UNIVERSIDAD AUTÓNOMA DE SINALOA
Facultad de Informática Culiacán
Facultad de Ciencias de la Tierra y el Espacio
Posgrado en Ciencias de la Información

RECONOCIMIENTO DE PATRONES PLANTA-PATÓGENO USANDO VISIÓN POR COMPUTADORA MULTIESPECTRAL

Como requisito para obtener el grado de:
MAESTRO EN CIENCIAS DE LA INFORMACIÓN
Presenta:

Lic. Angélica Sarahy Trujillo López

Directores:
Dr. Jesús Roberto Millán Almaraz
Dr. Arturo Yee Rendón

Culiacán, Sinaloa, Julio 2023

Dedicatoria

A mis padres:
Juan Carlos Trujillo Rocha y Emilia Angélica López Cárdenas por todo su apoyo, amor e impulso a desarrollar el potencial que veían en mí.

A mi pareja:
Jesús Alejandro Inzunza Ibarra por su gran apoyo, paciencia, amor y Por brindarme esa chispa que lo caracteriza.

A mis hermanos:
Ximena Trujillo López, Carlos Trujillo López y Yahir Trujillo López porque más mis hermanos son mis bros y siempre será así.

Agradecimientos

A la Universidad Autónoma de Sinaloa y al Posgrado en Ciencias de la Información, que han sido mi casa de estudios y han contribuido a mi crecimiento académico y personal.

A CONACyT por otorgarme la beca que me permitió dedicarme a mis estudios de maestría.

A mis directores de tesis, el Dr. Jesús Roberto Millán Almaraz y el Dr. Arturo Yee Rendón, por compartir sus conocimientos, comprensión y apoyo durante el desarrollo de mi tesis. Han sido pieza fundamental en mi desarrollo académico.

Asimismo, a los profesores del posgrado por compartir sus conocimientos con tanta dedicación.

A mis compañeros de generación, Karen Romero, Martín Galaviz, Osiris Chávez, Ángel Martínez y Jacqueline Samano. A pesar de los desafíos que enfrentamos durante la pandemia, hemos sido grandes compañeros superando las dificultades y apoyándonos mutuamente en nuestro camino académico.

Resumen

La necesidad de evaluar y analizar grandes volúmenes de datos, como lo es en los problemas de clasificación, surge de diversas áreas de conocimiento como lo son la física, medicina, biología, ingeniería civil, entre otras. La clasificación supervisada es un tipo de aprendizaje de máquina en la que los objetos (instancias a clasificar) se clasifican dentro de clases definidas de manera previa. El aprendizaje de máquina es un área de la Inteligencia Artificial (IA), la cual se encarga del desarrollo de algoritmos computacionales, los cuales, son capaces de inferir y deducir conocimiento a partir de un conjunto de datos, con la finalidad de obtener modelos capaces de evaluar otros datos similares de los cuales no se contaba con información previa.

En los años más recientes, el gran auge del aprendizaje de máquina en todas las áreas del conocimiento es evidente. El aprendizaje de máquina ha sido el área de elección para desarrollar aplicaciones en reconocimiento de patrones, identificación y clasificación de objetos, entre otras.

En el caso del aprendizaje supervisado, se generan modelos que utilizan un conjunto de instancias etiquetadas para relacionar los datos que entran(características) con los que salen(etiquetas), o, en otras palabras, los modelos utilizan un conjunto de instancias de entrenamiento para aprender a relación entre una entrada y su salida deseada. Este tipo de aprendizaje puede tener entonces dos tipos de modelo en función de su salida. Si su salida es un valor continuo, estamos hablando de un modelo de regresión, en cambio, si su salida es un valor categórico, entonces es un modelo de clasificación.

Por otra parte, en aprendizaje no supervisado, se generan modelos que utilizan conjuntos de instancias no etiquetadas con el objetivo de aumentar el conocimiento estructural de los conjuntos. Por ejemplo, agrupar los datos según su similitud, extraer la estructura interna de forma en que se distribuyen los datos en el espacio original, o simplificar la estructura de

los datos manteniendo solamente sus características fundamentales.

La finalidad de este trabajo de tesis es desarrollar un sistema de identificación y clasificación de enfermedades en cultivos de tomate, empleando una metodología de visión por computadora para el procesamiento de imágenes y, la aplicación de técnicas de aprendizaje de máquina para la creación de modelos de clasificación. Esta propuesta estará enfocada en un problema en particular, el virus rugoso del tomate (ToBRFV).

El virus rugoso del tomate (ToBRFV) es un virus relativamente emergente con grandes afectaciones en la producción de tomate. Las investigaciones con relación a este virus han estado enfocadas únicamente en el área de la genética del tomate y su afectación por virus rugoso del tomate (ToBRFV), dejando al área de las ciencias de la información de lado. En consecuencia, este trabajo de investigación es de vital importancia para el área de la agricultura, pero de igual manera para el área de las ciencias de la información al ser un problema que no ha sido explorado.

Abstract

The need to evaluate and analyze large amounts of data, such as classification problems, arises from various areas of knowledge such as physics, medicine, biology, civil engineering, among others. Supervised classification is a kind of machine learning in which objects(instances to classify) are classified into predefined classes. Machine learning is an area of Artificial Intelligence (AI), which is responsible for the development of computational algorithms. These algorithms are capable of infer and deduce knowledge from a set of data, in order to obtain models able to assess other similar data for which no prior information was available.

In recent years, the great heyday of machine learning in all areas of knowledge is evident. Machine learning has been the area of choice to develop applications in pattern recognition, identification and objects classification, among others.

In the case of supervised learning, models that use a set of labeled instances to relate the input data (features) with the output data (labels) are generated. In other words, the models use a set of training instances to learn the relationship between an input and its desired output. This type of learning can then have two types of model depending on its output. If its output is a continuos value, we are talking about a regression model. Whereas if its output is a categorical value, then it is a classification model.

On the other hand unsupervised learning models generate models that use sets of unlabeled instances with the aim of increasing the structural knowledge of the sets. For example, grouping the data according to their similarity, extracting the internal structure of the way the data is distributed in the original space, or to simplify the structure of the data keeping only its fundamental characteristics.

The aim of this thesis work is to develop a system for the identification and classification of diseases in tomato and pepper crops. It will be do-

ne through a computer vision methodology for image processing and the application of machine learning techniques to create classification models. In particular, this proposal will be focused on a particular problem, the Tomato brown rugose fruit virus (ToBRFV).

Tomato brown rugose fruit virus (ToBRFV) is a relatively emerging virus with major effects on tomato production. Research related to this virus has been focused only on the area of tomato genetics and its affectation by tomato brown rugose fruit virus (ToBRFV), leaving the area of information science aside. Therefore, this research work is of vital importance for the agriculture, but equally for the information science as it is a problem that has not been explored.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Existen problemas de la vida cotidiana que son complejos y necesitan ser modelados y analizados de una manera correcta, como la detección de enfermedades, tanto en el área de la medicina como en el área de la agricultura. De ahí la necesidad de contar con herramientas computacionales que sean capaces de agilizar el análisis de estos problemas.

En particular en el área de la agricultura, cuando alguna enfermedad se presenta en una planta, ya sea en la hoja, raíz, tallo, o fruto, en muchas ocasiones se debe remover de raíz la planta debido a que es muy probable que contamine todo el cultivo. Los efectos que las enfermedades causan en las plantas son problemas importantes debido a que disminuyen el índice de producción y la calidad del producto agrícola, además de las pérdidas económicas que generarían. Por ello, un análisis para la detección temprana de las enfermedades en la agricultura es fundamental para no interrumpir la producción sostenible del cultivo (Hanssen et al., 2010). Al proceso que engloba el conjunto de técnicas orientadas a optimizar el uso de los insumos agrícolas y la disminución de pérdidas a partir de estos análisis de enfermedades se le denomina agricultura de precisión (Roel and Terra, 2013). El ciclo general que compone a la agricultura de precisión, consta de los siguientes procesos, primero una acción en el campo inicial (siembra), seguido de una recolección de datos (una vez la planta comienza a crecer), un procesamiento e interpretación de los datos recolectados previamente, una toma de decisiones con base en el procesamiento e interpretación y por último, una nueva actuación en el campo con base en estas decisiones. Este proceso lo podemos ver ilustrado en la Figura 1.1.

Cada uno de los procesos que componen la agricultura de precisión tienen sus respectivas variaciones de acuerdo al problema que se requiere tratar. Se pueden tratar problemas de nutrientes, de riego, de plagas, de

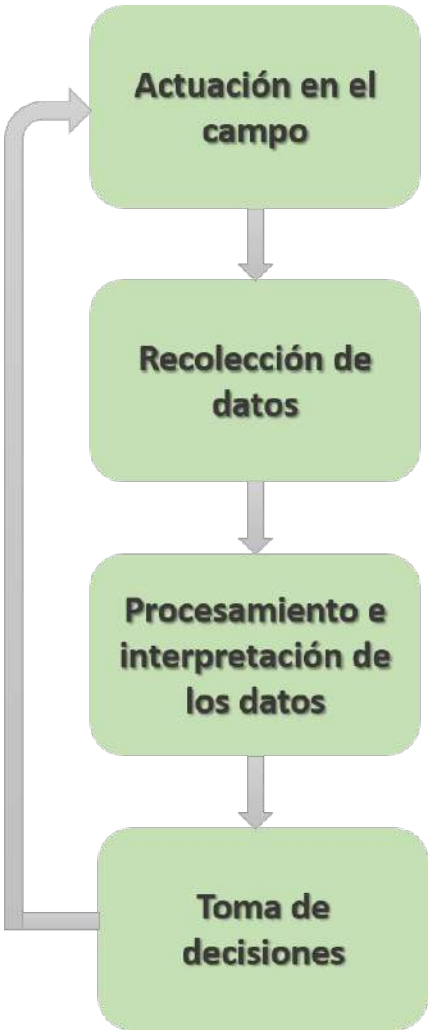

Figura 1.1: Diagrama del ciclo de la agricultura de precisión.

enfermedades, entre otros más. En particular, para las enfermedades en plantas existen diversas metodologías para su control.

En particular, en el cultivo de tomate, las enfermedades virales son notoriamente difíciles de controlar, dado la aparición continua de nuevas cepas de virus existentes o virus nuevos. Los virus tienen un gran potencial para evolucionar y adaptarse rápidamente bajo presión de selección natural (Hanssen et al., 2010).

El virus rugoso del fruto del tomate (ToBRFV) apareció en los plantíos de tomate en Jordania en 2015, actualmente este virus ha producido brotes en diferentes países del mundo, por ejemplo en China, España, México, entre otros (Virol et al., 2015). En territorio mexicano se identificó por primera vez en el año 2018 (Cambrón-Crisantos, 2018).

El virus ToBRFV es un patógeno que puede infectar tomate y pimien-

to. El principal cultivo afectado por el ToBRFV es el tomate. Este virus tiene la capacidad de superar todas las resistencias genéticas conocidas a los tobamovirus y causa síntomas severos en los frutos, lo cual lo hace muy problemático (Euroseeds, 2020). El virus se transmite por contacto, de ahí su gran peligrosidad. En las hojas, la infección se manifiesta mediante la aparición de secciones amarillentas o secas, manchas en variaciones de tono y color y mediante el estrechamiento de las mismas. En ocasiones muestran completo deterioro en la sección de tallo de la planta que sostiene el fruto. Por otro lado, en los frutos pueden aparecer decoloraciones, deformaciones y lesiones cafés que muestran grave deterioro. Debido a estos efectos, en poco tiempo se ha convertido en uno de los principales problemas a nivel nacional en el cultivo del tomate y se está en continua búsqueda para encontrar métodos tanto para su control como para su detección.

La detección de enfermedades en plantas usualmente se realiza mediante técnicas visuales a través de la exploración, sin embargo, estas exploraciones son de alto costo porque las personas capacitadas para realizarlas tienen un gran nivel de experiencia y conlleva extensa mano de obra. Además, puede no ser una detección correcta del padecimiento. Esto debido a que utilizar técnicas de detección visual comúnmente se puede ver comprometida, resultado de la dificultad de diferenciar los síntomas que muchas veces son muy similares, aun siendo provocados por diferentes padecimientos. Por otro lado, la visión por computadora ha evolucionado hasta un punto en el que incluso las alteraciones visuales más leves en el color y la morfología de la planta pueden detectarse usando la tecnología adecuada (Arnal Barbedo, 2019). Por ello, la visión computadora hoy es de las principales herramientas en la agricultura de precisión y más específicamente en la detección de enfermedades. El área de la visión por computadora que da solución al problema de detección de virus es el aprendizaje de máquina.

1.1. Planteamiento del Problema

En este trabajo de investigación, el problema a resolver es la extracción y clasificación de características del virus rugoso del tomate (ToBRFV) a través de técnicas de visión por computadora y técnicas de aprendizaje de máquina. El objetivo, entonces, es encontrar un vector de características con información capaz de diferenciar lo suficiente las dos clases para su correcta clasificación.

El aporte de este trabajo de investigación es una metodología capaz de identificar si una planta está infectada con el Virus Rugoso del Tomate (ToBRFV) o no. Así como la creación de una base de datos. Esta última es fundamental debido a que no hay una base de datos en existencia que tenga la información visual necesaria para la detección del Virus Rugoso del Tomame en hojas.

1.2. Justificación

Una de las principales preocupaciones sobre el cultivo del tomate es su susceptibilidad a la variedad de enfermedades virales. Los Tobamovirus están entre los virus más destructivos en hortalizas a nivel mundial (Koh, 2018), esto debido a que pueden inducir enfermedades graves en los cultivos. El ToBRFV es un tipo de Tobamovirus que se ha convertido en una gran amenaza para el cultivo del tomate. Se transmite de manera mecánica, además es sumamente peligroso por su variabilidad genética y duración, lo que ocasiona pérdidas sustanciales en este cultivo. El efecto negativo que causan los Tobamovirus, especialmente el ToBRFV, en los cultivos de tomate en México, da como resultado priorizar su estudio. Es por todas estas razones, que el estudio constante de este emergente virus en todos los ámbitos es de alta prioridad. Especialmente al ser tan contagioso y poco tratable. Su detección actualmente se da únicamente de manera visual mediante expertos que luego pasan a comprobarse con pruebas químicas especializadas; sin embargo, esta detección pudiese ser mejorada a partir de la visión por computadora en gran medida.

La caracterización de distintas enfermedades en plantas tiene nivel de dificultad que depende de los síntomas que muestran las hojas o frutos de la planta en cuestión y la constancia de estos síntomas. Es decir, si en la mayoría de las plantas enfermas se presentan los mismos síntomas. Pensando en esto, hay que resaltar que el ToBRFV es un virus con gran dificultad de caracterización. El hecho de que los síntomas se presenten de manera simultánea o independiente y además de manera inconstante, nos hace comprobar la dificultad antes mencionada de que la caracterización del ToBRFV es de dificultad alta.

El ToBRFV al ser un virus emergente con alto impacto negativo es el objetivo perfecto para caso de estudio. En el estado del arte no existen trabajos con aportaciones en el área de las ciencias de la información en-

focadas en su detección mediante algún método computacional. Al ser un virus reciente, los principales aportes que se han realizado son dentro de las áreas de la biotecnología, dejando de lado la gran aportación que puede dar la visión por computadora al control de este virus mediante su clasificación.

1.3. Objetivos

1.3.1. Objetivo General

Desarrollar modelos predictivos para la detección automática del virus rugoso del tomate (ToBRFV) en plantas de tomate mediante una metodología de visión por computadora.

1.3.2. Objetivos Específicos

- Realizar un estudio exhaustivo del estado del arte sobre el tema de investigación para la identificación de metodologías que permitan detectar enfermedades en las plantas mediante visión por computadora.
- Recolectar y formar el conjunto de datos integrado por imágenes de hojas de tomate sanas y hojas con presencia de síntomas causados por el ToBRFV. Conjunto que es necesario para el desarrollo del trabajo de tesis.
- Explorar técnicas de visión por computadora que permitan extraer características representativas del virus ToBRFV.
- Desarrollar modelos predictivos que permitan identificar el virus rugoso del tomate.
- Evaluar los resultados obtenidos de la metodología y modelos predictivos con el uso de métricas de desempeño existentes en la literatura.

1.4. Hipótesis

Mediante técnicas de visión por computadora y de aprendizaje de máquina es posible identificar eficientemente patrones del ToBRFV en imágenes digitales de hojas de tomate que permitan detectar y clasificar de manera oportuna y temprana patologías en plantas como el virus rugoso del tomate, obteniendo modelos de clasificación con una tasa de aciertos mayor al 80 %.

Estructura de la Tesis

A continuación, se presenta la organización de este documento de tesis. En el Capítulo II se definen conceptos a utilizar en la metodología tales como: Procesamiento de imágenes, aprendizaje de máquina, agricultura de precisión, entre otros. En el Capítulo III se describe la metodología propuesta para dar solución al problema planteado. El diseño conceptual de la metodología, la base de datos y el desarrollo de los modelos predictivos. Además, de los puntos principales de una metodología de clasificación, siendo estos preprocesamientos, segmentación, extracción de características y clasificación. Continuando con el Capítulo IV en donde se describen los resultados experimentales de la propuesta metodológica del presente trabajo de tesis. Finalmente, en el Capítulo V se presenta la conclusión del trabajo de investigación y algunas propuestas de trabajos a futuro.

Capítulo 2

Marco teórico y estudio del estado del arte

2.1. Marco teórico

En este capítulo vemos el estudio que se realizó del estado del arte de los principales temas que engloba este trabajo de tesis. Además, se describen los conceptos base necesarios para su desarrollo.

2.1.1. Imágenes

El término imagen se refiere a una función bidimensional de intensidad de luz $f(x, y)$ donde x e y representan las coordenadas espaciales y el valor de f es un valor en un rango de $(0, 255)$ que representa la intensidad de color de la imagen en ese punto. Los elementos de una distribución digital de este tipo se denominan elementos de la imagen, o más comúnmente píxeles (Mendez, 2008).

Una imagen a color o también conocida como imagen RGB, es representada en cada pixel como una combinación de ciertas intensidades de los colores rojo, verde y azul. Se puede interpretar como una matriz de tres canales donde cada canal corresponde a la intensidad de color de los componentes rojo, verde y azul, de manera visual lo podemos interpretar como la Figura 2.1.

Los espacios de color son una forma de representar la información importante de una imagen durante el procesamiento digital. Permiten analizar cada píxel desde otro punto de vista, y así aprovechar toda la información presente dentro de la imagen. Los trabajos más recientes realizados en esta área se relacionan con la segmentación de imágenes a color, la localización

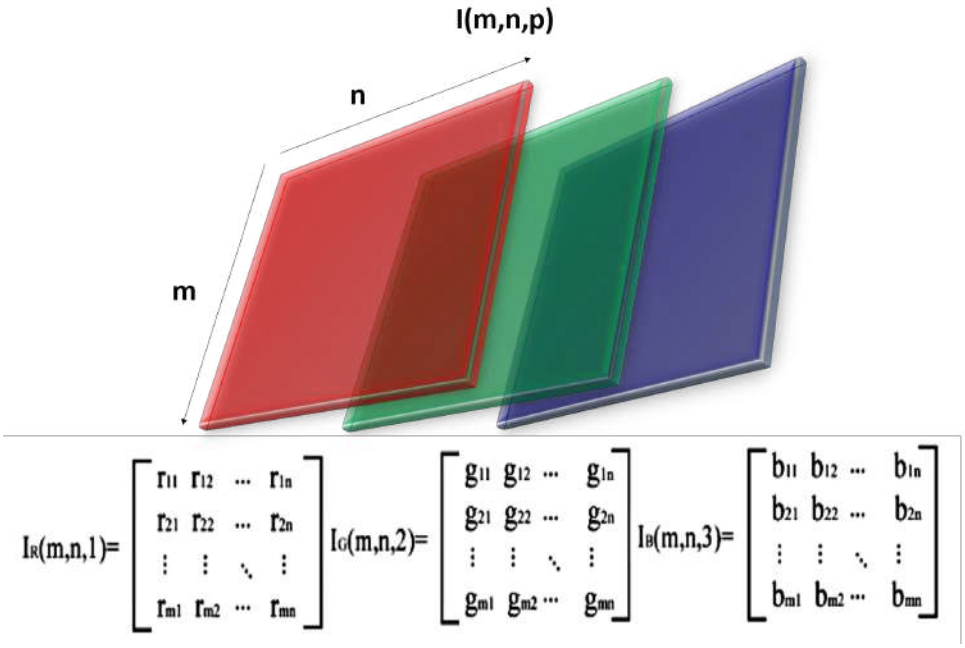

Figura 2.1: Representación gráfica y matricial de una imagen RGB.

de objetos, análisis de textura, morfología matemática, estandarización de imágenes a color, entre otros (Aguirre Dobernack, s.f.).

Por otra parte, las escalas de color son otra manera útil de representación para resaltar pequeñas diferencias en imágenes en escala de grises. Se utilizan para mejorar la visualización de los seres humanos en aplicaciones como índices de vegetación, cámaras térmicas y diferentes tipos de aplicaciones de detección remota. Las escalas de color consisten en una discretización de la escala de grises de 8 bits en dieciséis bandas de colores sólidos, donde los colores se asignan según la escala de colores que elija (Yee-Rendon et al., 2021). Las escalas de color más populares son las que se muestran en la Figura 2.2, en ella podemos ver además el rango de colores de cada una de ellas.

El procesamiento de imágenes implica cinco pasos básicos que son, adquisición de los datos, conformación de la base de datos, procesamiento de las imágenes, extracción de características y por último, clasificación (Shruthi et al., 2019).

La visión por computadora o visión artificial es un área de la computación que se utiliza para extraer, analizar y comprender una imagen digital

Figura 2.2: Representación visual de las escalas de color. (Bianco et al., 2014)

que contiene información del mundo real al visualizarla tal como lo hace el humano (Singh et al., 2019).

Su función principal es reconocer y localizar objetos en el ambiente mediante el procesamiento de las imágenes. La visión por computadora es el estudio de estos procesos, para entenderlos y construir máquinas con capacidades similares (Sucar and Gomez, 2011).

Un área muy ligada a la de visión por computadora es la de procesamiento de imágenes. Aunque ambos campos tienen mucho en común, el objetivo final es diferente. El objetivo de procesamiento de imágenes es mejorar la calidad de las imágenes para su posterior utilización o interpretación, por ejemplo: remover ruido, mejorar propiedades como color, contraste, estructura, etc. Mientras que el objetivo de la visión por computadora es extraer características de una imagen para su descripción e interpretación por una computadora. Por ejemplo: determinar la localización de objetos, construir una representación tridimensional de un objeto, etc. (Sucar and Gomez, 2011).

2.1.2. Análisis y detección de enfermedades en plantas

Las deficiencias de nutrición en las plantas son expresadas de distintas maneras en diferentes especies y ambientes de crecimiento, lo cual, hace que una evaluación unificada estándar sea difícil. Adicionalmente, existen limitaciones biológicas con deficientes de nutrientes en las plantas que pueden ocurrir concurrentemente (Li et al., 2020). El análisis rápido y preciso de las plantas requiere una segmentación efectiva de la parte de interés de la planta. (Asefpour Vakilian and Massah, 2017).

Los índices de vegetación son algoritmos que se enfocan en estimar la cobertura vegetal verde fraccional mediante la estimación indirecta de la superficie de vegetación a través de operaciones de álgebra lineal entre máscaras de imagen como rojo, verde y azul, o bandas hiperespectrales (Gao et al., 2020).

NGBVI es un índice de vegetación que se enfoca en resaltar las variaciones que puedan existir en las hojas de plantas por medio de las máscaras verde y azul. (Yee-Rendon et al., 2021). Su expresión matemática se muestra en la Ecuación (2.1).

$$NGBI = (VERDE - AZUL)/(MAX(VERDE - AZUL)) \tag{2.1}$$

NRBVI es otro índice de vegetación que se enfoca en resaltar las variaciones que puedan existir en las hojas de plantas por medio de las máscaras roja y azul, este índice también ayuda a descartar píxeles que no son vegetación en el entorno del invernadero (Yee-Rendon et al., 2021). Su expresión matemática se muestra en la Ecuación (2.2).

$$NRBVI = (ROJO - AZUL)/(MAX(ROJO - AZUL)) \tag{2.2}$$

2.1.3. Segmentación

La segmentación de imágenes es el proceso de separar o agrupar una imagen en diferentes partes. Actualmente, existen muchas formas diferentes de realizar la segmentación de imágenes, que van desde la desde un simple método de umbral hasta métodos avanzados de segmentación de imágenes en color (Singh, 2016). Una de las técnicas más utilizadas es el método de selección de umbral a partir de un histograma en escala de grises (OTSU, 1979).

2.1.4. Extracción y selección de características

Una característica de interés en una imagen de una planta, por ejemplo una hortaliza, está relacionado con el color, la forma, el tamaño, la composición, el sabor o un defecto. Los vectores de características representan una imagen o parte de ella reteniendo información útil mientras que la redundante se deja de lado. Se utilizan para detección y clasificación de objetos en imágenes digitales (Naik, 2017).

Las características de Gabor, el filtro de Gabor, los patrones binarios locales, la transformada discreta de Fourier, la derivada de primer orden del gradiente medio, la derivada de segundo orden de Laplacian media, la media y la desviación estándar son métodos típicos para la extracción de características. Otros métodos, como la extracción de características profundas, que se basa en redes neuronales profundas, son útiles cuando la estructura de datos es compleja.(Frederic et al., 2019).

Una de las principales características que se observan en procesos de reconocimiento de patrones en plantas es la textura. Podemos reconocer la textura fácilmente cuando la vemos, su definición no es tan sencilla (Tuceryan and Jain, 1993). Textura es la denominación utilizada con el propósito de caracterizar la superficie de un fenómeno u objeto. En imágenes, la textura es en esencia una propiedad de vecindario (Unser, 1986). Su estructura se atribuye a los patrones repetitivos en los que sus elementos están distribuidos de acuerdo a alguna regla de distribución (Tuceryan and Jain, 1993). Según (Vijayakumar, 2020) hay cuatro tipos de métodos que se utilizan para describir la textura, métodos estadísticos, métodos estructurales, métodos basados en modelos y métodos basados en transformación.

La matriz de co-ocurrencia a niveles de escala de grises (GLCM, por sus siglas en inglés), es una matriz que describe la frecuencia en que un nivel de intensidad de gris aparece en relación con otro valor de gris con una relación de posición específica, dentro del área de una ventana determinada. En resumen, es la manera en que los valores de los píxeles ocurren uno al lado de otro en una ventana (Presutti, 2004).

La GLCM considera la relación espacial entre dos píxeles, a los que se les denomina como pixel de referencia y pixel vecino. Cada uno de estos píxeles ubicados dentro de la ventana se va siendo el nuevo pixel de referencia sucesivamente, iniciando con el ubicado arriba a la izquierda y terminando

con el ubicado abajo a la derecha. Los píxeles ubicados en el margen de la imagen original, tienden a no ser usados en los cálculos debido a las restricciones que tienen estas posiciones.

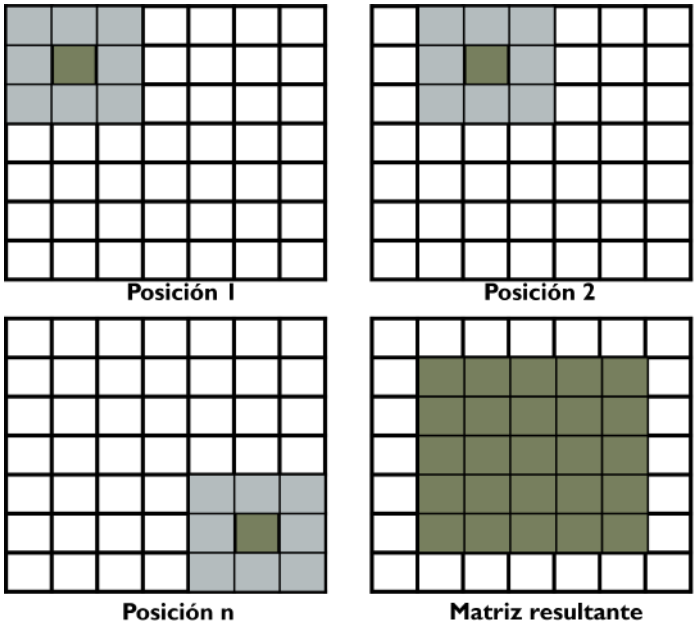

Figura 2.3: Representación visual del recorrido de la comparación de píxeles, posición por posición y matriz resultante para una comparación de 9 posiciones.

La Figura 2.3, representa el recorrido comentado en el párrafo anterior de la comparación de píxeles, posición por posición y matriz resultante para una comparación de 9 posiciones.

Las posibles combinaciones de niveles de grises se colocan en una nueva matriz. Existen, por lo tanto, diferentes matrices de co-ocurrencia para cada relación espacial, según se considere el vecino de arriba, al costado o en diagonal. Después de obtener esta matriz, se le suma su transpuesta y se obtiene una matriz simétrica. Por último, se normaliza obteniendo así la matriz que contiene la probabilidad de ocurrencia de las configuraciones de color en la imagen. A partir de esta matriz de probabilidad se pueden hacer los cálculos necesarios para las características de homogeneidad, disimilaridad, entropía, etc(Presutti, 2004).

Las ecuaciones con las que se calculan las características a partir de la matriz de probabilidades según (Presutti, 2004) son las siguientes:

- Homogeneidad

$$\sum_{i,j=0}^{N-1} \frac{P_{i,j}}{1+(i-j)^2} \tag{2.3}$$

$$\sum_{i,j=0}^{N-1} P_{i,j}\,(i-j)^2 \tag{2.4}$$

- Disimilaridad

$$\sum_{i,j=0}^{N-1} P_{i,j}\,|i-j| \tag{2.5}$$

- Entropía

$$\sum_{i,j=0}^{N-1} -P_{i,j} ln\,(P_{i,j}) \tag{2.6}$$

- Correlación

$$\sum_{i,j=0}^{N-1} P_{i,j} \left[\frac{(i-\mu_i)\,(j-\mu_i)}{\sqrt{(\sigma_i^2)\,(\sigma_i^2)}} \right] \tag{2.7}$$

- ASM(Angular Second Moment)

$$\sum_{i,j=0}^{N-1} P_{i,j}^2 \tag{2.8}$$

En las ecuaciones 2.3, 2.4, 2.5, 2.6, 2.7 y 2.8 se tiene $P_{i,j}$ como la probabilidad de ocurrencia delos valores de gris i y j, para una distancia dada.

Los histogramas de sumas y diferencias fueron introducidos como una alternativa para los métodos usuales para el análisis de textura (Unser, 1986), los cuales están basados en matrices de co-ocurrencia como el de la GLCM descrito anteriormente.

Tómese en cuenta una imagen rectangular KxL denotada por $y_{k,l}$, donde $(k = 1, ..., K; l = 1....L)$ y $G = 1, 2, ..., N_g$, con N_g representando los niveles de grises cuantificados de la imagen. Entonces, consideramos dos elementos de una imagen en posiciones relativas por un par de distancias (d_1, d_2) como las ecuaciones 2.9 y 2.10 respectivamente (Unser, 1986):

$$y1 = y_{k,l} \tag{2.9}$$

$$y2 = y_{k+d1,l+d2} \tag{2.10}$$

El desplazamiento relativo (d_1, d_2) es equivalentemente caracterizado como una distancia d en unidades radiales y un ángulo Θ con respecto al eje horizontal. Entonces, la función de probabilidad de unión discreta de estas dos variables que representan un punto en la imagen es $P(y_1, y_2)$. Y la probabilidad de observar los niveles de gris i, j en una posición relativa fija de dos píxeles especificada por (d_1, d_2) es la representación que podemos observar en la ecuación 2.11 (Unser, 1986) :

$$P_{rob}\{y_{k,l} = i, y_{k+d_1,l+d_2} = j\} = P(i, j; d_1, d_2) = P(i, j) \tag{2.11}$$

La suma y la diferencia no normalizadas, asociados con el desplazamiento relativo (d_1, d_2), se definen como lo podemos ver en las ecuaciones 2.12 y 2.13 respectivamente.

$$s_{k,l} = y_{k,l} + y_{k+d1,l+d2} \tag{2.12}$$

$$d_{k,l} = y_{k,l} - y_{k+d1,l+d2} \tag{2.13}$$

Por lo tanto, el rango de la suma y diferencia es generalmente el doble del rango del tamaño original.

Los histogramas de sumas y diferencias(h_s y h_d) con parámetros (d_1, d_2) sobre el dominio(D) están definidos según las ecuaciones 2.14 y 2.15 respectivamente.

$$h_s(i; d_1, d_2) = h_s(i) = C_{ard}\{(k, l) \in D, s_{k,l} = i\} \tag{2.14}$$

$$h_d(j; d_1, d_2) = h_d(j) = C_{ard}\{(k, l) \in D, d_{k,l} = j\} \tag{2.15}$$

El número total de la cuenta sería descrito por la ecuación 2.16.

$$N = C_{ard}\{D\} = \sum_i h_s(i) = \sum_j h_d(j) \tag{2.16}$$

Los histogramas de sumas y diferencias normalizados estarían dados entonces matemáticamente como las ecuaciones 2.17 y 2.18 los describen.

$$\hat{p}_s(i) = h_s(i)/N; (i = 2, ..., 2N_g) \tag{2.17}$$

$$\hat{p}_d(j) = h_d(j)/N; (j = -N_g + 1, ..., N_g - 1) \tag{2.18}$$

Existen ocasiones en las que surge la necesidad de reducir la cantidad de características utilizadas para describir la textura. La información estadística puede ser extraída a partir de ecuaciones basadas en los histogramas obtenidos (Unser, 1986).

En la Tabla 2.1 podemos ver las ecuaciones que corresponden a las distintas características de textura generalmente utilizadas. Podemos verlas en distintos formatos, tanto para los métodos con matrices de ocurrencias y el método del que hablamos en cuestión, que es el método de histogramas de sumas y diferencias (Unser, 1986).

Tabla 2.1: Fórmulas para extracción de características para los métodos de matrices de co-ocurrencia y los histogramas de sumas y diferencias

Característica de textura	Matriz de co-ocurrencias	Histogramas de sumas y diferencias
Media	$\sum_i \sum_j i \cdot \hat{p}(i,j)$	$\frac{1}{2} \sum_i i \cdot \hat{p}_s(i) = \mu$
Varianza	$\sum_i \sum_j (i-\mu)^2 \cdot \hat{p}(i,j)$	$\frac{1}{2}\{\sum_i (i-2\mu)^2 \cdot \hat{p}_s(i) + \sum_j j^2 \cdot \hat{p}_d(j)\}$
Energía	$\sum_i \sum_j \hat{p}(i,j)^2$	$\sum_i \hat{p}_s(i)^2 \cdot \sum_j \hat{p}_d(j)^2$
Correlación	$\sum_i \sum_j (i-\mu) \cdot (j-\mu) \cdot \hat{p}(i,j)$	$\frac{1}{2}\{\sum_i (i-2\mu)^2 \cdot \hat{p}_s(i) - \sum_j j^2 \cdot \hat{p}_d(j)\}$
Entropía	$\sum_i \sum_j -\hat{p}(i,j) \cdot \log(\hat{p}(i,j))$	$-\sum_i \hat{p}_s(i) \cdot \log\{\hat{p}_s(i)\} - \sum_j \hat{p}_d(j) \cdot \log\{\hat{p}_d(j)\}$
Contraste	$\sum_i \sum_j (i-j)^2 \cdot \hat{p}(i,j)$	$\sum_j j^2 \cdot \hat{p}_d(j)$
Homogeneidad	$\sum_i \sum_j \frac{1}{1+(i-j)^2} \cdot \hat{p}(i,j)$	$\sum_j \frac{1}{1+j^2} \cdot \hat{p}_d(j)$

2.1.5. Conceptos y técnicas básicas de aprendizaje de máquina

Un patrón es la representación de una relación estadística entre señales y se obtiene mediante el análisis matemático de ejemplos de señales adquiridas previamente. Por lo que, un patrón representa a una clase de señales, que a su vez representa una clase de entidades individuales (objetos, acciones, eventos, procesos, etc.). Normalmente, se asocia una etiqueta a cada patrón que indica el nombre de la clase correspondiente. Los patrones se obtienen a partir de la extracción de características distintivas o atributos de dichas entidades, normalmente del medio ambiente, por lo que las señales casi siempre contienen características que no aportan al proceso de aprendizaje y se requiere reducir la información obtenida, de tal forma que se representen solo los rasgos significativos de las clases (Reyes García, 2017).

El reconocimiento de patrones consiste en clasificar una o varias señales desconocidas en la clase correspondiente. Un sistema de reconocimiento de patrones se construye a partir de analizar y manipular instancias o ejemplos de un problema. La construcción de este tipo de sistemas tiene dos etapas principales: diseño y evaluación. En la primera se generan reglas de decisión para clasificar instancias del problema y en la segunda se verifica de forma intensiva que dichas reglas sean apropiadas para tomar buenas decisiones (Reyes García, 2017).

Matemáticamente, la clasificación consiste en partir un espacio n-dimensional, con una dimensión por cada característica relevante, y cada región de este espacio corresponde a una clase determinada (Reyes García, 2017).

El aprendizaje automático es un campo muy dinámico de la inteligencia artificial que ha experimentado avances significativos a lo largo de los años. Se han creado varias técnicas para aplicaciones en el mundo real. Las técnicas de aprendizaje de máquina son potentes algoritmos capaces de resolver una gran variedad de problemas. Actualmente, la dificultad principal recae en el tamaño y calidad de los conjuntos de datos disponibles para entrenar y validar los modelos (Arnal Barbedo, 2019).

Para este trabajo de investigación, se emplearon cinco técnicas de aprendizaje automático: k-vecinos más cercanos (KNN, por sus siglas en inglés de K-Nearset Neighbor), máquinas de soporte vectorial(SVM, por sus siglas en inglés de Support Vector Machine), redes neuronales artificiales (ANN, por sus siglas en inglés de Artificial Neural Network), árboles de decisión(DT, por sus siglas en inglés de Decision Tree) y bosque aleatorio(RF, por sus siglas en inglés de Random Forest).

Dentro del aprendizaje de máquina existen tres principales tipos de aprendizaje: aprendizaje supervisado, no supervisado, y por refuerzo. Nos enfocaremos en describir las técnicas de aprendizaje supervisado y no supervisado.

En el caso del aprendizaje supervisado, se generan modelos que utilizan un conjunto de instancias etiquetadas para relacionar los datos que entran(características) con los que salen(etiquetas), o, en otras palabras, los modelos se ajustan a un conjunto de instancias de entrenamiento con las que aprenden la relación entre una entrada y su salida deseada. Este tipo de aprendizaje puede tener entonces dos tipos de modelo en función de su salida. Si su salida es un valor de un espacio continuo, estamos hablando de un modelo de regresión, en cambio, si su salida es un valor categórico, entonces es un modelo de clasificación. Algunos ejemplos de técnicas de aprendizaje supervisado son: k-vecinos más cercanos (KNN, por sus siglas en inglés de K-Nearset Neighbor), máquinas de vectores de soporte (SVM por sus siglas en inglés de Support Vector Machine), redes neuronales artificiales, entre otras.

El método de los k-vecinos más cercanos es una de las técnicas de clasificación supervisada más simples y fundamentales. Consiste en calcular

las distancias entre una nueva instancia y las instancias existentes, y luego ordenar esas distancias de menor a mayor para determinar a qué clase pertenece la instancia. La clase asignada será aquella que tenga la mayor frecuencia entre las instancias más cercanas en distancia (J. M. Keller and Givens, 1985). En el algoritmo KNN, se considera que las instancias que se encuentran cerca unas de otras son vecinos. El número de vecinos más cercanos a examinar se especifica mediante el valor de k. En situaciones donde se produce un empate entre dos o más clases, es recomendable aplicar reglas heurísticas para desempatar. Por ejemplo, se puede seleccionar la clase que contiene al vecino más cercano, o elegir la clase con una distancia media menor entre los vecinos, entre otras posibles reglas. Estas reglas ayudan a asignar una clase definitiva a la instancia en cuestión cuando hay ambigüedad en la clasificación.

En los últimos años, las máquinas de soporte vectorial se han convertido en una de las técnicas de aprendizaje automático más utilizadas para problemas de clasificación binaria, y también se han extendido a problemas de clasificación múltiple y regresión. La idea principal de las MVS es encontrar un hiperplano entre los diversos hiperplanos que separan las clases, que maximice el margen entre los vectores de soporte. Los vectores de soporte son un subconjunto de datos que corresponden a las instancias más cercanas de las clases y definen la posición del hiperplano (Wanjun and Xiaoguang, 2010). Cuando el problema de clasificación no puede ser separado de manera lineal, la técnica de SVM recurre al uso de una función de kernel. Esta función se encarga de mapear las instancias a un espacio de características de mayor dimensión, donde las instancias se vuelven linealmente separables. Al aplicar una función de kernel, como el polinómico, el de base radial o la función sigmoidal, las SVM pueden abordar problemas de clasificación no lineal al encontrar un hiperplano de separación óptimo en el espacio transformado.

Las redes neuronales artificiales son modelos que imitan el funcionamiento de las neuronas en el cerebro humano. Estas redes están compuestas por unidades de procesamiento organizadas en estructuras interconectadas llamadas capas. Una red neuronal artificial se forma por una o más capas. En la literatura especializada se distinguen tres tipos de capas según su función: la capa de entrada, la capa intermedia y la capa de salida. Cada capa está formada por un conjunto de nodos, también conocidos como neuronas. Estos nodos reciben las entradas de la capa anterior y generan salidas que se convierten en las entradas de la siguiente capa. La capa de entrada

recibe las instancias a procesar, mientras que la capa de salida produce los resultados finales de la red. Las capas intermedias se encuentran entre la capa de entrada y la capa de salida, y ofrecen flexibilidad a la red, ya que desconocen tanto las entradas como las salidas, permitiendo que la red crezca en profundidad y complejidad (Bis, 2011).

Los árboles de decisión son estructuras jerárquicas que generan decisiones secuenciales, de ahí su denominación como "árboles". Los componentes fundamentales de los árboles de decisión son los nodos de decisión y los nodos hoja. Los nodos de decisión representan las características de las instancias en el conjunto de datos. A partir de los nodos de decisión se generan ramas, una por cada posible valor de las características, y cada rama se conecta a otro nodo de decisión o a un nodo hoja. Los nodos hoja representan las clases a las que pertenecen las instancias. En esta técnica, se utiliza la estrategia "divide y vencerás" para descomponer el problema en subproblemas más pequeños y manejables. Los árboles de decisión ofrecen un enfoque sistemático y estructurado para la toma de decisiones al organizar la información de manera jerárquica, permitiendo una interpretación clara y lógica de las reglas de decisión (Gavankar and Sawarkar, 2017).

El bosque aleatorio es una técnica que consiste en combinar un conjunto de árboles de decisión entrenados de manera independiente. Cada árbol de decisión en el bosque contiene diferentes evaluaciones en sus nodos. El objetivo de esta técnica en un problema de clasificación es obtener predicciones más precisas. La predicción se obtiene al seleccionar la clase que obtiene la mayoría de votos entre todos los árboles del bosque. La técnica del bosque aleatorio genera una serie de árboles de decisión, donde cada árbol es entrenado utilizando un conjunto de instancias de entrenamiento. Luego, las predicciones de estos modelos se combinan mediante el método de votación mayoritaria. Al combinar múltiples árboles de decisión entrenados de forma independiente, el bosque aleatorio aprovecha la diversidad de los modelos para mejorar la precisión de las predicciones. Esta técnica es ampliamente utilizada en problemas de clasificación para obtener resultados más robustos y confiables (Breiman, 2017).

Por otra parte, el aprendizaje no supervisado genera modelos en los que se utilizan conjuntos de instancias no etiquetadas con el objetivo de aumentar el conocimiento estructural de los conjuntos, para casos como, agrupar los datos según su similitud, extraer la estructura interna de forma en que se distribuyen los datos en el espacio original, o simplificar la estructura

de los datos manteniendo solamente sus características fundamentales. Un ejemplo de técnica de este tipo de aprendizaje es K-medias (k-means en inglés). El K-means utiliza datos no etiquetados para su clasificación. El principio de este clasificador es encontrar grupos en los datos, con el número de grupos representado por la variable K. El clasificador de K-media funciona de forma iterativa para asignar cada punto de datos a uno de los K grupos en función de las características proporcionadas. Luego, los puntos de datos se agrupan en función de la similitud de características (Bis, 2011).

2.1.6. Conceptos relacionados con la agricultura de precisión

Se le denomina agricultura de precisión al conjunto de técnicas orientadas a optimizar el uso de insumos agrícolas en función de la cuantificación de la variabilidad espacial y temporal de la producción agrícola (Roel and Terra, 2013). Es una adaptación de tecnologías, dispositivos y métodos emergentes en la agricultura para mejorar la productividad de los cultivos. Los dispositivos como cámaras, sensores, dispositivos wifi, etc. con tecnologías como aprendizaje de máquina, visión por computadora, Internet de las cosas, etc. se utilizan para desarrollar el sistema de apoyo a la toma de decisiones que controla y gestiona automáticamente la productividad de los cultivos (Singh et al., 2019). La fitopatología es la ciencia dedicada al estudio de las enfermedades o patógenos padecidos por las plantas. Comprende el ciclo de vida, crecimiento y muerte de las plantas y el cómo las enfermedades que padecen son influenciadas por factores tanto bióticos como abióticos. Actualmente, existen muchos estudios para el análisis y detección de virus utilizando imágenes digitales de plantas con algoritmos de visión por computadora y técnicas de aprendizaje de máquina.

El virus rugoso del tomate (ToBRFV por sus siglas en inglés de Tomato Brown Rugose Fruit Virus) presenta síntomas en frutos en forma de coloración amarilla, manchas verdes, malformación de frutos, estriado verde, manchas irregulares color marrón, hojas con mosaicos y moteado amarillento (Nolasco-García et al., 2020). El ToBRFV se puede presentar en las especies de vegetales tales como tomate (Solanum lycopersicum), chile (Capsicum sp.) y berenjena (Solanum melongena) (SADER, 2018).

Los síntomas del ToBRFV son los descritos a continuación la producción (Caades, 2015).

- necrosis del pedúnculo y cáliz
- Amarillamiento de las hojas.
- Tallos y hojas secos.
- Manchas amarillas en hojas y frutos.
- Rugosidad en hojas y frutos.
- Textura en forma de ampollas en hojas.

El virus puede permanecer estable en semillas, restos de plantas y en suelos durante meses o años. Frecuentemente, los frutos reducen su tamaño. Cuanto más precoz es la infección, mayor es la repercusión en la producción (Caades, 2015).

Por otra parte, podemos observar los síntomas del virus ToBRFV en el órgano distintivo del fruto. Algunos ejemplos se ilustran en la Figura 2.4.

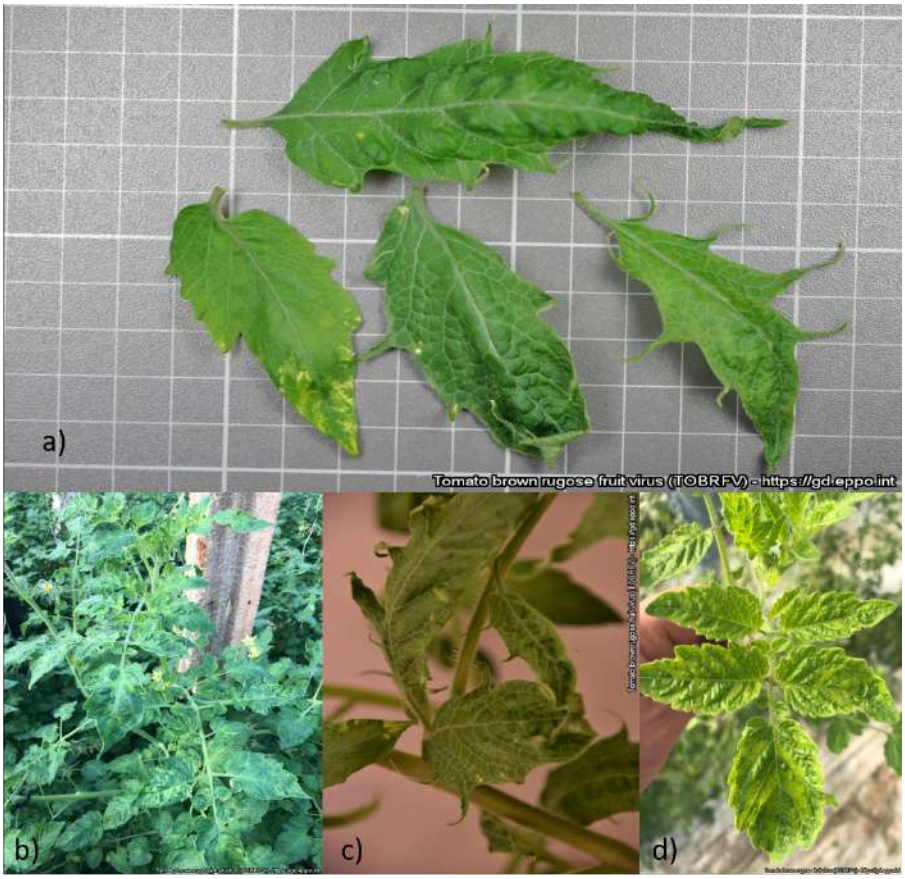

Figura 2.4: Síntomas del ToBRFV en hojas, a) varios síntomas, b) Necrosis, c) Enroscamiento y d) Ampollas. Tomadas de https://gd.eppo.int/taxon/TOBRFV/photos.

En la Figura 2.5 podemos ver ejemplos de los síntomas que presentan los frutos debido al virus rugoso del tomate.

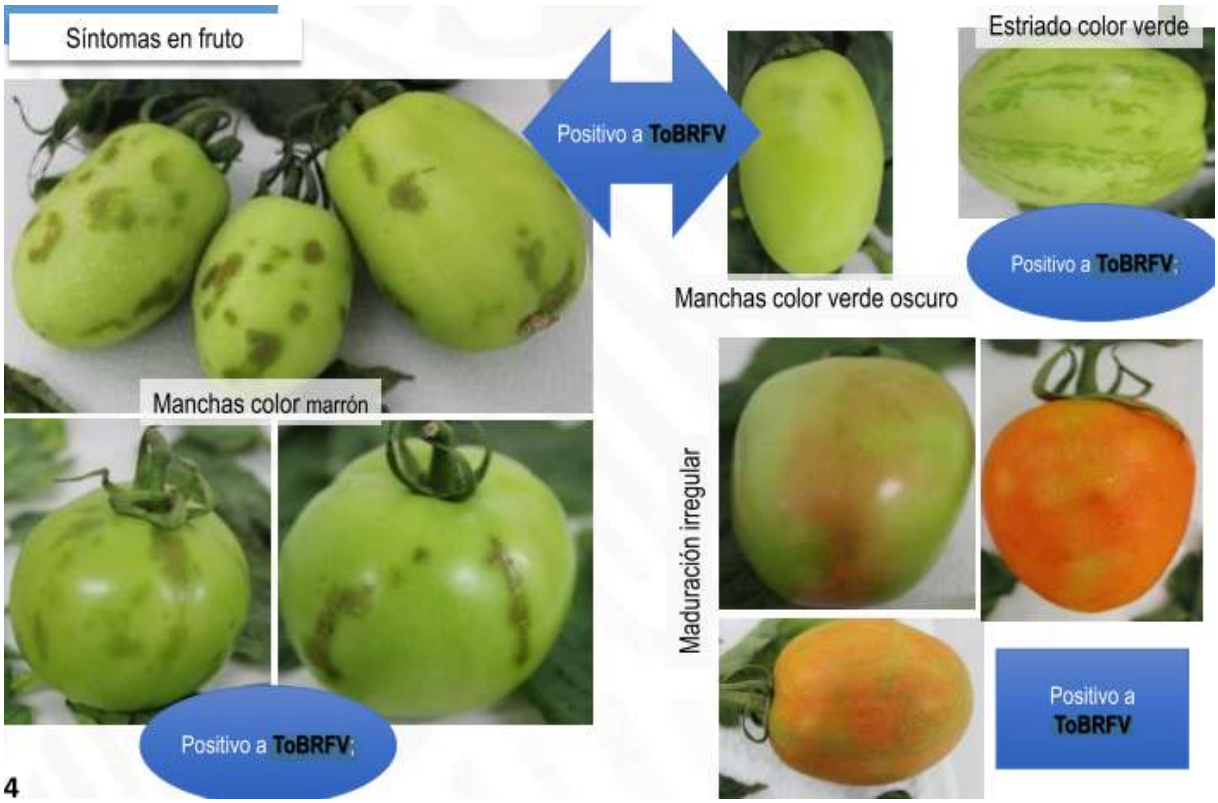

Figura 2.5: Síntomas del ToBRFV en frutos. Tomada de (SAGARPA SENASICA, 2018).

La infección del ToBRFV se basa en la dispersión de larga distancia por medio del movimiento de semillas infectadas de un país a otro, y en la dispersión a corta distancia mediante operadores, insectos polinizadores y el contacto de planta a planta que representa el aspecto más peligroso para la rápida propagación del ToBRFV (Davino et al., 2020).

2.2. Estudio de técnicas recientes para la detección de enfermedades en plantas

A continuación, se describen algunos trabajos del estado del arte relacionados con el objeto de estudio del presente trabajo de investigación.

Phadikar y Sil en (Phadikar and Sil, 2008) analizaron dos enfermedades que afectan a los cultivos de arroz mediante imágenes digitales, para

ello, transformaron las imágenes a un espacio de color HSI, y aplicaron una técnica de segmentación utilizando un umbral basado en entropía. Después, a las imágenes segmentadas se les aplicó un detector, y luego las manchas se detectan utilizando la intensidad de los componentes verdes.

Yao et al. en (Yao et al., 2009) propusieron un método para identificar y clasificar tres tipos de enfermedades que afectan a los cultivos de arroz. Una imagen es segmentada por el método de Otsu, después las regiones enfermas son aisladas. Las características de color, forma y textura son extraídas, este último proviene del espacio de color HSV. Finalmente, las características obtenidas son la entrada para un modelo de clasificación basado en una máquina de soporte vectorial.

Kurniawati et al. (Kurniawati et al., 2009) propusieron un método para identificar y etiquetar tres tipos diferentes de enfermedades que afectan a los cultivos de arroz. Utilizaron la umbralización como técnica de segmentación de las regiones sanas y enfermas. Wang y col. en (Kurniawati et al., 2009) desarrollaron un método para discriminar entre pares de enfermedades en el trigo y vides. Las imágenes fueron segmentadas utilizando K-means, y luego 50 características de color, forma y textura son extraídos. El trabajo propuesto por Macedo-Cruz et al. en (Macedo-cruz, 2011) analizó el daño causado por las heladas en los cultivos de avena. Primero realizó una conversión de RGB al espacio de color $L * a * b$. Los autores emplearon tres estrategias de umbral diferentes: El método de Otsu, el algoritmo Isodata y el umbral difuso.

En el 2013, Arivazhagan et al. (Arivazhagan S., 2013) propusieron un sistema para la detección y clasificación de enfermedades en hojas de plantas, las cuales pueden clasificarse ampliamente en tres tipos: bacterianas, fúngicas y virales. El esquema de procesamiento desarrollado consta de cuatro pasos principales, el primero paso es crear una estructura de transformación de color para una imagen RGB de entrada, luego se enmascaran y eliminan los píxeles verdes utilizando un valor de umbral específico, seguido de un proceso de segmentación, las estadísticas de textura se calculan para los segmentos útiles, finalmente las características extraídas son las entradas a un clasificador basado en una máquina de soporte vectorial. El conjunto de datos que se utilizó fue de aproximadamente 500 hojas de plantas de plátano, frijoles, yaca, limón, mango, papa, tomate y sapote. La exactitud del modelo de clasificación alcanzó un 94 %.

En 2016, Singh y Misra (Singh, 2016) realizaron un estudio comparativo de diferentes técnicas de calificación. Para la extracción de características utilizaron el método de coocurrencia del color, en la cual, tanto la textura como el color de la imagen son consideradas como las únicas características. Todos los experimentos se realizaron en MATLAB. Para datos de entrada de enfermedades, muestras de plantas, hojas como rosa con enfermedad bacteriana, hojas de frijoles con enfermedad bacteriana, hojas de limón con quemaduras solares, hoja de plátano con enfermedad de quemadura temprana y enfermedad fúngica en hoja de frijol. Primero se utilizó el método de clasificación K-means que mostró una exactitud del 86.54 %. Luego se utilizó el método SVM con el que se obtuvo una exactitud del 95.71 %.
Xuebin Bai et al. (Bai et al., 2017) propusieron un método de segmentación de agrupamiento difuso basado en vecindario información en escala de grises para analizar imágenes con enfermedades de la mancha de la hoja del pepino. Para evaluar la solidez y precisión del método de segmentación propuesto, se realizaron pruebas con 129 imágenes de enfermedades del pepino. Los resultados muestran que el error de segmentación promedio fue de solo 0,12 %.

Jiaquan Chen et al. (Chen et al., 2017) desarrollaron un método de segmentación que integra la información espacial en el proceso de establecimiento de umbrales. El método propuesto utiliza la entropía local de un píxel para caracterizar su información espacial. La entropía local de un píxel puede representar la variedad de nivel de gris en un vecino. El nivel de gris y la entropía local se utilizan juntos para construir un histograma bidimensional, que se denomina histograma GLLE. Un método de umbral bidimensional basado en el histograma GLLE se obtiene maximizando la entropía Tsallis del objeto y el fondo.

En 2017, Islam et al. (Islam et al., 2017) extrajeron 10 características de color y textura para la clasificación de enfermedades en plantas. Se utilizó la Matriz de Coocurrencia de Niveles de Grises (GLCM) para extraer características estadísticas de la textura como el contraste, la correlación, la energía y la homogeneidad. Además, a partir de los histogramas de los planos de color se calcularon indicadores numéricos como media, desviación estándar, entropía, sesgo y energía. La técnica de segmentación se basó en un conjunto de máscaras generadas mediante el análisis de los componentes de color y luminosidad de diferentes regiones de las imágenes en espacios de color $L * a * b$. Con este enfoque en la segmentación y utilizando SVM para la clasificación de enfermedades en 300 imágenes de hojas de papa

provenientes del conjunto de datos 'Plant Village' (100 sanas y 200 enfermas) se obtuvo una exactitud del 95 %.

En el 2018, Sengar et al. (Sengar et al., 2018) presentaron un método para la identificación y cuantificación de la enfermedad de cenicilla (*powdery mildew*) en imágenes de hojas de cultivos de cerveza, para ello, utilizaron un umbral basado en la intensidad adaptativa para la segmentación automática del área afectada por la enfermedad a partir de imágenes de hojas. El método propuesto se probó en un conjunto de imágenes de hojas de cultivos de cerezas, 50 imágenes de hojas sanas y 50 de hojas enfermas, y alcanzó una exactitud del 99 %.

Peng Wang et al. (Wan et al., 2018) propusieron un método para detectar los niveles de madurez (verde, naranja y rojo) de tomates frescos de mercado (variedades Roma y Pera) combinando el valor del color de la característica con un clasificador basado en redes neuronales. Se diseñó un dispositivo de detección de madurez basado en tecnología de visión por computadora específicamente para adquirir las imágenes de tomate en el laboratorio. Las imágenes de tomate fueron procesadas y los objetivos de los tomates se obtuvieron con base en procesamiento de imágenes. Después de eso, se utilizaron los círculos máximos inscritos en la superficie de los tomates para determinar la región de extracción del color de la característica. La región de extracción se dividió en cinco subregiones de colores en la misma área con círculos concéntricos de diferentes radios. Los valores de color de las funciones RGB y HSI se extrajeron de las áreas de color de la característica (FCA por sus siglas en inglés). Luego, los valores de color B y H de los FCA se consideraron como el valor de la característica de color del tomate. Finalmente, se adoptó un clasificador, utilizó tres capas para clasificar el nivel de madurez de dos variedades de tomate diferentes. Los resultados de esta investigación muestran que la tasa de precisión para este método de extraer los valores de H del tomate imagen y luego detectar y clasificar los niveles de madurez de las muestras de tomate fue del 99,31 %.

En una revisión realizada por Rehman et al. (Rehman et al., 2019), se concluyó que la técnica de aprendizaje de máquina basada en SVM tiene éxito y se aplica en diferentes áreas de los sistemas de visión artificial agrícola, como la detección de enfermedades de las plantas. Otros algoritmos estadísticos de aprendizaje de máquina que pueden usarse potencialmente para los sistemas de visión artificial agrícolas, según esta revisión, incluyen árboles de decisión, bosques aleatorios y regresión logística.

En 2019, Abdu et al. (Abdu et al., 2019) probaron que la aplicación del concepto patológico permite cuantificar las áreas de lesión de la enfermedad real de acuerdo con su verdadera analogía. Con esto, se pueden rastrear las características del patrón individual de cada lesión a lo largo de la superficie de la hoja y luego se pueden extraer las características para caracterizarlas mediante el aprendizaje automático. Los resultados obtenidos son alentadores, localizando y cuantificando con éxito lesiones de enfermedades individuales. Esto también indica la aplicabilidad del enfoque propuesto en la discriminación de enfermedades de las plantas en función de su disimilitud analógica. Además, brinda oportunidades para la identificación y detección tempranas de cambios finos en el crecimiento de las plantas, la etapa de la enfermedad y la estimación de la gravedad para ayudar al diagnóstico de cultivos en la agricultura de precisión.

Shruthi et al. (Shruthi et al., 2019) realizaron una revisión donde se llevó a cabo un estudio comparativo sobre cinco tipos de técnicas de clasificación de aprendizaje automático para el reconocimiento de enfermedades de las plantas. Las técnicas comparadas fueron SVM, Redes Neuronales Artificiales (ANN o RNA), KNN, clasificador de tipo Fuzzy y aprendizaje profundo CNN (Convolutional Neural Network). La extracción de características se realizó mediante la matriz de coocurrencia a escala de grises (GLCM). El resultado mostró que el clasificador CNN detecta una mayor cantidad de enfermedades con alta exactitud (mayor al 96 % en todos los casos de estudio de este trabajo).

En un estudio propuesto por Arthur Z. da Costa (Arthur et al., 2019) analizaron la detección de defectos externos en tomates utilizando aprendizaje profundo. Los autores construyeron un conjunto de datos de 43,843 imágenes tomates con defectos externos. El conjunto presenta un alto grado de desbalance hacia la clase sana (aquellas imágenes que no presentan defectos externos). Se entrenaron modelos de clasificación basados en redes neuronales residuales profundas (ResNet) que son capaces de detectar defectos externos mediante aprendizaje por transferencia con las dos técnicas, extracción de características y fine-tuning. Los resultados muestran que los modelos entrenados con la técnica fine-tuning tuvieron mejores resultados que con la de extracción de características, revelando el beneficio de entrenar capas adicionales cuando hay suficientes muestras de datos disponibles. El mejor modelo fue un ResNet50 con todas sus capas reentrenadas. Este modelo logró una precisión promedio del 94: 6 % en el conjunto de prueba.

La investigación realizada en 2020 por Nturambirwe y Opara (Frederic et al., 2019) ha demostrado que los métodos de aprendizaje de máquina son efectivos para mejorar la precisión de la detección, ya sea al permitir la fusión de datos y sensores, al permitir la reducción de la dimensionalidad de los datos o la extracción de características. Permiten un procesamiento espectral y de imágenes más rápido que los métodos de segmentación tradicionales, la detección y automatización de objetos más rápida se vuelve muy factible mediante algoritmos de aprendizaje entrenados como ANN, SVM, CNN, entre otros.

Vijayakumar (Vijayakumar, 2020), logró diagnosticar enfermedades del arroz, el banano y la caña de azúcar mediante métodos de visión artificial y aprendizaje automático tales como el Sistema de Inferencia Adaptativo Neuro Fuzzy (ANFIS). La detección asistida por computadora usando segmentación se realiza usando la técnica de umbral para localizar el lugar enfermo de la imagen previamente procesada. Al final utilizó la curva del receptor de características operativas (ROC) para evaluar el esquema sugerido y se obtuvo una exactitud del 97.5 %.

Recientemente, Yee-Rendon et al. (Yee-Rendon et al., 2021) propusieron una nueva metodología de análisis predictivo para identificar los síntomas visuales del virus del mosaico del tabaco (TMV) y el virus de la vena amarilla del pimiento huasteco (PHYVV) en chile jalapeño (Capsicum annuum) usando procesamiento de imágenes y modelos de clasificación de aprendizaje profundo. El enfoque de procesamiento de imágenes se basó en el uso del Índice de Vegetación Rojo-Azul Normalizado (NRBVI) y el Índice de Vegetación Verde-Azul Normalizado (NGBVI) como nuevos índices de vegetación basados en RGB. Para la clasificación se utilizaron modelos basados en CNN los cuales fueron entrenados usando el Gradiente del Descenso Estocástico (SGD por su nombre en inglés Stochastic Gradient Descent). Los modelos fueron VGG-16 obteniendo una exactitud promedio del 73.3 %, Xception con 83.3 %, Inception v3 con 82.3 % y MobileNet v2 con 77.5 %.

Capítulo 3

Metodología

En este capítulo se presenta la propuesta metodológica para este trabajo de investigación. La cual tiene como propósito desarrollar un sistema de identificación y clasificación de enfermedades en plantas utilizando una metodología de visión por computadora para el procesamiento de imágenes y la aplicación de técnicas de aprendizaje de máquina para la creación de modelos de clasificación.

Se utilizarán las hojas como principales órganos distintivos. Algunos síntomas de ToBRFV en las hojas de tomate incluyen:

- Manchas cloróticas o patrones de mosaico en las hojas
- Curvatura o distorsión de las hojas
- Amarillamiento de las hojas
- Necrosis o muerte del tejido de la hoja
- Manchas o lesiones marrones en las hojas
- Rugosidad en forma de ampollas
- Enanismo del crecimiento de la planta
- Reducción del rendimiento de los tomates

En la Figura 3.1 se puede ver un esquema de una metodología de visión por computadora general y los módulos con los que esta cuenta.

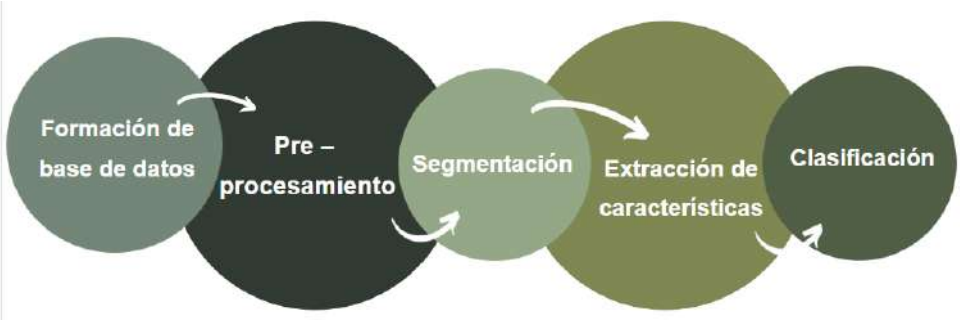

Figura 3.1: Diagrama de la metodología general de un proceso de clasificación de imágenes de plantas o enfermedades en plantas.

3.1. Propuesta metodológica

La propuesta metodológica para este trabajo de investigación se muestra en la Figura 3.2. La primera etapa es la adquisición de imágenes, la cual es fundamental en cualquier sistema de detección y clasificación debido a que las imágenes deben de tener la calidad suficiente para poder extraer características distintivas del problema en cuestión. Se propone hacer una búsqueda exhaustiva de repositorios libres con los cuales poder trabajar. Sin dejar de lado el trabajo de campo, ya que, debido a la novedad del virus rugoso del tomate, la disponibilidad de repositorios de calidad es limitada, sino que nula.
La siguiente etapa es el preprocesamiento, en esta etapa se analizan las técnicas que permiten la reducción de ruido y realzan los detalles.

En la etapa de segmentación el objetivo es aislar el objeto de interés del resto de la imagen. Para esta etapa se implementarán técnicas basadas en el color de la imagen. En particular, se utilizó la técnica de segmentación OTSU.

Para el caso de la extracción de características se utilizarán técnicas descritas en el estado del arte con el fin de obtener un conjunto de características idóneas para diferenciar los tipos de objetos y clases entre sí. Para finalizar tenemos la clasificación, en ella se exploran variadas técnicas de aprendizaje de máquina para el desarrollo de modelos de clasificación.

Figura 3.2: Propuesta metodológica de visión por computadora realizada en este trabajo de tesis.

3.2. Ubicación geográfica del lugar de toma de fotografías

La adquisición de fotografías se realizó en la Facultad de Ciencias Químico-Biológicas de la Universidad Autónoma de Sinaloa, bajo la supervisión del Dr. José Antonio Garzón Tiznado, ubicada en Calzada de las Américas Nte 2771, Cd Universitaria, 80030 Culiacán Rosales, Sin. 24°49'34.19"N, 107°22'55.45"O. En la Figura 3.3 podemos ver como se ve el interior del invernadero.

La toma de fotografías requiere ser de una forma muy específica para la obtención óptima de los datos. El primer problema con la obtención de los datos fue la dificultad de encontrar un lugar donde se permitiera la toma de fotografías. Una vez se pudo contar con el lugar fue notable que la cantidad de plantas posibles a fotografiar era poca y que además las condiciones para la toma de fotografía no eran las óptimas, sin embargo, se comenzó con la obtención de los datos.

Figura 3.3: Imagen del interior del invernadero de la base de datos.

3.3. Información biológica

La siguiente información fue proporcionada por la alumna encargada del experimento en la Facultad de Ciencias Químico-Biológicas.

Se utilizó Solanum lycopersicum variedad Río Grande, como hospedadoras del virus. De material enfermo recolectado en campo, se tomó tejido foliar y se maceró en un mortero frío estéril, la pulpa se exprimió a través de una gasa y se recogieron 2 mL de savia El extracto recién obtenido se utilizó para inocular plantas sanas de tomate, lo que permitió determinar si el virus seguía viable y aislarlo.

Para la inoculación mecánica por frotis se siguió la metodología descrita, cada planta fue espolvoreada con fibra de vidrio de manera ligera y uniforme, que actuó como abrasivo, posteriormente se frotó el extracto con ayuda de un hisopo de algodón, y se lavó con agua destilada para retirar el exceso de abrasivo que pudiera dañar el tejido. Las plantas se desarrollaron en invernadero, aisladas y bajo condiciones óptimas de crecimiento.

Se realizaron evaluaciones diarias durante 30 días, contándose el número de plantas sintomáticas, y se utilizaron tiras reactivas (Agdia-ImmunoStrip® para TMV) para detectar de forma temprana aquellas plantas infectadas con ToBRFV.

Todas estaban distribuidas de manera individual y sin contacto entre las diferentes clases. No existía ningún tipo de control de temperatura o humedad.

En el invernadero presentado se contaba con 4 tipos de clases posibles a fotografiar, las cuales son las siguientes:

- Tomate Sano
- Tomate ToBRFV
- Chile Sano
- Chile ToBRFV

3.4. Base de datos adquirida

Se lograron obtener alrededor de 200 fotografías de la clase de tomate sano y de la clase de tomate con ToBRFV. Todas ellas tomadas enfocadas en la hoja y con fondo blanco. Se muestra a continuación la Figura 3.4 en la cual aparecen una muestra de cada uno de los conjuntos de datos de las clases.

Del resto de las clases se obtuvieron por lo menos 100 imágenes de cada una, sin embargo, una vez finalizó el experimento en la Facultad de Ciencias Químico – Biológicas se nos informó que los experimentos con chile no dieron como resultado plantas enfermas, es decir que la inoculación de ellas no fue efectiva, esto debido a algún gen particular del tipo de chile

que rechazo los patógenos inducidos en ellos. Razón por la cual solo trabajaremos con un problema binario, como está establecido en el Capítulo I de este documento.

Las fotos tomadas fueron del tipo RGB de hoja por hoja en fondo blanco con el fin de tener un mejor procesamiento de los datos y fueron tomadas con luz natural y dentro del invernadero con un celular LG LM-K52HM con dimensiones de 4000x1800 y 7.2MP sin flash y con fondo blanco mediante hoja de papel. Se obtuvieron a día de hoy 69 fotografías de tomate sano, 98 fotografías de tomate con ToBRFV, 61 fotografías de chile sano, 77 fotografías de chile con ToBRFV y 60 fotografías de chile con Tospovirus.

A pesar de todo este trabajo realizado, solo se tomaron en cuenta las dos clases comentadas en la sección anterior. Esto debido a que al finalizar el experimento realizado en la Facultad de Ciencias Químico-Biológicas, los expertos nos comentaron que las especies de chile no retuvieron el Virus Rugoso del Tomate que les fue inoculado.

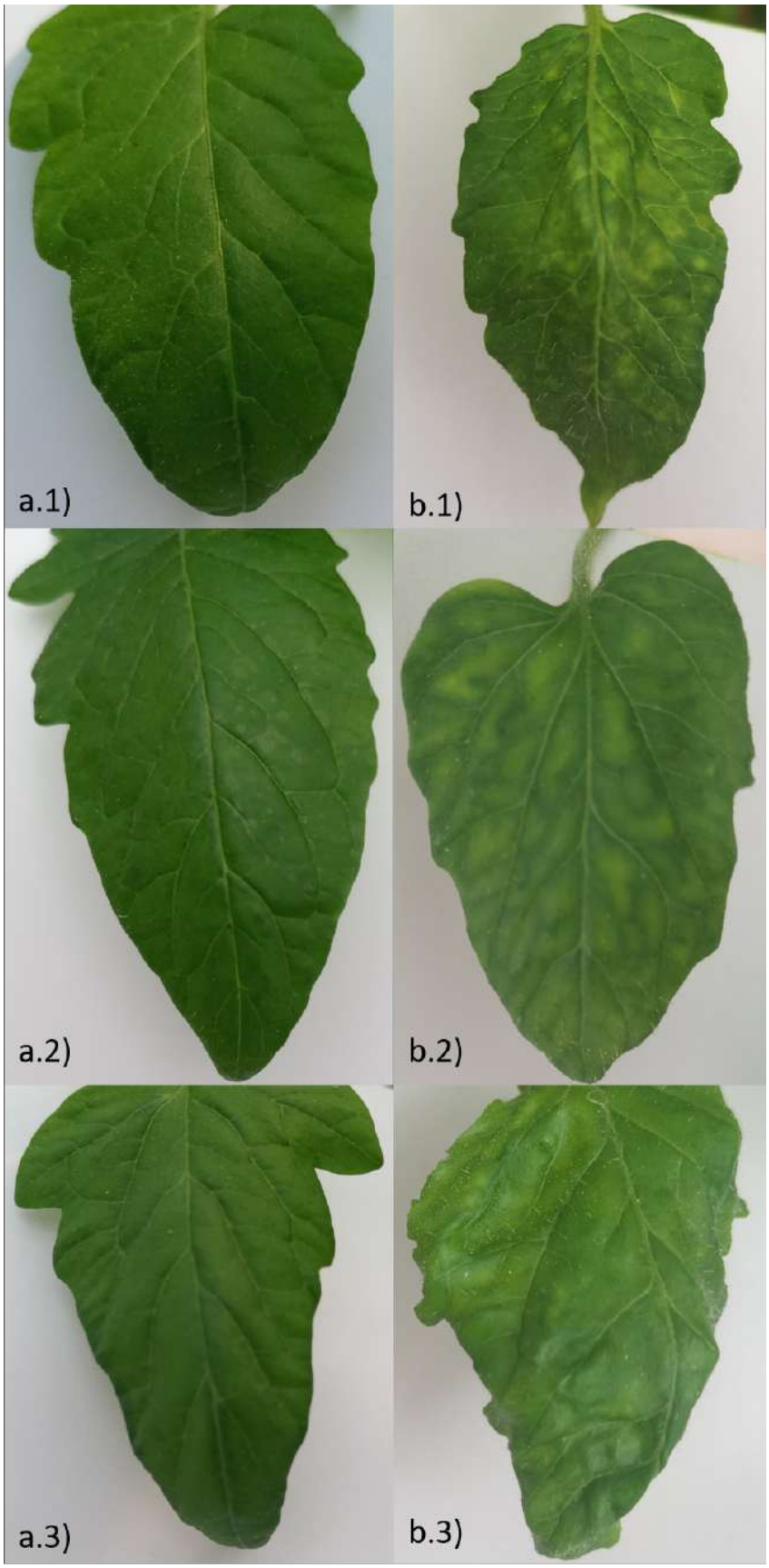

Figura 3.4: Ejemplo de las fotografías tomadas de cada clase. a.1),a.2),a.3) Tomate sano, b.1),b.2),b.3) Tomate con ToBRFV.

3.5. Preprocesamiento

En el caso del preprocesamiento se comienza con el recorte de las imágenes. Estos recortes fueron realizados de manera manual debido a las circunstancias de las imágenes. Una vez se obtuvieron los recortes se procedió a aplicar algunas técnicas de aumento de datos con el fin de tener una base de datos mucho más amplía.

Se comenzó con un total de 122 imágenes luego de depurar el conjunto de datos y separar un total de 50 instancias por clase a la cuales no se les aplicaría todo este proceso, esto debido a que estas 50 instancias serán nuestra muestra de prueba una vez tengamos nuestro clasificador.

Los procesos que se le aplicaron a las imágenes con su respectiva sección de código fueron los siguientes:

- Rotación a 90, 180 y 270 grados.

```
image = cv2.imread('t_sano'+str(i)+'.jpg')
imgr = rotate(image, 90, resize=True)
imgr = cv2.normalize(imgr, None, 0, 255,
cv2.NORM_MINMAX, cv2.CV_8U)
```

- Transformación espejo vertical y horizontal.

```
#espejo vertical
image = Image.open('t_sano'+str(i)+'.jpg')
imgf = image.transpose(method=Image.FLIP_TOP_BOTTOM)
imgf.save('t_sano'+str(i+488)+'.jpg')
#espejo horizontal
image = Image.open('t_sano'+str(i)+'.jpg')
imgf = image.transpose(method=Image.FLIP_LEFT_RIGHT)
imgf.save('t_sano'+str(i+610)+'.jpg')
```

Y lo vemos de manera visual en la figura 3.5.

Dándonos como resultado un total de 732 imágenes por clase para procesar.

3.6. Segmentación

La fase de segmentación fue aplicada una vez se tenían las 732 fotos obtenidas de la fase de

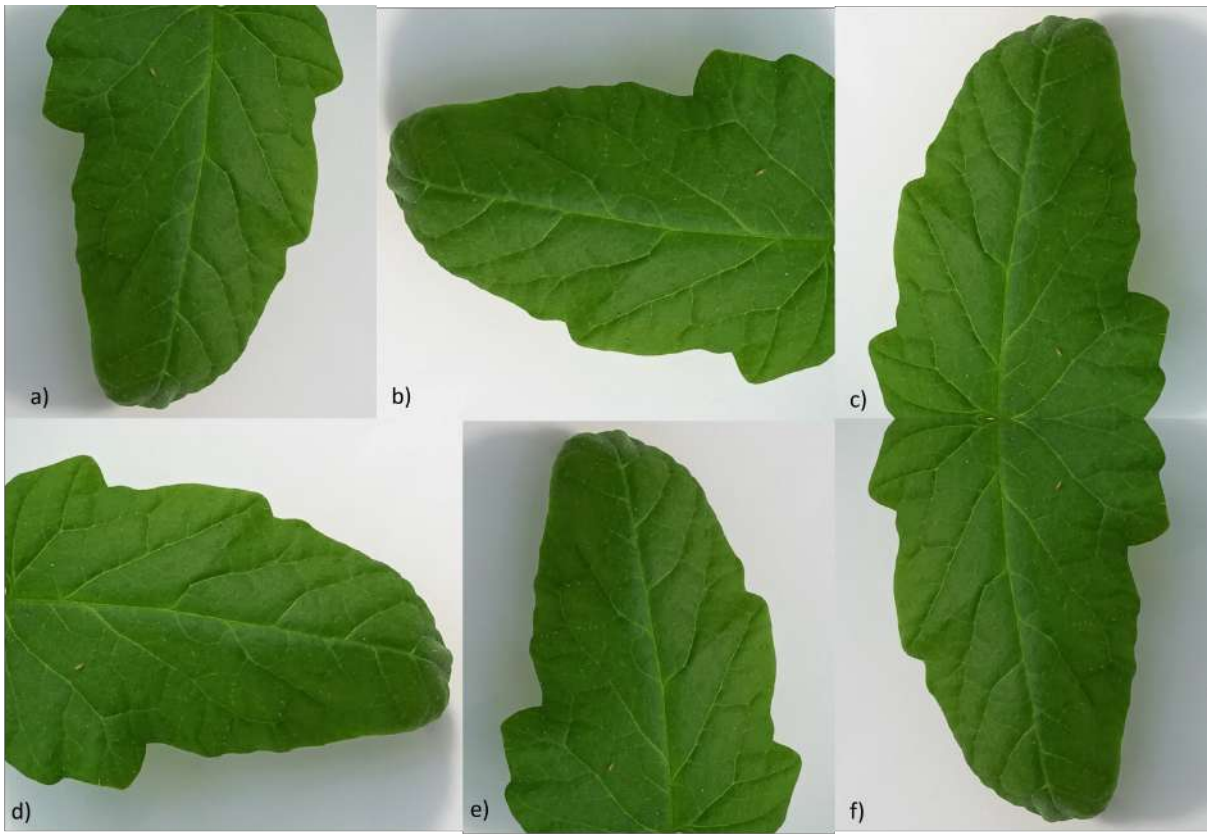

Figura 3.5: Ejemplo de las rotaciones ejercidas sobre cada una de las imágenes. a)Original, b)Rotación de 90 grados, c)Rotación de 180 grados, d)Rotación de 190 grados, e)Rotación espejo vertical y f)Rotación espejo horizontal.

3.6.1. Fase 1: Umbralización adaptativa

La primera fase de la segmentación es hecha de manera automática mediante el método de OTSU (OTSU, 1979), el cual es un método de umbralización adaptativa que toma en consideración la distribución de los valores según su histograma. A partir de esto toma el valor adecuado para hacer una máscara, en este caso se configura para que sea una máscara binaria donde el área de interés tenga un valor de 1 y el resto un valor de 0. Esto se hace a partir de la imagen convertida en escala de grises, como se muestra en la sección de código a continuación.

```
img = cv2.imread('t_sano'+str(i)+'.jpg')
ib,ig,ir=cv2.split(img)

gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
_, thresh = cv2.threshold(gray,0,1,
cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
```

Como se ha mencionado previamente, las imágenes a color o imágenes

en el modelo de color RGB están compuestas por tres canales: azul, verde y rojo. En esta etapa del proceso, la imagen original se divide en sus tres canales individuales. Estos canales, también conocidos como máscaras de color, se combinan con la máscara binaria resultante del proceso de segmentación OTSU. Para lograr la segmentación total, se realiza una multiplicación entre la máscara binaria y cada canal de color. Dando como resultado tres imágenes segmentadas, una con valores de su respectivo canal de color. Una vez que se han obtenido las nuevas máscaras segmentadas, se vuelven a combinar para obtener la imagen RGB de la hoja segmentada. Esta operación se muestra en la siguiente sección de código. Así como los resultados de este proceso en la Figura 3.6.

```
segr=cv2.multiply(thresh,ir)
segb=cv2.multiply(thresh,ib)
segg=cv2.multiply(thresh,ig)
segm=cv2.merge([segb,segg,segr])
```

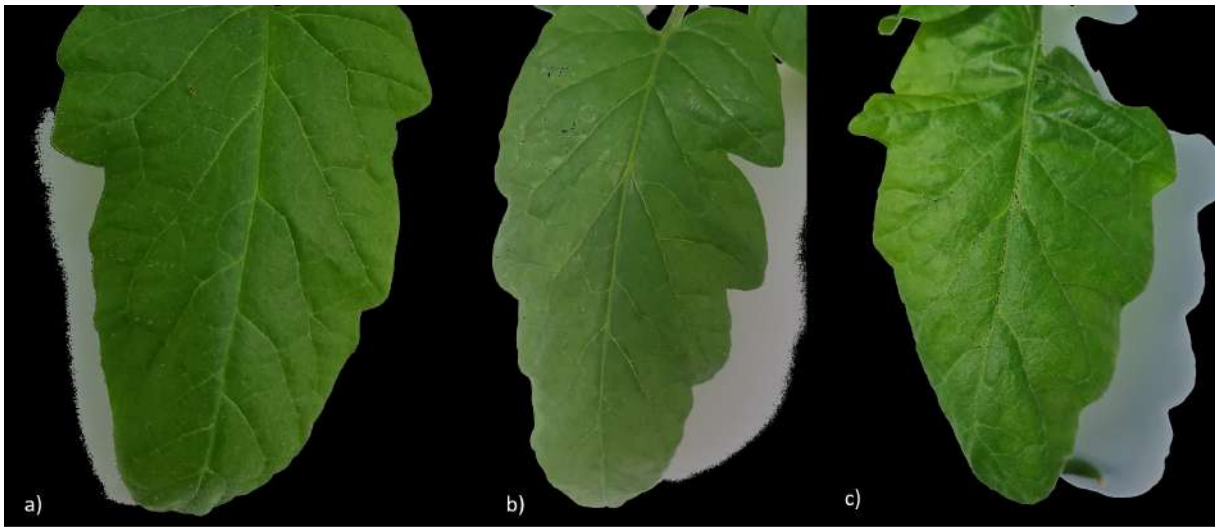

Figura 3.6: Ejemplo de algunas fotografías segmentadas de la clase sana después de la primera etapa.

3.6.2. Fase dos: Eliminación manual de sombra

La segunda parte de la segmentación surge a partir de que la segmentación anterior no eliminaba por completo el fondo al dejar no solo la hoja sino su sombra.

La solución a esto se dio mediante un análisis de la composición de los canales de color en el área de la sombra de la hoja. Este análisis dio como

resultado reconocer que el canal que estaba manteniendo la sombra es el canal azul. En el área de la sombra el valor del canal azul es mucho más alto que en el área de superficie de la hoja, por lo tanto, se hace una depuración mediante este canal respecto a los valores de cada pixel. Haciendo que estos píxeles tengan el valor de cero para hacer de esta segmentación la mejor posible.

A continuación se muestra la sección del código en la se realiza esta acción y los resultados de este proceso en la Figura 3.7.

```
img = cv2.imread('t_sano_seg'+str(r)+'.jpg')
ib,ig,ir=cv2.split(img)

m,n=np.shape(ib)
for i in range (0,m-1):
    for j in range (0, n-1):
        if ib[i][j]>100 :
            ib[i][j]=0
            ig[i][j]=0
            ir[i][j]=0

out1=cv2.merge([ib,ig,ir])
```

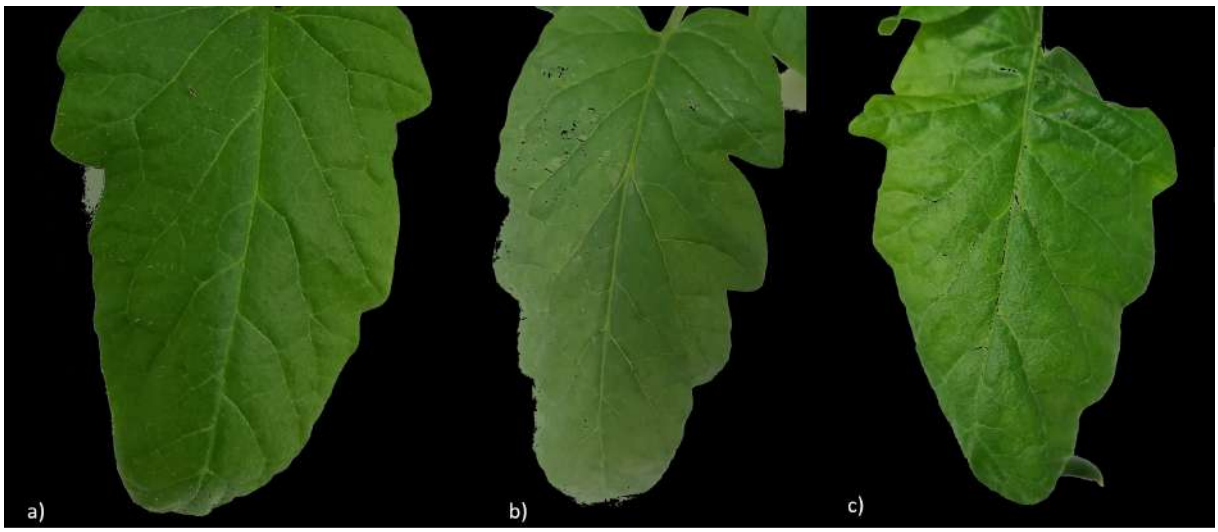

Figura 3.7: Ejemplo de algunas fotografías segmentadas de la clase sana después de la segunda etapa.

3.6.3. Comparación de las etapas de segmentación con las imágenes originales

En la Figura 3.8 podemos ver de manera más comparativa cómo quedan las imágenes después de cada proceso.

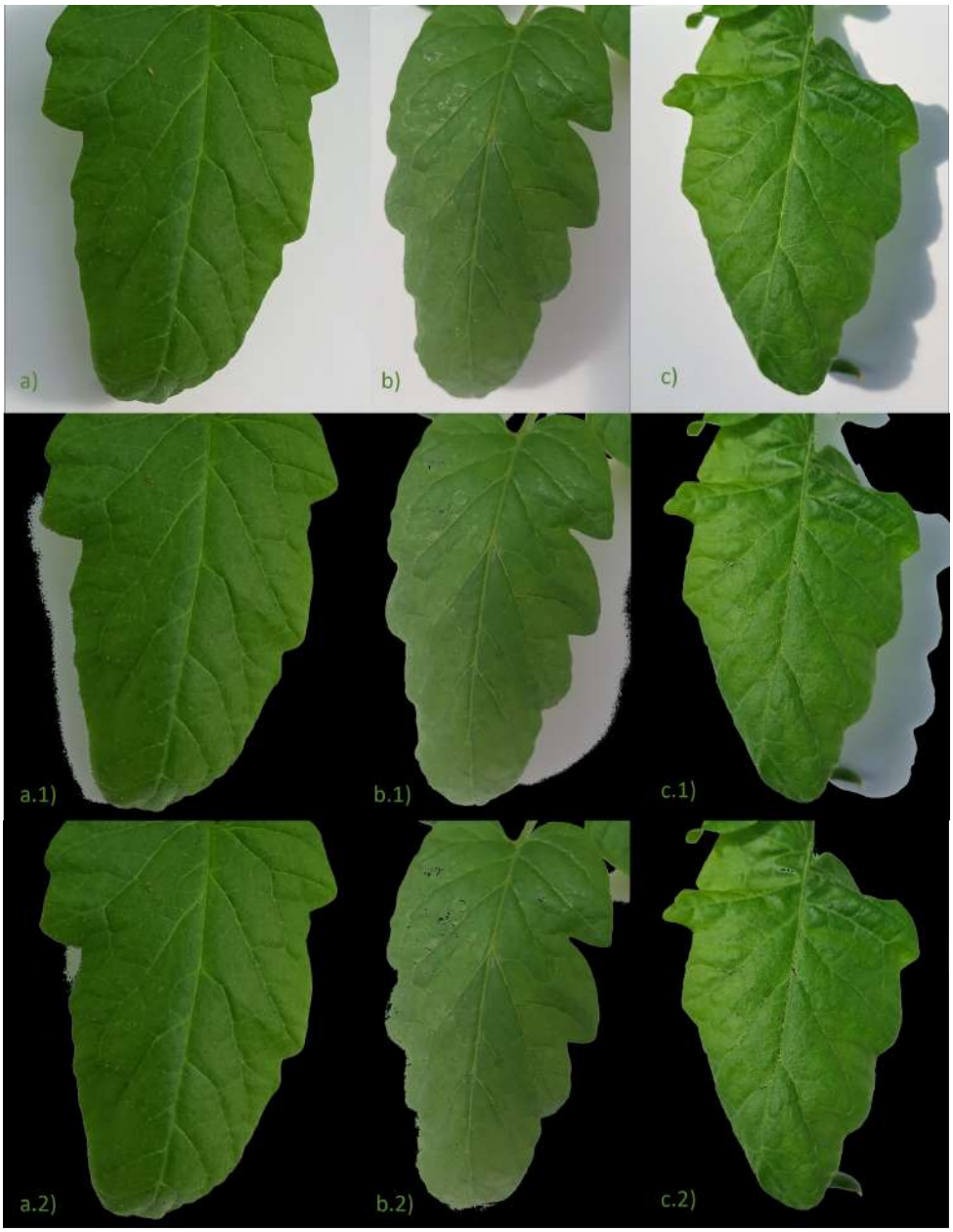

Figura 3.8: Comparación de las diferentes etapas de la segmentación, De manera descendente tenemos imágenes originales, primera etapa de segmentación y segunda etapa de segmentación.

3.7. Extracción de características

Para la extracción de características se comenzó con un análisis de las imágenes completas.

3.7.1. Matriz de coocurrencia a nivel de escala de grises

Se les aplicó la Matriz de coocurrencia a nivel de escala de grises de la siguiente manera. Primero que nada como el nombre lo indica, este proceso se trabaja a escala de grises, por lo cual se hace una transformación de la imagen original a este formato.

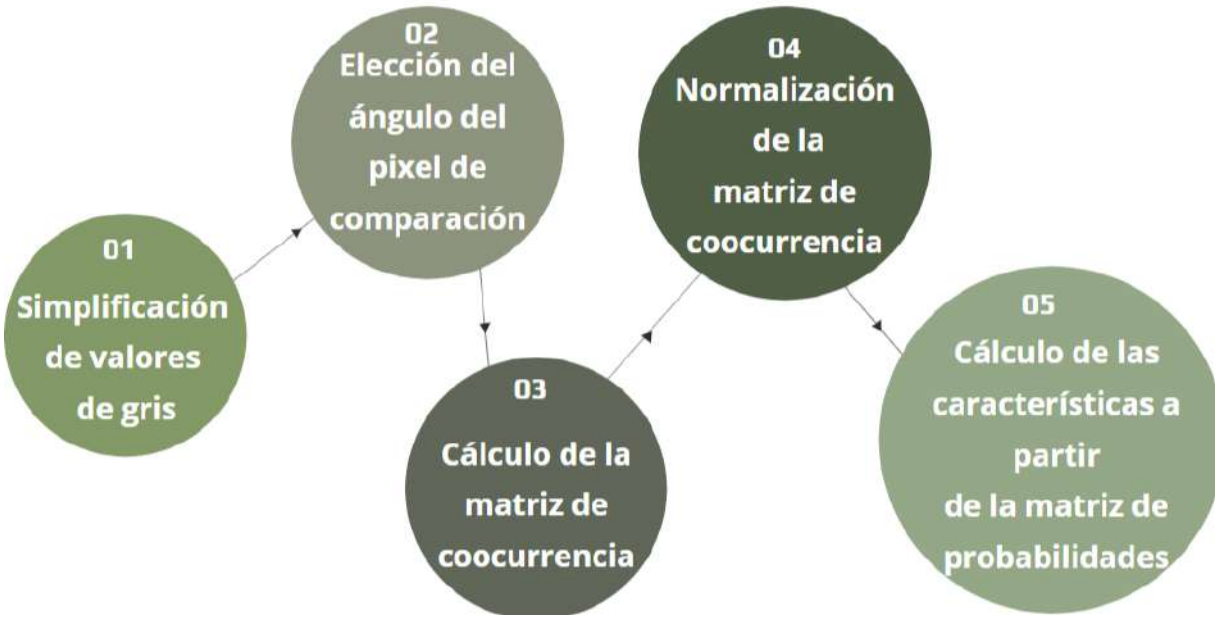

Figura 3.9: Diagrama de la metodología de la GLCM.

El proceso englobado que realiza la GLCM se puede ver en la Figura 3.9. Para hacer de este proceso uno más rápido, lo siguiente fue cambiar los niveles de gris que hay de 255 a 16, esto con el fin de que el procesamiento de todo el conjunto de datos no sea algo que demande demasiado tiempo de procesamiento, ya que como se comentó en el marco teórico, este procedimiento lo que hace es calcular cuantas veces un valor de gris está en cierta posición respecto a otro, por lo cual, en lugar de tener que evaluar 65,025 posibilidades de combinaciones de valores de pixel se prueban solamente 256 posibles combinaciones. El proceso continúa con el diseño de nuestra matriz de coocurrencia, lo más importante es el ángulo a tomar en cuenta para el análisis de las combinaciones de valores de gris. Recordemos que tenemos 8 posibilidades, una para cada pixel vecino posible, cuatro en las direcciones que conocemos como norte, sur, este y oeste y cuatro más para cada vecino en diagonal. Además, podemos elegir a que distancia entre píxeles queremos que se haga esta comparación. En este caso se toman en cuenta los 4 propuestos por el estado del arte que son norte, sur, este y oeste y a una distancia de 1 pixel.

Una vez es creada nuestra matriz de coocurrencia, se procede con el cálculo de las características a partir de ella, para lo cual existen funciones definidas, las características calculadas son las siguientes:

- Contraste
- Disimilaridad
- Homogeneidad
- Energía
- Correlación
- ASM

El cálculo de estas características es almacenado en dos archivos diferentes, uno en el cual tenemos los datos obtenidos en cada una de las 4 direcciones que elegimos como valores individuales y otro en el cual se hace un promedio de estos 4 valores registrando únicamente este valor y no los individuales.

3.7.2. Procesamiento de un área de interés

Con el objetivo de obtener los mejores resultados se decidió hacer un estudio más direccionado hacia el centro de la imagen con la esperanza de obtener la información más significativa de las hojas. Para ello se tomó en cuenta las dimensiones de cada imagen de manera individual. A partir de esto se encontró el centro de la imagen y se tomó como el centro del recuadro a ser analizado.

De acuerdo a las dimensiones de las imágenes se decidió tomar un recuadro de $99x99$ píxeles con centro en el centro de la imagen. Esta referencia se trató de hacer desde las imágenes con menor área foliar. Por lo tanto, el cálculo de la matriz de coocurrencia y las características derivadas de ella se calcularon a partir de este recuadro solamente y no de toda la imagen como anteriormente se realizó. El propósito de esta nueva área de estudio tiene el fin de reducir el área de fondo que se está analizando y tratar de obtener las características más congruentes posibles con la hoja sin obstrucción del fondo.

En la figura 3.10 se muestra una imagen representativa del área que se estará tomando en cuenta respecto a algunas de las imágenes de la base de

datos. Del lado izquierdo tenemos una imagen de la clase sana y del lado derecho de la clase infectada con ToBRFV.

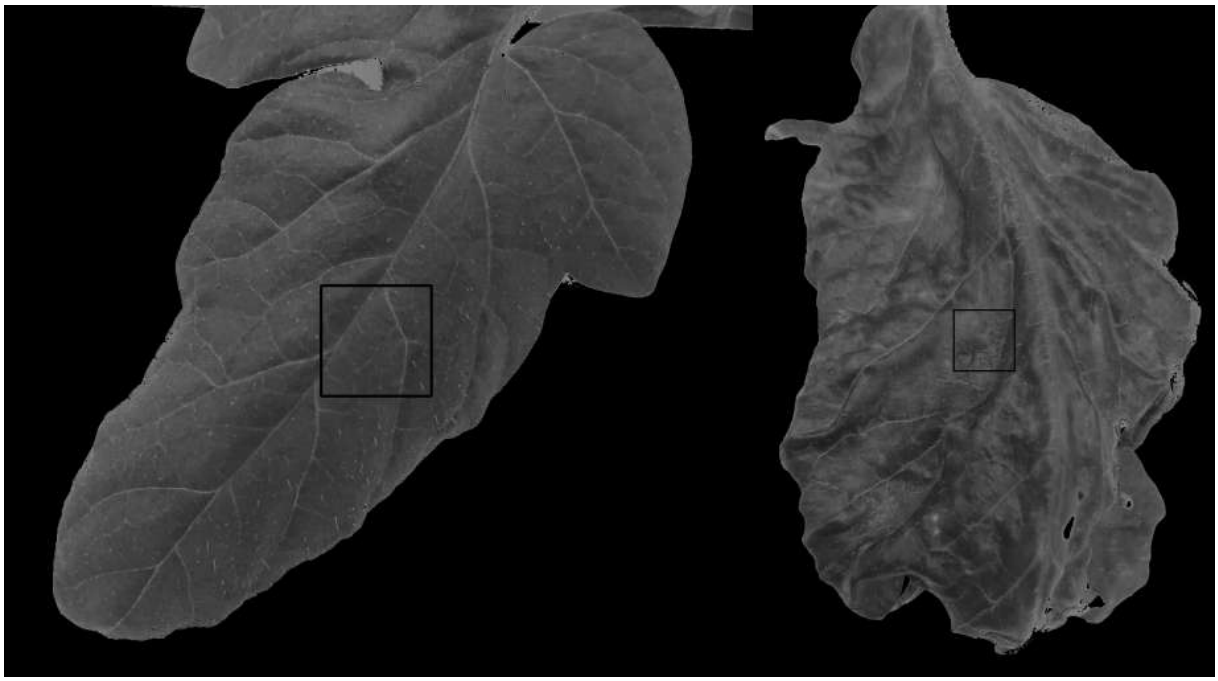

Figura 3.10: Muestra del área sobre la cual se hacen las evaluaciones respecto a una imagen de tamaño original.

3.7.3. Aplicación de los índices de vegetación NGBVI y NRBVI

Una vez se realizaron los procesos anteriores a las imágenes originales, se realizó la aplicación de los índices de vegetación NGBVI y NRBVI. Este tipo de procesamiento son cálculos algebraicos entre las bandas de color de la imagen.

Posteriormente a esta imagen resultante en escala de grises se le aplica una paleta de color, para mejor visualización de las diferencias entre los valores de la imagen. A continuación se muestra una sección del código donde se calculan estos índices.

```
#NGBVI
gb=GS-BS
max_gb = np.amax(gb)
NGBVI = (gb/max_gb)
resultimage = np.zeros((1500, 1500))
norm_img = cv2.normalize(NGBVI, resultimage ,
```

```
0, 255,cv2.NORM_MINMAX)

#NRBVI
rb=R-B
max_value = np.amax(rb)
NRBVI = (rb/max_value)
resultimage = np.zeros((1500, 1500))
norm_img = cv2.normalize(NRBVI,resultimage,
0, 255,cv2.NORM_MINMAX)
```

Este proceso nos da como resultado las siguientes imágenes mostradas en la Figura 3.11. En la parte superior de la figura tenemos el resultado para NGBVI y en la inferior para NRBVI. Así como en la parte izquierda están las imágenes resultantes originales y en la derecha están a las cuales se les ha aplicado la paleta de color tipo JET.

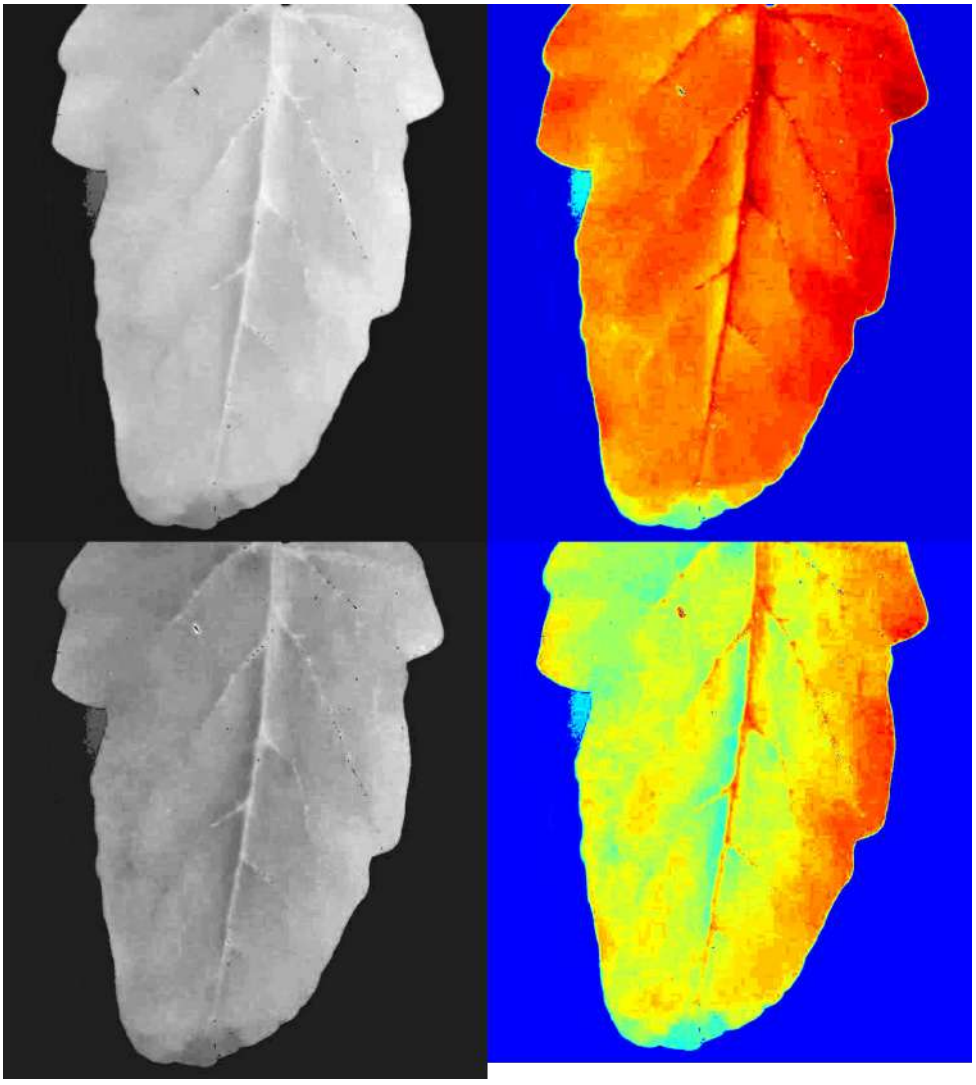

Figura 3.11: Comparación de los dos distintos índices aplicados a las imágenes.

3.7.4. Histogramas de sumas y diferencias

Otro método de extracción de características son los histogramas de sumas y diferencias. Este método fue elegido con base en los resultados del método GLCM anteriormente ejecutado.

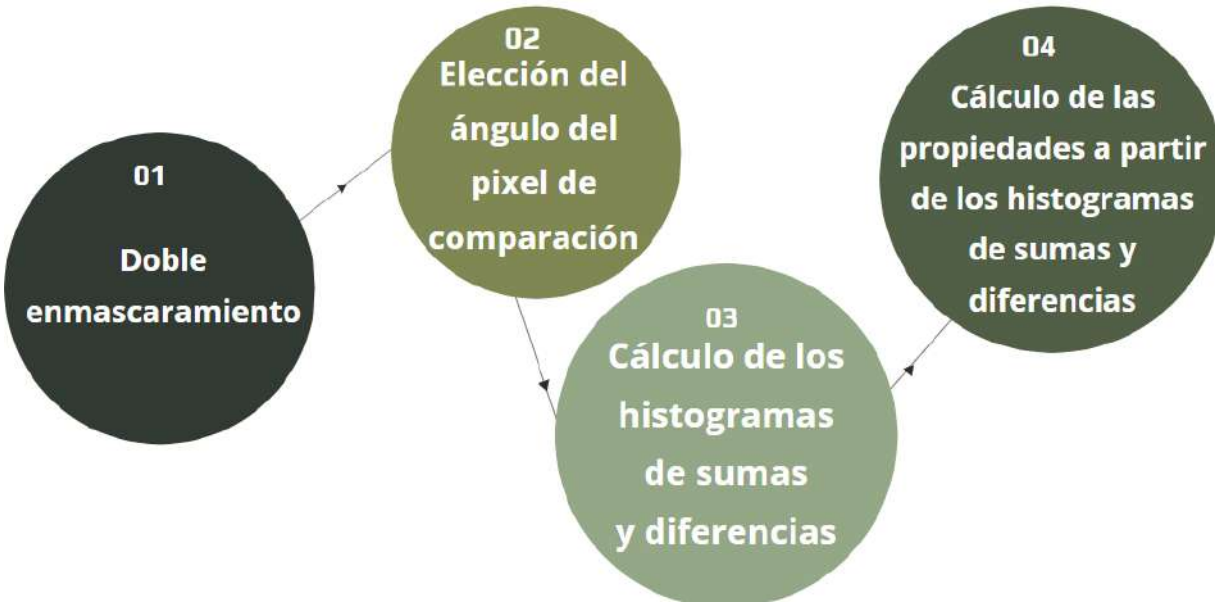

Figura 3.12: Diagrama de la metodología de la GLCM.

Los procesos realizados en este método se pueden ver en la Figura 3.12. Para este nuevo método se realizó previamente un doble enmascaramiento. Este doble enmascaramiento consistió primero en una máscara binaria de la imagen, resaltando solamente el área de la hoja con base en una segmentación del método OTSU como se muestra a continuación.

```
ib,ig,ir=cv2.split(img)
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
_, thresh = cv2.threshold
(gray,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)

#eliminacion de la sombra
thresh[ib>100]=0
```

A partir de esta primera máscara se construyó la segunda máscara. Para esto, se dilató la primera máscara con el fin de evitar los huecos más pequeños. La dilatación es una operación importante en el procesamiento de imágenes que se utiliza para ampliar o resaltar los objetos de interés en

una imagen, y puede mejorar la segmentación, la detección de bordes y la extracción de características en una imagen.

```
thresh = cv2.blur(thresh,(3,3))
thresh[thresh!=0]=1
```

Luego de la dilatación, se procedió a buscar los píxeles que cumplían el requisito necesario para su análisis. Este requisito consistía en que dentro de una matriz de una dimensión específica que rodeaba el píxel actual, no hubiera ningún píxel con valor cero. Si se cumplía esta condición, entonces el píxel era considerado válido para su evaluación mediante el método de extracción de características.

Para llevar a cabo esta parte del proceso, se utilizó un enfoque de recorrido de imagen en el que se examinó cada píxel de la imagen de forma secuencial. Para cada píxel, se definió una matriz de una dimensión específica que rodeaba al píxel actual. La dimensión de la matriz se determinó en función del tamaño del kernel utilizado en la dilatación.

Luego, se verificó si había algún píxel dentro de la matriz con valor cero. Si no había ningún píxel con valor cero, entonces se consideraba que el píxel actual cumplía con el requisito y se lo marcaba como válido para su evaluación mediante el método de extracción de características.

De esta forma, se logró identificar aquellos píxeles que cumplían con las condiciones necesarias para su análisis y se descartaron aquellos que no cumplían con estos requisitos. Este proceso de selección de píxeles válidos es crucial para garantizar la precisión y eficiencia del método de extracción de características utilizado en el análisis de imágenes.

```
def mask_hsyd(thresh,n):
x,y=np.shape(thresh)
thresh2=np.zeros((x,y,1),np.uint8)
for i in range ((n//2)+1,x-1-(n//2)):
    for j in range ((n//2)+1,y-1-(n//2)):
        roi=thresh[i-(n//2):i+(n//2),
        j-(n//2):j+(n//2)]
        if(np.all(roi==1)):
            thresh2[i][j]=1
return thresh2
```

Una vez termina el proceso del doble enmascaramiento se procede a

aplicar la evaluación del método de Histogramas de sumas y diferencias. Los histogramas de sumas y diferencias se usan comúnmente en el análisis de textura. Proporcionan una representación estadística de la relación entre los valores de píxeles vecinos en una imagen.

La primera parte del proceso consta de un análisis de comparación de píxeles vecinos, en este caso se tomaron en cuenta los píxeles vecinos a la derecha. Como el nombre del proceso lo dice, se realizan sumas y diferencias entre estos píxeles, ambos de manera individual, por lo cual, se tendrán como resultado dos valores distintos, la suma y la diferencia no normalizadas, asociados con el desplazamiento relativo de estos valores. Como la imagen tiene un rango de valores que van de 0 a 255, las variables de la suma y diferencia entre dos píxeles será del doble, es decir 511.

Este proceso se realiza mediante ventanas de dimensiones nxn donde n es adaptable dependiendo la necesidad del detalle y las medidas de la imagen.

El siguiente paso es guardar esta información en una variable, por lo que tendremos una lista o vector de valores donde cada punto es representado por la respuesta al procesamiento de estas ventanas. Estas listas son mejor conocidas como los histogramas de sumas y diferencias. Los cuales son normalizados posteriormente.

```
for i in range ((n//2)+1,x-1-(n//2)):
    for j in range ((n//2)+1,y-1-(n//2)):
        if (thresh2[i][j]==1):
            #print(i,j)
            roi=img[i-(n//2):i+(n//2),
            j-(n//2):j+(n//2)]
            hs=np.zeros(m,np.uint8)
            hd=np.zeros(m,np.uint8)
            for p in range(0,n-2):
                for q in range(0,n-1):
                    s=int(roi[p][q])
                    +int(roi[p+1][q])
                    r=int(roi[p][q])
                    -int(roi[p+1][q])
                    hs[s]+=1; hd[r+255]+=1
```

```
        for k in range(0,m):
            hsn[k]=(float(hs[k]))/(m)
            hdn[k]=(float(hd[k]))/(m)
```

A partir de estos resultados y con base en las ecuaciones anteriores, se procede con los cálculos para extraer las características de textura de la imagen. Estos cálculos están dados de la siguiente manera:

```
#mediana
def mediahsyd(hs,m):
    s=0.0
    for i in range (0,m-1):
        s+=float(i*hs[i])
    s=s/2
    return s

#varianza
def varianzahsyd(hs,hd,m,med):
    s=0; r=0
    for i in range (0,m-1):
        s+=float((i-(2*med))*(i-(2*med))*hs[i])
    for j in range(0,m-1):
        r+=float((j-255)*(j-255)*hd[j])
    return (s+r)/2

#correlacion
def correlacionhsyd(hs,hd,m,med):
    s=0; r=0
    for i in range (0,m-1):
        s+=float((i-(2*med))*(i-(2*med))*hs[i])
    for j in range(0,m-1):
        r+=float((j-255)*(j-255)*hd[j])
    return (s-r)/2

#energia
def energiahsyd(hs,hd,m):
    s=0; r=0
    hs=cv2.multiply(hs,hs)
    s=sum(hs)
    hd=cv2.multiply(hd,hd)
```

```
    r=sum(hd)
    return s*r

#contraste
def contrastehsyd(hd,m):
    r=0
    for j in range(0,m-1):
        r+=(j-255)*(j-255)*hd[j]
    return r

#homogeneidad
def hmghsyd(hd,m):
    r=0
    for j in range (0,m-1):
        r+=float((1/(1+((j-255)*(j-255))))*hd[j])
    return r
```

Para finalizar este proceso, los histogramas son calculados y posteriormente normalizados.

3.7.5. Procesamiento de toda el área de la hoja mediante Histogramas de sumas y diferencias

El primer modo en que se realizó la extracción de características mediante el método de histogramas de sumas y diferencias fue aplicándolo a toda el área de la hoja.

El objetivo era obtener la máxima cantidad de información posible. Incluso teniendo en cuenta las limitaciones de la base de datos mencionadas anteriormente, se esperaba lograr resultados satisfactorios con este nuevo enfoque.

Después de aplicar este proceso, se hizo evidente que el hecho de enfocarse únicamente en áreas pequeñas que presentan síntomas claros en relación con el tamaño de la hoja y el fondo no resulta muy útil. Esto se debe a que la información prioritaria se diluye entre los datos generales, que son predominantemente más numerosos.

Por esto extraer información de toda la hoja no es eficiente para los resultados deseados. Por lo tanto, es más beneficioso concentrarse en áreas más

específicas y con síntomas más claros para obtener la información deseada.

3.7.6. Procesamiento de recortes de áreas de interés mediante Histogramas de sumas y diferencias

Como último procesamiento se decidió volver a implementar la metodología de histogramas de sumas y diferencias justamente descrita en el proceso anterior. En este nuevo proceso se utilizaron recortes de las imágenes originales completas como entrada de nuestro proceso de extracción. Estos recortes contenían en su totalidad solo píxeles hoja, es decir, no existía ningún pixel de fondo.

Los recortes fueron hechos de manera manual con el fin de obtener la mayor información de cada una de las imágenes de la base de datos. Esto trajo como consecuencia que las imágenes tuvieran tamaños distintos entre sí. Se planteó la posibilidad de re-escalar estas imágenes o buscar alguna otra manera de homogeneizar los tamaños de estas, pero, todos estos procesos podían causar una pérdida de información. Principalmente, a razón de que los histogramas de sumas y diferencias se basan en la distribución espacial de la intensidad de los píxeles, y las alteraciones de re-escalado de las imágenes perjudican estas distribuciones.

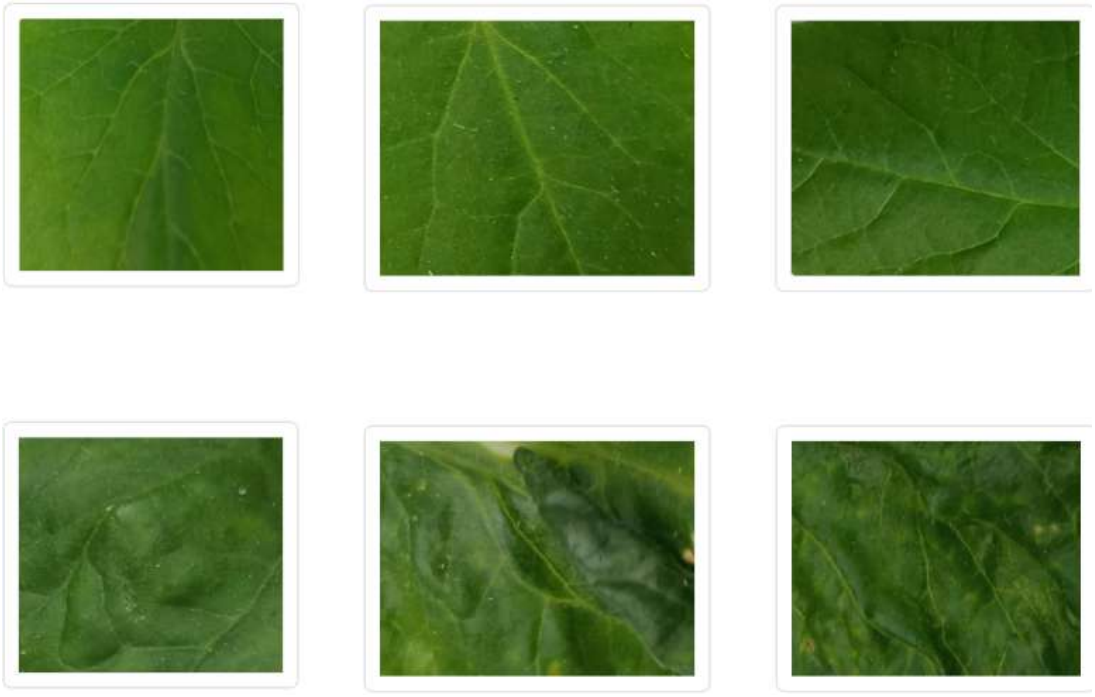

Figura 3.13: Diagrama de la metodología de la GLCM.

En la Figura 3.13 se observa una muestra de los recortes utilizados en este nuevo proceso. El resto de este proceso fue tal cual se describió en la sección anterior mediante histogramas de sumas y diferencias.

Capítulo 4

Resultados y Discusión

4.1. Resultados de extracción de características

En este apartado se irán mostrando y explicando los resultados de las distintas etapas y tipos de metodologías que se utilizaron para lograr obtener los resultados deseados.

Se observaron resultados en ciertas etapas con el objetivo de decidir si el trabajo realizado hasta ese momento era lo suficientemente bueno para seguir sobre esa línea o cambiar el rumbo. Como se ha mencionado, la metodología eficaz para la detección de enfermedades en plantas no es una sola para todas. El estado del arte nos muestra y dice que dependiendo el caso de estudio se va detectando sobre la marcha que sirve y que no para esa detección o clasificación en particular.

Los resultados mostrados en esta sección están descritos en orden cronológico, y se están mostrando aun cuando en ciertos casos después de observarlos se decidió no continuar sobre esa línea por no ser lo suficientemente buenos. Esto hará parecer algunos procesos como inconclusos, sin embargo, solo se trata de un cambio de enfoque dentro de la metodología.

4.1.1. Mediante la matriz de coocurrencia a nivel de escala de grises

A continuación se mostrarán los primeros resultados del análisis de la extracción de características por medio de la matriz de coocurrencia a nivel de escala de grises. Esta primera etapa se realizó a partir de las imágenes RGB segmentadas y se tomó en cuenta toda la imagen al momento de hacer los cálculos. Partiendo de esto, se muestran los siguientes resultados. Primero procedemos a ver cada característica de manera individual,

para los 4 ángulos distintos establecidos en la metodología. La Figura 4.1 muestra los resultados para la característica de contraste.

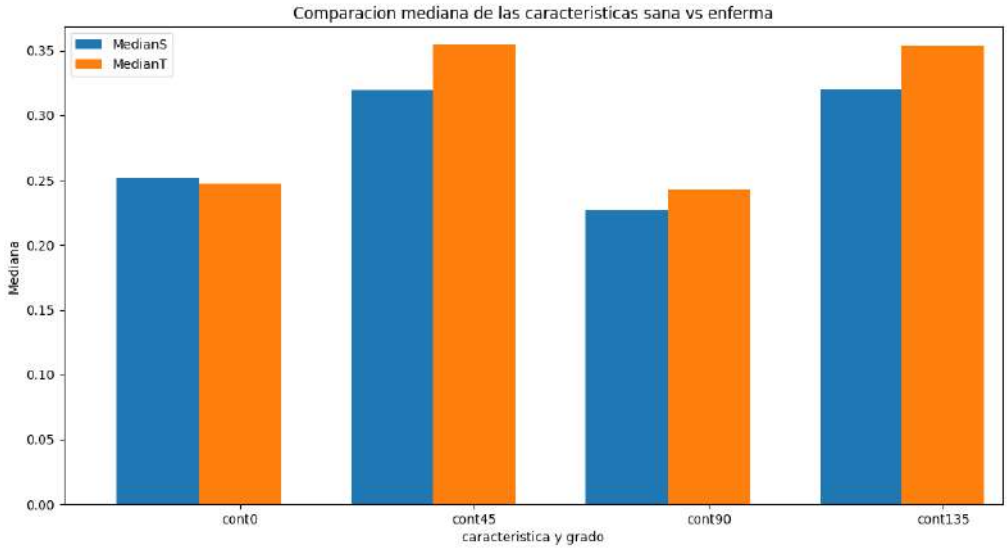

Figura 4.1: Gráfica de barras representante del contraste para los 4 ángulos definidos.

La Figura 4.2 muestra los resultados para la característica de disimilaridad.

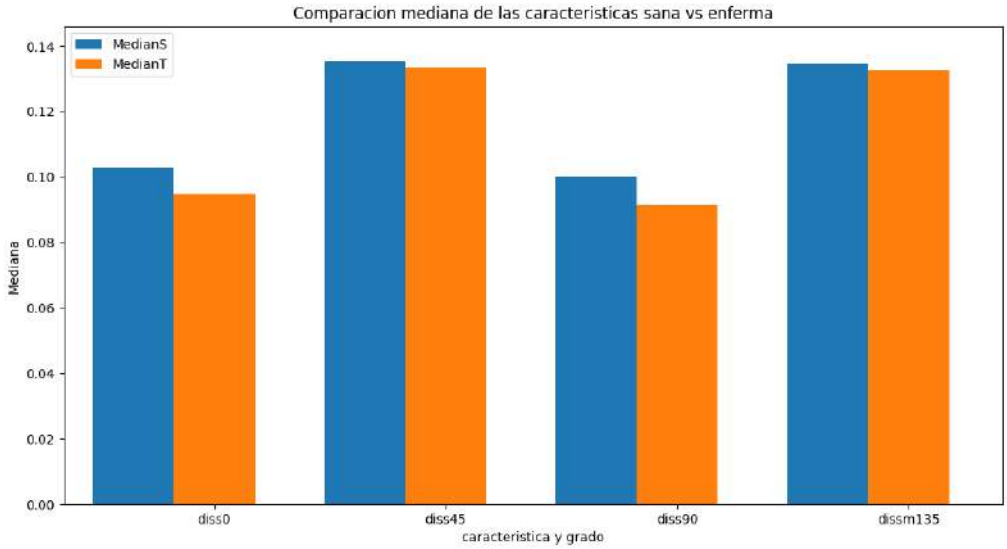

Figura 4.2: Gráfica de barras representante de la disimilaridad para los 4 ángulos definidos.

La Figura 4.3 muestra los resultados para la característica de homogeneidad.

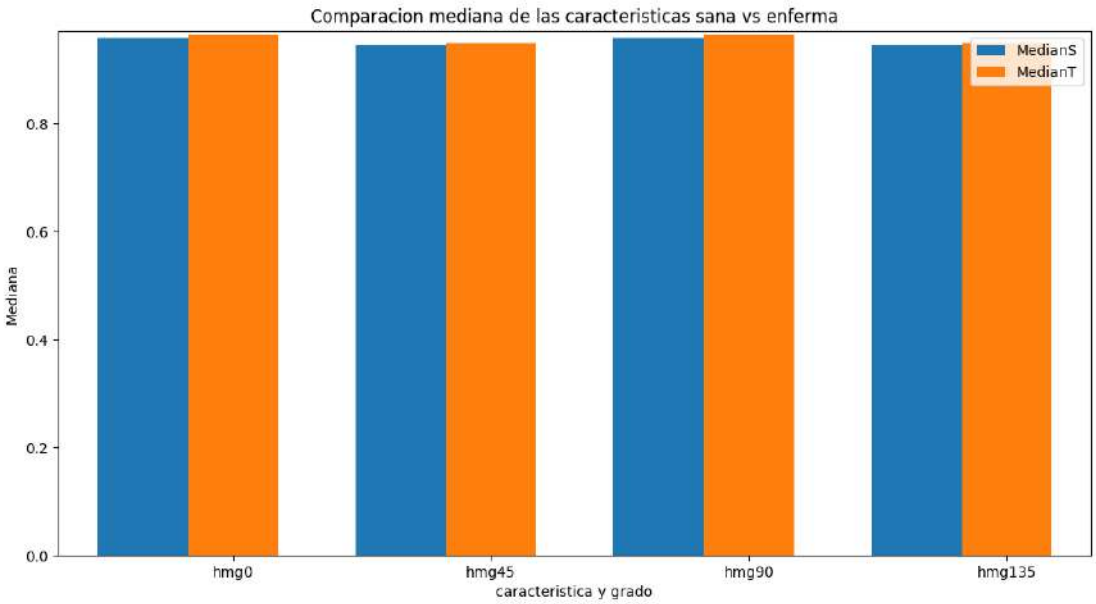

Figura 4.3: Gráfica de barras representante de la homogeneidad para los 4 ángulos definidos.

La Figura 4.4 muestra los resultados para la característica de energía.

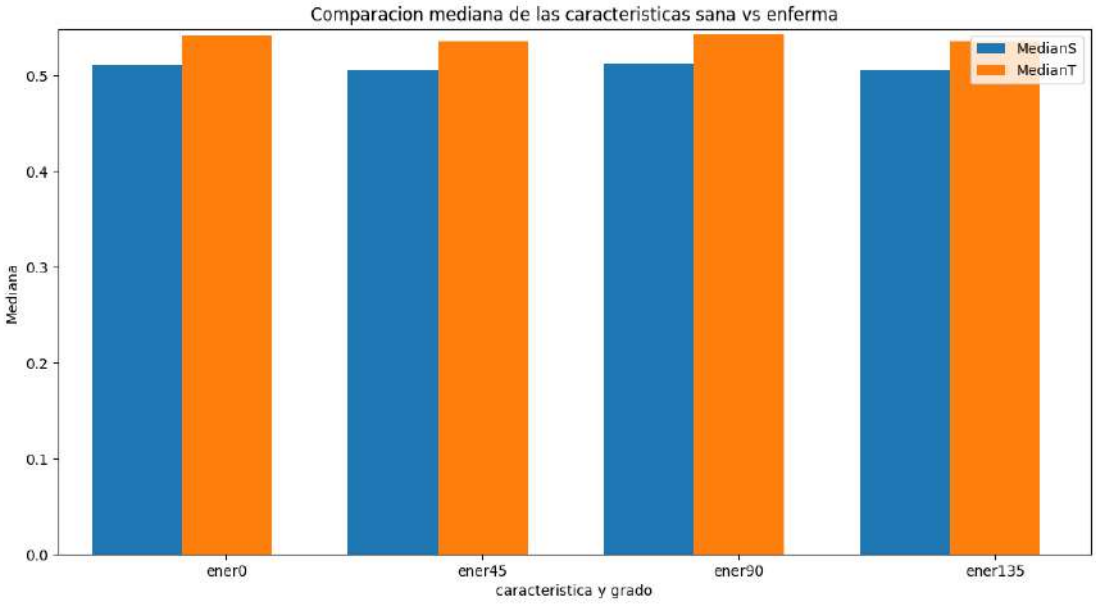

Figura 4.4: Gráfica de barras representante de la energía para los 4 ángulos definidos.

La Figura 4.5 muestra los resultados para la característica de correlación.

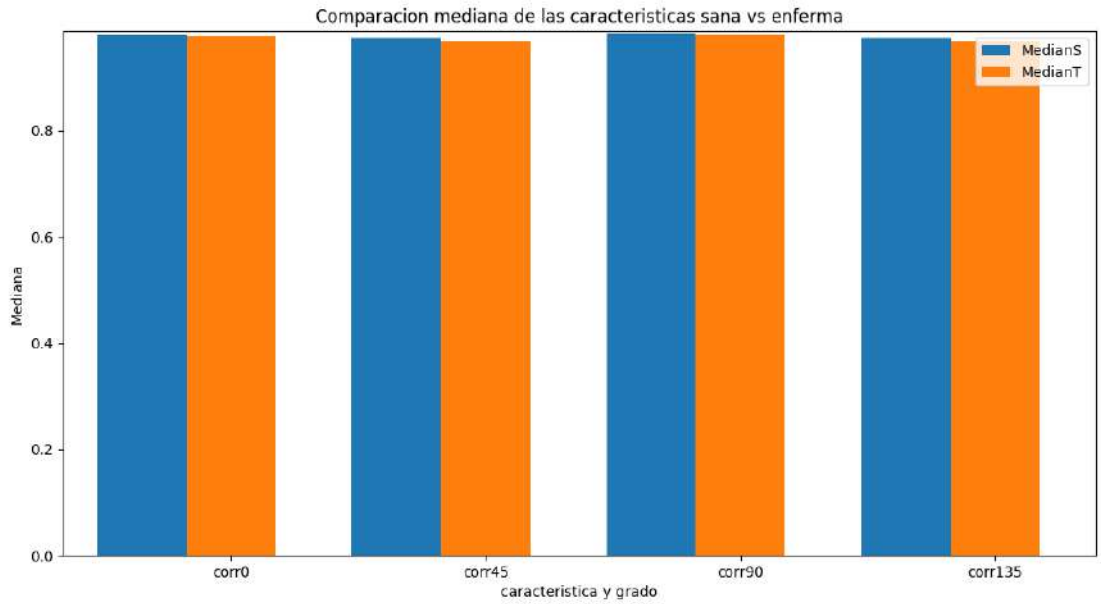

Figura 4.5: Gráfica de barras representante de la correlación para los 4 ángulos definidos.

La Figura 4.6 muestra los resultados para la característica de ASM.

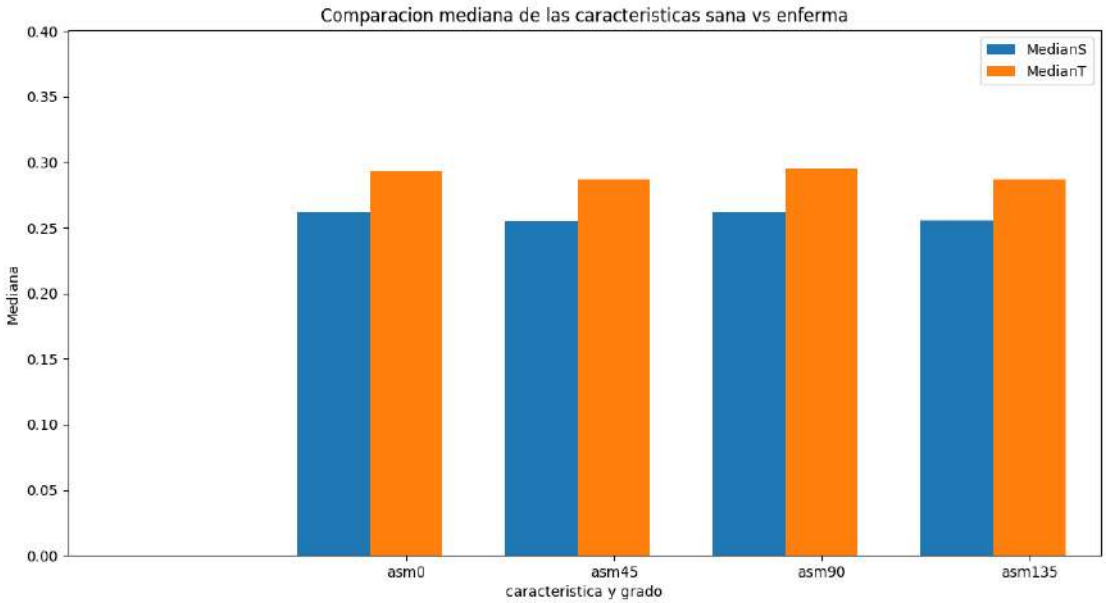

Figura 4.6: Gráfica de barras representante del ASM para los 4 ángulos definidos.

Una vez se hicieron estos análisis y se observó que entre los cuatro diferentes ángulos no había tal diferencia como el estado del arte lo predijo, la siguiente acción fue promediar estos cuatro valores para obtener valor único por característica para cada imagen. Entonces, a partir de aquí se hicieron las siguientes comparativas entre los valores obtenidos a partir de

la clase sana y la clase enferma de manera más global.

La Figura 4.7 muestra los resultados para la característica de contraste.

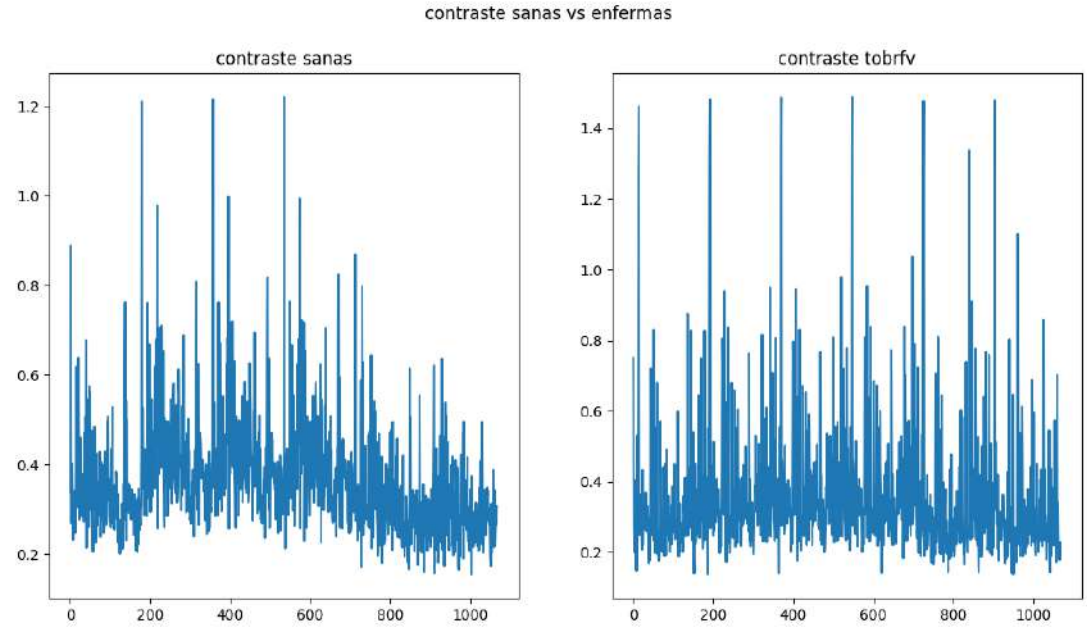

Figura 4.7: Gráfica representante del contraste sanas contra enfermas.

La Figura 4.8 muestra los resultados para la característica de disimilaridad.

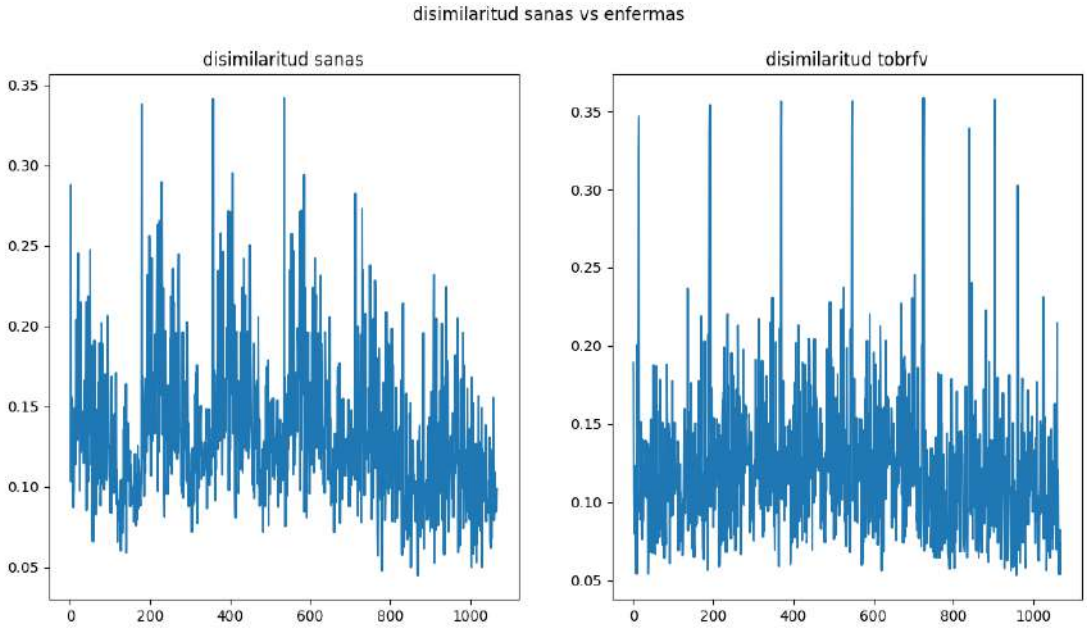

Figura 4.8: Gráfica representante de la disimilaridad sanas contra enfermas.

La Figura 4.9 muestra los resultados para la característica de homoge-

neidad.

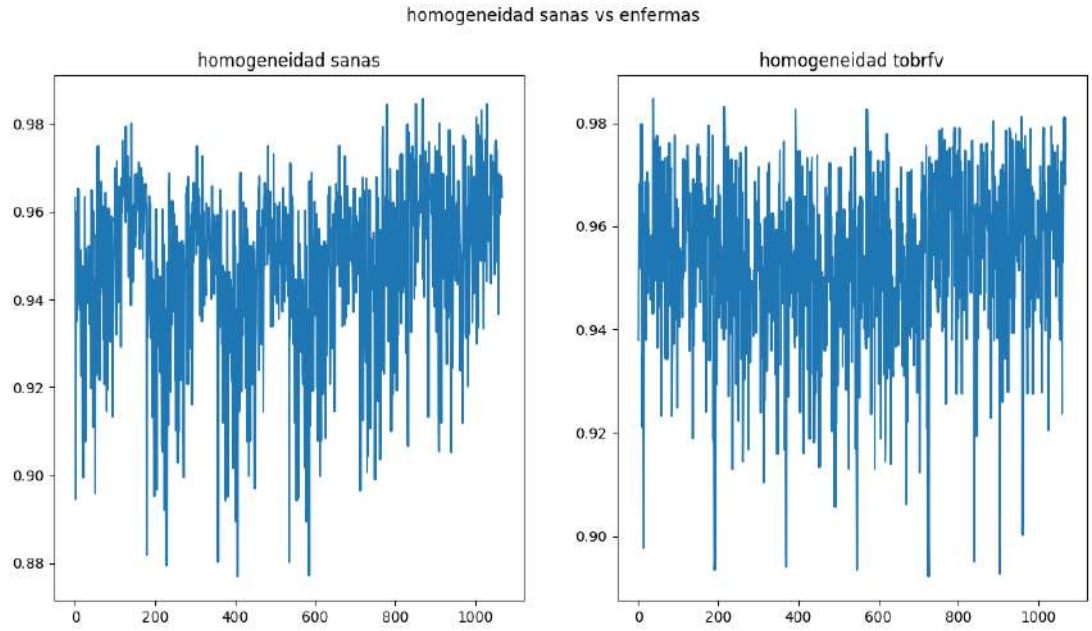

Figura 4.9: Gráfica representante de la homogeneidad sanas contra enfermas.

La Figura 4.10 muestra los resultados para la característica de energía.

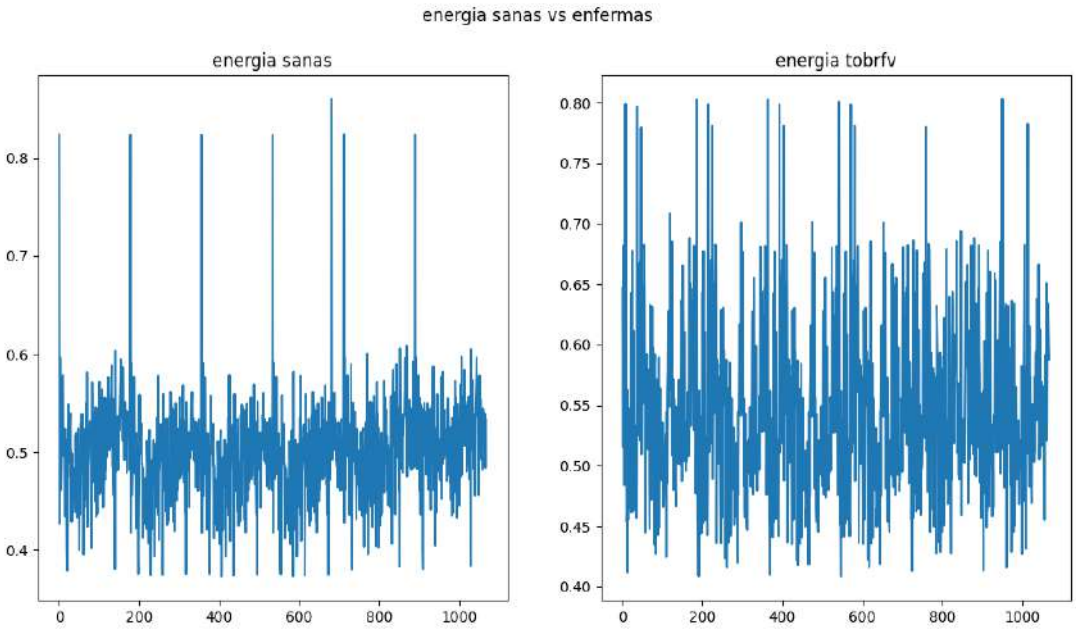

Figura 4.10: Gráfica representante de la energía sanas contra enfermas.

La Figura 4.11 muestra los resultados para la característica de correlación.

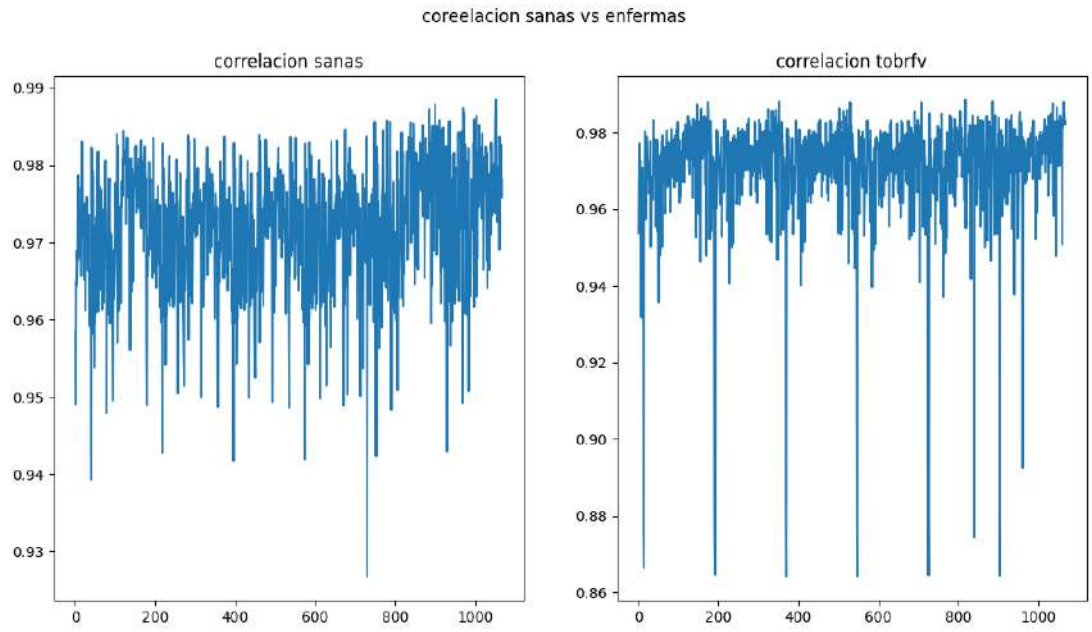

Figura 4.11: Gráfica representante de la correlación sanas contra enfermas.

La Figura 4.12 muestra los resultados para la característica de ASM.

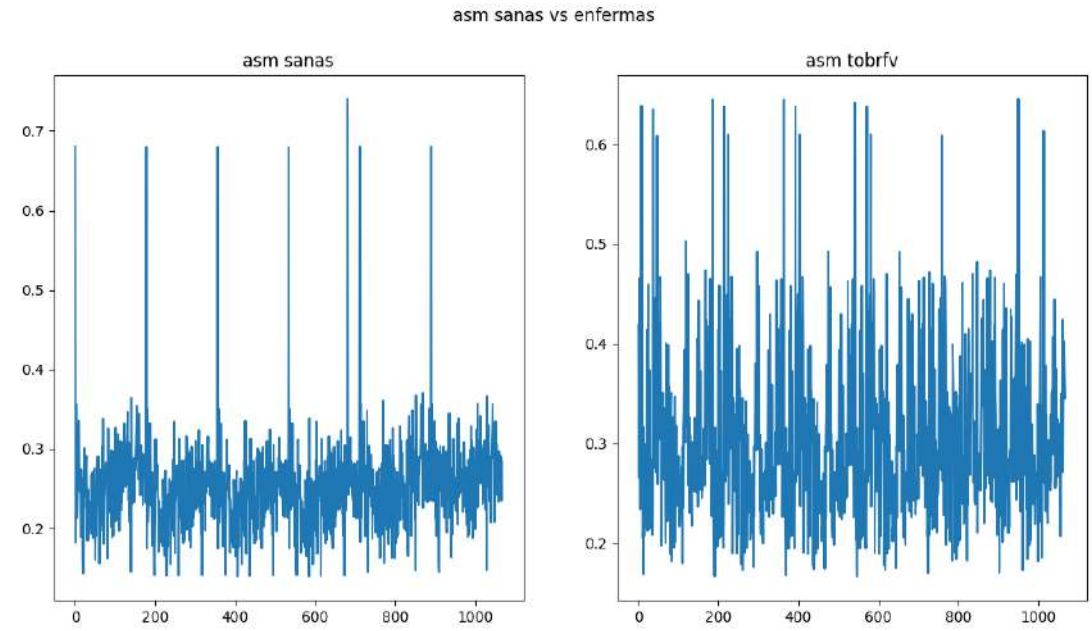

Figura 4.12: Gráfica representante del ASM sanas contra enfermas.

A partir de los resultados anteriores es evidente que la diferencia que existe entre las distribuciones del total de las imágenes sanas contra la distribución del total de las imágenes enfermas no es suficiente para hacer una clasificación exitosa de las clases. Por ello se decidió intentar hacer una mejora mediante el cambio del área de estudio de la imagen. En lugar de analizar toda la hoja, incluyendo el fondo, se analizó un área más centrada. Los siguientes resultados muestran las distribuciones de los datos obtenidos a partir de esta nueva extracción.

La Figura 4.13 muestra los resultados para la característica de contraste.

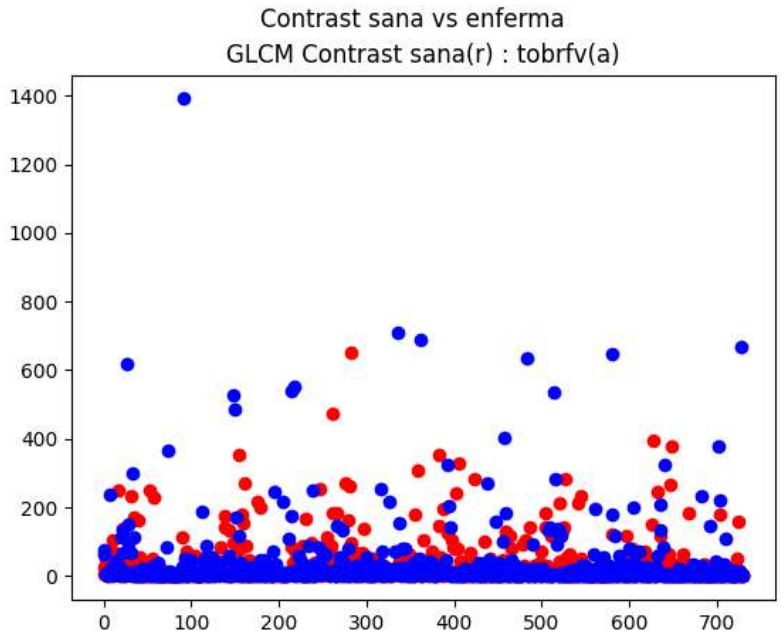

Figura 4.13: Gráfica de distribución representante del contraste sanas contra enfermas.

La Figura 4.14 muestra los resultados para la característica de disimilaridad.

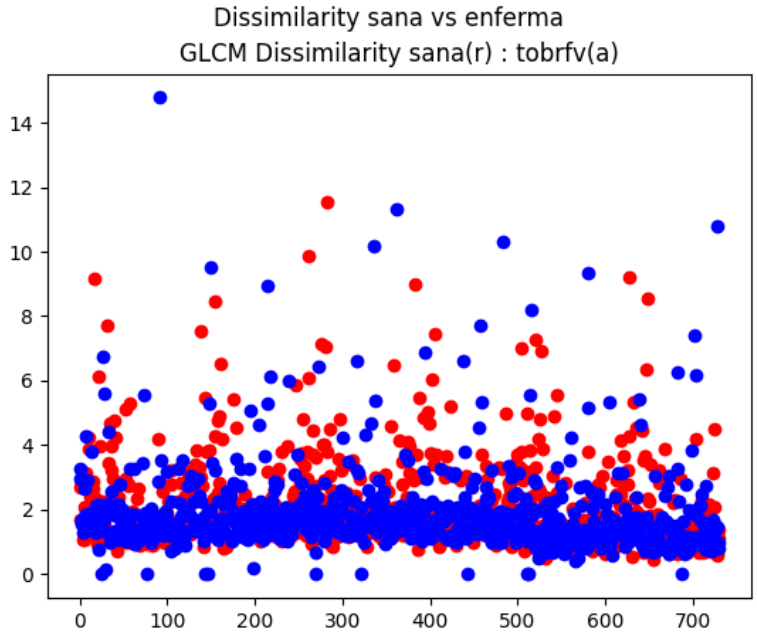

Figura 4.14: Gráfica de distribución representante de la disimilaridad sanas contra enfermas.

La Figura 4.15 muestra los resultados para la característica de homogeneidad.

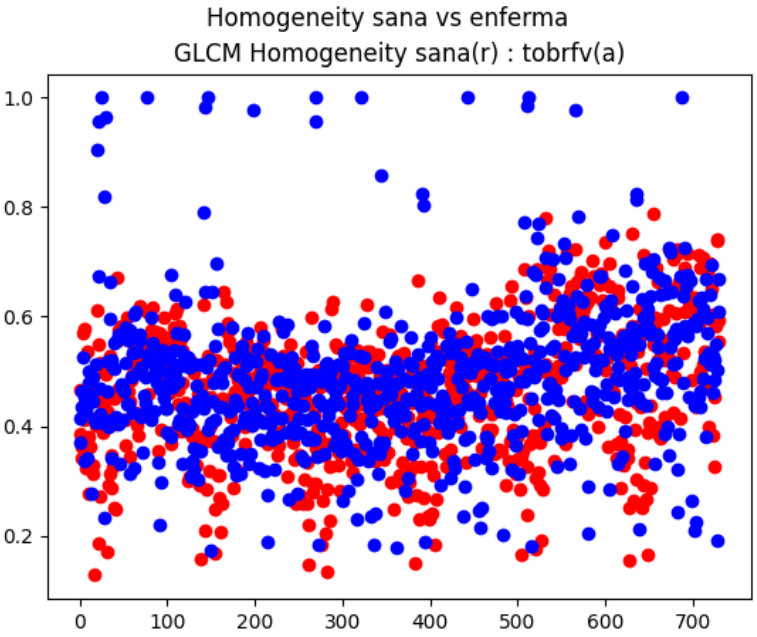

Figura 4.15: Gráfica de distribución representante de la homogeneidad sanas contra enfermas.

La Figura 4.16 muestra los resultados para la característica de energía.

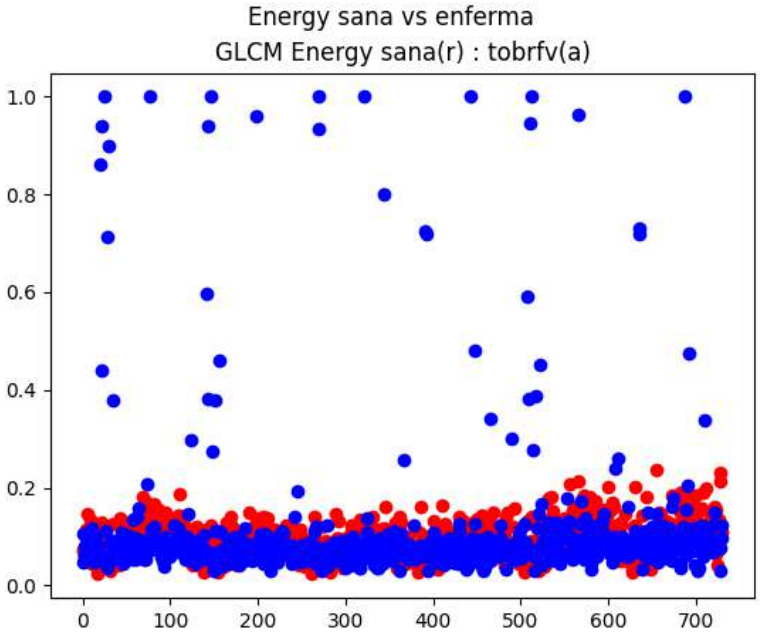

Figura 4.16: Gráfica de distribución representante de la energía sanas contra enfermas.

La Figura 4.17 muestra los resultados para la característica de correlación.

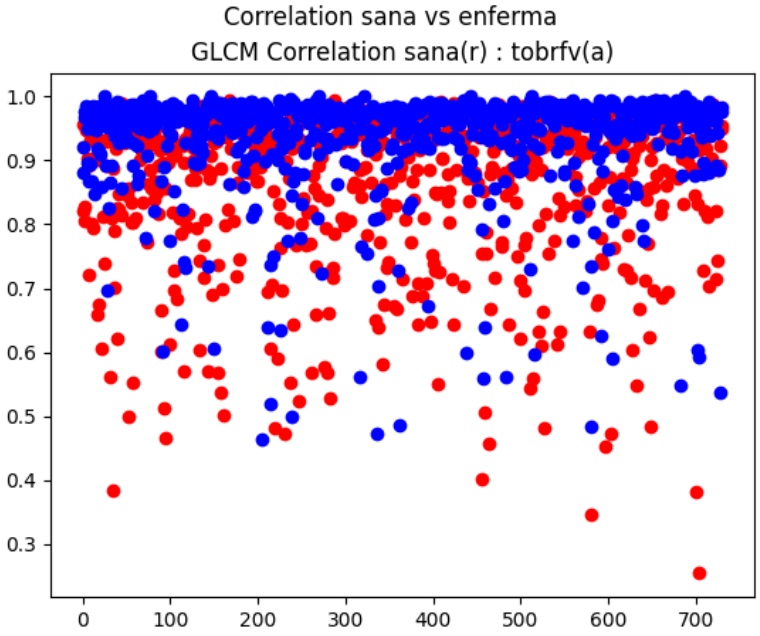

Figura 4.17: Gráfica de distribución representante de la correlación sanas contra enfermas.

La Figura 4.18 muestra los resultados para la característica de ASM.

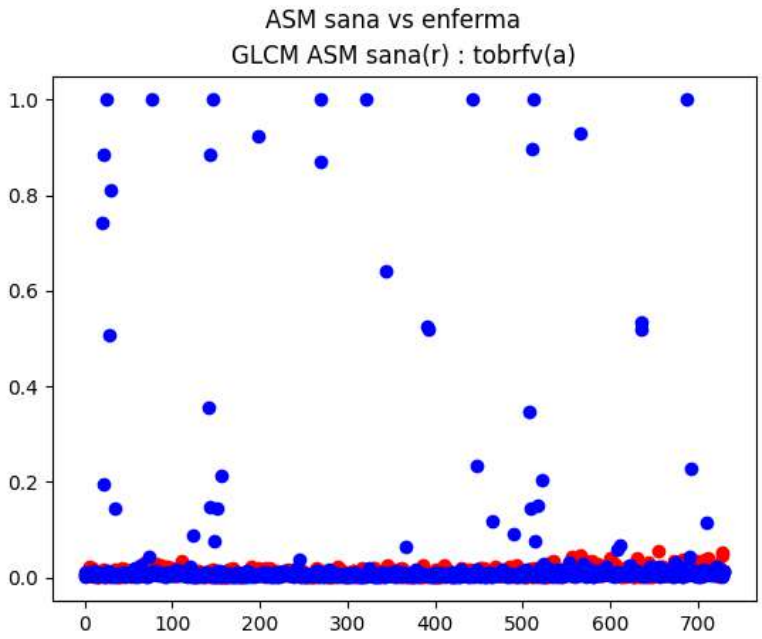

Figura 4.18: Gráfica de distribución representante del ASM sanas contra enfermas.

En estos nuevos resultados, al igual que en los primeros que se mostraron, es sencillo percatarse que existe una diferencia, pero de la misma manera esta diferencia tampoco es suficiente para una clasificación exitosa. Motivo por el cual se decidió hacer un cambio en la metodología de extracción de características y dejar atrás a la GLCM. Motivándonos a cambiar a una nueva metodología que produjo los resultados mostrados en la siguiente subsección.

4.1.2. Mediante histogramas de sumas y diferencias

La nueva metodología implementada para la extracción de características fueron los histogramas de sumas y diferencias. En teoría, esta metodología es considerablemente más sencilla en términos de cálculos en comparación con el uso de la GLCM. Sin embargo, se aplicaron de manera distinta. A pesar de que el método de histogramas de sumas y diferencias ofrece cálculos más rápidos y menos complejos, este proceso requería considerablemente más tiempo de procesamiento en comparación con el uso de la GLCM, ya que en este proceso sí se procesó la imagen pixel por pixel en su tamaño regular a diferencia de en el proceso que se simplificó una imagen con 256 variaciones en una imagen de solo 16 variaciones, haciendo las operaciones más rápidas. Los primeros resultados que se observan en

la Figura 4.19 son el total de histogramas obtenidos (122,uno por imagen) en cada una de las características que se extrajeron para plantas sanas y en la Figura 4.20 para plantas enfermas.

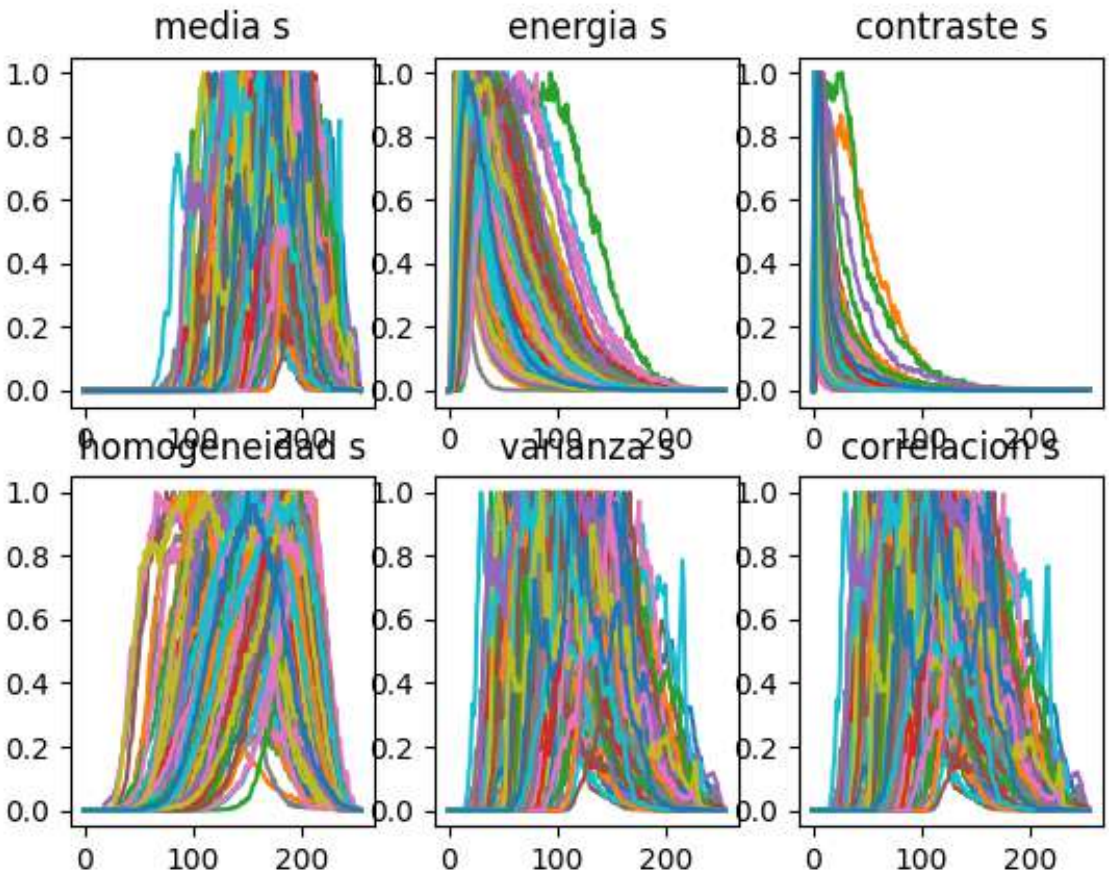

Figura 4.19: Distribución de las seis características de la clase sana.

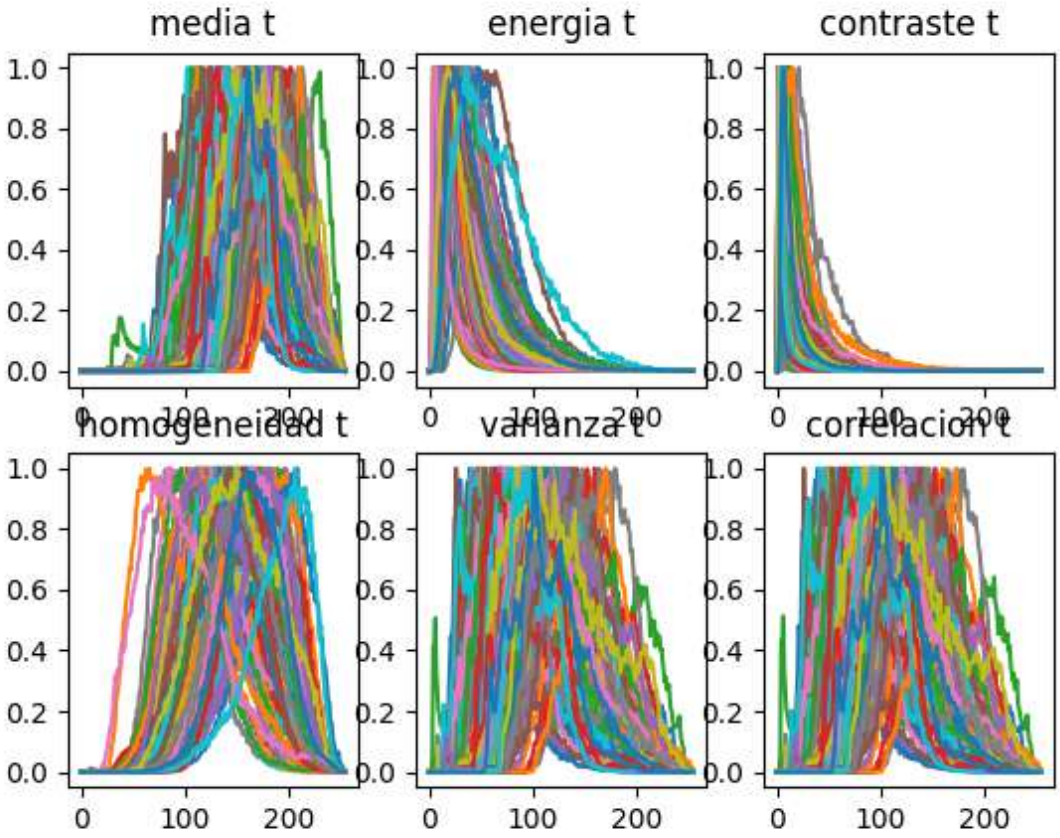

Figura 4.20: Distribución de las seis características de la clase enferma.

A modo de ver mejor comparación entre cada característica respecto a su clase, se configuraron las siguientes figuras:

La Figura 4.21 representa la comparación del contraste.

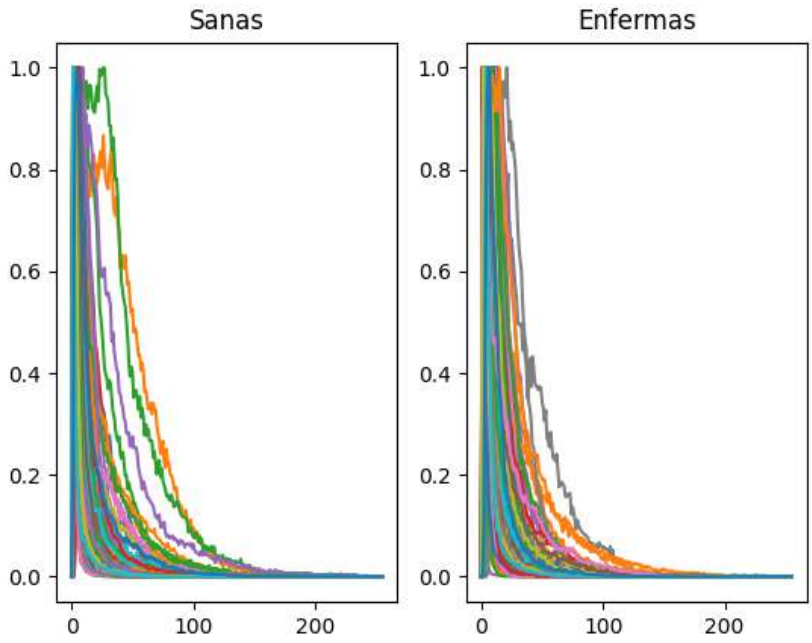

Figura 4.21: Contraste sanas contra enfermas de todas las imágenes por cada clase.

La Figura 4.22 representa la comparación de la correlación.

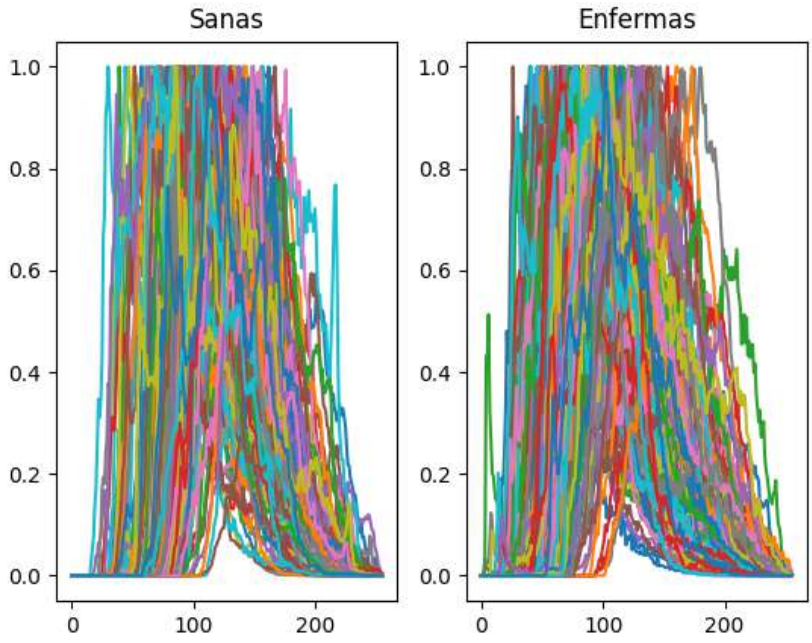

Figura 4.22: Correlación sanas contra enfermas de todas las imágenes por cada clase.

La Figura 4.23 representa la comparación de la energía.

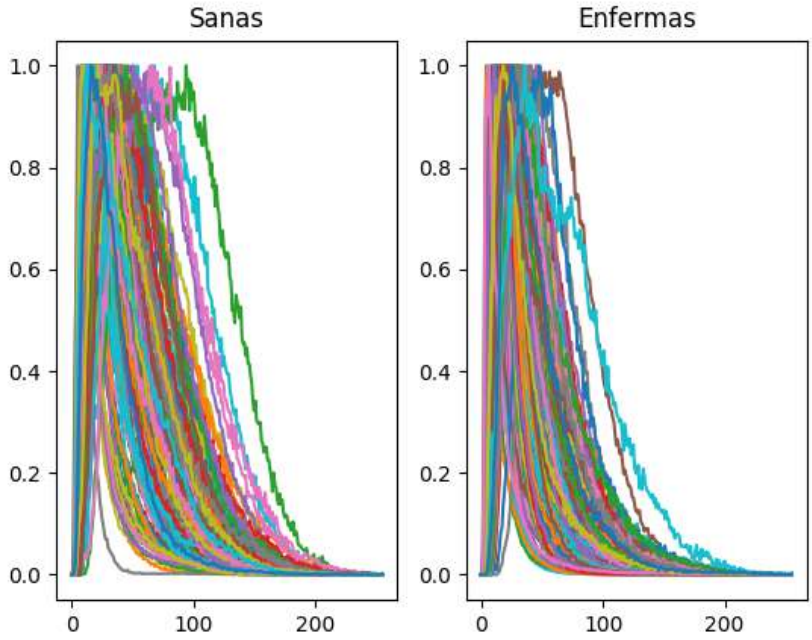

Figura 4.23: Energía sanas contra enfermas de todas las imágenes por cada clase.

La Figura 4.24 representa la comparación de la homogeneidad.

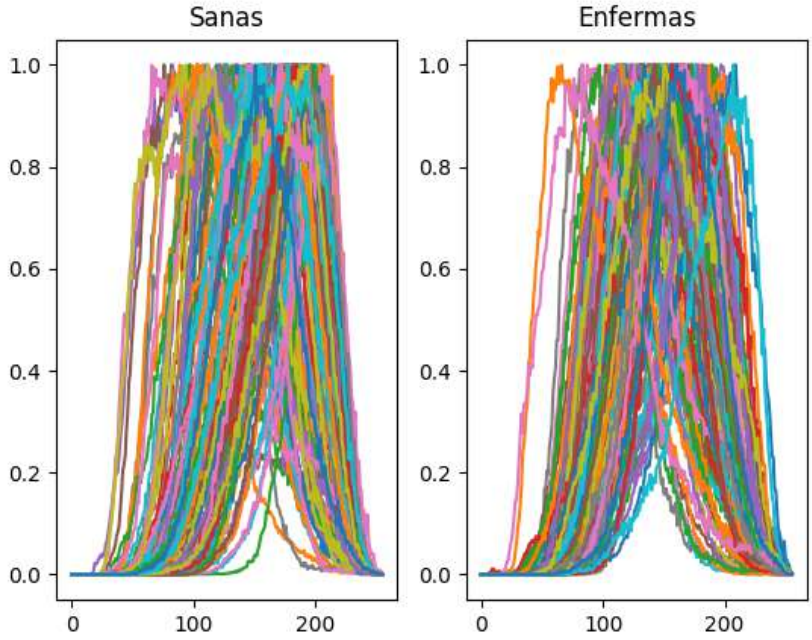

Figura 4.24: Homogeneidad sanas contra enfermas de todas las imágenes por cada clase.

La Figura 4.25 representa la comparación de la media.

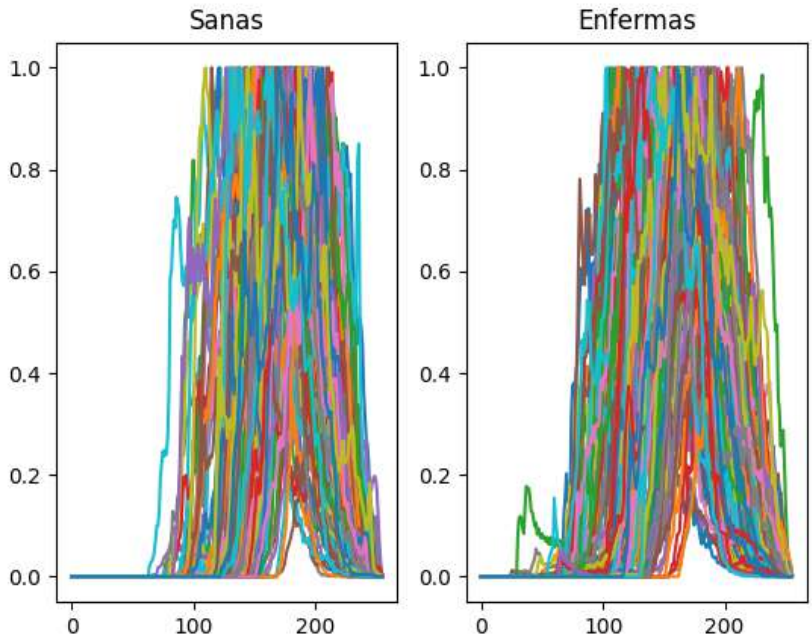

Figura 4.25: Media sanas contra enfermas de todas las imágenes por cada clase.

La Figura 4.26 representa la comparación de la varianza.

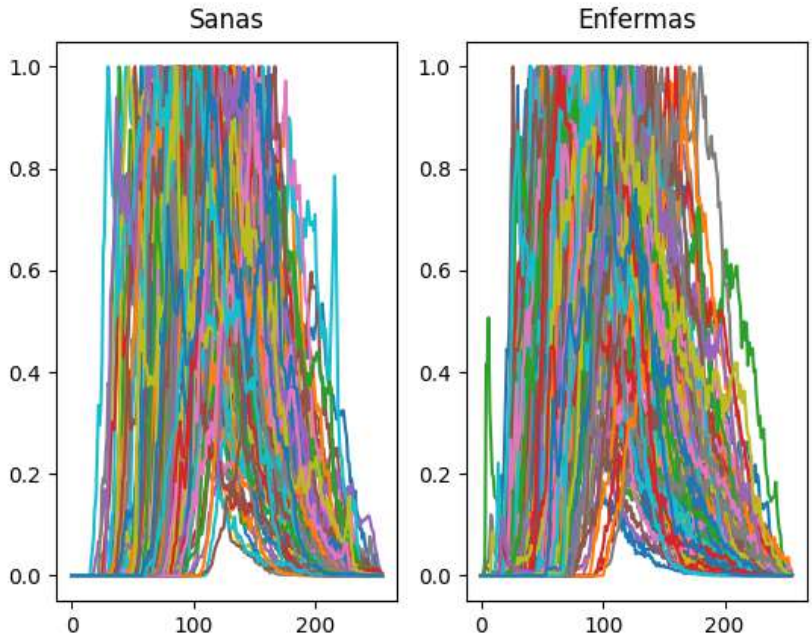

Figura 4.26: Varianza sanas contra enfermas de todas las imágenes por cada clase.

Para una vista más particular, en las siguientes figuras se observan los

resultados individuales de algunas imágenes. Cada figura muestra las seis características correspondientes a la imagen del respectivo número. Primero veremos ejemplos de las plantas sanas y luego de las plantas enfermas.

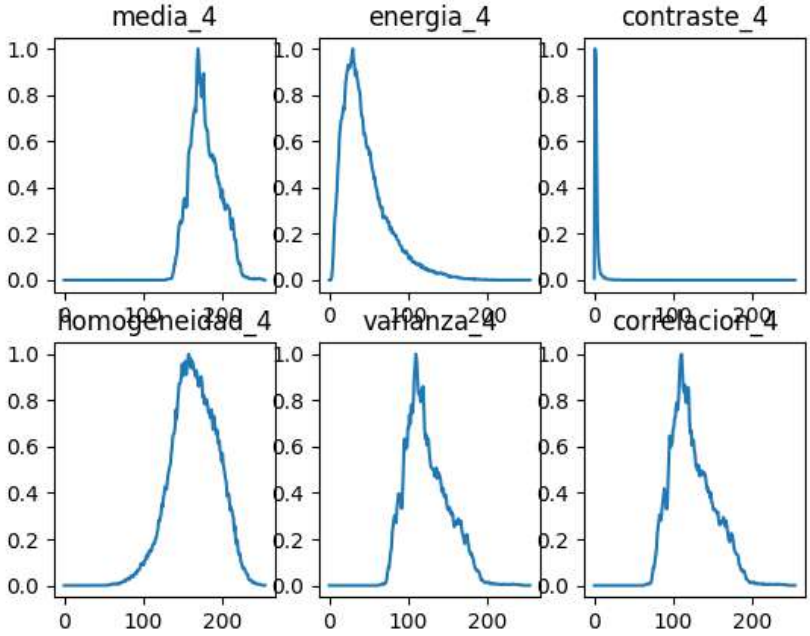

Figura 4.27: Ejemplo 1: Muestra de las seis características extraídas de una hoja enferma

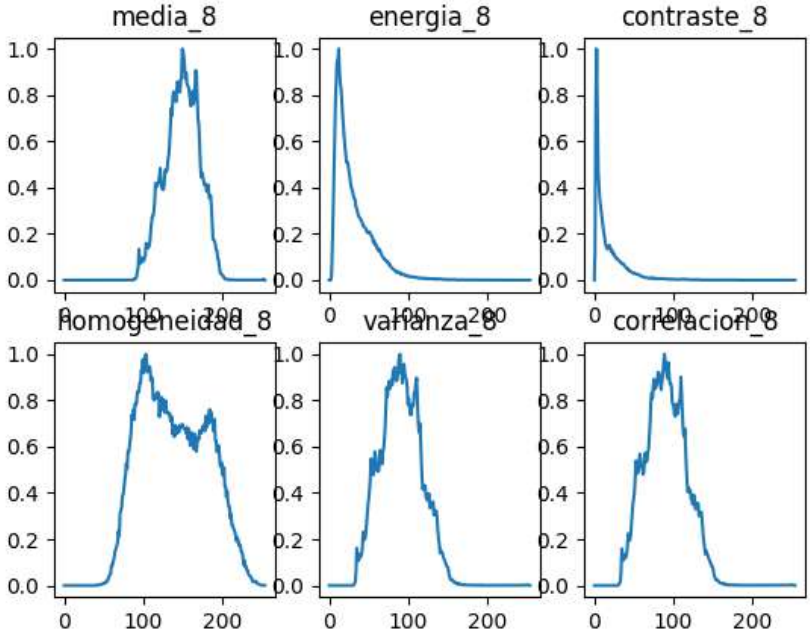

Figura 4.28: Ejemplo 2: Muestra de las seis características extraídas de otra hoja sana

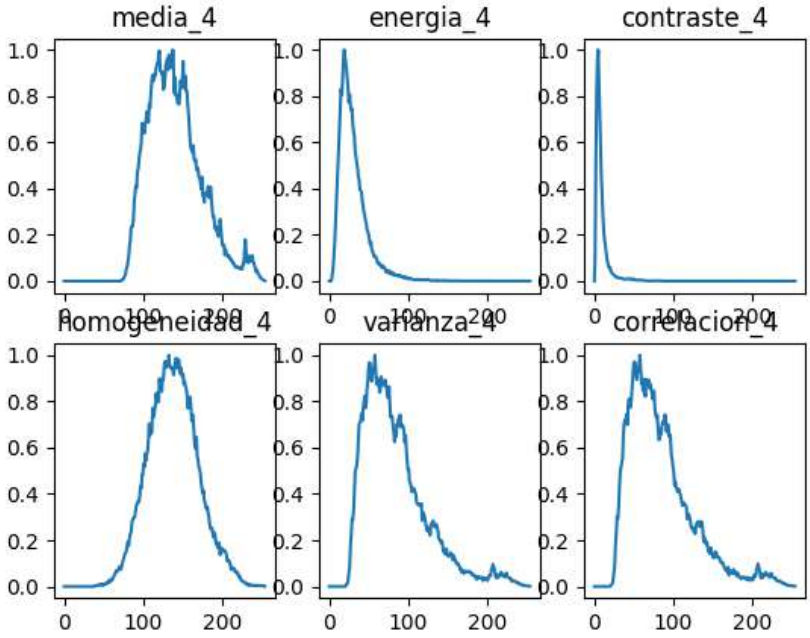

Figura 4.29: Ejemplo 1: Muestra de las seis características extraídas de una hoja enferma

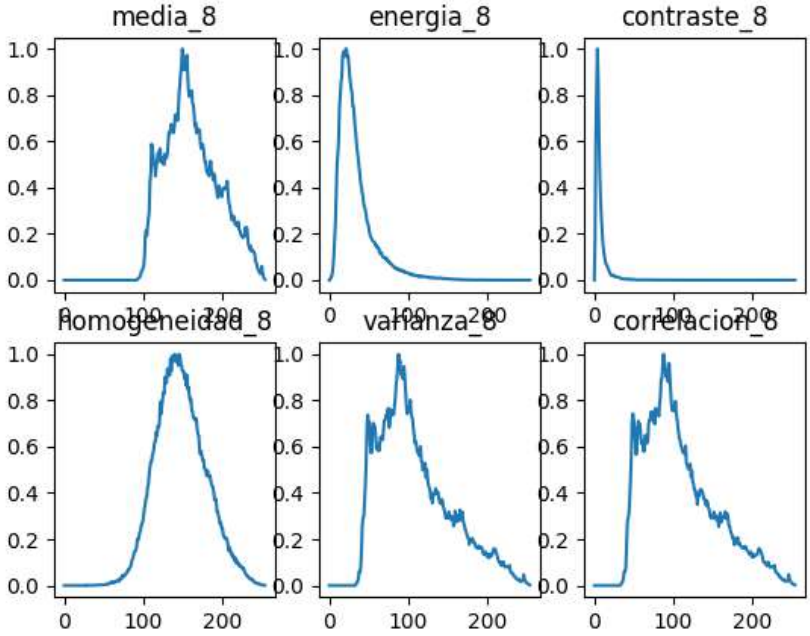

Figura 4.30: Ejemplo 2: Muestra de las seis características extraídas de otra hoja enferma

En el caso del método de histogramas de sumas y diferencias con la doble segmentación se evaluó únicamente la totalidad de la hoja mediante pequeñas ventanas. Los resultados obtenidos de estas evaluaciones, es decir, los anteriormente mostrados, no fueron los deseados, por lo cual se realizó el último experimento.

Este último experimento consistía en cambiar los datos de entrada por completo. Se sustituyeron las imágenes de hoja completa con fondo negro por imágenes de recortes con síntomas visibles, así como secciones de las hojas sanas que no muestran ningún tipo de síntoma. Los resultados fueron mejores, lo que nos indica que los síntomas se perdían en la inmensidad de la hoja y el fondo vacío. Por ende se prosiguió a la siguiente sección de la metodología que consiste en la clasificación de las dos clases.

4.1.3. Resultados de los 5 clasificadores para validación cruzada

Los resultados esperados de la clasificación de enfermedades en plantas de acuerdo al estado del arte son aquellos que están entre el 85 % y el 90 % de efectividad. Con esto como entendido, se comenzó la búsqueda del clasificador más adecuado para nuestro caso de estudio que es el ToBRFV. Recodemos, sin embargo, que no podemos contrastar estos resultados con otros trabajos de investigación que tengan el mismo caso de estudio con el objetivo de comprar los valores obtenidos a partir de las distintas metodologías, ya que estos estudios aún no se han realizado, y en el caso de haber sido realizados no se encuentran documentados.

Con el objetivo de hacer el mejor análisis posible se probaron cinco distintos clasificadores que son: Redes neuronales, máquinas de soporte vectorial. árboles de decisión, k-vecinos más cercanos y bosque aleatorio. Las métricas más importantes para evaluar él desempeñó de una clasificación son: Exactitud, precisión, exhaustividad y Valor F, mejor conocidas por su representación en Inglés como acccuracy, precision, recall y f1-score. A continuación se presentan los resultados de estos cuatro valores para cada uno de los cinco clasificadores seleccionados.

Tabla 4.1: Resultados del método "Redes neuronales" con validación cruzada.

	Accuracy	**Precision**	**Recall**	**F1 Score**
1	0.8333	0.85	0.8095	0.8292
2	0.8809	0.8636	0.9047	0.8837
3	0.8571	0.8947	0.8095	0.85
4	0.8809	0.9	0.8571	0.8780
5	0.8095	0.8095	0.8095	0.8095
6	0.7619	0.7037	0.9047	0.7916
7	0.8536	0.9375	0.75	0.8333
8	0.9024	0.9444	0.85	0.8947
9	0.9268	0.9090	0.9523	0.9023
10	0.7804	0.75	0.8571	0.8
AVG	***0.8487***	***0.8562***	***0.8504***	***0.8500***

Tabla 4.2: Resultados del método "Máquinas de soporte vectorial" con validación cruzada.

	Accuracy	**Precision**	**Recall**	**F1 Score**
1	0.8095	0.8095	0.8095	0.8095
2	0.9285	0.95	0.9047	0.9268
3	0.7619	0.7391	0.8095	0.7727
4	0.8333	0.8181	0.8571	0.8372
5	0.8809	0.9	0.8571	0.8780
6	0.8095	0.7826	0.8571	0.8181
7	0.8536	0.7916	0.95	0.8636
8	0.8048	0.7727	0.85	0.8095
9	0.9024	0.84	1	0.9130
10	0.8536	0.8260	0.9047	0.8636
AVG	***0.8438***	***0.8229***	***0.88***	***0.8492***

Tabla 4.3: Resultados del método "Árboles de decisión" con validación cruzada.

	Accuracy	**Precision**	**Recall**	**F1 Score**
1	0.8333	0.7692	0.9523	0.8510
2	0.6904	0.75	0.5714	0.6486
3	0.7857	0.8333	0.7142	0.7692
4	0.7857	0.7727	0.8095	0.7906
5	0.6666	0.6666	0.6666	0.6666
6	0.7857	0.7227	0.8095	0.7906
7	0.7804	0.8235	0.7	0.7567
8	0.8048	08	0.8	0.8
9	0.8292	0.85	0.8095	0.8292
10	0.7317	0.7777	0.6666	0.7179
AVG	***0.7693***	***0.7815***	***0.75***	***0.7620***

Tabla 4.4: Resultados del método "K-Vecinos más cercanos" con validación cruzada.

	Accuracy	**Precision**	**Recall**	**F1 Score**
1	0.8095	0.7241	1	0.84
2	0.7619	0.72	0.8571	0.7826
3	0.7380	0.6923	0.8571	0.7659
4	0.8333	0.8181	0.8571	0.8372
5	0.8095	0.7407	0.9523	0.8333
6	0.8333	0.7692	0.9523	0.8510
7	0.8292	0.8095	0.85	0.8292
8	0.7317	0.9656	0.8	0.7441
9	0.6097	0.5862	0.8095	0.68
10	0.7560	0.72	0.8571	0.7826
AVG	***0.0.7712***	***0.0.7275***	***0.8792***	***0.7946***

Tabla 4.5: Resultados del método "Bosque aleatorio" con validación cruzada.

	Accuracy	**Precision**	**Recall**	**F1 Score**
1	0.7142	0.6666	0.8571	0.75
2	0.8095	0.76	0.9047	0.8260
3	0.8333	0.8181	0.8571	0.8372
4	0.8571	0.8947	0.8095	0.85
5	0.8095	0.7826	0.8571	0.8181
6	0.7380	0.6785	0.9047	0.7755
7	0.9024	0.8333	1	0.9090
8	0.8536	0.85	0.85	0.85
9	0.8048	0.8421	0.7619	0.8
10	0.8536	0.8947	0.8095	0.85
AVG	***0.8176***	***0.8020***	***0.8611***	***0.8266***

Capítulo 5

Conclusiones

Podemos concluir que los dos mejores clasificadores para la detección del ToBRFV en hojas de tomate fueron sin duda las redes neuronales y las máquinas de soporte vectorial. Obteniendo un 84 % de exactitud, así como 85 % y 82 % de precisión respectivamente.

Esto confirma nuestra hipótesis de que es posible clasificar de manera certera la existencia del virus rugoso del tomate en una fotografía de una hoja de tipo RGB mediante la metodología planteada en este trabajo de investigación.

Debe destacarse este resultado teniendo en cuenta que es la primera vez que se intenta y logra clasificar el ToBRFV mediante imágenes RGB de hojas de tomate. Además, deja como resultado adicional una base de datos de hojas infectadas con ToBRFV anteriormente inexistente que es de gran utilidad para futuras investigaciones.

Como trabajo a futuro se tiene en cuenta que existen incontables posibilidades de mejora. Como en todo proceso siempre hay espacio para mejorar. En el caso de este trabajo de estudio en particular, es posible mencionar que a pesar de que la base de datos creada es una gran aportación, se puede formar una aún mejor. Además, se pueden realizar una gran cantidad de distintos preprocesamientos que van desde la transformación de color hasta la aplicación de los anteriormente mencionados índices de vegetación.

Bibliografía

Pattern Recognition. 2011. ISBN 9780387310732.

A. M. Abdu, M. M. Mokji, and U. U. Sheikh. An Automatic Plant Disease Symptom Segmentation Concept Based on Pathological Analogy. *ICSGRC 2019 - 2019 IEEE 10th Control and System Graduate Research Colloquium, Proceeding*, (August):94–99, 2019. doi: 10.1109/ICSGRC.2019.8837076.

N. Aguirre Dobernack. Procesamiento de imágenes. s.f.

A. S. V. V. S. Arivazhagan S., Newlin Shebiah R. Detection of unhealthy region of plant leaves and classification of plant leaf diseases using texture features. 2013.

J. G. Arnal Barbedo. Barbedo - 2019 - Detection of nutrition deficiencies in plants using proximal images and machine learning A review es.pdf, 2019.

Z. Arthur, H. E. H. Figueroa, and J. A. Fracarolli. Computer vision based detection of external defects on tomatoes using deep learning. *Biosystems Engineering*, 190:131–144, 2019. ISSN 1537-5110. doi: 10.1016/j.biosystemseng.2019.12.003. URL `https://doi.org/10.1016/j.biosystemseng.2019.12.003`.

K. Asefpour Vakilian and J. Massah. A farmer-assistant robot for nitrogen fertilizing management of greenhouse crops. *Computers and Electronics in Agriculture*, 139:153–163, 2017. ISSN 01681699. doi: 10.1016/j.compag.2017.05.012. URL `http://dx.doi.org/10.1016/j.compag.2017.05.012`.

X. Bai, X. Li, Z. Fu, X. Lv, and L. Zhang. Original papers A fuzzy clustering segmentation method based on neighborhood grayscale information for defining cucumber leaf spot disease images. *Computers and Electronics in Agriculture*, 136, 2017. doi: 10.1016/j.compag.2017.03.004. URL `http://dx.doi.org/10.1016/j.compag.2017.03.004`.

S. Bianco, F. Gasparini, and R. Schettini. *Color Coding for Data Visualization*, pages 85–95. 01 2014. doi: 10.4018/978-1-4666-5888-2.ch161.

L. Breiman. Random forests. *Mach. Learn., 45(1):5–32*, 2017.

Caades. Medidas de Prevención para nuevo Virus de Tomate (ToBRFV) ESTIMADOS, BOLETÍN URGENTE. Technical Report 56, 2015.

J. V.-L. J. B. A.-R. S. G.- C. D. J. L.-B. J. A. O.-M. D. L. Cambrón-Crisantos, José Manuel Rodríguez-Mendoza. Primer reporte de Tomato brown rugose fruit virus (ToBRFV) en Michoacán, México. *Revista Mexicana de Fitopatología, Mexican Journal of Phytopathology*, 2018. ISSN 0185-3309.

J. Chen, B. Guan, H. Wang, X. Zhang, Y. Tang, and W. Hu. Image Thresholding Segmentation Based on Two Dimensional Histogram Using Gray Level and Local Entropy Information. *IEEE Access*, 6(c):5269–5275, 2017. ISSN 21693536. doi: 10.1109/ACCESS.2017.2757528.

S. Davino, A. G. Caruso, S. Bertacca, S. Barone, and S. Panno. Tomato Brown Rugose Fruit Virus : Seed Transmission. *MDPI*, pages 1–13, 2020.

Euroseeds. Commitment of the European Seed Industry to help manage risk from ToBRFV. Technical report, 2020.

J. Frederic, I. Nturambirwe, and U. L. Opara. CienciaDirecta Aplicaciones de aprendizaje automático para la detección no destructiva de defectos en productos hortícolas. 9, 2019. doi: 10.1016/j.biosystemseng.2019.11.011.

Z. Gao, Z. Luo, W. Zhang, Z. Lv, and Y. Xu. Deep Learning Application in Plant Stress Imaging: A Review. *AgriEngineering*, 2, 2020. doi: 10.3390/agriengineering2030029.

S. S. Gavankar and S. D. Sawarkar. Eager decision tree. *2017 2nd International Conference for Convergence in Technology*, 2017.

I. M. Hanssen, R. Mumford, D. R. Blystad, I. Cortez, B. Hasiów-Jaroszewska, D. Hristova, I. Pagán, A. M. Pereira, J. Peters, H. Pospieszny, M. Ravnikar, I. Stijger, L. Tomassoli, C. Varveri, R. van der Vlugt, and S. L. Nielsen. Seed transmission of Pepino mosaic virus in tomato. *European Journal of Plant Pathology*, 126, 2010. ISSN 09291873. doi: 10.1007/s10658-009-9528-x.

M. Islam, A. Dinh, and K. Wahid. Detection of potato diseases using image segmentation and multiclass support vector machine. 2017.

M. R. G. J. M. Keller and J. A. Givens. A fuzzy k-nearest neighbor algorithm. *IEEE Transactions on Systems, Man, and Cybernetics*, 1985.

L. H. S. K. A.-R. J. M. G. K. . W.-S. J. Koh, S. H. Koh, s. h., li, h., sivasithamparam, k., admiraal, r., jones, m. g. k., wylie, s. j. 2018.

N. N. Kurniawati, S. Norul, H. Sheikh, S. Abdullah, and S. Abdullah. Investigation on Image Processing Techniques for Diagnosing Paddy Diseases. 2009. doi: 10.1109/SoCPaR.2009.62.

D. Li, C. Li, Y. Yao, M. Li, and L. Liu. Modern imaging techniques in plant nutrition analysis: A review. *Computers and Electronics in Agriculture*, 174(February 2019):105459, 2020. ISSN 01681699. doi: 10.1016/j.compag.2020.105459. URL `https://doi.org/10.1016/j.compag.2020.105459`.

G. S. M. V.-r. I. Macedo-cruz, Antonia Pajares. Digital Image Sensor-Based Assessment of the Status of Oat (Avena sativa L.) Crops after Frost Damage. 2011.

R. Mendez. Etapas de la visión artificial. In *Procesamiento digital de imágenes*, page 58. 2008.

B. Naik, Sapan Patel. Machine Vision based Fruit Classification and Grading - A Review. *International Journal of Computer Applications*, 170, 2017.

L. I. Nolasco-García, J. L. Marín-León, J. E. Ruiz-Nieto, and J. Hernández-Ruíz. Identification methods for tomato brown rugose fruit virus (tobrfv) in méxico. *Agronomy Mesoamerican*, 31(3):835–844, 2020. ISSN 22153608. doi: 10.15517/AM.V31I3.40655.

N. OTSU. A Tlreshold Selection Method from Gray-Level Histograms. *Czasopismo stomatologiczne*, 26, 1979. ISSN 00114553.

S. Phadikar and J. Sil. Rice Disease Identification using Pattern Recognition Techniques. (Iccit):25–27, 2008.

M. Presutti. La matriz de co-ocurrencia en la clasificación multiespectral: tutorial para la enseñanza de medidas texturales en cursos de grado universitario. *Jornada de Educação em Sensoriamento Remoto no Âmbito do Mercosul*, page 9, 2004.

T. U. Rehman, M. S. Mahmud, Y. K. Chang, J. Jin, and J. Shin. Current and future applications of statistical machine learning algorithms for agricultural machine vision systems. 2019. doi: 10.1016/j.compag.2018.12.006.

C. A. Reyes García. Análisis de Señales y Reconocimiento de Patrones. 2017.

Á. Roel and J. Terra. *Muestreo de suelos y factores limitantes del rendimiento*. 2013. ISBN 9290397411.

SADER. Consideraciones regulatorias en torno al virus rugoso del tomate. 06. 2018.

N. Sengar, M. K. Dutta, and C. M. Travieso. Computer vision based technique for identification and quantification of powdery mildew disease in cherry leaves. *Computing*, 100(11):1189–1201, 2018. ISSN 0010485X. doi: 10.1007/s00607-018-0638-1. URL `https://doi.org/10.1007/s00607-018-0638-1`.

U. Shruthi, V. Nagaveni, and B. K. Raghavendra. A Review on Machine Learning Classification Techniques for Plant Disease Detection. *2019 5th International Conference on Advanced Computing Communication Systems (ICACCS)*, pages 281–284, 2019.

S. Singh, C. Uday, P. Singh, and S. Jain. Applications of Computer Vision in Plant Pathology : A Survey. *Archives of Computational Methods in Engineering*, (0123456789), 2019. ISSN 1886-1784. doi: 10.1007/s11831-019-09324-0. URL `https://doi.org/10.1007/s11831-019-09324-0`.

V. Singh. IIP : DOI : Técnicas de Computación. 2016. doi: 10.1016/j.inpa.2016.10.005.

L. E. Sucar and G. Gomez. Visión Computacional. 2011.

M. Tuceryan and A. K. Jain. Handbook of Pattern Recognition and Computer Vision. pages 235–276, 1993.

M. Unser. Sum and difference histograms for texture classification. *IEEE*, 1986.

A. A. P. Vijayakumar. Banana disease diagnosis using computer vision and machine learning methods. *Journal of Ambient Intelligence and Humanized Computing*, (2016), 2020. ISSN 1868-5145. doi: 10.1007/s12652-020-02273-8. URL `https://doi.org/10.1007/s12652-020-02273-8`.

A. Virol, N. S. A. Mansour, and M. C. B. W. Falk. A new tobamovirus infecting tomato crops in jordan. *Archives of Virology*, 2015. ISSN 1432-8798. doi: 10.1007/s00705-015-2677-7.

P. Wan, A. Toudeshki, H. Tan, and R. Ehsani. A methodology for fresh tomato maturity detection using computer vision. *Computers and Electronics in Agriculture*, 146(February 2017):43–50, 2018. ISSN 0168-1699. URL `https://doi.org/10.1016/j.compag.2018.01.011`.

Y. Wanjun and S. Xiaoguang. Research on text categorization based on machine learning. *IEEE International Conference on Advanced Management Science*, 2010.

Q. Yao, Z. Guan, and Y. Zhou. Application of support vector machine for detecting rice diseases using shape and color texture features. 2009. doi: 10.1109/ICEC.2009.73.

A. Yee-Rendon, I. Torres-pacheco, A. S. Trujillo-lopez, K. P. Romero-bringas, and J. R. Millan-almaraz. Identification in Jalapeño Pepper (Capsicum annuum) Leaves Using CNNs-Based Model. 2021.

Anexos

Códigos

Aumento de datos

```
import numpy as np
import cv2
from skimage import data
from skimage.transform import rotate #, resize,
downscale_local_mean
from PIL import Image

''' Proceso de reescalado de las imagenes
for i in range(1,179):
    image = cv2.imread('t_sano'+str(i)+'.jpg')
    r = 600.0 / image.shape[1]
    dim = (600, int(image.shape[0] * r))
    imgr = cv2.resize(image, dim,
    interpolation=cv2.INTER_AREA)
    cv2.imwrite('t_sano'+str(i)+'.jpg',imgr)
'''
'''
#Proceso de rotacion de las imagenes
for i in range(1,123):
    image = cv2.imread('t_sano'+str(i)+'.jpg')
    image2 = cv2.imread('t_tobrfv'+str(i)+'.jpg')
    imgr = rotate(image, 90, resize=True)
    imgr2 = rotate(image2, 90, resize=True)
    imgr = cv2.normalize(imgr, None, 0, 255,
    cv2.NORM_MINMAX, cv2.CV_8U)
    imgr2 = cv2.normalize(imgr2, None, 0, 255,
    cv2.NORM_MINMAX, cv2.CV_8U)
    cv2.imwrite('t_sano'+str(i+366)+'.jpg',imgr)
```

```
        cv2.imwrite('t_tobrfv'+str(i+366)+'.jpg',imgr2)
        #cv2.imshow('rotada',imgr)
        #cv2.waitKey(0)
        #cv2.destroyAllWindows()
'''
'''
#Proceso espejo vertical de las imagenes
for i in range(1,123):
        image = Image.open('t_sano'+str(i)+'.jpg')
        imgf = image.transpose
        (method=Image.FLIP_TOP_BOTTOM)
        imgf.save('t_sano'+str(i+488)+'.jpg')

        image2 = Image.open('t_tobrfv'+str(i)+'.jpg')
        imgf2 = image2.transpose
        (method=Image.FLIP_TOP_BOTTOM)
        imgf2.save('t_tobrfv'+str(i+)+'.jpg')
'''
#'''
#Proceso espejo horizontal de las imagenes
for i in range(1,123):
        image = Image.open('t_sano'+str(i)+'.jpg')
        imgf = image.transpose
        (method=Image.FLIP_LEFT_RIGHT)
        imgf.save('t_sano'+str(i+610)+'.jpg')

        image2 = Image.open('t_tobrfv'+str(i)+'.jpg')
        imgf2 = image2.transpose
        (method=Image.FLIP_LEFT_RIGHT)
        imgf2.save('t_tobrfv'+str(i+610)+'.jpg')

#'''
```

Segmentación fase 1

```
import numpy as np
import cv2
for i in range (1,733):
        #tomate sano
        #image = np.array(cv2.imread
```

```
('t_tobrfv89.jpg',cv2.IMREAD_GRAYSCALE))
img = cv2.imread('t_sano'+str(i)+'.jpg')
ib,ig,ir=cv2.split(img)
#cv2.imshow('original',img)

gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
_, thresh = cv2.threshold(gray,0,1,
cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)

segr=cv2.multiply(thresh,ir)
segb=cv2.multiply(thresh,ib)
segg=cv2.multiply(thresh,ig)
segm=cv2.merge([segb,segg,segr])
#cv2.imshow('segmentada rgb',segm)
cv2.imwrite('t_sano_seg'+str(i)+'.jpg',segm)

#tomate con tobrfv
#image2 = np.array(cv2.imread
('t_tobrfv89.jpg',cv2.IMREAD_GRAYSCALE))
img2 = cv2.imread('t_tobrfv'+str(i)+'.jpg')
ib2,ig2,ir2=cv2.split(img2)
#cv2.imshow('original',img2)

gray2 = cv2.cvtColor
(img2,cv2.COLOR_BGR2GRAY)
_, thresh2 = cv2.threshold(gray2,0,1,
cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
#cv2.imshow('thresh',thresh2)

segr2=cv2.multiply(thresh2,ir2)
segb2=cv2.multiply(thresh2,ib2)
segg2=cv2.multiply(thresh2,ig2)
segm2=cv2.merge([segb2,segg2,segr2])
#cv2.imshow('segmentada rgb',segm2)
cv2.imwrite('t_tobrfv_seg'+str(i)
+'.jpg',segm2)
```

Segmentación fase 2

```
import numpy as np
```

```
import cv2
for r in range (1,733):
    img = cv2.imread('t_sano_seg'+str(r)+'.jpg')
    img2 = cv2.imread
    ('t_tobrfv_seg'+str(r)+'.jpg')
    #cv2.imshow('original',img)
    ib,ig,ir=cv2.split(img)
    ib2,ig2,ir2=cv2.split(img2)

    m,n=np.shape(ib)
    for i in range (0,m-1):
        for j in range (0, n-1):
            if ib[i][j]>100 :
                ib[i][j]=0
                ig[i][j]=0
                ir[i][j]=0

    p,k=np.shape(ib2)
    for s in range (0,p-1):
        for t in range (0, k-1):
            if ib2[s][t]>100 :
                ib2[s][t]=0
                ig2[s][t]=0
                ir2[s][t]=0

    out1=cv2.merge([ib,ig,ir])
    out2=cv2.merge([ib2,ig2,ir2])
    cv2.imwrite
    ('t_sano_seg'+str(r+732)+'.jpg',out1)
    cv2.imwrite
    ('t_tobrfv_seg'+str(r+732)+'.jpg',out2)
    #cv2.imshow('eliminacion azul',out1)
    #cv2.imshow('eliminacion azul tobrfv',out2)
    #cv2.waitKey(0)
    #cv2.destroyAllWindows()
```

Extracción de características mediante GLCM

```
import numpy as np
import cv2
```

```
import pandas as pd
from skimage.feature
import greycomatrix, greycoprops
from skimage import io, color, img_as_ubyte

#columnas=['asm0','asm45','asm90','asm135','corr0',
'corr45','corr90','corr135','ener0','ener45',
'ener90','ener135','hmg0','hmg45','hmg90',
'hmg135','diss0','diss45','diss90','dissm135',
'cont0','cont45','cont90','cont135']
col_mean=['asm','correlacion','energia',
'homogeneidad','disimilaritud','contraste']
#df = pd.DataFrame(columns=columnas)
dfm=pd.DataFrame(columns=col_mean)
for i in range (1,1069): #
        img = cv2.imread('t_sano_seg'+str(i)+'.jpg')

        gray = color.rgb2gray(img)
        image = img_as_ubyte(gray)

        bins = np.array([0, 16, 32, 48, 64,
        80, 96, 112, 128, 144, 160, 176, 192,
        208, 224, 240, 255]) #16-bit
        inds = np.digitize(image, bins)

        max_value = inds.max()+1
        matrix_coocurrence = greycomatrix
        (inds, [1], [0, np.pi/4, np.pi/2, 3*np.pi/4],
        levels=max_value, normed=True, symmetric=True)

        # GLCM properties
        def contrast_feature(matrix_coocurrence):
                contrast = greycoprops
                (matrix_coocurrence, 'contrast')
                return contrast

        def dissimilarity_feature(matrix_coocurrence):
                dissimilarity =
                greycoprops(matrix_coocurrence,
                'dissimilarity')
```

```
        return dissimilarity

def homogeneity_feature(matrix_coocurrence):
        homogeneity = greycoprops
        (matrix_coocurrence, 'homogeneity')
        return homogeneity

def energy_feature(matrix_coocurrence):
        energy = greycoprops(matrix_coocurrence,
        'energy')
        return energy

def correlation_feature(matrix_coocurrence):
        correlation = greycoprops
        (matrix_coocurrence, 'correlation')
        return correlation

def asm_feature(matrix_coocurrence):
        asm = greycoprops(matrix_coocurrence, 'ASM')
        return asm

contr=contrast_feature(matrix_coocurrence)
diss=dissimilarity_feature(matrix_coocurrence)
hmg=homogeneity_feature(matrix_coocurrence)
ener=energy_feature(matrix_coocurrence)
corr=correlation_feature(matrix_coocurrence)
asm=asm_feature(matrix_coocurrence)
#vecar=np.concatenate
((asm,corr,ener,hmg,diss,contr), axis=1)

contrm=np.mean(contr)
dissm=np.mean(diss)
hmgm=np.mean(hmg)
enerm=np.mean(ener)
corrm=np.mean(corr)
asmm=np.mean(asm)
vecarm=[(asmm,corrm,enerm,hmgm,dissm,contrm)]
print(vecarm)

#print(len(vecar))
```

```
            #guardado de las variables
            #dfn = pd.DataFrame(vecar,columns=columnas)
            dfnm=pd.DataFrame(vecarm,columns=col_mean)
            print(dfnm)
            #df=df.append(dfn, ignore_index=False)
            dfm=dfm.append(dfnm,ignore_index=True)
            print(dfm)

#df.to_csv('resultados_tobrfv.csv')
dfm.to_csv('res_sano_mean.csv')
```

Centrado del área de ectracción

```
import matplotlib.pyplot as plt
import cv2
from skimage.feature
import greycomatrix, greycoprops
from skimage import data

for i in range (733,734): #1464+1
    images = cv2.imread
    ('t_sano_seg'+str(i)+'.jpg')
    #imaget = cv2.imread
    ('t_tobrfv_seg'+str(i)+'.jpg')
    val=0
    gray = cv2.cvtColor
    (images,cv2.COLOR_BGR2GRAY)
    _, thresh = cv2.threshold(gray,0,255,
    cv2.THRESH_BINARY+cv2.THRESH_OTSU)
    cv2.imshow('mascara',thresh)
    for i in range(gray.shape[0]):
        for j in range (gray.shape[1]):
            if thresh[i][j]==255:
                topx=i
                topy=j
                val=1
            if(val==1):
                break
        if(val==1):
            break
```

```
print(topx)
print(topy)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Observación de los resultados de la extracción mediante GLCM

```
import matplotlib.pyplot as plt
import numpy as np
import cv2
import pandas as pd
import csv

#col_tobrfv=['asm0', 'asm45', 'asm90', 'asm135',
'corr0','corr45','corr90','corr135','ener0',
'ener45','ener90','ener135','hmg0','hmg45',
'hmg90','hmg135','diss0','diss45','diss90',
'dissm135','cont0','cont45','cont90','cont135']
#columnas=['asm0', 'asm45', 'asm90', 'asm135',
'corr0','corr45','corr90','corr135','ener0',
'ener45','ener90','ener135','hmg0','hmg45',
'hmg90','hmg135','diss0','diss45','diss90',
'dissm135','cont0','cont45','cont90','cont135']
col_mean=['asm','correlacion','energia',
'homogeneidad','disimilaritud','contraste']

data_s=pd.read_csv('resultados_sano.csv')
data_t=pd.read_csv('resultados_tobrfv.csv')
data_sm=pd.read_csv('res_sano_mean.csv')
data_tm=pd.read_csv('res_tobrfv_mean.csv')

#data_s.plot("hindfoot_length", "weight", kind="scatter")
dfs=pd.DataFrame(data_s)
dft=pd.DataFrame(data_t)
dfsm=pd.DataFrame(data_sm)
dftm=pd.DataFrame(data_tm)

del(dfs['Unnamed: 0'])
del(dft['Unnamed: 0'])
```

```
del(dfsm['Unnamed: 0'])
del(dftm['Unnamed: 0'])
#histogramas
#dfsm.plot.hist()
#dftm.plot.hist()

#PLOTEO ASM
plt.subplot(1,2,1)
plt.plot(dfsm['asm'])
plt.title("asm sanas")
plt.subplot(1,2,2)
plt.plot(dftm['asm'])
plt.title("asm tobrfv")
plt.suptitle("asm sanas vs enfermas")
plt.show()

#PLOTEO CORRELACION
plt.subplot(1,2,1)
plt.plot(dfsm['correlacion'])
plt.title("correlacion sanas")
plt.subplot(1,2,2)
plt.plot(dftm['correlacion'])
plt.title("correlacion tobrfv")
plt.suptitle("coreelacion sanas vs enfermas")
plt.show()

#PLOTEO ENERGIA
plt.subplot(1,2,1)
plt.plot(dfsm['energia'])
plt.title("energia sanas")
plt.subplot(1,2,2)
plt.plot(dftm['energia'])
plt.title("energia tobrfv")
plt.suptitle("energia sanas vs enfermas")
plt.show()

#PLOTEO HOMOGENEIDAD
plt.subplot(1,2,1)
plt.plot(dfsm['homogeneidad'])
plt.title("homogeneidad sanas")
```

```
plt.subplot(1,2,2)
plt.plot(dftm['homogeneidad'])
plt.title("homogeneidad tobrfv")
plt.suptitle("homogeneidad sanas vs enfermas")
plt.show()

#PLOTEO DISIMILARITUD
plt.subplot(1,2,1)
plt.plot(dfsm['disimilaritud'])
plt.title("disimilaritud sanas")
plt.subplot(1,2,2)
plt.plot(dftm['disimilaritud'])
plt.title("disimilaritud tobrfv")
plt.suptitle("disimilaritud sanas vs enfermas")
plt.show()

#PLOTEO CONTRASTE
plt.subplot(1,2,1)
plt.plot(dfsm['contraste'])
plt.title("contraste sanas")
plt.subplot(1,2,2)
plt.plot(dftm['contraste'])
plt.title("contraste tobrfv")
plt.suptitle("contraste sanas vs enfermas")
plt.show()
'''
#print(dfs)
#print(dft)

dfs['estado']='sana'
dfs.to_csv('dfs.csv')
#print(dfs)
dft['estado']='enferma'
dft.to_csv('dft.csv')
#print(dft)

dfs_mean= dfs.mean(axis=0)
dfs_std= dfs.std(axis=0)
dfs_median=dfs.median(axis=0)
dft_mean= dft.mean(axis=0)
```

```
dft_std= dft.std(axis=0)
dft_median=dft.median(axis=0)

dfval=pd.DataFrame({'media_s':dfs_mean,'media_t'
:dft_mean,'std_s':dfs_std,'std_t':dft_std,
'mediana_s':dfs_median,'mediana_t':dft_median})
#print(dfval)

dfval.to_csv('comparacion_valores.csv')

numero_de_grupos = 24
indice_barras = np.arange(numero_de_grupos)
ancho_barras =0.35

plt.bar(indice_barras, dfs_median,
width=ancho_barras, label='MedianS')
plt.bar(indice_barras + ancho_barras,
dft_median, width=ancho_barras, label='MedianT')
plt.legend(loc='best')
plt.xticks(indice_barras + ancho_barras,
columnas)
plt.ylabel('Mediana')
plt.xlabel('caracteristica y grado')
plt.title('Comparacion mediana
de las caracteristicas sana vs enferma')
plt.savefig("grafica_barras.png")
#plt.show()

plt.scatter(indice_barras, dfs_median,
label='MedianS')
plt.scatter(indice_barras, dft_median,
label='MedianT')
plt.legend(loc='best')
plt.xticks(indice_barras, columnas)
plt.ylabel('Mediana')
plt.xlabel('caracteristica y grado')
plt.title('Comparacion mediana de
las caracteristicas sana vs enferma')
plt.savefig("grafica_scatter.png")
#plt.show()
```

```
,,,
```

Extracción de características mediante Histogramas de Sumas y Diferencias en toda la hoja

```
from pickletools import uint8
import cv2
from matplotlib.pyplot import hist2d, imshow
import numpy as np
import matplotlib.pyplot as plt
import time
import pandas as pd

def mask_hsyd(thresh,n):
    x,y=np.shape(thresh)
    thresh2=np.zeros((x,y,1),np.uint8)
    for i in range ((n//2)+1,x-1-(n//2)):
     for j in range ((n//2)+1,y-1-(n//2)):
        roi=thresh[i-(n//2):i+(n//2),
        j-(n//2):j+(n//2)]
              if(np.all(roi==1)):
                  thresh2[i][j]=1
    return thresh2

def mediahsyd(hs,m):
    s=0.0
    for i in range (0,m-1):
        s+=float(i*hs[i])
    s=s/2
    return s

def varianzahsyd(hs,hd,m,med):
    s=0; r=0
    for i in range (0,m-1):
        s+=float((i-(2*med))*
        (i-(2*med))*hs[i])
    for j in range(0,m-1):
        r+=float((j-255)*(j-255)*hd[j])
    return (s+r)/2
```

```
def correlacionhsyd(hs,hd,m,med):
    s=0; r=0
    for i in range (0,m-1):
        s+=float((i-(2*med))*
        (i-(2*med))*hs[i])
    for j in range(0,m-1):
        r+=float((j-255)*(j-255)*hd[j])
    return (s-r)/2

def energiahsyd(hs,hd,m):
    s=0; r=0
    hs=cv2.multiply(hs,hs)
    s=sum(hs)
    hd=cv2.multiply(hd,hd)
    r=sum(hd)
    return s*r

def contrastehsyd(hd,m):
    r=0
    for j in range(0,m-1):
        r+=(j-255)*(j-255)*hd[j]
    return r

def hmghsyd(hd,m):
    r=0
    for j in range (0,m-1):
        r+=float((1/(1+((j-255)*
        (j-255))))*hd[j])
    return r

def hsyd_exe(thresh2,img,n,m):
    x,y=np.shape(img)
    # print(x,y)
    # print(np.shape(thresh2))
    c1=np.zeros((x,y,1),np.float32)
    c2=np.zeros((x,y,1),np.float32)
    c3=np.zeros((x,y,1),np.float32)
    c4=np.zeros((x,y,1),np.float32)
    c5=np.zeros((x,y,1),np.float32)
    c6=np.zeros((x,y,1),np.float32)
```

```
    # hs=np.zeros(m,np.uint8)
    # hd=np.zeros(m,np.uint8)
    hdn=np.zeros(m,dtype=np.float32)
    hsn=np.zeros(m,dtype=np.float32)
    for i in range ((n//2)+1,x-1-(n//2)):
     for j in range ((n//2)+1,y-1-(n//2)):
       if (thresh2[i][j]==1):
           #print(i,j)
           roi=img[i-(n//2):i+(n//2),
           j-(n//2):j+(n//2)]
           hs=np.zeros(m,np.uint8)
           hd=np.zeros(m,np.uint8)
           for p in range(0,n-2):
               for q in range(0,n-1):
                 s=int(roi[p][q])+
                 int(roi[p+1][q])
                 r=int(roi[p][q])-
                 int(roi[p+1][q])
                 hs[s]+=1; hd[r+255]+=1
               for k in range(0,m):
                 hsn[k]=(float(hs[k]))/(m)
                 hdn[k]=(float(hd[k]))/(m)
               # print(hs)
               # print(hd)
               # time.sleep(1)
               c1[i][j]=mediahsyd(hsn,m)
               c2[i][j]=energiahsyd(hsn,hdn,m)
               c3[i][j]=contrastehsyd(hdn,m)
               c4[i][j]=hmghsyd(hdn,m)
               c5[i][j]=varianzahsyd
               (hsn,hdn,m,c1[i][j])
               c6[i][j]=correlacionhsyd
               (hsn,hdn,m,c1[i][j])
    return c1,c2,c3,c4,c5,c6

#CREACION DEL DATAFRAME PARA GUARDAR LOS DATOS
#estado sano=0, enfermas=1
columnas=["media","energia",
"contraste","homogeneidad",
"varianza","correlacion"]
```

```
df=pd.DataFrame(columns=columnas)
df["media"]=df["media"].astype("object")
df["energia"]=df["energia"].astype("object")
df["contraste"]=
df["contraste"].astype("object")
df["homogeneidad"]=
df["homogeneidad"].astype("object")
df["varianza"]=df["varianza"].astype("object")
df["correlacion"]=
df["correlacion"].astype("object")

#leer imagenes
for i in range (1,2): #1-733
    #tomate sano
    read = cv2.imread('t_sano'+str(i)+'.jpg')
    m=511
    n=15
    #read = cv2.imread('img_test.jpg')
    print(i)
    scale_percent = 30 # percent of original size
    if(read.shape[0]>read.shape[1]):
     scale_percent=(640.0/read.shape[0])*100.0
    else:
     scale_percent=(640.0/read.shape[1])*100.0

    width = int(read.shape[1] *
    scale_percent / 100)
    height = int(read.shape[0] *
    scale_percent / 100)
    dim = (width, height)
    print(dim)
    img = cv2.resize(read, dim,
    interpolation = cv2.INTER_AREA)
    #cv2.imwrite("img_reesc_ngbvi.png",img)
    #tomate tobrfv
    #imgt = cv2.imread('t_tobrfv89.jpg')
    #cv2.imshow("original",img)

    #separacion de mascaras
    #ibt,igt,irt=cv2.split(imgt)
```

```
ib,ig,ir=cv2.split(img)

#segmentacion por otsu
gray = cv2.cvtColor
(img,cv2.COLOR_BGR2GRAY)
_, thresh = cv2.threshold(gray,0,255,
cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)

#eliminacion de la sombra
thresh[ib>100]=0

#dilatacion para no tener tantos
huecos en la mascara
#de procesamiento del hsyd
thresh = cv2.blur(thresh,(3,3))
thresh[thresh!=0]=1

#EJECUCION DEL SEGUNDO ENMASCARADO
thresh2=mask_hsyd(thresh,n)

# cv2.imwrite("roit1_ngbvi.png",thresh)
# cv2.imwrite("roit_ngbvi.png",thresh2)

#IMAGEN SEGMENTADA RGB
#segr=cv2.multiply(thresh,ir)
#segb=cv2.multiply(thresh,ib)
#segg=cv2.multiply(thresh,ig)
#segm=cv2.merge([segb,segg,segr])
#cv2.imwrite("img_seg_rgb_t_ngbvi.png",
segm)

#INDICES DE VEGETACION RGB
#Y RESULTADO SEGMENTADO
# ibf=ib+0.0
# igf=ig+0.0
# irf=ir+0.0
#      #NGBVI
# gb=igf-ibf
# max_gb=np.amax(gb)
# ngbvi=gb/max_gb
```

```
        # ngbvi = cv2.normalize
        (ngbvi, None, 0, 255,
        cv2.NORM_MINMAX, cv2.CV_8U)
        # ngbvi_t=cv2.multiply(thresh,ngbvi)
        # cv2.imwrite("ngbvi_seg_t_ngbvi.png",
        ngbvi_t)
        #      #NRBVI
        # rb=irf-ibf
        # max_rb=np.amax(rb)
        # nrbvi=rb/max_rb
        # nrbvi = cv2.normalize(nrbvi, None, 0,
        255, cv2.NORM_MINMAX, cv2.CV_8U)
        # nrbvi_t=cv2.multiply(thresh,nrbvi)
        # cv2.imwrite("ngbvi_seg_t_nrbvi.png",
        nrbvi_t)

        #MASCARA PARA EXAMINACION
        #DE IMAGEN COMPLETA
        #thresh3=np.ones((height, width),
        np.uint8)
        #segbyw=cv2.multiply(thresh,gray)
        #print(type(thresh[0,0]))
        #EXTRACCION DE CARACTERISTICAS
        c1,c2,c3,c4,c5,c6=hsyd_exe
        (thresh2,gray,n,m)
            #NORMALIZACION DE LAS CARACTERISTICAS
        c1 = cv2.normalize(c1, None, 0, 255,
        cv2.NORM_MINMAX, cv2.CV_8U)
        c2 = cv2.normalize(c2, None, 0, 255,
        cv2.NORM_MINMAX, cv2.CV_8U)
        c3 = cv2.normalize(c3, None, 0, 255,
        cv2.NORM_MINMAX, cv2.CV_8U)
        c4 = cv2.normalize(c4, None, 0, 255,
        cv2.NORM_MINMAX, cv2.CV_8U)
        c5 = cv2.normalize(c5, None, 0, 255,
        cv2.NORM_MINMAX, cv2.CV_8U)
        c6 = cv2.normalize(c6, None, 0, 255,
        cv2.NORM_MINMAX, cv2.CV_8U)
#GUARDADO DE LAS IMAGENES DE CARACTERISTICAS
        #cv2.imwrite("c1_media_T_ngbvi.png",c1)
```

```
        #cv2.imwrite("c2_energia_T_ngbvi.png",c2)
        #cv2.imwrite("c3_contraste_T_ngbvi.png",c3)
        #cv2.imwrite("c4_hmg_T_ngbvi.png",c4)
        #cv2.imwrite("c5_var_T_ngbvi.png",c5)
        #cv2.imwrite("c4_corr_T_ngbvi.png",c6)

#CALCULO Y GUARDADO DE LOS HISTOGRAMAS
#A PARTIR DE LAS IMAGENES DE CARACTERISTICAS
        #nbl = np.sum(thresh2 == 1)
        #print(thresh2)
        histr1 = cv2.normalize(cv2.calcHist([c1],
        [0],thresh2,[256],[0,256]),None, 0, 1,
        cv2.NORM_MINMAX, cv2.CV_32F)
        #plt.plot(histr1)
        #plt.savefig("hist_t_c1_ngbvi_1.png")

        histr2 = cv2.normalize(cv2.calcHist
        ([c2],[0],thresh2,[256],[0,256]),
        None, 0, 1, cv2.NORM_MINMAX,
        cv2.CV_32F)
        #plt.plot(histr2)
        #plt.savefig("hist_t_c2_ngbvi_1.png")

        histr3 = cv2.normalize(cv2.calcHist
        ([c3],[0],thresh2,[256],[0,256]),
        None, 0, 1, cv2.NORM_MINMAX, cv2.CV_32F)
        #plt.plot(histr3)
        #plt.savefig("hist_t_c3_ngbvi_1.png")

        histr4 = cv2.normalize(cv2.calcHist
        ([c4],[0],thresh2,[256],[0,256]),
        None, 0, 1, cv2.NORM_MINMAX, cv2.CV_32F)
        #plt.plot(histr4)
        #plt.savefig("hist_t_c4_ngbvi_1.png")

        histr5 = cv2.normalize(cv2.calcHist
        ([c5],[0],thresh2,[256],[0,256]),
        None, 0, 1, cv2.NORM_MINMAX, cv2.CV_32F)
        #plt.plot(histr5)
        #plt.savefig("hist_t_c5_ngbvi_1.png")
```

```
        histr6 = cv2.normalize(cv2.calcHist
        ([c6],[0],thresh2,[256],[0,256]),
        None, 0, 1, cv2.NORM_MINMAX, cv2.CV_32F)
        #plt.plot(histr6)
        #plt.savefig("hist_t_c6_ngbvi_1.png")

        #GENERACION DE CSV
        histc1=np.array(histr1)
        histc1=np.array2string(histc1)
        histc2=np.array(histr2)
        histc2=np.array2string(histc2)
        histc3=np.array(histr3)
        histc3=np.array2string(histc3)
        histc4=np.array(histr4)
        histc4=np.array2string(histc4)
        histc5=np.array(histr5)
        histc5=np.array2string(histc5)
        histc6=np.array(histr6)
        histc6=np.array2string(histc6)
        df.at[i,"media"]=histc1
        df.at[i,"energia"]=histc2
        df.at[i,"contraste"]=histc3
        df.at[i,"homogeneidad"]=histc4
        df.at[i,"varianza"]=histc5
        df.at[i,"correlacion"]=histc6

        # cv2.waitKey(0)
        # cv2.destroyAllWindows()
df["estado"]=1
#print(df)
#df.to_csv('res_sano_mean2.csv')
```

Observación de los resultados de la extracción mediante HSD

```
from matplotlib.pyplot import figure, plot
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import csv
```

```
def matxchar(dfx):
    car=["media","energia","contraste",
    "homogeneidad","varianza","correlacion"]
    mat=np.zeros((6,122,256),dtype=float)
    for k in range(len(car)):
        #diccionario=dict['media':None,'energia':
        None,'contraste':None,'homogeneidad':
        None,'varianza':None,'correlacion':None]
        for i in range(1,122):
            tmp = dfx[car[k]][i].replace('[','')
            tmp = tmp.replace(']','')
            tmp = tmp.replace('/n',' ')
            #print(tmp)

            #floats_list = []
            a=0
            for item in tmp.split():
                mat[k,i,a]=float(item)
                a=a+1

    return mat

data_sh=pd.read_csv('res_sano_mean.csv')
data_th=pd.read_csv('res_tobrfv_mean.csv')

dfsh=pd.DataFrame(data_sh)
dfth=pd.DataFrame(data_th)
#print(dfsh["media"][0])
#print(type(dfsh["media"][0]))

mats=matxchar(dfsh)
matt=matxchar(dfth)

#plot 1 car
l=np.linspace(0,255,256)
#ploteo sana vs enfermas
plt.subplot(1,2,1)
for i in range(100,120):
    plt.plot(l,mats[1,i])
```

```
plt.title("Sanas 100-120")
plt.subplot(1,2,2)
for i in range(100,120):
    plt.plot(l,matt[1,i])
plt.title("Enfermas 100-120")
plt.savefig('100-120-svst.png')
plt.show()

#ploteo gral sanas
# plt.subplot(2,3,1)
# for i in range(1,122):
#     plt.plot(l,mats[0,i])
# plt.title("media s")
# plt.subplot(2,3,2)
# for i in range(1,122):
#     plt.plot(l,mats[1,i])
# plt.title("energia s")
# plt.subplot(2,3,3)
# for i in range(1,122):
#     plt.plot(l,mats[2,i])
# plt.title("contraste s")
# plt.subplot(2,3,4)
# for i in range(1,122):
#     plt.plot(l,mats[3,i])
# plt.title("homogeneidad s")
# plt.subplot(2,3,5)
# for i in range(1,122):
#     plt.plot(l,mats[4,i])
# plt.title("varianza s")
# plt.subplot(2,3,6)
# for i in range(1,122):
#     plt.plot(l,mats[5,i])
# plt.title("correlacion s")

# plt.savefig('gral_s.png')
# plt.show()

#ploteo gral tobrfv
# plt.subplot(2,3,1)
# for i in range(1,122):
```

```
#       plt.plot(l,matt[0,i])
# plt.title("media t")
# plt.subplot(2,3,2)
# for i in range(1,122):
#       plt.plot(l,matt[1,i])
# plt.title("energia t")
# plt.subplot(2,3,3)
# for i in range(1,122):
#       plt.plot(l,matt[2,i])
# plt.title("contraste t")
# plt.subplot(2,3,4)
# for i in range(1,122):
#       plt.plot(l,matt[3,i])
# plt.title("homogeneidad t")
# plt.subplot(2,3,5)
# for i in range(1,122):
#       plt.plot(l,matt[4,i])
# plt.title("varianza t")
# plt.subplot(2,3,6)
# for i in range(1,122):
#       plt.plot(l,matt[5,i])
# plt.title("correlacion t")

# plt.savefig('gral_t.png')
# plt.show()

#ploteo individual
# for j in range(1,122):
#       plt.subplot(2,3,1)
#       plt.plot(l,matt[0,j])
#       plt.title("media_"+str(j))
#       plt.subplot(2,3,2)
#       plt.plot(l,matt[1,j])
#       plt.title("energia_"+str(j))
#       plt.subplot(2,3,3)
#       plt.plot(l,matt[2,j])
#       plt.title("contraste_"+str(j))
#       plt.subplot(2,3,4)
#       plt.plot(l,matt[3,j])
#       plt.title("homogeneidad_"+str(j))
```

```
#       plt.subplot(2,3,5)
#       plt.plot(l,matt[4,j])
#       plt.title("varianza_"+str(j))
#       plt.subplot(2,3,6)
#       plt.plot(l,matt[5,j])
#       plt.title("correlacion_"+str(j))
#       plt.savefig('t_'+str(j)+'.png')
#       plt.clf()

# for k in range(1,122):
#       plt.subplot(2,3,1)
#       plt.plot(l,mats[0,k])
#       plt.title("media_"+str(k))
#       plt.subplot(2,3,2)
#       plt.plot(l,mats[1,k])
#       plt.title("energia_"+str(k))
#       plt.subplot(2,3,3)
#       plt.plot(l,mats[2,k])
#       plt.title("contraste_"+str(k))
#       plt.subplot(2,3,4)
#       plt.plot(l,mats[3,k])
#       plt.title("homogeneidad_"+str(k))
#       plt.subplot(2,3,5)
#       plt.plot(l,mats[4,k])
#       plt.title("varianza_"+str(k))
#       plt.subplot(2,3,6)
#       plt.plot(l,mats[5,k])
#       plt.title("correlacion_"+str(k))
#       plt.savefig('s_'+str(k)+'.png')
#       plt.clf()

# print(mats)
# print(mats.shape)

#plt.plot(dfsh["media"][0])
#plt.show()
```

Extracción de características mediante Histogramas de Sumas y Diferencias en recortes

```
from pickletools import uint8
import cv2
from matplotlib.pyplot import hist2d, imshow
import numpy as np
import matplotlib.pyplot as plt
import time
import pandas as pd

def mask_hsyd(thresh,n):
    x,y=np.shape(thresh)
    thresh2=np.zeros((x,y,1),np.uint8)
    for i in range ((n//2)+1,x-1-(n//2)):
        for j in range ((n//2)+1,y-1-(n//2)):
            roi=thresh[i-(n//2):i+(n//2),
            j-(n//2):j+(n//2)]
            if(np.all(roi==1)):
                thresh2[i][j]=1
    return thresh2

def mediahsyd(hs,m):
    s=0.0
    for i in range (0,m-1):
        s+=float(i*hs[i])
    s=s/2
    return s

def varianzahsyd(hs,hd,m,med):
    s=0; r=0
    for i in range (0,m-1):
        s+=float((i-(2*med))*(i-(2*med))*hs[i])
    for j in range(0,m-1):
        r+=float((j-255)*(j-255)*hd[j])
    return (s+r)/2

def correlacionhsyd(hs,hd,m,med):
    s=0; r=0
    for i in range (0,m-1):
        s+=float((i-(2*med))*(i-(2*med))*hs[i])
    for j in range(0,m-1):
        r+=float((j-255)*(j-255)*hd[j])
```

```
    return (s-r)/2

def energiahsyd(hs,hd,m):
    s=0; r=0
    hs=cv2.multiply(hs,hs)
    s=sum(hs)
    hd=cv2.multiply(hd,hd)
    r=sum(hd)
    return s*r

def contrastehsyd(hd,m):
    r=0
    for j in range(0,m-1):
        r+=(j-255)*(j-255)*hd[j]
    return r

def hmghsyd(hd,m):
    r=0
    for j in range (0,m-1):
        r+=float((1/(1+((j-255)*(j-255))))*hd[j])
    return r

def hsyd_exe(thresh2,img,n,m):
    x,y=np.shape(img)
    #print(x,y)
    #print(np.shape(thresh2))
    c1=np.zeros((x,y,1),np.float32)
    c2=np.zeros((x,y,1),np.float32)
    c3=np.zeros((x,y,1),np.float32)
    c4=np.zeros((x,y,1),np.float32)
    c5=np.zeros((x,y,1),np.float32)
    c6=np.zeros((x,y,1),np.float32)
    # hs=np.zeros(m,np.uint8)
    # hd=np.zeros(m,np.uint8)
    hdn=np.zeros(m,dtype=np.float32)
    hsn=np.zeros(m,dtype=np.float32)
    for i in range ((n//2)+1,x-1-(n//2)):
        #print(i)
        for j in range ((n//2)+1,y-1-(n//2)):
            if (thresh2[i][j]==1):
```

```
                    #print(i,j)
                    roi=img[i-(n//2):i+(n//2),
                    j-(n//2):j+(n//2)]
                    hs=np.zeros(m,np.uint8)
                    hd=np.zeros(m,np.uint8)
                    for p in range(0,n-2):
                        for q in range(0,n-1):
                            s=int(roi[p][q])+
                            int(roi[p+1][q])
                            r=int(roi[p][q])-
                            int(roi[p+1][q])
                            hs[s]+=1; hd[r+255]+=1
                    for k in range(0,m):
                        hsn[k]=(float(hs[k]))/(m)
                        hdn[k]=(float(hd[k]))/(m)
                    # print(hs)
                    # print(hd)
                    # time.sleep(1)
                    c1[i][j]=mediahsyd(hsn,m)
                    c2[i][j]=energiahsyd(hsn,hdn,m)
                    c3[i][j]=contrastehsyd(hdn,m)
                    c4[i][j]=hmghsyd(hdn,m)
                    c5[i][j]=varianzahsyd
                (hsn,hdn,m,c1[i][j])
                    c6[i][j]=correlacionhsyd
                    (hsn,hdn,m,c1[i][j])
    return c1,c2,c3,c4,c5,c6

#CREACION DEL DATAFRAME PARA GUARDAR LOS DATOS
#estado sano=0, enfermas=1
columnas=["media","energia","contraste",
"homogeneidad","varianza","correlacion"]
df=pd.DataFrame(columns=columnas)
df["media"]=df["media"].astype("object")
df["energia"]=df["energia"].astype("object")
df["contraste"]=df["contraste"].astype("object")
df["homogeneidad"]=df["homogeneidad"].
astype("object")
df["varianza"]=df["varianza"].astype("object")
df["correlacion"]=df["correlacion"].astype("object")
```

```
#leer imagenes
for i in range (1,168): #1-167
    #tomate sano
    read = cv2.imread('t_tobrfv_cut'+str(i)+'.jpg')
    m=511
    n=15
    #read = cv2.imread('img_test.jpg')
    print(i)
    # scale_percent = 30 # percent of original size

    if(read.shape[0]>=301 and read.shape[1]<301):
            scale_percent=(301.0/read.shape[0])*100.0
    elif(read.shape[0]<301 and read.shape[1]>=301):
            scale_percent=(301.0/read.shape[1])*100.0
    elif(read.shape[0]>=301 and read.shape[1]>=301
    and read.shape[0]>read.shape[1]):
            scale_percent=(301.0/read.shape[0])*100.0
    elif(read.shape[0]>=301 and read.shape[1]>=301
    and read.shape[0]<read.shape[1]):
            scale_percent=(301.0/read.shape[1])*100.0
    else:
            scale_percent=100

    width = int(read.shape[1] * scale_percent / 100)
    height = int(read.shape[0] * scale_percent / 100)
    dim = ( height,width)
    #print(dim)
    img = cv2.resize(read, dim, interpolation
    = cv2.INTER_AREA)
    #print(img.shape)
    #cv2.imwrite("img_reesc_ngbvi.png",img)
    #tomate tobrfv
    #imgt = cv2.imread('t_tobrfv89.jpg')
    #cv2.imshow("original",img)

    #separacion de mascaras
    #ibt,igt,irt=cv2.split(imgt)
    #ib,ig,ir=cv2.split(img)
```

```
#segmentacion por otsu
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
#_, thresh = cv2.threshold(gray,0,255,
cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)

#eliminacion de la sombra
#thresh[ib>100]=0
#print(gray.shape[0])
thresh=np.zeros((gray.shape[0],gray.shape[1]),
dtype=np.uint8)
thresh[(int(n/2)):thresh.shape[0]-(int(n/2)),
(int(n/2)):thresh.shape[1]-(int(n/2))]=1
#print(thresh)
#dilatacion para no tener tantos huecos
en la mascara
#de procesamiento del hsyd
#thresh = cv2.blur(thresh,(3,3))
#thresh[thresh!=0]=1

#EJECUCION DEL SEGUNDO ENMASCARADO
#thresh2=mask_hsyd(thresh,n)
# cv2.imwrite("roit1_ngbvi.png",thresh)
# cv2.imwrite("roit_ngbvi.png",thresh2)

#IMAGEN SEGMENTADA RGB
#segr=cv2.multiply(thresh,ir)
#segb=cv2.multiply(thresh,ib)
#segg=cv2.multiply(thresh,ig)
#segm=cv2.merge([segb,segg,segr])
#cv2.imwrite("img_seg_rgb_t_ngbvi.png", segm)

#INDICES DE VEGETACION RGB Y RESULTADO SEGMENTADO
# ibf=ib+0.0
# igf=ig+0.0
# irf=ir+0.0
#     #NGBVI
# gb=igf-ibf
# max_gb=np.amax(gb)
# ngbvi=gb/max_gb
# ngbvi = cv2.normalize(ngbvi, None, 0, 255,
```

```
cv2.NORM_MINMAX, cv2.CV_8U)
# ngbvi_t=cv2.multiply(thresh,ngbvi)
# cv2.imwrite("ngbvi_seg_t_ngbvi.png",ngbvi_t)
#       #NRBVI
# rb=irf-ibf
# max_rb=np.amax(rb)
# nrbvi=rb/max_rb
# nrbvi = cv2.normalize(nrbvi, None, 0, 255,
cv2.NORM_MINMAX, cv2.CV_8U)
# nrbvi_t=cv2.multiply(thresh,nrbvi)
# cv2.imwrite("ngbvi_seg_t_nrbvi.png",nrbvi_t)

#MASCARA PARA EXAMINACION DE IMAGEN COMPLETA
#thresh3=np.ones((height, width),np.uint8)
#segbyw=cv2.multiply(thresh,gray)

#EXTRACCION DE CARACTERISTICAS
#print(type(thresh[0,0]))
c1,c2,c3,c4,c5,c6=hsyd_exe(thresh,gray,n,m)
    #NORMALIZACION DE LAS CARACTERISTICAS
# c1 = cv2.normalize(c1, None, 0, 255,
cv2.NORM_MINMAX, cv2.CV_8U)
# c2 = cv2.normalize(c2, None, 0, 255,
cv2.NORM_MINMAX, cv2.CV_8U)
# c3 = cv2.normalize(c3, None, 0, 255,
cv2.NORM_MINMAX, cv2.CV_8U)
# c4 = cv2.normalize(c4, None, 0, 255,
cv2.NORM_MINMAX, cv2.CV_8U)
# c5 = cv2.normalize(c5, None, 0, 255,
cv2.NORM_MINMAX, cv2.CV_8U)
# c6 = cv2.normalize(c6, None, 0, 255,
cv2.NORM_MINMAX, cv2.CV_8U)
#GUARDADO DE LAS IMAGENES DE CARACTERISTICAS
# cv2.imwrite("c1_media_T_ngbvi.png",c1)
# cv2.imwrite("c2_energia_T_ngbvi.png",c2)
# cv2.imwrite("c3_contraste_T_ngbvi.png",c3)
# cv2.imwrite("c4_hmg_T_ngbvi.png",c4)
# cv2.imwrite("c5_var_T_ngbvi.png",c5)
# cv2.imwrite("c4_corr_T_ngbvi.png",c6)
```

```
#CALCULO Y GUARDADO DE LOS HISTOGRAMAS
#A PARTIR DE LAS IMAGENES DE CARACTERISTICAS
#nbl = np.sum(thresh2 == 1)

histr1 = cv2.calcHist([c1],[0],thresh,[256],
[0,256])#,None, 0, 1, cv2.NORM_MINMAX,
cv2.CV_32F)
#plt.plot(histr1)
#plt.savefig("hist_t_c1_ngbvi_1.png")

histr2 = cv2.calcHist([c2],[0],thresh,[256],
[0,256])#,None, 0, 1, cv2.NORM_MINMAX,
cv2.CV_32F)
#plt.plot(histr2)
#plt.savefig("hist_t_c2_ngbvi_1.png")

histr3 = cv2.calcHist([c3],[0],thresh,[256],
[0,256])#,None, 0, 1, cv2.NORM_MINMAX,
cv2.CV_32F)
#plt.plot(histr3)
#plt.savefig("hist_t_c3_ngbvi_1.png")

histr4 = cv2.calcHist([c4],[0],thresh,[256],
[0,256])#,None, 0, 1, cv2.NORM_MINMAX,
cv2.CV_32F)
#plt.plot(histr4)
#plt.savefig("hist_t_c4_ngbvi_1.png")

histr5 = cv2.calcHist([c5],[0],thresh,[256],
[0,256])#,None, 0, 1, cv2.NORM_MINMAX,
cv2.CV_32F)
#plt.plot(histr5)
#plt.savefig("hist_t_c5_ngbvi_1.png")

histr6 = cv2.calcHist([c6],[0],thresh,[256],
[0,256])#,None, 0, 1, cv2.NORM_MINMAX,
cv2.CV_32F)
#plt.plot(histr6)
#plt.savefig("hist_t_c6_ngbvi_1.png")
```

```
        #GENERACION DE CSV
        histc1=np.array(histr1)
        histc1=np.array2string(histc1)
        histc2=np.array(histr2)
        histc2=np.array2string(histc2)
        histc3=np.array(histr3)
        histc3=np.array2string(histc3)
        histc4=np.array(histr4)
        histc4=np.array2string(histc4)
        histc5=np.array(histr5)
        histc5=np.array2string(histc5)
        histc6=np.array(histr6)
        histc6=np.array2string(histc6)
        df.at[i,"media"]=histc1
        df.at[i,"energia"]=histc2
        df.at[i,"contraste"]=histc3
        df.at[i,"homogeneidad"]=histc4
        df.at[i,"varianza"]=histc5
        df.at[i,"correlacion"]=histc6

        # cv2.waitKey(0)
        # cv2.destroyAllWindows()
df["estado"]=1
#print(df)
df.to_csv('res_tobrfv_cut.csv')
```

Clasificación mediante RNN

```
import sklearn
from sklearn.neural_network import MLPClassifier
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import classification_report,
confusion_matrix
from sklearn.metrics import mean_squared_error
from math import sqrt
from sklearn.metrics import r2_score
from matplotlib.pyplot import figure, plot
import numpy as np
```

```
import pandas as pd
import matplotlib.pyplot as plt
import csv

def matxchar(dfx,r):
    car=["media","energia","contraste","homogeneidad",
    "varianza","correlacion"]
    mat=np.zeros((208,256*6),dtype=float)
    for i in range(1-r,208-r):
        #diccionario=dict['media':None,'energia':None,
        'contraste':None,'homogeneidad':None,
        'varianza':None,'correlacion':None]
        for k in range(0,5):
            tmp = dfx[car[k]][i].replace('[','')
            tmp = tmp.replace(']','')
            tmp = tmp.replace('/n',' ')
            #print(tmp)

            #floats_list = []
            a=0
            if(r==0):
                for item in tmp.split():
                    mat[i-1,(k*256)+a]=float(item)
                    a=a+1
            else:
                for item in tmp.split():
                    mat[i,(k*256)+a]=float(item)
                    a=a+1

    return mat

data_sh=pd.read_csv('res_sano_cut_norm.csv',index_col=0)
data_th=pd.read_csv('res_tobrfv_cut_norm.csv',index_col=0)
#print(data_th)

#print(type(data_sh["media"][0]))

mats=matxchar(data_sh,1)
matt=matxchar(data_th,0)
#print(mats)
```

```
#print(mats.shape)
#print(matt)
#print(matt.shape)

matf=np.concatenate((mats,matt),axis=0)
#print(matf)
print(matf.shape)

s=np.zeros((208),dtype=int)
e=np.ones((208),dtype=int)
etiq=np.concatenate((s,e))
print(etiq.shape)

# Split dataset into training set and test set
#semilla = 484,1,4,23, 84,101, 117, 136,195,234
X_train, X_test, y_train, y_test = train_test_split
(matf, etiq, test_size=0.3,random_state=234)
# 70% training and 30% test

#Create the Classifier
#activation{ identity , logistic , tanh , relu
#solver{ lbfgs , sgd , adam }, default= adam ,
mlp = MLPClassifier
(activation='tanh', solver='sgd', max_iter=500)

# Train Classifer
mlp.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = mlp.predict(X_test)

# #imprimir reporte
# print(confusion_matrix(y_test, y_pred))
# print(classification_report(y_test, y_pred))

# Model Accuracy: how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

# Model Precision: what percentage of positive tuples are
labeled as such?
```

```
print("Precision:",metrics.precision_score(y_test, y_pred))

# Model Recall: what percentage of positive tuples are
labelled as such?
print("Recall:",metrics.recall_score(y_test, y_pred))

print("F1_score:",metrics.f1_score(y_test, y_pred))
# # K-Fold Cross-Validation
# from sklearn.model_selection import cross_validate
# def cross_validation(model, _X, _y, _cv=10):
#'''Function to perform 10 Folds Cross-Validation
#       Parameters
#       ----------
#      model: Python Class, default=None
#      This is the machine learning algorithm
to be used for training.
#       _X: array
#      This is the matrix of features.
#       _y: array
#      This is the target variable.
#       _cv: int, default=5
#   Determines the number of folds for cross-validation.
#       Returns
#       -------
#The function returns a dictionary containing
#the metrics 'accuracy', 'precision',
# 'recall', 'f1' for both training set and validation set.
# '''
#    _scoring = ['accuracy', 'precision', 'recall', 'f1']
#    results = cross_validate(estimator=model,
#                             X=_X,
#                             y=_y,
#                             cv=_cv,
#                             scoring=_scoring,
#                             return_train_score=True)

# return {"Training Accuracy scores":
results['train_accuracy'],
#       "Mean Training Accuracy":
results['train_accuracy'].mean()*100,
```

```
#     "Training Precision scores":
results['train_precision'],
#     "Mean Training Precision":
results['train_precision'].mean(),
#     "Training Recall scores": results['train_recall'],
#     "Mean Training Recall":
results['train_recall'].mean(),
# "Training F1 scores": results['train_f1'],
# "Mean Training F1 Score": results['train_f1'].mean(),
# "Validation Accuracy scores":
results['test_accuracy'],
# "Mean Validation Accuracy":
results['test_accuracy'].mean()*100,
# "Validation Precision scores":
results['test_precision'],
#  "Mean Validation Precision":
results['test_precision'].mean(),
#  "Validation Recall scores": results['test_recall'],
#  "Mean Validation Recall":
results['test_recall'].mean(),
#"Validation F1 scores": results['test_f1'],
#"Mean Validation F1 Score": results['test_f1'].mean()
#   }

## Grouped Bar Chart for both training and validation data
# def plot_result(x_label, y_label, plot_title, train_data,
val_data):
# '''Function to plot a grouped bar chart showing
the training and validation
#   results of the ML model in each fold after
applying K-fold cross-validation.
#  Parameters
#  ----------
#  x_label: str,
#  Name of the algorithm used for training e.g 'SVM'
#  y_label: str,
#  Name of metric being visualized e.g 'Accuracy'
#  plot_title: str,
#  This is the title of the plot e.g 'Accuracy Plot'
#  train_result: list, array
```

```
#  This is the list containing either training precision,
accuracy, or f1 score.
#  val_result: list, array
#  This is the list containing either validation
precision, accuracy, or f1 score.
#  Returns
#  ---------
#  The function returns a Grouped Barchart showing the
training and validation result in each fold.
#          '''
#  # Set size of plot
#    plt.figure(figsize=(12,6))
#    labels = ["1st Fold", "2nd Fold", "3rd Fold",
"4th Fold","5th Fold","6th Fold", "7th Fold",
"8th Fold", "9th Fold", "10th Fold"]
#    X_axis = np.arange(len(labels))
#    ax = plt.gca()
#    plt.ylim(0.40000, 1)
#    plt.bar(X_axis-0.2, train_data, 0.4, color='black',
label='Training')
#    plt.bar(X_axis+0.2, val_data, 0.4, color='gray',
label='Validation')
#    plt.title(plot_title, fontsize=30)
#    plt.xticks(X_axis, labels)
#    plt.xlabel(x_label, fontsize=14)
#    plt.ylabel(y_label, fontsize=14)
#    plt.legend()
#    plt.grid(True)
#    plt.savefig(plot_title+"_rnn.png")
#    plt.show()

# def matxchar(dfx,r):
# car=["media","energia","contraste",
"homogeneidad","varianza","correlacion"]
#      mat=np.zeros((208,256*6),dtype=float)
#      for i in range(1-r,208-r):
# #diccionario=dict['media':None,'energia':
None,'contraste':None,'homogeneidad':
None,'varianza':None,'correlacion':None]
#    for k in range(0,5):
```

```
#    tmp = dfx[car[k]][i].replace('[','')
#    tmp = tmp.replace(']','')
#    tmp = tmp.replace('/n',' ')
#    #print(tmp)

#    #floats_list = []
#    a=0
#    if(r==0):
#      for item in tmp.split():
#           mat[i-1,(k*256)+a]=float(item)
#           a=a+1
#     else:
#       for item in tmp.split():
#           mat[i,(k*256)+a]=float(item)
#           a=a+1
#      return mat

# data_sh=pd.read_csv
('res_sano_cut_norm.csv',index_col=0)
# data_th=pd.read_csv
('res_tobrfv_cut_norm.csv',index_col=0)
# #print(data_th)

# #print(type(data_sh["media"][0]))

# mats=matxchar(data_sh,1)
# matt=matxchar(data_th,0)
# #print(mats)
# #print(mats.shape)
# #print(matt)
# #print(matt.shape)

# matf=np.concatenate((mats,matt),axis=0)
# #print(matf)
# print(matf.shape)
# #print(matf)

# s=np.zeros((208),dtype=int)
# e=np.ones((208),dtype=int)
# etiq=np.concatenate((s,e))
```

```
# print(etiq.shape)
# #print(etiq)
# ## Split dataset into training set and test set
#semilla = 132
# #X_train, X_test, y_train, y_test =
train_test_split(matf, etiq,
test_size=0.3,random_state=132)
# 70% training and 30% test

# #shuffle the data for classification problems
# from sklearn.model_selection
import StratifiedKFold
# skf=StratifiedKFold
(10,shuffle=True,random_state=1)
# # Create RNN classifier
# clf = MLPClassifier
(activation='logistic', solver='adam',
max_iter=1000,random_state=0)
# clf_res=cross_validation(clf,matf,etiq,skf)
# print(clf_res)

# # Plot Accuracy Result
# model_name = "SVM"
# plot_result(model_name,"Accuracy",
#     "Accuracy scores in 10 Folds",
#      clf_res["Training Accuracy scores"],
#      clf_res["Validation Accuracy scores"])

# # Plot Precision Result
# plot_result(model_name,
#      "Precision",
#      "Precision scores in 10 Folds",
#      clf_res["Training Precision scores"],
#      clf_res["Validation Precision scores"])

# # Plot Recall Result
# plot_result(model_name,
#      "Recall",
#      "Recall scores in 10 Folds",
#      clf_res["Training Recall scores"],
```

```
#         clf_res["Validation Recall scores"])

## Plot F1-Score Result
# plot_result(model_name,
#         "F1",
#         "F1 Scores in 10 Folds",
#         clf_res["Training F1 scores"],
#         clf_res["Validation F1 scores"])
```

Clasificación mediante SVM

```
from sklearn import svm
from sklearn.model_selection
import train_test_split
from sklearn import metrics
from matplotlib.pyplot import figure, plot
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import csv

def matxchar(dfx,r):
    car=["media","energia","contraste",
    "homogeneidad","varianza","correlacion"]
    mat=np.zeros((208,256*6),dtype=float)
    for i in range(1-r,208-r):
        #diccionario=dict['media':None,'energia':
        None,'contraste':None,'homogeneidad':
        None,'varianza':None,'correlacion':None]
        for k in range(0,5):
            tmp = dfx[car[k]][i].replace('[','')
            tmp = tmp.replace(']','')
            tmp = tmp.replace('/n',' ')
            #print(tmp)

            #floats_list = []
            a=0
            if(r==0):
                for item in tmp.split():
                    mat[i-1,(k*256)+a]=float(item)
```

```
                    a=a+1
            else:
                for item in tmp.split():
                    mat[i,(k*256)+a]=float(item)
                    a=a+1

    return mat

data_sh=pd.read_csv
('res_sano_cut_norm.csv',index_col=0)
data_th=pd.read_csv
('res_tobrfv_cut_norm.csv',index_col=0)
#print(data_th)

#print(type(data_sh["media"][0]))

mats=matxchar(data_sh,1)
matt=matxchar(data_th,0)
#print(mats)
#print(mats.shape)
#print(matt)
#print(matt.shape)

matf=np.concatenate((mats,matt),axis=0)
#print(matf)
print(matf.shape)

s=np.zeros((208),dtype=int)
e=np.ones((208),dtype=int)
etiq=np.concatenate((s,e))
print(etiq.shape)

# Split dataset into training set and test set
#semilla = 894,701,632, 508,448,342,309,1,23
X_train, X_test, y_train, y_test =
train_test_split(matf, etiq,
test_size=0.3,random_state=94)
# 70% training and 30% test

#Create a svm Classifier
```

```
#kernel{ linear , poly , rbf ,
    sigmoid , precomputed }
clf = svm.SVC(kernel='poly')

#Train the model using the training sets
clf.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

# Model Accuracy:
#how often is the classifier correct?
print("Accuracy:",
metrics.accuracy_score(y_test, y_pred))

# Model Precision: what percentage
#of positive tuples are
labeled as such?
print("Precision:",
metrics.precision_score(y_test, y_pred))

# Model Recall: what
#percentage of positive tuples are
labelled as such?
print("Recall:",
metrics.recall_score(y_test, y_pred))

print("F1_score:",metrics.f1_score(y_test, y_pred))

# # K-Fold Cross-Validation
# from sklearn.model_selection
import cross_validate
# def cross_validation(model, _X, _y, _cv=10):
# '''Function to perform 10 Folds Cross-Validation
#  Parameters
# ----------
# model: Python Class, default=None
# This is the machine learning algorithm to be used
for training.
# _X: array
```

```
# This is the matrix of features.
# _y: array
# This is the target variable.
# _cv: int, default=5
# Determines the number of folds for cross-validation.
# Returns
# -------
# The function returns a dictionary containing the
metrics 'accuracy', 'precision',
#  'recall', 'f1'
#for both training set and validation set.
#  '''
#  _scoring = ['accuracy', 'precision', 'recall', 'f1']
#  results = cross_validate(estimator=model,
#                  X=_X,
#                  y=_y,
#                  cv=_cv,
#                  scoring=_scoring,
#                  return_train_score=True)

# return {"Training Accuracy scores":
results['train_accuracy'],
#  "Mean Training Accuracy":
results['train_accuracy'].mean()*100,
#  "Training Precision scores":
results['train_precision'],
#  "Mean Training Precision":
results['train_precision'].mean(),
#   "Training Recall scores":
results['train_recall'],
#   "Mean Training Recall":
results['train_recall'].mean(),
#   "Training F1 scores": results['train_f1'],
#   "Mean Training F1 Score":
results['train_f1'].mean(),
#   "Validation Accuracy scores":
results['test_accuracy'],
#   "Mean Validation Accuracy":
results['test_accuracy'].mean()*100,
#   "Validation Precision scores":
```

```
results['test_precision'],
#     "Mean Validation Precision":
results['test_precision'].mean(),
#     "Validation Recall scores":
results['test_recall'],
#     "Mean Validation Recall":
results['test_recall'].mean(),
#     "Validation F1 scores": results['test_f1'],
#     "Mean Validation F1 Score":
results['test_f1'].mean()
#     }

## Grouped Bar Chart for
#both training and validation data
# def plot_result(x_label, y_label,
plot_title, train_data,val_data):
# '''Function to plot a grouped bar chart showing the
training and validation
#    results of the ML model in each fold after applying
K-fold cross-validation.
#    Parameters
#    ----------
#    x_label: str,
#    Name of the algorithm used for training e.g 'SVM'

#    y_label: str,
#    Name of metric being visualized e.g 'Accuracy'
#    plot_title: str,
#    This is the title of the plot e.g 'Accuracy Plot'

#    train_result: list, array
#  This is the list containing either training precision,
accuracy, or f1 score.
#    val_result: list, array
#    This is the list containing either validation
precision, accuracy, or f1 score.
#    Returns
#    -------
#    The function returns a Grouped Barchart showing the
training and validation result
```

```
#            in each fold.
#           '''

#    # Set size of plot
#    plt.figure(figsize=(12,6))
#    labels = ["1st Fold", "2nd Fold", "3rd Fold",
"4th Fold","5th Fold","6th Fold", "7th Fold",
"8th Fold", "9th Fold", "10th Fold"]
#    X_axis = np.arange(len(labels))
#    ax = plt.gca()
#    plt.ylim(0.40000, 1)
#    plt.bar(X_axis-0.2, train_data, 0.4, color='black',
label='Training')
#    plt.bar(X_axis+0.2, val_data, 0.4, color='gray',
label='Validation')
#    plt.title(plot_title, fontsize=30)
#    plt.xticks(X_axis, labels)
#    plt.xlabel(x_label, fontsize=14)
#    plt.ylabel(y_label, fontsize=14)
#    plt.legend()
#    plt.grid(True)
#    plt.savefig(plot_title+"_svm.png")
#    plt.show()

# def matxchar(dfx,r):
#car=["media","energia","contraste","homogeneidad",
"varianza","correlacion"]
#mat=np.zeros((208,256*6),dtype=float)
#for i in range(1-r,208-r):
##diccionario=dict['media':None,'energia':
None,'contraste':None,'homogeneidad':None,
'varianza':None,'correlacion':None]
#     for k in range(0,5):
#         tmp = dfx[car[k]][i].replace('[','')
#         tmp = tmp.replace(']','')
#         tmp = tmp.replace('/n',' ')
#         #print(tmp)

#         #floats_list = []
#         a=0
```

```
#         if(r==0):
#           for item in tmp.split():
#               mat[i-1,(k*256)+a]=float(item)
#               a=a+1
#         else:
#           for item in tmp.split():
#               mat[i,(k*256)+a]=float(item)
#               a=a+1

#       return mat

# data_sh=pd.read_csv
('res_sano_cut_norm.csv',index_col=0)
# data_th=pd.read_csv
('res_tobrfv_cut_norm.csv',index_col=0)
# #print(data_th)

# #print(type(data_sh["media"][0]))

# mats=matxchar(data_sh,1)
# matt=matxchar(data_th,0)
# #print(mats)
# #print(mats.shape)
# #print(matt)
# #print(matt.shape)

# matf=np.concatenate((mats,matt),axis=0)
# #print(matf)
# print(matf.shape)

# s=np.zeros((208),dtype=int)
# e=np.ones((208),dtype=int)
# etiq=np.concatenate((s,e))
# print(etiq.shape)

# ## Split dataset into training set and test set
#semilla = 132
# #X_train, X_test, y_train, y_test = train_test_split
(matf, etiq, test_size=0.3,random_state=132)
# 70% training and 30% test
```

```
## shuffle the data for classification problems
# from sklearn.model_selection import StratifiedKFold
# skf=StratifiedKFold(10, shuffle=True, random_state=0)
## Create SVM classifier
##kernel{ linear , poly , rbf ,
    sigmoid , precomputed }
# clf = svm.SVC(kernel='rbf', random_state=0)
# clf_res=cross_validation(clf, matf, etiq, skf)
# print(clf_res)

## Plot Accuracy Result
# model_name = "SVM"
# plot_result(model_name,"Accuracy",
#"Accuracy scores in 10 Folds",
#clf_res["Training Accuracy scores"],
#clf_res["Validation Accuracy scores"])

## Plot Precision Result
# plot_result(model_name,
# "Precision",
# "Precision scores in 10 Folds",
# clf_res["Training Precision scores"],
# clf_res["Validation Precision scores"])

## Plot Recall Result
# plot_result(model_name,
# "Recall",
# "Recall scores in 10 Folds",
# clf_res["Training Recall scores"],
# clf_res["Validation Recall scores"])

## Plot F1-Score Result
# plot_result(model_name,
# "F1",
# "F1 Scores in 10 Folds",
# clf_res["Training F1 scores"],
# clf_res["Validation F1 scores"])
```

Clasificación mediante Desition Trees

```
from sklearn.tree import DecisionTreeClassifier
# Import Decision Tree Classifier
from sklearn.model_selection import train_test_split
# Import train_test_split function
from sklearn import metrics
#Import scikit-learn metrics
#module for accuracy calculation
from sklearn.model_selection import train_test_split
from sklearn import metrics
from matplotlib.pyplot import figure, plot
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import csv

## K-Fold Cross-Validation
# from sklearn.model_selection import cross_validate
# def cross_validation(model, _X, _y, _cv=10):
# '''Function to perform 10 Folds Cross-Validation
# Parameters
# ----------
# model: Python Class, default=None
# This is the machine learning algorithm to be
used for training.
#  _X: array
#  This is the matrix of features.
#  _y: array
#  This is the target variable.
#  _cv: int, default=5
#  Determines the number of folds
for cross-validation.
#  Returns
#  -------
#  The function returns a dictionary containing
the metrics 'accuracy', 'precision',
# 'recall', 'f1' for both training set
and validation set.
#  '''
```

```
#_scoring=['accuracy', 'precision', 'recall', 'f1']
# results = cross_validate(estimator=model,
#                          X=_X,
#                          y=_y,
#                          cv=_cv,
#                          scoring=_scoring,
#                         return_train_score=True)

# return {"Training Accuracy scores":
results['train_accuracy'],
# "Mean Training Accuracy":
results['train_accuracy'].mean()*100,
# "Training Precision scores":
results['train_precision'],
# "Mean Training Precision":
results['train_precision'].mean(),
# "Training Recall scores": results['train_recall'],
# "Mean Training Recall":
results['train_recall'].mean(),
# "Training F1 scores": results['train_f1'],
# "Mean Training F1 Score": results['train_f1'].mean(),
# "Validation Accuracy scores":
results['test_accuracy'],
# "Mean Validation Accuracy":
results['test_accuracy'].mean()*100,
# "Validation Precision scores":
results['test_precision'],
# "Mean Validation Precision":
results['test_precision'].mean(),
# "Validation Recall scores": results['test_recall'],
# "Mean Validation Recall":
results['test_recall'].mean(),
# "Validation F1 scores": results['test_f1'],
# "Mean Validation F1 Score": results['test_f1'].mean()
# }

## Grouped Bar Chart for both training
and validation data
# def plot_result(x_label, y_label, plot_title,
train_data, val_data):
```

```
# '''Function to plot a grouped bar chart showing the
training and validation
# results of the ML model in each fold after applying
K-fold cross-validation.
# Parameters
# ----------
# x_label: str,
# Name of the algorithm used for training e.g 'Decision
Tree'

# y_label: str,
# Name of metric being visualized e.g 'Accuracy'
# plot_title: str,
# This is the title of the plot e.g 'Accuracy Plot'

# train_result: list, array
#This is the list containing either training precision,
accuracy, or f1 score.

# val_result: list, array
# This is the list containing either validation
precision, accuracy, or f1 score.
# Returns
# -----
# The function returns a Grouped Barchart showing the
training and validation result
# in each fold.
#  '''

#  # Set size of plot
#   plt.figure(figsize=(12,6))
#   labels = ["1st Fold", "2nd Fold", "3rd Fold",
"4th Fold","5th Fold","6th Fold", "7th Fold",
"8th Fold", "9th Fold", "10th Fold"]
# X_axis = np.arange(len(labels))
# ax = plt.gca()
# plt.ylim(0.40000, 1)
# plt.bar(X_axis-0.2, train_data, 0.4, color='black',
label='Training')
# plt.bar(X_axis+0.2, val_data, 0.4, color='gray',
```

```
label='Validation')
# plt.title(plot_title, fontsize=30)
# plt.xticks(X_axis, labels)
# plt.xlabel(x_label, fontsize=14)
# plt.ylabel(y_label, fontsize=14)
# plt.legend()
# plt.grid(True)
# plt.savefig(plot_title+"_dtree.png")
# plt.show()

# def matxchar(dfx,r):
# car=["media","energia","contraste","homogeneidad",
"varianza","correlacion"]
#       mat=np.zeros((208,256*6),dtype=float)
#       for i in range(1-r,208-r):
##diccionario=dict['media':None,'energia':
None,'contraste':None,'homogeneidad':None,'varianza':
None,'correlacion':None]
# for k in range(0,5):
#     tmp = dfx[car[k]][i].replace('[','')
#     tmp = tmp.replace(']','')
#     tmp = tmp.replace('/n',' ')
#     #print(tmp)

#     #floats_list = []
#     a=0
#     if(r==0):
#       for item in tmp.split():
#         mat[i-1,(k*256)+a]=float(item)
#         a=a+1
#     else:
#         for item in tmp.split():
#             mat[i,(k*256)+a]=float(item)
#             a=a+1

#return mat

# data_sh=pd.read_csv
('res_sano_cut_norm.csv',index_col=0)
# data_th=pd.read_csv
```

```
('res_tobrfv_cut_norm.csv',index_col=0)
##print(data_th)

##print(type(data_sh["media"][0]))

# mats=matxchar(data_sh,1)
# matt=matxchar(data_th,0)
##print(mats)
##print(mats.shape)
##print(matt)
##print(matt.shape)

# matf=np.concatenate((mats,matt),axis=0)
##print(matf)
# print(matf.shape)

# s=np.zeros((208),dtype=int)
# e=np.ones((208),dtype=int)
# etiq=np.concatenate((s,e))
# print(etiq.shape)

##shuffle the data for classification problems
# from sklearn.model_selection import StratifiedKFold
# skf=StratifiedKFold(10,shuffle=True,random_state=1)
## Create Decision Tree classifer object
##criterion{ gini , entropy , log_loss },
default= gini
# clf = DecisionTreeClassifier
(criterion='gini',random_state=1)
# clf_res=cross_validation(clf,matf,etiq,skf)
# print(clf_res)

## Plot Accuracy Result
# model_name = "Decision Tree"
# plot_result(model_name,"Accuracy",
#     "Accuracy scores in 10 Folds",
#     clf_res["Training Accuracy scores"],
#     clf_res["Validation Accuracy scores"])

## Plot Precision Result
```

```
# plot_result(model_name,
#      "Precision",
#      "Precision scores in 10 Folds",
#      clf_res["Training Precision scores"],
#      clf_res["Validation Precision scores"])

## Plot Recall Result
# plot_result(model_name,
#       "Recall",
#       "Recall scores in 10 Folds",
#        clf_res["Training Recall scores"],
#        clf_res["Validation Recall scores"])

## Plot F1-Score Result
# plot_result(model_name,
#       "F1",
#       "F1 Scores in 10 Folds",
#        clf_res["Training F1 scores"],
#        clf_res["Validation F1 scores"])

# clf2 = DecisionTreeClassifier
(min_samples_split=10,random_state=1)
# clf_res2 = cross_validation(clf2, matf, etiq, 10)
# print(clf_res2)

## Plot Accuracy Result
# plot_result(model_name,
#       "Accuracy",
#       "Accuracy scores in 10 Folds",
#       clf_res2["Training Accuracy scores"],
#       clf_res2["Validation Accuracy scores"])

## Plot Precision Result
# plot_result(model_name,
#      "Precision",
#      "Precision scores in 10 Folds",
#      clf_res2["Training Precision scores"],
#      clf_res2["Validation Precision scores"])
## Plot Recall Result
# plot_result(model_name,
```

```
#    "Recall",
#    "Recall scores in 10 Folds",
#    clf_res2["Training Recall scores"],
#    clf_res2["Validation Recall scores"])
## Plot F1-Score Result
# plot_result(model_name,
#    "F1",
#    "F1 Scores in 10 Folds",
#     clf_res2["Training F1 scores"],
#     clf_res2["Validation F1 scores"])

def matxchar(dfx,r):
car=["media","energia","contraste","homogeneidad",
 "varianza",     "correlacion"]
  mat=np.zeros((208,256*6),dtype=float)
  for i in range(1-r,208-r):
#diccionario=dict['media':None,'energia':None,
    'contraste':None,'homogeneidad':None,'varianza':
     None,'correlacion':None]
     for k in range(0,5):
        tmp = dfx[car[k]][i].replace('[','')
        tmp = tmp.replace(']','')
        tmp = tmp.replace('/n',' ')
        #print(tmp)
        #floats_list = []
        a=0
        if(r==0):
            for item in tmp.split():
                mat[i-1,(k*256)+a]=float(item)
                a=a+1
        else:
            for item in tmp.split():
                mat[i,(k*256)+a]=float(item)
                a=a+1
    return mat

data_sh=pd.read_csv
('res_sano_cut_norm.csv',index_col=0)
data_th=pd.read_csv
('res_tobrfv_cut_norm.csv',index_col=0)
```

```
#print(data_th)

#print(type(data_sh["media"][0]))

mats=matxchar(data_sh,1)
matt=matxchar(data_th,0)
#print(mats)
#print(mats.shape)
#print(matt)
#print(matt.shape)

matf=np.concatenate((mats,matt),axis=0)
#print(matf)
print(matf.shape)

s=np.zeros((208),dtype=int)
e=np.ones((208),dtype=int)
etiq=np.concatenate((s,e))
print(etiq.shape)

# Split dataset into training set and test set
#semilla = 132,0,1
X_train, X_test, y_train, y_test = train_test_split(
matf, etiq, test_size=0.3,random_state=1)
# 70% training and 30% test

# Train Decision Tree Classifer
clf=DecisionTreeClassifier()

clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",
metrics.accuracy_score(y_test, y_pred))

# Model Precision: what percentage of positive tuples
are labeled as such?
```

```
print("Precision:",
metrics.precision_score(y_test, y_pred))

# Model Recall: what percentage of positive tuples
are labelled as such?
print("Recall:",metrics.recall_score(y_test, y_pred))
```

Clasificación mediante KNN

```
from sklearn.neighbors import KNeighborsClassifier
# Import KNN Classifier
from sklearn.model_selection import train_test_split
# Import train_test_split function
from sklearn import metrics
#Import scikit-learn metrics
module for accuracy calculation
from sklearn.model_selection import train_test_split
from sklearn import metrics
from matplotlib.pyplot import figure, plot
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import csv

# K-Fold Cross-Validation
from sklearn.model_selection import cross_validate
def cross_validation(model, _X, _y, _cv=10):
  '''Function to perform 10 Folds Cross-Validation
 Parameters
 ----------
model: Python Class, default=None
This is the machine learning algorithm
to be used for training.
_X: array
 This is the matrix of features.
_y: array
 This is the target variable.
_cv: int, default=5
 Determines the number of folds for cross-validation.
  Returns
```

```
    _______
    The function returns a dictionary containing
    the metrics 'accuracy', 'precision',
    'recall', 'f1' for both training set and validation set.
          '''
      _scoring = ['accuracy', 'precision', 'recall', 'f1']
      results = cross_validate(estimator=model,
                               X=_X,
                               y=_y,
                               cv=_cv,
                               scoring=_scoring,
                               return_train_score=True)

        return {"Training Accuracy scores":
        results['train_accuracy'],
        "Mean Training Accuracy":
        results['train_accuracy'].mean()*100,
        "Training Precision scores":
        results['train_precision'],
        "Mean Training Precision":
        results['train_precision'].mean(),
        "Training Recall scores":
        results['train_recall'],
        "Mean Training Recall":
        results['train_recall'].mean(),
        "Training F1 scores": results['train_f1'],
        "Mean Training F1 Score":
        results['train_f1'].mean(),
        "Validation Accuracy scores":
        results['test_accuracy'],
        "Mean Validation Accuracy":
        results['test_accuracy'].mean()*100,
        "Validation Precision scores":
        results['test_precision'],
        "Mean Validation Precision":
        results['test_precision'].mean(),
        "Validation Recall scores": results['test_recall'],
        "Mean Validation Recall":
        results['test_recall'].mean(),
        "Validation F1 scores": results['test_f1'],
```

```
    "Mean Validation F1 Score":
    results['test_f1'].mean()
    }

# Grouped Bar Chart for both training and validation data
def plot_result(x_label, y_label, plot_title,
train_data, val_data):
'''Function to plot a grouped bar chart showing
the training and validation
results of the ML model in each fold after
applying K-fold cross-validation.
        Parameters
        ----------
        x_label: str,
Name of the algorithm used for training
e.g 'Decision Tree'

        y_label: str,
Name of metric being visualized e.g 'Accuracy'
     plot_title: str,
This is the title of the plot e.g 'Accuracy Plot'

        train_result: list, array
This is the list containing either training precision,
     accuracy, or f1 score.

        val_result: list, array
This is the list containing either validation precision,
    accuracy, or f1 score.
        Returns
        -------
The function returns a Grouped Barchart showing
the training and validation result in each fold.
    '''
        # Set size of plot
        plt.figure(figsize=(12,6))
        labels = ["1st Fold", "2nd Fold", "3rd Fold",
        "4th Fold", "5th Fold","6th Fold", "7th Fold",
        "8th Fold", "9th Fold", "10th Fold"]
        X_axis = np.arange(len(labels))
```

```
        ax = plt.gca()
        plt.ylim(0.40000, 1)
        plt.bar(X_axis-0.2, train_data, 0.4,
        color='black', label='Training')
        plt.bar(X_axis+0.2, val_data, 0.4,
        color='gray', label='Validation')
        plt.title(plot_title, fontsize=30)
        plt.xticks(X_axis, labels)
        plt.xlabel(x_label, fontsize=14)
        plt.ylabel(y_label, fontsize=14)
        plt.legend()
        plt.grid(True)
        plt.savefig(plot_title+"_dtree.png")
        plt.show()

def matxchar(dfx,r):
    car=["media","energia","contraste",
    "homogeneidad","varianza","correlacion"]
    mat=np.zeros((208,256*6),dtype=float)
    for i in range(1-r,208-r):
        #diccionario=dict['media':None,'energia':
        None,'contraste':None,'homogeneidad'
        :None,'varianza':
        None,'correlacion':None]
        for k in range(0,5):
            tmp = dfx[car[k]][i].replace('[','')
            tmp = tmp.replace(']','')
            tmp = tmp.replace('/n',' ')
            #print(tmp)

            #floats_list = []
            a=0
            if(r==0):
                for item in tmp.split():
                    mat[i-1,(k*256)+a]=float(item)
                    a=a+1
            else:
                for item in tmp.split():
                    mat[i,(k*256)+a]=float(item)
                    a=a+1
```

```
    return mat

data_sh=pd.read_csv
('res_sano_cut_norm.csv',index_col=0)
data_th=pd.read_csv
('res_tobrfv_cut_norm.csv',index_col=0)
#print(data_th)

#print(type(data_sh["media"][0]))

mats=matxchar(data_sh,1)
matt=matxchar(data_th,0)
#print(mats)
#print(mats.shape)
#print(matt)
#print(matt.shape)

matf=np.concatenate((mats,matt),axis=0)
#print(matf)
print(matf.shape)

s=np.zeros((208),dtype=int)
e=np.ones((208),dtype=int)
etiq=np.concatenate((s,e))
print(etiq.shape)

#shuffle the data for classification problems
from sklearn.model_selection import StratifiedKFold
skf=StratifiedKFold(10,shuffle=True,random_state=1)
# Create KNN
clf = KNeighborsClassifier(n_neighbors=5)
clf_res=cross_validation(clf,matf,etiq,skf)
print(clf_res)

## Plot Accuracy Result
# model_name = "Decision Tree"
# plot_result(model_name,"Accuracy",
#                "Accuracy scores in 10 Folds",
#                clf_res["Training Accuracy scores"],
#                clf_res["Validation Accuracy scores"])
```

```
## Plot Precision Result
# plot_result(model_name,
#             "Precision",
#             "Precision scores in 10 Folds",
#             clf_res["Training Precision scores"],
#             clf_res["Validation Precision scores"])

## Plot Recall Result
# plot_result(model_name,
#             "Recall",
#             "Recall scores in 10 Folds",
#             clf_res["Training Recall scores"],
#             clf_res["Validation Recall scores"])

## Plot F1-Score Result
# plot_result(model_name,
#             "F1",
#             "F1 Scores in 10 Folds",
#             clf_res["Training F1 scores"],
#             clf_res["Validation F1 scores"])

# clf2 = DecisionTreeClassifier
(min_samples_split=10,random_state=1)
# clf_res2 = cross_validation(clf2, matf, etiq, 10)
# print(clf_res2)

## Plot Accuracy Result
# plot_result(model_name,
#             "Accuracy",
#             "Accuracy scores in 10 Folds",
#             clf_res2["Training Accuracy scores"],
#             clf_res2["Validation Accuracy scores"])

## Plot Precision Result
# plot_result(model_name,
#             "Precision",
#             "Precision scores in 10 Folds",
#             clf_res2["Training Precision scores"],
#             clf_res2["Validation Precision scores"])
```

```
## Plot Recall Result
# plot_result(model_name,
#                 "Recall",
#                 "Recall scores in 10 Folds",
#                 clf_res2["Training Recall scores"],
#                 clf_res2["Validation Recall scores"])

## Plot F1-Score Result
# plot_result(model_name,
#                 "F1",
#                 "F1 Scores in 10 Folds",
#                 clf_res2["Training F1 scores"],
#                 clf_res2["Validation F1 scores"])

# def matxchar(dfx,r):
#     car=["media","energia","contraste",
"homogeneidad","varianza","correlacion"]
#     mat=np.zeros((208,256*6),dtype=float)
#     for i in range(1-r,208-r):
#         #diccionario=dict['media':None,'energia':
None,'contraste':None,'homogeneidad':None,'varianza':
None,'correlacion':None]
#         for k in range(0,5):
#             tmp = dfx[car[k]][i].replace('[','')
#             tmp = tmp.replace(']','')
#             tmp = tmp.replace('/n',' ')
#             #print(tmp)

#             #floats_list = []
#             a=0
#             if(r==0):
#                 for item in tmp.split():
#                     mat[i-1,(k*256)+a]=float(item)
#                     a=a+1
#             else:
#                 for item in tmp.split():
#                     mat[i,(k*256)+a]=float(item)
#                     a=a+1
```

```
#      return mat

# data_sh=pd.read_csv
('res_sano_cut_norm.csv',index_col=0)
# data_th=pd.read_csv
('res_tobrfv_cut_norm.csv',index_col=0)
# #print(data_th)

# #print(type(data_sh["media"][0]))

# mats=matxchar(data_sh,1)
# matt=matxchar(data_th,0)
# #print(mats)
# #print(mats.shape)
# #print(matt)
# #print(matt.shape)

# matf=np.concatenate((mats,matt),axis=0)
# #print(matf)
# print(matf.shape)

# s=np.zeros((208),dtype=int)
# e=np.ones((208),dtype=int)
# etiq=np.concatenate((s,e))
# print(etiq.shape)

## Split dataset into training set and test set
#semilla = 132,1,0,4,13,94
# X_train, X_test, y_train, y_test =
train_test_split
(matf, etiq, test_size=0.3,random_state=132)
# 70% training and 30% test

## Train Decision Tree Classifer
# clf = KNeighborsClassifier(n_neighbors=5)

##Train the model using the training sets
# clf.fit(X_train, y_train)

##Predict the response for test dataset
```

```
# y_pred = clf.predict(X_test)

## Model Accuracy,
how often is the classifier correct?
# print("Accuracy:",metrics.
accuracy_score(y_test, y_pred))

## Model Precision:
what percentage of positive
tuples are labeled as such?
# print("Precision:",metrics.
precision_score(y_test, y_pred))

## Model Recall:
what percentage of positive
tuples are labelled as such?
# print("Recall:",metrics.
recall_score(y_test, y_pred))
```

Clasificación mediante Random Forest

```
from sklearn.neighbors import KNeighborsClassifier
# Import KNN Classifier
from sklearn.model_selection import train_test_split
# Import train_test_split function
from sklearn import metrics
#Import scikit-learn metrics module
for accuracy calculation
from sklearn.model_selection import train_test_split
from sklearn import metrics
from matplotlib.pyplot import figure, plot
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import csv

# K-Fold Cross-Validation
from sklearn.model_selection import cross_validate
def cross_validation(model, _X, _y, _cv=10):
'''Function to perform 10 Folds Cross-Validation
```

```
 Parameters
        ----------
model: Python Class, default=None
This is the machine learning algorithm to be used for trainin
      _X: array
This is the matrix of features.
      _y: array
This is the target variable.
      _cv: int, default=5
Determines the number of folds for cross-validation.
       Returns
        -------
The function returns a dictionary
containing the metrics 'accuracy', 'precision',
'recall', 'f1' for both training set and validation set.
       '''
_scoring = ['accuracy', 'precision', 'recall', 'f1']
results = cross_validate(estimator=model,
       X=_X,
       y=_y,
      cv=_cv,
       scoring=_scoring,
     return_train_score=True)

return {"Training Accuracy scores":
results['train_accuracy'],
"Mean Training Accuracy":
results['train_accuracy'].mean()*100,
"Training Precision scores":
results['train_precision'],
"Mean Training Precision":
results['train_precision'].mean(),
"Training Recall scores":
results['train_recall'],
"Mean Training Recall":
results['train_recall'].mean(),
"Training F1 scores":
results['train_f1'],
"Mean Training F1 Score":
results['train_f1'].mean(),
```

```
"Validation Accuracy scores":
results['test_accuracy'],
"Mean Validation Accuracy":
results['test_accuracy'].mean()*100,
"Validation Precision scores":
results['test_precision'],
"Mean Validation Precision":
results['test_precision'].mean(),
"Validation Recall scores":
results['test_recall'],
"Mean Validation Recall":
results['test_recall'].mean(),
"Validation F1 scores": results['test_f1'],
"Mean Validation F1 Score":
results['test_f1'].mean()
}

# Grouped Bar Chart for both training and
validation data
def plot_result(x_label, y_label, plot_title,
train_data, val_data):
'''Function to plot a grouped
bar chart showing the training and validation
results of the ML model in each
fold after applying K-fold cross-validation.
        Parameters
        ----------
        x_label: str,
    Name of the algorithm used for training
    e.g 'Decision Tree'

        y_label: str,
    Name of metric being visualized
    e.g 'Accuracy'
        plot_title: str,
    This is the title of the plot
    e.g 'Accuracy Plot'

        train_result: list, array
  This is the list containing either
```

```
    training precision, accuracy, or f1 score.

          val_result: list, array
      This is the list containing either
      validation precision, accuracy, or f1 score.
          Returns
          ---------
      The function returns a Grouped Barchart
      showing the training and validation result
      in each fold.
          '''

      # Set size of plot
      plt.figure(figsize=(12,6))
      labels = ["1st Fold", "2nd Fold", "3rd Fold",
      "4th Fold", "5th Fold","6th Fold", "7th Fold",
      "8th Fold", "9th Fold", "10th Fold"]
      X_axis = np.arange(len(labels))
      ax = plt.gca()
      plt.ylim(0.40000, 1)
      plt.bar(X_axis-0.2, train_data, 0.4,
      color='black', label='Training')
      plt.bar(X_axis+0.2, val_data, 0.4,
      color='gray', label='Validation')
      plt.title(plot_title, fontsize=30)
      plt.xticks(X_axis, labels)
      plt.xlabel(x_label, fontsize=14)
      plt.ylabel(y_label, fontsize=14)
      plt.legend()
      plt.grid(True)
      plt.savefig(plot_title+"_dtree.png")
      plt.show()

def matxchar(dfx,r):
    car=["media","energia","contraste","homogeneidad",
    "varianza","correlacion"]
    mat=np.zeros((208,256*6),dtype=float)
    for i in range(1-r,208-r):
      #diccionario=dict['media':None,'energia':
      None,'contraste':None,'homogeneidad':None,'varianza':
```

```
        None,'correlacion':None]
        for k in range(0,5):
            tmp = dfx[car[k]][i].replace('[','')
            tmp = tmp.replace(']','')
            tmp = tmp.replace('/n',' ')
            #print(tmp)

            #floats_list = []
            a=0
            if(r==0):
                for item in tmp.split():
                    mat[i-1,(k*256)+a]=float(item)
                    a=a+1
            else:
                for item in tmp.split():
                    mat[i,(k*256)+a]=float(item)
                    a=a+1

    return mat

data_sh=pd.read_csv
('res_sano_cut_norm.csv',index_col=0)
data_th=pd.read_csv
('res_tobrfv_cut_norm.csv',index_col=0)
#print(data_th)

#print(type(data_sh["media"][0]))

mats=matxchar(data_sh,1)
matt=matxchar(data_th,0)
#print(mats)
#print(mats.shape)
#print(matt)
#print(matt.shape)

matf=np.concatenate((mats,matt),axis=0)
#print(matf)
print(matf.shape)

s=np.zeros((208),dtype=int)
```

```
e=np.ones((208),dtype=int)
etiq=np.concatenate((s,e))
print(etiq.shape)

#shuffle the data for classification problems
from sklearn.model_selection import StratifiedKFold
skf=StratifiedKFold(10,shuffle=True,random_state=1)
# Create KNN
clf = KNeighborsClassifier(n_neighbors=5)
clf_res=cross_validation(clf,matf,etiq,skf)
print(clf_res)

## Plot Accuracy Result
# model_name = "Decision Tree"
# plot_result(model_name,"Accuracy",
#                "Accuracy scores in 10 Folds",
#                clf_res["Training Accuracy scores"],
#                clf_res["Validation Accuracy scores"])

## Plot Precision Result
# plot_result(model_name,
#                "Precision",
#                "Precision scores in 10 Folds",
#                clf_res["Training Precision scores"],
#                clf_res["Validation Precision scores"])

## Plot Recall Result
# plot_result(model_name,
#                "Recall",
#                "Recall scores in 10 Folds",
#                clf_res["Training Recall scores"],
#                clf_res["Validation Recall scores"])

## Plot F1-Score Result
# plot_result(model_name,
#                "F1",
#                "F1 Scores in 10 Folds",
#                clf_res["Training F1 scores"],
#                clf_res["Validation F1 scores"])
```

```
# clf2 = DecisionTreeClassifier
(min_samples_split=10,random_state=1)
# clf_res2 = cross_validation(clf2, matf, etiq, 10)
# print(clf_res2)

## Plot Accuracy Result
# plot_result(model_name,
#             "Accuracy",
#             "Accuracy scores in 10 Folds",
#             clf_res2["Training Accuracy scores"],
#             clf_res2["Validation Accuracy scores"])

## Plot Precision Result
# plot_result(model_name,
#             "Precision",
#             "Precision scores in 10 Folds",
#             clf_res2["Training Precision scores"],
#             clf_res2["Validation Precision scores"])

## Plot Recall Result
# plot_result(model_name,
#             "Recall",
#             "Recall scores in 10 Folds",
#             clf_res2["Training Recall scores"],
#             clf_res2["Validation Recall scores"])

## Plot F1-Score Result
# plot_result(model_name,
#             "F1",
#             "F1 Scores in 10 Folds",
#             clf_res2["Training F1 scores"],
#             clf_res2["Validation F1 scores"])

# def matxchar(dfx,r):
#     car=["media","energia","contraste","homogeneidad",
"varianza","correlacion"]
#     mat=np.zeros((208,256*6),dtype=float)
#     for i in range(1-r,208-r):
#         #diccionario=dict['media':None,'energia':None,
'contraste':None,'homogeneidad':None,'varianza':
```

```
None,'correlacion':None]
#           for k in range(0,5):
#                tmp = dfx[car[k]][i].replace('[','')
#                tmp = tmp.replace(']','')
#                tmp = tmp.replace('/n',' ')
#                #print(tmp)

#                #floats_list = []
#                a=0
#                if(r==0):
#                    for item in tmp.split():
#                        mat[i-1,(k*256)+a]=float(item)
#                        a=a+1
#                else:
#                    for item in tmp.split():
#                        mat[i,(k*256)+a]=float(item)
#                        a=a+1
#      return mat

# data_sh=pd.read_csv
('res_sano_cut_norm.csv',index_col=0)
# data_th=pd.read_csv
('res_tobrfv_cut_norm.csv',index_col=0)
# #print(data_th)

# #print(type(data_sh["media"][0]))

# mats=matxchar(data_sh,1)
# matt=matxchar(data_th,0)
# #print(mats)
# #print(mats.shape)
# #print(matt)
# #print(matt.shape)

# matf=np.concatenate((mats,matt),axis=0)
# #print(matf)
# print(matf.shape)

# s=np.zeros((208),dtype=int)
# e=np.ones((208),dtype=int)
```

```
# etiq=np.concatenate((s,e))
# print(etiq.shape)

## Split dataset into training set and test set
#semilla = 132,1,0,4,13,94
# X_train, X_test, y_train, y_test =
train_test_split(matf, etiq,
test_size=0.3,random_state=132)
# 70% training and 30% test

## Train Decision Tree Classifer
# clf = KNeighborsClassifier(n_neighbors=5)

##Train the model using the training sets
# clf.fit(X_train, y_train)

##Predict the response for test dataset
# y_pred = clf.predict(X_test)

## Model Accuracy,
how often is the classifier correct?
# print("Accuracy:",
metrics.accuracy_score(y_test, y_pred))

## Model Precision: what percentage of
positive tuples are labeled as such?
# print("Precision:",metrics.
precision_score(y_test, y_pred))

## Model Recall: what percentage of
positive tuples are labelled as such?
# print("Recall:",metrics.
recall_score(y_test, y_pred))
```