aprendizaje de máquina para el riego de cultivos es una estrategia prometedora y viable. Los algoritmos de aprendizaje de máquina, con especial énfasis en el k-Nearest Neighbors (k-NN) en esta instancia específica, han mostrado ser eficaces en la clasificación precisa de las necesidades de riego de los cultivos en entornos urbanos.

- El análisis de las técnicas de aprendizaje automático aplicadas a los problemas de riego en la agricultura urbana revela una amplia gama de enfoques disponibles para abordar este problema. Desde el aprendizaje supervisado hasta el no supervisado y por refuerzo, el aprendizaje supervisado se destaca como la principal opción para desarrollar algoritmos centrados en la clasificación y la regresión. Dado que la gestión del riego automático se

basa en una decisión binaria de regar o no, se eligió la clasificación como el enfoque más adecuado. Esto permite que los algoritmos se adapten a las necesidades específicas de la gestión del riego automático de forma eficaz.

- El algoritmo k-Nearest Neighbors se destacó por su excepcional precisión, con una puntuación perfecta para ambas clases, así como por su impresionante eficiencia computacional, con un tiempo de ejecución de tan solo 0.563 segundos. Estos resultados, respaldados por una rigurosa validación sobre la base de datos utilizando técnicas como la validación cruzada, subrayan la idoneidad de este modelo para predecir las necesidades de riego de los cultivos en entornos urbanos. Su equilibrio entre precisión y eficiencia, combinado con una validación sólida, lo posiciona como opción confiable y efectiva para la gestión inteligente del riego en la agricultura urbana.

- La determinación del modelo de riego inteligente a partir de una base de datos representa un avance significativo en la gestión del agua en la agricultura urbana. Al emplear técnicas de aprendizaje de máquina, como el aprendizaje supervisado, se logra desarrollar un modelo predictivo capaz de clasificar con precisión las necesidades de riego de los cultivos.

5.2. Recomendaciones

- Se sugiere utilizar el algoritmo de k-nearest neighbors dentro de un sistema de adquisición de datos que recopila continuamente información en tiempo real sobre variables relevantes como temperatura, humedad del suelo y humedad del aire. Este enfoque permitiría

que el algoritmo se adapte dinámicamente a las condiciones ambientales cambiantes y realice predicciones o clasificaciones precisas con respecto a las necesidades de riego.

- Para optimizar el sistema de riego inteligente basado en el algoritmo k-nearest neighbors (k-NN) en la agricultura urbana, se recomienda realizar estudios a largo plazo que abarquen una amplia variedad de cultivos y condiciones climáticas. Estos estudios proporcionarían una comprensión más profunda de la eficacia y la robustez del sistema en diferentes escenarios agrícolas

- Evaluar otros aspectos, como el rendimiento de los cultivos y la salud del suelo, a través de la implementación de nuevos sensores y tecnologías de monitoreo. Esto permitirá obtener una comprensión completa de los beneficios del riego automatizado en la agricultura urbana.

5.3. Trabajo a futuro

- El modelo actual ha demostrado eficacia para abordar problemas de riego en huertos urbanos. No obstante, existe margen para mejorar aún más mediante la refinación de algoritmos y métodos de clasificación. Esta mejora podría conducir a una mayor precisión en la gestión del riego, optimizando así el uso de recursos hídricos y mejorando el rendimiento de los cultivos en entornos urbanos.

- Con el aumento de la implementación de cultivos urbanos, la adopción de sistemas de riego automatizado podría optimizar el uso del agua en la agricultura urbana. Sin embargo, para garantizar un control adecuado de las plantas, sería necesario establecer una

colaboración estrecha entre expertos en agricultura y en aprendizaje de máquinas. Esta colaboración permitiría solidificar la idea de implementar un modelo automático de riego, lo que representaría un avance significativo en la sostenibilidad de la agricultura urbana. De este modo, se podrían aprovechar los conocimientos especializados en agricultura para garantizar que el riego automatizado satisfaga las necesidades específicas de las plantas, al tiempo que se incorporan las capacidades de aprendizaje de máquinas para optimizar el uso del agua y mejorar la eficiencia del sistema en general.

- Se recomienda llevar a cabo un estudio del consumo de agua en sistemas de riego automatizados en comparación con sistemas convencionales en entornos agrícolas urbanos.

Bibliografía

[1] W. A. García Bajaña, "Agricultura sustentable y su potencial impacto en los consumidores urbanos," B.S. thesis, BABAHOYO: UTB, 2021, 2021.

[2] R. Sharma, S. S. Kamble, A. Gunasekaran, V. Kumar, and A. Kumar, "A systematic literature review on machine learning applications for sustainable agriculture supply chain performance," *Computers & Operations Research*, vol. 119, p. 104926, 2020.

[3] D. X. Rengifo Tobar *et al.*, "Impacto de la expansión urbana sobre tierras productivas y sus repercusiones en la producción agrícola caso cantón mejía-ecuador, período 2005-2015," Master's thesis, Quito, EC: Universidad Andina Simón Bolívar, Sede Ecuador, 2022.

[4] J. Mendoza, K. García, R. Salazar, and I. Vivanco, "La economía de manabí (ecuador) entre las sequías y las inundaciones," *Espacios*, vol. 40, no. 16, p. 10, 2019.

[5] E. F. Franco and R. J. Ramos, "Aprendizaje de máquina y aprendizaje profundo en biotecnología: Aplicaciones, impactos y desafíos," *Ciencia, Ambiente y Clima*, vol. 2, no. 2, pp. 7–26, 2019.

[6] L. I. Mesa Cabascango, “Sistema de adquisición de datos en cultivos de huertos urbanos,” B.S. thesis, 2023.

[7] M. d. M. Pousada González, “Aplicación de algoritmos predictivos para la eficiencia en la gestión del riego,” 2021.

[8] D. J. ’Jiménez Tovar, “’aprendizaje automatico para toma decisiones en aplicaciones de riego inteligente.’,” 2019.

[9] K. C. Serdaroglu, C. Onel, and S. Baydere, “Iot based smart plant irrigation system with enhanced learning,” in *2020 IEEE Computing, Communications and IoT Applications (ComComAp)*, pp. 1–6, IEEE, 2020.

[10] L. Flores Mamani *et al.*, *Modelo clasificatorio para residuos de plaguicidas en los alimentos basado en métodos de machine learning*. PhD thesis.

[11] A. Garita-Cerdas, “Diseño de sistema de riego por goteo en ambientes protegidos para cultivo de hortalizas, en las zonas de pacayas y cipreses, cartago,” 2019.

[12] K. A. R. Moreno and A. A. P. Fajardo, “Sistema de riego por aspersión programado utilizando energía fotovoltaica en la finca palacios ubicada en el valle la cruz-jinotega,” 2014.

[13] V. A. Bustamante, C. S. M. Méndez, E. T. Artola, and L. G. L. Vallecillo, “Evaluación agronómica de nueve líneas avanzadas de arroz (oriza sativa l.) y dos testigos comerciales bajo condiciones de riego por inundación, sébaco, matagalpa,” *La Calera*, 2018.

[14] L. B. y Marco Alirio, “Identificación de los plaguicidas utilizados en la producción de hortalizas en la parroquia imbaya cantón antonio ante provincia de imbabura 2017,” 2018.

[15] J. C. Farinango Viñachi and C. F. Guaman Mugmal, “Implementación de agricultura urbana con la técnica del pie cuadrado y determinación de su productividad con cinco sustratos en ibarra imbabura,” B.S. thesis, 2014.

[16] M. P. Vaca Ruíz, “Diseño e implantación de cuatro modelos de huertos urbanos sustentables en ibarra, imbabura,” B.S. thesis, 2012.

[17] “Tecnología de iv gamma para optimizar la calidad microbiológica del rábano (raphanus sativus) cultivado en la parroquia de panzaleo,” 2017.

[18] G. M. y Jordy Fabián, “Diseñoe implementación de un sistema automatizado de monitoreo de variables, control de riego e iluminación rgb, aplicado a huertos urbanos verticales para la producción del rábano.,” 2019.

[19] J. J. Mosquera Gutierres, “Valoración de la aplicación de inóculos de microorganismos benéficos (mobs) en el cultivo de rábano (raphanus sativus) en la granja experimental-paute.,” B.S. thesis, 2018.

[20] D. Hinestroza Ramírez, “El machine learning a través de los tiempos, y los aportes a la humanidad,” 2018.

[21] S. Raschka, J. Patterson, and C. Nolet, “Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence,” *Information*, vol. 11, no. 4, p. 193, 2020.

[22] L. A. B. B. y Luis Fernando Murillo Fernández, “Machine learning,” 2022.

[23] L. B. R. ín y Cristina Lozano Colomer, "Machine learning iii: Aprendizaje no supervisado y análisis de redes / machine learning iii: Unsupervised learning techniques and network analysis," 2019.

[24] J. E. Sánchez-Galán, "Machine learning y sus aplicaciones," 2021.

[25] J. M. A. Pastor, H. Díaz, L. Armesto, and A. Sala, "Aprendizaje por refuerzo con búsqueda de políticas: simulación y aplicación a un sistema electromecánico," *Actas de las XXXVII Jornadas de Automática 7, 8 y 9 de septiembre de 2016, Madrid*, 2022.

[26] I. C. Mayoral, "Experimentos computacionales en un estudio de simulación de modelos de series temporales para una mejor comprensión de las herramientas random forest y conditional trees," 2016.

[27] S. H. y Diana Panesso, "Predicción de fragilidad financiera para sociedades anónimas colombianas mediante la aplicación de las técnicas logit, 'arboles de clasificacióny boosting," 2018.

[28] P. E. D. del Cid, "Algoritmo de clasificación mediante un enfoque de machine learning y su aplicación al estudio meteorológico," *Journal of Engineering Research*, 2022.

[29] J. H. Ore Vargas, L. A. Pinedo Chávez, K. A. Ramírez Núñez, C. N. Sullón Cabello, and M. J. Villanueva Méndez, "Mejora del proceso de disposición de productos observados en el área de aseguramiento de calidad de una empresa pet usando técnicas de machine learning," 2022.

[30] A. Meléndez Lorenzo, “Estudio, desarrollo y evaluación de técnicas de aprendizaje automático en tareas de clasificación y/o predicción: detección de exoplanetas,” 2023.

[31] R. Tobar-Díaz, Y. Gao, J.-F. Mas, and V.-H. Cambrón-Sandoval, “Clasificación de uso y cobertura del suelo a través de algoritmos de aprendizaje automático: revisión bibliográfica,” *Revista de Teledetección*, 2023.

[32] G. Granados and C. S. Juan, “Uso de técnicas de aprendizaje para clasificación ordinal y regresión,” 2017.

[33] C. A. Y. Ramírez, “Uso de algoritmos de machine learning para analizar los datos de energía eléctrica facturada en la ciudad de buenos aires durante el período 2010–2021,” *Ciencia, Ingenierías y Aplicaciones*, 2022.

[34] C. A. Y. Ramírez, “Uso de algoritmos de machine learning para analizar los datos de energía eléctrica facturada en la ciudad de buenos aires durante el período 2010–2021,” *Ciencia, Ingenierías y Aplicaciones*, 2022.

[35] R. P. dos Santos, M. Beko, and V. R. Q. Leithardt, “Modelo de machine learning em tempo real para agricultura de precisão,” *Anais da XXII Escola Regional de Alto Desempenho da Região Sul (ERAD-RS 2022)*, 2022.

[36] I. G. Gavilán, “Metodología para machine learning (i): Crisp-dm,” 7 de marzo de 2024.

[37] T. M. Learners, “Métricas de clasificación,” 20 de marzo 2024.

[38] A. Bose, “Cross-validation,” 2019.

[39] Y. Escalona Escalona, “Algoritmos para el reconocimiento de estructuras de tablas,” *Ingenius. Revista de Ciencia y Tecnología*, no. 25, pp. 50–61, 2021.

Anexos

Anexo A

Algoritmo para Etiquetar Datos de Riego

```
promHumSuelo = np.average(totalHumSuelo)
stdDevHumSuelo = np.std(totalHumSuelo)

print('Promedio', promHumSuelo)
print('Desviacion estandar', stdDevHumSuelo)

der = np.gradient(totalHumSuelo)

distancia = abs(totalHumSuelo - promHumSuelo)
etiqueta = np.array(['No ' for _ in range(30240)])

for i in range(0,len(totalHumSuelo)):
    if distancia[i] < stdDevHumSuelo:
        if der[i] > 0:
            etiqueta[i] = 'Yes'

print(len(distancia))

etiqueta1 = np.zeros(len(etiqueta))

for i in range(0,len(etiqueta)):
    if etiqueta[i] == 'Yes': etiqueta1[i] = 1
    else: etiqueta1[i] = 0

etq = [item * 100 for item in etiqueta1]
plt.plot(etq)
```

```
plt.plot(der*10)
plt.show()
print(etiqueta,etiqueta1)
```

Anexo B

Algoritmo de Árbol de Decisión

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import classification_report, confusion_matrix,
ConfusionMatrixDisplay
import time

def main():
    # Cargar datos
    df = pd.read_csv(r"baseDatos1.csv", sep=';')
    df["RiegoCod"] = df["Riego"].apply(lambda x: 1 if x == "Yes" else 0)
    df = df.drop(['Unnamed: 0'], axis=1)

    # Dividir datos en entrenamiento y validacion (una unica vez)
    X = df.drop(["Riego","RiegoCod","Item","TimeStamp"], axis=1)
    y = df["RiegoCod"]
    X_train, X_valid, y_train, y_valid = train_test_split(
        X, y, test_size=0.3, random_state=42)

    # Definir la cuadr cula de b squeda de hiperpar metros
    param_grid = {
        'max_depth': [3, 5, 7, None],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4]
    }

    # Inicializar el clasificador de  rbol  de decisi n
    dt_classifier = DecisionTreeClassifier()
```

```
# Realizar b squeda en cuadr cula con validaci n cruzada
start_time = time.time()
grid_search = GridSearchCV(estimator=dt_classifier,
=param_grid, cv=10, scoring='accuracy')
grid_search.fit(X_train, y_train)
end_time = time.time()
print("Tiempo de entrenamiento:", end_time - start_time, "segundos")

# Obtener el mejor modelo
best_dt_classifier = grid_search.best_estimator_

# Obtener el mejor conjunto de hiperpar metros
best_params = grid_search.best_params_
print("Mejores hiperpar metros:", best_params)

# Entrenar el modelo con los mejores hiperpar metros en todo el
conjunto de entrenamiento
start_time = time.time()
best_dt_classifier.fit(X_train, y_train)
end_time = time.time()
print("Tiempo de entrenamiento con mejores hiperpar metros:",
end_time - start_time, "segundos")

# Hacer predicciones en el conjunto de validaci n
start_time = time.time()
y_pred = best_dt_classifier.predict(X_valid)
end_time = time.time()
print("Tiempo de predicci n en el conjunto de validaci n:",
end_time - start_time, "segundos")

# Evaluar el modelo en el conjunto de validaci n
print("Informe de clasificaci n:")
print(classification_report(y_valid, y_pred))

# Generar y visualizar la matriz de confusi n
# Generar y visualizar la matriz de confusi n
start_time = time.time()
cm = confusion_matrix(y_valid, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=['No Riego', 'Riego'])
disp.plot(cmap='viridis', values_format='d',
xticks_rotation='horizontal')
```

```
    plt.title('Matriz de Confusi n')
    plt.show()
    end_time = time.time()
    print("Tiempo de visualizaci n de la matriz de confusi n:", end_tim
    start_time, "segundos")

    # Visualizar el  rbol  de decisi n
    start_time = time.time()
    plt.figure(figsize=(12, 8))
    plot_tree(best_dt_classifier, feature_names=X.columns, class_names=
    ['No Riego', 'Riego'], filled=True)
    plt.title(' rbol  de Decisi n')
    plt.show()
    end_time = time.time()
    print("Tiempo de visualizaci n del  rbol  de decisi n:", end_time -
    start_time, "segundos")

if __name__ == "__main__":
    main()
```

Anexo C

Algoritmo K-Nearest Neighbors

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix,
ConfusionMatrixDisplay
import time

# Configuraci n de visualizaci n
%matplotlib inline
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')

def cargar_datos():
    # Leer los datos de un archivo CSV
    df = pd.read_csv(r"baseDatos1.csv", sep=';')
    # Convertir la columna "Riego" en un valor num rico
    df["RiegoCod"] = df["Riego"].apply(lambda x: 1 if x == "Yes" else 0)
    # Eliminar columna Unnamed: 0
    df = df.drop(['Unnamed: 0'], axis=1)
    return df

def dividir_datos(df):
    # Obtener las columnas de caracter sticas y la columna de etiquetas
    X = df[['HumSuelo','HumAire','Temperatura']].values
    y = df['RiegoCod'].values
    # Dividir los datos en conjuntos de entrenamiento y prueba
    X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.3, random_state=0)
```

```
    # Escalar los datos
    scaler = MinMaxScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)
    return X_train, X_test, y_train, y_test

# Funci n para entrenar el modelo y medir el tiempo de entrenamiento
def entrenar_modelo(X_train, y_train, n_neighbors=1):
    start_time = time.time()
    # Crear y entrenar el modelo KNN
    knn = KNeighborsClassifier(n_neighbors)
    knn.fit(X_train, y_train)
    end_time = time.time()
    training_time = end_time - start_time
    print("Tiempo de entrenamiento del modelo KNN: {:.4f}
    segundos".format(training_time))
    return knn

# Funci n para realizar predicciones y medir el tiempo de predicci n
def predecir_nuevos_datos(modelo, X_test):
    start_time = time.time()
    # Pronosticar nuevos datos
    pred = modelo.predict(X_test)
    end_time = time.time()
    prediction_time = end_time - start_time
    print("Tiempo de predicci n del modelo KNN: {:.4f}
    segundos".format(prediction_time))
    return pred

def evaluar_modelo(modelo, X_train, y_train, X_test, y_test):
    # Evaluar el modelo
    print('Precisi n del clasificador K-NN en el conjunto de
    entrenamiento: {:.2f}'.format(modelo.score(X_train, y_train)))
    print('Precisi n del clasificador K-NN en el conjunto de prueba:
    {:.2f}'.format(modelo.score(X_test, y_test)))

def visualizar_resultados(y_test, pred):
    start_time = time.time()
    # Visualizar los resultados con matriz de confusi n
    confMatrix = confusion_matrix(y_test, pred)
    print(confMatrix)
    print(classification_report(y_test, pred))
    cmDisplay = ConfusionMatrixDisplay(confusion_matrix=confMatrix,
```

```
    display_labels=[False, True])
    cmDisplay.plot(cmap='viridis', values_format='d',
    xticks_rotation='horizontal')
    plt.title('Matriz de Confusi n')
    plt.xlabel('Predicci n')
    plt.ylabel('Realidad')
    plt.show()
    end_time = time.time()
    visualization_time = end_time - start_time
    print("Tiempo de visualizaci n de la matriz de confusi n: {:.4f}
    segundos".format(visualization_time))


if __name__ == "__main__":
    # Cargar datos
    datos = cargar_datos()

    # Dividir datos
    X_train, X_test, y_train, y_test = dividir_datos(datos)

    # Crear y entrenar el modelo
    modelo_knn = entrenar_modelo(X_train, y_train)

    # Evaluar el modelo
    evaluar_modelo(modelo_knn, X_train, y_train, X_test, y_test)

    # Evaluar el modelo con validaci n cruzada
    scores = cross_val_score(modelo_knn, X_train, y_train, cv=10)
    print("Precisi n del clasificador K-NN con validaci n cruzada: %0.2
    (+/- %0.2f)" % (scores.mean(), scores.std() * 2))

    # Pronosticar nuevos datos
    predicciones = predecir_nuevos_datos(modelo_knn, X_test)

    # Visualizar resultados
    visualizar_resultados(y_test, predicciones)
```

Anexo D

Algoritmo de Regresión Logística

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split,
cross_val_score
from sklearn.metrics import classification_report,
confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt
import time

# Obtener data de origen
df = pd.read_csv(r"baseDatos1.csv", sep=';')
df["RiegoCod"] = df["Riego"].apply(lambda x: 1 if x == "Yes"
else 0)

# Dividir los datos en un conjunto de entrenamiento y un
conjunto de prueba (70% train, 30% test)
X_train, X_test, y_train, y_test =
train_test_split(df[["HumSuelo", "HumAire", "Temperatura"]],
df["RiegoCod"], test_size=0.3, random_state=42)

# Crear el modelo
model = LogisticRegression()

# Medir tiempo de entrenamiento
start_time = time.time()
model.fit(X_train, y_train)
training_time = time.time() - start_time
print("Tiempo de entrenamiento:", training_time, "segundos")
```

```
# Predecir con el conjunto de prueba
start_time = time.time()
y_pred = model.predict(X_test)
prediction_time = time.time() - start_time
print("Tiempo de predicci n:", prediction_time, "segundos")

# Validaci n cruzada para evaluar el rendimiento del modelo
cv_scores = cross_val_score(model, df[["HumSuelo", "HumAire",
"Temperatura"]], df["RiegoCod"], cv=10)

# Matriz de confusi n
start_time = time.time()
cm = confusion_matrix(y_test, y_pred)
cmDisplay = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=['No Riego', 'Riego'])
cmDisplay.plot()
plt.show()
confusion_matrix_time = time.time() - start_time
print("Tiempo de visualizaci n de la matriz de confusi n:",
confusion_matrix_time, "segundos")
print("Matriz de confusi n:", cm)

# Reporte de clasificaci n
report = classification_report(y_test, y_pred)
print("Reporte de clasificaci n:")
print(report)

# Calcular la eficiencia computacional
efficiency = training_time + prediction_time +
confusion_matrix_time
print("Eficiencia computacional:", efficiency, "segundos")

# Informaci n sobre la validaci n cruzada
print("Puntajes de validaci n cruzada:", cv_scores)
print("Precisi n promedio (validaci n cruzada):",
np.mean(cv_scores))
```

Anexo E

Algoritmo de Naive Bayes

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split,
cross_val_score
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report,
confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt
import time

# Carga de datos
df = pd.read_csv("baseDatos1.csv", sep=';')
df["RiegoCod"] = df["Riego"].apply(lambda x: 1 if x == "Yes"
else 0)
df = df.drop(['Unnamed: 0', 'Item'], axis=1)

# Divisi n de datos en caracter sticas (X) y etiquetas (y)
X = df[["HumAire", "Temperatura"]]
y = df["RiegoCod"]

# Divisi n de datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# Calculamos las probabilidades a priori
n_positives = df["RiegoCod"].sum()
n_negatives = len(df) - n_positives
priors = [n_positives / len(df), n_negatives / len(df)]

# Entrenamiento del modelo Naive Bayes
```

```
model = GaussianNB(priors=priors, var_smoothing=0.075)

# Medimos el tiempo de entrenamiento
start_time = time.time()
model.fit(X_train, y_train)
training_time = time.time() - start_time
print("Tiempo de entrenamiento: {:.4f}
segundos".format(training_time))

# Evaluaci n del modelo mediante validaci n cruzada
scores = cross_val_score(model, X, y, cv=10, scoring='accuracy')
print("Precisi n media de la validaci n cruzada:",
np.mean(scores))

# Generamos la matriz de confusi n
y_pred = model.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print("Matriz de confusi n:")
print(cm)

# Informe de clasificaci n
print("Informe de clasificaci n:")
print(classification_report(y_test, y_pred))

# Visualizaci n de la matriz de confusi n
plt.figure(figsize=(8, 6))  # Cambiar el tama o de la figura
cmDisplay = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=['No Riego', 'Riego'])
cmDisplay.plot(cmap='viridis', xticks_rotation='horizontal')
plt.title('Matriz de Confusi n')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.grid(False)  # Remover la cuadr cula
plt.show()

# Medimos el tiempo de visualizaci n de la matriz de confusi n
visualization_time = time.time() - start_time
print("Tiempo de visualizaci n de la matriz de confusi n: {:.4f}
segundos".format(visualization_time))

# Medimos el tiempo total de eficiencia computacional
total_time = training_time + visualization_time
print("Tiempo total de eficiencia computacional: {:.4f}
```

```
segundos".format(total_time))
```

Anexo F

Algoritmo de Máquinas de Vectores de Soporte

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split,
cross_val_score
from sklearn.svm import SVC
from sklearn.metrics import classification_report,
confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
import sklearn.metrics as metrics
import time

# Funci n para calcular la eficiencia computacional
def calcular_eficiencia_computacional(training_time,
prediction_time, visualization_time):
    efficiency = training_time + prediction_time +
    visualization_time
    return efficiency

# Carga de datos
df = pd.read_csv("baseDatos1.csv", sep=';')
df["RiegoCod"] = df["Riego"].apply(lambda x: 1 if x == "Yes"
else 0)
df = df.drop(['Unnamed: 0', 'Item'], axis=1)

# Divisi n de datos en caracter sticas (X) y etiquetas (y)
X = df[["HumAire", "Temperatura", "HumSuelo"]]
y = df["RiegoCod"]
```

```
# Divisi n de datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# Entrenamiento del modelo SVM
model = SVC(kernel='linear')
# Puedes probar tambi n con otros kernels como 'rbf' o 'poly'

# Tiempo de entrenamiento
start_training_time = time.time()
model.fit(X_train, y_train)
end_training_time = time.time()
training_time = end_training_time - start_training_time

# Evaluaci n del modelo mediante validaci n cruzada
scores = cross_val_score(model, X, y, cv=5, scoring='accuracy')

# Informe de clasificaci n promedio
print("Informe de clasificaci n promedio:")
print("--------------------------------")
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))

# Precisi n media de la validaci n cruzada
print("Precisi n media de la validaci n cruzada:",
np.mean(scores))

# Tiempo de predicci n
start_prediction_time = time.time()
y_pred = model.predict(X_test)
end_prediction_time = time.time()
prediction_time = end_prediction_time - start_prediction_time

# Generamos la matriz de confusi n
cm = metrics.confusion_matrix(y_test, y_pred)

# Visualizamos la matriz de confusi n
cmDisplay = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=[False, True])
start_visualization_time = time.time()
cmDisplay.plot()
end_visualization_time = time.time()
```

```
visualization_time = end_visualization_time -
start_visualization_time
print("Tiempo de visualizaci n de la matriz de confusi n: {:.4f}
segundos".format(visualization_time))
# Calcular la eficiencia computacional
efficiency = calcular_eficiencia_computacional(training_time,
prediction_time, visualization_time)
print("Eficiencia computacional:", efficiency, "segundos")
```

Anexo G

Algoritmo de Bosques Aleatorios

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split,
GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report,
confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
import seaborn as sns
import matplotlib.pyplot as plt
import time

# Obtener data de origen
df = pd.read_csv(r"baseDatos1.csv", sep=';')
df = df.drop(['Item','TimeStamp', 'Unnamed: 0'], axis=1)
df["RiegoCod"] = df["Riego"].apply(lambda x: 1 if x == "Yes"
else 0)

# Cargar los datos y dividir en caracter sticas (X) y etiquetas
(y)
X = df[['HumSuelo', 'HumAire', 'Temperatura']]
y = df['RiegoCod']

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# Definir la cuadr cula de valores de hiperpar metros
param_grid = {
```

```
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 10, 20, 30]
}

# Inicializar el clasificador de Bosque Aleatorio
rf_classifier = RandomForestClassifier(random_state=42)

# Realizar b squeda en cuadr cula con validaci n cruzada
grid_search = GridSearchCV(estimator=rf_classifier,
param_grid=param_grid, cv=10)
start_time_training = time.time()
grid_search.fit(X_train, y_train)
end_time_training = time.time()

# Obtener los mejores valores de hiperpar metros
best_params = grid_search.best_params_
print("Mejores hiperpar metros:", best_params)

# Entrenar el clasificador de Bosque Aleatorio con los
mejores hiperpar metros
rf_classifier = RandomForestClassifier(**best_params,
random_state=42)
start_time_prediction = time.time()
rf_classifier.fit(X_train, y_train)
end_time_prediction = time.time()

# Hacer predicciones en el conjunto de prueba
y_pred = rf_classifier.predict(X_test)

# Imprimir el informe de clasificaci n
print("Informe de clasificaci n:")
print(classification_report(y_test, y_pred))

# Generar la matriz de confusi n
conf_matrix = confusion_matrix(y_test, y_pred)

# Visualizar la matriz de confusi n
start_time_visualization = time.time()
conf_matrix_display =
ConfusionMatrixDisplay(confusion_matrix=conf_matrix,
display_labels=['No Riego', 'Riego'])
conf_matrix_display.plot(cmap='viridis', ax=plt.gca())
plt.title('Matriz de Confusi n')
```

```
plt.grid(False)  # Remover la cuadr cula
plt.show()
end_time_visualization = time.time()

# Calcular los tiempos de entrenamiento, predicci n y
visualizaci n
training_time = end_time_training - start_time_training
prediction_time = end_time_prediction - start_time_prediction
visualization_time = end_time_visualization -
start_time_visualization

print("Tiempo de entrenamiento:", training_time, "segundos")
print("Tiempo de predicci n:", prediction_time, "segundos")
print("Tiempo de visualizaci n de la matriz de confusi n:",
visualization_time, "segundos")
# Calcular la eficiencia computacional total
training_time = end_time_training - start_time_training
prediction_time = end_time_prediction - start_time_prediction
confusion_matrix_time = end_time_visualization -
start_time_visualization

efficiency = training_time + prediction_time +
confusion_matrix_time
print("Eficiencia computacional:", efficiency, "segundos")
```