TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CHILPANCINGO

LGAC 2: INVESTIGACIÓN, DESARROLLO Y APLICACIONES DE TECNOLOGÍAS INTELIGENTES

MODELO CLASIFICADOR DE RESIDUOS SÓLIDOS URBANOS (RSU) DEL TIPO MADERA Y TEXTIL MEDIANTE INTELIGENCIA ARTIFICIAL Y MACHINE LEARNING

TESIS

PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS DE LA INGENIERÍA

PRESENTA:

Jonathan Jesús Carranza Vega

Director

Dr. Wilfrido Campos Francisco

INSTITUTO TECNOLÓGICO DE CHILPANCINGO

Codirector

Dr. Jonathan Villanueva Tavira

CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO

Revisor 1

Dra. Josefa Morales Morales

INSTITUTO TECNOLÓGICO DE CHILPANCINGO

Revisor 2

Dra. Yanet Evangelista Alcocer

INSTITUTO TECNOLÓGICO DE CHILPANCINGO

Chilpancingo de los Bravo, Guerrero, junio 2026

Agradecimientos

En primera instancia, expreso mis más sinceros agradecimientos a mi familia:

A mi esposa, quien con su amor, comprensión y respaldo incondicional está a mi lado en los momentos buenos y malos. A mis hijos, quienes en lugar de reclamar el tiempo que les resté para dedicarlo al estudio, se convirtieron en el impulso para concluir esta meta. A mi madre, quien con su amor infinito me ha alentado, motivado y apoyado en todo lo necesario para cumplir mis sueños desde que tengo uso de razón. Y a mis hermanos, que de una u otra manera siempre están presentes; aun sin llamarlos, me fortalecen con sus palabras, consejos o con el simple hecho de estar, recordándome que nunca estoy solo.

En segundo lugar, agradezco a mi institución laboral, la Subsecretaría de Planeación Educativa, por permitirme adaptar mis tiempos para cumplir con mis deberes. De manera especial, a la Dirección de Evaluación, a su directora y a mis compañeros de área, quienes siempre me brindaron su comprensión y ánimo.

En tercer lugar, al cuerpo académico del posgrado, a mi director, codirector y revisores de tesis. Gracias por su guía, retroalimentación, paciencia y por la confianza que depositaron en mí para culminar esta aventura. Asimismo, agradezco a los profesores que creyeron en mí; sus consejos, clases y charlas son, sin duda, parte fundamental de este logro.

Por último, pero no menos importante, a mis compañeros de maestría. Compartimos momentos de estrés, alegría y estudio, buscando siempre el apoyo mutuo. En cada uno de ellos encontré siempre una disposición de ayudar acompañada de una sonrisa.

El éxito de esta meta es, ante todo, el resultado de tener fe. No dudo que Dios unió el esfuerzo y el cariño de todas las personas aquí mencionadas para hacer este sueño realidad. Mi más sincero agradecimiento a todos.

Educación
Secretaría de Educación Pública

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Chilpancingo
DiEPI

Chilpancingo, Guerrero 12/junio/2026
Oficio No. 12DIT0002E/DEPI/047/2026

JONATHAN JESÚS CARRANZA VEGA
CANDIDATO A MAESTRO EN CIENCIAS DE LA INGENIERÍA

P R E S E N T E

Por este conducto me es grato comunicarle que el Comité Tutorial asignado a su trabajo de tesis titulada **"Modelo clasificador de Residuos Sólidos Urbanos (RSU) del tipo Madera y Textil mediante Inteligencia Artificial y Machine Learning"**, ha informado a esta División de Estudio de Posgrado e Investigación (DEPI) que están de acuerdo con el trabajo presentado. Por lo anterior, se le autoriza a que proceda con la impresión definitiva de su trabajo de tesis.

Sin otro particular, le envío un cordial saludo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
Crear Tecnología es Forjar Libertad

MIRNA CASTRO BELLO
JEFA DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE CHILPANCINGO
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

ccp. Subdireción Académica.
ccp. Archivo
MCB*rlgr.

2026 año de Margarita Maza

Av. José Francisco Ruiz Massieu No. 5, Colonia Villa Moderna, Chilpancingo de los Bravo, Guerrero. México.
Tel. (747) 45 4 1300, Ext. 1340, email: ssa@chilpancingo.tecnm.mx

Educación
Secretaría de Educación Pública

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Chilpancingo
DEPI

Chilpancingo, Guerrero 05/Junio/2026
Oficio No. 045/2026

DRA. MIRNA CASTRO BELLO
JEFA DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO
E INVESTIGACIÓN DEL INSTITUTO TECNOLÓGICO
DE CHILPANCINGO

PRESENTE

Por este medio los integrantes del comité Tutorial de la **C. Jonathan Jesús Carranza Vega** con número de control **G98520373** de la **Maestría en Ciencias de la Ingeniería**, le informamos que hemos revisado el trabajo de la tesis de grado titulado **"Modelo clasificador de Residuos Sólidos Urbanos (RSU) del tipo Madera y Textil mediante Inteligencia Artificial y Machine Learning"**, una vez que se han atendido todas las observaciones que se indicaron, hemos acordado aceptar el documento por lo que solicitamos la autorización de impresión definitiva.

Sin más por el momento, le enviamos un cordial saludo.

Director	Codirector
Dr. Wilfrido Campos Francisco	Jonathan Villanueva Tavira Dr. Jonathan Villanueva Tavira

Revisor 1	Revisor 2
Dra. Josefa Morales Morales	Dra. Yanet Evangelista Alcocer

2026 año de Margarita Maza

Av. José Francisco Ruiz Massieu No. 5, Colonia Villa Moderna, Chilpancingo de los Bravo, Guerrero, México.
Tel. (747) 65 4 1300, Ext. 1340, email: ext@chilpancingo.tecnm.mx
http://chilpancingo.tecnm.mx/ https://www.facebook.com/TecNMcampusChilpancingo

Contenido

Índice de tablas

Índice de figuras

Resumen

En varias ciudades del mundo los Residuos Sólidos Urbanos (RSU) son un problema en crecimiento, se estima que aumente a 3,400 millones de toneladas para el año 2050. En México se generan diariamente en promedio 108,146 toneladas de RSU. Los sistemas de Inteligencia Artificial (IA) se han fortalecido como herramientas informáticas para agilizar los procesos de gestión. No obstante, la mayoría de estos se centran principalmente en la clasificación de residuos como papel, plástico, vidrio y metal, por lo que, los desechos de madera y textil han recibido poca atención. Este trabajo propone un análisis comparativo sistemático de modelos de aprendizaje automático bajo condiciones controladas que evalúa la capacidad discriminativa de descriptores de textura clásicos aplicados a las categorías de RSU de madera y textil. El procedimiento consistió en la extracción de características de textura de un conjunto de 4,396 imágenes de residuos de madera y textil utilizando técnicas Local Binary Pattern (LBP), Gray-Level Co-occurrence Matrix (GLCM), Histogram of Oriented Gradients (HOG), Canny/Sobel edge detection y Fractal Dimension (FD). Al integrar todos estos descriptores se implementó el algoritmo Random Forest Importance para identificar las variables más discriminantes. Finalmente, el subconjunto óptimo se utilizó para entrenar tres modelos de Machine Learning (ML). Los resultados demostraron que Multilayer Perceptron (MLP) obtuvo el mejor desempeño con una precisión del 96.70 %, seguido por Random Forest (RF) con 95.45 % y Support Vector Machine (SVM) con 95.22 %. Esto confirma que al combinar técnicas clásicas de análisis de textura con modelos de ML logra una clasificación binaria de alta precisión para textiles y madera bajo condiciones controladas. La implementación de estas comparaciones servirá como base para el desarrollo de nuevas herramientas tecnológicas con bajo costo computacional que lleven a cabo una adecuada separación de residuos.

Palabras clave: Residuos Sólidos Urbanos, Machine Learning, Residuos de madera, Residuos de textil, Características de textura.

Abstract

In several cities around the world, Municipal Solid Waste (MSW) is a growing problem; it is estimated to increase to 3.4 billion tons by the year 2050. In Mexico, an average of 108,146 tons of MSW are generated daily. Artificial Intelligence (AI) systems have strengthened as computing tools to streamline management processes. However, most of these focus primarily on sorting waste such as paper, plastic, glass, and metal, leaving wood and textile waste with little attention. This work proposes a systematic comparative analysis of machine learning models under controlled conditions that evaluates the discriminative capacity of classic texture descriptors applied to the wood and textile MSW categories. The procedure consisted of extracting texture features from a dataset of 4396 images of wood and textile waste using Local Binary Pattern (LBP), Gray-Level Co-occurrence Matrix (GLCM), Histogram of Oriented Gradients (HOG), Canny/Sobel edge detection, and Fractal Dimension (FD) techniques. By integrating all these descriptors, the Random Forest Importance algorithm was implemented to identify the most discriminative variables. Finally, the optimal subset was used to train three Machine Learning (ML) models. The results demonstrated that the Multilayer Perceptron (MLP) achieved the best performance with an accuracy of 96.70 %, followed by Random Forest (RF) with 95.45 %, and Support Vector Machine (SVM) with 95.22 %. This confirms that combining classic texture analysis techniques with ML models achieves a high-precision binary classification for textiles and wood under controlled conditions. The implementation of these comparisons will serve as a foundation for developing new technological tools with low computational cost to carry out proper waste separation.

Keywords: Municipal Solid Waste, Machine Learning, Wood Waste, textile waste, texture feature.

I. Introducción

Los Residuos Sólidos Urbanos (RSU) son un problema para todas las ciudades del mundo. El reto para las administraciones públicas de gobierno es realizar una gestión que permita disminuir las consecuencias de los RSU en agua, aire, tierra para mitigar sus efectos de los contaminantes en la fauna y flora, se espera que los residuos aumenten en el mundo a 3,400 millones de toneladas para el año 2050. Los residuos de alimentos representan el 44 %, los residuos reciclables secos como el plástico, papel, cartón, metal y vidrio representan otro 38 % (Kaza et al., 2018). En este sentido, el uso eficiente de depósitos de desechos y los restos vinculados con el reciclaje, la valorización energética y las tecnologías de tratamiento al final de la vida útil de los residuos han adquirido una gran importancia (Salem et al., 2023). En el año 2022 en México, la cantidad promedio diaria de RSU recolectados fue de 108,146 toneladas, de las cuales 66.7 % se realizó mediante el sistema de recolección en los hogares, el 24.9 %, en un punto de recolección establecido y el 8.4 % mediante sistema de contenedores (INEGI, 2025). Nuestro país ha mantenido su participación activa en la implementación de la Agenda 2030, presentando avances sobre los Objetivos de Desarrollo Sostenible (ODS) ante el Foro Político de Alto Nivel en Desarrollo Sostenible (ONU México, 2025).

La Secretaría de Medio Ambiente y Recursos Naturales (SEMARNAT) establece un sistema de separación del reciclaje, lo organiza en una clasificación primaria, que contempla los residuos orgánicos e inorgánicos, y una secundaria, que agrupa las fracciones con mayor potencial de aprovechamiento, como papel, plástico, metal, vidrio, madera y tela (SEMARNAT, 2017). Los principios fundamentales de una economía circular son evitar la generación de residuos, mejorar la recuperación de recursos, aumentar el uso de elementos reciclados, gestionar mejor el flujo de materiales en beneficio del medio ambiente, economía y sociedad. A pesar de que la madera se utiliza abundantemente en todo el mundo, su potencial para integrarse en la economía circular ha recibido menos atención (Jahan et al., 2022). En lo que se refiere a textiles, la gestión circular requiere la creación de ciclos seguros de producción que generen menos residuos, fomentando la reutilización y el reciclaje, evitando la incineración o el vertido de residuos (Vercalsteren et

al., 2019). La SEMARNAT, a través de su Diagnóstico Básico para la Gestión Integral de los Residuos, estima que el porcentaje de la madera es de 0.79 % y textil (trapo) 2.82 % a nivel nacional (SEMARNAT, 2020).

Los sistemas de Inteligencia Artificial (IA) para la detección y el reconocimiento automático de desechos a partir de imágenes para reemplazar la clasificación manual, se está volviendo cada vez más posible (Liang y Gu, 2021), permitiendo desarrollar herramientas para clasificar RSU y contribuir al reciclaje (Ramos et al., 2024). Las Redes Neuronales Convolucionales (CNN), son modelos de Deep Learning (DL) eficientes para apoyar a procesar de manera automática grandes cantidades de imágenes y se están utilizando para la clasificación de residuos. Sin embargo, entre sus desventajas se encuentran que necesitan importantes recursos computacionales, mucho tiempo de entrenamiento y grandes cantidades de datos etiquetados; de lo contrario, el modelo puede tener un mal funcionamiento por el sobreajuste (Zhou et al., 2024).

La base de un buen modelo de IA son los conjuntos de datos de alta calidad, los cuales permite adaptar algoritmos a las características específicas de los residuos. En este sentido, los conjuntos de datos grandes ofrecen una cobertura amplia; los de escala media suelen equilibrar una cobertura integral con áreas de enfoque específicas, mientras que los pequeños tienden a destacar en aplicaciones especializadas o servir como referencias bien documentadas. Esto ofrece diversas opciones para entrenar y evaluar la clasificación de residuos. Una de las limitaciones de los conjuntos de datos es la escalabilidad y el mantenimiento, debido a la naturaleza dinámica de la producción de residuos y la evolución de los materiales, lo que obliga a actualizar y ampliar de manera periódicamente los conjuntos de datos. Abordar estos desafíos no solo mejora la precisión y la generalización de los algoritmos de clasificación, sino que también fortalece las bases para las técnicas de modelado impulsadas por IA (Fotovvatikhah et al., 2025). Para gestionar esta situación, se proponen enfoques para seleccionar la mejor combinación de descriptores que, en conjunto, proporcionen una mejor clasificación. Incluso si, se debe calcular más de un descriptor, este método sigue siendo lo suficientemente rápido para aplicaciones en tiempo real. El conjunto de descriptores resultante siempre mejora la calidad de la clasificación en comparación con el

mejor descriptor por sí solo; estas técnicas de selección pueden emplearse en diferentes conjuntos de datos y contextos (Merino et al., 2020).

Los algoritmos Machine Learning (ML) están diseñados para extraer información que se utilizan en tareas como regresión o clasificación. El flujo de trabajo de estos modelos consta del procesamiento de datos, extracción y selección de características (Dao et al., 2025). El ML puede ser más eficaz que las herramientas estadísticas tradicionales para manejar diversos formatos de datos como texto, imágenes y gráficos, al utilizar combinaciones de características desconocidas para determinar un resultado (Zhong et al., 2021). Existen estudios que han utilizado Redes Neuronales Artificiales (RNA) para la separación de materiales de desecho de manera automática (Abdallah et al., 2020). Se han desarrollado cestos de basura inteligentes que integran un sistema automático de clasificación; sin embargo, la mayoría clasifica cantidades limitadas y se centran mayormente en papel, plástico, vidrio, metal y otros (Zoumpoulis et al., 2024), dejando de lado los RSU de madera y textil.

El presente trabajo propone un análisis comparativo sistemático bajo condiciones controladas con modelos de ML, que evalúa la capacidad discriminativa de descriptores de textura clásicos aplicados en las categorías de RSU de madera y textil. Lo anterior permitirá establecer la base metodológica para el desarrollo de nuevas herramientas que abonará a la eficiencia y precisión para una mejor separación de estos residuos.

II. Antecedentes

Los avances en IA y visión artificial, generan la posibilidad de reemplazar la visión humana para la clasificación, detección y reconocimiento de objetos por sistemas automatizados. Las CNN han demostrado ser eficientes para este tipo de tareas, sin embargo, obligan a ocupar infraestructuras computacionales robustas y grandes volúmenes de imágenes. En este sentido, el ML apoyado con la extracción de descriptores de textura clásicos y algoritmos de clasificación supervisada, ofrece una alternativa para recursos computacionales limitados. Existen varias investigaciones que abordan diversos enfoques para la clasificación de residuos, los métodos basados en ML, DL y los híbridos que combinan ambos, de igual manera estos sistemas de visión artificial se integran a hardware especializado. A continuación, se presentan una serie de investigaciones en varias partes de mundo con respecto a la clasificación de RSU mediante IA.

Por mencionar algunos casos, en Indonesia proponen de manera inicial un sistema que integra las técnicas clásicas de extracción de características GLCM y LBP para la detección de objetos en 2D. Con los valores estadísticos extraídos de 4,437 imágenes en 2D se obtuvo un vector de características y se clasifican con algoritmos de ML (K-Nearest Neighbors y RF). Al combinar las características de textura se llegó a una precisión del 93.9 % y una puntuación F1 del 93.8 %, lo que demuestra la eficacia de la técnica de conjunto para aumentar la precisión en la detección de objetos (Kurniati et al., 2024). Por otro lado, en Turquía para la predicción de parámetros de tejido en la industria textil, se aplicaron diversos métodos de extracción de textura como GLCM, Gabor Filter Bank, LBP y Intertwined Frame Vector a 130 imágenes de tejidos capturadas con un microscopio digital portátil con una resolución de 640×480 píxeles, con algoritmos de ML como RF, XGBoost y MLP; en este caso XGBoost alcanzó la precisión de 98.7 % (Seckin et al., 2023).

Rahman et al. (2025) desarrollaron un modelo donde, a partir de un conjunto de datos de imágenes de astillas de madera con rango de niveles de humedad en tres categorías seco, medio y húmedo, se extrajeron cinco tipos de características de textura: Haralick, First-Order Statistics, Fast Fourier Power Spectrum, Gray Level Run Length Matrix y LBP. El modelo desarrollado AdaptMoist generaliza datos sin etiquetas, y obtuvo una precisión promedio del 80 %.

Asimismo, en Sudáfrica se emplearon simulaciones para pronosticar la generación de RSU con datos históricos de variables como población, empleo formal, desempleo, número de unidades familiares. Se entrenó una RNA con cinco neuronas diferentes (5, 10, 20, 30 y 40) y seis variantes de algoritmos SVM (lineal, cuadrático, cúbico, gaussiano fino, gaussiano medio y gaussiano grueso). Los resultados demuestran que la RNA con 10 neuronas obtuvo la predicción de 99.9 %, superando el modelo SVM lineal de 98.5 % (Ayeleru et al., 2021).

Por otra parte, en India se propone un modelo automatizado para identificar 12 especies de madera a partir de un conjunto de datos WOOD-AUTH, que contiene 4,272 imágenes macroscópicas de secciones transversales, radiales y tangenciales. El análisis integró dos métodos de textura: GLCM para extraer características estadísticas de relación espacial entre píxeles, y LBP, para evaluar diferencias entre un píxel central y sus vecinos. Con el Análisis de Componentes Principales (PCA) se descartaron datos redundantes, se utilizó el algoritmo de ML K-Vecinos más Cercanos (KNN), logrando clasificar las especies con una precisión del 64 % (Sharafudeen et al., 2021).

Del mismo modo, también en India se desarrolló un sistema de gestión de residuos inteligente para áreas residenciales que combina el Internet de las Cosas (IoT) y ML. Se equiparon contenedores de basura con un sistema Raspberry Pi y diversos sensores ultrasónicos para medir constantemente el nivel de capacidad general del basurero y el nivel de residuos metálicos acumulados. Además, utilizan un sensor MQ4 para medir la concentración de gases venenosos como el metano, los cuales se generan por la descomposición de residuos biodegradables. Esta tecnología permite enviar mensajes si los contenedores alcanzan o superan el 70 % de su capacidad, o si las emisiones de gas venenoso superan la marca de 300 ppm (0.003), lo que indica la necesidad de una limpieza urgente. Utilizaron varios modelos de ML para evaluar la clasificación automática como: SVM, Naive Bayes (NB), K-Nearest Neighbors (KNN), Árboles de Decisión (DT), Regresión Logística (LR) y RF. El mejor resultado lo obtuvo RF con una precisión de 85.29 % (Dubey et al., 2020).

Actualmente el DL mediante el uso de las CNN se emplean debido a su capacidad

para extraer automáticamente las características de las imágenes de RSU. En Iraq se utilizó la arquitectura VGG16 para abordar la complejidad de gestión de residuos orgánicos y reciclables; este modelo logró una precisión del 96 % para orgánicos y del 88 % reciclables; con las capacidades del modelo se alcanzó una tasa de reconocimiento del 97 % para las imágenes etiquetadas (Shafek et al., 2025). Bajo un enfoque similar, en Bangladesh presentan un sistema automatizado de clasificación de residuos en tres etapas, diseñado para mejorar la eficiencia del reciclaje y promover la sostenibilidad ambiental, especialmente en entornos con recursos limitados. Este modelo propone una red neuronal convolucional separable en profundidad ligero y paralelo (DP-CNN), combinada con un clasificador de aprendizaje extremo basado en conjuntos (En-ELM). Consiste en clasificar un conjunto de datos de 35,264 imágenes de residuos en tres etapas: la primera etapa separa los residuos en dos categorías (biodegradables y no biodegradables), logrando una precisión del 96 %. En la segunda subdivide la basura según sus características generales (por ejemplo: vidrio, metal, polímeros, residuos médicos, residuos electrónicos, etc.). En esta fase, el sistema alcanzó una precisión del 91 %. Por último, asigna cada imagen a una de las 36 clases granulares específicas (como botellas de plástico, jeringas, teléfonos inteligentes, etc.), logrando una precisión del 85.25 %. Además, se emplearon técnicas de XAI (como mapas de calor Grad-CAM y SHAP), las cuales son herramientas que apoyan a visualizar gráficamente las áreas exactas de la imagen (Nahiduzzaman et al., 2025).

En Turquía, Kaya (2023) propuso los modelos llamados Xception_CutLayer e InceptionResNetV2_CutLayer, basados en técnicas de aprendizaje por transferencia. Estos modelos consisten en la aplicación de un “corte de bloques” (block cutting) a las arquitecturas originales de los modelos DL: Xception e InceptionResNetV2, respectivamente. Dicha modificación logró transformar las dos arquitecturas clásicas en modelos más simples, reduciendo significativamente la cantidad de parámetros, la complejidad y el tiempo de ejecución. La validación se realizó utilizando el conjunto de datos Trashnet, que incluye seis tipos de residuos (cartón, vidrio, metal, papel, plástico y otros desechos); el modelo Xception_CutLayer obtuvo la tasa de éxito de clasificación más alta con un 89.72 %, seguido por el modelo

InceptionResNetV2_CutLayer con un 85.77 %. Por su parte, en México Castro-Bello et al. (2025a), mediante la red neuronal R3sNet, la cual, ocupa una combinación de técnicas de procesamiento y arquitectura optimizada con una precisión del 87 % para residuos orgánicos y del 94 % para residuos inorgánicos. R3sNet supera al modelo preentrenado ResNet50 hasta en un 6 % en la clasificación RSU.

Existen enfoques híbridos que se utilizan para la clasificación, en este sentido, en Viet Nam al combinar DL a través del modelo MobileNetV3 para la extracción de características; con apoyo del algoritmo de optimización Harris Hawk Optimization para eliminar las características irrelevantes y ML para clasificarla, se comparó el rendimiento de RF, SVM y Regresión Logística. SVM alcanzó la mejor precisión de 96.74 % (Dao et al., 2025).

En Emiratos Árabes Unidos, se presentó un sistema híbrido que integra la extracción de características con DL mediante una red Xception preentrenada y con algoritmos de ML (SVM, kNN, Bagged Trees, LDA y regresión logística) para clasificar escombros de construcción y demolición. Se recopiló un conjunto de datos integrado por 1,800 imágenes equilibradas y de alta calidad de cuatro categorías (cerámica/baldosas, hormigón, basura/residuos y madera). Los resultados demuestran que los sistemas híbridos obtuvieron una precisión de hasta el 99,45 %, superando a algunos enfoques de DL (Alashram et al., 2025). Con técnicas similares en Indonesia, se evaluó el rendimiento de los modelos SVM con núcleos Radial Basis Function (RBF) y Polinomial para la clasificación de residuos, utilizando SqueezeNet una CNN ligera para la extracción de características. Se preprocesaron y ampliaron conjuntos de datos de Kaggle de residuos orgánicos y reciclables para mejorar el entrenamiento del modelo. Los resultados muestran que el modelo SVM con núcleo RBF obtuvo el mejor rendimiento con una precisión del 97.9 % frente al 97.3 % del núcleo polinomial. Este hallazgo subraya la importancia de la selección del núcleo y el ajuste de parámetros para optimizar los modelos SVM en tareas de clasificación no lineal (Yenuargo et al., 2024).

En la misma línea, en Indonesia presentan un sistema clasificador de residuos que combina la extracción de características de un modelo EfficientNet preentrenado con el Análisis

de Componentes Principales (ACP) para reducir la dimensionalidad. Lo hacen en dos etapas para extraer, equilibrar y procesar la información visual de los residuos, en la primera etapa se utiliza la arquitectura EfficientNet-CNN para obtener las características, para ello modifica la arquitectura del modelo para extraer información de patrones generales y locales, así como estructuras específicas de las imágenes. El segundo paso, es reducir la dimensionalidad de las características mediante ACP para crear una vector de características único y equilibrado. El sistema propuesto alcanzó una notable precisión del 99.07 % (Santoso et al., 2024).

En Vietnam, se desarrolló una comparación exhaustiva de tres paradigmas: algoritmos de ML que utilizan características diseñadas manualmente; arquitecturas de DL, incluyendo variantes de ResNet y EfficientNetV2S; y un enfoque híbrido que utiliza modelos de DL para la extracción de características combinados con clasificadores clásicos. Los experimentos realizados en tres conjuntos de datos públicos demuestran que el método híbrido supera sistemáticamente a los demás, alcanzando hasta un 100 % de precisión en TrashNet y el conjunto refinado de basura doméstica, y un 99,87 % en Clasificación de Basura. Además, la selección de características reduce la dimensionalidad de las mismas en más del 95 % sin comprometer la precisión, lo que resulta en un entrenamiento e inferencia más rápidos (Nguyen et al., 2025).

En India abordan la complejidad de los fondos en las imágenes de residuos médicos, se presenta EnSegNet-CFE-DNN-TC (Enhanced Segmentation Network with Combined Feature Extraction and Deep Neural Network Transfer Classifier). Este enfoque híbrido integra el DL con técnicas avanzadas de extracción de características de textura. Primero las imágenes de residuos (desechos infecciosos, químicos, cortopunzantes, farmacéuticos y patológicos) se segmentan utilizando la arquitectura para aislar mejor la información relevante, después se extraen múltiples características de textura con cuatro técnicas clásicas: la GLCM, el Patrón binario local multinivel (MLBP), el Patrón derivativo local (LDP) y el Patrón ternario local (LTP). Para finalizar, se introduce una nueva capa de combinación para fusionar las características extraídas y construir un nuevo vector híbrido. Los resultados demuestran que el sistema EnSegNet-CFE-DNN-TC alcanzó una precisión de clasificación del 93,7 % para 100 imágenes de basura (Mythili y Anbarasi, 2022).

En China Yang et al. (2025) con un enfoque de clasificación de basura que emplea una red neuronal de grafos dinámica basada en la fusión de características multimodales. Con el objetivo de integrar la información visual profunda, extraída por redes neuronales, con datos estadísticos globales sobre el color de la imagen. El modelo extrae características semánticas profundas mediante un Módulo de Atención de Red Residual (RNAM), y emplea el histograma de color CIELAB (LAB) capturar la distribución global del color de la imagen. Con un algoritmo adaptativo de KNN para construir la estructura gráfica dinámica. Se obtuvo una precisión de 99.33 % en la clasificación de basura de cocina, basura reciclable, basura peligrosa y otro tipo de basura.

Al integrar un modelo de IA con hardware especializado, se ha dado lugar a sistemas inteligentes con la finalidad de que funcionen en entornos reales. En este ámbito, en Japón se desarrolló un robot de clasificación automática para el reciclaje de envases de bebidas (plástico, latas, y vidrio transparente y de color). La detección de objetos se realizó a través de YOLO (You Only Look Once) en su versión siete, logrando una precisión media promedio (mAP) de 99.67 %. Mediante Segment Anything Meta AI (SAM) se generaron máscaras de segmentación para calcular el punto exacto de agarre del robot y para ayudar a la red neuronal en la correcta lectura de los colores del vidrio. Para validar su funcionamiento se introdujeron 495 envases en la cinta transportadora, el robot alcanzó una velocidad máxima de 45 recolecciones por minuto (2.4 m³/H) con una tasa de clasificación correcta del 93 % sobre los artículos que logró agarrar exitosamente (Cheng et al., 2024).

Sumando a lo anterior, en España se diseñó e implemento un prototipo de contenedor inteligente que clasifica automáticamente los residuos mediante un sistema multiagente e integración de blockchain. El sistema cuenta con sensores que identifican residuos orgánicos, plásticos, papel, etc. y con la plataforma PANGEA que divide el proceso de reciclaje en módulos inteligentes y autónomos que colaboran en tiempo real para tomar decisiones de clasificación instantáneas. Blockchain se implementó como tecnología para el control del registro de residuos, favoreciendo la trazabilidad durante el proceso de clasificación. Se evaluó su rendimiento en la detección y clasificación con las versiones de YOLO 8, 11, 12. YOLOv12 obtuvo los valores de

mAP@50 de 86,2 %, una precisión general de 84,6 % y una puntuación F1 promedio de 80,1 % (González et al., 2025).

Bajo un enfoque similar en México al comparar la precisión de cuatro modelos de CNN de la familia YOLO mediante su rendimiento en términos de velocidad de inferencia y uso de recursos en un sistema embebido instalado en una Raspberry Pi para la clasificación de residuos orgánicos e inorgánicos, los resultados fueron: YOLOv4-tiny, 91,71 %; YOLOv7-tiny, 91,34 %; YOLOv8-nano, 93 %; y YOLOv9-tiny, 92 % (Castro-Bello et al., 2025b). Aunado a esto en Colombia, Echeverry y López (2024) proponen un sistema de clasificación automatizada basado en visión artificial que reconoce y separa materiales como: plástico, vidrio, cartón y metal a través de una cámara web conectada en tiempo real a la placa de programación Nvidia Jetson Nano de 2 GB, la cual cuenta con una CNN entrenada con la arquitectura SSD-Mobilenet-v2 para la correcta clasificación de los residuos. El sistema tuvo una precisión del 95 % en la separación de plástico, 96 % en vidrio y metal, y 94 % en cartón.

Además, en Eslovenia se presenta un modelo conceptual para un sistema inteligente de separación de residuos plásticos de un solo uso. Para ello, se compara dos enfoques de clasificación automatizada: Sistema basado en sensores (infrarrojo cercano o NIR, humedad, temperatura, CO2, CH4 y un sensor de perfil láser). El sensor NIR identifica el tipo de polímero (PET, PVC, etc.), los sensores de gas, humedad y temperatura detectan la presencia de materia orgánica (envases sucios), y el láser determina si el objeto es bidimensional o tridimensional. Por otro lado, un sistema basado en una cámara RGB junto con modelos CNN para clasificar los materiales basándose en su apariencia visual. Los resultados principales demuestran que a través del modelo DL llamado Inception-v3, alcanzó una precisión del 78 %. Mientras que a través de la clasificación mediante sensores identifico mejor la composición química del material. Ambos métodos presentaron confusión y errores en la clasificación de algunos materiales, por lo que concluyeron que la combinación de ambos métodos tiene el potencial de superar las limitaciones de cada método por separado (Pučnik et al., 2024). Finalmente, en Italia mediante el uso de un fotodetector de germanio sobre silicio (Ge-on-Si) sintonizable por voltaje, se clasificaron a través de ML los datos obtenidos

de desechos de plásticos, vidrio, aluminio y papel. El núcleo del sistema funciona con doble banda que integra dos fotodiodos y puede operar tanto en el espectro de luz visible (400–1100 nm) como en el infrarrojo de onda corta (SWIR, 1000–1600 nm). El sistema permite extraer información de los diferentes materiales a partir de la luz que reflejan. El algoritmo de los KNN resultó ser el más eficaz, superando a otros métodos como RF y SVM y alcanzó una precisión global del 95 % (Manakkakudy Kumaran et al., 2024).

II.1. Conceptos

Residuo: Material o producto cuyo propietario o poseedor desecha y que se encuentra en estado sólido o semisólido, o es un líquido o gas contenido en recipientes o depósitos, y que puede ser susceptible de ser valorizado o requiere sujetarse a tratamiento o disposición final conforme a lo dispuesto en esta Ley y demás ordenamientos que de ella deriven (Cámara de Diputados, 2023).

Residuos Sólidos Urbano: Los generados en las casas habitación, que resultan de la eliminación de los materiales que utilizan en sus actividades domésticas, de los productos que consumen y de sus envases, embalajes o empaques; los residuos que provienen de cualquier otra actividad dentro de establecimientos o en la vía pública que genere residuos con características domiciliarias, y los resultantes de la limpieza de las vías y lugares públicos, siempre que no sean considerados por esta Ley como residuos de otra índole (Cámara de Diputados, 2023).

Reciclado: Transformación de los residuos a través de distintos procesos que permiten restituir su valor económico, evitando así su disposición final, siempre y cuando esta restitución favorezca un ahorro de energía y materias primas sin perjuicio para la salud, los ecosistemas o sus elementos (Cámara de Diputados, 2023).

Orgánicos: Todo desecho de origen biológico que alguna vez estuvo vivo o fue parte de un ser vivo (SEMARNAT, 2017).

Inorgánicos: Todo desecho que no es de origen biológico (SEMARNAT, 2017).

Residuo textil: El procedente de la ropa, calzado y otro material textil como ropa del hogar, bolsas, paños, etc., que una vez utilizado durante un periodo de tiempo determinado se convierte

en residuo. Por otra parte, también se incluyen los excedentes de la industria textil o de cualquier industria que utilice tejido textil, hilos, etc., en su proceso productivo (SEMARNAT, 2015).

Residuos de madera: Existen tres tipos de residuos de madera, sin tratar, industrializada y tratada con conservantes o pintada. Residuos de madera sin tratar son aquellos que no han sido tratados con conservantes; residuos de madera industrializada, en la construcción se utilizan diversos productos de madera como el contrachapado, los tableros de virutas orientadas, la madera laminada enchapada, la madera laminada encolada, los tableros de partículas y los tableros de fibra de densidad media; residuos de madera con conservantes se refiere a la madera recubierta o pintada para mejorar la calidad de los productos (Jahan et al., 2022).

Inteligencia Artificial: Es la tecnología que permite que las computadoras simulen la inteligencia humana y las capacidades humanas de resolución de problemas (IBM, 2023).

Visión Artificial: Campo de la IA que incluye la clasificación de imágenes, la detección de objetos y la segmentación semántica. Emplea el ML y las redes neuronales para enseñar a las computadoras y a los sistemas de aprendizaje a derivar información significativa de imágenes digitales, videos y otras entradas visuales, y a hacer recomendaciones o tomar medidas cuando el sistema detecta defectos o problemas (IBM, 2023).

Redes Neuronales Artificiales: Son sistemas adaptativos que aprenden empleando neuronas o nodos interconectados en una estructura en capas que se asemeja al cerebro humano. Las redes neuronales pueden aprender a partir de datos; por lo tanto, pueden entrenarse para reconocer patrones, clasificar datos y predecir eventos futuros. Las redes neuronales descomponen la entrada en capas de abstracción. Pueden entrenarse con muchos ejemplos para reconocer patrones en voz o imágenes del mismo modo que el cerebro humano. El comportamiento de las redes neuronales está definido por la forma en que se conectan sus elementos individuales y la fortaleza, o pesos, de esas conexiones. Estos pesos se ajustan automáticamente durante el entrenamiento conforme a una regla de aprendizaje especificada hasta que las redes neuronales artificiales ejecutan la tarea deseada correctamente (MathWorks, 2024).

Multilayer Perceptron: Es una red neuronal feedforward profunda formada por una o más

capas ocultas, cuya capacidad de representación depende de la composición sucesiva de funciones no lineales y de la profundidad del modelo (Goodfellow et al., 2016).

Arquitecturas ligeras: corresponden a modelos optimizados para reducir el consumo de memoria y el costo computacional, generalmente mediante el uso de menor número de parámetros, operaciones eficientes y representaciones de baja precisión numérica (Goodfellow et al., 2016).

Modelos preentrenados: Son redes neuronales cuyos parámetros han sido ajustados previamente utilizando grandes conjuntos de datos, con el propósito de servir como punto de partida para nuevas tareas relacionadas (Goodfellow et al., 2016).

Normalización por lotes: Es una técnica aplicada a las capas de una red neuronal que reescala las activaciones internas para mejorar la estabilidad del entrenamiento y facilitar la optimización (Goodfellow et al., 2016).

Función de activación: Son transformaciones no lineales aplicadas a la salida de una combinación afín en una neurona, permitiendo que las redes neuronales representen relaciones complejas entre entradas y salidas (Goodfellow et al., 2016).

Función de pérdida: Es una medida escalar que cuantifica la discrepancia entre as predicciones del modelo y los valores reales, y constituye el criterio que se busca minimizar durante el entrenamiento (Goodfellow et al., 2016).

Optimización del modelo: Es el proceso de ajuste de los parámetros de una red neuronal mediante algoritmos basados en gradiente, con el fin de minimizar la función de pérdida definida (Goodfellow et al., 2016).

Conjunto de datos: Es una colección estructurada de ejemplos que representan la experiencia utilizada por un algoritmo de aprendizaje para entrenar, validar y evaluar un modelo (Goodfellow et al., 2016).

Características de las imágenes: Estas características se pueden clasificar en tres tipos principales: color, textura y forma. Cada una se asocia con métricas de similitud que miden la semejanza o la distancia entre estas características de dos imágenes u objetos. Los dos primeros tipos de características se utilizan para representar tanto el contenido global como el regional o

específico de un objeto, mientras que las características de forma se utilizan únicamente para describir objetos. También existen muchas características derivadas de las imágenes, como las relaciones entre los objetos que las componen (Zhang, 2002).

Reducción de dimensionalidad: Es el proceso de transformar datos de alta dimensión a una representación de menor dimensión, preservando la mayor cantidad posible de información relevante (Goodfellow et al., 2016).

Algoritmo de aprendizaje automático: Es un conjunto de reglas o procesos que utiliza un sistema de IA para realizar tareas, generalmente para descubrir nuevos datos y patrones, o para predecir valores de salida a partir de un conjunto determinado de variables de entrada (IBM, 2023).

Extracción de características: Es una técnica que reduce la dimensionalidad o complejidad de los datos para mejorar el rendimiento y la eficiencia de los algoritmos de ML. Este proceso facilita las tareas de ML y mejora el análisis de datos al simplificar el conjunto de datos para incluir solo sus variables o atributos significativos (IBM, 2023).

Aprendizaje profundo: Es una aplicación específica de las funciones avanzadas que ofrecen los algoritmos de aprendizaje automático. La diferencia radica en cómo aprende cada algoritmo. Los modelos de aprendizaje automático "profundo" pueden usar conjuntos de datos etiquetados, también conocidos como aprendizaje supervisado, para fundamentar su algoritmo, pero no necesariamente requieren datos etiquetados (IBM, 2023).

Redes neuronales convolucionales (CNN o ConvNets): Se emplean principalmente en aplicaciones de visión artificial y clasificación de imágenes. Pueden detectar características y patrones dentro de imágenes y videos, lo que permite tareas, como la detección de objetos, el reconocimiento de imágenes, de patrones y facial. Estas redes aprovechan los principios del álgebra lineal, en particular la multiplicación de matrices, para identificar patrones dentro de una imagen (IBM, 2023).

Unidad Central de Procesamiento (CPU): El CPU puede tener varios núcleos y comúnmente se la conoce como el "cerebro" de la computadora. Es esencial para todos los sistemas informáticos modernos, ya que ejecuta las órdenes y los procesos que necesitan la computadora

y el sistema operativo. La CPU también es importante para determinar la rapidez con que los programas pueden ejecutarse mientras realizan tareas como navegar por internet, hacer cálculos de física para juegos y otros programas o crear hojas de cálculo (Intel, 2024).

Unidad de Procesamiento de Gráficos (GPU): La GPU tiene muchos núcleos más pequeños y especializados. Estos núcleos ofrecen un enorme desempeño al trabajar juntos y dividir las tareas de procesamiento entre muchos núcleos simultáneamente (o en paralelo). La GPU se destaca en tareas altamente paralelas, como renderizar imágenes durante un juego, manipular datos de video durante la creación de contenido y calcular resultados en cargas de trabajo de IA intensivas (Intel, 2024).

Python: Es un lenguaje de programación potente y fácil de aprender. Tiene estructuras de datos de alto nivel eficientes y un simple pero efectivo sistema de programación orientado a objetos. La elegante sintaxis de Python y su tipado dinámico, junto a su naturaleza interpretada lo convierten en un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en muchas áreas, para la mayoría de plataformas (Python, 2024).

TensorFlow: Es una biblioteca de software para el diseño e implementación de cálculos numéricos, centrada en aplicaciones de aprendizaje automático (NVIDIA, 2023).

Métricas de evaluación: Pueden ayudar a supervisar continuamente el rendimiento de los modelos de IA para proporcionar información a lo largo del ciclo de vida de la IA (IBM, 2023).

Métricas de evaluación de exactitud: Mide la exactitud de las predicciones de su modelo calculando la proporción de resultados correctos entre el número total de resultados (IBM, 2023).

La precisión se calcula con la siguiente ecuación (1):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

- TP = Verdaderos positivos.
- TN = Negativos verdaderos.
- FP = Falsos positivos.

- FN = Falsos negativos.

Métricas de precisión: Es la proporción de previsiones de clase positivas que realmente pertenecen a la clase en cuestión. Otra forma de entender la precisión es que mide la probabilidad de que una instancia elegida de manera aleatoria pertenezca a una determinada clase. La precisión también puede denominarse valor previsto positivo (PPV). Se representa mediante la ecuación (2) (IBM, 2023):

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

- TP = Verdaderos positivos.
- FP = Falsos positivos.

Métricas de coincidencia: Denota el porcentaje de instancias de clase detectadas por un modelo. En otras palabras, indica la proporción de previsiones positivas para una clase dada de todas las instancias reales de esa clase. La coincidencia también se conoce como sensibilidad o tasa de verdaderos positivos (TPR) y se representa mediante la ecuación (3) (IBM, 2023):

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

- TP = Verdaderos positivos.
- FN = Falsos negativos.

Métricas de puntuación F1: La puntuación F1, también llamada puntuación F, medida F o media armónica de precisión y coincidencia, combina precisión y coincidencia para representar la precisión total de clase de un modelo (4) (IBM, 2023):

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4}$$

III. Justificación

La contaminación a largo plazo por desechos es uno de los factores más determinantes de la pérdida de biodiversidad y pone en riesgo la integridad de los ecosistemas. Los desechos que se descomponen en la tierra pueden provocar contaminación a largo plazo en las fuentes de agua dulce por patógenos, metales pesados, productos químicos que alteran el sistema endocrino y otros compuestos peligrosos (PNUMA, 2024). México ha sostenido una alianza estratégica con la Organización de Naciones Unidas (ONU) y mantenido su participación activa en la implementación de la Agenda 2030. El ODS 11 (que se refiere a lograr que las ciudades y asentamientos humanos sean inclusivos, seguros, resilientes y sostenibles) y el ODS 12 (cuyo objetivo es garantizar modalidades de consumo y producción sostenibles) hacen hincapié en la reducción del impacto ambiental mediante la gestión y disminución de los desechos municipales a través de actividades de prevención, reducción, reciclado y reutilización (ONU México, 2025).

El camino hacia el progreso es avanzar hacia una economía circular y adoptar un enfoque de cero residuos. Las herramientas que se utilicen y el ritmo del cambio los determinarán las circunstancias nacionales de cada gobierno. Se recomiendan el uso de datos y digitalización para priorizar la prevención y la gestión de los desechos (PNUMA, 2024). Las capacidades de clasificación de los contenedores inteligentes, revelaron que las categorías de RSU más frecuentes que ocupan los modelos de clasificación: metal (47 %), plásticos (44 %), vidrio (28 %), materiales húmedos y secos (25 %), reciclable y no reciclable (19 %), papel y cartón (19 %), biodegradable y no biodegradable (11 %), electrónicos (11 %), orgánicos (11 %), mixtos (6 %) y madera (3 %); donde los porcentajes son la ocurrencia de escenario de las instancias que presentaron funcionalidades de clasificación de residuos (Zoumpoulis et al., 2024).

Los modelos basados en ML son algoritmos clásicos, que destacan por ser confiables al interpretar datos estructurados y dependen de la extracción manual de características a través de las imágenes de residuos, lo que resulta en un proceso laborioso que no garantiza precisión óptima, a diferencia de los modelos de DL que lo hacen de manera automática. El uso de modelos preentrenados, permite lograr mejores precisiones, lo que los hace ideales para la clasificación en

tiempo real. El DL exige altos recursos computacionales y grandes volúmenes de datos. Además, sufren caídas severas de precisión cuando se enfrentan a escenarios del mundo real con mala iluminación, objetos superpuestos o fondos desordenados. Los modelos híbridos combinan el ML con el DL, esta integración permite aprovechar lo mejor de cada técnica, lo que permite lograr mejores tasas de precisión, la desventaja es que su arquitectura múltiple suele ser compleja al diseñarse e implementarse (Fotovvatikhah et al., 2025). El MLP es una estructura de red neuronal dentro del ML y su integración mejora significativamente la calidad de representación de los datos y aumenta el rendimiento del sistema al transferir aprendizaje a tareas complejas (Umar, 2026).

Algunos trabajos aprovechan la extracción de características para mejorar la velocidad y precisión, especialmente en muestras complejas con texturas similares (Nežerka et al., 2024), existen diferencias muy sutiles entre muchas imágenes debido a su fondo complejo, lo que lleva a juicios erróneos del sistema, para tratar de resolver esta problemática se utilizan técnicas clásicas de extracción de características como GLCM (Mythili y Anbarasi, 2022) LBP, HOG (Cheng et al., 2023), FD (Su et al., 2024), bordes Canny/Sobel (Bieringer et al., 2024).

En este sentido, la presente investigación se justifica en la necesidad de desarrollar una base metodológica eficiente y precisa para la clasificación de residuos de madera y textil, mediante la integración de técnicas de extracción de características texturales y su clasificación con ML. Este enfoque busca mejorar la precisión, reducir el costo computacional y contribuir al desarrollo de sistemas automatizados accesibles para la gestión inteligente de residuos.

IV. Objetivos

IV.1. Objetivo general

Desarrollar un modelo clasificador de RSU del tipo Madera y Textil mediante ML ocupando técnicas clásicas de extracción de características.

IV.2. Objetivos específicos

1. Recopilar y seleccionar un conjunto de imágenes (Dataset) de RSU de madera y textil en diversas fuentes públicas y repositorios científicos.
2. Analizar algoritmos de extracción y selección de características, orientados a mejorar la eficiencia y precisión del modelo clasificador.
3. Entrenar modelos de ML para la clasificación de RSU de madera y textil.
4. Evaluar y ajustar el modelo clasificador.

V. Planteamiento del problema

Para el año 2050, se proyecta que la generación global de basura alcance los 3,400 millones de toneladas, evidenciando que el incremento acelerado de los RSU representa un desafío ambiental crítico a nivel mundial (Kaza et al., 2018). Ante esta problemática, la urgencia de optimizar las tecnologías de tratamiento y recuperación se hace evidente en México, donde se genera un promedio de 108,146 toneladas diarias de residuos (INEGI, 2025). Para facilitar y automatizar los procesos de separación y reciclaje, la IA ha emergido como una herramienta computacional clave; sin embargo, la gran mayoría de los sistemas actuales están enfocados en clasificar materiales como papel, plástico, vidrio y metal (Zoumpoulis et al., 2024). Como consecuencia, residuos con un gran potencial para reintegrarse a la economía circular, tales como la madera (Jahan et al., 2022) y los textiles (Vercalsteren et al., 2019), han recibido muy poca atención y han quedado rezagados de la automatización. La SEMARNAT estima que el porcentaje de la madera es de 0.79 % y textil (trapo) 2.82 % a nivel nacional (SEMARNAT, 2020).

Los enfoques predominantes de IA para la clasificación automática de basura se basan en el reconocimiento de desechos a partir de imágenes para reemplazar la clasificación manual (Liang y Gu, 2021), en este sentido existen modelos de IA de ML, DL y híbridos que combinan ambos. Las CNN actualmente son las más ocupadas para procesar de manera automática grandes cantidades de datos. Sin embargo, presentan la desventaja de requerir una cantidad significativa de recursos computacionales (Zhou et al., 2024).

Por lo tanto, el problema central radica en la necesidad de desarrollar nuevas herramientas tecnológicas que sean capaces de identificar de manera automática aquellas categorías de RSU ignorados (madera y textiles), pero que al mismo tiempo operen con un bajo costo computacional. Surge entonces el reto de validar si la implementación de técnicas clásicas de extracción de características visuales orientadas a la textura combinadas con algoritmos tradicionales de ML, puede ofrecer una solución accesible para la separación de estos desechos sin depender de los exhaustivos recursos que exigen los modelos actuales.

V.1. Hipótesis

La integración de descriptores de textura clásicos (LBP, GLCM, HOG, FD y Canny/Sobel) como vector de características, con selección de los atributos más discriminantes, combinada con algoritmos de ML supervisado (SVM, RF y MLP), permite clasificar automáticamente RSU de madera y textil con mayor exactitud que el uso de descriptores individuales, bajo condiciones controladas de adquisición de imagen.

VI. Metodología

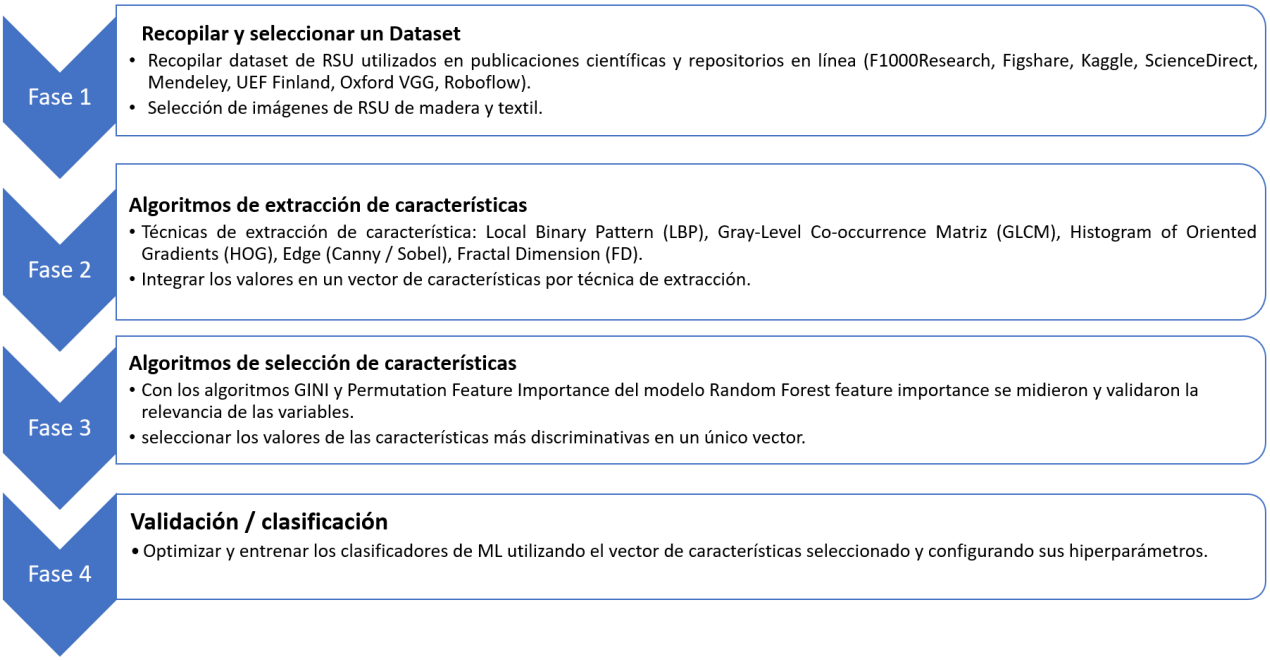
VI.1. Materiales

La experimentación se realizó con una computadora de escritorio ensamblada con sistema operativo Windows 10 de 64 bits, procesador Intel(R) Core (TM) i7-7700 CPU @ 3.60 GHz, 32 GB en RAM, tarjeta gráfica NVIDIA Quadro M6000 de 24 GB y lenguaje de programación Python versión 3.10.11.

VI.2. Metodología

Se utiliza una metodología cuantitativa - experimental, estructurada en etapas secuenciales, iniciando con la integración de un conjunto de datos, extracción de características mediante descriptores de textura (LBP, GLCM, HOG, FD), selección de atributos con RF por importancia Gini/Permutación, y validación cruzada mediante clasificadores SVM, RF y MLP (Figura. VI.2.0.1).

Figura VI.2.0.1. Metodología propuesta



Nota. Elaboración propia

VI.2.1. Integración del Dataset

Las imágenes son parte fundamental para la clasificación automatizada de residuos; a través de éstas, los modelos son capaces de operar en entornos reales. No obstante, la disponibilidad de fotografías con las características necesarias suele ser limitada. Por lo tanto, con el objetivo de construir un conjunto de datos para las categorías de RSU de madera y textil, se buscaron y recopilaron imágenes a partir de diferentes fuentes públicas y repositorios científicos; donde cada conjunto de datos fue originalmente construido en distintos escenarios y con un propósito en específico, como se muestra en la Tabla VI.2.1.1. Se realizó una inspección visual para identificar y eliminar imágenes duplicadas seleccionando aquellas muestras que representaran de una mejor manera las características texturales de cada material, patrones en textil como tejidos, bordados, mezclilla, superficies deterioradas y en madera vetas, nudos, grietas, con la intención de aproximarse a condiciones reales de operación.

Tabla VI.2.1.1. *Conjuntos de datos recopilados*

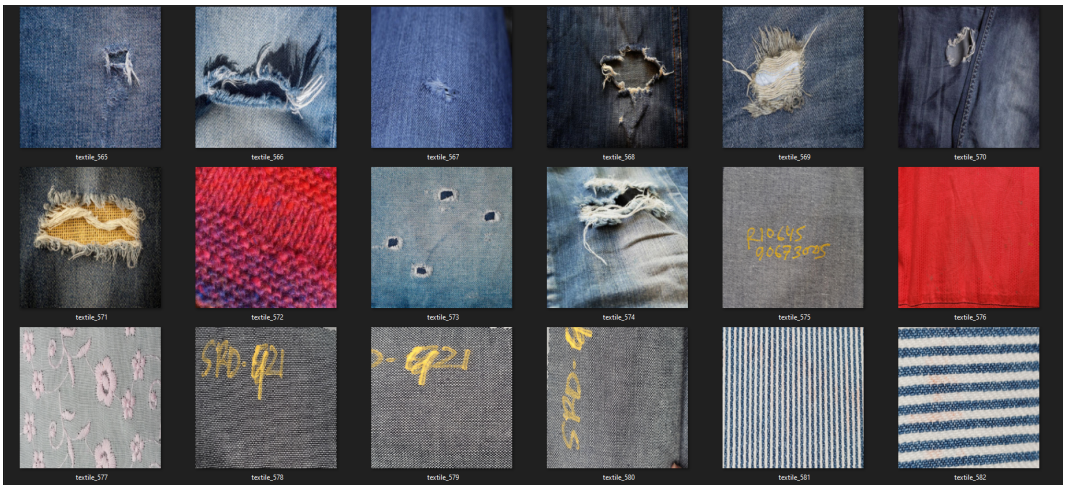
Clase	Origen	Tamaño	Referencia
Wood	Investigación F1000	640×640	Kodytek et al. (2022)
Textile	Figshare	640×640	Neamah (2024)
Textile	Kaggle	256×256	Shakir y Topal (2022)
Textile	Sciencedirect	365×365	Gil-Arroyo et al. (2025)
Textile	Mendeley	Tamaño variado	Sumaya et al. (2024)
Textile	Springer Nature	640×640	Mirhashemi (2018)
Textile	IEEE	Tamaño variado	Cimpoi et al. (2014)
Textile	Roboflow	640×640	Sinems Workplace (2025)
Textile	Roboflow	640×640	Usuario defect-8n3b5 (2025)
Textile	Roboflow	640×640	meraldanma (2024)

Nota."Comparative Analysis of Techniques for Texture Feature Extraction for Supervised Classification of Wood and Textile Waste," por W. C. Francisco, J. V. Tavira, J. J. C. Vega, B. D. V. Robles, E. R. Tamariz y A. B. Ortega, 2026, *Recycling*, *11*(5), p. 86.

El conjunto final está integrado por 4,396 imágenes, distribuidas de manera balanceada en 2,198 muestras por clase, como se muestra en la Figura VI.2.1.1 y Figura VI.2.1.2. Todas las imágenes fueron normalizadas a una resolución de 640 × 640 píxeles y organizadas en una estructura de carpetas por cada clase, tal como se ilustra en la Figura. VI.2.1.3, lo que es necesario

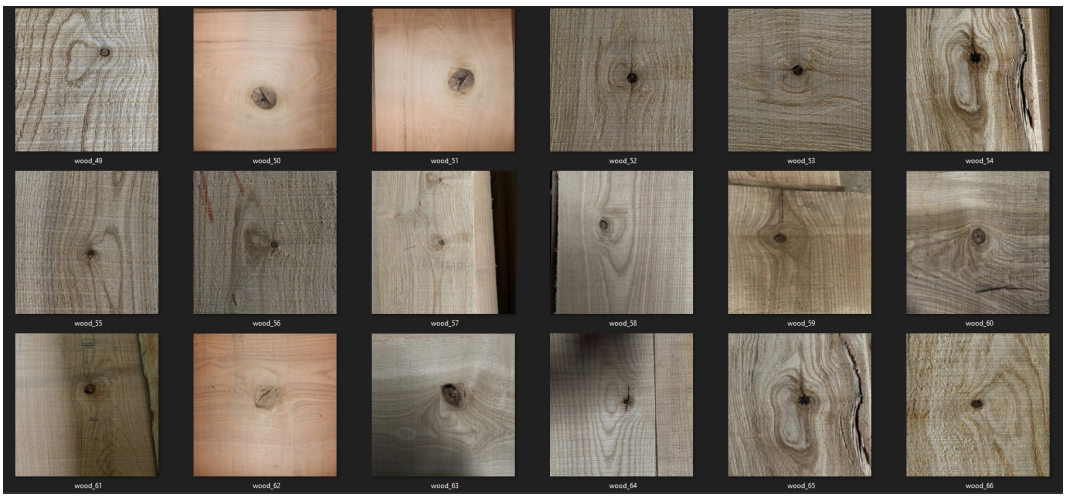
para que los algoritmos clásicos de extracción de características recorran las carpetas de origen, extraigan sus descriptores de textura de cada imagen, calculen y guarden sus valores.

Figura VI.2.1.1. Imágenes de la clase textil



Nota."Comparative Analysis of Techniques for Texture Feature Extraction for Supervised Classification of Wood and Textile Waste," por W. C. Francisco, J. V. Tavira, J. J. C. Vega, B. D. V. Robles, E. R. Tamariz y A. B. Ortega, 2026, *Recycling*, *11*(5), p. 86.

Figura VI.2.1.2. Imágenes de la clase madera



Nota."Comparative Analysis of Techniques for Texture Feature Extraction for Supervised Classification of Wood and Textile Waste," por W. C. Francisco, J. V. Tavira, J. J. C. Vega, B. D. V. Robles, E. R. Tamariz y A. B. Ortega, 2026, *Recycling*, *11*(5), p. 86.

Figura VI.2.1.3. Estructura jerárquica de directorios del conjunto de datos

```
DataSetWT/
wood/
    wood_1/
      .
      .
      .
    wood_N/
textile/
    textile_1/
      .
      .
      .
    textile_N/
```

Nota."Comparative Analysis of Techniques for Texture Feature Extraction for Supervised Classification of Wood and Textile Waste," por W. C. Francisco, J. V. Tavira, J. J. C. Vega, B. D. V. Robles, E. R. Tamariz y A. B. Ortega, 2026, *Recycling*, *11*(5), p. 86.

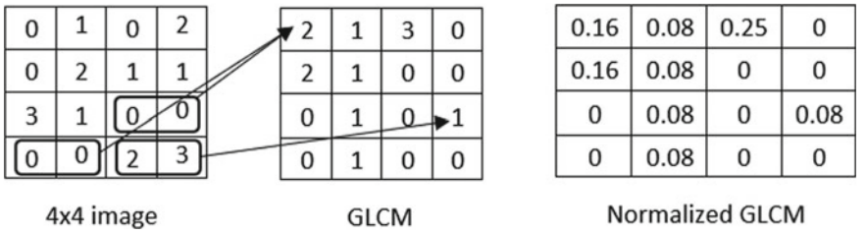
VI.2.2. Técnicas de extracción de características

Con técnicas clásicas como: LBP, GLCM, HOG, Canny/Sobel edge detection y FD, se calcularon los valores de características de texturas de cada imagen.

Gray-Level Co-occurrence Matrix (GLCM)

La técnica de GLCM es un método estadístico basado en matrices utilizado para analizar y medir características de textura en una imagen. Esta técnica funciona evaluando simultáneamente dos píxeles a la vez (un píxel de referencia y un píxel vecino). Durante el recorrido de una imagen se cuenta y registra el número de veces que un par específico de intensidades aparece siguiendo la relación espacial estipulada. Con la GLCM calculada, se extrae una amplia variedad de propiedades estadísticas, valores que se normalizan de manera de que cada valor describa las intensidades de gris de la imagen (Chaki y Dey, 2020), como se muestra en la Figura. VI.2.2.1. Con cada imagen se obtiene información de contraste, correlación, energía, homogeneidad, disimilitud, momento angular, en cuatro direcciones $(0°, 45°, 90°, 135°)$, la ecuación de cada característica se ponen a continuación (Utaminingrum et al., 2023):

Figura VI.2.2.1. Creación de GLCM a partir de una imagen



Nota."*Texture Feature Extraction Techniques for Image Recognition,*" por J. Chaki y N. Dey, 2020, Springer Singapore.

Correlación (5): Es una medida de la dependencia lineal del nivel de gris entre píxeles en sus respectivas ubicaciones, su valor es 1 o -1 para una imagen con correlación completamente positiva o negativa (Chaki y Dey, 2020).

$$\text{Correlation} = \sum_i \sum_j \left((i\, j)\, p(i,j,d,\theta) - \mu_x \mu_y \right) \tag{5}$$

Contraste (6): Es la diferencia de color que distingue un objeto.

$$\text{Contrast} = \sum_i \sum_j (i-j)^2\, p(i,j,d,\theta) \tag{6}$$

Disimilitud (7): Mide la diferencia entre los niveles de gris de píxeles vecinos en una imagen.

$$\text{Dissimilarity} = \sum_i \sum_j p(i,j \mid d,\theta)\, |i-j| \tag{7}$$

Homogeneidad (8): Mide la distancia entre los elementos del GLCM y la diagonal del mismo.

$$\text{Homogeneity} = \sum_i \sum_j \frac{p(i,j,d,\theta)}{1+|i-j|} \tag{8}$$

La energía (9) y el segundo momento angular (ASM)(10): La energía es una medida de la uniformidad de la textura de una imagen. Mientras que el ASM se obtiene a partir de la raíz

cuadrada de la energía.

$$\text{Energy} = \sum_i \sum_j \left(p(i,j,d,\theta)\right)^2 \tag{9}$$

$$\text{ASM} = \sqrt{Energy} \tag{10}$$

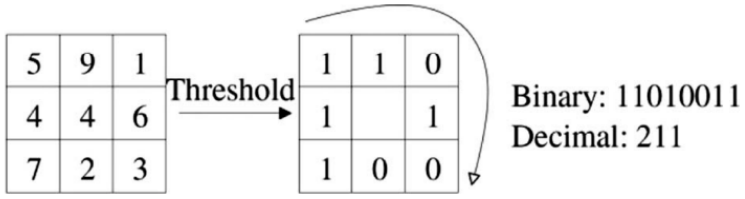
Información:

- p: probabilidad.
- i: nivel de gris de los píxeles en la ubicación (x, y).
- j: nivel de gris de los píxeles a la distancia d y ángulo de orientación θ.
- d: distancia de píxeles en el ángulo de orientación $\theta°$.
- θ: ángulo de orientación del píxel principal ($0°$, $45°$, $90°$, $135°$).
- $\mu_x\mu_y$: promedio de las probabilidades en las direcciones x y y.

Local Binary Pattern (LBP)

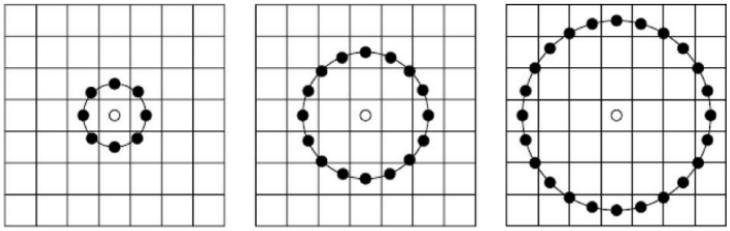
LBP etiqueta los píxeles de una imagen con números decimales que codifican la estructura local alrededor de cada píxel, cada uno se compara con sus ocho vecinos en una circunferencia de 3 × 3 restando el valor del píxel central; los valores estrictamente negativos resultantes se codifican con 0 y los demás con 1. Al hacer el recorrido, se obtiene un número binario concatenando todos estos valores obtenidos en sentido horario, comenzando por el de su vecino superior izquierdo. El valor decimal correspondiente al número binario generado se utiliza para etiquetar el píxel (Huang et al., 2011), como se muestra en la Figura. VI.2.2.2. Una circunferencia local se define como un conjunto de puntos de muestreo. La Figura. VI.2.2.3 muestra LBP extendido en 8, 16 y 24.

Figura VI.2.2.2. Operador LBP



Nota."Local Binary Patterns and Its Application to Facial Image Analysis: A Survey," por D. Huang, C. Shan, M. Ardabilian, Y. Wang y L. Chen, 2011, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *41*(6), pp. 765–781. Copyright 2011 por IEEE.

Figura VI.2.2.3. Vecindades circulares del operador LBP (8, 1), (16, 2) y (24, 3)



Nota."Local Binary Patterns and Its Application to Facial Image Analysis: A Survey," por D. Huang, C. Shan, M. Ardabilian, Y. Wang y L. Chen, 2011, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *41*(6), pp. 765–781. Copyright 2011 por IEEE.

Con los resultados obtenidos se calcula la media, varianza, asimetría, kurtosis, entropía (Sanjaya et al., 2020), desviación estándar y energía (Khan et al., 2022).

Histogram of Oriented Gradients (HOG)

HOG es una técnica de extracción de características que analiza las orientaciones de los gradientes locales. Una imagen de 640 x 640 pixeles, se divide en 80 X 80 celdas de 8 x 8 pixeles. Se calculan los gradientes de la imagen usando filtros como [-1, 0, 1] en X y en Y, obteniendo la magnitud y la dirección del borde en cada píxel. En cada celda se genera un histograma que cuenta cuántos gradientes apuntan en cada dirección; después, se agrupan celdas en bloques de 2 × 2 superpuestos y se normalizan los histogramas para reducir los efectos de la iluminación. Todos los histogramas normalizados se concatenan para formar un descriptor HOG final (Dalal y Triggs, 2005), luego se calcula la media, entropía, energía, desviación estándar.

Canny/Sobel edge detection

Canny: El algoritmo de detección de bordes introducido por John Canny, plantea una fórmula matemática para el rendimiento de una buena detección, localización y una sola respuesta a un solo borde. Canny demuestra que para detectar un borde afectado por ruido puede aproximarse mediante la ecuación (11) de la primera derivada de gaussiana $G'(x)$ (Canny, 1986).

$$\mathrm{G(x)} = \exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{11}$$

y el rendimiento teórico de una primera derivada de un filtro gaussiano con el operador óptimo (12).

$$\mathrm{f(x)} = -\frac{x}{\chi^2}\exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{12}$$

Sobel: Es un operador de diferenciación discreta que calcula una aproximación del gradiente de la función de intensidad de la imagen y permite identificar bordes significativos. El cálculo de la probabilidad de un borde es más fiable. Matemáticamente se emplean dos vectores bidimensionales de 3 x 3 cuyos componentes están dados por las derivadas en las direcciones horizontal y vertical para detectar bordes en ambas direcciones (Vincent y Folorunso, 2009). Se estima la magnitud del gradiente en cada píxel (13).

$$\mathrm{G(i,j)} = \sqrt{(I_1)^2 + (I_2)^2} \tag{13}$$

Fractal Dimension (FD)

La FD se ha utilizado ampliamente como característica en aplicaciones de visión artificial para caracterizar la rugosidad y la autosimilitud de los objetos en una imagen. Estas características se han adoptado con éxito principalmente en la segmentación y clasificación de texturas. La FD es una medida cuantitativa que muestra cuánto espacio adquiere un conjunto fractal (Panigrahy et al., 2019). Diversas técnicas se encuentran en la literatura para estimar la FD, una de ellas es el Differential Box Counting. Su rendimiento se ve influenciado por numerosos parámetros. En

particular, la altura de la caja está directamente relacionada con las variaciones en el nivel de gris sobre la cuadrícula de la imagen (Panigrahy et al., 2017). La fórmula estándar para calcular FD de un conjunto fractal acotado ideal, se expresa mediante la ecuación (14).

$$\mathrm{FD_B} = \frac{\log(N_r)}{\log\left(\frac{1}{r}\right)}, r \neq 1 \tag{14}$$

dónde, N_r es el número de casillas distintas de escala r que se requieren para cubrir B. Si B no es un conjunto fractal ideal, la ecuación dará diferentes valores de FD para diferentes valores de r. En ese caso, la pendiente de la línea obtenida al ajustar los puntos $(\log N_r, \log \frac{1}{r})\ \forall r$ Se conoce como la FD del conjunto fractal no ideal. El método de Canny en combinación con FD se ocuparon para un sistema eficiente de detección de cerebros patológicos basado en IA; demostrando que se puede extraer bordes con mayor eficiencia sin eliminar texturas importantes (Zhang et al., 2016).

VI.2.3. Extracción e integración de características

A continuación, se desglosan los procesos que cada método empleo para garantizar consistencia en los descriptores basados en intensidad y reducir la complejidad computacional.

GLCM: El proceso que se le realiza al conjunto de datos es que cada imagen es convertida a escala de grises para homogenizar la información de intensidad y reducir la complejidad del procesamiento. Seguido se calcula la GLCM utilizando diferentes distancias espaciales y orientaciones angulares, a partir de cada matriz GLCM, se extraen propiedades textuales estadísticas (Algoritmo 1).

Algorithm 1 Extracción de características GLCM

Input: Carpeta $Input\_Dir$, distancias D, ángulos θ, niveles L
Output: Archivo CSV con características GLCM
Obtener lista de imágenes desde $Input\_Dir$
for cada imagen img **do**
    Leer imagen
    Convertir a escala de grises
    Reducir niveles de gris a L
    Calcular matriz GLCM usando:
        $D = \{1, 2, 3\}$
        $\theta = \{0°, 45°, 90°, 135°\}$
    Extraer:
        contraste, correlación
        energía, homogeneidad
        disimilitud y ASM
    Calcular promedio de características
    Guardar resultados
end for
Exportar resultados a archivo `.csv`

LBP: Se inicia leyendo las imágenes almacenadas en la carpeta de entrada. Cada imagen es convertida a escala de grises y normalizada en un rango de intensidad de 0 a 255, con el propósito de homogenizar la información visual y garantizar estabilidad durante el procesamiento. Se calcula el descriptor utilizando un radio de vecindad de 3 píxeles y 48 vecinos. A partir de este proceso se extraen la entropía, energía, skewness, kurtosis, desviación estándar y número de bins no vacíos (Algoritmo 2).

Algorithm 2 Extracción de características LBP

Input: Carpeta $Base\_Dir$, radio R, vecinos P, método LBP
Output: Archivo CSV con características LBP
Obtener lista de imágenes desde $Base\_Dir$
for cada imagen img **do**
    Leer imagen
    Convertir a escala de grises
    Normalizar intensidades
    Calcular operador LBP usando:
        radio $R = 3$
        vecinos $P = 48$
        método uniforme
    Generar histograma LBP
    Normalizar histograma
    Extraer:
        entropía y energía
        skewness y kurtosis
        desviación estándar
        bins no vacíos
    Guardar características
end for
Exportar resultados a archivo `.csv`

HOG: Inicia leyendo las imágenes almacenadas en la carpeta de entrada. Cada imagen es convertida a escala de grises y redimensionada a 256 × 256 píxeles, para mejorar el contraste local se aplica la técnica CLAHE. Se analiza la distribución de gradientes utilizando 9 orientaciones

angulares, celdas de 16 × 16 y bloques de 2 × 2 celdas, empleando normalización L2-Hys (Dalal y Triggs, 2005), para mejorar la estabilidad frente a variaciones de iluminación y contraste (Algoritmo 3).

Algorithm 3 Extracción de características HOG

Input: Carpeta $Base\_Dir$, tamaño de imagen, parámetros HOG
Output: Archivo CSV con características HOG
Obtener lista de imágenes desde $Base\_Dir$
for cada imagen img **do**
    Leer imagen
    Convertir a escala de grises
    Redimensionar imagen a 256×256
    Aplicar mejora de contraste CLAHE
    Calcular descriptor HOG usando:
        9 orientaciones
        celdas de 16×16
        bloques de 2×2
        normalización L2-Hys
    Generar vector HOG
    Extraer:
        media y desviación estándar
        energía y entropía
    Guardar características
end for
Exportar resultados a archivo `.csv`

Canny/Sobel: Inicia leyendo las imágenes almacenadas en la carpeta de entrada. Cada imagen es convertida a escala de grises y se aplica la técnica CLAHE. Se calcula los gradientes horizontales y verticales mediante el operador Sobel utilizando un kernel de 3 × 33, para obtener la magnitud y dirección que permiten describir la intensidad y orientación de los bordes presentes en la imagen. El operador Canny utiliza umbrales inferior y superior para identificar regiones con cambios significativos de intensidad (Algoritmo 4).

Algorithm 4 Extracción de características Sobel y Canny

Input: Carpeta $Base\_Dir$, parámetros Sobel y Canny
Output: Archivo CSV con características de bordes
Obtener lista de imágenes desde $Base\_Dir$
for cada imagen img **do**
    Leer imagen
    Convertir a escala de grises
    Aplicar mejora de contraste CLAHE
    Calcular gradientes Sobel usando:
        kernel 3×3
    Obtener:
        magnitud del gradiente
        dirección del gradiente
    Extraer características Sobel:
        media y desviación estándar
        percentil 95
        energía
        entropía direccional
    Detectar bordes usando operador Canny
        umbral bajo = 50
        umbral alto = 150
    Extraer características Canny:
        proporción de bordes
        componentes conectados
        magnitud promedio de bordes
    Guardar características
end for
Exportar resultados a archivo `.csv`

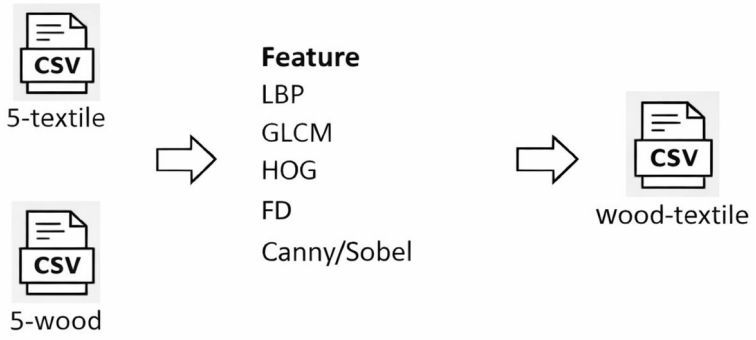
FD: Inicia leyendo las imágenes en la carpeta de entrada, convirtiéndolas a escala de grises, las intensidades son normalizadas para garantizar uniformidad durante el procesamiento. El algoritmo calcula la FD utilizando el método Differential Box Counting (DBC), el cual analiza la variación de intensidades en múltiples escalas de caja para estimar la complejidad estructural de la textura. De manera complementaria, se aplica el operador Canny para detectar bordes y calcular una segunda medida de FD basada en la distribución geométrica. Opcionalmente, el algoritmo puede generar mapas fractales locales mediante ventanas deslizantes, permitiendo calcular estadísticas regionales como la media y desviación estándar de la dimensión fractal (Algoritmo 5).

Algorithm 5 Extracción de características fractales

Input: Carpeta $Base\_Dir$, parámetros fractales y Canny
Output: Archivo CSV con características fractales
Obtener lista de imágenes desde $Base\_Dir$
for cada imagen img **do**
    Leer imagen
    Convertir a escala de grises
    Normalizar intensidades
    Calcular dimensión fractal usando:
        método Differential Box Counting (DBC)
        múltiples escalas de caja
    Detectar bordes usando operador Canny
        umbral bajo = 50
        umbral alto = 150
    Calcular dimensión fractal sobre bordes
    if mapa fractal habilitado **then**
        Dividir imagen en ventanas locales
        Calcular dimensión fractal por región
        Obtener:
            media fractal local
            desviación estándar fractal
    end if
    Guardar:
        FD_DBC
        FD_edges
        estadísticas fractales locales
end for
Exportar resultados a archivo `.csv`

Posteriormente, los archivos generados fueron integrados en un único conjunto de datos mediante un proceso de alineación basado en el nombre de la imagen, lo que permitió asociar correctamente las características provenientes de cada técnica en un solo archivo CSV con 25 características por imagen, como se muestra en la Figura. VI.2.3.1.

Figura VI.2.3.1. Integración de características



Nota."Comparative Analysis of Techniques for Texture Feature Extraction for Supervised Classification of Wood and Textile Waste," por W. C. Francisco, J. V. Tavira, J. J. C. Vega, B. D. V. Robles, E. R. Tamariz y A. B. Ortega, 2026, *Recycling*, *11*(5), p. 86.

El conjunto final está compuesto por un concentrado de los descriptores individuales. Cada técnica clásica incorpora propiedades propias permite obtener una representación más completa y

discriminativa del contenido visual, como se muestra en la Tabla VI.2.3.1.

Tabla VI.2.3.1. *Descripción de las características extraídas para el análisis de textura*

Técnica	Característica	Descripción
Fractal	FD_DBC	Dimensión fractal estimada mediante Differential Box Counting.
Fractal	FD_edges	Dimensión fractal calculada sobre mapas de bordes.
LBP	entropy_lbp	Entropía del histograma LBP, mide complejidad local.
LBP	energy_lbp	Energía del histograma LBP, indica uniformidad.
LBP	skew_lbp	Asimetría de la distribución de patrones LBP.
LBP	kurtosis_lbp	Curtosis del histograma LBP, mide concentración de valores.
LBP	std_lbp	Variabilidad de los valores del histograma LBP.
HOG	mean_HOG	Valor medio del histograma de gradientes orientados.
HOG	std_HOG	Desviación estándar del histograma HOG.
HOG	entropy_HOG	Entropía del histograma HOG, refleja complejidad estructural.
HOG	energy_HOG	Energía del histograma HOG.
GLCM	Contrast_mean_glcm	Contraste promedio entre niveles de gris.
GLCM	Correlation_mean_glcm	Correlación promedio de intensidades vecinas.
GLCM	Energy_mean_glcm	Uniformidad promedio de la matriz GLCM.
GLCM	Homogeneity_mean_glcm	Homogeneidad promedio de textura.
GLCM	Dissimilarity_mean_glcm	Variación promedio entre niveles vecinos.
GLCM	Asm_mean_glcm	Momento angular promedio de textura.
Sobel	sobel_mag_mean	Media de la magnitud del gradiente Sobel.
Sobel	sobel_mag_std	Variabilidad de la magnitud Sobel.
Sobel	sobel_mag_p95	Percentil 95 de la magnitud Sobel.
Sobel	sobel_energy	Energía del gradiente Sobel.
Sobel	sobel_dir_entropy	Entropía direccional Sobel.
Canny	canny_edge_ratio	Proporción de píxeles detectados como bordes.
Canny	canny_components	Número de componentes conectados.
Canny	canny_mean_edge_mag	Intensidad promedio de bordes detectados.

Nota."Comparative Analysis of Techniques for Texture Feature Extraction for Supervised Classification of Wood and Textile Waste," por W. C. Francisco, J. V. Tavira, J. J. C. Vega, B. D. V. Robles, E. R. Tamariz y A. B. Ortega, 2026, *Recycling*, *11*(5), p. 86.

VI.2.4. Selección de características

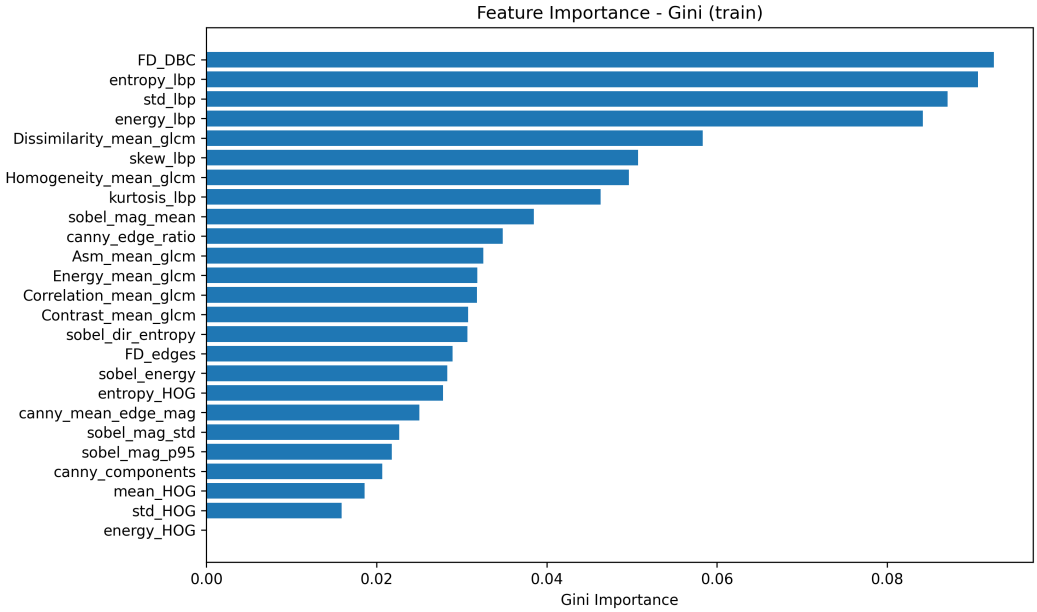
RF es una combinación de características de árboles, de modo que cada árbol depende de los valores de un vector aleatorio seleccionado de forma independiente y con la misma distribución para todos los árboles del bosque. El error de generalización de los bosques converge hasta un límite a medida que aumenta el número de árboles en el bosque. Las estimaciones internas monitorizan el error, fuerza, correlación, y se utilizan para mostrar la respuesta al aumentar el número de características utilizadas en la división. Las estimaciones internas también se utilizan para medir la importancia de las variables (Breiman, 2001).

Para evaluar la influencia de las variables incluidas en el conjunto de datos, se utilizan

métodos como el índice Gini, la entropía cruzada o la permutación (Nikitin y Stepashkin, 2025). Para la importancia de las características, la permutación de una característica se calcula como la disminución promedio en la precisión del modelo en las muestras datos out-of-bag cuando los valores de la característica respectiva se permutan aleatoriamente y Gini utiliza la disminución del índice de Gini (impureza) después de una división de nodos como una medida de la relevancia de la característica (Altmann et al., 2010).

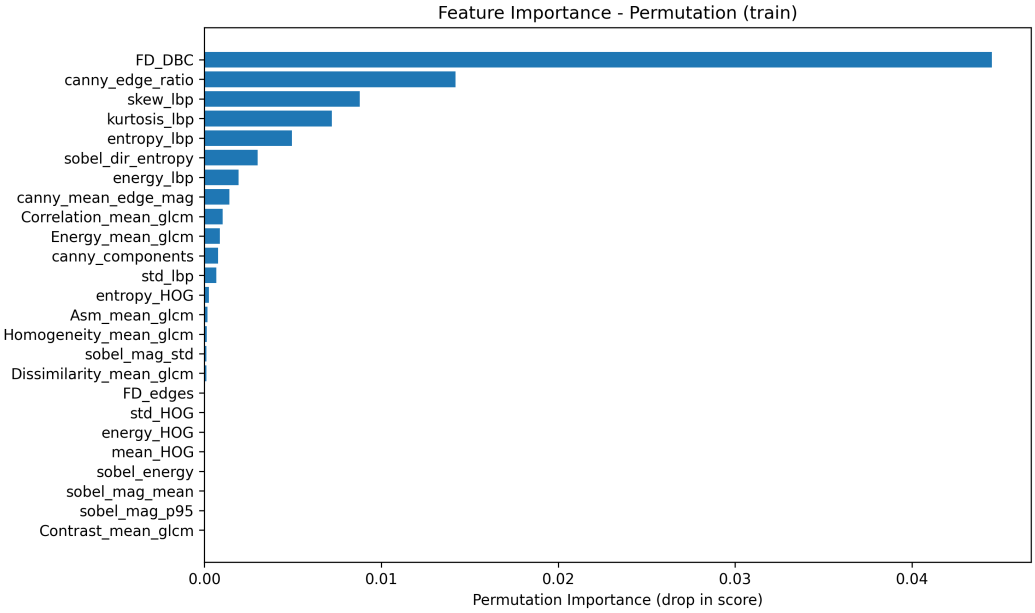
Gini muestra que las características con mayor importancia son la *FD_DBC, entropy_lbp, std_lbp, energy_lbp, Dissimilarity_mean_glcm*, las primeras cinco ordenadas de marera, como se observa en la Figura. VI.2.4.1. La importancia por permutación indica que las variables más representativas son: *FD_DBC, canny_edge_ratio, skew_lbp, kurtosis_lbp, entropy_lbp*, como se muestra en la Figura. VI.2.4.2.

Figura VI.2.4.1. Importancia de características usando GINI



Nota."Comparative Analysis of Techniques for Texture Feature Extraction for Supervised Classification of Wood and Textile Waste," por W. C. Francisco, J. V. Tavira, J. J. C. Vega, B. D. V. Robles, E. R. Tamariz y A. B. Ortega, 2026, *Recycling*, *11*(5), p. 86.

Figura VI.2.4.2. Importancia de características usando Permutación



Nota."Comparative Analysis of Techniques for Texture Feature Extraction for Supervised Classification of Wood and Textile Waste," por W. C. Francisco, J. V. Tavira, J. J. C. Vega, B. D. V. Robles, E. R. Tamariz y A. B. Ortega, 2026, *Recycling*, *11*(5), p. 86.

Se implemento un análisis y evaluación de características, el programa divide los datos en conjuntos de entrenamiento y prueba mediante (80/20), en la etapa de entrenamiento, el código construye un modelo compuesto por 300 árboles de decisión y lo entrena utilizando el conjunto de entrenamiento completo. Finalizado el entrenamiento se calcula la importancia de las características Gini y por permutación. Automáticamente se construyen diferentes subconjuntos de características para realizar un estudio de comparación entre los subconjuntos, como se muestra en la Tabla VI.2.4.1.

Tabla VI.2.4.1. *Conjunto y subconjuntos de características*

Selección	Características	Nombre
Conjunto completo	FD_DBC, FD_edges, entropy_lbp, energy_lbp, skew_lbp, kurtosis_lbp, std_lbp, mean_HOG, std_HOG, energy_HOG, entropy_HOG, Contrast_mean_glcm, Correlation_mean_glcm, Energy_mean_glcm, Homogeneity_mean_glcm, Dissimilarity_mean_glcm, Asm_mean_glcm, sobel_mag_mean, sobel_mag_std, sobel_mag_p95, sobel_energy, sobel_dir_entropy, canny_edge_ratio, canny_components, canny_mean_edge_mag	Full_25
Gini + Permutation	FD_DBC, entropy_lbp, std_lbp, energy_lbp, skew_lbp, kurtosis_lbp, canny_edge_ratio, canny_components, canny_mean_edge_mag, Homogeneity_mean_glcm, Energy_mean_glcm, Correlation_mean_glcm, FD_edges, sobel_mag_mean, sobel_dir_entropy, sobel_energy	Selected_16
Ranking: Gini rank + Permutation rank	FD_DBC, entropy_lbp, skew_lbp, kurtosis_lbp, std_lbp, energy_lbp, canny_edge_ratio, Dissimilarity_mean_glcm, Homogeneity_mean_glcm, Correlation_mean_glcm, canny_mean_edge_mag, Asm_mean_glcm, sobel_dir_entropy, sobel_mag_mean, canny_components, Energy_mean_glcm	Top_16_rank
Mayor relevancia	FD_DBC, entropy_lbp, skew_lbp, kurtosis_lbp, std_lbp, energy_lbp, canny_edge_ratio, Dissimilarity_mean_glcm, Homogeneity_mean_glcm, Correlation_mean_glcm	Top_10_rank
Más discriminativas	FD_DBC, entropy_lbp, skew_lbp, kurtosis_lbp, std_lbp	Top_5_rank
Textura local + fractal	FD_DBC, FD_edges, entropy_lbp, energy_lbp, skew_lbp, kurtosis_lbp, std_lbp	LBP_plus_FD
Sin FD	entropy_lbp, energy_lbp, skew_lbp, kurtosis_lbp, std_lbp, mean_HOG, std_HOG, energy_HOG, entropy_HOG, Contrast_mean_glcm, Correlation_mean_glcm, Energy_mean_glcm, Homogeneity_mean_glcm, Dissimilarity_mean_glcm, Asm_mean_glcm, sobel_mag_mean, sobel_mag_std, sobel_mag_p95, sobel_energy, sobel_dir_entropy, canny_edge_ratio, canny_components, canny_mean_edge_mag	Without_FD
Sin descriptores de textura local	FD_DBC, FD_edges, mean_HOG, std_HOG, energy_HOG, entropy_HOG, Contrast_mean_glcm, Correlation_mean_glcm, Energy_mean_glcm, Homogeneity_mean_glcm, Dissimilarity_mean_glcm, Asm_mean_glcm, sobel_mag_mean, sobel_mag_std, sobel_mag_p95, sobel_energy, sobel_dir_entropy, canny_edge_ratio, canny_components, canny_mean_edge_mag	Without_LBP

Nota."Comparative Analysis of Techniques for Texture Feature Extraction for Supervised Classification of Wood and Textile Waste," por W. C. Francisco, J. V. Tavira, J. J. C. Vega, B. D. V. Robles, E. R. Tamariz y A. B. Ortega, 2026, *Recycling*, *11*(5), p. 86.

El subconjunto Selected_16 obtuvo el mejor desempeño, con una precisión de 96.59 % y una exactitud media en validación cruzada de 96.73 %, superando ligeramente al conjunto completo de 25 variables (96.14 % de accuracy y 96.67 % de validación cruzada). Este resultado

indica que la eliminación de variables redundantes o poco informativas reduce la dimensionalidad, y al no participar mejora la capacidad de generalización del modelo.

Además, el subconjunto Top_16_rank, obtenido a partir del ranking automático, presentó un desempeño ligeramente inferior al subconjunto Selected_16. Por otra parte, al excluir las variables derivadas de LBP se observó una caída importante en el rendimiento (94.55 %), lo que confirma que estas características constituyen el componente más discriminativo del sistema. En contraste, la exclusión de las variables de dimensión fractal produjo una disminución menor (96.25 %), indicando que FD aporta información estructural complementaria, aunque no actúa de manera aislada.

De igual forma, los subconjuntos reducidos LBP+FD y Top-5 mostraron pérdidas notables de desempeño, confirmando su limitada capacidad discriminativa de manera independiente. En conjunto, estos resultados demuestran que el subconjunto Selected_16 representa un equilibrio adecuado entre relevancia, complementariedad y reducción de dimensionalidad, como se muestra en la Tabla VI.2.4.2.

Tabla VI.2.4.2. *Comparación del rendimiento de los subconjuntos de características*

Conjuntos	Número de características	Accuracy (test)	CV mean
Selected_16	16	96.59 %	96.73 %
Top_16_rank	16	96.48 %	96.64 %
Without_FD	23	96.25 %	96.10 %
Full_25	25	96.14 %	96.67 %
Top_10_rank	10	95.91 %	96.30 %
Without_LBP	20	94.55 %	94.65 %
LBP_plus_FD	7	94.20 %	94.57 %
Top_5_rank	5	92.39 %	93.47 %

Nota."Comparative Analysis of Techniques for Texture Feature Extraction for Supervised Classification of Wood and Textile Waste," por W. C. Francisco, J. V. Tavira, J. J. C. Vega, B. D. V. Robles, E. R. Tamariz y A. B. Ortega, 2026, *Recycling*, *11*(5), p. 86.

El análisis valida experimentalmente que el subconjunto Selected_16 características constituyen una representación óptima y permitirá maximizar el rendimiento con una menor dimensionalidad. Este comportamiento evidencia que variables con menor importancia relativa pueden aportar información complementaria que mejora el desempeño global del modelo. En

consecuencia, este subconjunto representa un equilibrio adecuado entre relevancia, diversidad descriptiva y capacidad de generalización, validando su selección desde un punto de vista tanto estadístico como estructural.

VI.2.5. Entrenamientos de modelos de ML

El entrenamiento de los modelos de ML se llevó a cabo entre las etapas de ajuste de hiperparámetros y evaluación. En primer lugar, el conjunto de datos fue dividido mediante una partición estratificada en 80 % para entrenamiento y 20 % para prueba, garantizando que la distribución de clases se mantuviera. Con el conjunto obtenido de 16 características mediante Random Forest Importance se entrenaron SVM, RF y MLP.

El ajuste de hiperparámetros para cada modelo se realizó mediante una búsqueda en malla (GridSearchCV) (Sun et al., 2024), combinada con validación cruzada estratificada a siete particiones, aplicada únicamente sobre el conjunto de entrenamiento. Este procedimiento permitió seleccionar la mejor configuración de parámetros sin involucrar el conjunto de prueba. El conjunto de prueba se reservó estrictamente para la evaluación final del desempeño, con métricas como exactitud (accuracy), matriz de confusión, y log-loss. Adicionalmente, se generaron curvas de aprendizaje con el fin de analizar la estabilidad y capacidad de generalización de los modelos sin introducir sesgos.

Support Vector Machine (SVM)

SVM es un algoritmo que maximiza el margen entre los patrones de entrenamiento, la frontera de decisión (Boser et al., 1996); pueden gestionar tanto tareas de clasificación simples y lineales como complejos, es decir, gestionan tantos problemas separables como no separables. La idea de la SVM es mapear los puntos de datos originales del espacio de entrada a un espacio de características de alta dimensión, o incluso de dimensión infinita, de modo que el problema de clasificación se simplifique en dicho espacio (Luts et al., 2010).

la Tabla VI.2.5.1 contiene los valores numéricos de las métricas de modelo entrenado.

Tabla VI.2.5.1. *Resultados del modelo SVM-kernel RBF*

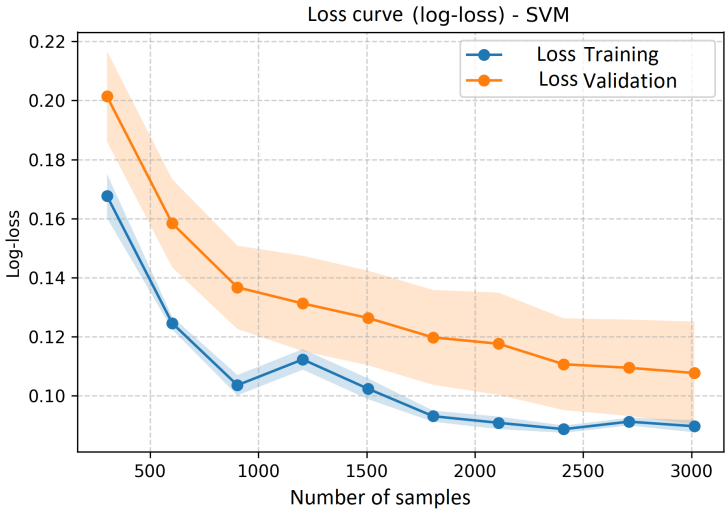
Parámetro / Métrica	**Valor**
C	1
gamma	scale
Accuracy (CV 7-fold)	0.9593
Accuracy (Test)	0.9522
AUC (Test)	0.9920
Log-loss (Test)	0.1133
Training time (s)	50.164
Reporte de Clasificación por Clase	
Clase 0 – Precision	0.9715
Clase 0 – Recall	0.9318
Clase 0 – F1-score	0.9512
Clase 1 – Precision	0.9344
Clase 1 – Recall	0.9727
Clase 1 – F1-score	0.9532

Clase 0 = textile, Clase 1 = wood

Nota."Comparative Analysis of Techniques for Texture Feature Extraction for Supervised Classification of Wood and Textile Waste," por W. C. Francisco, J. V. Tavira, J. J. C. Vega, B. D. V. Robles, E. R. Tamariz y A. B. Ortega, 2026, *Recycling*, *11*(5), p. 86.

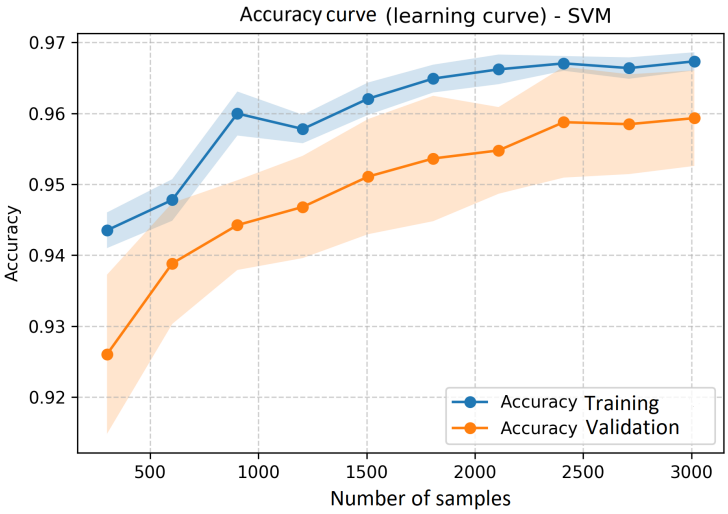
Lo que respecta al comportamiento de las curvas de pérdida y precisión, en la primera, ambas líneas caen conforme aumenta el número de muestras, se interpreta que el modelo generaliza bien, como se muestra en la Figura. VI.2.5.1. La línea de precisión crece al momento que aumentan los datos, la diferencia entre ambas es pequeña, lo que demuestra estabilidad, véase la Figura. VI.2.5.2.

Figura VI.2.5.1. Gráfico de pérdida para el modelo SVM durante el entrenamiento



Nota."Comparative Analysis of Techniques for Texture Feature Extraction for Supervised Classification of Wood and Textile Waste," por W. C. Francisco, J. V. Tavira, J. J. C. Vega, B. D. V. Robles, E. R. Tamariz y A. B. Ortega, 2026, *Recycling*, *11*(5), p. 86.

Figura VI.2.5.2. Gráfico de precisión para el modelo SVM durante el entrenamiento

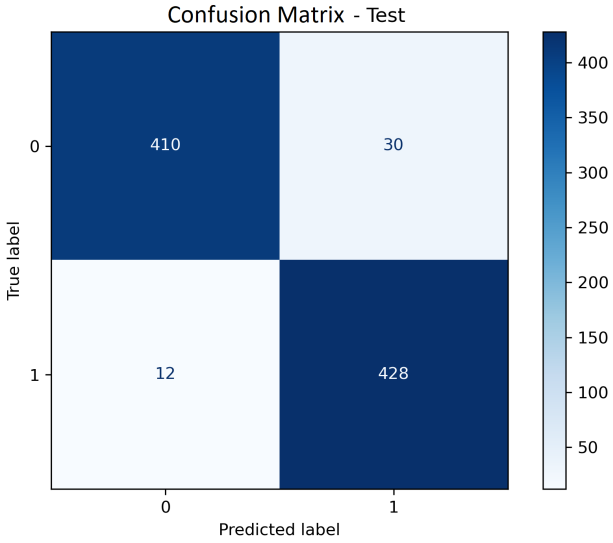


Nota."Comparative Analysis of Techniques for Texture Feature Extraction for Supervised Classification of Wood and Textile Waste," por W. C. Francisco, J. V. Tavira, J. J. C. Vega, B. D. V. Robles, E. R. Tamariz y A. B. Ortega, 2026, *Recycling*, *11*(5), p. 86.

La matriz de confusión generada con los datos de prueba, demuestra que 410 imágenes de textiles (clase: 0) son separadas correctamente y solo en 30 existe confusión. La clase wood

(clase: 1) en 12 hay una confusión y 428 son separadas correctamente, como se muestra en la Figura. VI.2.5.3.

Figura VI.2.5.3. Matriz de confusión del modelo SVM



Nota."Comparative Analysis of Techniques for Texture Feature Extraction for Supervised Classification of Wood and Textile Waste," por W. C. Francisco, J. V. Tavira, J. J. C. Vega, B. D. V. Robles, E. R. Tamariz y A. B. Ortega, 2026, *Recycling*, *11*(5), p. 86.

Random Forests (RF)

RF es una técnica de ML que también se utilizada para predecir la generación de residuos. Este algoritmo combina múltiples árboles de decisión para crear un potente modelo predictivo. Al entrenar el modelo RF con el archivo CSV que contienen patrones de residuos y características importantes, obtener relaciones complejas y proporcionar pronósticos (Pillai et al., 2023). La Tabla VI.2.5.2 contiene los valores numéricos de las métricas del modelo entrenado.

Tabla VI.2.5.2. *Resultados del modelo Random Forest*

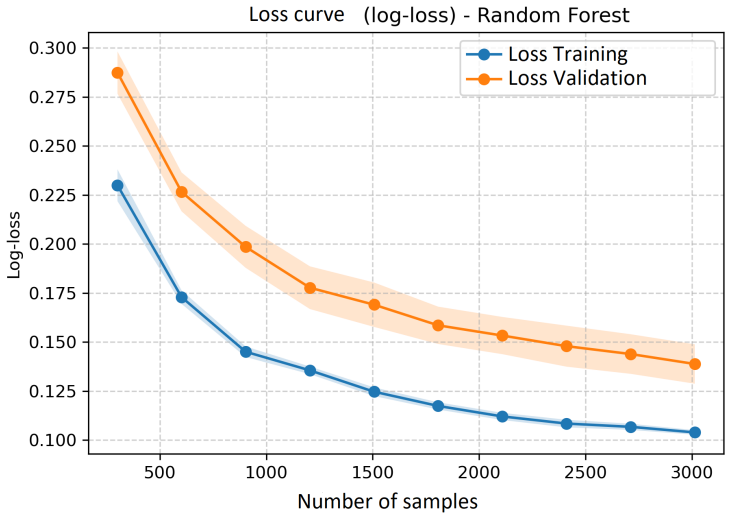
Parámetro / Métrica	Valor
n_estimators	200
max_depth	10
max_features	sqrt
min_samples_split	10
min_samples_leaf	10
Accuracy (CV 7-fold)	0.9573
Accuracy (Test)	0.9545
AUC (Test)	0.9907
Log-loss (Test)	0.1382
Training time (s)	144.83
Reporte de Clasificación por Clase	
Clase 0	Precision = 0.9545 Recall = 0.9545 F1-score = 0.9545
Clase 1	Precision = 0.9545 Recall = 0.9545 F1-score = 0.9545

Clase 0 = textile, Clase 1 = wood

Nota."Comparative Analysis of Techniques for Texture Feature Extraction for Supervised Classification of Wood and Textile Waste," por W. C. Francisco, J. V. Tavira, J. J. C. Vega, B. D. V. Robles, E. R. Tamariz y A. B. Ortega, 2026, *Recycling*, *11*(5), p. 86.

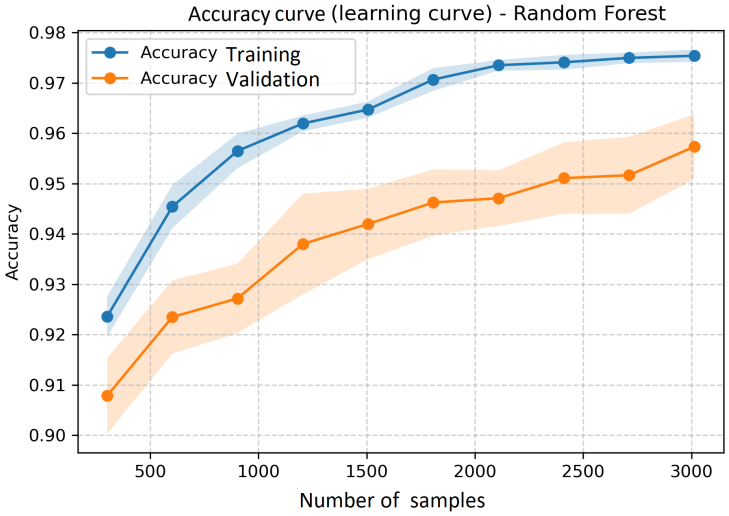
El comportamiento de entrenamiento muestra que la curva de pérdida es constante, con solo algunas diferencias que la línea de prueba, como se ilustra en la Figura. VI.2.5.4. en la Figura. VI.2.5.5, se puede observar que la precisión ambas líneas tienen un incremento constante, la diferencia de comportamiento es poca, lo que da a entender que no existe sobreajuste. Por otra parte, la matriz de confusión muestra que los valores de prueba de las dos clases tienen 20 errores por igual, ver Figura. VI.2.5.6.

Figura VI.2.5.4. Gráfico de pérdida para el modelo RF durante el entrenamiento



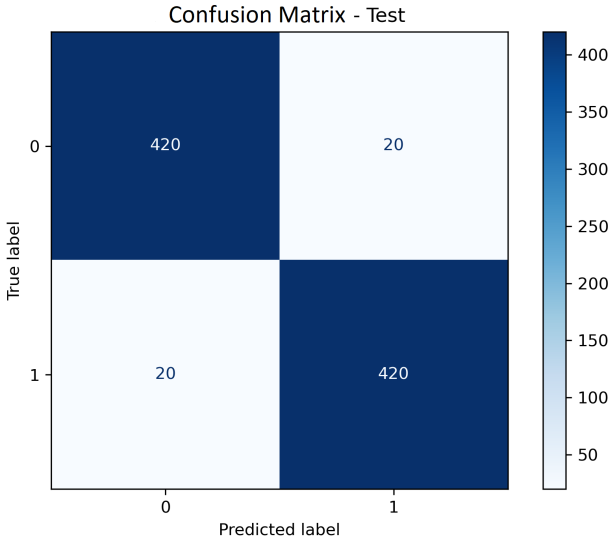
Nota."Comparative Analysis of Techniques for Texture Feature Extraction for Supervised Classification of Wood and Textile Waste," por W. C. Francisco, J. V. Tavira, J. J. C. Vega, B. D. V. Robles, E. R. Tamariz y A. B. Ortega, 2026, *Recycling*, *11*(5), p. 86.

Figura VI.2.5.5. Gráfico de precisión para el modelo RF durante el entrenamiento



Nota."Comparative Analysis of Techniques for Texture Feature Extraction for Supervised Classification of Wood and Textile Waste," por W. C. Francisco, J. V. Tavira, J. J. C. Vega, B. D. V. Robles, E. R. Tamariz y A. B. Ortega, 2026, *Recycling*, *11*(5), p. 86.

Figura VI.2.5.6. Matriz de confusión del modelo RF



Nota."Comparative Analysis of Techniques for Texture Feature Extraction for Supervised Classification of Wood and Textile Waste," por W. C. Francisco, J. V. Tavira, J. J. C. Vega, B. D. V. Robles, E. R. Tamariz y A. B. Ortega, 2026, *Recycling*, *11*(5), p. 86.

Perceptrón Multicapa (MLP)

MLP es una RNA inspirada en el comportamiento del sistema nervioso de los organismos superiores basadas en el procesamiento de neuronas artificiales con el fin de resolver problemas como clasificación de patrones, minería de datos, regresión/aproximación de funciones y el procesamiento de información (Piccioni Costa et al., 2023). Este algoritmo es ocupado para clasificar automáticamente residuos (Chu et al., 2018).

Además de la evaluación con el umbral predeterminado (0,5), se realizó un análisis complementario seleccionando un umbral de decisión óptimo basado en la puntuación F1 mediante la curva de precisión-exhaustividad. Esto resultó en un umbral óptimo de 0.6095, como se muestra en Tabla VI.2.5.3. Es importante destacar que este umbral se obtuvo utilizando las probabilidades predichas en el conjunto de prueba como un análisis post hoc, y no se utilizó durante el entrenamiento del modelo, el ajuste de hiperparámetros ni la calibración de probabilidad. Por lo tanto, los resultados correspondientes al umbral predeterminado (0,5) representan la evaluación principal del modelo, mientras que el umbral optimizado proporciona información adicional sobre

el comportamiento del clasificador bajo criterios de decisión alternativos.

Tabla VI.2.5.3. *Resultados del modelo MLP calibrado*

Parámetro / Métrica	Valor
Hiperparámetros	
hidden_layer_sizes	(256, 128, 64)
activation	relu
alpha	0.003
learning_rate_init	0.002
Rendimiento general (thr = 0.5)	
Accuracy (CV 7-fold)	0.9772
Accuracy (Test)	0.9625
AUC (Test)	0.9956
Log-loss (Test)	0.0852
Training time (s)	1148.20
Overall Performance (thr optimum F1 = 0.6095)	
Accuracy (Test)	0.9670
Reporte de clasificación por clase	
Clase 0	
Precision	0.9702
Recall	0.9636
F1-score	0.9669
Clase 1	
Precision	0.9638
Recall	0.9704
F1-score	0.9672

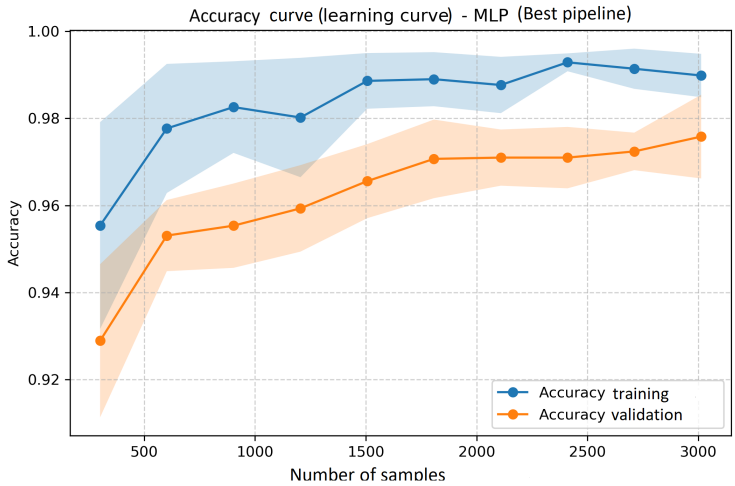
Clase 0 = textile, Clase 1 = wood

Nota."Comparative Analysis of Techniques for Texture Feature Extraction for Supervised Classification of Wood and Textile Waste," por W. C. Francisco, J. V. Tavira, J. J. C. Vega, B. D. V. Robles, E. R. Tamariz y A. B. Ortega, 2026, *Recycling*, *11*(5), p. 86.

Con un entrenamiento de hasta 3,000 muestras, el comportamiento de la curva de precisión con los valores de entrenamiento y prueba demuestra un entrenamiento equilibrado, ver Figura. VI.2.5.7. De igual manera en las líneas de pérdida disminuyen progresivamente, lo que corrobora que la clasificación de clases es confiable, como se muestra en la Figura. VI.2.5.8. La matriz de confusión muestra que, de 440 imágenes, solo 419 de la clase textile (clase: 0) fueron clasificadas correctamente, en lo que respecta a la madera (clase: 1) solo 21 no se clasificaron correctamente, esto indica que existe un equilibrio en la clasificación de clases, acorde a lo expuesto

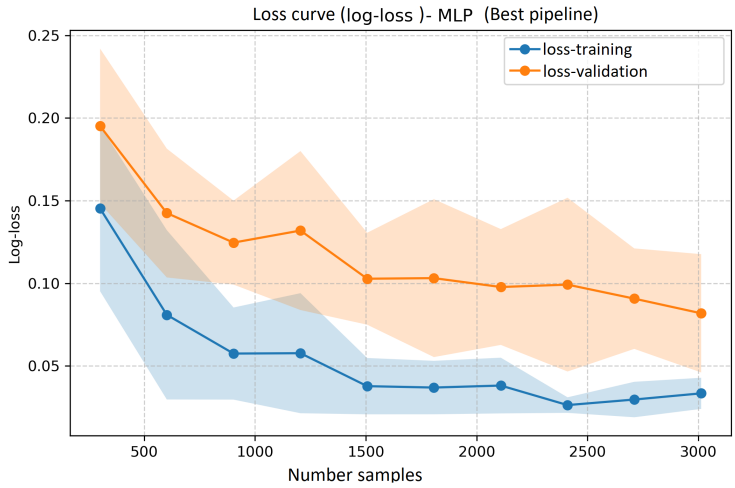
en la Figura. VI.2.5.9.

Figura VI.2.5.7. Gráfico de precisión para el modelo MLP durante el entrenamiento



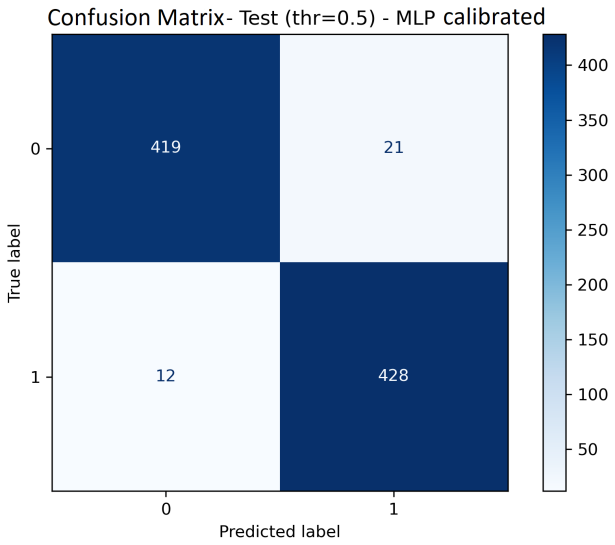
Nota."Comparative Analysis of Techniques for Texture Feature Extraction for Supervised Classification of Wood and Textile Waste," por W. C. Francisco, J. V. Tavira, J. J. C. Vega, B. D. V. Robles, E. R. Tamariz y A. B. Ortega, 2026, *Recycling*, *11*(5), p. 86.

Figura VI.2.5.8. Gráfico de pérdida para el modelo MLP durante el entrenamiento



Nota."Comparative Analysis of Techniques for Texture Feature Extraction for Supervised Classification of Wood and Textile Waste," por W. C. Francisco, J. V. Tavira, J. J. C. Vega, B. D. V. Robles, E. R. Tamariz y A. B. Ortega, 2026, *Recycling*, *11*(5), p. 86.

Figura VI.2.5.9. Matriz de confusión del modelo MLP



Nota."Comparative Analysis of Techniques for Texture Feature Extraction for Supervised Classification of Wood and Textile Waste," por W. C. Francisco, J. V. Tavira, J. J. C. Vega, B. D. V. Robles, E. R. Tamariz y A. B. Ortega, 2026, *Recycling*, *11*(5), p. 86.

VII. Resultados

Los tres modelos de ML muestran diferencias en sus resultados. En términos de exactitud el modelo MLP obtuvo el valor de 96.70 %, junto con un AUC de 99.56 %, lo que indica un alto poder discriminativo. Este valor corresponde al uso de un umbral óptimo basado en F1, mientras que con el umbral estándar de 0.5 el modelo alcanza una exactitud de 96.25 %. Por su parte, SVM logró una exactitud de 95.22 % con un AUC de 99.20 %, mientras que RF obtuvo una exactitud de 95.45 % y un AUC de 99.07 %. En cuanto al costo computacional, SVM es el modelo más rápido, seguido de RF y finalmente MLP, lo cual es consistente con la complejidad de cada enfoque. MLP requiere el mayor tiempo de entrenamiento, como se muestra en la Tabla VII.0.0.1; sin embargo, presenta ventajas adicionales, como la capacidad de modelar relaciones no lineales complejas mediante arquitecturas profundas y el ajuste adaptativo del aprendizaje (Bieringer et al., 2019), lo que, combinado con las técnicas de extracción de características empleadas, fortalece la clasificación de las texturas de residuos de madera y textil.

Tabla VII.0.0.1. *Comparación de resultados de los modelos SVM, RF y MLP*

Métricas/Parámetros	SVM (RBF)	RF	MLP

Hiperparámetros	`C = 1`, $\gamma =$`scale`	`n_estimators = 200`
		`max_depth = 10`
		`max_features = sqrt`
		`min_samples_split = 10`
		`min_samples_leaf = 10`
			`hidden_layer_sizes = (256,128,64)`
			`activation = relu`
			$\alpha = 0{,}003$
			`learning_rate_init = 0.002`

Accuracy (CV)	0.9593	0.9573	0.9772
Accuracy (Test)	0.9522	0.9545	0.9670
AUC (Test)	0.9920	0.9907	0.9956
Log-loss (Test)	0.1133	0.1382	0.0852
Training time (s)	50.164	144.83	1148.20
Reporte por clase (F1-score Test)			
Clase 0 (textile)	0.9512	0.9545	0.9669
Clase 1 (wood)	0.9532	0.9545	0.9672

Nota: La precisión de prueba reportada del modelo MLP corresponde al umbral de decisión óptimo (basado en F1, umbral = 0,6095).

Nota."Comparative Analysis of Techniques for Texture Feature Extraction for Supervised Classification of Wood and Textile Waste," por W. C. Francisco, J. V. Tavira, J. J. C. Vega, B. D. V. Robles, E. R. Tamariz y A. B. Ortega, 2026, *Recycling*, *11*(5), p. 86.

VIII. Discusión

Los resultados obtenidos en esta investigación presentaron el mejor desempeño mediante el modelo MLP en términos de exactitud (96.70 %), AUC y log-loss, favoreciendo la efectividad de los modelos clásicos cuando son alimentados con vectores de características altamente discriminativas. De manera similar Dao et al. (2025) analizaron el rendimiento de clasificadores en contextos generales, modelos híbridos de DL combinados con el algoritmo Harris Hawk Optimization, reportado la selección de características una precisión de 96.74 %. En cuanto al análisis por tipo de residuo, las propiedades físicas de textura evaluadas en este trabajo permitieron resultados idóneos de clasificación, la FD capturó la complejidad multiescala e irregularidad intrínseca características de la madera. A diferencia de enfoques como AdaptMoist, el cual logró un 80 % de exactitud promedio en la predicción de humedad en dicho sector mediante descriptores de textura (Rahman et al., 2025). Por otra parte, para el material textil, las herramientas GLCM, LBP y los detectores de bordes de Canny/Sobel resultaron ideales para modelar sus texturas homogéneas y periódicas. Estos hallazgos son semejantes con el contexto específico de la industria textil, donde combinaciones de descriptores (GLCM, GFB, LBP e IFV) junto con el algoritmo XGBoost alcanzaron un 98.7 % de precisión (Seckin et al., 2023).

En este estudio, el dataset se desarrolló bajo condiciones controladas con un mínimo ruido de fondo para facilitar la extracción de características textuales. Este enfoque metodológico de captura coincide con lo reportado por Seckin et al. (2023), quienes utilizaron 130 imágenes controladas bajo microscopio digital, así como con el banco WOOD-AUTH, el cual empleó 4,272 imágenes macroscópicas estables de secciones de madera (Sharafudeen et al., 2021). Por el contrario, el dataset utilizado contrasta con investigaciones que optan por conjuntos de datos masivos o multiclase; tal es el caso de las 35,264 imágenes de residuos utilizadas por Nahiduzzaman et al. (2025), o las 1,800 imágenes equilibradas de escombros de construcción (cerámica/baldosas, hormigón, basura/residuos y madera) de Alashram et al. (2025). Si bien la naturaleza controlada de nuestro entorno garantiza la estabilidad y evita el sesgo en el entrenamiento de los modelos de ML, se reconoce que esto representa una limitante de

generalización frente a escenarios urbanos reales.

En términos de costo computacional, esta investigación se fortalece como una base metodológica para la extracción y selección de características de textura, demostrando que el uso de técnicas clásicas permite alcanzar una óptima precisión disminuyendo significativamente la latencia. Esto resulta fundamental en entornos prácticos donde la viabilidad de un sistema de clasificación de RSU depende críticamente del balance entre la complejidad del algoritmo y la infraestructura de hardware requerida. Mientras que el enfoque de este trabajo optimiza el procesamiento lógico para reducir la carga computacional, otros autores han abordado esta problemática mediante la incorporación de hardware dedicado. Tal es el caso de Echeverry y López (2024), quienes emplearon una tarjeta Nvidia Jetson Nano de 2 GB utilizando la arquitectura SSD-Mobilenet-v2 para lograr precisiones del 94 % al 96 %, o de Dubey et al. (2020), quien implementó el algoritmo RF (85.29 % de precisión) directamente en una placa Raspberry Pi apoyada por sensores ultrasónicos y de gas.

IX. Conclusiones

Los pasos secuenciales realizados demuestran el cumplimiento de los objetivos establecidos; en primer lugar, se logró la integración del conjunto de datos, el cual fungió como la base experimental para el entrenamiento y validación. Posteriormente, se analizaron e implementaron las técnicas de extracción de textura, lo que hizo posible conformar un vector de valores estadísticos robusto para representar visualmente los residuos. En tercer lugar, mediante la selección de características, se optimizó dicho vector al eliminar los datos redundantes y priorizar los descriptores textuales con mayor poder discriminatorio. Finalmente, se entrenaron y contrastaron los tres modelos de ML con el vector de 16 características resultantes, donde la etapa de evaluación y ajuste de hiperparámetros permitió refinar el sistema hasta obtener los mejores resultados posibles.

La comparación de modelos de ML (SVM, RF, MLP) con el vector de valores estadísticos, derivado de las técnicas de extracción de características textuales (GLCM, LBP, HOG, FD, bordes Canny/Sobel), demostró que el algoritmo MLP se desempeñó mejor; los valores asignados a los hiperparámetros permitieron modelar adecuadamente las relaciones no lineales entre las características de textura, alcanzando una precisión de 96.70 % y un AUC de 99.56 %.

La experimentación demostró que al combinar las técnicas de texturas clásicas con modelos ML permite alcanzar, bajo condiciones controladas, un alto desempeño en la clasificación binaria de RSU (madera y textil), demostrando la hipótesis planteada. No obstante, los hallazgos deben interpretarse dentro del alcance experimental del estudio, por lo que la metodología propuesta se plantea como una línea base para la extracción de descriptores de texturas en entornos controlados, más que como una solución directamente aplicable a entornos reales. Como trabajo futuro, se propone ampliar el conjunto de datos incorporando condiciones más reales, incluyendo variabilidad en la imagen con fondo y materiales mixtos. También incorporar etapas de detección o segmentación con un enfoque basado en CNN antes de la clasificación, lo que podría mejorar la capacidad del sistema al aislar regiones de interés relevantes.

Este enfoque híbrido podría contribuir a reducir la brecha entre los entornos experimentales

controlados y los escenarios reales. Los resultados confirman que el análisis de textura sigue siendo una herramienta eficaz para la clasificación de materiales; sin embargo, su efectividad en aplicaciones reales depende de su capacidad para adaptarse a condiciones ambientales variables y escenarios complejos, con el objetivo de avanzar hacia el desarrollo de herramientas tecnológicas aplicables en contextos reales para la gestión de RSU.

El aporte de esta investigación radica en que se establece una línea base para futuras investigaciones enfocadas en la gestión de RSU y contribuye al desarrollo de modelos de visión artificial orientados a su clasificación automatizada. Mediante la selección de características de textura más discriminativas extraídas a partir de las técnicas clásicas, se logró la clasificación de RSU de madera y textil en escenarios controlados a través de sus texturas. El modelo basado en MLP requirió un tiempo de entrenamiento de 1,148.20 segundos, demostrando la viabilidad operativa y el bajo costo computacional del sistema. Esta propuesta aporta una metodología transferible para optimizar los procesos de recuperación y aprovechamiento de estos materiales dentro del marco de la economía circular, favoreciendo así la sostenibilidad ambiental

X. Referencias

Abdallah, M., Abu Talib, M., Feroz, S., Nasir, Q., Abdalla, H., & Mahfood, B. (2020). Artificial intelligence applications in solid waste management: A systematic research review. *Waste Management, 109*, 231–246. https://doi.org/10.1016/j.wasman.2020.04.057

Alashram, O., Alagha, N., AlKakuri, M., Swaveel, Z., & Copiaco, A. (2025). Hybrid Deep Feature Extraction and ML for Construction and Demolition Debris Classification. *Proceedings of the 2025 8th International Conference on Signal Processing and Information Security (ICSPIS)*, 1–5. https://doi.org/10.1109/icspis67605.2025.11318390

Altmann, A., Toloşi, L., Sander, O., & Lengauer, T. (2010). Permutation importance: a corrected feature importance measure. *Bioinformatics, 26*(10), 1340–1347. https://doi.org/10.1093/bioinformatics/btq134

Ayeleru, O. O., Fajimi, L. I., Oboirien, B. O., & Olubambi, P. A. (2021). Forecasting municipal solid waste quantity using artificial neural network and supported vector machine techniques: A case study of Johannesburg, South Africa. *Journal of Cleaner Production, 289*, 125671. https://doi.org/10.1016/j.jclepro.2020.125671

Bieringer, A., Wysocki, O., Tuttas, S., Hoegner, L., & Holst, C. (2024). Analyzing the impact of semantic LoD3 building models on image-based vehicle localization. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, X-4/W5-2024*, 55–62. https://doi.org/10.5194/isprs-annals-X-4-W5-2024-55-2024

Boser, B., Guyon, I., & Vapnik, V. (1996). A Training Algorithm for Optimal Margin Classifier. *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory, 5*. https://doi.org/10.1145/130385.130401

Breiman, L. (2001). Random Forests. *Machine Learning, 45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Canny, J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8*(6), 679–698. https://doi.org/

10.1109/TPAMI.1986.4767851

Castro-Bello, M., Roman-Padilla, D. B., Morales-Morales, C., Campos-Francisco, W., Marmolejo-Vega, C. V., Marmolejo-Duarte, C., Evangelista-Alcocer, Y., & Gutiérrez-Valencia, D. E. (2025). Convolutional Neural Network Models in Municipal Solid Waste Classification: Towards Sustainable Management. *Sustainability, 17*(8), 3523. https://doi.org/10.3390/su17083523

Castro-Bello, M., Romero-Juárez, V. M., Fuentes-Pacheco, J., Morales-Morales, C., Marmolejo-Vega, C. V., Zagal-Barrera, S. R., Gutiérrez-Valencia, D. E., & Marmolejo-Duarte, C. (2025). R3sNet: Optimized Residual Neural Network Architecture for the Classification of Urban Solid Waste via Images. *Sustainability, 17*(8), 3502. https://doi.org/10.3390/su17083502

Chaki, J., & Dey, N. (2020). *Texture Feature Extraction Techniques for Image Recognition*. Springer Singapore. https://doi.org/10.1007/978-981-15-0853-0

Cheng, G., Chen, J., Wei, Y., Chen, S., & Pan, Z. (2023). A Coal Gangue Identification Method Based on HOG Combined with LBP Features and Improved Support Vector Machine. *Symmetry, 15*(1), 202. https://doi.org/10.3390/sym15010202

Cheng, T., Kojima, D., Hu, H., Onoda, H., & Pandyaswargo, A. H. (2024). Optimizing Waste Sorting for Sustainability: An AI-Powered Robotic Solution for Beverage Container Recycling. *Sustainability, 16*(23), 10155. https://doi.org/10.3390/su162310155

Chu, Y., Huang, C., Xie, X., Tan, B., Kamal, S., & Xiong, X. (2018). Multilayer Hybrid Deep-Learning Method for Waste Classification and Recycling. *Computational Intelligence and Neuroscience, 2018*(1), 5060857. https://doi.org/10.1155/2018/5060857

Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., & Vedaldi, A. (2014). Describing Textures in the Wild. *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.https://openaccess.thecvf.com/content_cvpr_2014/html/Cimpoi_Describing_Textures_in_2014_CVPR_paper.html

Cámara de Diputados. (2023). *LEY GENERAL PARA LA PREVENCIÓN Y GESTIÓN INTEGRAL DE LOS RESIDUOS*. https://www.diputados.gob.mx/LeyesBiblio/pdf/LGPGIR.pdf

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), 1*, 886–893 vol. 1. https://doi.org/10.1109/CVPR.2005.177

Dao, S. V. T., Le, T. M., Tran, H. M., Pham, H. V., Vu, M. T., & Chu, T. (2025). Integrating artificial intelligence for sustainable waste management: Insights from machine learning and deep learning. *Watershed Ecology and the Environment, 7*, 353–382. https://doi.org/10.1016/j.wsee.2025.07.001

Dubey, S., Singh, M. K., Singh, P., & Aggarwal, S. (2020). Waste Management of Residential Society using Machine Learning and IoT Approach. *2020 International Conference on Emerging Smart Computing and Informatics (ESCI)*, 293–297. https://doi.org/10.1109/ESCI48226.2020.9167526

Echeverry, A. P., & López, C. F. (2024). AUTORECYCLER: Prototype based on artificial vision to automate the material classification process (Plastic, Glass, Cardboard and Metal). *HardwareX, 19*, e00575. https://doi.org/10.1016/j.ohx.2024.e00575

Francisco, W. C., Tavira, J. V., Vega, J. J. C., Robles, B. D. V., Tamariz, E. R., & Ortega, A. B. (2026). Comparative Analysis of Techniques for Texture Feature Extraction for Supervised Classification of Wood and Textile Waste. *Recycling*, *11*(5), 86. https://doi.org/10.3390/recycling11050086

Fotovvatikhah, F., Ahmedy, I., Noor, R. M., & Munir, M. U. (2025). A Systematic Review of AI-Based Techniques for Automated Waste Classification. *Sensors, 25*(10), 3181. https://doi.org/10.3390/s25103181

Gil-Arroyo, B., Sanz, J. M., Arroyo, Á., Urda, D., Basurto, N., & Herrero, Á. (2025). Dataset for defect detection in textile manufacturing. *Data in Brief, 59*, 111451. https://doi.org/10.1016/j.dib.2025.111451

González, S. G., García, D. C., Pérez, R. H., Sanchez, A. Á., & González, G. V. (2025). Autonomous Waste Classification Using Multi-Agent Systems and Blockchain: A Low-Cost Intelligent Approach. *Sensors, 25*(14), 4364. https://doi.org/10.3390/s25144364

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org

Huang, D., Shan, C., Ardabilian, M., Wang, Y., & Chen, L. (2011). Local Binary Patterns and Its Application to Facial Image Analysis: A Survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 41*(6), 765–781. https://doi.org/10.1109/TSMCC.2011.2118750

IBM. (2023). *Deep Learning*. https://www.ibm.com/mx-es/topics/deep-learning

INEGI. (2025). *Estadísticas a Propósito del Día Mundial del Medio Ambiente (5 de Junio): Datos Nacionales* (Technical Report). Instituto Nacional de Estadística y Geografía, Aguascalientes, México. https://www.inegi.org.mx/contenidos/saladeprensa/aproposito/2025/EAP_MedioAmb_25.pdf

Intel. (2024). *CPU o GPU: opciones interesantes para sus necesidades informáticas*. https://www.intel.la/content/www/xl/es/products/docs/processors/cpu-vs-gpu.html

Jahan, I., Zhang, G., Bhuiyan, M., & Navaratnam, S. (2022). Circular Economy of Construction and Demolition Wood Waste—A Theoretical Framework Approach. *Sustainability, 14*(17), 10478. https://doi.org/10.3390/su141710478

Kaya, V. (2023). Classification of waste materials with a smart garbage system for sustainable development: a novel model. *Frontiers in Environmental Science, 11*. https://doi.org/10.3390/fenvs.2023.1228732

Kaza, S., Yao, L., Bhada-Tata, P., & Van Woerden, F. (2018). *What a waste 2.0: a global snapshot of solid waste management to 2050*. World Bank Publications.

Khan, A. R., Saba, T., Sadad, T., Nobanee, H., & Bahaj, S. A. (2022). Identification of Anomalies in Mammograms through Internet of Medical Things (IoMT) Diagnosis System. *Computational Intelligence and Neuroscience, 2022*, 1100775. https://doi.org/10.1155/2022/1100775

Kodytek, P., Bodzas, A., & Bilik, P. (2022). A large-scale image dataset of wood surface defects for automated vision-based quality control processes. *F1000Research, 10*(581). https://doi.org/10.12688/f1000research.52903.2

Kurniati, F. T., Manongga, D. H., Sembiring, I., Wijono, S., & Huizen, R. R. (2024). The object detection model uses combined extraction with KNN and RF classification. *Indonesian Journal of Electrical Engineering and Computer Science, 35*(1), 436–445. http://dx.doi.org/10.11591/ijeecs.v35.i1.pp436-445

Liang, S., & Gu, Y. (2021). A deep convolutional neural network to simultaneously localize and recognize waste types in images. *Waste Management, 126*, 247–257. https://doi.org/10.1016/j.wasman.2021.03.017

Luts, J., Ojeda, F., Van de Plas, R., De Moor, B., Van Huffel, S., & Suykens, J. A. K. (2010). A tutorial on support vector machine-based methods for classification problems in chemometrics. *Analytica Chimica Acta, 665(2)*, 129–145. https://doi.org/10.1016/j.aca.2010.03.030

Manakkakudy Kumaran, A., De Iacovo, A., Ballabio, A., Frigerio, J., Isella, G., & Colace, L. (2024). Waste Material Classification Based on a Wavelength-Sensitive Ge-on-Si Photodetector. *Sensors, 24(21)*, 6970. https://doi.org/10.3390/s24216970

MathWorks. (2024). *¿Qué son las redes neuronales?*. https://la.mathworks.com/discovery/neural-network.html

meraldanma. (2024). *Textile 2 Dataset*. Roboflow Universe. https://universe.roboflow.com/meraldanma/textile-2

Merino, I., Azpiazu, J., Remazeilles, A., & Sierra, B. (2020). Histogram-Based Descriptor Subset Selection for Visual Recognition of Industrial Parts. *Applied Sciences, 10*(11), 3701.

https://doi.org/10.3390/app10113701

Mirhashemi, A. (2018). Introducing spectral moment features in analyzing the SpecTex hyperspectral texture database. *Machine Vision and Applications, 29*(3), 415–432. https://doi.org/10.1007/s00138-017-0892-9

Mythili, T., & Anbarasi, A. (2022). A concatenation of deep and texture features for medicinal trash image classification using EnSegNet-DNN-based transfer learning. *Materials Today: Proceedings, 62*, 4691–4698. https://doi.org/10.1016/j.matpr.2022.03.129

Nahiduzzaman, M., Ahamed, M. F., Naznine, M., Karim, M. J., Kibria, H. B., Ayari, M. A., Khandakar, A., Ashraf, A., Ahsan, M., & Haider, J. (2025). An automated waste classification system using deep learning techniques: Toward efficient waste recycling and environmental sustainability. *Knowledge-Based Systems, 310*, 113028. https://doi.org/10.1016/j.knosys.2025.113028

Neamah, O. N. (2024). *Fabric Defects Object Detection Dataset*. Figshare. https://doi.org/10.6084/m9.figshare.25546465.v2

Nežerka, V., Zbíral, T., & Trejbal, J. (2024). Machine-learning-assisted classification of construction and demolition waste fragments using computer vision: Convolution versus extraction of selected features. *Expert Systems with Applications, 238*, 121568. https://doi.org/10.1016/j.eswa.2023.121568

Nguyen, N. B. Q., Do, T. M., Phan, C. T., & Phan, T. T. H. (2025). Towards Accurate and Efficient Waste Image Classification: A Hybrid Deep Learning and Machine Learning Approach. *arXiv e-prints*, arXiv:2510.21833. https://arxiv.org/abs/2510.21833

Nikitin, N. Y., & Stepashkin, A. A. (2025). Classification of tensile test results of unidirectional carbon fiber-polysulfone composite material based on random forest, KNN and CNN methods. *Results in Materials, 28*, 100788. https://doi.org/10.1016/j.rinma.2025.100788

NVIDIA. (2023). *GPU-Accelerated Applications: TensorFlow*. https://www.nvidia.com/

en-sg/data-center/gpu-accelerated-applications/tensorflow/

ONU México. (2025). *Objetivos de Desarrollo Sostenible*. https://mexico.un.org/es/sdgs

Panigrahy, C., Garcia-Pedrero, A., Seal, A., Rodríguez-Esparragón, D., Mahato, N. K., & Gonzalo-Martín, C. (2017). An Approximated Box Height for Differential-Box-Counting Method to Estimate Fractal Dimensions of Gray-Scale Images. *Entropy, 19*(10), 534. https://doi.org/10.3390/e19100534

Panigrahy, C., Seal, A., Mahato, N. K., & Bhattacharjee, D. (2019). Differential box counting methods for estimating fractal dimension of gray-scale images: A survey. *Chaos, Solitons & Fractals, 126*, 178–202. https://doi.org/10.1016/j.chaos.2019.06.007

Piccioni Costa, L., Guerreiro, M., Puchta, E., Tadano, Y., Antonini Alves, T., Kaster, M., & Siqueira, H. (2023). *Multilayer Perceptron*. Zenodo. https://doi.org/10.5281/zenodo.7562924

Pillai, K. S., M L, S., S, A., Anand, A. B., & Prasad, G. (2023). Municipal Solid Waste Management: A Review of Machine Learning Applications. *E3S Web Conferences, 455*, 02018. https://doi.org/10.1051/e3sconf/202345502018

PNUMA. (2024). *Perspectiva Mundial de la Gestión de Residuos 2024*. https://www.unep.org/es/resources/perspectiva-mundial-de-la-gestion-de-residuos-2024

Pučnik, R., Dokl, M., Fan, Y. V., Vujanović, A., Novak Pintarič, Z., Aviso, K. B., Tan, R. R., Pahor, B., Kravanja, Z., & Čuček, L. (2024). A waste separation system based on sensor technology and deep learning: A simple approach applied to a case study of plastic packaging waste. *Journal of Cleaner Production, 450*, 141762. https://doi.org/10.1016/j.jclepro.2024.141762

Python. (2024). *El tutorial de Python*. https://docs.python.org/es/3.13/tutorial/index.html

Rahman, A., Marufuzzaman, M., Street, J., Wang, H., Gude, V. G., & Buchanan, R. (2025).

Robust Cross-Domain Adaptation in Texture Features Transferring for Wood Chip Moisture Content Prediction. *arXiv e-prints*, arXiv:2510.16832. https://arxiv.org/abs/2510.16832

Ramos, E., Lopes, A. G., & Mendonça, F. (2024). Application of Machine Learning in Plastic Waste Detection and Classification: A Systematic Review. *Processes, 12*(8), 1632. https://doi.org/10.3390/pr12081632

Salem, K.S.; Clayson, K.; Salas, M.; Haque, N.; Rao, R.; Agate, S.; Singh, A.; Levis, J.W.; Mittal, A.; Yarbrough, J.M.; et al. "A critical review of existing and emerging technologies and systems to optimize solid waste management for feedstocks and energy conversion," *Matter*, vol. 6, no. 10, pp. 3348–3377, 2023. https://doi.org/10.1016/j.matt.2023.08.003

Sanjaya, S., Adzkia, U., Handayani, L., & Yanto, F. (2020). Local Binary Pattern and Learning Vector Quantization for Classification of Principal Line of Palm-Hand. *Indonesian Journal of Artificial Intelligence and Data Mining, 3*(2), 71–77. https://doi.org/10.24014/ijaidm.v3i2.10236

Santoso, H., Hanif, I., Magdalena, H., & Afiyati, A. (2024). A Hybrid Model for Dry Waste Classification using Transfer Learning and Dimensionality Reduction. *JOIV: International Journal on Informatics Visualization, 8*(2), 623–634. http://dx.doi.org/10.62527/joiv.8.2.1943

Seckin, M., Seckin, A. C., Demircioglu, P., & Bogrekci, I. (2023). FabricNET: A Microscopic Image Dataset of Woven Fabrics for Predicting Texture and Weaving Parameters through Machine Learning. *Sustainability, 15*(21), 15197. https://doi.org/10.3390/su152115197

SEMARNAT. (2015). *Guía de buenas prácticas : para el reciclaje de los residuos textiles y de calzado en Cataluña (recurso electrónico)*. https://biblioteca.semarnat.gob.mx/janium/Documentos/Ciga/Libros2011/CD003675.pdf

SEMARNAT. (2017). *Secretaría de Medio Ambiente y Recursos Naturales, Residuos Sólidos*

Urbanos (RSU). https://www.gob.mx/semarnat/acciones-y-programas/residuos-solidos-urbanos-rsu

SEMARNAT. (2020). *Secretaría de Medio Ambiente y Recursos Naturales, Diagnóstico Básico para la Gestión Integral de los Residuos*. https://www.gob.mx/cms/uploads/attachment/file/554385/DBGIR-15-mayo-2020.pdf

Shafek, D., Hilow, H. W., & Ahmed, M. (2025). Clasificación de imágenes de residuos mediante aprendizaje profundo. *Revista de Investigación Aplicada y Tecnología, 23*(4), 381–391. https://doi.org/10.22201/icat.24486736e.2025.23.4.2578

Shakir, S., & Topal, C. (2022). *Ten Fabrics Dataset (TFD)*. Kaggle. https://doi.org/10.34740/KAGGLE/DSV/3012635

Sharafudeen, M., Manju, R. A., & Simon, P. (2021). Wood texture classification using GLCM and LBP. En *Proceedings of Sinems Workplace Conference* (Vol. 2349, p. 6002).

Sinems Workplace. (2025). *DefectDetectionwk2 Classification Dataset*. Roboflow Universe. https://universe.roboflow.com/sinems-workplace-gzxdr/defectdetectionwk2

Su, L., Dong, X., Peng, J., Cheng, H., Craig, N. J., Hu, B., & Li, J. Y. (2024). Segmentation of beach plastic fragments' contours based on self-organizing map and multi-shape descriptors: A rapid indication of fragmentation and wearing types. *Journal of Hazardous Materials, 478*, 135564. https://doi.org/10.1016/j.jhazmat.2024.135564

Sumaya, Islam, F., Monir, M. F., & Islam, A. (2024). *FabricSpotDefect: An Annotated Dataset for Identifying Spot Defects in Different Fabric Types*. Mendeley Data. https://doi.org/10.17632/6574nhzm8x.1

Sumsion, G. R., Bradshaw, M. S., Hill, K. T., Pinto, L. D. G., & Piccolo, S. R. (2019). Remote sensing tree classification with a multilayer perceptron. *PeerJ, 7*, e6101. https://doi.org/10.7717/peerj.6101

Sun, Y., Wang, J., Wang, T., Li, J., Wei, Z., Fan, A., Liu, H., Chen, S., Zhang, Z., Chen, Y., & Huang, L. (2024). Post-Fracture Production Prediction with Production Segmentation

and Well Logging: Harnessing Pipelines and Hyperparameter Tuning with GridSearchCV. *Applied Sciences, 14*(10), 3954. https://doi.org/10.3390/app14103954

Umar, N. (2026). Type Deep Learning Model for Multi-Label Waste Classification in Canal Environments: A Comparative Study with CNN Architectures. *Journal of Applied Data Sciences, 7*, 261–276. https://doi.org/10.47738/jads.v7i1.1066

Usuario defect-8n3b5. (2025). *Dataset de Segmentación: Proyecto segment-vx0be*. Roboflow Universe. https://universe.roboflow.com/defect-8n3b5/segment-vx0be

Utaminingrum, F., Alqadri, A. M., Somawirata, I. K., Karim, C., Septiarini, A., Lin, C. Y., & Shih, T. K. (2023). Feature selection of gray-level Cooccurrence matrix using genetic algorithm with Extreme learning machine classification for early detection of Pole roads. *Results in Engineering, 20*, 101437. https://doi.org/10.1016/j.rineng.2023.101437

Vercalsteren, A., Nicolau, M., & Lafond, E. (2019). Textiles and the environment in a circular economy. *ETC/WMGE: Copenhagen, Denmark*. https://vito.be/sites/vito/files/upload/etc-wmge_report_textiles_2020.pdf

Vincent, O., & Folorunso, O. (2009). A Descriptive Algorithm for Sobel Image Edge Detection. *Proceedings of the Sobel Image Edge Detection Project*. https://doi.org/10.28945/3351

Yang, Y., Luo, Y., Yang, Y., & Kang, S. (2025). Dynamic Graph Neural Network for Garbage Classification Based on Multimodal Feature Fusion. *Applied Sciences, 15*(14), 7688. https://doi.org/10.3390/app15147688

Yenuargo, D., Fatchan, M., & Hadikristanto, W. (2024). Valuation of svm kernel performance in organic and non-organic waste classification. *International Journal of Integrated Science and Technology, 2*(5), 1–12. https://doi.org/10.59890/ijist.v2i5.1873

Zhang, H. (2002). Introduction to Content–Based Image Indexing and Retrieval. En *Readings in Multimedia Computing and Networking* (pp. 247–253). Morgan Kaufmann. https://doi.org/10.1016/B978-155860651-7/50107-8

Zhang, Y. D., Chen, X. Q., Zhan, T. M., Jiao, Z. Q., Sun, Y., Chen, Z. M., Yao, Y., Fang, L. T., Lv, Y. D., & Wang, S. H. (2016). Fractal Dimension Estimation for Developing Pathological Brain Detection System Based on Minkowski-Bouligand Method. *IEEE Access, 4*, 5937–5947. https://doi.org/10.1109/ACCESS.2016.2611530

Zhong, S., Zhang, K., Bagheri, M., Burken, J. G., Gu, A., Li, B., Ma, X., Marrone, B. L., Ren, Z. J., Schrier, J., Shi, W., Tan, H., Wang, T., Wang, X., Wong, B. M., Xiao, X., Yu, X., Zhu, J. J., & Zhang, H. (2021). Machine Learning: New Ideas and Tools in Environmental Science and Engineering. *Environmental Science & Technology, 55*(19), 12741–12754. https://doi.org/10.1021/acs.est.1c01339

Zhou, Y., Wang, Z., Zheng, S., Zhou, L., Dai, L., Luo, H., Sui, M. (2024). Optimization of automated garbage recognition model based on ResNet-50 and weakly supervised CNN for sustainable urban development. *Alexandria Engineering Journal, 108*, 415–427. https://doi.org/10.1016/j.aej.2024.07.066

Zoumpoulis, P., Konstantinidis, F. K., Tsimiklis, G., & Amditis, A. (2024). Smart bins for enhanced resource recovery and sustainable urban waste practices in smart cities: A systematic literature review. *Cities, 152*, 105150. https://doi.org/10.1016/j.cities.2024.105150

XI. Anexos

Anexo 1. Código GLCM

```
        import os, glob
        from pathlib import Path
        import numpy as np
        import pandas as pd
        import cv2
        from skimage.feature import graycomatrix, graycoprops
        INPUT_DIR = r"DataSedWT/wood"
        OUT_CSV   = "DataSedWT/resultados/features_glcm_wood.csv"
        DISTANCES = [1, 2, 3]
        ANGLES_DEG = [0, 45, 90, 135]
        ANGLES_RAD = [np.deg2rad(a) for a in ANGLES_DEG]
        GLCM_PROPS = ["contrast", "correlation", "energy", "homogeneity",
        "dissimilarity", "ASM"]
        LEVELS = 8

        def list_images(input_dir: str) -> list[str]:
        paths = []
        if input_dir and os.path.isdir(input_dir):
        for ext in ("*.jpg","*.jpeg","*.png","*.bmp","*.tif","*.tiff"):
        paths.extend(glob.glob(os.path.join(input_dir, ext)))
        paths = sorted(list(dict.fromkeys(paths)))
        return paths

        def read_gray_8bit(path: str) -> np.ndarray:
        img = cv2.imdecode(np.fromfile(path, dtype=np.uint8),
        cv2.IMREAD_UNCHANGED)
        if img is None:
        raise ValueError(f"No se pudo leer: {path}")
        if img.ndim == 3 and img.shape[2] == 4:
        img = cv2.cvtColor(img, cv2.COLOR_BGRA2BGR)
        if img.ndim == 3:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        if img.dtype != np.uint8:
        img = cv2.normalize(img, None, 0, 255, cv2.NORM_MINMAX)
        img = img.astype(np.uint8)
        return img

        def quantize_levels(gray_u8: np.ndarray, levels: int = 8)
        -> np.ndarray:
        if levels <= 1:
        return np.zeros_like(gray_u8)
        bins = np.linspace(0, 256, levels+1, endpoint=True).astype(np.int32)
        q = np.digitize(gray_u8, bins[:-1], right=False) - 1
        q = np.clip(q, 0, levels-1).astype(np.uint8)
        return q

        def glcm_features(gray_levels: np.ndarray,
        distances=DISTANCES, angles=ANGLES_RAD,
        props=GLCM_PROPS, levels=LEVELS) -> dict:
        glcm = graycomatrix(gray_levels,
        distances=distances,
        angles=angles,
        levels=levels,
        symmetric=True,
```

```
        normed=True)
        feats = {}
        for p in props:
        val = graycoprops(glcm, p)
        feats[f"{p.capitalize()}_mean"] = float(val.mean())
        return feats

        def infer_label_from_name(path: str) -> str:
        name = Path(path).name.lower()
        if "wood" in name or "madera" in name:
        return "wood"
        if "textile" in name or "textil" in name or "fabric" in name:
        return "textile"
        return "unknown"

        all_paths = list_images(INPUT_DIR)
        print(f"Imágenes encontradas: {len(all_paths)}")
        if len(all_paths) == 0:
        raise SystemExit("No se encontraron imágenes.
        Verifica INPUT_DIR o FILE_PATTERNS.")

        rows = []
        for p in all_paths:
        try:
        gray = read_gray_8bit(p)
        q = quantize_levels(gray, LEVELS)
        feats = glcm_features(q, DISTANCES, ANGLES_RAD, GLCM_PROPS, LEVELS)
        rows.append({
            "name": Path(p).stem,
            "label": infer_label_from_name(p),
            **feats
        })
        except Exception as e:
        print(f"[WARN] {p}: {e}")
                df = pd.DataFrame(rows)
        df

        df.to_csv(OUT_CSV, index=False, encoding="utf-8")
        print(f"Archivo guardado: {OUT_CSV}\nFilas: {len(df)}
        | Columnas: {len(df.columns)}")

```

Listing 1: Script de extracción de características con la técnica GLCM.

Anexo 2. Código LBP

```
import os, glob
from pathlib import Path
import numpy as np
import pandas as pd
import cv2
from scipy.stats import entropy, skew, kurtosis
from skimage.feature import local_binary_pattern

BASE_DIR = r"DataSedWT/textile"
OUT_CSV  = "DataSedWT/resultados/features_lbp_textile_stats.csv"
R = 3
P = 16 * R
METHOD = "uniform"          # 'uniform' => N_BINS = P + 2
N_BINS = P + 2

def list_images_flat(base_dir):
items = []
for ext in ("*.jpg", "*.jpeg", "*.png", "*.bmp", "*.tif", "*.tiff"):
for p in glob.glob(os.path.join(base_dir, ext)):
label = "wood" if "wood" in Path(p).stem.lower() else (
"textile" if "textil" in Path(p).stem.lower() else "unknown")
items.append((p, label))
return sorted(items)

def read_gray_u8(path):
img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
if img is None:
raise ValueError(f"No se pudo leer: {path}")
img = cv2.normalize(img, None, 0, 255, cv2.NORM_MINMAX)
.astype(np.uint8)
return img

def lbp_features(gray, P, R, method=METHOD, n_bins=None):
if n_bins is None:
n_bins = (P + 2) if method == "uniform" else int(2**P)
lbp = local_binary_pattern(gray, P=P, R=R, method=method)
hist_counts, _ = np.histogram(lbp.ravel(), bins=n_bins,
range=(0, n_bins), density=False)
total = hist_counts.sum()
if total == 0:
p = np.zeros_like(hist_counts, dtype=np.float64)
else:
p = hist_counts.astype(np.float64) / total
feats = {
    "entropy": float(entropy(p, base=2)),
    "energy": float(np.sum(p**2)),
    "skew": float(skew(p)),
    "kurtosis": float(kurtosis(p)),
    "std": float(np.std(p)),
    "nonzero_bins": int(np.count_nonzero(hist_counts)),
    "num_pixels": int(lbp.size),
}
return feats

pairs = list_images_flat(BASE_DIR)
print(f"Imágenes encontradas: {len(pairs)}")
```

```
        rows = []
        for path, label in pairs:
        try:
        gray = read_gray_u8(path)
        feats = lbp_features(gray, P=P, R=R, method=METHOD, n_bins=N_BINS)
        rows.append({
            "name": Path(path).stem,
            "label": label,
            **feats
        })
        except Exception as e:
        print(f"[WARN] {path}: {e}")
        df_lbp = pd.DataFrame(rows)
        df_lbp.head()

        df_lbp.to_csv(OUT_CSV, index=False, encoding="utf-8")
        print(f"Archivo guardado: {OUT_CSV} | filas: {len(df_lbp)}")
        if "label" in df_lbp.columns and df_lbp["label"].nunique() > 1:
        display(df_lbp.groupby("label")[["entropy","energy","skew","kurtosis",
        "std","nonzero_bins","num_pixels"]].mean().round(5))

```

Listing 2: Script de extracción de características con la técnica LBP

Anexo 3. Código HOG

```
        import os, glob
        from pathlib import Path
        import numpy as np
        import pandas as pd
        import cv2
        from scipy.stats import entropy
        from skimage.feature import hog

        BASE_DIR = r"DataSedWT/wood"
        OUT_CSV  = "DataSedWT/features_hog_wood_test.csv"

        IMG_SIZE   = (256, 256)
        USE_CLAHE  = True
        CLAHE_CLIP = 2.0
        CLAHE_TILE = (8, 8)

        HOG_ORIENTATIONS    = 9
        HOG_PIXELS_PER_CELL = (16, 16)
        HOG_CELLS_PER_BLOCK = (2, 2)
        HOG_BLOCK_NORM      = 'L2-Hys'
        HOG_TRANSFORM_SQRT  = True
        HOG_VISUALIZE       = False

        def infer_label_from_name(path: str) -> str:
        stem = Path(path).stem.lower()
        if ("wood" in stem) or ("madera" in stem):
        return "wood"
        if ("textil" in stem) or ("textile" in stem) or ("fabric" in stem):
        return "textile"
        return "unknown"

        def list_images_flat(base_dir: str):
```

```
        items = []
        for ext in ("*.jpg","*.jpeg","*.png","*.bmp","*.tif","*.tiff"):
        items.extend(glob.glob(os.path.join(base_dir, ext)))
        return sorted(items)

        def read_gray_u8(path: str) -> np.ndarray:
        img = cv2.imdecode(np.fromfile(path, dtype=np.uint8),
        cv2.IMREAD_UNCHANGED)
        if img is None:
        raise ValueError(f"No se pudo leer: {path}")
        if img.ndim == 3:
        if img.shape[2] == 4:
        img = cv2.cvtColor(img, cv2.COLOR_BGRA2BGR)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        if img.dtype != np.uint8:
        img = cv2.normalize(img, None, 0, 255, cv2.NORM_MINMAX)
        .astype(np.uint8)
        return img

        def apply_clahe(gray_u8: np.ndarray, clip=2.0, tile=(8,8))
        -> np.ndarray:
        clahe = cv2.createCLAHE(clipLimit=clip, tileGridSize=tile)
        return clahe.apply(gray_u8)

        def hog_stats(vec: np.ndarray) -> dict:
        v = vec.astype(np.float64)
        mean_HOG   = float(np.mean(v))
        std_HOG    = float(np.std(v))
        energy_HOG = float(np.sum(v**2))
        p = np.abs(v)
        s = p.sum()
        p = p / s if s > 0 else p
        entropy_HOG = float(entropy(p, base=2)) if s > 0 else 0.0
        return {
            "mean_HOG": mean_HOG,
            "std_HOG": std_HOG,
            "energy_HOG": energy_HOG,
            "entropy_HOG": entropy_HOG
        }

        def hog_vector(gray_u8: np.ndarray) -> np.ndarray:
        if IMG_SIZE is not None:
        gray_u8 = cv2.resize(gray_u8, (IMG_SIZE[1], IMG_SIZE[0]),
        interpolation=cv2.INTER_AREA)
        if USE_CLAHE:
        gray_u8 = apply_clahe(gray_u8, CLAHE_CLIP, CLAHE_TILE)
        gray_f = (gray_u8.astype(np.float32) / 255.0)
        feat = hog(
        gray_f,
        orientations=HOG_ORIENTATIONS,
        pixels_per_cell=HOG_PIXELS_PER_CELL,
        cells_per_block=HOG_CELLS_PER_BLOCK,
        block_norm=HOG_BLOCK_NORM,
        transform_sqrt=HOG_TRANSFORM_SQRT,
        visualize=HOG_VISUALIZE,
        feature_vector=True
        )
        return feat.astype(np.float64)

```

```
        paths = list_images_flat(BASE_DIR)
        print(f"Imágenes encontradas: {len(paths)}")
        if len(paths) == 0:
        raise SystemExit("No se encontraron imágenes. Verifica BASE_DIR.")
        rows = []
        hog_dim = None
        for p in paths:
        try:
        gray = read_gray_u8(p)
        vec  = hog_vector(gray)
        stats = hog_stats(vec)
        hog_dim = len(vec) if hog_dim is None else hog_dim
        rows.append({
            "name": Path(p).stem,
            "label": infer_label_from_name(p),
            "hog": vec,
            **stats
        })
        except Exception as e:
        print(f"[WARN] {p}: {e}")
        df = pd.DataFrame(rows)
        df.to_csv(OUT_CSV, index=False, encoding="utf-8")
        print(f"Guardado: {OUT_CSV} | filas: {len(df)}
        | cols: {len(df.columns)}")


```

Listing 3: Script de extracción de características con la técnica HOG

Anexo 4. Código Canny/Sobel

```
        import os, glob
        from pathlib import Path
        import numpy as np
        import pandas as pd
        import cv2

        BASE_DIR = r"DataSedWT/wood"
        OUT_CSV  = "DataSedWT/resultados/features_edges_wood_SobelCandy.csv"
        USE_CLAHE  = True
        CLAHE_CLIP = 2.0
        CLAHE_TILE = (8, 8)
        CANNY_LOW  = 50
        CANNY_HIGH = 150
        GAUSS_BLUR = True
        BLUR_KSIZE = (3, 3)
        SOBEL_KSIZE = 3


        def infer_label_from_name(path: str) -> str:
        stem = Path(path).stem.lower()
        if ("wood" in stem) or ("madera" in stem):
        return "wood"
        if ("textil" in stem) or ("textile" in stem) or ("fabric" in stem):
        return "textile"
        return "unknown"
        def list_images_flat(base_dir: str):
        items = []
```

```
        for ext in ("*.jpg","*.jpeg","*.png","*.bmp","*.tif","*.tiff"):
        items.extend(glob.glob(os.path.join(base_dir, ext)))
        return sorted(items)
        def read_gray_u8(path: str) -> np.ndarray:
        img = cv2.imdecode(np.fromfile(path, dtype=np.uint8),
        cv2.IMREAD_UNCHANGED)
        if img is None:
        raise ValueError(f"No se pudo leer: {path}")
        if img.ndim == 3:
        if img.shape[2] == 4:
        img = cv2.cvtColor(img, cv2.COLOR_BGRA2BGR)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        if img.dtype != np.uint8:
        img = cv2.normalize(img, None, 0, 255, cv2.NORM_MINMAX)
        .astype(np.uint8)
        return img
        def apply_clahe(gray_u8: np.ndarray, clip=2.0, tile=(8,8))
        -> np.ndarray:
        clahe = cv2.createCLAHE(clipLimit=clip, tileGridSize=tile)
        return clahe.apply(gray_u8)

        from scipy.stats import entropy as shannon_entropy
        def sobel_maps(gray: np.ndarray, ksize=3):
        sx = cv2.Sobel(gray, cv2.CV_32F, 1, 0, ksize=ksize)
        sy = cv2.Sobel(gray, cv2.CV_32F, 0, 1, ksize=ksize)
        mag = cv2.magnitude(sx, sy)
        ang = cv2.phase(sx, sy, angleInDegrees=False)  # 0..pi
        return sx, sy, mag, ang
        def sobel_features(gray: np.ndarray, ksize=3, bins_dir=18) -> dict:
        _, _, mag, ang = sobel_maps(gray, ksize=ksize)
        mag_f = mag.astype(np.float64)
        h, w = gray.shape[:2]
        npx = h * w
        mag_mean = float(mag_f.mean())
        mag_std  = float(mag_f.std())
        mag_p95  = float(np.percentile(mag_f, 95))
        energy   = float(np.sum(mag_f**2) / (npx if npx > 0 else 1))
        ang_flat = ang.ravel()
        mag_flat = mag_f.ravel()
        ang_bins = np.linspace(0.0, np.pi, bins_dir+1, endpoint=True)
        hist, _ = np.histogram(ang_flat, bins=ang_bins, weights=mag_flat,
        range=(0.0, np.pi))
        p = hist.astype(np.float64)
        s = p.sum()
        p = p / s if s > 0 else p
        dir_entropy = float(shannon_entropy(p, base=2)) if s > 0 else 0.0
        return {
            "sobel_mag_mean": mag_mean,
            "sobel_mag_std": mag_std,
            "sobel_mag_p95": mag_p95,
            "sobel_energy": energy,
            "sobel_dir_entropy": dir_entropy
        }

        def canny_edges(gray: np.ndarray, low=50, high=150, use_blur=True,
        ksize=(3,3)) -> np.ndarray:
        img = gray
        if use_blur:
        img = cv2.GaussianBlur(gray, ksize, 0)
```

```
        edges = cv2.Canny(img, threshold1=low, threshold2=high)
        return edges
        def canny_features(gray: np.ndarray, low=50, high=150, use_blur=True,
        ksize=(3,3)) -> dict:
        edges = canny_edges(gray, low=low, high=high, use_blur=use_blur,
        ksize=ksize)
        h, w = gray.shape[:2]
        npx = h * w if h*w > 0 else 1
        edge_ratio = float((edges > 0).sum() / npx)
        num_labels, _ = cv2.connectedComponents((edges > 0).astype(np.uint8))
        components = int(max(0, num_labels - 1))
        _, _, mag, _ = sobel_maps(gray, ksize=SOBEL_KSIZE)
        mag_edge = mag[edges > 0]
        mean_edge_mag = float(mag_edge.mean()) if mag_edge.size > 0 else 0.0
        return {
            "canny_edge_ratio": edge_ratio,
            "canny_components": components,
            "canny_mean_edge_mag": mean_edge_mag
        }

        paths = list_images_flat(BASE_DIR)
        print(f"Imágenes encontradas: {len(paths)}")
        if len(paths) == 0:
        raise SystemExit("No se encontraron imágenes. Verifica BASE_DIR.")
        rows = []
        for p in paths:
        try:
        gray = read_gray_u8(p)
        if USE_CLAHE:
        gray = apply_clahe(gray, CLAHE_CLIP, CLAHE_TILE)

        feats_sobel = sobel_features(gray, ksize=SOBEL_KSIZE, bins_dir=18)
        feats_canny = canny_features(gray, low=CANNY_LOW, high=CANNY_HIGH,
        use_blur=GAUSS_BLUR, ksize=BLUR_KSIZE)
        rows.append({
            "name": Path(p).stem,
            "label": infer_label_from_name(p),
            "height": gray.shape[0],
            "width": gray.shape[1],
            **feats_sobel,
            **feats_canny
        })
        except Exception as e:
        print(f"[WARN] {p}: {e}")
        df = pd.DataFrame(rows)
        df.to_csv(OUT_CSV, index=False, encoding="utf-8")
        print(f"Guardado: {OUT_CSV} | filas: {len(df)}
        | columnas: {len(df.columns)}")
        df.head()

```

Listing 4: Script de extracción de características con la técnica Canny/Sobel

Anexo 5. Código FD

```
        import os, glob
        from pathlib import Path
        import numpy as np
```

```
        import pandas as pd
        import cv2
        from numpy.lib.stride_tricks import sliding_window_view

        BASE_DIR = r"DataSedWT/wood"
        OUT_CSV  = "DataSedWT/resultados/features_wood_fd.csv"

        GENERATE_FD_MAP = False
        FD_MAP_WIN   = 32
        FD_MAP_STEP  = 16
        BOX_SCALES = None
        CANNY_LOW, CANNY_HIGH = 50, 150

        def list_images_flat(base_dir):
        """Lista imágenes en una sola carpeta y deduce etiqueta por nombre."""
        items = []
        for ext in ("*.jpg","*.jpeg","*.png","*.bmp","*.tif","*.tiff"):
        for p in glob.glob(os.path.join(base_dir, ext)):
        stem = Path(p).stem.lower()
        if "wood" in stem or "madera" in stem:
        label = "wood"
        elif "textil" in stem or "textile" in stem or "fabric" in stem:
        label = "textile"
        else:
        label = "unknown"
        items.append((p, label))
        return sorted(items)
        def read_gray_u8(path):
        img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
        if img is None:
        raise ValueError(f"No se pudo leer {path}")
        img = cv2.normalize(img, None, 0, 255, cv2.NORM_MINMAX)
        .astype(np.uint8)
        return img

        def differential_box_counting(gray, scales=None):
        H, W = gray.shape
        if scales is None:
        max_pow = int(np.floor(np.log2(min(H, W))))
        s_list = [2**k for k in range(1, max_pow)]
        else:
        s_list = [int(s) for s in scales if s >= 2]
        if len(s_list) < 2:
        return np.nan
        N_list = []
        for s in s_list:
        h_cells = H // s
        w_cells = W // s
        if h_cells == 0 or w_cells == 0:
        continue
        gy = gray[:h_cells*s, :w_cells*s]
        cells = gy.reshape(h_cells, s, w_cells, s).swapaxes(1,2)
        .reshape(h_cells*w_cells, s, s)
        vmin = cells.min(axis=(1,2)).astype(np.int32)
        vmax = cells.max(axis=(1,2)).astype(np.int32)
        Q = int(np.ceil(256 / s))
        n_i = (vmax // (256 // Q)) - (vmin // (256 // Q)) + 1
        n_i = np.clip(n_i, 1, None)
        N_s = int(n_i.sum())
```

```
        N_list.append((s, N_s))
        if len(N_list) < 2:
        return np.nan
        s_vals = np.array([s for s,_ in N_list], dtype=np.float64)
        N_vals = np.array([N for _,N in N_list], dtype=np.float64)
        x = np.log(1.0 / s_vals + 1e-12)
        y = np.log(N_vals + 1e-12)
        A = np.polyfit(x, y, 1)
        FD = A[0]
        return float(FD)

        def fractal_dimension_edges(gray, scales=None, low=50, high=150):
        edges = cv2.Canny(gray, threshold1=low, threshold2=high)
        H, W = edges.shape
        if scales is None:
        max_pow = int(np.floor(np.log2(min(H, W))))
        s_list = [2**k for k in range(1, max_pow)]
        else:
        s_list = [int(s) for s in scales if s >= 2]
        if len(s_list) < 2:
        return np.nan
        N_list = []
        for s in s_list:
        h_cells = (H + s - 1) // s
        w_cells = (W + s - 1) // s
        N_s = 0
        for i in range(h_cells):
        for j in range(w_cells):
        y0, y1 = i*s, min((i+1)*s, H)
        x0, x1 = j*s, min((j+1)*s, W)
        tile = edges[y0:y1, x0:x1]
        if np.any(tile):
        N_s += 1
        N_list.append((s, N_s))
        s_vals = np.array([s for s,_ in N_list], dtype=np.float64)
        N_vals = np.array([N for _,N in N_list], dtype=np.float64)
        x = np.log(1.0 / s_vals + 1e-12)
        y = np.log(N_vals + 1e-12)
        A = np.polyfit(x, y, 1)
        FD = A[0]
        return float(FD)

        def fd_map(gray, win=32, step=16, method="dbc"):
        H, W = gray.shape
        if H < win or W < win:
        return np.array([]), np.nan, np.nan
        fds = []
        for y in range(0, H - win + 1, step):
        for x in range(0, W - win + 1, step):
        patch = gray[y:y+win, x:x+win]
        if method == "dbc":
        f = differential_box_counting(patch)
        else:
        f = fractal_dimension_edges(patch, low=CANNY_LOW, high=CANNY_HIGH)
        if np.isfinite(f):
        fds.append(f)
        fds = np.array(fds, dtype=np.float64)
        if fds.size == 0:
        return fds, np.nan, np.nan
```

```
        return fds, float(np.nanmean(fds)), float(np.nanstd(fds))

        pairs = list_images_flat(BASE_DIR)
        print(f"Imágenes encontradas: {len(pairs)}")
        rows = []
        for path, label in pairs:
        try:
        gray = read_gray_u8(path)
        scales = BOX_SCALES
        fd_dbc   = differential_box_counting(gray, scales)
        fd_edges = fractal_dimension_edges(gray, scales,
        low=CANNY_LOW, high=CANNY_HIGH)
        row = {
            "name": Path(path).stem,
            "label": label,
            "FD_DBC": fd_dbc,
            "FD_edges": fd_edges,
        }
        if GENERATE_FD_MAP:
        _, m, s = fd_map(gray, win=FD_MAP_WIN, step=FD_MAP_STEP, method="dbc")
        row["fdmap_mean"] = m
        row["fdmap_std"]  = s
        rows.append(row)
        except Exception as e:
        print(f"[WARN] {path}: {e}")
        df_fd = pd.DataFrame(rows)
        df_fd.head()

        df_fd.to_csv(OUT_CSV, index=False, encoding="utf-8")
        print(f"Archivo guardado: {OUT_CSV} | filas: {len(df_fd)}")
        if "label" in df_fd.columns and df_fd["label"].nunique() > 1:
        display(df_fd.groupby("label").mean(numeric_only=True).round(5))

```

Listing 5: Script de extracción de características con la técnica FD

Anexo 6. Código Random forest feature importance

```
        import os
        import warnings
        import re
        from datetime import datetime

        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt

        from sklearn.model_selection import train_test_split,
        StratifiedKFold, cross_val_score
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.metrics import accuracy_score, f1_score
        from sklearn.inspection import permutation_importance
        from sklearn.preprocessing import LabelEncoder

        warnings.filterwarnings("ignore")

        CSV_PATH = r"/content/feactures_wood_textile_all.csv"
        TARGET_COLUMN = None
```

```
        TEST_SIZE = 0.20
        N_ESTIMATORS = 300
        SEED = 42
        N_SPLITS_CV = 5
        N_REPEATS_PERM = 10

        def read_csv_safely(path):
        for enc in ["utf-8", "latin-1", "cp1252"]:
        try:
        return pd.read_csv(path, encoding=enc)
        except Exception:
        pass
        return pd.read_csv(path)

        df = read_csv_safely(CSV_PATH)
        print(f"CSV cargado: {df.shape[0]} filas x {df.shape[1]} columnas")

        if TARGET_COLUMN is None:
        posibles = ["label", "Etiqueta_Final", "Clase", "class",
        "Target", "target", "tipo", "Tipo"]
        TARGET_COLUMN = next((c for c in posibles if c in df.columns),
        df.columns[-1])

        print(f"Columna objetivo: {TARGET_COLUMN}")

        DROP_CANDIDATES = {
            "name", "name_a", "name_b", "filename", "file", "image",
            "img", "id", "ids", "path", "path_a", "path_b", "ruta",
            "ruta_a", "ruta_b"
        }
        to_drop = [c for c in df.columns if c.lower() in DROP_CANDIDATES
        and c != TARGET_COLUMN]
        df.drop(columns=to_drop, inplace=True, errors="ignore")

        y_raw = df[TARGET_COLUMN].astype(str)
        le = LabelEncoder()
        y = le.fit_transform(y_raw)
        print(f"Clases detectadas: {list(le.classes_)}")

        X_df = df.drop(columns=[TARGET_COLUMN], errors="ignore").copy()

        def looks_like_path(s):
        if s.dtype == object:
        sample = s.dropna().astype(str).head(50)
        pattern = r"(\.jpg|\.png|\\\\|/|:|^wood_|^madera_|^textile_|^textil_)"
        return sample.str.contains(pattern, flags=re.IGNORECASE, regex=True)
        .any()
        return False

        bad_cols = [c for c in X_df.columns if looks_like_path(X_df[c])]
        if bad_cols:
        print("Quitando columnas tipo ruta/nombre:", bad_cols)
        X_df.drop(columns=bad_cols, inplace=True, errors="ignore")

        obj_cols = X_df.select_dtypes(include=["object", "string",
        "category"]).columns.tolist()
        for c in obj_cols:
        s = pd.to_numeric(X_df[c], errors="coerce")
        if s.notna().mean() >= 0.8:
```

```
        X_df[c] = s

        X_df = X_df.select_dtypes(include=[np.number])

        if X_df.shape[1] == 0:
        raise ValueError("No hay columnas numéricas para entrenar.
        Revisa tu CSV.")

        mask = ~(X_df.isna().any(axis=1) | pd.isna(y))
        X_df, y = X_df[mask], y[mask]

        feature_names = X_df.columns.tolist()
        print(f"Features finales: {len(feature_names)} columnas")


        X_train_df, X_test_df, y_train, y_test = train_test_split(
        X_df,
        y,
        test_size=TEST_SIZE,
        stratify=y,
        random_state=SEED
        )

        print(f"Train: {X_train_df.shape[0]} muestras
        | Test: {X_test_df.shape[0]} muestras")

        OUTDIR = f"resultados_rf_ablation_clean_{datetime.now()
        .strftime('%Y%m%d_%H%M%S')}" os.makedirs(OUTDIR, exist_ok=True)

        clf_full = RandomForestClassifier(
        n_estimators=N_ESTIMATORS,
        max_depth=None,
        random_state=SEED,
        n_jobs=-1
        )
        clf_full.fit(X_train_df, y_train)

        gini_df = pd.DataFrame({
            "Feature": X_train_df.columns,
            "Gini": clf_full.feature_importances_
        }).sort_values("Gini", ascending=False)

        perm = permutation_importance(
        clf_full,
        X_train_df,
        y_train,
        n_repeats=N_REPEATS_PERM,
        random_state=SEED,
        n_jobs=-1
        )

        perm_df = pd.DataFrame({
            "Feature": X_train_df.columns,
            "Perm_Mean": perm.importances_mean,
            "Perm_STD": perm.importances_std
        }).sort_values("Perm_Mean", ascending=False)

        rank_df = gini_df.merge(perm_df, on="Feature", how="inner")
        rank_df["Gini_rank"] = rank_df["Gini"].rank(ascending=False,
```

```
        method="min")
        rank_df["Perm_rank"] = rank_df["Perm_Mean"].rank(ascending=False,
        method="min")
        rank_df["Mean_rank"] = (rank_df["Gini_rank"] +
        rank_df["Perm_rank"]) / 2 rank_df = rank_df.sort_values("Mean_rank")

        gini_df.to_csv(os.path.join(OUTDIR, "importance_gini_train.csv"),
        index=False)
        perm_df.to_csv(os.path.join(OUTDIR,
        "importance_permutation_train.csv"), index=False)
        rank_df.to_csv(os.path.join(OUTDIR,
        "ranking_combinado_gini_perm_train.csv"),index=False)

        gini_sorted = gini_df.sort_values("Gini", ascending=True)
        perm_sorted = perm_df.sort_values("Perm_Mean", ascending=True)

        plt.figure(figsize=(10, 6))
        plt.barh(gini_sorted["Feature"], gini_sorted["Gini"])
        plt.xlabel("Gini Importance")
        plt.title("Feature Importance - Gini (train)")
        plt.tight_layout()
        plt.savefig(os.path.join(OUTDIR, "feature_importance_gini_train.png"),
        dpi=300)
        plt.close()

        plt.figure(figsize=(10, 6))
        plt.barh(perm_sorted["Feature"], perm_sorted["Perm_Mean"])
        plt.xlabel("Permutation Importance (drop in score)")
        plt.title("Feature Importance - Permutation (train)")
        plt.tight_layout()
        plt.savefig(os.path.join(OUTDIR,
        "feature_importance_permutation_train.png"), dpi=300)
        plt.close()

        top5 = rank_df.head(5)["Feature"].tolist()
        top10 = rank_df.head(10)["Feature"].tolist()
        top16 = rank_df.head(16)["Feature"].tolist()

        selected_16 = [
        "FD_DBC",
        "entropy_lbp",
        "std_lbp",
        "energy_lbp",
        "skew_lbp",
        "kurtosis_lbp",
        "canny_edge_ratio",
        "canny_components",
        "canny_mean_edge_mag",
        "Homogeneity_mean_glcm",
        "Energy_mean_glcm",
        "Correlation_mean_glcm",
        "FD_edges",
        "sobel_mag_mean",
        "sobel_dir_entropy",
        "sobel_energy",
        ]
        selected_16 = [f for f in selected_16 if f in X_train_df.columns]

        lbp_fd = [f for f in X_train_df.columns if ("lbp" in f.lower())
```

```
        or ("fd" in f.lower())]
        no_fd = [f for f in X_train_df.columns if "fd" not in f.lower()]
        no_lbp = [f for f in X_train_df.columns if "lbp" not in f.lower()]

        subsets = {
            "Full_25": X_train_df.columns.tolist(),
            "Selected_16_table3": selected_16,
            "Top_16_rank": top16,
            "Top_10_rank": top10,
            "Top_5_rank": top5,
            "LBP_plus_FD": lbp_fd,
            "Without_FD": no_fd,
            "Without_LBP": no_lbp
        }

        subset_rows = []
        for name, cols in subsets.items():
        subset_rows.append({
            "Subset": name,
            "Num_features": len(cols),
            "Features": ", ".join(cols)
        })
        pd.DataFrame(subset_rows).to_csv(os.path.join(OUTDIR,
        "subset_definitions.csv"), index=False)

        def evaluate_subset(subset_name, cols):
        Xtr = X_train_df[cols]
        Xte = X_test_df[cols]

        clf = RandomForestClassifier(
        n_estimators=N_ESTIMATORS,
        max_depth=None,
        random_state=SEED,
        n_jobs=-1
        )

        skf = StratifiedKFold(n_splits=N_SPLITS_CV,
        shuffle=True, random_state=SEED)
        cv_scores = cross_val_score(
        clf,
        Xtr,
        y_train,
        cv=skf,
        scoring="accuracy",
        n_jobs=-1
        )

        clf.fit(Xtr, y_train)

        y_pred = clf.predict(Xte)
        acc = accuracy_score(y_test, y_pred)
        f1w = f1_score(y_test, y_pred, average="weighted")

        return {
            "Subset": subset_name,
            "Num_features": len(cols),
            "Accuracy_test": acc,
            "F1_weighted": f1w,
            "CV_mean_train": cv_scores.mean(),
```

```
            "CV_std_train": cv_scores.std(),
            "Features": ", ".join(cols)
        }

        results = []
        for name, cols in subsets.items():
        if len(cols) == 0:
        continue
        print(f"Evaluando: {name} ({len(cols)} features)")
        results.append(evaluate_subset(name, cols))

        results_df = pd.DataFrame(results).sort_values(
        ["Accuracy_test", "CV_mean_train"],
        ascending=False
        )

        results_df.to_csv(os.path.join(OUTDIR,
        "ablation_results_clean.csv"), index=False)

        print("\n=== RESULTADOS ===")
        print(results_df[[
        "Subset", "Num_features", "Accuracy_test",
        "F1_weighted", "CV_mean_train", "CV_std_train"]])

        print(f"\nResultados guardados en: {OUTDIR}")

```

Listing 6: Script de selección de características Random forest feature importance

Anexo 7. Código SVM

```
        import os, json, time, sys
        from datetime import datetime
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt

        from sklearn.preprocessing import StandardScaler, LabelEncoder
        from sklearn.pipeline import Pipeline
        from sklearn.svm import SVC
        from sklearn.model_selection import train_test_split,
        StratifiedKFold, GridSearchCV, learning_curve
        from sklearn.metrics import (accuracy_score, confusion_matrix,
        classification_report,
        roc_auc_score, RocCurveDisplay, ConfusionMatrixDisplay)

        from sklearn.metrics import log_loss
        import joblib

        CSV_PATH = "C:/Users/jcarr/Documents/2025/Ejercicios_CodigoLimpio
        /DataSedWT/resultados/integracion_caracteristicas
        /feactures_wood_textile_Gini_Permutación.csv"
        DROP_COLS = ["name"]
        TARGET_COL = "label"

        PREF_X = "Energy_mean_glcm"
        PREF_Y = "Dissimilarity_mean_glcm"

```

```
        PARAM_GRID = {
            "svm__C": [0.001, 0.01, 0.1, 1],
            "svm__gamma": [0.0001, 0.001, 0.01,"scale"]
        }
        N_SPLITS = 7
        TEST_SIZE = 0.2
        RANDOM_STATE = 42

        STAMP = datetime.now().strftime("%Y%m%d_%H%M%S")
        OUT_DIR = f"resultados_svm_{STAMP}"
        os.makedirs(OUT_DIR, exist_ok=True)
        print(f"Carpeta de salida: {OUT_DIR}")

        read_ok = False
        for enc in ["utf-8", "latin-1", "cp1252"]:
        try:
        df = pd.read_csv(CSV_PATH, encoding=enc)
        read_ok = True
        break
        except Exception as e:
        last_err = str(e)

        if not read_ok:
        raise RuntimeError(f"No se pudo leer el CSV '{CSV_PATH}'
        . Último error: {last_err}")

        print(f"CSV cargado: {df.shape[0]} filas x {df.shape[1]} columnas")

        df = df.drop(columns=[c for c in DROP_COLS if c in df.columns],
        errors="ignore")

        if TARGET_COL not in df.columns:
        raise ValueError(f"La columna objetivo '{TARGET_COL}' no existe
        .Columnas: {list(df.columns)}")

        X = df.drop(columns=[TARGET_COL])
        y = df[TARGET_COL]

        le = LabelEncoder()
        y_enc = le.fit_transform(y.astype(str))

        def to_py(o):
        import numpy as np
        if isinstance(o, (np.integer,)):
        return int(o)
        if isinstance(o, (np.floating,)):
        return float(o)
        if isinstance(o, (np.ndarray,)):
        return o.tolist()
        return str(o)

        label_mapping = {int(i): to_py(cls) for i,
        cls in enumerate(le.classes_)}

        with open(os.path.join(OUT_DIR, "label_mapping.json"),
        "w", encoding="utf-8") as f:
        json.dump(label_mapping, f, ensure_ascii=False, indent=2)

        plt.figure(figsize=(5.5, 4.2))
```

```
        class_counts = pd.Series(y).value_counts()
        class_counts.plot(kind="bar")
        plt.title("Distribución de clases")
        plt.xlabel("Clase")
        plt.ylabel("Número de muestras")
        plt.grid(axis="y", linestyle="--", alpha=0.6)
        plt.tight_layout()
        plt.savefig(os.path.join(OUT_DIR,
        "01_distribucion_clases.png"), dpi=300)
        plt.close()

        X_train, X_test, y_train, y_test = train_test_split(
        X, y_enc, test_size=TEST_SIZE,
        random_state=RANDOM_STATE, stratify=y_enc)

        pipe = Pipeline([
        ("scaler", StandardScaler()),
        ("svm", SVC(kernel="rbf", probability=True,
        random_state=RANDOM_STATE))
        ])

        cv = StratifiedKFold(n_splits=N_SPLITS, shuffle=True,
        random_state=RANDOM_STATE)
        gs = GridSearchCV(pipe, PARAM_GRID, cv=cv, scoring="accuracy",
        n_jobs=-1, refit=True, verbose=0) t0 = time.time()
        gs.fit(X_train, y_train)
        t_train = time.time() - t0

        best_model = gs.best_estimator_
        best_params = gs.best_params_
        cv_mean = gs.best_score_

        print(f"Best params: {best_params} | CV mean acc: {cv_mean:.4f}
        | Train time: {t_train:.1f}s")

        test_logloss = log_loss(y_test, best_model.predict_proba(X_test))

        y_pred = best_model.predict(X_test)
        y_proba = best_model.predict_proba(X_test)[:, 1]
        if len(np.unique(y_enc)) == 2 else None

        acc = accuracy_score(y_test, y_pred)
        report = classification_report(y_test, y_pred,
        target_names=[str(c) for c in le.classes_], output_dict=True)
        cm = confusion_matrix(y_test, y_pred)

        metrics = {
            "best_params": best_params,
            "cv_mean_accuracy": float(cv_mean),
            "test_accuracy": float(acc),
            "test_logloss": float(test_logloss),
            "train_time_sec": float(t_train),
            "classification_report": report,
        }

        if y_proba is not None:
        auc = roc_auc_score(y_test, y_proba)
        metrics["test_auc"] = float(auc)

```

```
        with open(os.path.join(OUT_DIR, "metrics.json"), "w",
        encoding="utf-8") as f: json.dump(metrics, f,
        ensure_ascii=False, indent=2)
        print("Métricas guardadas en metrics.json")

        plt.figure(figsize=(5.2, 4.6))
        disp = ConfusionMatrixDisplay(confusion_matrix=cm,
        display_labels=le.classes_)
        disp.plot(values_format='d', cmap="Blues", colorbar=True)
        plt.title("Matriz de confusión - Test")
        plt.tight_layout()
        plt.savefig(os.path.join(OUT_DIR, "02_matriz_confusion.png"), dpi=300)
        plt.close()

        train_sizes, train_scores_acc, val_scores_acc = learning_curve(
        best_model, X_train, y_train, cv=cv,
        train_sizes=np.linspace(0.1, 1.0, 10), scoring="accuracy", n_jobs=-1
        )
        tr_acc_mean = np.mean(train_scores_acc, axis=1)
        tr_acc_std  = np.std(train_scores_acc, axis=1)
        vl_acc_mean = np.mean(val_scores_acc, axis=1)
        vl_acc_std  = np.std(val_scores_acc, axis=1)

        plt.figure(figsize=(6.2, 4.4))
        plt.plot(train_sizes, tr_acc_mean,'o-',label="Accuracy Entrenamiento")
        plt.plot(train_sizes, vl_acc_mean,'o-', label="Accuracy Validación")
        plt.fill_between(train_sizes, tr_acc_mean - tr_acc_std,
        tr_acc_mean + tr_acc_std, alpha=0.2)
        plt.fill_between(train_sizes, vl_acc_mean - vl_acc_std,
        vl_acc_mean + vl_acc_std, alpha=0.2)
        plt.title("Curva de Precisión (learning curve) - SVM")
        plt.xlabel("Tamaño del entrenamiento"); plt.ylabel("Accuracy")
        plt.legend(); plt.grid(True, linestyle="--", alpha=0.6)
        plt.tight_layout()
        plt.savefig(os.path.join(OUT_DIR, "04_learningcurve_accuracy.png"),
        dpi=300)
        plt.close()

        train_sizes, tr_scores_ll, vl_scores_ll = learning_curve(
        best_model, X_train, y_train, cv=cv,
        train_sizes=np.linspace(0.1, 1.0, 10), scoring="neg_log_loss",
        n_jobs=-1)
        tr_loss_mean = -np.mean(tr_scores_ll, axis=1)
        tr_loss_std  =  np.std(tr_scores_ll, axis=1)
        vl_loss_mean = -np.mean(vl_scores_ll, axis=1)
        vl_loss_std  =  np.std(vl_scores_ll, axis=1)

        plt.figure(figsize=(6.2, 4.4))
        plt.plot(train_sizes, tr_loss_mean,'o-',label="Pérdida Entrenamiento")
        plt.plot(train_sizes, vl_loss_mean, 'o-', label="Pérdida Validación")
        plt.fill_between(train_sizes, tr_loss_mean - tr_loss_std,
        tr_loss_mean + tr_loss_std, alpha=0.2)
        plt.fill_between(train_sizes, vl_loss_mean - vl_loss_std,
        vl_loss_mean + vl_loss_std, alpha=0.2)
        plt.title("Curva de Pérdida (log-loss) - SVM")
        plt.xlabel("Tamaño del entrenamiento"); plt.ylabel("Log-loss")
        plt.legend(); plt.grid(True, linestyle="--", alpha=0.6)
        plt.tight_layout()
        plt.savefig(os.path.join(OUT_DIR, "05_learningcurve_logloss.png")
```

```
        , dpi=300)
        plt.close()

        if y_proba is not None and len(np.unique(y_enc)) == 2:
        plt.figure(figsize=(5.4, 4.6))
        RocCurveDisplay.from_predictions(y_test, y_proba)
        plt.title("Curva ROC - Test")
        plt.tight_layout()
        plt.savefig(os.path.join(OUT_DIR, "06_roc_auc.png"), dpi=300)
        plt.close()

        print("Listo. Imágenes y métricas guardadas en:", OUT_DIR)

        joblib.dump(best_model, "svm_best_model.pkl")

        print("Mejor modelo (pipeline) guardado en svm_best_model.pkl")


```

Listing 7: Script SVM

Anexo 8. Código RF

```
        import os, json, time
        from datetime import datetime
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt

        from sklearn.preprocessing import LabelEncoder
        from sklearn.model_selection import train_test_split, StratifiedKFold,
        GridSearchCV, learning_curve
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.metrics import (accuracy_score, confusion_matrix,
        classification_report, roc_auc_score, RocCurveDisplay,
        ConfusionMatrixDisplay, log_loss)
        from sklearn.decomposition import PCA

        CSV_PATH = "C:/Users/jcarr/Documents/2025/Ejercicios_CodigoLimpio
        /DataSedWT/resultados/integracion_caracteristicas
        /feactures_wood_textile_Gini_Permutación.csv"
        DROP_COLS = ["name"]
        TARGET_COL = "label"

        PREF_X = "Energy_mean_glcm"
        PREF_Y = "Dissimilarity_mean_glcm"

        PARAM_GRID = {
            "n_estimators": [200, 400],
            "max_depth": [8, 10, 12],
            "min_samples_split": [10, 20],
            "min_samples_leaf": [10, 20],
            "max_features": ["sqrt", "log2"]
        }
        N_SPLITS = 7
        TEST_SIZE = 0.2
        RANDOM_STATE = 42

```

```
        STAMP = datetime.now().strftime("%Y%m%d_%H%M%S")
        OUT_DIR = f"resultados_rf_{STAMP}"
        os.makedirs(OUT_DIR, exist_ok=True)
        print(f" Carpeta de salida: {OUT_DIR}")

        def json_safe(obj):
        import numpy as np
        if isinstance(obj, dict):
        return {json_safe(k): json_safe(v) for k, v in obj.items()}
        if isinstance(obj, (list, tuple)):
        return [json_safe(x) for x in obj]
        if isinstance(obj, (np.integer,)):
        return int(obj)
        if isinstance(obj, (np.floating,)):
        return float(obj)
        if isinstance(obj, (np.ndarray,)):
        return obj.tolist()
        return obj

        read_ok, last_err = False, ""
        for enc in ["utf-8", "latin-1", "cp1252"]:
        try:
        df = pd.read_csv(CSV_PATH, encoding=enc)
        read_ok = True
        break
        except Exception as e:
        last_err = str(e)

        if not read_ok:
        raise RuntimeError(f"No se pudo leer el CSV '{CSV_PATH}'.
        Último error: {last_err}")

        print(f"CSV cargado: {df.shape[0]} filas x {df.shape[1]} columnas")

        df = df.drop(columns=[c for c in DROP_COLS if c in df.columns],
        errors="ignore")

        if TARGET_COL not in df.columns:
        raise ValueError(f"La columna objetivo '{TARGET_COL}' no existe
        . Columnas: {list(df.columns)}")

        X = df.drop(columns=[TARGET_COL])
        y = df[TARGET_COL]

        le = LabelEncoder()
        y_enc = le.fit_transform(y)
        label_mapping = {int(i): (str(cls) if not isinstance(cls, str)
        else cls) for i, cls in enumerate(le.classes_)}
        with open(os.path.join(OUT_DIR, "label_mapping.json"), "w",
        encoding="utf-8") as f:
        json.dump(json_safe(label_mapping), f, ensure_ascii=False, indent=2)
        print("Mapeo de etiquetas guardado en label_mapping.json")

        plt.figure(figsize=(5.5, 4.2))
        pd.Series(y).value_counts().plot(kind="bar")
        plt.title("Distribución de clases")
        plt.xlabel("Clase"); plt.ylabel("Número de muestras")
        plt.grid(axis="y", linestyle="--", alpha=0.6)
        plt.tight_layout()
```

```
        plt.savefig(os.path.join(OUT_DIR, "01_distribucion_clases.png")
        , dpi=300)
        plt.close()

        X_train, X_test, y_train, y_test = train_test_split(
        X, y_enc, test_size=TEST_SIZE, random_state=RANDOM_STATE,
        stratify=y_enc)

        rf = RandomForestClassifier(random_state=RANDOM_STATE, n_jobs=-1)
        cv = StratifiedKFold(n_splits=N_SPLITS, shuffle=True,
        random_state=RANDOM_STATE)

        gs = GridSearchCV(rf, PARAM_GRID, cv=cv, scoring="accuracy",
        n_jobs=-1, refit=True, verbose=0)
        t0 = time.time()
        gs.fit(X_train, y_train)
        t_train = time.time() - t0

        best_model = gs.best_estimator_
        best_params = gs.best_params_
        cv_mean = gs.best_score_
        print(f"Best params: {best_params} | CV mean acc: {cv_mean:.4f}
        | Train time: {t_train:.1f}s")

        y_pred = best_model.predict(X_test)
        y_proba = best_model.predict_proba(X_test)[:, 1]
        if len(np.unique(y_enc)) == 2 else None

        acc = accuracy_score(y_test, y_pred)
        report = classification_report(y_test, y_pred,
        target_names=[str(c) for c in le.classes_], output_dict=True)
        cm = confusion_matrix(y_test, y_pred)

        metrics = {
            "best_params": best_params,
            "cv_mean_accuracy": float(cv_mean),
            "test_accuracy": float(acc),
            "train_time_sec": float(t_train),
            "classification_report": report,
        }

        if y_proba is not None:
        auc = roc_auc_score(y_test, y_proba)
        metrics["test_auc"] = float(auc)
        metrics["test_log_loss"] = float(log_loss(y_test,
        best_model.predict_proba(X_test)))

        with open(os.path.join(OUT_DIR, "metrics.json"),
        "w", encoding="utf-8") as f:
        json.dump(json_safe(metrics), f, ensure_ascii=False, indent=2)
        print("Métricas guardadas en metrics.json")

        plt.figure(figsize=(5.2, 4.6))
        disp = ConfusionMatrixDisplay(confusion_matrix=cm,
        display_labels=le.classes_)
        disp.plot(values_format='d', cmap="Blues", colorbar=True)
        plt.title("Matriz de confusión - Test")
        plt.tight_layout()
        plt.savefig(os.path.join(OUT_DIR, "02_matriz_confusion.png"
```

```
        , dpi=300)
        plt.close()


        train_sizes, train_scores_acc, val_scores_acc = learning_curve(
        best_model, X_train, y_train, cv=cv,
        train_sizes=np.linspace(0.1, 1.0, 10), scoring="accuracy", n_jobs=-1
        )
        tr_acc_mean = np.mean(train_scores_acc, axis=1)
        tr_acc_std  = np.std(train_scores_acc, axis=1)
        vl_acc_mean = np.mean(val_scores_acc, axis=1)
        vl_acc_std  = np.std(val_scores_acc, axis=1)

        plt.figure(figsize=(6.2, 4.4))
        plt.plot(train_sizes, tr_acc_mean,'o-',label="Accuracy Entrenamiento")
        plt.plot(train_sizes, vl_acc_mean,'o-', label="Accuracy Validación")
        plt.fill_between(train_sizes, tr_acc_mean - tr_acc_std,
        tr_acc_mean + tr_acc_std, alpha=0.2)
        plt.fill_between(train_sizes, vl_acc_mean - vl_acc_std,
        vl_acc_mean + vl_acc_std, alpha=0.2)
        plt.title("Curva de Precisión (learning curve) - Random Forest")
        plt.xlabel("Tamaño del entrenamiento"); plt.ylabel("Accuracy")
        plt.legend(); plt.grid(True, linestyle="--", alpha=0.6)
        plt.tight_layout()
        plt.savefig(os.path.join(OUT_DIR, "04_learningcurve_accuracy.png"),
         dpi=300)
        plt.close()

        if y_proba is not None:
        train_sizes, tr_scores_ll, vl_scores_ll = learning_curve(
        best_model, X_train, y_train, cv=cv,
        train_sizes=np.linspace(0.1, 1.0, 10), scoring="neg_log_loss",
        n_jobs=-1)
        tr_loss_mean = -np.mean(tr_scores_ll, axis=1)
        tr_loss_std  =  np.std(tr_scores_ll, axis=1)
        vl_loss_mean = -np.mean(vl_scores_ll, axis=1)
        vl_loss_std  =  np.std(vl_scores_ll, axis=1)

        plt.figure(figsize=(6.2, 4.4))
        plt.plot(train_sizes, tr_loss_mean,'o-',label="Pérdida Entrenamiento")
        plt.plot(train_sizes, vl_loss_mean,'o-', label="Pérdida Validación")
        plt.fill_between(train_sizes, tr_loss_mean - tr_loss_std,
        tr_loss_mean + tr_loss_std, alpha=0.2)
        plt.fill_between(train_sizes, vl_loss_mean - vl_loss_std,
        vl_loss_mean + vl_loss_std, alpha=0.2)
        plt.title("Curva de Pérdida (log-loss) - Random Forest")
        plt.xlabel("Tamaño del entrenamiento"); plt.ylabel("Log-loss")
        plt.legend(); plt.grid(True, linestyle="--", alpha=0.6)
        plt.tight_layout()
        plt.savefig(os.path.join(OUT_DIR, "05_learningcurve_logloss.png"),
        dpi=300)
        plt.close()

        if y_proba is not None and len(np.unique(y_enc)) == 2:
        plt.figure(figsize=(5.4, 4.6))
        RocCurveDisplay.from_predictions(y_test, y_proba)
        plt.title("Curva ROC - Test (Random Forest)")
        plt.tight_layout()
        plt.savefig(os.path.join(OUT_DIR, "06_roc_auc.png"), dpi=300)
```

```
        plt.close()

        report = classification_report(y_test, y_pred,
        target_names=[str(c) for c in le.classes_])
        print(report)

        txt_path = os.path.join(OUT_DIR, "09_classification_report.txt")
        with open(txt_path, "w", encoding="utf-8") as f:
        f.write("Reporte de clasificación:\n")
        f.write(report)
        print(f" Reporte de clasificación guardado en: {txt_path}")

        model_filename = "rf_best_model.pkl"
        model_save_path = os.path.join(OUT_DIR, model_filename)

        try:
        joblib.dump(best_model, model_save_path)
        print(f"\n Modelo (Random Forest) guardado exitosamente en:")
        print(f"{model_save_path}")

        print(f"\nRecuerda que las métricas y gráficos
        (incluido 'label_mapping.json') están en:")
        print(f"{OUT_DIR}")

        except NameError:
        print("\n Error: La variable 'best_model' no está definida.")
        print("Asegúrate de haber ejecutado la celda de GridSearchCV.")
        except Exception as e:
        print(f"\n Ocurrió un error inesperado al guardar:")
        print(e)
```

Listing 8: Script RF

Anexo 9. Código MLP

```
import os, json, time
from datetime import datetime
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder, StandardScaler,
PowerTransformer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split,
StratifiedKFold, GridSearchCV, learning_curve
from sklearn.neural_network import MLPClassifier
from sklearn.calibration import CalibratedClassifierCV
from sklearn.metrics import (
accuracy_score, confusion_matrix, classification_report,
roc_auc_score, RocCurveDisplay, ConfusionMatrixDisplay, log_loss,
precision_recall_curve
)
from sklearn.decomposition import PCA

CSV_PATH = r"C:/Users/jcarr/Documents/2025/CSV files and algorithms
/Feature Extraction Results/Feature integration
/feactures_wood_textile_Gini_Permutación.csv"
DROP_COLS = ["name"]
TARGET_COL = "label"

PARAM_GRID = {
    "mlp__hidden_layer_sizes": [
    (128,), (256,), (128, 64), (256, 128), (256, 128, 64)
    ],
    "mlp__activation": ["relu", "tanh"],
    "mlp__alpha": [1e-5, 3e-5, 1e-4, 3e-4, 1e-3, 3e-3, 1e-2],
    "mlp__learning_rate_init": [5e-4, 7e-4, 1e-3, 2e-3],
}

N_SPLITS = 7
TEST_SIZE = 0.2
RANDOM_STATE = 42
MAX_ITER = 700
EARLY_STOPPING = True
N_ITER_NO_CHANGE = 20
VALIDATION_FRACTION = 0.12

STAMP = datetime.now().strftime("%Y%m%d_%H%M%S")
OUT_DIR = f"resultados_mlp_{STAMP}"
os.makedirs(OUT_DIR, exist_ok=True)
print(f"Carpeta de salida: {OUT_DIR}")

def json_safe(obj):
import numpy as np
if isinstance(obj, dict):
return {json_safe(k): json_safe(v) for k, v in obj.items()}
if isinstance(obj, (list, tuple)):
return [json_safe(x) for x in obj]
```

```
        if isinstance(obj, (np.integer,)):
        return int(obj)
        if isinstance(obj, (np.floating,)):
        return float(obj)
        if isinstance(obj, (np.ndarray,)):
        return obj.tolist()
        return obj

        read_ok, last_err = False, ""
        for enc in ["utf-8", "latin-1", "cp1252"]:
        try:
        df = pd.read_csv(CSV_PATH, encoding=enc)
        read_ok = True
        break
        except Exception as e:
        last_err = str(e)
        if not read_ok:
        raise RuntimeError(f"No se pudo leer el CSV '{CSV_PATH}'
        . Último error: {last_err}")

        print(f" CSV cargado: {df.shape[0]} filas x {df.shape[1]} columnas")
        df = df.drop(columns=[c for c in DROP_COLS if c in df.columns],
        errors="ignore")
        if TARGET_COL not in df.columns:
        raise ValueError(f"La columna objetivo '{TARGET_COL}' no existe.
        Columnas: {list(df.columns)}")

        X = df.drop(columns=[TARGET_COL])
        y = df[TARGET_COL]

        le = LabelEncoder()
        y_enc = le.fit_transform(y)
        label_mapping = {int(i): (str(cls) if not isinstance(cls, str)
        else cls) for i, cls in enumerate(le.classes_)}
        with open(os.path.join(OUT_DIR, "label_mapping.json"),
         "w", encoding="utf-8") as f:
        json.dump(json_safe(label_mapping), f, ensure_ascii=False, indent=2)

        plt.figure(figsize=(5.5, 4.2))
        pd.Series(y).value_counts().plot(kind="bar")
        plt.title("Distribución de clases")
        plt.xlabel("Clase"); plt.ylabel("Número de muestras")
        plt.grid(axis="y", linestyle="--", alpha=0.6)
        plt.tight_layout()
        plt.savefig(os.path.join(OUT_DIR, "01_distribucion_clases.png")
        , dpi=300)
        plt.close()

        X_train, X_test, y_train, y_test = train_test_split(
        X, y_enc, test_size=TEST_SIZE, random_state=RANDOM_STATE,
        stratify=y_enc)

        num_cols = X.columns.tolist()
        pre = ColumnTransformer(
        transformers=[
        ("num", Pipeline([
        ("power", PowerTransformer(method="yeo-johnson", standardize=False)),
        ("scaler", StandardScaler())
        ]), num_cols)
```

```
        ],
        remainder="drop"
        )

        base_mlp = MLPClassifier(
        solver="adam",
        learning_rate="adaptive",
        max_iter=MAX_ITER,
        tol=1e-5,
        random_state=RANDOM_STATE,
        early_stopping=EARLY_STOPPING,
        n_iter_no_change=N_ITER_NO_CHANGE,
        validation_fraction=VALIDATION_FRACTION,
        verbose=False
        )

        pipe = Pipeline([
        ("prep", pre),
        ("mlp", base_mlp)
        ])

        cv = StratifiedKFold(n_splits=N_SPLITS, shuffle=True,
        random_state=RANDOM_STATE)
        gs = GridSearchCV(pipe, PARAM_GRID, cv=cv, scoring="accuracy",
        n_jobs=-1, refit=True, verbose=0)

        t0 = time.time()
        gs.fit(X_train, y_train)
        t_train = time.time() - t0

        best_model = gs.best_estimator_
        best_params = gs.best_params_
        cv_mean = gs.best_score_
        print(f" Best params: {best_params} | CV mean acc: {cv_mean:.4f}
        | Train time: {t_train:.1f}s")

        cal = CalibratedClassifierCV(best_model, method="isotonic", cv=3)
        cal.fit(X_train, y_train)

        is_binary = (len(np.unique(y_enc)) == 2)
        if is_binary:
        y_proba = cal.predict_proba(X_test)[:, 1]
        auc = roc_auc_score(y_test, y_proba)
        ll  = log_loss(y_test, cal.predict_proba(X_test))
        else:
        y_proba, auc, ll = None, None, None

        y_pred_05 = cal.predict(X_test) if not is_binary else
        (y_proba >= 0.5).astype(int)
        acc_05 = accuracy_score(y_test, y_pred_05)
        cm_05  = confusion_matrix(y_test, y_pred_05)
        rep_text_05 = classification_report(y_test, y_pred_05,
        target_names=[str(c) for c in le.classes_])
        rep_dict_05 = classification_report(y_test, y_pred_05,
        target_names=[str(c) for c in le.classes_], output_dict=True)

        if is_binary:
        prec, rec, thr = precision_recall_curve(y_test, y_proba)
        f1 = (2*prec*rec)/(prec+rec+1e-12)
```

```
        t_star = float(thr[np.nanargmax(f1)]) if len(thr) else 0.5
        y_pred_star = (y_proba >= t_star).astype(int)
        acc_star = accuracy_score(y_test, y_pred_star)
        cm_star  = confusion_matrix(y_test, y_pred_star)
        rep_dict_star = classification_report(y_test, y_pred_star,
        target_names=[str(c) for c in le.classes_], output_dict=True)
        else:
        t_star, acc_star, cm_star, rep_dict_star = None, None, None, None

        metrics = {
            "best_params": best_params,
            "cv_mean_accuracy": float(cv_mean),
            "train_time_sec": float(t_train),
            "test_accuracy_thr_0.5": float(acc_05),
            "classification_report_thr_0.5": rep_dict_05,
        }
        if auc is not None: metrics["test_auc_calibrated"] = float(auc)
        if ll  is not None: metrics["test_log_loss_calibrated"] = float(ll)
        if t_star is not None:
        metrics["threshold_opt_f1"] = float(t_star)
        metrics["test_accuracy_thr_opt"] = float(acc_star)
        metrics["classification_report_thr_opt"] = rep_dict_star

        with open(os.path.join(OUT_DIR, "metrics.json"), "w",encoding="utf-8")
        as f: json.dump(json_safe(metrics), f, ensure_ascii=False, indent=2)

        with open(os.path.join(OUT_DIR,
        "09_classification_report_thr_0.5.txt"),"w", encoding="utf-8") as f:
        f.write("Reporte de clasificación (umbral 0.5):\n")
        f.write(rep_text_05)

        if is_binary:
        with open(os.path.join(OUT_DIR, "09b_threshold_optimo_f1.txt"),
        "w", encoding="utf-8") as f:
        f.write(f"Umbral óptimo por F1: {t_star:.5f}\n")

        plt.figure(figsize=(5.6, 4.8))
        ConfusionMatrixDisplay(confusion_matrix=cm_05,
        display_labels=le.classes_).plot(values_format='d', cmap="Blues",
        colorbar=True)
        plt.title("Matriz de confusión - Test (thr=0.5) - MLP calibrado")
        plt.tight_layout()
        plt.savefig(os.path.join(OUT_DIR, "02_matriz_confusion_thr_0.5.png")
        , dpi=300)
        plt.close()


        train_sizes, train_scores_acc, val_scores_acc = learning_curve(
        best_model, X_train, y_train, cv=cv,
        train_sizes=np.linspace(0.1, 1.0, 10), scoring="accuracy", n_jobs=-1
        )
        tr_acc_mean = np.mean(train_scores_acc, axis=1)
        tr_acc_std  = np.std(train_scores_acc, axis=1)
        vl_acc_mean = np.mean(val_scores_acc, axis=1)
        vl_acc_std  = np.std(val_scores_acc, axis=1)

        plt.figure(figsize=(7.2, 4.8))
        plt.plot(train_sizes, tr_acc_mean,'o-',label="Accuracy Entrenamiento")
        plt.plot(train_sizes, vl_acc_mean,'o-',label="Accuracy Validación")
```

```
        plt.fill_between(train_sizes, tr_acc_mean - tr_acc_std,
        tr_acc_mean + tr_acc_std, alpha=0.22)
        plt.fill_between(train_sizes, vl_acc_mean - vl_acc_std,
        vl_acc_mean + vl_acc_std, alpha=0.22)
        plt.title("Curva de Precisión(learning curve)- MLP (mejor pipeline)")
        plt.xlabel("Tamaño del entrenamiento (muestras)")
        plt.ylabel("Accuracy")
        plt.legend(); plt.grid(True, linestyle="--", alpha=0.6)
        plt.tight_layout()
        plt.savefig(os.path.join(OUT_DIR, "04_learningcurve_accuracy.png"),
        dpi=300)
        plt.close()

        if is_binary:
        train_sizes_ll, tr_scores_ll, vl_scores_ll = learning_curve(
        best_model, X_train, y_train, cv=cv,
        train_sizes=np.linspace(0.1, 1.0, 10), scoring="neg_log_loss",
        n_jobs=-1)
        tr_loss_mean = -np.mean(tr_scores_ll, axis=1)
        tr_loss_std  =  np.std(tr_scores_ll, axis=1)
        vl_loss_mean = -np.mean(vl_scores_ll, axis=1)
        vl_loss_std  =  np.std(vl_scores_ll, axis=1)

        plt.figure(figsize=(7.2, 4.8))
        plt.plot(train_sizes_ll,tr_loss_mean,'o-',
        label="Pérdida Entrenamiento")
        plt.plot(train_sizes_ll,vl_loss_mean,'o-',label="Pérdida Validación")
        plt.fill_between(train_sizes_ll,tr_loss_mean-tr_loss_std,tr_loss_mean
        + tr_loss_std, alpha=0.22)
        plt.fill_between(train_sizes_ll, vl_loss_mean - vl_loss_std,
        vl_loss_mean + vl_loss_std, alpha=0.22)
        plt.title("Curva de Pérdida (log-loss) - MLP (mejor pipeline)")
        plt.xlabel("Tamaño del entrenamiento (muestras)")
        plt.ylabel("Log-loss")
        plt.legend(); plt.grid(True, linestyle="--", alpha=0.6)
        plt.tight_layout()
        plt.savefig(os.path.join(OUT_DIR, "05_learningcurve_logloss.png")
        , dpi=300)
        plt.close()

        if is_binary:
        plt.figure(figsize=(5.6, 4.8))
        RocCurveDisplay.from_predictions(y_test, y_proba)
        plt.title("Curva ROC - Test (MLP calibrado)")
        plt.tight_layout()
        plt.savefig(os.path.join(OUT_DIR, "06_roc_auc.png"), dpi=300)
        plt.close()
        print("Listo. Imágenes y métricas guardadas en:", OUT_DIR)

```

Listing 9: Script MLP