UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE CUENCA

CARRERA DE TELECOMUNICACIONES

ANÁLISIS DE MÉTODOS DE ML PARA EL DESARROLLO DE UN SISTEMA RECOMENDADOR EN LA PRODUCCIÓN DE ROSAS

Trabajo de titulación previo a la obtención del título de Ingeniero en Telecomunicaciones

AUTOR: VLADIMIRO RICARDO CORDERO GALARZA

TUTOR: ING. JUAN PAÚL INGA ORTEGA, MgT.

Cuenca – Ecuador

2025

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Yo, Vladimiro Ricardo Cordero Galarza con documento de identificación N° 0107377921; manifiesto que:

Soy el autor y responsable del presente trabajo; y, autorizo a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Cuenca, 10 de febrero de 2025

Atentamente,

Vladimiro Ricardo Cordero Galarza

0107377921

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Yo, Vladimiro Ricardo Cordero Galarza con documento de identificación N° 0107377921, expreso mi voluntad y por medio del presente documento cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del Proyecto Técnico: "Análisis de métodos de ML para el desarrollo de un sistema recomendador en la producción de rosas", el cual ha sido desarrollado para optar por el título de: Ingeniero en Telecomunicaciones, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribo este documento en el momento que hago la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Cuenca, 10 de febrero de 2025

Atentamente,

Vladimiro Ricardo Cordero Galarza
0107377921

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Juan Paúl Inga Ortega con documento de identificación N° 0104166491, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: ANÁLISIS DE MÉTODOS DE ML PARA EL DESARROLLO DE UN SISTEMA RECOMENDADOR EN LA PRODUCCIÓN DE ROSAS, realizado por Vladimiro Ricardo Cordero Galarza con documento de identificación N° 0107377921, obteniendo como resultado final el trabajo de titulación bajo la opción proyecto técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Cuenca, 10 de febrero de 2025

Atentamente,

Firmado electrónicamente por:
JUAN PAUL INGA ORTEGA

Ing. Juan Paúl Inga Ortega, MgT.
0104166491

AGRADECIMIENTOS

Agradezco profundamente al Ing. Juan Inga, tutor del presente trabajo, quien no solo brindó su apoyo en la realización de este proyecto, sino que además sirvió como guía durante mi estadía en la universidad. Su vocación como docente es admirable, y sus enseñanzas las llevaré siempre en el corazón.

DEDICATORIAS

A mis padres, Fernando y Patricia, por su paciencia y apoyo incondicional durante mi vida universitaria; a mi hermano Christopher, por ser una figura irreemplazable en mi vida y un modelo de lo que quiero llegar a ser; a mi abuela Gladys, por depositar su fe en mí y apoyarme cuando más lo necesité; y a mi abuelo Vladimiro, quien, aunque ya no se encuentre con nosotros, estoy seguro de que nunca dejó de creer en mí.

Índice general

Índice de figuras

Índice de tablas

Resumen

En el presente trabajo se muestra la implementación un sistema recomendador para el cultivo de rosas con un modelo de IA Generativa como lo es Gemini. Para ello se analizaron diferentes modelos de ML para clasificar los valores de Nitrógeno, Fósforo y Potasio en base a las variables ambientales de Temperatura, Humedad, pH y precipitación. Para escoger los modelos de ML se hizo un análisis exploratorio del dataset para entrenamiento, se definieron las limitaciones que este podría tener y posteriormente se escogieron los modelos mas adecuados dada la distribución de cada variable. Otro criterio importante a la hora de escoger los modelos fue la naturaleza de estos, pues se escogieron 3 modelos con enfoques diferentes, lo cual sienta una base solida para comparar resultados y comprender que enfoques de ML se comportan de mejor manera a la hora de clasificar variables de nutrientes del suelo en el agro. Una vez que se conocieron los modelos que mejor se comportan para cada variable se procedió a implementar el sistema recomendador, para ello se uso la plataforma de IoT Thingsboard donde el sistema se muestra en un Dashboard, la plataforma IoT se ejecuta en una instancia de AWS. Para conformar el sistema recomendador se uso un script de Python que mediante la API de Gemini crea consultas a partir de la función de NLP del modelo Gemini-1.5 flash usando los datos precedidos por los modelos de ML para generar reportes y recomendaciones como si de un experto de cultivos de rosas se tratase y adicionalmente se mostraron las variables ambientales con las que fueron entrenadas los modelos en un dashboard adicional para monitorear el comportamiento de las mismas, brindando así un sistema mas completo en cuanto al control de cultivos de rosas.

Palabras clave: ML; SVM; Arboles de Decisión; KNN; Dashboards

Abstract

This work shows the implementation of a recommender system for cultivating roses with a Generative IA model such as Gemini. For this purpose, different ML models were analyzed to classify the values of Nitrogen, Phosphorus, and Potassium based on the environmental variables of Temperature, Humidity, pH, and Precipitation. In order to choose the ML models, an exploratory analysis of the training dataset was made, the limitations that it could have were defined. Then the most appropriate models were chosen given the distribution of each variable. Another important criterion when choosing the models was the nature of each one; because of that, 3 models with different approaches were chosen, which provides a solid basis for comparing results and understanding which approaches of ML behave better when classifying soil nutrient variables in agriculture. Once the best-performing models for each variable were known, we proceeded to implement the recommender system, using the IoT Thingsboard platform where the system is displayed on a Dashboard, the IoT platform runs on an AWS instance. In order to create the recommender system, a Python script was used to create queries through the Gemini API, the NLP function of the Gemini-1.5 flash model, and using the data preceded by the ML models to generate reports and recommendations as if it were a rose crop expert. In addition, the environmental variables with which the models were trained were shown on an additional dashboard to monitor their behavior, thus providing a more complete system for the control of rose crops.

Keywords: ML; SVM; Decision Tree; KNN; Dashboards

Antecedentes

Con el creciente aumento de la población en el mundo, los países se ven envueltos en el problema de garantizar una agricultura saludable y sostenible para suplir la demanda de la población. Sin embargo, la Organización de las Naciones Unidas para la Alimentación y Agricultura (FAO, del inglés *Food and Agriculture Organization*) estima que cada año hasta un 40 % de los cultivos se pierden debido a pestes y enfermedades en las plantas [1]. Por este motivo los agricultores a través de todo el mundo optan por sistemas de protección en sus cultivos que les brinden información del estado actual que podrían tener sus granjas para poder prevenir pérdidas al máximo. Es en este contexto, las tecnologías que permiten implementar el concepto de Internet de las Cosas (IoT, del inglés *Internet of Things*) representan una parte fundamental en la agricultura ya que facilita el monitoreo remoto. Además, los sensores de la red proveen información valiosa a los agricultores como lo pueden ser la humedad, calidad del suelo y plagas, lo que reduce en gran medida el trabajo a los agricultores. No obstante, estas tecnologías no pueden predecir enfermedades o niveles de producción solamente con los datos, es ahí donde radica la importancia de inteligencia artificial (IA) ya que gracias a esta, es posible trabajar con técnicas de modelos predictivos para facilitar el trabajo a los agricultores al predecir que es lo que podría ocurrir con los cultivos a corto y mediano plazo con alto porcentaje de efectividad.

También, las bases de datos han probado ser una pieza fundamental en la agricultura en los últimos años debido a su capacidad y flexibilidad para almacenar los diferentes tipos de información que se pueden recolectar de los sistemas de IoT. Es por ello que se han llevado a cabo diversas implementaciones de bases de datos relacionales que son accesibles a nivel global mediante una interfaz web utilizando

tecnologías como las de Oracle, MySQL y Google Cloud Platform. El uso de estas bases de datos permiten destacar la importancia de los cultivos en la promoción de sistemas alimentarios sostenibles y la agrobiodiversidad [2].

La implementación de dashboards (paneles) y su integración con las bases de datos han probado ser una manera efectiva de transmitir la información en los sistemas de IoT, se ha logrado implementar diversos conjuntos de datos, incluyendo humedad del suelo, producción de cultivos, indicadores económicos y métricas de salud. A través de opciones de visualización como mapas coropléticos, gráficos de líneas y matrices de correlación, los usuarios pueden explorar las relaciones entre diferentes variables a lo largo del tiempo. Un caso de estudio sobre la producción de maní y mijo en los distritos de Kaolack y Kaffrine de Senegal demuestra cómo esta herramienta puede revelar variaciones en la producción de alimentos y su relación con factores socio económicos y de mortalidad infantil [3].

Con la llegada de IA y el desarrollo de sus conceptos derivados como aprendizaje automático (ML, del inglés *Machine Learning*) y aprendizaje profundo (DP, del inglés *Deep Learning*) varios campos de la industria se han visto beneficiados y la agricultura no es ninguna excepción, además cada día los métodos evolucionan y se optimizan lo que supone un campo en pleno crecimiento. Así, la aplicación del DP en la agricultura ha ganado una atención significativa debido a su potencial para mejorar la gestión de cultivos y el rendimiento. Diversas arquitecturas de DP, como Faster R-CNN, SSD e Inception v3, se han utilizado para diagnosticar enfermedades en plantas con diferentes niveles de éxito.[4]

Se ha visto también que las CNN (Redes Neuronales Convolucionales) pueden ser altamente efectivas para la clasificación de imágenes, lo que las hace adecuadas para la identificación de enfermedades en las plantas, alcanzando precisiones de 99.2 % y 98.42 % para los modelos de DenseNet201 y EfficientNet respectivamente.[5]

Los trabajos realizados hasta la fecha revelan que una buena implementación de un sistema de análisis con IoT en la agricultura facilitan en gran medida a la toma de decisiones de las empresas del sector, sin embargo, al existir una gran cantidad de herramientas se deben tomar en cuenta los requerimientos del proyecto en cuestión para lograr una selección adecuada.

En el país se han realizado estudios donde se analiza la factibilidad de la implementación de estas tecnologías para el sector agrícola nacional, donde se proponen soluciones en el óptimo consumo de agua y el monitoreo de algunas variables importantes en el sector, así como también se han usado algoritmos de ML en otros campos como para la identificación de temas en tweets de la red social **X** en el país [6].

Justificación

Actualmente con los avances en los sectores de IoT, bases de datos y visualización de datos existen demasiadas herramientas que pueden resultar confusas a la hora de escoger la adecuada para el proyecto en el que se pretenda incursionar con estas tecnologías. Además, escoger un modelo adecuado de Aprendizaje de Máquina es crucial para garantizar la precisión y veracidad de las predicciones realizadas en base a los datos. En este contexto el presente trabajo de titulación tiene como objetivo implementar un sistema de predicción para las diferentes variables de nutrientes en el suelo (Nitrógeno, Fósforo y Potasio) para luego elaborar una recomendación mediante IA para la toma de decisiones y aportar así un sistema mas completo donde, al poder predecir los nutrientes en el suelo, el usuario final reduciría los costos de producción ya que estos pueden resultar elevados en la mayoría de casos, todo esto mediante el uso de las tecnologías actuales de IA y ML.

Objetivos

Objetivo General

- Analizar Métodos de ML para el desarrollo de un sistema recomendador en la producción de rosas.

Objetivos específicos:

- Analizar la revisión bibliográfica necesaria para la identificación de las variables de monitoreo más importantes en el agro y posibles métodos de ML para la producción de rosas.
- Comparar los modelos de ML identificados en el estado del arte para establecer el mas adecuado en relación a la producción de rosas.
- Implementar el modelo ML mas adecuado en un sistema recomendador a ser presentado en un dashboard de los datos de nutrientes de suelo.

Introducción

En el país existe un retraso en cuanto a sistemas IoT si lo comparamos con el resto del mundo, por lo que es necesario incursionar en nuevas tecnologías y sistemas de control sobretodo en la agricultura teniendo en cuenta que esta representa un 7,57 % del producto interno bruto del Ecuador en 2024 [7]. Además, las pérdidas por mal manejo de cultivos pueden crecer exponencialmente si no se tiene un control adecuado del mismo. En este contexto, el presente trabajo busca aportar con un sistema basado en Internet de las Cosas en donde no solo se apliquen las tecnologías mas populares y estándar ya que ademas se pretende incursionar en otras tecnologías mas modernas y acordes a los tiempos actuales como lo pueden ser los métodos de ML, IA y NLP para brindar un sistema mas completo en el campo del IoT.

Este trabajo tiene un beneficiario principal que es la florícola "Florlago.ª través del proyecto de Investigación SISMO-ROSAS a cargo del grupo de investigación en Telecomunicaciones y Telemática (GITEL) de la Universidad Politécnica Salesiana donde se busca brindar un sistema de vanguardia en el país con varias etapas, donde una de ellas involucra un sistema recomendador para la producción de rosas. Este sistema funciona recibiendo información de métodos de ML previamente entrenados para predecir el estado de los nutrientes de Nitrógeno, Fósforo y Potasio del suelo. Luego se envían estos datos a un modelo de IA generativa mediante una API que permita usar las capacidades de NLP y obtener recomendaciones a través de lenguaje natural al usuario final por medio de la plataforma de IoT de Thingsboard.

Entonces, el presente trabajo analiza diversos métodos de ML para identificar los más adecuados para cada variable implementada. También, funciona como una guía de cómo implementar modelos de ML en el contexto de variables de cultivos en general, además de mostrar los procesos de análisis exploratorios que se pueden

seguir para identificar la distribución de nuestras variables y así tener un mejor criterio a la hora de escoger los modelos adecuados para la clasificación de variables en el agro.

Capítulo 1

Fundamento Teórico

El presente capítulo pretende brindar al lector los conceptos relevantes que se relacionan el desarrollo de este trabajo y así pueda familiarizarse con aspectos relacionados con la agricultura digital y la ciencia de datos. Para ello, se presenta una breve descripción de lo que busca la agricultura digital y sus conceptos inherentes, se presenta una perspectiva de cómo aprovechar el uso de modelos generativos de IA; también, se describen los aspectos más relevantes relacionados con la ciencia de datos y el aprendizaje de máquina para finalizar con una breve exploración bibliográfica de las prácticas actuales en el uso de clasificadores en la agricultura.

1.1. Agricultura Digital

Cuando hablamos de agricultura digital, por lo general se refiere a todos aquellos procesos en los que se implementan las tecnologías digitales en el contexto de los sistemas de agricultura. Esto incluye el paradigma de IoT aunque no está limitado a su uso, incluye también las bases de datos tanto relacionales como no relacionales (pues dependerá de la arquitectura), las comunicaciones inalámbricas, robótica y electrónica en general [8].

Varios estudios señalan que la agricultura digital se encuentra en una fase de gran crecimiento, esto no es sorpresa pues viene de la mano de la evolución constante de las tecnologías mencionadas, por lo que el agro digital se ha convertido en un campo bastante cotizado al que cada vez más agricultores se suman para mejorar,

optimizar y automatizar sus cultivos [9].

1.1.1. Variables del agro Relevantes

En el agro existe una amplia cantidad de variables que pueden ser monitoreadas por las tecnologías actuales y luego de revisar los artículos pertinentes se identificaron las siguientes como las más importantes a destacar:

- **Condiciones climáticas:** Incluyen varios factores del clima sin embargo destacan la temperatura, humedad y precipitación [10].
- **Suelo:** Se tiene un interés especial en la calidad del suelo, midiendo humedad, ph y nutrientes los cuales influyen en los cultivos [9].
- **Recursos Hídricos:** La calidad del agua, su disponibilidad y gestión eficiente son de gran interés en la agricultura digital [10].

1.1.2. Tecnologías en el agro

Un tema de gran importancia a la hora de implementar un sistema de agricultura digital son las tecnologías a emplear en dicho sistema, cada día estas tecnologías avanzan, mejoran y surgen nuevas también, a continuación se muestran las más utilizadas en la industria en la actualidad.

- **IoT:** Tal vez la principal tecnología del campo es el internet de las cosas ya que permite monitorear las variables de elección en tiempo real [11].
- **Robótica:** Los robots cada vez están más presentes en la agricultura digital y se pueden observar por ejemplo en el ordeñamiento de las vacas [12].
- **Nanotecnología:** Algunas soluciones proponen nanotecnología para el monitoreo de algunas variables del suelo [12].
- **IA:** Hoy en día con el creciente aumento de las capacidades de la IA se implementan modelos de aprendizaje de máquina y aprendizaje profundo [12].

- **Bases de datos:** Con la gran cantidad de datos que se recolectan en el sector es imprescindible tener una base de datos en cualquier sistema de agricultura digital [13].

1.2. Inteligencia Artificial

Según Google la definición de inteligencia artificial no es tan simple pues se refiere a varias tecnologías que permiten a las computadoras razonar de una forma más compleja. En concreto, la IA es una parte de la ciencia que está relacionada con la creación de algoritmos y máquinas que sean capaces de aprender y razonar de una manera que los algoritmos comunes no pueden y que por lo general requeriría de un criterio humano para resolver los problemas que se le presentan [14].

Existen varios tipos de inteligencia artificial los cuales se muestran a continuación:

- **Máquinas Reactivas**: Es la forma más básica de IA, no tiene memoria, solo reacciona a reglas programadas con antelación así que no puede aprender en base a datos nuevos [14].
- **Memoria limitada**: Hoy en día se puede decir que toda inteligencia artificial es de memoria limitada ya que si tienen memoria y pueden usarla para mejorar a través del tiempo conforme entrenan con datos nuevos [14].
- **Teoría de la mente**: Hasta ahora no existe ninguna IA que se encuentre en esta categoría, sin embargo diversos estudios evalúan las posibilidades de implementarla, se trata de emular la mente humana para que la máquina pueda tomar decisiones basadas en emociones como lo haría un ser humano y desenvolverse sin problemas en situaciones sociales [14].
- **Auto conocimiento**: Un poco más avanzado que la teoría de la mente, en este punto se piensa en una máquina que sea consciente de su propia existencia y replique las características de un ser humano [14].

La IA tiene aplicaciones en todos los campos lo que la convierten en una herramienta universal hoy en día para la mayoría de proyectos y se estima mediante

un reporte del 2024 que un 40 % de las empresas a nivel mundial están usando algún tipo de IA en sus procesos y el 82 % de empresas a nivel mundial están usando o considerando usar IA en los procesos empresariales [15].

1.2.1. IA Generativa y NLP

Se entiende por Inteligencia Artificial Generativa a el campo de la IA que se enfoca en la generación de nuevo contenido, ya sea texto, imágenes, video, audio, etc. A partir del entrenamiento con grandes conjuntos de datos[16]. Para generar este nuevo contenido se ocupa los llamados **Modelos Generativos** que son algoritmos matemáticos y arquitecturas de ML que están diseñados para generar datos nuevos a partir de lo aprendido [16]. Las principales arquitecturas para la IA Generativa son:

- Redes Generativas Antagónicas
- Variational Encoders
- Modelos Difusos

La elección de cada arquitectura dependerá de la aplicación [16]. En cuanto a la generación de nuevo contenido por parte de la IA es indispensable tratar el tema de que los modelos generen lenguaje natural, es decir, lenguaje que sea similar o igual al lenguaje humano esto se conoce como procesamiento de lenguaje natural (NLP, del inglés *Natural Language Processing*). Podemos definir NLP como una rama de la inteligencia artificial que se centra en la interacción entre las máquinas y el lenguaje humano. Su objetivo es permitir que las computadoras comprendan y generen contenido en lenguaje humano. Esto incluye una variedad de aplicaciones, como la búsqueda semántica, herramientas conversacionales, traducción de documentos y generación de texto [17].

En la actualidad los modelos más usados en el campo de la IA Generativa se denominan Transformadores. Un modelo transformador es una red neuronal que se usa principalmente para NLP, fue propuesta en 2017 por Vaswani y se diferencia de los modelos tradicionales de redes neuronales en que permite modelar de forma directa las relaciones entre todas las posiciones de entrada, en lugar de procesarlas de

forma secuencial [18]. Esto facilita el manejo de dependencias temporales y mejora la eficiencia en el entrenamiento de modelos a gran escala [18] lo cual es fundamental debido a que las herramientas que usan estos modelos como lo pueden ser chatGPT y Gemini se entrenan en grandes volúmenes de datos y luego se ajustan para tareas más especificas, logrando así una interacción con el usuario de una forma natural como si se tratase de otro ser humano respondiendo a las preguntas que se le asignen.

Es importante destacar que los modelos no son perfectos, un estudio que evaluó el rendimiento de los modelos de lenguaje ChatGPT 4.0 y Gemini Ultra 1.0 en la generación de código NONMEM para tareas farmacométricas mostró que aunque ambos modelos proporcionaron marcos útiles y generaron código, las salidas contenían errores, lo que requería revisiones antes de su ejecución [19].

1.3. Ciencia de datos y ML

La Ciencia de Datos hace referencia a un conjunto de destrezas y habilidades multidisciplinarias que cada día cobran más importancia en las aplicaciones cotidianas, entre estas disciplinas se encuentran la estadística, las ciencias computacionales y también la habilidad para interpretar resultados basándose en la experiencia en la industria [20]. En la figura 1.1 se observa un diagrama de venn realizado por Drew Conway's en 2010.

1.3.1. Python para la Ciencia de Datos

Python suele ser el principal lenguaje de programación a la hora de manejar datos por lo que resulta esencial para la Ciencia de Datos, esto se debe a que, además de ser un poderoso lenguaje de alto nivel, tiene una gran variedad de librerías dedicadas a la estadística y manejo de conjuntos de datos como lo son las librerías Pandas y NumPy, además tenemos librerías para la visualización de los datos como pueden ser MatplotLib o Seaborn y otras para Machine Learning y Deep Learning como lo son ScikitLearn o Tensorflow [20].

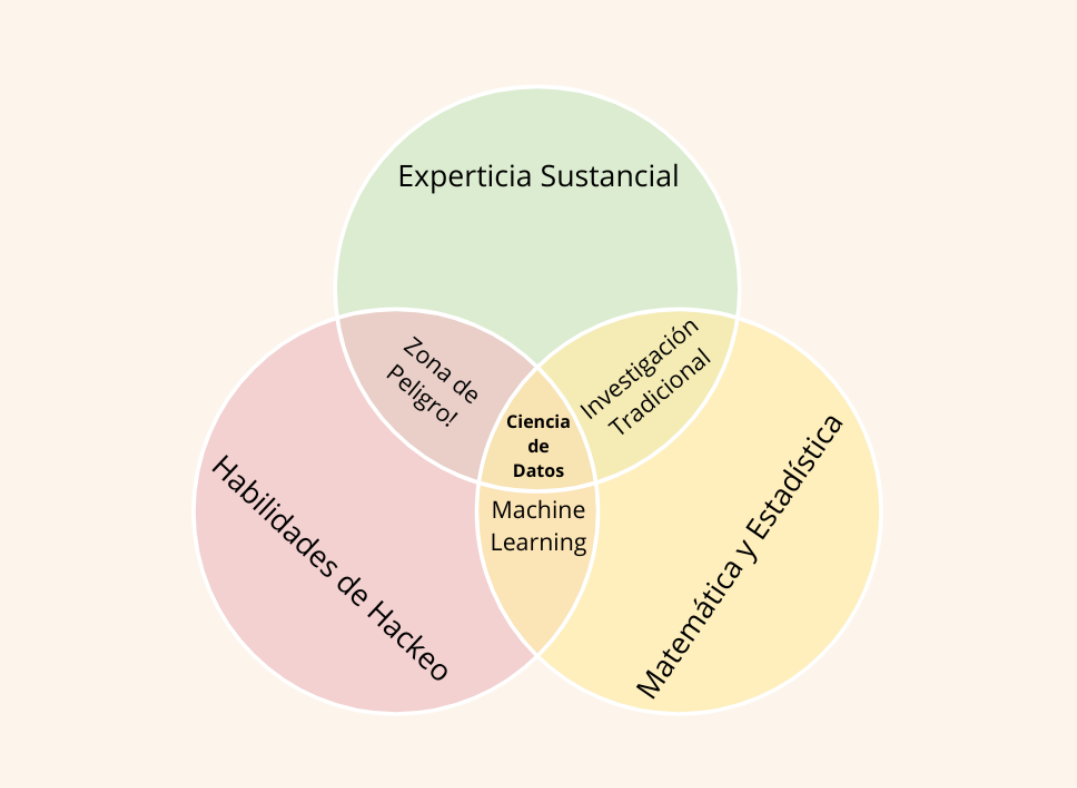


Figura 1.1: Diagrama de Venn de la Ciencia de Datos.
Fuente [20].

1.3.2. Métodos de ML

El aprendizaje de máquina (Machine learning) se refiere a un sub campo de la inteligencia artificial que se centra en crear modelos a partir de los datos, involucra en gran parte construir modelos matemáticos para ayudar a entender los datos, el término de Aprendizaje viene del hecho de que a estos modelos se los alimenta con parámetros ajustables que pueden ser adaptados a los datos observables, el primer modelo de ML fue propuesto en 1949 por Donald Hebb el cual fue llamado la "Regla de aprendizaje de Hebb" que está basada en la neuropsicología y se trata de una regla de aprendizaje no supervisado que extrae estadísticas de un set de entrenamiento para clasificar datos que tengan similitud con los del entrenamiento [21]. Una vez que los modelos se han ajustado a los datos de entrenamiento estos se pueden usar para predecir y entender aspectos de los nuevos conjuntos de datos que ingresemos [20].

Existen 2 tipos de Aprendizaje de Máquina [falta referencia]:

- **Aprendizaje Supervisado**: El modelo se entrena con datos etiquetados, es decir,

con ejemplos que incluyen tanto las entradas como las salidas correctas. El objetivo es que el modelo aprenda a predecir la salida a partir de nuevas entradas [20]

- **Aprendizaje no Supervisado**: El modelo se entrena con datos no etiquetados, es decir, solo tiene las entradas y debe identificar patrones o estructuras subyacentes en los datos, esto implica tareas como el clustering y la reducción dimensional [20].

1.3.3. Árboles de Decisión

Este algoritmo que puede usarse para problemas de clasificación y regresión, sin embargo para los objetivos de este trabajo se profundizara en los árboles de decisión y clasificación. Cada árbol tiene 3 componentes clave: un único nodo raíz, nodos de decisión y nodos terminales o también llamados hojas, en la figura 1.2 se observan estos elementos.

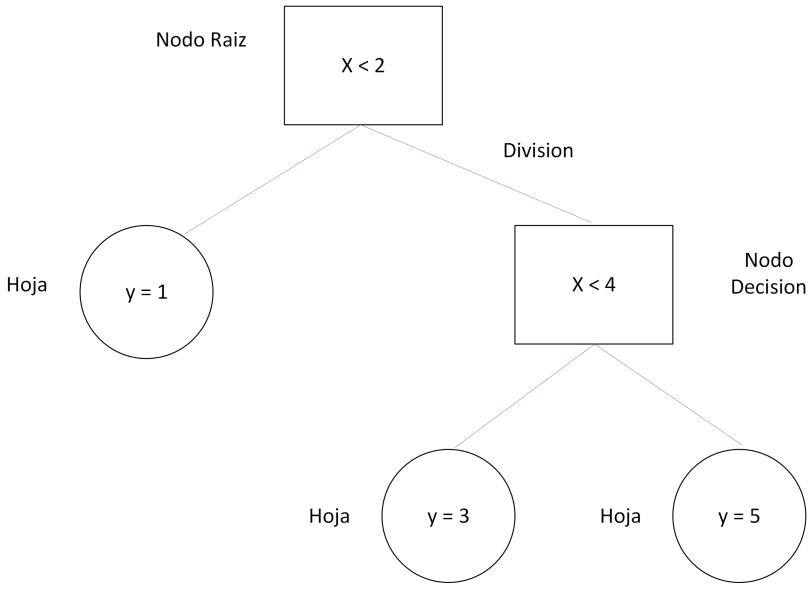


Figura 1.2: Estructura básica de un árbol de decisión.
Fuente: [22]

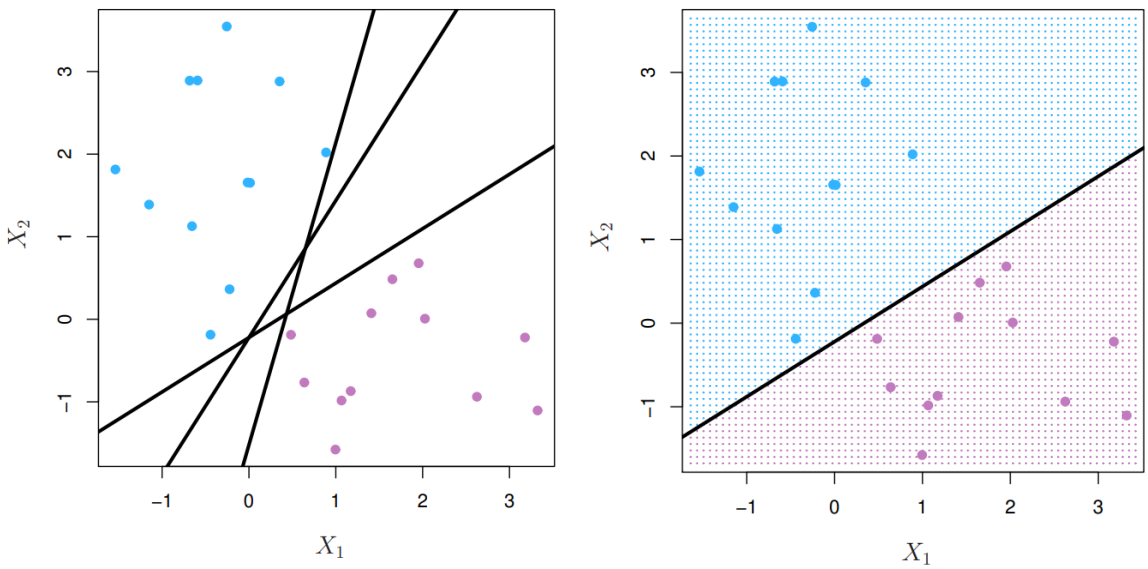
.

En un árbol de clasificación (como en cualquier clasificador) el objetivo es asignar una observación a una clase específica, esto se consigue a través de la clase

de mayor ocurrencia en la región de las observaciones del entrenamiento, es decir, en cada nodo hoja se asignara la clase que tenga mayor ocurrencia en dicho nodo. [23]

1.3.4. Máquinas de Soporte Vectorial

Las máquinas de soporte vectorial (SVM, del inglés *Support Vector Machine*) fueron desarrolladas en los años 90 y hoy en día representan uno de los clasificadores más usados en la comunidad de Ciencia de Datos [23]. La idea de un SVM es segmentar la data, para ello ocupamos hiperplanos donde se maximice el margen el cual es la distancia entre el hiperplano y la entrada más cercana a dicho plano [24]. En la figura 1.3 se muestra la forma más básica de segmentar las clases a través de un plano, donde se observan solo 2 clases separadas por varias líneas posibles para dividirlas en la figura de la izquierda, mientras que en la de la derecha se observa un hiperplano escogido para separar las clases.



(a) Diferentes hiperplanos para separar el mismo conjunto de clases.

(b) Un hiperplano en concreto separando las clases.

Figura 1.3: Formas de separar un mismo conjunto de clases [23].

En las máquinas de soporte vectorial podemos escoger varios kernels para dividir los datos, podemos ver al kernel como la forma en la que vamos a separar las clases, es decir, qué tipo de hiperplano vamos a usar, existen 4 tipos:

- Lineal: Usamos planos rectos para dividir las clases [25].
- Polinómico: Usamos planos de grado polinómico por lo que pueden ser curvos [25].
- RBF: Es un kernel de base radial por lo que podemos encerrar a las clases en círculos, además es el más usado [25].
- Sigmoid: Se basa en la tangente hiperbólica para lograr la separación entre clases [25].

Kernel RBF

El kernel RBF se define por la ecuación 1.1 para el caso de 2 clases:

$$K(x_1, x_2) = exp(-\gamma \cdot || x_1 - x_2 ||^2) \tag{1.1}$$

Así podemos ver que γ es quien controla la influencia de cada entrenamiento sobre las decisiones de frontera.[25]

1.3.5. K vecinos más próximos

El algoritmo de K Vecinos más Próximos (KNN, del inglés *K Nearest Neighbor*) es un algoritmo de aprendizaje de máquina bastante popular para desarrollar clasificadores, aunque también puede ser usado para problemas de regresión.
Para poder etiquetar una clase se toma la distancia entre ese punto y todas las k clases cercanas a este punto, luego la etiqueta se asignara a la clase que más se repite entre los vecinos, esto se conoce en la literatura como "votación por mayoría"[26]. En la figura 1.4 se observa como se asignan las etiquetas de clase en el algoritmo de KNN para el caso más básico donde solo tenemos 2 clases.

Figura 1.4: Algoritmo de clasificación de KNN cuando K=1 [27].

Para calcular la distancia existen 2 formas tradicionales, que son la distancia Manhattan y la distancia euclidiana, sin embargo, para generalizar estas 2 podemos usar la distancia de Minkowski que se muestra en la ecuación 1.2.

$$d\left(\mathrm{x}^{[a]}, \mathbf{x}^{[b]}\right) = \left[\sum_{j=1}^{m} \left(\left|x^{[a]} - x^{[b]}\right|\right)^{p}\right]^{\frac{1}{p}} \tag{1.2}$$

La ecuación 1.2 se vuelve distancia Manhattan cuando $p = 1$ y euclidiana cuando $p = 2$, esto es importante pues esta notación es la que sigue la librería sckit-learn para aprendizaje de máquina en Python.

1.4. Fundamentos de ML

1.4.1. Balanceo de clases

El balanceo de clases consiste en varias técnicas destinadas a reducir el sesgo que puede ocurrir cuando existe una desproporción grande entre las muestras de las diferentes clases en un conjunto de datos, esto resulta de suma importancia en los clasificadores estándar ya que estos tienden a ser abrumados por las clases mayoritarias y pueden ignorar las clases minoritarias, lo que resulta en un rendimiento deficiente. Existen varias estrategias para el balanceo, como aumentar la penalización por clasificar incorrectamente la clase positiva, sobre-muestrear la clase minoritaria o sub-muestrear la clase mayoritaria [28].

Sobre muestreo

Una técnica bastante popular es aumentar con muestras sintéticas a las clases minoritarias para que lleguen a tener la misma cantidad que la clase mayoritaria. Para generar una muestra sintética podemos usar el método SMOTE de la librería de imbalanced-learn en python, este método trabaja de la siguiente manera:

1. **Definición de las muestras a agregar**: Primero se establece cuantas muestras sintéticas se deberán agregar en base a la clase mayoritaria.

2. **Selección del punto**: Se elige de forma aleatoria una instancia (o punto) de la clase minoritaria.

3. **Búsqueda de vecinos**: Se identifican los 5 vecinos más cercanos a este punto.

4. **Interpolación**: Se seleccionan de forma aleatoria algunos de esos vecinos cercanos para generar nuevas instancias. Para ello primero se calcula la diferencia entre la instancia original y cada vecino seleccionado. Luego, esa diferencia se multiplica por un valor aleatorio entre 0 y 1, y se suma al valor original. Y así se crea una nueva instancia sintética entre la instancia original y su vecino [29].

De aquí se repiten los pasos 2-4 hasta lograr balancear las clases, en la figura 1.5 se ilustra este proceso.

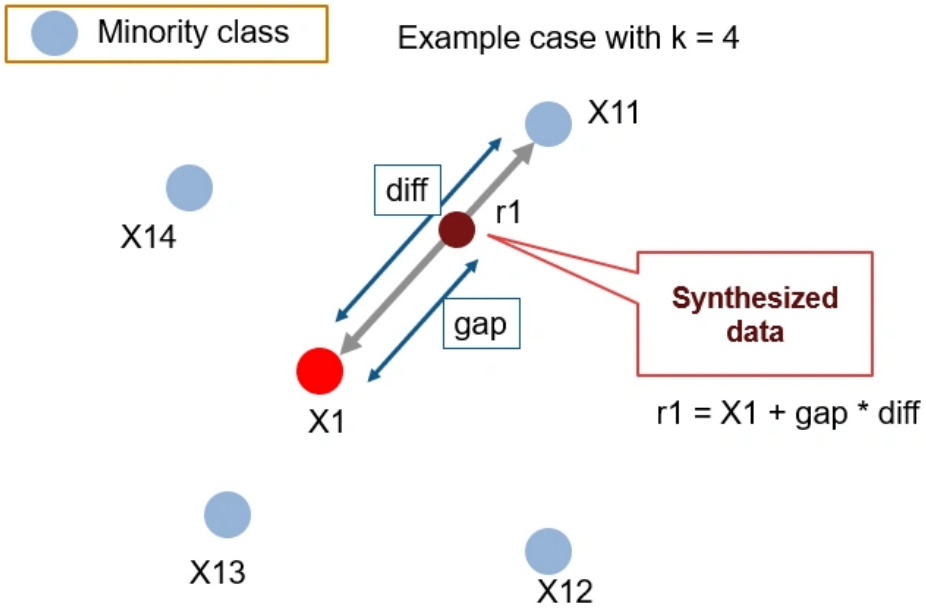


Figura 1.5: Balanceo por agregación de muestras sintéticas [29].

Métodos Híbridos

Las técnicas híbridas involucran aplicar tanto sobre muestreo como sub muestreo, existen 2 bastante importantes en la librería de inbalanced-learn:

- **SMOTETOMEK**: Este método inicia aplicando SMOTE para agregar las muestras sintéticas pero después se identifican y eliminan los pares de muestras de clases opuestas que son mutuamente sus vecinos más cercanos. Estas muestras suelen encontrarse cerca de los límites de decisión y representan casos donde las clases están solapadas [29].
- **SMOTEEN**: De igual manera aplica SMOTE para agregar las muestras pero luego utiliza una técnica de sub muestreo en la que se evalúan los 3 vecinos más cercanos de cada instancia y si una de ellas es mal clasificada por sus vecinos se considera ruidosa y por lo tanto se elimina [29].

1.4.2. Validación Cruzada

La validación cruzada es un procedimiento utilizado en estadística, minería de datos y ciencia de datos para estimar el rendimiento de generalización de un

modelo. Consiste en dividir los datos en dos partes: una parte se utiliza para entrenar el modelo y la otra para pruebas. Este procedimiento es ampliamente aceptado en la comunidad de ML como un estándar para evaluar el rendimiento de los modelos [30]. El procedimiento que se sigue es:

1. Dividir el conjunto de datos en k partes iguales.
2. Entrenar el modelo usando $k-1$ partes y evaluarlo con la parte restante, esto se repite k veces donde en cada iteración se deja un diferente conjunto de prueba.
3. Se promedian los resultados obtenidos en el paso anterior.

Aplicando estos pasos se obtiene una estimación más robusta de como se comportara el modelo con datos no vistos.

1.4.3. Métricas de Evaluación

Tal vez la métrica más intuitiva y general para evaluar la calidad de un clasificador es la exactitud la cual se refiere a la proporción de casos que fueron identificados o clasificados de forma correcta, esto se calcula mediante la ecuación 1.3, donde TP son los verdaderos positivos, TN son los verdaderos negativos, FP los falsos positivos, FN los falsos negativos y N hace referencia al total de casos [31].

$$exactitud = \frac{TP + TN}{TP + TN + FN + FP} = \frac{TP + TN}{N} \tag{1.3}$$

Una métrica bastante importante a la hora de evaluar un clasificador es el recall, este valor cuantifica que tan bien clasifica nuestro modelo como positivo a un caso que en realidad si es positivo (identificar correctamente una clase) [31], esto se puede calcular mediante la ecuación 1.4.

$$recall = \frac{TP}{TP + FN} \tag{1.4}$$

Aun así, el recall no captura en su totalidad la información sobre la exactitud total del modelo. Otra métrica importante es la precisión, esta nos ofrece una perspectiva diferente ya que evalúa la probabilidad de que una predicción positiva sea correcta

[31], esta se calcula mediante la ecuación 1.5.

$$precision = \frac{TP}{TP + FP} \tag{1.5}$$

El puntaje F1 (en inglés F1-score) es la media armónica de la precisión y el recall, por lo que se calcula como en la ecuación 1.6.

$$F1 - score = 2 \cdot \frac{recall \cdot precision}{recall + precision} = \frac{2TP}{2TP + FN + FP} \tag{1.6}$$

1.5. Métodos más usados en el agro

Dentro del campo de la agricultura digital y la Ciencia de Datos aplicada a la agricultura se pueden destacar los métodos de ML de Máquinas de soporte Vectorial, Random Forest y Stochastic Gradient Descent ya que han presentado resultados bastante prometedores para la clasificación de variables en el agro de acuerdo con [5]. Las redes neuronales convolucionales se consideran los más avanzados, pues ofrecen precisiones más altas en la detección de plantas, varios modelos como AlexNet, GoogleNet y LeNet han sido utilizados de forma amplia y han mostrado resultados superiores en comparación con los métodos de ML tradicionales a la hora de detectar enfermedades en plantas[5]. A pesar de mostrar buenos resultados, los modelos de DP se asocian por lo general a un costo computacional mucho más elevado, además de resultar más complejos de implementar que un modelo tradicional de ML, esto resulta en un problema crucial pues es necesario tener acceso a una gran capacidad de cómputo para avanzar en el estudio de detección de enfermedades en plantas y predicción de las variables relacionadas con la agricultura [4]. Otra limitación importante resulta ser que muchos conjuntos de datos utilizados en investigaciones se crean por los propios autores y no están disponibles para el público en general, esto afecta de forma directa a la validación de los modelos pues al no contar con el conjunto de datos también conocido en inglés como *dataset* que fue usado durante una investigación se afecta la replicación de la técnica usada y su evaluación [4].

Trabajos como "A Review on Machine Learning Classification Techniques for Plant Disease Detection" presentan una revisión de las técnicas de clasificación en

aprendizaje automático para la detección de enfermedades en plantas, destacando el impacto de estas enfermedades en la producción agrícola. Se comparan distintos enfoques, incluyendo métodos de ML como SVM y KNN, pero además se exploran métodos de DP como lo son las redes neuronales y CNN. En el estudio se concluye que las CNN logran la mayor precisión en la detección de múltiples enfermedades en diferentes cultivos. En el estudio se enfatiza la necesidad de explorar métodos automáticos y precisos para apoyar a los agricultores en la identificación temprana de enfermedades y mejorar la producción agrícola [32]. Además, para entrenar los modelos se requieren parámetros, los cuales pueden ser obtenidos de las variables de interés en el agro, diversos estudios señalan que las variables meteorológicas o del ambiente resultan ser las de más interés dentro de la agricultura digital [33].

La tabla 1.1 presenta un resumen de los trabajos analizados para el desarrollo del trabajo donde se detallan las áreas que cada artículo profundiza, incluyendo además aquellas áreas en las que el presente trabajo pretende incursionar.

Tabla 1.1: Resumen de los trabajos relacionados el tema de titulación.

Artículo	**Problema de Telecomunicaciones**					**Restricciones**					**Propósito**			
Autor	Machine Learning	Redes Neuronales	Dashboards	Deep Learning	Internet de las Cosas	Tipo de plantas	ML supervisado	Modelos de base de datos	ML no supervisado	Evaluación de modelos	Fiabilidad	Apariencia	Optimización	Integración de datos
Nizar, 2021 [3]			✠		✠			✠		✠	✠		✠	✠
Shruthi , 2019 [32]	✠	✠		✠		✠	✠				✠		✠	
Adedoja, 2020 [4]	✠	✠		✠		✠	✠		✠	✠	✠		✠	
Biswas, 2023[5]		✠		✠	✠	✠	✠			✠	✠		✠	
Santiago, 2023 [2]			✠		✠			✠			✠	✠		✠
kSilawarawet, 2021 [34]			✠		✠			✠			✠	✠		✠
Chergui, 2022 [35]	✠	✠		✠			✠		✠	✠	✠		✠	
Este trabajo	✠		✠		✠	✠	✠			✠	✠	✠		✠

Capítulo 2

Diseño e Implementación

Lo más importante para lograr cumplir los objetivos propuestos en el presente trabajo de titulación es definir las herramientas necesarias para llegar a la solución, sin embargo, para saber qué herramientas utilizar se debe conocer la naturaleza del problema, en este caso se inicia planteando un dataset a usar para entrenar los modelos y luego se hace un análisis exploratorio del mismo para escoger aquellos modelos que resultaran más adecuados así como el ajuste de dichos modelos. Con todo esto se puede implementar un sistema recomendador mediante la API de Gemini para mandar los datos recopilados junto con la recomendación de la IA a la plataforma en la nube Thingsboard.

2.1. Enfoque Metodológico

En la presente sección se detallan los procesos que fueron seguidos para sentar las bases para el entrenamiento y evaluación de los modelos ML realizado posterior a este análisis.

Para implementar las predicciones de NPK de los clasificadores con el sistema recomendador, se usa la API de Gemini, ya que este modelo de IA generativa ofrece una capa gratuita para explorar y es suficiente para cumplir los objetivos del presente trabajo. En cuanto a los dashboards estos fueron trabajados sobre la plataforma de Thingsboard que está alojada en una instancia de AWS. De esta manera los datos que se reciben de los módulos de medición a través de alguna tecnología IoT, sean

presentados en el dashboard. Según los datos recibidos, otra instancia alojada en AWS también, se encarga de procesar y clasificar los datos para que, a través de Gemini, se de la respuesta del recomendador. Para el entrenamiento, se estableció que el usuario desacargue el historial en un archivo con extensión CSV para que el encargado del mantenimiento del sistema pueda actualizar el modelo.

Cabe mencionar que previo al proceso de entrenamiento, es necesario curar los datos. Esta gestión de datos se describen en las secciones siguientes.

La figura 2.1 muestra la estructura de todo el sistema de acuerdo a lo antes comentado, donde se incluyen las etapas de análisis y entrenamiento para luego pasar a la instancia en la nube y posterior a ello enviar los datos procesados a la plataforma de IoT.

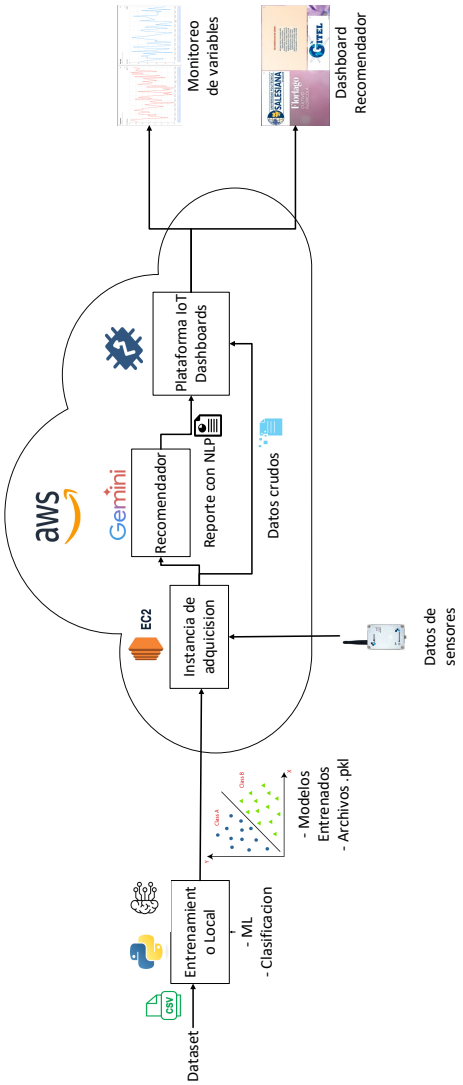


Figura 2.1: Diagrama de bloques del sistema completo.
Fuente: El Autor.

2.1.1. Aproximación al problema

En la florícola se necesita saber el estado de los valores de los nutrientes NPK en el suelo, es decir, se clasificarán en bajo, medio y alto, para esto es necesario usar clasificadores de ML. Sin embargo, en el estado actual del proyecto no se cuentan con los nodos de recolección de datos por lo que no se dispone de un dataset específico de la florícola, aun así es posible entrenar modelos con datasets parecidos para luego migrar las técnicas a la data específica de la florícola.

Para entrenar los modelos se usó un dataset descargado de la página de la comunidad más grande de científicos de datos llamada Kaggle, este dataset contiene lo necesario para trabajar con los modelos y clasificar los valores de los nutrientes del suelo, el aporte fue realizado por Aryan Sinha [36].

Antes de empezar a explorar se debe tener claro las librerías que son necesarias para poder tener un mejor entendimiento de los datos que se presentan en el dataset, para ello se manejaron las librerías de python más comunes para manipular y visualizar conjuntos de datos las cuales se listan a continuación:

- Pandas
- Numpy
- Matplotlib
- Seaborn

Se inicia el trabajo declarando las librerías que son necesarias para realizar el análisis exploratorio de un dataframe.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Se empieza dando un vistazo general a los datos, para ello vamos a cargar el dataset y usar el método head de pandas para poder observar las primeras 5 filas del dataset.

Es necesario comprender como están estructurados los datos, para ello se debe crear un dataframe a partir del dataset y mediante el método *"head"* de pandas visualizar las 5 primeras filas, el código se muestra a continuación:

```
datos_df = pd.read_csv("Crop_recommendation.csv")
datos_df.head()
```

La salida de esta celda se muestra a continuación:

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice

Esto brinda una idea general de como se ve el dataset, sin embargo, es necesario obtener más información de los datos y esto se logra mediante los metadatos, los cuales se pueden obtener mediante el método describe, por lo que se ejecuta el siguiente comando:

```
datos_df.describe()
```

El cual a su salida nos muestra:

	N	P	K	temperature	humidity	ph	rainfall
count	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000
mean	50.551818	53.362727	48.149091	25.616244	71.481779	6.469480	103.463655
std	36.917334	32.985883	50.647931	5.063749	22.263812	0.773938	54.958389
min	0.000000	5.000000	5.000000	8.825675	14.258040	3.504752	20.211267
25 %	21.000000	28.000000	20.000000	22.769375	60.261953	5.971693	64.551686
50 %	37.000000	51.000000	32.000000	25.598693	80.473146	6.425045	94.867624
75 %	84.250000	68.000000	49.000000	28.561654	89.948771	6.923643	124.267508
max	140.000000	145.000000	205.000000	43.675493	99.981876	9.935091	298.560117

Esta salida es importante, pues muestra los valores máximos y mínimos de nuestras variables, esto lo usaremos más adelante para etiquetar los valores en bajo, medio y alto.

Usando el método *"shape"* se puede obtener la cantidad de columnas y filas del dataset, es de interés la cantidad de filas, ya que esto se traduce en la cantidad de observaciones que se han hecho para cada variable de NPK.

```
datos_df.shape
```

A la salida de esta celda se obtiene:

```
(2200, 8)
```

2.1.2. Selección y limpieza de datos

Como se vio existen 2200 observaciones, sin embargo, no se puede incluir todas ellas, esto es debido a que cada observación viene etiquetada a un tipo de cultivo. Para elaborar un modelo que capture de mejor manera el comportamiento de las variables NPK se debe limitar el análisis a los datos que pertenecen a los cereales, granos y legumbres, ya que su comportamiento es parecido y consumen más o menos los mismos valores en las mismas circunstancias, esto deja afuera a las frutas en el dataset, para aplicar el filtrado se usa el código a continuación:

```
granos_df = datos_df[(datos_df['label']=='lentil') |
                     (datos_df['label']=='chickpea') |
                     (datos_df['label']=='kidneybeans') |
                     (datos_df['label']=='pigeonpeas') |
                     (datos_df['label']=='mothbeans') |
                     (datos_df['label']=='mungbean') |
                     (datos_df['label']=='blackgram') |
                     (datos_df['label']=='rice') |
                     (datos_df['label']=='maize')]
```

Así mismo mediante el método *"shape"* obtenemos la siguiente salida:

```
(900,8)
```

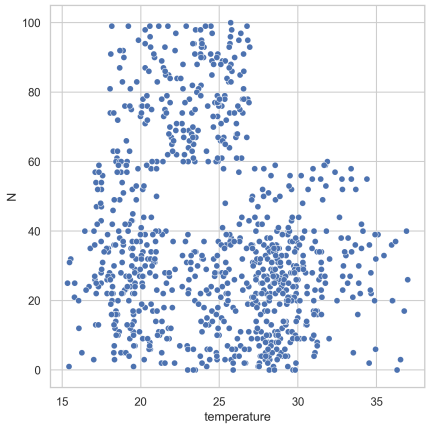
Esto indica que el dataframe se ha reducido a 900 observaciones, este número será importante más adelante a la hora de analizar los resultados.

2.1.3. Análisis Exploratorio

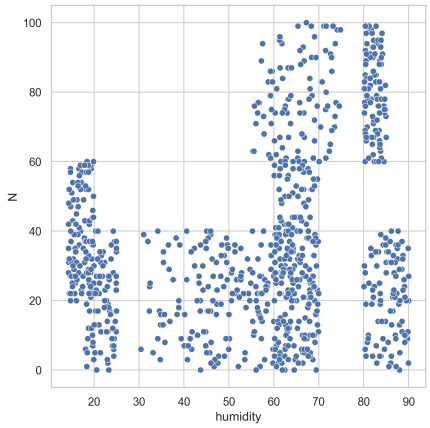
Hasta ahora se ha filtrado el dataset para conseguir la información que es de interés, sin embargo, es necesario explorar un poco más a fondo el comportamiento de las variables, en concreto se requiere hacerlo de una forma visual, para ello se hace uso de una librería bastante popular en la ciencia de datos llamada *"seaborn"*.

En este punto se crearon tres gráficos de dispersión para cada variable, donde se relacione dicha variable con los valores de temperatura, humedad y Ph.

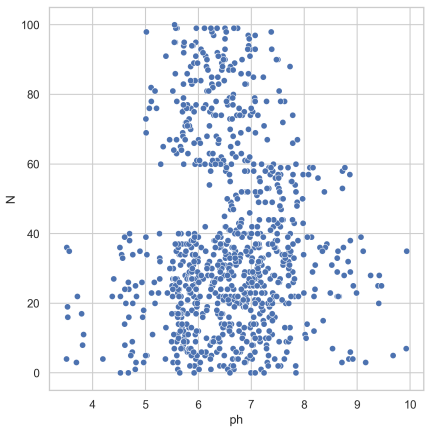
En las figuras 2.2, 2.3 y 2.4 se muestran los diagramas de dispersión para las 3 variables **N**, **P** y **K** frente a temperatura, humedad y ph. Donde se observa que K tiene más agrupados los puntos, esto podría influir en gran medida en la precisión de los modelos como se verá más adelante, por otro lado, las variables de N y P presentan una dispersión menos segmentada y en un rango de valores más amplio.



(a) Temperatura vs N.



(b) Humedad vs N.



(c) pH vs N.

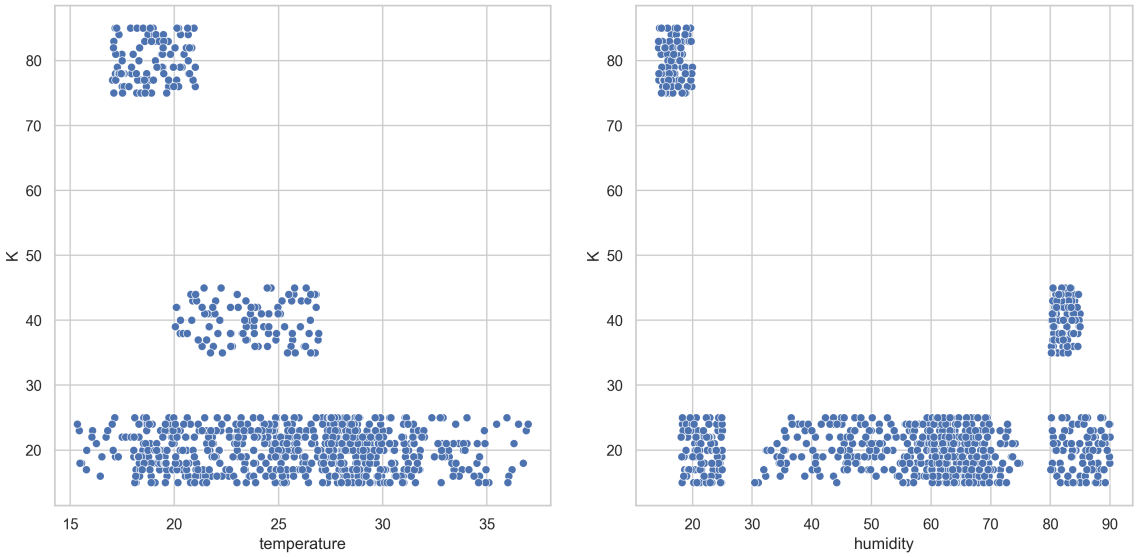
Figura 2.2: Dispersión de la variable N.
Fuente: El Autor.

(a) Temperatura vs P.

(b) Humedad vs P.

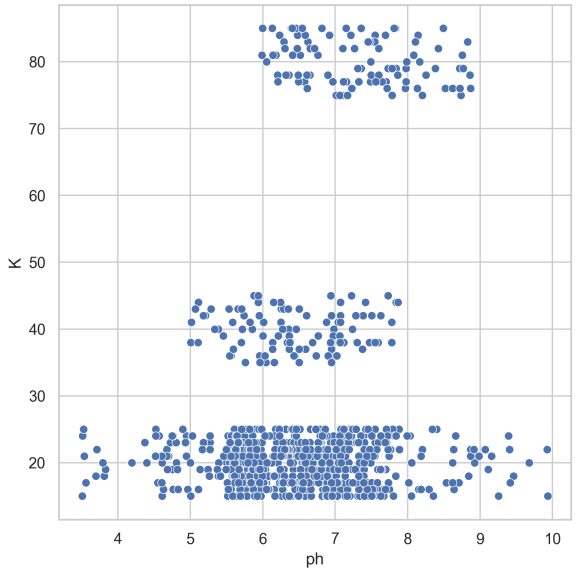
(c) pH vs P.

Figura 2.3: Dispersión de la variable P.
Fuente: El Autor.



(a) Temperatura vs K.

(b) Humedad vs K.



(c) pH vs K.

Figura 2.4: Dispersión de la variable K.
Fuente: El Autor.

Es necesario entender que tipo de relación existe entre las variables con las que alimentaremos los modelos y los valores de NPK, esto se logra con la correlación entre las variables mediante un mapa de calor, de igual manera con la librería *"seaborn"* se obtiene la figura 2.5 donde se muestra dicho mapa de calor de las correlaciones.

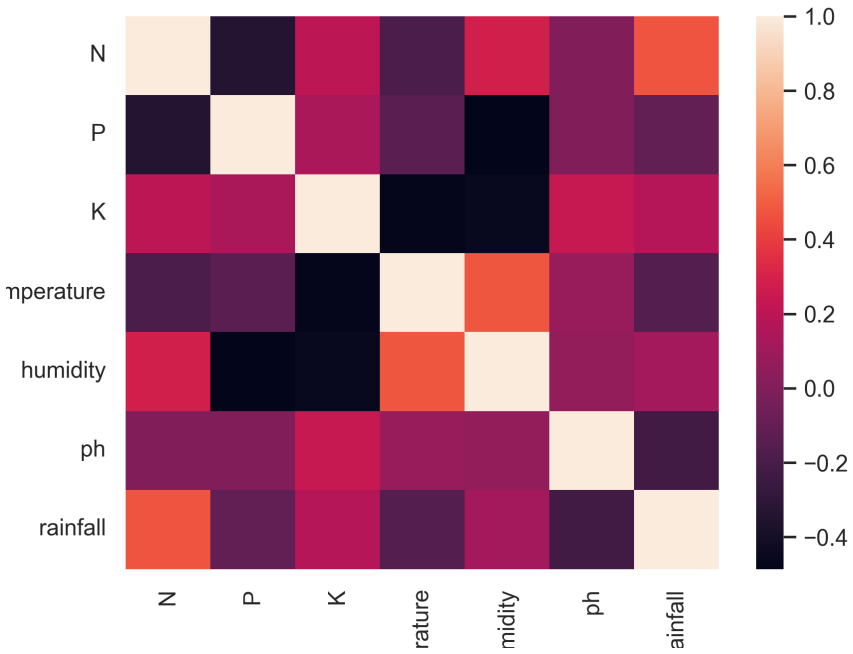


Figura 2.5: Mapa de calor acerca de la correlación entre las variables del dataset.
Fuente: El Autor.

Para tener un mejor criterio a la hora de seleccionar los modelos, se muestran las distribuciones de cada variable NPK graficadas con la librería *"matplotlib"*, las cuales se ilustran en las figuras 2.6, 2.7 y 2.8.

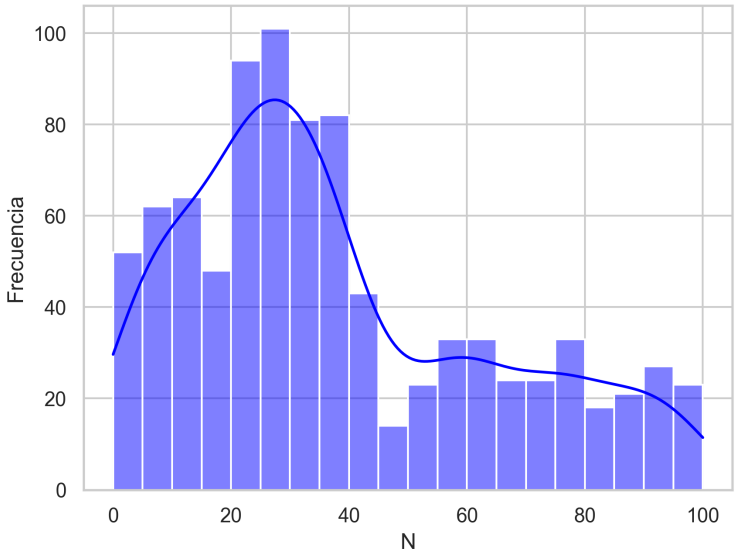


Figura 2.6: Distribución de la variable N en el dataset.
Fuente: El Autor.

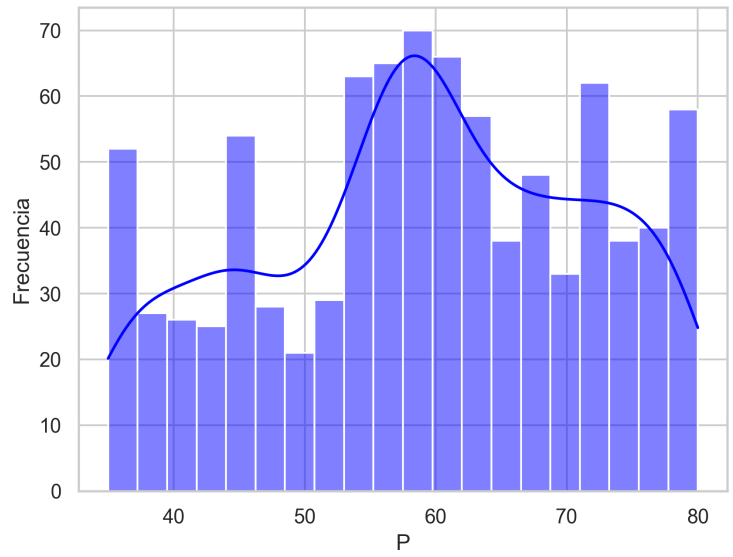


Figura 2.7: Distribución de la variable P en el dataset.
Fuente: El Autor.

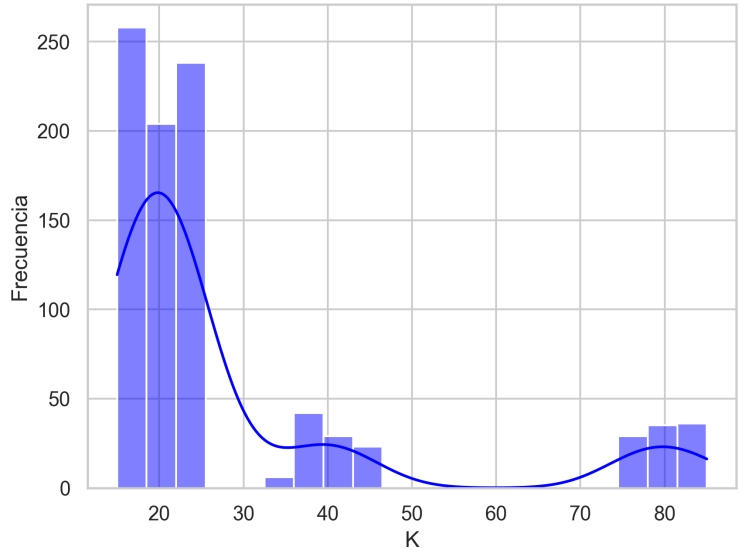


Figura 2.8: Distribución de la variable K en el dataset.
Fuente: El Autor.

Se puede entonces observar en las figuras de las distribuciones que la variable K tiene concentrado la mayor parte de sus datos en valores bajos si lo comparamos con las variables de N y P, además las distribuciones no son normales, por lo que esto sugiere el uso de un escalador estándar para escalar los datos de entrenamiento

y prueba en los casos en que el modelo de ML así lo requiera.

2.2. Modelos de ML

Un punto clave del trabajo es seleccionar los modelos más adecuados para tratar el problema planteado, es por esto que se realizó una evaluación de 3 modelos importantes en el área de la Ciencia de datos que se consideraron adecuados por su naturaleza, así mismo se ajustaron algunos parámetros para lograr la mejor eficiencia entre dichos modelos.

2.2.1. Modelos de ML adecuados y preparación de los datos

Para el desarrollo de los modelos, se ocupó la librería de python para el aprendizaje de máquina llamada *"sckit-learn"*, esta brinda métodos muy poderosos para implementar modelos de una forma relativamente sencilla. Como se vio en la revisión de la literatura, existen clasificadores más eficientes que otros a la hora de implementar los modelos para su uso en la agricultura digital, entre ellos podemos destacar SVM y KNN, además existe un método que no aparece mucho en la literatura, pero es de interés en el estudio para tener otro clasificador con el cual contrastar resultados, hablamos de los Árboles de decisión, los cuales también fueron implementados en el presente trabajo.

Árboles de Decisión

Los árboles de decisión son altamente interpretables, ya que se puede visualizar las reglas de decisión que el modelo utiliza, esto nos ayuda a entender que factores afectan más a la clasificación, en el contexto del presente trabajo sabemos que los datos pueden interactuar de maneras no triviales por lo que usar árboles de decisión puede ser una buena aproximación al problema.

SVM

SVM al ser un clasificador robusto maneja muy bien las fronteras gracias a sus diferentes tipos de kernel, en el caso del presente trabajo esta aproximación puede ser beneficiosa, ya que en el análisis exploratorio se observó fronteras complejas y no lineales.

KNN

KNN en un método más tradicional y simple, este método ofrece una perspectiva diferente a los anteriores, ya que solo se basa en distancias para clasificar los elementos del dataset, además es útil para identificar patrones locales de las subregiones del espacio de características del problema.

Preparación de Datos

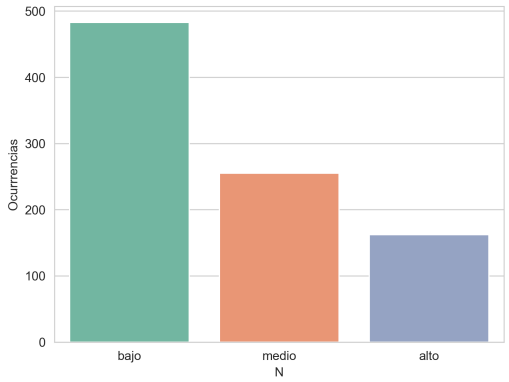
Antes de implementar dichos modelos hay que preparar los datos, como se vio en el análisis exploratorio, la variable N presenta valores en el rango de 0 a 100, por lo que teniendo este rango en cuenta se dividieron las etiquetas de la siguiente manera:

- 0-33: Bajo
- 33-66: Medio
- 66-100: Alto

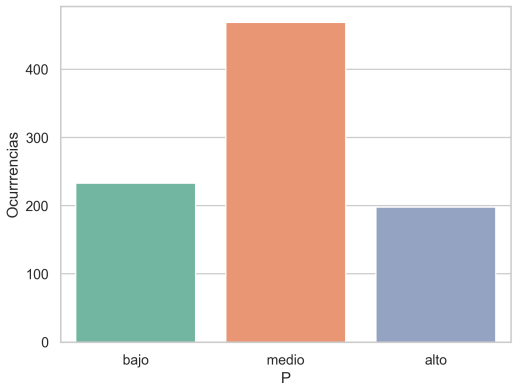
El código correspondiente para lograr esta categorización es:

```
granos_df['N'] = pd.cut(granos_df['N'], bins=[-float('inf'), 33, 66,
                                                float('inf')],
                        labels=['bajo', 'medio', 'alto'])
nombres=['bajo', 'medio', 'alto']
```

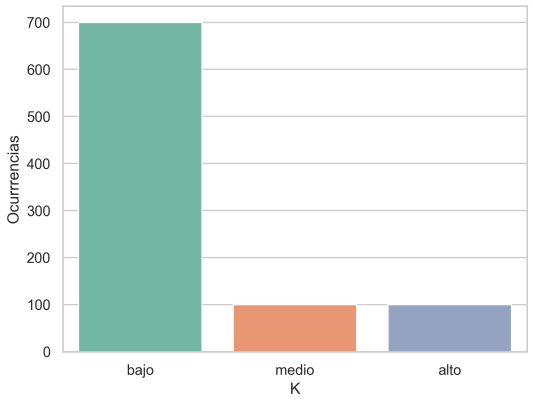
Al usar el método value_counts se obtiene el conteo de valores para cada clase, esto se observa en la figura 2.9.



(a) Ocurrencias de la variable N.



(b) Ocurrencias de la variable P.



(c) Ocurrencias de la variable K.

Figura 2.9: Ocurrencias de las variables N, P, K.
Fuente: El Autor.

Aplicando el mismo procedimiento a las variables de P y K se obtuvieron las figuras 2.9(b) y 2.9(c).

Viendo las figuras 2.9(a), 2.9(b) y 2.9(c) es evidente que es necesario aplicar alguna técnica de balanceo de clases, pues existe un sesgo considerable entre las clases, sobre todo cuando hablamos de la variable K donde el sesgo favorece en gran medida a la etiqueta bajo.

2.2.2. Entrenamiento y Ajuste

Para esta parte se detalla el proceso completo para clasificar N con los diferentes clasificadores escogidos con anterioridad.

Árboles de Decisión

Se inició entrenado un clasificador de árboles de decisión, mediante la librería sckitlearn es sencillo implementarlo en pocas líneas de código, primero se definen las variables independientes y dependientes del modelo, además se dividen los datos en data de entrenamiento y data de prueba, para esto se escogieron los valores típicos recomendados en la Ciencia de Datos que son 80 % de los datos para entrenamiento y 20 % serán usados para evaluar los modelos.

```
X = granos_df[['temperature', 'humidity', 'ph', 'rainfall']]
y = granos_df[['N']]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state=42)
```

Se uso Smote Tomek para balancear las clases.

```
from imblearn.combine import SMOTETomek


smote = SMOTETomek(random_state=139)
X_train, y_train = smote.fit_resample(X_train, y_train)
```

Ahora se declara el clasificador de árboles de decisión con 2 parámetros esenciales, la profundidad del árbol y el estado aleatorio, luego de esto, entrenamos con la data de entrenamiento. Se debe aclarar que en este caso se usa una profundidad de árboles de 6, esto se justificara más adelante.

```
clf = DecisionTreeClassifier(max_depth=6, random_state=42)
clf.fit(X_train, y_train)
```

Ahora resta evaluar el modelo con el código que se muestra a continuación.

```
    y_pred = clf.predict(X_test)
    print("Exactitud: {:.3f} %".format(accuracy_score(y_test, y_pred) *
        ↪  100))
    print(classification_report(y_test, y_pred))
```

Esta última celda tiene 2 salidas, la exactitud del modelo entrenado y evaluado con esa data en específico y un reporte de clasificación, donde se nos muestran métricas importantes como lo son la precisión, el recall y el f1-score.

Exactitud: 66.111 %

	Precision	Recall	f1-score	Support
alto	0.79	0.81	0.80	32
bajo	0.77	0.68	0.72	96
medio	0.44	0.54	0.49	52
accuracy			0.66	180
macro avg	0.67	0.68	0.67	180
weighted av	0.68	0.66	0.67	180

La matriz de confusión del rendimiento del modelo se observa en la figura 2.10 donde se puede contrastar los valores precedidos por el modelo con los valores reales.

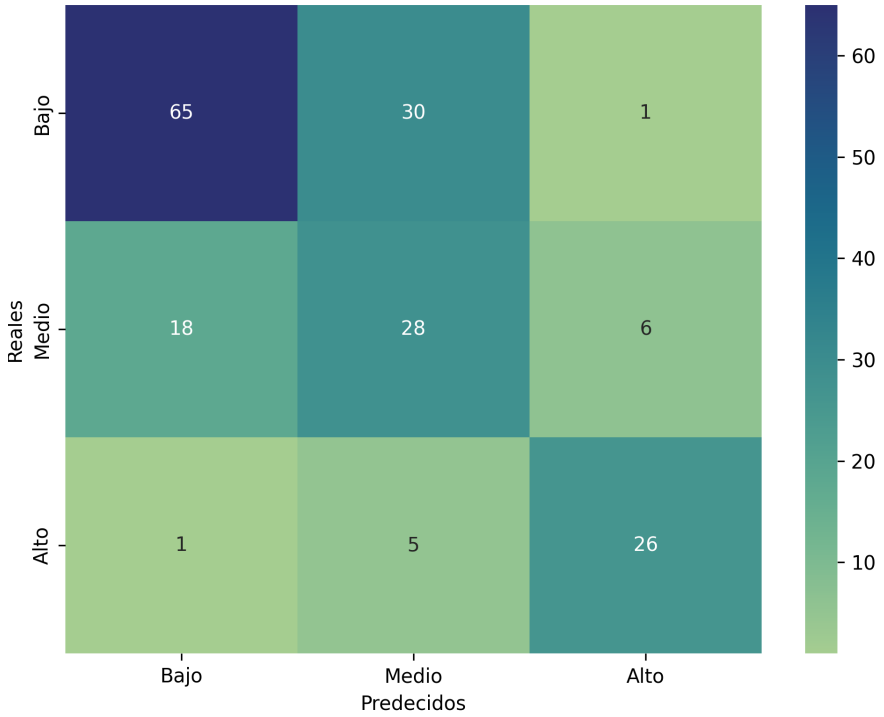


Figura 2.10: Matriz de confusión para árboles de decisión con profundidad=6.
Fuente: El Autor.

Es importante evaluar la precisión del modelo con validación cruzada k-fold, para obtener una métrica acertada del rendimiento del modelo, al ejecutar el código necesario para evaluar este modelo con 10 seccionamientos se obtuvo:

Precisión con Validación Cruzada de: 66.89 %

Para obtener el mejor rendimiento del modelo en términos de la profundidad de los árboles se hace uso del lazo for para comprobar mediante validación cruzada la precisión del modelo para diferentes valores del parámetro maxdepth.

```
import csv

nombre_archivo = "resultadosArbolDecision.csv"

with open(nombre_archivo, mode='w', newline='') as archivo_csv:
    escritor = csv.writer(archivo_csv)

    #Encabezado
    escritor.writerow(["MaxDepth", "Precision"])

    #Bucle para la validación cruzada
    for i in range(1,30):
        clf = DecisionTreeClassifier(max_depth=i, random_state=42)
        clf.fit(X_train, y_train)
        scores = cross_val_score(clf, X, y, cv=10)
        escritor.writerow([i, round(scores.mean() * 100,3)])
```

Luego de esto se obtiene un archivo .csv y los datos que contiene este se ilustran en la figura 2.11.

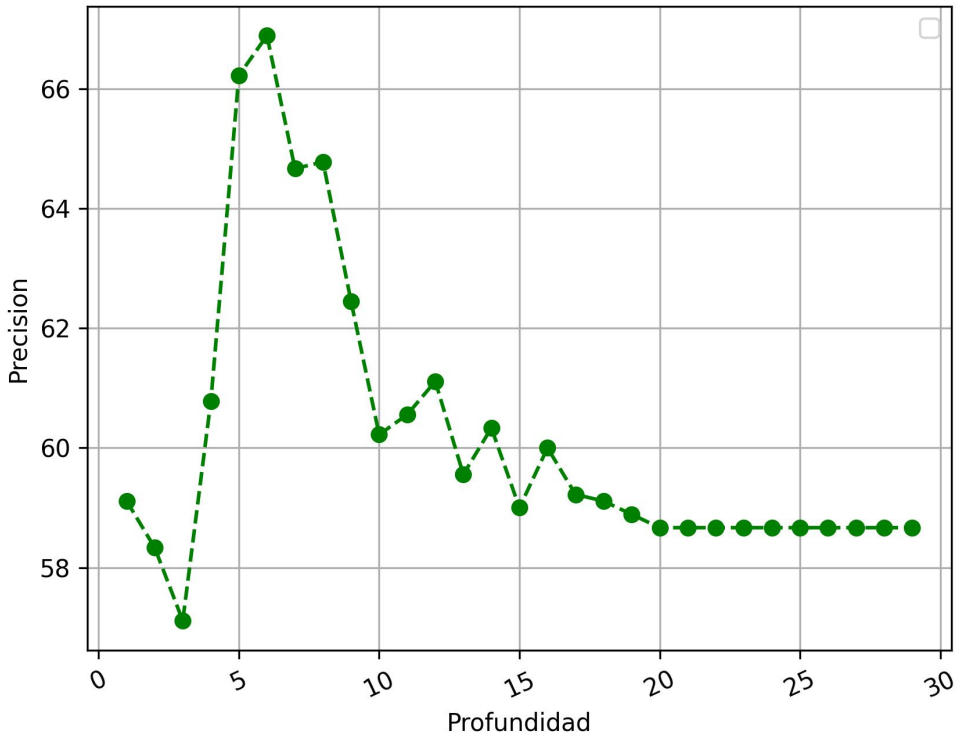


Figura 2.11: Curva de validación cruzada para árboles de decisión.
Fuente: El Autor.

Máquinas de soporte vectorial (SVM)

Una vez dividida la data en entrenamiento y evaluación se declara el clasificador SVM con un kernel rbf, ya que debido a la naturaleza de los datos, esto podrá capturar patrones más complejos en las fronteras y mejorar el rendimiento.

```
svm_model = SVC(kernel='rbf', gamma=0.011)
svm_model.fit(X_train, y_train)
```

Se hace uso del método *"clasification_report"* para generar el reporte con todas las métricas importantes, además se muestra la exactitud de este clasificador para la data de entrenamiento en concreto.

Exactitud: 69.44 %

	Precision	Recall	f1-score	Support
alto	0.76	0.97	0.85	32
bajo	0.78	0.72	0.75	96
medio	0.50	0.48	0.49	52
accuracy			0.69	180
macro avg	0.68	0.72	0.70	180
weighted av	0.69	0.69	0.69	180

La matriz de confusión se ilustra en la figura 2.12.

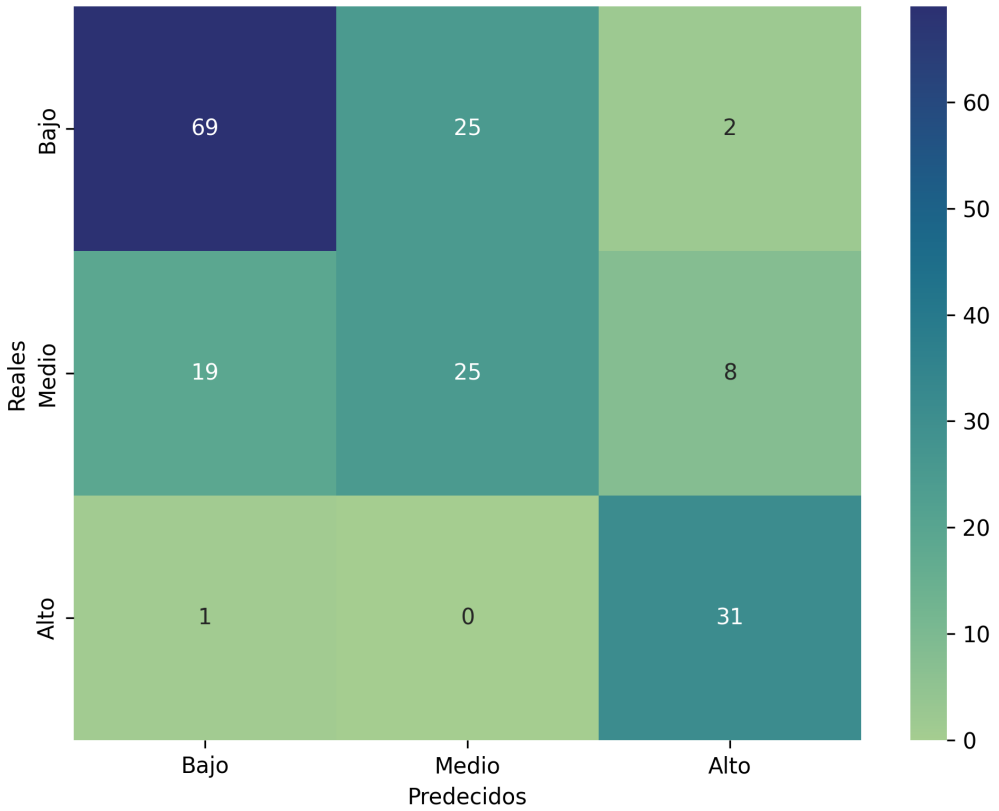


Figura 2.12: Matriz de confusión para SVM con gamma=0.011.
Fuente: El Autor.

La precisión con validación cruzada con 10 folds resulta ser:

Precisión con Validación Cruzada de: 70.44 %

Ahora mediante el lazo for se obtiene el valor de gamma que brinda la mejor precisión, el resultado para cada una de las iteraciones realizadas se muestra en la figura 2.13.

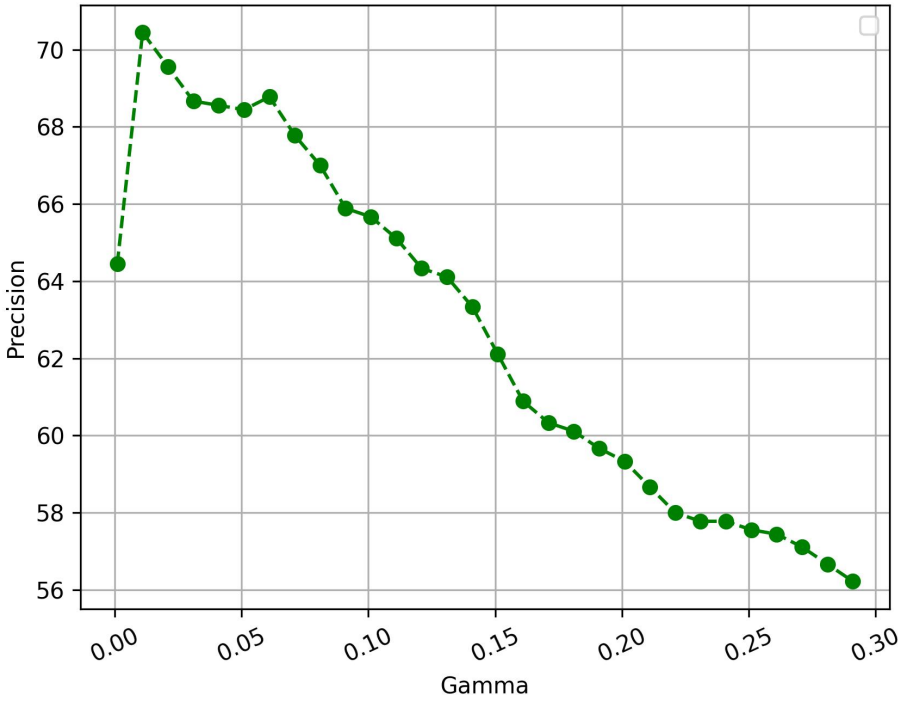


Figura 2.13: Curva de validación cruzada para SVM.
Fuente: El Autor.

K vecino más próximo (KNN)

Para este método, aparte de reducir el sesgo con SMOTE se debe escalar los datos con un escalamiento estándar dado que los datos no siguen una distribución normal.

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Ahora se declara el clasificador, donde se coloca 15 k vecinos y p=2, este último para manejar la distancia euclidiana a la hora de calcular distancias.

```
knn = KNeighborsClassifier(n_neighbors=15, metric='minkowski', p=2)
knn.fit(X_train_scaled, y_train)
```

La precisión de este modelo para esos datos de entrenamiento en concreto resulta ser:

Exactitud: 73.889 %

El reporte de clasificación resultan ser:

	Precision	Recall	f1-score	Support
alto	0.76	0.97	0.85	32
bajo	0.77	0.81	0.79	96
medio	0.63	0.46	0.53	52
accuracy			0.74	180
macro avg	0.72	0.75	0.72	180
weighted av	0.73	0.74	0.73	180

La matriz de confusión obtenida para este clasificador, es la figura 2.14.

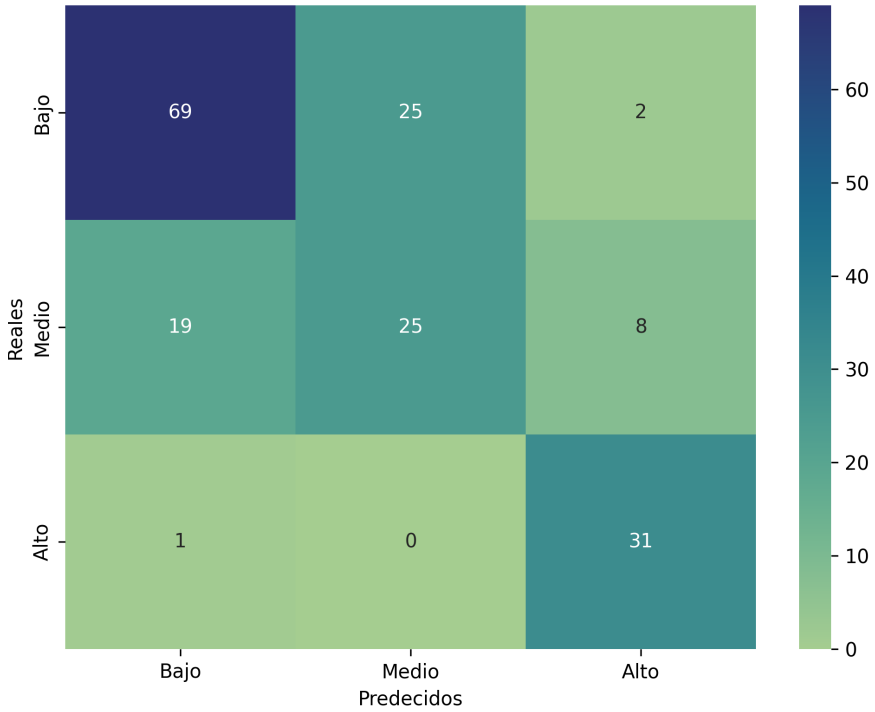


Figura 2.14: Matriz de confusión para KNN con vecinos=15.
Fuente: El Autor.

Al aplicar la validación cruzada se obtuvo:

Precisión con Validación Cruzada de: 69.11 %.

La curva de precisión usando el bucle for para encontrar el No de vecinos que mejor rendimiento otorga se muestra en la figura 2.15.

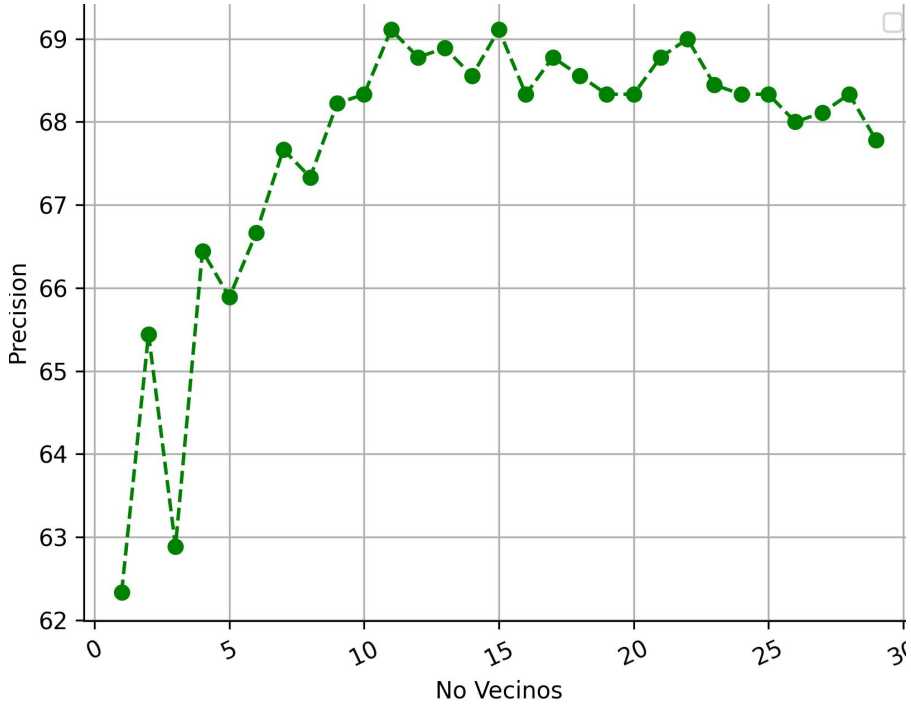


Figura 2.15: Curva de validación cruzada para KNN.
Fuente: El Autor.

2.3. Dashboard y Sistema Recomendador

En esta sección se describe el desarrollo final del trabajo donde se implementa el modelo que mejor precisión dio para cada valor de nutrientes del suelo (NPK) junto con la IA de Gemini mediante su API para mostrar recomendaciones en un dashboard en conjunto con las gráficas de las variables meteorológicas de los nodos.

2.3.1. Herramientas de IA y nube utilizadas

La primera parte de la figura 2.1 ya se realizó en la sección anterior, ahora se crea el script para el sistema recomendador, para ello necesitaremos usar una API de Gemini, esta la generamos mediante el servicio de computación en la nube de Google, en la figura 2.16 se muestra el entorno de desarrollo de nuestro proyecto en la nube donde se puede monitorear el tráfico de la API y así controlar su uso.

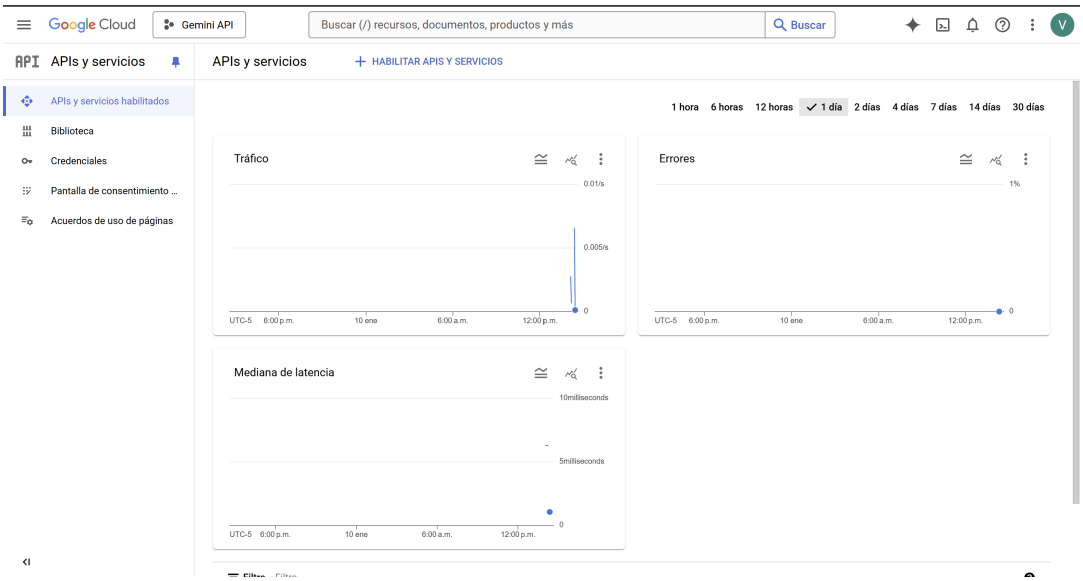


Figura 2.16: Entorno de la API de Google para Gemini.
Fuente: El Autor.

2.3.2. Sistema Recomendador

Se debe crear una clave de API llamada **Generative Language API Key**, luego esta la usaremos para crear el script de python. La clave de API no se muestra en el ejemplo de código por motivos de confidencialidad. Se declara primero el modelo que en este caso para fines educativos está puesto Gemini 1.5 flash porque es rápido y funciona bastante bien para la mayoría de casos, luego se indican las instrucciones en el modelo donde damos el contexto de como se quiere que responda y luego se envía otro string que se refiere a lo que se quiere preguntar. La implementación del sistema recomendador se la realiza en conjunto con los modelos para N, P y K que ya fueron entrenados con anterioridad y guardados en archivos .pkl para ser usados cuando sea necesario, entonces en este script solo se llama a dichos modelos para clasificar

NPK basándose en los parámetros de temperatura, humedad, pH y precipitación, también hay que darle contexto al modelo, es decir, se debe armar el escenario para que sus respuestas sean acorde a ello, lo cual se logra mediante la instrucción *system_instruction*, esta última recibe todo lo referente al proyecto, entonces el script queda de la siguiente manera:

```
import pandas as pd
import numpy as np
import joblib
import google.generativeai as genai

#Cargamos el clasificador
clasificadorN = joblib.load('modeloSVM_N.pkl')
clasificadorP = joblib.load('modeloArboles_P.pkl')
clasificadorK = joblib.load('modeloSVM_K.pkl')
#Declaramos los nombres de las columnas que se usaron para entrenar el
    ↪ modelo
nombres_columnas = ['temperature', 'humidity', 'ph', 'rainfall']

#Simulamos el envio de datos de los nodos
LecturaNodos = pd.DataFrame([[20, 20, 7, 200]],
    ↪ columns=nombres_columnas)
prediccionN = clasificadorN.predict(LecturaNodos)
prediccionP = clasificadorP.predict(LecturaNodos)
prediccionK = clasificadorK.predict(LecturaNodos)

#Configuramos la API de Google
genai.configure(api_key="ApiOcultaPorSeguridad")
model = genai.GenerativeModel(model_name="gemini-1.5-flash",
```

```
                                    system_instruction="Eres un biologo
                                        ↪ experto en cultivos de rosas y
                                        ↪ necesito que me des recomendaciones
                                        ↪ basadas en los valores de NPK del
                                        ↪ suelo, cuando yo te diga que N es
                                        ↪ bajo significa que esta entre un
                                        ↪ valor de 0-33, si te digo N medio
                                        ↪ es que esta en un valor de 33-66 y
                                        ↪ si digo alto es un valor mayor a
                                        ↪ 66. para P los valores son bajo=
                                        ↪ 0-50, medio= 50-70 y alto mayor a
                                        ↪ 70, para K los valores son bajo=
                                        ↪ 0-30, medio= 30-70 y alto mayor a
                                        ↪ 70, ademas necesito respuestas que
                                        ↪ no superen los 200 caracteres y que
                                        ↪ expliquen las consecuencias de
                                        ↪ dejar el cultivo con dichos
                                        ↪ valores")
pregunta = "Que opinas si mi cultivo de rosas tiene valores de
    ↪ Nitrogeno:" + str(prediccionN[0] + ", Fosforo:" +
    ↪ str(prediccionP[0]) + ", Potasio:" + str(prediccionK[0] ))
response = model.generate_content(pregunta)
print(response.text)
```

Este último script se ejecuta en una instancia nueva de AWS para generar reportes periódicos con Gemini.

2.3.3. Integración de los datos con la nube

Debido a que hasta la fecha de elaboración de este trabajo los nodos no están disponibles para capturar la información se usa un script de python para simular el envío de datos y así poder levantar los dashboards en la plataforma de thingsboard,

en la figura 2.17 se observa el dispositivo declarado en el servidor IoT donde se le ha colocado el nombre de **MonitoreoNodos** debido a que este recibirá toda la temeraria de los nodos.

Primero se declara un dispositivo en thingsboard que recibirá los datos de los nodos de la florícola y también las recomendaciones que realice Gemini.

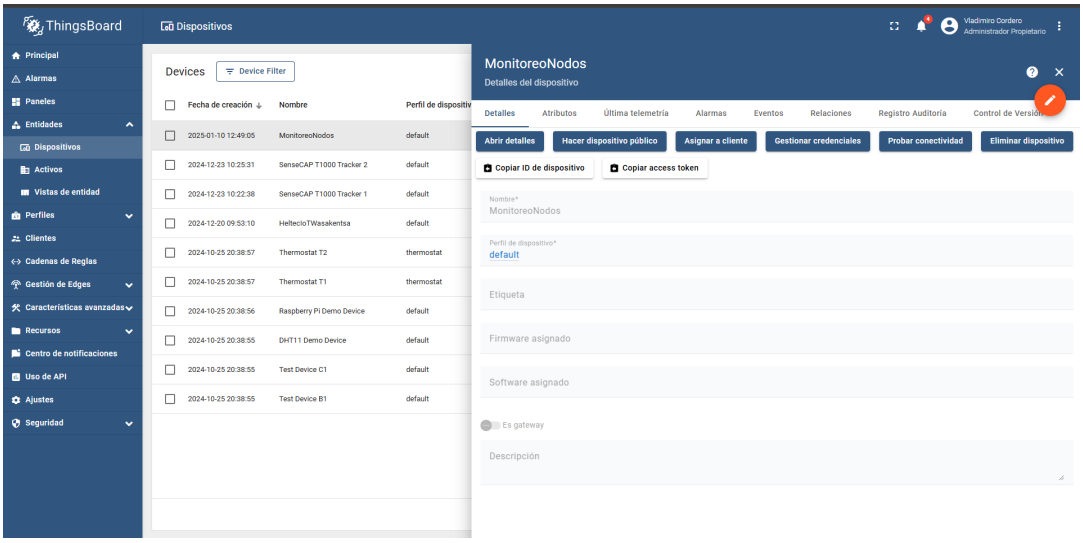


Figura 2.17: Dispositivo declarado en Thingsboard.
Fuente: El Autor.

Para simular un envío de datos hacia la plataforma IoT se toman los datos del mismo archivo csv que fue usado para entrenar los modelos ya que cuenta con una cantidad considerable de datos, el token del dispositivo de Thingsboard fue censurado en esta parte por fines de confidencialidad.

```
import csv
import time
import requests


#Configuramos la direccion de envio, junto con el token de nuestro
    ↪ dispositivo
THINGSBOARD_URL =
    ↪ "http://3.15.222.219:8080/api/v1/token-censurado/telemetry"
CSV_FILE_PATH = "ML\Crop_recommendation.csv"
```

```python
#Declaramos las columnas que se enviaran
COLUMNAS_FILTRADAS = ['temperature', 'humidity', 'ph', 'rainfall']

#Definimos funcion que recibe el archivo csv, la url y las columnas a
    ↪ enviar
def enviar_telemetria(csv_file_path, url, columnas_filtradas):
    with open(csv_file_path, 'r') as file:

        reader = csv.DictReader(file)  #Leemos el archivo por columnas

        for row in reader:
            #Filtramos solo las columnas especificadas
            payload = {
                columna: float(row[columna]) for columna in
                    ↪ columnas_filtradas if columna in row
            }
            #Convertimos a formato JSON conforme lo requiere thinsboard
            payload_formateado = "{" + ", ".join([f"{key}:{value}" for
                ↪ key, value in payload.items()]) + "}"
            headers = {'Content-Type': 'application/json'}

            try:
                #Enviamos los datos al servidor
                response = requests.post(url, headers=headers,
                    ↪ data=payload_formateado)
                if response.status_code == 200:
                    print(f"Datos enviados: {payload_formateado}")
                else:
                    print(f"Error al enviar los datos: {response.text}")
            except Exception as e:
                print(f"Error de conexión: {e}")
```

```
        time.sleep(1)  # Pausa de 1 segundo entre envíos

#Ejecutamos la funcion
enviar_telemetria(CSV_FILE_PATH, THINGSBOARD_URL, COLUMNAS_FILTRADAS)
```

Se ejecuta entonces el script para enviar los datos y se levantan las gráficas en tiempo real dentro de un nuevo dashboard al que llamaremos **Monitoreo florícola** y así podemos comprobar su funcionamiento en las figuras 2.18 y 2.19 donde se muestran las 4 gráficas usadas en el monitoreo las cuales se encuentran en el mismo dashboard.

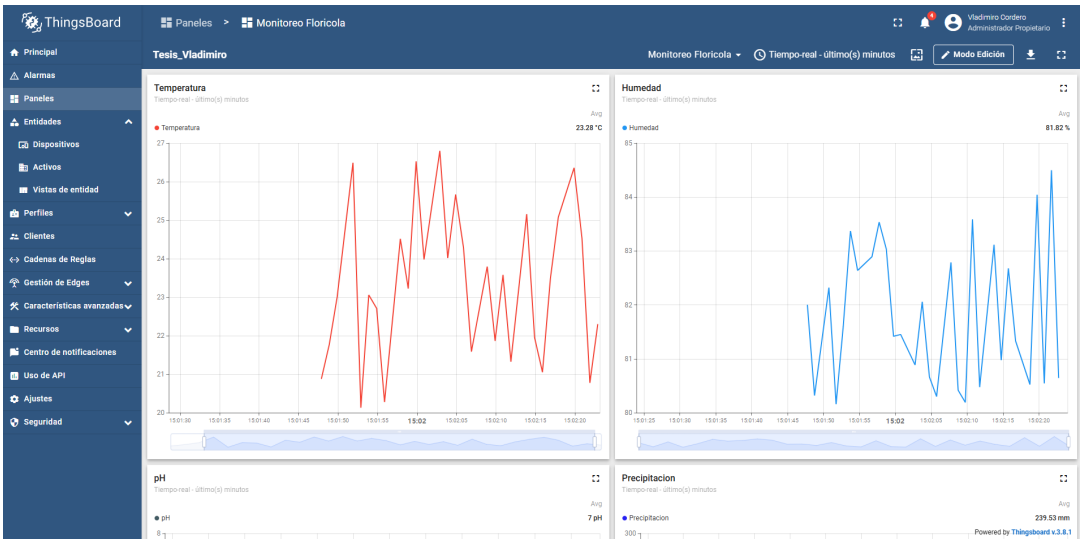


Figura 2.18: Gráficas de Temperatura y Humedad en el dashboard.
Fuente: El Autor.

Figura 2.19: Gráficas de pH y Precipitación en el dashboard.
Fuente: El Autor..

El sistema recomendador con Gemini se implementó en un nuevo dashboard, esto se logra mediante el envío de texto al mismo dispositivo que se creó en la plataforma IoT para el monitoreo de las variables ambientales, para ello será necesario agregar la parte del envío de datos al script del sistema recomendador y configurar un widget HTML en Thingsboard para mostrar el texto recibido. La parte del envío de datos de nuestro script queda de la siguiente manera:

```
#Iniciamos el envio del texto a Thingsboard

texto = re.sub(r"(\*\*|##|[*-]+)", "", response.text)

#Quitamos los saltos de linea adicionales que pueda ocacionar la
    ↪ generación de texto
texto = re.sub(r"\n\s*\n", "\n", texto)

url="http://3.15.222.219:8080/api/v1/BbYct4EbzZLpCUDLbWds/telemetry"

data = {
    "RespuestaGemini": texto
}
```

```
json_data = json.dumps(data)

headers = {'Content-Type': 'application/json'}
response = requests.post(url, headers=headers, data=json_data)

if response.status_code == 200:
    print("Texto enviado correctamente a ThingsBoard")
else:
    print(f"Error al enviar el texto: {response.status_code},
        ↪ {response.text}")
```

El código HTML para el widget se muestra a continuación:

```
<div class='card'>
    <div class='content'>
        <div class='column'>
            <h1>Recomendacion de Gemini:</h1>
            <div class='value'>
                ${RespuestaGemini}
            </div>
        </div>
    </div>
</div>
```

Nótese que se está usando RespuestaGemini en el script de HTML como el nombre de la variable que llega al dispositivo declarado en thingsboard, esto es acorde con el script de envío de datos, donde se envían las telemetrias de la respuesta de Gemini con ese mismo nombre a la plataforma IoT de thingboard.

El código CSS para el widget se muestra a continuación:

```
.card {
   width: 100%;
   height: 100%;
   border: 2px solid #67a2cf;
```

```
    box-sizing: border-box;
}

.card .content {
    padding: 20px;
    display: flex;
    flex-direction: row;
    align-items: center;
    justify-content: space-around;
    height: 100%;
    box-sizing: border-box;
}

.card .content .column {
    display: flex;
    flex-direction: column;
    justify-content: space-around;
    height: 100%;
}

.card h1 {
     text-transform: uppercase;
     color: #c23e11;
     font-size: 30px; /* Puedes ajustar este valor */
     font-weight: bold;
     margin: 0;
     padding-bottom: 10px;
     line-height: 24px; /* Puedes ajustar este valor si es necesario */
}

.card .value {
```

```
    font-size: 30px; /* Ajustado para ser más pequeño */
    font-weight: 300;
    color: #09385c;
}

.card .description {
    font-size: 16px; /* Ajustado para ser más pequeño */
    color: #909;
}
```

Capítulo 3

Resultados

Los modelos presentaron un rendimiento aceptable para el dataset con el que se entrenaron, así mismo se escogieron los modelos que mejor rendimiento dieron para cada variable de NPK y se implementaron con éxito en un sistema recomendador con la IA de Google Gemini.

3.1. Precisión de los Modelos

Los modelos que mejor rendimiento mostraron fueron SVM para el caso de N y K y Árboles de decisión para el caso de P. Esto es evidente pues cada valor de NPK presenta una distribución diferente lo que resulta en que diferentes clasificadores presenten rendimientos diferentes para cada caso puesto que no existe un clasificador que sea mejor que los demás en todas las aplicaciones. A continuación se muestran los resultados obtenidos en cuanto a la matriz de confusión y precisión mediante validación cruzada para las variables de P y K ya que los resultados de N se expusieron en el capítulo anterior como parte del desarrollo y se pueden consultar en las figuras 2.10-2.15, así entonces los resultados para P y K se muestran en las figuras 3.1-3.12.

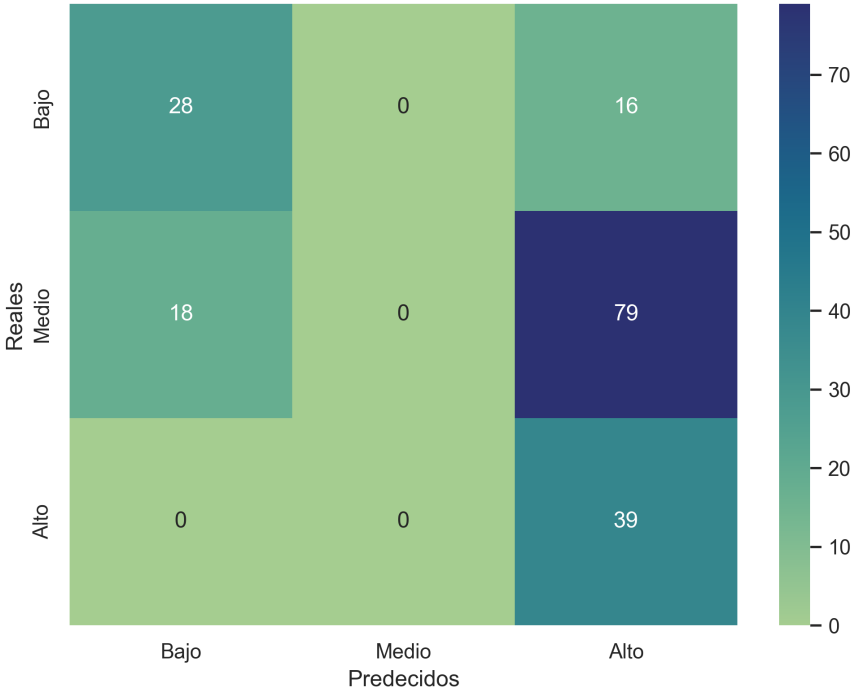


Figura 3.1: Matriz de confusión de P para árboles de decisión con Profundidad=1.
Fuente: El Autor.

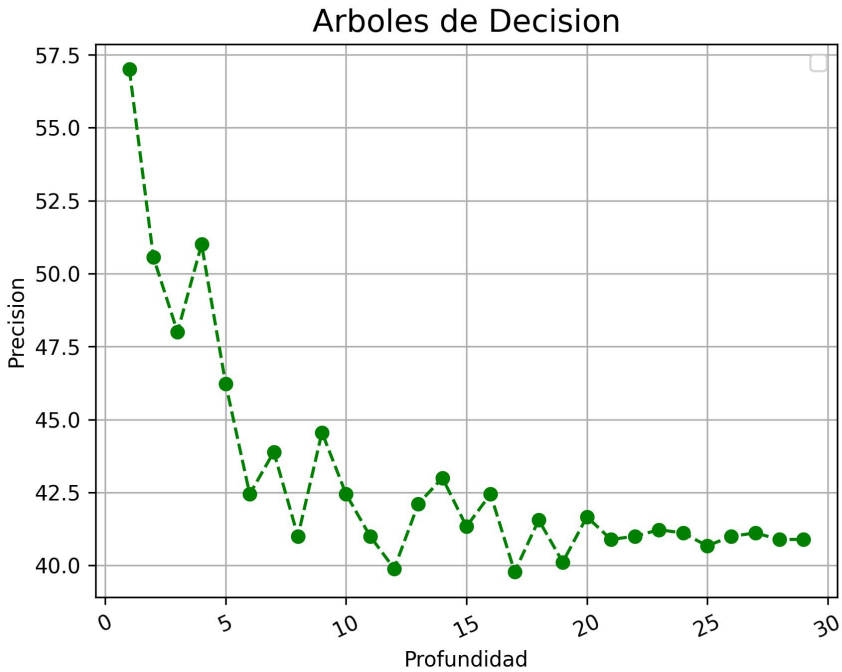


Figura 3.2: Curva de validación cruzada de P para árboles de decisión.
Fuente: El Autor.

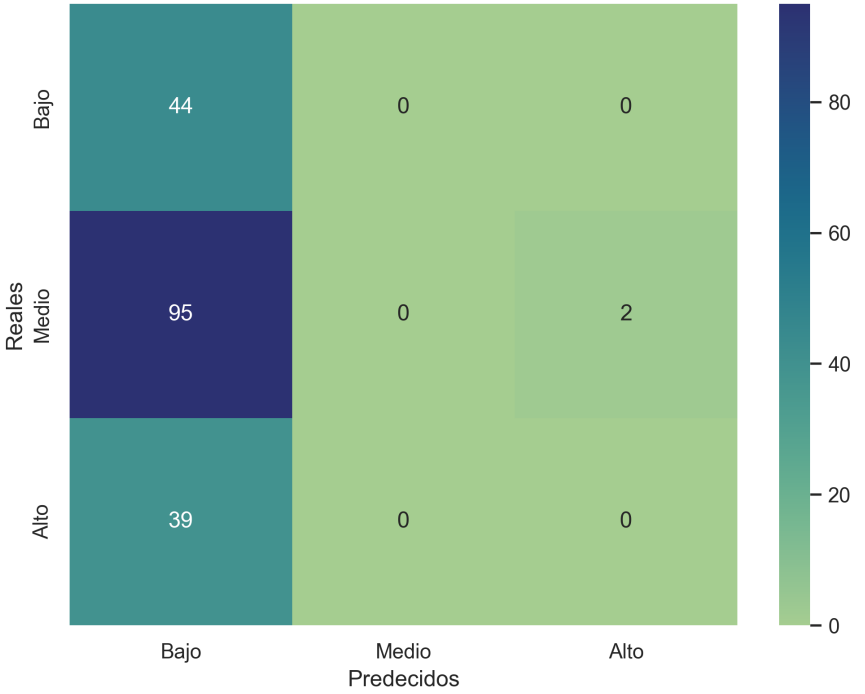


Figura 3.3: Matriz de confusión de P para SVM con Gamma=2.
Fuente: El Autor.

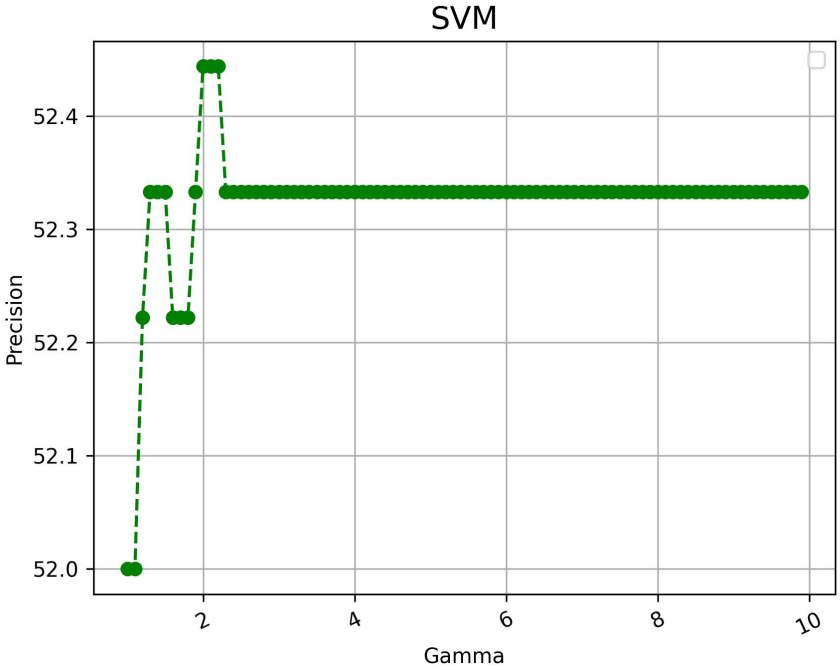


Figura 3.4: Curva de validación cruzada de P para SVM.
Fuente: El Autor.

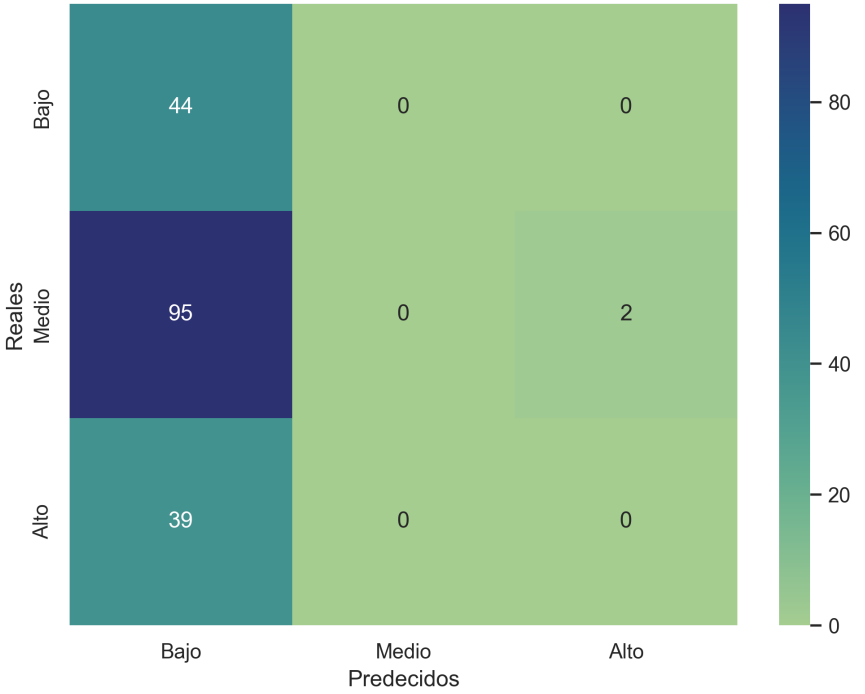


Figura 3.5: Matriz de confusión de P para KNN con vecinos=1.
Fuente: El Autor.

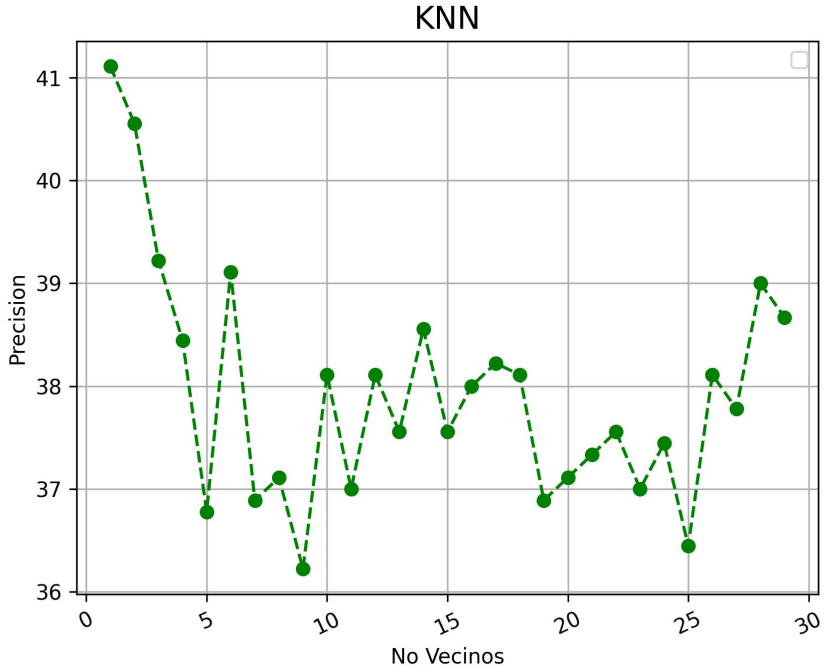


Figura 3.6: Curva de validación cruzada de P para KNN.
Fuente: El Autor.

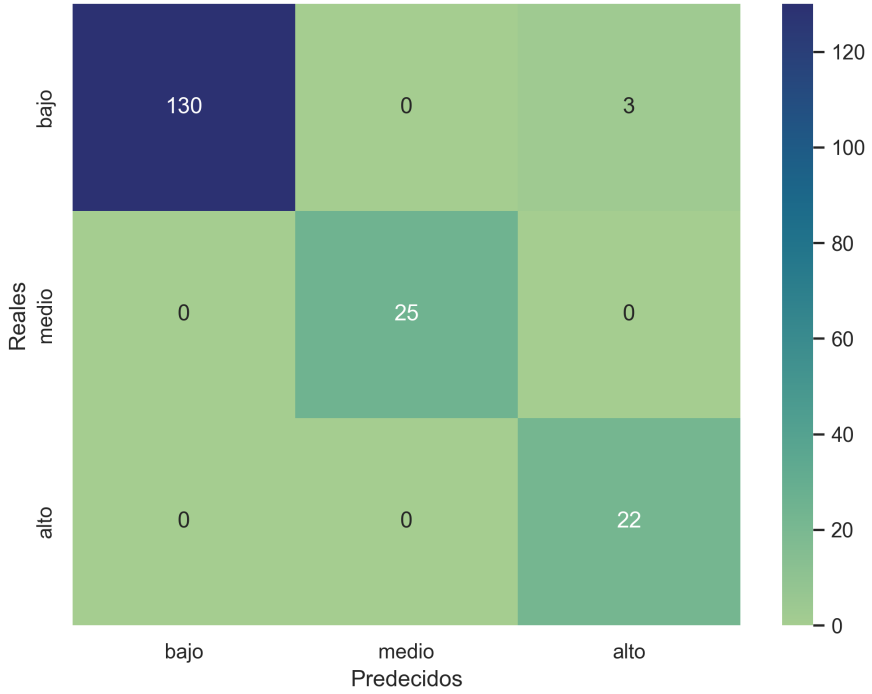


Figura 3.7: Matriz de confusión de K para árboles de decisión con Profundidad=5.
Fuente: El Autor.

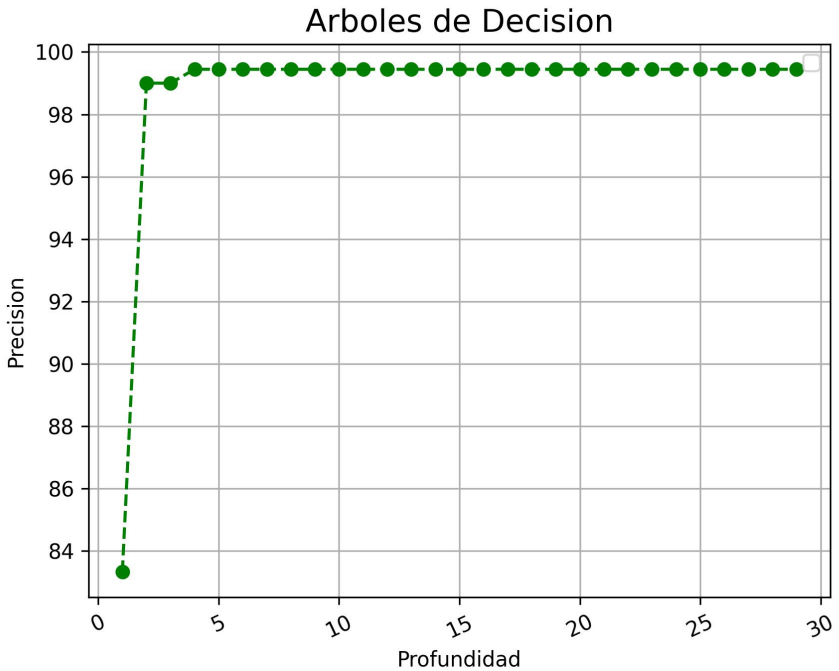


Figura 3.8: Curva de validación cruzada de K para árboles de decisión.
Fuente: El Autor.

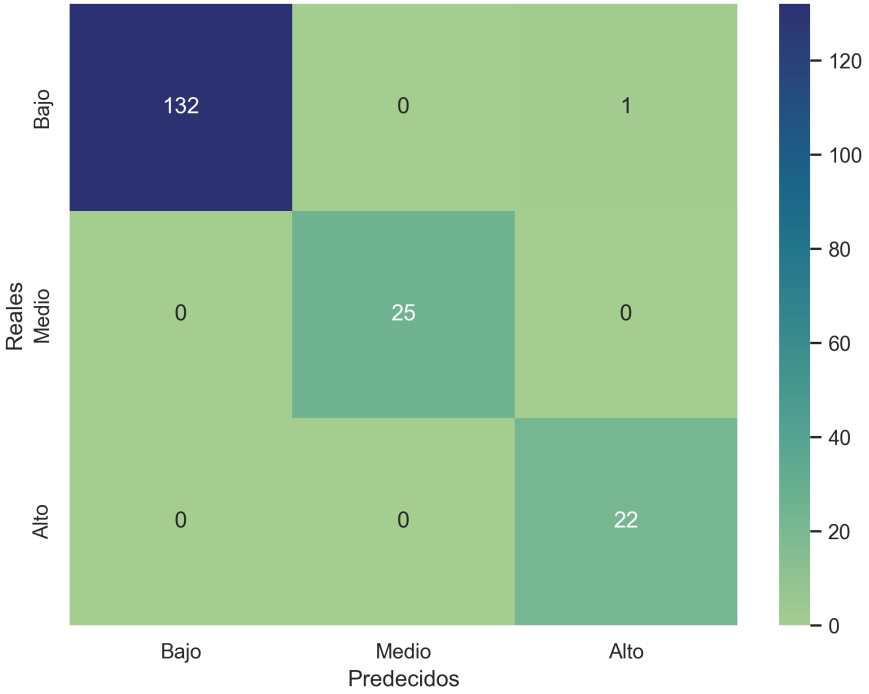


Figura 3.9: Matriz de confusión de K para SVM con Gamma=0,051.
Fuente: El Autor.

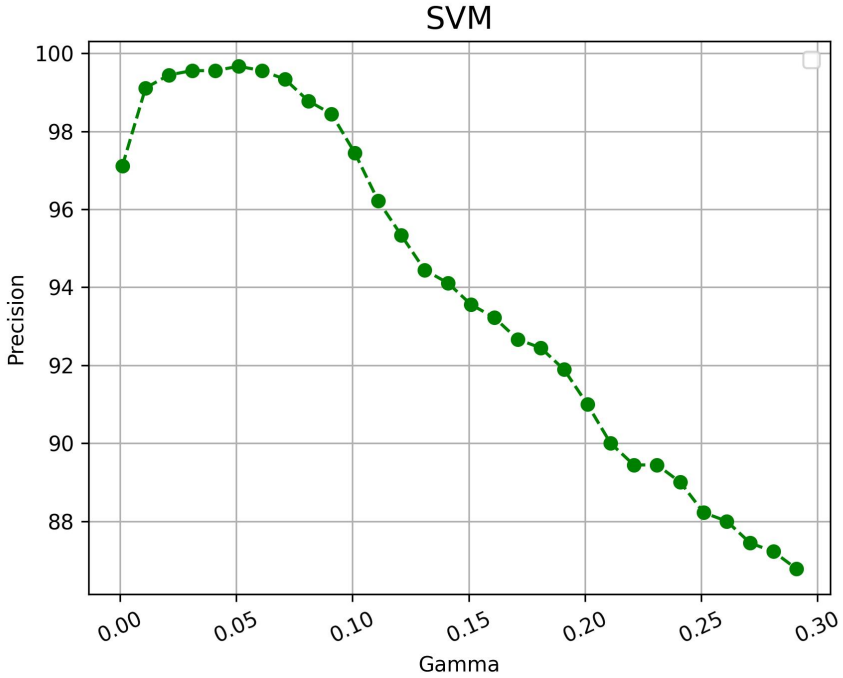


Figura 3.10: Curva de validación cruzada de K para SVM.
Fuente: El Autor.

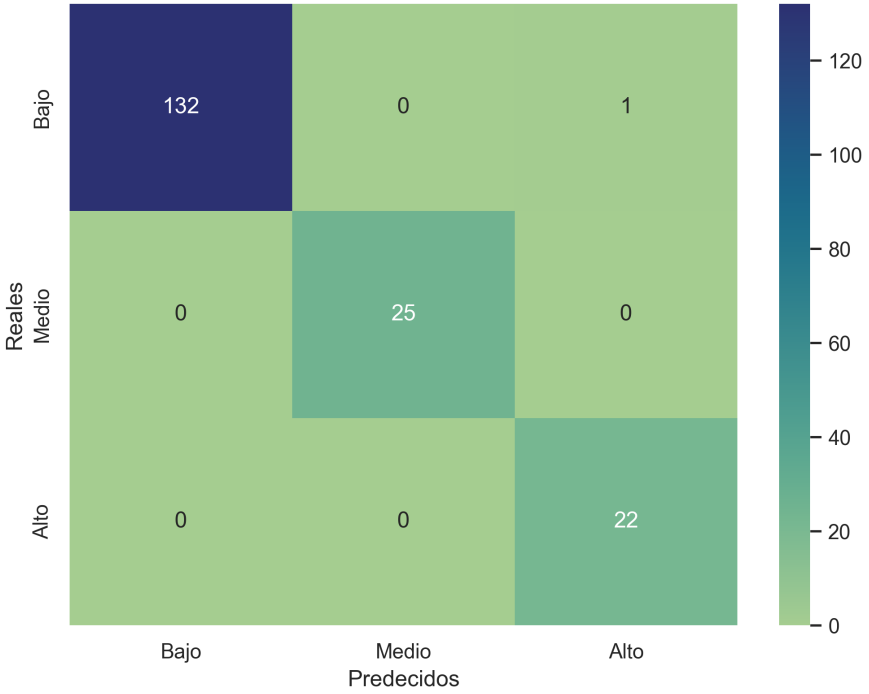


Figura 3.11: Matriz de confusión de K para KNN con vecinos=1.
Fuente: El Autor.

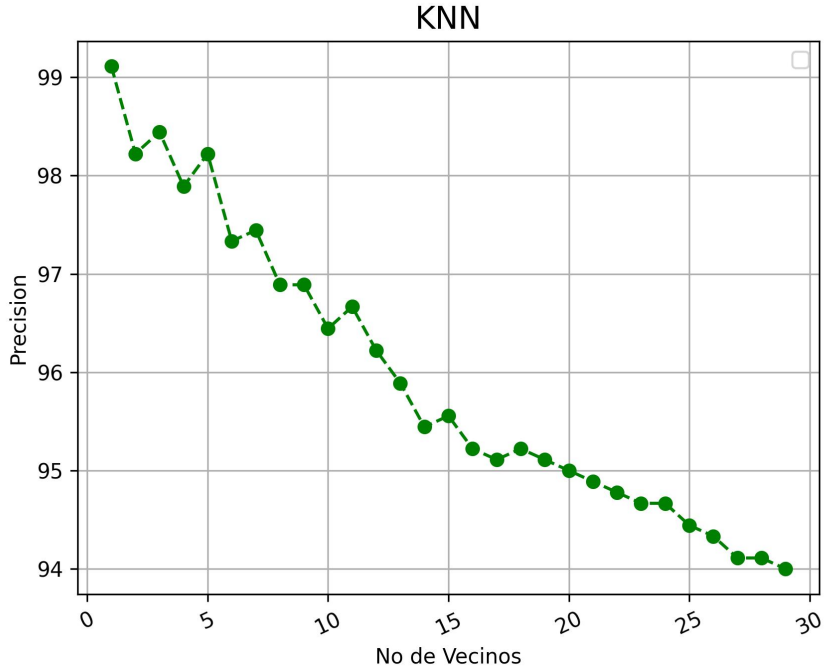


Figura 3.12: Curva de validación cruzada de K para KNN.
Fuente: El Autor.

Las figuras 3.3 y 3.5 muestran un comportamiento interesante de los clasificadores SVM y KNN debido a que se puede deducir que estos clasificadores no capturan bien los patrones y clasifican la mayor cantidad de datos en bajo, de hecho, tienen un comportamiento muy similar para la misma semilla aleatoria a la hora de segmentar los datos en entrenamiento y prueba, sin embargo, las figuras 3.4 y 3.6 muestran la diferencia sustancial entre estos 2 clasificadores, donde se observa que cuando se evalúa mediante validación cruzada el modelo de SVM tiene un comportamiento que difiere bastante a KNN llegando a tener mejores valores de precisión.

Cuando se habla de K el panorama se vuelve más complejo de evaluar, esto es debido a que en los datos de prueba existe un des-balance total en las clases y la mayor parte está contenida en la clase bajo, esto es evidente si observando las gráficas de dispersión en la figura 2.4 en el análisis exploratorio, donde existe una gran cantidad de observaciones en los valores bajos de K, es así entonces que las matrices de confusión que corresponden al análisis de esta variable K van a tener un gran sesgo que favorecerá a la etiqueta bajo, sin embargo, es importante destacar que como los modelos se han entrenado usando técnicas de balanceo de clases estos pueden rendir de una forma aceptable en datos no vistos hasta la fecha de elaboración de este documento, lo que se presenta en las figuras 3.7-3.12 son los resultados de la matriz de confusión para una parte específica de los datos y la curva precisión usando la métrica de validación cruzada conforme cambiamos los parámetros clave de cada modelo.

En la tabla 3.1 se muestra un resumen de la precisión de cada modelo para cada variable a etiquetar NPK mediante la métrica de validación cruzada, nótese que AD significa árboles de decisión.

Tabla 3.1: Resumen de las precisiones obtenidas con validación cruzada para cada variable.

	AD	SVM	KNN
N	66.89 %	69.44 %	69.11 %
P	57 %	52.44 %	41.11 %
K	99.44 %	99.67 %	99.11 %

Así mismo en las tablas 3.2-3.7 se presentan los reportes de clasificación donde constan las métricas de precisión, recall y f1-score para las variables de P y K, estos valores se obtuvieron mediante la elección aleatoria de un grupo de entrenamiento y evaluación, luego estos grupos se usaron en los 3 modelos por lo que todas las tablas hacen referencia a un mismo grupo de entrenamiento y evaluación en el caso de P y así mismo para el caso de K.

Tabla 3.2: Reporte de clasificación para Arboles de Decisión con P.

	Precision	Recall	f1-score	Support
alto	0.29	1.00	0.45	39
bajo	0.61	0.64	0.62	44
medio	0.00	0.00	0.00	97
accuracy			0.37	180
macro avg	0.30	0.55	0.36	180
weighted av	0.21	0.37	0.25	180

Tabla 3.3: Reporte de clasificación para SVM con P.

	Precision	Recall	f1-score	Support
alto	0.00	0.00	0.00	39
bajo	0.25	1.00	0.40	44
medio	0.00	0.00	0.00	97
accuracy			0.24	180
macro avg	0.08	0.33	0.13	180
weighted av	0.06	0.24	0.10	180

Tabla 3.4: Reporte de clasificación para KNN con P.

	Precision	Recall	f1-score	Support
alto	0.37	0.41	0.39	39
bajo	0.57	0.55	0.56	44
medio	0.57	0.56	0.56	97
accuracy			0.52	180
macro avg	0.50	0.50	0.50	180
weighted av	0.53	0.52	0.52	180

Tabla 3.5: Reporte de clasificación para Árboles de Decisión con K.

	Precision	Recall	f1-score	Support
alto	0.88	1.00	0.94	22
bajo	1.00	0.98	0.99	133
medio	1.00	1.00	1.00	25
accuracy			0.98	180
macro avg	0.96	0.99	0.97	180
weighted av	0.99	0.98	0.98	180

Tabla 3.6: Reporte de clasificación para SVM con K.

	Precision	Recall	f1-score	Support
alto	0.96	1.00	0.98	22
bajo	1.00	0.99	1.00	133
medio	1.00	1.00	1.00	25
accuracy			0.99	180
macro avg	0.99	1.00	0.99	180
weighted av	0.99	0.99	0.99	180

Tabla 3.7: Reporte de clasificación para KNN con K.

	Precision	Recall	f1-score	Support
alto	1.00	1.00	1.00	22
bajo	1.00	1.00	1.00	133
medio	1.00	1.00	1.00	25
accuracy			1.00	180
macro avg	1.00	1.00	1.00	180
weighted av	1.00	1.00	1.00	180

En las tablas 3.5, 3.6 y 3.7 se evidencia el sesgo en el dataset para la variable K, por lo que se llegan a tener valores irreales en la práctica como lo es el caso del reporte de KNN donde todas las métricas apuntan a un 100 % de exactitud lo cual indica un problema con el modelo, sin embargo, en este caso podemos decir que no se trata de un problema de implementación sino más bien una limitación de los datos usados para evaluarlo.

3.2. Visualización de la plataforma en la nube

El sistema recomendador implementado en la plataforma de IoT se muestra en la figura 3.13, se debe aclarar que en este caso que debido a que los nodos todavía no están operativos la recomendación que se ve en la figura es generada con datos arbitrarios de temperatura, humedad, pH y precipitación para evaluar el funcionamiento del sistema.

Figura 3.13: Sistema recomendador implementado en Thingsboard.
Fuente: El Autor.

Los dashboards para la visualización de las variables ambientales ya se mostraron durante el desarrollo del trabajo, los cuales se observan en las figuras 2.18 y 2.19.

Capítulo 4

Conclusiones y Trabajos Futuros

Tal como se identificó en el estado del arte, el método de SVM resulta ser efectivo a la hora de clasificar variables en el agro, esto no es diferente en la clasificación de valores de NPK como se ha demostrado en el presente trabajo. Los resultados de SVM presentaron una gran precisión para clasificar los nutrientes del suelo y además se contrastó con otros métodos como KNN y árboles de decisión, donde los resultados mostraron que estos métodos también pueden resultar favorables a la hora de clasificar los valores de NPK. Por lo tanto, este trabajo sienta una base y motivación para probar la factibilidad de otros clasificadores de ML a la hora de clasificar nutrientes del suelo. tAREASgUEVARA10 La obtención de precisiones altas en el entrenamiento de modelos de ML se relaciona de forma directa con la calidad de los datos, en otras palabras, es necesario tener una cantidad sustancial de datos para capturar la variabilidad completa del sistema, de lo contrario puede haber relaciones entre las variables que no sean completamente capturadas por los modelos debido a las limitaciones de los datos. En el caso del presente trabajo se entrenaron los modelos con un dataset externo al grupo de investigación GITEL por lo que la calidad de los datos no fue la óptima para obtener resultados con precisiones mayores. Sin embargo, los resultados son prometedores y sientan una base sólida para trabajar con la clasificación de las variables del suelo NPK una vez se consolide la obtención de datos de los nodos en la florícola.

La implementación del dashboard con las variables meteorológicas de interés y el sistema recomendador pueden resultar bastante beneficiosas para los usuarios

de la florícola, pues constituyen la base para buenas prácticas de agricultura digital al poder monitorear las variables en tiempo real de los cultivos además de recibir recomendaciones de la IA de Gemini basada en los datos recolectados. Sin embargo, es importante destacar que la IA puede cometer errores y ser imprecisa con las recomendaciones, por lo que a la final se recomienda tomar los resultados con cautela y siempre consultar con un experto en el tema para tomar decisiones finales en cuanto al cuidado de los cultivos.

Se demostró que la implementación de un sistema de clasificación con aprendizaje de máquina y un modelo de IA generativa pueden complementarse para dar lugar a sistemas más completos en el campo de la agricultura digital y aportar valor a la hora de la toma de decisiones en los cultivos de rosas de la florícola flores del lago.

Recomendaciones

Es importante implementar un sistema de validación cruzada robusto para evaluar futuros modelos de clasificación. Además, se debe tomar muy en cuenta el balanceo de clases en los conjuntos de datos cuando el dataset presenta grandes sesgos debido a que esto puede influenciar negativamente en los modelos a la hora de clasificar las clases. Esto se debe a que los datos favorecen más a la clase mayoritaria, haciendo que el modelo clasifique mal a las clases minoritarias y esto puede no representar las características completas del sistema.

Asegurar la calidad de los datos tomados de los nodos resulta indispensable para asegurar resultados precisos, ya que, a pesar de que se pueden usar técnicas de limpieza de datos para mejorar la calidad de estos, esto solo se puede extender hasta cierto punto y es necesario que la calidad de los datos crudos también sea buena para que el proceso de limpieza sea efectivo, además se debe tomar una gran cantidad de datos para capturar de forma completa la aleatoriedad del sistema.

Trabajos Futuros

El análisis del presente trabajo se centró en modelos de ML para clasificar los valores de NPK en bajo, medio y alto, además se implementó un sistema recomendador con un modelo de IA generativa para NLP como lo es Gemini, sin embargo, se podría explorar en trabajos futuros otros métodos del campo de DP y otros modelos generativos como lo pueden ser chatGPT para evaluar la precisión de estos en contraste con los resultados de los modelos expuestos en el presente trabajo.

Se podría explorar otros modelos extensos de lenguaje para NLP como lo es Gemini, con esto se tendría más información para contrastar el rendimiento del recomendador implementado en el presente trabajo y así seleccionar el mejor a la hora de actuar como un experto en cultivos de rosas y mejorar la calidad de las recomendaciones.

Glosario

CNN Redes Neuronales Convolucionales – Convolutional Neural Network.

DP Aprendizaje Profundo – Deep Learning.

FAO Organización para la Alimentación y la Agricultura de las Naciones Unidas – Food and Agriculture Organization, is a United Nations agency that works to end wold hunger.

IA Inteligencia Artificial.

IoT Internet de las Cosas – Internet of Things.

KNN K Vecino mas proximo – K Nearest Neighbor.

ML Aprendizaje de Máquina – Machine Learning.

NLP Procesamiento de Lenguaje Natural – Natural Language Processing.

NPK Nitrogeno, Fosforo y Potasio.

SVM Máquinas de Soporte Vectorial – Support Vector Machine.

Referencias

[1] «Plant-Production-and-Protection.» Accedido en Enero 7, 2025. (2023), dirección: https://www.fao.org/plant-production-protection/about/en#:~:text=Every%20year%2C%20up%20to%2040,at%20least%20USD%2070%20billion..

[2] G. N. Santiago, A. J. P. Carcedo, M. E. Brown, A. P. Nejadhashemi, P. V. V. Prasad e I. A. Ciampitti, «Data integration dashboard for assessing and planning sustainable intensification agricultural interventions: a case study in Senegal,» *Frontiers in Sustainable Food Systems*, vol. 7, pág. 1 208 286, 2023. DOI: 10.3389/fsufs.2023.1208286.

[3] N. M. Mohd Nizar, E. Jahanshiri, A. S. Tharman et al., «Underutilised crops database for supporting agricultural diversification,» *Computers and Electronics in Agriculture*, vol. 180, pág. 105 920, 2021. DOI: 10.1016/j.compag.2020.105920.

[4] A. O. Adedoja, P. A. Owolawi, T. Mapayi y C. Tu, «Progress on Deep Learning Models for Plant Disease Detection: A Survey,» en *2021 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)*, 2021, págs. 1-9. DOI: 10.1109/icABCD51485.2021.9519323.

[5] B. Biswas y R. K. Yadav, «A Review of Convolutional Neural Network-based Approaches for Disease Detection in Plants,» en *Proceedings of the International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT 2023)*, IEEE, 2023, págs. 517-518, ISBN: 978-1-6654-7451-1. DOI: 10.1109/IDCIoT56793.2023.10053428.

[6] J. O. Ordoñez-Ordoñez, L. F. Guerrero-Vásquez, P. A. Chasi-Pesántez, D. P. Barros-Piedra, E. J. Coronel-González y K. C. Bustamante-Cacao, «LDA Algorithm for the Identification of Topics: A Case of Study in the Most Influential Twitter Accounts in Ecuador,» en *Proceedings of Sixth International Congress on Information and Communication*

Technology, X.-S. Yang, S. Sherratt, N. Dey y A. Joshi, eds., London: Springer, 2021, págs. 359-367. DOI: 10.1007/978-981-16-1781-2_33.

[7] Dirección de Estadísticas Agropecuarias y Ambientales, «Encuesta de Superficie y Producción Agropecuaria Continua (ESPAC),» INEC, Ecuador, inf. téc., 2024. dirección: %7Bhttps://www.ecuadorencifras.gob.ec/documentos/web-inec/Estadisticas_agropecuarias/espac/2023/Boletin_tecnico_ESPAC_2023.pdf%7D.

[8] S. e. a. Van der Burg, «Digitalization and agricultural knowledge and innovation systems,» *Innovation studies, science and technology studies, communication science, economics*, vol. 27, págs. 1-20, 2019, Available at: https://doi.org/10.1016/j.njas.2018.06.001.

[9] G. Ozdogan y A. Aktas, «The Current Situation of Digital Agriculture in Turkey,» *Journal of Economics, Finance and Accounting*, vol. 4, n.º 2, págs. 184-191, 2017. DOI: 10.17261/Pressacademia.2017.448.

[10] S. Ionitescu, A. Popescu y N.-L. Gudanescu, «Digitalization and agriculture - impact on human resources in the European Union and Romania,» *Scientific Papers Series Management, Economic Engineering in Agriculture and Rural Development*, vol. 23, n.º 3, págs. 361-371, 2023.

[11] C. Sharma, P. Pathak, A. Kumar et al., «Sustainable regenerative agriculture allied with digital agri-technologies and future perspectives for transforming Indian agriculture,» *Environment, Development and Sustainability*, 2024. DOI: 10.1007/s10668-024-05231-y. dirección: https://doi.org/10.1007/s10668-024-05231-y.

[12] C. Parra-López, «Technologies for Climate Change Adaptation and Mitigation in Agriculture,» *Computers and Electronics in Agriculture*, vol. 226, pág. 109 412, 2024, Available online 7 September 2024. DOI: 10.1016/j.compag.2024.109412.

[13] V. K. Shankarnarayan y H. Ramakrishna, «Paradigm change in Indian agricultural practices using big data: Challenges and opportunities from field to plate,» *Information Processing in Agriculture*, vol. 7, n.º 3, págs. 355-368, 2020. DOI: 10.1016/j.inpa.2020.01.001.

[14] «¿Qué es la inteligencia artificial o ia?» Accedido en Diciembre 15, 2024. (2024), dirección: https://cloud.google.com/learn/what-is-artificial-intelligence?h.

[15] «How Many Companies Use AI? (New Data).» Accedido en Enero 18, 2025. (2024), dirección: https://explodingtopics.com/blog/companies-using-ai.

[16] P. Hua, «Generative Models,» en *Neural Networks with TensorFlow and Keras : Training, Generative Models, and Reinforcement Learning*. Berkeley, CA: Apress, 2024, págs. 105-129, ISBN: 979-8-8688-1020-6. DOI: 10.1007/979-8-8688-1020-6_6. dirección: https://doi.org/10.1007/979-8-8688-1020-6_6.

[17] H. Assoudi, «NLP Essentials,» en *Natural Language Processing on Oracle Cloud Infrastructure: Building Transformer-Based NLP Solutions Using Oracle AI and Hugging Face*. Berkeley, CA: Apress, 2024, págs. 3-34, ISBN: 979-8-8688-1073-2. DOI: 10.1007/979-8-8688-1073-2_1. dirección: https://doi.org/10.1007/979-8-8688-1073-2_1.

[18] J. Campino, «Transformers NLP models in AI writing detection,» *Journal of Computers in Education*, vol. 1, págs. 1-29, 2023. DOI: 10.1007/s40979-023-00146-z.

[19] E. Shin y M. Ramanathan, «Evaluation of prompt engineering strategies for pharmacokinetic data analysis with the ChatGPT large language model,» *Journal of Pharmacokinetics and Pharmacodynamics*, vol. 51, n.º 2, págs. 101-108, 2024. DOI: 10.1007/s10928-024-09921-y.

[20] J. VanderPlas, *Python Data Science Handbook: Essential Tools for Working with Data*. O'Reilly Media, 2017, ISBN: 978-1-491-91205-8.

[21] Y. Jiang, X. Li, H. Luo, S. Yin y O. Kaynak, «Quo vadis artificial intelligence?» *Springer*, vol. 2, n.º 4, 2022. DOI: https://doi.org/10.1007/s44163-022-00022-8.

[22] «Decision tree: Structure, training and Python Code.» Accedido en Diciembre 27, 2024. (2024), dirección: https://www.insidealgorithms.com/blog/decision-tree.

[23] G. James, D. Witten, T. Hastie y R. Tibshirani, *An Introduction to Statistical Learning: with Applications in R* (Springer Texts in Statistics). New York, NY: Springer, 2013, vol. 112, ISBN: 978-1-4614-7137-0.

[24] O. Colliot, *Machine Learning for Brain Disorders*. Humana, 2023, ISBN: 9781071631959.

[25] «Plot classification boundaries with different SVM kernels.» Accedido en Diciembre 28, 2024. (2024), dirección: https://scikit-learn.org/1.5/auto_examples/svm/plot_svm_kernels.html.

[26] R. O. Duda, P. E. Hart y D. G. Stork, *Pattern Classification*. John Wiley & Sons, 2012, cap. 4, ISBN: 9780071154673. dirección: https://books.google.com/books?id=EoYBngEACAAJ.

[27] «What is the K-nearest neighbors algorithm?» Accedido en Diciembre 28, 2024. (2024), dirección: https://www.ibm.com/think/topics/knn.

[28] S. H. Javaheri, M. M. Sepehri y B. Teimourpour, «Chapter 6 - Response Modeling in Direct Marketing: A Data Mining-Based Approach for Target Selection,» en *Data Mining Applications with R*, Y. Zhao e Y. Cen, eds., Boston: Academic Press, 2014, págs. 153-180, ISBN: 978-0-12-411511-8. DOI: https://doi.org/10.1016/B978-0-12-411511-8.00006-2. dirección: https://www.sciencedirect.com/science/article/pii/B9780124115118000062.

[29] «Smote for imbalanced classification with python.» Accedido en Diciembre 28, 2024. (2024), dirección: https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/.

[30] S. Yadav y S. Shukla, «Analysis of k-Fold Cross-Validation over Hold-Out Validation on Colossal Datasets for Quality Classification,» en *2016 IEEE 6th International Conference on Advanced Computing (IACC)*, 2016, págs. 78-83. DOI: 10.1109/IACC.2016.25.

[31] G. M. Foody, «Challenges in the real world use of classification accuracy metrics: From recall and precision to the Matthews correlation coefficient,» *PLOS ONE*, vol. 18, n.º 10, págs. 1-27, oct. de 2023. DOI: 10.1371/journal.pone.0291908. dirección: https://doi.org/10.1371/journal.pone.0291908.

[32] U. Shruthi, V. Nagaveni y B. Raghavendra, «A Review on Machine Learning Classification Techniques for Plant Disease Detection,» en *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, 2019, págs. 281-284. DOI: 10.1109/ICACCS.2019.8728415.

[33] L. Klerkx, E. Jakku y P. Labarthe, «A review of social science on digital agriculture, smart farming and agriculture 4.0: New contributions and a future research agenda,» *NJAS - Wageningen Journal of Life Sciences*, vol. 90, pág. 100 315, 2019. DOI: 10.1016/j.njas.2019.100315.

[34] K. Silawarawet y T. Phurat, «Building a Dashboard Farm by Simulation with Node-Red,» en *2023 18th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP)*, 2023, págs. 1-6. DOI: 10.1109/iSAI-NLP60301.2023.10354673.

[35] A. Chergui y T. Kechadi, «Data analytics for crop management: a big data view,» *Journal of Big Data*, vol. 9, n.º 123, 2022. DOI: 10.1186/s40537-022-00668-2. dirección: https://doi.org/10.1186/s40537-022-00668-2.

[36] A. Sinha, *crop-dataset*, https://www.kaggle.com/datasets/aryan2003/crop-dataset, Dataset, 2022.