Santiago de Cali, 4 de Junio del 2024

Doctor
Diego Luis Linares Ospina
Director Maestría en Ciencia de Datos
Facultad de Ingeniería y Ciencias
Pontificia Universidad Javeriana de Cali

Asunto: Presentación para evaluación del proyecto aplicado

Cordial Saludo,

Con el fin de cumplir con los requisitos exigidos por la Universidad para llevar a cabo el Trabajo de Grado y posteriormente optar por el título de Magíster en Ciencia de Datos, nos permitimos presentar a su consideración el proyecto de Trabajo de Grado denominado Modelo predictivo para la identificación de enfermedades producidas por la plaga Heilipus Lauri en el cultivo de aguacate Hass en Colombia, por medio del procesamiento y clasificación de imágenes con aplicación de técnicas de Machine Learning, el cual será realizado por el (la) estudiante Karen Andrea Carvajal Jaramillo, Mauricio Castro Collazos, Ramón Siddartha Riveros Pulgarín con código 8979545, 8980327, 8979748 perteneciente a la Maestría en Ciencia de Datos, bajo la dirección del profesor David Arango Londoño.

El suscrito director del Trabajo de Grado autoriza para que se proceda a hacer la evaluación de este Proyecto ante el Tribunal que para el efecto se designe, toda vez que ha revisado cuidadosamente el documento y avala que ya se encuentra listo para ser presentado oficialmente.

Atentamente,

Estudiante

Karen Andrea Carvajal Jaramillo

CC.1033785074 de Bogotá

Estudiante

Ramón Siddartha Riveros Pulgarín

CC. 71273717 de Itagui

Estudiante

Mauricio Castro Collazos

CC. 14624939 de Cali

David Arango Londoño

Director

David Arango Londoño

CC. 1130586950 de Cali

Documentación anexa:

Resumen del Proyecto Aplicado en formato digital (máximo 1 página).

Una copia digital (PDF) del documento del proyecto aplicado

FICHA RESUMEN

PROYECTO APLICADO – MAESTRÍA EN CIENCIA DE DATOS

TÍTULO: Modelo predictivo para la identificación de la enfermedad producida por la plaga *Heilipus Lauri* en el cultivo de aguacate Hass en Colombia, por medio del procesamiento y clasificación de imágenes con aplicación de técnicas de Machine Learning

1. ÁREA DE TRABAJO: Sector Agroindustrial
2. TIPO DE PROYECTO (Aplicado, Innovación, Investigación): Aplicado
3. ESTUDIANTE(S): Karen Andrea Carvajal Jaramillo, Mauricio Castro Collazos, Ramón Siddartha Riveros Pulgarín
4. CORREO ELECTRÓNICO: kacarvaj@javerianacali.edu.co, maucasco@javerianacali.edu.co, rsriveros@javerianacali.edu.co
5. DIRECCIÓN Y TELÉFONO: Calle 7 # 90 –61 Bogotá D.C, 3112907978 – 3043722618 - 3186387629
6. DIRECTOR: David Arango Londoño
7. VINCULACIÓN DEL DIRECTOR: Docente Catedrático
8. CORREO ELECTRÓNICO DEL DIRECTOR: david.arango@javerianacali.edu.co
9. CODIRECTOR (Si aplica): No
10. GRUPO O EMPRESA QUE LO AVALA (Si aplica): Pontificia Universidad Javeriana Cali
11. OTROS GRUPOS O EMPRESAS: No
12. PALABRAS CLAVE (al menos 5): Machine Learning, Clasificación de imágenes, Ciencia de datos, Procesamiento de imágenes, Plagas, Cultivos, Aguacate Hass, tecnología en el sector agroindustrial
13. FECHA DE INICIO: 24 de mayo de 2023
14. FECHA DE FINALIZACIÓN: 04 de Junio 2024
15. RESUMEN:

Las enfermedades causadas por el insecto-plaga *Heilipus Lauri* son una de las principales causantes de los daños en los cultivos de aguacate Hass, adicionalmente reducen la calidad de los cultivos al

generar problemas fitosanitarios que dificultan alcanzar el potencial exportador de este producto. Por lo cual, el presente proyecto plantea entrenar y evaluar un algoritmo de clasificación de imágenes con el uso de técnicas de aprendizaje automático, con el uso de un banco de imágenes recolectadas por Agrosavia, para la construcción de una herramienta que pueda ser usada por los pequeños y medianos productores de este fruto, que sirva para el control y monitoreo del daño causado por estas plagas. Además, se espera que este proyecto sirva como marco de referencia para futuras investigaciones en el sector agropecuario y académico en el ámbito de procesamiento y clasificación de imágenes.

Modelo predictivo para la identificación de enfermedades producidas por la plaga *Heilipus Lauri* en el cultivo de aguacate Hass en Colombia, por medio del procesamiento y clasificación de imágenes con aplicación de técnicas de Machine Learning

Karen Andrea Carvajal Jaramillo

Mauricio Castro Collazos

Ramón Siddartha Riveros Pulgarín

Proyecto Aplicado para optar al título de Magíster en Ciencia de Datos

Director

David Arango Londoño

Pontificia Universidad Javeriana de Cali

Facultad de Ingeniería y Ciencias

Santiago de Cali

2024

07 de Junio 2024

TABLA DE CONTENIDO

LISTA DE ILUSTRACIONES

LISTA DE TABLAS

Introducción

El aguacate Hass es un fruto tropical de alto valor comercial y con gran potencial de exportación, Colombia se destaca como el cuarto país exportador de este producto, siendo el departamento del Tolima el que presenta mayor expansión territorial en el sembrado del aguacate Hass [1].

Se ha observado un incremento en las plagas que generan varias afectaciones en los frutos del aguacate Hass en Colombia, especialmente en departamentos como el Valle del Cauca, Tolima, Caldas y Antioquia. En donde se identificó la causa de la enfermedad causada por la plaga tipo barrenadora que es escarabajo picudo (*Heilipus Lauri*), que viene creando grandes pérdidas económicas significativas para los cultivadores.

Este proyecto tuvo como objetivo principal identificar las posibles afectaciones causadas al fruto del aguacate Hass por diferentes plagas, para este caso de estudio corresponde a la plaga *Heilipus Lauri* en los cultivos de aguacate Hass en Colombia, en donde se utilizó las imágenes generadas por Agrosavia a los frutos afectados. Para lograr la identificación en las imágenes se utilizaron técnicas de Machine Learning aplicadas al preprocesamiento y selección de características en las imágenes. Se emplearon varios algoritmos avanzados que han permitido la segmentación de imágenes en donde se detalla las áreas afectadas por la plaga [3].

En el desarrollo del proyecto se entrenaron diferentes modelos de Machine Learning como son las Redes Neuronales Convolucionales (CNN), Máquinas de Soporte Vectorial (SVM), entre otros modelos. En donde se empleó el conjunto de datos descritos anteriormente, generando un reentrenamiento de los modelos mencionados utilizando métricas de validación como es el Recall, F1-Score para determinar su confiabilidad en la predicción del daño en los frutos y determinar el algoritmo a utilizar para la fase de despliegue.

1. Definición del problema

1.1. Planteamiento del problema

El aguacate Hass es un fruto tropical de alto valor comercial y con gran potencial de exportación, Colombia se destaca como el cuarto país exportador de este producto, y en diferentes departamentos del país como Antioquia, Bolívar, Caldas, Cesar, Quindío, Santander, Valle del Cauca y Tolima, representan el 86% total del área sembrada de aguacate Hass en el país, y siendo el departamento del Tolima con la mayor participación con un 18% de sembrado para la producción nacional [1].

El aguacate Hass se ha convertido en una de las frutas de mayor crecimiento en el mercado mundial, y sus perspectivas son las mejores en cuanto a crecimiento sostenido en las exportaciones futuras, no solamente considerando la campaña de promoción en los Estados Unidos para aumentar su consumo, sino también la demanda existente en los mercados europeos y asiáticos [2].

A este auge productivo enfocado en cubrir la demanda externa, se le suma la contraparte del desarrollo del mercado y son las plagas y enfermedades que se están generando en el cultivo del aguacate Hass en Colombia, siendo esta la principal limitante para el desarrollo y sostenibilidad productiva [4].

Este uno de los tipos de plagas invasivos en los frutos del aguacate Hass son los conocidos por los cultivadores son los picudo (*Heilipus Lauri*) [2], los cuales no existen en los principales mercados importadores del aguacate Hass como es el caso de Estados Unidos, por lo que es necesario generar herramientas que permitan el monitoreo y control de estas plagas, y poder identificar las afectaciones al cultivo, con el fin de elaborar estrategias para generar cosechas de alta calidad y mitigar el riesgo de daños en los frutos, para llevarlos al mercado internacional.

1.2. Formulación del problema

En este contexto, la investigación se destina a responder las siguientes preguntas:

¿Cómo identificar el daño en el fruto de aguacate Hass con el uso de imágenes? ¿Qué tipo de imágenes se deben utilizar para el entrenamiento de diferentes modelos de Machine Learning que puedan predecir la presencia de las enfermedades a causa de la plaga *Heilipus Lauri* en el fruto de aguacate Hass? ¿Cuál técnica de Machine Learning es más adecuada para realizar una correcta clasificación para la detección de las enfermedades causadas por estas plagas en los cultivos de aguacate Hass con el uso de imágenes?

2. Objetivos del proyecto

2.1. Objetivo General

Clasificar las imágenes de aguacate Hass con presencia de daño causado por la plaga Heilipus Lauri con el uso de técnicas de Machine Learning a través del preprocesamiento y selección de características en imágenes, utilizando el banco de imágenes recolectadas por Agrosavia.

2.2. Objetivos Específicos

- Realizar la exploración, limpieza y clasificación del banco de imágenes suministradas para el proyecto, con el fin de preparar los datos necesarios para el entrenamiento de los modelos de clasificación de imágenes con presencia de daño de los frutos de aguacate Hass causado por la plaga *Heilipus Lauri*.
- Entrenar dos modelos de Machine Learning para la clasificación de imágenes de los frutos del aguacate Hass sanos y frutos del aguacate Hass con daño causado por la plaga *Heilipus Lauri* como son las Redes Neuronales Convolucionales (CNN) y Máquinas de Soporte Vectorial (SVM), utilizando el conjunto de datos preprocesados.
- Evaluar los modelos entrenados para determinar la confiabilidad de los resultados de estos.
- Desarrollar una API en línea que permita la exposición y utilización del modelo propuesto para la predicción de daño en los frutos de aguacate causado por la plaga *Heilipus Lauri*.

3. Marco Teórico y Antecedentes

3.1. Marco Teórico

El cultivo de aguacate Hass fue patentado en 1935 por Rudolph Hass en Habra Heights (California), en virtud de la calidad de sus frutos, el alto rendimiento en su producción y su maduración tardía, comparado con otras variedades importantes para la época. Produce frutos esféricos, ovalados, con corteza gruesa y quebradiza, la pulpa es cremosa, con excelente sabor y sin fibra; la semilla es pequeña (bien pegada a la cavidad) y se pela fácilmente. De acuerdo con el estado de madurez, presenta un color que va desde verde opaco hasta morado oscuro. Los frutos son retenidos en la planta hasta por 6 meses posterior a su madurez fisiológica [5].

En nuestro país se adapta muy bien en altitudes entre los 1800 msnm y los 2100 msnm y temperaturas de 5 a 19 °C, precipitación pluvial: 1,200 mm a 2,000 mm anuales bien distribuidas, humedad relativa de 60%, no tolera encharcamientos de agua en las raíces, susceptible a vientos fuertes. pH entre 5.5 a 6.5. [6].

Colombia presenta un gran potencial para ocupar los primeros lugares en la producción de aguacate Hass, sus condiciones ambientales son adecuadas en varios lugares para productores y comercializadoras; para los técnicos agroambientales estos presentan un programa específico para el manejo de las plagas de acuerdo con los ecotipos existentes en cada región del país [4].

La agricultura desempeña un papel crucial en la economía. Los agricultores están buscando maneras de maximizar la producción y rendimientos de sus cultivos. Por lo que, deben realizar un seguimiento, para medir y dar respuesta a variables de tipo ambiental como la temperatura, altitud, precipitación, viento, suelos, entre otras, el llevar un control efectivo sobre estas variables ambientales ayuda aumentar el éxito en la cadena productiva. Sin duda los productores deben ser conscientes de cuáles son las estaciones ideales para la siembra, el mantenimiento y recolección de las cosechas [5].

Para los agricultores dar un entendimiento a los datos recopilados en todo el proceso pre y post producción del aguacate Hass, evidencia un ingreso a diferentes mercados donde puedan tener productos de calidad estableciendo medidas fitosanitarias que permitan prevenir, controlar o erradicar las plagas que afectan los cultivos y sus frutos.

A lo largo del proceso productivo del aguacate Hass, se han identificado varias plagas y enfermedades que afectan la producción desde los pequeños a grandes cultivadores. Para aumenta la producción del aguacate Hass y mejorar la calidad del fruto se determinaron 4 etapas, la primera es la de establecer umbrales de acción para identificar el foco de la plaga en zonas específica del cultivo, la segunda es el monitoreo y tratamiento de elementos químicos o naturales a las plagas que

afectan el cultivo, el tercero es la prevención para la optimizar los costos, mitigar los riesgos ambientales generados por el cultivo, el cuarto es de llevar un control a las estrategias planteadas en las tres primeras etapas para mejorar el mejor desempeño del cultivo [5].

3.1.1. Plagas

Partiendo de los datos históricos obtenidos por los agricultores y grupos de investigación orientados al mejoramiento en la producción del aguacate Hass en Colombia, se identificaron diferentes plagas que afectan el fruto del aguacate Hass [3].

3.1.1.1. Barrenador de la semilla - *Heilipus Laur*i

Es una plaga que está presente en países como México, Guatemala, Costa Rica, Nicaragua, y Colombia, presenta un ciclo biológico de 200 días aproximadamente, en donde los primeros 14 días pasa en incubación con un tamaño de huevo de 1.40 mm de largo y 0.87 mm de ancho, su color al inicio de la oviposición es blanco brillante, pero en su desarrollo final de incubación cambia a un color café, en su estado larval tiene una duración aproximada de 63 días y presenta un tamaño entre 12 mm y 25 mm de longitud, en su estado pupa tiene una duración de 16 días aproximadamente y mide 16.97 mm aproximante con un tórax formado y color blanco cremoso, y en la etapa adulta tiene una duración de 90 a 150 días, aproximadamente presenta un tamaño diferente para hembras de 14.77 aproximadamente y en machos de 13.78 mm aproximadamente [7].

Tipo de Insecto	Afectación del fruto
	
Imagen 1, Adulto *Heilipus Lauri* [37]	**Imagen 2, Fruto Afectado por *Heilipus Lauri***

3.1.2. Metodología de proyectos de datos

Aplicar una metodología de proyecto de datos es importante para definir, analizar y ejecutar una serie de fases para la identificación de las afecciones presentadas en los frutos del aguacate Hass. La ejecución de un proyecto de Machine Learning es necesario para esta metodología.

3.1.2.2. Exploración de datos

La exploración es uno de los primeros pasos en la preparación de los datos, que son utilizados en grandes conjuntos de datos para realizar un análisis profundo y estructurado, para descubrir nuevos conocimientos basados en patrones, utilizando gráficos estadísticos y métodos de visualización de datos [8].

3.1.2.3. Limpieza de datos

La limpieza de los datos es un proceso esencial que permite resolver anomalías identificados en el conjunto de datos, y es conocida por como Data Cleansing o Data Scrubbing, se definen como un conjunto de procesos que permiten identificar y sustituir los datos o registros incompletos,

inexactos, corruptos o irrelevantes.

El proceso de limpieza de datos es gradual, basado siempre en un plan de trabajo plenamente en donde permita definir las estrategias a utilizar en la identificación de las fuentes de error, y determinar las acciones a realizar sobre los orígenes de datos que presentan afectación [9].

3.1.2.4. Clasificación de datos

La clasificación de los datos es un proceso que presenta diferentes ramificaciones que requieren ser validadas a detalle, utilizando diferentes métodos o técnicas avanzadas como Tomek Link o CNN, que son utilizadas para eliminar el ruido de los datos de origen y su aplicación principal se da en el ordenamiento de datos no estructurados [10].

3.1.2.5. Algoritmos de segmentación

La segmentación es un proceso de clasificación de imágenes digitales que se puede realizar por características como el color, la forma y la textura, cada una de estas juega un papel importante en el análisis de los objetos, siendo el color uno de los más afectados debido a factores ambientales como la luz solar, las sombras, lluvia, o la niebla. Los algoritmos de segmentación permiten clasificar objetos o fondos utilizando la información generada por el color de la imagen [11].

3.1.2.6. Modelos de Machine Learning

El Machine Learning surge como un tipo de inteligencia artificial, orientado a la creación de caminos de aprendizaje autónomos, en donde se utilizan algoritmos especializados para identificar patrones en los conjuntos de datos y utilizar sobre estos modelos estadísticos que permiten elaborar predicciones cada vez más acertadas de acuerdo a la definición más óptima de sus parámetros [12].

3.1.2.7. Aprendizaje supervisado

El aprendizaje supervisado es un método que utiliza un conjunto de datos etiquetados para realizar un entrenamiento de los algoritmos para clasificar los datos y predecir resultados, el aprendizaje supervisado se define en dos tipos, el primero es la Clasificación que emplea algoritmos de

segmentación como es una regresión logística, k-NN, random forest, árboles de decisión, clasificador bayesiano, descenso de gradiente estocástico y el segundo es la Regresión que utiliza un método estadístico para validar la relación entre la variable dependiente y las variables independientes, algunos algoritmos son la regresión de Rigde, Lasso, regresión de redes neuronales y regresión logística [13].

El aprendizaje no supervisado utiliza algoritmos para agrupar el conjunto de datos no etiquetados, en donde es posible identificar patrones en los datos que no es posible conocer los de forma simple, se definen en tres categorías, la primera es la Agrupación de clústeres utilizando el algoritmo K-means de segmentación por puntos similares, el segundo Asociación permite identificar el relacionamiento entre las variables del conjunto de datos no etiquetado y el tercero es la Reducción de dimensiones que permite a depurar el ruido generado en los datos, definido de otra forma elimina el ruido de una imagen y mejora su calidad [13].

3.1.2.8. Redes neuronales convolucionales – CNN

Las Redes Neuronales Convolucionales (CCN) es una red tipo profunda, de una variación del perceptrón multicapa o una Red Neuronal Artificial (RNA) que está formada por múltiples capas y su aplicación se realiza en matrices bidimensionales, convirtiéndola en un algoritmo muy efectivo respecto a las tareas de visión artificial como son la segmentación y la clasificación de imágenes, compresión de imágenes y asociación de patrones [14].

Las redes neuronales convolucionales son capaces de aprender a diferenciar y reconocer una gran variedad de objetos dentro de imágenes utilizando el proceso de extracción de características, lo que requiere un conjunto de datos lo suficientemente grande para realizar un procesamiento complejo. Las neuronas de la red van a ser capaces de modelar las características únicas de cada objeto y poder generalizar las a nuevas imágenes de entrada, lo que significa que no solo se reconocer los objetos observados en el entrenamiento, sino que se podrá inferir la clase de imágenes no observadas respecto al relacionamiento y similitudes generadas en el entrenamiento [14].

3.1.2.9. Máquinas de Soporte Vectorial

La Máquina de Soporte Vectorial es un sistema utilizado para realizar entrenamiento de aprendizaje lineal respecto a la clasificación, aplicado a la segmentación de imágenes, reconocimiento de caracteres, clasificación de patrones, entre otras aplicaciones [15].

3.1.2.10. OpenCV (Open Source Computer Vision Library)

Esta es una librería de visión artificial utilizada para la identificación de objetos propios de una imagen, a continuación, se listan diferentes métodos [16]:

- Threshold: Este método permite fijar el umbral de una segmentación de una imagen en una escala de grises en una imagen binaria.
- FindContours: Este método permite definir los contornos de las figuras en una imagen binaria.
- Bitwise And: Este método permite crear una máscara en la imagen procesada.
- Dilate: Este método permite aumentar el área de las regiones blancas o aumentar el tamaño.
- MorpholyEx: Este método permite aplicar transformaciones a la imagen.
- CvtColor: Este método permite cambiar la imagen y pasar de la escala de colores RGB a una escala de grises.

3.2. Antecedentes

En el desarrollo de este proyecto se ha tomado como base varias fuentes de información enmarcadas en artículos científicos, tesis, investigaciones entre otros documentos.

El primer documento fuente de antecedentes para el proyecto planteado es un estudio realizado por la Corporación Colombiana de Investigación Agropecuaria (Agrosavia) y la Universidad Nacional de Colombia (UNAL) Sede Palmira, desarrollaron un proyecto con Inteligencia Artificial (IA) de un modelo predictivo que permite identificar las plagas cuarentenarias o plagas de tipo estacionario que provocan un impacto económico, ambiental o social [1], el estudio se realizó en varias parcelas de los municipios de Sotará y Timbío del departamento del Cauca tomando diferentes muestras, generando un referenciamiento geográfico por árbol censado, lotes de los frutos sanos y no sanos causados por la plaga *Heilipus Lauri* [17].

Con el proyecto realizado por Agrosavia el ingeniero agrónomo Juan Camilo Calero y Magíster en Ciencias Agrarias y el investigador PhD Arturo Carabalí Muñoz de Agrosavia y el profesor John Josepharj Selvaraj de la UNAL (sede palmira), fueron las personas encargadas de realizar el procesamiento de los datos utilizando algoritmos que permitan identificar las variables que más influyentes en las predicciones generadas por los daños causados por las plagas cuarentenarias en donde se encontró una precisión del 80% de afectación en los predios estudiados [17].

Con los resultados de este estudio se generó el modelo que será la base para una aplicación web que Agrosavia está desarrollando para que cualquier productor y asesor técnico pueda hacer análisis con solo ingresar las coordenadas geográficas del predio del cultivo [17].

Se han realizado diferentes investigaciones similares que buscan apoyar y potencializar el sector agrícola de los países con presencia de cultivo de aguacate Hass, así como la investigación presentada en enero de 2015 en la Conferencia Internacional sobre Tecnologías de la Comunicación, la Información y la Computación (ICCICT), en Mumbai, India donde se presentó el proyecto “A Novel Cloud Computing based Smart Farming System for Early Detection of Borer Insects in Tomatoes”. Un novedoso sistema de agricultura inteligente basado en la computación en la nube para la detección temprana de insectos barrenadores en tomates en el artículo se plantea un enfoque

novedoso para resolver este problema mediante monitoreo constante a los cultivos utilizando el procesamiento de datos de los videos generados y la aplicación de robótica orienta al mejoramiento del cultivo.

El documento se concentra en metodologías para detectar plagas en una de las frutas más populares en el mundo como lo es el tomate, en donde usan una metodología que plantea el desarrollo de un algoritmo de procesamiento de video eficiente utilizando el lenguaje de programación Java. En la primera fase del proyecto, el algoritmo se ocupa de la detección de tomate inmaduro en la planta y en la fase posterior el algoritmo se concentra en encontrar el insecto barrenador, en este documento se compara el rendimiento con dos algoritmos como es el clasificación no supervisada (K-means) y el de Análisis de Componentes Principales (PCA), con el resultado de esta investigación se encontró un modelo muy eficiente aumentado la velocidad de procesamiento en 1.71 veces más rápido que otros modelos. Con estos tiempos obtenidos por el modelo, la toma de decisiones es mucho más rápida y precisa para definir la cantidad de pesticida ha rociar en el cultivo de tomate, minimizar la pérdida de pesticida, mitigar los riesgos en salud en los agricultores, consumidores y menos contaminación del ambiente [18].

Otro documento que sirve como antecedente a este proyecto es el artículo titulado “Detección de enfermedades de las hojas de banano y frutas uso de redes neuronales” presentado en la segunda Conferencia Internacional sobre Investigación Inventiva en Aplicaciones Informáticas (ICIRCA-2020), en donde se plantea analizar el crecimiento de la planta de banano de forma eficaz y automática con un coste mínimo. Se propone un sistema para la detección de enfermedades en estas de crecimiento de la plata hasta la etapa final del fruto para ser recolectado, mediante el procesamiento y clasificación de imágenes de las plantas enfermas de banano usando una Red Neuronal Artificial (ANN) para el procesamiento de imágenes. El sistema propuesto implica varios pasos, entre estos la adquisición de las imágenes, el preprocesamiento de imágenes, la extracción de características y la detección y clasificación de enfermedades basadas en redes neuronales artificiales [19].

Por último, el documento usa como antecedente es el artículo titulado “Detección y clasificación de enfermedades de la fruta mediante procesamiento de imágenes y computación en la nube”, el cual

plantea una metodología que permite clasificar las imágenes, analizar las propiedades de la imagen, implementar algoritmos de Clasificación no Supervisados (K-mean), Redes neuronales y Máquinas de Soporte Vectorial (SVM), para la clasificación de datos de entrenamiento en función de las clases impartidas como etiquetas de clase de formación.

Como resultado de este proyecto se evidencia un desarrollo basado en la nube, que permita ayudar a los agricultores de la India, para analizar los datos que sus productos ofrecen, de una forma práctica y sencilla para obtener mejores resultados

Imagen 3, Fruto Afectado por *Heilipus lauri* [38]

3.3. Tecnología

En el desarrollo del proyecto de Machine Learning fue requerido comprender y conocer el ecosistema tecnológico para realizar clasificación de imágenes, a continuación, se describen conceptos y tecnologías usadas para el proyecto:

3.3.1. Computer Vision

La visión por computadora es concepto que inició en los años 50 por parte de Frank Rosenblatt en donde introdujo el Perceptrón, un tipo de red neuronal artificial que podía reconocer patrones visuales simples. Durante los años 60, se desarrollaron más trabajos en el área, enfocándose en la detección de bordes y reconocimiento de formas básicas [16].

3.3.2. OpenCV

En 1999 se desarrolló la primera librería de código abierto para aplicar la visión por computadora [27]OpenCV (Open Source Computer Vision Library) en Intel, liderado por Gary Bradski. La primera versión oficial, OpenCV 1.0, se lanzó en el año 2000. Desde entonces, OpenCV ha crecido significativamente en popularidad y uso.

Esta funcionalidad está implementada en la actualidad en múltiples lenguajes, en la investigación llegamos a la conclusión de realizar 2 pruebas de concepto con R y Python , lenguajes de programación que hemos estudiado ampliamente en la Maestría de Ciencia de Datos y usado de manera masiva por ingenieros y científicos de datos.

3.3.3. Lenguaje R

R es un lenguaje de programación y un entorno de software para el análisis estadístico, la visualización gráfica y la elaboración de informes. Fue creado por Ross Ihaka y Robert Gentleman en la Universidad de Auckland, Nueva Zelanda. R es ampliamente utilizado en estadísticas, investigación biomédica, economía y una multitud de campos para análisis de datos y modelado estadístico [21].

3.3.4. Python

Python es un lenguaje de programación de alto nivel, interpretado y de propósito general, conocido por su legibilidad y sintaxis clara. Fue creado por Guido van Rossum y lanzado por primera vez en 1991. Python es ampliamente utilizado en diversas áreas como desarrollo web, análisis de datos, inteligencia artificial, aprendizaje automático, desarrollo de software, automatización y muchas otras aplicaciones [22].

3.3.5. TensorFlow

TensorFlow es un marco de trabajo integral y de código abierto para el aprendizaje automático, desarrollado por Google en 2015. Se destaca en el procesamiento de imágenes, proporcionando las herramientas necesarias para construir y entrenar modelos avanzados de aprendizaje profundo, como las redes neuronales convolucionales (CNN), que son cruciales para la clasificación de imágenes [23].

TensorFlow proporciona una API flexible que permite definir los modelos de CNNs personalizados, ajustados a las necesidades específicas del proyecto de clasificación de enfermedades del aguacate Hass.

3.3.6. Keras

En el contexto de la clasificación de imágenes relacionadas a las afectaciones en el fruto del aguacate Hass, Keras proporciona herramientas potentes y flexibles que permiten a los investigadores implementar y modificar rápidamente las arquitecturas de CNNs. Ofrece una amplia gama de capas predefinidas, como las convolucionales (Conv2D), de pooling (MaxPooling2D) y totalmente conectadas (Dense) [24].

3.3.7. Data Augmentation

El Data Augmentation, es una técnica crucial en el procesamiento y análisis de imágenes, especialmente en el ámbito del aprendizaje profundo y la clasificación de imágenes. Esta técnica implica la creación de nuevas imágenes de entrenamiento, a partir de imágenes existentes mediante la aplicación de diversas transformaciones, con el objetivo de aumentar la diversidad y cantidad del conjunto de datos sin necesidad de recopilar más datos reales [25].

3.4. Modelos Machine Learning

Para el proyecto se definió utilizar un modelo de regresión logística para iniciar el entendimiento de vectorización de las imágenes , support vector machine el cual es un algoritmo de clasificación y redes neuronales convolucionales para probar con aprendizaje profundo.

3.4.1. Support Vector Machine

Las Máquinas de Vectores de Soporte (SVM) son un modelo de aprendizaje supervisado utilizado para la clasificación y regresión [15]. En el marco de la clasificación de enfermedades en los aguacates Hass, las SVM se utilizan para distinguir entre aguacates sanos y aquellos afectados por enfermedades específicas como *Heilipus lauri.*

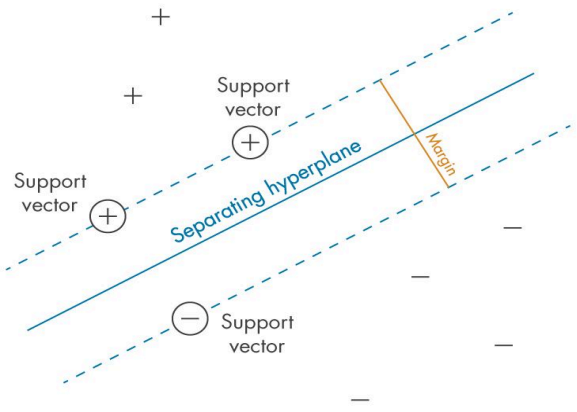


Imagen 4, Máquina de soporte vectorial [39]

El objetivo del algoritmo SVM es encontrar un hiperplano que separe de la mejor forma posible dos clases diferentes de puntos de datos. Lo que implica que el hiperplano con el margen más amplio entre las dos clases. El margen se define como la anchura máxima de la región paralela al hiperplano que no tiene puntos de datos interiores [26].

3.4.2. Convolutional Neural Network

Es un tipo de red neuronal artificial especializada en procesar datos que tienen una estructura en forma de cuadrícula, es utilizada ampliamente en el campo de la visión por computadora para tareas como la clasificación de imágenes, detección de objetos y análisis de video.

Las Redes Neuronales Convolucionales (CNN, por sus siglas en inglés, Convolutional Neural Network), permite capturar automáticamente las características importantes de las imágenes, sin necesidad de extracción manual de características [14].

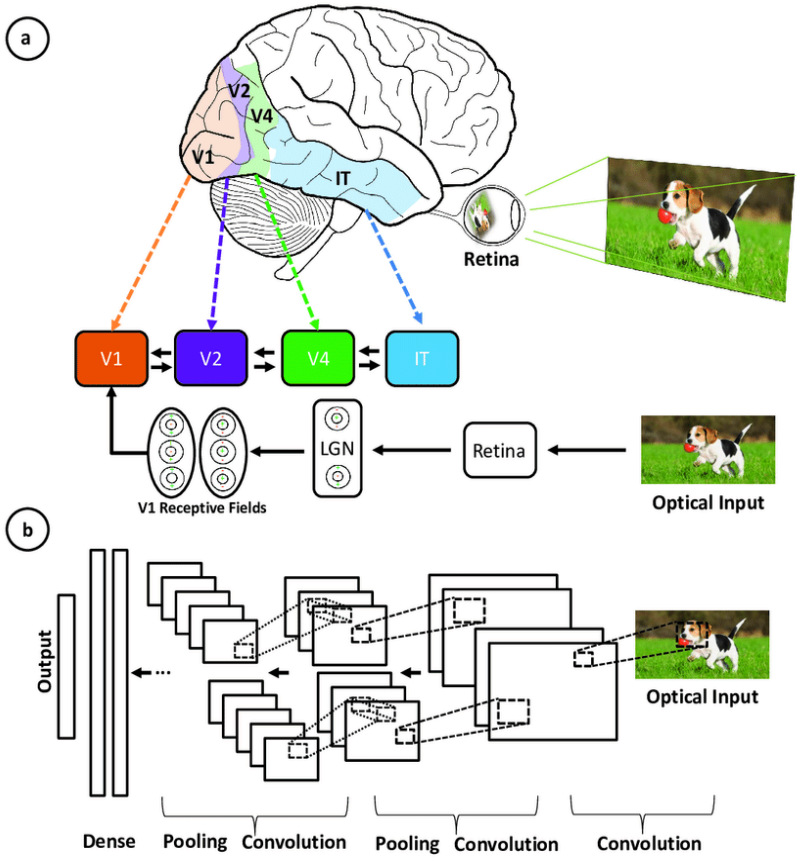


Imagen 5, Red Neuronal Convolucional [40]

3.4.3. Capas de Convolución

Una capa convolucional es el componente central de una red convolucional (CNN) que consta de un conjunto de filtros, diseñados para detectar patrones o características de los datos de entrada por medio operaciones matemáticas de convolución de datos de entrada.

Cada filtro se convierte en una pequeña región de los datos de entrada, realizando una operación matemática que es una multiplicación de una pequeña matriz y la suma de los elementos del filtro y la región correspondiente a los datos del registro, para obtener un único valor del mapa características de la salida [27].

Las pequeñas matrices se deslizan sobre la imagen para extraer características, por medio de filtros que detectan bordes, texturas o patrones más complejos en niveles más profundos de la red.

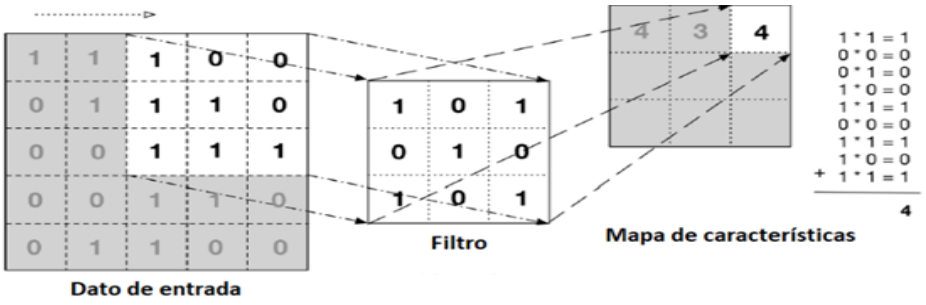


Imagen 6, Capa Convolucional [41]

3.4.3 Capas de Agrupamiento (Pooling)

El pooling es una reducción de muestreo utilizado para reducir las dimensiones espaciales de los datos de entrada en las redes convolucionales (CNN). El Maxpooling es una técnica permite realizar una agrupación de los datos de entrada que son divididos en regiones rectangulares para generar un valor máximo en el mapa de caracteres de salida para ser tomado como un único valor [27].

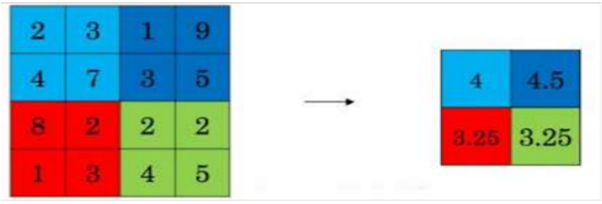


Imagen 7, Capas de Agrupamiento Promedio Pooling [28]

Imagen 8, Capas de Agrupamiento Maxpooling [28]

Imagen 8, Capas de Agrupamiento Maxpooling [28]

3.4.5. Capas de Aplanamiento (Flatten)

La capa de apalancamiento, en una red neuronal convolucional, generalmente va ubicada después de las capas convolucionales y de las capas de agrupamiento. Esta capa se encarga de convertir a las matrices de resultantes de las capas anteriores de vectores planos, para luego pasar la información a las capas densas [28].

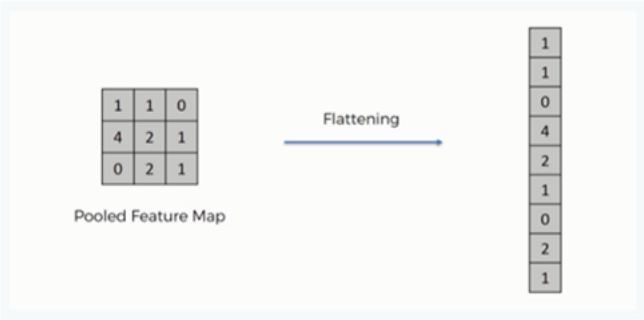


Imagen 9, Capa de apalancamiento [42]

3.4.6. Capas Densa (Fully Connected)

La capa densa es un tipo de neurona que recibe toda la información de las neuronas de la capa anterior, esta combinación de información permite obtener un resultado total de toda la red relacionando las propiedades previamente conocidas en las capas iniciales o anteriores [29].

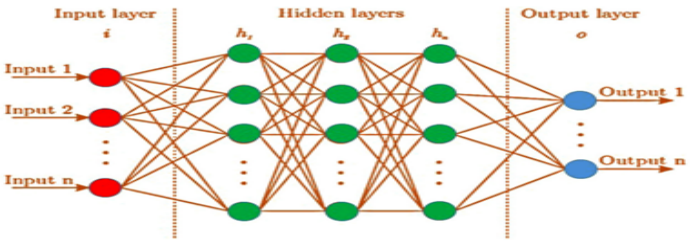


Imagen 10, Fully Connected [43]

3.4.7. Función de Activación

Las capas convolucionales suelen ir seguidas de funciones de activación, como ReLU (Rectified Linear Unit), que introducen la no linealidad en la red y le permiten aprender representaciones más complejas de los datos de entrada. Se pueden apilar varias capas convolucionales una encima de la otra para aprender características cada vez más abstractas de los datos de entrada [27].

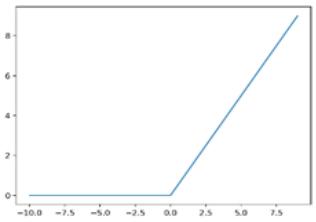


Imagen 11, Función de Activación - ReLU [44]

La función Sigmoide, es utilizada para la activación de las capas ocultas de una red neuronal, estas funciones toman los valores de entrada en un rango entre [0,1], lo que permite realizar una clasificación binaria, basada en una clase [29].

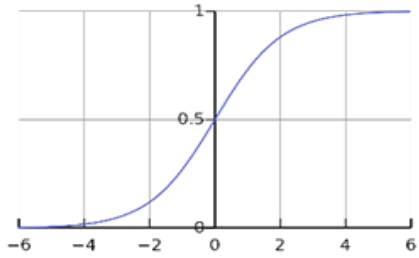


Imagen 12, Función de Activación - Sigmoide [45]

La función Tanh o Tangente Hiperbólica es una función matemática que toma valores en un rango entre [-1, 1], su comportamiento es similar al de la función Sigmoide [29].

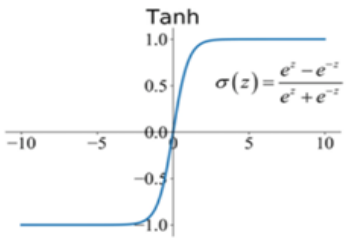


Imagen 13, Función de Activación - Tanh [46]

3.4.8. Aprendizaje y Optimización

Los algoritmos de optimización permiten encontrar la mejor solución disponible para un determinado problema, mediante la búsqueda de los valores precisos de las variables que intervienen. La red ajusta los parámetros de los filtros en las capas de convolucional, ajusta los pesos en la capa densa, para minimizar la diferencia entre la predicción del modelo y los valores reales.

Algoritmo de optimización - Descenso del gradiente estocástico (SGD): Se basa en la idea de actualizar iterativamente los parámetros del modelo dando pequeños pasos en la dirección del gradiente negativo de la función de pérdida.

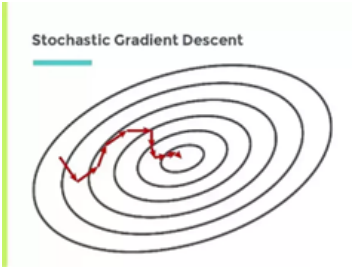


Imagen 14, Algoritmo de optimización - SGD [47]

Algoritmo de optimización – Adam: El algoritmo permite realizar una búsqueda en el menor tiempo o número de pasos, su simplicidad, eficiencia computacional, el poco uso de memoria [30].

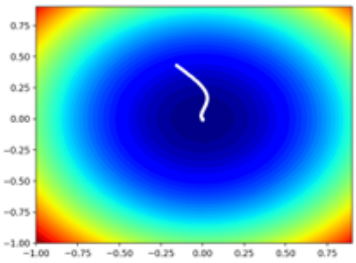


Imagen 15, Algoritmo de optimización - Adam [48]

4. Metodología

4.1. Metodología

La metodología utilizada de referenciamiento respecto al proyecto Modelo predictivo para la identificación de enfermedades producidas por la plaga *Heilipus Lauri* en el cultivo de aguacate Hass en Colombia, por medio del procesamiento y clasificación de imágenes con aplicación de técnicas de Machine Learning es CRISP- DM [31], permite iniciar la fase de planeación hasta la fase de despliegue del modelo de Machine Learning.

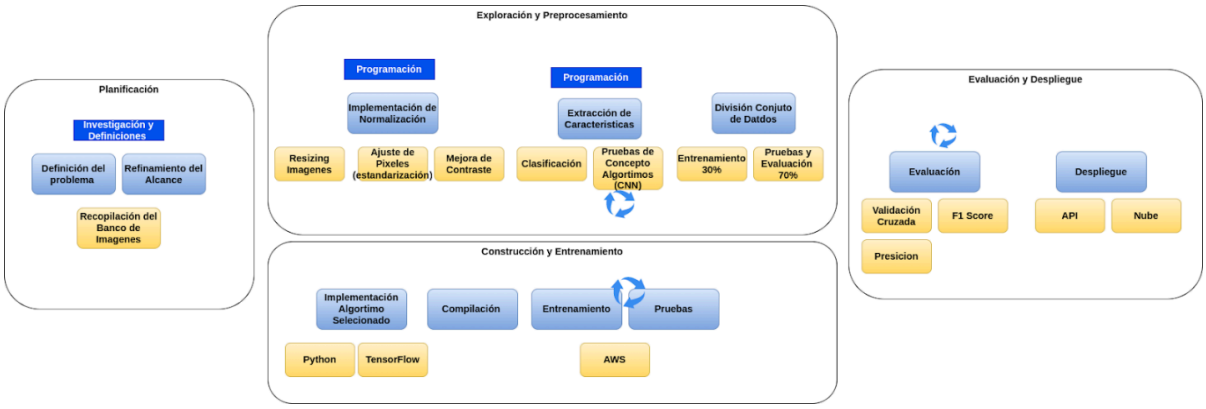


Imagen 16, Metodología Proyecto

A continuación, se describen las fases de la metodología CRISP- DM empleadas en el proyecto:

4.1.1. Planificación

En esta fase el primer paso es entender la problemática a resolver definiendo los objetivos del proyecto.

El objetivo principal de este proyecto fue desarrollar un modelo predictivo de clasificación de imágenes para identificar los frutos del aguacate Hass afectado por la plaga *Heilipus Lauri* y los frutos sanos, por medio de esta segmentación de identificar los beneficios esperados del modelo para los productores del aguacate Hass en Colombia, como la detección temprana de la plaga, mejorar los procesos preventivos e incrementar la calidad del producto con un cultivo sano.

4.1.2. Exploración y preprocesamiento de datos

En esta fase, se realizó un análisis exploratorio de los datos, para obtener un entendimiento de su estructura, la calidad del dato y las características principales del conjunto de datos a validar. En este análisis se identificaron patrones y datos que no cumple con el propósito requerido del proyecto como imágenes borrosas o mal etiquetadas, las cuales requieren un tratamiento adecuado en la etapa de preprocesamiento.

La preparación de los datos implicó una serie de actividades encaminadas al mejoramiento del conjunto de datos, como es la limpieza del banco de imágenes, la normalización y la segmentación de las imágenes. Se aplicaron diferentes técnicas para mejorar la calidad y poder extraer las características más relevantes del conjunto de imágenes. Esto permitió realizar la separación del conjunto universal en tres subconjuntos: el primer conjunto de entrenamiento, el segundo conjunto de validación y el tercer conjunto de pruebas.

4.1.3. Construcción y entrenamiento

En esta fase, se seleccionó y se aplicaron distintos algoritmos de clasificación con el objetivo de desarrollar un modelo predictivo robusto, se realizó una exploración con diferentes arquitecturas de redes neuronales convolucionales (CNN), buscando la mejor predicción respecto a la clasificación de imágenes, además, se ejecutó un entrenamiento con una máquina de soporte vectorial (SVM), para mejor los problemas presentados con la clasificación binaria de forma eficiente y eficaz.

Los modelos CNN y SVM, se entrenaron con el conjunto de datos preparado previamente, en donde metódicamente se ajustaron los hiperparámetros de forma exhaustiva para obtener el mejor rendimiento del modelo. En esta fase se seleccionó la arquitectura de la red neuronal convolucional que mejor predicción arrojó, posterior a esta elección se configuraron los parámetros e hiperparámetros correspondientes. Este enfoque detallado permitió optimizar la capacidad del modelo para capturar y aprender las características relevantes de las imágenes de aguacate Hass, garantizando así un rendimiento óptimo en la clasificación de estas.

4.1.4. Evaluación y despliegue

En esta fase, se realizó la evaluación de los modelos, utilizando métricas de rendimiento estándar, de presión, Recall, F1-Score y la matriz de confusión, se comparó el resultado de dos modelos para

seleccionar la mejor predicción que permite cumplir con los objetivos establecidos en el proyecto, además, se realizó una validación cruzada que permite validar que el modelo seleccionado no estuviera sobre ajustado.

En la fase de despliegue, se desarrolló una API en AWS con el objetivo de exponer y acceder al modelo en el futuro desde una interfaz de usuario o una herramienta dedicada para el consumo del AP; el enfoque que se le dio a esta solución es para beneficiar a los pequeños y medianos productores de aguacate Hass, brindándoles la capacidad de monitorear y certificar sus cultivos de manera efectiva.

5. Resultados de exploración y preparación de imágenes

Se propone realizar una exploración, limpieza y clasificación a fondo del banco de imágenes asignado para el desarrollo del proyecto, con el fin de preparar los datos necesarios para el entrenamiento de los modelos de clasificación de imágenes, en donde se evidencia frutos del aguacate Hass con afectación por la plaga *Heilipus Lauri*, a continuación se presentan los siguientes resultados:

5.1. Exploración y preprocesamiento de datos (Manual)

El proceso de identificación de las características desde la perspectiva humana permite identificar las características de los frutos del aguacate Hass en el conjunto de datos suministrado para el proyecto, en donde se define en un primera clasificación, limpieza y separación del conjunto de datos definido para la extracción de características asociadas a los frutos del aguacate Hass sanos y no sanos.

5.1.1. Reconocimiento de características

En esta fase se identificaron tres tipos de características asociadas al fruto del aguacate Hass correspondientes al conjunto de imágenes a estudiar en el proyecto

- **Característica primaria**: Aguacate Hass sano con o sin pedicelo (tallo), aguacate Hass con perforaciones en la exocarpio o piel con o sin pedicelo (tallo), aguacate Hass con manchas con pedicelo o tallo.
- **Característica secundaria**: Imágenes tomadas en alta resolución, imágenes con poca luz.
- **Característica terciaria**: Para realizar la fotografía del fruto se presentan varias condiciones que alteran el resultado final del procesamiento de la imagen como son un aumento mayor en el brillo o menos iluminación, incremento de sombras, entre otras condiciones.

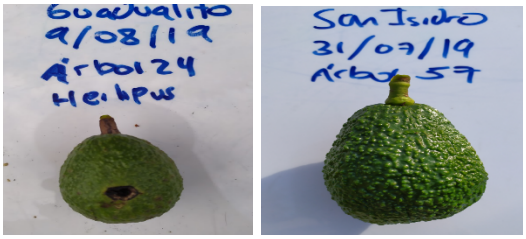


Imagen 17, Tipos de toma

5.1.2. Limpieza del banco de imágenes

En esta fase el objetivo es reducir el ruido o efecto que se puede presentar en una imagen respecto al proceso de captura, digitalización y transmisión, esta transformación es necesaria antes de aplicar la detección de los bordes. Existen diferentes tipos de algoritmos para reducir el ruido.

Los parámetros de configuración permiten adecuar el filtro a las diversas características de la imagen y adaptarlo a diversas aplicaciones o propósitos. Un valor adecuado para estos parámetros puede inferirse sólo después de observar la imagen, por lo tanto, la efectividad del filtrado queda sujeta a la capacidad visual. Desde luego, sería interesante tener un filtro de auto calibrable, es decir, que se ajuste sin supervisión a las condiciones de la imagen. Esto implicaría tener una hipótesis o un modelo de cómo el observador percibe una imagen, para que, a partir de este modelo, sean derivados los valores adecuados de los parámetros [32].

Imagen 18, Imágenes con sombra

5.1.3. Estandarización del tamaño de imágenes, calidad y contraste

En esta fase se identifica una heterogeneidad en la calidad de las imágenes, presentando diferentes tamaños que afectan directamente la proporción de píxeles de la imagen, entre otras características. Para esta actividad se utilizaron algoritmos que permitieron la segmentación e identificaciones de contrastes, máscaras y filtros no identificados a simple vista.

5.1.4. Características de la imagen identificadas

El dispositivo móvil que permitio la captura de las imágenes de los frutos de aguacate Hass con la afectación de la plaga *Heilipus Lauri* o no, presenta diferentes propiedades como el formato de la imagen, el ancho y el alto basado en píxeles, el peso de la imagen, son características principales para el desarrollo del proyecto.

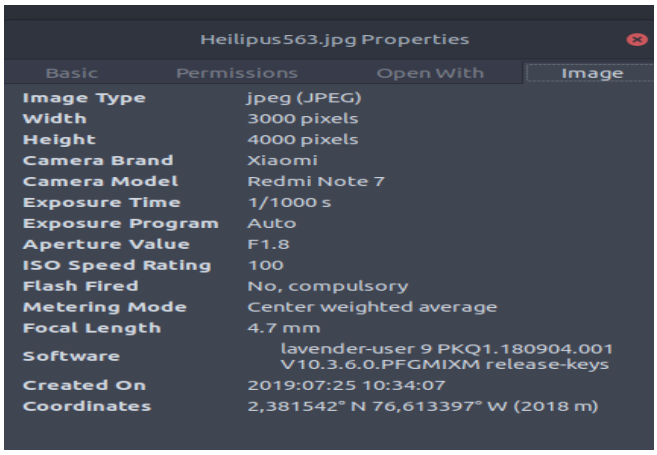


Imagen 19, Características de la imagen

5.2. Extracción de características

En la exploración manual se identificó varias características que son perceptibles a la visión humana, útiles para una primera clasificación de las imágenes definidas el desarrollo del proyecto, pero al involucrar el tema de visión por computadora o visión artificial se obtiene automáticamente una extracción de características más aplica del objeto analizar, cómo son estructuras y propiedades basadas en imágenes bidimensionales o tridimensionales, sean estas en color o en escala gris, iluminación, textura y composición, tamaños.

Se realizaron dos pruebas de concepto rápido o pruebas ad hoc, utilizando los lenguajes de programación R y Python, los cuales son los más demandados para realizar proyectos de Machine Learning, a continuación se lista los siguientes criterios para realizar el preprocesamiento de las imágenes.

- Manipular la imagen
- Obtención de propiedades
- Redimensionamiento
- Remover ruidos externos (sombras, texto, etc.)

5.2.1. Prueba de concepto usando R

La prueba de concepto utilizando el lenguaje de programa de R [21] fue de 2 semanas, en esta actividad se realizó la manipulación de las imágenes y el redimensionamiento del conjunto de datos utilizado en la prueba no fue el más óptimo, debido a la complejidad del lenguaje de programación R y la falta de documentación para el proceso de implementación en proyectos similares al nuestro era poca.

5.2.2. Prueba de concepto usando Python

La prueba de concepto utilizando el lenguaje de programación de Python [22] se desarrolló en las mismas condiciones del lenguaje de programación R, en donde se obtuvo mejor respuesta y desde el proyecto se decidió desarrollarlo en Python, debido a su versatilidad y framework especializados para el preprocesamiento de las imágenes e implementación de los modelos analíticos.

5.3. Limpieza banco de imágenes

Se desarrolló una clase en python (.py) **ProcessImages** para automatizar el tratamiento de las imágenes, utilizando la librería OpenCV que ofrece diferentes funciones utilitarias para analizar imágenes, detectar objetos, realizar el seguimiento de objetos, identificar patrones, segmentación, entre otras funciones

```
    return filtrada

def segmentar_imagen(self,imagen_sin_texto):

# Eliminar sombras
    imagen_sin_texto= self.eliminar_sombras(imagen_sin_texto)
    gray = cv2.cvtColor(imagen_sin_texto, cv2.COLOR_BGR2GRAY)

# Aplicar la corrección gamma para eliminar las sombras
    gamma_corrected = np.array(255*(gray / 255) ** 0.2 , dtype='uint8')
    # Usar thresholding de Otsu
    _, imagen_segmentada = cv2.threshold(gamma_corrected, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)

    # Detección de contornos y conservar solo el contorno más grande
    contours, _ = cv2.findContours(imagen_segmentada, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    # Filtrar contornos por área
    min_area = 3000  # Puedes ajustar este valor
    contours = [c for c in contours if cv2.contourArea(c) > min_area]

    mask = np.zeros_like(gamma_corrected)
    if contours:
        # Ordenar contornos y dibujar el más grande
        contours = sorted(contours, key=cv2.contourArea, reverse=True)
        cv2.drawContours(mask, [contours[0]], -1, (255), thickness=cv2.FILLED)

    aguacate = cv2.bitwise_and(gamma_corrected, gamma_corrected, mask=mask)

    # Mejorar el contraste y brillo del aguacate
    aguacate_resaltado = self.mejorar_contraste(aguacate)

    return aguacate_resaltado
```

Imagen 20, Segmentación de imágenes

```
def eliminar_texto(self,imagen, x_inicio, y_inicio, x_fin, y_fin):
    mascara = np.zeros_like(imagen[:, :, 0])
    mascara[y_inicio:y_fin, x_inicio:x_fin] = 255
    dilatada = cv2.dilate(mascara, (1,1), iterations = 2)  # Dilatar para cubrir más región del texto
    return cv2.inpaint(imagen, dilatada, inpaintRadius=3, flags=cv2.INPAINT_TELEA)
```

Imagen 21, Método eliminar Texto

```
# Aplicar la corrección gamma al canal V
gamma_corrected = np.array(255*(v / 255) ** 0.5 , dtype='uint8')

# Reemplazar el canal V en la imagen HSV con el canal V corregido
hsv[:,:,2] = gamma_corrected

# Convertir la imagen HSV de vuelta a BGR
shadow_removed = cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)
```

Imagen 22, Método conversión

Con la aplicación de los métodos para el tratamiento de las imágenes, se removieron los textos descriptivos que etiquetan al fruto sano y al fruto afectado por la plaga *Heilipus lauri*, eliminado sombras y fijando contornos, a continuación se visualiza el resultado del tratamiento de las imágenes.

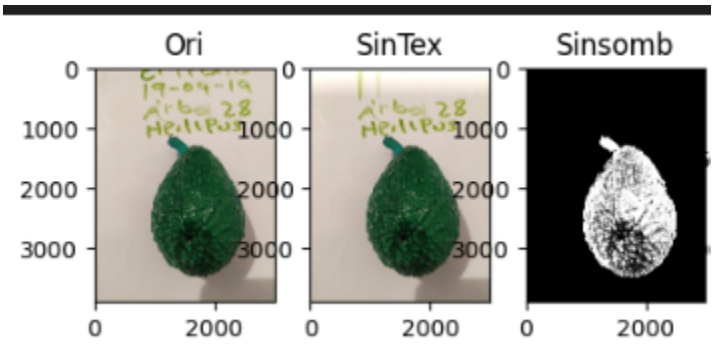


Imagen 23, Preprocesamiento imagen Sana

La remoción del texto se da de forma manual por medio de un recorte de píxeles y se lleva la imagen a una escala de grises, en donde la imagen como resultado final presenta una remoción de sombras.

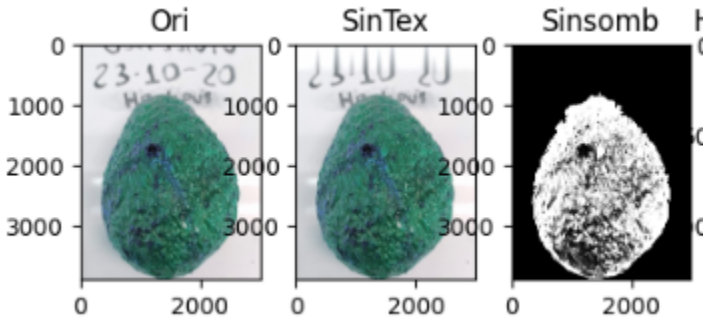


Imagen 24, Preprocesamiento imagen con Helipus Lauri

Al finalizar el tratamiento de la imagen se aprecia un punto negro dentro del fruto que corresponde a la afectación generada por la plaga *Heilipus lauri*, para llegar a este punto se aplicaron tecnicas de Vision Computacional (CV) que permiten resaltar el area afectada en este caso de color negro definido como un circulo.

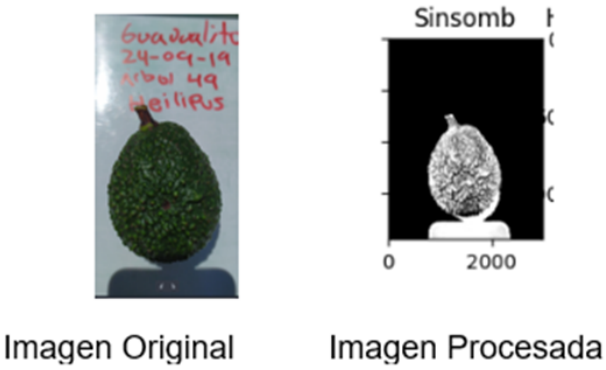


Imagen 25,Sombras no procesadas

Al finalizar el proceso se observa que aún existe ruido en la imagen procesada como sombras, debido al color del fondo que tiene tiene la imagen, y no se logra a identificar si esta sombra está relacionada al fruto del aguacate Hass o el fondo, debido a esta particularidad las imágenes son removidas del set de datos para mitigar el riesgo de valores atípicos en el conjunto de datos de entrenamiento.

5.4. Redimensionamiento de imágenes

Las imágenes entregadas para el desarrollo del proyecto por Agrosavia son fotografías capturadas por dispositivos móviles de gama media lo que no genera un estándar en la calidad de la imagen, el

tamaño de las imágenes es muy grande para realizar el modelamiento del aprendizaje automático, lo que conlleva un coste computacional demasiado alto para el procesamiento y entrenamiento, su tamaño original es de (3000 x 4000 pixeles, 2000 x 1500 pixels, 4000 x 2000 pixeles), por lo que en el redimensionamiento, fue necesario aplicar una conversión distinta a las propiedades de las imágenes para mantener la uniformidad y posterior entrenamiento. Como parte del preprocesamiento y basado en la densidad de píxeles de las imágenes se aplicó un algoritmo de redimensionamiento para transformar las imágenes aún en menor tamaño.

```
if imagen.size>1000000 and imagen.size<=9999999:
    imagen_sin_texto = self.eliminar_texto(imagen, 0, 0, 3000, 250)
elif imagen.size>10000000 and imagen.size<=29999999:
    imagen_sin_texto = self.eliminar_texto(imagen, 0, 0, 3000, 1000)
elif imagen.size>=30000000 and imagen.size<=35000000:
    imagen_sin_texto = imagen
else:
    imagen_sin_texto = self.eliminar_texto(imagen, 0, 0, 3000, 500)
```

```
# Carga y preprocesamiento
def redimensionar(self, image):
    original_height, original_width = image.shape[:2]
    desired_width = 100

    # Calcula el factor de escala y el alto deseado
    scale_factor = desired_width / original_width
    desired_height = int(original_height * scale_factor)

    # Redimensiona la imagen proporcionalmente
    resized_img = cv2.resize(image, (desired_width, desired_height))
    return resized_img
```

Imagen 26,Algoritmo de redimensionamiento

Para nuestro caso el ancho óptimo de la dimensión es de 100 pixeles y poder tener un dataset vectorial, para construir y ejecutar pruebas de densidad y modelos de aprendizaje supervisado para evaluación.

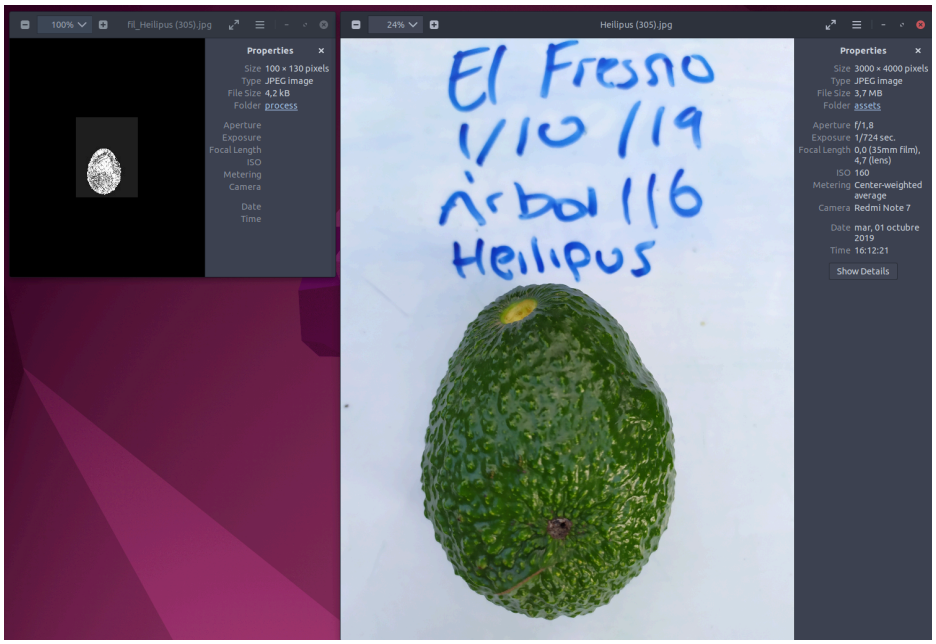


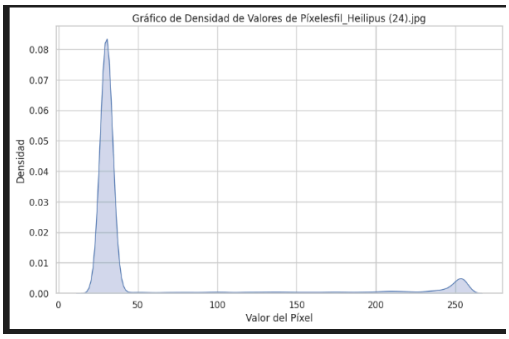
Imagen 27, Conversión

5.5. Construcción del dataset de píxeles

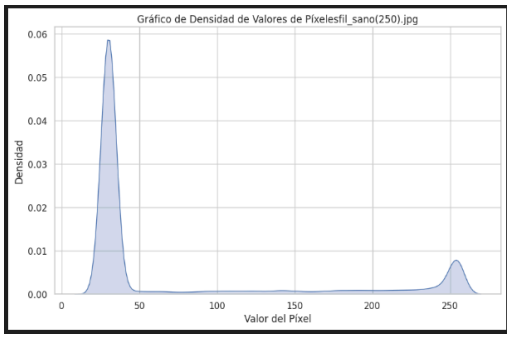
Se generó un dataset en píxeles que permita conocer cómo las imágenes procesadas previamente están a nivel de un arreglo de pixeles, con el fin de realizar pruebas de densidad y realizar una prueba en un modelo de regresión logística y redes convoluciones y para testear el nivel de predicción que ofrece el modelo. Para este dataset se creo las siguientes columnas

- Arreglo de pixeles
- Alto y Ancho
- Etiqueta: 0-Sano, 1-Enfermo

Pruebas de densidad pixeles: Para dar entendimiento entre las diferencias en las imágenes a nivel de su distribución de píxeles, se realizó las pruebas de densidad y se identificó en las imágenes de los frutos de aguacate Hass enfermos, presentan una fluctuación en la densidad del 0.05 comparado con un fruto sano.



Sano



Enfermo

Imagen 28, Densidad Pixelar

6. Resultados de modelación y entrenamiento

Respecto al objetivo específico de entrenar dos modelos de aprendizaje automático para la clasificación de imágenes de frutos de aguacates Hass sanos y frutos de aguacate Hass dañados por la plaga *Heilipus Lauri*.

Se dio inició al proceso de programación y entrenamiento de los modelos, a partir del dataset de imágenes del aguacate Hass, con 564 imágenes de Helipus Lauri y 95 imágenes sanas, en donde se obtuvo un resultado de precisión bajo, por lo cual se amplió el datos set de las imágenes sanas para realizar balanceo de clases y remover el sesgo hacia Helipus Lauri.

6.1. Data Augmentation

Debido a que el dataset del proyecto es bastante reducido respecto a las imágenes de aguacates Hass sanos, fue necesario realizar una ampliación de las imágenes, para poder completar el dataset suministrado por Agrosavia, se realizó un trabajo de campo con el equipo del proyecto obteniendo buscando nuevas imagenes de frutos sanos de aguacate Hass, para poder obtener un balanceo de clases las eficiente, la ampliación permitió obtener pequeñas variaciones de base en las imágenes y posibilita realizar un mejor entrenamiento de los modelos.

Para el entrenamiento del modelo, se generaron 384 nuevas imágenes a partir de 62 imágenes base, todas pertenecientes a la clase "Sano", que era la clase menos representada. Se utilizó la clase **Data**

Augmentation (Es una técnica para aumentar la diversidad de su conjunto de entrenamiento mediante la aplicación de transformaciones aleatorias, con la rotación de imágenes [34]) dentro del proyecto, que hace uso del módulo **ImageDataGenerator** de la librería Keras con la función **random_datagen**. Las transformaciones aplicadas incluyeron:

```python
class DataAugmentation:
    # Función para crear un generador de aumentación de datos con parámetros aleatorios
    def random_datagen(self):
        return ImageDataGenerator(
            rotation_range=random.randint(0, 30),  # Reducir el rango de rotación
            width_shift_range=random.uniform(0.05, 0.15),  # Reducir el rango de cambio de ancho
            height_shift_range=random.uniform(0.05, 0.15),  # Reducir el rango de cambio de altura
            shear_range=random.uniform(0.05, 0.15),  # Reducir el ángulo de cizallamiento
            zoom_range=random.uniform(0.05, 0.15),  # Reducir el rango de zoom
            horizontal_flip=random.choice([True, False]),  # Volteo horizontal aleatorio
            fill_mode='nearest'  # Relleno de pixeles faltantes
        )


    def augment_images(self,directory):
        # Inicializar el generador de aumentación

        # Obtener una lista de archivos de imagen en el directorio
        image_files = [f for f in os.listdir(directory) if os.path.isfile(os.path.join(directory, f))]
        save_dir = directory + '/augmented_images'
        print(save_dir)
        # Procesar cada imagen
        for image_file in image_files:
            img_path = os.path.join(directory, image_file)
            img = load_img(img_path)  # Cargar imagen
            x = img_to_array(img)     # Convertir la imagen a un array
            x = np.expand_dims(x, axis=0)  # Añadir una dimensión extra
            save_prefix = 'aug_' + image_file.split('.')[0]
            datagen = self.random_datagen()
            # Generar y guardar imágenes aumentadas
            i = 0
            for batch in datagen.flow(x, batch_size=1, save_to_dir=save_dir, save_prefix=save_prefix, save_format='jpg'):
                i += 1
                if i >= 20:  # Generar 10 imágenes aumentadas por archivo
                    break
```

Imagen 29, Data Augmentation

- **Rotación (rotation_range):** Las imágenes se rotan en un rango aleatorio de grados entre 0° y 30°. Esto simula la orientación variable de los aguacates Hass en imágenes reales, ayudando al modelo a aprender a reconocer las características de las enfermedades independientemente de la orientación de la fruta.
- **Desplazamiento en ancho y altura (width_shift_range y height_shift_range):** Se aplica un desplazamiento horizontal y vertical a las imágenes dentro de un rango del 5% al 15% del tamaño total de la imagen. Esto permite que el modelo sea menos sensible a la posición exacta de las características de la enfermedad en la imagen.

- **Cizallamiento (shear_range):** Las imágenes se distorsionan aplicando un cizallamiento con un ángulo aleatorio de 5% a 15%. Esta transformación ayuda al modelo a reconocer las enfermedades incluso cuando la forma de los aguacates o las características de la enfermedad están ligeramente distorsionadas.
- **Zoom (zoom_range):** Se aplica un zoom aleatorio a las imágenes dentro de un rango de 5% al 15%, lo que permite al modelo aprender a identificar las características de las enfermedades a diferentes escalas.
- **Rotación horizontal (horizontal_flip):** Las imágenes tienen una probabilidad de ser volteadas horizontalmente. Este espejo refleja la posibilidad de encontrar aguacates orientados de manera diferente, aumentando la invarianza del modelo a la orientación.

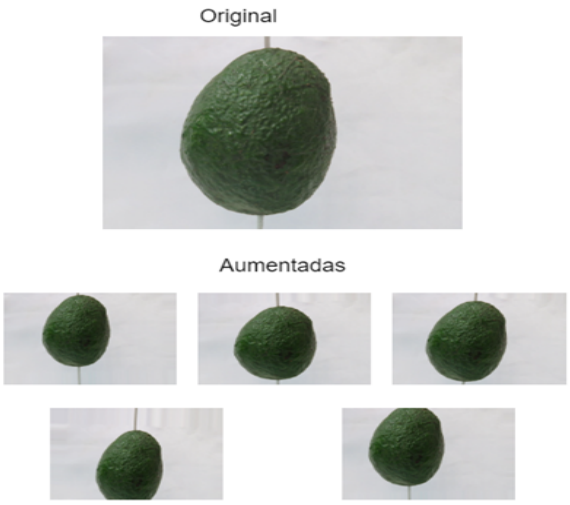


Imagen 30, Ejemplo ejecución Data Augmentation [49]

6.2. Modelos aprendizaje automático

A continuación se describen los modelos seleccionados para entrenar y comparar: Regresión Logística , Máquina de soporte vectorial (SVM) y la Red neuronal convolucional (CNN), utilizando el conjunto de datos preprocesados se obtuvieron los siguientes resultados:

6.2.1. Máquinas de soporte vectorial

La Máquina de Soporte Vectorial (SVM), son un modelo de aprendizaje supervisado, utilizado para la clasificación como para la regresión. En el contexto de este proyecto, se utilizaron SVM para distinguir entre aguacates Hass sanos y afectados por la plaga *Heilipus Lauri*.

Para evaluar el rendimiento del modelo SVM, se llevaron a cabo varias pruebas de validación. En general, las métricas más altas se obtuvieron para las imágenes de frutos afectados por la enfermedad, lo que indica una mayor eficacia del modelo en la detección de aguacates Hass enfermos.

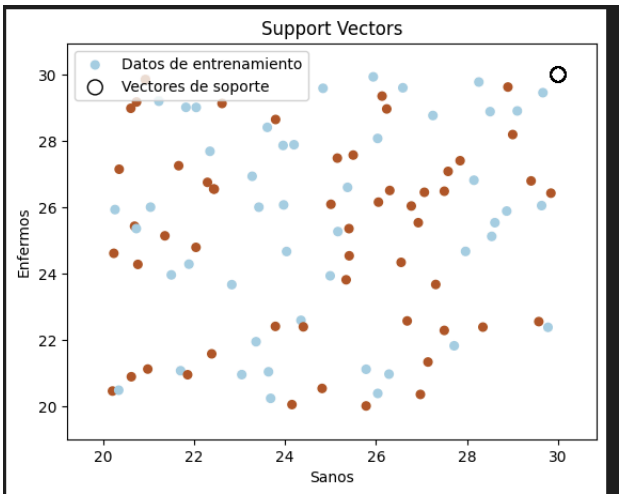


Imagen 31, Distribución de Vectores

Tener solo un vector de soporte es inusual y es un indicio de que el modelo no está bien ajustado. Para que SVM tenga buena precisión es necesario tener varios vectores de soporte que definan la frontera de decisión. Este punto único sugiere que el modelo puede ser demasiado simple o que los datos no están representados adecuadamente.

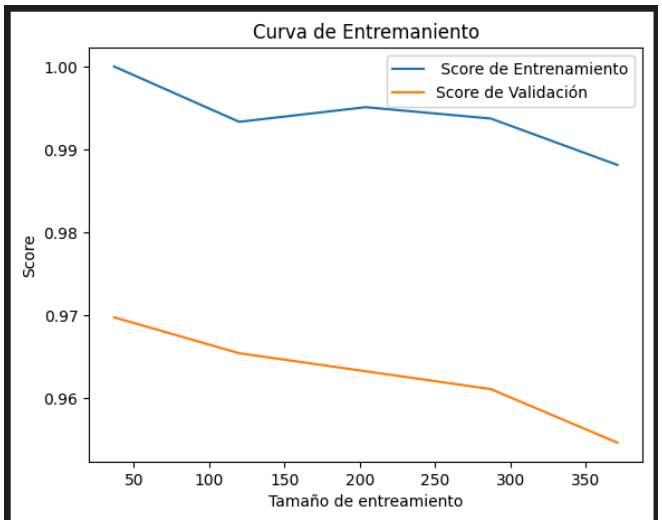


Imagen 32 , SVM Curva de Aprendizaje

Como se puede apreciar en la imagen se marca una tendencia hacia disminución de precisión, lo que indica un claro indicio de sobreajuste, por esta razón se robustece la implementación en la siguiente sesión.

6.2.2. Máquinas de soporte vectorial + Grid Search (Final)

Para mejorar los resultados de la SVM y evitar el sobreajuste que claramente está dando inicialmente se implementó una normalización y GridSearchCV (GridSearchCV es una técnica de validación cruzada incluida en el paquete de scikit learn [36]).

```python
X = np.array(features)
y = np.array(labels)

# Dividir el conjunto de datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
pipeline = Pipeline([
    ('scaler', StandardScaler()),  # Normalizar los datos
    ('svm', svm.SVC(kernel='linear', probability=True))

])
param_grid = {
    'svm__C': [0.1, 1, 10, 100],
    'svm__kernel': ['linear', 'rbf'],
    'svm__gamma': ['scale', 'auto']
}
# Usar GridSearchCV para encontrar los mejores hiperparámetros
grid_search = GridSearchCV(pipeline, param_grid, cv=5, n_jobs=-1, scoring='accuracy')

# Entrenar el modelo SVM usando GridSearchCV
grid_search.fit(X_train, y_train)

# Obtener el mejor modelo
best_model = grid_search.best_estimator_

# Predecir etiquetas para el conjunto de prueba
y_pred = best_model.predict(X_test)
```

Imagen 33, Implementación mejorada SVM

La normalización implementada con ***StandardScaler*** para los datos asegura que todas las características contribuyen equitativamente al modelo, evitando que características con mayores magnitudes dominen el proceso de aprendizaje. Esto es crucial en modelos como el SVM, donde la distancia entre puntos es fundamental para la creación de la frontera de decisión.

GridSearchCV es una implementación de validación cruzada para este caso dividiendo los datos en 5 partes iguales sv=5, lo que permite hacer la búsqueda exhaustiva de los mejores hiperparámetros para el modelo. La ventaja de usar GridSearchCV es que evalúa múltiples combinaciones de parámetros a través de la validación cruzada, garantizando que el modelo esté optimizado para el conjunto de datos específico.

Este proceso de ajuste hiperparámetrico permite encontrar el equilibrio óptimo entre ajuste y generalización, reduciendo así el riesgo de sobreajuste o subajuste. En conjunto, estos enfoques

metodológicos robustecen el rendimiento y la precisión del modelo SVM, asegurando resultados más fiables y reproducibles.

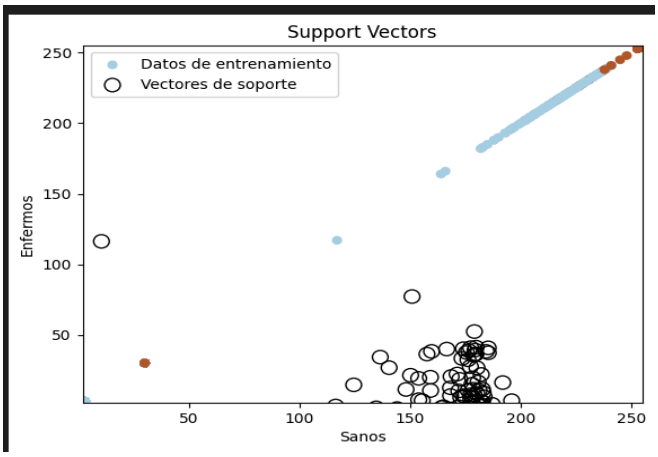


Imagen 34, Distribución de vectores de soporte

Como se puede ver en la imagen , Los círculos con borde negro son los vectores de soporte. La dispersión de los puntos a lo largo de los ejes indica la distribución de las características de los datos, y la concentración de vectores de soporte en ciertas áreas muestra dónde el modelo SVM encuentra más difícil separar las clases, en este caso los Sanos.

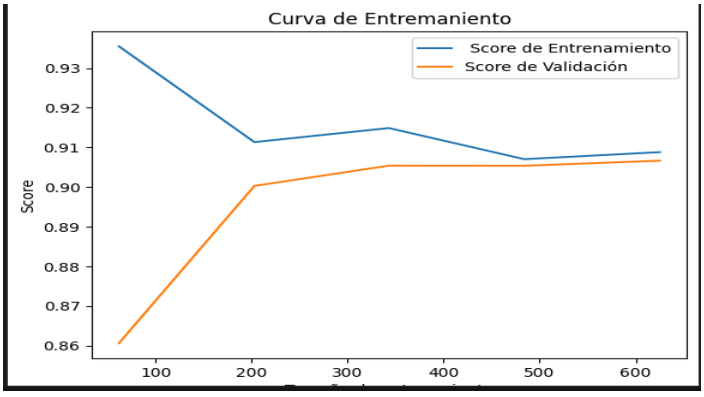


Imagen 35, Distribución de vectores de soporte

La curva de entrenamiento, en esta versión mejorada se observa claramente cómo ataca el sobreajuste y hace converger las curvas de entrenamiento y validación conforme aumenta el número de imágenes.

6.3.2 Convolutional Neural Network

En el proyecto se probaron varias arquitecturas de redes convolucionales, aumentando su profundidad y modificando los hiperparámetros además de la forma en que se cargan las imágenes al modelo con generadores y de la manera tradicional.

6.3.2.1. ImageDataGenerator

```
datagen = ImageDataGenerator(rescale=1./255,shear_range=0.15,zoom_range=0.15,horizontal_flip=True)
print(directorio
+'/train')
generator = datagen.flow_from_directory(
directorio+'/train',
target_size=(200, 200),
batch_size=32,
class_mode='categorical' )


# Generador de datos de validación
validation_datagen = ImageDataGenerator(rescale=1./255)
validation_generator = validation_datagen.flow_from_directory(
    directorio+'/test',
    target_size=(500, 500),
    batch_size=32,
    class_mode='categorical'
)
```

Imagen 36, ImageDataGenerator

Para iniciar la implementación de la CNN se inició la carga de imágenes con **ImageDataGenerator**, para darle eficiencia al uso de la memoria en la etapa de exploración y facilitar el preprocesamiento de las imágenes, alimentando el modelo con lotes de imágenes tensoriales para entrenamiento y validación. Se configuro las siguientes variables al generador:

- **Normalización:** rescale=1./255 normaliza los valores de los píxeles, lo que es esencial para asegurar que los valores de entrada estén en un rango adecuado para el modelo.
- **Corte:** shear_range=0.15 aplica una transformación de corte a las imágenes, ayudando al modelo a aprender características que son invariantes a esta transformación.
- **Zoom:** zoom_range=0.15 aplica un zoom aleatorio, lo cual puede simular diferentes distancias de la cámara al objeto.
- **Rotación Horizontal:** horizontal_flip=True voltea las imágenes horizontalmente, incrementando la variabilidad del conjunto de datos.

6.3.2.2. Cargue Tradicional

```
train_dir = directorio+'/train'  # Ajusta esta ruta
val_dir = directorio+'/test'  # Ajusta esta ruta

# Cargar datos
X_train, y_train, class_names_train = CreateDataSet().load_images_from_directory(train_dir)
X_val, y_val, class_names_val = CreateDataSet().load_images_from_directory(val_dir)

# Normalizar las imágenes
X_train = X_train / 255.0
X_val = X_val / 255.0

# Convertir las etiquetas a categóricas
y_train = to_categorical(y_train, num_classes=len(class_names_train))
y_val = to_categorical(y_val, num_classes=len(class_names_val))

class_weights = compute_class_weight('balanced', classes=np.unique(y_train.argmax(axis=1)), y=y_train.argmax(axis=1))
class_weights = dict(enumerate(class_weights))
```

Imagen 37, Cargue Tradicional en Memoria

Al ejecutar múltiples pruebas con los generadores, se pudo comprobar que aunque era un procesamiento más eficiente en el uso de la memoria, no permite que la CNN aprenda adecuadamente de las características de las imágenes. Cabe recordar que la diferencia entre un aguacate Hass sano y un enfermo, es que en el fruto hay algunas cicatrices de color marron, la cual se puede confundir con las protuberancias del mismo fruto. Por esta razón decidimos hacer una carga de la información de manera tradicional, adicionalmente aplicar un balanceo forzado de clases previo a la compilación del modelo.

Ventajas

- **Control sobre el Preprocesamiento:** Permite un control detallado sobre cómo se cargan y procesan las imágenes.
- **Manejo del Desequilibrio de Datos:** La incorporación de pesos de clases ayuda en proyectos como estos donde se tiene mayor cantidad de imágenes de frutos enfermos que de sanos, ayuda a manejar el desequilibrio en los datos, mejorando la capacidad del modelo para aprender de clases menos representadas, para este caso los sanos.

Desventajas

- **Consumo de memoria**: Cargar todas las imágenes en la memoria es ineficiente para conjuntos de datos grandes.
- **Tiempo de procesamiento**: Debido al alto consumo de memoria se encontró una duplicidad del tiempo del entrenamiento versus que ImageDataGenerator.

6.3.2.3. Arquitectura de la red

Se realizó pruebas con distintas arquitecturas de red neuronal agregando nuevas capas convolucionales, hiperparametros, capas densas, capa de salida y funciones de pérdida intentando obtener la mejor precisión y la menor pérdida en entrenamiento y validación.

Como se mencionó en el marco teórico se implementa una Red Neuronal Convolucional (CNN) utilizando Keras, un marco de trabajo de alto nivel para redes neuronales en Python. El objetivo de implementar esta CNN es clasificar imágenes de aguacate HASS sanas o enfermas por la plaga *Heilipus Lauri*.

- **Capas Convolucionales (Conv2D):** Estas capas aplican un conjunto de filtros a la imagen para extraer características relevantes. Cada capa Conv2D tiene tamaño de kernel 3x3 y profundidades de filtro (32, 64, 128,256 ,512) para aprender una jerarquía de características visuales.
- **Capas de Agrupación (MaxPooling2D):** Después de cada capa convolucional, una capa de agrupación reduce la dimensión espacial de los mapas de características, resumiendo la información y reduciendo la cantidad de parámetros y computación en la red.
- **Capa de Aplanado (Flatten):** Transforma los mapas de características multidimensionales en un vector unidimensional, preparándonos para la clasificación.
- **Capa Densa (Dense):** Capa de neuronas fully connected que interpreta las características extraídas por las capas convolucionales y de agrupación, con una activación ReLu para introducir no lineales.
- **Capa de Abandono (Dropout):** Se utiliza para la regularización, es decir, para prevenir el sobreajuste durante el entrenamiento, descartando aleatoriamente un conjunto de activaciones.

- **Capa de Salida:** La última capa Dense con activación Softmax of Sigmoid se utiliza para la clasificación final, produciendo probabilidades para cada una de las dos clases.

6.3.2.3.1. 3 Capas Convolucionales + Función de Pérdida Binaria

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_1 (Conv2D)	(None, 70, 72, 64)	30,784
max_pooling2d_1 (MaxPooling2D)	(None, 35, 36, 64)	0
conv2d_2 (Conv2D)	(None, 33, 34, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 16, 17, 128)	0
flatten (Flatten)	(None, 34816)	0
dense (Dense)	(None, 128)	4,456,576
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 1)	129

Imagen 38, CNN 3 Capas Convolucionales + Función de Pérdida Binaria

El modelo está compuesto por varias capas: tres capas convolucionales (Conv2D) con funciones de activación ReLU y las capas de pooling (MaxPooling2D), seguidas de una capa de aplanamiento (Flatten), una capa densa (Dense) con 128 neuronas y función de activación ReLU, una capa de abandono (Dropout) con una tasa de 0.5 para regularización, y una capa de salida densa con una neurona y función de activación sigmoide para la clasificación binaria. El modelo se compila con el optimizador Adam y la función de pérdida de **categorical_crossentropy**. El proceso de entrenamiento se lleva a cabo durante 50 épocas con un tamaño de lote de 32, utilizando datos de entrenamiento y validación generados de manera tradicional.

Particularmente la ejecución tiene muy buenos números en la precisión de entrenamiento y validación como se puede apreciar en la imagen

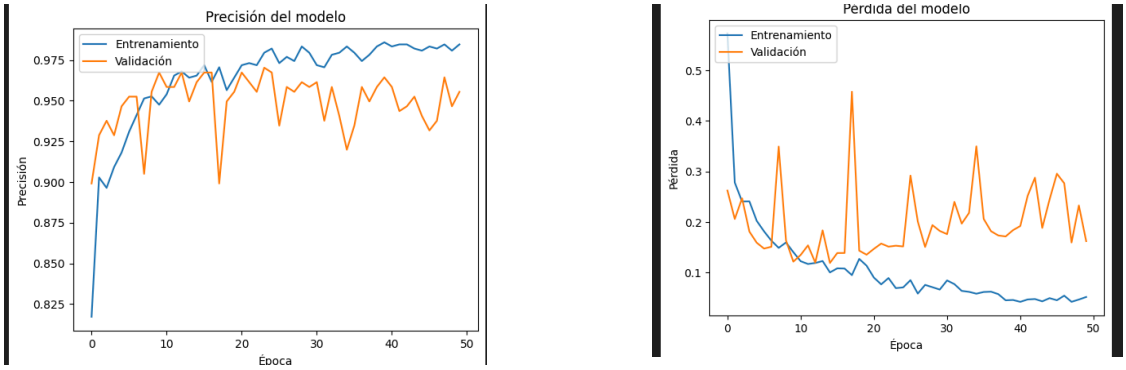


Imagen 39, Precisión y pérdida del modelo Red Neuronal Convolucional 1

Pero al revisar la matriz de confusión y las métricas´ de prueba el desempeño del modelo es bastante regular, debido al desbalance de clase y el no poder identificar de manera adecuada las características de los frutos

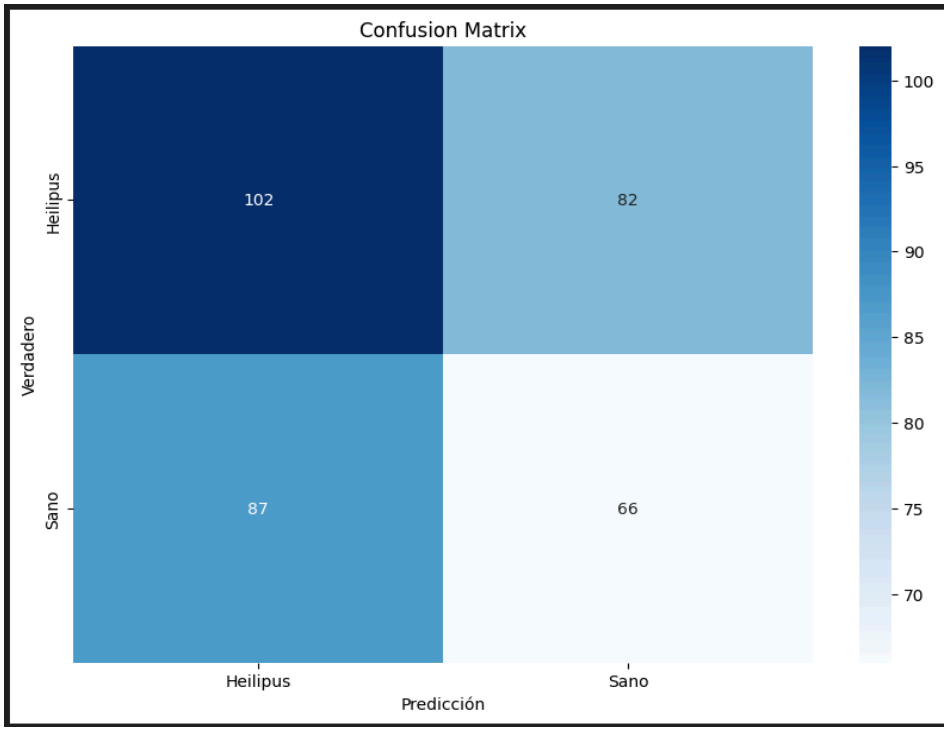


Imagen 40, Matriz de confusión Red Neural Convolucional 1

Esta arquitectura tiene dificultades en la clasificación de imágenes, específicamente, el modelo

identificó correctamente 102 imágenes de aguacates enfermos como enfermos, pero también clasificó erróneamente 82 imágenes de aguacates enfermos como sanos. Además, clasificó correctamente 66 imágenes de aguacates sanos como sanos, pero identificó incorrectamente 87 imágenes de aguacates sanos como enfermos.

Este comportamiento sugiere que el modelo puede tener una precisión y una pérdida aceptables, no permite diferenciar entre aguacates sanos y enfermos y puede verse como un resultado aleatorio en términos de predicción. Esta inexactitud es provocada por el desbalance de clases y a la falta de profundidad en la red neuronal utilizada. Por lo tanto, se optó por una nueva arquitectura para abordar estos problemas y mejorar la capacidad del modelo para clasificar correctamente las imágenes la cual se presenta en el siguiente numeral.

6.3.2.3.2 6 Capas Convolucionales + Función de Pérdida Binaria(final)

Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 200, 200, 64)	1,792
max_pooling2d_5 (MaxPooling2D)	(None, 100, 100, 64)	0
conv2d_6 (Conv2D)	(None, 100, 100, 128)	73,856
max_pooling2d_6 (MaxPooling2D)	(None, 50, 50, 128)	0
conv2d_7 (Conv2D)	(None, 50, 50, 256)	295,168
max_pooling2d_7 (MaxPooling2D)	(None, 25, 25, 256)	0
conv2d_8 (Conv2D)	(None, 25, 25, 512)	1,180,160
max_pooling2d_8 (MaxPooling2D)	(None, 12, 12, 512)	0
conv2d_9 (Conv2D)	(None, 12, 12, 512)	2,359,808
max_pooling2d_9 (MaxPooling2D)	(None, 6, 6, 512)	0
flatten_1 (Flatten)	(None, 18432)	0
dropout_1 (Dropout)	(None, 18432)	0
dense_2 (Dense)	(None, 256)	4,718,848
dense_3 (Dense)	(None, 2)	514

Imagen 41, Red Neuronal Convolucional # Parámetros

El modelo final es una red neuronal convolucional (CNN) que consta de seis capas convolucionales

(Conv2D) con funciones de activación ReLU y padding same, permitiendo mantener las dimensiones espaciales de las imágenes. Estas capas convolucionales tienen un número creciente de filtros (64, 128, 256, 512, 512), que permite capturar características más complejas y detalladas, claves para distinguir entre las protuberancias del fruto y las enfermedades. Las capas convolucionales están seguidas por capas de agrupación máxima (MaxPooling2D) que reducen las dimensiones espaciales y extraen las características más relevantes. Después de aplanar (Flatten) las salidas de las capas convolucionales, se incluye una capa densa (Dense) con 256 neuronas y una capa de abandono (Dropout) con una tasa del 50% para evitar el sobreajuste. La capa de salida es una capa densa con una función de activación softmax para la clasificación multiclase. El modelo se compila con el optimizador Adam, la función de pérdida de **binary_crossentropy** y se entrena utilizando estrategias avanzadas como **EarlyStopping** y **ReduceLROnPlateau** para ajustar dinámicamente la tasa de aprendizaje y prevenir el sobreentrenamiento. Esta arquitectura tiene mayor profundidad y configuraciones que ofrecen las siguientes ventajas:

- **Filtros:** Utiliza más filtros en las capas convolucionales (64, 128, 256, 512, 512), lo que permite capturar más detalles de las imágenes.
- **padding="same":** en todas las capas convolucionales preserva las dimensiones espaciales de las imágenes, evitando la pérdida de características en los bordes.
- **Regularización Mejorada:** Usa una capa densa con 256 neuronas, mejorando la capacidad del modelo para aprender relaciones complejas.
- **Optimizador Adam:** El modelo mejorado utiliza el optimizador Adam con una tasa de aprendizaje ajustable y se complementa con los callbacks EarlyStopping y ReduceLROnPlateau, que optimizan dinámicamente la tasa de aprendizaje y previenen el sobreentrenamiento.
- **Entrenamiento Prolongado:**El modelo mejorado se entrena durante 50 épocas con un batch size de 64 teniendo un entrenamiento más exhaustivo y estable.

El modelo final tiene una arquitectura más robusta y avanzada que permite una extracción de características más profundas, una mejor regularización y una optimización dinámica. Lo que se traduce en un mejor rendimiento y una mayor capacidad de generalización en la clasificación de imágenes de aguacates Hass sanos y enfermos.

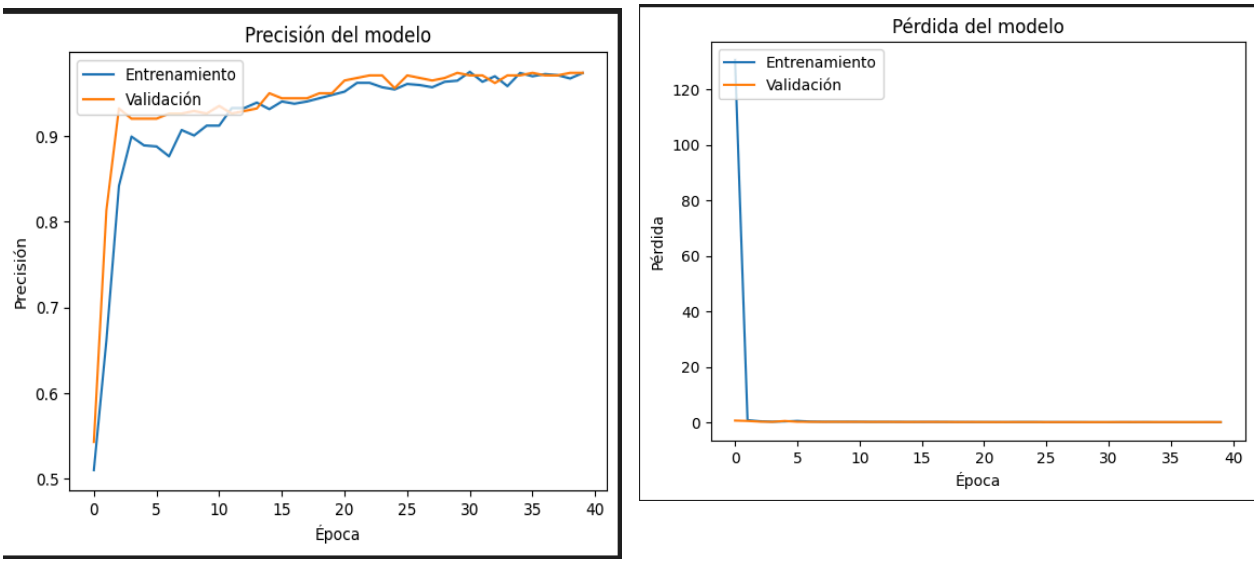


Imagen 42,Precisión y pérdida del modelo Red Neuronal Convolucional 2

La precisión de validación y entrenamiento se mantiene muy cercana a lo largo de todo el proceso, indicando que el modelo tiene una buena capacidad de generalización sin signos evidentes de sobreajuste. En cuanto a la pérdida al inicio del entrenamiento, la pérdida en el conjunto de entrenamiento es extremadamente alta, superando los 120, pero disminuye drásticamente en las primeras 5 épocas, estabilizándose cerca de 0. La pérdida en el conjunto de validación se mantiene consistentemente baja a lo largo de todo el proceso, sin mostrar fluctuaciones significativas.

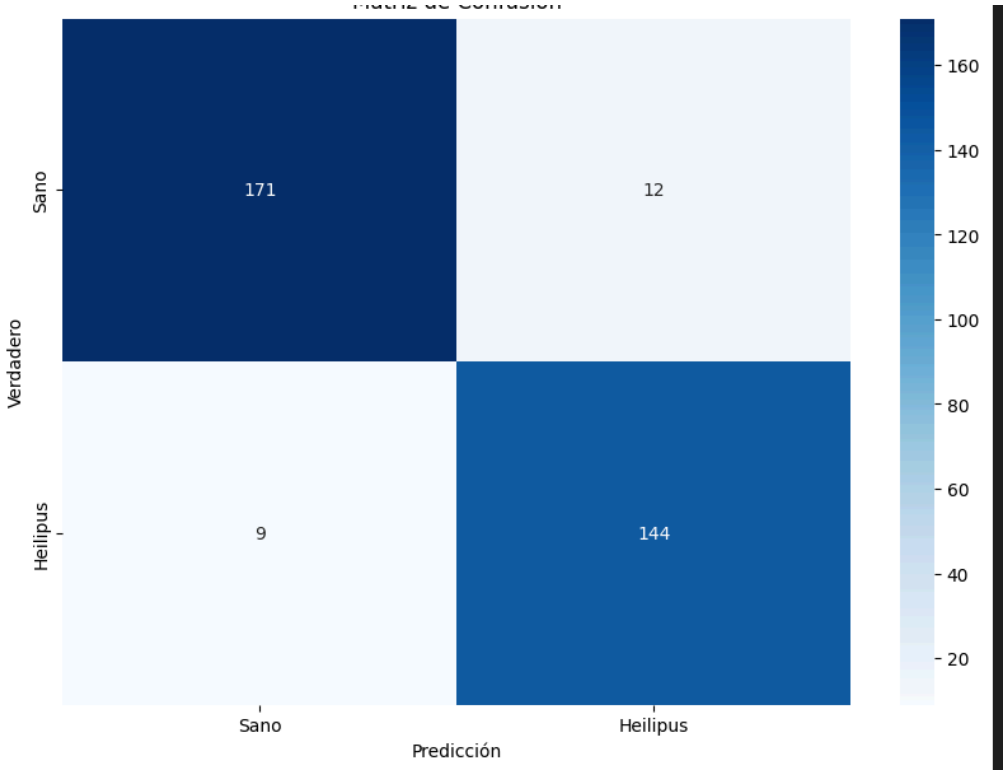


Imagen 43, Matriz de confusión Red Neural Convolucional final

En la nueva matriz de confusión, se observa una reducción significativa en los errores de clasificación. El modelo ahora identifica correctamente un mayor número de imágenes de aguacates sanos y enfermos, disminuyendo tanto los falsos positivos como los falsos negativos. Esta mejora demuestra que la nueva arquitectura trabaja el desbalance de clases y además captura de manera más precisa las características distintivas de cada clase, mejorando así la precisión del modelo.

7. Resultado de evaluación de modelos

7.1. Máquinas de soporte vectorial

7.1.1. Máquinas de soporte vectorial 1

	precisión	recall	f1-score	support
0 (Sano)	0,84375	0,7941176471	0,8181818182	34
1 (Heilipus)	0,9583333333	0,9698795181	0,9640718563	166
accuracy	0,94	0,94	0,94	0,94
macro avg	0,9010416667	0,8819985826	0,8911268372	200
weighted avg	0,9388541667	0,94	0,9392705498	200

Tabla 1, Reporte de modelo de máquinas de soporte vectorial 1

Los resultados obtenidos muestran que la precisión global del modelo es del 94%. Sin embargo, al analizar la precisión por clase, se observa que el modelo tiene una mejor precisión para identificar imágenes con daños causados por enfermedades, con un accuracy del 96%, mientras que tiene una precisión más baja para detectar imágenes de aguacates sanos, con un accuracy del 84%. Además, se nota que el modelo presenta una sensibilidad menor para la clase de aguacates sanos, con un recall del 79%. A pesar de esto, el F1-score del 81% es una medida sólida, especialmente considerando el desbalance significativo entre las clases.

7.1.1.1. Validación cruzada máquinas de soporte vectorial 1

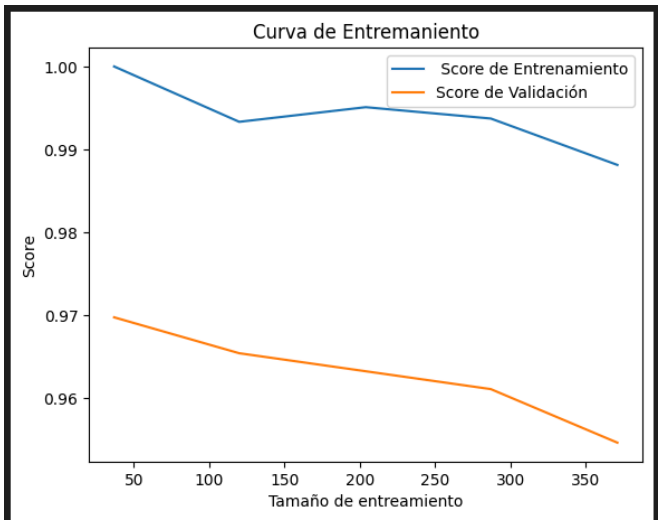


Imagen 44, Validación cruzada SVM 1

En la imagen de validación cruzada se evidencia que el modelo, tanto para el conjunto de datos de entrenamiento como para el de prueba, comienza con un puntaje alto. Sin embargo, a medida que aumentan las iteraciones, se nota una tendencia decreciente. Esta disminución en el rendimiento sugiere posibles problemas de sobreajuste, donde el modelo aprende demasiado de los datos de entrenamiento iniciales y no generaliza bien a nuevos datos.

7.1.2.2. Matriz de confusión máquinas de soporte vectorial 1

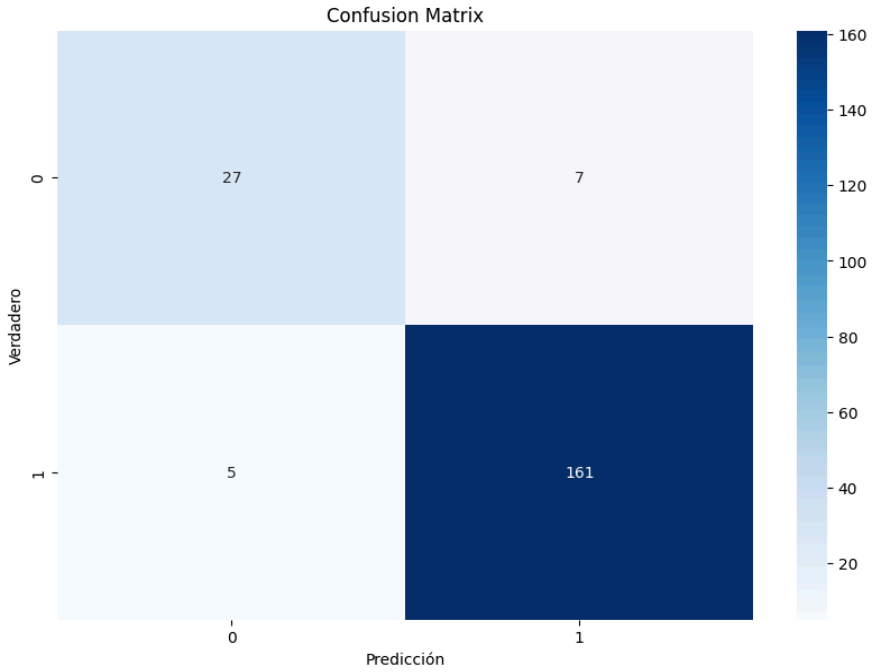


Imagen 45, Matriz de confusión SVM 1

De acuerdo con la matriz de confusión, se puede observar que la cantidad de falsos negativos y falsos positivos es significativamente inferior a la de verdaderos positivos y verdaderos negativos. Esto indica que, a pesar del fuerte desbalance en las clases, el modelo logra clasificar correctamente la mayoría de las instancias.

7.1.3. Máquinas de soporte vectorial 2 con balanceo y data augmentation

	precisión	recall	f1-score	support
0 (Sano)	0,9763313609	0,9537572254	0,9649122807	173
1 (Heilipus)	0,9520958084	0,9754601227	0,9636363636	163
accuracy	0,9642857143	0,9642857143	0,9642857143	0,9642857143
macro avg	0,9642135847	0,9646086741	0,9642743222	336
weighted avg	0,9645742328	0,9642857143	0,964293309	336

Tabla 2, Reporte de modelo de máquinas de soporte vectorial 2 con balanceo y data augmentation

Al comparar las métricas antes y después de aplicar técnicas como balanceo de datos y aumento de datos al modelo, se observa una mejora significativa en su desempeño. Los resultados muestran que

la precisión global del modelo ha aumentado al 96%. Este incremento se refleja especialmente en la precisión de la clasificación de imágenes de frutos sanos, alcanzando un 95% de precisión para ambas clases. Además, se ha notado una mejora notable en las medidas de recall y F1-score, lo que indica un rendimiento general muy superior del modelo después de aplicar estas técnicas. En resumen, los resultados sugieren que el balanceo de datos y el aumento de datos han tenido un impacto positivo en la capacidad del modelo para generalizar y clasificar con precisión las imágenes de aguacates, lo que se traduce en métricas mejoradas y más confiables.

7.1.3.1. Matriz de confusión máquinas de soporte vectorial 2 con balanceo y data augmentation

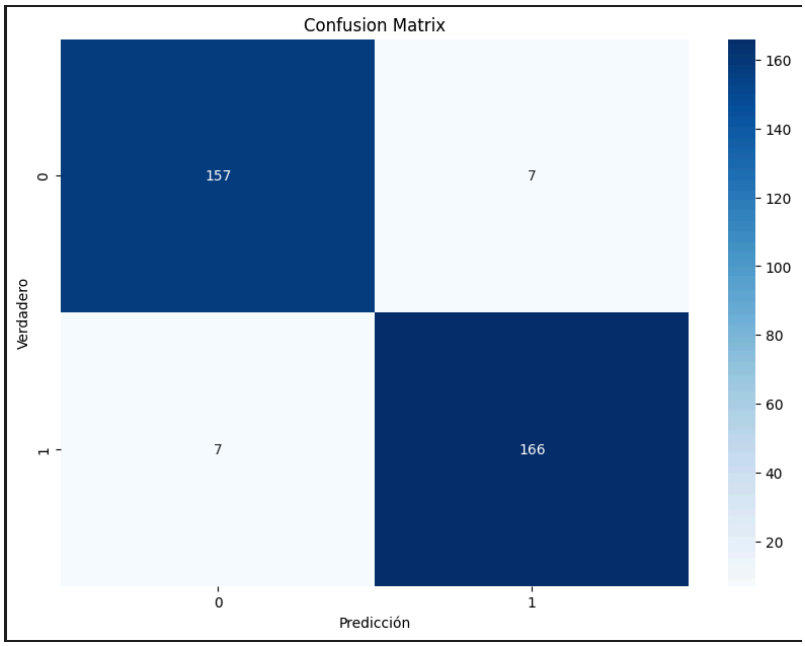


Imagen 46, Matriz de confusión SVM 2 con balanceo y data augmentation

De acuerdo con la matriz de confusión, el modelo ha predicho correctamente 166 instancias como pertenecientes a la clase sanos (0) (Verdaderos Positivos) y 157 instancias como no pertenecientes a la clase *Heilipus Laur*i (1) (Verdaderos Negativos). Los errores del modelo son relativamente bajos, con solo 7 instancias incorrectas predichas como no pertenecientes a la clase sanos (Falsos

Negativos) y 7 instancias incorrectas predichas como pertenecientes a la clase *Heilipus Lauri* (Falsos Positivos). Estos resultados indican que, a pesar del desbalance significativo en las clases, el modelo logra clasificar correctamente la mayoría de las instancias.

7.1.3.2. Curva ROC máquinas de soporte vectorial 2 con balanceo y data augmentation

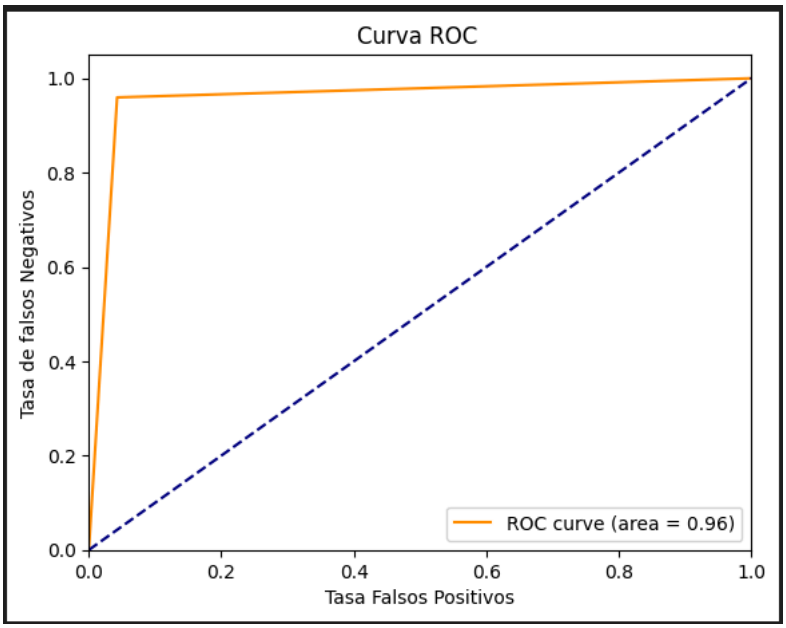


Imagen 47, Curva ROC máquinas de soporte vectorial 2

El área bajo la curva del modelo es bastante buena, de 0.96 muy cercana a 1 lo que indica que la tasa de falsos positivos es muy baja en la clasificación de clases, al combinarlo con los reportes de la clasificación garantiza que el modelo aprendió de manera adecuada.

7.1.3.3. Validación cruzada soporte vectorial 2 con balanceo y data augmentation

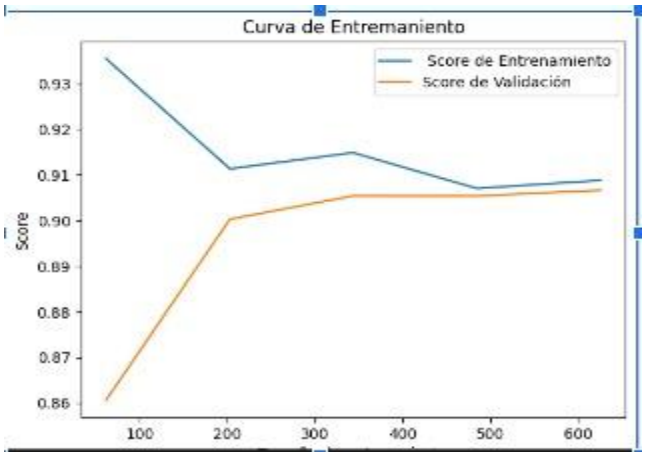


Imagen 48, Validación cruzada máquinas de soporte vectorial 2

De acuerdo con el gráfico, se observa que el modelo comienza con una precisión más alta para los datos de entrenamiento que para los datos de prueba. Sin embargo, el rendimiento del modelo se estabiliza con una buena precisión, lo que indica que no se sobreajuste. Esto se logró después de balancear las clases y aplicar aumentación de datos para incrementar la cantidad de imágenes de frutos de aguacate sanos, que inicialmente eran menos numerosas.

7.2. Red Neuronal Convolucional

7.2.1 Red Neuronal Convolucional 1

	precision	recall	f1-score	support
Heilipus	0,8705882353	0,8554913295	0,8629737609	173
Sano	0,1666666667	0,1851851852	0,1754385965	27
accuracy	0,765	0,765	0,765	0,765
macro avg	0,518627451	0,5203382573	0,5192061787	200
weighted avg	0,7755588235	0,765	0,7701565137	200

Tabla 3, Reporte red neuronal convolucional 1

De acuerdo con la tabla de reporte, se observa que la precisión global del modelo es del 77%. Sin embargo, la precisión por clase para la identificación de imágenes de frutos sanos es solo del 16%, lo cual es una métrica extremadamente baja para la clasificación de estos casos. Por otro lado, la

precisión para las imágenes con daño por la plaga *Heilipus laur*i, presenta una precisión del 87%. Estos resultados pueden explicarse debido a la gran proporción de desbalance entre las clases, lo que impide que el modelo aprenda adecuadamente las características de las imágenes de frutos de aguacate sanos.

7.2.1.2. Precisión del modelo red neuronal convolucional 1

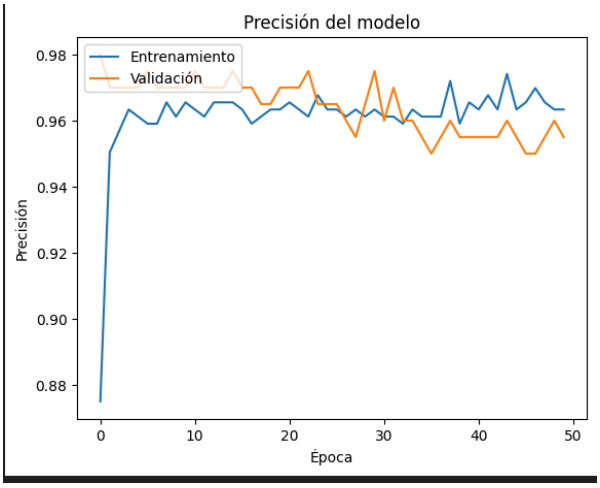


Imagen 49, Precisión del modelo red neuronal convolucional 1

Al realizar la validación cruzada del modelo, se observa que la precisión (accuracy) se mantiene con una tendencia similar tanto para los datos de entrenamiento como para los datos de prueba. Esto sugiere que el modelo ha aprendido correctamente y no debería tener problemas al momento de ingresar nuevos datos. Sin embargo, es importante tener en cuenta otros aspectos como el balance de clases y las métricas de precisión por clase para asegurar que el modelo funcione adecuadamente en todos los escenarios.

7.2.1.3. Pérdida del modelo red neuronal convolucional 1

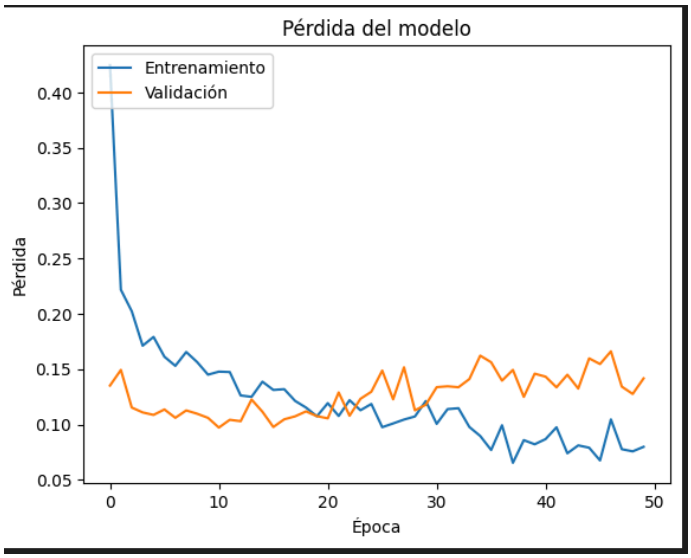


Imagen 50, Pérdida del modelo red neuronal convolucional 1

La gráfica de pérdida del modelo muestra que al inicio del entrenamiento, la pérdida para los datos de entrenamiento comienza en 0.4, mientras que para los datos de prueba empieza en 0.14. A medida que el entrenamiento avanza, ambas curvas de pérdida disminuyen y tienden a estabilizarse en una tendencia horizontal. La gráfica de pérdida sugiere que el modelo está aprendiendo de manera efectiva y que ha alcanzado un punto de estabilidad, lo cual es una buena indicación de que el modelo está listo para su uso en datos nuevos, siempre y cuando se hayan considerado adecuadamente otros factores importantes en la evaluación como el resto de las métricas de validación, sin embargo se observa que el modelo no presenta sobreajuste.

7.2.1.3. Matriz de confusión red neuronal convolucional 1

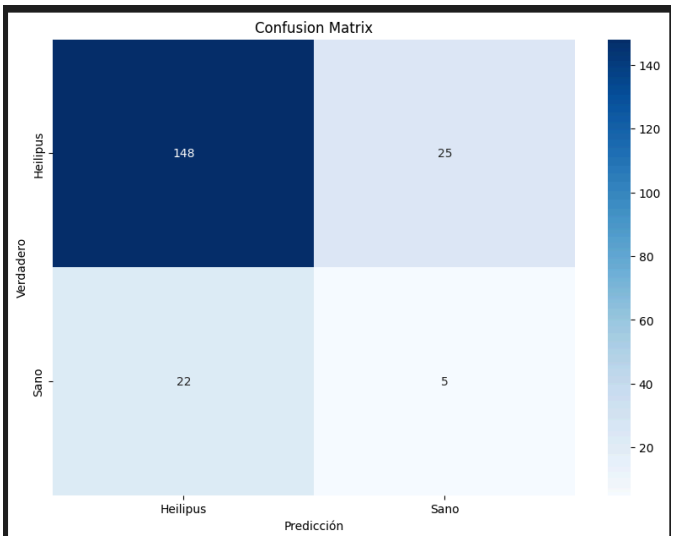


Imagen 51, Matriz de confusión red neuronal convolucional 1

De acuerdo con la imagen de la matriz de confusión, se observa que el modelo tiene un menor número de verdaderos negativos, lo que indica que tiene dificultades para clasificar correctamente las imágenes de aguacates sanos, clasificándolas erróneamente como imágenes de aguacates enfermos. Sin embargo, el modelo predice adecuadamente los verdaderos positivos, es decir, las imágenes de aguacates con daño. Esto sugiere que, aunque el modelo es efectivo en la identificación de frutas dañadas, necesita mejoras en la identificación de frutas sanas para reducir los falsos positivos.

7.2.2. Red neuronal convolucional 2

	Precisión	Recall	f1-score	support
Sano	0,95	0,9344262295	0,9421487603	183
Heilipus	0,9230769231	0,9411764706	0,932038835	153
accuracy	0,9375	0,9375	0,9375	0,9375
macro avg	0,9365384615	0,93780135	0,9370937976	336
weighted avg	0,9377403846	0,9375	0,9375451336	336

Tabla 4, Reporte red neuronal convolucional 2

El informe muestra que las métricas de precisión son excelentes, tanto para el modelo global como para cada clase individual, con una precisión del 98% en cada clase y en el modelo global. Además, se observa un incremento en las medidas de recall y F1-score, que también oscilan alrededor del mismo valor. Sin embargo, es necesario realizar una validación cruzada para verificar que el modelo no presenta problemas de sobreajuste.

7.2.2.1. Validación cruzada red neuronal convolucional 2

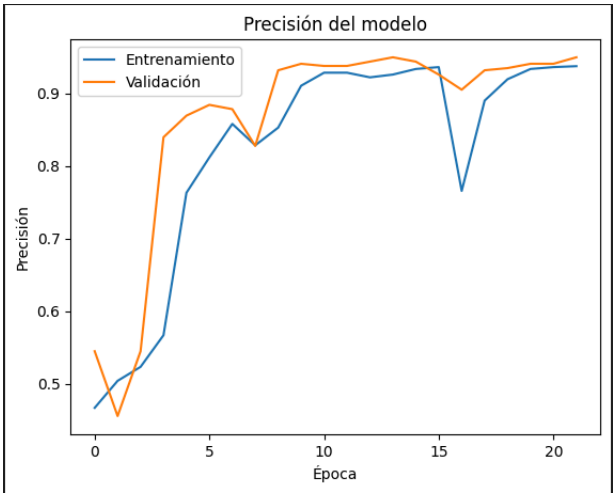


Imagen 52, Validación cruzada red neuronal convolucional 2

La gráfica muestra una subida pronunciada de las métricas desde aproximadamente 0.5 a valores que oscilan entre 0.8 y 0.9, con una tendencia horizontal, indica que el modelo ha aprendido bien y generaliza adecuadamente, demostrando un rendimiento sólido y consistente tanto en los datos de entrenamiento como en los de prueba.

La similitud en el comportamiento de las métricas tanto para los datos de entrenamiento como para los de prueba sugiere que el modelo generaliza bien y no está sobreajustado (overfitting).

7.2.2.2. Pérdida del modelo red neuronal convolucional 2

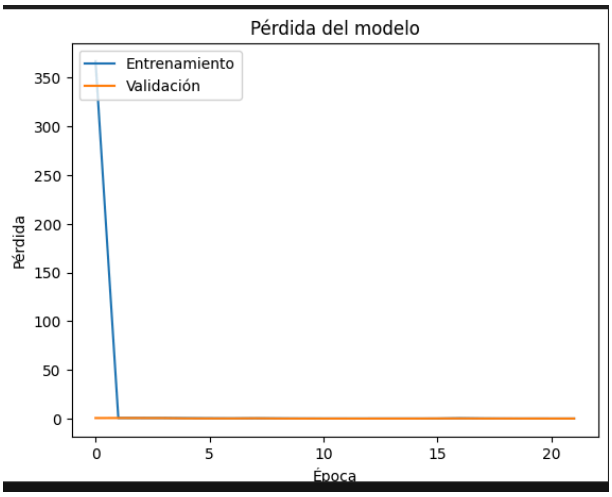


Imagen 53, Pérdida del modelo red neuronal convolucional 2

De acuerdo al gráfico de pérdida del modelo se puede evidenciar que hay un posible problema de sobreajuste, una pérdida de entrenamiento que desciende rápidamente y una pérdida de prueba que permanece horizontal sugiere que el modelo está sobreajustado. Es necesario tomar medidas para mejorar la generalización del modelo, asegurando que no solo aprenda las peculiaridades de los datos de entrenamiento, sino que también se desempeñe bien con datos nuevos.

7.2.2.3. Matriz de confusión red neuronal convolucional 2

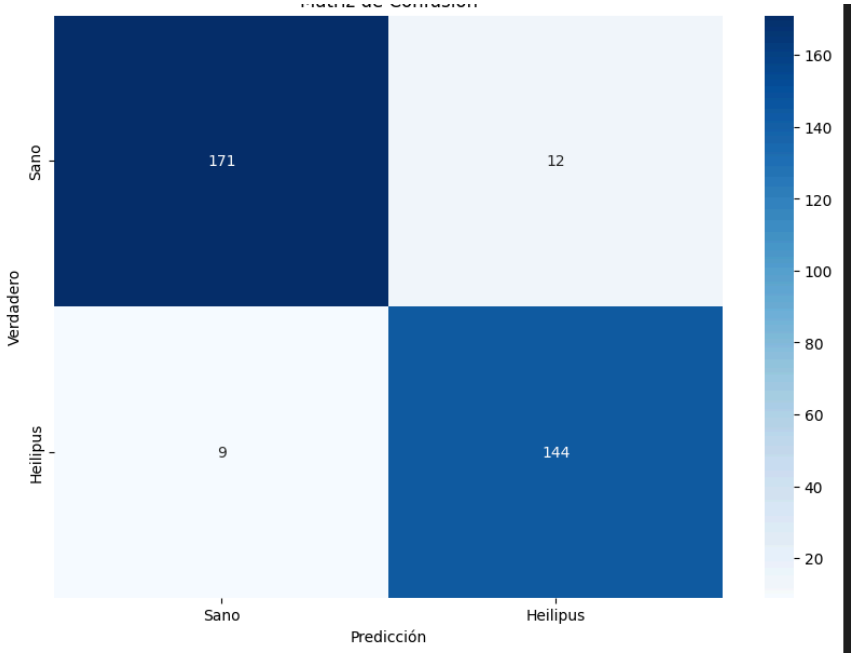


Imagen 54, Matriz de confusión CNN

En el gráfico de la matriz de confusión, se puede observar que, después de realizar el balanceo de clases y aplicar la técnica de aumento de datos para incrementar la cantidad de imágenes de la clase con menor representación, el modelo mejora en la clasificación de verdaderos negativos. Esto indica que el modelo está aprendiendo a clasificar adecuadamente las imágenes de aguacates sanos.

7.2.2.4. Curva ROC modelo red neuronal convolucional 2

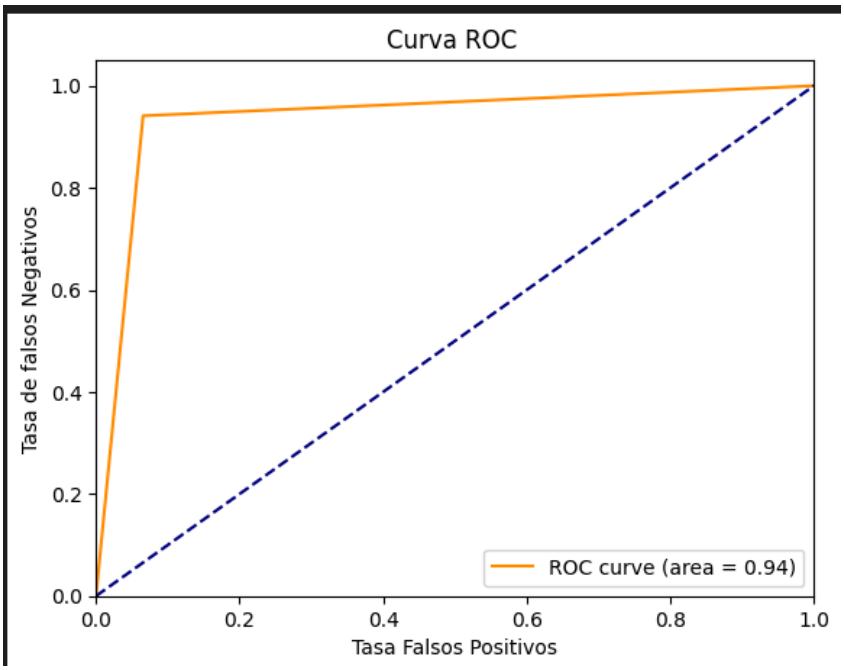


Imagen 55, Matriz de confusión CNN

La curva ROC de modelo, con un área bajo la curva del 94%, indica un rendimiento sólido en la capacidad de distinguir entre las clases positivas y negativas. Este valor sugiere que el modelo es capaz de realizar predicciones con una alta tasa de verdaderos positivos y una baja tasa de falsos positivos. Sin embargo, es importante tener en cuenta otros aspectos del modelo y su aplicación específica para una evaluación más completa de su rendimiento y utilidad.

7.5 API Reconocimiento de imágenes , Predicción CNN

La implementación del modelo contiene una api desarrollada en Python con el framework *Flask* , la cual realiza el consumo del modelo de la Red Neuronal Convolucional y se encarga de otorgar un Score de si el fruto tiene alguna enfermedad.

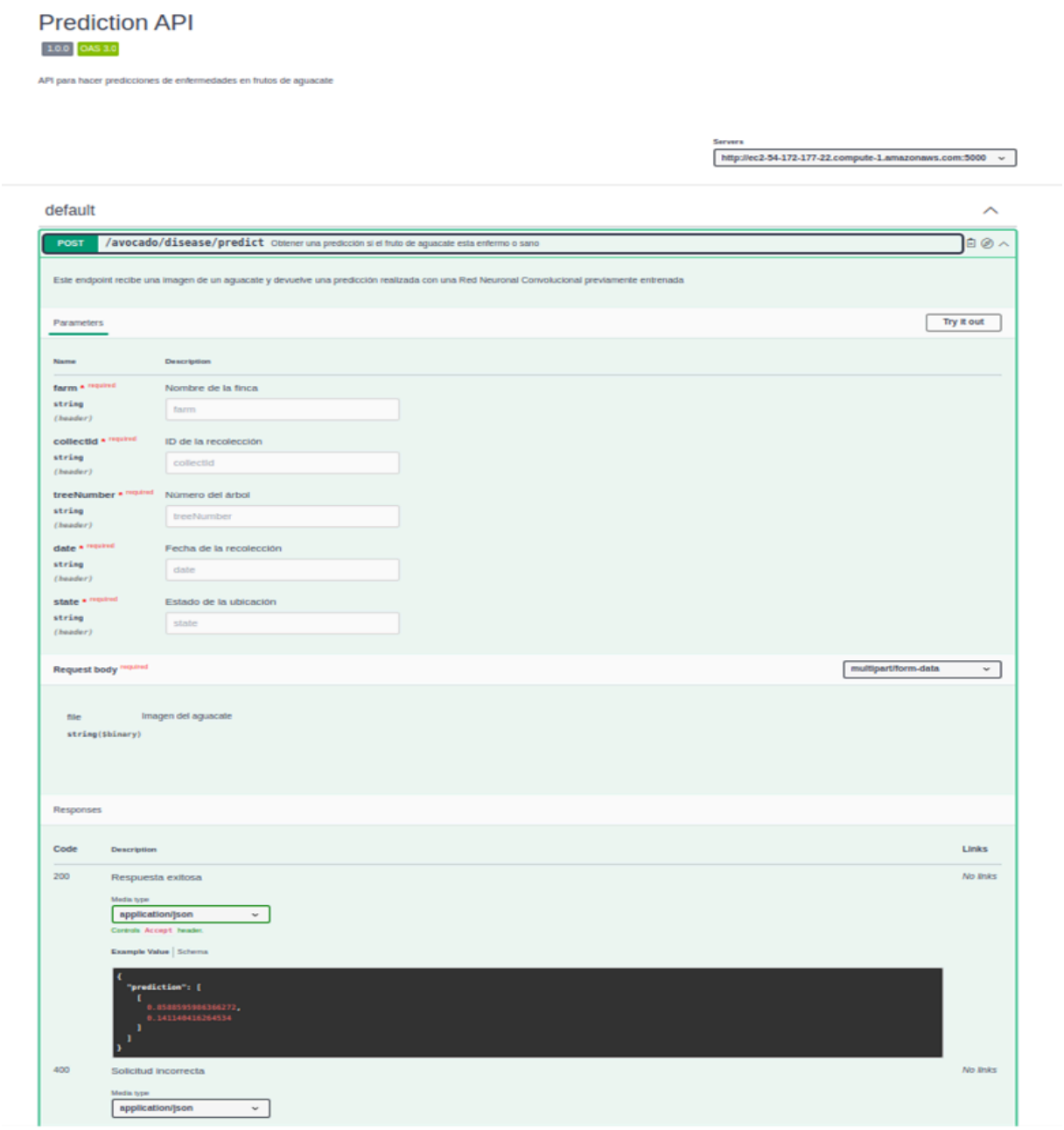


Imagen 56, Open Api Specification

El Open API Specification es el mecanismo idóneo para compartir a los consumidores el contrato de cómo debe invocar y qué esperar del resultado de la invocación de un recurso HTTP . Para el proyecto se expone un método POST llamado */avocado/disease/predict* el cual obtiene una predicción si el fruto de aguacate está enfermo o sano.

7.5.1. Desarrollo API (https://github.com/maucasco/pujc-advocato-clasification-api)

El desarrollo del API consiste en exponer un servicio REST por medio del Framework Flask de Python , el cual carga el modelo pre entrenado y evaluado en formato H5 como se puede apreciar en la imagen 36:

```
# Guardar el modelo en un archivo
modelo.save('/data/maestria/maestriasinpro/proyectodegrado/pujc-advocato-filter-project-mngr/modelo_cnn.h5')
```

Imagen 57, Open Api Specification

Previo a pasar la imagen al modelo primero se convierte a un Array de Numpy y luego se usa la misma clase *ProcessImage* usada para entrenar el modelo la cual se encarga de aplicar las mismas conversiones explicadas en la sesión 5.4 y 5.5 del documento. Una vez se realiza el preprocesamiento de la imagen se realiza la normalización de la imagen para que cumpla con las características requeridas para pasar a través de la red convolucional y poder obtener el score de Sano o Enfermo.

Para ejecutar el código se crea un ambiente virtual de python usando las librerías

- jsonpatch==1.32
- jsonpointer==2.0
- jsonschema==4.10.3
- numpy==1.26.4
- opencv-python==4.9.0.80
- pillow==10.2.0
- six==1.16.0
- tensorflow==2.16.1
- flask==3.0.3

```python
def load_model():
    model = tf.keras.models.load_model('modelo_cnn.h5')
    return model
model = load_model()
@app.route('/avocado/disease/predict', methods=['POST'])
def predict():
    try:
        if 'file' not in request.files:
            return jsonify({'error': 'No file uploaded'}), 400
        file = request.files['file']
        image = Image.open(file)
        image_np = np.array(image)
        procesador = ProcessImages()

        _, _, redimensionada = procesador.procesar(image_np, 200, 200)
        if redimensionada.shape[-1] == 1:  # Si la imagen tiene un solo canal
            redimensionada = cv2.cvtColor(redimensionada, cv2.COLOR_GRAY2RGB)  # Convertir a RGB

        imagen_redimensionada = Image.fromarray(np.uint8(redimensionada))
        ruta_redimensionada = os.path.join('', f'redimensionada_{file.filename}')
        imagen_redimensionada.save(ruta_redimensionada)
        # Convertir la imagen redimensionada a un arreglo de imágenes
        img_array = tf.keras.preprocessing.image.img_to_array(redimensionada)
        img_array = np.expand_dims(img_array, axis=0) / 255.0
        predictions = model.predict(img_array)
        return jsonify({'prediction': predictions.tolist()})
    except Exception as e:
        print (e)
        error_trace = traceback.format_exc()
        return jsonify({'error': 'Bad Request','error-trace':error_trace, 'message': str(e)}), 400
```

Imagen 58, Función de predicción /avocado/disease/predict

7.5.2. Despliegue API

Esta api fue desplegada en Amazon EC2 sistema operativo Ubuntu en modalidad de contratación a demanda, en una cuenta personal de los integrantes del proyecto con las siguientes características:

Instance type	**Platform**	**AMI name**
t2.medium	Ubuntu (Inferred)	ubuntu/images/hvm-ssd-gp3/ubuntu-noble-24.04-amd64-server-20240423
Number of vCPUs	**Storage**	**Memory**
2	8gb	4gb

Tabla 5, Tabla configuración API

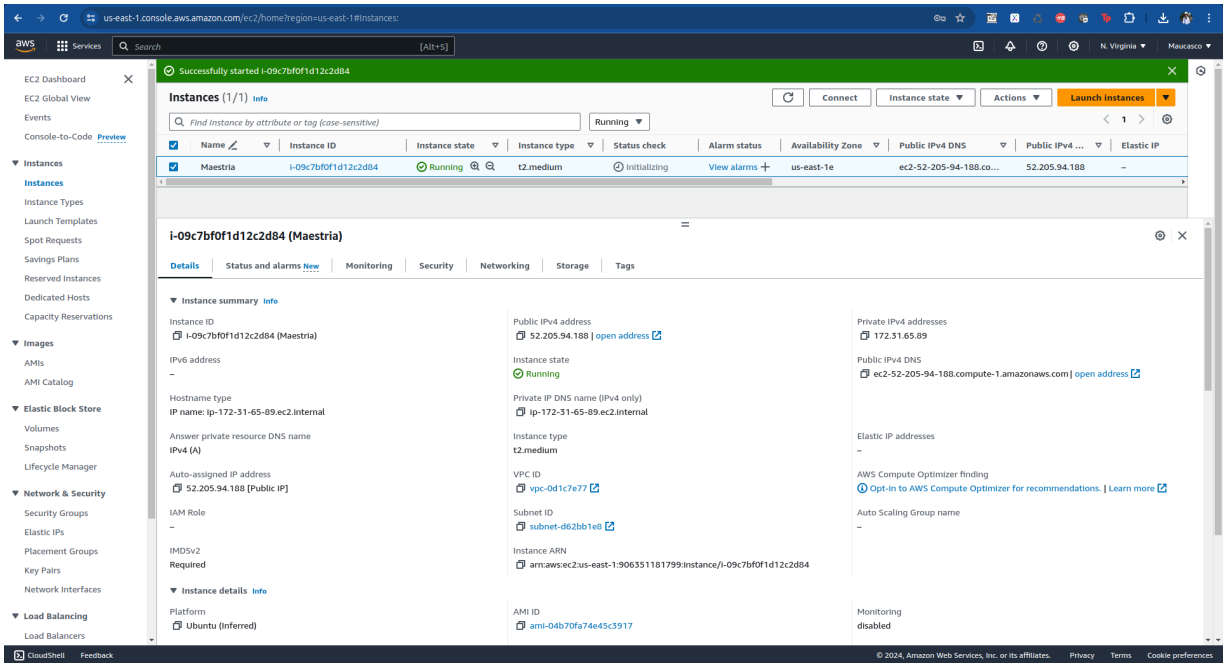


Imagen 59, Despliegue AWS

Al trabajar con Virtual Environments se facilitó el despliegue en el sistema operativo Ubuntu provisionado en nube ya que solo fue ejecutar la instalación con los comandos descritos a continuación

Comandos de aprovisionamiento

```
10  sudo apt install python3-pip
11  pip3
12  sudo apt install python3-pip
13  sudo apt update
14  sudo apt install python3-pip
15  source myenv/bin/activate
16  python3.12 -m venv env_layer
17  source ./env_layer/bin/activate
18  python3.12 -m venv env_layer
23  sudo apt-get install python
24  sudo apt-get install python3
25  sudo pip3 install -r requirements.txt
26  python3 -m venv myenv
27  sudo  apt install python3.12-venv
28  source myenv/bin/activate
29  python3.12 -m venv env_layer
30  source ./env_layer/bin/activate
31  pip install -r requirements.txt
```

19 sudo python3.12 -m venv env_layer

20 pip install -r requirements.txt

21 pip3 install -r requirements.txt

22 apt-get install python

32 /env_layer/bin/python run.py

33 /home/ubuntu/env_layer/bin/python run.py

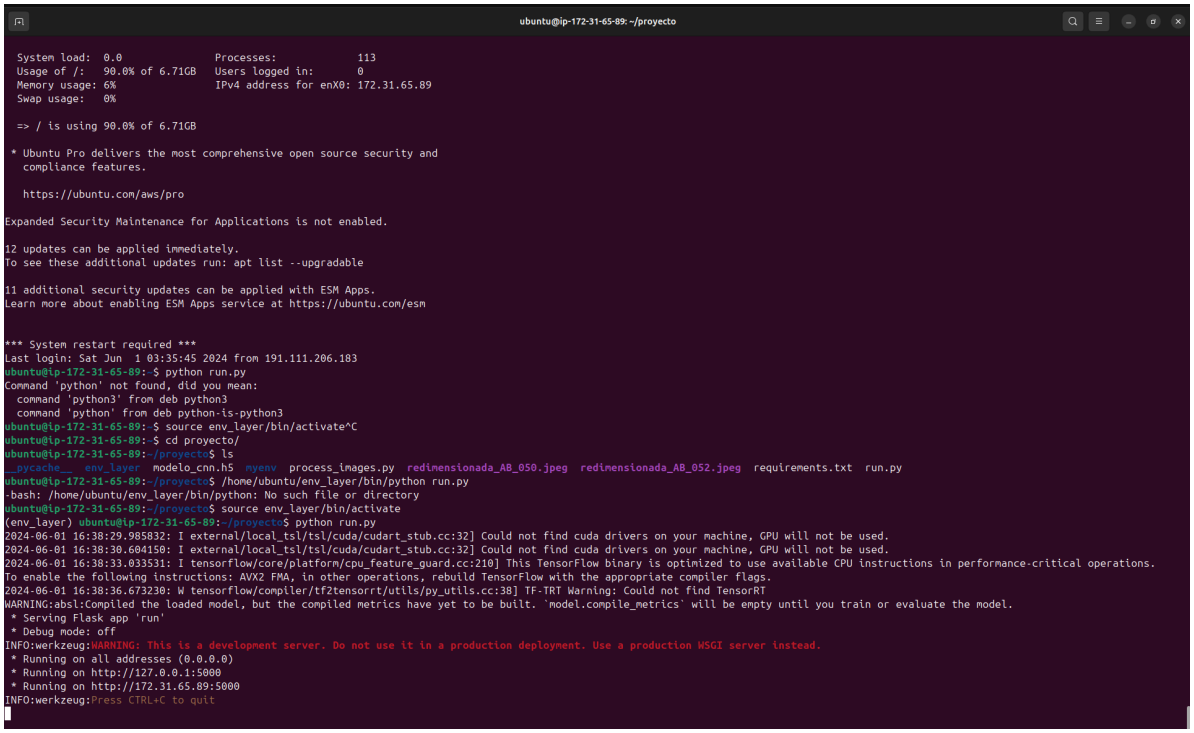


Imagen 60, Ambiente AWS

7.5.3. Pruebas API

Las pruebas se realizaron con imágenes AB que son imágenes Sanas y Heilipus que son enfermas, que no fueron parte del entrenamiento. Lo que se puede identificar en la siguiente tabla es que solo una de las imágenes fue mal clasificada Heilipus (322).jpg la cual estuvo en un 49% de posibilidades de ser una categorizada como Heilipus y un 51% de ser Sana.

Enfermo **Bueno**

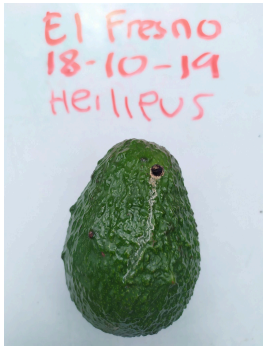


Imagen 61, Heilipus (322).jpg

Como se puede apreciar en la imagen es un fruto bastante sano excepto por el circulo superior negro, a la derecha esta un fruto sano , que aunque no tiene un circulo tan pronunciado si tiene puntos pequeños negros. Por eso es tan importante la profundidad de la red que aprende hasta de esos pequeños detalles.

imagen	predicted_class	prediction 0/0	prediction/0/1
AB_001.jpeg	Sano	0,1019	0,8981
AB_002.jpeg	Sano	0,0250	0,9750
AB_003.jpeg	Sano	0,1827	0,8173
AB_004.jpeg	Sano	0,0000	1,0000
AB_005.jpeg	Sano	0,0000	1,0000
AB_006.jpeg	Sano	0,0002	0,9998
AB_007.jpeg	Sano	0,0005	0,9995
AB_008.jpeg	Sano	0,0000	1,0000
AB_009.jpeg	Sano	0,0047	0,9953
AB_010.jpeg	Sano	0,0048	0,9952
AB_011.jpeg	Sano	0,0001	0,9999
AB_012.jpeg	Sano	0,0000	1,0000
AB_013.jpeg	Sano	0,0011	0,9989
AB_014.jpeg	Sano	0,0050	0,9950

AB_015.jpeg	Sano	0,0001	0,9999
AB_016.jpeg	Sano	0,0000	1,0000
AB_017.jpeg	Sano	0,0001	0,9999
AB_018.jpeg	Sano	0,0001	0,9999
AB_019.jpeg	Sano	0,0004	0,9996
AB_020.jpeg	Sano	0,0006	0,9994
AB_021.jpeg	Sano	0,0000	1,0000
AB_022.jpeg	Sano	0,0133	0,9867
AB_023.jpeg	Sano	0,0006	0,9994
AB_024.jpeg	Sano	0,0000	1,0000
AB_025.jpeg	Sano	0,0027	0,9973
AB_026.jpeg	Sano	0,0001	0,9999
AB_027.jpeg	Sano	0,0000	1,0000
AB_028.jpeg	Sano	0,0000	1,0000
AB_029.jpeg	Sano	0,0004	0,9996
AB_030.jpeg	Sano	0,0005	0,9995
AB_031.jpeg	Sano	0,0028	0,9972
AB_032.jpeg	Sano	0,0023	0,9977
AB_033.jpeg	Sano	0,0002	0,9998
AB_034.jpeg	Sano	0,0002	0,9998
AB_035.jpeg	Sano	0,0006	0,9994
AB_036.jpeg	Sano	0,0003	0,9997
AB_037.jpeg	Sano	0,0007	0,9993
AB_038.jpeg	Sano	0,0157	0,9843
AB_039.jpeg	Sano	0,0006	0,9994
AB_040.jpeg	Sano	0,0014	0,9986
AB_041.jpeg	Sano	0,0000	1,0000
AB_042.jpeg	Sano	0,0002	0,9998
AB_043.jpeg	Sano	0,0004	0,9996
AB_044.jpeg	Sano	0,0003	0,9997
AB_045.jpeg	Sano	0,0000	1,0000
AB_046.jpeg	Sano	0,0007	0,9993
AB_047.jpeg	Sano	0,0005	0,9995
AB_048.jpeg	Sano	0,0004	0,9996
AB_049.jpeg	Sano	0,0003	0,9997
AB_050.jpeg	Sano	0,0000	1,0000
AB_051.jpeg	Sano	0,0004	0,9996

AB_052.jpeg	Sano	0,0001	0,9999
AB_053.jpeg	Sano	0,0007	0,9993
Heilipus (163).jpg	Heilipus	0,6930	0,3070
Heilipus (318).jpg	Heilipus	0,7036	0,2964
Heilipus (319).jpg	Heilipus	0,9509	0,0491
Heilipus (320).jpg	Heilipus	0,9868	0,0132
Heilipus (321).jpg	Heilipus	0,9448	0,0552
Heilipus (322).jpg	Sano	0,4976	0,5024
Heilipus (323).jpg	Heilipus	0,5419	0,4581
Heilipus (324).jpg	Heilipus	0,9288	0,0712
Heilipus (325).jpg	Heilipus	0,8141	0,1859
Heilipus (326).jpg	Heilipus	0,6639	0,3361
Heilipus (327).jpg	Heilipus	0,8762	0,1238
Heilipus (328).jpg	Heilipus	0,8463	0,1537
Heilipus (329).jpg	Heilipus	0,8935	0,1065
Heilipus (330).jpg	Heilipus	0,9049	0,0951
Heilipus (466).jpg	Heilipus	0,8549	0,1451
Heilipus (68).jpg	Heilipus	0,8060	0,1940
Heilipus (69).jpg	Heilipus	0,5831	0,4169

Tabla 6, Pruebas con imágenes a través del API

8. Conclusiones y trabajos futuros

8.1. Conclusiones

- El tratamiento de imágenes requiere un estándar de tamaño de píxel y calidad, mientras que el brillo en las imágenes representa un desafío significativo y difícil de abordar. En este proyecto, se enfrentó un gran reto al tener que procesar las imágenes de manera diferente para llevarlas a un estándar único que permitiera un modelado adecuado.
- La disponibilidad de un repositorio limitado y desbalanceado de imágenes implicó la necesidad de aplicar nuevas técnicas como Data Augmentation y balanceo forzado. Aunque estas prácticas son válidas en un entorno académico, en el ámbito productivo es crucial contar con fuentes de datos adecuadas desde el inicio del proyecto. De lo contrario, el resultado final puede no ser satisfactorio debido a la falta de representatividad en los datos.
- Durante este trabajo de grado, se investigaron dos enfoques de clasificación para identificar imágenes de aguacates afectados por la plaga *Heilipus lauri* y aguacates Hass sanos. El primer enfoque se basó en el algoritmo de Máquinas de Vectores de Soporte (SVM), el cual mostró un rendimiento sólido desde el principio, incluso sin aplicar técnicas de balanceo de clases ni aumento de datos. Después de implementar estas técnicas para mejorar aún más su rendimiento, el modelo SVM exhibió un incremento significativo en sus métricas sin evidencia de sobreajuste.
- Por otro lado, se evaluó un segundo enfoque utilizando una red neuronal. Aunque este modelo mostraba una precisión inicial prometedora, se identificaron dificultades para predecir correctamente la clase minoritaria de aguacates sanos. Después de aplicar técnicas de balanceo de clases y aumento de datos para mejorar su desempeño, se observó un aumento en las métricas de rendimiento, pero también se manifestó un problema de sobreajuste.
- Estos resultados resaltan la importancia de considerar cuidadosamente las características de los datos y las técnicas de preprocesamiento en el desarrollo de modelos de clasificación. Mientras que el SVM demostró ser robusto y efectivo incluso con un conjunto de datos desbalanceado, la red neuronal mostró una mayor sensibilidad en la identificación de

características de los frutos, la distribución de clases y requirió un enfoque más cauteloso en el tratamiento de los datos.

- En resumen, este estudio destaca la relevancia de seleccionar apropiadamente algoritmos y técnicas de preprocesamiento en la clasificación de imágenes de aguacates, así como la necesidad de evaluar exhaustivamente el rendimiento y la generalización de los modelos para evitar problemas como el sobreajuste. Estos resultados proporcionan una base sólida para investigaciones futuras en el desarrollo de sistemas de clasificación de imágenes agrícolas.

8.2. Trabajos futuros

- La implementación actual puede predecir de manera adecuada si un fruto está Sano o Enfermo, pero la codificación del proyecto quedó abierta a extenderse a enfermedades distintas a *Heilipus Laur*i.
- El API está disponible como una instancia IaaS (infraestructura como servicio) en AWS por lo cual dejarla encendida generaría un costo aproximado de 60 dólares mensuales lo cual no es viable de sostener en un proyecto educativo. Se debe trabajar en un despliegue estilo serverless que le permita a los productores pequeños hacer uso de este recurso.
- Disponibilizar una página o una aplicación móvil que realice el consumo de la API, para que los productores en campo puedan hacer uso del servicio desde sus propios dispositivos móvil y de esta manera categorizar los frutos antes de disponibilizar los a los distribuidores.
- Mejora del Preprocesamiento de Datos: Se podría investigar y desarrollar técnicas más avanzadas de preprocesamiento de datos para mejorar la calidad y la representación de las imágenes de aguacates. Esto podría incluir métodos para manejar el brillo en las imágenes y técnicas más efectivas de aumento de datos para abordar el desbalance en el conjunto de datos.
- Recopilación de Datos Adicionales: Dado el desafío de trabajar con un repertorio limitado y desbalanceado de imágenes de aguacates, sería beneficioso recopilar más datos de manera

exhaustiva y equilibrada para entrenar modelos más precisos y confiables.

- Optimización de Hiperparámetros: Se pueden explorar técnicas de optimización de hiperparámetros para encontrar la configuración óptima de los modelos, lo que podría mejorar su rendimiento y capacidad de generalización.
- Evaluación de Otras Técnicas de Aprendizaje Automático: Además de SVM y CNN, sería interesante evaluar otros algoritmos de aprendizaje automático para la detección de enfermedades en imágenes de aguacates, como árboles de decisión, bosques aleatorios o clasificadores de vecinos más cercanos, para determinar cuál proporciona los mejores resultados.

9. Referencias Bibliográficas

[1] Cadena productiva del aguacate, Minagricultura, Articulo, 2020, En línea, https://sioc.minagricultura.gov.co/Aguacate/Documentos/2020-03-30%20Cifras%20Sectoriales.pdf (2023, 26 de Junio)

[2] Camilo Alvaro Velez, El cultivo de aguacate como alternativa y posibilidad de crear empresa, Tesis pregrado, Medellín, 2010, En línea,

https://repository.usta.edu.co/bitstream/handle/11634/47809/2010camiloalvarez.pdf?sequence=1, (2024, 28 de Mayo)

[3] E. Avendaño - A. Caicedo - M. Gonzaléz - D. Jaramillo, Resultados piloto de vigilancia fitosanitaria escama verde (Coccus viridis (Green) (Hemiptera: occidae)) pasador del fruto (Stenoma catenifer Walsingham (Lepidoptera: Elaschistidae)) y barrenadores (Heilipus lauri Boheman, H. sp. cerca pittieri Barber, H. trifasciatus (Fabricius) y H. elegans Guérin-Méneville (Coleoptera: Curculionidae)) en Antioquia , Subgerencia de Protección Vegetal Dirección técnica de Epidemiologia y Vigilancia Fitosanitaria , Antiquia, 2011, En línea, https://www.ica.gov.co/areas/agricola/servicios/epidemiologia-agricola/boletines/departamentales/2012/boletin_aguacate_antioquia.aspx, (2024, 28 de Mayo)

[4] C. Urrea y J. Cardona, Manejo integrado de las principales plagas y enfermedades en aguacate Hass 1 (Persea americana) en el departamento de Caldas, Tesis pregrado, Manizalez, 2020, En linea, https://repository.unad.edu.co/bitstream/handle/10596/38446/dosquebradas.pdf?sequence=3&isAllowed=y, (2024, 28 de Mayo)

[5] E. Avendaño - M. Gonzaléz - J. Galindo - J. Alrcon, Manejo fitosanitario del cultivo del aguacate Hass, Articulo,ICA, 2012, En linea, https://www.ica.gov.co/getattachment/4b5b9b6f-ecfc-46e1-b9ca-b35cc1cefee2/-, (2023, 20 de Junio)

[6] D. Salazar y M. Rojas, Establecimiento y manejo del cultivo de aguacate Hass para 25 familias productoras como alternativa de generación de empleo y mejoramiento de la calidad de vida en el municipio de Capitanejo, Santander, Tesis especialización, Malaga, 2018, En linea, https://repository.unad.edu.co/bitstream/handle/10596/25698/dmsalazard.pdf?sequence=1&isAllowed=y , (2024, 28 de Mayo)

[7] I. Ruiz - C. Garcia - D. Bravo - A. Quezada - S. Hernandez - J. Florencio , El barrenador grande de la semilla del Aguacate Heilipus lauri Boheman, Articulo, Laboratorio de Entomología y Acarología, 2012, En linea, https://www.gob.mx/cms/uploads/attachment/file/155685/Ficha_T_cnica_Helipus_lauri_EPF_2016_1_.pdf, (2024, 28 de Mayo)

[8] P. Mila - E. Gomez - Y. Jaime , Análisis exploratorio de datos a una base de datos de la biblioteca de la universidad de la Salle, Articulo, Universidad de La Salle, Bogota, 2019, https://acofipapers.org/index.php/eiei/article/download/196/189/376, (2023, 26 de Junio)

[9] A. Sandoval, Análisis de métodos y técnicas de Limpieza de Datos existentes y aplicación en un Sistema CRM para una institución educativa limeña, Tesis pregrado, Lima, Pontificia Universidad Católica del Perú, 2018, En línea, https://tesis.pucp.edu.pe/repositorio/bitstream/handle/20.500.12404/12619/SANDOVAL_LINARES_ANGEL_ANÁLISIS_MÉTODOS_TÉCNICAS.pdf?sequence=6&isAllowed=y, (2024, 27 de Mayo)

[10] M. Kraiem, Clasificación a partir de conjuntos de datos no equilibrados. Un marco para mejorar la aplicación de las estrategias de remuestreo,Tesis Doctoral, Salamanca, 2020, https://gredos.usal.es/bitstream/handle/10366/145514/Kraiem%2c%20Mohamed%20%28v.r%29.pdf?sequence=1&isAllowed=y, (2024, 27 de Mayo)

[11] S. Gonzalez - M. Lopez - E. Zavala - I. YAñez - J. Guerrero, Estudio comparativo de algoritmos de segmentación de piel usando atributos de color, Universidad Polit´ ecnica de Juventino Rosas, Articulo, 2016, En línea, https://rcs.cic.ipn.mx/2016_114/Estudio%20comparativo%20de%20algoritmos%20de%20segmentacion%20de%20piel%20usando%20atributos%20de%20color.pdf, (2024, 27 de Mayo)

[12] E. Vargas, Modelado basado en datos para la clasificación semiautomática de correspondencia electrónica: caso de estudio para la Administración Pública Colombiana, Tesis maestria, Escuela Colombiana de Ingeniería Julio Garavito, Bogota,2018, En línea, https://repositorio.escuelaing.edu.co/flip/index.jsp?pdf=/bitstream/handle/001/829/Vargas%20Antol%c3%adnez%2c%20Edwin%20Alberto%20-%202018%20.pdf?sequence=1&isAllowed=y, (2024, 26 Mayo)

[13] R. Garcia, Aprendizaje auto-supervisado para análisis de imágenes en dominios no convencionales, Universidad Autónoma de Madrid, Madrid, Tesis, 2021, En línea, https://repositorio.uam.es/bitstream/handle/10486/698397/lopez_garcia_rafael_tfg.pdf?sequence=1, (2024, 18 Marzo)

[14] J. Benavides y J. Franco, Universidad Católica de Colombia, Bogota, Tesis pregrado, 2021, En línea, https://repository.ucatolica.edu.co/server/api/core/bitstreams/feb71e6c-7ac0-404e-8e13-7ee5f6731745/content, (2024, 20 Febrero)

[15] J. Reséndiz Las máquinas de vectores de soporte para identificación en línea, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional,Tesis maestría, México DF, 2006, En línea, https://www.ctrl.cinvestav.mx/~yuw/pdf/MaTesJAR.pdf, (2024, 18 de Marzo)

[16] OpenCV Image Processing in OpenCV, En linea, https://docs.opencv.org/4.x/d9/df8/tutorial_root.html, (2024, 15 de abril)

[17] Fin - PA - Dmh - Lof,Inteligencia artificial predice con alta precisión plagas en aguacate Hass, Agencia UNAL: Universidad Nacional de Colombia, Agencia UNAL, Artículo, 2023. http://agenciadenoticias.unal.edu.co/detalle/inteligencia-artificial-predice-con-alta-precision-plagas-en-aguacate-hass, (2023, 26 de Junio)

[18] S. R. Rupanagudi, Ranjani B. S., P. Nagaraj, V. G. Bhat and Thippeswamy G, "A novel cloud computing based smart farming system for early detection of borer insects in tomatoes," 2015 International Conference on Communication, Information & Computing Technology (ICCICT), Mumbai, India, 2015, pp. 1-6, doi: 10.1109/ICCICT.2015.7045722.

[19] N. Saranya, L. Pavithra, N. Kanthimathi, B. Ragavi and P. Sandhiyadevi, "Detection of Banana Leaf and Fruit Diseases Using Neural Networks," 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2020, pp. 493-499, doi: 10.1109/ICIRCA48905.2020.9183006.

[20] F, Alvi, Why You Need To Start Learning OpenCV in 2024! OpenCV, En linea, https://opencv.org/blog/learning-opencv/, (2024, 22 de Enero)

[21] Ross Ihaka, A Brief History R: Past and Future History, Statistics Department, The University of Auckland, Auckland, New Zealand, En linea,

https://cran.r-project.org/, (2024, 22 de Enero)

[22] Changelog - Python Documentation, En Línea, https://docs.python.org/3.12/whatsnew/changelog.html, (2024, 22 de Enero)

[23] D. Jeffrey, TensorFlow: Open source machine learning It is machine learning software being used for various kinds of perceptual and language understanding tasks, 2015,En línea, https://www.tensorflow.org/?hl=es-419, (2024, 22 de Enero)

[24] C. François, keras.io, 2018, En línea, https://keras.io/about/, (2024, 22 de Enero)

[25] Data augmentation, TensorFlow Core, En linea, https://www.tensorflow.org/tutorials/images/data_augmentation, (2024, 22 de Enero)

[26] Introducción a Support Vector Machine (SVM), En Línea, https://la.mathworks.com/discovery/support-vector-machine.html, (2024, 22 de Enero)

[27] J. Naizaque - E. Bermudez - J. Vargas, Modelo CNN para la clasificación de presencia de ruido en resonancias del cerebro humano,Tesis,2023,

https://repository.javeriana.edu.co/bitstream/handle/10554/64951/attachment_7_Anexo-7---Modelo-CNN.pdf?sequence=8&isAllowed=y, (2023, 23 de Mayo)

[28] E. Gangotena, Desarrollo de un sistema distribuido para la clasifiación de fichas basado en imagenes lego subsistema de clasificación,Escuela Politecnica Nacional,Tesis,Quito, 2022, En linea, https://repository.udistrital.edu.co/bitstream/11349/27950/1/VasquezRomeroDiegoFederico2020.pdf , (2024, 22 de Enero)

[29] M. Catalán, Análisis y modelización de los procesos de aprendizaje profundo, Universitat Jaume I, Tesis doctorado, Castellano de la Plana, En línea, 2024

https://www.tdx.cat/bitstream/handle/10803/690425/2024_Tesis_Catalán%20Carbó_Mar.pdf?sequence=1&isAllowed=y, (2024, 15 de abril)

[30] A. Rojas - R. Salazar - L. Miranda - W. Ojeda, Algoritmo Adam en la inteligencia artificial, Universidad Autónoma Chapingo, México DF, Artículo, En línea, 2021,

[31] Conceptos básicos de ayuda de CRISP-DM, En linea, 2021, https://www.ibm.com/docs/es/spss-modeler/saas?topic=dm-crisp-help-overview (2023, 15 de Agosto)

[32] C. Estada, Un filtro para la eliminación de ruido mediante correlación espacial en imágenes, Tesis maestria, Centro de investigación en matemáticas a.c., Guadalajara, En línea, 2021, https://cimat.repositorioinstitucional.mx/jspui/bitstream/1008/280/2/TE%20405.pdf, (2024, 25 de Mayo)

[33] What is Computer Vision? | IBM. (n.d.). https://www.ibm.com/topics/computer-vision

[34] Aumento de datos, TensorFlow Core, https://www.tensorflow.org/tutorials/images/data_augmentation?hl=es-419

[35] H. Mejia - J. ARcila - V. Tuesta - E.Rodriguez, Avocado-DB: Dataset of Hass Avocado Images with Seed, Peel and Pulp Weights, Universidad Señor de Sipán, Articulo, En línea, 2023, https://data.mendeley.com/datasets/b2d83zft4s/1, (2024, 10 de Febrero)

[36] GridSearchCV, scikit-learn, En línea,2024, https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html, (2024, 10 de Febrero)

[37] Imagen 1, Barrenador de la semilla del aguacate (Heilipus sp.), En línea, 2017, https://www.citricaldas.com.co/barrenador-de-la-semilla-del-aguacate-heilipus-sp/, (2024, 25 de Mayo)

[38] Imagen 3, Fruto Afectado por Heilipus lauri, Redagricola, Bgota, En linea,https://redagricola.com/tag/heilipus-lauri/, (2024, 25 de Mayo)

[39] Imagen 4, Máquina de soporte vectorial, MathWorks, En línea, 2024 https://la.mathworks.com/discovery/support-vector-machine.html, (2024, 25 de Mayo)

[40] Imagen 5, Red Neuronal Convolucional, researchgate, En línea, 2024, https://www.researchgate.net/figure/2-Illustration-of-the-corrispondence-between-the-areas-associated-with-the-primary_fig7_317679065, (2024, 25 de Mayo)

[41] Imagen 6, Capa Convolucional, dsotilccama, En línea, 2022, https://rpubs.com/dsotilccama/CNN_Markdown, (2024, 25 de Mayo)

[28] Imagen 7, Capas de Agrupamiento Promedio Pooling

[29] Imagen 8, Capas de Agrupamiento Maxpooling

[42] Imagen 9, Capa de apalancamiento, frogamesformacion, En línea,2020, https://cursos.frogamesformacion.com/pages/blog/la-guia-definitiva-de-las-redes-neuronales-convolucionales, (2024, 28 de Mayo)

[43] Imagen 10, Fully Connected, animalia-life, En línea,2023, https://ita.animalia-life.club/reti-neurali, (2024, 28 de Mayo)

[44] Imagen 11, Función de Activación - ReLU, paperswithcode, En línea, 2017, https://paperswithcode.com/method/relu (2024, 28 de Mayo)

[45] Imagen 12, Función de Activación - Sigmoide, paperswithcode, En línea, 2017, https://paperswithcode.com/method/sigmoid-activation, (2024, 28 de Mayo)

[46] Imagen 13, Función de Activación - Tanh, paperswithcode, En línea, 2017, https://paperswithcode.com/method/tanh-activation, (2024, 28 de Mayo)

[47] Imagen 14, Algoritmo de optimización - SGD,Satwik Muthappa, En línea, 2020, https://www.linkedin.com/pulse/stochastic-gradient-descent-sgd-satwik-muthappa-p-ph-d-/, (2024, 28 de Mayo)

[48]Imagen 15, Algoritmo de optimización - Adam,Universidad Autonoma de Chapingo, Articulo, 2021, https://www.riego.mx/congresos/comeii2021/files/ponencias/extenso/COMEII-21005.pdf, (2024, 28 de Mayo)

[49] Imagen 30, Ejemplo ejecución Data Augmentation, data.mendeley, 2023, En línea, https://data.mendeley.com/datasets/b2d83zft4s/1, (2024, 5 de Febrero)

10. Anexos

Anexos 1 - Código Entrenamiento del modelo

Se implementó un proyecto de código hecho en python para las operaciones comunes a los modelos y júpiter notebooks en donde se tiene la implementación de los modelos. La estructura de código en donde se separan las responsabilidades de las clases generadas.

Repositorio Github: https://github.com/maucasco/pujc-advocato-filter-project-mngr

- assets/: Directorio que contiene las imágenes para ejecutar el entrenamiento
- edu/: Directorio principal del proyecto con los módulos de Python.
    - create_dataset.py: Script para crear el conjunto de datos.
    - data_augmentation.py: Script para aumentar el conjunto de datos mediante técnicas de aumento de datos.
    - extract_properties.py: Script para extraer propiedades de las imágenes.
    - process_images.py: Script para preprocesar las imágenes.
    - show_images.py: Script para visualizar las imágenes.
- cnn_final.ipynb: Notebook con la implementación final de la CNN.
- cnn_imagedatagenerator.ipynb: Notebook que utiliza ImageDataGenerator para la CNN.
- install.sh: Script para instalar dependencias.
- installDependencies.sh: Script para instalar dependencias específicas.
- requirements.txt: Archivo con las dependencias del proyecto.
- run copy 2.ipynb: Copia de un notebook de ejecución.
- run copy 3.ipynb: Otra copia de un notebook de ejecución.
- svm_final.ipynb: Notebook con la implementación final del SVM.
- svm_pipeline_model.h5: Modelo entrenado de la SVM.
- svm.ipynb: Notebook principal para el SVM.
- svmsinbal.ipynb: Notebook del SVM sin balanceo.

Anexos 2 - Código del api que hace uso del modelo

Repositorio Github: https://github.com/maucasco/pujc-advocato-clasification-api

- prediction-api-1.0.0-resolved.json: Archivo JSON con la configuración de la API de predicción.
- modelo_cnn.h5: Archivo del modelo entrenado de la CNN.
- process_images.py: Script para el preprocesamiento de imágenes.
- requirements.txt: Archivo con las dependencias del proyecto.
- run.py: Script principal para ejecutar el modelo.
- test.sh: Script para ejecutar las pruebas contra el api.

Anexos 3- Curl Consumo de APIs

```
curl --location 'http://localhost:5000/avocado/disease/predict' \
--header 'farm: san-isidro' \
--header 'collectId: 345' \
--header 'treeNumber: 50' \
--header 'date;' \
--form 'file=@"/home/maucasco/Downloads/drive-download-20240601T021149Z-001/Heilipus (322).jpg"'
```