FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

TESIS

Análisis De Algoritmos De Aprendizaje Automático Para Detectar La Roya De Café

PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO DE SISTEMAS

Autor(es)

Bach. Gonzales Inoñan, Oscar Eduardo
ORCID: https://orcid.org/0000-0002-6172-8877

Bach. Lopez Cruz, Alex Fabian
ORCID: https://orcid.org/0000-0002-9170-4585

Asesor

Dr. Vásquez Leyva, Oliver
ORCID:-https://orcid.org/0000-0003-4425-0688

Línea de Investigación

Ciencias de la información como herramienta multidisciplinares y estratégicas en el contexto industrial y de organizaciones

Sublínea de Investigación

Informática y transformación digital en el contexto industrial y organizacional

Pimentel – Perú

2024

Análisis De Algoritmos De Aprendizaje Automático Para Detectar La Roya De Café

Aprobación del Jurado

MG. ARCILA DIAZ JUAN CARLOS

Presidente del Jurado de Tesis

MG. ENRIQUE ASENJO CARRANZA

Secretario del Jurado de Tesis

MG. HEBER IVAN MEJIA CABRERA

Vocal del Jurado de Tesis

REPORTE DE SIMILITUD TURNITIN

Lopez Cruz / Gonzales Inoñan

Análisis De Algoritmos De Aprendizaje Automático Para Detectar La Roya De Café

My Files

My Files

Universidad Señor de Sipan

Detalles del documento

Identificador de la entrega
trn:oid:::26396:409298100

Fecha de entrega
25 nov 2024, 8:29 a.m. GMT-5

Fecha de descarga
25 nov 2024, 8:32 a.m. GMT-5

Nombre de archivo
tesis-turnitin.docx

Tamaño de archivo
15.0 MB

34 Páginas

7,523 Palabras

40,125 Caracteres

11% Similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para ca...

Filtrado desde el informe

- Bibliografía
- Texto mencionado
- Coincidencias menores (menos de 8 palabras)

Fuentes principales

8% Fuentes de Internet
1% Publicaciones
5% Trabajos entregados (trabajos del estudiante)

Marcas de integridad

N.º de alertas de integridad para revisión

No se han detectado manipulaciones de texto sospechosas.

Los algoritmos de nuestro sistema analizan un documento en profundidad para buscar inconsistencias que permitirían distinguirlo de una entrega normal. Si advertimos algo extraño, lo marcamos como una alerta para que pueda revisarlo.

Una marca de alerta no es necesariamente un indicador de problemas. Sin embargo, recomendamos que preste atención y la revise.

Universidad Señor de Sipán

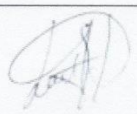
ANEXO 01: DECLARACIÓN JURADA DE ORIGINALIDAD

Quien(es) suscribe(n) la DECLARACIÓN JURADA, soy(somos) **Gonzales Inoñan Oscar Eduardo, Lopez Cruz Alex Fabian** Del Programa de Estudios de **Ingeniería de Sistemas** de la Universidad Señor de Sipán S.A.C, declaro (amos) bajo juramento que soy (somos) autor(es) del trabajo titulado:

ANÁLISIS DE ALGORITMOS DE APRENDIZAJE AUTOMÁTICO PARA DETECTAR LA ROYA DE CAFÉ

El texto de mi trabajo de investigación responde y respeta lo indicado en el Código de Ética del Comité Institucional de Ética en Investigación de la Universidad Señor de Sipán, conforme a los principios y lineamientos detallados en dicho documento, en relación con las citas y referencias bibliográficas, respetando el derecho de propiedad intelectual, por lo cual informo que la investigación cumple con ser inédito, original y auténtico.

En virtud de lo antes mencionado, firman:

Gonzales Inoñan, Oscar Eduardo	DNI: 71934306	
Lopez Cruz, Alex Fabian	DNI: 75463065	

Pimentel, 15 de diciembre de 2023.

Dedicatoria

A mi querido Padre Tito, cuya sabiduría y amor incondicional han sido la guía en cada paso de mi vida; a mi Madre Delci, por su apoyo constante y su fe inquebrantable en mis sueños; y a mi hermana Patricia, por su compañía y aliento en los momentos de desafío y celebración.

Este trabajo es un reflejo de su amor y sacrificio, y les dedico este trabajo con todo mi corazón.

Con amor y gratitud.

Oscar Gonzales

A mis padres, Alex y Marleni, por su amor incondicional, apoyo constante y por inspirarme a seguir mis sueños. Esta tesis es el reflejo de su inquebrantable fe en mí. A mi hermanos, primos y amigos, quienes han estado a mi lado apoyando y celebrando mis éxitos, como también en los momentos más difíciles. Este logro es fruto del esfuerzo y la dedicación colectiva de todos los involucrados. Expreso mi más sincero agradecimiento y dedico este trabajo a cada uno de ustedes.

Alex Lopez

Agradecimientos

Expresamos nuestro más sincero agradecimiento a los ilustres docentes de la Universidad Señor de Sipán, por su invaluable guía, paciencia y apoyo a lo largo de este proyecto. Sus conocimientos y consejos han sido fundamentales para el desarrollo de esta investigación. Agradecemos también a nuestros padres y familiares, por su amor, comprensión y apoyo incondicional, que han sido nuestra mayor fortaleza durante todo el proceso. A nuestros colegas y amigos, por sus ideas, apoyo moral y por compartir este viaje académico. Sin su valiosa ayuda, este proyecto no habría alcanzado el nivel de excelencia que hoy celebramos.

Los autores

Índice de contenidos

Dedicatoria 6

Agradecimientos 7

Resumen 11

Abstract 12

I. INTRODUCCIÓN 13

II. Materiales y Método 16

2.1. Materiales 16

2.2. Método 17

III. RESULTADOS Y DISCUSIÓN 35

3.1. Resultados 35

3.2. Discusión 41

IV. CONCLUSIONES Y RECOMENDACIONES 43

4.1. Conclusiones 43

4.2. Recomendaciones 44

REFERENCIAS 46

ANEXOS 50

Índice de Tablas

Tabla I: Lista de materiales 16
Tabla II: Características del Dataset 18
Tabla III: Técnicas para el aumento de datos 20
Tabla IV: Distribución del Dataset 21
Tabla V: Lista de los 10 mejores algoritmos 22
Tabla VI: Notaciones matemáticas 24
Tabla VII: Características 26
Tabla VIII: Comunicación entre capas 34
Tabla IX: Consumo de RAM y tiempo de entrenamiento de los modelos 35
Tabla X: Resultados de la exactitud de los algoritmos 38
Tabla XI: Resultados de precisión de los algoritmos 39
Tabla XII: Resultados del recall de los algoritmos 39
Tabla XIII: Resultados del F1 – score de los algoritmos 40

Índice de Figuras

Figura 1: Diagrama del método utilizado 17
Figura 2: Método técnico 17
Figura 3: Segmentación con OTSU 19
Figura 4: Segmentación con REMBG 19
Figura 5: Espacio de búsqueda de bayesiana 23
Figura 6: Histograma de color HSV 26
Figura 7: Tamaño del vector 26
Figura 8: Matriz GLCM 27
Figura 9: Arquitectura de Decision Tree 27
Figura 10: Arquitectura de Random Forest 28

Figura 11: Arquitectura de Support Vector Machine 29
Figura 12: Entrenamiento de SVM..... 30
Figura 13: Arquitectura de Custom Convolutional Neural Networks..... 30
Figura 14: Creación del modelo Custom CNN 31
Figura 15: Compilar modelo Custom CNN..... 31
Figura 16: Arquitectura del Sistema..... 34
Figura 17: Matriz de confusión de Decision Tree..... 36
Figura 18: Matriz de confusión de Random Forest 37
Figura 19: Matriz de confusión SVM..... 38
Figura 20: Matriz de confusión Custom CNN..... 38
Figura 21: Comparación de los algoritmos aplicados..... 41

Resumen

El objetivo de la investigación consiste en analizar algoritmos de aprendizaje automático para la detección de roya de café. En este contexto, se emplearon algoritmos supervisados, entre los cuales se incluyen Random Forest (RF), Decision Tree (DT), Support Vector Machine (SVM) y Custom Convolutional Neural Networks (CNN). Para llevar a cabo el entrenamiento de dichos algoritmos, se utilizó un conjunto de datos compuesto por 6260 imágenes de hojas de café, tanto saludables como no saludables. Es importante señalar que se implementó un proceso de preprocesamiento inicial que implicó la eliminación del fondo de las imágenes. Dentro de la metodología empleada, se asignó el 80% de las imágenes al conjunto de entrenamiento, mientras que el 20% restante se destinó a la validación del modelo. Asimismo, se documenta detalladamente el proceso de selección e implementación de cada algoritmo, acompañado de las correspondientes métricas de precisión, exactitud, sensibilidad y la puntuación F1. Los resultados obtenidos indican que el algoritmo Decision Tree se destaca como el más eficaz en la detección del patógeno en cuestión, alcanzando un rendimiento del 99%. En conclusión, este estudio demuestra de manera concluyente que los algoritmos de aprendizaje automático representan una herramienta sumamente prometedora para la detección de la roya del café.

Palabras Clave: Aprendizaje automático, Algoritmos de detección, Hemileia Vastratix, Coffee Rust.

Abstract

The objective of the research is to analyze machine learning algorithms for the detection of coffee rust. In this context, supervised algorithms were used, including Random Forests (RF), Decision Trees (DT), Support Vector Machines (SVM) and Custom Convolutional Neural Networks (CNN). To carry out the training of these algorithms, a dataset composed of 6260 images of coffee leaves, both healthy and unhealthy, was used. It is important to note that an initial preprocessing process was implemented that involved the removal of the background of the images. Within the methodology employed, 80% of the images were assigned to the training set, while the remaining 20% were used for model validation. The process of selection and implementation of each algorithm is also documented in detail, along with the corresponding metrics of precision, accuracy, sensitivity and F1 score. The results obtained indicate that the Decision Trees algorithm stands out as the most effective in detecting the pathogen in question, achieving 100% performance. In conclusion, this study conclusively demonstrates that machine learning algorithms represent an extremely promising tool for the detection of coffee rust.

Keywords: Machine learning, Detection algorithms, Hemileia Vastratix, Coffee Rust

I. INTRODUCCIÓN

El café representa uno de los cultivos más icónicos y de gran relevancia económica en múltiples naciones, incluyendo Perú, un país reconocido por la producción y variedades como el café arábica y robusta. Principalmente el café robusto es cultivado en las regiones amazónicas del país, además, es valorado tanto en el mercado local como internacional por su resistencia a climas adversos y su uso en mezclas comerciales. Sin embargo, este tipo de cultivo se ve expuesto a una enfermedad grave que es la roya, la cual vive en casi todas las plantaciones de café [1]. La enfermedad conocida como roya del café es originada por el hongo Hemileia vastratrix y es clasificada como una de las afecciones fitopatológicas con mayores impactos económicos a nivel global [2], comúnmente esta enfermedad debilita a las plantas, las hojas caen prematuramente y reduce la capacidad para producir café. Esta enfermedad es de gran importancia para los agricultores ya que están en constante lucha para disminuir esta plaga, la detección e identificación temprana de esta enfermedad puede ser crucial para superar grandes desafíos en industrias cafetaleras [3]. La forma más común de detectar esta enfermedad es caminando por el cultivo y realizando una inspección visual humana, lo que hace que existan errores y un tratamiento incorrecto de esta enfermedad puede dañar seriamente los cultivos [4]. Por eso, la detección de roya de café es un desafío importante para los agricultores y la industria cafetalera en general, ya que buscan soluciones innovadoras que permitan detectar de manera rápida, precisa y eficiente este mal de cultivos, lo que podría resultar a una toma de acciones y/o medidas preventivas de tal modo que se trate de contrarrestar esta enfermedad. Por ello, en el presente informe de investigación se centra en analizar y comparar algoritmos para la detección de roya de café basado en el aprendizaje automático.

El propósito de esta investigación es brindar un instrumento con una solución prometedora para la detección temprana y precisa de la roya del café. Para lograrlo, se ha trazado como objetivo principal, realizar un análisis de algoritmos de aprendizaje automático para detectar roya de café, teniendo en cuenta su rendimiento en términos de precisión, exactitud, sensibilidad y la medida F1, como también el tiempo de respuesta y el consumo de

RAM, ante ello, se plantea la siguiente pregunta de investigación ¿qué algoritmo de aprendizaje automático presenta mejor eficiencia para detectar la roya de café? Para lo cual se determinó la siguiente hipótesis, el algoritmo Random Forest presenta mejor eficiencia en la detección de roya de café. Así mismo se abordarán otros aspectos importantes, como el procesamiento de imágenes, selección de características relevantes y evaluación del sistema propuesto en condiciones del mundo real.

Por ende, el presente informe se justifica en que los algoritmos de aprendizaje automático aparecen como una solución prometedora para la detección temprana y precisa de la roya del café. Estos algoritmos tienen la capacidad única de analizar grandes volúmenes de datos, en este caso imágenes de hojas de café, y de discernir patrones y características asociadas con la presencia de enfermedades de manera objetiva y eficiente. Mediante el uso de métodos como el aprendizaje supervisado y el aprendizaje profundo, los modelos de aprendizaje automático pueden generalizar su conocimiento a partir de conjuntos de datos de entrenamiento y luego aplicar ese conocimiento de manera sistemática y efectiva en nuevas imágenes.

Por eso la tecnología ha hecho que las prácticas artesanales pasen a un primer plano por sus costes y tiempo; la idea principal de los agricultores es realizar las labores agrícolas con el mínimo tiempo, personal y precio posibles [5]. En respuesta a esta situación, se han creado diversos enfoques para identificar la presencia de la enfermedad conocida como roya en los cultivos de café. En el trabajo [6] se presenta un enfoque basado en arboles de decisión y evaluó un método validación cruzada con un total de 10 carpetas: Logistic Model Tree (LMT); J48; Extra Tree; REP-Tree; Functional Trees (FT); Random Tree (RT), y Random Forest (RF), donde da a conocer que Logistic Model Tree obtuvo una tasa más alta de predicción precisa de clases de infestación temprana, arrojando valores de medida F de 0.915 y 0.875, por lo que es un buen indicador de CLR temprano (2 a 5% roya) y etapas posteriores de CLR (20 a 40% roya). En [7] presentaron un enfoque de aprendizaje profundo para la identificación y clasificación de enfermedades del café utilizando redes neuronales convolucionales, donde combinan la arquitectura basada en Google Net y RESNET, las

cuales pueden capturar características más complejas y significativas de las imágenes de entrada, como formas, objetos y patrones que son importantes, logrando una puntuación de precisión del 99,08% en el conjunto de datos.

Por otro lado, Pereira et al. [8] hicieron uso de algoritmos genéticos, donde evaluaron la máscara del núcleo calculada y la segmentación de un conjunto de data, asimismo entrenaron solamente para distinguir solo entre las regiones normales y afectas de la hoja, donde obtuvieron como resultado un 0.90 de la media y la mediana con un total de 0,92 del coeficiente. En el trabajo [9] se presentó un método rápido y no destructivo que puede ser utilizada para monitorear el café, donde identifica la ocurrencia de infestación de la hoja de café mediante un nuevo índice de vegetación, utilizando imágenes multiespectrales del satélite Sentinel – 2 y la plataforma Google Earth Engine, además, aplicaron el método de aprendizaje automático mediante algoritmos supervisados entre ellos se aplicó Support Vector Machine con una exactitud del 81.8% y Random Forest tuve un ligero incremento con una exactitud del 89.5%. Mientras que el número de árboles para el clasificador Random Forest fue de 100. El modelo Support Vector Machine presentó un menor rendimiento que el algoritmo Random Forest, pero el rendimiento de ambos fue superior al 80%, por lo que el algoritmo Random forest presentó mayor rendimiento en la detección de infestación en la hoja de café. Por otro lado, [10] presentó un método robusto de clasificación y reconocimiento de enfermedades de las hojas de café basada en algoritmos de aprendizaje automático, la exactitud, precisión, recall se emplearon como indicadores de rendimiento para respaldar este método, es por ello, por lo que los algoritmos con mejor rendimiento son SVM y RF brindando porcentajes superiores a 97%, es por eso que son de suma importancia para la detección de enfermedades de café.

En ese sentido, el trabajo [11] propusieron un sistema de Alerta Temprana para la enfermedad de la roya del café basado en Códigos de Salida con Corrección de Errores y SVM, donde se considera excelente para aplicar algoritmos de aprendizaje binario a problemas de aprendizaje multiclase, mientras que SVM es el algoritmo más preciso para problemas de clasificación. En [12] presenta un sistema embebido basado en aprendizaje

automático con modelo basado en red neuronal profunda, donde la precisión de las hojas enfermas obtuvo 84% de precisión y de las hojas sanas un 92% de precisión, mientras que recall para las hojas que están sanas baja hasta el 41%, y el valor para las hojas enfermas de Roya del Cafeto sube hasta el 88%.

Es por eso por lo que se ha optado por el aprendizaje automático ya que brinda soluciones o patrones que permiten identificar los patógenos relacionados con el café, es por eso por lo que se opta por entrenar un modelo de aprendizaje automático capaz de facilitar la identificación con precisión de la roya de café, con el objetivo de brindar una herramienta innovadora eficaz y confiable a los agricultores.

II. Materiales y Método

2.1. Materiales

Tabla I: Lista de materiales

N°	Nombre	Descripción
1	Dataset Roya de Café	Consta de 6260 imágenes de hojas de cafeto, saludables y no saludables de la variedad robusta
2	Software	Visual Studio Code
3	Lenguaje Python	Cuenta con librerías especializadas en aprendizaje automático, además se destaca por la legibilidad de su código
4	Laptop y Pc	**LAPTOP** Procesador: Intel(R) Core (TM) i5-1035G1 CPU @1.00 GHz 1.19 GHz Gráficos Intel(R) UHD Graphics RAM: 8.00 GB Sistema operativo Windows 10 PRO **COMPUTADORA** Procesador: Intel(R) Core (TM) i5-10400F CPU @ 2.90GHz Gráficos NVIDIA GeForce GTX 1650 RAM: 24.00 GB Sistema operativo Windows 11 Home

Fuente: Elaboración propia.

2.2. Método

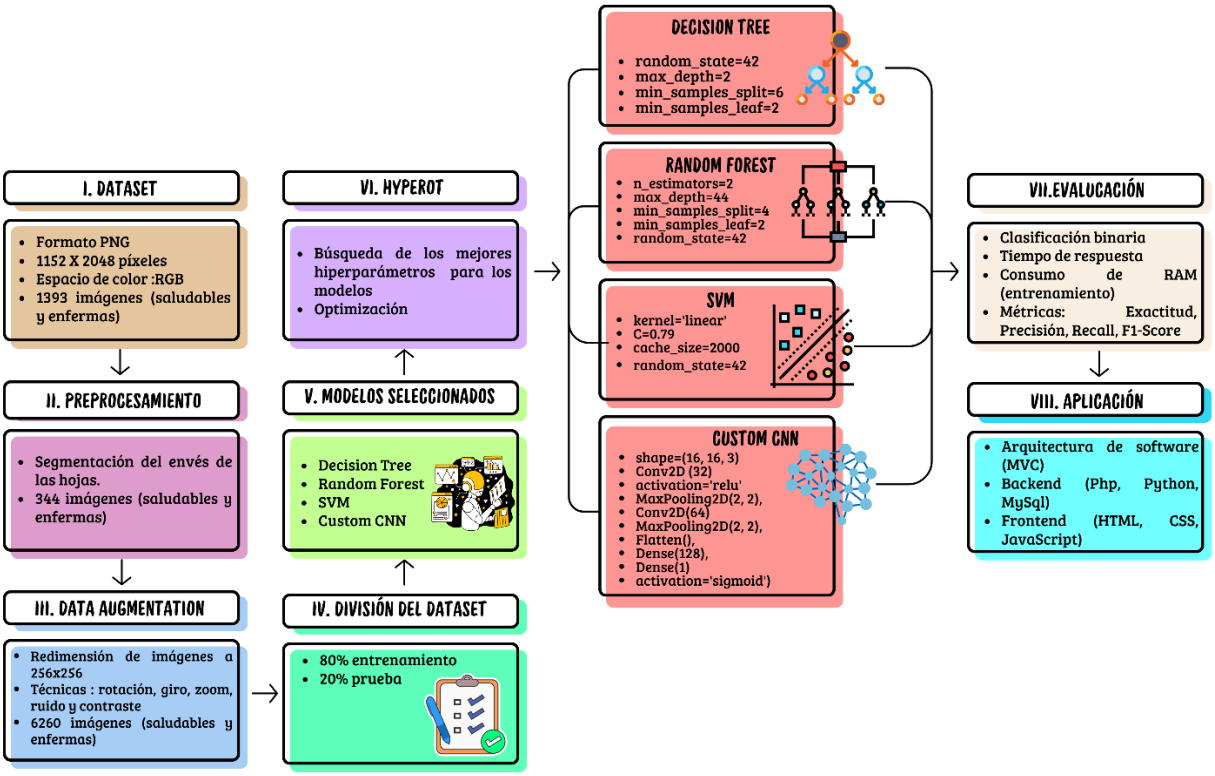


Figura 1: Diagrama del método utilizado

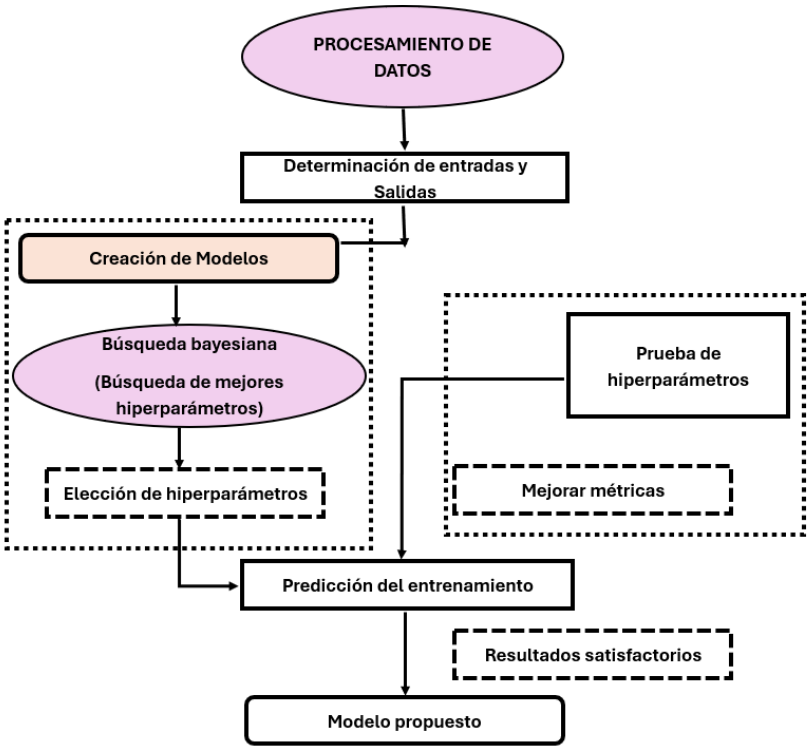


Figura 2: Método técnico

2.2.1. Dataset

Cómo punto de partida el dataset obtenido consta de imágenes de hojas de cafeto de la variedad Coffea canephora, también conocida como café robusto, las cuales fueron extraídas de un conjunto de 1393 imágenes tanto saludables como no saludables. Este conjunto de datos fue extraído de la investigación de [13], donde mencionan que la recolección se hizo utilizando una cámara de teléfono inteligente de 5 MP a una distancia de trabajo de 200-300 mm sin zoom. Las imágenes se tomaron diariamente en condiciones reales, como múltiples brillos de iluminación (días nublados, soleados y ventosos), fondos (otras plantas y maleza) y niveles de temperatura (niveles altos y bajos de humedad) para tener muestras reales y representativas de plantas de café. Por otro lado, las dimensiones de las fotografías son de 1152x2048 pixeles. Además, es importante mencionar que el café robusto se cultiva en muchas partes del mundo, también se cultiva en el Perú, especialmente en zonas cálidas y de baja altitud, por eso, este conjunto de datos se convierte en una pieza clave, ya que se usa información de una especie que se cultiva en el país.

Tabla II: Características del Dataset

Herramienta	Coordenadas	Tipo de café	Fuente de datos Lugar
Cámara de Teléfono Inteligente 5MP.	Latitud -0.834206 y Longitud -80.177159.	Café robusto	ESPAM MFL, Calceta, Manabí, Ecuador.

Fuente: Obtenido de [13].

2.2.2. Preprocesamiento

Para la segmentación de imágenes se hace uso de la segmentación basada en bordes, por eso se puso a prueba dos modelos de segmentación para ver cuál de los dos es mejor para este contexto, el primero se trata de OTSU, este modelo permite la segmentación de la imagen a través de umbrales, es decir, segmenta la imagen en diferentes regiones (ver imagen 03), este modelo de segmentación hace uso de la siguiente fórmula empleada por [14]: $W_0 = \sum_{(i,j,k)\in C_0} p_{ijk}$, $W_1 = \sum_{(i,j,k)\in C_1} p_{ijk}$.

Por otro lado, REMBG, es una herramienta que utiliza el modelo U-2-NET, el cual está

entrenado para segmentar el primer plano de la imagen (objeto importante de la imagen), donde únicamente se aprecia la hoja de café (ver imagen 04). Por lo tanto, se optó por usar REMBG, ya que permite de manera precisa enfocarse en el objeto principal de la imagen (hoja de café), mientras que OTSU te presenta las capas umbrales, lo que hace que pierda características de la imagen, es por ello, REMBG es el modelo adecuado para segmentar la imagen. Fórmula de REMBG: $i_{img\ (x,y)=I(x,y)*M(x,y)}$

Figura 3: Segmentación con OTSU

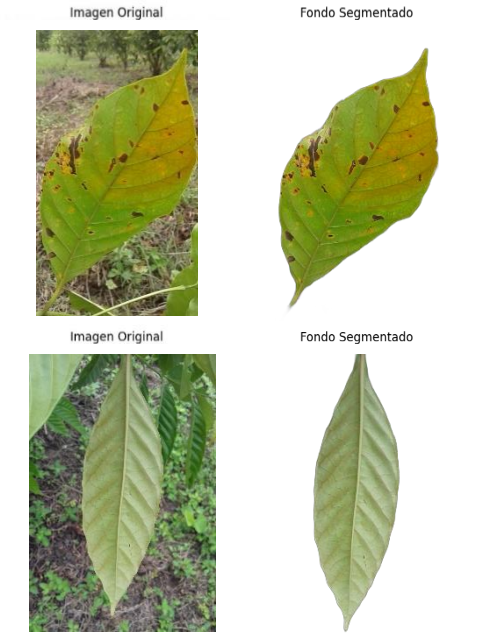


Figura 4: Segmentación con REMBG

Después, se realizó la selección de imágenes que presentan el envés de la hoja de café, tanto en su estado saludable como no saludable, dando como resultado la obtención de 164 imágenes relacionadas a hojas no saludables y 180 imágenes de hojas saludables. En total, se dispone de un conjunto de datos compuesto por 344 imágenes, las cuales se utilizarán a lo largo del desarrollo de esta investigación.

2.2.3. Data Augmentation

Continuando con el método, se realizó el aumento de datos, permitiendo generar más imágenes que se usarán para el entrenamiento a partir del conjunto de datos previamente construido, para el desarrollo se hace uso de cinco técnicas como rotación aleatoria, giro aleatorio, zoom aleatorio, ruido gaussiano y contraste aleatorio. Donde la rotación aleatoria permite rotar la imagen en un rango aleatorio entre -40 y 40 grados (ver tabla II), además, permite ajustar el tamaño de la imagen, de manera que no se recorte ningún píxel cuando procesa. Por otro lado, giro aleatorio permite voltear la imagen o no de forma horizontal, donde hace uso de decisiones como True y False, si el valor de decisión es verdadero, entonces realiza el giro, sino deja la imagen tal y como está. Además, zoom aleatorio permite acercar o alejar la imagen al azar entre valores de 70% - 130%, es decir, recorta o amplia la imagen para luego devolver dicha imagen en las dimensiones originales. Por otro lado, se aplica ruido gaussiano, ya que permite ajustar la desviación estándar del ruido de la imagen, donde hace una distribución media de 0 y una desviación estándar de 25, por último, el contraste aleatorio, los cuales van dentro de los rangos 0.6 a 1.4, permitiendo ajustar de manera uniforme el contraste de la imagen, dando como resultado una diferencia relativa de la intensidad entre el punto más claro de una imagen y el más oscuro.

Tabla III: Técnicas para el aumento de datos

IMAGEN ORIGINAL	TÉCNICA	IMAGEN PROCESADA	CARACTERÍSTICAS
	Rotación Aleatoria		Rotación entre -40 y 40 grados

	**Giro Aleatorio**		Giro horizontal
	Zoom Aleatorio		Zoom entre 70% - 130%
	Ruido Gaussiano		Media 0 y desviación estándar de 25
	Contraste Aleatorio		Ajustes entre 0.6 a 1.4

Fuente: Elaboración propia.

2.2.4. División del Dataset

Con respecto a la división del conjunto de datos, se optó por usar el 80% para el entrenamiento y el 20% para las pruebas. Donde se sabe que el 80% de los efectos provienen del 20% de las causas, es decir, los datos de prueba permiten que el entrenamiento sea exitoso. Por otro lado, se optó por esta distribución ya que permite a los modelos a que no memoricen los datos, sino que puedan aprender de ellos y así evitar un sobreajuste, además, permite mejorar la eficiencia y la toma de decisiones en los modelos a usar.

Tabla IV: Distribución del Dataset

Dataset	Entrenamiento	Prueba
100%	80%	20%
6260	5008	1252

Fuente: Elaboración propia.

2.2.5. Modelos seleccionados

Para la selección de los algoritmos de aprendizaje automático, se llevó a cabo una revisión sistemática de la literatura, explorando bases de datos como ScienceDirect, Scopus e IEEExplorer. Posteriormente, se llevó a cabo el proceso de selección de algoritmos, rigiéndose por criterios de inclusión, en este caso el criterio de inclusión es que el algoritmo se encuentre dentro de los 10 mejores algoritmos para la detección de roya (ver tabla II). Se optó por seleccionar estos algoritmos porque se han aplicado en investigaciones relacionadas con la detección de la roya y otras enfermedades como el minador de café, ojo de gallo, la araña roja y la mancha de hierro. Aunque algunos de estos algoritmos también detectan otras enfermedades, en este caso el enfoque está en detectar este mal de hoja.

Tabla V: Lista de los 10 mejores algoritmos

Artículos	Algoritmo	Precisión	Exactitud	Recall	F1 – Score
[15][16][17][18][19][20][21][22][23][24]	Convolution Neural Network	0.99	---	0.98	0.98
[15][18][25][26]	Support Vector Machine	0.97	0.96	0.95	0.95
[16]	K-nearest neighbors	0.90	---	0.90	0.91
[27]	Neural networks	0.95	---	---	---
[28]	XGBoost	0.96	0.86	---	0.74
[28]	CatBoost	0.69	0.86	---	0.74
[29][6][10]	Decision tree	0,82	0,82	0,82	0,83
[6][9][7]	Random Forest	0.97	---	0.97	0.97
[26]	K-Means	0.88	0.88	0.88	0.88
[30]	Artificial Neural Network	0.90	0.90	---	---

Fuente: Elaboración propia.

Es importante destacar que la selección se enfocó exclusivamente en algoritmos de aprendizaje automático, basándonos en las métricas de evaluación como la exactitud, precisión, recall y F1-Score, donde se hizo selección de los algoritmos como Random Forest, Custom Convolutional Neural Networks, Support Vector Machine, ya que estos tres algoritmos cuentan con buenos resultados ante los criterios mencionados. Sin embargo, también se optó por seleccionar Decision Tree, ya que es un modelo con una estructura

simple, además, consume pocos recursos computacionales y permite entrenar grandes volúmenes de datos en tiempos aceptables, por esa razón es que se optó por este algoritmo.

2.2.6. Búsqueda Bayesiana

Para a la implementación de hiperparámetros, se hace uso del método de búsqueda bayesiana que permite encontrar los mejores hiperparámetros para cada modelo, donde [31] menciona que la búsqueda bayesiana tiene un rendimiento consistentemente mejor que otros enfoques en términos de precisión y predicción, asimismo, permite crear espacios de búsqueda complejos, por otro lado, este método incluye algoritmos de optimización avanzada, los cuales son más eficientes que los métodos tradicionales, por eso, se optó por usar la técnica de búsqueda bayesiana, ya que permitió realizar un espacio de búsqueda de los hiperparámetros, donde crea un espacio de búsqueda definiendo los rangos e indica entre qué valores puede probar para mejorar el modelo. Posterior a ello, se toman los mejores resultados, los cuales se almacenan para posteriormente ser implementados en la creación y entrenamiento de los algoritmos.

```
space = {
    'max_depth': hp.choice('max_depth', range(1, 20)),
    'min_samples_split': hp.uniform('min_samples_split', 2, 10),
    'min_samples_leaf': hp.uniform('min_samples_leaf', 1, 5)
}
```

Figura 5: Espacio de búsqueda de bayesiana

2.2.7. Evaluación

También se tiene en cuenta la evaluación de los modelos, donde se calculan las métricas, haciendo uso de los valores de Verdaderos Positivos (VP), Verdaderos Negativos (VN), Falsos Positivos (FP) y Falsos Negativos (FN), permite evaluar el rendimiento del modelo de manera adecuada.

Exactitud

Permite conocer las predicciones correctas de las clases positivas como de las clases negativas, la exactitud proporciona una visión clara de la predicción correcta, clasificando hojas sanas como hojas no saludables.

$$\boldsymbol{Exactitud} = \frac{\boldsymbol{VP + VN}}{\boldsymbol{VP + VN + FP + FN}}$$

Precisión

Permite evaluar los valores de las predicciones positivas, la precisión proporciona una de predicción correcta de roya en hojas de café frente a las predicciones incorrectas, asegurando que no se esté sobre diagnosticando la presencia de la enfermedad.

$$\boldsymbol{Precisión} = \frac{\boldsymbol{VP}}{\boldsymbol{VP + FP}}$$

Recall

Permite identificar correctamente todas las instancias positivas, es decir determina la capacidad de hojas que tienen roya y que fueron correctamente identificadas.

$$\boldsymbol{Recall} = \frac{\boldsymbol{VP}}{\boldsymbol{VP + FN}}$$

F1-Score

Permite evaluar el rendimiento del modelo de clasificación, determina la capacidad para detectar todas las hojas con roya como la capacidad para minimizar los falsos positivos.

$$\boldsymbol{F1 - Score} = \boldsymbol{2} * \frac{\boldsymbol{Precisión * Recall}}{\boldsymbol{Precisión + Recall}}$$

Tabla VI: Notaciones matemáticas

Algoritmo	Notación	Donde
Arboles de Decisión	$Gini\ (t) = 1 - \sum_{i=1}^{C} p_i^2$	- $Gini\ (t) =$, es el índice de Gini cercano a 0 indica que el nodo esta puro, asi mismo C, es el numero total de clases en el conjunto de datos (hojas saludables y hojas no saludables) p_i, es la proporción de elementos en el nodo t que pertenecen a clase i.
Random Forest	$H(x) = nodo\{h_b(x) \mid b = 1, \dots, B\}$	- $h(x), x$ es una imagen de una hoja de café y $H(x)$, sería la clasificación final que el modelo hace sobre esa hoja, asi mismo $h_b(x)$, es la predicción realizada por el modelo, en este caso una hoja

		de café, y hace su propia clasificación, $b = 1, \dots, B,$ representa los arboles en el modelo, asi mismo entrena un conjunto de datos.
Custom CNN	$h\frac{(l)}{ij} = \sigma(\sum_{m,n} W\frac{(l)}{mn} * x\ \frac{(l-1)}{(i+m)(j+n)} + b^{(l)}$	- $h\frac{(l)}{ij}$, mapa de características que representa activaciones de la capa convolucional, σ función de activación que ayuda a identificar las areas de la imagen que tienen roya, $W\frac{(l)}{mn}$, es el filtro que a través de la imagen extrae características, como color y textura, $x\ \frac{(l-1)}{(i+m)(j+n)}$, filtro a través de la imagen y/o activación de la capa anterior, por otro lado $b^{(l)}$, es el sesgo que se agrega al resultado de la convolución.
Support Vector Machine	$K(x_i - x_j) = \exp(-\gamma \lvert\lvert x_i - x_j \rvert\rvert^2)$	- $K(x_i - x_j)$, vectores de características extraídas de las imágenes de hoja de café (color y textura), $\exp$, es la función exponencial, $\lvert\lvert x_i - x_j \rvert\rvert^2$, representa la diferencia entre los vectores de características de las dos muestras.

Fuente: Elaboración propia.

Otro aspecto a considerar es la extracción de características de las imágenes del conjunto de datos, en este caso se trabaja con el color y la textura de la imagen de hoja de café, la imagen que entra está en un espacio de color RGB que al ser procesada se transforma al espacio de color HSV, es decir que transforman los valores de los colores para representar de manera diferente la información de color de las imágenes de hoja de cafeto, donde se calcula el histograma y los valores pasan a convertirse en un vector de características, el cual es un vector de 13 valores. Además, se calculan las características de textura, donde la imagen pasa de RGB a escala de grises para sí poder calcular la matriz

GLCM para un ángulo de 45°.

Tabla VII: Características

FEATURES	RESULTADO	FÓRMULA
Contraste	2977.6533	$\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(i-j)^2P(i,j)$
Disimilitud	30.9333	$\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\lvert i-j\rvert^2P(i,j)$
Homogeneidad	0.4444	$\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\frac{P(i,j)}{1+(i-j)^2}$
ASM	0.1778	$\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}P(i-j)^2)$
Energía	0.4217	$\sqrt{ASM}=\sqrt{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}P(i-j)^2)}$

Fuente: Elaboración propia.

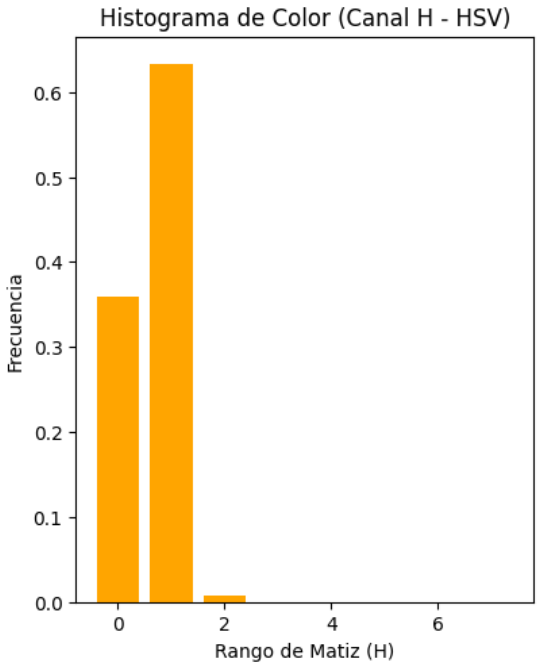


Figura 6: Histograma de color HSV

0.3594	0.6328	0.0078	0.0000	0.0000	0.0000	0.0000	0.0000	2977.6533	30.9333	0.4444	0.1778	0.4217

Figura 7: Tamaño del vector

94	4	1	...	0	0	0
4	0	0	...	0	0	0
2	0	0	...	0	0	0
			...			
0	0	0	...	0	0	0
0	0	0	...	0	0	0
0	0	0	...	0	0	0

Figura 8: Matriz GLCM

Decision Tree

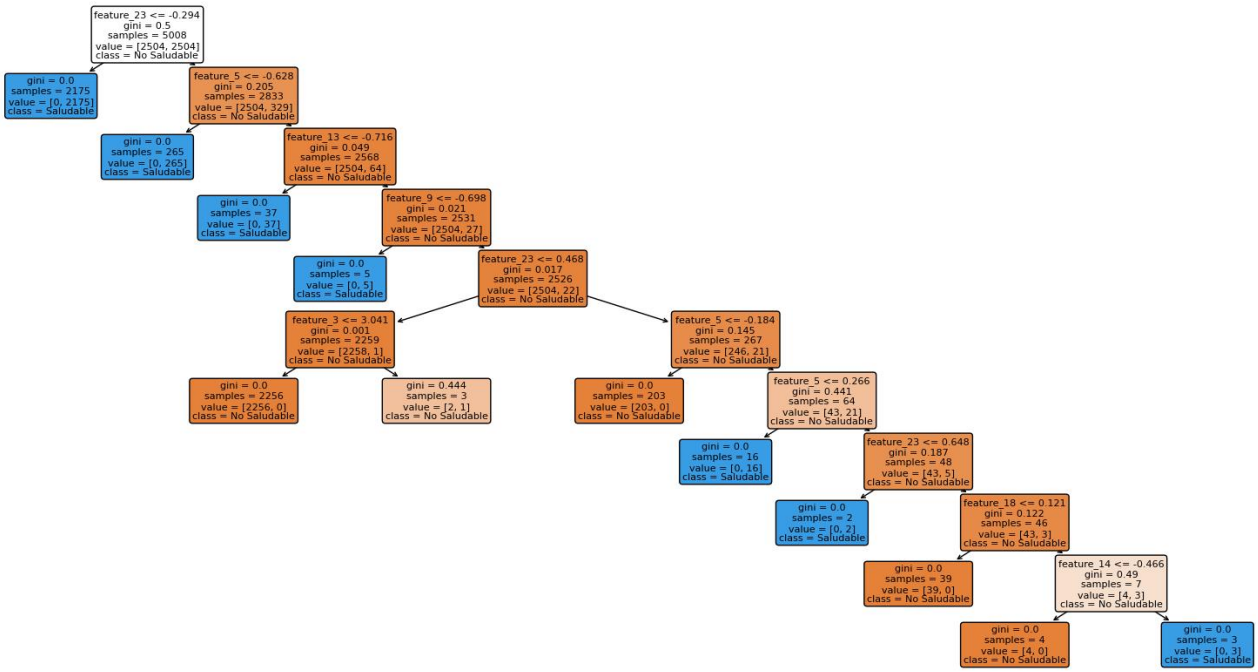


Figura 9: Arquitectura de Decision Tree

Primero, extraemos las características de color y textura de cada imagen, las cuales será almacenadas en una matriz 2D, de tal manera que dividimos el conjunto de datos en entrenamiento y prueba, garantizando que las clases estén balanceadas, se crea el modelo con los hiperparámetros encontrados con la búsqueda bayesiana como la profundidad, el número mínimo de muestras por nodo y el número mínimo de muestras en las hojas, en este

apartado se crea un espacio donde se define el rango de los hiperparámetros, para la profundidad del árbol se asignan valores enteros entre 5 y 49, es decir no se incluye el 50, mientras que el número de muestra requeridas esta entre el rango 4 y 10, esto sucede con los demás hiperparámetros, después de realizar la búsqueda bayesiana se crea el modelo y se entrena el modelo . Por último, se calcula el rendimiento del modelo, además se calculan las métricas, como precisión, exactitud, recall y F1-Score, también el informe de clasificación y la matriz de confusión para evaluar los aciertos y errores del modelo, para finalmente imprimir los resultados obtenidos.

Random Forest

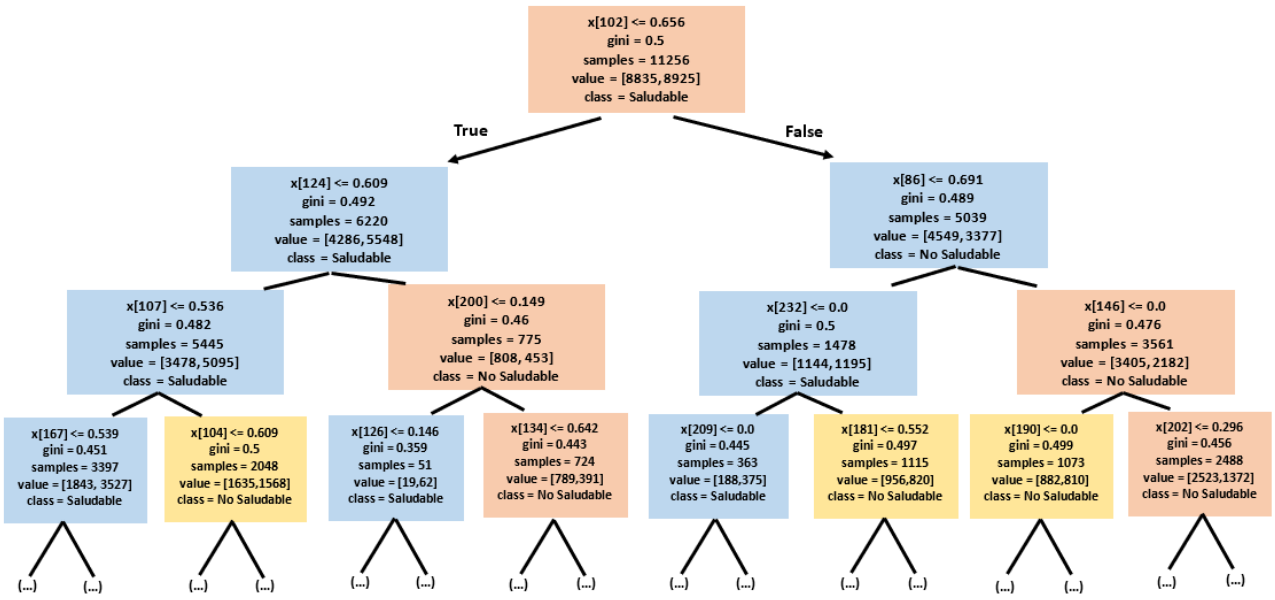


Figura 10: Arquitectura de Random Forest

Para el desarrollo de este modelo, primero se realiza la extracción de características de la hoja por cada imagen, en este caso se hace uso de la extracción de color y textura, las cuales se irán almacenando en una matriz 2D, después se crea el espacio de búsqueda bayesiana donde se asigna el rango de los valores que tendrán los hiperparámetros como la cantidad de árboles que oscila entre 50 y 300, permitiendo tener un modelo más robusto, también, tenemos los rangos de profundidad que están entre 5 y 21, así sucesivamente se asignan los demás valores para los demás hiperparámetros, posteriormente se hace uso de los mejor hiperparámetros encontrados, para luego dividir los datos en 80 y 20, en este punto

se crea el modelo con los mejores hiperparámetros, Asimismo, se llega a conocer el límite de profundidad de los árboles y el número de muestras para dividir un nodo, además del número que puede contener cada nodo hoja. Cada uno de estos árboles es entrenado con un subconjunto aleatorio de los datos y, para hacer predicciones, cada árbol vota por una clase (saludable o no saludable).

Después de entrenar el modelo, se realizan las predicciones sobre el conjunto de prueba, se calculan las métricas de evaluación como precisión, exactitud, recall y F1-Score, y visualizas la matriz de confusión para ver cuántas veces el modelo acertó o falló en las predicciones.

Support Vector Machine

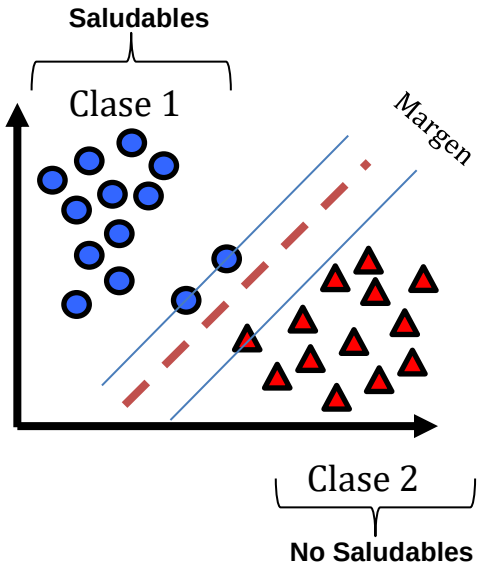


Figura 11: Arquitectura de Support Vector Machine

El modelo SVM usa métodos que permiten la extracción de características de las imágenes, en este caso se hace uso de la extracción de color y textura de la hoja, permitiendo guardar la información en matrices, posteriormente se configuró el clasificador con un kernel linear, el cual sirve para relacionar características complejas entre los datos. El modelo se inicializo con un valor de C, el cual se define en la búsqueda, lo que controla el equilibrio entre la precisión de la clasificación y gamma, ajusta automáticamente cada punto de soporte en el margen de decisión. Asimismo, se estableció un tamaño de cache de 2000 MB el cual

permite mejorar el rendimiento computacional y un sistema aleatorio de 42.

```
svm_classifier = SVC(
    kernel='linear',
    C=best_params['C'],
    cache_size=2000,
    random_state=42
)
```

Figura 12: Entrenamiento de SVM

Finalmente se hizo el cálculo de métricas que reflejan que tan eficaz es este modelo frente a la detección de roya mediante el procesamiento de imágenes, asimismo, se imprimen las métricas para posteriormente graficar la matriz de confusión.

Custom Convolutional Neural Networks

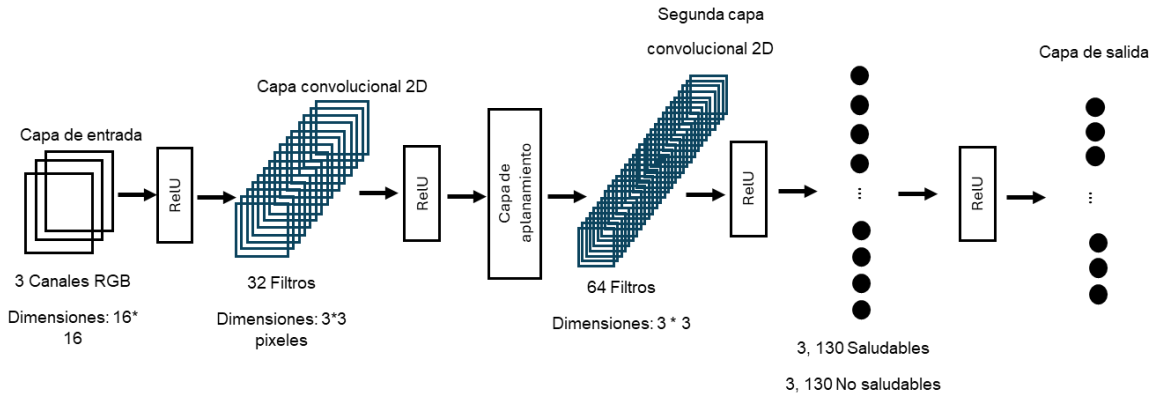


Figura 13: Arquitectura de Custom Convolutional Neural Networks

Para la normalización de los datos se hacen uso de los valores de los píxeles en un rango de 0 a 1, los cuales se dividen entre 255, permitiendo convertir los valores de intensidad, que van de 0 a 255, en valores pequeños y uniformes permitiendo mejorar la estabilidad y eficiencia del entrenamiento del modelo. Después, se define el modelo secuencial con Keras, para posteriormente definir la entrada de las imágenes con una resolución de 16x16 pixeles con 3 canales de color en este caso RGB, Luego, se crea una primera capa convolucional con 32 filtros cada uno con un tamaño (3,3), con activación ReLU, el cual permite introducir no linealidades al modelo, posteriormente se reducen las dimensiones de las imágenes usando con dimensiones 2x2, permitiendo extraer

características importantes, asimismo, se crea una capa de 64 filtros cada uno con un tamaño de 3x3 y de la misma manera con activación ReLU, seguido a ello se realiza la capa de Pooling donde nuevamente se redimensionan las imágenes para volver a extraer características, asimismo se crea una capa de aplanamiento con activación ReLU y transforman la salida 2D en un vector unidimensional, mientras que la capa densa con activación ReLU aprende combinaciones complejas. Finalmente, la capa de salida con activación sigmoide genera un valor entre 0 y 1 para clasificar imágenes como saludables y no saludables.

```
model = Sequential([
    Input(shape=(16, 16, 3)),
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Flatten(),
    Dense(best['dense_units'], activation='relu'),
    Dense(1, activation='sigmoid')
])
```

Figura 14: Creación del modelo Custom CNN

Luego se hace uso de 32 épocas y un tamaño de lote de 10, permitiendo entrenar el modelo, por otro lado, se compila el modelo creado haciendo uso del optimizador de Adam, el cual permite adaptarse y ajustar las tasas de aprendizaje automáticamente durante el entrenamiento, en este caso haremos uso de un valor establecido por la búsqueda bayseaiana, además se define la función de perdida y se define la métrica a evaluar.

```
model.compile(optimizer=Adam(learning_rate=best['learning_rate']),
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

Figura 15: Compilar modelo Custom CNN

Posterior a ello, se calculan las métricas y consumo de RAM durante el entrenamiento, además del tiempo de respuesta del modelo, para luego imprimir los resultados obtenidos.

2.2.8. Desarrollo de la aplicación

Posteriormente, se lleva a cabo el desarrollo de la aplicación para detectar la presencia de roya en hojas de café, en este punto se engloban los pasos que se siguieron para la construcción del software, a continuación, se detalla lo siguiente:

- **Requisitos**
  - **Alcance del proyecto**
    - **<u>Funciones principales</u>**
      - Carga de imágenes de hoja de café.
      - Procesamiento y análisis de las imágenes usando algoritmos de aprendizaje automático.
      - Mostrar resultados indicando la presencia o ausencia de roya de café.
    - **<u>Limitaciones</u>**
      - El sistema solo analiza imágenes de hojas de café y no de otro tipo.
  - **Recursos**
      - Entorno de desarrollo (IDE, librerías, base de datos).
      - Herramientas de análisis de imágenes (Python, OpenCV, Framework, Flask para integrar el modelo, entre otras dependencias y/o librerías).
      - Servidor para despliegue de aplicación.
- **ANÁLISIS DE REQUERIMIENTOS**
  - **Requerimientos funcionales**
    - ✓ El sistema debe permitir al usuario cargar imágenes de hojas de café en formato jpg, png y jgeg.
    - ✓ El sistema debe procesar la imagen cargada y analizarla usando el modelo (Decisión Tree), identificando la presencia o ausencia de roya.
    - ✓ El sistema presenta al usuario los resultados de la detección, indicando si la hoja tiene signos de roya o está en un estado saludable.
    - ✓ El sistema debe permitir el registro y autenticación de usuarios.

- ✓ Para guardar las imágenes se debe de hacer una comunicación entre sistemas y cuando un usuario carga una imagen, estas se guardan en una carpeta destinada para ello.

- **Requerimientos no funcionales**

  - ✓ **Usabilidad**

- La interfaz debe ser intuitiva, fácil de manejar y especialmente para usuarios sin conocimientos.
- Los tiempos de respuesta del sistema para la carga y procesamiento de imágenes no debe exceder los 5 segundos.

  - ✓ **Rendimiento**

- El tiempo de todo el proceso para cargar imagen no debe superar los 3 – 5 segundos para efectos de eficiencia.

  - ✓ **Escalabilidad**

- El sistema debe estar diseñado para soportar una mayor carga de usuarios en el futuro, de esta manera se garantiza y permite la integración de nuevos modelos de detección de roya de café.

  - ✓ **Seguridad**

- Los datos personales de los usuarios que utilizan el sistema deben de estar protegidos mediante cifrado y ciertas medidas de autenticación.

  - ✓ **Compatibilidad**

- El sistema debe ser accesible desde diferentes navegadores llámese (Chrome, Mozilla, Brave etc.).
- El sistema debe tener compatibilidad con disipativos móviles, asegurando una buena experiencia con el usuario final.

  - ✓ **Mantenibilidad**

- El sistema debe de estar diseñado de forma modular, facilitando el reemplazo o actualización de algunos componentes.

Después de eso, se procedió con el desarrollo de una aplicación web diseñada para abordar las necesidades de los agricultores en nuestro país. Este sistema, creado mediante tecnologías web, ofrece una interfaz intuitiva que permite a los usuarios cargar imágenes de sus plantaciones de café.

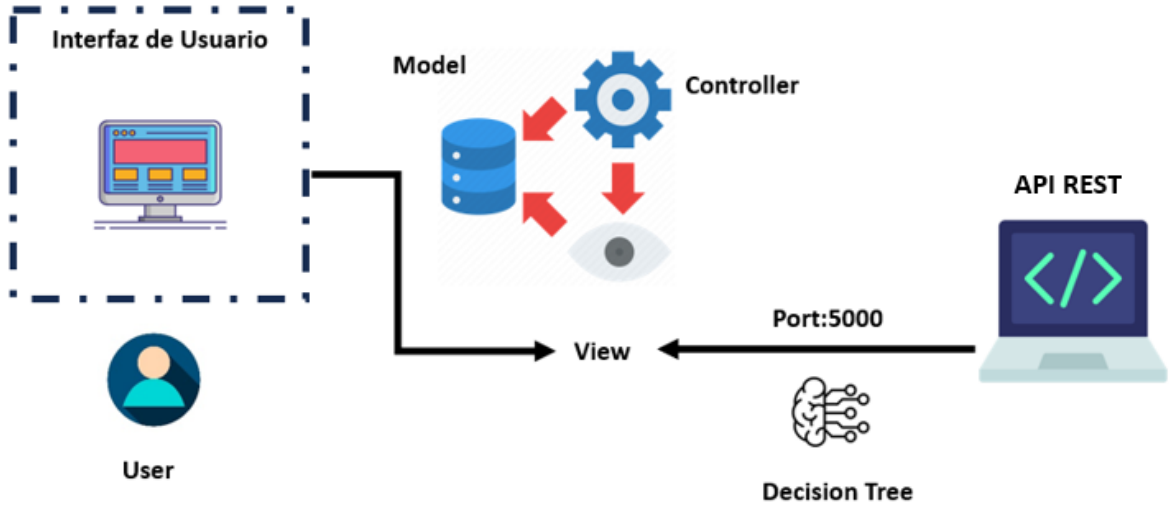


Figura 16: Arquitectura del Sistema

Para este proyecto de investigación, se eligió desarrollar una aplicación web con el propósito principal de detectar la enfermedad de la roya del café. La elección del framework Flask se basa en su ligereza y versatilidad para el desarrollo web utilizando el lenguaje de programación Python. Asimismo, se hizo uso de la arquitectura MVC, la cual representa la comunicación entre componente (ver tabla VIII). Donde el modelo (M) se encarga de gestionar y/o organizar la lógica de datos, el cual accede a nuestra base de datos mediante consultas, vista (V), es la parte frontend y/o interfaz, encargada de mostrar información para el usuario final permitiendo iteración, controlador (C), actúa como intermediario entre la vista y el modelo, procesando información.

Tabla VIII: Comunicación entre capas

Elemento	Responsabilidad	Propiedades
Usuarios	Los usuarios son consumidores de la API REST.	- Peticiones de tipo GET y tipo POST.
Base de datos	Permite realizar consultas	- Sistema para detección de roya de café (SPDRC)

API REST	Permiten intercambiar información de manera eficiente y segura.	- Peticiones de tipo HTTP - Framework Flask (Python)
Modelo DT	Permite detectar si una imagen tiene roya o no.	- Etiquetado
Interfaz de Usuario	Muestra la vista para la detección de roya de café.	- Interacción usuario y web.

Fuente: Elaboración propia.

Consecuentemente, la aplicación web fue desarrollada usando tecnologías como el framework Flask, lenguaje PHP, Python, los cuales corresponden al backend, mientras que para el frontend se hizo uso de HTML, CSS, JavaScript. La aplicación para la detección de roya de café está diseñada para ser intuitiva y fácil de usar, además cuenta con un menú destinado a la detección de la roya de café. Los usuarios pueden cargar imágenes de las hojas, ya sean saludables o no saludables. Una vez que se cargan las imágenes, la aplicación procesa las imágenes y proporciona un mensaje claro que indica si la hoja es positiva para roya o no. Finalmente, se han implementado medidas de seguridad para proteger la información de los usuarios y garantizar que los datos sensibles estén resguardados adecuadamente.

III. RESULTADOS Y DISCUSIÓN

3.1. Resultados

Tabla IX: Consumo de RAM y tiempo de entrenamiento de los modelos

Modelo aplicado	Consumo de RAM	Tiempo de respuesta
Decision Tree	31.02 MB	1.18 segundos
Random Forest	237.07 MB	25.02 segundos
Support Vector Machine	38 MB	257.36 segundos
Custom Convolutional Neural Networks	21.43 MB	36.42 segundos

Fuente: Elaboración propia.

La tabla contiene un resumen exhaustivo del desempeño de diversos modelos aplicados a la problemática de la detección de roya en imágenes digitales vinculadas al cultivo

de café. Cada fila de la tabla corresponde a un modelo particular, y se proporcionan tres métricas fundamentales para la evaluación del rendimiento y eficiencia de cada modelo en el contexto de la detección de roya en las imágenes: consumo de RAM y tiempo de respuesta. Estas métricas revisten una importancia crucial al evaluar y comparar los modelos en términos de su capacidad computacional y velocidad de respuesta. La consideración de tales factores resulta esencial al seleccionar el modelo más idóneo para la aplicación específica de detección de roya en imágenes digitales asociadas al cultivo de café.

Decision Tree

La figura 17 presenta la matriz de confusión de Decision Tree, se puede apreciar que 622 casos de verdaderos positivos (1,1) que fueron correctamente clasificadas como no saludables, por otro lado, 626 casos de verdaderos negativos (0,0) fueron correctamente clasificadas como saludables, así también tenemos 0 casos falsos positivos y 4 casos de falsos negativos (1,0).

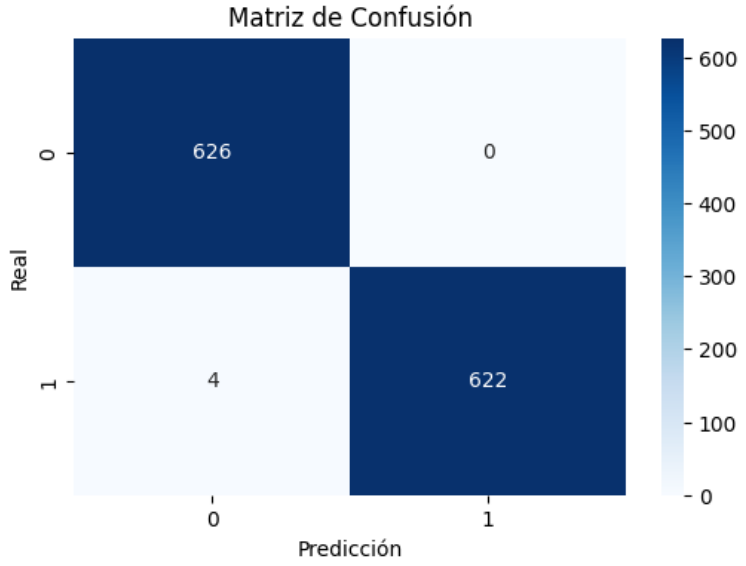


Figura 17: Matriz de confusión de Decision Tree

Random Forest

La figura 18 presenta la matriz de confusión de Random Forest, se puede apreciar que 492 casos de verdaderos positivos (1,1) que fueron correctamente clasificadas como no saludables, por otro lado, 480 casos de verdaderos negativos (0,0) fueron correctamente clasificadas como saludables, así también tenemos 115 casos falsos positivos y 164 casos

de falsos negativos (1,0).

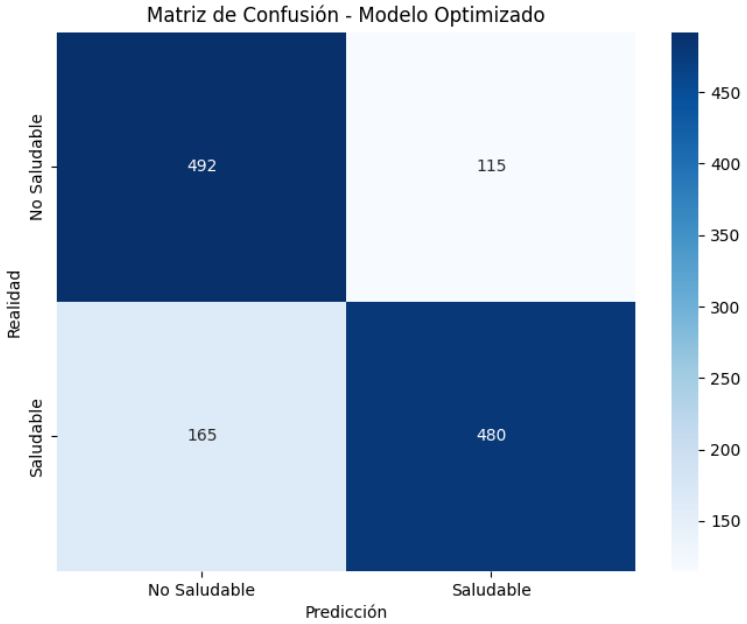


Figura 18: Matriz de confusión de Random Forest

Support Vector Machine

La figura 19 presenta la matriz de confusión de SVM, se puede apreciar que 442 casos de verdaderos positivos (1,1) que fueron correctamente clasificadas como no saludables, por otro lado, 385 casos de verdaderos negativos (0,0) fueron correctamente clasificadas como saludables, así también tenemos 222 casos falsos positivos y 203 casos de falsos negativos (1,0).

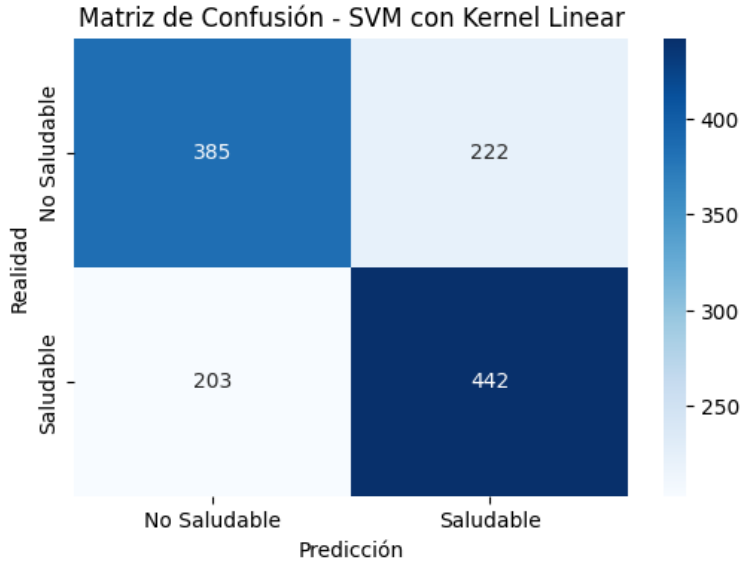


Figura 19: Matriz de confusión SVM

Custom Convolutional Neural Networks

La figura 20 presenta la matriz de confusión de Custom CNN, se puede apreciar que 620 casos de verdaderos positivos (1,1) que fueron correctamente clasificadas como no saludables, por otro lado, 554 casos de verdaderos negativos (0,0) fueron correctamente clasificadas como saludables, así también tenemos 53 casos falsos positivos y 25 casos de falsos negativos (1,0).

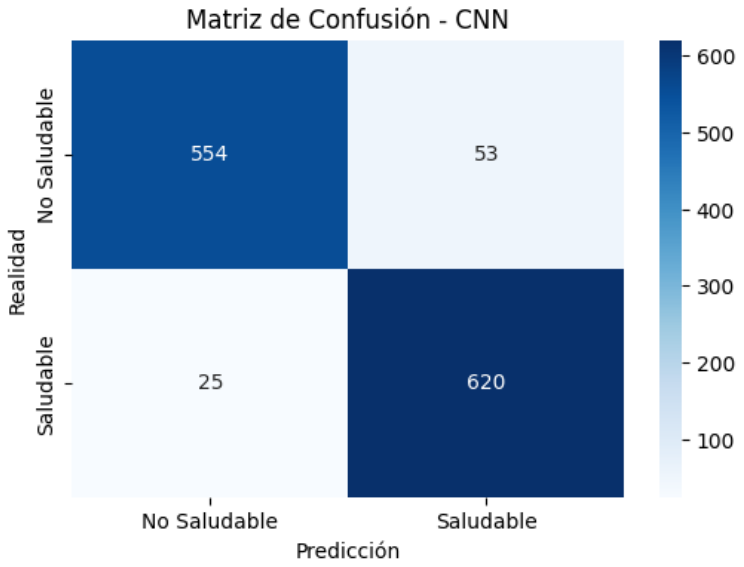


Figura 20: Matriz de confusión Custom CNN

Tabla X: Resultados de la exactitud de los algoritmos

Algoritmo	Exactitud
Decision Tree	99.00%
Random Forest	77.00%
Support Vector Machine	66.00%
Custom Convolutional Neural Networks	93.00%

Fuente: Elaboración propia.

Los resultados emanaron de la comparativa entre cuatro algoritmos de clasificación, a saber: son Random Forest, Support Vector Machine, Custom Convolutional Neural Networks y Decision Tree. Estos modelos fueron implementados con el propósito de diagnosticar la presencia de la roya en plantaciones de café. En un análisis inicial, se evaluó

la métrica de exactitud de cada algoritmo. De manera destacada, el algoritmo de Decisión Tree exhibió un desempeño óptimo al alcanzar una exactitud del 99%, posicionándose como el más preciso en el diagnóstico de la roya en plantaciones de café. Asimismo, Random Forest exhibió un rendimiento más modesto, con una exactitud del 77%, mientras que el algoritmo Support Vector Machine registró una precisión muy inferior, con un 66%. Por otro lado, el algoritmo Custom CNN demostró un 93% de exactitud.

Tabla XI: Resultados de precisión de los algoritmos

Algoritmo	Precisión
Decision Tree	99.00%
Random Forest	80.00%
Support Vector Machine	66.00%
Custom Convolutional Neural Networks	92.00%

Fuente: Elaboración propia.

Por otro lado, la tabla presenta tasas de precisión para diversos algoritmos de aprendizaje automático aplicados a la detección de roya mediante imágenes digitales. Los Decision Tree lideran con un impresionante 99%, evidenciando su eficacia para modelar relaciones complejas en los patrones visuales asociados con la roya. A pesar de su menor precisión del 80%, Random Forest ofrecen robustez y resistencia al sobreajuste, características valiosas en un contexto de detección de enfermedades en plantas. Las Support Vector Machine con 66% demuestran su eficacia en espacios de alta dimensión, mientras que las Custom Convolutional Neural Networks con 92%, las cuales resaltan su capacidad para abordar tareas visuales complejas.

Tabla XII: Resultados del recall de los algoritmos

Algoritmo	Recall
Decision Tree	99.00%
Random Forest	74.00%
Support Vector Machine	68.00%
Custom Convolutional Neural Networks	96.00%

Fuente: Elaboración propia.

La presente tabla (ver tabla XII) presenta tasas de recall para diversos algoritmos de aprendizaje automático aplicados en la detección de roya mediante análisis de imágenes digitales. Destaca que los Decision Tree lideran con un recall del 99%, indicando una capacidad integral para identificar de manera exhaustiva los casos positivos de roya. Por su parte, los Random Forest, con un recall del 74%, mantienen un nivel significativo de sensibilidad, sugiriendo robustez en la identificación de la enfermedad en plantas. Las Support Vector Machine 68% exhiben una buena capacidad para recuperar casos positivos en entornos de alta dimensión, mientras que las Custom Convolutional Neural Networks mostró un comportamiento del 96%.

Tabla XIII: Resultados del F1 – score de los algoritmos

Algoritmo	F1 – Score
Decision Tree	100.00%
Random Forest	77.00%
Support Vector Machine	67.00%
Convolutional Neural Networks	94.00%

Fuente: Elaboración propia.

La tabla presenta puntuaciones F1 (Ver tabla XIII) para diferentes algoritmos de aprendizaje automático en la tarea de detección de roya mediante imágenes digitales. Los Decision Tree lideran con un F1-Score del 99%, indicando un equilibrio excepcional entre precisión y recall en la identificación de casos positivos de roya. El algoritmo Random Forest, con un F1-Score del 77%, mantienen un buen equilibrio entre precisión y recall, lo que sugiere una capacidad robusta en la detección de la enfermedad en plantas. El algoritmo de Support Vector Machine con un 67%, estas muestran un rendimiento sólido al considerar ambas métricas, especialmente en entornos de alta dimensión. Las Custom Convolutional Neural Networks con 94%, estas se destacan por su capacidad inherente para comprender patrones visuales complejos, lo que puede ser crucial en tareas de detección de imágenes. Al seleccionar un algoritmo, es esencial considerar el F1-Score como una métrica integral que

pondera precisión y recall, proporcionando una evaluación holística del rendimiento del modelo en la detección de roya en imágenes digitales.

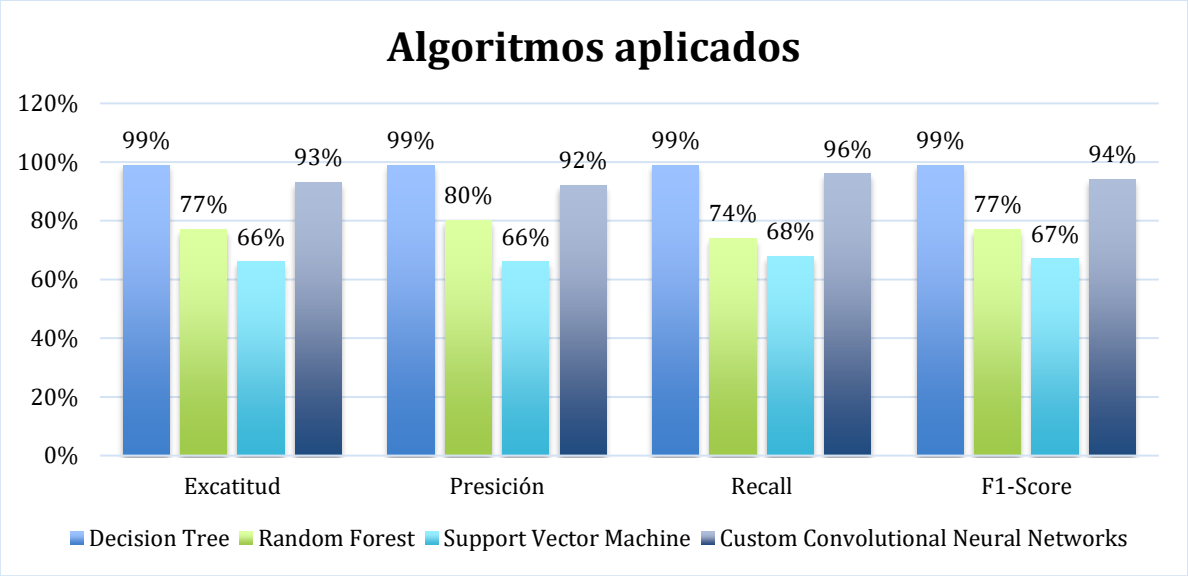


Figura 21: Comparación de los algoritmos aplicados

En resumen, podemos apreciar el rendimiento de diferentes modelos aplicados a la detección de roya de café mediante imágenes digitales en un conjunto de datos de 6260 imágenes. Se destaca que los dos mejores algoritmos en términos de métricas son Decision Tree y Random Forest. Se evidencia que Decision Tree exhibe un impresionante 99% de exactitud, precisión, recall y f1-score, indicando una capacidad casi perfecta de clasificación. Por otro lado, Random Forest demuestra un rendimiento equilibrado con un 77% de exactitud, precisión 80%, recall 74% y un f1-score del 77%. También se destacan otros algoritmos como Support Vector Machine y Convolutional Neural Networks tal y como se evidencia en la figura 20, pero los dos primeros algoritmos se destacan por su capacidad para identificar la roya del café con alta precisión y exactitud, siendo Decision Tree superior en todos los términos.

3.2. Discusión

El algoritmo de Decision Tree es una opción eficaz para diagnosticar la presencia de roya en las hojas de café, según los resultados obtenidos, evidenciando un rendimiento superior en todas las métricas evaluadas. Por lo tanto, el algoritmo de Random Forest exhibió un desempeño ligeramente inferior, donde la investigación de [7], dan a conocer que el algoritmo de Decision Tree, tiene un 82% de exactitud y precisión, mientras que el resultado

de esta investigación es muy optimo ya que supera el resultado del autor mencionado, donde lidera con un 99% de exactitud y de precisión, siendo el más optimo en la detección de roya de café, además, la indagación realizada [9] emplea la metodología de Random Forest, cuyos resultados sobresalen al alcanzar un 83% de exactitud en la detección de la infestación. No obstante, comparando con estos hallazgos, en el presente estudio, se observa que el desempeño del algoritmo de Random Forest con 77, cuenta con una ligera disminución y esto posiblemente sea por la cantidad del conjunto de datos y la división para el entrenamiento. Por otro lado, la investigación de [10], muestra que utilizó el algoritmo de Support Vector Machine y Random Forest en la clasificación y reconocimiento de enfermades de la hoja de café basados en algoritmos de aprendizaje automático, donde obtuvo una exactitud del 100%, siendo superior al resultado obtenido en esta investigación, donde Support Vector Machine obtuvo un 66% y Random Forest un 77% de exactitud, cabe resaltar que existen diversas diferencias con respecto a los datos utilizados. Los investigadores previamente mencionados utilizaron un dataset para entrenar y probar los modelos trabajados, contando con un dataset de 31518 imágenes de hojas de café, las cuales están divididas en cuatro clases (Phoma, Cercospora, Rust y saludables), los cuales fueron divididos en 60% para el entrenamiento, 20% para la validación y el 20% restante para realizar pruebas. Por el contrario, en esta investigación se contó con un dataset de 6260 imágenes relacionadas a hojas de café (saludables y no saludables), donde el 80% fue para entrenamiento y el 20% para la prueba.

Además, la presente investigación se distingue de otros estudios debido a la incorporación de Hyperopt, una herramienta de búsqueda bayesiana y optimización, permitiendo conocer los mejores hiperparámetros, además, hacemos se hace uso de la extracción de características como el color y textura de las imágenes de hojas de café. Por otro lado, se evidencia el consumo de RAM y tiempo de ejecución de cada algoritmo en tiempo real. Mientras que los trabajos referenciados en esta investigación utilizan valores predeterminados de hiperparámetros o realizan ajustes manuales. Por otro lado, estos ajustes permiten mejorar el rendimiento de los modelos empleados, logrando resultados respetables para detectar la roya de café.

Asimismo, los indicadores evaluativos revelaron que el algoritmo de Decision Tree demostró un rendimiento óptimo, logrando una exactitud y una precisión del 99% en la detección de roya en las hojas de café.

Por último, en la fase inicial de nuestra investigación se formuló la hipótesis de que el algoritmo de Random Forest sería el más eficaz para la detección de roya de café. Sin embargo, los resultados obtenidos presentados en este estudio no respaldan esta afirmación. Por el contrario, este algoritmo no logró los resultados deseados, mientras que el algoritmo de Decision Tree alcanzó un 99% en todas sus métricas. Estos hallazgos desafían la suposición inicial de que Random Forest sería superior en términos de rendimiento. La disparidad en los resultados sugiere que la elección del algoritmo puede no ser tan clara como se pensaba inicialmente, y destaca la importancia de una evaluación más profunda de los factores que podrían influir en el desempeño de los algoritmos en la detección de roya de café en nuestro contexto específico.

IV. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

- La rigurosa selección y preparación del dataset, compuesto por 6260 imágenes de hojas de café, constituye una base sólida para asegurar la validez y confiabilidad de los resultados obtenidos en esta investigación de aprendizaje automático. Este cuidadoso proceso garantiza la calidad y consistencia de los datos, aspectos esenciales para alcanzar el éxito en este estudio académico.
- La selección de los algoritmos fue un componente clave en esta investigación, fundamentada en una revisión sistemática de la literatura especializada en la detección de roya de café, la cual permitió adquirir una comprensión profunda de los algoritmos más adecuados para enfrentar este problema, además de identificar las métricas de evaluación más relevantes para su análisis y se logró una selección estratégica de los algoritmos, mejorando tanto la precisión como la fiabilidad de los resultados obtenidos.

- En la implementación de los algoritmos de aprendizaje automático seleccionados, Decision Tree destacó por su alta eficacia en la detección de la roya en hojas de café, alcanzando una exactitud y precisión del 99%. Aunque el rendimiento del Random Forest fue más bajo, con una exactitud del 83% y una precisión del 82%, sigue demostrando ser una opción valiosa en la clasificación de datos complejos. Estos resultados ponen de relieve la importancia de una adecuada variabilidad y cantidad de datos en el conjunto de entrenamiento para maximizar el rendimiento de los modelos y asegurar predicciones precisas.
- La comparación de los resultados reveló que el algoritmo de aprendizaje automático " Decision Tree " sobresale como el más eficiente para diagnosticar la presencia de roya en hojas de café. Este juicio se fundamenta en el rendimiento superior observado al evaluar todos los indicadores propuestos en esta investigación.
- La creación de un aplicativo para la carga y análisis de imágenes de hojas de café facilita a los usuarios la evaluación precisa del estado fitosanitario de las hojas, permitiendo tomar decisiones informadas para el manejo de los cultivos. Esta solución tiene el potencial de mejorar las prácticas agrícolas al proporcionar un medio accesible y confiable para la identificación temprana de la roya. Lo que contribuye directamente a la salud y productividad de los cultivos de café.

4.2. Recomendaciones

La selección de algoritmos de aprendizaje automático para la detección de roya del café exige la realización de una revisión sistemática exhaustiva en diversas bases de datos científicas. Este enfoque se erige como un medio eficaz para la recopilación de un conjunto más amplio de resultados, posibilitando así la confección de una tabla clasificatoria que exhiba un ranking de los algoritmos más destacados. Se buscará resaltar aquellos que manifiesten un rendimiento óptimo en la tarea específica de detección.

Por otro lado, se recomienda limitar el uso de la interfaz diseñada únicamente a la carga de imágenes relacionadas con hojas de café, ya que esta restricción garantizará un rendimiento óptimo del sistema al enfocarse específicamente en el propósito previsto, la

detección de la roya en hojas de café. Al restringir la carga de imágenes a este contexto, se promoverá la eficacia y precisión del análisis, proporcionando así resultados más confiables y relevantes para los usuarios.

Las conclusiones derivadas de esta investigación podrían ser susceptibles a variaciones al modificar las asignaciones de porcentajes destinados al entrenamiento y prueba de los modelos, así como al realizar ajustes en la aleatoriedad de los datos.

REFERENCIAS

[1] E. Lasso, D. C. Corrales, J. Avelino, E. de Melo Virginio Filho, and J. C. Corrales, “Discovering weather periods and crop properties favorable for coffee rust incidence from feature selection approaches,” *Comput Electron Agric*, vol. 176, Sep. 2020, doi: 10.1016/j.compag.2020.105640.

[2] E. J. G. Buitrón, D. C. Corrales, J. Avelino, J. A. Iglesias, and J. C. Corrales, “Rule-based expert system for detection of coffee rust warnings in colombian crops,” *Journal of Intelligent and Fuzzy Systems*, vol. 36, no. 5, pp. 4765–4775, 2019, doi: 10.3233/JIFS-179025.

[3] P. Wei *et al.*, “Coffee flower identification using binarization algorithm based on convolutional neural network for digital images,” *Plant Phenomics*, vol. 2020, 2020, doi: 10.34133/2020/6323965.

[4] M. A. Genaev, E. S. Skolotneva, E. I. Gultyaeva, E. A. Orlova, N. P. Bechtold, and D. A. Afonnikov, “Image-based wheat fungi diseases identification by deep learning,” *Plants*, vol. 10, no. 8, Aug. 2021, doi: 10.3390/plants10081500.

[5] C. Jackulin and S. Murugavalli, “A comprehensive review on detection of plant disease using machine learning and deep learning approaches,” *Measurement: Sensors*, vol. 24, Dec. 2022, doi: 10.1016/j.measen.2022.100441.

[6] D. B. Marin *et al.*, “Detecting coffee leaf rust with UAV-based vegetation indices and decision tree machine learning models,” *Comput Electron Agric*, vol. 190, Nov. 2021, doi: 10.1016/j.compag.2021.106476.

[7] B. M. Abuhayi and A. A. Mossa, “Coffee disease classification using Convolutional Neural Network based on feature concatenation,” *Inform Med Unlocked*, vol. 39, Jan. 2023, doi: 10.1016/j.imu.2023.101245.

[8] Pereira Marcos, Silva Rodovalho Natan, and Backes Andre, *Coffee Leaf Rust Detection Using Genetic Algorithm*. 2019.

[9] E. F. Vilela *et al.*, “New Spectral Index and Machine Learning Models for Detecting Coffee Leaf Miner Infestation Using Sentinel-2 Multispectral Imagery,” *Agriculture*

(Switzerland), vol. 13, no. 2, Feb. 2023, doi: 10.3390/agriculture13020388.

[10] K. J. Ayikpa, D. Mamadou, P. Gouton, and K. J. Adou, “Experimental Evaluation of Coffee Leaf Disease Classification and Recognition Based on Machine Learning and Deep Learning Algorithms,” *Journal of Computer Science*, vol. 18, no. 12, pp. 1201–1212, 2022, doi: 10.3844/jcssp.2022.1201.1212.

[11] D. C. Corrales, A. J. Peña, C. León, A. Figueroa, and J. Carlos Corrales, “Early warning system for coffee rust disease based on error correcting output codes: a proposal.”

[12] F. Martínez, F. José, C. Bogotá, D. C. Colombia, H. Montiel, and C. Fernando Martínez, “A Machine Learning Model for the Diagnosis of Coffee Diseases.” [Online]. Available: www.ijacsa.thesai.org

[13] J. Parraga-Alava, K. Cusme, A. Elica Loor, and E. Santander, “RoCoLe: A robusta coffee leaf images dataset for evaluation of machine learning based methods in plant diseases recognition,” 2019, doi: 10.17632/c5yvn32dzg.2.

[14] *Algorithm for segmentation based on an improved three-dimensional Otsu’s thresholding*. IEEE, 2013.

[15] S. Divakar, A. Bhattacharjee, and R. Priyadarshini, “Smote-DL: A Deep Learning Based Plant Disease Detection Method,” in *2021 6th International Conference for Convergence in Technology, I2CT 2021*, Institute of Electrical and Electronics Engineers Inc., Apr. 2021. doi: 10.1109/I2CT51068.2021.9417920.

[16] L. X. Boa Sorte, C. T. Ferraz, F. Fambrini, R. D. R. Goulart, and J. H. Saito, “Coffee leaf disease recognition based on deep learning and texture attributes,” in *Procedia Computer Science*, Elsevier B.V., 2019, pp. 135–144. doi: 10.1016/j.procs.2019.09.168.

[17] K. Ashokkumar, S. Parthasarathy, S. Nandhini, and K. Ananthajothi, “Prediction of grape leaf through digital image using FRCNN,” *Measurement: Sensors*, vol. 24, Dec. 2022, doi: 10.1016/j.measen.2022.100447.

[18] J. V. Y. Bordin Yamashita and J. P. R. R. Leite, “Coffee disease classification at the edge using deep learning,” *Smart Agricultural Technology*, vol. 4, Aug. 2023, doi:

10.1016/j.atech.2023.100183.

[19] S. Bhowmik, A. K. Talukdar, and K. Kumar Sarma, “Detection of Disease in Tea Leaves Using Convolution Neural Network,” in *International Conference on Advanced Communication Technologies and Signal Processing, ACTS 2020*, Institute of Electrical and Electronics Engineers Inc., Dec. 2020. doi: 10.1109/ACTS49415.2020.9350413.

[20] M. Nawaz, T. Nazir, A. Javed, S. Tawfik Amin, F. Jeribi, and A. Tahir, “CoffeeNet: A deep learning approach for coffee plant leaves diseases recognition,” *Expert Syst Appl*, vol. 237, p. 121481, Mar. 2024, doi: 10.1016/J.ESWA.2023.121481.

[21] D. A. Lyimo, V. Lakshmi Narasimhan, and Z. A. Mbero, “Sensitivity Analysis of Coffee Leaf Rust Disease using Three Deep Learning Algorithms,” in *2021 IEEE AFRICON*, 2021, pp. 1–6. doi: 10.1109/AFRICON51333.2021.9571007.

[22] E. B. Paulos and M. M. Woldeyohannis, “Detection and Classification of Coffee Leaf Disease using Deep Learning,” in *2022 International Conference on Information and Communication Technology for Development for Africa (ICT4DA)*, 2022, pp. 1–6. doi: 10.1109/ICT4DA56482.2022.9971300.

[23] A. K. Lelis, E. G. I. Ferriols, K. M. A.Vallesteros, and J. A. B. Delmo, “A Comparative Analysis of Convolutional Neural Network Architectures for Coffee Leaf Rust Detection,” in *2023 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)*, 2023, pp. 213–218. doi: 10.1109/I2CACIS57635.2023.10193074.

[24] A. Pereira Marcos, N. L. Silva Rodovalho, and A. Backes, *Coffee Leaf Rust Detection Using Convolutional Neural Network*. [IEEE Computer Society], 2019.

[25] C. Sarkar, D. Gupta, U. Gupta, and B. B. Hazarika, “Leaf disease detection using machine learning and deep learning: Review and challenges,” *Appl Soft Comput*, p. 110534, Jun. 2023, doi: 10.1016/j.asoc.2023.110534.

[26] R. S and S. R, “Clustering-based Hemileia Vastatrix Disease Prediction in Coffee Leaf using Deep Belief Network,” in *2023 8th International Conference on Communication and Electronics Systems (ICCES)*, 2023, pp. 1094–1100. doi: 10.1109/ICCES57224.2023.10192835.

[27] A. F. Chavarro, D. Renza, and D. M. Ballesteros, “Influence of Hyperparameters in Deep Learning Models for Coffee Rust Detection,” *Applied Sciences (Switzerland)*, vol. 13, no. 7, Apr. 2023, doi: 10.3390/app13074565.

[28] M. L. J. Cortez, M. de C. Alves, G. R. Carvalho, and E. A. Pozza, “Relationship between Sentinel-2 orbital data and in situ monitoring of coffee rust,” *SN Appl Sci*, vol. 2, no. 8, Aug. 2020, doi: 10.1007/s42452-020-03257-1.

[29] W. Pinheiro Claro Gomes, L. Gonçalves, C. Barboza da Silva, and W. R. Melchert, “Application of multispectral imaging combined with machine learning models to discriminate special and traditional green coffee,” *Comput Electron Agric*, vol. 198, Jul. 2022, doi: 10.1016/j.compag.2022.107097.

[30] E. M. Torres Caballero and A. M. Reyes Duke, *Implementation of Artificial Neural Networks Using NVIDIA Digits and OpenCV for Coffee Rust Detection*. IEEE, 2020.

[31] K. R. Venugopal, *A Modified Bayesian Optimization based Hyper-Parameter Tuning Approach for Extreme Gradient Boosting*. IEEE, 2019.

ANEXOS

Anexo 01: Acta de aprobación del asesor

Universidad Señor de Sipán

ACTA DE APROBACIÓN DEL ASESOR

Yo **Vásquez Leyva Oliver** quien suscribe como asesor designado mediante Resolución Pregrado **N° 0426-2023/FIAU-USS**, del proyecto de investigación titulado **ANALISIS DE ALGORITMOS DE APRENDIZAJE AUTOMATICO PARA DETECTAR LA ROYA DE CAFE**, desarrollado por los estudiantes: **GONZALES INOÑAN OSCAR EDUARDO, LOPEZ CRUZ ALEX FABIAN**, del programa de estudios de **Ingeniería de Sistemas**, acredito haber revisado, y declaro expedito para que continúe con el trámite pertinentes.

En virtud de lo antes mencionado, firman:

Vásquez Leyva Oliver (**Asesor**)	DNI: 40283413	
Gonzales Inoñan Oscar Eduardo **(Autor 1)**	DNI: 71934306	
López Cruz Alex Fabian **(Autor 2)**	DNI: 75463065	

Pimentel, 19 de diciembre del 2023

Anexo 02: Actas de revisión de asesoría

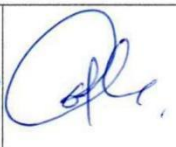
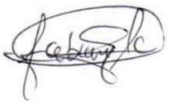
Universidad Señor de Sipán

ACTA DE REVISIÓN DE ASESORÍA

Yo **VÁSQUEZ LEYVA OLIVER**. quien suscribe como asesor designado mediante Resolución Pregrado N° 0426-2023, del proyecto de investigación titulado **ANÁLISIS DE ALGORITMOS DE APRENDIZAJE AUTOMÁTICO PARA DETECTAR LA ROYA DE CAFÉ**, desarrollado por el(los) estudiante(s): **GONZALES INOÑAN OSCAR EDUARDO**, **LOPEZ CRUZ ALEX FABIAN** del programa de estudios de Ingeniería de Sistemas, acredito haber revisado, realizado observaciones y recomendaciones pertinentes tal como se detalla en el siguiente cuadro:

Fecha de revisión:	Modalidad de Asesoría:	Medio de Asesoría:	Veredicto de Asesoría:
26/09/23	Presencial	—	

En virtud de lo antes mencionado, firman:

VÁSQUEZ LEYVA OLIVER	DNI: 40283413	
GONZALES INOÑAN, OSCAR EDUARDO	DNI: 71934306	
LOPEZ CRUZ, ALEX FABIAN	DNI: 75463065	

Pimentel, 26 de setiembre de 2023

Universidad
Señor de Sipán

ACTA DE REVISIÓN DE ASESORÍA

Yo **VÁSQUEZ LEYVA OLIVER**. quien suscribe como asesor designado mediante Resolución Pregrado N° 0426-2023, del proyecto de investigación titulado **ANÁLISIS DE ALGORITMOS DE APRENDIZAJE AUTOMÁTICO PARA DETECTAR LA ROYA DE CAFÉ**, desarrollado por el(los) estudiante(s): **GONZALES INOÑAN OSCAR EDUARDO**, **LOPEZ CRUZ ALEX FABIAN** del programa de estudios de Ingeniería de Sistemas, acredito haber revisado, realizado observaciones y recomendaciones pertinentes tal como se detalla en el siguiente cuadro:

Fecha de revisión:	Modalidad de Asesoría:	Medio de Asesoría:	Veredicto de Asesoría:
03/10/23	Presencial	—	

En virtud de lo antes mencionado, firman:

VÁSQUEZ LEYVA OLIVER	DNI: 40283413	
GONZALES INOÑAN, OSCAR EDUARDO	DNI: 71934306	
LOPEZ CRUZ, ALEX FABIAN	DNI: 75463065	

Pimentel, 03 de octubre de 2023

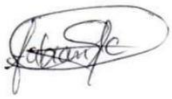
Universidad
Señor de Sipán

ACTA DE REVISIÓN DE ASESORÍA

Yo **VÁSQUEZ LEYVA OLIVER**. quien suscribe como asesor designado mediante Resolución Pregrado N° 0426-2023, del proyecto de investigación titulado **ANÁLISIS DE ALGORITMOS DE APRENDIZAJE AUTOMÁTICO PARA DETECTAR LA ROYA DE CAFÉ**, desarrollado por el(los) estudiante(s): **GONZALES INOÑAN OSCAR EDUARDO**, **LOPEZ CRUZ ALEX FABIAN** del programa de estudios de Ingeniería de Sistemas, acredito haber revisado, realizado observaciones y recomendaciones pertinentes tal como se detalla en el siguiente cuadro:

Fecha de revisión:	Modalidad de Asesoría:	Medio de Asesoría:	Veredicto de Asesoría:
16/10/23	Presencial	—	

En virtud de lo antes mencionado, firman:

VÁSQUEZ LEYVA OLIVER	DNI: 40223412	
GONZALES INOÑAN, OSCAR EDUARDO	DNI: 71934306	
LOPEZ CRUZ, ALEX FABIAN	DNI: 75463065	

Pimentel, 18 de octubre de 2023

Universidad
Señor de Sipán

ACTA DE REVISIÓN DE ASESORÍA

Yo **VÁSQUEZ LEYVA OLIVER**. quien suscribe como asesor designado mediante Resolución Pregrado N° 0426-2023, del proyecto de investigación titulado **ANÁLISIS DE ALGORITMOS DE APRENDIZAJE AUTOMÁTICO PARA DETECTAR LA ROYA DE CAFÉ**, desarrollado por el(los) estudiante(s): **GONZALES INOÑAN OSCAR EDUARDO**, **LOPEZ CRUZ ALEX FABIAN** del programa de estudios de Ingeniería de Sistemas, acredito haber revisado, realizado observaciones y recomendaciones pertinentes tal como se detalla en el siguiente cuadro:

Fecha de revisión:	Modalidad de Asesoría:	Medio de Asesoría:	Veredicto de Asesoría:
25/10/23	Presencial	—	

En virtud de lo antes mencionado, firman:

VÁSQUEZ LEYVA OLIVER	DNI: 40283417	
GONZALES INOÑAN, OSCAR EDUARDO	DNI: 71934306	
LOPEZ CRUZ, ALEX FABIAN	DNI: 75463065	

Pimentel, 25 de octubre de 2023

Universidad
Señor de Sipán

ACTA DE REVISIÓN DE ASESORÍA

Yo **VÁSQUEZ LEYVA OLIVER**. quien suscribe como asesor designado mediante Resolución Pregrado N° 0426-2023, del proyecto de investigación titulado **ANÁLISIS DE ALGORITMOS DE APRENDIZAJE AUTOMÁTICO PARA DETECTAR LA ROYA DE CAFÉ**, desarrollado por el(los) estudiante(s): **GONZALES INOÑAN OSCAR EDUARDO**, **LOPEZ CRUZ ALEX FABIAN** del programa de estudios de Ingeniería de Sistemas, acredito haber revisado, realizado observaciones y recomendaciones pertinentes tal como se detalla en el siguiente cuadro:

Fecha de revisión:	Modalidad de Asesoría:	Medio de Asesoría:	Veredicto de Asesoría:
19/12/23	Presencial	—	

En virtud de lo antes mencionado, firman:

VÁSQUEZ LEYVA OLIVER	DNI: 40283413	
GONZALES INOÑAN, OSCAR EDUARDO	DNI: 71934306	
LOPEZ CRUZ, ALEX FABIAN	DNI: 75463065	

Pimentel, 19 de diciembre de 2023

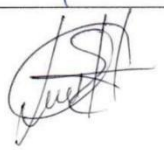
Universidad
Señor de Sipán

ACTA DE REVISIÓN DE ASESORÍA

Yo **VÁSQUEZ LEYVA OLIVER**. quien suscribe como asesor designado mediante Resolución Pregrado N° 0426-2023, del proyecto de investigación titulado **ANÁLISIS DE ALGORITMOS DE APRENDIZAJE AUTOMÁTICO PARA DETECTAR LA ROYA DE CAFÉ**, desarrollado por el(los) estudiante(s): **GONZALES INOÑAN OSCAR EDUARDO**, **LOPEZ CRUZ ALEX FABIAN** del programa de estudios de Ingeniería de Sistemas, acredito haber revisado, realizado observaciones y recomendaciones pertinentes tal como se detalla en el siguiente cuadro:

Fecha de revisión:	Modalidad de Asesoría:	Medio de Asesoría:	Veredicto de Asesoría:
20/12/23	Presencial	—	

En virtud de lo antes mencionado, firman:

VÁSQUEZ LEYVA OLIVER	DNI: 40883413	
GONZALES INOÑAN, OSCAR EDUARDO	DNI: 71934306	
LOPEZ CRUZ, ALEX FABIAN	DNI: 75463065	

Pimentel, 20 de diciembre de 2023

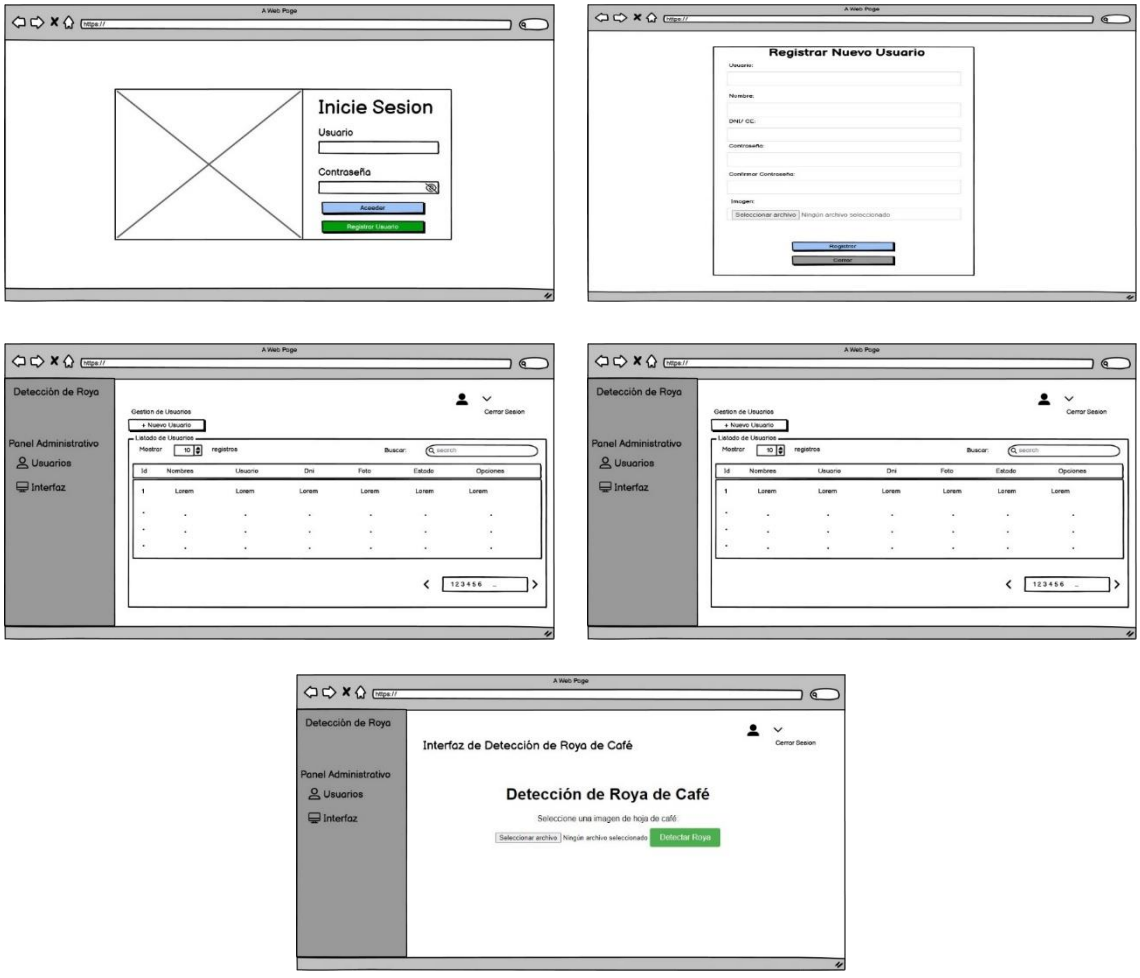
Anexo 03: Matriz de consistencia lógica

Problema	Hipótesis	Objetivo General	Objetivo Específico	Tipo de Investigación	Diseño de Investigación
¿Qué algoritmo de aprendizaje automático presenta mejor eficiencia para detectar la roya de café?	El algoritmo Random Forest (RF) presenta mejor eficiencia para la detección de roya en hojas de café.	Analizar algoritmos de aprendizaje automático para la detección de roya de café.	1. Determinar un Data set a partir de imágenes digitales. 2. Identificar algoritmos de aprendizaje automático para la detección de roya de café. 3. Implementar los algoritmos de aprendizaje automático seleccionados. 4. Analizar los resultados obtenidos. 5. Construir un aplicativo.	Tecnología Aplicada Cuantitativa.	Cuasiexperimental

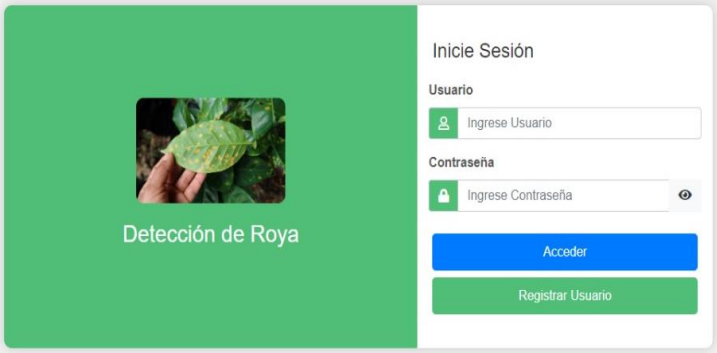
Anexo 04: Tabla de matriz de operacionalización

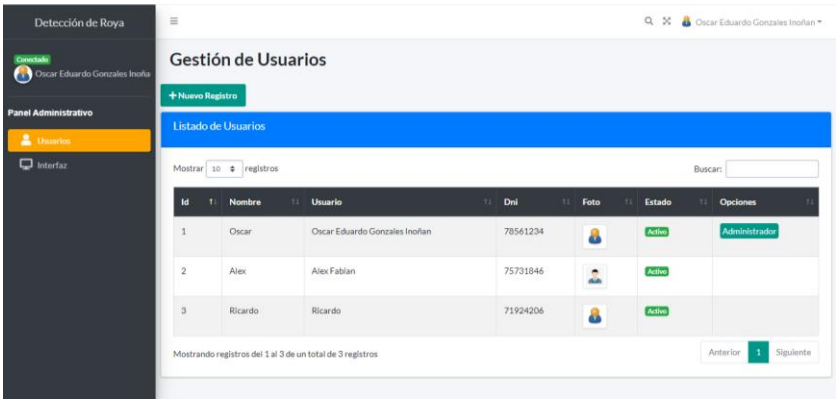
Variable de estudio	Definición conceptual	Definición operacional	Dimensiones	Indicadores	Ítems	Instrumento	Tipo de variable	Escala de medición
Algoritmos de aprendizaje automático	Conjunto de técnicas y métodos	Desempeño	Consumo de recursos	Tiempo de respuesta	$T = Tinicial - Tfinal$	Ficha de Observación	Independiente	Segundos (s)
				Consumo de RAM	$cm = \sum_{i}^{n} \frac{cm_i}{n}$			
Detección de roya de café	Campo de estudio y desarrollo que involucra diversas técnicas y enfoques	Eficiencia	Rendimiento	Precisión	$\frac{VP}{(VP + FP)}$		Dependiente	Porcentaje (%)
				Exactitud	$\frac{VP + VN}{VP + VN + FP + FN}$			
				Recall	$\frac{VP}{(VP + FN)}$			
				F1 – Score	$F1 = 2\frac{(Recall * Precisión)}{Recall + Precisión}$			

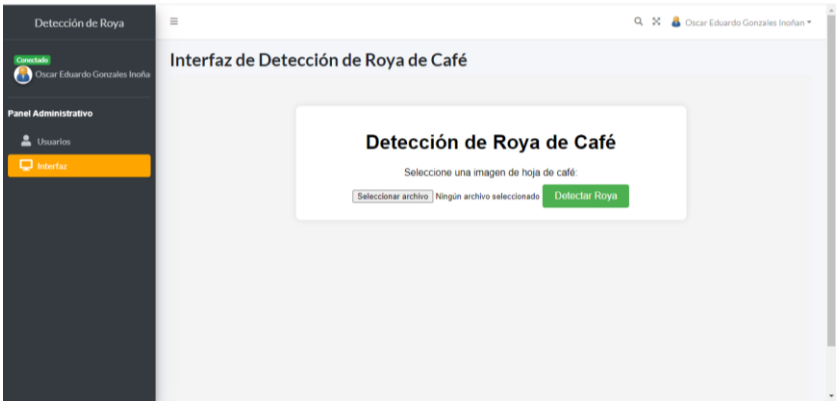
Anexo 05: Mockups



Anexo 06: Interfaces del sistema







Anexo 08: Código fuente de cada modelo

DECISION TREE

```
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score, confusion_matrix,
roc_curve, auc, classification_report
)
from sklearn.preprocessing import StandardScaler
from skimage import io, color, transform
from skimage.feature import graycomatrix, graycoprops
import os
import matplotlib.pyplot as plt
import seaborn as sns
from hyperopt import fmin, tpe, hp, Trials, STATUS_OK

# Función para extraer características de color y textura de cada imagen
def extract_features(image_path, size=(16, 16)):
    image = io.imread(image_path)
    if image.shape[2] == 4:  # Eliminar canal alfa si existe
        image = image[:, :, :3]

    # Redimensionar imagen
```

```
    image_resized = transform.resize(image, size)

    # Características de color en HSV
    hsv_image = color.rgb2hsv(image_resized)
    hist = np.histogram(hsv_image[:, :, 0], bins=8, range=(0, 1))[0]  # Canal H
(matiz)
    hist = hist / np.sum(hist)  # Normalizar histograma
    color_features = hist.flatten()

    # Características de textura usando GLCM
    gray_image = color.rgb2gray(image_resized)
    glcm = graycomatrix((gray_image * 255).astype('uint8'), distances=[1],
angles=[0, np.pi/4, np.pi/2, 3*np.pi/4])
    contrast = graycoprops(glcm, 'contrast').mean()
    dissimilarity = graycoprops(glcm, 'dissimilarity').mean()
    homogeneity = graycoprops(glcm, 'homogeneity').mean()
    ASM = graycoprops(glcm, 'ASM').mean()
    energy = graycoprops(glcm, 'energy').mean()
    texture_features = [contrast, dissimilarity, homogeneity, ASM, energy]

    # Combinar características de color y textura
    return np.concatenate([color_features, texture_features])

# Función para cargar características de imágenes de un directorio
def cargar_caracteristicas_de_imagenes(ruta, cantidad):
    caracteristicas = []
    for archivo in os.listdir(ruta)[:cantidad]:
        if archivo.endswith(('.jpg', '.jpeg', '.png')):
            image_path = os.path.join(ruta, archivo)
            caracteristicas.append(extract_features(image_path))
    return StandardScaler().fit_transform(caracteristicas)

# Rutas de los directorios con las imágenes
ruta_saludables = "D:/TESIS/NuevaDataProcesada/SaludablesAugmentation"
ruta_no_saludables = "D:/TESIS/NuevaDataProcesada/NoSaludablesAugmentation"

# Cargar características de imágenes y etiquetas
imagenes_saludables = cargar_caracteristicas_de_imagenes(ruta_saludables,
cantidad=3130)
imagenes_no_saludables = cargar_caracteristicas_de_imagenes(ruta_no_saludables,
cantidad=3130)

X = np.concatenate((imagenes_saludables, imagenes_no_saludables))
y = np.concatenate((np.ones(len(imagenes_saludables)),
np.zeros(len(imagenes_no_saludables))))

# Dividir los datos en conjunto de entrenamiento y prueba
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42, stratify=y)

# Definir el espacio de búsqueda para los hiperparámetros del Árbol de Decisión
space = {
    'max_depth': hp.choice('max_depth', range(5, 50)),
    'min_samples_split': hp.quniform('min_samples_split', 4, 10, 1),
    'min_samples_leaf': hp.quniform('min_samples_leaf', 1, 3, 1),
    'criterion': hp.choice('criterion', ['gini', 'entropy'])
}

# Función objetivo para la optimización
def objective(params):
    params['min_samples_split'] = int(params['min_samples_split'])
    params['min_samples_leaf'] = int(params['min_samples_leaf'])

    clf = DecisionTreeClassifier(
        random_state=42,
        max_depth=params['max_depth'],
        min_samples_split=params['min_samples_split'],
        min_samples_leaf=params['min_samples_leaf'],
        criterion=params['criterion']
    )
    # Validación cruzada con 5 folds
    score = cross_val_score(clf, X_train, y_train, cv=5,
scoring='accuracy').mean()
    return {'loss': -score, 'status': STATUS_OK}

# Realizar la optimización con hyperopt
trials = Trials()
best_params = fmin(
    fn=objective,
    space=space,
    algo=tpe.suggest,
    max_evals=10,
    trials=trials
)

print("Mejores parámetros encontrados:", best_params)

# Entrenar el modelo con los mejores parámetros
clf = DecisionTreeClassifier(
    random_state=42,
    max_depth=best_params['max_depth'],
    min_samples_split=int(best_params['min_samples_split']),
    min_samples_leaf=int(best_params['min_samples_leaf']),
    criterion=['gini', 'entropy'][best_params['criterion']]
)
```

```
clf.fit(X_train, y_train)

# Realizar predicciones
y_pred = clf.predict(X_test)

# Calcular métricas de evaluación
precision = precision_score(y_test, y_pred)
exactitud = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Imprimir métricas
print(f'Precisión: {precision:.4f}')
print(f'Exactitud: {exactitud:.4f}')
print(f'Recall: {recall:.4f}')
print(f'F1-Score: {f1:.4f}')

# Imprimir el informe de clasificación
print(classification_report(y_test, y_pred))

# Mostrar matriz de confusión
conf_mat = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(conf_mat, annot=True, fmt='d', cmap='Blues')
plt.title('Matriz de Confusión')
plt.xlabel('Predicción')
plt.ylabel('Real')
plt.show()

# Calcular y mostrar la curva ROC
y_prob = clf.predict_proba(X_test)[:, 1]
fpr, tpr, _ = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'AUC = {roc_auc:.2f}')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('Tasa de Falsos Positivos')
plt.ylabel('Tasa de Verdaderos Positivos')
plt.title('Curva ROC')
plt.legend(loc="lower right")
plt.show()

# Graficar el árbol de decisión entrenado
plt.figure(figsize=(20, 10))
plot_tree(
    clf,
    filled=True,
```

```
    feature_names=[f'feature_{i}' for i in range(X.shape[1])],  # Etiquetas de las
características
    class_names=['No Saludable', 'Saludable'],  # Etiquetas de las clases
    rounded=True
)
plt.title("Árbol de Decisión para la Detección de Roya de Café")
plt.show()
```

RANDOM FOREST

```
import numpy as np
import os
import time
import psutil
from skimage import io, color, transform
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
)
import matplotlib.pyplot as plt
import seaborn as sns
from hyperopt import fmin, tpe, hp, STATUS_OK, Trials
from memory_profiler import memory_usage
import cv2

# Función para extraer características de color
def extract_color_features(image):
    # Asegurarse de que la imagen esté en el rango [0, 255] y convertir a uint8
    image = (image * 255).astype(np.uint8)

    # Convertir la imagen a espacio de color HSV
    hsv_image = cv2.cvtColor(image, cv2.COLOR_RGB2HSV)

    # Calcular los histogramas para los tres canales HSV
    hist_hue = cv2.calcHist([hsv_image], [0], None, [8], [0, 256])
    hist_saturation = cv2.calcHist([hsv_image], [1], None, [8], [0, 256])
    hist_value = cv2.calcHist([hsv_image], [2], None, [8], [0, 256])

    # Aplanar y concatenar los histogramas
    color_features = np.concatenate([hist_hue.flatten(),
hist_saturation.flatten(), hist_value.flatten()])
    return color_features

# Función para extraer características de textura
def extract_texture_features(image):
    # Asegurarse de que la imagen esté en el rango [0, 255] y convertir a uint8
    image = (image * 255).astype(np.uint8)
```

```
    # Convertir la imagen a escala de grises
    gray_image = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)

    # Usar GLCM (matriz de co-ocurrencia de niveles de gris) para obtener
características de textura
    glcm = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8)).apply(gray_image)
    mean = np.mean(glcm)
    std_dev = np.std(glcm)

    texture_features = np.array([mean, std_dev])
    return texture_features

# Función para cargar y extraer características de las imágenes
def cargar_y_extraer_caracteristicas(desde_directorio, cantidad=3130,
nuevo_tamano=(16, 16)):
    imagenes = []
    for i, img_file in enumerate(os.listdir(desde_directorio)):
        if i >= cantidad:
            break
        img = io.imread(os.path.join(desde_directorio, img_file))

        # Eliminar canal alfa si es necesario
        if img.shape[2] == 4:
            img = img[:, :, :3]

        # Redimensionar imagen
        img_resized = transform.resize(img, nuevo_tamano)

        # Extraer características de color y textura
        color_features = extract_color_features(img_resized)
        texture_features = extract_texture_features(img_resized)

        # Combinar ambas características
        imagenes.append(np.concatenate((color_features, texture_features)))

    return imagenes

# Rutas de los directorios con las imágenes
ruta_saludables = "D:/TESIS/NuevaDataProcesada/SaludablesAugmentation"
ruta_no_saludables = "D:/TESIS/NuevaDataProcesada/NoSaludablesAugmentation"

# Cargar imágenes y etiquetas
imagenes_saludables = cargar_y_extraer_caracteristicas(ruta_saludables,
cantidad=3130)
etiquetas_saludables = np.ones(len(imagenes_saludables), dtype=int)

imagenes_no_saludables = cargar_y_extraer_caracteristicas(ruta_no_saludables,
cantidad=3130)
```

```
etiquetas_no_saludables = np.zeros(len(imagenes_no_saludables), dtype=int)

# Concatenar imágenes y etiquetas
X = np.concatenate((imagenes_saludables, imagenes_no_saludables))
y = np.concatenate((etiquetas_saludables, etiquetas_no_saludables))

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

# Definir el espacio de búsqueda para los hiperparámetros
space = {
    'n_estimators': hp.choice('n_estimators', range(50, 301, 50)),
    'max_depth': hp.choice('max_depth', range(5, 21)),
    'min_samples_split': hp.choice('min_samples_split', range(2, 11)),
    'min_samples_leaf': hp.choice('min_samples_leaf', range(1, 5)),  # Ajustado
para evitar 0
    'max_features': hp.choice('max_features', ['sqrt', 'log2', None]),
}

# Función objetivo para Hyperopt
def objetivo(params):
    clf = RandomForestClassifier(**params, random_state=42)
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    return {'loss': -accuracy, 'status': STATUS_OK}

# Optimización de hiperparámetros con Hyperopt
trials = Trials()
best_params = fmin(
    fn=objetivo,
    space=space,
    algo=tpe.suggest,
    max_evals=10,
    trials=trials
)

print("Mejores hiperparámetros encontrados:")
print(best_params)

# Entrenar el modelo final con los mejores hiperparámetros
clf = RandomForestClassifier(
    n_estimators=best_params['n_estimators'] * 50 + 50,
    max_depth=best_params['max_depth'] + 5,
    min_samples_split=best_params['min_samples_split'] + 2,
    min_samples_leaf=best_params['min_samples_leaf'] + 1,
    max_features=['sqrt', 'log2', None][best_params['max_features']],
```

```
    random_state=42
)

# Medir memoria máxima durante el entrenamiento y tiempo de ejecución
start_time = time.time()
mem_usage = memory_usage((clf.fit, (X_train, y_train)))
end_time = time.time()
training_time = end_time - start_time

print(f'Tiempo de entrenamiento del modelo: {training_time:.2f} segundos')
print(f'Consumo máximo de memoria durante el entrenamiento: {max(mem_usage):.2f}
MB')

# Realizar predicciones
y_pred = clf.predict(X_test)

# Calcular y mostrar métricas de evaluación
precision = precision_score(y_test, y_pred)
exactitud = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
conf_mat = confusion_matrix(y_test, y_pred)

print(f'Precisión: {precision:.4f}')
print(f'Exactitud: {exactitud:.4f}')
print(f'Recall: {recall:.4f}')
print(f'F1-Score: {f1:.4f}')
print('Matriz de Confusión:')
print(conf_mat)

# Gráfico de la matriz de confusión
plt.figure(figsize=(8, 6))
sns.heatmap(conf_mat, annot=True, fmt='d', cmap='Blues', xticklabels=["No
Saludable", "Saludable"], yticklabels=["No Saludable", "Saludable"])
plt.xlabel('Predicción')
plt.ylabel('Realidad')
plt.title('Matriz de Confusión - Modelo Optimizado')
plt.show()
```

SUPPORT VECTOR MACHINE

```
import os
import numpy as np
from joblib import Parallel, delayed
from skimage.feature import graycomatrix, graycoprops
from skimage.color import rgb2gray
from PIL import Image
from sklearn.svm import SVC
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
from hyperopt import fmin, tpe, hp, STATUS_OK, Trials
import tracemalloc
import time

# Directorios con las imágenes de café saludables y no saludables
ruta_saludables = "D:/TESIS/NuevaDataProcesada/SaludablesAugmentation"
ruta_no_saludables = "D:/TESIS/NuevaDataProcesada/NoSaludablesAugmentation"

# Función para extraer características de color
def extraer_color(imagen):
    promedio_r = np.mean(imagen[:, :, 0])
    promedio_g = np.mean(imagen[:, :, 1])
    promedio_b = np.mean(imagen[:, :, 2])
    return [promedio_r, promedio_g, promedio_b]

# Función para extraer características de textura
def extraer_textura(imagen):
    imagen_gris = rgb2gray(imagen)
    glcm = graycomatrix((imagen_gris * 255).astype(np.uint8), [1], [0])
    contraste = graycoprops(glcm, 'contrast').mean()
    homogeneidad = graycoprops(glcm, 'homogeneity').mean()
    energia = graycoprops(glcm, 'energy').mean()
    correlacion = graycoprops(glcm, 'correlation').mean()
    return [contraste, homogeneidad, energia, correlacion]

# Función para cargar y extraer características
def cargar_y_extraer_caracteristicas(desde_directorio, etiqueta, cantidad=3130):
    archivos = os.listdir(desde_directorio)[:cantidad]

    def cargar_y_extraer(img_file):
        img = Image.open(os.path.join(desde_directorio, img_file)).convert('RGB')
        img_array = np.array(img.resize((16, 16)))
        color_features = extraer_color(img_array)
        texture_features = extraer_textura(img_array)
        return np.concatenate([color_features, texture_features]), etiqueta

    datos = Parallel(n_jobs=-1)(delayed(cargar_y_extraer)(img_file) for img_file
in archivos)
    características, etiquetas = zip(*datos)
    return np.array(características), np.array(etiquetas)

# Cargar las imágenes y etiquetas
```

```
características_saludables, etiquetas_saludables =
cargar_y_extraer_caracteristicas(ruta_saludables, etiqueta=1, cantidad=3130)
características_no_saludables, etiquetas_no_saludables =
cargar_y_extraer_caracteristicas(ruta_no_saludables, etiqueta=0, cantidad=3130)

# Combinar los datos
X = np.vstack((características_saludables, características_no_saludables))
y = np.hstack((etiquetas_saludables, etiquetas_no_saludables))

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

# Definir el espacio de búsqueda para los hiperparámetros
space = {
    'C': hp.uniform('C', 0.1, 1)
}

# Función objetivo para Hyperopt
def objetivo(params):
    clf = SVC(kernel='linear', **params, cache_size=2000, random_state=42)
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    return {'loss': -accuracy, 'status': STATUS_OK}

# Optimización de hiperparámetros
trials = Trials()
best_params = fmin(
    fn=objetivo,
    space=space,
    algo=tpe.suggest,
    max_evals=3,
    trials=trials
)

print("Mejores hiperparámetros encontrados:")
print(best_params)

# Entrenar el modelo final con los mejores hiperparámetros
svm_classifier = SVC(
    kernel='linear',
    C=best_params['C'],
    cache_size=2000,
    random_state=42
)

# Iniciar el monitoreo de memoria
```

```
tracemalloc.start()
start_time = time.time()

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

# Monitorear el uso de memoria
end_time = time.time()
current, peak = tracemalloc.get_traced_memory()
execution_time = end_time - start_time
memory_used_mb = peak / 1024 / 1024
tracemalloc.stop()

# Predicciones
y_pred = svm_classifier.predict(X_test)

# Métricas
precision = precision_score(y_test, y_pred)
exactitud = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Resultados
print('Resultados del SVM con Kernel Linear:')
print(f'Precisión: {precision:.4f}')
print(f'Exactitud: {exactitud:.4f}')
print(f'Recall: {recall:.4f}')
print(f'F1-Score: {f1:.4f}')
print(f'Tiempo de ejecución para entrenar el modelo: {execution_time:.2f}
segundos')
print(f'Consumo de memoria para entrenar el modelo: {memory_used_mb:.2f} MB')

# Tamaño del conjunto
print(f'Tamaño del conjunto de entrenamiento: {len(X_train)}')
print(f'Tamaño del conjunto de prueba: {len(X_test)}')

# Matriz de confusión
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=["No
Saludable", "Saludable"], yticklabels=["No Saludable", "Saludable"])
plt.xlabel("Predicción")
plt.ylabel("Realidad")
plt.title("Matriz de Confusión - SVM con Kernel Linear")
plt.show()
```

CUSTOM CNN

```
import os
```

```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Input
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, confusion_matrix
from PIL import Image
import matplotlib.pyplot as plt
import seaborn as sns
import time
import psutil
from hyperopt import fmin, tpe, hp, Trials, STATUS_OK

# Directorios con las imágenes de café saludables y no saludables
ruta_saludables = "D:/TESIS/NuevaDataProcesada/SaludablesAugmentation"
ruta_no_saludables = "D:/TESIS/NuevaDataProcesada/NoSaludablesAugmentation"

# Función para cargar y etiquetar imágenes desde un directorio
def cargar_y_etiquetar_imagenes(desde_directorio, etiqueta, cantidad=3130):
    imagenes = []
    etiquetas = []
    for i, img_file in enumerate(os.listdir(desde_directorio)):
        if i >= cantidad:
            break
        img = Image.open(os.path.join(desde_directorio, img_file))

        # Asegurarse de que las imágenes tengan 3 canales
        if img.mode != 'RGB':
            img = img.convert('RGB')

        img = img.resize((16, 16))  # Redimensionar a 16x16
        img = np.array(img)
        imagenes.append(img)
        etiquetas.append(etiqueta)
    return imagenes, etiquetas

# Cargar y etiquetar las imágenes de cada ruta
imagenes_saludables, etiquetas_saludables =
cargar_y_etiquetar_imagenes(ruta_saludables, etiqueta=1, cantidad=3130)
imagenes_no_saludables, etiquetas_no_saludables =
cargar_y_etiquetar_imagenes(ruta_no_saludables, etiqueta=0, cantidad=3130)

# Combinar los datos de café saludables y no saludables
X = np.array(imagenes_saludables + imagenes_no_saludables)
y = np.array(etiquetas_saludables + etiquetas_no_saludables)
```

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

# Normalizar los datos (escala de 0 a 1)
X_train = X_train / 255.0
X_test = X_test / 255.0

# Definir el espacio de búsqueda para HyperOpt
space = {
    'filters1': hp.choice('filters1', [16, 32, 64]),
    'filters2': hp.choice('filters2', [32, 64, 128]),
    'dense_units': hp.choice('dense_units', [64, 128, 256]),
    'learning_rate': hp.loguniform('learning_rate', np.log(1e-5), np.log(1e-2))
}

# Función objetivo para HyperOpt
def objective(params):
    model = Sequential([
        Input(shape=(16, 16, 3)),
        Conv2D(params['filters1'], (3, 3), activation='relu'),
        MaxPooling2D(2, 2),
        Conv2D(params['filters2'], (3, 3), activation='relu'),
        MaxPooling2D(2, 2),
        Flatten(),
        Dense(params['dense_units'], activation='relu'),
        Dense(1, activation='sigmoid')
    ])

    model.compile(optimizer=Adam(learning_rate=params['learning_rate']),
                  loss='binary_crossentropy',
                  metrics=['accuracy'])

    # Entrenar el modelo y medir el tiempo y la memoria
    start_time = time.time()
    memory_before = psutil.virtual_memory().used

    early_stopping = EarlyStopping(monitor='val_loss', patience=3)
    history = model.fit(X_train, y_train,
                        validation_split=0.2,
                        epochs=32,
                        batch_size=10,
                        callbacks=[early_stopping],
                        verbose=0)

    end_time = time.time()
    memory_after = psutil.virtual_memory().used
```

```
    # Evaluar el modelo
    y_pred = model.predict(X_test)
    y_pred = (y_pred > 0.5).astype(int)

    f1 = f1_score(y_test, y_pred)

    # Resultados de consumo de tiempo y memoria
    tiempo_entrenamiento = end_time - start_time
    consumo_memoria = (memory_after - memory_before) / 1024 / 1024

    print(f"Tiempo de entrenamiento: {tiempo_entrenamiento:.2f} segundos")
    print(f"Consumo de memoria para entrenamiento: {consumo_memoria:.2f} MB")

    return {'loss': -f1, 'status': STATUS_OK}

# Ejecutar la optimización con HyperOpt
trials = Trials()
best = fmin(fn=objective, space=space, algo=tpe.suggest, max_evals=5,
trials=trials)

print("Mejores parámetros:", best)

# Entrenar el modelo final con los mejores parámetros
best_filters1 = [16, 32, 64][best['filters1']]
best_filters2 = [32, 64, 128][best['filters2']]
best_dense_units = [64, 128, 256][best['dense_units']]
best_learning_rate = best['learning_rate']

model = Sequential([
    Input(shape=(16, 16, 3)),
    Conv2D(best_filters1, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Conv2D(best_filters2, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Flatten(),
    Dense(best_dense_units, activation='relu'),
    Dense(1, activation='sigmoid')
])

model.compile(optimizer=Adam(learning_rate=best_learning_rate),
              loss='binary_crossentropy',
              metrics=['accuracy'])

start_time = time.time()
memory_before = psutil.virtual_memory().used
model.fit(X_train, y_train, epochs=32, batch_size=10)
end_time = time.time()
memory_after = psutil.virtual_memory().used
```

```
# Evaluar el modelo en el conjunto de prueba
y_pred = model.predict(X_test)
y_pred = (y_pred > 0.5).astype(int)

precision = precision_score(y_test, y_pred)
exactitud = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print('Resultados de la Red Neuronal Convolucional (CNN):')
print(f'Precisión: {precision}')
print(f'Exactitud: {exactitud}')
print(f'Recall: {recall}')
print(f'F1-Score: {f1}')
print(f'Tiempo de entrenamiento: {end_time - start_time:.2f} segundos')
print(f'Consumo de memoria para entrenamiento: {(memory_after - memory_before) /
1024 / 1024:.2f} MB')

# Crear y graficar la matriz de confusión
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=["No
Saludable", "Saludable"], yticklabels=["No Saludable", "Saludable"])
plt.xlabel("Predicción")
plt.ylabel("Realidad")
plt.title("Matriz de Confusión - CNN")
plt.show()
```

Anexo 08: Código fuente de las técnicas

AUMENTO DE DATOS

```
import os
from PIL import Image, ImageEnhance, ImageOps
import random
import numpy as np

# Rutas de las carpetas
input_folder = r'D:\TESIS\NuevaDataProcesada\Saludables'
output_folder = r'D:\TESIS\NuevaDataProcesada\SaludablesAugmentation'

# Crear la carpeta de salida si no existe
if not os.path.exists(output_folder):
    os.makedirs(output_folder)

# Definir el número de aumentaciones y el tamaño uniforme
num_augmentations = 20
output_size = (256, 256)
```

```python
# Definir funciones de augmentación
def random_rotation(image):
    return image.rotate(random.uniform(-40, 40), expand=True)

def random_flip(image):
    if random.choice([True, False]):
        return ImageOps.mirror(image)
    return image

def random_zoom(image):
    width, height = image.size
    zoom_factor = random.uniform(0.7, 1.3)
    x_crop = int(width * zoom_factor)
    y_crop = int(height * zoom_factor)
    image = image.resize((x_crop, y_crop), Image.LANCZOS)
    return image.resize((width, height), Image.LANCZOS)

def add_random_noise(image):
    np_img = np.array(image)
    noise = np.random.normal(0, 25, np_img.shape)  # Ajusta la desviación estándar
del ruido
    np_img = np.clip(np_img + noise, 0, 255)
    return Image.fromarray(np_img.astype('uint8'), 'RGBA')

def random_contrast(image):
    """Ajusta el contraste de la imagen."""
    enhancer = ImageEnhance.Contrast(image)
    return enhancer.enhance(random.uniform(0.6, 1.4))

# Procesar cada imagen en la carpeta
for filename in os.listdir(input_folder):
    img_path = os.path.join(input_folder, filename)
    img = Image.open(img_path).convert("RGBA")  # Asegurarse de que tiene canal
alfa

    for i in range(num_augmentations):
        # Aplicar augmentaciones
        augmented_img = random_rotation(img)
        augmented_img = random_flip(augmented_img)
        augmented_img = random_zoom(augmented_img)
        augmented_img = add_random_noise(augmented_img)
        augmented_img = random_contrast(augmented_img)

        # Redimensionar al tamaño uniforme
        augmented_img = augmented_img.resize(output_size, Image.LANCZOS)

        # Guardar la imagen aumentada
        output_path = os.path.join(output_folder, f'aug_{i}_{filename}')
```

```
        augmented_img.save(output_path, format='PNG')
```

REMBG – ELIMINACIÓN DE FONDO DE IMÁGENES

```
import os
import rembg
import io

def remove_background(input_path, output_path):
    with open(input_path, "rb") as input_file:
        # Leer el contenido del archivo
        input_data = input_file.read()

    # Utilizar rembg para quitar el fondo
    result = rembg.remove(input_data)

    with open(output_path, "wb") as output_file:
        output_file.write(result)

def process_images(input_folder, output_folder):
    # Asegurarse de que la carpeta de salida exista
    os.makedirs(output_folder, exist_ok=True)

    # Iterar sobre las imágenes en la carpeta de entrada
    for filename in os.listdir(input_folder):
        if filename.endswith(".jpg"):
            input_path = os.path.join(input_folder, filename)
            output_path = os.path.join(output_folder, filename.replace(".jpg",
"_no_background.png"))

            # Eliminar el fondo
            remove_background(input_path, output_path)

if __name__ == "__main__":
    # Rutas de las carpetas de entrada y salida
    input_folder_path = r'D:\Dataset\DatasetRoya_48Mpx\No_Saludables'
    output_folder_path = r'D:\Dataset\DatasetRoya_48Mpx\No_Saludables_Sin_Fondo'

    # Procesar las imágenes
    process_images(input_folder_path, output_folder_path)

# Definir el espacio de búsqueda de hiperparámetros para Hyperopt

space = {
    'max_depth': hp.choice('max_depth', range(1, 20)),
    'min_samples_split': hp.uniform('min_samples_split', 2, 10),
    'min_samples_leaf': hp.uniform('min_samples_leaf', 1, 5)
}
```