



UNIVERSIDAD RAMON LLULL

Facultat Internacional de Comerç i Economia Digital La Salle

Trabajo Final de Máster

Máster Universitario en Ciencias de los Datos / Data Science

**SISTEMA DE CLASIFICACIÓN DE
DESTINO FINAL PARA PRENDAS
SEGUNDA MANO**

Alumno

Juan Pedro Montero Gil

Profesor Ponente

David Sirera i Pulido

ACTA DEL EXAMEN DEL TRABAJO FINAL DE MÁSTER

Reunido el Tribunal calificador en el día de la fecha, el alumno

Juan Pedro Montero Gil

Expuso su Trabajo de Final de Máster, el cual trató sobre el tema siguiente:

SISTEMA DE CLASIFICACIÓN DE DESTINO FINAL PARA PRENDAS DE SEGUNDA MANO

Acabada la exposición y contestadas por parte del alumno las objeciones formuladas por los miembros del tribunal, éste valoró el mencionado Trabajo con la calificación de

VOCAL DEL TRIBUNAL

VOCAL DEL TRIBUNAL

PRESIDENTE DEL TRIBUNAL

Porque un algoritmo puede ser *verde*

Agradecimientos a mi familia por su apoyo incondicional,
a mis amigos de máster por las risas y los buenos momentos,
y a los profesores por todo el conocimiento y pasión entregada.

Resumen

Ante el creciente impacto ambiental de la moda rápida, resulta imprescindible desarrollar tecnologías bajo los principios de la Green AI que faciliten su gestión sostenible e impulsen modelos circulares. Esta investigación propone el desarrollo de un *meta-learner* capaz de clasificar el destino final de prendas de segunda mano desde sus imágenes. La metodología propuesta se desarrolla en tres fases: identificación y selección de las características más relevantes; extracción de dichos atributos mediante descriptores visuales y construcción de sub-modelos para su predicción; y ensamblado y entrenamiento de un *meta-learner*. Los resultados revelan que las características *condition*, *price* y *pilling* presentan el mayor nivel predictivo ante *usage*; el descriptor LBP-U resulta ser óptimo para la extracción de características; y que aplicado en sub-modelos alcanzan un F1 Macro de 44.09 % para *condition* con Random Forest, 51.79 % para *price* y 48.09 % para *pilling* con XGBoost; y que un *meta-learner* con CatBoost, entrenado sobre las predicciones ensambladas, logra un F1 Macro de 41.52 % para *usage* vs un 65.07 % logrado por el modelo base entrenado con metadatos. El proceso completo emitió 2.39 kg de CO₂, demostrando que es posible combinar precisión y sostenibilidad en la clasificación de prendas de segunda mano.

palabras claves: ropa, prendas de segunda mano, moda circular, IA verde, huella de carbono, visión por computador, Local Binnary Pattern Uniform (LBP-U), aprendizaje automático, ensamblado de modelo, meta modelo.

Abstract

Given the growing environmental impact of fast fashion, it is essential to develop technologies based on the principles of Green AI that facilitate its sustainable management and promote circular models. This research proposes the development of a meta-learner capable of classifying the final destination of second-hand garments from their images. The proposed methodology unfolds in three phases: identification and selection of the most relevant features; extraction of these attributes using visual descriptors and construction of sub-models for their prediction; and the assembly and training of a meta-learner. The results reveal that the features *condition*, *price*, and *pilling* exhibit the highest predictive power for *usage*; the LBP-U descriptor proves optimal for feature extraction and, when applied to sub-models, achieves a test F1-Macro of 44.09 % for *condition* with Random Forest, 51.79 % for *price*, and 48.09 % for *pilling* with XGBoost; and that a CatBoost meta-learner trained on the assembled predictions attains a test F1-Macro of 41.52 % for *usage* versus 65.07 % achieved by the base model trained on metadata. The entire process emitted 2.39 kg of CO₂, demonstrating that it is possible to combine accuracy and sustainability in the classification of second-hand garments.

keywords: clothing, second-hand garments, circular fashion, Green AI, carbon footprint; computer vision, Local Binary Pattern Uniform (LBP-U), machine learning, stacking, meta-learner.

Índice general

Resumen	ii
Abstract	iii
1 Introducción	1
1.1 Definición del problema	1
1.2 Rol de la inteligencia artificial en la Economía Circular	2
1.3 Motivación	3
1.4 Objetivos e hipótesis	3
1.5 Estructura del documento	4
2 Teoría y Estado del arte	5
2.1 Green AI	5
2.1.1 Medición de emisiones con CodeCarbon	6
2.2 Computer Vision	7
2.2.1 Tareas fundamentales	7
2.2.2 Representación digital de imágenes	8
2.2.3 Preprocesamiento clásico de imágenes	8
2.2.4 Descriptores visuales	9
2.3 Machine Learning	11

2.3.1	Proceso de entrenamiento, validación y test	12
2.3.2	Overfitting y underfitting	13
2.3.3	Análisis de relevancia de variables	13
2.3.4	Algoritmos de aprendizaje supervisado	14
2.3.5	Stacking y Meta Learners	16
2.3.6	Interpretabilidad de los modelos: SHAP Values	17
2.3.7	Métricas de evaluación	18
2.4	Estado del Arte	21
3	Dataset	25
3.1	Contexto	25
3.2	Descripción del dataset	26
3.3	Generación de los metadatos	26
4	Metodología	28
4.1	Visión general del pipeline	28
4.2	Fase 1: Análisis exploratorio de los datos y selección de <i>k</i> - <i>features</i>	30
4.2.1	Revisión general de los metadatos	30
4.2.2	Limpieza, transformación y distribución de variables (<i>feature engineer</i>)	32
4.2.3	Selección preliminar de variables explicativas (<i>feature selection</i>)	36
4.2.4	Ánálisis de variable objetivo	36
4.2.5	División del conjunto de datos	39
4.2.6	Entrenamiento inicial de modelo base y selección final de variables (<i>k</i> - <i>features</i>)	40
4.2.7	Optimización y re-entrenamiento de modelo base con <i>k</i> - <i>features</i>	45
4.2.8	Resultados entrenamiento y selección final de modelo base	46

4.3	Fase 2: Análisis exploratorio de las imágenes y entrenamiento de sub-modelos visuales	48
4.3.1	Análisis exploratorio y preprocesamiento de las imágenes	48
4.3.2	Extracción de características visuales	54
4.3.3	Selección de algoritmos, métricas de evaluación y cálculos de eficiencia para sub-modelos	56
4.3.4	Búsqueda de hiperparámetros y entrenamiento de sub-modelos	57
4.3.5	Resultados entrenamiento selección de sub-modelos	57
4.4	Fase 3: Generación de <i>meta-features</i> y ensamblado del <i>meta-learner</i> .	61
4.4.1	Generación de <i>meta-features</i>	61
4.4.2	Ensamblado (<i>staking</i>) y configuración de <i>meta-learner</i>	64
4.4.3	Resultados finales del <i>meta-learner</i>	65
5	Conclusiones	69
5.1	Estimación de la huella de carbono	69
5.2	Conclusiones	70
5.3	Líneas de investigación futuras	71
5.4	Consideraciones éticas y sociales	72
5.5	Costo económico del proyecto	73
Apéndices		79
A	Variables	80
A.1	Lista de variables	80
A.2	Imagen ejemplo por categoría original de <i>usage</i>	86
B	Sub-modelos visuales	88
B.1	Entrenamiento de sub-modelos con categorías de variables originales .	88

B.2 Entrenamiento de sub-modelo visual para <i>usage</i>	90
--	----

Listas de figuras

1.1	Vertedero de ropa de Alto Hospicio, Chile, en desierto de Atacama. Fuente: La Nación, 2023.	2
2.1	Ejemplo de tareas de visión por computadora. Fuente: Loy, 2019.	7
2.2	Imagen digital en escala de grises representada como una matriz bidimensional. Fuente: Procesamiento de Imágenes, Máster Data Science, La Salle Universitat Ramón Llull (2024-2025).	8
2.3	Tres ejemplos de vecindarios utilizados para definir una textura y calcular un patrón binario local (LBP). Fuente: https://en.wikipedia.org/wiki/Local_binary_patterns .	10
2.4	Ejemplo de aplicación de LBP: imagen original (izquierda), imagen transformada con el descriptor LBP (centro) y distribución de los patrones binarios locales en forma de histograma (derecha). Fuente: Procesamiento de Imágenes, Máster Data Science, La Salle Universitat Ramón Llull (2024-2025).	10
2.5	Proceso de extracción del descriptor HOG: (a) imagen original, (b) mapa de gradientes, (c) vectores de orientación, (d) celdas y bloques normalizados, y (e) histograma de gradientes por bloque. Fuente: Procesamiento de Imágenes, Máster Data Science, La Salle Universitat Ramón Llull (2024-2025).	11
2.6	Ilustración de (a) aprendizaje supervisado y (b) no supervisado. Fuente: Simeone, 2018.	12
2.7	Estructura de <i>stacking</i> y <i>meta learner</i> . Fuente: Inteligencia Artificial para la Ciencia de Datos, Máster Data Science, La Salle Universitat Ramón Llull (2024-2025).	17

2.8	Ejemplo gráfica de análisis de interpretabilidad de SHAP Values (Trevisan, 2022).	18
2.9	Matriz de Confusión multiclase (Grandini et al., 2020).	19
2.10	Matriz de Confusión dos clases (Grandini et al., 2020).	20
2.11	Propuesta de pipeline en la investigación de Sen et al. (2025).	22
2.12	Propuesta de pipeline en la investigación de Seçkin et al. (2023).	23
2.13	a) Imagen CAD del sistema de cámara para escaneo textil. b) Imagen de ejemplo de una prenda escaneada y un defecto en la tela. Weimer et al. (2024).	24
3.1	Ejemplo de imágenes de una prenda y sus principales variables. Station: 1, Brand: SomeWear, Category: Unisex, Type: Sweater, Size: XL, Colors: White, Yellow Price: <50 SEK, Use: Export, Trend: None, Material: 79 % cotton, 21 % polyester, Holes: None, Stains: Minor, Smell: Minor, Pilling: 3, Condition: 3. Fuente: Elaboración propia. . .	26
3.2	Pipeline desarrollado para la generación de los metadatos.	27
4.1	Pipeline fase 1: EDA datos, selección de <i>k-features</i> y selección de mejor modelo base inicial.	29
4.2	Pipeline fase 2: EDA imágenes, extracción de características visuales y selección de mejor sub-modelo visual para cada <i>k-feature</i>	29
4.3	Fase 3: Generación de <i>meta-features</i> , ensamblado y <i>train/test</i> modelo base y <i>meta-learner</i>	30
4.4	Estructura tabular metadatos.	31
4.5	Distribución de <i>category</i> , <i>type</i> , <i>colors</i> , <i>season</i> , <i>pilling</i> y <i>condition</i>	33
4.6	Distribución de <i>price</i> , <i>cut</i> , <i>pattern</i> , <i>smell</i> , <i>stains</i> y <i>holes</i>	34
4.7	Distribución de <i>trends</i> , <i>brand</i> , <i>size</i> y <i>usage</i>	35
4.8	Distribuciones de imágenes por vista <i>station</i> y <i>gold</i>	38
4.9	Registros por categoría en <i>usage</i>	39
4.10	Matriz de confusión LightGBM.	43

4.11	Importancia de variables con SHAP Values.	44
4.12	Matriz de confusión para Random Forest (CW) y CatBoost.	48
4.13	Distribución de resolución por vista de imágen.	49
4.14	Ejemplos de imágenes por tipo de fondo.	49
4.15	Análisis de brillo medio.	50
4.16	Ejemplos de <i>outliers</i> en análisis de brillo medio.	51
4.17	Distribución de medias RGB por canal y vista	51
4.18	Ejemplos de <i>outliers</i> en análisis de color RGB.	52
4.19	Distribuciones Otsu y Laplacian.	53
4.20	Ejemplos de <i>outliers</i> para Otsu y Laplacian.	53
4.21	Prueba de parámetros para LBP-U en prenda con <i>condition</i> nivel 3 -para vista frontal y posterior-, en escala de grises, blur = 1 y ksize = 5,5.	56
4.22	Matriz de confusión para Random Forest (<i>condition</i>), XGBoot (<i>price</i>) y XGBoost (<i>pilling</i>) en validación.	60
4.23	Matriz de confusión para Random Forest (<i>condition</i>), XGBoot (<i>price</i>) y XGBoost (<i>pilling</i>) en <i>test</i> .	64
4.24	Matriz de confusión para modelo base, modelo base híbrido, <i>meta-learner</i> híbrido y <i>meta-learner</i> sobre <i>usage</i> .	68
A.1	Ejemplo de prenda etiquetada como <i>Reuse</i> .	86
A.2	Ejemplo de prenda etiquetada como <i>Export</i> .	86
A.3	Ejemplo de prenda etiquetada como <i>Recycle</i> .	86
A.4	Ejemplo de prenda etiquetada como <i>Remake</i> .	87
A.5	Ejemplo de prenda etiquetada como <i>Repair</i> .	87
A.6	Ejemplo de prenda etiquetada como <i>Energy Recovery</i> .	87
B.1	Matriz de confusión de Random Forest para <i>condition</i> , <i>price</i> y <i>pilling</i> , con sus categorías originales en validación.	90

B.2 Matriz de confusión de Random Forest y XGBoost para <i>usage</i> en validación.	92
---	----

Listas de tablas

2.1 Resumen comparativo de algoritmos de aprendizaje no supervisado utilizados.	16
4.1 Variables por categorías de datos.	31
4.2 Resultados obtenidos de las pruebas y análisis de relevancia.	37
4.3 División de los registros en los conjuntos de entrenamiento, validación y prueba.	40
4.4 Configuración de búsqueda y selección de hiperparámetros para modelo base inicial con <i>metadata</i>	42
4.5 Métricas de evaluación modelo base inicial.	42
4.6 Reporte de clasificación LightGBM.	43
4.7 Métricas de eficiencia computacional de algoritmos para modelo base inicial.	43
4.8 Categorías originales y reducidas de las <i>k-features</i> seleccionadas.	45
4.9 Selección de mejores hiperparámetros para modelo base entrenado con <i>k-features</i> seleccionadas.	46
4.10 Métricas de evaluación de algoritmos para modelo base con <i>k-features</i> seleccionadas.	47
4.11 Reporte de clasificación Random Forest (CW) para <i>k-features</i> originales.	47
4.12 Reporte de clasificación CatBoost para <i>k-features</i> reducidas.	47
4.13 Métricas de eficiencia computacional modelo base con <i>k-features</i> seleccionadas.	48

4.14	Grilla de parámetros y resultados para LBP-U.	54
4.15	Grilla de parámetros y resultados para filtro gaussiano.	55
4.16	Resultados F1 Macro para experimentos de LBP y LBP+HOG. . . .	55
4.17	Configuración de búsqueda y selección de hiperparámetros para sub-modelos.	58
4.18	Métricas de evaluación sub-modelos en validación.	59
4.19	Reporte de clasificación Random Forest para <i>condition</i> en validación. . . .	59
4.20	Reporte de clasificación XGBoost para <i>price</i> en validación.	59
4.21	Reporte de clasificación XGBoost para <i>pilling</i> en validación.	60
4.22	Métricas de eficiencia computacional de sub-modelos en validación. . . .	61
4.23	Métricas de eficiencia computacional en generación de <i>meta-features</i> para conjunto de entrenamiento (OOF).	62
4.24	Métricas de evaluación de sub-modelos en <i>test</i>	62
4.25	Reporte de clasificación Random Forest para <i>condition</i> en <i>test</i>	63
4.26	Reporte de clasificación XGBoost para <i>price</i> en <i>test</i>	63
4.27	Reporte de clasificación XGBoost para <i>pilling</i> en <i>test</i>	63
4.28	Métricas de eficiencia computacional de sub-modelos en <i>test</i>	63
4.29	Matriz de comparación de escenarios modelo base vs <i>meta-learner</i>	66
4.30	Reporte de clasificación modelo base Catboost en <i>usage</i>	67
4.31	Reporte de clasificación modelo base híbrido Catboost en <i>usage</i>	67
4.32	Reporte de clasificación <i>meta-learner</i> híbrido Catboost en <i>usage</i>	67
4.33	Reporte de clasificación <i>meta-learner</i> Catboost en <i>usage</i>	67
4.34	Métricas de eficiencia computacional para modelo base, modelo base híbrido, <i>meta-learner</i> híbrido y <i>meta-learner</i> sobre <i>usage</i>	68
5.1	Emisiones de CO ₂ equivalente por fase de desarrollo.	70

5.2	Emisiones de CO ₂ equivalentes de entrenar modelos de NLP comunes (Strubell et al., 2019) y el <i>meta-learner</i> de esta investigación.	70
5.3	Horas dedicadas por actividad	73
5.4	Coste total del proyecto	74
A.1	Listas de las variables de la metadata con su descripción y resumen de su preprocesamiento.	80
B.1	Métricas de evaluación sub-modelos con sus categorías originales en validación.	88
B.2	Reporte de clasificación Random Forest para <i>condition</i> con categorías originales en validación.	89
B.3	Reporte de clasificación Random Forest para <i>price</i> con categorías originales en validación.	89
B.4	Reporte de clasificación Random Forest para <i>pilling</i> con categorías originales en validación.	89
B.5	Métricas de evaluación sub-modelo <i>usage</i> en validación.	91
B.6	Reporte de clasificación XGBoost para <i>usage</i> en validación.	91
B.7	Reporte de clasificación Random Forest para <i>usage</i> en validación. . .	91

Capítulo 1

Introducción

1.1 Definición del problema

La industria textil ha experimentado un crecimiento acelerado durante las últimas dos décadas, impulsado principalmente por el modelo *fast fashion* que promueve la producción masiva y el consumo excesivo de prendas a bajo costo. En los últimos 15 años, la producción de ropa se ha duplicado, mientras que su vida útil media se ha reducido en un 36 %, alcanzando hasta un 70 % en mercados como China. Este patrón de consumo conlleva graves problemas de residuos: el 73 % de las prendas producidas terminan como desecho y menos del 1 % de los materiales se recicla para nuevas prendas, generando pérdidas superiores a los 100 billones de dólares anuales en valor de materiales. El impacto ambiental es igual de alarmante: la industria textil emite alrededor de 1,2 millones de toneladas de CO₂ equivalente al año -superando las emisiones conjuntas de todos los vuelos internacionales y de transporte marítimo- y es la segunda mayor consumidora de agua a nivel mundial, necesitando hasta 7,500 litros de agua para producir un solo par de jeans, equivalente al consumo doméstico de una persona durante siete años (Ellen MacArthur Foundation, 2017).

En el desierto de Atacama, al norte de Chile, existe un cementerio de ropa usada que ejemplifica la crisis del desperdicio textil (Figura 1.1). Cada año llegan a esta zona alrededor de 40,000 toneladas de prendas *fast fashion* desechadas, provenientes principalmente de Estados Unidos y Europa, convirtiéndolo en el mayor vertedero de ropa del mundo (Alonso, 2025).

Por otro lado, el mercado global de ropa de segunda mano está experimentando un crecimiento exponencial, impulsado por el cambio de hábitos de los consumidores y por plataformas de reventa en línea líderes como Vinted, ThredUp, Wallapop, entre

otras. En 2023 el mercado alcanzó un volumen aproximado de 197 billones de dólares, tras crecer un 18 %, y se proyecta que triplique su valor para el 2028, llegando a los 350 billones de dólares (ThredUp, 2024).

Esta transición hacia una moda circular está dando lugar al desarrollo de nuevos modelos de negocios, enfocados en gestionar de forma sostenible la ropa post-consumo, y de nuevas empresas especializadas en recoger, clasificar y revender prendas usadas a gran escala. En este escenario, tecnologías emergentes como la inteligencia artificial, con sus distintas aplicaciones, cumplen un papel fundamental para potenciar el diseño y desarrollo de soluciones innovadoras para una economía más sostenible.



(a) Foto satelital del vertedero. (b) Una mujer buscando ropa en el vertedero.

Figura 1.1: Vertedero de ropa de Alto Hospicio, Chile, en desierto de Atacama. Fuente: La Nación, 2023.

1.2 Rol de la inteligencia artificial en la Economía Circular

La inteligencia artificial se posiciona como una herramienta clave para acelerar la transición hacia una economía circular, ofreciendo soluciones que optimizan el diseño, el uso y el fin de vida de los productos (Ellen MacArthur Foundation & Google, 2019). En el ámbito textil, técnicas de *Computer Vision* permiten identificar con alta precisión características clave de una prenda —como presencia de roturas, manchas, desgaste, color o tipo de tejido— que, integradas en modelos de *Machine Learning* de aprendizaje supervisado, posibilitan su clasificación automática según su estado por ejemplo. Estas tecnologías, combinadas con sensores avanzados y sistemas robóticos, hacen posible una gestión más eficiente y escalable de los flujos de ropa de segunda mano, sustituyendo procesos manuales de clasificación que hoy prevalecen en muchas instalaciones. A su vez, si son diseñados bajo un enfoque de *Green AI* o Algoritmos Verdes, permiten desarrollar y entrenar modelos que minimicen su huella de carbono,

garantizando que la tecnología que impulsa la economía circular también actúe en coherencia con los principios de sostenibilidad que la inspiran.

1.3 Motivación

Vivimos aún bajo el paradigma de un modelo económico lineal de extraer, producir y desechar que consume recursos finitos y devuelve al planeta montañas de residuos. Este esquema está dejando una huella que vemos día a día en los titulares sobre contaminación, pérdida de biodiversidad y crisis climática. Transformar este paradigma hacia una economía circular no es solo una alternativa deseable, es imperativo.

Durante décadas, imaginar e implementar modelos circulares que sean sostenibles en el tiempo parecía un ideal lejano, limitado por la falta de tecnología y las capacidades de gestionarlos a gran escala. Hoy, sin embargo, las nuevas herramientas que ofrece la inteligencia artificial nos acercan a ese camino. La transición hacia una economía circular no es solo viable, si no que puede convertirse en una ventaja competitiva única para quienes sepan integrar innovación tecnológica, eficiencia y propósito en el diseño de nuevas soluciones (Lacy & Rutqvist, 2015).

En cada iniciativa profesional y académica en la que he participado, lo he hecho bajo la convicción de aportar, aunque sea en un pequeño paso, hacia el diseño de una economía más sostenible. Esta investigación, al contar con una base de datos única de prendas de segunda mano, junto con el conocimiento en modelos circulares y técnicas avanzadas de inteligencia artificial, abre la oportunidad de desarrollar una solución que optimice la clasificación de ropa usada y aporte un valor real a la cadena textil circular.

1.4 Objetivos e hipótesis

El presente trabajo de investigación tiene como objetivo principal desarrollar un *meta-learner* capaz de clasificar el destino final de prendas de segunda mano (p.ej. reutilización, exportación, reciclaje o valorización energética) a partir de técnicas de *Computer Vision* y *Machine Learning*.

Para alcanzar este objetivo, se plantean cuatro objetivos específicos. En primer lugar, identificar las características clave de una prenda (p. ej., condición, nivel de *pilling*, precio, color, materialidad, entre otras) que permitan predecir su destino final dentro

de la cadena de valor textil. En segundo lugar, investigar y seleccionar las técnicas más eficaces de *Computer Vision* para la extracción de *features* y los algoritmos de *Machine Learning* supervisado más adecuados para el caso de uso, evaluando su rendimiento de forma comparativa. En tercer lugar, desarrollar sub-modelos visuales que generen predicciones de dichos atributos y ensamblarlos para entrenar y evaluar el meta-learner. Por último, aplicar un enfoque de *Green AI*, midiendo y minimizando la huella de carbono generada durante los procesos de extracción, entrenamiento e inferencia, de manera que la tecnología desarrollada esté alineada con los principios de sostenibilidad que sustentan la economía circular.

Como hipótesis de trabajo se plantea que la aplicación de técnicas de *Computer Vision* y *Machine Learning*, integradas en un esquema de ensamblado (*stacking*) y entrenadas con características (*features*) seleccionadas de la metadata, que luego son extraídas por descriptores visuales desde las imágenes de prendas de segunda mano, permitirá clasificar su destino final (p.ej. reutilización, reciclaje, exportación, valorización energética), alcanzando niveles de rendimiento comparables a los de un clasificador humano experto.

Asimismo, se espera que la incorporación de un enfoque de *Green AI* no solo reduzca la huella de carbono del proceso de entrenamiento, sino que demuestre que es posible desarrollar modelos de alto rendimiento técnico sin comprometer los principios de sostenibilidad que sustentan la economía circular.

1.5 Estructura del documento

La investigación se organiza en cinco capítulos que guían al lector desde los fundamentos hasta la puesta en práctica y cierre del estudio. Tras esta introducción, el Capítulo 2 desarrolla el marco teórico y el estado del arte, presentando los principios de *Green AI*, las técnicas de visión por computadora y los algoritmos de machine learning para la clasificación supervisada de prendas. El Capítulo 3 describe el conjunto de datos empleado: su origen, estructura y el proceso de unificación de los metadatos. A continuación, el Capítulo 4 explica la metodología seguida en tres etapas —selección de variables, extracción de descriptores visuales y ensamblado de sub-modelos en un meta-learner—, detallando los experimentos, las decisiones de diseño y las métricas de evaluación. Finalmente, el Capítulo 5 cierra la investigación con la estimación total de la huella de carbono del pipeline, las conclusiones sobre los objetivos cumplidos, las líneas futuras, las consideraciones éticas y sociales y el análisis del coste económico del proyecto.

Capítulo 2

Teoría y Estado del arte

Este capítulo desarrolla los fundamentos teóricos y técnicos que sustentan la investigación. Se presentan los conceptos clave asociados a la sostenibilidad en inteligencia artificial bajo el enfoque *Green AI*, las bases del análisis de imágenes mediante técnicas de *Computer Vision*, y los principales algoritmos de *Machine Learning* utilizados para la clasificación supervisada y sus métricas de evaluación. Finalmente, se revisan estudios previos relacionados con la clasificación de prendas de segunda mano, integrando metadatos e información visual.

2.1 Green AI

La creciente demanda de recursos computacionales para entrenar modelos de inteligencia artificial ha generado una preocupación por su impacto ambiental, especialmente en términos de emisiones de CO₂. Frente a este escenario, emerge el concepto de *Green AI*, una aproximación que busca desarrollar modelos de alto rendimiento sin comprometer la sostenibilidad del planeta (Schwartz et al., 2019). Este enfoque propone priorizar tanto la precisión como la eficiencia computacional, incorporando como métrica de evaluación el coste energético y la huella de carbono asociada a los procesos de entrenamiento y despliegue de modelos.

En contraposición, el enfoque dominante hasta ahora ha sido el de la *Red AI*, centrado en maximizar el rendimiento de los modelos sin considerar los costes computacionales involucrados (Schwartz et al., 2019). Este enfoque ha impulsado la carrera por modelos cada vez más grandes y complejos, como GPT o BERT, entrenados con billones de parámetros y una enorme demanda energética. Si bien han logrado avances significativos en precisión, también han evidenciado una huella de carbono

desproporcionada.

Desde una perspectiva teórica, se ha evidenciado que los modelos de *Deep Learning* suelen implicar mayores costes computacionales debido a su arquitectura compleja, tiempos de entrenamiento excesivos y requerimientos de *hardware* intensivos. En contraste, algoritmos tradicionales de *Machine Learning* presentan una menor huella de carbono, ya que demandan menos recursos computacionales y tiempos de cómputo más bajos, sin necesariamente comprometer el rendimiento en tareas específicas (Barbierato & Gatti, 2024).

Para facilitar este camino, han surgido diversas herramientas que permiten cuantificar el impacto ambiental del desarrollo de modelos de inteligencia artificial. Estas incluyen librerías como *CodeCarbon*, *Carbon Tracker* o Green Algorithms que estiman las emisiones de CO₂ durante el entrenamiento de modelos, considerando variables como el tipo de hardware, la duración del cómputo y la intensidad energética del país o región.

2.1.1 Medición de emisiones con CodeCarbon

CodeCarbon es una librería de código abierto diseñada para estimar las emisiones de carbono generadas durante el desarrollo de modelos. Su funcionalidad principal consiste en monitorear en tiempo real el uso de CPU, GPU y memoria, y calcular la huella de carbono resultante en función del consumo energético y la fuente eléctrica del país o región donde se ejecuta el experimento (CodeCarbon, 2024).

El cálculo se basa en factores de emisión obtenidos de bases de datos como *ElectricityMap* y la *Agencia Internacional de Energía (IEA)*, lo que permite estimar el dióxido de carbono equivalente (kgCO₂eq) asociado al consumo computacional. La fórmula utilizada por *CodeCarbon* es:

$$\text{CO}_2\text{equivalente} = C \times E \quad (2.1)$$

donde:

- C : Intensidad de carbono de la red eléctrica consumida en la computación (gCO₂/kWh).
- E : Energía consumida por la infraestructura computacional (kWh).

Esta métrica permite comparar la eficiencia energética y el impacto ambiental de

diferentes modelos y configuraciones, promoviendo así el desarrollo de soluciones de inteligencia artificial más sostenibles bajo el enfoque de *Green AI*.

2.2 Computer Vision

La visión por computadora o *Computer Vision* es una rama de la inteligencia artificial que estudia cómo los sistemas informáticos pueden adquirir, procesar, analizar y comprender las imágenes del mundo real para extraer información útil. Su objetivo principal es automatizar tareas que el sistema visual humano puede realizar, como la identificación de objetos, el seguimiento de movimientos o la reconstrucción de escenarios (Szeliski, 2022).

2.2.1 Tareas fundamentales

Las tareas en visión por computadora se agrupan comúnmente en tres categorías principales: clasificación, detección de objetos y segmentación. La clasificación de imágenes consiste en asignar una etiqueta única a una imagen completa, identificando la clase predominante presentada en ella. La detección de objetos extiende la capacidad de clasificación, al localizar múltiples objetos dentro de una misma imagen. Utiliza estructuras como cajas delimitadoras o *bounding boxes* para identificar tanto la clase como la posición de cada objeto. Finalmente, la segmentación de imágenes busca una comprensión aún más detallada al clasificar cada píxel de una imagen, ya sea a nivel de clase (segmentación semántica) o por objeto específico (segmentación por instancia) (Szeliski, 2022).

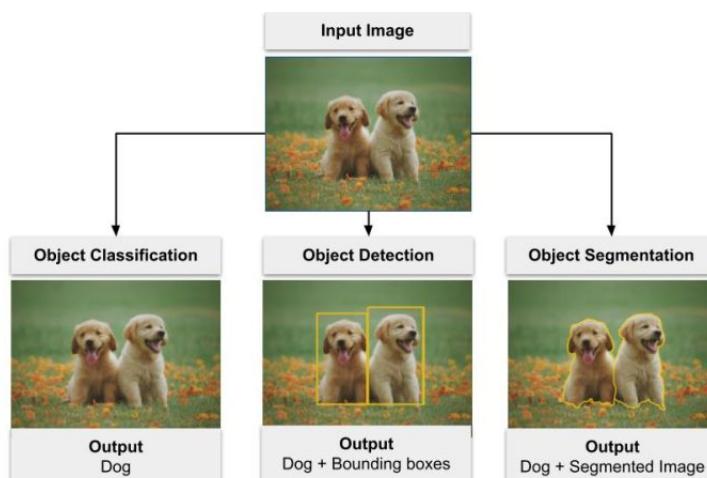


Figura 2.1: Ejemplo de tareas de visión por computadora. Fuente: Loy, 2019.

2.2.2 Representación digital de imágenes

En visión por computador, una imagen digital se representa como una matriz bidimensional en la que cada elemento, denominado píxel, almacena un valor numérico que describe la intensidad luminosa o el color en una posición determinada (Szeliski, 2022). En las imágenes en escala de grises, cada píxel contiene un único valor de intensidad, mientras que en imágenes en color se emplean modelos como RGB, donde tres canales —rojo, verde y azul— combinan sus valores para representar una amplia gama cromática (Gonzalez & Woods, 2018). La elección del espacio de color, como RGB, HSV o CIELAB, afecta directamente la eficacia de los algoritmos de análisis y debe seleccionarse según las características de la tarea (Szeliski, 2022). Asimismo, el preprocesado de imágenes —que incluye operaciones como la normalización, el filtrado de ruido o el ajuste de contraste— es fundamental para mejorar la calidad de la información y optimizar la posterior extracción de características (Gonzalez & Woods, 2018).

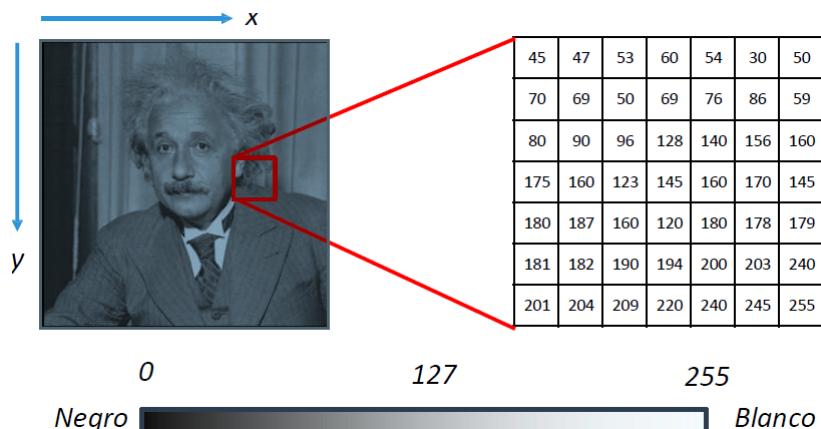


Figura 2.2: Imagen digital en escala de grises representada como una matriz bidimensional.
Fuente: Procesamiento de Imágenes, Máster Data Science, La Salle Universitat Ramón Llull (2024-2025).

2.2.3 Preprocesamiento clásico de imágenes

El procesamiento clásico de imágenes comprende un conjunto de técnicas que permiten mejorar la calidad visual de las imágenes y facilitar su análisis computacional posterior. Estas técnicas operan directamente sobre la representación matricial de la imagen, aplicando transformaciones locales mediante máscaras o kernels convolucionales (Gonzalez & Woods, 2018).

Una de las herramientas más utilizadas en este contexto es el filtro gaussiano, un filtro de suavizado que reduce el ruido preservando estructuras relevantes. Su

funcionamiento se basa en ponderar los píxeles vecinos según una distribución normal, otorgando mayor peso al píxel central. Este filtro es ampliamente utilizado como etapa previa a la extracción de características, ya que mejora la uniformidad de la imagen de entrada sin distorsionar sus patrones esenciales (Szeliski, 2022).

Junto con los filtros de suavizado, existen otras técnicas comunes de preprocesamiento, como la normalización de intensidad, la conversión a escala de grises, el ajuste de contraste o la ecualización del histograma. Estas operaciones permiten homogeneizar las imágenes y reducir la variabilidad innecesaria entre muestras, favoreciendo la eficacia de los algoritmos de análisis posteriores (Gonzalez & Woods, 2018).

2.2.4 Descriptores visuales

Los descriptores visuales son herramientas fundamentales en visión por computador para representar y extraer información local relevante de una imagen. Capturan patrones, texturas y estructuras a nivel de píxel o vecindario, permitiendo describir visualmente regiones específicas mediante vectores de características que pueden ser utilizados en tareas como clasificación o detección (Ojala et al., 1996).

2.2.4.1 Local Binary Patterns (LBP) y LBP-Uniforme

El descriptor *Local Binary Patterns* (LBP) codifica la textura local de una imagen mediante la comparación entre la intensidad del píxel central y sus vecinos en una ventana, usualmente de 3×3 . Si el valor de un píxel vecino es mayor o igual al del centro, se asigna un 1; de lo contrario, un 0. El patrón resultante se convierte en un número binario, generando así un valor característico por píxel. La expresión matemática que lo representa es:

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c) \cdot 2^p \quad (2.2)$$

donde g_c representa el nivel de gris del píxel central, g_p corresponde a los niveles de gris de los P vecinos distribuidos en un radio R , y la función $s(x)$ se define como:

$$s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

Una extensión ampliamente utilizada es el *LBP uniforme* (LBP_U), que considera únicamente aquellos patrones con un máximo de dos transiciones entre 0 y 1 en la secuencia binaria circular. Esta variante reduce significativamente la dimensionalidad del descriptor —de 256 a 59 dimensiones en el caso clásico de $P = 8$ vecinos— y proporciona invariancia a la rotación, siendo especialmente útil para describir texturas complejas como las presentes en prendas textiles (Ojala et al., 1996; Pietikäinen et al., 2011).

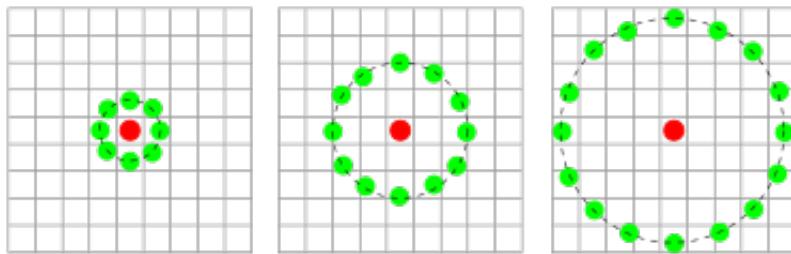


Figura 2.3: Tres ejemplos de vecindarios utilizados para definir una textura y calcular un patrón binario local (LBP). Fuente: https://en.wikipedia.org/wiki/Local_binary_patterns.



Figura 2.4: Ejemplo de aplicación de LBP: imagen original (izquierda), imagen transformada con el descriptor LBP (centro) y distribución de los patrones binarios locales en forma de histograma (derecha). Fuente: Procesamiento de Imágenes, Máster Data Science, La Salle Universitat Ramón Llull (2024-2025).

2.2.4.2 Histogram of Oriented Gradients (HOG)

El descriptor *Histogram of Oriented Gradients* (HOG) representa la distribución local de gradientes de intensidad en una imagen. La técnica consiste en dividir la imagen en celdas pequeñas y calcular, para cada una, un histograma de las orientaciones de los gradientes, ponderados por su magnitud. Estos histogramas se normalizan por bloques superpuestos, aumentando así la robustez frente a variaciones de iluminación y contraste. HOG ha demostrado ser eficaz en tareas como la detección de objetos y el reconocimiento de formas, ya que captura información estructural relevante (Dalal & Triggs, 2005).

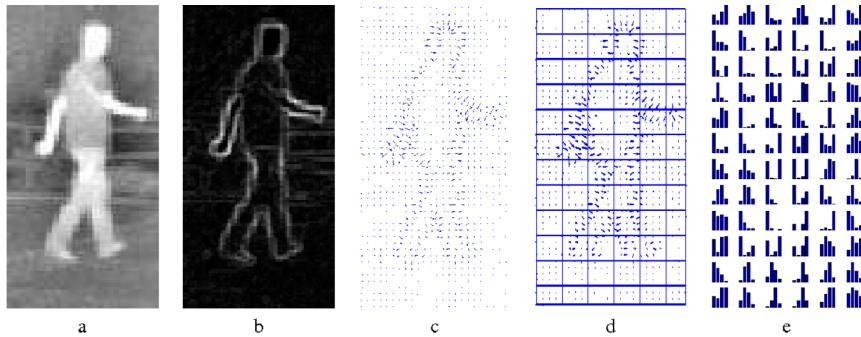


Figura 2.5: Proceso de extracción del descriptor HOG: (a) imagen original, (b) mapa de gradientes, (c) vectores de orientación, (d) celdas y bloques normalizados, y (e) histograma de gradientes por bloque. Fuente: Procesamiento de Imágenes, Máster Data Science, La Salle Universitat Ramón Llull (2024-2025).

2.3 Machine Learning

El aprendizaje automático o *Machine Learning* (ML por sus siglas en inglés) es una rama de la inteligencia artificial que se centra en el desarrollo de algoritmos capaces de aprender patrones a partir de los datos y de mejorar su desempeño sin ser programados explícitamente para cada tarea (Mitchell, 1997).

En lugar de seguir un conjunto fijo de instrucciones, los algoritmos de *Machine Learning* construyen modelos matemáticos a partir de datos de entrenamiento, con el objetivo de realizar predicciones o tomar decisiones sobre nuevos datos (Simeone, 2018). Este enfoque permite abordar problemas complejos en los que el modelado explícito resulta costoso o inviable, como el reconocimiento de imágenes, el procesamiento natural del lenguaje o la detección de anomalías.

El proceso general de *Machine Learning* implica varias etapas: recopilación y preparación de los datos, selección de un modelo, entrenamiento mediante optimización de una función de pérdida y evaluación de su rendimiento con datos no vistos (Janiesch et al., 2021). La calidad de los datos dependerá tanto de la cantidad y representatividad de los datos como la adecuación del algoritmo y los hiperparámetros seleccionados.

El *Machine Learning* abarca múltiples paradigmas, entre los que destacan el aprendizaje supervisado, el aprendizaje no supervisado y el aprendizaje por refuerzo. En el primero, el modelo recibe pares de entrada-salida con el objetivo de aprender la función que los relaciona. En el segundo, el modelo trabaja con datos sin etiquetas, buscando descubrir estructuras o patrones subyacentes, como en la agrupación (*clustering*) o la reducción de dimensionalidad. Por su parte, el último se basa en

un agente que interactúa con un entorno y aprende a tomar decisiones a través de un sistema de recompensas y penalizaciones (Kaur, 2021). En el contexto de esta investigación, el foco se sitúa en el aprendizaje supervisado.

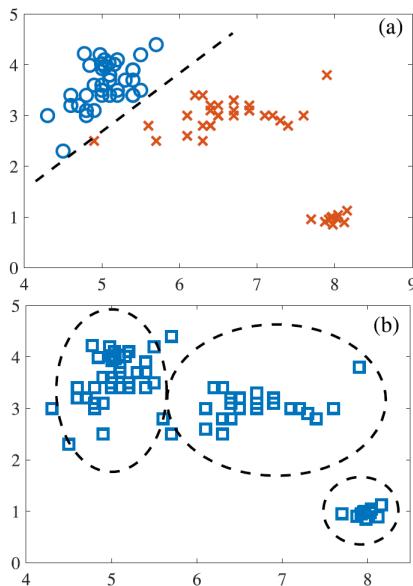


Figura 2.6: Ilustración de (a) aprendizaje supervisado y (b) no supervisado. Fuente: Simeone, 2018.

2.3.1 Proceso de entrenamiento, validación y test

El desarrollo de un modelo de *Machine Learning* implica la correcta división del conjunto de datos para garantizar una evaluación objetiva de su rendimiento. Generalmente, el *dataset* se divide en tres subconjuntos: entrenamiento (*train*), validación (*validation*) y pruebas (*test*). El conjunto de entrenamiento se utiliza para ajustar parámetros del modelo; el de validación para seleccionar hiperparámetros y prevenir sobreajuste; y el de prueba para estimar su capacidad de generalización (Géron, 2019).

Las proporciones más comunes son 60-20-20 o 70-15-15, aunque pueden variar en función del tamaño y naturaleza de los datos. En conjuntos pequeños o con clases desbalanceadas, es recomendable aplicar validación cruzada (*cross-validation*), técnica que reutiliza de forma eficiente los datos para entrenamiento y validación, reduciendo la varianza en la estimación del rendimiento (Janiesch et al., 2021).

Una división adecuada evita el *data leakage*, es decir, la filtración de información del conjunto de validación o prueba hacia el modelo durante el entrenamiento, lo que podría inflar artificialmente las métricas de rendimiento (Géron, 2019).

2.3.2 Overfitting y underfitting

En el entrenamiento de modelos, el sobreajuste (*overfitting*) se produce cuando el modelo aprende de patrones demasiado específicos del conjunto de entrenamiento, incluyendo ruido o variaciones irrelevantes, lo que provoca una disminución de su capacidad para generalizar datos nuevos. Por el contrario, el sobreajuste (*underfitting*) ocurre cuando el modelo es demasiado simple para capturar la complejidad de los datos, generando un bajo rendimiento tanto en el conjunto de entrenamiento como en el de prueba (Géron, 2019)

Existen diversas estrategias para mitigar estos problemas. Entre las más comunes se encuentran la regularización, que introduce penalizaciones para limitar la complejidad del modelo, y el *early stopping*, que interrumpe el entrenamiento cuando las métricas de validación dejan de mejorar (Géron, 2019).

Otra estrategia es el aprendizaje por ensamblado (*ensemble learning*), técnica utilizada en esta investigación, la que combina múltiples modelos para mejorar la capacidad predictiva y la robustez. El primer enfoque de estas técnicas es Bagging (*Bootstrap Aggregating*), método que entrena múltiples modelos independientes sobre subconjuntos aleatorios del conjunto de datos y promedia sus predicciones, reduciendo la varianza. Este enfoque es la base de algoritmos como Random Forest (Amat Rodrigo, s.f.). El segundo enfoque es Boosting, técnica secuencial en la que cada modelo intenta corregir los errores del anterior, asignando mayor peso a las observaciones mal clasificadas. Entre sus implementaciones se encuentran CatBoost (Prokhorenkova et al., 2018), XGBoost (Chen & Guestrin, 2016) y LightGBM (Ke et al., 2017), que optimizan tanto el rendimiento como la eficiencia en problemas de clasificación supervisada.

2.3.3 Análisis de relevancia de variables

La etapa de análisis de variables permite comprender la relación entre los predictores y la variable objetivo, guiando procesos posteriores como la selección de atributos, el diseño de modelos o la interpretación de resultados. Este análisis se vuelve especialmente relevante cuando se trabaja con múltiples variables categóricas, ya que muchas técnicas de aprendizaje automático no manejan eficientemente predictores irrelevantes o redundantes.

Para variables categóricas, se utilizan comúnmente pruebas estadísticas que miden distintos tipos de asociación. Una de las más utilizadas es el estadístico χ^2 (chi-

cuadrado), que evalúa si existe dependencia entre dos variables categóricas. A partir de este valor, es posible calcular el coeficiente de *Cramér's V*, que estandariza la magnitud de la asociación en un rango entre 0 y 1, permitiendo comparar directamente la fuerza de asociación entre variables con distintos números de categorías (Sheskin, 2011).

Además, desde la teoría de la información, métricas como la Información Mutua permiten estimar cuánta información se gana sobre la variable objetivo al conocer un predictor dado. Otras pruebas complementarias incluyen ANOVA (cuando se evalúan diferencias de medias para variables ordinales transformadas) y el test no paramétrico de Kruskal-Wallis, útil para detectar diferencias significativas en rangos de distribución (James et al., 2021).

Estas técnicas permiten tomar decisiones informadas sobre qué variables incorporar, transformar o eliminar, y son especialmente útiles en tareas de clasificación.

2.3.4 Algoritmos de aprendizaje supervisado

En el aprendizaje supervisado, la elección del algoritmo es un factor determinante para el rendimiento del modelo. Cada técnica presenta características particulares que influyen en su capacidad de generalizar, su eficiencia computacional y su adaptabilidad a distintos tipos de datos. Algunos algoritmos están diseñados para trabajar de forma nativa con variables categóricas, numéricas o ambas, lo que les otorga flexibilidad en contextos con datos heterogéneos. En esta investigación se emplearon cinco algoritmos representativos -Support Vector Machine (SVM), Random Forest, CatBoost, XGBoost y LightGBM- seleccionados por su eficacia en tareas de clasificación supervisada y su capacidad para manejar conjuntos de datos complejos y específicos.

2.3.4.1 Support Vector Machine (SVM)

Support Vector Machine busca el hiperplano que maximiza el margen de separación entre clases. Mediante el uso de funciones *kernel*, puede proyectar los datos a espacios de mayor dimensión, permitiéndole manejar relaciones no lineales. Aunque trabaja principalmente con variables numéricas, también puede utilizar datos categóricos tras una codificación previa. Sus parámetros claves para controlar el *overfitting* incluyen la constante de penalización (C), que regula el equilibrio entre margen y error, y el parámetro del *kernel* ($gamma$), que influye en la complejidad del modelo y su capacidad de generalización (Mammone et al., 2009).

2.3.4.2 Random Forest (RF)

Random Forest, algoritmo basado en la técnica de *bagging* (*Bootstrap Aggregating*), combina múltiples árboles de decisión entrenados sobre conjuntos aleatorios de datos y características. Cada árbol realiza una predicción independiente, y el resultado final se obtiene mediante votación mayoritaria en clasificación o promedio en regresión. Este enfoque reduce la varianza y mejora la capacidad de generalización respecto a un único árbol. RF puede trabajar con variables numéricas y categóricas sin requerir transformaciones extensas, y es robusto frente a valores atípicos y ruido. Entre sus parámetros más relevantes se incluyen el número de árboles (*n_estimators*), la profundidad máxima (*max_depth*) y el número de características en cada división (*max_features*), que permiten controlar el sobre ajuste y el rendimiento computacional (Amat Rodrigo, s.f.).

2.3.4.3 CatBoost

CatBoost, algoritmo basado en árboles de decisión, introduce mejoras específicas para el manejo eficiente de variables categóricas. Utiliza una técnica de codificación ordenada (*ordered boosting*), que transforma las variables categóricas en valores numéricos evitando el *target leakage*. Además, implementa un esquema de crecimiento simétrico de árboles (*oblivious trees*), lo que reduce el riesgo de sobre ajuste y mejora la velocidad de predicción. Sus parámetros claves incluyen el número de iteraciones (*iterations*), la tasa de aprendizaje (*learning_rate*) y la profundidad máxima de los árboles (*depth*), que permiten ajustar el equilibrio entre precisión, velocidad y capacidad de generalización (Prokhorenkova et al., 2018).

2.3.4.4 XGBoost

El algoritmo XGBoost (*Extreme Gradient Boosting*) es una implementación optimizada de *gradient boosting* sobre árboles de decisión, diseñada para maximizar el rendimiento y la eficiencia computacional. Introduce mejoras como la regularización explícita en la función objetivo para controlar el sobre ajuste, el manejo eficiente de valores perdidos y un esquema de paralelización que acelera el entrenamiento. Puede trabajar con variables numéricas y categóricas (tras codificación previa) y se adapta a grandes volúmenes de datos. Sus parámetros claves incluyen la tasa de aprendizaje (*eta*), la profundidad máxima de los árboles (*max_depth*) y el número de rondas de entrenamiento (*n_estimators*), así como parámetros de regularización (*lambda*, *alpha*) que equilibran sesgo y varianza (Chen & Guestrin, 2016).

2.3.4.5 LightGBM

LightGBM es una implementación de *gradient boosting* sobre árboles de decisión enfocada en alta eficiencia y escalabilidad. Introduce dos técnicas clave: GOSS (*Gradient-based One-Side Sampling*), que conserva instancias con gradientes altos y submuestra las de gradientes pequeños ajustando sus pesos para estimar con precisión la ganancia de división; y EFB (*Exclusive Feature Bundling*), que agrupa características mutuamente excluyentes para reducir dimensionalidad efectiva sin pérdida sustancial de información. Emplea un esquema *leaf-wise* con *histogram-based splitting*, lo que acelera la búsqueda de cortes y reduce memoria. Maneja datos numéricos y categorizados (tras codificación) y destaca por su rendimiento en grandes volúmenes de datos. Parámetros relevantes incluyen *num_leaves*, *max_depth*, *learning_rate*, *feature_fraction*, *bagging_fraction* y *min_data_in_leaf*, que regulan sobreajuste, velocidad y uso de memoria (Ke et al., 2017).

Tabla 2.1: Resumen comparativo de algoritmos de aprendizaje no supervisado utilizados.

Función	SVM	Random Forest	CatBoost	XGBoost	LightGBM
Parámetros para controlar overfitting	C, kernel, gamma	max_depth, min_samples_split, min_samples_leaf, max_features	learning_rate, depth, l2_leaf_reg	learning_rate, max_depth, min_child_weight	learning_rate, max_depth, num_leaves, min_data_in_leaf
Parámetros para categóricas	No aplica	No aplica	cat_features, one_hot_max_size	No aplica	categorical_feature
Parámetros para velocidad	max_iter, tipo de kernel	n_estimators, max_features, n_jobs	rsm, iterations	colsample_bytree, subsample, n_estimators	feature_fraction, bagging_fraction, num_iterations
Tipo de técnica	Método de margen	Bagging	Boosting	Boosting	Boosting
Ventajas clave	Robusto en alta dimensionalidad, buen separador no lineal	Maneja bien datos ruidosos y mixtos	Óptimo para categóricas sin preprocesar	Alta precisión y control fino del modelo	Entrenamiento rápido y bajo consumo de memoria

2.3.5 Stacking y Meta Learners

El *stacking* es una técnica de meta-aprendizaje propuesta por Wolpert (1992) que combina las predicciones de múltiples modelos base (*base learners* o sub-modelos) mediante un modelo de nivel superior denominado *meta learner*. Su objetivo es mejorar el rendimiento y la capacidad de generalización frente a los modelos individuales.

Este enfoque se estructura en dos niveles. En el nivel base, se entrena un conjunto de modelos o *-weak learners-* heterogéneos sobre los datos originales, pudiendo emplear algoritmos de distinta naturaleza para capturar diferentes patrones en la información.

En el nivel meta, el modelo recibe como entrada las predicciones generadas por los modelos base y aprende a combinarlas, optimizando la predicción final.

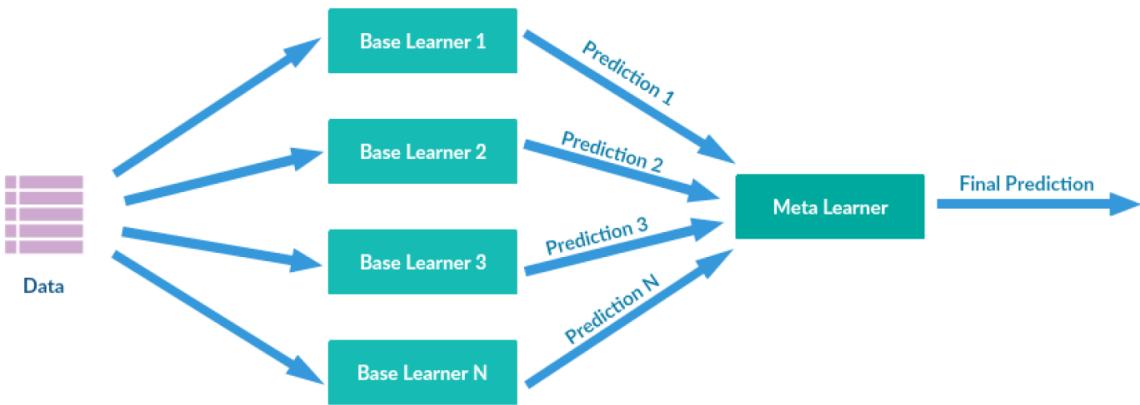


Figura 2.7: Estructura de *stacking* y *meta learner*. Fuente: Inteligencia Artificial para la Ciencia de Datos, Máster Data Science, La Salle Universitat Ramón Llull (2024-2025).

Estudios recientes, han extendido el *stacking* a contextos de aprendizaje multivista (*multiview learning*), en los que cada modelo base puede estar especializado en un subconjunto de características o en una representación distinta de los datos (Van Loon et al., 2024). En este enfoque, el *meta learner* no solo combina predicciones, si no que también selecciona de forma adaptativa las más relevantes, mejorando la robustez y precisión en entornos complejos y heterogéneos.

El *stacking* presenta beneficios clave, entre ellos, el incremento de la capacidad de generalización, al integrar múltiples perspectivas del problema; la reducción del riesgo de *overfitting* en comparación con un único modelo, gracias a la combinación de predicciones; y la posibilidad de aprovechar la complementariedad de modelos diversos para capturar patrones que podrían pasar inadvertidos a un algoritmo individual.

2.3.6 Interpretabilidad de los modelos: SHAP Values

La interpretabilidad en modelos de *Machine Learning* hace referencia a la capacidad de comprender y explicar las razones detrás de una predicción. Este aspecto es fundamental para generar confianza, detectar sesgos y mejorar el rendimiento del modelo. Sin embargo, a menudo existe un compromiso entre *accuracy* y explicabilidad. Los modelos simples son más fáciles de interpretar, mientras que los complejos, como redes neuronales profundas (*deep networks*) o ensamblados (*ensemble methods*), ofrecen mayor precisión a costa de menos transparencia (Lundberg & Lee, 2017).

SHAP (SHapley Additive exPlanations) es un marco unificado para la interpretación de predicciones, basado en los valores de Shapley de la teoría de juegos cooperativos. Este método asigna a cada característica un valor que cuantifica su contribución al resultado de una instancia, descomponiendo la predicción en un valor base más las aportaciones individuales (Lundberg & Lee, 2017).

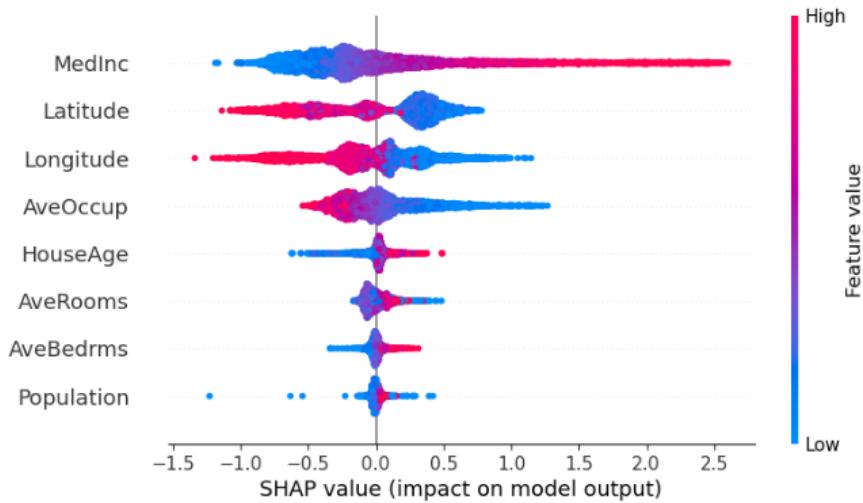


Figura 2.8: Ejemplo gráfica de análisis de interpretabilidad de SHAP Values (Trevisan, 2022).

2.3.7 Métricas de evaluación

En problemas de clasificación supervisada, especialmente en contexto de multiclase, la evaluación del rendimiento del modelo es fundamental para comparar alternativas y tomar decisiones informadas. Si bien métricas como el *accuracy* son ampliamente utilizadas por su simplicidad, estas pueden ofrecer una visión parcial en situaciones donde existen clases desbalanceadas o con distintas relevancia. Por esto, es esencial considerar un conjunto más de métricas que consideren tanto la calidad global como el comportamiento del modelo respecto a cada clase individual (Grandini et al., 2020).

2.3.7.1 Matriz de Confusión

La Matriz de Confusión es una tabla que resume el rendimiento de un modelo de clasificación al mostrar cuántas instancias fueron correctamente o incorrectamente clasificadas. En problemas multiclase, las filas indican las clases reales y las columnas predichas, siendo la diagonal principal la que refleja las predicciones correctas (Grandini et al., 2020).

		PREDICTED classification				Total
Classes		a	b	c	d	
ACTUAL classification	a	6	0	1	2	9
	b	3	9	1	1	14
	c	1	0	10	2	13
	d	1	2	1	12	16
	Total	11	11	13	17	52

Figura 2.9: Matriz de Confusión multiclas (Grandini et al., 2020).

2.3.7.2 Precision y Recall

La *precision* y el *recall* son métricas fundamentales derivadas de la matriz de confusión. En problemas de clasificación binaria, permiten evaluar qué tan bien el modelo distingue entre clases positivas y negativas, y constituyen la base para métricas más complejas como el *Balanced Accuracy* y *F1-Score* (Grandini et al., 2020).

La *precision* mide la proporción de instancias que el modelo predijo como positivas y que efectivamente lo son. Se interpreta como una medida de confiabilidad cuando el modelo predice una clase positiva.

$$\text{Precision} = \frac{TP}{TP + FP}$$

El *recall*, por otro lado, evalúa la capacidad del modelo para detectar todos los casos positivos reales, es decir, la fracción de verdaderos positivos sobre el total de positivos reales.

$$\text{Recall} = \frac{TP}{TP + FN}$$

Estas métricas pueden extenderse a contextos multiclas, calculando valores por clase y promediando posteriormente (*macro* o *weighted*), lo que se discutirá más adelante.

		PREDICTED		Total
Classes		Positive (1)	Negative (0)	
ACTUAL	Positive (1)	TP = 20	FN = 5	25
	Negative (0)	FP = 10	TN = 15	25
	Total	30	20	50

Figura 2.10: Matriz de Confusión dos clases (Grandini et al., 2020).

2.3.7.3 Accuracy y sus variantes

El *accuracy* mide la proporción de predicciones correctas sobre el total de instancias. Es una métrica simple e intuitiva, pero puede resultar poco informativa en *datasets* desbalanceados (Grandini et al., 2020).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.3)$$

Balanced Accuracy calcula el promedio del *recall* por clase, otorgando el mismo peso a cada una sin importar su frecuencia. Esta métrica resulta especialmente útil cuando existe un desbalance significativo entre clases, ya que evita que las mayoritarias dominen el resultado. *Weighted Balanced Accuracy* ajusta el *recall* de cada clase según su frecuencia en el dataset. Así, logra representar de manera proporcional la contribución de cada clase, conservando sensibilidad frente a las menos representadas.

2.3.7.4 F1-Score y sus variantes

El F1-Score evalúa el rendimiento de un modelo de clasificación combinando las métricas de *precision* y *recall* mediante la media armónica. Este indicador penaliza los modelos que presentan un desbalance entre ambas métricas, y resulta especialmente útil en contextos con clases desbalanceadas (Grandini et al., 2020). Se calcula como:

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

F1-Score Macro calcula la *precision* y el *recall* por clase y luego promedia ambos valores. De esta forma, todas las clases tienen el mismo peso, independientemente de su frecuencia. Esto lo hace particularmente adecuado para evaluar el rendimiento del

modelo sobre todas las clases por igual, incluyendo aquellas menos representadas.

F1-Score Weighted también calcula el F1 para cada clase, pero pondera su contribución según el número de instancias de dicha clase en el *dataset*. Así, las clases más frecuentes influyen más en la métrica final. Esta variante permite evaluar el rendimiento considerando tanto el balance de clases como el desempeño específico por clase.

F1-Score Micro agrega todos los verdaderos positivos, falsos positivos y falsos negativos a nivel global antes de calcular *precision* y *recall*. De este modo, se prioriza el rendimiento general sobre el conjunto completo, favoreciendo a las clases más representadas. En problemas de clasificación multiclase, esta métrica equivale directamente a la *accuracy*.

2.4 Estado del Arte

La identificación de características visuales en prendas de vestir ha sido una línea de investigación clave en la intersección entre visión por computador y el análisis de moda. Un estudio pionero en esta línea es el realizado por Lorenzo-Navarro et al. (2014), quienes compararon el rendimiento de distintos descriptores visuales -como HOG, LBP y color- para la clasificación de atributos específicos en ropa, utilizando como clasificadores modelos de SVM lineal y Random Forest. La investigación fue realizada sobre un conjunto de datos público que contenía 26 atributos relacionados con la vestimenta, como presencia de cuellos, largo de la manga y patrones de diseño. Los resultados mostraron que HOG superó consistentemente a LBP en atributos definidos por bordes y orientaciones (con un *accuracy* de hasta el 80 %), mientras que LBP resultó más adecuado para capturar texturas locales relacionadas en los atributos de patrón.

Recientes investigaciones han demostrado que los enfoques basados en *feature engineering* y modelos clásicos de *Machine Learning* siguen siendo altamente competitivos en clasificación de imágenes, especialmente cuando se prioriza la eficiencia computacional y la interpretabilidad frente a arquitecturas más profundas. Sen et al. (2025) desarrollaron un sistema que fusiona Permutación de Entropía (PE), HOG y LBP como descriptores visuales, alimentando un modelo SVM optimizado. En *datasets* como Fashion-MINIST, lograron un 91.15 % *accuracy* en test, superando a métodos clásicos (89.7 %) y aproximándose al rendimiento de redes neuronales profundas, pero con menor coste computacional.

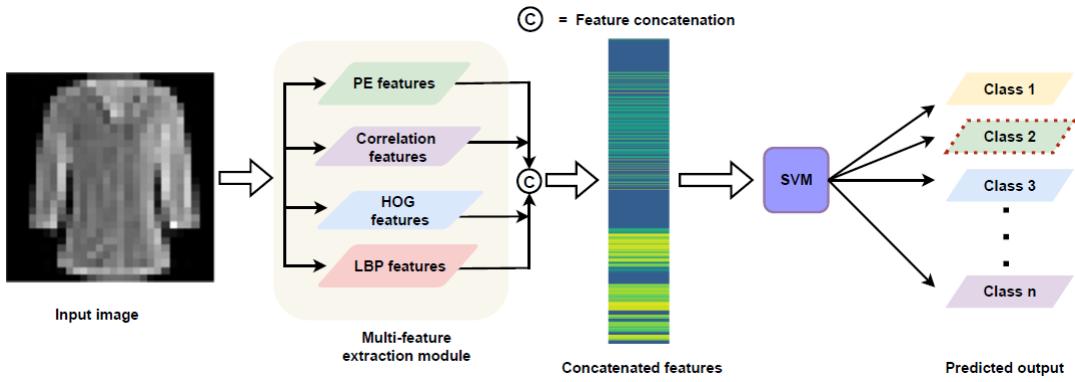


Figura 2.11: Propuesta de pipeline en la investigación de Sen et al. (2025).

A pesar del creciente uso de la visión por computadora en la industria de la moda, existe una brecha significativa en el desarrollo de *datasets* orientados al análisis de destino de ropa de segunda mano. Si bien existen numerosas bases de datos y más de 200 investigaciones para tareas como detección, recomendación y análisis de tendencias en prendas (Cheng et al., 2021), no se han identificado conjuntos públicos que permitan evaluar el estado de una prenda de segunda mano (Wargön Innovation AB et al., 2024). En respuesta a esta problemática, Mizunuma et al. (2024) desarrollaron un dataset compuesto por más de 14,000 imágenes tomadas desde múltiples ángulos en tiendas de segunda mano, las que fueron etiquetadas por expertos y no expertos como reutilizables o reciclables, con el objetivo de entrenar modelos que pudieran eliminar el fondo y posteriormente clasificarlas en su uso. Por otro lado, Wargön Innovation AB, en colaboración con RISE Research Institutes of Sweden AB y Myrorna AB, levantaron entre 2021 y 2024 el "*Clothing Dataset for Second-Hand Fashion*", primer *dataset* público de ropa de segunda mano, compuesto por más de 30,000 prendas (Nauman, 2024). Este es el *dataset* utilizado en la presente investigación, el que será explicado en profundidad en la siguiente sección.

Un caso especialmente relevante para esta investigación es la tesis de Hermansson (2023), que abordó la predicción del destino de ropa usada (reutilización o exportación) y del precio estimado, utilizando una versión inicial del *dataset* de Wargön Innovation y el conjunto Sellpy. El primero -que en esos momentos aun se encontraba en construcción- contenía 9,000 imágenes de 3,000 prendas (vistas frontal, posterior y etiqueta), mientras que el segundo incluía 100,000 prendas en condiciones más controladas. En la investigación, se compararon dos modelos: CLIP, de tipo multimodal, y MAE, auto-supervisado. CLIP fue el que mostró mejor rendimiento, logrando un F1-score de 59.04 % para la clasificación de destino (*usage* con *Reuse* y *Export* como clases) y de 38.08 % para la predicción de precio categorizado (*price*), marcando un punto de referencia relevante para este trabajo.

En cuanto al análisis de textura en prendas y el entrenamiento de modelos específicos según característica objetivo, una contribución destacada es el trabajo de Seçkin et al. (2023), quienes desarrollaron un *dataset* tabular a partir de imágenes de tejidos de prendas, extrayendo descriptores como Gray-Level Co-occurrence Matrix (GLCM), Gabor Filter Banck (GFB) y LBP, entre otros. A partir de esta matriz de características, entrenaron modelos como XGBoost, Random Forest y Multi layer Preception (MLP) para predecir parámetros textiles (p.ej. textura, masa específica, entre otros), empleando un enfoque de submodelos independientes por variable. Su sistema alcanzó una alta *accuracy* en clasificación de texturas, destacando XGBoost con una exactitud de 98.7 %. Si bien su trabajo no contempla una etapa de ensamblaje, su estructura modular y el uso de descriptores visuales clásicos dialogan estrechamente con la propuesta de esta investigación.

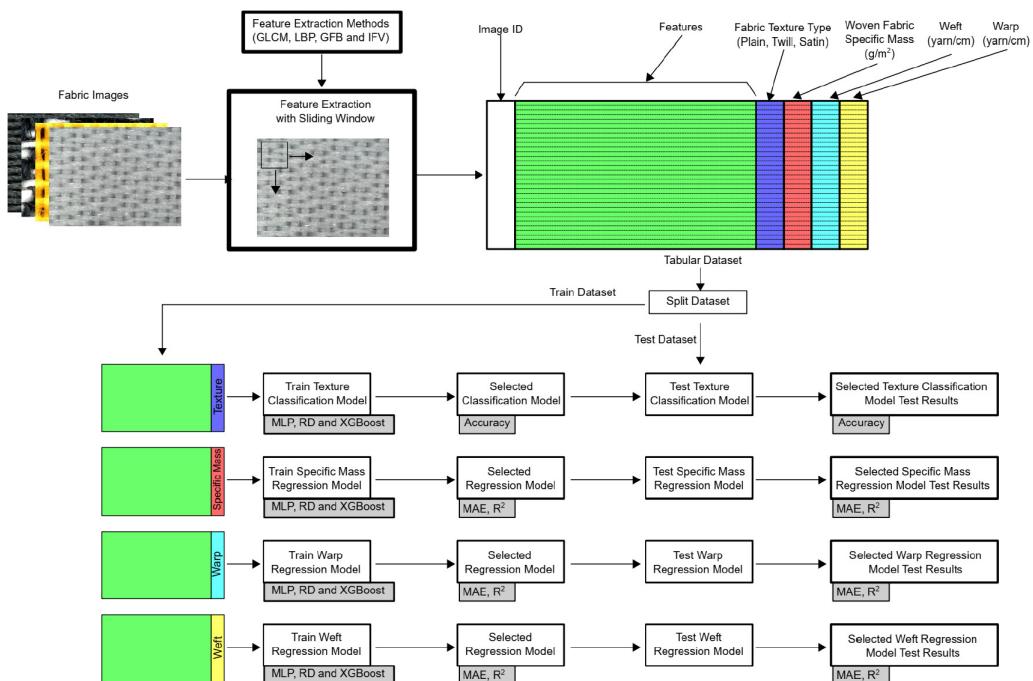


Figura 2.12: Propuesta de pipeline en la investigación de Seçkin et al. (2023).

Finalmente, el estudio de Weimer et al. (2024) propone una visión práctica y escalable para la automatización del proceso de clasificación textil. Mediante el diseño de un scanner de ropa compuesto por un sistema de cámaras de alta resolución montado sobre cintas transportadoras, lograron capturar imágenes de prendas en movimiento y clasificarlas según su tipo, calidad y tipo de tejido. Para validar su propuesta, los autores entrenaron modelos de *Deep Learning* basados en ResNet50 y ViT-L/16, alcanzando F1-Score superiores a 99 % en las tareas de clasificación. Esta investigación resulta especialmente relevante, ya que demuestra cómo soluciones

basadas en visión por computador pueden implementarse en entornos reales de clasificación textil, optimizando procesos, mejorando la trazabilidad de las prendas y fomentando modelos de reutilización más eficientes y sostenibles.

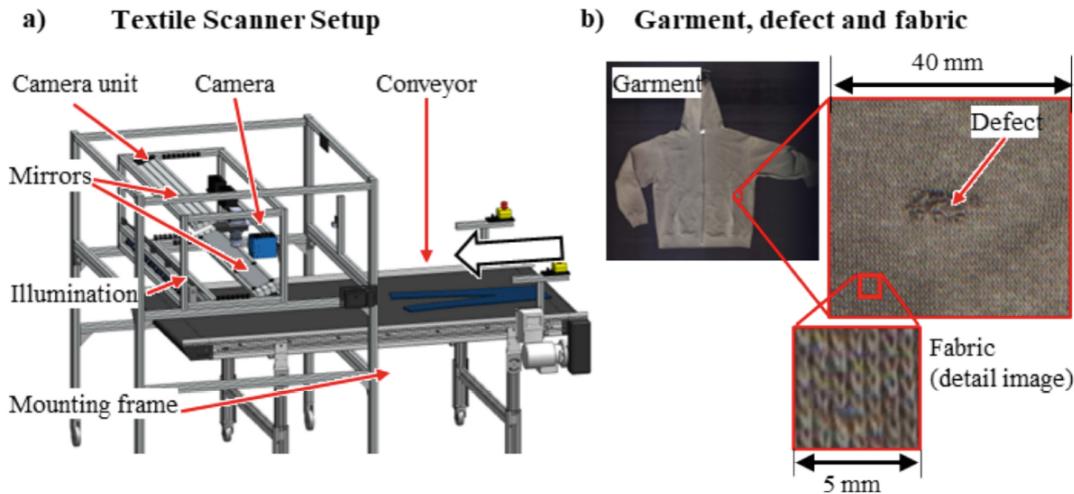


Figura 2.13: a) Imagen CAD del sistema de cámara para escaneo textil. b) Imagen de ejemplo de una prenda escaneada y un defecto en la tela. Weimer et al. (2024).

Capítulo 3

Dataset

En este capítulo se introduce el conjunto de datos utilizados en esta investigación. Se inicia explicando el contexto de su origen, seguido de una descripción general del *dataset* y, por último, una breve explicación del proceso realizado para la unificación y generación de la versión definitiva de los metadatos.

3.1 Contexto

El *Clothing Dataset for Second-Hand Fashion* es el primer conjunto de datos abiertos de ropa de segunda mano, provenientes de proyectos relacionados a la clasificación de ropa usada. Su objetivo principal es clasificar cada prenda en una de varias categorías según su destino final: reutilización, reutilización fuera de Suecia (exportación), reciclaje, reparación, rehacer o valorización energética (Nauman, 2024).

La recolección de los datos del dataset comenzó en la primavera de 2022, bajo el proyecto financiado por Vinnova “*AI for resource-efficient circular fashion*”, e involucra la colaboración de RISE Research Institutes of Sweden AB, Wargön Innovation AB y Myrorna AB, recibiendo además apoyo del proyecto europeo CISUTAC (Nauman, 2024). En septiembre del 2024 lanzaron la tercera versión -utilizada en esta investigación- publicándose bajo licencia CC BY 4.0, lo que permite su reutilización y distribución.

3.2 Descripción del dataset

El conjunto está formado por 31,638 prendas, identificadas mediante un ID en formato fecha-hora, y proviene de tres estaciones de etiquetado: station1, station2 (Wargön Innovation AB) y station3 (Myrorna AB). Todo el etiquetado fue realizado por clasificadores expertos de ambas organizaciones y, además, existe un subconjunto "gold" (test100) de 100 prendas anotadas por varios evaluadores para medir la concordancia de cada anotador (Nauman, 2024).

Cada artículo dispone de tres imágenes —vista frontal, vista posterior y primer plano de la etiqueta de marca— y un archivo JSON que contiene todas las anotaciones. Aunque faltan unas 4,000–5,000 imágenes de etiqueta por motivos de privacidad y, en algunos casos, por ausencia de marca, las fotografías restantes tienen una resolución, en su mayoría, de 1280×720 o 1920×1080 píxeles, y el fondo de captura evolucionó de una cinta métrica a una cuadrícula de 10×10 cm desde enero de 2023 (Nauman, 2024).

El archivo JSON de cada prenda incluye principalmente el campo *usage* (destino de la prenda: *Reuse*, *Export*, *Recycle*, *Repair*, *Remake*, *Energy recovery*), *price* (precio en moneda de suecia -SEK-), *trend* (estilo) y material (derivado de escáner NIR o etiqueta), junto a atributos de daño: *condition* y *pilling* en escala 1–5 (siguiendo estándares ISO 14644-1), *stains*, *holes* y *smell* con niveles *None/Minor/Major*, y para station1/2 la localización detallada de cada daño, entre otros campos.



Figura 3.1: Ejemplo de imágenes de una prenda y sus principales variables. Station: 1, Brand: SomeWear, Category: Unisex, Type: Sweater, Size: XL, Colors: White, Yellow Price: <50 SEK, Use: Export, Trend: None, Material: 79 % cotton, 21 % polyester, Holes: None, Stains: Minor, Smell: Minor, Pilling: 3, Condition: 3. Fuente: Elaboración propia.

3.3 Generación de los metadatos

Para facilitar el análisis de la información contenida en los archivos JSON y la carga dinámica de las imágenes en el desarrollo de la investigación, se diseñó en Python un pipeline modular de cuatro scripts que las unificara (Figura 3.2). El primer script

centralizó la configuración del proyecto (rutas base, directorios de datos y de salida) y creó automáticamente la estructura de directorios necesarias. El segundo recorrió los directorios de las tres estaciones de etiquetado y la del test100, agrupando las rutas relativas de las tres vistas de imágenes y del fichero JSON asociado a cada prenda, y generó un registro preliminar por ítem. El tercero leyó cada JSON, extrae sus campos de anotación y los incorporó al registro correspondiente. Finalmente, el cuarto aplanó las estructuras anidadas, transformando cada atributo en una columna independiente dentro de un único DataFrame.

La fase de normalización garantizó la homogeneidad de formatos y tipos de datos: las escalas ISO, para *condition* y *pilling*, se presentaron con valores numéricos, mientras que las categorías (*usage*, *stain*, *holes*, etc.) se mantuvieron como variables textuales. El resultado final fue un archivo .csv denominado *Metadata*, que incluyó todas las variables extraídas junto a las rutas relativas de cada imagen, generando un total de 44 variables y 32,036 registros (398 instancias adicionales que no estaban consideradas en la descripción original del *dataset*).

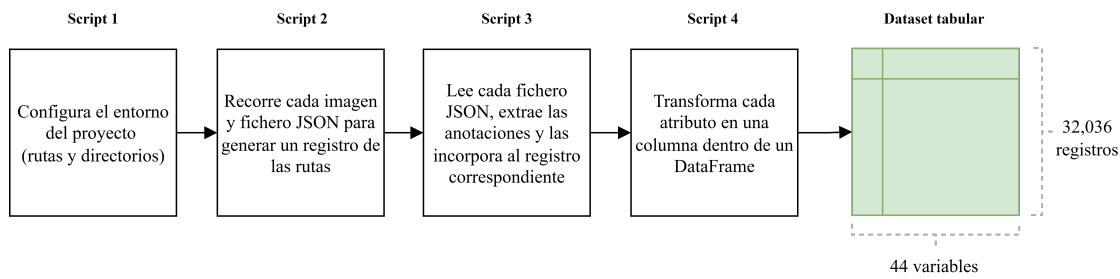


Figura 3.2: Pipeline desarrollado para la generación de los metadatos.

Capítulo 4

Metodología

En este capítulo se presenta el enfoque metodológico utilizado para desarrollar el sistema de clasificación de destino final de prendas de segunda mano. A lo largo de este se aborda de manera ordenada todos los pasos del proceso, iniciando con una explicación general del esquema de trabajo empleado, seguida de una explicación detallada de los experimentos y decisiones tomadas en cada fase y, por último, se presentan los resultados finales del ensamblado, entrenamiento y evaluación del *meta-learner*.

4.1 Visión general del pipeline

El esquema de trabajo seguido en esta investigación se estructuró en tres fases principales.

En la Fase 1 (Figura 4.1), se realizó el análisis exploratorio de los datos de la *metadata*: limpieza y preselección de variables, la división del *dataset* en dos subconjuntos -uno para entrenamiento/validación y otro para re-entrenamiento/prueba-, y el entrenamiento de un modelo base inicial para identificar las variables predictoras clave (*k-features*). Con ellas se re-entrenó y optimizó dicho modelo base, seleccionando la mejor configuración para su uso posterior en el *meta-learner*.

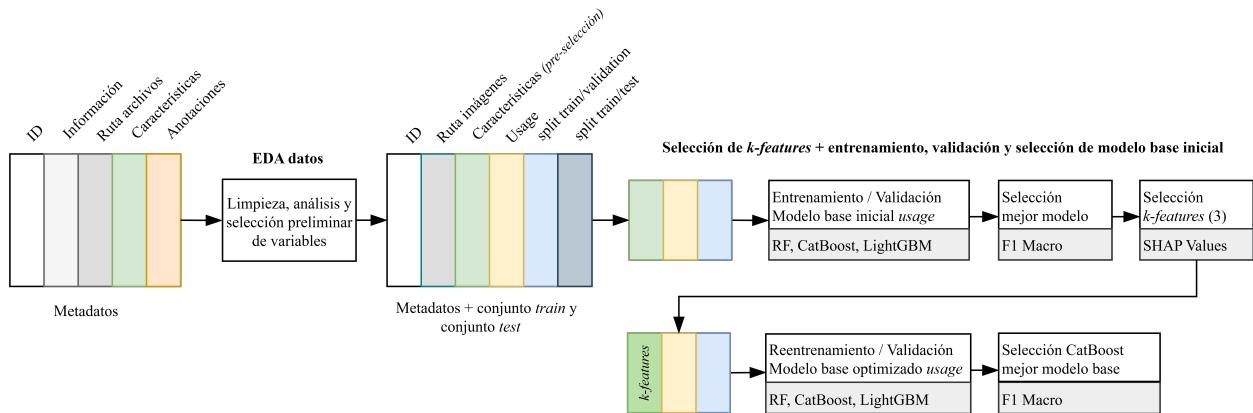


Figura 4.1: Pipeline fase 1: EDA datos, selección de *k-features* y selección de mejor modelo base inicial.

La Fase 2 (Figura 4.2) abordó el trabajo con las imágenes de las prendas: tras su análisis y procesamiento, se extrajeron descriptores visuales (LBP-U) que, junto a las *k-features* de la *metadata*, sirvieron de entrada al entrenamiento y validación de sub-modelos independientes para cada *k-feature*, eligiendo para cada variable el clasificador con mayor rendimiento predictivo a ser utilizado en la siguiente fase.

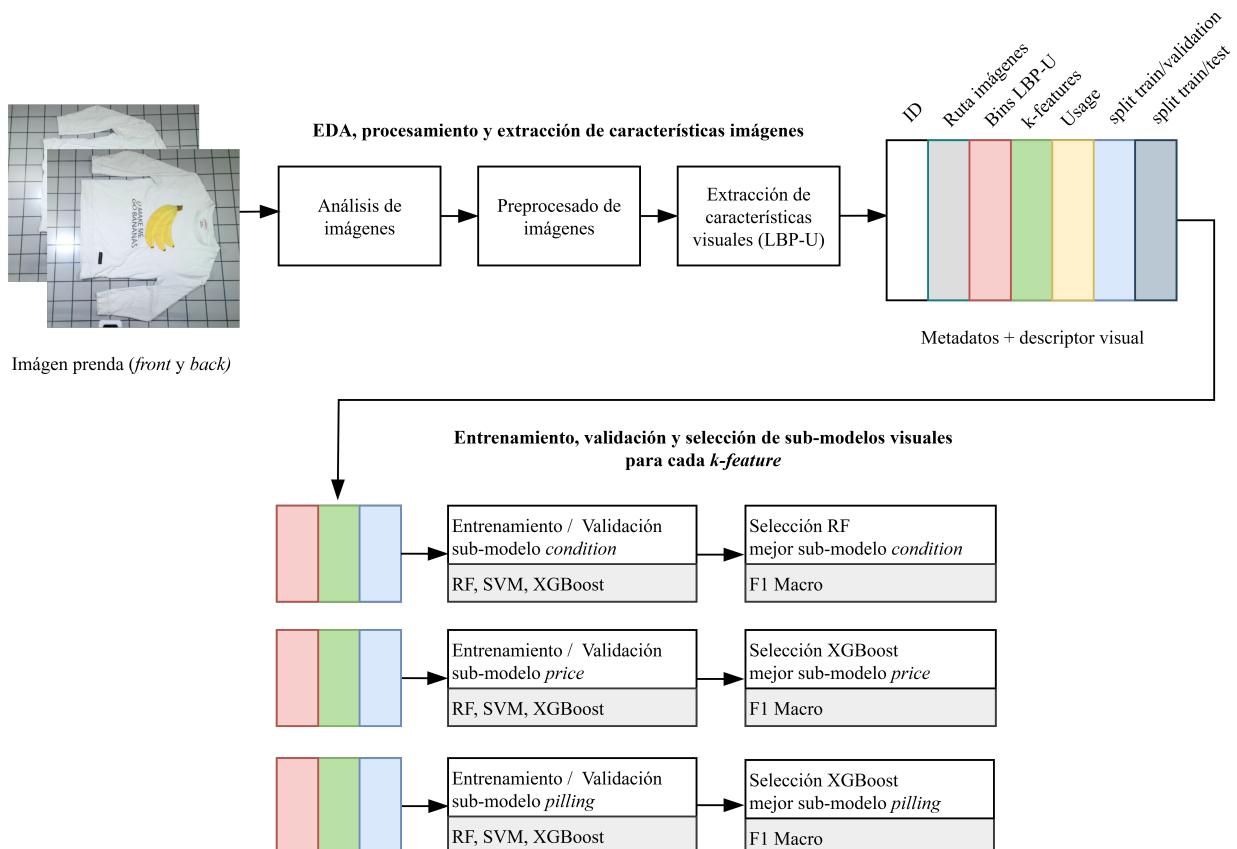


Figura 4.2: Pipeline fase 2: EDA imágenes, extracción de características visuales y selección de mejor sub-modelo visual para cada *k-feature*.

En la Fase 3 (Figura 4.3) se llevó a cabo el ensamblado y evaluación del meta-learner. Primero se generaron las *meta-features* —predicciones out-of-fold sobre todo el conjunto de *train* y predicciones puras sobre *test*— con los sub-modelos seleccionados; luego, con las *meta-features* de entrenamiento se entrenó el *meta-learner* y, en paralelo, se re-entrenó el modelo base sobre el conjunto completo de entrenamiento de la *metadata* original. Finalmente, se compararon cuatro escenarios para evaluar la capacidad predictiva del *meta-learner*: el primero estableció la línea base (modelo base entrenado y probado con *metadata* real); el segundo validó al *meta-learner* puro (entrenamiento y prueba con predicciones) y se comparó su capacidad predictiva con la línea base del escenario 1; y dos escenarios híbridos que examinaron, respectivamente, al modelo base entrenado con *metadata* real y probado con predicciones, y al *meta-learner* entrenado con predicciones y probado con *metadata* real, con el fin de cuantificar las ganancias y pérdidas al usar información real frente a predicha.

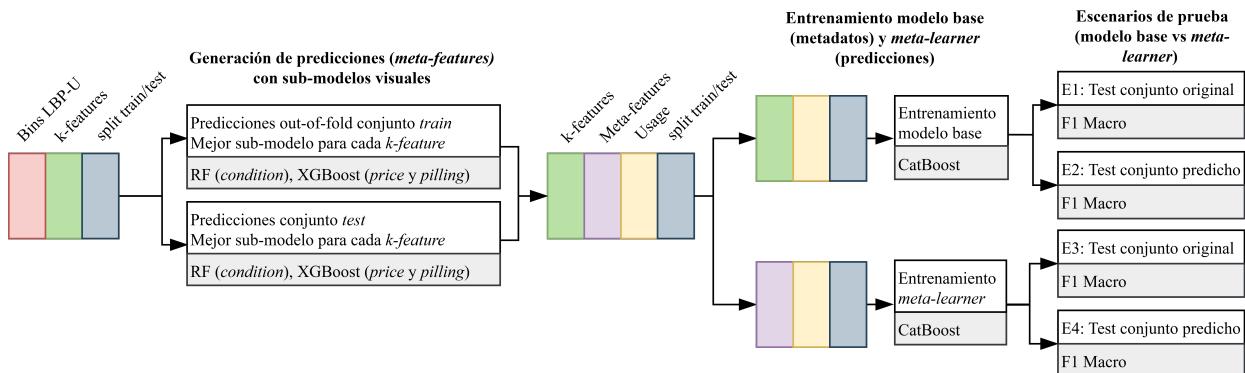


Figura 4.3: Fase 3: Generación de *meta-features*, ensamblado y *train/test* modelo base y *meta-learner*.

4.2 Fase 1: Análisis exploratorio de los datos y selección de *k-features*

4.2.1 Revisión general de los metadatos

Se inició la investigación realizando un análisis general del *dataset*, compuesto por un total de 32,036 registros y 44 variables (Figura 4.4). Del análisis de variables se identificaron 5 categorías principales (Figura 4.1): variables de identificación (*item_id*), de información de cuándo y dónde se tomaron las imágenes de la prenda (*month*, *station*, entre otras), de rutas de las imágenes y archivos (*front*, *back*, *brand* y *labels*), de características de las prendas (p.ej. *material*, *category*, *type*, *size*, *price*,

condition, pilling, entre otras) y de anotaciones de texto realizado por los anotadores (p.ej. *comments, brand.text, sizetext, description*, entre otras).

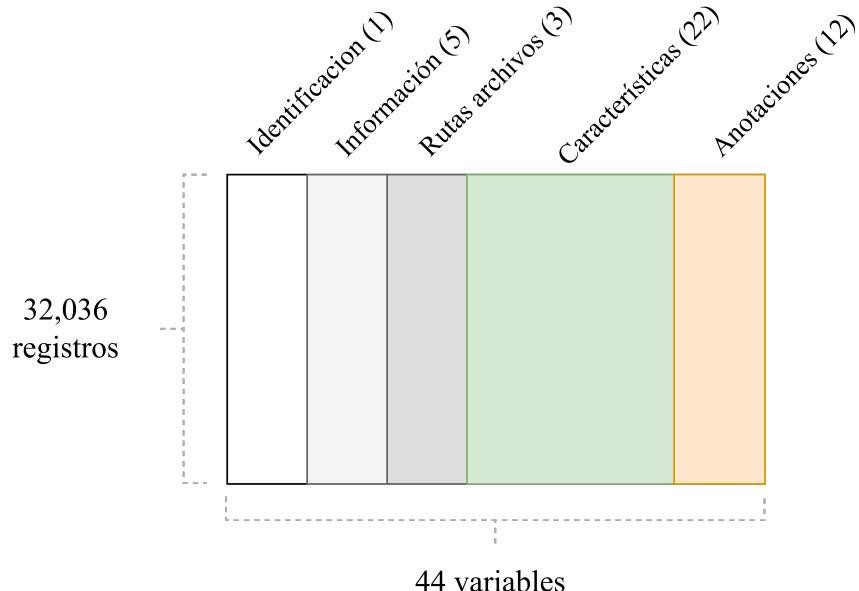


Figura 4.4: Estructura tabular metadatos.

Tabla 4.1: Variables por categorías de datos.

Categoría	variables originales metadatos
Identificación	item_id
Información	station, month, gold, annotator, brand_missing
Rutas archivos	front, back, brand, labels
Características	brand.1, brand.text, category, type, size, colors, season, pilling, condition, price, cut, pattern, trends, smell, stains, holes, damage, materials, usage, sizetext, styleholes, weight
Anotaciones	comments, usage_confidence, price_confidence, description, damageimage, damageloc, damage2image, damage2loc, damage3image, damage3loc, damage2, damage3

En esta revisión inicial, se realizó una verificación del tipo de dato de cada variable y la existencia de valores nulos. Se identificaron 100 registros faltantes en *front* y *back* (0.31 %), 5,290 en *brand* (20 %), 21,068 en *brand.text* (66 %), 6,094 en *size* (19 %), 6,415 en *season* (20 %) y entre 20,000 a 30,000 en variables como *smell*, *stains* y *holes*, entre otros. Este análisis inicial permitió priorizar la limpieza y tratamiento de las variables antes de avanzar en la transformación y selección de características.

4.2.2 Limpieza, transformación y distribución de variables (*feature engineer*)

Para cada variable se realizó un análisis univariado que incluyó la inspección del tipo de dato, análisis de frecuencias y distribuciones, y verificación de valores únicos y ausencias. A partir de este diagnóstico, se aplicaron transformaciones genéricas según la naturaleza de los atributos (ver Apéndice A.1). Se imputaron los valores faltantes rellenando con “*Unknown*” o un *placeholder* cuando fuera necesario. Las variables numéricas y ordinales (*pling, condition, price*) se estandarizaron. Los campos textuales (*brand, brand.text, color, pattern, cut, trends, etc.*) se normalizaron con técnicas de Procesamiento Natural del Lenguaje (p.ej. limpieza de espacios, corrección tipográfica y normalización de mayúsculas) y se consolidaron en variables limpias. Las rutas de imagen (*front, back, brand*) se llenaron con un *placeholder missing_image* cuando no existía imagen y se renombraron a **_path* (p.ej. *front_path*). Identificadores y metadatos irrelevantes (*item_id, annotator, comment, damage, weight, etc.*) se combinaron o eliminaron según su utilidad.

Posterior a la limpieza, transformación y unificación de las variables, se realizó un análisis de distribución y frecuencia para cada una de las categorías de las 16 variables descriptivas principales de una prenda (Figura 4.5, 4.6, 4.7). A modo de resumen, en *category* -categoría de la prenda según público objetivo-, la línea *Ladies* representó el 68 % de los registros, mientras que *Children* y *Men* se situaron en torno al 14 % y *Unisex* al 4 %. En *type* -tipo de prenda-, destacan especialmente *Top* (4,284 prendas), *T-shirt* (3,522) y *Trousers* (4,219), mientras que categorías específicas como *Robe*, *Denim jacket* y *Winter trousers* mostraban muy baja representación. En *colors* -colores de la prenda-, los colores neutros dominan el catálogo, con *Black* y *White* con cerca de 10,000 apariciones cada uno, seguidos de *Blue* (8,300) y *Grey* (3,900). En *season* -estación del año asociada a la prenda-, el 67 % de las prendas no se vincula a una estación (*All*) y un 20 % carece de etiqueta. Los indicadores de desgaste —*pling* y *condition*— se inclinaron hacia valores altos: el 51 % de los artículos presentó *pling* nivel 5 (mayor presencia de pelotitas en la prenda) y el 80 % *condition* entre 3 y 5 (niveles de peor condición). El rango de *price* -precio estimado de la prenda- mostró que el 89 % de las prendas cuesta menos de 100 SEK (9 euros app). En *cut* -tipo de ajuste de la prenda-, más de la mitad (53 %) fue *Regular*, seguida de *C-collar* (14 %) y 8 % sin identificación. En *pattern* -estampado de la prenda- se concentraron en “*Unknown*” (50 %) y “*Other*” (13 %), con *Floral*, *Striped* y *Logo Print* como los más frecuentes. La variable *trends* -tendencia general de la prenda-, quedó mayoritariamente sin tendencia clara “*Unknown*”(85 %), con un 7 % de “*No trend*” y con pequeñas apariciones de *Denim* y *Sports*. Las características de *Smell*,

stains, y *holes* -presencia de olor, mancha y orificio en la prenda- registraron más del 90 % de valores desconocidos. En *size* -tamaño de la prenda-, las tallas convencionales M (13 %), L (11 %) y S (10 %) fueron mayoría, aunque un 19 % quedó sin asignar (Unknown). En *brand* -marca de la prenda-, el 20 % de los registros se concentran en *H&M*, *Lindex* y *Kappahl*, un 10 % a ropa sin registro, quedando una larga cola de casi 5,085 marcas con baja representación. Por ultimo *usage* -destino final de uso la prenda- el 99 % de las prendas se clasifican como *Reuse* (60 %), *Export* (32 %) y *Recycle* (7 %).



Figura 4.5: Distribución de *category*, *type*, *colors*, *season*, *pilling* y *condition*.

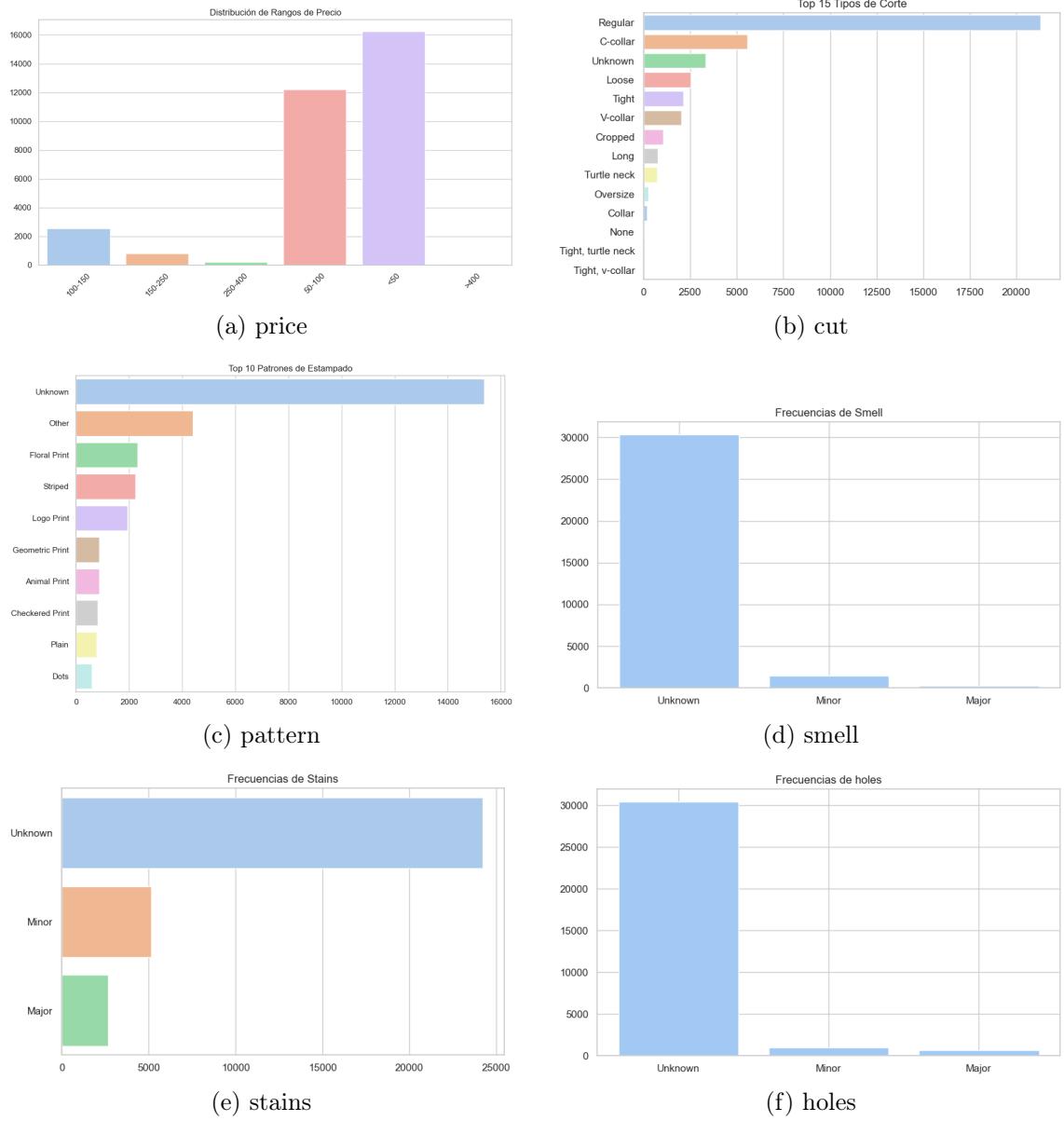


Figura 4.6: Distribución de *price*, *cut*, *pattern*, *smell*, *stains* y *holes*.

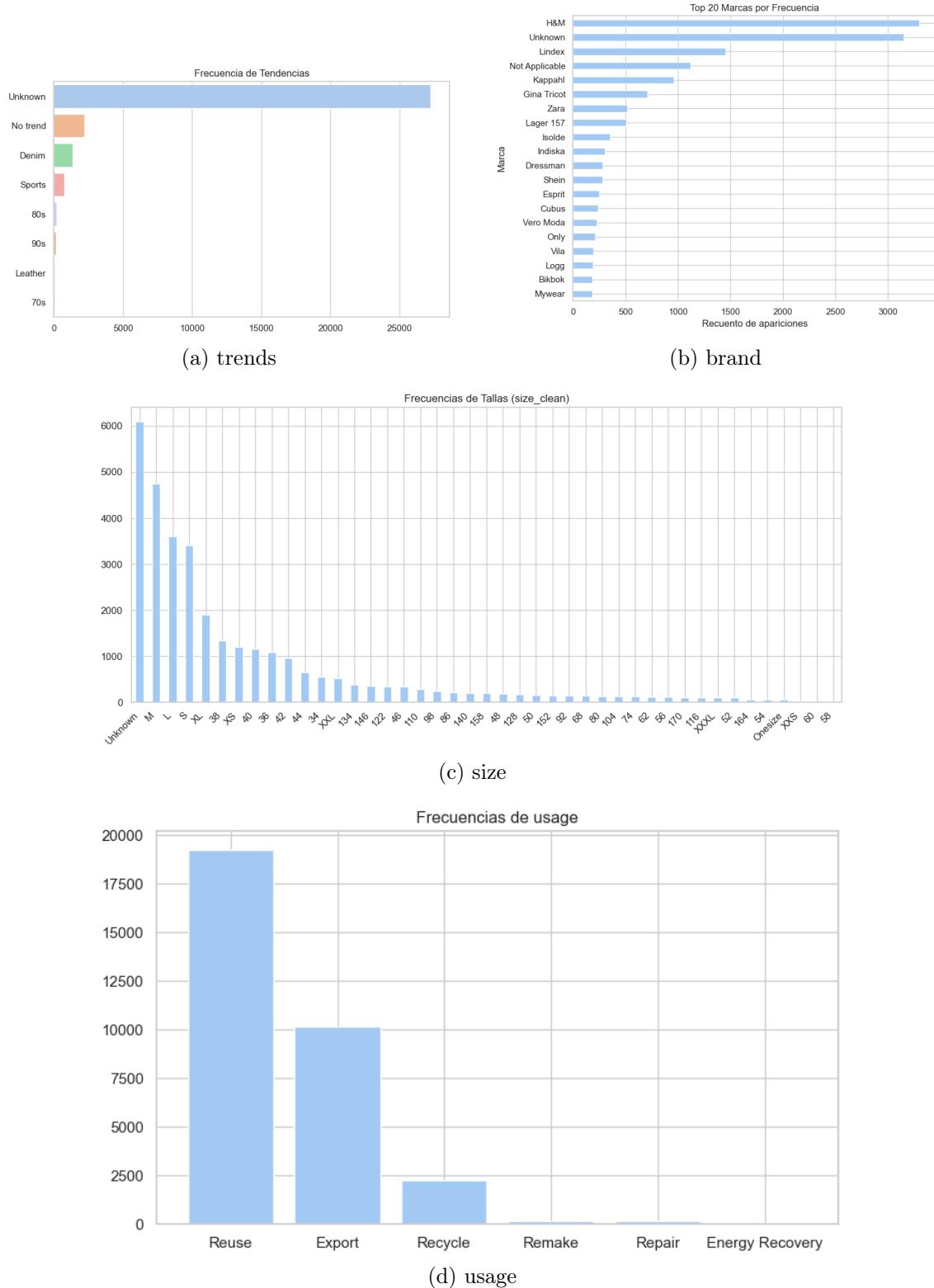


Figura 4.7: Distribución de *trends*, *brand*, *size* y *usage*.

4.2.3 Selección preliminar de variables explicativas (*feature selection*)

Para seleccionar aquellas variables con mayor capacidad de predicción ante la variable objetivo *usage*, se realizó un análisis de relevancia. Para ello, primero se realizó una revisión general previa de la variable objetivo *usage* (en la siguiente sección se presenta y analiza en detalle). En segundo lugar, se descartaron las columnas irrelevantes para el modelado -rutas de imagen, identificadores y variables auxiliares- y se confeccionó un subconjunto de 14 atributos principales —todas variables categóricas— sobre las que se evaluó su asociación con *usage*. Si bien en el paso anterior se observaron algunos registros sin imágenes, en este análisis se consideraron las 32,036 instancias, dado que todas contaban con información valiosa para la evaluación. Sobre este conjunto de variables se aplicaron cuatro pruebas de análisis: Test estadístico Chi-cuadrado (χ^2) normalizado con coeficiente Cramér's V, análisis de Información Mutua, análisis de ANOVA (*one way*) y el test paramétrico Kruskal-Wallis.

Del análisis se obtuvieron los resultados mostrados en la Tabla 4.1, donde mediante la selección de umbrales -Cramér V ≥ 0.15 (efecto moderado), Mutual Information ≥ 0.05 (ganancia práctica), F-statistic $\gg 1$ (diferencias significativas de medias) y H-statistic $\gg 1$ (diferencias significativas de rangos)- se identificaron de forma consistente cinco variables con alta capacidad predictiva: *condition*, *price*, *stains*, *pilling* y *holes*. Estas características conformaron el conjunto preliminar de *k-features* a utilizar en la fase de entrenamiento de modelo base con metadatos.

4.2.4 Análisis de variable objetivo

Una vez identificadas las variables explicativas relevantes, se procedió a analizar en detalle la variable objetivo *usage* y su representación en las distintas imágenes.

Como paso previo, se realizó una limpieza para asegurar que cada registro de *usage* contara con su imagen frontal y posterior, condición fundamental para la extracción de características y entrenamiento de sub-modelos. Esta revisión detectó que las variables de ruta (*front*, *back* y *brand*) contemplaban un total de 32,036 registros, de los cuales en *front* y *back* 31,936 contenían rutas correctas a las imágenes y 100 no (0,31 %), mientras que para *brand* 26,746 registros contaban con su imagen y 5,290 no (16,51 %). Se procedió a eliminar los 100 registros sin imágenes para *front* y *back*, quedando un *dataset* preliminar de 31,936 instancias. Se eliminó la variable de ruta de imágenes de *brand* por su falta de registros, no utilizándose en esta investigación. Posteriormente, se revisó la distribución de imágenes por estación, observándose

Tabla 4.2: Resultados obtenidos de las pruebas y análisis de relevancia.

Variable	Cramér V	Mutual Information	F-stat (ANOVA)	H-stat (Kruskal)
condition	0.369	0.312	5730.18	16038.64
price	0.337	0.260	3252.85	13701.88
stains	0.346	0.125	3514.72	7128.38
pilling	0.175	0.060	605.78	3007.06
holes	0.155	0.024	356.18	1392.56
colors	0.294	0.101	1.09	3357.41
cut	0.151	0.047	8.59	2095.29
type	0.125	0.031	44.91	1522.44
brand_group	0.119	0.023	5.69	790.00
pattern	0.097	0.018	43.81	786.74
season	0.086	0.017	36.32	604.63
size	0.077	0.012	11.30	555.30
trend	0.076	0.011	36.28	400.94
category	0.061	0.008	100.39	340.16

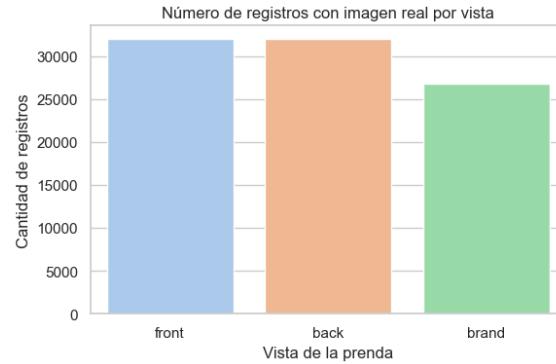
Nota: Todas las variables presentaron un p-value <0.05 tanto para Chi-cuadrado como para ANOVA. Aunque *cut* y *colors* mostraron relevancia en el análisis global, al descomponerlos en dummies –por ser multietiqueta– no presentaron métricas robustas y quedaron fuera de esta fase de selección.

que *station1* aportaba 21,839 registros, *station2* 7,416 y *station3* 2,681. Finalmente, respecto del conjunto de control *gold_garment*, se identificaron 298 registros con imágenes consideradas como parte del *goldset*. Debido a su baja representatividad se decide no utilizar este set como *test* set. Estas distribuciones se resumen en la Figura 4.8.

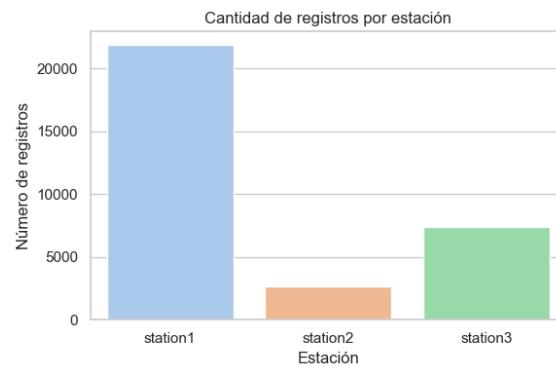
Con la verificación de imágenes por registro realizada, se procedió a analizar *usage*. Esta variable se compone de 6 categorías: *Reuse* -prendas que se pueden reusar- con 19,159 registros (60,0 %), *Export* -prendas que se pueden exportar, en este caso fuera de Suecia- con 10,145 registros (31,8 %), *Recycle* -prendas que se pueden reciclar- con 2,242 registros (7,0 %), *Remake* -prendas que se deben rehacer- con solo 156 instancias (0,5 %), *Repair* -prendas que se deben reparar- con 146 registros (0,4 %) y *Energy Recovery* -prendas que se deben incinerar- con solo 88 registros (0,3 %). En Apéndice A.2 se muestra un ejemplo de imagen (frontal, posterior y de marca) para cada categoría. Esta distribución original evidenció un fuerte desbalanceo, donde *Reuse* y *Export* concentran más del 92 % de las observaciones, mientras que las 3 últimas aportan menos del 1 %. La Figura 4.9 muestra la distribución original de las seis categorías y la versión final reducida a tres clases.

Debido a su baja representatividad, las clases *Repair*, *Remake* y *Energy Recovery* fueron descartadas del análisis, al no disponer de un volumen suficiente para entrenar clasificadores consistentes. En consecuencia, la variable objetivo se redefinió en tres

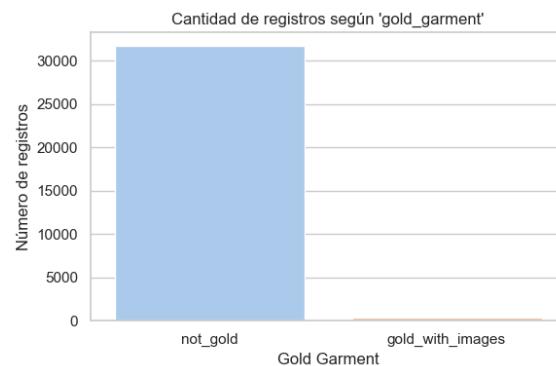
clases principales —*Reuse*, *Export* y *Recycle*— que concentran más del 99 % de los registros. Para sus uso en modelos de *Machine Learning*, estas etiquetas fueron codificadas a enteros: *Reuse* = 0, *Export* = 1 y *Recycle* = 2.



(a) Registro por vista

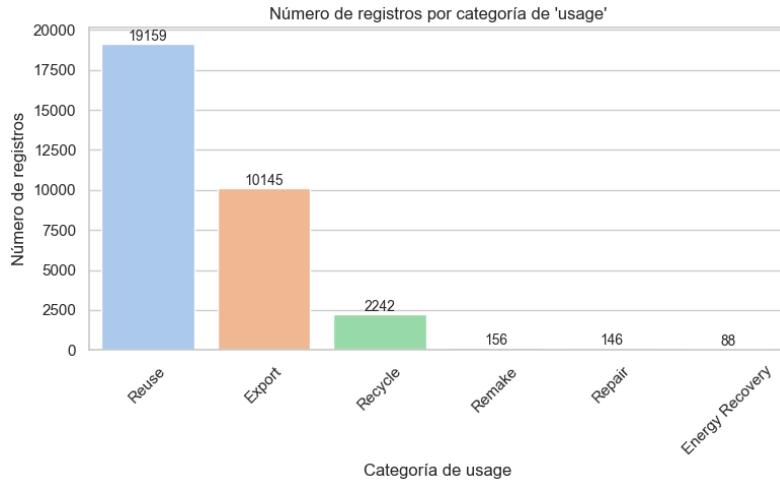


(b) Registro por *station*

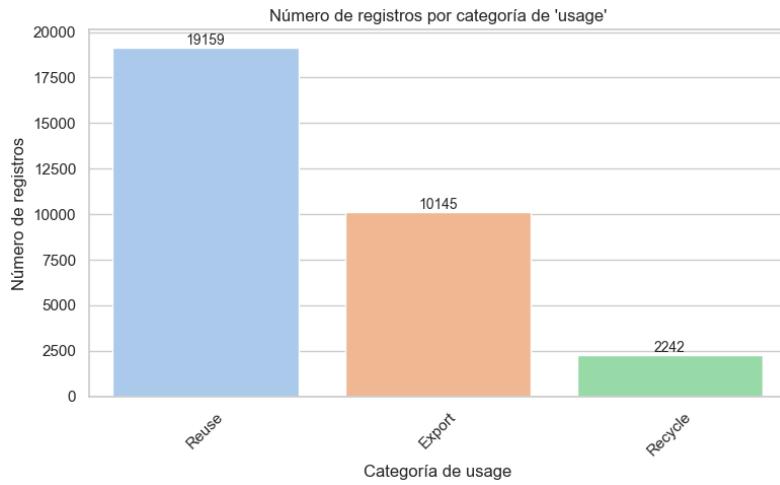


(c) Registro por etiqueta *gold*

Figura 4.8: Distribuciones de imágenes por vista *station* y *gold*.



(a) Registros iniciales



(b) Registros posterior a procesamiento

Figura 4.9: Registros por categoría en *usage*.

4.2.5 División del conjunto de datos

Con la *metadata* revisada, la existencia de imágenes verificadas -para *front* y *back*- y la variable *usage* reducida a las clases finales, se procedió a realizar una división estratificada del *dataset*. Se optó por una estrategia de dos conjuntos: un primer conjunto de *train* y *validation* para disponer con una muestra representativa para entrenar, ajustar y validar los modelos y un segundo conjunto de *train* y *test*, para re-entrenar los modelos -con sus mejores configuraciones y con todos los datos- y evaluar en un conjunto aislado no visto en ninguna fase.

El procedimiento se ejecutó en dos etapas (Tabla 4.3). Primero se realizó un *split* estratificado sobre el conjunto de datos de 80% *train* y 20% *test*. Posteriormente, dentro del conjunto *train* creado, se aplicó un segundo *split* estratificado de 75% para *train* y un 25% para *validation*. Para utilizarlos según la etapa del proceso de

modelamiento se crearon dos columnas auxiliares: *split_test* con el primer conjunto de datos *train* (*train_full*) y *test*, y *split_train* con el segundo conjuntos de datos *train* (*train_train*) y *validation* (*train_validation*).

Tabla 4.3: División de los registros en los conjuntos de entrenamiento, validación y prueba.

Clase usage	Proporción global	(1) Split_test		(2) Split_train	
		<i>train_full</i>	<i>test</i>	<i>train_train</i>	<i>train_validation</i>
0: Reuse	61 %	15,327	3,832	11,495	3,832
1: Export	32 %	8,116	2,029	6,087	2,029
2: Recycle	7 %	1,793	449	1345	448
Total		25,236	6,310	18,927	6,309
Porcentaje		80 %	20 %	75 %	25 %

Nota: *split_test* corresponde a la división original 80/20 de *train* y *test*, mientras que *split_train* corresponde a la subdivisión del conjunto *train* para fines de entrenamiento y validación (75/25).

Como resultado de este proceso, se consolidó una *metadata* final de 31,546 registros limpios y depurados, con rutas de imágenes verificadas, variables explicativas y objetivo trabajadas y un *dataset* correctamente dividido, listo para ser utilizado en las fases de entrenamiento y validación de modelos base y sub-modelos, así como en la etapa final de entrenamiento del *meta-learner* y prueba.

4.2.6 Entrenamiento inicial de modelo base y selección final de variables (*k-features*)

4.2.6.1 Selección de algoritmos, métricas de evaluación y cálculos de eficiencia

Para seleccionar las variables explicativas finales, que se utilizaron tanto en el entrenamiento del modelo base con *metadata* como para los sub-modelos desarrollados mediante técnicas de *Computer Vision* y *Machine Learning*, se implementó una estrategia comparativa basada en tres algoritmos de clasificación robustos para el manejo de variables categóricas: Random Forest, CatBoost y LightGBM. Estos modelos fueron entrenados sobre el subconjunto del *dataset* de *train* y *validation* (*split 2*), utilizando únicamente las variables obtenidas tras el proceso de selección preliminar *-condition*, *price*, *stains*, *pilling* y *holes*- y evaluados en función de su capacidad predictiva sobre la variable objetivo *usage*.

Como métrica principal de evaluación se empleó F1 Macro, dado que permite evaluar

el desempeño del modelo en contextos de desbalance de clases -como ocurre con la categoría *Recycle* en *usage*-, otorgando el mismo peso a cada clase independiente de su frecuencia. Como métrica complementaria se consideró Recall Macro, la que entrega información relevante respecto de la capacidad del modelo de recuperar la mayor cantidad de casos positivos en cada categoría. Ambas métricas fueron utilizadas de forma sistemática a lo largo de todo el proceso de desarrollo de modelos de esta investigación.

De modo complementario a las métricas de evaluación, se incorporó un análisis de eficiencia computacional de cada algoritmo, que incluyó la medición del tiempo de entrenamiento en segundos, el uso de memoria (medido en uso máximo de memoria en CPU, dado que todos los modelos fueron ejecutados sin GPU) y la estimación de emisiones de CO₂ -medido con la librería CodeCarbon, el cual se configuró para que estimara la huella de carbono emitida en base al consumo y fuente eléctrica para Barcelona, España-, tanto para la fase de entrenamiento como de inferencia.

4.2.6.2 Búsqueda de hiperparámetros y entrenamiento inicial de modelos

Para cada uno de los algoritmos evaluados se realizó una búsqueda de hiperparámetros, empleando distintas estrategias de configuración y búsqueda, según correspondiera. El proceso consideró técnicas de balanceo para mitigar el desequilibrio de clases en *usage* y la selección final se basó en la métrica F1 Macro obtenida mediante validación cruzada estratificada.

La búsqueda de hiperparámetros se estructuró de forma específica para cada algoritmo (Tabla 4.4). En el caso de Random Forest se exploraron tres enfoques de balanceo —class weights, Random Over Sampler y SMOTE— y se variaron el número de árboles, la profundidad máxima y los tamaños mínimos de división y hoja. Para CatBoost se definió una grilla de learning rate, profundidad y regularización L2, evaluando iteraciones crecientes para equilibrar convergencia y sobreajuste. Por su parte, LightGBM incorporó en su búsqueda tasas de aprendizaje, número de hojas, profundidad máxima y penalizaciones L1/L2, con el objetivo de optimizar tanto precisión como velocidad de entrenamiento.

Tabla 4.4: Configuración de búsqueda y selección de hiperparámetros para modelo base inicial con *metadata*.

Algoritmo	Balanceo	Parámetros	Método	Selección
Random Forest (CW)	Class weights	n_estimators: [100, 200, 300], max_depth: [5, 10, 15, None], min_samples_split: [2, 5, 10], min_samples_leaf: [1, 2, 4], class_weight:[None, balanced, balanced_subsample]	Random Search, iterations: 30, CV: 5, Scoring: F1_macro	n_estimators: 100, max_depth: 5, min_samples_split: 2, min_samples_leaf: 4, class_weight: balanced
Random Forest (SMOTE)	SMOTE	n_estimators: [100, 200, 300], max_depth: [5, 10, 15, None], min_samples_split: [2, 5, 10], min_samples_leaf: [1, 2, 4], class_weight: [None]	Random Search, iterations: 30, CV: 5, Scoring: F1_macro	n_estimators: 200, max_depth: None, min_samples_split: 5, min_samples_leaf: 2, class_weight: None
Random Forest (ROS)	Random Over Sampler	n_estimators: [100, 200, 300], max_depth: [5, 10, 15, None], min_samples_split: [2, 5, 10], min_samples_leaf: [1, 2, 4], class_weight: [None]	Random Search, iterations: 30, CV: 5, Scoring: F1_macro	n_estimators: 100, max_depth: 10, min_samples_split: 10, min_samples_leaf: 2, class_weight: None
CatBoost	Class weights	learning_rate: [0.03, 0.05, 0.1, 0.2], depth: [4, 6, 8, 10], l2_leaf_reg: [1, 3, 5, 7], iterations: [100, 200, 300]	Grid Search, CV: 5, Scoring: F1_macro	class_weights: (0: 1.6), (1: 3.1), (2: 14.1), depth: 4, iterations: 200, l2_leaf_reg: 7, learning_rate: 0.03
LightGBM	Class weights	learning_rate: [0.03, 0.05, 0.1, 0.2], num_leaves: [15, 31, 63], depth: [4, 6, 8, 10], n_estimators: [100, 200, 300], reg_alpha: [0, 1, 3], reg_lambda: [0, 1, 3],	Grid Search, CV: 5, Scoring: F1_macro	class_weights: (0: 1.6), (1: 3.1), (2: 14.1), learning_rate: 0.05, max_depth: 4, n_estimators: 100, num_leaves: 15, reg_alpha: 3, reg_lambda: 1

Nota: Con Random Forest se probaron las distintas técnicas de balanceo, en base a su resultado se seleccionó la mejor para utilizar en los otros algoritmos.

4.2.6.3 Resultados del entrenamiento inicial de modelo base

Tras comparar el desempeño de todos los modelos entrenados, LightGBM destacó como el mejor modelo en esta primera experimentación, alcanzando un F1 Macro de 65.55 % -ligeramente superior a Random Forest (CW) y CatBoost- y un Recall Macro de 69.50 %, apenas superado por Random Forest ROS y SMOTE (Tabla 4.5). En relación a su matriz de confusión (Figura 4.10) y reporte de clasificación (Tabla 4.6), este modelo base inicial presentó un excelente rendimiento para clasificar *Reuse* y *Export* -Recall de 0.88 y 0.68 respectivamente- y tendió a confundirse con Recycle (Recall 0.53). Estos resultados justificaron su selección para la etapa de interpretación con SHAP y de definición de *k-features*.

Tabla 4.5: Métricas de evaluación modelo base inicial.

Algoritmo	F1-Score Macro	Recall Macro
LightGBM	65.55 %	69.50 %
Random Forest (CW)	65.32 %	68.77 %
CatBoost	65.25 %	69.24 %
Random Forest (ROS)	64.92 %	69.59 %
Random Forest (SMOTE)	64.92 %	69.59 %

Tabla 4.6: Reporte de clasificación LightGBM.

Clase	Precision	Recall	F1-score	Support
Reuse	0.96	0.88	0.91	3832
Export	0.73	0.68	0.70	2029
Recycle	0.26	0.53	0.35	448
accuracy			0.79	6309
macro avg	0.65	0.70	0.66	6309
weighted avg	0.83	0.79	0.81	6309

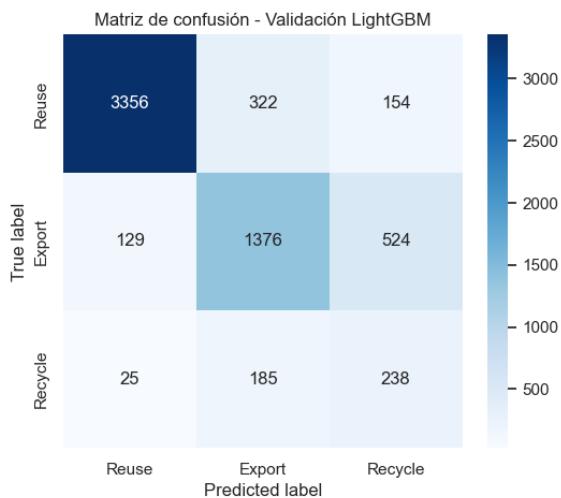


Figura 4.10: Matriz de confusión LightGBM.

Adicionalmente, para cada modelo entrenado se midió su tiempo, coste computacional y emisión de CO₂ en su fase de entrenamiento como en inferencia (Tabla 4.7). En conjunto, el proceso de entrenamiento y validación del modelo base inicial generó 0.13 kg de CO₂.

Tabla 4.7: Métricas de eficiencia computacional de algoritmos para modelo base inicial.

Algoritmo	Tiempo E (s)	Tiempo I (s)	Mem CPU E (MB)	Mem CPU I (MB)	CO ₂ E (kg)	CO ₂ I (kg)	CO ₂ total (kg)
LightGBM	3199.41	0.04	102.79	0.11	0.059504	0.059506	0.119010
Random Forest (CW)	18.19	0.06	3.18	0.07	0.000325	0.000327	0.000652
CatBoost	228.27	0.04	14.45	0.25	0.005057	0.005059	0.011577
Random Forest (ROS)	21.32	0.09	1.07	0.00	0.000750	0.001126	0.001876
Random Forest (SMOTE)	23.79	0.18	9.77	0.00	0.000747	0.000750	0.001497
							Total 0.134612

Nota: E = Entrenamiento, I = Inferencia, Mem CPU = Memoria en CPU.

Este primer experimento supuso un punto de inflexión para la investigación, al demostrar la viabilidad de predecir *usage* a partir de atributos cualitativos claves y

que si estos, se logran reproducir a partir de imágenes y descriptores visuales, un *meta-learner* podría ser factible para esta tarea. Asimismo, confirmó la robustez de los algoritmos seleccionados para datos categóricos y la utilidad de incorporar Recall Macro como métrica complementaria. Las técnicas de balanceo y búsqueda de hiperparámetros mostraron su eficacia al incrementar el F1 Macro y mejorar la predicción de las clases minoritarias, mientras que el análisis de tiempo de entrenamiento, uso de memoria y emisiones de CO₂ garantizó la alineación con los principios de *Green AI*.

4.2.6.4 Selección final de *k-features* mediante SHAP Values

Se utilizó la librería SHAP para analizar la importancia de cada variable sobre las predicciones del modelo LightGBM entrenado con metadatos. A partir del diagrama de valores medios absolutos de SHAP (Figura 4.11), se observó que las variables con mayor impacto en la salida del modelo fueron *condition*, *price* y *pilling*, en ese orden. Aunque SHAP también identificó la categoría *stains_Unknown* entre los predictores, ésta se descartó por tratarse de una variable capciosa: no es posible determinar si “*Unknown*” implica ausencia o presencia de manchas, a la vez que la distribución de la variable original *stains* está fuertemente desbalanceada (ver Figura 4.6). En consecuencia, la selección final de *k-features* quedó fijada en *condition*, *price* y *pilling*.

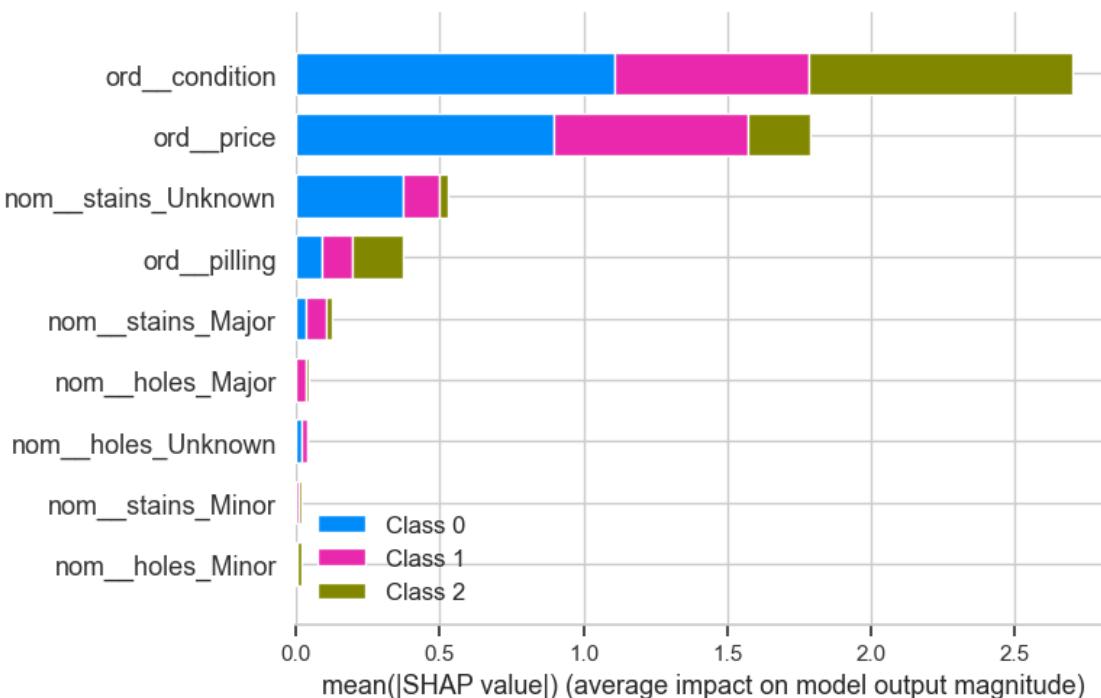


Figura 4.11: Importancia de variables con SHAP Values.

4.2.7 Optimización y re-entrenamiento de modelo base con *k-features*

Tras seleccionar las tres *k-features* (*condition*, *price* y *pilling*), se reentrenó el modelo base para evaluar su capacidad predictiva usando únicamente estas variables. A su vez, se definieron dos esquemas de categorización: el primero mantenía la granularidad original de las variables, mientras que el segundo reducía las clases a solo tres niveles (Tabla 4.8). Este experimento partió de la hipótesis de que un esquema de tres clases que estuviera más balanceado podría facilitar el entrenamiento de modelos de visión por computadora de la siguiente fase, al reducir la complejidad de la tarea multicategoría. De este modo, se buscó evaluar si una menor granularidad mejoraba o empeoraba la capacidad predictiva del modelo y tomar una decisión fundamentada sobre cual esquema seguir.

Tabla 4.8: Categorías originales y reducidas de las *k-features* seleccionadas.

k-feature	Categoría original	Categoría reducida
condition	1; 1,020 (3 %) 2; 4,964 (16 %) 3; 9,580 (30 %) 4; 7,755 (25 %) 5; 8,227 (26 %)	1 = bajo (1–2); 5,984 (19 %) 3 = medio (3–4); 17,335 (55 %) 5 = alto (5); 8,227 (26 %)
price	<50; 15,865 (50 %) 50–100; 12,099 (38 %) 100–150; 2,534 (8 %) >150; 1,048 (3 %)	<50; 15,865 (50 %) 50–100; 12,099 (38 %) >100; 3,582 (11 %)
pilling	1; 669 (2 %) 2; 2,211 (7 %) 3; 4,895 (16 %) 4; 7,553 (24 %) 5; 16,218 (51 %)	1 = bajo (1–2); 2,880 (9 %) 3 = medio (3–4); 12,448 (39 %) 5 = alto (5); 16,218 (51 %)

Nota: Para cada categoría se presentan su etiqueta, número de registros y porcentaje (p.ej. 1; 1,020 (3 %)). *Price* está contabilizado en SEK (moneda Sueca) equivalente a 0.09 euros (al 26/08/2025).

Para la selección de hiperparámetros en el entrenamiento de los modelos, se utilizó la misma grilla de búsqueda anterior (ver sección 4.2.6.2). En la Tabla 4.9 se presentan los mejores hiperparámetros encontrados para cada algoritmo según esquema de categorización.

Tabla 4.9: Selección de mejores hiperparámetros para modelo base entrenado con *k*-*features* seleccionadas.

Algoritmo	Hiperparámetros k-features original	Hiperparámetros k-features reducida
Random Forest (CW)	n_estimators: 100, max_depth: 5, min_samples_split: 2, min_samples_leaf: 4, class_weight: balanced	n_estimators: 300, max_depth: 10, min_samples_split: 2, min_samples_leaf: 1, class_weight: balanced
Random Forest (SMOTE)	n_estimators: 300, max_depth: 10, min_samples_split: 2, min_samples_leaf: 2, class_weight: None	n_estimators: 200, max_depth: 15, min_samples_split: 5, min_samples_leaf: 1, class_weight: None
Random Forest (ROS)	n_estimators: 100, max_depth: 10, min_samples_split: 5, min_samples_leaf: 1, class_weight: None	n_estimators: 100, max_depth: 15, min_samples_split: 10, min_samples_leaf: 2, class_weight: None
CatBoost	class_weights: (0: 1.6), (1: 3.1), (2: 14.1), depth: 6, iterations: 200, l2_leaf_reg: 7, learning_rate: 0.03	class_weights: (0: 1.6), (1: 3.1), (2: 14.1), depth: 4, iterations: 100, l2_leaf_reg: 1, learning_rate: 0.03
LightGBM	class_weights: (0: 1.6), (1: 3.1), (2: 14.1), learning_rate: 0.05, max_depth: 4, n_estimators: 100, num_leaves: 31, reg_alpha: 0, reg_lambda: 2	class_weights: (0: 1.6), (1: 3.1), (2: 14.1), learning_rate: 0.05, max_depth: 4, n_estimators: 100, num_leaves: 15, reg_alpha: 3, reg_lambda: 1

4.2.8 Resultados entrenamiento y selección final de modelo base

Los resultados de la Tabla 4.10 muestran que, en el esquema original de categorías, Random Forest (CW) obtuvo el mejor desempeño con un F1 Macro de 66,98 % y un Recall Macro de 67,56 %, mientras que en el esquema reducido CatBoost alcanzó el máximo rendimiento con un F1 Macro de 65,89 % y un Recall Macro de 67,10 %, presentando un rendimiento levemente menor en comparación a su contraparte. Los reportes de clasificación (Tablas 4.11 y 4.12) y las matrices de confusión (Figura 4.12) evidencian que ambos modelos mantienen una sólida discriminación entre *Reuse* (Recall de 0.91 y 0.82 respectivamente) y *Export* (Recall de 0.72 y 0.80), aunque presentan mayor confusión en la clase minoritaria *Recycle* -ambos con un Recall de 0.40- en comparación con el modelo base inicial (LightGBM). Asimismo, ambos candidatos mostraron eficiencia computacional competitiva en entrenamiento (15.25 s para RF y 149 s para CatBoost) y emisiones de CO₂ comparables a las de los demás algoritmos, 0.000538 kg para RF y 0.006349 kg para CatBoost (Tabla 4.13). En conjunto, el proceso de entrenamiento y validación de todos los modelos generó aproximadamente 0.17 kg de CO₂.

Si bien la reducción de categorías penalizó levemente el F1 Macro y el Recall Macro respecto al esquema original, el rendimiento de CatBoost entrenado sobre las *k*-*features* con categorías reducidas se mantuvo robusto y competitivo frente a su

versión con granularidad completa. Como resultado, CatBoost, con sus mejores hiperparámetros, fue seleccionado como el mejor modelo para ser utilizado en la fase final de ensamblado y entrenamiento del *meta-learner*. Asimismo, se decidió emplear de forma definitiva el esquema simplificado de las *k-features* en el entrenamiento posterior de los sub-modelos, garantizando un balance más homogéneo y reduciendo la complejidad de la clasificación multicategoría.

Tabla 4.10: Métricas de evaluación de algoritmos para modelo base con *k-features* seleccionadas.

Algoritmo	k-features originales		k-features reducidas	
	F1 Macro	Recall Macro	F1 Macro	Recall Macro
Random Forest (CW)	66.98 %	67.56 %	65.55 %	67.63 %
CatBoost	66.94 %	68.17 %	65.89 %	67.10 %
LightGBM	66.74 %	68.07 %	65.55 %	67.63 %
Random Forest (ROS)	65.23 %	68.19 %	65.61 %	67.61 %
Random Forest (SMOTE)	65.23 %	68.19 %	65.61 %	67.61 %

Tabla 4.11: Reporte de clasificación Random Forest (CW) para *k-features* originales.

Clase	Precision	Recall	F1-score	Support
Reuse	0.92	0.91	0.91	3832
Export	0.73	0.72	0.73	2029
Recycle	0.35	0.40	0.37	448
accuracy			0.81	6309
macro avg	0.67	0.68	0.67	6309
weighted avg	0.82	0.81	0.81	6309

Tabla 4.12: Reporte de clasificación CatBoost para *k-features* reducidas.

Clase	Precision	Recall	F1-score	Support
Reuse	0.95	0.82	0.88	3832
Export	0.64	0.80	0.71	2029
Recycle	0.38	0.40	0.39	448
accuracy			0.78	6309
macro avg	0.66	0.67	0.66	6309
weighted avg	0.81	0.78	0.79	6309

Figura 4.12: Matriz de confusión para Random Forest (CW) y CatBoost.

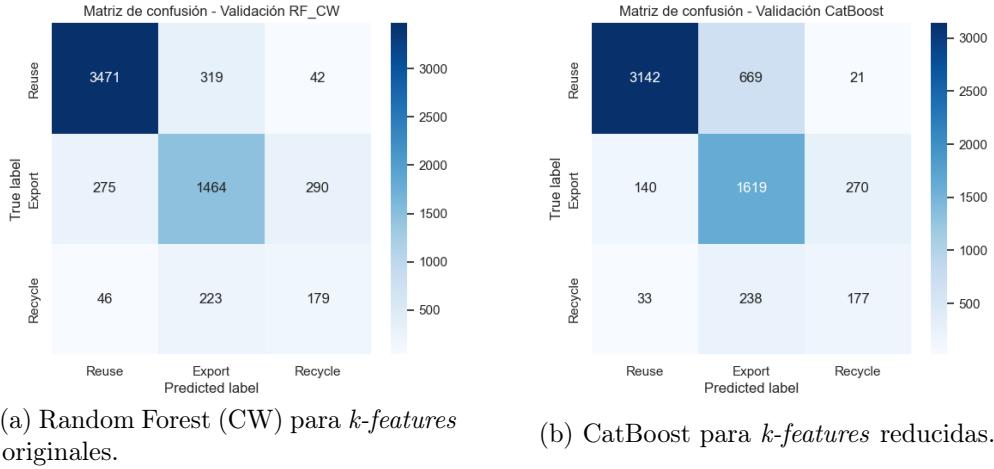


Tabla 4.13: Métricas de eficiencia computacional modelo base con *k*-features seleccionadas.

Algoritmo	<i>k</i> features	Tiempo E (s)	Tiempo I (s)	Mem CPU E (MB)	Mem CPU I (MB)	CO ₂ E (kg)	CO ₂ I (kg)	CO ₂ total (kg)
RF (CW)	original	15.25	0.06	4.48	0.04	0.000268	0.000270	0.000538
	reducida	13.43	0.10	4.57	0.10	0.000238	0.000241	0.000479
CatBoost	original	180.47	0.04	16.25	0.25	0.003896	0.003898	0.007794
	reducida	148.81	0.03	12.65	0.25	0.003174	0.003175	0.006349
LightGBM	original	2473.35	0.04	217.71	0.44	0.045979	0.045981	0.091960
	reducida	1467.24	0.04	8.14	0.06	0.028173	0.028175	0.056348
RF (ROS)	original	13.82	0.09	0.46	0.00	0.000554	0.000803	0.001357
	reducida	10.08	0.05	2.16	0.00	0.000438	0.000616	0.001054
RF (SMOTE)	original	15.60	0.15	6.29	0.00	0.000551	0.000554	0.001105
	reducida	10.72	0.07	4.06	0.00	0.000436	0.000438	0.000874
								Total 0.167858

Nota: E = Entrenamiento, I = Inferencia, Mem CPU = Memoria en CPU, RF = Random Forest.

4.3 Fase 2: Análisis exploratorio de las imágenes y entrenamiento de sub-modelos visuales

4.3.1 Análisis exploratorio y preprocesamiento de las imágenes

Se realizó un análisis exploratorio general de las imágenes *front* y *back* con el objetivo de caracterizar en detalle sus propiedades visuales y poder extraer *insights* relevantes para el preprocesamiento de las imágenes de cara a la extracción de características.

Del análisis de integridad de las imágenes se verificó la existencia de 63,902 imágenes

en formato .jpg, y no se encontraron archivos corruptos, ni ausentes. El 95.1 % presentaba una resolución de 1280x720 píxeles y solo un 4.9 % en 1920x1080 píxeles (Figura 4.13). Ambas vistas presentaron un *aspect ratio* constante de 1.78. Respecto al fondo, el 92.1 % de las imágenes presentaban un fondo cuadriculado y el 7.9 % con cinta de medición (Figura 4.14).

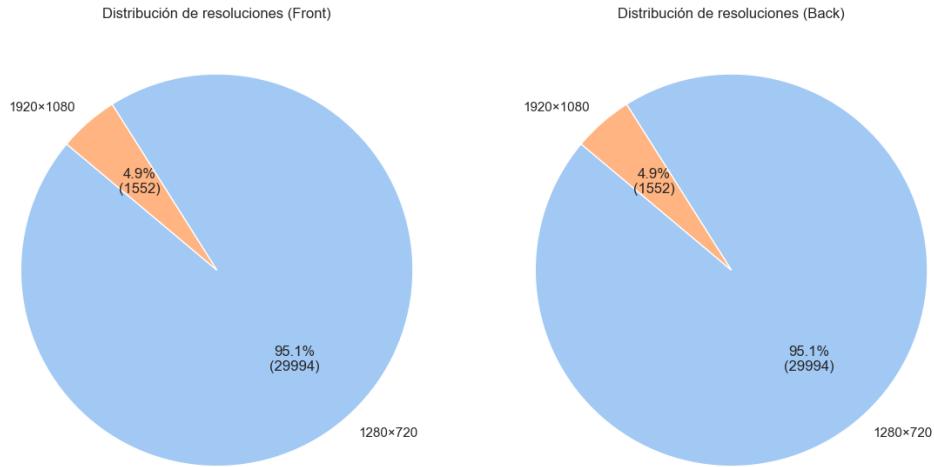


Figura 4.13: Distribución de resolución por vista de imagen.

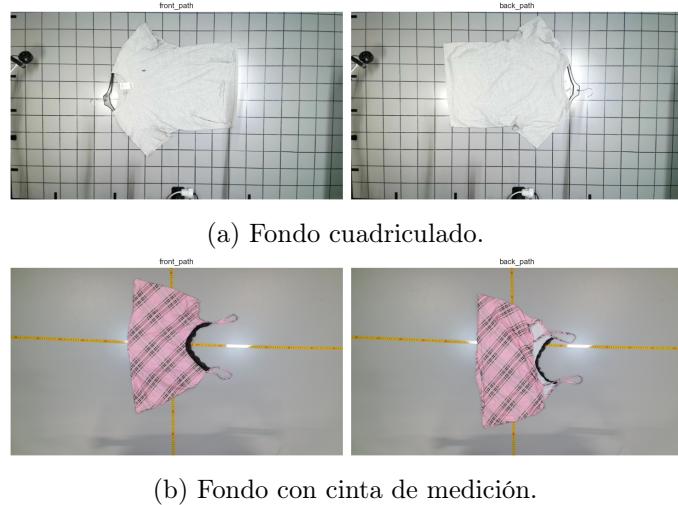


Figura 4.14: Ejemplos de imágenes por tipo de fondo.

Se analizó el brillo medio de cada imagen y se compararon sus distribuciones globales por tipo de imagen, fondo y estación de captura (Figura 4.15). En el análisis por tipo de imagen, los valores de brillo medio oscilaron entre 160-185, con pocos casos por debajo de 100 o encima de 190. Por tipo de fondo, los valores medio se mantuvieron similares al anterior. Por estación, *station1* registró una mayor proporción de imágenes con brillo bajo (inferior a 120) y alto (superior a 180) que *station2* y *station3*, lo que

sugiere diferencias en la iluminación, configuración del fondo o cámara. Al hacer una revisión de los *outliers*, se confirmó que estos no eran fallos de configuración, sino prendas reales con colores o tamaños extremos (Figura 4.16).

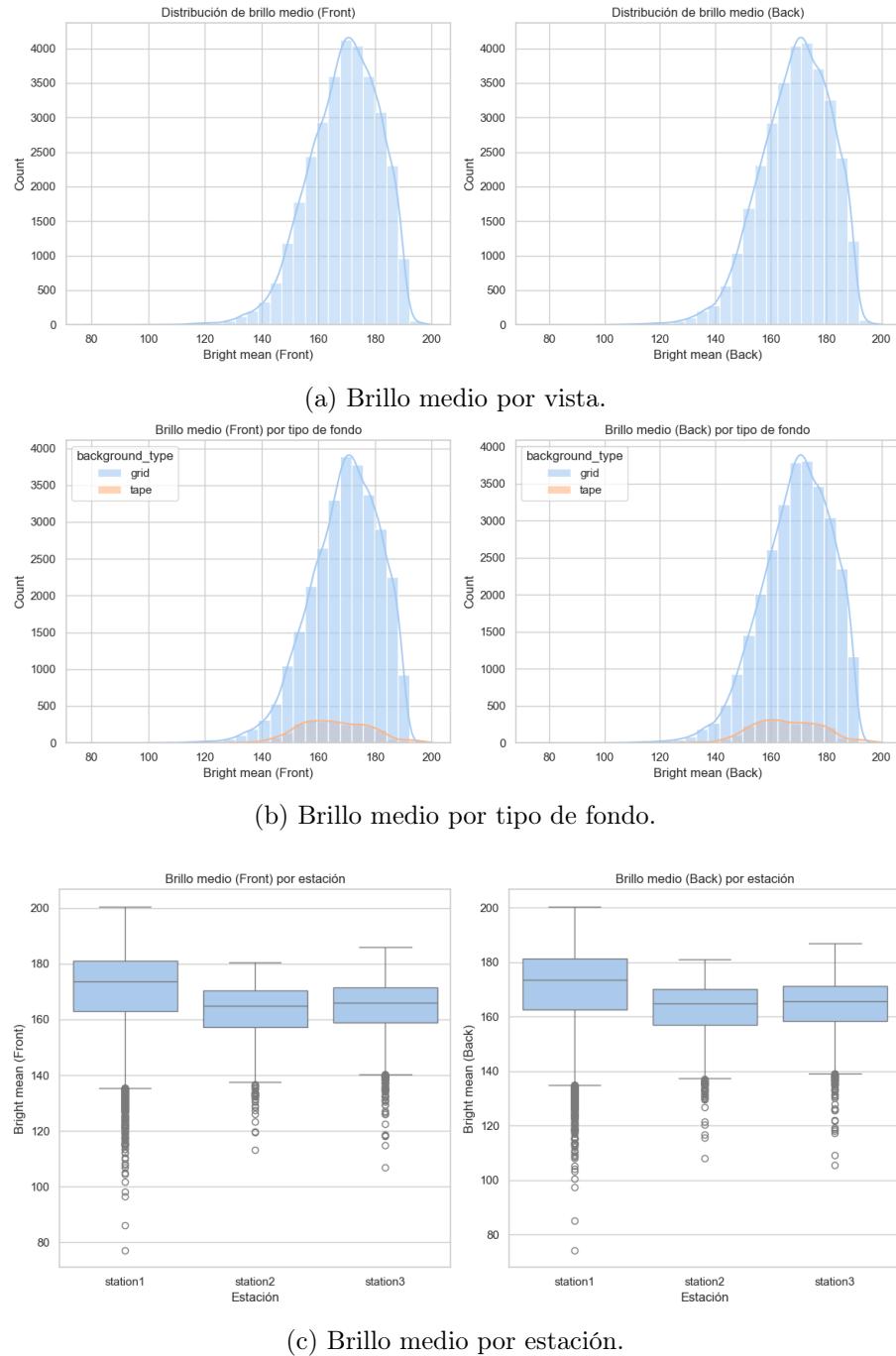
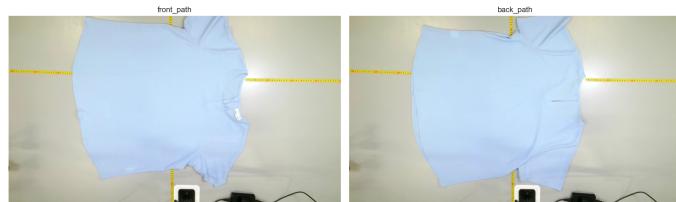


Figura 4.15: Análisis de brillo medio.



(a) *Outlier* oscuro (brillo medio de 86).



(b) *Outlier* oscuro (brillo medio de 200).

Figura 4.16: Ejemplos de *outliers* en análisis de brillo medio.

Se calculó la media y la desviación estándar de los canales R, G y B para las vistas *front* y *back* (Figura 4.17). Las distribuciones mostraron medianas muy similares (R: 167, G: 170, B: 172) y rangos intercuartílicos prácticamente idénticos, sin sesgos de cámara entre *front* y *back* (Figura 4.18). El canal B presentó la mayor variabilidad (desviación estándar 14.6 frente a 2.9 en G), reflejando prendas con tonos azules más intensos. Del análisis de *outliers* se identificó que correspondían a prendas brillantes/saturadas (R: 230), confirmando variabilidad legítima de colores (no errores de captura).

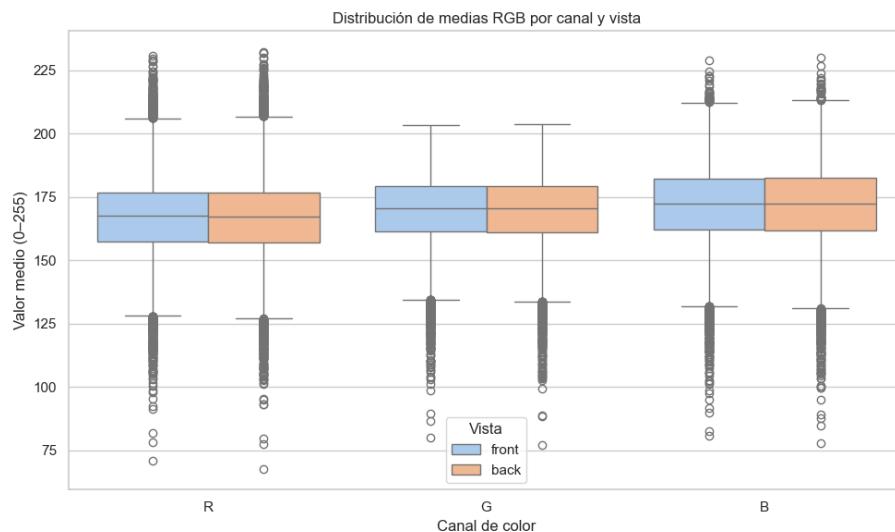


Figura 4.17: Distribución de medias RGB por canal y vista

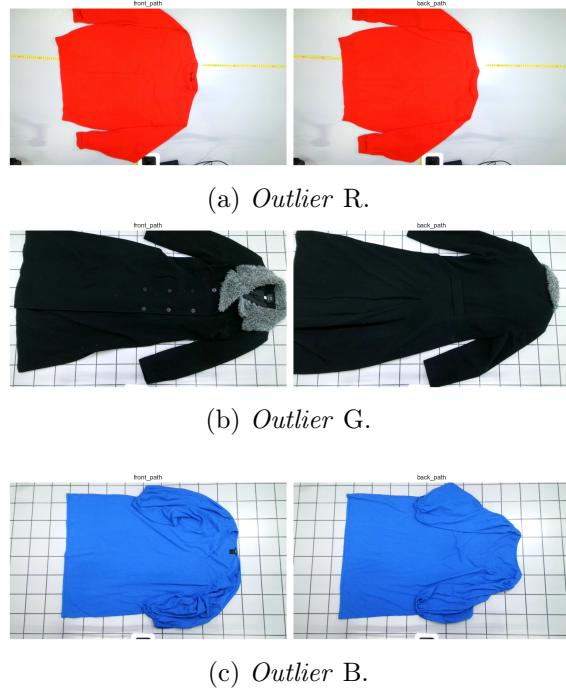
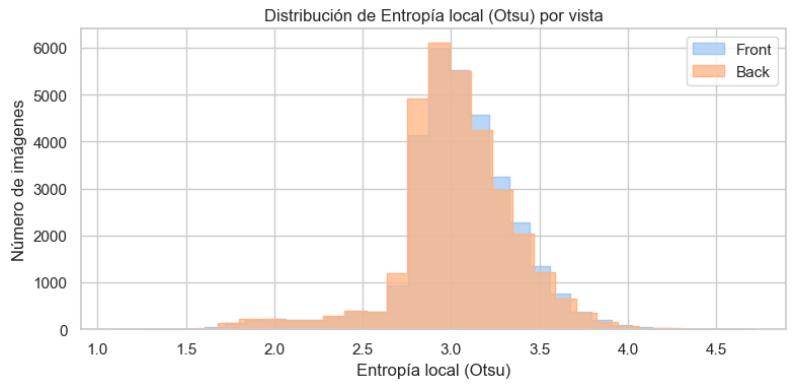


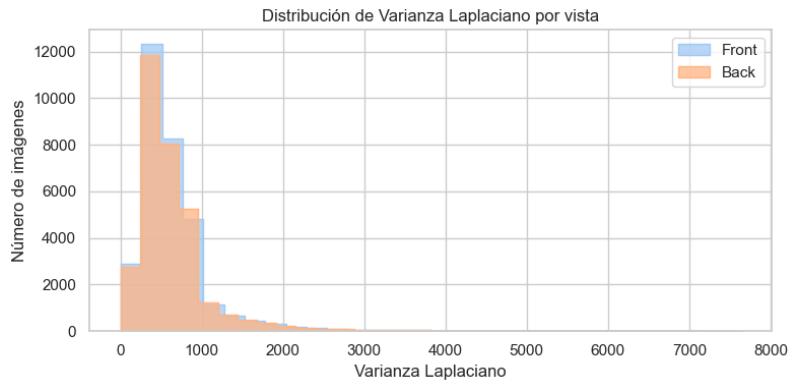
Figura 4.18: Ejemplos de *outliers* en análisis de color RGB.

Se midió la complejidad de textura de las prendas mediante dos métricas: entropía local (umbralización Otsu) y varianza del Laplacian (indicador de nitidez). La entropía media fue de 3.04 bits en *front* y *back* (IQR [2.89–3.23]), con pocos *outliers* bajos (<2.37 ; 3.7 %) o altos (>3.74 ; 1.6 %) que correspondieron a tejidos muy lisos o estampados complejos (Figura 4.20). La varianza del Laplacian presentó medias de 632 (*front*) y 622 (*back*), IQR [360–775], sin *outliers* inferiores y un 5–6 % de valores superiores a 1,398, asociados a zonas de alto contraste (bordados, arrugas marcadas). Las distribuciones *front/back* son casi idénticas (Figura 4.19), lo que confirma uniformidad de textura.

Del análisis se concluyó que las imágenes podían estandarizarse sin perder información relevante, mediante ajustes simples previos a la extracción de descriptores. En primer lugar, se redimensionaron todas las imágenes a 1280×720 píxeles, manteniendo alta definición de la prenda y asegurando uniformidad. En cuanto al brillo y al color, dado que los casos extremos correspondían a prendas reales y no a fallos de captura, no se aplicó ninguna corrección ni filtrado adicional. Respecto a los fondos, pese a identificarse dos tipos distintos (cuadriculado y con cinta), se decidió conservarlos y posponer cualquier técnica de enmascarado o segmentación para trabajos futuros. Por último, se identificó la necesidad de aplicar un filtro gaussiano para suavizar el ruido y estabilizar las métricas de entropía y de Laplacian, cuya parametrización se explora en la siguiente sección.



(a) Distribución de Entropía local (Otsu) por vista.

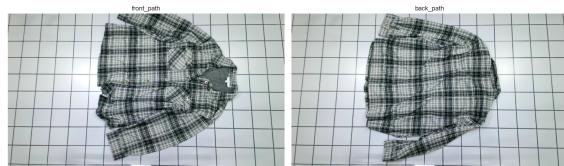


(b) Distribución de varianza Laplacian por vista.

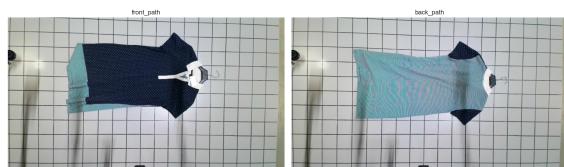
Figura 4.19: Distribuciones Otsu y Laplacian.



(a) *Outlier* Otsu bajo.



(b) *Outlier* Otsu alto.



(c) *Outlier* Laplacian alto.

Figura 4.20: Ejemplos de *outliers* para Otsu y Laplacian.

4.3.2 Extracción de características visuales

Tras el análisis y procesamiento de las imágenes, se extrajeron características con los descriptores Local Binary Pattern Uniform (LBP-U) e Histogram of Oriented Gradients (HOG). Para cada uno se exploró una grilla de parámetros y se seleccionó la mejor configuración, que luego se evaluó —tanto para LBP-U solo como combinado con HOG— sobre un subconjunto de 5,000 prendas, usando F1 Macro en *condition*, *price* y *pilling* para medir su capacidad predictiva. Finalizada esta validación, se aplicó la configuración óptima al conjunto completo, se midió su emisión de CO₂ y se concatenaron los histogramas resultantes en un *dataset* para el entrenamiento de los sub-modelos del *meta-learner*.

Se inició el proceso con LBP-U, un descriptor de textura invarianta a la rotación, ligero de calcular y de histogramas compactos. Para encontrar su configuración óptima, se definió una grilla de parámetros (Tabla 4.14) para P (número de vecinos) y R (radio) y varias configuraciones de filtro gaussiano (Tabla 4.15), que se evaluaron sobre una muestra de prendas de *condition* (Figura 4.21). La combinación seleccionada fue P = 24 y R = 3, que preservó la dispersión bimodal y capturó microtexturas, junto con un filtro gaussiano de blur = 1 y tamaño de kernel 5×5, que eliminó ruido fino sin borrar detalles.

Tabla 4.14: Grilla de parámetros y resultados para LBP-U.

Configuración	Nº Bins (P+2)	Argumento de prueba	Resultado
P=16, R=2	18	Resolución intermedia: 16 vecinos a distancia 2. Balance entre detalle y robustez, útil para texturas de desgaste moderado.	Distribución razonable, pero menos detalle intermedio. Se descartó.
P=24, R=3	26	Alta resolución: 24 vecinos a distancia 3. Captura micro-texturas finas.	Excelente dispersión bimodal y captación de microtexturas. Se seleccionó como configuración óptima.
P=32, R=4	34	Resolución ultra-fina: 32 vecinos a distancia 4. Máxima granularidad para detectar las transiciones más pequeñas, a costa de mayor ruido y tiempo de cómputo	Muchos bins vacíos, señal concentrada en pocos picos. Se descartó.

Una vez definida la configuración óptima de LBP-U y de filtro gaussiano, se evaluó su capacidad predictiva sobre una muestra de 5,000 prendas con las categorías originales de *condition*, *price* y *pilling*. Se extrajeron los histogramas LBP-U (52 bins: 24+2 por vista), se dividieron los datos en 80/20 y se entrenó un Random Forest sencillo, con *class_weight*=balanced. Los F1 Macro obtenidos fueron 28 % en *condition*, 35 % en *price* y 22 % en *pilling*, confirmando su utilidad predictiva incluso sin optimización de hiperparámetros en los modelos.

Posteriormente, se evaluó el aporte de HOG extrayendo el descriptor con configuración estándar (9 orientaciones, celdas de 16×16 píxeles y bloques de 2×2) sobre imágenes

Tabla 4.15: Grilla de parámetros y resultados para filtro gaussiano.

Parámetros	Argumento de prueba	Resultados
Escala de grises	Configuración base	Máximo detalle, pero mucho “ruido” de fondo. Bimodal (picos en bins bajos y altos). Captura todo el grano fino. Captura todo el grano fino. Se descartó.
Escala de grises Blur (σ)= 0.5 ksize= 3,3	Atenúa el ruido fino sin “borrar” casi nada de textura relevante.	Muy poco suavizado; ruido apenas atenuado. Bimodal muy parecida a “sin blur”, ligeros cambios. Efecto imperceptible. Se descartó
Escala de grises Blur (σ)= 1 ksize= 5,5	Más suavizado, reduce aún más picos de ruido, pero puede diluir patrones muy finos.	Limpia el ruido fino sin borrar microtexturas. Bimodal clara, picos preservados y bins intermedios. Mejor balance detalle/ruido. Se seleccionó.
Escala de grises Blur (σ)= 2 ksize= 7,7	Mínimo suavizado, casi imperceptible visualmente, pero filtra algo de ruido “grano-fino”.	Suavizado excesivo; microtexturas difuminadas. Histograma colapsado en pocos bins, señal intermedia baja. Pierde capacidad de discriminar niveles medios. Se descartó.

en escala de grises con filtro gaussiano de 1. La extracción generó 250,272 *bins* adicionales que se añadieron a los 52 *bins* de LBP-U, produciendo un conjunto de 5,000 registros y 250,324 características. Se volvió a entrenar Random Forest -integrando la información de LBP-U + HOG- obteniéndose unos F1 Macro de 32 % en *condition*, 31 % en *price* y 24 % en *pilling*. Aunque al integrar HOG mejoró ligeramente *condition* y *pilling* y empeoró *price*, el incremento masivo de bins y su poco aporte predictivo, llevó a descartar HOG como descriptor visual para este estudio (Tabla 4.16).

Tabla 4.16: Resultados F1 Macro para experimentos de LBP y LBP+HOG.

Variable	F1 Macro LBP-U	F1 Macro LBP-U + HOG
condition	28 %	32 %
price	35 %	31 %
pilling	22 %	24 %

Con LBP-U seleccionado como mejor descriptor y con sus parámetros y filtro gaussiano validado, se procedió a la extracción definitiva de características de las 31,546 prendas, actividad que generó una emisión estimada de 0.3256 kg de CO₂. El resultado fue un único *dataset* que agrupó los identificadores y rutas de las imágenes *front* y *back*, las 52 variables de histograma LBP-U y las etiquetas originales de *condition*, *price* y *pilling* para el entrenamiento de sub-modelos, además de la variable *usage* destinada al *meta-learner*.

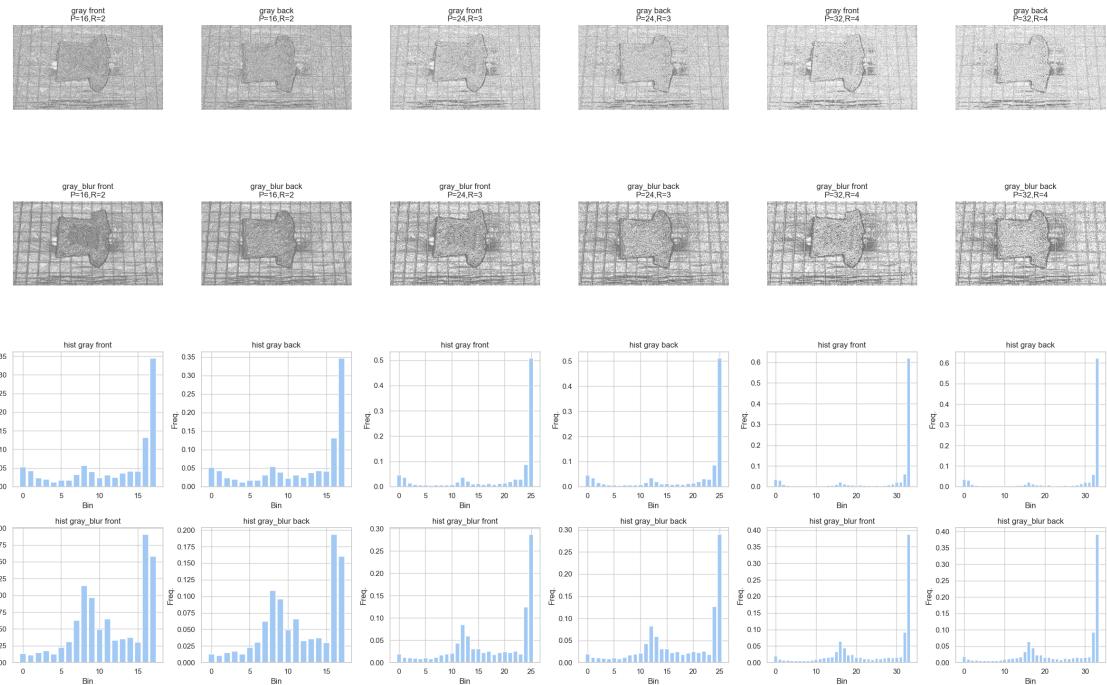


Figura 4.21: Prueba de parámetros para LBP-U en prenda con *condition* nivel 3 -para vista frontal y posterior-, en escala de grises, blur = 1 y kszie = 5,5.

4.3.3 Selección de algoritmos, métricas de evaluación y cálculos de eficiencia para sub-modelos

Para seleccionar el algoritmo más competitivo en la predicción de cada *k-feature* (*condition*, *price* y *pilling*) en su versión reducida, se compararon tres modelos de clasificación supervisada multiclasa que fueran robustos para el manejo de variables numéricas: Random Forest (RF), Support Vector Machine (SVM) y XGBoost. Estos modelos fueron entrenados y ajustados sobre el subconjunto del *dataset* de *train* y *validation* (*split_train*), empleando únicamente los 52 bins LBP-U extraídos de las vistas *front* y *back* de cada prenda, y asegurando la distribución estratificada en cada partición para cada variable objetivo (*condition*: 1 (19 %), 3 (55 %) y 5 (26 %); *price*: <50 (50 %), 50-100 (38 %) y >10 (11 %); *pilling*: 1 (9 %), 3 (39 %) y 5 (51 %)).

Como métricas de evaluación principal se utilizó F1 Macro —para garantizar igual peso a cada clase en escenarios de desbalance— y Recall Macro —para medir la capacidad de recuperación de casos positivos en cada categoría— como métrica secundaria. De forma complementaria, se midió la eficiencia computacional de cada algoritmo en términos de tiempo de entrenamiento, uso máximo de memoria en CPU y emisiones de CO₂. Este enfoque comparativo permitió identificar, para cada

k-feature, el clasificador con mejor rendimiento predictivo, sentando las bases para el entrenamiento final de los sub-modelos.

4.3.4 Búsqueda de hiperparámetros y entrenamiento de sub-modelos

Para la optimización de cada modelo se empleó una búsqueda de hiperparámetros con distintas configuraciones y utilizando principalmente Random Search como método de búsqueda. El proceso consideró técnicas de balanceo y preprocesamiento de variables según el tipo de algoritmo. En Random Forest se variaron el número de estimadores, la profundidad máxima y los tamaños mínimos de división y hoja; en SVM se ajustaron los parámetros C, kernel, gamma y degree; y en XGBoost se exploraron tasas de aprendizaje, configuración de los árboles (n_estimators, max_depth, learning_rate, subsample, colsample_bytree, gamma) y regularizaciones alpha/lambda. La selección final se basó en el métrica F1 Macro obtenida mediante validación cruzada estratificada para cada variable objetivo (*condition*, *price* y *pilling*). En la tabla 4.17 se resume el detalle de búsqueda y selección de hiperparámetros para cada uno.

4.3.5 Resultados entrenamiento selección de sub-modelos

Para la selección del mejor clasificador de cada submodelo, se compararon los resultados de F1 Macro y Recall Macro obtenidos en validación (Tabla 4.18). En *condition*, Random Forest alcanzó un F1 Macro de 43.41 % y un Recall Macro de 44.24 %; aunque XGBoost superó ligeramente en Recall (46.73 %), la diferencia en F1 (43.05 % vs. 43.41 %) justificó la elección de Random Forest. En *price*, XGBoost presentó el mejor equilibrio con un F1 Macro de 49.66 % y un Recall Macro de 50.86 %. Para *pilling*, XGBoost de nuevo mostró mayor capacidad predictiva con un F1 Macro 44.37 % y Recall Macro 45.22 %.

Las matrices de confusión (Figura 4.22) revelaron que, en *condition*, Random Forest clasificó con solidez la clase intermedia (nivel 3) con un Recall de 0.57, pero confundió con frecuencia los extremos (niveles 1 y 5, Recall 0.42 y 0.34 respectivamente); en *price* XGBoost clasificó mejor las prendas de precio intermedio (<50 SEK) con un Recall de 0.70, y mostró errores principalmente entre rangos adyacentes (50–100 vs. >100 SEK); y en *pilling* destacó en la detección del nivel 5 (Recall de 0.61), con aciertos suficientes en el nivel 3 y confusiones con el nivel 1 (Recall 0.61 y 0.26).

Tabla 4.17: Configuración de búsqueda y selección de hiperparámetros para sub-modelos.

Algoritmo	Random Forest	SVM	XGBoost
Parámetros	sampling: passthrough, SMOTE, scaler: passthrough, StandardScaler, n_estimators: [100, 200, 300], max_depth: [5, 10, 15, None], min_samples_split:[2, 5, 10], min_samples_leaf: [1, 2, 4], class_weight: [None, 'balanced']	sampling: passthrough, SMOTE, scaler: passthrough, StandardScaler, C: [0.1, 1, 10, 100], kernel: [linear, rbf, poly], gamma: [scale, 0.01], degree: 3,	sampling: SMOTE, scaler: StandardScaler, n_estimators: [100, 200, 300, 500], max_depth: [3, 6, 9, 12], learning_rate: [0.05, 0.01, 0.1, 0.2], subsample: [0.7, 0.6, 1.0], colsample_bytree: [0.6, 0.7, 1.0], gamma:[0, 0.5, 1, 5], reg_alpha: [0, 0.1, 1], reg_lambda: [1, 1.5, 2]
Método	Random Search, iterations: 30, CV: 5, Scoring: F1_macro	Random Search, iterations: 10, CV: 3, Scoring: F1_macro	Random Search, iterations: 300, CV: 5, Scoring: F1_macro
Selección <i>condition</i>	sampling: SMOTE, scaler: passthrough, n_estimators: 300, max_depth: None, min_samples_split: 2, min_samples_leaf: 4, class_weight: None	sampling: SMOTE, scaler: StandardScaler, C: 0.1, kernel: rbf, gamma: scale, degree: 3, class_weight: None	sampling: SMOTE, scaler: StandardScaler, n_estimators: 300, max_depth: 12, learning_rate: 0.05, subsample: 0.7, colsample_bytree: 0.6, gamma: 1, reg_alpha: 1, reg_lambda: 1
Selección <i>pilling</i>	sampling: SMOTE, scaler: passthrough, n_estimators: 300, max_depth: None, min_samples_split: 2, min_samples_leaf: 2, class_weight: balanced	sampling: SMOTE, scaler: StandardScaler, C: 0.1, kernel: rbf, gamma: scale, degree: 3, class_weight: None	sampling: SMOTE, scaler: StandardScaler, n_estimators: 500, max_depth: 12, learning_rate: 0.05, subsample: 0.7, colsample_bytree: 0.7, gamma: 0.5, reg_alpha: 0, reg_lambda: 2
Selección <i>price</i>	sampling: passthrough, scaler: StandardScaler, n_estimators: 300, max_depth: 15, min_samples_split: 5, min_samples_leaf: 4, class_weight: balanced	sampling: SMOTE, scaler: StandardScaler, C: 100, kernel: poly, gamma: scale, degree: 3, class_weight: None	sampling: SMOTE, scaler: StandardScaler, n_estimators: 500, max_depth: 12, learning_rate: 0.01, subsample: 0.6, colsample_bytree: 0.7, gamma: 0, reg_alpha: 0, reg_lambda: 1.5

Nota: Las técnicas de balanceo se integraron en la búsqueda de mejore hiperparámetros.

En términos de eficiencia computacional (Tabla 4.22), Random Forest para *condition* fue el más rápido en entrenamiento (278 s), demandando una memoria media de 206 MB y generando la menor huella de carbono en esta fase (0.005006 kg CO₂). Para *price*, XGBoost registró un tiempo de entrenamiento de 3239 s, un consumo de 170 MB y emisiones de 0.148659 kg CO₂ en entrenamiento, siendo el segundo más ágil tras Random Forest y con un coste energético intermedio. En *pilling*, XGBoost también ocupó la segunda posición en velocidad (3249 s), con un uso de memoria de 228 MB y la mayor emisión de CO₂ en entrenamiento (0.155633 kg). En conjunto, el proceso de entrenamiento y validación de los tres sub-modelos generó aproximadamente 1.59 kg de CO₂, balanceando capacidades predictivas y criterios de sostenibilidad.

Tabla 4.18: Métricas de evaluación sub-modelos en validación.

Sub-modelo	Algoritmo	F1-Score Macro	Recall Macro
condition	Random Forest	43.41 %	44.24 %
	SVM	39.82 %	45.25 %
	XGBoost	43.05 %	46.73 %
price	Random Forest	49.13 %	49.66 %
	SVM	46.84 %	51.58 %
	XGBoost	49.66 %	50.86 %
pilling	Random Forest	44.12 %	44.05 %
	SVM	40.33 %	45.23 %
	XGBoost	44.37 %	45.22 %

Tabla 4.19: Reporte de clasificación Random Forest para *condition* en validación.

Clase	Precision	Recall	F1-score	Support
1	0.30	0.42	0.35	1197
3	0.63	0.57	0.60	3472
5	0.36	0.34	0.35	1640
accuracy			0.48	6309
macro avg	0.43	0.44	0.43	6309
weighted avg	0.50	0.48	0.49	6309

Tabla 4.20: Reporte de clasificación XGBoost para *price* en validación.

Clase	Precision	Recall	F1-score	Support
50-100	0.47	0.37	0.42	2405
<50	0.63	0.70	0.67	3171
>100	0.37	0.45	0.41	733
accuracy			0.55	6309
macro avg	0.49	0.51	0.50	6309
weighted avg	0.54	0.55	0.54	6309

Tabla 4.21: Reporte de clasificación XGBoost para *pilling* en validación.

Clase	Precision	Recall	F1-score	Support
1	0.16	0.26	0.20	573
3	0.55	0.48	0.51	2460
5	0.62	0.61	0.62	3276
accuracy			0.53	6309
macro avg	0.44	0.45	0.44	6309
weighted avg	0.55	0.53	0.54	6309

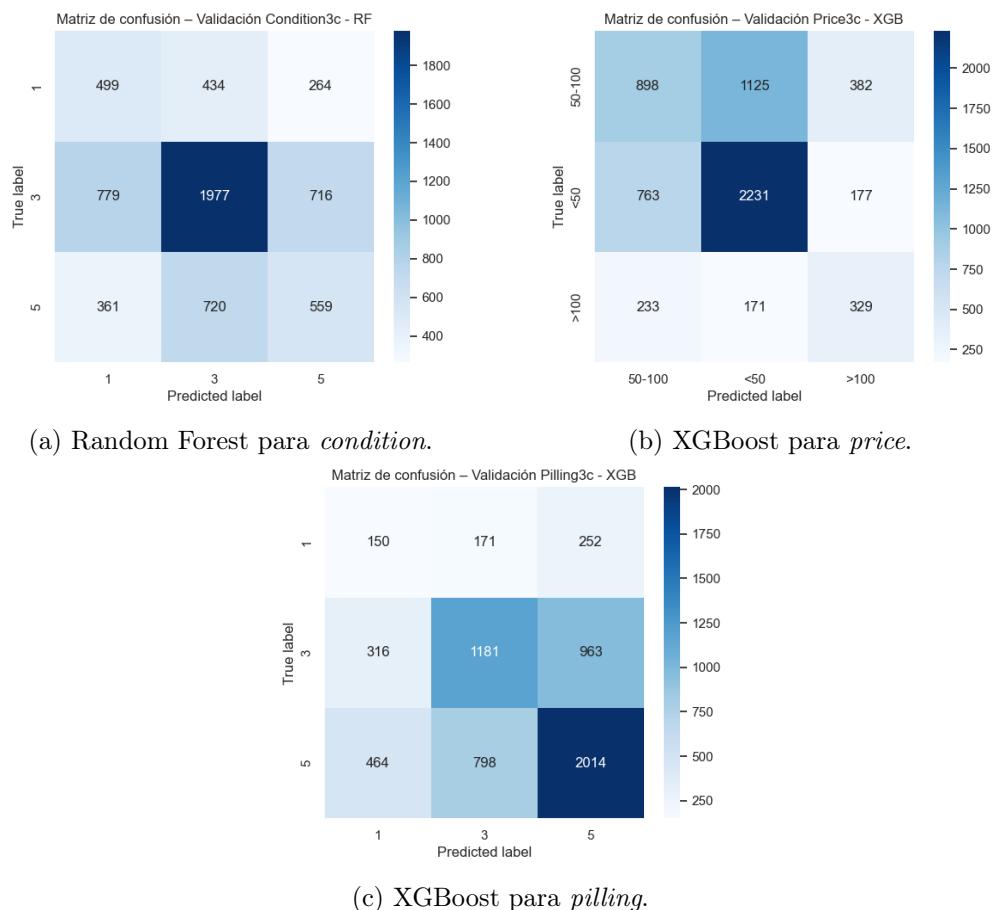


Figura 4.22: Matriz de confusión para Random Forest (*condition*), XGBoost (*price*) y XGBoost (*pilling*) en validación.

Tabla 4.22: Métricas de eficiencia computacional de sub-modelos en validación.

sub-modelo	Algoritmo	Tiempo E (s)	Tiempo I (s)	Mem CPU E (MB)	Mem CPU I (MB)	CO ₂ E (kg)	CO ₂ I (kg)	CO ₂ total (kg)
condition	RF	278.47	0.23	205.52	0.05	0.005006	0.005011	0.010017
	SVM	4908.58	17.10	389.07	2.92	0.088574	0.088867	0.177441
	XGBoost	3625.23	0.10	112.9	0.10	0.150742	0.000241	0.150983
price	RF	1698.88	0.18	82.18	0.02	0.030416	0.030419	0.060835
	SVM	3678.61	1.59	11.52	0.31	0.093534	0.093563	0.187097
	XGBoost	3238.91	0.08	170.03	0.05	0.148659	0.148662	0.297321
pilling	RF	1926.75	0.26	188.04	0.18	0.039836	0.039842	0.100677
	SVM	3533.80	7.96	350.20	2.86	0.100058	0.100195	0.247932
	XGBoost	3248.79	0.11	227.89	0.39	0.155633	0.155637	0.358156
								Total 1.590459

Nota: E = Entrenamiento, I = Inferencia, Mem CPU = Memoria en CPU, RF = Random Forest, SVM = Support Vector Machine.

4.4 Fase 3: Generación de *meta-features* y ensamblado del *meta-learner*

4.4.1 Generación de *meta-features*

Tras completar el entrenamiento y validación de los sub-modelos, se procedió a generar las *meta-features* -predicciones que cada sub-modelo genera de una variable con las que se entrenará el *meta-learner* final. Para ello, se aplicó una estrategia de predicción out-of-fold (OOF) en la que un modelo realiza predicciones sobre partes del conjunto de datos que no han sido usadas para entrenarlo, durante un proceso de validación cruzada. De este modo, cada predicción se genera por un modelo que no vio esa muestra durante su propio ajuste, generando predicciones "honestas".

Para generar las predicciones OOF se empleó el conjunto completo de entrenamiento train_full (Tabla 4.3). Este se partió en cinco pliegues estratificados, de modo que en cada iteración el sub-modelo se ajustara con cuatro pliegues y, mediante el método *predict*, obtuviera un vector de etiquetas para el pliegue restante. El vector resultante de predicciones de cada sub-modelo se volcó directamente a las columnas *oof_condition*, *oof_price* y *oof_pilling* para su almacenamiento en el *dataset* a utilizar en el entrenamiento del *meta-learner*. De este modo, las 31,546 muestras de train_full recibieron las predicciones de cada *k-feature*. La generación OOF consumió un total de 0,011 kg de CO₂ y recursos de cómputo moderados (Tabla 4.23).

Tabla 4.23: Métricas de eficiencia computacional en generación de *meta-features* para conjunto de entrenamiento (OOF).

sub-modelo	Algoritmo	Tiempo E (s)	Mem CPU E (MB)	CO ₂ E (kg)
condition	Random Forest	71.37	1.18	0.001292
price	XGBoost	52.15	2.45	0.003512
pilling	XGBoost	140.11	2.06	0.006195
Total				0.010999

Nota: E = Entrenamiento y Mem CPU = Memoria en CPU.

Tras completar la generación OOF, se pasó a producir las *meta-features* del conjunto de prueba (*test*). Para ello, cada sub-modelo seleccionado -*condition*, *price* y *pilling*- se reentrenó con sus hiperparámetros óptimos empleando la totalidad del *train_full* (80 % del *split_test*), para que pudiera inferir con el máximo de información en el conjunto de *test*. Al modelo, entrenado con todos los datos de entrenamiento, se le aplicó el método *predict* sobre el *hold-out test* (20 % del *split_test*), volcando las etiquetas predichas de cada *k-features* en las columnas de *pred_condition*, *pred_price* y *pred_pilling*. Con ello, se obtuvieron las predicciones de *test* que, junto a las OOF de entrenamiento, conformaron el *dataset* completo de *meta-features* para el entrenamiento y prueba del *meta-learner*.

En relación a los resultados en *test*, los sub-modelos mantuvieron o mejoraron su desempeño respecto a validación (Tabla 4.24): el sub-modelo de *condition* alcanzó un F1 Macro de 44,09 % (vs. 43,41 % en validación) y un Recall Macro de 43,47 % (vs. 44,24 %), con una ligera ganancia en F1 y una mínima pérdida en recall; el de *price* elevó su F1 Macro a 51,79 % (vs. 49,66 %) y mantuvo un Recall macro de 50,07 % (vs. 50,86 %); y el de *pilling* mejoró ambas métricas hasta 48,09 % y 46,86 % respectivamente (vs. 44,37 % y 45,22 %). Las matrices de confusión (Figura 4.23) confirmaron estos cambios: en *condition* aumentó la clasificación correcta del nivel 3; en *price* mejoró la detección de prendas <50 SEK, con leves desplazamientos entre categorías adyacentes; y en *pilling* crecieron los aciertos en los niveles 5 y 3 a costa de menos errores hacia el nivel 1. En la fase de *test*, los sub-modelos emitieron en conjunto 0.026627 kg de CO₂ (Tabla 4.28).

Tabla 4.24: Métricas de evaluación de sub-modelos en *test*.

Sub-modelo	Algoritmo	F1-Score Macro	Recall Macro
condition	Random Forest	44.09 %	43.47 %
price	XGBoost	51.79 %	50.07 %
pilling	XGBoost	48.09 %	46.86 %

Tabla 4.25: Reporte de clasificación Random Forest para *condition* en *test*.

Clase	Precision	Recall	F1-score	Support
1	0.31	0.41	0.35	1227
3	0.62	0.58	0.60	3441
5	0.38	0.33	0.35	1642
accuracy			0.48	6310
macro avg	0.43	0.44	0.43	6310
weighted avg	0.50	0.48	0.49	6310

Tabla 4.26: Reporte de clasificación XGBoost para *price* en *test*.

Clase	Precision	Recall	F1-score	Support
50-100	0.50	0.37	0.42	2399
<50	0.65	0.72	0.68	3213
>100	0.34	0.46	0.39	698
accuracy			0.56	6310
macro avg	0.50	0.52	0.50	6310
weighted avg	0.56	0.56	0.55	6310

Tabla 4.27: Reporte de clasificación XGBoost para *pilling* en *test*.

Clase	Precision	Recall	F1-score	Support
1	0.20	0.32	0.25	607
3	0.56	0.49	0.53	2462
5	0.64	0.63	0.64	3241
accuracy			0.55	6310
macro avg	0.47	0.48	0.47	6310
weighted avg	0.57	0.55	0.56	6310

Tabla 4.28: Métricas de eficiencia computacional de sub-modelos en *test*.

sub-modelo	Algoritmo	Tiempo E (s)	Tiempo I (s)	Mem CPU E (MB)	Mem CPU I (MB)	CO ₂ E (kg)	CO ₂ I (kg)	CO ₂ total (kg)
condition	RF	76.20	0.24	220.72	0.30	0.002606	0.002612	0.005218
price	XGBoost	16.04	0.07	154.82	0.25	0.003793	0.003795	0.007588
pilling	XGBoost	41.66	0.11	132.18	0.21	0.006909	0.006912	0.013821
Total								0.026627

Nota: E = Entrenamiento, I = Inferencia, Mem CPU = Memoria en CPU, RF = Random Forest.

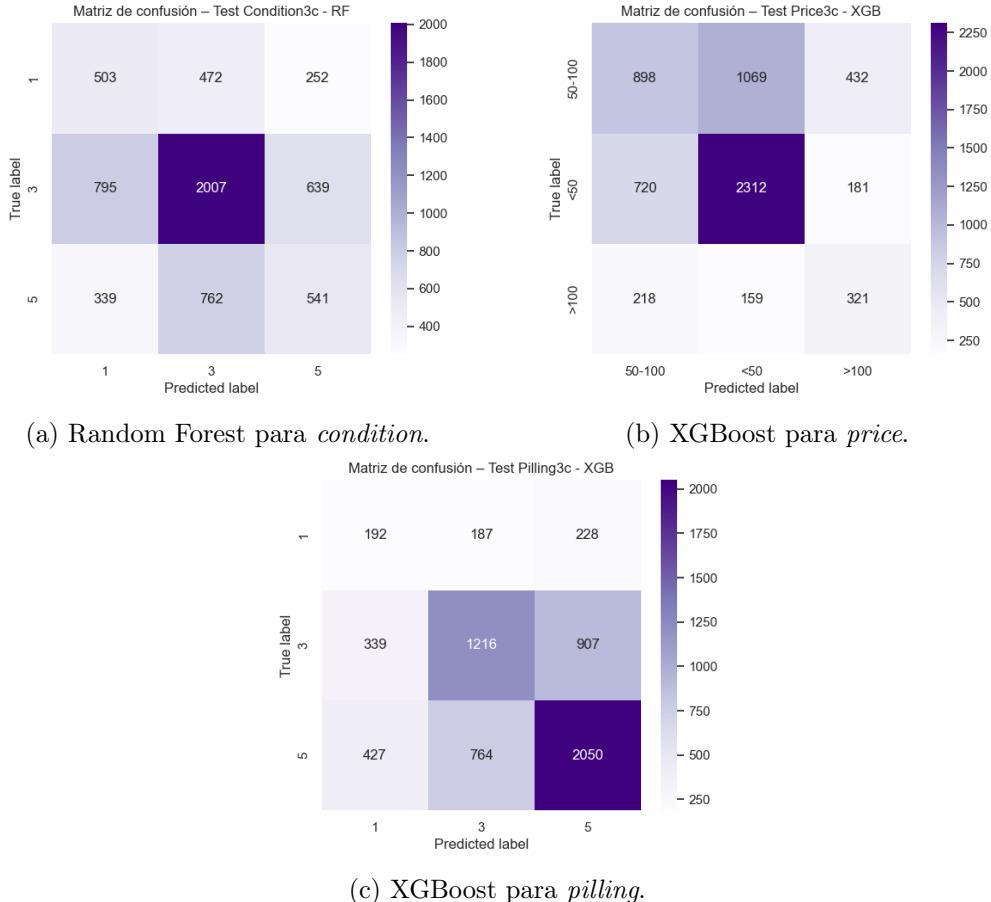


Figura 4.23: Matriz de confusión para Random Forest (*condition*), XGBoot (*price*) y XGBoost (*pilling*) en *test*.

4.4.2 Ensamblado (*staking*) y configuración de *meta-learner*

Para el entrenamiento y evaluación del *meta-learner* en el esquema de ensamblado (*staking*) se utilizó el *dataset* de *meta-features* generado en la sección anterior, conformado por las predicciones OOF de cada sub-modelo sobre el conjunto de entrenamiento, las predicciones sobre test y las variables originales de la *metadata*. Este conjunto mantuvo la partición inicial de *split_test*, destinando *train_full* para el entrenamiento del *meta-learner* y *test* para su prueba final.

Como algoritmo de entrenamiento se seleccionó CatBoost, clasificador ganador en la fase de experimentación de modelo base, empleando sus mejores hiperparámetros (ver Sección 4.2.8).

Para evaluar el rendimiento predictivo del *meta-learner* sobre la variable objetivo principal *usage* y sus categorías (0: *Reuse*, 1: *Export* y 2: *Recycle*), se diseñaron cuatro escenarios de comparación. En el Escenario 1 (entrenamiento y test con *metada*), se entrenó y evaluó el modelo base usando únicamente las variables originales de la

metadata, estableciendo la línea base “ideal” al contar con información perfecta en ambas fases. En el Escenario 2 (entrenamiento con *metadata* y test con predicciones), se ajustó un modelo que entrenó con la *metadata* real y se evaluó sustituyendo éstas por las predicciones de los sub-modelos, con el fin de cuantificar la pérdida de rendimiento cuando solo se dispone de salidas automáticas en producción. En el Escenario 3 (entrenamiento con predicciones OOF y test con *metadata*), el *meta-learner* se entrenó con las predicciones out-of-fold de los sub-modelos y se evaluó con la *metadata* real, sirviendo como cota optimista al combinar predicciones en entrenamiento con datos “perfectos” en prueba. Finalmente, en el Escenario 4 (entrenamiento y test con predicciones), el *meta-learner* puro se entrenó con las predicciones OOF y se probó con las salidas de los sub-modelos sobre *test*, reflejando el flujo realista en producción y constituyendo el referente para comparar con el Escenario 1.

4.4.3 Resultados finales del *meta-learner*

Una vez completados los entrenamientos en los cuatro escenarios —con CatBoost y sus hiperparámetros óptimos— sobre el conjunto completo de entrenamiento y evaluados en *test* (*hold-out*), se obtuvieron los siguientes resultados de rendimiento predictivo en *usage* (Tabla 4.29). En el Escenario 1 (modelo base), que emplea únicamente las variables originales de *metadata*, el F1 Macro alcanzó 65.07 % y el Recall Macro 66.24 %, confirmando la capacidad predictiva del modelo cuando cuenta con información real completa. En el Escenario 4 (*meta-learner* puro), entrenado y probado exclusivamente con las predicciones de los submodelos, se registró un F1 Macro de 41.52 % y un Recall Macro de 47.65 %, reflejando la efectividad del esquema ensamblado en condiciones reales de producción. Aunque supone una caída de 23.55 p.p. en F1 Macro frente al modelo base, éste valida la viabilidad del *meta-learner* para predecir *usage* a partir de la descripción visual de las imágenes frontal y posterior de una prenda. Los resultados intermedios de los Escenarios 2 (modelo base híbrido) y 3 (*meta-learner* híbrido) registraron F1 Macro de 44.31 % y 57.57 %, y Recall Macro de 46.19 % y 62.88 %, respectivamente, ilustrando la pérdida en la capacidad predictiva del modelo al entrenar con *metadata* real y depender de predicciones en *test*, así como la ganancia en predicción al combinar predicciones en entrenamiento con datos perfectos en la evaluación.

En el análisis de las matrices de confusión y los reportes de clasificación de los cuatro escenarios, el Escenario 1 (modelo base) destacó por un Recall de 0.81 tanto en *Reuse* como en *Export* (Tabla 4.30, Figura 4.24), aunque mostró un rendimiento muy bajo en *Recycle* (Recall de 0.37), confundiendo buena parte de sus casos con

Export. Por su parte, el Escenario 4 (*meta-learner* puro) mejoró notablemente la recuperación de *Recycle* hasta 0.51, mantuvo un Recall competitivo de 0.53 en *Reuse* y evidenció la mayor confusión en *Export*, con un Recall de 0.39 (Tabla 4.33, Figura 4.24). Los escenarios intermedios confirman estos patrones: el Escenario 2 (modelo base híbrido) logró un Recall de 0.61 en *Reuse*, pero retrocedió a 0.51 en *Export* y a 0.27 en *Recycle* (Tabla 4.31, Figura 4.24), mientras que el Escenario 3 (*meta-learner* híbrido) mostró ser el más equilibrado, con Recalls de 0.75, 0.61 y 0.53 en *Reuse*, *Export* y *Recycle* respectivamente (Tabla 4.32, Figura 4.24), reflejando la ventaja de entrenar con predicciones y evaluar con *metadata* real.

En relación a la eficiencia computacional de los modelos, el Escenario 1 (modelo base) fue el que más tiempo requirió en entrenamiento (5.48 s) y, a pesar de registrar el mayor uso de CPU (26.77 MB), emitió la menor cantidad de CO₂ total (0.000207 kg). El Escenario 2 (modelo base híbrido) entrenó en 4.30 s, consumió 13.11 MB memoria de CPU y generó 0.017443 kg de CO₂ total, superando levemente la emisión del modelo base. Los Escenarios 3 (*meta-learner* híbrido) y 4 (*meta-learner* puro) mostraron tiempos de entrenamiento parejos —4.38 s y 4.43 s respectivamente— con uso de CPU similar (13.66 MB y 11.21 MB), y fueron los modelos que más CO₂ produjeron, con 0.048478 kg y 0.062522 kg respectivamente. Con ello, la fase completa de entrenamiento e inferencia sumó un total de 0.128650 kg de CO₂.

Tabla 4.29: Matriz de comparación de escenarios modelo base vs *meta-learner*.

	Test: Metadata	Test: Predicciones
Train: metadata	Escenario 1 <i>(modelo base)</i> F1 Macro: 65.07 % Recall Macro: 66.24 %	Escenario 2 <i>(modelo base híbrido)</i> F1 Macro: 44.31 % Recall Macro: 46.19 %
Train: OOF	Escenario 3 <i>(meta-learner híbrido)</i> F1 Macro: 57.57 % Recall Macro: 62.88 %	Escenario 4 <i>(meta-learner)</i> F1 Macro: 41.52 % Recall Macro: 47.65 %

Tabla 4.30: Reporte de clasificación modelo base Catboost en *usage*.

Clase	Precision	Recall	F1-score	Support
Reuse	0.95	0.81	0.87	3832
Export	0.63	0.81	0.71	2029
Recycle	0.37	0.37	0.37	449
accuracy			0.78	6310
macro avg	0.65	0.66	0.65	6310
weighted avg	0.81	0.78	0.78	6310

Tabla 4.31: Reporte de clasificación modelo base híbrido Catboost en *usage*.

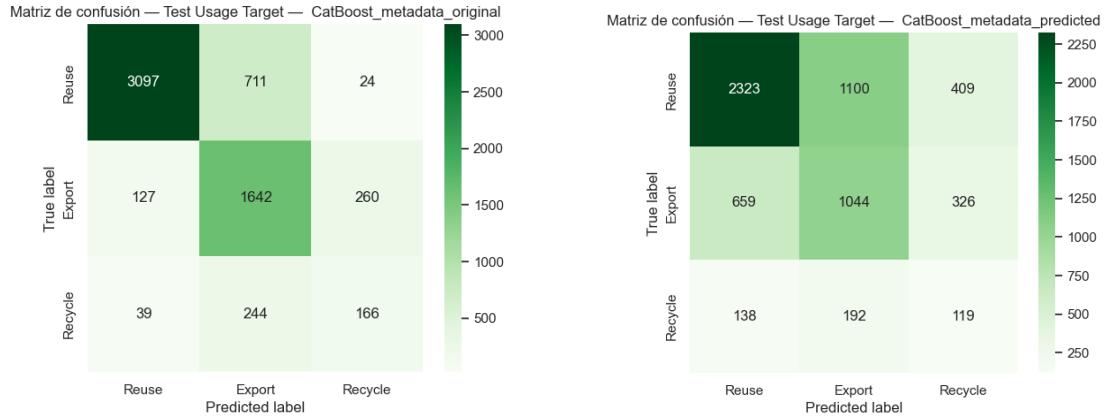
Clase	Precision	Recall	F1-score	Support
Reuse	0.74	0.61	0.67	3832
Export	0.45	0.51	0.48	2029
Recycle	0.14	0.27	0.18	449
accuracy			0.55	6310
macro avg	0.44	0.46	0.44	6310
weighted avg	0.61	0.55	0.57	6310

Tabla 4.32: Reporte de clasificación *meta-learner* híbrido Catboost en *usage*.

Clase	Precision	Recall	F1-score	Support
Reuse	0.95	0.75	0.84	3832
Export	0.66	0.61	0.64	2029
Recycle	0.17	0.53	0.25	449
accuracy			0.69	6310
macro avg	0.59	0.63	0.58	6310
weighted avg	0.80	0.69	0.73	6310

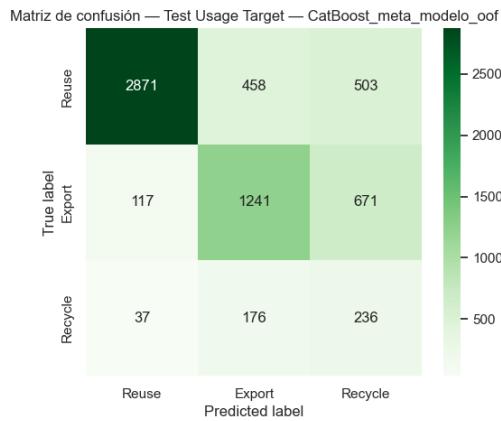
Tabla 4.33: Reporte de clasificación *meta-learner* Catboost en *usage*.

Clase	Precision	Recall	F1-score	Support
Reuse	0.79	0.53	0.63	3832
Export	0.49	0.39	0.43	2029
Recycle	0.11	0.51	0.18	449
accuracy			0.48	6310
macro avg	0.46	0.48	0.42	6310
weighted avg	0.65	0.48	0.54	6310

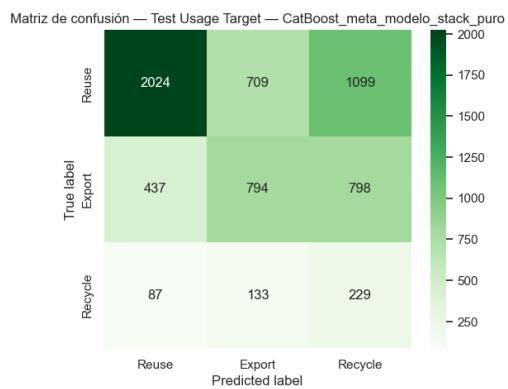


(a) Modelo base final para *usage*.

(b) Modelo base híbrido para *usage*.



(c) *Meta-learner* híbrido para *usage*.



(d) *Meta-learner* para *usage*.

Figura 4.24: Matriz de confusión para modelo base, modelo base híbrido, *meta-learner* híbrido y *meta-learner* sobre *usage*.

Tabla 4.34: Métricas de eficiencia computacional para modelo base, modelo base híbrido, *meta-learner* híbrido y *meta-learner* sobre *usage*.

Escenario	Tiempo E (s)	Tiempo I (s)	Mem CPU E (MB)	Mem CPU I (MB)	CO ₂ E (kg)	CO ₂ I (kg)	CO ₂ total (kg)
Modelo base	5.48	0.02	26.77	0.32	0.000103	0.000104	0.000207
Modelo bases híbrido	4.30	0.03	13.11	0.64	0.006987	0.010456	0.017443
Meta-learner híbrido	4.38	0.02	13.66	0.26	0.019233	0.021657	0.048478
Meta-learner	4.43	0.02	11.21	0.62	0.030531	0.031991	0.062522
Total							0.128650

Nota: E = Entrenamiento, I = Inferencia, Mem CPU = Memoria en CPU.

Capítulo 5

Conclusiones

Para concluir esta investigación, en este capítulo se abordan cinco dimensiones: la estimación de la huella de carbono del *pipeline* de meta-aprendizaje y su comparación con modelos de referencia; las conclusiones finales que sintetizan el cumplimiento de los objetivos, los hallazgos y las contribuciones metodológicas; las líneas futuras de investigación propuestas para ampliar y robustecer el enfoque; las consideraciones éticas, sociales y el análisis del coste económico del proyecto.

5.1 Estimación de la huella de carbono

En el marco de esta investigación, en el que se implementó un *meta-learner* diseñado bajo los principios de *Green AI* para optimizar tanto el rendimiento como la sostenibilidad, se cuantificó la emisión de huella de carbono generada durante todo el proceso de desarrollo. El cálculo, utilizando la calculadora de emisión y equivalencia comparativa de CodeCarbon, arrojó un total de 2.39 kg de CO₂ equivalente (Tabla 5.1). Para ponerlo en perspectiva, estas emisiones equivalen a recorrer aproximadamente 20 km en un coche (0.12 kg CO₂/km) y representan cerca del 1 % de las emisiones semanales de un ciudadano medio en Estados Unidos (256 kg CO₂/semana).

Al comparar las emisiones de CO₂ generadas durante el entrenamiento de un modelo estándar de NLP con las correspondiente al *meta-learner* desarrollado en esta investigación, se evidencia una reducción drástica en el coste de emisión. Por ejemplo, mientras que entrenar un Transformer “big” emite 87 kg de CO₂ y ejecutar un *pipeline* común de NLP emite 18 kg (Strubell et al., 2019), el *meta-learner* generó únicamente 2.39 kg de CO₂ (Tabla 5.2).

Estos resultados evidencian que es posible combinar técnicas avanzadas de aprendizaje automático con prácticas responsables, minimizando el impacto medioambiental sin sacrificar la calidad de un modelo.

Tabla 5.1: Emisiones de CO₂ equivalente por fase de desarrollo.

Fase	Actividad	CO ₂ (kg)
Fase 1	Entrenamiento/validación modelo base y selección <i>k-features</i>	0.13
	Optimización Modelo base con <i>k-features</i>	0.17
	Sub total Fase 1	0.30
Fase 2	Extracción de características visuales (LBP-U)	0.33
	Entrenamiento/validación y selección de sub-modelos visuales	1.59
	Sub total Fase 2	1.92
Fase 3	Generación de <i>meta-features</i> conjunto entrenamiento (OOF)	0.01
	Generación de <i>meta-features</i> conjunto prueba	0.03
	Entrenamiento y comparación <i>meta-learner</i> vs modelo base	0.13
Sub total Fase 3		0.17
Total		2.39

Tabla 5.2: Emisiones de CO₂ equivalentes de entrenar modelos de NLP comunes (Strubell et al., 2019) y el *meta-learner* de esta investigación.

Entrenamiento de modelo	CO ₂ (kg) emitido
NLP pipeline (parsing, SRL)	18
con Tuning y experimentación (pipeline completo)	35,593
Transformer (big)	87
con arquitectura de búsqueda neuronal	284,019
Meta-learner de la investigación	2.39

Nota: Los primeros tres modelos consideran solo entrenamiento de un modelo en GPU. El meta-learner considera todo el proceso de desarrollo en CPU.

5.2 Conclusiones

En esta investigación se planteó como objetivo principal desarrollar un *meta-learner* capaz de predecir el destino final de prendas de segunda mano a partir de sus imágenes. Para ello, se definieron cuatro objetivos específicos: identificar las características de la prenda más relevantes para la predicción de *usage* (variable objetivo de destino final); extraer dichos atributos mediante descriptores visuales aplicados a las imágenes y, con ellos, construir sub-modelos independientes; ensamblar las predicciones de estos sub-modelos en un *meta-learner* capaz de clasificar con precisión el destino final de

cada prenda; y, finalmente, enmarcar todo el flujo de trabajo bajo un enfoque de *Green AI*, midiendo y optimizando el rendimiento y la huella de carbono en cada una de sus fases.

En relación al primer objetivo, las variables *condition*, *price* y *pilling* resultaron ser las características de mayor relevancia y con mayor capacidad predictiva ante *usage*, las que utilizadas para entrenar un modelo base inicial con CatBoost permitieron alcanzar un F1 Macro de 65.89 % en validación. Para la extracción de estas características a partir de la imagen frontal y posterior de la prenda, el descriptor visual de texturas LBP-U (con P = 24 y R = 3) con filtro gaussiano (con blur = 1 y ksize = 5,5) resultó ser la configuración óptima, permitiendo mantener una alta resolución y capturar micro-texturas finas. Los sub-modelos visuales entrenados sobre estas características, alcanzaron un F1 Macro de 43.41 % en validación y 44.09 % en *test* para *condition* con Random Forest, 49.66 % (validación) y 51.79 % (*test*) para *price* con XGBoost y 44.37 % (validación) y 48.09 % (*test*) para *pilling* con XGBoost. Finalmente, el meta-learner ensambló eficazmente estas predicciones y logró un F1 Macro de 41.52 % con CatBoost, apenas 23.55 puntos porcentuales menos que el modelo base re-entrenado con metadata completa, que alcanzó un F1 Macro de 65.07 % en *test*. Además, el proceso completo emitió tan solo 2.39 kg de CO₂ equivalente, lo que supone un ahorro de emisiones de 7.5 veces frente a un pipeline básico de NLP y 36 veces frente a un Transformer “big” entrenado en GPU.

Con este trabajo se introducen dos aportaciones claves al campo de la clasificación de ropa de segunda mano mediante técnicas de inteligencia artificial. En primer lugar, demuestra la eficacia de un *meta-learner* para ensamblar de forma coherente sub-modelos visuales independientes, logrando un sistema unificado que aproxima al rendimiento de modelos con metadatos perfectos. En segundo lugar, integra de manera pionera la medición de la huella de carbono como criterio operativo: la cuantificación y monitoreo de las emisiones se conciben desde el diseño del *pipeline*, validando así la viabilidad de los principios de *Green AI* en cada fase del desarrollo. Este doble enfoque —técnico y medioambiental— ofrece un referente reproducible y ligero para futuras aplicaciones de meta-aprendizaje responsable en la industria textil y otros dominios.

5.3 Líneas de investigación futuras

Las perspectivas de ampliación de esta investigación abarcan desde la exploración de nuevas variables predictoras, la generación de nuevas imágenes que robustezcan los

modelos, hasta la validación y despliegue del sistema en escenarios reales.

En primer lugar, se propone profundizar en el estudio de las variables no consideradas —como *material*, *colors* y *brand*, entre otras— para evaluar su capacidad predictiva, su correlación con *usage* y otras variables, y su potencial aporte informativo en el modelo. Asimismo, resulta fundamental enriquecer y equilibrar el conjunto de imágenes de prendas de segunda mano, incorporando muestras adicionales que cubran de manera más uniforme las distintas categorías de destino, así como los niveles de *condition* y *pilling*, con el fin de robustecer los sub-modelos visuales y las categorías de predicción destino final del *meta-learner*.

Luego, es recomendable diversificar los descriptores de textura, probando distintas configuraciones de LBP-U, añadiendo otras técnicas de extracción y aplicando preprocesados específicos (p.ej. enmascarado por tipo de prenda) para evitar dependencia de un único descriptor y optimizar la captura de información relevante. En paralelo, la evaluación de arquitecturas de aprendizaje profundo que trabajen directamente sobre las imágenes —tanto para la predicción de atributos intermedios como de *usage* final— permitiría comparar rendimiento predictivo y huella de carbono, identificando el punto óptimo entre precisión y sostenibilidad.

Finalmente, explorar el desarrollo de un dispositivo de escaneo integrado —*hardware* con sensores y *software* con el *meta-learner*— abriría la vía al uso del modelo en producción, facilitando la validación en entornos industriales como centros de clasificación de ropa de segunda mano y la calibración continua del modelo según condiciones reales de operación.

5.4 Consideraciones éticas y sociales

La adopción de soluciones de inteligencia artificial en el ámbito textil exige un análisis riguroso de sus implicaciones éticas y sociales. Este apartado explora tres dimensiones clave: equidad y sesgos; privacidad y consentimiento; y responsabilidad ambiental y social.

En relación a la equidad y los sesgos, es fundamental reconocer que los estilos, marcas y preferencias de uso varían significativamente entre culturas y regiones. Un modelo entrenado mayoritariamente con datos de un país o mercado concreto puede perpetuar estereotipos o desfavorecer minorías. Por ello, se recomienda auditar regularmente los resultados según atributos demográficos y de moda, y enriquecer el conjunto de entrenamiento con muestras diversas para mitigar posibles desequilibrios.

Respecto a la privacidad y el consentimiento, es necesario garantizar la protección de datos personales mediante la eliminación de cualquier elemento identificable en las imágenes —p. ej. en la versión del *dataset* utilizada en esta investigación se descartaron entre 4,000 y 5,000 fotografías con rostros, manos o información personal sensible— y asegurar que todas las imágenes cuenten con las licencias y permisos necesarios. Al mismo tiempo, debe cumplirse la normativa de protección de datos (GDPR), aplicando principios de minimización, transparencia y derecho a la supresión, de modo que el uso del material visual sea lícito, leal y respetuoso con la privacidad de las personas. Además, al trabajar con prendas de marcas comerciales, se recomienda obtener las autorizaciones para evitar posibles infracciones de copyright o de marca registrada, cuando sea necesario.

Finalmente, la responsabilidad ambiental y social se extiende más allá de la huella de carbono del modelo. Al potenciar una economía de moda circular, esta herramienta puede optimizar la reutilización y el reciclaje de prendas, reduciendo el desperdicio textil. Integrar y democratizar esta tecnología en la cadena de valor fomenta prácticas responsables y contribuye a la transición hacia un sector de la moda más sostenible y respetuoso con el medio ambiente.

5.5 Costo económico del proyecto

Para el desarrollo completo de esta investigación, se requirió la dedicación de un *Data Scientist Junior* a jornada parcial (20 h/semana) durante 22 semanas (5 meses de trabajo, de abril a agosto del 2025), lo que equivale a un total de 440 h (Tabla 5.3). Con un salario bruto promedio anual de 30,000 € (Glassdoor, 2025), equivalente a 14 €/h, el coste total de personal equivale a 6,346 €.

Tabla 5.3: Horas dedicadas por actividad

Actividad	Horas dedicadas	Distribución
Configuración entorno y metadatos	40	9 %
Fase 1: EDA, k-features y modelo base	160	36 %
Fase 2: EDA imágenes y sub-modelos	120	27 %
Fase 3: Ensamblabo y meta-learner	40	9 %
Redacción investigación	80	18 %
Total	440	100 %

Respecto a la infraestructura tecnológica, se utilizó un portátil ASUS ROG Zephyrus G16 GA605 (ASUS ROG, 2024), con un costo de adquisición de 2,500 € y cuya vida

útil estimada es de 3 años o 26,280 h de uso. Con un coste por hora calculado en 0.10 €/h y 440 h de uso, la depreciación atribuible al proyecto representa 42 €.

Por último, el consumo energético asociado al uso de CPU (TDP de 28 W durante 440 h) generó un gasto de tan solo 3 € en electricidad, calculado a 0.24 €/kWh según la factura real de suministro.

En conjunto, el coste económico total del proyecto ascendió a 6,391 € (Tabla 5.4), cifra que cubre la mano de obra, el equipamiento computacional y la energía, sin incurrir en gastos adicionales de licencias o suscripciones. Este balance pone de manifiesto la eficiencia económica de un pipeline de meta-aprendizaje diseñado para ser a la vez efectivo y sostenible.

Tabla 5.4: Coste total del proyecto

Ítem	Costo (€)
Data Scientist Junior	6,346
Portátil	42
Electricidad	3
Total	6,391

Bibliografía

- Alonso, J. (2025, 17 de mayo). *Una campaña «rescata» prendas de ropa que se acumulan en el desierto de Atacama*. Deutsche Welle. Consultado el 10 de agosto de 2025, desde <https://www.dw.com/es/una-campa%C3%B1a-rescata-prendas-de-ropa-que-se-acumulan-en-el-desierto-de-atacama/a-72577489>
- Amat Rodrigo, J. (s.f.). *Random Forest con Python*. Ciencia de Datos. Consultado el 13 de agosto de 2025, desde https://www.cienciadedatos.net/documentos/py08_random_forest_python.html Disponible bajo licencia Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0).
- ASUS ROG. (2024). ROG Zephyrus G16 (2024, GA605) Specifications [Accessed: 2025-09-03].
- Barbierato, E., & Gatti, A. (2024). Toward Green AI: A Methodological Survey of the Scientific Literature. *IEEE Access*, 12, 23989-24013. <https://doi.org/10.1109/ACCESS.2024.3360705>
- Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785-794.
- Cheng, W.-H., Song, S., Chen, C.-Y., Hidayati, S. C., & Liu, J. (2021). Fashion meets computer vision: A survey. *ACM Computing Surveys (CSUR)*, 54(4), 1-41.
- CodeCarbon. (2024). *Methodology*. CodeCarbon Documentation. Consultado el 10 de agosto de 2025, desde <https://mlco2.github.io/codcarbon/methodology.html>
- 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.
- Ellen MacArthur Foundation. (2017). *A New Textiles Economy: Redesigning Fashion's Future*. Ellen MacArthur Foundation. <https://ellenmacarthurfoundation.org/a-new-textiles-economy>
- Ellen MacArthur Foundation & Google. (2019). *Artificial Intelligence and the Circular Economy: AI as a Tool to Accelerate the Transition* (With research and analytical support from McKinsey & Company). Ellen MacArthur Foundation.

- <https://www.ellenmacarthurfoundation.org/artificial-intelligence-and-the-circular-economy>
- Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (2nd). O'Reilly Media.
- Glassdoor. (2025). Junior Data Scientist Salary in Barcelona [Accessed: 2025-09-03].
- Gonzalez, R. C., & Woods, R. E. (2018). *Digital Image Processing* (4th). Pearson.
- Grandini, M., Bagli, E., & Visani, G. (2020). Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756*.
- Hermansson, S. (2023). *Learning Embeddings for Fashion Images* [Tesis de maestría, Linköping University, Computer Vision]. <https://www.diva-portal.org/smash/record.jsf?pid=diva2:1763534>
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An Introduction to Statistical Learning: with Applications in R* (2nd). Springer. <https://www.statlearning.com>
- Janiesch, C., Zschech, P., & Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets*, 31(3), 685-695.
- Kaur, R. (2021). Review on machine learning algorithms. *International Journal of Engineering Applied Sciences and Technology*, 5(6), 64-68.
- Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, 30.
- La Nación. (2023, mayo). Una foto satelital deja claras las dimensiones descomunales de un “cementerio de ropa” que se formó en Atacama [Accedido: 2025-09-03].
- Lacy, P., & Rutqvist, J. (2015). *Waste to Wealth: The Circular Economy Advantage*. Palgrave Macmillan.
- Lorenzo-Navarro, J., Castrillón, M., Ramón, E., & Freire, D. (2014). Evaluation of LBP and HOG descriptors for clothing attribute description. *International Workshop on Video Analytics for Audience Measurement in Retail and Digital Signage*, 53-65.
- Loy, J. (2019). *Neural Network Projects with Python: The Ultimate Guide to Using Python to Explore the True Power of Neural Networks Through Six Projects* (1st) [Accessed: 2025-09-03]. Packt Publishing. <https://www.oreilly.com/library/view/neural-network-projects/9781789138900/>
- Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.
- Mammone, A., Turchi, M., & Cristianini, N. (2009). Support vector machines. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(3), 283-289. <https://doi.org/10.1002/wics.41>

- Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.
- Mizunuma, C., Ikeya, T., Kurosaki, H., Okumura, N., Kami, Y., & Hanada, M. (2024). A Construction of the Data Set and Background Removal Identifier for Classification of Used Clothes. *Procedia Computer Science*, 246, 2062-2070.
- Nauman, F. (2024). Clothing Dataset for Second-Hand Fashion (Version 3) [Data set]. <https://doi.org/10.5281/zenodo.13788681>
- Ojala, T., Pietikäinen, M., & Harwood, D. (1996). A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29(1), 51-59.
- Pietikäinen, M., Hadid, A., Zhao, G., & Ahonen, T. (2011). *Computer Vision Using Local Binary Patterns*. Springer.
- Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: unbiased boosting with categorical features. *Advances in Neural Information Processing Systems*, 31.
- Schwartz, R., Dodge, J., Smith, N. A., & Etzioni, O. (2019). Green AI. *Communications of the ACM*, 63(12), 54-63. <https://doi.org/10.1145/3381831>
- Seçkin, M., Seçkin, A. Ç., 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.
- Sen, A., Maiti, G., Parida, B. K., Mishra, B. P., Arya, M., & Bondar, D. I. (2025). Feature Engineering is Not Dead: Reviving Classical Machine Learning with Entropy, HOG, and LBP Feature Fusion for Image Classification. *arXiv preprint arXiv:2507.13772*.
- Sheskin, D. J. (2011). *Handbook of Parametric and Nonparametric Statistical Procedures, Fifth Edition* (5th). Chapman; Hall/CRC. <https://doi.org/10.1201/9780429186196>
- Simeone, O. (2018). A very brief introduction to machine learning with applications to communication systems. *IEEE Transactions on Cognitive Communications and Networking*, 4(4), 648-664.
- Strubell, E., Ganesh, A., & McCallum, A. (2019). Energy and Policy Considerations for Deep Learning in NLP. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 3645-3650. <https://doi.org/10.18653/v1/P19-1355>
- Szeliski, R. (2022). *Computer vision: algorithms and applications*. Springer Nature.
- ThredUp. (2024). *2024 Resale Report*. ThredUp. Consultado el 10 de agosto de 2025, desde https://cf-assets-tup.thredup.com/resale_report/2024/ThredUp_2024_Report.pdf
- Trevisan, V. (2022, 117). *Using SHAP Values to Explain How Your Machine Learning Model Works* [Imagen por el autor]. <https://towardsdatascience.com/>

- using-shap-values-to-explain-how-your-machine-learning-model-works-732b3f40e137
- Van Loon, W., Fokkema, M., Szabo, B., & De Rooij, M. (2024). View selection in multi-view stacking: choosing the meta-learner. *Advances in Data Analysis and Classification*, 1-39.
- Wargön Innovation AB, RISE Research Institutes of Sweden AB & Myrorna AB. (2024, octubre). *Final Report: AI for Resource-Efficient Circular Fashion* (inf. téc.). Wargön Innovation AB. <https://wargoninnovation.se/content/uploads/sites/3/2024/10/final-project-report-w-all-apps.pdf>
- Weimer, J., Pufahl, K., Jagodzinski, M., Cisowski, A., Kanngiesser, B., & Oberschmidt, D. (2024). AI-Based System for Enhanced Textile Recycling and Reuse. *Global Conference on Sustainable Manufacturing*, 626-634.
- Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, 5(2), 241-259. [https://doi.org/10.1016/S0893-6080\(05\)80023-1](https://doi.org/10.1016/S0893-6080(05)80023-1)

Apéndices

Apéndice A

Variables

A.1 Lista de variables

Lista exhaustiva de las 44 variables generadas con la metadata con un resumen de su descripción y del preprocesamiento realizado en la etapa de limpieza de datos.

Tabla A.1: Listas de las variables de la metadata con su descripción y resumen de su preprocesamiento.

#	Variable	Tipo	Descripción
1	station	I	Estación desde la que se tomó la fotografía de la prenda (station1, station2, station3). Se conservó tal cual.
2	month	I	Mes y año en que se tomó la fotografía de la prenda en formato (p. ej., apr2023, may2023). Se generó la columna month_dt como datetime para orden temporal y se mantuvo month para reconstruir rutas de archivo.
3	item_id	ID	Identificador único temporal de la captura de la prenda, basado en la fecha y hora exactas en formato (p. ej. 2023_04_04_11_37_25). Se creó item_station_id concatenando item_id y station para garantizar un identificador.

Continúa en la siguiente página

#	Variable	Tipo	Descripción
4	front	R	Ruta relativa a la imagen frontal de la prenda. Presentaba 100 valores faltantes (0,31 %), que fueron imputados con el placeholder “missing_image.jpg” y renombrada a front_path.
5	back	R	Ruta relativa a la imagen posterior de la prenda. Presentaba 100 valores faltantes (0,31 %), que fueron imputados con el placeholder “missing_image.jpg” y renombrada a back_path.
6	brand	R	Ruta relativa a la imagen posterior de la prenda. Presentaba 5,290 valores faltantes (16,5 %), que fueron imputados con el placeholder “missing_image.jpg” y renombrada a back_path.
7	labels	R	Ruta al archivo JSON con las anotaciones detalladas de cada prenda. Se mantuvo para cargar los JSON asociados, pero no se utilizó como variable predictora y se descartó antes del modelado.
8	gold	I	Indica si la prenda forma parte del “gold dataset” (booleano True/False). Se usó junto con la ausencia o presencia de imágenes para crear una nueva variable categórica gold_garment -categorías: not_gold, gold_with_images, gold_no_images-, para identificar las 100 prendas gold sin imagen, posteriormente se eliminó gold.
9	brand.1	C	Etiqueta de marca asignada de un listado predefinido. Contiene placeholders, puntuación o mayúsculas inconsistentes. Se normalizó con técnicas de NLP y luego se integró en la variable brand_clean, eliminando brand.1.
10	brand.text	C	Texto libre de la marca. Se limpió de la misma forma que brand.1 y se utilizó para completar brand_clean, posteriormente se eliminó.
11	category	C	Categoría de la prenda según público objetivo (Ladies, Men, Children, Unisex). Se limpió el texto con técnicas de NLP (minúsculas, eliminación de espacios y corrección de alias) y se generó category_clean, conservando únicamente las cuatro clases principales.

Continúa en la siguiente página

#	Variable	Tipo	Descripción
12	type	C	Tipo de prenda (p. ej. Blazer, Dress, T-shirt). Se normalizó con técnicas de NLP (eliminación de espacios, minúsculas y corrección de alias tipográficos) usando un listado de categorías válidas, generando type_clean y descartando la columna original.
13	size	C	Talla de la prenda (XS, S, M, L, XL, XXL, etc.). Se limpió texto (minúsculas y eliminación de espacios), se unificaron variantes tipográficas, se llenaron los NaN y valores como “none” con “Unknown” y se generó size_clean, descartando la columna original.
14	colors	C	Colores asociados a la prenda (lista de strings o string). Se convirtieron todas las entradas a listas de colores, se eliminaron los valores nulos, se aplicó limpieza de texto, se reemplazaron listas vacías por Unknowns se aplanaron las listas de un solo color a string, generando colors_clean y descartando la columna original.
15	season	C	Estación del año de la prenda (Spring, Summer, Autumn, Winter). Se llenaron los NaN con “Unknown”, se normalizó texto (eliminación de espacios y capitalización) y se generó season_clean, descartando la columna original.
16	pilling	C	Grado de formación de bolitas (“pilling”) en la superficie de la prenda (escala ISO 1–5). Se transformó a string.
17	condition	C	Estado general de la prenda en escala ISO 1–5, donde 1 representa muy mal estado y 5 excelente. Se convirtió a string.
18	price	C	Valor de venta de la prenda en moneda Sueca (SEK) por rangos (p.ej. <50, 50-100, 100-150). Se corrigieron rangos erróneos.
19	annotator	I	Identificador del anotador que etiquetó cada prenda. No aporta valor predictivo y se elimina del conjunto antes del modelado.

Continúa en la siguiente página

#	Variable	Tipo	Descripción
20	cut	C	Tipo de corte de la prenda (p. ej. Slim, Regular, Oversize). Se convirtieron las entradas a listas de strings, se normalizó texto, se reemplazaron listas vacías por “Unknown” y se aplanaron listas de un solo elemento, generando cut_clean, se eliminó la original.
21	pattern	C	Tipo de estampado de la prenda (p. ej. Floral print, Striped, Polka dot, Solid). Se llenaron NaN y “none” con “Unknown”, se limpiaron espacios y se capitalizó, generando pattern_clean y eliminando la columna original.
22	trends	C	Tendencia o estilo de la prenda (p. ej. 70s, 80s, Denim, Sports). Se renombraron NaN como “Unknown” y “none” como “No trend”, se limpiaron espacios y se capitalizó para crear trend_clean, eliminando la columna original.
23	smell	C	Severidad de olor residual en la prenda (None, Minor, Major). Se llenaron los NaN con “Unknown” y se creó smell_clean, eliminando la columna original.
24	stains	C	Grado de presencia de manchas en la prenda (Major, Minor, None). Se llenaron los NaN con “Unknown”, se limpiaron espacios y se capitalizó el texto, se unificaron sinónimos y se creó stains_clean, eliminando la columna original.
25	holes	C	Gravedad de agujeros en la prenda (None, Minor, Major). Se llenaron NaN y valores “none” con “Unknown”, se normalizó texto y se generó holes_clean, eliminando la columna original.
26	damage	C	Descripción del grado de daño físico en la superficie de la prenda. No aportó valor predictivo y se eliminó antes del modelado.
27	materials	C	Materiales que componen la prenda (p. ej. “100 % Cotton”, “Polyester Blend”, “Wool”). Se dejó sin procesar debido a la complejidad de su normalización y se conservó tal cual para posible tratamiento posterior.

Continúa en la siguiente página

#	Variable	Tipo	Descripción
28	comments	C	Observaciones libres de los anotadores. Presentaba diversidad textual y alto porcentaje de NaN, por lo que se eliminó antes del modelado.
29	usage	C	Destino de la prenda al final de su ciclo de vida (Reuse, Repair, Remake, Recycle, Export, Energy recovery). Se limpiaron espacios, se aplicó capitalización uniforme y se corrigieron variantes tipográficas, generando usage_clean y eliminando la columna original.
30	brand_missing	I	Indicador de ausencia de marca (True/False), con un alto porcentaje de NaN. No aportó valor predictivo y se eliminó antes del modelado.
31	sizetext	C	Texto libre de la talla asignada (e.g., “Small for the size L, Test 11”), con alta diversidad (y predominio de “missing”). No aportó valor estructural y se eliminó antes del modelado.
32	usage_confidence	A	Nivel de confianza asignado al etiquetado de usage (Low, Medium, High). Se empleó como variable predictora y se eliminó antes del modelado.
33	price_confidence	A	Nivel de confianza en el valor registrado de price (Low, Medium, High). Presentó gran proporción de NaN y no aportó señal, por lo que fue eliminado.
34	description	A	Texto libre con anotaciones sobre la prenda (e.g., “t-shirt med tryck”). Alta diversidad y muchos NaN, se descartó antes de la fase de modelado.
35	styleholes	C	Indicador booleano de presencia de agujeros estilísticos (True/False). No aportó valor predictivo y se eliminó del dataset.
36	damageimage	A	Ruta de la imagen donde se detectó el primer daño. Se eliminó por alto porcentaje de NaN y falta de uso predictivo.
37	damageloc	A	Ubicación del primer daño en la prenda. Se eliminó.
38	damage2image	A	Ruta de la segunda imagen de daño. Se eliminó.
39	damage2loc	A	Ubicación del segundo daño. Se eliminó
40	damage3image	A	Ruta de la tercera imagen de daño. Se eliminó.
41	damage3loc	A	Ubicación del tercer daño. Se eliminó.

Continúa en la siguiente página

#	Variable	Tipo	Descripción
42	damage2	A	Descripción textual del segundo daño. Se eliminó.
43	damage3	A	Descripción textual del tercer daño. Se eliminó.
44	weight	C	Peso de la prenda en gramos. No aportó señal consistente para el modelado y se eliminó.

Nota: Los tipos de variables se identifican como A = anotación, C = característica prenda, I = informativa, ID = identificador y R = ruta de archivo.

A.2 Imagen ejemplo por categoría original de *usage*

Para cada categoría original de *usage* se presenta un ejemplo aleatorio de sus imágenes: frontal, posterior y de la marca. Se evidencia la falta imágenes de marca para ciertas prendas.



Figura A.1: Ejemplo de prenda etiquetada como *Reuse*.

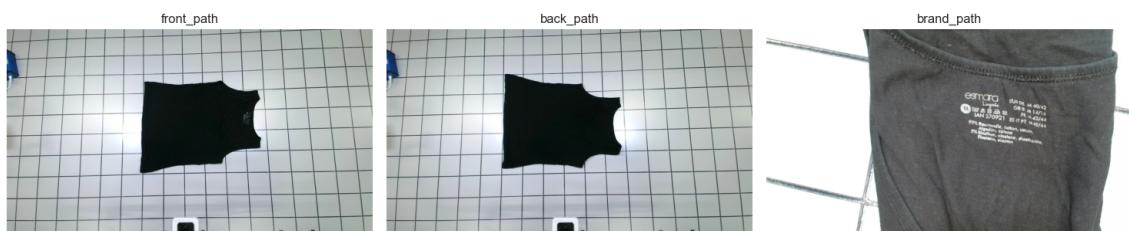


Figura A.2: Ejemplo de prenda etiquetada como *Export*.



Figura A.3: Ejemplo de prenda etiquetada como *Recycle*.



Figura A.4: Ejemplo de prenda etiquetada como *Remake*.



Figura A.5: Ejemplo de prenda etiquetada como *Repair*.



Figura A.6: Ejemplo de prenda etiquetada como *Energy Recovery*.

Apéndice B

Sub-modelos visuales

B.1 Entrenamiento de sub-modelos con categorías de variables originales

En la sección 4.2.7, se introdujo la hipótesis de que el entrenamiento de sub-modelos visuales, para cada k -*features*, tendría un mejor rendimiento si la variable objetivo contara con sus categorías reducidas. A modo de prueba, se realizó el experimento de entrenar los sub-modelos con sus categorías originales, validando la hipótesis inicial.

A continuación, se presenta las métricas de evaluación de rendimiento de cada sub-modelo, el reporte de clasificación y la matriz de confusión del modelo con mejor rendimiento predictivo.

Tabla B.1: Métricas de evaluación sub-modelos con sus categorías originales en validación.

Sub-modelo	Algoritmo	F1-Score Macro	Recall Macro
condition	Random Forest	37.49 %	32.06 %
	SVM	27.28 %	31.57 %
	XGBoost	31.07 %	31.83 %
price	Random Forest	40.22 %	41.35 %
	SVM	34.70 %	40.30 %
	XGBoost	38.37 %	39.29 %
pilling	Random Forest	27.83 %	27.85 %
	SVM	23.19 %	30.60 %
	XGBoost	27.65 %	27.50 %

Tabla B.2: Reporte de clasificación Random Forest para *condition* con categorías originales en validación.

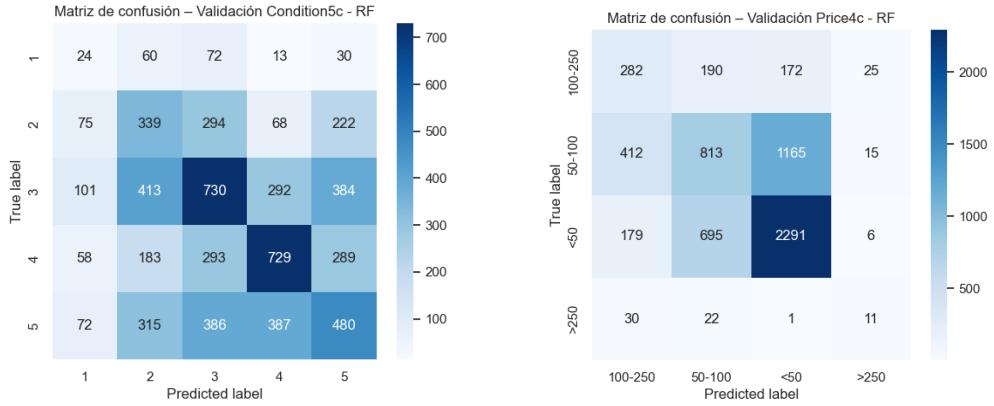
Clase	Precision	Recall	F1-score	Support
1	0.07	0.12	0.09	199
2	0.26	0.34	0.29	998
3	0.41	0.38	0.40	1920
4	0.49	0.47	0.48	1552
5	0.34	0.29	0.32	1640
accuracy			0.36	6309
macro avg	0.31	0.32	0.31	6309
weighted avg	0.38	0.36	0.37	6309

Tabla B.3: Reporte de clasificación Random Forest para *price* con categorías originales en validación.

Clase	Precision	Recall	F1-score	Support
100-250	0.31	0.42	0.36	669
50-100	0.47	0.34	0.39	2405
<50	0.63	0.72	0.67	3171
>250	0.19	0.17	0.18	64
accuracy			0.54	6309
macro avg	0.40	0.41	0.40	6309
weighted avg	0.53	0.54	0.53	6309

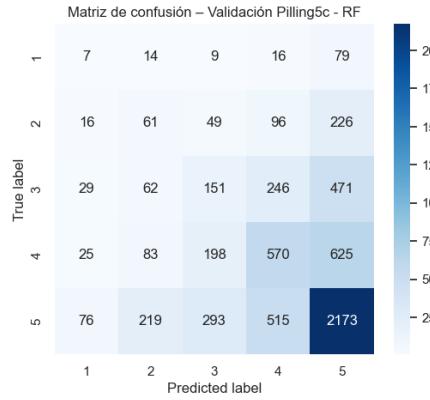
Tabla B.4: Reporte de clasificación Random Forest para *pilling* con categorías originales en validación.

Clase	Precision	Recall	F1-score	Support
1	0.05	0.06	0.05	125
2	0.14	0.14	0.14	448
3	0.22	0.16	0.18	959
4	0.40	0.38	0.39	1501
5	0.61	0.66	0.63	3276
accuracy			0.47	6309
macro avg	0.28	0.28	0.28	6309
weighted avg	0.45	0.47	0.46	6309



(a) Random Forest para *condition* categorías originales.

(b) Random Forest para *price* categorías originales.



(c) Random Forest para *pilling* categorías originales.

Figura B.1: Matriz de confusión de Random Forest para *condition*, *price* y *pilling*, con sus categorías originales en validación.

B.2 Entrenamiento de sub-modelo visual para *usage*

Al contar con la descripción visual de cada prenda -obtenida a partir del descriptor LBP-U- y con la variable objetivo *usage*, se realizó el experimento de entrenar un sub-modelo que clasificará directamente la variable. Para ello, se seleccionaron los algoritmos de Random Forest y XGBoost -presentaron el mejor rendimiento para clasificar las *k-features*-, se configuraron reutilizando la grilla de búsqueda de hiperparámetros de los sub-modelos y se entrenaron sobre los descriptores visuales en el conjuntos de entrenamiento y validación (*split_train*).

En validación sobre *usage* utilizando únicamente los descriptores LBP-U, XGBoost obtuvo un F1 Macro de 47,33 % y un Recall Macro de 46,73 %, mientras que Random Forest alcanzó un F1 Macro de 47,25 % y un Recall Macro de 50,10 % (Tabla B.5). Estos resultados superan el rendimiento del *meta-learner* puro (Escenario 4), que registró un F1 Macro de 41,52 % y un Recall Macro de 47,65 %, pero quedan por

debajo de la cota ideal establecida por el modelo base inicial (Escenario 1), con 65,07 % y 66,24 %, respectivamente. Aunque el sub-modelo directo para *usage* mostró un mejor desempeño que el ensamblado, este carece de interpretabilidad de variables (p.ej. saber que una prenda se recicla por que su *condition* y *pilling* son bajos), por otro lado, la cota de referencia la marca el modelo base con *metadata* real, por lo que seguir optimizando los sub-modelos de cada *k-feature*, para mejorar las predicciones que alimentan al *meta-learner*, es fundamental para que el *meta-learner* alcance un rendimiento predictivo más cercano a los modelos con *metadata* real.

En el reporte de clasificación (tabla B.6) y en la matriz de confusión (Figura B.2), XGBoost mostró un buen desempeño en las clases *Reuse* y *Export*, con un Recall de 0,60 y 0,58 respectivamente, situándose de forma competitiva frente al *meta-learner* pero aún por debajo del Recall alcanzado por el modelo base. Por su parte, *Recycle* fue la clase con mayor confusión, obteniendo un Recall de tan solo 0,27, inferior tanto al rendimiento del *meta-learner* como al del modelo base.

Tabla B.5: Métricas de evaluación sub-modelo *usage* en validación.

Sub-modelo	Algoritmo	F1-Score Macro	Recall Macro
<i>usage</i>	XGBoost	47.33 %	46.73 %
	Random Forest	47.25 %	50.10 %

Tabla B.6: Reporte de clasificación XGBoost para *usage* en validación.

Clase	Precision	Recall	F1-score	Support
0: Reuse	0.78	0.60	0.68	3832
1: Export	0.48	0.58	0.53	2029
2: Recycle	0.16	0.32	0.21	448
accuracy			0.58	6309
macro avg	0.47	0.50	0.47	6309
weighted avg	0.64	0.58	0.60	6309

Tabla B.7: Reporte de clasificación Random Forest para *usage* en validación.

Clase	Precision	Recall	F1-score	Support
0: Reuse	0.78	0.60	0.68	3832
1: Export	0.47	0.62	0.54	2029
2: Recycle	0.17	0.27	0.21	448
accuracy			0.58	6309
macro avg	0.47	0.49	0.47	6309
weighted avg	0.64	0.58	0.60	6309

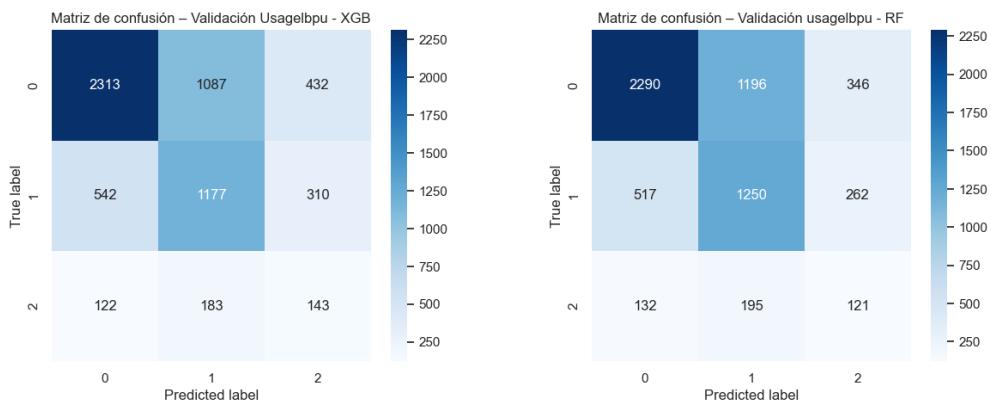


Figura B.2: Matriz de confusión de Random Forest y XGBoost para *usage* en validación.