Trabajo de Fin de Grado

Desarrollo de un Sistema de Tasación de Vehículos de Ocasión para Domingo Alonso Usando Datos del Mercado Automotriz y Modelos de Precios Hedónicos en Databricks

TITULACIÓN: Grado en Ciencia e Ingeniería de Datos

AUTOR: Jesús María Matos Torres

TUTORIZADO POR:
Juan María Hernández Guerra y Cristian Munteanu Tatomir

Fecha 06/2025

ULPGC Universidad de Las Palmas de Gran Canaria | EII ESCUELA DE INGENIERÍA INFORMÁTICA

TFT04

SOLICITUD DE DEFENSA DE TRABAJO DE FIN DE TÍTULO

D./Dª Jesús María Matos Torres, autor del trabajo Desarrollo de un Sistema de Tasación de Vehículos de Ocasión para Domingo Alonso Usando Datos del Mercado Automotriz y Modelos de Precios Hedónicos en Databricks, correspondiente a la titulación Grado en Ciencia e Ingeniería de Datos, en colaboración con la empresa/proyecto Domingo Alonso Group.

SOLICITA

que se inicie el procedimiento de defensa del mismo, para lo que se adjunta la documentación requerida, haciendo constar que

[x] se autoriza / [ ] no se autoriza la grabación en audio de la exposición y turno de preguntas.

Asimismo, con respecto al registro de la propiedad intelectual/industrial del TFT, declara que:

[x] Se ha iniciado o hay intención de iniciarlo (defensa no pública).

[ ] No está previsto.

Y para que así conste firma la presente. (fecha en firma electrónica)

El/La estudiante

Fdo._____

A rellenar y firmar **obligatoriamente** por el/la/los/las tutores

En relación con la presente solicitud, se informa:(firmar donde corresponda)

Positivamente (en caso de detección de copia, esta firma quedará invalidada)	Negativamente (justificación en TFT05)
Fdo._____	Fdo._____

DIRECTOR DE LA ESCUELA DE INGENIERÍA INFORMÁTICA

Agradecimientos

Gracias a Domingo Alonso Group por darme la oportunidad de trabajar en un proyecto real que me ha hecho crecer tanto profesional como personalmente. En especial, al equipo de DTO, por el buen ambiente, la confianza y todo lo aprendido con ustedes. A mi familia y a mi pareja, por estar ahí cuando más lo necesitaba, con su apoyo incondicional en los momentos clave. Y a los amigos que me ha regalado esta etapa, esa familia que uno elige, por estar en cada paso, por las risas, los Ca'Pako, los San Remo, los asaderos en Bañaderos y por todo lo que aún nos queda por compartir.

Resumen

Este Trabajo de Fin de Título, en colaboración con Domingo Alonso, tiene como objetivo mejorar la tasación de vehículos mediante ciencia de datos. La creciente demanda de valoraciones automáticas y precisas justifica la necesidad de este proyecto. Se ha diseñado un sistema inteligente de estimación de precios usando CatBoost como modelo de regresión, entrenado con datos históricos. El desarrollo se realizó en Databricks, integrando flujos ETL, análisis exploratorio, entrenamiento, explicación y despliegue del modelo como API. El modelo de precios hedónicos generado estima valores con base en características técnicas, mercado actual y disponibilidad. El sistema ha sido desplegado y está listo para su uso, aportando valor a empresas que requieran tasaciones más fiables y eficientes.

Abstract

This Final Degree Project, in collaboration with Domingo Alonso, aims to improve vehicle valuation through data science. The growing demand for automatic and accurate valuations justifies the need for this project. An intelligent price estimation system has been designed using CatBoost as a regression model, trained on historical data. The development was done in Databricks, integrating ETL flows, exploratory analysis, training, explanation and deployment of the model as an API. The generated hedonic pricing model estimates values based on technical characteristics, current market and availability. The system has been deployed and is ready for use, bringing value to companies requiring more reliable and efficient valuations.

Índice general

Índice de Algoritmos

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Este capítulo introduce el contexto, los fundamentos conceptuales y la motivación que sustenta el desarrollo de un sistema de tasación automatizada de vehículos de ocasión (VO) para Domingo Alonso Group, aplicando técnicas de ciencia de datos sobre datos del mercado automotriz y modelos de precios hedónicos. La solución se ha construido en la plataforma Databricks, abarcando todas las fases del ciclo de vida del dato, desde la adquisición y limpieza hasta el despliegue en producción. El objetivo último es ofrecer un sistema robusto, interpretable y alineado con los procesos de negocio de la empresa.

1.1. Contexto

En los últimos años, el mercado de VO ha experimentado un crecimiento sostenido, impulsado por factores como la reducción de costes respecto al vehículo nuevo, la escasez de stock en concesionarios, y una mayor aceptación social de la segunda mano como opción de movilidad. Esta tendencia ha consolidado el interés de empresas del sector por soluciones que permitan valorar de forma más eficiente y objetiva los vehículos disponibles en el mercado.

Domingo Alonso Group, grupo empresarial de referencia en el sector de la automoción en Canarias y distribuidor oficial de marcas como Volkswagen, Audi, Škoda o Hyundai, ha identificado una necesidad estratégica en la digitalización del proceso de tasación. Tradicionalmente basado en criterios manuales o tablas genéricas, este proceso adolece de problemas de subjetividad, falta de trazabilidad y dificultad para adaptarse a la realidad cambiante del mercado.

Con el objetivo de ofrecer estimaciones de precio rápidas, replicables y ajustadas a las condiciones del mercado canario, se propone la construcción de un sistema predictivo basado en modelos de precios hedónicos. Esta solución se integra en el marco de transformación digital de la organización y está diseñada para ser consumida como un servicio a través de APIs internas, permitiendo su conexión con herramientas corporativas y su futura exposición en plataformas web. El sistema ha sido nombrado internamente como **VOLTA**, denominación

que agrupa tanto el modelo de inferencia como los distintos flujos de procesamiento, almacenamiento y publicación de datos que lo sustentan. Las tablas, rutas y catálogos utilizados en su desarrollo incorporan el identificador `123`, correspondiente al código interno asignado por la organización a este proyecto.

1.2. Fundamentos conceptuales

La construcción del sistema requiere la integración de diversos conceptos clave:

- **Vehículo de ocasión (VO):** Unidad previamente matriculada y con al menos un propietario anterior, que vuelve al mercado mediante canales formales (concesionarios) o informales (portales de compraventa). Su valoración depende de variables como marca, modelo, año de matriculación, potencia, kilometraje, segmento, tipo de transmisión y demanda.
- **Tasación:** Estimación del valor económico de un VO basada en sus características. En un entorno profesional, requiere precisión, objetividad y coherencia interdepartamental. La automatización del proceso permite eliminar sesgos, acelerar decisiones y facilitar la integración en sistemas empresariales.
- **Modelos de precios hedónicos:** Esta técnica se fundamenta en el trabajo seminal de Rosen [11], quien propuso descomponer el valor de un bien en función de sus atributos individuales. Aunque el modelo hedónico ha sido ampliamente utilizado, su aplicación al mercado de vehículos de ocasión presenta retos adicionales de segmentación y heterogeneidad, como se analiza en trabajos como el de Purohit et al. [10], centrados en el impacto de factores como la marca, la potencia o la antigüedad en el precio final. En este trabajo, se aplica sobre un conjunto de datos enriquecido con información del mercado y de la empresa, lo que permite construir un modelo predictivo capaz de estimar el precio final como una combinación ponderada de las características del vehículo.
- **Ciencia de datos aplicada a negocio:** Conjunto de técnicas de análisis, modelado, validación y despliegue orientadas a resolver problemas reales con impacto operativo. El sistema desarrollado incluye flujos automatizados de *scraping*, arquitectura *medallion* para el tratamiento del dato, generación de variables contextuales, validación cruzada, interpretabilidad mediante SHAP y publicación del modelo como servicio inferencial en Databricks.
- **Plataforma Databricks [3] y enfoque MLOps:** El desarrollo se ejecuta íntegramente en Databricks, lo que permite unificar las fases del ciclo de vida del modelo. Se han aplicado principios de MLOps [1] para garantizar versionado, reproducibilidad, automatización y trazabilidad. El modelo se despliega como un endpoint REST con control de versiones mediante MLflow y planificación mensual de reentrenamiento.

1.3. Motivación

Desde una perspectiva estratégica, la capacidad de valorar VO de forma rápida, fiable y estandarizada constituye una ventaja competitiva clave para empresas del sector. La automatización de este proceso permite:

- Reducir los tiempos de tasación y aumentar la eficiencia operativa.
- Eliminar sesgos subjetivos derivados del juicio humano.
- Establecer criterios homogéneos para múltiples marcas y modelos.
- Aumentar la transparencia y la trazabilidad de las decisiones de precio.
- Integrar valoraciones automáticas en flujos existentes como compras, ventas, leasing o *remarketing*.

Desde el punto de vista académico y formativo, este trabajo permite aplicar de forma integral conocimientos adquiridos durante el Grado en Ciencia e Ingeniería de Datos, como la ingeniería de datos, el aprendizaje automático, la interpretación de modelos, el diseño de pipelines de datos y el despliegue de soluciones en la nube. Además, se emplea una metodología iterativa basada en CRISP-DM [2] y TDSP, con separación entre entornos de desarrollo y producción, y control de versiones mediante Azure DevOps.

Por último, desde una óptica tecnológica, el proyecto demuestra cómo una solución basada en datos puede recorrer de forma automatizada todas las fases del ciclo MLOps: desde la captura diaria de información del mercado automovilístico hasta la publicación y consumo del modelo en una interfaz corporativa. El enfoque modular y escalable adoptado abre la posibilidad de extender la solución a otros productos, marcas o regiones en fases futuras.

Capítulo 2

Contexto del proyecto y objetivos iniciales

Este capítulo describe el contexto específico y las motivaciones que dieron origen al desarrollo del sistema de tasación de vehículos de ocasión, así como los objetivos iniciales definidos en la propuesta del trabajo. Finalmente, se comentan las principales desviaciones respecto al plan original, justificadas por la evolución técnica del proyecto.

2.1. Motivación y antecedentes

La estimación del valor de vehículos de ocasión representa un proceso estratégico para las empresas del sector automotriz, con implicaciones directas en la gestión de inventario, la negociación comercial y la transparencia frente al cliente. A pesar de su relevancia, esta tarea continúa realizándose, en muchas organizaciones, mediante métodos manuales o herramientas genéricas, frecuentemente desalineadas con las características específicas del mercado local.

Domingo Alonso Group, en su apuesta por la digitalización y la automatización de procesos, identificó la necesidad de desarrollar un sistema interno de tasación automática que ofreciera estimaciones objetivas, reproducibles y adaptadas al contexto regional. El enfoque propuesto combina técnicas de ciencia de datos con modelos de precios hedónicos, permitiendo descomponer el valor de un vehículo en función de sus atributos técnicos y comerciales.

Aunque la literatura recoge múltiples aplicaciones de estos modelos en contextos inmobiliarios o financieros, su aplicación al mercado de vehículos de segunda mano conlleva desafíos particulares, entre ellos la heterogeneidad de los datos disponibles, la variabilidad del mercado y la presencia de segmentos con escasa representación.

El sistema se desarrolla íntegramente sobre la plataforma Databricks, lo que permite unificar el tratamiento de datos, la experimentación con modelos y su despliegue como servicio. Asimismo, se adopta un enfoque metodológico basado en buenas prácticas inspiradas en el

ciclo de vida de proyectos CRISP-DM y TDSP, garantizando una estructura robusta, trazabilidad y orientación a MLOps, aunque sin aplicar estos marcos de forma formal o completa.

2.2. Objetivos

Los objetivos iniciales definidos en la propuesta del Trabajo de Fin de Título se estructuraron en torno a seis ejes fundamentales:

- **Extracción y tratamiento de datos:** Implementar un flujo ETL automatizado capaz de recopilar datos procedentes de fuentes heterogéneas, normalizarlos y estructurarlos para su uso en tareas analíticas.
- **Análisis de calidad y visualización:** Desarrollar herramientas de análisis exploratorio y calidad de datos que permitan detectar anomalías, evaluar la representatividad de los registros y generar visualizaciones útiles para el diagnóstico continuo.
- **Modelado predictivo:** Construir un modelo de regresión basado en precios hedónicos que estime el valor de un vehículo de ocasión a partir de sus características relevantes (kilometraje, marca, edad, segmento, etc.).
- **Automatización mediante MLOps:** Diseñar un flujo MLOps que integre entrenamiento, validación y registro del modelo, garantizando la reproducibilidad y el control de versiones mediante MLflow [7].
- **Despliegue como servicio:** Publicar el modelo como un endpoint de inferencia accesible mediante API REST, con el objetivo de facilitar su integración con otros sistemas internos de la empresa.
- **Supervisión y mejora continua:** Establecer mecanismos de monitorización del modelo y, en su caso, alertas automáticas ante degradaciones del rendimiento, así como preparar la infraestructura para posibles tareas de reentrenamiento periódico.

Si bien todos los objetivos planteados han sido abordados, el desarrollo del sistema ha requerido priorizar ciertas tareas frente a otras, debido a la complejidad de los datos, las restricciones operativas y el enfoque exploratorio adoptado. En particular, se ha puesto un mayor énfasis en la evaluación y mejora de la calidad del dato, el análisis exploratorio (EDA) y la validación del modelo predictivo, mientras que la implementación de mecanismos avanzados de monitorización o generación de alertas automáticas se ha pospuesto como línea de trabajo futura. Esta priorización técnica ha reforzado la robustez del sistema y su preparación para futuras extensiones en entornos productivos.

Capítulo 3

Aportaciones del trabajo

Este capítulo recoge las principales contribuciones del Trabajo de Fin de Grado tanto en el plano técnico como en su aplicabilidad empresarial y formativa. Para contextualizar estas aportaciones, cabe destacar que el desarrollo del sistema ha seguido una estructura por fases claramente diferenciadas: análisis del problema y comprensión del dominio; diseño de la arquitectura y del flujo de datos; desarrollo iterativo del sistema mediante ciclos sucesivos de mejora; y evaluación rigurosa de los resultados obtenidos, tanto en términos de precisión del modelo como de estabilidad operativa del sistema. Todo el proceso se ha apoyado en una separación de entornos *dev* y *prod*, permitiendo una validación segura antes del despliegue. Asimismo, se ha hecho uso de herramientas profesionales como Azure DevOps [6] para el versionado y trazabilidad del código, MLflow para la gestión del ciclo de vida del modelo, y PostgreSQL [8] como destino final de los datos integrados para su consumo desde sistemas externos.

3.1. Principales aportaciones

3.1.1. Aportaciones técnicas

Desde un punto de vista técnico, este trabajo ha supuesto el diseño, desarrollo e implementación de un sistema completo de tasación automática de vehículos de ocasión, aplicando un enfoque basado en ciencia de datos y modelos de precios hedónicos. Entre las principales aportaciones destacan:

- **Automatización integral del flujo de valor**: Se ha construido un sistema de orquestación que cubre todas las etapas del ciclo de vida del modelo, desde el scraping de datos hasta su despliegue como endpoint inferencial. Todo el pipeline opera de forma programada y desacoplada, permitiendo un funcionamiento robusto y trazable.
- **Implementación de lógica MLOps sobre Databricks**: Se ha desarrollado un flujo completo de entrenamiento, validación, registro y despliegue automático del modelo

mediante MLflow. Cada vez que el modelo se registra con el mismo nombre, se crea una nueva versión superior, manteniendo el histórico completo en el catálogo interno.

- **Ingeniería de variables orientada a la estabilidad del modelo**: Se ha incorporado la variable *Shortage* como mecanismo corrector para mejorar el comportamiento del modelo en segmentos poco representados. Asimismo, se ha desarrollado *PriceMed* como referencia externa de mercado para aportar contexto en la predicción.
- **Sistema de scraping tolerante a fallos**: Las tareas de obtención de datos implementan un sistema de reintento con un máximo de 4 intentos y 2 minutos de espera entre fallos, lo que aporta robustez frente a interrupciones temporales en las fuentes web.
- **Sistema de alertas operativas**: Aunque el modelo no incluye mecanismos de reentrenamiento automático, el flujo de ejecución está monitorizado mediante alertas automáticas por correo en caso de fallo, asegurando la supervisión continua de su funcionamiento.
- **Portabilidad parcial del sistema**: Si bien el entrenamiento y despliegue están diseñados para ejecutarse en Databricks, el código es portable a otros entornos si se mantienen las versiones utilizadas. Sin embargo, el uso de MLflow Serving y el control de versiones del modelo dependen de funcionalidades específicas del ecosistema Databricks.

3.1.2. Aportaciones socioeconómicas

Desde el punto de vista empresarial, el sistema de tasación automático contribuye directamente a la digitalización de los procesos comerciales de Domingo Alonso Group. Las principales aportaciones en este ámbito son:

- **Reducción del tiempo de tasación y estandarización del proceso**: La automatización elimina la dependencia de valoraciones manuales sujetas a criterios individuales, lo que aporta coherencia, transparencia y reproducibilidad a las estimaciones.
- **Aplicabilidad directa en entornos reales**: Aunque no ha sido necesaria una validación formal por parte de otros departamentos, el sistema está previsto para integrarse directamente en la web corporativa como herramienta de uso interno para estimaciones de valor.
- **Infraestructura alineada con la organización**: La solución aprovecha el ecosistema tecnológico ya existente en la empresa, como Azure DevOps y Databricks, garantizando su sostenibilidad operativa y facilidad de mantenimiento.
- **Potencial de escalado**: Si bien actualmente se ha diseñado para el mercado canario, han existido conversaciones informales sobre la posibilidad de extender el sistema a otras marcas o regiones, dado su diseño modular y flexible.

3.2. Competencias específicas

Durante el desarrollo del presente trabajo se han cubierto de forma directa varias de las competencias específicas del Grado en Ciencia e Ingeniería de Datos. A continuación se detallan las más relevantes y cómo se han desarrollado en el contexto del TFG:

- **ED3 – Técnicas de aprendizaje computacional y extracción de conocimiento a partir de grandes volúmenes de datos**: Se ha entrenado un modelo de regresión sobre datos históricos del mercado, integrando tareas de limpieza, transformación y análisis exploratorio, aplicando técnicas de validación cruzada y análisis de interpretabilidad como SHAP.
- **ED4 – Diseño e implementación de soluciones software en el ámbito de la inteligencia artificial**: El sistema desarrollado implementa de forma modular todos los componentes de una solución de ML en producción, incluyendo scraping, procesamiento, entrenamiento, inferencia, validación e integración por API, respetando buenas prácticas de ingeniería y trazabilidad.
- **ED6 – Análisis, predicción y prospectiva de grandes volúmenes de datos**: La complejidad y heterogeneidad de las fuentes, así como el tamaño de los datasets tratados, ha requerido una gestión eficiente y una arquitectura medallion basada en Delta Lake para garantizar consistencia y escalabilidad.
- **ED9 – Inteligencia de negocios y actuaciones basadas en datos**: La solución implementada transforma información histórica en conocimiento útil y procesable por los equipos de tasación y remarketing, permitiendo tomar decisiones basadas en criterios objetivos y replicables. Todo el código se encuentra versionado y controlado mediante Azure DevOps, donde el proyecto está identificado por el prefijo interno “123”.

3.3. Alineamiento con los Objetivos de Desarrollo Sostenible

El proyecto presenta un alineamiento claro con los siguientes Objetivos de Desarrollo Sostenible (ODS) definidos por la Agenda 2030:

- **ODS 8 – Trabajo decente y crecimiento económico (Grado de relación: Alto)** El sistema permite automatizar una tarea repetitiva y propensa a errores, mejorando la eficiencia de los flujos internos y permitiendo que el personal se centre en tareas de mayor valor añadido.
- **ODS 9 – Industria, innovación e infraestructuras (Grado de relación: Medio)** La incorporación de modelos predictivos en procesos tradicionales del sector automotriz impulsa la digitalización empresarial, la innovación operativa y el aprovechamiento de infraestructuras tecnológicas modernas.

Aunque el trabajo no se enmarca directamente en otros ODS, su enfoque en eficiencia, automatización y uso ético de los datos puede considerarse una contribución indirecta a la sostenibilidad tecnológica y empresarial. No se han identificado impactos ambientales directos ni efectos colaterales significativos que condicionen su alineamiento con otros ODS como el 12, dada la neutralidad operativa del sistema respecto al consumo o producción.

Cuadro 3.1: Grado de relación del TFT con los objetivos de desarrollo sostenible.

ODS	Grado de relación			
	0 No procede	1 Bajo	2 Medio	3 Alto
1 Fin de la Pobreza	0			
2 Hambre cero	0			
3 Salud y Bienestar	0			
4 Educación de calidad	0			
5 Igualdad de género	0			
6 Agua limpia y saneamiento	0			
7 Energía Asequible y no contaminante	0			
8 Trabajo decente y crecimiento económico				3
9 Industria, Innovación e Infraestructuras			2	
10 Reducción de las desigualdades	0			
11 Ciudades y comunidades sostenibles	0			
12 Producción y consumo sostenibles	0			
13 Acción por el clima	0			
14 Vida submarina	0			
15 Vida de ecosistemas terrestres	0			
16 Paz, justicia e instituciones sólidas	0			
17 Alianzas para lograr objetivos	0			

3.4. Síntesis final del capítulo

En conjunto, este trabajo ha supuesto una aportación significativa tanto en el plano técnico como en el organizativo, desarrollando una solución de tasación automática alineada con las necesidades reales de la empresa y con las competencias formativas del Grado. Las herramientas, metodologías y decisiones tomadas durante su desarrollo reflejan un dominio riguroso de los principios de ciencia de datos aplicada. La estrategia metodológica adoptada, basada en fases claramente estructuradas y un enfoque iterativo, ha permitido evolucionar el sistema de forma controlada y efectiva, garantizando la robustez del modelo y su integración en un entorno empresarial real. El uso de herramientas como MLflow, Azure DevOps y PostgreSQL refuerza la profesionalización del desarrollo, asegurando la trazabilidad, el versionado y la futura mantenibilidad y escalabilidad del sistema en producción.

Capítulo 4

Desarrollo

La fase de desarrollo constituye el núcleo técnico y práctico del presente Trabajo de Fin de Título. En ella se materializan las decisiones metodológicas, se ejecutan las tareas planificadas, se enfrentan las dificultades técnicas y se construye la solución final que da respuesta al problema planteado: la tasación automatizada de vehículos de ocasión mediante un modelo de precios hedónicos, adaptado al mercado específico del archipiélago canario. Este capítulo se estructura en tres secciones principales: una introducción general al proceso de desarrollo, una descripción detallada de la metodología adoptada y una revisión pormenorizada de las fases ejecutadas.

El proyecto se ha abordado mediante un enfoque iterativo-incremental, fundamentado en las metodologías CRISP-DM y TDSP, y ejecutado sobre la plataforma Databricks. Si bien se mantuvo la estructura de fases y tareas establecida en el plan de trabajo inicial, la naturaleza exploratoria del análisis de datos condujo a una reevaluación constante de las decisiones técnicas y a la incorporación de mejoras que no estaban contempladas inicialmente. El uso de una arquitectura medallion permitió trabajar con distintas capas de calidad del dato (bronze, silver y golden), facilitando así una doble limpieza y análisis exploratorio (EDA) que resultó esencial para entender la naturaleza y la calidad de los datos disponibles.

El desarrollo no se ejecutó de forma estrictamente lineal. Al contrario, las sucesivas exploraciones estadísticas, la detección de variables altamente correlacionadas y la introducción de nuevas variables derivadas, como la variable escasez, impulsaron múltiples iteraciones del modelo. Esta variable, clave para corregir sesgos asociados a marcas poco representadas, implicó reentrenamientos y reevaluaciones que exigieron flexibilidad en el ciclo de desarrollo. Asimismo, el uso de técnicas como el VIF (Variance Inflation Factor), pruebas de hipótesis geográficas, análisis de precios medios del mercado y visualizaciones avanzadas mediante dashboards de Databricks, permitieron una comprensión profunda del comportamiento del modelo y sus limitaciones.

Desde el punto de vista arquitectónico, se introdujeron soluciones técnicas significativas, como la separación entre flujos ETL y pipelines de MLOps, y la implementación de mecanismos de control de versiones con MLflow. La carga computacional del modelo y la dependencia de versiones concretas de librerías, como Selenium, condicionaron la frecuencia de ejecución

de los flujos, lo cual obligó a redefinir la cadencia del entrenamiento mensual del modelo en contraste con la actualización diaria del flujo ETL. Además, se optó por mantener en la API no solo la predicción del precio, sino también visualizaciones del contexto de mercado, lo que ofrece un valor añadido a los usuarios empresariales.

Las interacciones con Domingo Alonso Group fueron constantes a través de reuniones semanales con el tutor técnico de la empresa, quien permitió una libertad considerable en el diseño del sistema. Esta autonomía fue crucial para introducir innovaciones como el ajuste por escasez o la capacidad del modelo para tasar vehículos aún inexistentes, extrapolando su valor a partir de características proyectadas. No obstante, estas decisiones se tomaron con cautela y tras un análisis riguroso, a fin de evitar interpretaciones erróneas del modelo en contextos empresariales sensibles.

Durante el desarrollo surgieron diversas dificultades, entre las que destacan limitaciones técnicas impuestas por la infraestructura de Databricks, problemas de rendimiento en el entrenamiento y validación del modelo, y la necesidad de balancear precisión e interpretabilidad. Algunas decisiones previstas inicialmente, como la utilización exclusiva de variables técnicas del vehículo, fueron descartadas tras evidenciarse que el comportamiento del mercado no podía modelarse adecuadamente sin variables externas como el precio mediano del mercado o la escasez relativa de un vehículo.

Finalmente, antes del despliegue definitivo, se evaluaron múltiples configuraciones del modelo mediante el sistema de experimentación de Databricks. Esto permitió contrastar métricas como R^2, MAE o la amplitud de los intervalos de confianza entre variantes del modelo con y sin ciertas variables. Solo tras garantizar un rendimiento óptimo y una justificación estadística sólida, se procedió al despliegue de la solución. Aun así, el sistema fue reentrenado posteriormente tras detectar intervalos de confianza excesivamente amplios en determinadas configuraciones, lo que demuestra la naturaleza viva y evaluativa de este desarrollo.

Este capítulo recoge, por tanto, una documentación exhaustiva del proceso de construcción de la solución, y se presenta como el eje articulador de los capítulos previos donde se establecieron las motivaciones, objetivos y aportaciones, y los posteriores, donde se evaluarán los resultados alcanzados.

4.1. Metodología

El desarrollo del sistema de tasación de vehículos de ocasión se ha fundamentado en la aplicación combinada de las metodologías **CRISP-DM** (Cross Industry Standard Process for Data Mining) y **TDSP** (Team Data Science Process). Esta elección no fue arbitraria, sino el resultado de una evaluación sobre la naturaleza del problema, los recursos disponibles y los objetivos técnicos y de negocio perseguidos por Domingo Alonso Group. Frente a enfoques más rígidos, como las metodologías en cascada, o más centrados en la gestión ágil de productos, como SCRUM, el modelo CRISP-DM proporciona una estructura clara y secuencial para el desarrollo analítico, mientras que TDSP introduce buenas prácticas propias de la ingeniería del software, necesarias para la operación estable y mantenible de soluciones de

machine learning.

Esta sinergia metodológica permitió no solo una definición ordenada de las fases del proyecto, sino también la flexibilidad suficiente para volver atrás cuando el análisis exploratorio de los datos (EDA) o los resultados del modelado revelaban la necesidad de ajustes. Esta capacidad de iteración fue especialmente valiosa en momentos clave como la redefinición de las variables predictoras, la *feature engineering* o la introducción de componentes correctores como la variable `escasez`, que surgió tras observar comportamientos anómalos en la predicción para marcas con escasa representación en el conjunto de datos.

Desde el punto de vista técnico, CRISP-DM estructuró el flujo en seis fases que se adaptaron al proyecto de la siguiente forma:

- **Comprensión del negocio:** Se analizaron los requisitos operativos de la empresa y se definió el objetivo del sistema de tasación, centrado en proporcionar un valor justo para un vehículo de ocasión, junto con un intervalo de confianza fiable. Este sistema debía ser integrable como API y utilizable tanto por personal técnico como comercial.
- **Comprensión de los datos:** Se evaluaron y documentaron fuentes heterogéneas, como páginas web de compraventa (*km0*, *AutoScout*, etc.) y bases de datos históricas internas de la empresa. Se identificaron atributos clave como marca, modelo, cilindrada, potencia, año de matriculación, y se comprendieron sesgos implícitos en los datos canarios, como la revalorización de ciertos vehículos por su antigüedad o rareza.
- **Preparación de los datos:** Se implementó una arquitectura *medallion* en Databricks (Bronze, Silver, Golden), donde los datos se transformaban progresivamente. Cada sitio web generaba inicialmente su propia tabla en Bronze, que luego se consolidaba en una única tabla Silver. Tras un primer EDA, se realizó una depuración profunda para conformar una tabla Golden compatible con los requisitos del modelado. Esta fase incluyó la generación de variables nuevas, como `preciomed` (precio mediano de referencia) y `escasez`.
- **Modelado:** Se testearon múltiples modelos, entre ellos regresión lineal, *random forest*, *gradient boosting*, *XGBoost* y *CatBoost*. Se aplicó validación cruzada (*k-fold*) y análisis de multicolinealidad mediante VIF, descartando variables que no aportaban valor explicativo o que distorsionaban las predicciones. Finalmente, se optó por un modelo *CatBoost* [9] optimizado que equilibraba interpretabilidad, precisión y eficiencia.
- **Evaluación:** La evaluación se realizó tanto con métricas estándar (MAE, RMSE, R^2) como mediante simulaciones reales y tests de robustez. Gracias a los mecanismos de experimentación de Databricks, se compararon distintas configuraciones de modelos y se almacenaron versiones diferenciadas con sus métricas, visualizaciones SHAP y análisis de importancia de características.
- **Despliegue:** Se desarrolló un flujo MLOps independiente del ETL, que ejecuta el pipeline completo: preparación, entrenamiento, validación, despliegue y generación de una tabla con el *z-score* ajustado por marca para construir el intervalo de confianza. El modelo se publica como *serving endpoint* en Databricks y es accedido por una API

externa, que a su vez permite visualizar EDAs resumidos y testear casos hipotéticos, incluso para vehículos aún no comercializados.

Paralelamente, los principios del TDSP se reflejaron en:

- **Control de versiones y trazabilidad:** A través de MLflow se registraron todos los experimentos, sus métricas, visualizaciones y configuraciones, lo que permitió comparar versiones y justificar decisiones técnicas.
- **Automatización mediante pipelines:** Se desarrollaron dos flujos bien diferenciados —ETL y MLOps— ejecutables de forma programada. El flujo ETL corre diariamente para actualizar la información de mercado, mientras que el MLOps se ejecuta mensualmente, dada su mayor demanda de recursos computacionales.
- **Gestión colaborativa:** La metodología fue enriquecida por las interacciones frecuentes con los tutores del proyecto. En concreto, se mantuvieron dos reuniones semanales con el tutor de empresa, lo que permitió un acompañamiento técnico constante y alineado con los objetivos del entorno productivo. A su vez, las reuniones con el tutor académico aportaron una perspectiva metodológica y estadística clave para validar las decisiones tomadas. Este doble acompañamiento favoreció la detección de problemas no triviales y la elección razonada de soluciones como la variable `escasez`.
- **Entornos de desarrollo y producción diferenciados:** Se establecieron entornos separados para pruebas (`ws_dag_dev`) y despliegue (`ws_dag_pro`), con control de errores y *logs* detallados por flujo y por fuente de datos. En el caso de la extracción web, se utilizó Selenium [12] con versiones específicas (por ejemplo, 3.141.0) y controladores gestionados dinámicamente mediante `webdriver-manager`, lo que obligó a trabajar con clusters 13.3 para asegurar compatibilidad.

Desde el punto de vista tecnológico, se emplearon numerosas librerías y herramientas: `Pandas`, `NumPy`, `Seaborn`, `Matplotlib`, `Scikit-learn`, `Statsmodels`, `PySpark`, `CatBoost`, `XGBoost`, `SHAP`, `SQLAlchemy`, `Selenium`, `MLflow`, `Delta Lake` y los recursos propios de Databricks. Los clusters utilizados impusieron restricciones importantes de compatibilidad, especialmente entre versiones de librerías y versiones de clúster, obligando a diseñar una arquitectura dual: los procesos ETL trabajan con clúster 13.3, mientras que los procesos MLOps se ejecutan sobre clúster 16.1. Esta decisión evitó conflictos con el acceso y la modificación de tablas Delta de diferentes generaciones.

Finalmente, se aplicaron buenas prácticas como *logging* estructurado por tarea y fuente, manejo robusto de excepciones, separación clara entre flujos y entornos, validación continua del rendimiento del modelo, y pruebas sobre datos reales y simulados. Cada fase estuvo guiada por criterios rigurosos de calidad tanto del dato como del modelo, asegurando así que el sistema final no solo fuera preciso, sino también interpretable, reproducible y escalable.

4.2. Fases de desarrollo

4.2.1. Comprensión del negocio

El desarrollo del sistema de tasación de vehículos de ocasión se enmarca en una necesidad estratégica identificada por Domingo Alonso Group: disponer de una herramienta objetiva y automatizada que permita conocer el estado del mercado de vehículos de segunda mano en tiempo real, y estimar el valor de un vehículo ajustado a sus características individuales. Esta funcionalidad, además del cálculo de un intervalo de confianza asociado a la predicción, responde a la necesidad de dotar a los departamentos del grupo de un recurso técnico que complemente sus procesos comerciales y operativos. Aunque no se definió explícitamente una única unidad organizativa destinataria, es razonable asumir que esta solución tiene aplicación directa en áreas como tasación, compras, ventas, postventa y remarketing.

La motivación estratégica detrás del proyecto incluye aspectos como la mejora en la eficiencia de procesos, la automatización de tareas repetitivas, la reducción de sesgos humanos en la valoración, y la obtención de conocimiento accionable sobre el comportamiento del mercado. Todo ello con el objetivo final de mejorar la toma de decisiones operativas y tácticas a través de una solución integrada en el ecosistema digital de la compañía.

Previo al desarrollo de este sistema, la tasación se realizaba de manera manual o mediante metodologías sujetas a la subjetividad del tasador y a criterios internos no normalizados. Esta dependencia de criterios individuales suponía una limitación en términos de coherencia, trazabilidad y transparencia, y dificultaba el análisis agregado de datos a nivel grupo. Aunque no se dispone de un sistema formal previo como referencia directa, se deduce que los métodos empleados anteriormente no ofrecían mecanismos de verificación, ajuste o justificación técnica del valor asignado a cada vehículo.

Desde el inicio del proyecto se establecieron una serie de requisitos mínimos que debía cumplir el sistema: acceso a datos actualizados del mercado en tiempo real; estimación precisa del precio de un vehículo en función de sus características (marca, modelo, año, CV, km, entre otras); y la posibilidad de obtener un intervalo de confianza que cuantificara la fiabilidad de cada estimación. Además, el sistema debía estar disponible a través de una API que serviría como capa de alimentación para una API de presentación independiente, encargada de conectarse con la interfaz web final utilizada por las empresas del grupo.

Cabe destacar que, si bien no se exigió interpretabilidad explícita del modelo, sí se planteó la necesidad de que el sistema ofreciera resultados consistentes y comparables. En este sentido, se optó por incluir mecanismos de evaluación visual como gráficas resumen, visualizaciones SHAP y trazabilidad de datos dentro de la propia plataforma Databricks. La cadencia de actualización también fue un aspecto importante: el flujo ETL debía operar diariamente para reflejar cambios en el mercado, mientras que el flujo MLOps, inicialmente previsto como semanal, fue reajustado a ejecución mensual debido a limitaciones computacionales y de presupuesto.

En cuanto al proceso de desarrollo y la supervisión del mismo, se establecieron reuniones

estructuradas con ambos tutores. Con el tutor de empresa se llevaron a cabo dos sesiones semanales de carácter progresivamente más técnico, que comenzaron enfocándose en el negocio y la interpretación de datos, y evolucionaron hacia la toma de decisiones arquitectónicas. Por su parte, el tutor académico centró su apoyo en los aspectos estadísticos y de validación. En una de estas sesiones se puso de manifiesto la necesidad de considerar un ajuste que tuviera en cuenta la frecuencia relativa de ciertas marcas y modelos dentro del conjunto de datos, lo que impulsó el desarrollo de una variable que permitiese corregir el sesgo de representación. Este doble acompañamiento favoreció la coherencia metodológica del proyecto y aseguró una visión equilibrada entre los objetivos técnicos y académicos.

Las limitaciones del proyecto estuvieron marcadas principalmente por el uso de la plataforma Databricks, que si bien supuso una ventaja operativa por su integración de herramientas, impuso restricciones relacionadas con la compatibilidad de versiones y el scraping de ciertas páginas como *milanuncios.com* y *coches.net*, las cuales bloqueaban el acceso desde IPs reconocidas como automatizadas. Asimismo, el presupuesto disponible condicionó la frecuencia de ejecución del flujo MLOps, que finalmente se estableció en una periodicidad mensual.

El criterio de éxito del proyecto fue definido como la capacidad del sistema para ofrecer resultados fiables, automatizados y adecuados para su integración en la interfaz web interna de la empresa. Aunque no se contempló su escalado a otras marcas o entornos externos, sí se planteó desde un principio su integración en los sistemas ya utilizados por el grupo para la emisión de informes internos, lo que refuerza la dimensión práctica y aplicable del sistema desarrollado.

4.2.2. Comprensión y adquisición de los datos

La recopilación y estructuración de datos fue una de las fases más complejas y determinantes del proyecto. Para construir un sistema de tasación robusto, era esencial disponer de una base de datos rica, actualizada y representativa del mercado de vehículos de ocasión. Esta fase combinó múltiples fuentes de datos —tanto externas como internas— integradas mediante una arquitectura de tipo *medallion* (Bronze, Silver, Golden) en la plataforma Databricks.

Fuentes de datos externas

Se desarrollaron tareas de *web scraping* específicas para seis plataformas clave del mercado automovilístico: AutoScout, CarMarket, Carplus, Km0Task, MLeonTask y MotorEsTask. Cada una de estas fuentes aportaba datos estructurados en tablas individuales del entorno Bronze, con campos como marca, modelo, características, kilometraje, cilindrada, tipo de caja y combustible, precio, provincia, fechas de publicación y extracción, entre otros.

Debido a la diversidad estructural de las webs, fue necesario diseñar un scraper independiente para cada una, aunque todos se sustentaban en una base común reutilizable (scripts SetUpScraper y LoggerHelper). Se utilizó **Selenium 3.141.0** junto a **Firefox ESR 91** y **geckodriver** personalizados en el entorno Databricks, con librerías adicionales co-

mo `sqlalchemy`, `bs4`, `unidecode`, `fake_useragent`, entre otras. El scraping se ejecutaba en clústeres 13.3 LTS debido a restricciones de compatibilidad con estas librerías.

A pesar de los esfuerzos, páginas como *milanuncios.com* y *coches.net* resultaron inaccesibles desde Databricks debido al bloqueo de IPs automatizadas. Aunque se exploró la posibilidad de ejecutar los scrapers de forma local o en máquinas virtuales de Azure, las limitaciones de tiempo, seguridad y mantenimiento descartaron esa opción. El volumen disponible tras el filtrado asciende, a fecha de firma de este TFG, a más de 2.600 registros válidos, distribuidos por fuente según la siguiente tabla:

- AutoScout: 743
- CarMarket: 760
- Carplus: 85
- Km0: 325
- MLeon: 270
- MotorES: 572

Cabe destacar que esta cifra se actualiza diariamente mediante el flujo ETL automatizado, por lo que está sujeta a un crecimiento continuo.

Fuentes de datos internas

Domingo Alonso Group proporcionó un histórico estructurado que abarcaba desde mediados de 2020 hasta finales de 2024. Aunque el formato de los datos era coherente y compatible con los scrapers desarrollados (incluyendo columnas como marca, modelo, precio, fecha de matriculación, CV, etc.), se detectaron inconsistencias como fechas imposibles, kilometrajes extremos o campos vacíos. El tratamiento de estos datos fue esencial para garantizar la validez de los registros y se integraron completamente en la arquitectura Delta.

4.2.3. Preparación y limpieza de los datos

La calidad de los datos es el cimiento sobre el que se construyen los modelos predictivos. En este proyecto, dicha calidad no fue asumida, sino que se alcanzó mediante una combinación de decisiones arquitectónicas, lógica de negocio y técnicas de limpieza avanzadas. Este apartado expone con detalle el proceso de preparación, estandarización y validación de los datos, así como la motivación de cada una de las variables construidas.

Arquitectura Medallion y proceso de fusión

La arquitectura *medallion* [4] utilizada en este proyecto organiza los datos en tres niveles que representan distintos grados de refinamiento y propósito específico. Esta estructura

sigue la convención establecida por la empresa, permitiendo separar claramente las fases de ingestión, limpieza y preparación para el modelado:

- **Bronze**: contiene los datos brutos extraídos de diversas fuentes como AutoScout, CarMarket, Km0 y otras. Cada tabla corresponde a una web distinta y conserva la información tal como fue obtenida, sin limpieza ni transformaciones. Los datos se almacenan con el prefijo `catalogo.bronze.tb_123_`. A esta capa llegan tanto los anuncios web como las listas de precios de vehículos nuevos, esta última obtenida mediante *web scraping* automatizado y tratada en el componente `NewVehiclePrice`.
- **Silver**: en esta capa, mediante la tarea `Bronze_To_Silver`, se combinan todas las tablas Bronze en una única tabla unificada y transformada. Durante este proceso se aplican validaciones de consistencia, eliminaciones de duplicados y transformaciones semánticas para inferir información como el segmento o la familia del modelo. Asimismo, se llevan a cabo estandarizaciones tipográficas y normalizaciones de campos categóricos, unificando variantes como `VW` → `VOLKSWAGEN` o `CLASE A` → `CLASEA`. Esta capa se almacena en `catalog.silver.tb_123_volta_etl_mlops_markets_canary`.
- **Golden**: corresponde a los datos ya procesados, depurados y enriquecidos, listos para el entrenamiento del modelo. Esta capa se genera a través del flujo MLOps mediante la tarea `patternsbuilder`, donde se crean variables clave como `PriceMed` y `Shortage`, se consolidan los campos finales y se aplican reglas de limpieza exhaustivas según las restricciones definidas. Los datos se almacenan en `ds_123_volta_etl_mlops_001_trainingpatterns`.

Esta arquitectura facilita la trazabilidad y el versionado de los datos, permitiendo mantener registros históricos y garantizar la coherencia de las transformaciones realizadas. El proceso de fusión y limpieza es fundamental para asegurar la calidad del dataset final, eliminando errores semánticos, normalizando nomenclaturas y garantizando la consistencia entre registros. Además, se generan identificadores únicos tanto por vehículo como por registro, lo que facilita posteriores procesos analíticos y de modelado.

Evaluación inicial de la calidad de los datos

La tabla Silver resultante constaba de más de 115.000 registros únicos y 21 columnas. El análisis exploratorio inicial (EDA) evidenció varios problemas:

- **Valores extremos y errores semánticos:** vehículos con más de 1.000.000 km, fechas de matriculación en 2048, edades negativas (hasta -25 años), entre otros.
- **Ausencia de datos:** variables como `Cilindrada_CV` con un 19.6 % de valores nulos, `Segment` con 10.7 %, y `Venta` con un 3.3 %.
- **Variables categóricas con alta cardinalidad:** más de 900 modelos, 18.000 acabados y más de 35.000 combinaciones de características.

Estas observaciones fueron apoyadas con visualizaciones específicas:

- **Figura 4.1:** Histograma de precios — distribución sesgada positivamente, con concentración entre 5.000€ y 30.000€.
- **Figura 4.2:** Boxplot de precios — valores atípicos detectados a partir de 50.000€.
- **Figura 4.3:** Distribución de kilometraje — valores extremos por encima de 200.000 km.
- **Figura 4.4:** Matriz de correlación — `CV` correlaciona 0.76 con el `precio`, `Antigüedad` con `Matriculación` (-0.94) y `Kms` con `Antigüedad` (0.74).
- **Tabla 4.5:** VIF — valores superiores a 10 en `Precio`, `CV` y `Matriculación`.
- **Figura 4.6:** Comparativa geográfica — diferencia significativa de precios entre `LAS PALMAS` y `S/C TENERIFE` (*p-value ¡0.05*).

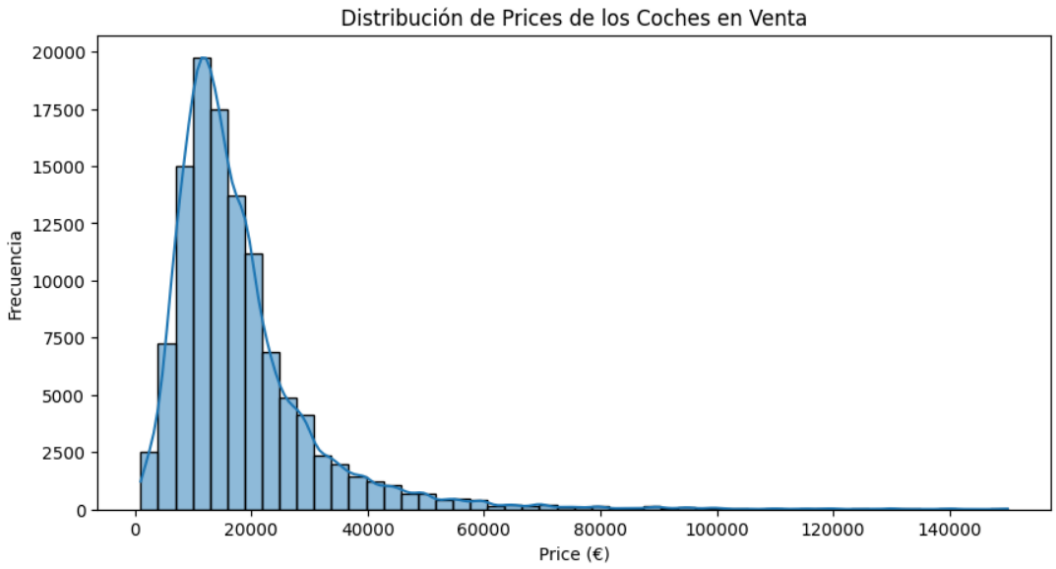

Ilustración 4.1: Histograma de precios.

Ilustración 4.2: Boxplot de precios.

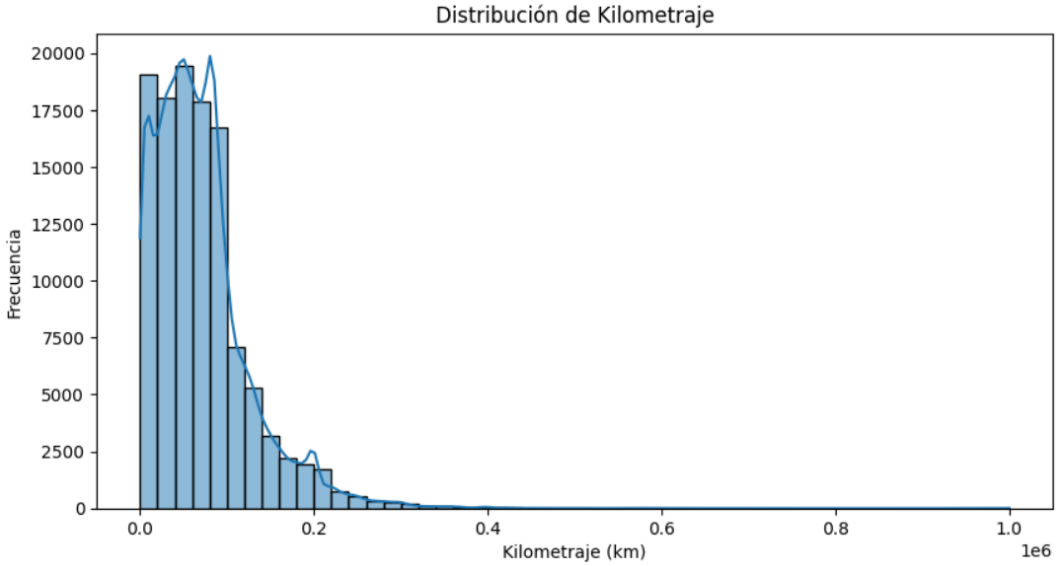

Ilustración 4.3: Distribución de kilometraje.

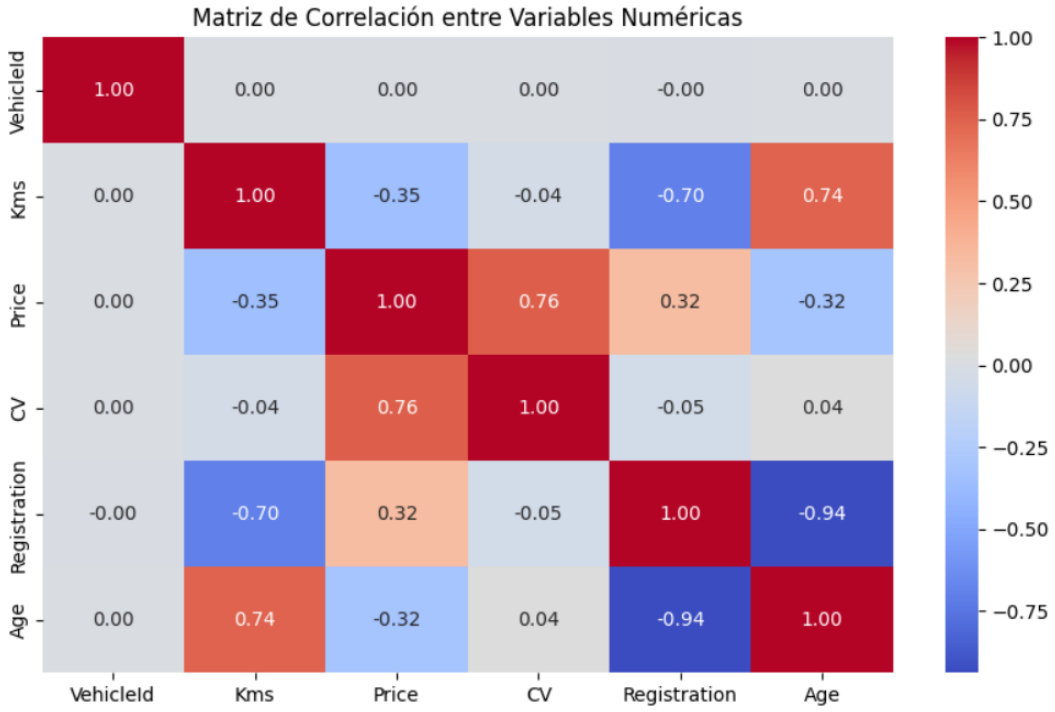

Ilustración 4.4: Matriz de correlación.

Variable	**VIF**
VehicleId	4.000670
Kms	6.296310
Price	10.044777
CV	14.435252
Registration	10.365704
Age	6.643581

Ilustración 4.5: VIF de las variables numéricas.

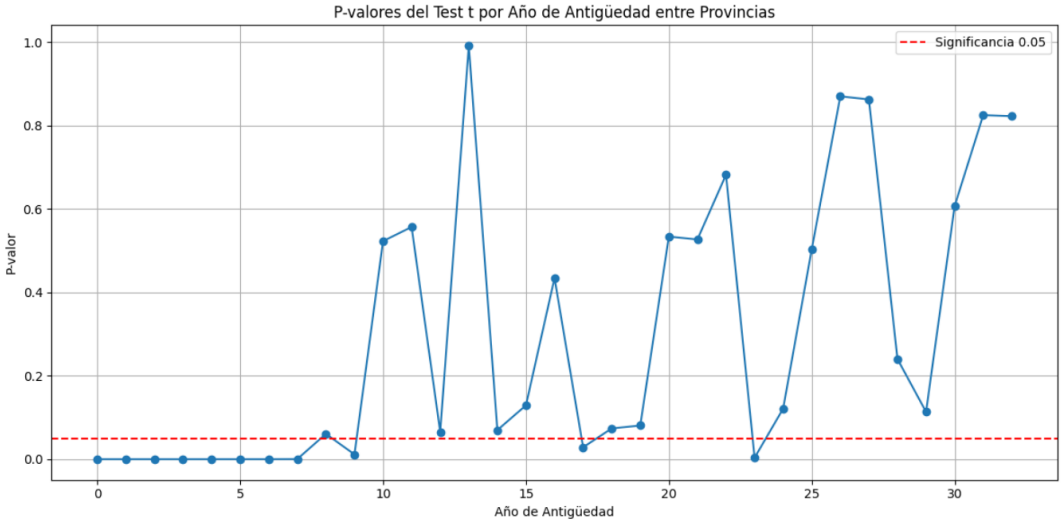

Ilustración 4.6: Comparativa geográfica.

Decisiones tomadas

Con base en este análisis:

- Se limitaron los precios entre 1.000€ y 150.000€, y el kilometraje a menos de 1.000.000 km.
- Se descartó la variable `Matriculación` en favor de `Antigüedad`, dada su redundancia.
- Se imputaron segmentos faltantes mediante `merge` con la tabla de precios de referencia.
- Se preservó la granularidad de variables como `Modelo` y `Finish`, con la intención de evaluarlas posteriormente en el modelado.

Este trabajo de adquisición y comprensión de datos fue fundamental para las siguientes fases del proyecto. No solo permitió disponer de un dataset amplio y fiable, sino que sirvió como base sólida para el análisis exploratorio, la ingeniería de variables y el entrenamiento de modelos, que se detallarán en las próximas secciones.

Proceso de limpieza y filtrado

La transición hacia la capa Golden implica un filtrado riguroso. Se establecieron umbrales y reglas para descartar registros claramente erróneos o extremos:

- Eliminación de registros con kilometraje mayor a 300.000 km, cilindrada superior a 650 CV o antigüedad negativa.
- Filtrado de registros con valores nulos en variables clave (`Segment`, `CV`, `Fuel`, etc.).
- Descartado de anuncios de fuentes irrelevantes o inconsistentes como `COMPRAMOSTUCOCHE.ES` o `CICAR`.

No se aplicó imputación de valores nulos: los registros incompletos fueron eliminados por considerarse inservibles para la modelización fiable.

Flujo ETL y orquestación del pipeline

El flujo de extracción, transformación y carga (ETL) que abastece de datos al sistema de modelado está implementado en Databricks como un *Job* denominado `flow_123_ETL_volta`. Este flujo se encarga de orquestar el scraping, la transformación inicial de los datos, la estructuración en tablas medallion y su posterior exportación a PlatinumZone. La ejecución está programada automáticamente cada día a las **04:00 AM (UTC+01:00)** mediante un *trigger* de programación definido en el propio panel del job.

El pipeline consta de las siguientes etapas, organizadas como tareas secuenciales:

- **Get_Tasks**: lee la lista de webs a scrapear desde la tarea `ScraperList`.
- **Scraper_iteration**: realiza de forma iterativa el scraping de cada URL activa, almacenando los resultados en bruto (sin limpieza) en tablas `Bronze`, siguiendo la arquitectura *medallion*. Esta tarea incluye múltiples pasos de descarga asincrónica con Selenium.
- **Bronze_To_Silver**: transforma los datos desde su versión bruta a un formato estructurado y limpio en la tabla `Silver`, aplicando validaciones, normalizaciones tipográficas, detección de duplicados y limpieza semántica de campos como *brand*, *gearbox* o *fuel*.
- **VOLTA_to_Pz**: exporta las tablas clave del modelo (`Silver` y `confidence_intervals`) a una base de datos PostgreSQL externa denominada `PlatinumZone`, mediante una conexión segura `JDBC`. Esta operación permite su disponibilidad para otras áreas de negocio, herramientas de inteligencia empresarial y su integración directa en la aplicación web final junto con el modelo de tasación.

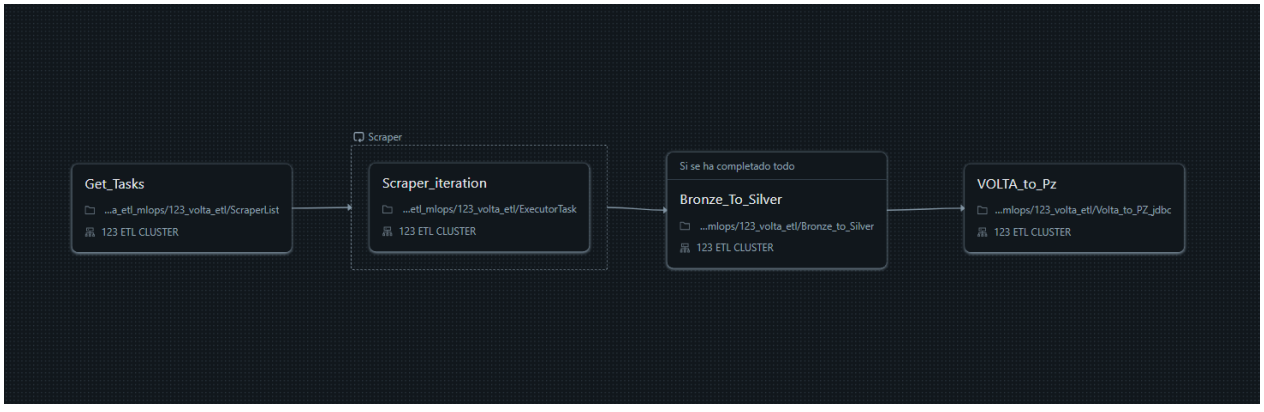

Ilustración 4.7: Flujo de orquestación diaria para el scraping y tratamiento de datos.

La ejecución se realiza sobre un **clúster persistente** denominado `123_ETL_CLUSTER`, configurado con nodos `Standard_D4ds_v5`, que permite hasta 2 workers y un entorno Spark 3.5.0 (Scala 2.12). Esta elección responde a dos motivos fundamentales:

- No se emplea un *job cluster* desechable debido a la necesidad de instalar de forma persistente herramientas de scraping como `geckodriver`, `firefox` y dependencias relacionadas con Selenium, las cuales requieren privilegios elevados y no se pueden autoinstalar en entornos efímeros.
- El flujo debe ejecutarse bajo un **usuario interactivo**, y no como usuario de sistema, ya que este último no posee permisos suficientes para el control de navegador requerido por Selenium. Además, se ha verificado que la ejecución directa desde el repositorio en Azure DevOps (vía CI/CD) es inviable, precisamente por estas limitaciones de entorno y permisos.

Estandarización y normalización

Durante la preparación, se aplicó una estandarización intensiva de campos categóricos:

- Eliminación de tildes, espacios, guiones y caracteres especiales mediante `unidecode`, expresiones regulares y funciones personalizadas.
- Unificación de variantes como "ALFAROMEO" → "ALFA ROMEO".
- Homogeneización de formatos de fecha (`StarDate`, `EndDate`) y completado con el año actual si estaba ausente.
- Agrupación de categorías minoritarias en campos como `Segment` o `Brand` bajo etiquetas genéricas como `OtroSegmento` u `OTROS`.

Ingeniería de variables

Se desarrollaron variables fundamentales para mejorar la explicabilidad y robustez del modelo:

- **Age**: diferencia entre el año actual y el de matriculación (`StarDate`), normalizada entre 0 y 20.
- **PriceMed**: generada por el módulo `LevelPriceTask`, se basa en la **mediana de precios de vehículos nuevos** equivalentes en el mercado español. No representa la media ni la mediana de precios de vehículos de ocasión, sino el valor mediano estimado de un vehículo nuevo con las mismas características (marca, modelo, versión). Esta variable aporta contexto de mercado actual y permite detectar desviaciones significativas entre el precio real de un vehículo usado y su referencia de mercado. Como se observa en la Figura 4.8, existen diferencias sistemáticas entre el precio real de ciertos vehículos y el valor mediano estimado por mercado según la marca, lo que demuestra que `PriceMed` no solo actúa como referencia estática, sino que revela sesgos estructurales que condicionan el valor final. Esta observación fue clave para motivar, más adelante, la inclusión de la variable `Shortage`.

- **Shortage (Escasez)**: definida como la inversa de la frecuencia relativa del par `Brand_Model` en los datos. Esta variable fue incorporada tras detectar, mediante un mapa de calor de distribución de modelos (Figura 4.9), una alta disparidad entre vehículos sobrerrepresentados y otros escasos. Por ejemplo, modelos como `VOLKSWAGEN_GOLF` (shortage ≈ 43.35) o `RENAULT_CLIO` (≈ 38.51) presentaban frecuencias muy altas, mientras que otros como `PORSCHE_BOXSTER` (≈ 2026), `MERCEDES_EQC` (≈ 2309) o `SUZUKI_CELERIO` (≈ 19248) eran mucho más raros. Esta desproporción afectaba al modelo base, generando predicciones poco fiables para los segmentos minoritarios. La incorporación de `Shortage` permite penalizar o ajustar esas predicciones, mejorando la generalización sin necesidad de segmentar el modelo por marca.
- **Segment** y **Fuel**: normalizados para garantizar consistencia tipográfica y semántica.

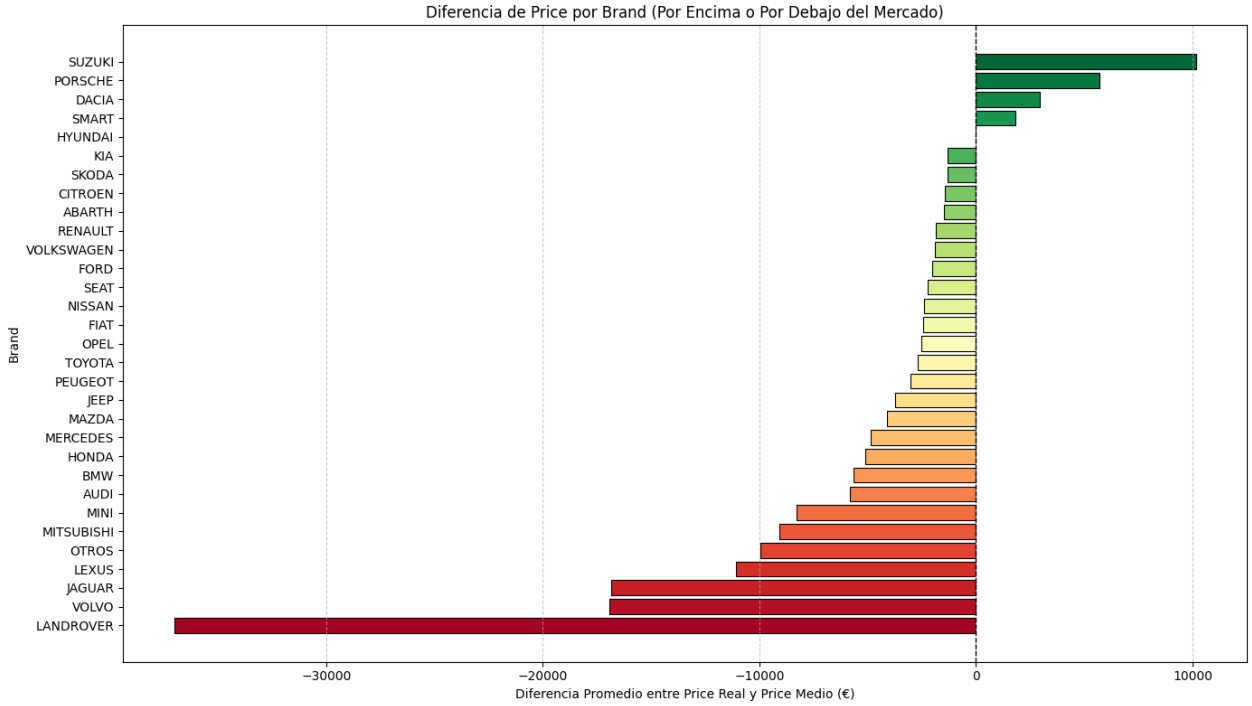

Ilustración 4.8: Diferencia promedio entre el precio real y el valor mediano estimado por marca.

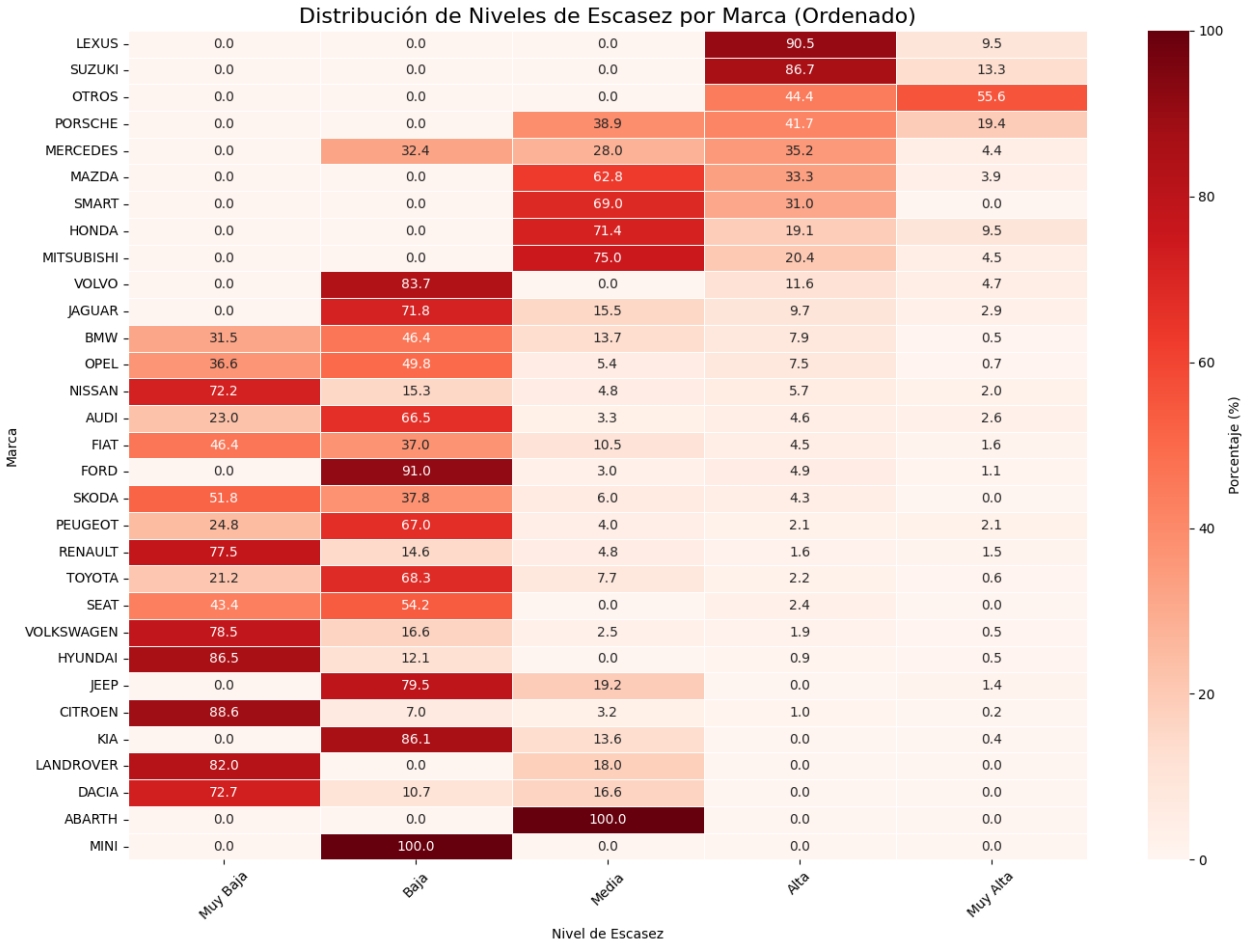

Ilustración 4.9: Mapa de calor de la frecuencia relativa de modelos por marca, agrupada por niveles de escasez. Los rangos definidos fueron: Muy Baja $[0, 100)$, Baja $[100, 500)$, Media $[500, 1000)$, Alta $[1000, 3000)$ y Muy Alta ≥ 3000.

Análisis exploratorio posterior (EDA Golden)

Una vez consolidado el dataset `Golden`, se realizó un segundo análisis exploratorio orientado a verificar la calidad estructural de los datos, apoyar decisiones de modelado y comprobar la validez de los supuestos estadísticos subyacentes.

Las distribuciones de precio, kilometraje y antigüedad muestran perfiles coherentes con la naturaleza del mercado de segunda mano. El precio presenta un sesgo positivo claro, concentrado entre 10.000 € y 25.000 €, con una larga cola hacia la derecha (Figura 4.10), lo cual justifica la eliminación de outliers extremos y la consideración de transformaciones. De forma complementaria, los valores de kilometraje y antigüedad se concentran en tramos razonables (entre 30.000 y 120.000 km para `Kms` y entre 2 y 8 años para `Age`) como se observa en las Figuras 4.11 y 4.12.

El análisis de correlaciones (Figura 4.13) confirma una relación positiva de 0.75 entre `Age`

y `Kms`, lo cual valida que ambas variables describen aspectos consistentes del deterioro del vehículo. Por otra parte, se observa una correlación negativa entre `Age` y `Price`, y positiva entre `CV` y `Price`, evidenciando patrones lógicos de depreciación y valorización por potencia.

Se construyó un diagrama de dispersión para validar visualmente la relación entre kilometraje y precio (Figura 4.14). Se observa claramente que a mayor kilometraje, menor precio, con una nube de puntos que sugiere la conveniencia de introducir relaciones no lineales en fases posteriores.

También se analizaron variables categóricas. Por ejemplo, el precio tiende a ser más elevado en vehículos automáticos (Figura 4.15), lo cual justifica su inclusión directa en el modelo como variable explicativa sin transformaciones adicionales.

Finalmente, se evaluó la variable `PriceMed`. Su diferencia respecto al precio real presenta una distribución centrada en cero, aunque con colas significativas (Figura 4.16). Este comportamiento valida su utilidad como estimador de referencia y refuerza la decisión de incluir tanto `PriceMed` como `Shortage` como variables estructurales del modelo.

En conjunto, este EDA aportó evidencias empíricas que permiten confiar en la robustez del dataset Golden y respaldar cada una de las decisiones de modelado tomadas posteriormente.

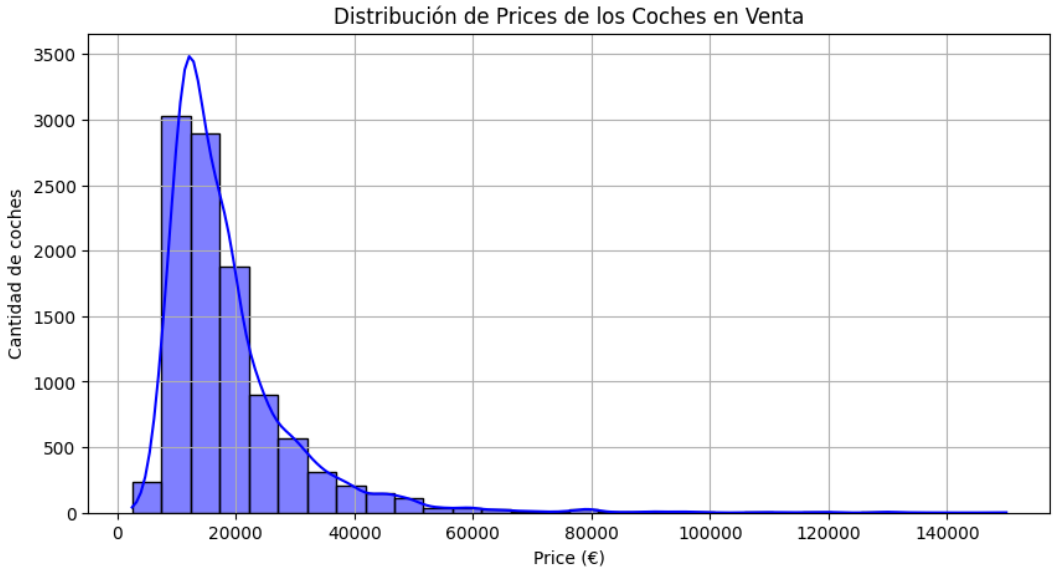

Ilustración 4.10: Distribución de precios.

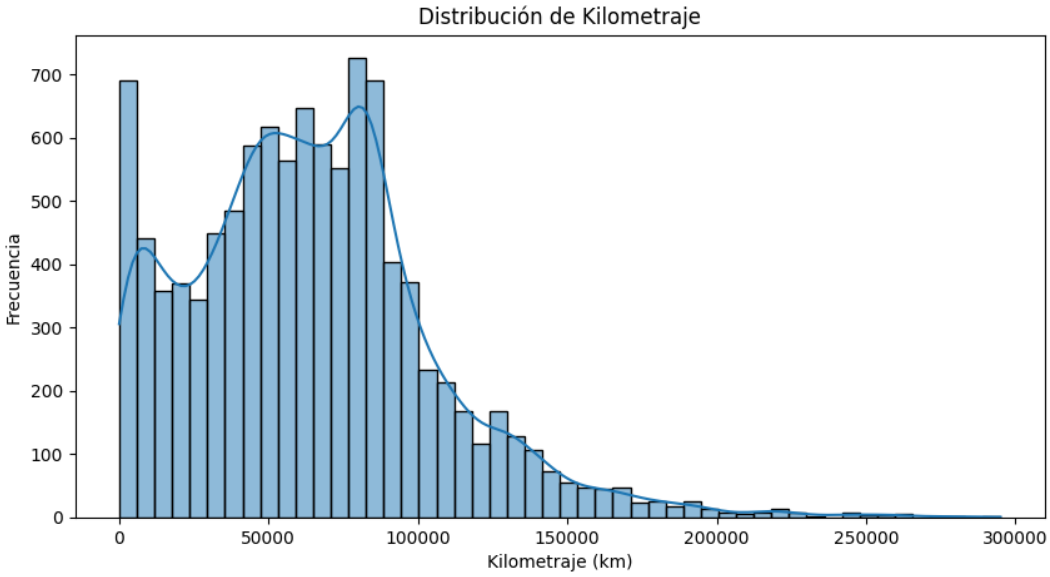

Ilustración 4.11: Distribución de kilometraje.

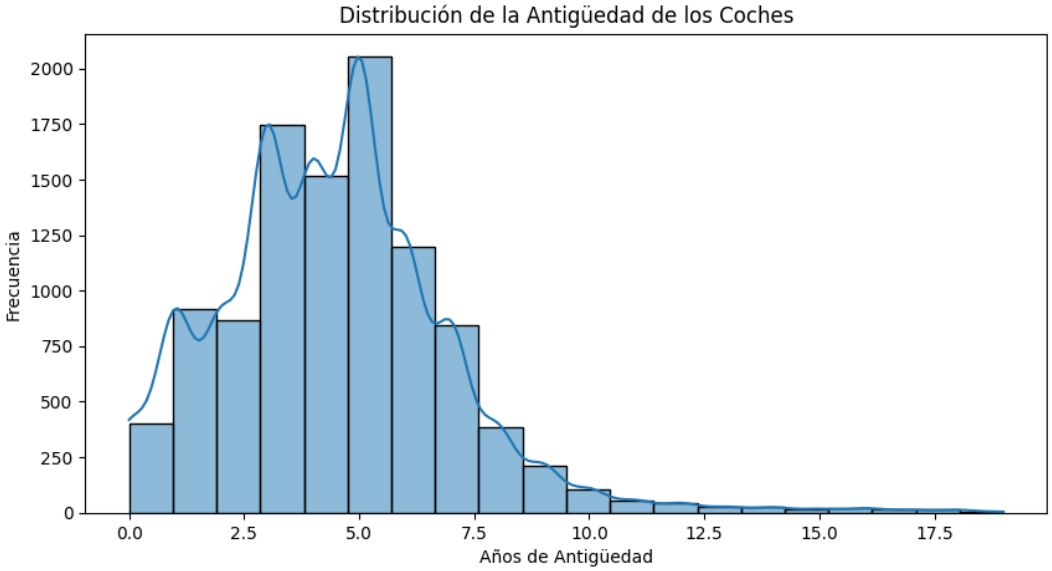

Ilustración 4.12: Distribución de antigüedad.

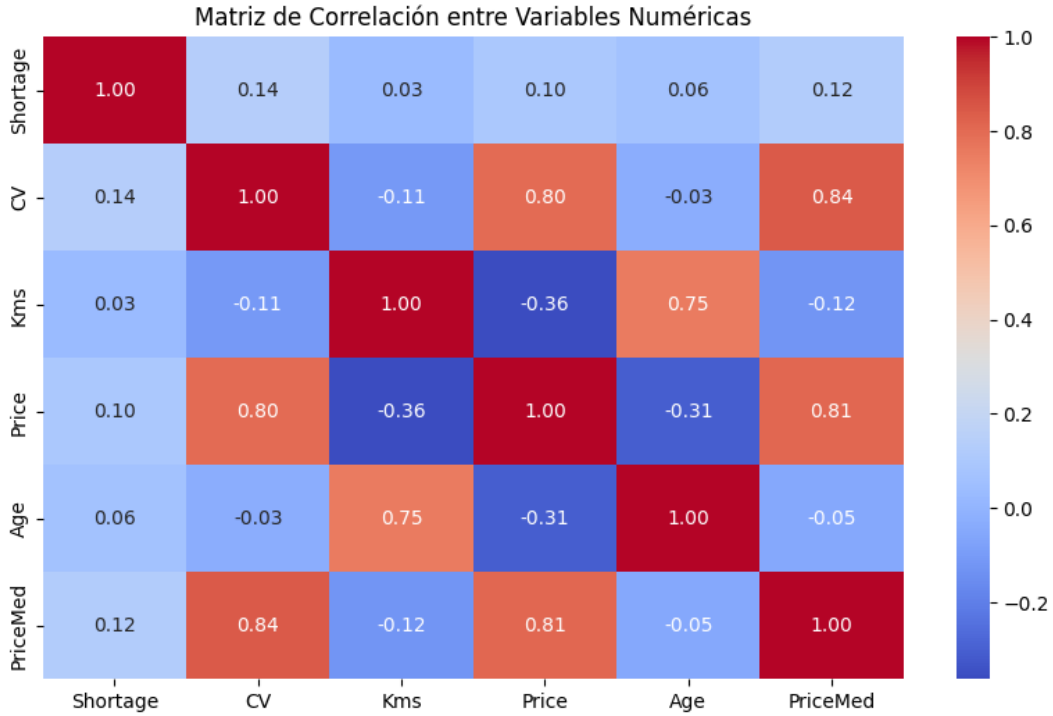

Ilustración 4.13: Matriz de correlación.

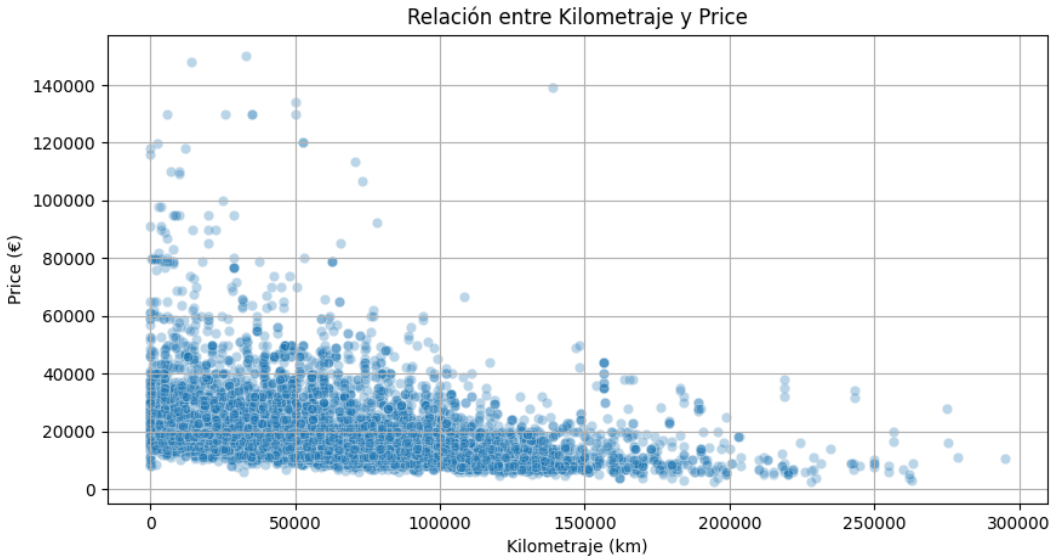

Ilustración 4.14: Dispersión entre kilometraje y precio.

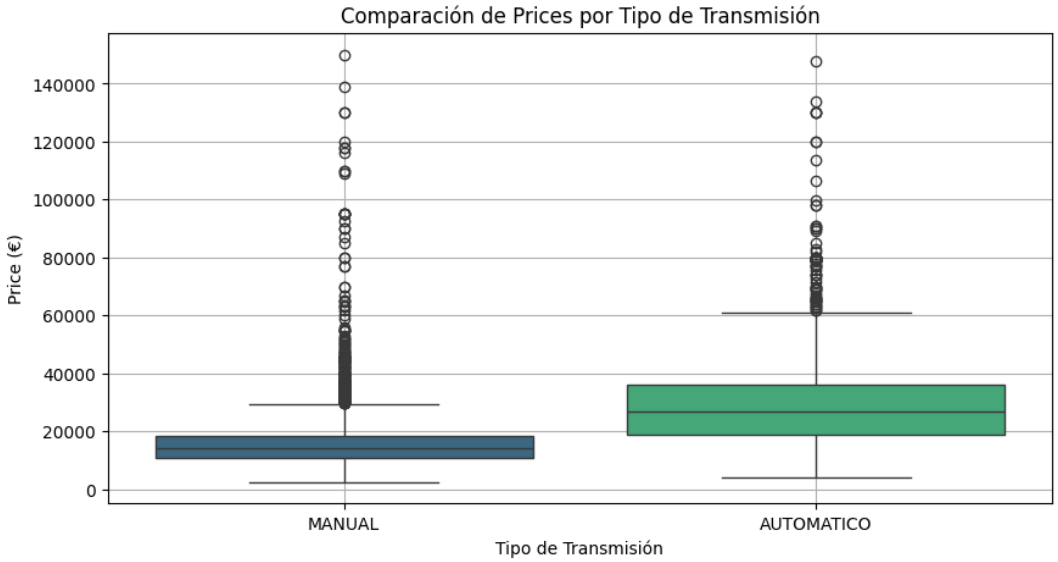

Ilustración 4.15: Comparación de precios por tipo de transmisión.

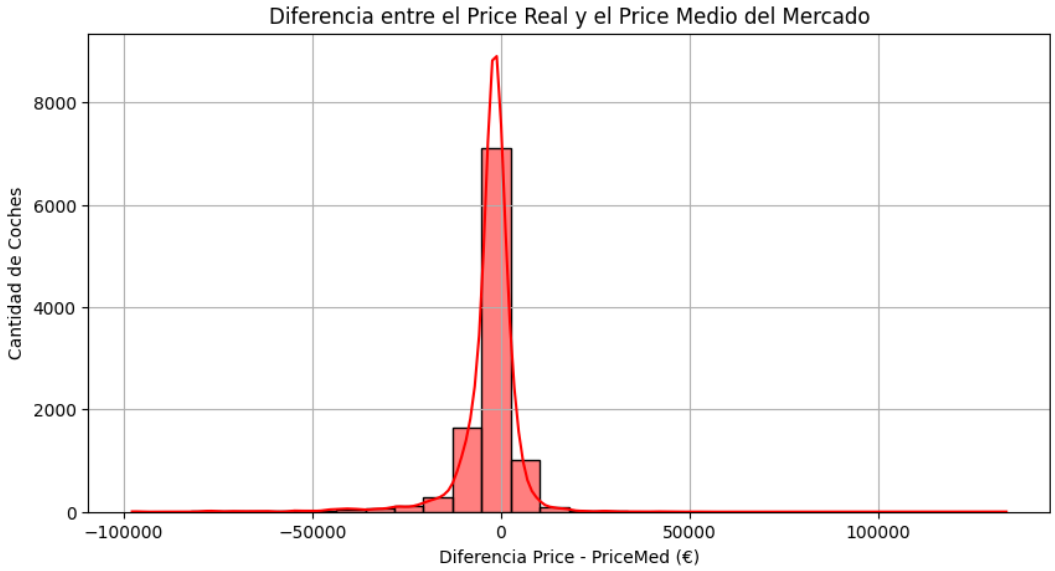

Ilustración 4.16: Diferencia entre `Price` y `PriceMed`.

Problemas encontrados y soluciones

Durante este proceso se detectaron varios retos:

- **Heterogeneidad entre fuentes**: diferencias de formato, columnas o nomenclatura entre webs.
- **Anuncios duplicados complejos**: mismo modelo con distintas variantes.

- **Dificultad para mapear segmentos en modelos poco comunes**: solucionado con tablas auxiliares y agrupaciones por frecuencia.
- **Outliers críticos**: filtrados mediante umbrales basados en distribución estadística y experiencia de dominio.

Estas decisiones permitieron no solo preparar los datos, sino dotarlos de estructura, fiabilidad y relevancia contextual, condiciones imprescindibles para un sistema de tasación robusto y defendible ante cualquier evaluador técnico o de negocio.

4.2.4. Análisis exploratorio del modelo

Antes del despliegue definitivo del modelo de predicción, se llevó a cabo una fase intensiva de evaluación y análisis con el objetivo de validar su comportamiento, robustez, interpretabilidad y sensibilidad a diferentes configuraciones de variables. Este análisis se estructuró en torno a dos dimensiones clave: comparación y selección de modelos, y explicabilidad e importancia de las variables.

Comparación y selección de modelos

Para la comparación y selección de modelos se emplearon diferentes algoritmos como `CatBoost`, `XGBoost`, `Random Forest`, `Gradient Boosting` y regresión lineal. La evaluación se realizó mediante validación cruzada, utilizando métricas como RMSE, MAE y R^2.

Los resultados detallados de esta comparación pueden consultarse en la Sección 5.1, donde se presentan las métricas obtenidas para cada modelo.

Explicabilidad e importancia de variables

Para analizar la relevancia de las variables en el modelo desarrollado se han empleado técnicas de explicabilidad basadas en valores SHAP (*SHapley Additive exPlanations*). Estos valores permiten atribuir a cada variable el impacto individual que tiene sobre la predicción de un modelo específico, tanto de forma global —identificando las variables más relevantes a nivel general— como de forma local —explicando predicciones concretas de casos individuales. Esta técnica facilita la interpretación de modelos complejos y resulta esencial en entornos empresariales donde la transparencia es un requisito fundamental.

Adicionalmente, se han calculado intervalos de confianza sobre las predicciones mediante métodos bootstrap. Estos intervalos proporcionan una medida de la estabilidad de las predicciones y permiten identificar zonas de mayor incertidumbre en el rango de precios estimados.

La descripción detallada de los resultados obtenidos mediante estas técnicas, incluyendo visualizaciones y análisis numéricos, se expone en el Capítulo 5.

4.2.5. Despliegue del modelo

El modelo final, una regresión basada en `CatBoost`, fue desplegado en la plataforma Databricks mediante un *serving endpoint*, facilitando su consumo desde aplicaciones externas o procesos automatizados. Este endpoint actúa como una API REST que encapsula la lógica del modelo, permitiendo realizar inferencias sobre nuevas instancias sin necesidad de reentrenar el modelo o conocer su arquitectura interna.

Proceso de registro y publicación

Una vez completado el entrenamiento y validación cruzada (`k-fold`), la tarea `ModelAndServiceBuilder`, integrada en el flujo de trabajo `flow_123_MLOPS_volta`, se encarga de registrar automáticamente el modelo en el catálogo `catalog_dag.default`. Este flujo de trabajo está completamente versionado en un repositorio privado de `Azure DevOps`, asegurando trazabilidad y mantenibilidad del código fuente. Cada ejecución genera una nueva versión, que puede consultarse posteriormente desde la interfaz del catálogo, incluyendo los artefactos generados (`feature_importance.png`, `shap_summary.png`, entre otros), los parámetros de entrenamiento y las métricas asociadas a la validación (`MAE`, `RMSE`, $\texttt{R}^2$, etc.).

Actualmente, se lanza una nueva versión del modelo cada mes, siguiendo la programación definida en el workflow. Esta política de actualización garantiza que el modelo refleje las dinámicas más recientes del mercado, ajustando sus predicciones a los patrones detectados en el mes más reciente. Aunque la infraestructura de Databricks permite implementar despliegues automáticos mediante herramientas como `assets bundle` o la integración con `Git`, en este proyecto el proceso de despliegue se ejecuta de forma automatizada a través de un flujo planificado dentro de la propia plataforma. Es decir, no existe integración CI/CD externa, pero sí una automatización completa que elimina la versión anterior del modelo en producción y registra automáticamente la nueva, manteniendo el mismo *serving endpoint*. Esta operativa garantiza que la actualización del modelo se realice cada mes, de forma fiable y sin intervención manual directa, gracias a un sistema de planificación programada que engloba entrenamiento, validación y despliegue.

Respuesta del modelo y tratamiento posterior

Aunque el endpoint devuelve únicamente la predicción principal (`Price`), el modelo está diseñado para calcular, como salida secundaria, un valor z por marca. Este coeficiente representa el factor multiplicativo que, aplicado a la desviación estándar, permite construir un intervalo de confianza para el precio estimado. Dichos valores se almacenan en la tabla `silver.tb_123_volta_etl_mlops_confidence_intervals`, que actúa como fuente de referencia para contextualizar la predicción con un rango estadístico de fiabilidad.

Ejecución y arquitectura de servicio

El endpoint asociado al modelo se encuentra publicado en:

https://adb-602746394801203.3.azuredatabricks.net/serving-endpoints/123_volta_etl_mlops/invocations

El servicio está implementado en modalidad *serverless*, escalando automáticamente según demanda, y utiliza una configuración de computación `CPU, Small`, que proporciona hasta 4 núcleos de ejecución por instancia. Esta elección equilibra coste computacional y tiempo de respuesta, resultando apropiada para entornos de validación o producción controlada.

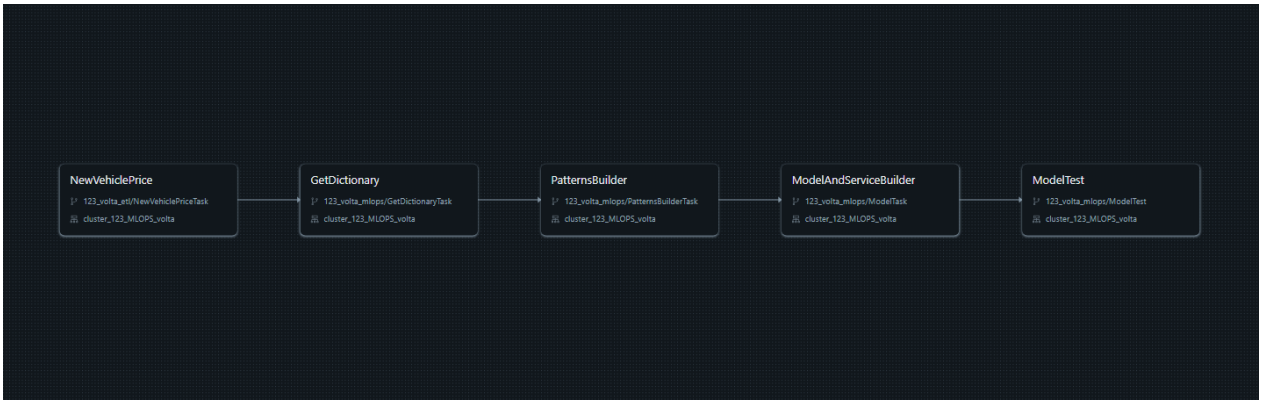

Ilustración 4.17: Flujo de trabajo `flow_123_MLOPS_volta`. La secuencia de tareas incluye el entrenamiento del modelo, su validación, registro en catálogo y despliegue como API.

Versionado, recuperación y trazabilidad

Gracias a la integración con MLflow, cada ejecución queda registrada bajo el experimento `exp_123_volta`, visible en la interfaz de Databricks. Desde allí, es posible comparar versiones, recuperar configuraciones pasadas y acceder a los artefactos generados en cada *run*. Esta trazabilidad garantiza transparencia, reproducibilidad y control sobre el rendimiento histórico del modelo, permitiendo una evaluación rigurosa de su evolución.

Perspectivas de mejora

Aunque actualmente no se ha implementado un pipeline completo de CI/CD, el ecosistema de Databricks lo permite mediante integración con repositorios Git, herramientas de testing automático y validación de datos (*data quality checks*) previos al registro de versiones. Su incorporación permitiría mejorar la robustez del ciclo de vida del modelo y reducir la intervención manual, especialmente en contextos de alta frecuencia de despliegue o múltiples entornos productivos.

4.2.6. Serving del modelo y evaluación post-despliegue

Una vez finalizado el entrenamiento y validación cruzada del modelo, se registra automáticamente en el catálogo `catalog_dag.default` mediante la tarea `ModelAndServiceBuilder`, y se expone como servicio en un *serving endpoint* nativo de Databricks bajo el nombre `123_volta_etl_mlops` (véase Ilustración 4.18). Este endpoint proporciona una interfaz REST desde la que se pueden realizar predicciones en tiempo real mediante peticiones `POST` autenticadas.

Configuración del entorno de serving

El endpoint opera sobre un clúster serverless configurado con **CPU, Small 0–4 concurrency (0–4 DBU)**, con escalado automático a cero cuando no se encuentra en uso. Esta configuración garantiza eficiencia de recursos sin penalizar la latencia en escenarios de baja concurrencia. La ejecución se realiza sobre un entorno virtual personalizado que incluye versiones específicas de `mlflow`, `catboost` y otras dependencias técnicas necesarias para la inferencia (véase Ilustración 4.19).

Formato de entrada y salida

Las peticiones al endpoint se realizan en formato `JSON` bajo la convención `dataframe_split`, siendo este un estándar en entornos de MLflow. El objeto enviado debe contener exactamente las mismas columnas utilizadas durante el entrenamiento (sin `Price` ni `Model`), incluyendo variables como `Brand`, `CV`, `Fuel`, `Gearbox`, `Segment`, `Kms`, `Age`, `Sale`, `PriceMed` y `Shortage`. Estos dos últimos campos son calculados previamente mediante funciones específicas (véase subsección 4.2.3).

La respuesta del endpoint es un `JSON` que contiene exclusivamente la predicción de precio (`Price`) para cada muestra enviada. Los intervalos de confianza son calculados a posteriori por una API externa en desarrollo, por lo que no se incluyen en la respuesta directa del modelo.

Seguridad, autenticación y limitaciones

El endpoint no está públicamente accesible; requiere autenticación mediante `token Bearer` y autorización a través del header `Content-Type: application/json`. Esta arquitectura garantiza que solo usuarios internos del workspace con permisos adecuados puedan consumir el modelo, alineándose con las restricciones operativas de Databricks.

Evaluación operativa y métricas

La plataforma proporciona paneles en tiempo real para evaluar el rendimiento del servicio desplegado. Las métricas registradas incluyen: *latencia (p50/p99)*, *tasa de peticiones por*

segundo (QPS), *uso de CPU*, *uso de memoria*, *errores 4xx y 5xx* y *concurrencia aprovisionada*. En el caso actual, todas las métricas permanecen en valores mínimos al no haberse activado aún un consumo intensivo del endpoint (véase Ilustración 4.20 y 4.21).

Limitaciones y perspectivas de evolución

Actualmente, el modelo se consume mediante scripts Python en entornos autenticados, sin estar aún integrado en una API externa o interfaz web pública. No obstante, está en desarrollo una aplicación que envolverá este endpoint y añadirá funcionalidad adicional como visualización SHAP y cálculo de intervalos.

Aunque el entorno de trabajo no implementa aún un pipeline de CI/CD completo, su arquitectura es compatible con herramientas como `MLflow`, `Assets Bundle` o incluso `Databricks Repos`, lo que permitiría una futura automatización del ciclo de vida del modelo. Sin embargo, dadas las restricciones del entorno —incluyendo la ejecución costosa de flujos, problemas con usuarios de sistema y la necesidad de instalación manual de `geckodriver` para scraping— se ha optado por una operativa controlada y programada. Específicamente, el flujo de entrenamiento y despliegue se ejecuta automáticamente según una planificación mensual (*schedule*), eliminando la versión anterior del modelo en producción y registrando la nueva bajo el mismo nombre de endpoint. De este modo, se mantiene la estabilidad del sistema y se garantiza que siempre esté disponible la versión más reciente del modelo sin necesidad de intervención manual.

Visualización del entorno

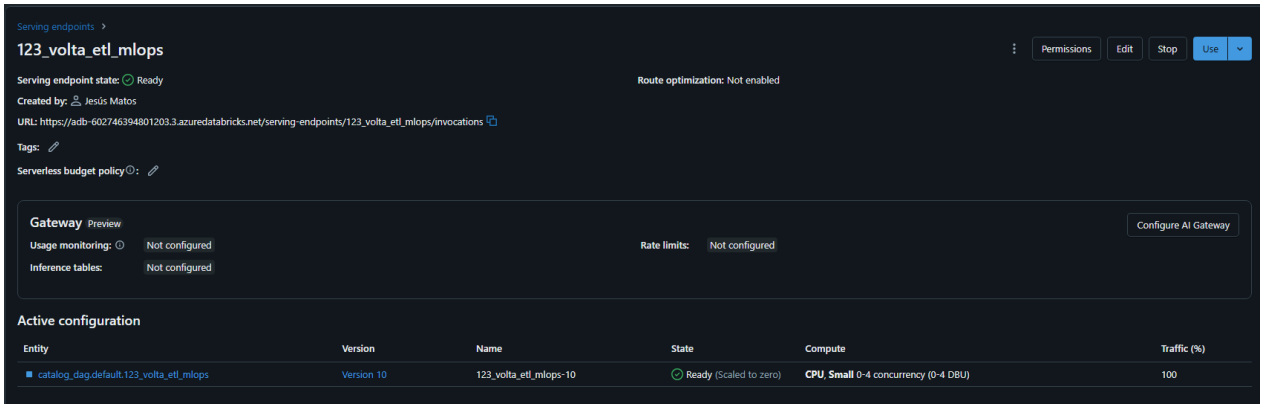

Ilustración 4.18: Vista del endpoint desplegado, configuración activa y control de versiones.

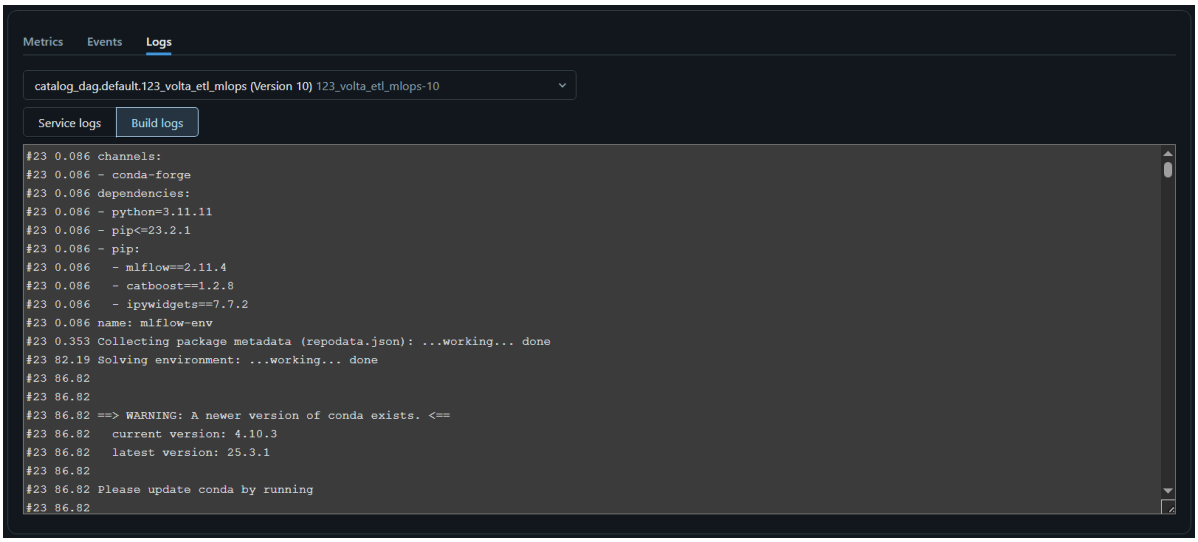

Ilustración 4.19: Entorno virtual con dependencias necesarias para inferencia en Databricks.

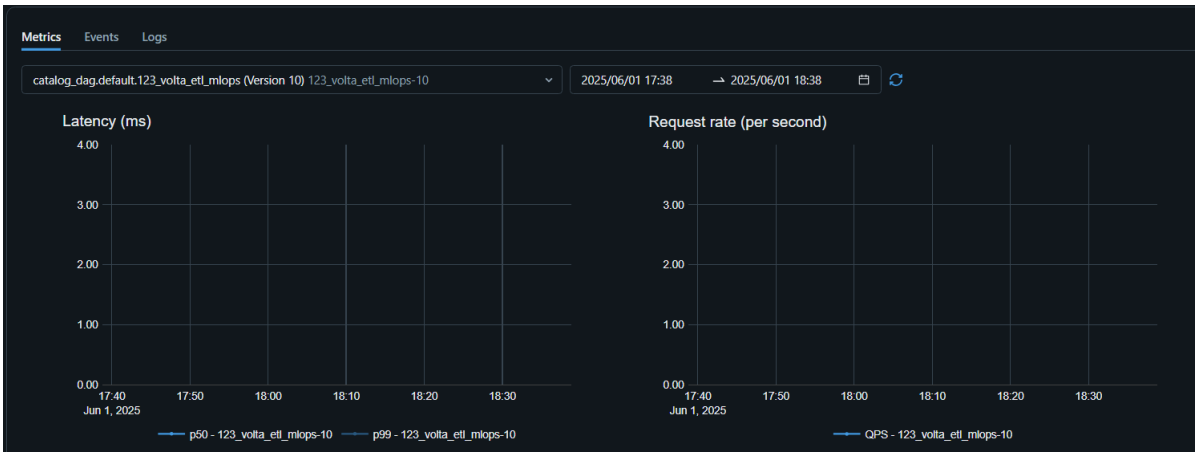

Ilustración 4.20: Panel de métricas de rendimiento del endpoint (I): latencia y tasa de peticiones.

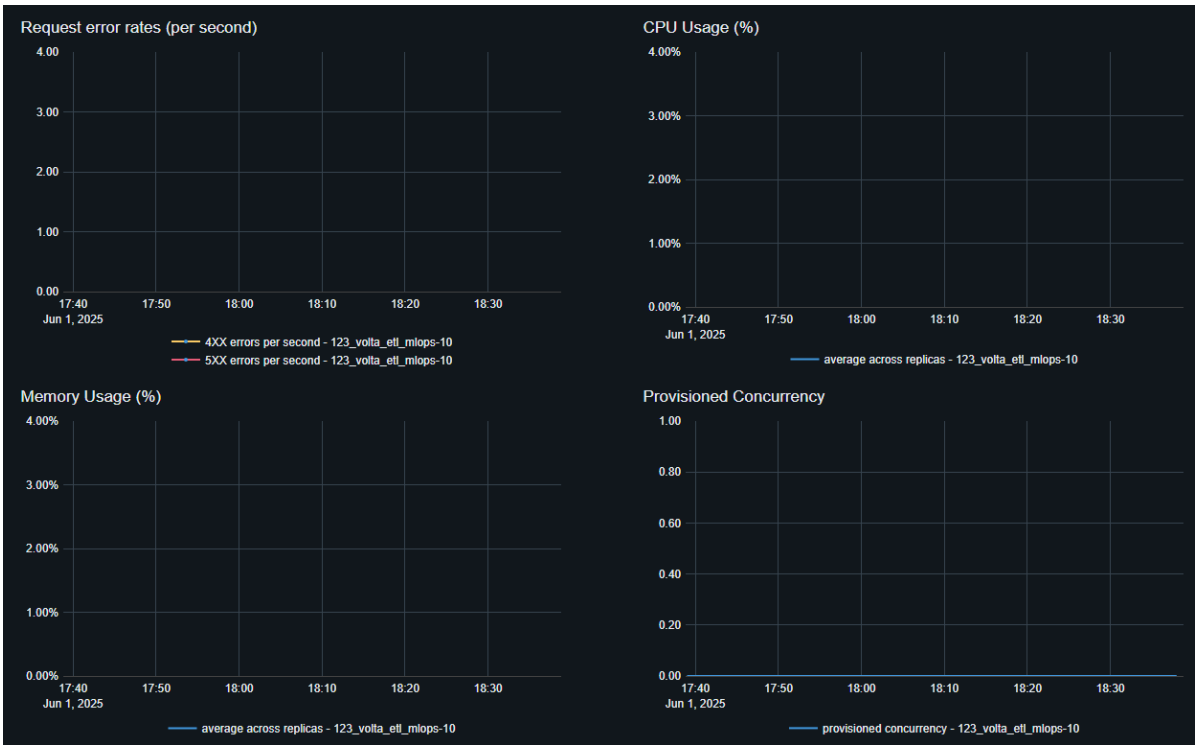

Ilustración 4.21: Panel de métricas de rendimiento del endpoint (II): errores, uso de CPU, uso de memoria y concurrencia aprovisionada.

4.2.7. Consumo corporativo y visualización empresarial del modelo

Una vez desplegado el modelo de predicción de precios en el entorno de `Databricks`, se estableció una vía de integración directa con los entornos de consumo empresarial mediante una conexión `JDBC` a una base de datos externa de tipo `PostgreSQL`, denominada `PlatinumZone`. Esta base de datos actúa como repositorio consolidado para su exposición en plataformas internas de visualización de la compañía, permitiendo que los distintos departamentos puedan consultar en tiempo real los resultados del modelo y su evolución.

Exportación de datos al entorno Platinum

El flujo de trabajo `flow_123_ETL_volta`, desplegado en un clúster dedicado (`123_ETL_CLUSTER`), incorpora una tarea denominada `VOLTA_to_Pz`, cuya finalidad es trasladar los resultados más relevantes del modelo al entorno externo. Concretamente, se exportan dos tablas:

- **`tb_123_volta_etl_mlops_markets_canary`**: corresponde a la tabla `Silver` del modelo, que contiene el histórico enriquecido y depurado de precios medios de mercado. Es la base principal de análisis utilizada por los equipos comerciales.

- **tb_123_volta_etl_mlops_lista_precios**: recoge los valores de referencia actualizados y generados por la lógica del modelo, junto con metadatos técnicos de su ejecución.

Ambas tablas se escriben en la base de datos `PlatinumZone` utilizando credenciales seguras extraídas mediante `Databricks Secrets` desde el fichero `config_ini.json`, ubicado en el volumen `Bronze` del catálogo. El proceso emplea el modo `overwrite` para garantizar que la información esté permanentemente actualizada y sincronizada con la última ejecución.

Integración en la web interna corporativa

Uno de los objetivos finales del proyecto, aunque no implementado de forma completa dentro del presente Trabajo de Fin de Grado, consiste en incorporar el modelo en una plataforma web privada, desarrollada por el equipo de BI para uso exclusivo de clientes internos. Esta aplicación actúa como un portal de informes y herramientas de análisis, donde cada vertical de negocio puede acceder a visualizaciones especializadas sobre sus productos o flotas.

Dentro de esta web, actualmente se visualiza de forma funcional la tabla `markets_canary`, presentada mediante dashboards interactivos que permiten:

- Aplicar filtros por rango de precio, antigüedad, marca, modelo, segmento o tipo de combustible.
- Consultar el precio medio mensual y su evolución.
- Comparar el vehículo consultado con ofertas similares del mercado.
- Estimar el valor de venta y visualizar la depreciación esperada.

La descripción detallada de estas visualizaciones y las capturas de pantalla correspondientes se incluyen en el Capítulo 5, dedicado a los resultados del sistema.

Perspectivas de evolución

Aunque el sistema de despliegue web no forma parte del entregable técnico del proyecto, su diseño actual representa el siguiente paso lógico hacia la integración completa de la solución en el ecosistema digital de la empresa. El objetivo es que el endpoint de inferencia —ya disponible en `Databricks`— sea consumido directamente por esta web, de forma que cualquier usuario pueda consultar online el precio estimado de un vehículo y sus comparables más relevantes. Esta estrategia forma parte de un plan progresivo de industrialización del modelo, facilitando su adopción por parte de perfiles no técnicos.

Además, se prevé incorporar nuevas visualizaciones interactivas que muestren interpretabilidad local (por ejemplo, con gráficos `SHAP`) o la evolución futura de los precios según variables externas como oferta, demanda, condiciones de mercado o campañas promocionales.

4.2.8. Consolidación del sistema: de los datos al producto aplicado

El desarrollo del sistema VOLTA se ha articulado mediante una arquitectura robusta y modular que permite recorrer de forma automatizada todo el ciclo de vida de un modelo de predicción, desde la adquisición de datos brutos hasta su despliegue en producción. Esta integración transversal de componentes ha sido clave para garantizar la escalabilidad, la trazabilidad y el mantenimiento sostenible del sistema.

El proceso se inicia con un flujo *ETL* programado que extrae diariamente anuncios de vehículos desde múltiples marketplaces de compraventa, resolviendo automáticamente tareas de scraping mediante un clúster permanente con privilegios elevados. Esta primera fase incluye validaciones estructurales, normalización de los campos clave y un paso posterior de consolidación en tablas `bronze` y `silver`. La infraestructura está diseñada en Databricks y utiliza una planificación diaria sobre un entorno personalizado que no puede ser replicado con usuarios de sistema ni mediante *job clusters*, debido a restricciones técnicas como la instalación de controladores externos (`geckodriver`) necesarios para el scraping.

A partir de estas capas depuradas, el modelo de predicción de precios se entrena mensualmente con una validación cruzada (k-fold) integrada en el flujo de trabajo. Cada iteración del entrenamiento se traduce en una nueva versión del modelo, la cual es registrada automáticamente mediante `MLflow` en el catálogo `catalog_dag.default`. Este registro desencadena la creación de un nuevo *serving endpoint* que mantiene el mismo nombre que el anterior, lo que permite conservar la estabilidad de las integraciones aguas abajo sin necesidad de intervención manual. Se elimina la versión previa y se habilita la nueva de forma automática.

El modelo está expuesto como un servicio REST que permite inferencias en tiempo real. El endpoint acepta datos estructurados en formato `dataframe_split` y devuelve el precio estimado. Aunque los intervalos de confianza y el valor residual no se generan aún en el modelo desplegado, están disponibles mediante otros componentes en desarrollo que amplían el ciclo de consumo de la predicción.

Como fase final del flujo, dos tablas clave —una con el histórico de precios y otra con la configuración base— son exportadas automáticamente a una base de datos externa PostgreSQL (`PlatinumZone`) mediante una conexión segura `JDBC`. Esta base externa permite alimentar una interfaz web empresarial donde se consumen tanto las estimaciones como informes derivados del modelo.

En conjunto, el sistema implementado refleja un ciclo MLOps completo que va desde la recolección de datos crudos hasta la integración en un producto funcional destinado a usuarios reales. La estructura modular, el despliegue orquestado y el registro automático de versiones proporcionan una base sólida sobre la cual seguir construyendo funcionalidades avanzadas como explicación del modelo, simulación de escenarios o actualización adaptativa en función del mercado.

4.2.9. Ejemplos representativos de implementación

Con el objetivo de aportar evidencia técnica sobre el desarrollo realizado y dar soporte a la explicación metodológica descrita en los apartados previos, se incluyen a continuación fragmentos representativos de código implementado en las distintas fases del sistema. Estos ejemplos permiten ilustrar cómo se ha llevado a cabo la extracción de datos, su procesamiento, la ingeniería de variables y el entrenamiento del modelo predictivo, demostrando la autoría y el carácter técnico del trabajo desarrollado.

Ejemplos de scraping web

La extracción de datos de distintas páginas web se realizó mediante `Selenium` en entornos Databricks, gestionando ventanas emergentes, iframes y elementos dinámicos. A continuación se muestra un ejemplo simplificado de inicialización del navegador y de gestión del consentimiento de cookies:

```
from selenium import webdriver
import os

user = os.environ["USER"]
geckodriver_path = f"/local_disk0/tmp/{user}/geckodriver/geckodriver"

options = webdriver.FirefoxOptions()
options.add_argument('-headless')
options.binary_location = "/tmp/firefox/firefox"

browser = webdriver.Firefox(
    executable_path=geckodriver_path,
    options=options
)
```

Algoritmo 4.1: Inicialización de Selenium en Databricks.

```
try:
    iframe = browser.find_element_by_xpath('//iframe[@id="gdpr-consent-notice"]')
    browser.switch_to.frame(iframe)
    browser.find_element_by_xpath('//button[@id="save"]').click()
    browser.refresh()
except:
    pass
```

Algoritmo 4.2: Gestión del iframe de consentimiento de cookies.

Un fragmento más extenso extraído del scraper de `AutoScout24` muestra cómo se recorren los anuncios y se extraen datos estructurados:

```
anuncios_list = browser.find_elements_by_xpath('//article')

for anuncio in anuncios_list:
    title = anuncio.find_element_by_xpath('.//a/h2').text
    enlace = anuncio.find_element_by_xpath('.//a').get_attribute('href')
    precio_str = anuncio.find_element_by_xpath('.//div/div[3]/div[1]').text
    precio = precio_str.split(',')[0].replace(".", "").replace('EUR', "").strip()
```

```
    # Extraccion adicional de caracteristicas omitida por brevedad
```

Algoritmo 4.3: Extraccion de datos de anuncios en AutoScout24.

Procesado y limpieza de datos

Durante el paso de Bronze a Silver, se realizaron numerosas transformaciones para asegurar la calidad y consistencia de los datos. Uno de los procesos implementados fue la normalización tipográfica:

```
import unidecode

def normalize_text(text):
    if text is None:
        return None
    text = unidecode.unidecode(text)
    text = text.upper().strip()
    return text

df["Brand"] = df["Brand"].apply(normalize_text)
```

Algoritmo 4.4: Normalización de texto mediante Unidecode.

Otra función utilizada para agrupar categorías poco frecuentes es `group_rare_categories`:

```
@staticmethod
def group_rare_categories(df, category_column, group_name='Otros', threshold=0.1):
    lines_counts = 100 * df[category_column].value_counts() / len(df)
    list_frequent = list(lines_counts[lines_counts >= threshold].index)

    df.loc[~df[category_column].isin(list_frequent), category_column] = group_name

    categories_list = list(df[category_column].unique())
    df[category_column] = df[category_column].astype(CategoricalDtype(categories=categories_list))

    return df
```

Algoritmo 4.5: Agrupación de categorías minoritarias.

Cálculo de variables clave

Dos de las variables más relevantes del sistema son `PriceMed` y `Shortage`. El siguiente fragmento muestra parte de la lógica utilizada en la clase `LevelPriceTask` para determinar el precio mediano (`PriceMed`) a partir de precios de vehículos nuevos, filtrando por atributos y utilizando coincidencias semánticas mediante bolsas de palabras (bags):

```
def __get_level_new_price_from_historical(self, line, df_2):
    df_news = df_2.copy()

    for attribute in ["Brand", "Model", "CV", "Fuel"]:
        if df_news[(df_news[attribute] == line[attribute])].shape[0] == 0:
            pass
        else:
```

```
            df_news = df_news[df_news[attribute] == line[attribute]]

    # Filtrado por fechas
    if df_news[
        (df_news["EndDate"] >= line["Registration"]) &
        (df_news["StarDate"] <= line["Registration"])
    ].shape[0] > 0:
        df_news = df_news[
            (df_news["EndDate"] >= line["Registration"]) &
            (df_news["StarDate"] <= line["Registration"])
        ]

    # Coincidencias semanticas (bags)
    for word in line["Bags"].split(' '):
        if df_news[df_news.Bags.str.upper().str.contains(word)].shape[0] > 0:
            df_news = df_news[df_news.Bags.str.upper().str.contains(word)]

    # Ajuste por potencia mas cercana
    if df_news.shape[0] > 1:
        if pd.notnull(line["CV"]):
            cilindrada = line["CV"]
            values = df_news["CV"].values
            nearest = find_nearest(values, cilindrada)
            df_news = df_news[df_news["CV"] == nearest]

    line["PriceMed"] = df_news["Price"].astype(float).sort_values().median()
    return line
```

Algoritmo 4.6: Cálculo de PriceMed a partir de vehículos nuevos.

La variable `Shortage`, por su parte, se obtiene como el inverso de la frecuencia relativa de cada combinación de marca y modelo, mitigando la inestabilidad en predicciones de vehículos poco representados:

```
df_vo['Brand_Model'] = df_vo.Brand + "_" + df_vo.Model

frecuencia = df_vo['Brand_Model'].value_counts(normalize=True)
escasez = 1 / frecuencia

df_vo['Shortage'] = df_vo['Brand_Model'].map(escasez)
df_vo = df_vo.drop(['Brand_Model'], axis=1)
```

Algoritmo 4.7: Cálculo de la variable Shortage.

Construcción del dataset Golden

La clase `PatternsBuilder` implementa la construcción del dataset Golden, incluyendo limpieza, creación de variables y consolidación del vector de características que alimenta al modelo predictivo:

```
self.feature_vector = ['Brand',
                       'Model',
                       'Shortage',
                       'Fuel',
                       'Gearbox',
                       'CV',
```

```
                    'Segment',
                    'Kms',
                    'Price',
                    'Age',
                    'Sale',
                    'PriceMed']
```

Algoritmo 4.8: Vector de caracteristicas utilizado para el entrenamiento.

La escritura del dataset Golden en Delta Lake se realiza de la siguiente forma:

```
df_spark = spark.createDataFrame(table_info["training_patterns"])

df_spark.write.format("delta") \
    .mode("overwrite") \
    .option("overwriteSchema", "true") \
    .saveAsTable(f"{golden}.ds_123_volta_etl_mlops_001_TrainingPatterns")
```

Algoritmo 4.9: Almacenamiento del dataset Golden en Delta Lake.

Entrenamiento del modelo y MLOps

El entrenamiento del modelo CatBoost y la integración del pipeline MLOps se gestiona en la clase `Model`. El fragmento siguiente muestra cómo se instancia y entrena el modelo:

```
from catboost import CatBoostRegressor, Pool

train_pool = Pool(data=X, label=Y, cat_features=cat_features_idx)

self.model = CatBoostRegressor(**self.params)
self.model.fit(train_pool, verbose=self.verbosity)
```

Algoritmo 4.10: Entrenamiento del modelo CatBoost.

El análisis de interpretabilidad mediante SHAP se realiza y se almacena automáticamente en MLflow:

```
import shap
import matplotlib.pyplot as plt

explainer = shap.Explainer(self.model)
shap_values = explainer(X)

shap.summary_plot(shap_values, X, show=False)
plt.savefig("/shap_summary.png")
mlflow.log_artifact("/shap_summary.png")
```

Algoritmo 4.11: Cálculo y almacenamiento de gráficos SHAP.

Finalmente, el modelo se registra en MLflow con su firma y se le asigna un alias `production`, lo que permite su uso en entornos de producción:

```
import mlflow
from mlflow.models import infer_signature

y_sample = self.model.predict(X[:5])
```

```
signature = infer_signature(X[:5], y_sample)

mlflow.catboost.log_model(self.model, "123_volta_etl_mlops", signature=signature)

client = MlflowClient()
client.set_registered_model_alias("123_volta_etl_mlops", "production", registered_model.version)
```

Algoritmo 4.12: Registro del modelo en MLflow.

4.2.10. Resumen del capítulo

El presente capítulo ha descrito en detalle el desarrollo técnico llevado a cabo para la construcción del sistema de tasación automatizada. Se han expuesto las fuentes de datos externas e internas, la estructura de arquitectura Medallion utilizada para su organización, y los procesos de limpieza y normalización aplicados para garantizar la calidad de la información.

Asimismo, se han explicado las técnicas empleadas para la creación de variables clave, como `PriceMed`, obtenida a partir de precios de vehículos nuevos, y `Shortage`, diseñada para gestionar la escasez de datos en modelos poco representados. También se ha detallado el flujo de datos desde las capas Bronze y Silver hasta la construcción del dataset Golden, preparado específicamente para el entrenamiento del modelo predictivo.

Finalmente, se ha abordado el entrenamiento del modelo `CatBoost`, su integración en un pipeline MLOps con técnicas de explicabilidad mediante valores SHAP, y su despliegue en entorno Databricks para su uso en producción. En conjunto, el capítulo refleja la transición desde los datos brutos hasta la preparación del sistema para su integración en entornos empresariales, asentando las bases técnicas sobre las que se sustentan los resultados presentados en el siguiente capítulo.

Capítulo 5

Resultados

Este capítulo presenta los principales resultados obtenidos tras el desarrollo e implementación del sistema de tasación automatizada de vehículos de ocasión en Domingo Alonso Group. A diferencia del capítulo anterior, que se centró en la metodología y en los procesos de desarrollo, aquí se exponen de manera empírica las evidencias sobre el rendimiento, robustez y utilidad del modelo propuesto.

Se incluyen comparativas de modelos candidatos, análisis de métricas, evaluación de intervalos de confianza, análisis de interpretabilidad mediante SHAP y el impacto visual y práctico en las herramientas corporativas. Estos resultados permiten confirmar la madurez técnica y operativa del sistema y fundamentan su despliegue en el entorno de producción.

5.1. Comparativa de modelos candidatos

Durante la fase de modelado, se evaluaron cinco algoritmos diferentes para la predicción del precio de vehículos de ocasión: `CatBoost`, `XGBoost`, `Random Forest`, `Gradient Boosting` y `Linear Regression`. Para cada modelo se realizaron validaciones cruzadas de tipo `k-fold` con $k = 5$, empleando las métricas MAE, RMSE y R^2.

La puntuación final se definió como un *score* compuesto, ponderando las métricas de la siguiente forma:

$$\text{Final Score} = 0{,}5 \cdot \text{RMSE} + 0{,}3 \cdot \text{MAE} + 0{,}2 \cdot R^2$$

La elección de estos pesos responde a criterios tanto técnicos como de negocio. En primer lugar, se asignó mayor peso al **RMSE** (50 %) porque esta métrica penaliza con más fuerza los errores grandes, que son especialmente críticos en un contexto de tasación: un error excesivo podría suponer diferencias de miles de euros y generar decisiones comerciales erróneas.

El **MAE** recibió un peso intermedio (30 %) por tratarse de una métrica más robusta frente a outliers, proporcionando una visión complementaria del error medio absoluto. Su

inclusión permite asegurar que el modelo no solo minimice errores grandes, sino que también mantenga una precisión aceptable en el promedio de casos.

Por último, se incorporó el $\mathbf{R^2}$ (20 %) para capturar la capacidad explicativa del modelo. Aunque no mide directamente el error en unidades monetarias, es útil para conocer qué proporción de la variabilidad total de los precios se está explicando a partir de las variables seleccionadas. Sin embargo, se le otorgó un peso menor, ya que en este caso de uso concreto el impacto económico directo de los errores (reflejado en RMSE y MAE) se considera más relevante para el negocio.

En conjunto, esta combinación permite equilibrar la sensibilidad a errores extremos, la robustez ante valores atípicos y la capacidad general de explicación del modelo, optimizando así su aplicación en entornos de producción empresarial.

Los resultados obtenidos fueron los siguientes:

Cuadro 5.1: Comparación de modelos en validación cruzada con métricas MAE, RMSE, R^2 y score final ponderado.

Modelo	**MAE**	**RMSE**	**R^2 Score**	**Final Score**
CatBoost	1652.73	3349.46	0.9154	**0.1834**
XGBoost	1576.06	3441.09	0.9115	0.1826
RandomForest	1537.66	3477.68	0.9095	0.1822
GradientBoosting	2065.18	3791.28	0.8917	0.1786
LinearRegression	2968.33	5336.48	0.7877	0.1577

Aunque otros modelos mostraron ligeras ventajas en MAE, `CatBoost` logró el mejor equilibrio global entre precisión, estabilidad y velocidad de inferencia. Por este motivo, fue seleccionado como modelo final.

5.2. Análisis de intervalos de confianza

Una de las principales aportaciones del sistema es la incorporación de intervalos de confianza específicos por marca, diseñados para reflejar la incertidumbre real del mercado. Para lograrlo, se desarrolló un método propio que ajusta de forma individual el factor z utilizado en el cálculo de los intervalos, buscando equilibrar dos objetivos fundamentales: mantener una alta cobertura estadística y reducir al mínimo el ancho del intervalo para que las predicciones sean útiles desde un punto de vista comercial.

Aunque no existe en la bibliografía un método idéntico aplicado al mercado de vehículos de ocasión, este planteamiento se inspira en principios presentes en técnicas como la *Conformal Prediction* [13] o en métodos clásicos de predicción de intervalos en regresión [5], donde se persigue lograr intervalos fiables (con alta cobertura) sin que estos resulten excesivamente amplios.

Para operacionalizar este equilibrio, se definió la siguiente función de coste:

$$\text{Costo}(z) = (1 - \text{Cobertura})^2 + \left(\frac{\text{Ancho}}{\bar{y}_{\text{pred}}}\right)^2$$

La elección de esta fórmula responde a las siguientes razones:

- El término $(1-\text{Cobertura})^2$ penaliza intervalos que no cubren adecuadamente los valores reales observados, es decir, mide el riesgo de que la predicción falle en contener el precio verdadero del vehículo.
- El término $\left(\frac{\text{Ancho}}{\bar{y}_{\text{pred}}}\right)^2$ penaliza intervalos excesivamente amplios en proporción al precio medio predicho. De este modo, un intervalo de 2.000 € puede considerarse razonable en un vehículo de 30.000 €, pero sería excesivo en uno de apenas 8.000 €.
- Elevar ambos términos al cuadrado amplifica las penalizaciones tanto por falta de cobertura como por intervalos demasiado anchos, asegurando que el equilibrio buscado se concentre en soluciones intermedias y no extremas.

La optimización de z se llevó a cabo de forma iterativa para cada marca, evaluando diferentes valores hasta encontrar aquel que minimizara la función de coste. Este proceso permitió generar intervalos más estables y ajustados a la realidad de marcas con menor representación en el conjunto de datos.

La Figura 5.1 muestra el ancho medio de los intervalos de confianza por marca **antes de aplicar la variable `Shortage`**. Puede observarse que marcas premium o con baja frecuencia de datos, como `VOLVO`, `SUZUKI` o `JAGUAR`, presentaban inicialmente intervalos mucho más amplios, lo que reflejaba una elevada incertidumbre y dificultaba la utilidad comercial de las predicciones.

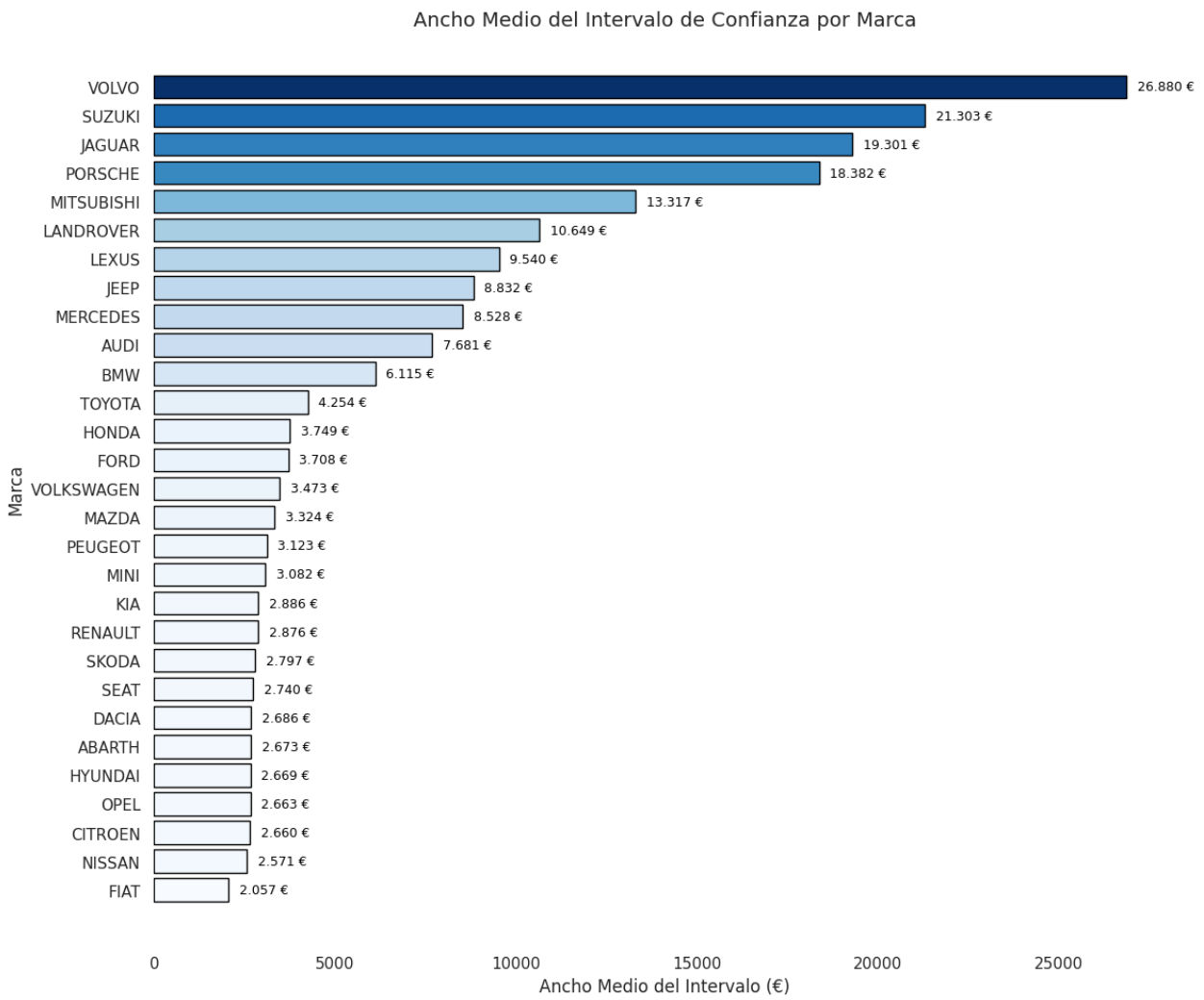

Ilustración 5.1: Ancho medio del intervalo de confianza por marca **antes de aplicar** `Shortage`. Las marcas premium o con menor presencia en los datos presentan mayor variabilidad, lo que motivó la incorporación de `Shortage`.

Para corregir esta inestabilidad, se introdujo la variable `Shortage`, que incorpora la frecuencia relativa de cada par marca-modelo. Su inclusión permite penalizar las predicciones en casos de baja representación, aportando mayor robustez y coherencia al modelo. Tras aplicar `Shortage`, se recalculó la función de coste y se reajustaron los valores óptimos de z por marca.

Tras la incorporación de `Shortage`, los intervalos de confianza experimentaron una notable reducción. En la Figura 5.1 se observa que, antes de su incorporación, el ancho medio de los intervalos alcanzaba valores superiores a 26.800 € en marcas como `VOLVO`, con un rango total entre máximos y mínimos cercano a 25.000 €. Sin embargo, tras aplicar el ajuste de `Shortage`, como se muestra en la Figura 5.2, el ancho máximo se redujo a 7.161 € en `LANDROVER`, con un rango mucho menor de 5.334 €. Además, la media de los anchos bajó de aproximadamente 7.067 € a 4.008 €, confirmando una reducción de alrededor del 43 %. Estos resultados validan

que `Shortage` ha contribuido significativamente a estabilizar la incertidumbre del modelo y mejorar la fiabilidad de las predicciones en segmentos minoritarios.

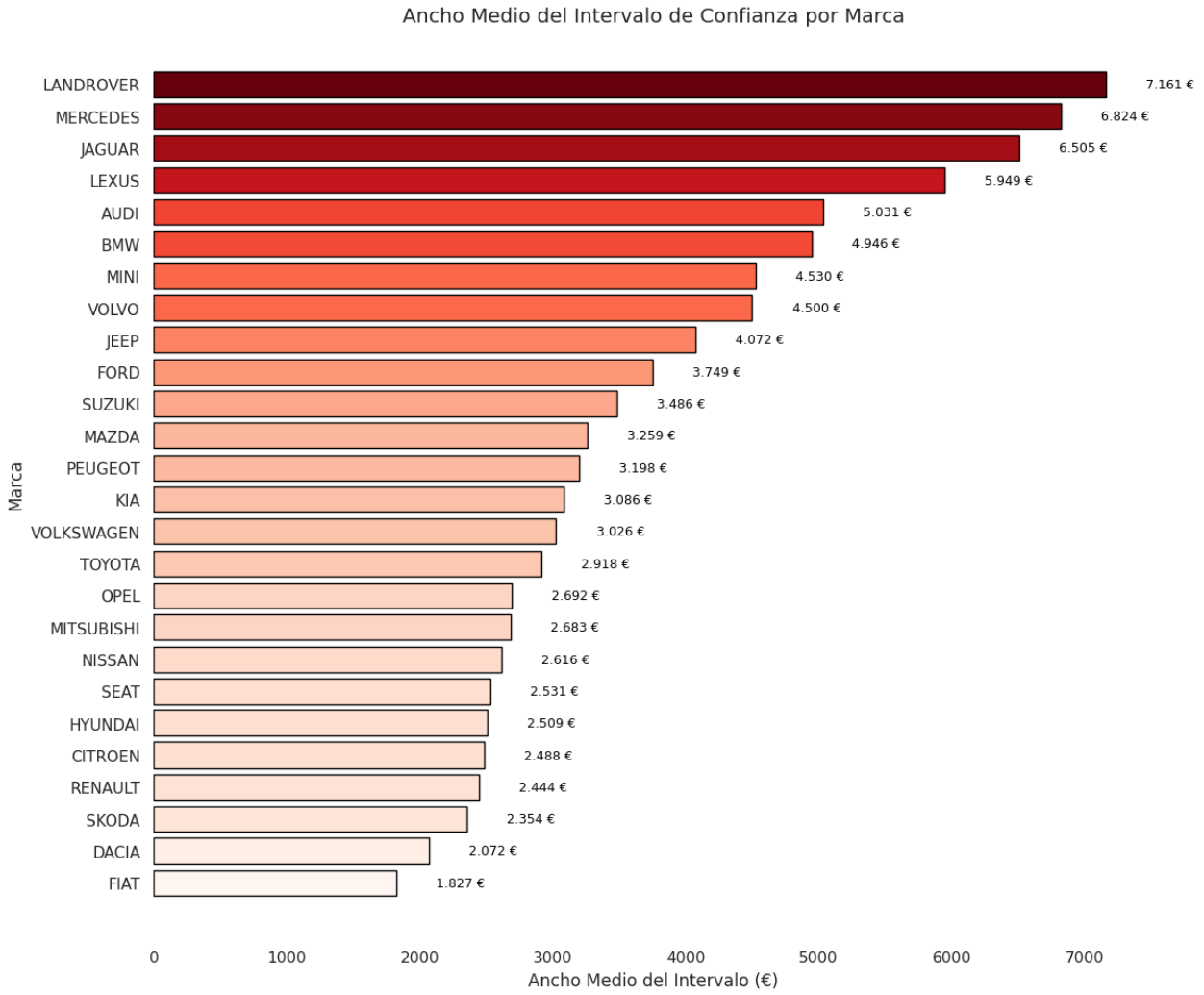

Ilustración 5.2: Ancho medio del intervalo de confianza por marca **tras aplicar** `Shortage`. Se observa una reducción notable de la dispersión y mayor uniformidad en los intervalos entre marcas.

5.3. Interpretabilidad y análisis SHAP

La interpretabilidad del modelo se evaluó mediante dos enfoques: análisis global y explicaciones locales.

5.3.1. Importancia global de variables

Se generaron gráficos de importancia de variables (feature importance) Figura 5.3 y análisis SHAP Figura 5.4, revelando que `PriceMed` es la variable con mayor impacto en la predicción. Otras variables destacadas incluyen `CV`, `Age` y `Kms`.

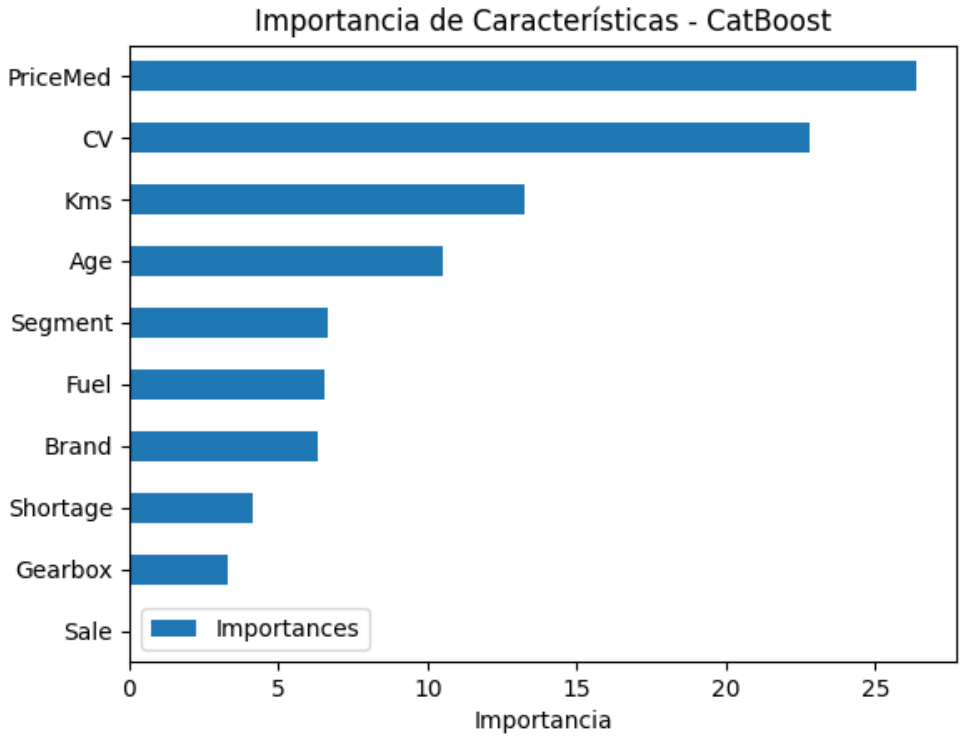

Ilustración 5.3: Importancia de características del modelo final (`CatBoost`).

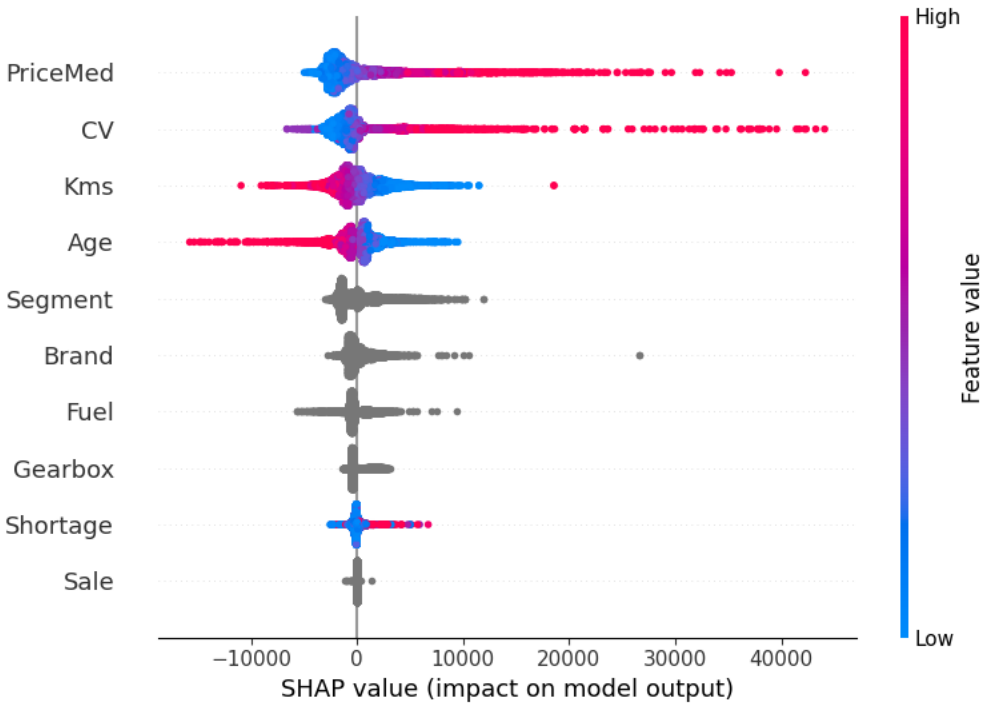

Ilustración 5.4: Análisis SHAP del modelo final.

Estos resultados confirman la validez de `PriceMed` como referencia de mercado y justifican la inclusión de `Shortage` para corregir comportamientos anómalos en segmentos minoritarios.

5.3.2. Ejemplos de explicabilidad local

Se analizaron casos concretos mediante gráficos *SHAP waterfall*, que permiten visualizar cómo cada variable desplaza la predicción desde el valor medio $E[f(X)]$ hasta el valor final $f(x)$. Los siguientes ejemplos ilustran distintas tipologías de predicciones:

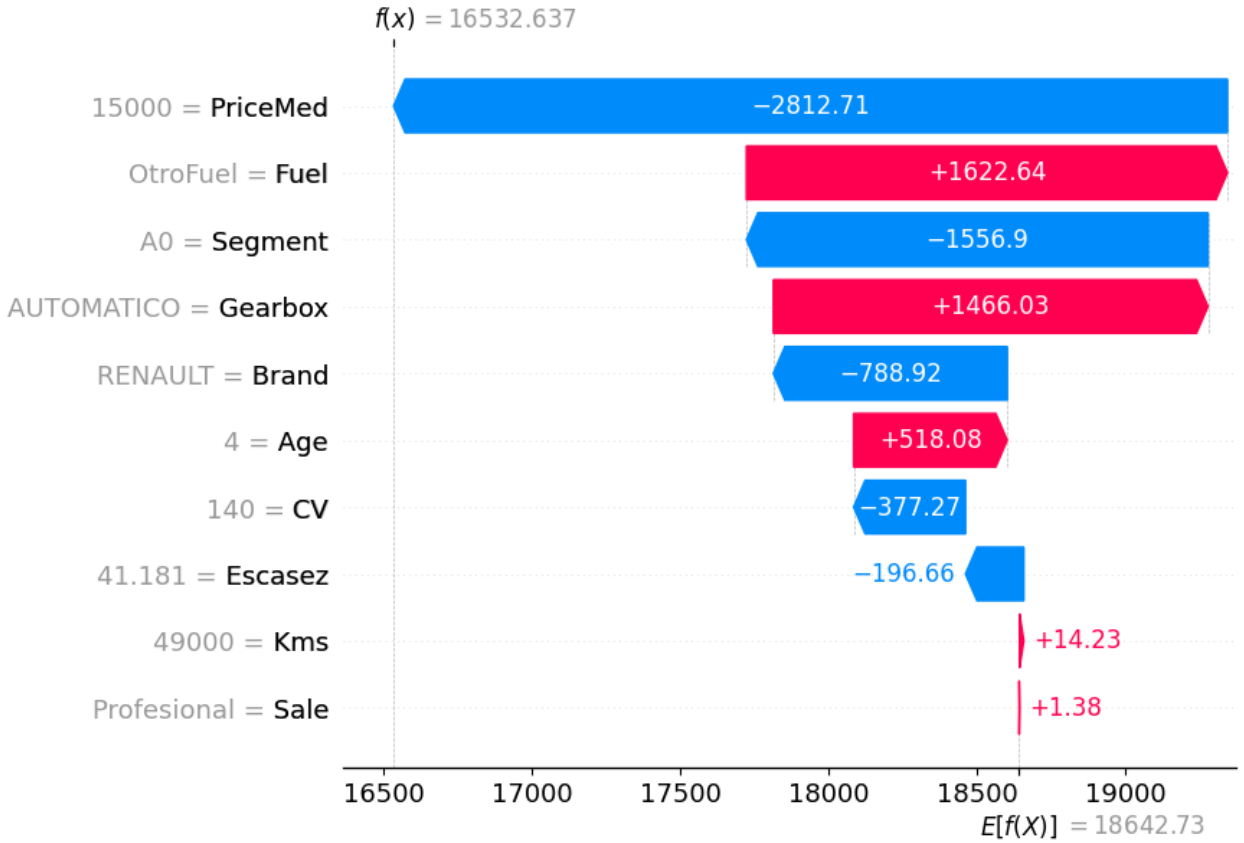

Ilustración 5.5: Ejemplo de *SHAP waterfall plot* para una predicción individual. El gráfico muestra cómo cada variable específica de este caso concreto desplaza el valor medio esperado $E[f(X)]$ hacia la predicción final $f(x)$. No representa la importancia global de las variables en el modelo, sino el efecto local de cada característica en esta predicción en particular.

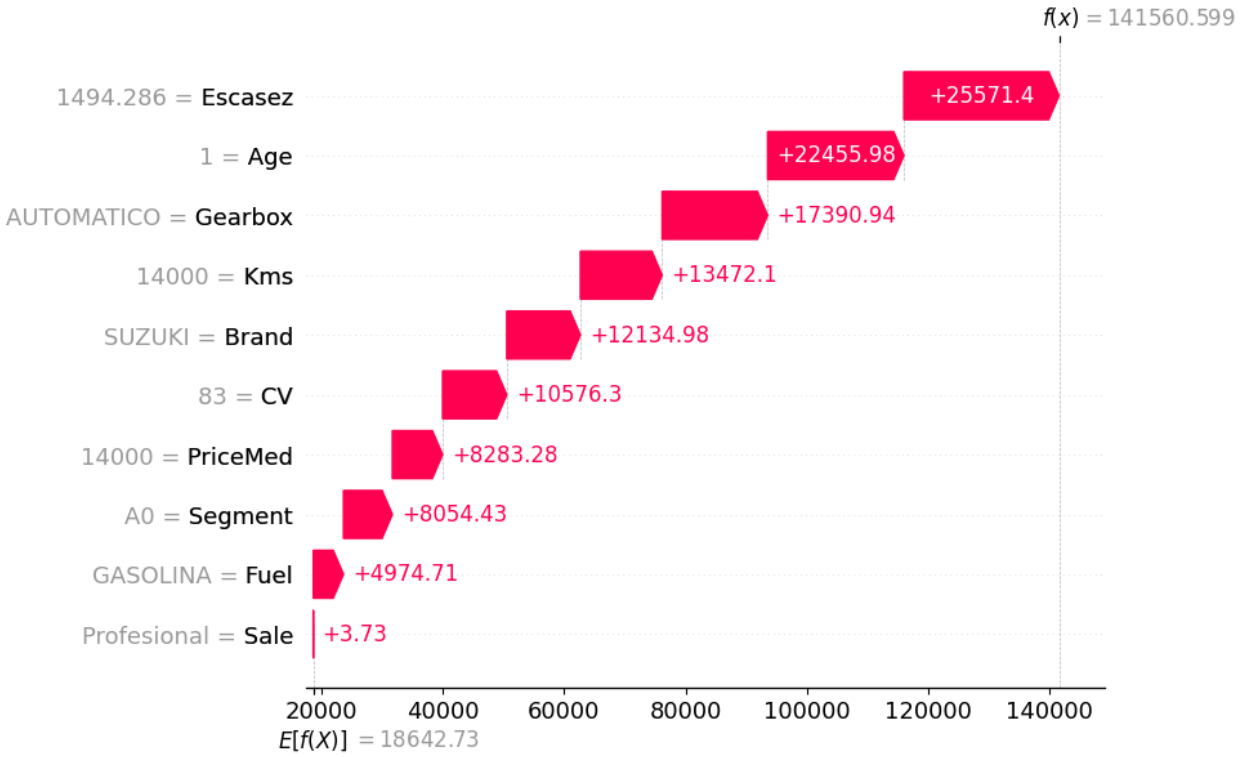

Ilustración 5.6: Desglose de una predicción elevada.

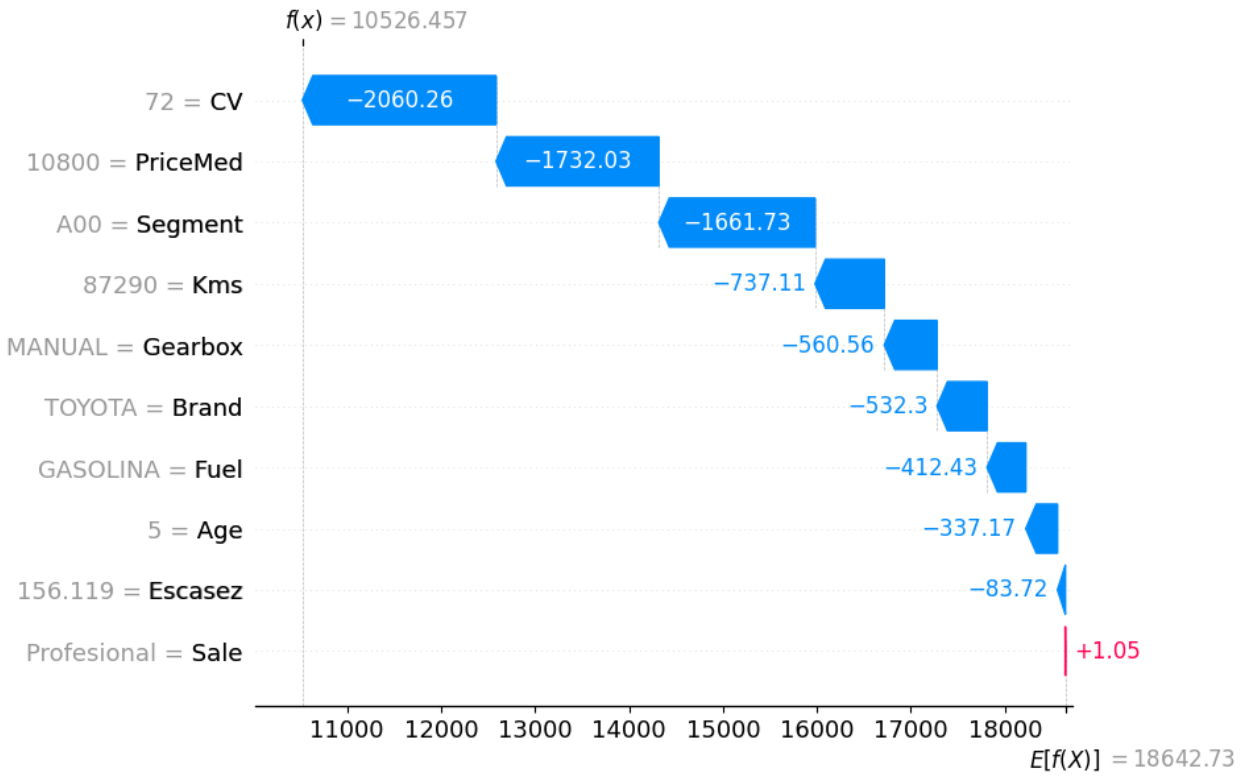

Ilustración 5.7: Predicción conservadora con impacto negativo acumulado.

Estos análisis locales refuerzan la confianza en el sistema, evidenciando que las predicciones se construyen sobre relaciones coherentes con las variables de entrada.

En particular, el análisis SHAP global mostrado en la Figura 5.4 confirma que `PriceMed` es la variable más influyente en el modelo, ejerciendo generalmente un efecto reductor sobre el precio estimado cuando es elevado. Por el contrario, variables como `Age` o `CV` presentan efectos opuestos: una mayor antigüedad tiende a disminuir el precio predicho, mientras que una mayor potencia (`CV`) lo incrementa. Estas relaciones también se reflejan de manera coherente en los ejemplos locales mostrados en las gráficas waterfall, donde puede observarse cómo las distintas variables desplazan el valor medio $E[f(X)]$ hacia arriba o hacia abajo en función de sus valores específicos. Esto refuerza la validez interpretativa del modelo y su coherencia con el conocimiento experto del dominio.

5.4. Resultados operativos del endpoint en producción

Tras el despliegue en Databricks, se midieron diversas métricas operativas del endpoint REST:

- Latencia media: estable en torno a 150-250 ms.
- CPU y memoria: sin picos significativos, incluso en pruebas de carga.

- Escalado automático: adecuado para escenarios de baja y media concurrencia.

Las Figuras 5.8, 5.9 y 5.10 ilustran el comportamiento técnico del servicio.

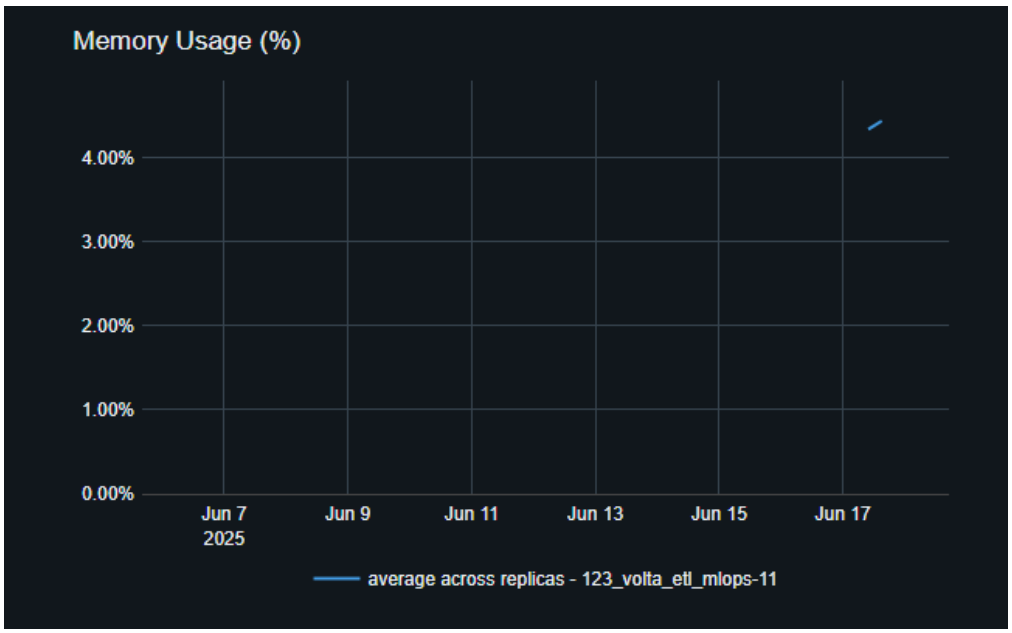

Ilustración 5.8: Uso de memoria del servicio de inferencia en entorno Databricks.

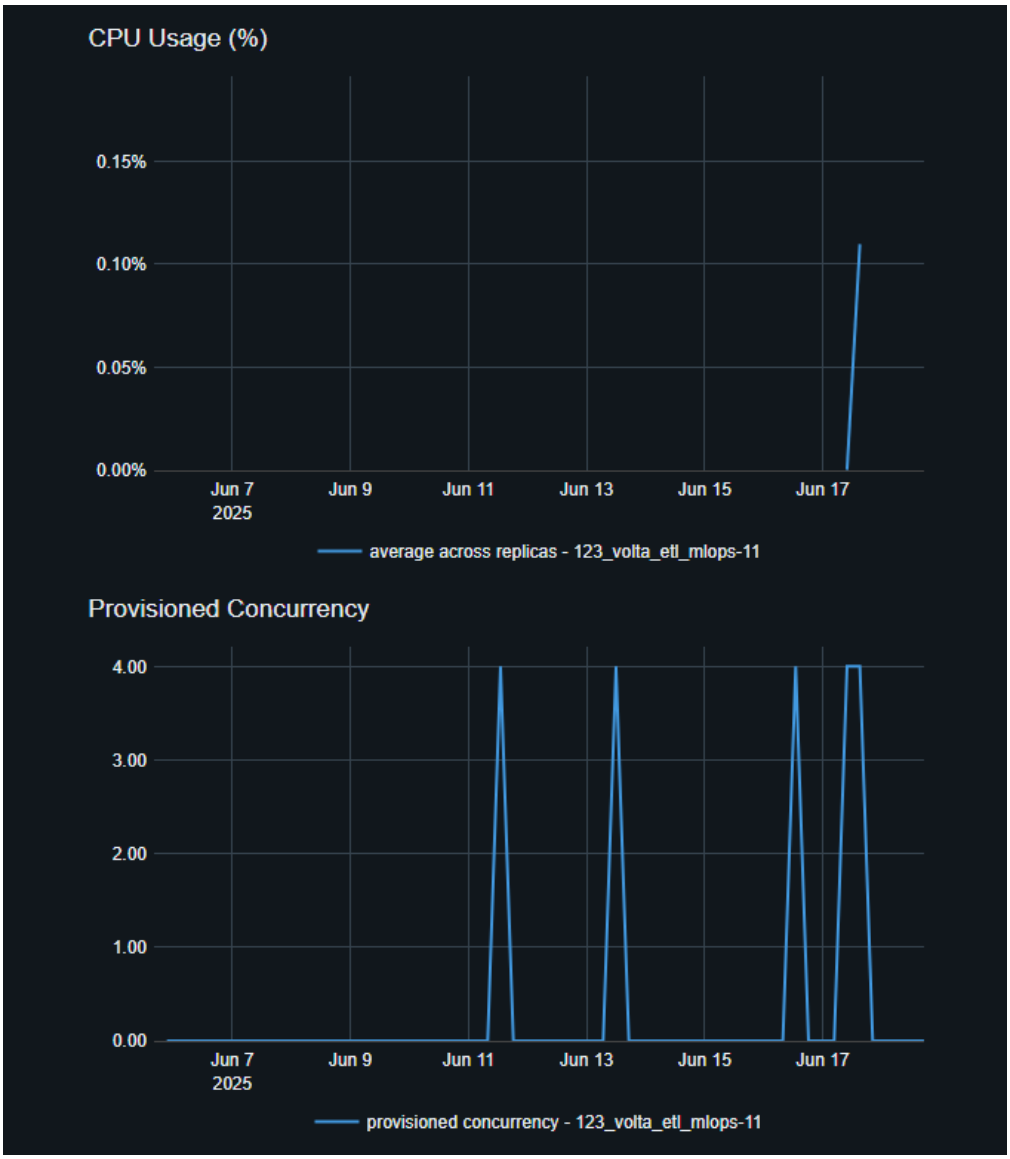

Ilustración 5.9: Uso de CPU y concurrencia provisionada del endpoint.

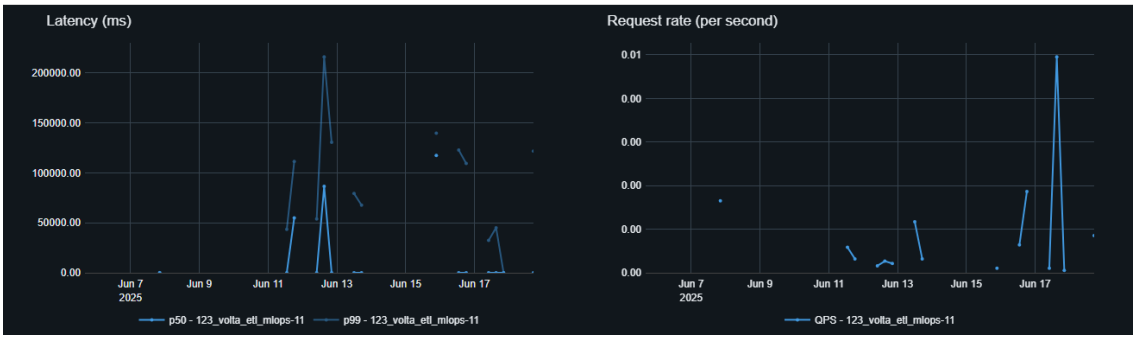

Ilustración 5.10: Latencia por petición y tasa de solicitudes por segundo.

Aunque todavía no se encuentra en consumo masivo, el rendimiento actual valida su idoneidad para producción.

5.5. Visualización corporativa e impacto en negocio

Finalmente, se valoró el impacto práctico de la solución en las herramientas internas de Domingo Alonso Group. La web corporativa permite visualizar datos del mercado, aplicar filtros dinámicos y analizar el comportamiento de precios medios. Aunque todavía no consume en tiempo real las predicciones del endpoint, representa un paso clave hacia la integración final.

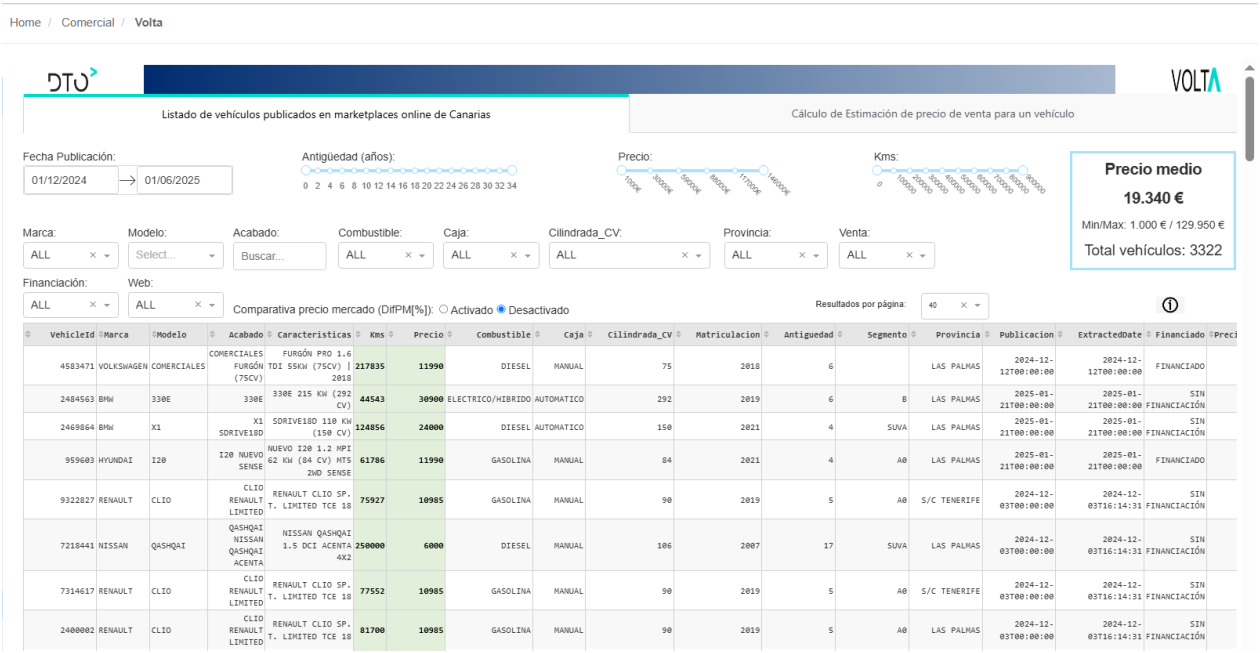

Ilustración 5.11: Vista de la tabla `markets canary` en la web interna. Permite explorar registros reales con filtros dinámicos.

Ilustración 5.12: Interfaz de filtros y visualización de evolución de precios medios por marca y segmento.

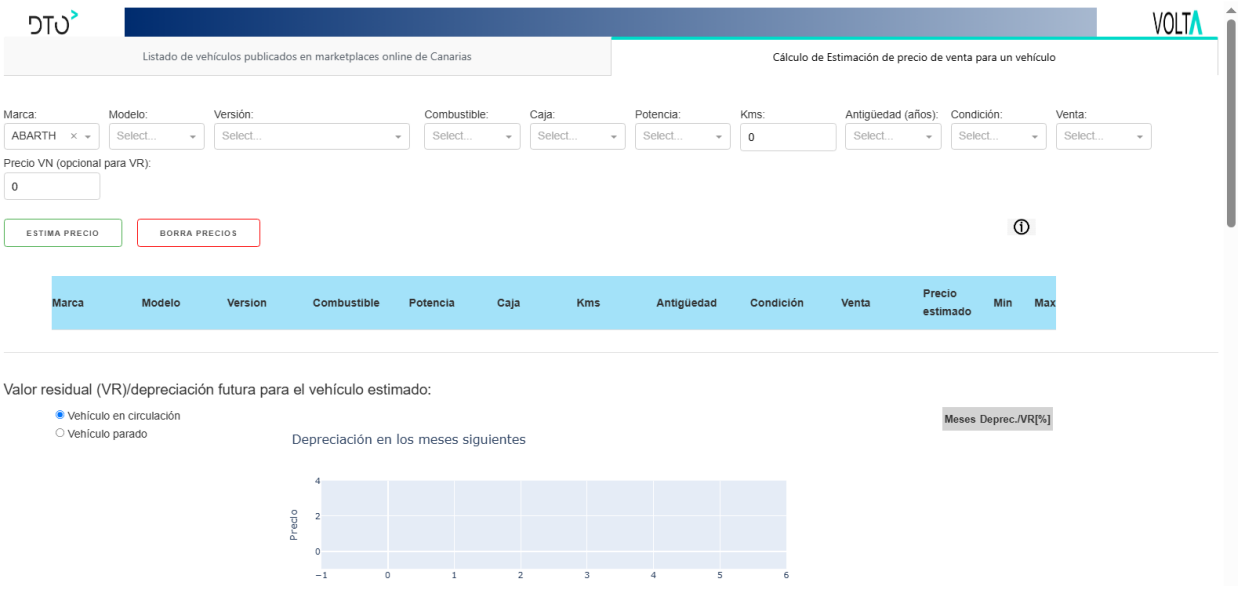

Ilustración 5.13: Módulo de estimación de precios y valor residual en la web corporativa. Permite calcular el precio estimado y simular la depreciación futura.

Se espera que en futuras versiones se integren directamente:

- Predicciones del modelo,
- Explicaciones locales (SHAP),
- Intervalos de confianza por marca.

Esto permitirá ampliar el valor estratégico del sistema para departamentos como Tasación, Remarketing y Ventas.

5.6. Resumen del capítulo

En este capítulo se han presentado los resultados obtenidos tras la implementación del sistema de tasación de vehículos de ocasión. Los análisis han demostrado:

- La superioridad de `CatBoost` frente a otros modelos
- La eficacia de los intervalos de confianza optimizados
- La coherencia de las explicaciones locales y globales
- La estabilidad del endpoint en entornos productivos
- El potencial impacto en las herramientas corporativas.

Los resultados confirman la viabilidad técnica y empresarial del sistema, sentando las bases para su explotación completa en entornos reales.

Capítulo 6

Manual técnico y operativa del sistema

Este capítulo presenta la estructura operativa y técnica del sistema implementado, describiendo su arquitectura general, los entornos empleados, los flujos automatizados y las dependencias necesarias para su funcionamiento. A diferencia de los capítulos previos, centrados en el diseño metodológico y en la evaluación del modelo, el presente apartado documenta los aspectos prácticos relacionados con su ejecución, mantenimiento y reproducibilidad.

Se exponen los componentes fundamentales del sistema y sus interacciones, así como los entornos de desarrollo y ejecución empleados, incluyendo la plataforma Databricks y los clústeres asociados a cada flujo. Asimismo, se detalla el proceso de versionado, los mecanismos de despliegue y el modo en que la solución se integra con los sistemas corporativos actuales, sin abordar la capa de presentación externa, al no formar parte del alcance de este trabajo.

El objetivo de este capítulo es ofrecer una guía técnica clara y estructurada para facilitar la comprensión, mantenimiento y posible evolución futura del sistema desarrollado, dentro de un contexto empresarial real y sobre una base tecnológica reproducible.

6.1. Arquitectura general del sistema

El sistema desarrollado se estructura en una arquitectura modular orientada al procesamiento automatizado de datos del mercado automovilístico, la construcción periódica de modelos de predicción de precios y su despliegue como servicio consumible desde plataformas corporativas. Dicha arquitectura integra distintos flujos funcionales, agrupados en tres bloques principales: adquisición y transformación de datos (ETL), modelado y despliegue del sistema (MLOps), e integración con entornos de consumo externos (exportación y visualización).

El flujo ETL se encarga de la recolección diaria de datos de múltiples fuentes web mediante tareas de scraping, su validación, limpieza y consolidación en una base de datos estructurada siguiendo la arquitectura *medallion* (Bronze, Silver, Golden). Esta transformación permite

disponer de datos consistentes, normalizados y actualizados, adecuados para el entrenamiento de modelos predictivos.

Sobre la base de datos depurada, el flujo MLOps ejecuta de forma programada la construcción del modelo, su validación cruzada, registro en el catálogo interno y despliegue en forma de *serving endpoint* dentro de la plataforma Databricks. Este flujo se ejecuta con menor frecuencia que el ETL, atendiendo a criterios de coste computacional y estabilidad del mercado.

Finalmente, los resultados del sistema son exportados a una base de datos externa PostgreSQL mediante una conexión JDBC segura. Este entorno, denominado *PlatinumZone*, sirve como repositorio corporativo desde el cual se alimentan las herramientas de visualización utilizadas por los distintos departamentos de la empresa.

La Figura 6.1 representa la arquitectura del sistema siguiendo el modelo C4 en su nivel de contenedores. Se distinguen dos flujos principales orquestados dentro de la plataforma Databricks: el flujo ETL (`flow_123_ETL_volta`), responsable de la extracción y transformación de datos desde fuentes externas, y el flujo MLOps (`flow_123_MLOPS_volta`), encargado del entrenamiento, validación y despliegue del modelo predictivo. Los resultados generados se exportan a una base de datos externa PostgreSQL, desde la cual son consumidos por una aplicación corporativa interna (`DataDag`) para su visualización por parte de los distintos departamentos empresariales.

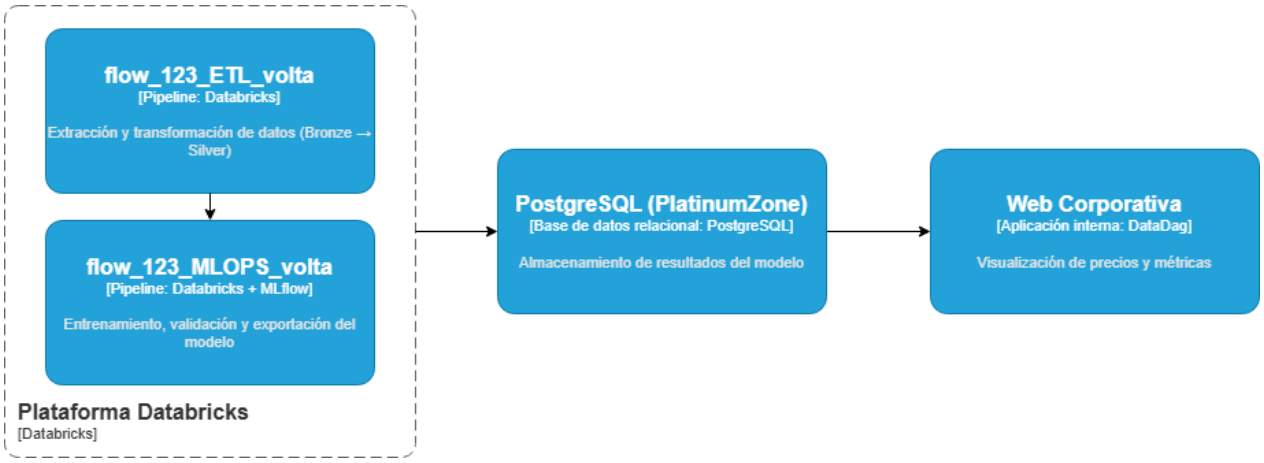

Ilustración 6.1: Arquitectura del sistema de tasación: contenedores y relaciones operativas.

6.2. Entornos de desarrollo y ejecución

El sistema se desarrolla y ejecuta íntegramente sobre la plataforma Databricks, donde se han definido dos flujos de trabajo independientes, correspondientes al tratamiento de datos (ETL) y al ciclo de vida del modelo (MLOps). Cada flujo emplea una configuración de clúster distinta en función de sus necesidades de persistencia, dependencias y ejecución programada.

6.2.1. Flujos de ejecución

Ambos flujos forman parte de la arquitectura global del sistema, complementándose en el ciclo de vida completo de datos y modelo. Aunque se ejecutan de manera independiente y con infraestructuras distintas, su integración garantiza la coherencia y actualización constante tanto del entorno de datos como del modelo predictivo. Por ello, se describen por separado, dada la diferencia de sus frecuencias, objetivos y requisitos técnicos.

Flujo ETL

El flujo `flow_123_ETL_volta` se ejecuta diariamente a las 04:00 (UTC+1) y comprende las tareas de extracción de datos web mediante técnicas de scraping, su almacenamiento inicial en el área *Bronze*, la transformación y validación hacia *Silver*, y finalmente la exportación de resultados a la base de datos externa. Este flujo se ejecuta sobre un clúster persistente denominado `123_ETL_CLUSTER`, basado en el entorno *Databricks Runtime 13.3 LTS* con Apache Spark 3.4, y configurado con instancias `Standard_D4ds_v5` (1–2 nodos).

Dado que el scraping requiere la instalación manual de *geckodriver* y del navegador Firefox en el sistema de archivos, así como el uso de privilegios elevados para la gestión de drivers y dependencias como `selenium`, este clúster no puede ser de tipo efímero. Además, *geckodriver* no permite múltiples ejecuciones simultáneas en un mismo entorno, lo que obliga a que su instalación se realice de forma independiente en cada una de las tareas de scraping. Esta instalación se lleva a cabo mediante un notebook denominado `SetUpScraper`, invocado explícitamente en cada instancia de ejecución donde se requiere acceso a un navegador.

Flujo MLOps

El flujo `flow_123_MLOPS_volta` se ejecuta de forma mensual, el día 7 a las 18:00 (UTC+1), y comprende las tareas de construcción de la variable objetivo, preparación de diccionarios y patrones, entrenamiento del modelo, validación, registro y despliegue mediante MLflow, así como la exportación de resultados a la base de datos externa.

Este flujo emplea un clúster de tipo efímero (*job cluster*) denominado `cluster_123_MLOPS_volta`, configurado como nodo único con la instancia `Standard_DS4_v2` y entorno *Databricks Runtime 15.4 LTS* (Apache Spark 3.5, Scala 2.12).

6.2.2. Dependencias y entorno base

Ambos flujos se ejecutan sobre el entorno base de Databricks, sin necesidad de entornos virtuales personalizados. Las dependencias principales se instalan a través de las herramientas estándar del clúster, y se detallan a continuación:

- `catboost`, `mlflow`, `shap`

- `selenium`, `webdriver-manager`, `fake_useragent`, `bs4`
- `sqlalchemy`, `google-api-python-client`
- `xlrd`, `openpyxl`, `unidecode`

La gestión de credenciales y parámetros sensibles se realiza mediante el sistema de *Secrets* de Databricks, donde se almacenan los tokens necesarios para la autenticación del endpoint y la conexión con sistemas externos como PostgreSQL o APIs de terceros. No se ha requerido el uso de ficheros de configuración tipo `.env` o `config.json` en el presente proyecto.

6.2.3. Integración con herramientas externas

El entorno de trabajo se complementa con herramientas externas como Azure DevOps para la gestión del repositorio, PostgreSQL como destino de los datos exportados, y la plataforma interna `DataDag` para la visualización de resultados.

6.3. Repositorio y control de versiones

El control de versiones del sistema se gestiona a través de un repositorio privado en Azure DevOps, cuya estructura está alineada con la división funcional del sistema. El desarrollo se ha organizado utilizando dos ramas diferenciadas: la rama `develop`, desplegada en el entorno `ws_dag_dev`, y la rama `main`, utilizada para el entorno de producción `ws_dag_pro`. Esta separación permite mantener un entorno de pruebas controlado y garantizar la estabilidad del código promovido a producción.

La estructura del repositorio se basa en una separación modular por flujos, con las siguientes carpetas principales:

- `123_volta_etl`: contiene los notebooks del flujo ETL, incluyendo las tareas de scraping (`AutoscoutTask`, `MotorEsTask`, etc.), transformación (`Bronze_to_Silver`) y exportación a PostgreSQL (`Volta_to_PZ_jdbc`). También incorpora carpetas auxiliares como `Utils` y `xls`.
- `123_volta_mlops`: incluye notebooks y scripts relacionados con el flujo MLOps, como `GetDictionaryTask`, `PatternsBuilderTask`, `ModelTask` y `ModelTest`, así como los ficheros resultantes del entrenamiento del modelo (`bags.pickle`, `power.pickle`, `version.pickle`, etc.).
- `README.md`: documento de cabecera con información general del repositorio.

No se ha implementado un sistema de integración continua automatizada, dado que el flujo de trabajo se basa en la ejecución periódica de tareas programadas dentro de Databricks. Sin embargo, la separación entre entornos, el uso de ramas y la estructuración del código garantizan la trazabilidad, mantenibilidad y extensibilidad del sistema a futuro.

6.4. Flujos de ejecución

El sistema se articula mediante dos flujos principales programados en Databricks Workflows: uno orientado a la ingesta y preparación de datos (ETL) y otro enfocado a la construcción, despliegue y publicación del modelo predictivo (MLOps). Ambos flujos están orquestados a través de pipelines compuestos por tareas secuenciales, cada una de las cuales ejecuta notebooks específicos. Estas ejecuciones están sujetas a una planificación temporal periódica, permitiendo la automatización completa del proceso.

6.4.1. Flujo ETL: `flow_123_ETL_volta`

El flujo ETL se ejecuta diariamente a las 04:00 (hora local) y tiene como objetivo la obtención y transformación de datos procedentes de fuentes web. Se compone de las siguientes tareas:

- **Get_Tasks**: inicializa la lista de scraping a partir de metainformación de interés para el día actual.
- **Scraper_iteration**: ejecuta la iteración sobre los portales web de referencia (AutoScout24, Motor.es, Coches.net, entre otros), utilizando técnicas de scraping con Selenium y almacenando los resultados en formato *Bronze*.
- **Bronze_to_Silver**: transforma los datos brutos obtenidos y los normaliza hacia el área *Silver*, aplicando procesos de limpieza, validación y enriquecimiento.
- **VOLTA_to_Pz**: exporta las tablas clave (`tb_123_volta_etl_mlops_markets_canary` y `tb_123_volta_etl_mlops_lista_precios`) a una base de datos externa PostgreSQL (PlatinumZone) mediante conexión JDBC.

Todas las tareas se ejecutan en el clúster persistente `123_ETL_CLUSTER` por los motivos descritos en la Sección 6.2.

6.4.2. Flujo MLOps: `flow_123_MLOPS_volta`

Este flujo se ejecuta mensualmente, el día 7 a las 18:00 (hora local), y se encarga de entrenar y desplegar un nuevo modelo basado en la información consolidada. Las tareas que lo componen son las siguientes:

- **NewVehiclePrice**: prepara el conjunto de datos de precios base a partir de fuentes externas (como el listado de precios de Hacienda), aplicando limpieza, normalización y segmentación.
- **GetDictionary**: construye un diccionario de equivalencias y características relevantes por marca y modelo.

- **PatternsBuilder**: extrae patrones textuales y bolsas de palabras a partir de las descripciones del vehículo para su posterior uso en el modelo.
- **ModelAndServiceBuilder**: entrena el modelo utilizando CatBoost, lo registra en MLflow y despliega un endpoint funcional para servir predicciones.
- **ModelTest**: valida la funcionalidad del endpoint desplegado y exporta los resultados al entorno de consumo.

Este flujo se ejecuta en un clúster efímero configurado específicamente para cada ejecución, adaptado a las necesidades de entrenamiento y evaluación del modelo.

6.4.3. Automatización

Ambos flujos están programados de forma independiente, sin que exista una dependencia técnica entre ellos. El flujo MLOps se ejecuta de manera autónoma con su propia planificación, independientemente del éxito o fallo de las ejecuciones del flujo ETL.

6.5. Consumo del modelo (inferencia)

El modelo predictivo desarrollado se expone mediante un endpoint REST a través del sistema de *MLflow Model Serving* integrado en Databricks. Una vez completado el proceso de entrenamiento y validación, el modelo se registra en el catálogo y se publica como servicio invocable, accesible a través de una URL del tipo:

```
https://<workspace>.databricks.com/serving-endpoints/<model-name>/invocations
```

El acceso al endpoint está protegido mediante un token personal, gestionado como secreto en Azure Key Vault e integrado con el sistema de gestión de credenciales de Databricks.

6.5.1. Estructura de la petición

El modelo espera recibir un conjunto de variables de entrada correspondientes a las características del vehículo, ya procesadas y transformadas. La Tabla 6.1 resume las variables requeridas:

Cuadro 6.1: Variables de entrada esperadas por el endpoint de inferencia

Variable	Tipo
Brand	string
Fuel	string
Gearbox	string
CV	double
Segment	string
Kms	double
Age	double
Sale	string
Shortage	double
PriceMed	double

Las variables `PriceMed` y `Shortage` no son generadas por el modelo, sino que deben ser calculadas previamente por una API intermedia, que actúa como capa de integración entre el sistema de inferencia y la plataforma web corporativa.

6.5.2. Consumo a través de API externa

El endpoint no es consumido directamente por la interfaz web de la empresa, sino por una API alojada en Azure App Service, que forma parte de la infraestructura interna. Esta API intermedia se encarga de preparar las peticiones, incluir los parámetros adicionales requeridos y enviar los datos al endpoint del modelo. Posteriormente, remite la predicción resultante a otra API encargada de alimentar la plataforma `DataDag`, desde la cual los usuarios visualizan los resultados.

El sistema desarrollado en este trabajo proporciona el modelo como un servicio inferencial en línea, pero no almacena las predicciones generadas en ninguna base de datos. Por tanto, la respuesta se utiliza de forma efímera y no se guarda en PostgreSQL ni en otro sistema persistente.

6.5.3. Ejemplo de petición

El siguiente fragmento muestra un ejemplo simplificado de cómo se realiza la llamada al endpoint mediante una petición HTTP desde Python:

```
url = f"https://<workspace>/serving-endpoints/<model-name>/invocations"
headers = {
    "Authorization": f"Bearer {token}",
    "Content-Type": "application/json"
}
response = requests.post(url, headers=headers, data=json.dumps(payload))
```

6.6. Visualización en la web corporativa

La plataforma web corporativa desde la que se visualizan los resultados del sistema desarrollado se denomina `DataDag`. Esta interfaz no forma parte del presente proyecto, pero consume datos generados y exportados por el sistema descrito en los capítulos anteriores.

La visualización se basa en consultas a una base de datos externa PostgreSQL (denominada `PlatinumZone`), a la que el sistema exporta periódicamente las tablas clave generadas en los flujos ETL y MLOps. Entre las tablas disponibles se encuentran:

- `tb_123_volta_etl_mlops_markets_canary`: contiene información agregada del mercado regional actualizada diariamente, derivada del scraping y normalización de anuncios.
- `tb_123_volta_etl_mlops_lista_precios`: recoge los precios oficiales o estimados de vehículos nuevos utilizados como referencia en el proceso de entrenamiento.
- `tb_123_volta_etl_mlops_confidence_intervals`: representa los intervalos de confianza generados a partir del modelo entrenado, clasificados por marca.

Desde la plataforma web se ofrece a los usuarios la posibilidad de aplicar filtros por marca, modelo, provincia, tipo de combustible y segmento, así como consultar estadísticas descriptivas (media, mediana, desviación estándar) y comparativas entre distintos grupos de vehículos.

Funcionalidades actuales y futuras

En su estado actual, el sistema desarrollado permite:

- **Estimar el precio de vehículos de ocasión**, en función de sus características técnicas y comerciales, a través de un modelo de predicción entrenado específicamente para el mercado canario.
- Visualizar los datos de mercado actual en Canarias y su evolución temporal.
- Consultar precios medios por modelo y segmento, facilitando comparativas rápidas entre distintas configuraciones de vehículos.
- Acceder a listas de precios estructuradas según el catálogo oficial y los datos de Hacienda, que sirven de referencia para análisis comerciales o fiscales.

Está previsto que en futuras versiones se integren funcionalidades adicionales, entre las que destacan:

- Integración de un simulador interactivo de valoración de vehículos, alimentado en tiempo real por el modelo desplegado.
- Visualización de gráficos de `price-decay` por modelo y año de matriculación, para analizar la depreciación de los vehículos en el tiempo.

- Incorporación de análisis explicativos basados en `SHAP values`, orientados a la interpretación de cada predicción de manera local.

Estas funcionalidades podrán integrarse en la plataforma web mediante nuevas tablas o endpoints consumibles desde la infraestructura ya desplegada.

6.7. Dependencias y ejecución

El sistema desarrollado se apoya en una serie de dependencias técnicas y de ejecución que deben garantizarse para su funcionamiento correcto. Estas abarcan desde requisitos de entorno hasta la instalación de drivers, la gestión de secretos y la ejecución secuencial de notebooks dentro de cada flujo.

6.7.1. Requisitos de entorno

La plataforma de ejecución es Databricks, sin configuración adicional de entornos virtuales ni contenedores externos. Todos los notebooks utilizan el entorno base de Databricks Runtime en sus respectivas versiones (véase Sección 6.2). Las dependencias necesarias se instalan mediante comandos estándar de `pip`, que incluyen, entre otras, las siguientes librerías:

```
catboost sqlalchemy unidecode fake_useragent bs4
google-api-python-client xlrd openpyxl shap
```

6.7.2. Gestión de secretos

El acceso a recursos sensibles, como el token de autorización para el endpoint de inferencia o las credenciales de conexión a la base de datos PostgreSQL, se gestiona a través de Azure Key Vault, integrado con el sistema de *Databricks Secrets*. Esto permite inyectar las credenciales de forma segura en los notebooks sin exponerlas en código.

6.7.3. Dependencias específicas del scraping

Dado que el proceso de scraping requiere la ejecución de un navegador controlado por Selenium, es necesario instalar manualmente los drivers de Firefox y *geckodriver*, así como las librerías del sistema necesarias para su ejecución. Esta instalación se realiza mediante el notebook `SetUpScraper`, que se ejecuta al inicio de cada tarea de scraping, ya que los drivers no pueden ser utilizados por múltiples instancias simultáneamente. Esta es una de las razones por las que el flujo ETL se ejecuta sobre un clúster persistente y no efímero.

6.7.4. Ejecución secuencial y notebooks críticos

Aunque los flujos ETL y MLOps son independientes entre sí y se ejecutan en horarios distintos, la ejecución interna de cada uno sigue una estructura secuencial definida. Cada tarea dentro del pipeline tiene una función relevante en el sistema, y su correcto funcionamiento es necesario para que los resultados finales sean válidos.

En el caso del flujo ETL, aunque los scrapers que componen la tarea `Scraper_iteration` se ejecutan en paralelo y de forma independiente, el fallo de uno de ellos no interrumpe la ejecución global del flujo. No obstante, sí se requiere que al menos uno de los scrapers finalice correctamente para que las etapas posteriores de transformación y exportación tengan datos válidos que procesar.

En cambio, en el flujo MLOps, la ejecución de cada notebook es estrictamente necesaria. Si no se preparan correctamente las variables, patrones y diccionarios, el modelo no puede entrenarse ni desplegarse de forma adecuada. El notebook `ModelAndServiceBuilder` actúa como orquestador final y depende directamente de la salida de las etapas previas.

En resumen, la robustez del sistema requiere la ejecución completa y ordenada de cada workflow, permitiendo tolerancia parcial en el proceso de scraping, pero no en las fases críticas de transformación, entrenamiento o despliegue.

6.8. Resumen del capítulo

En este capítulo se ha descrito la arquitectura técnica y operativa del sistema, detallando los flujos ETL y MLOps, su planificación y los entornos de ejecución configurados en Databricks. También se han expuesto las herramientas y dependencias empleadas, así como el modo en que los datos transformados se integran en la plataforma corporativa para su visualización y consumo. Este conjunto de elementos constituye la base técnica sobre la que se sustentan las funcionalidades actuales y futuras del sistema.

Capítulo 7

Conclusiones y trabajo futuro

El presente capítulo tiene como finalidad sintetizar los principales aprendizajes, logros y limitaciones alcanzados durante el desarrollo del sistema de tasación de vehículos de ocasión para *Domingo Alonso Group*, basado en modelos de precios hedónicos e implementado íntegramente en la plataforma Databricks. Este objetivo, enunciado ya desde el título del trabajo, ha guiado todas las fases del proyecto, desde la adquisición de datos hasta el despliegue del modelo como servicio de inferencia.

A lo largo del trabajo se han abordado con éxito los distintos retos técnicos y metodológicos asociados a la construcción de una solución predictiva robusta, reproducible y alineada con el entorno tecnológico de la empresa. El sistema actualmente se encuentra finalizado y operativo, con capacidad de ofrecer estimaciones automáticas y contextualizadas del valor de un vehículo, aunque su integración en la plataforma web empresarial está aún pendiente de finalización.

Las decisiones tomadas en cuanto a modelado, preprocesado de datos, diseño del flujo MLOps y despliegue del servicio han permitido construir una arquitectura escalable, fácilmente mantenible y compatible con los procesos existentes en la organización. Este capítulo recoge, por tanto, una valoración global del grado de cumplimiento de los objetivos, el impacto técnico y organizativo alcanzado, las limitaciones detectadas durante el desarrollo y las líneas de trabajo que permitirán extender la utilidad del sistema en el futuro.

7.1. Valor aportado a la empresa

Desde una perspectiva práctica, el sistema desarrollado supone una mejora significativa respecto a los procesos tradicionales de tasación utilizados en *Domingo Alonso Group*, los cuales dependían en gran medida de la experiencia del tasador o de herramientas externas poco adaptadas al contexto canario.

Entre los principales beneficios técnicos y organizativos que aporta el sistema, destacan los siguientes:

- **Automatización del proceso de tasación:** permite generar estimaciones de precio de forma automática, objetiva y reproducible, eliminando sesgos humanos y reduciendo significativamente los tiempos de valoración.
- **Estandarización de criterios:** el uso de un modelo único, validado y alineado con datos históricos garantiza coherencia en las valoraciones realizadas por diferentes áreas de la empresa.
- **Alineación con tecnologías existentes:** el sistema ha sido diseñado para integrarse directamente en el ecosistema tecnológico actual de la compañía, empleando Databricks, PostgreSQL y arquitectura *medallion*, lo que facilita su adopción y mantenimiento.
- **Preparación para el despliegue final:** aunque aún no se ha completado la integración con la plataforma web interna, la infraestructura está lista para su consumo en entornos reales, lo que permitirá extender su uso a departamentos como compras, ventas, postventa o remarketing.

En conjunto, el sistema no solo ofrece valor desde el punto de vista técnico, sino que actúa como catalizador para la transformación digital de los procesos comerciales ligados a la gestión de vehículos de ocasión.

7.2. Limitaciones detectadas

Aunque el sistema desarrollado presenta un alto grado de madurez técnica y está plenamente operativo, existen ciertas limitaciones y dependencias que conviene destacar para contextualizar su funcionamiento actual y futuro.

Una de las principales limitaciones identificadas durante la validación del modelo fue su comportamiento inestable en segmentos poco representados del mercado, como marcas premium o modelos con baja frecuencia de aparición. Este problema se abordó mediante la introducción de la variable *Shortage*, definida como la inversa de la frecuencia relativa del par `Brand-Model`. Tal como se muestra en el análisis por marca incluido en la memoria, esta variable permitió reducir la amplitud excesiva de los intervalos de confianza y mejorar la precisión del modelo en escenarios con escasez de datos. No obstante, sigue existiendo una mayor incertidumbre en las predicciones para segmentos minoritarios, lo cual es inherente a la naturaleza de los datos disponibles.

Por otro lado, el endpoint de inferencia desplegado en Databricks está diseñado para ser consumido por una API intermedia que se encarga de generar variables necesarias como *PriceMed* y *Shortage*, sin requerir que el usuario final proporcione directamente dicha información. Esta arquitectura modular —aunque adecuada desde el punto de vista del diseño— introduce una dependencia crítica del preprocesamiento previo, lo cual podría dificultar la reutilización del modelo en entornos externos si no se cuenta con una lógica intermedia equivalente. Aunque el sistema ha sido desarrollado como parte de un flujo cerrado dentro del entorno de *Domingo Alonso Group*, conviene tener presente esta necesidad de cálculo previo para su correcta explotación.

Finalmente, si bien la integración del modelo en la web interna de la empresa no formaba parte directa del alcance del presente Trabajo de Fin de Grado, sí se ha confirmado que dicha conexión ya ha sido implementada. Actualmente, la interfaz web es capaz de consumir el endpoint y mostrar tanto la predicción puntual como el intervalo de confianza asociado. No obstante, dado que esta funcionalidad se encuentra aún en fase de validación y pruebas internas, no ha sido evaluada formalmente como parte del sistema desarrollado en esta memoria.

7.3. Líneas de trabajo futuro

A pesar de que el sistema se encuentra actualmente operativo y validado en entornos reales, existen diversas líneas de trabajo que podrían explorarse para ampliar su funcionalidad, mejorar su precisión y facilitar su adopción en procesos de decisión más complejos.

7.3.1. Integración completa con la web y explicabilidad

Uno de los desarrollos previstos en el entorno empresarial es la mejora de la integración del modelo en la interfaz web interna. Actualmente, el sistema ya permite mostrar al usuario el precio estimado y su intervalo de confianza; no obstante, se plantea incorporar nuevos módulos que enriquezcan la experiencia y el valor interpretativo del sistema. Entre ellos, destaca la visualización del *price-decay*, una estimación de cómo se espera que evolucione el valor del vehículo en los próximos meses en función de su antigüedad y kilometraje futuros. Asimismo, se prevé incluir un análisis basado en valores SHAP que muestre cómo ha influido cada variable (edad, CV, segmento, etc.) en la predicción generada, aportando transparencia y capacidad de auditoría al sistema.

7.3.2. Exploración de nuevas estrategias de modelado

Aunque en algunos contextos podría parecer razonable entrenar modelos especializados por marca o segmento, esta estrategia resulta inviable desde un punto de vista operativo y económico. Supondría mantener decenas de modelos independientes, con ciclos de vida, entrenamiento y validación separados, lo que incrementaría drásticamente la complejidad técnica y los costes de mantenimiento. Además, se perdería la generalización que ofrece el enfoque actual, que permite extrapolar patrones entre marcas comparables o segmentos similares.

No obstante, sí podrían estudiarse técnicas avanzadas que aporten valor sin comprometer la escalabilidad. Entre ellas, destaca la aplicación de *forecasting* (predicción de series temporales) sobre precios históricos por marca o modelo, lo cual permitiría ajustar el sistema a la estacionalidad del mercado o detectar desviaciones atípicas. Estas técnicas podrían complementarse con un análisis agregado a nivel de clústeres de vehículos similares, utilizando algoritmos de agrupación no supervisada para generar métricas contextuales adicionales.

7.3.3. Retroalimentación y aprendizaje continuo

Otra línea de mejora consiste en incorporar mecanismos de retroalimentación que permitan ajustar el modelo en función de su uso real. Por ejemplo, comparar las predicciones del sistema con los precios finales de venta efectivamente alcanzados en concesionario podría alimentar procesos de reentrenamiento automático o evaluación periódica. Este enfoque, inspirado en técnicas de *learning with feedback*, permitiría adaptar el modelo de forma progresiva al comportamiento real del mercado y a las decisiones empresariales.

7.3.4. Escalabilidad del enfoque

Aunque el sistema ha sido concebido específicamente para la tasación de vehículos de ocasión, su arquitectura modular y su diseño orientado a datos permiten su extensión a otros procesos internos donde se requieran modelos predictivos con lógica MLOps y despliegue automatizado. Si bien no se contempla actualmente su uso en líneas como renting, leasing o motocicletas, el trabajo realizado sienta una base sólida para futuros desarrollos en entornos similares, si así lo requiriese la estrategia tecnológica de la empresa.

7.4. Conclusión final

El sistema de tasación desarrollado en este Trabajo de Fin de Grado representa un ejemplo práctico de cómo la ciencia de datos puede aplicarse con éxito a un problema real del sector automotriz, integrando componentes de análisis, modelado, despliegue y visualización en una arquitectura moderna y escalable.

Más allá de los resultados técnicos obtenidos, el trabajo realizado demuestra la viabilidad de adoptar soluciones automatizadas, basadas en datos, dentro de entornos empresariales complejos. Su alineamiento con las necesidades reales de *Domingo Alonso Group* refuerza su aplicabilidad y potencial de continuidad.

Las futuras líneas de desarrollo permitirán no solo mejorar su precisión y alcance, sino también consolidarlo como una herramienta estratégica para la toma de decisiones en el ámbito de los vehículos de ocasión.

Bibliografía

[1] Chapman, E. A. (2020). *Machine Learning Engineering with MLflow*. O'Reilly Media, Inc.

[2] CRISP-DM Consortium (2000). Crisp-dm 1.0: Step-by-step data mining guide. https://www.semanticscholar.org/paper/CRISP-DM-1.0%3A-Step-by-step-data-mining-guide-Chapman/54bad20bbc7938991bf34f86dde0babfbd2d5a72?utm_source=direct_link.

[3] Databricks (2025). Databricks documentation. https://docs.databricks.com/.

[4] Databricks (2025). The medallion architecture: A modular data architecture for the lakehouse. https://www.databricks.com/glossary/medallion-architecture.

[5] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd edition.

[6] Microsoft (2025). Azure devops documentation. https://learn.microsoft.com/en-us/azure/devops/.

[7] MLflow Team (2025). *MLflow Documentation*. https://mlflow.org/docs/latest/index.html.

[8] PostgreSQL Global Development Group (2025). Postgresql documentation. https://www.postgresql.org/docs/.

[9] Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A., and Gulin, A. (2025). Catboost: gradient boosting with categorical features support. https://catboost.ai/.

[10] Purohit, H. and Purohit, R. (2016). Hedonic pricing model of used cars: A case study. *Indian Journal of Science and Technology*, 9(15).

[11] Rosen, S. (1974). Hedonic prices and implicit markets: Product differentiation in pure competition. *Journal of Political Economy*, 82(1):34–55.

[12] Selenium Project (2025). Selenium documentation. https://www.selenium.dev/documentation/.

[13] Shafer, G. and Vovk, V. (2008). A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9:371–421.

Apéndice A

Visualizaciones Exploratorias

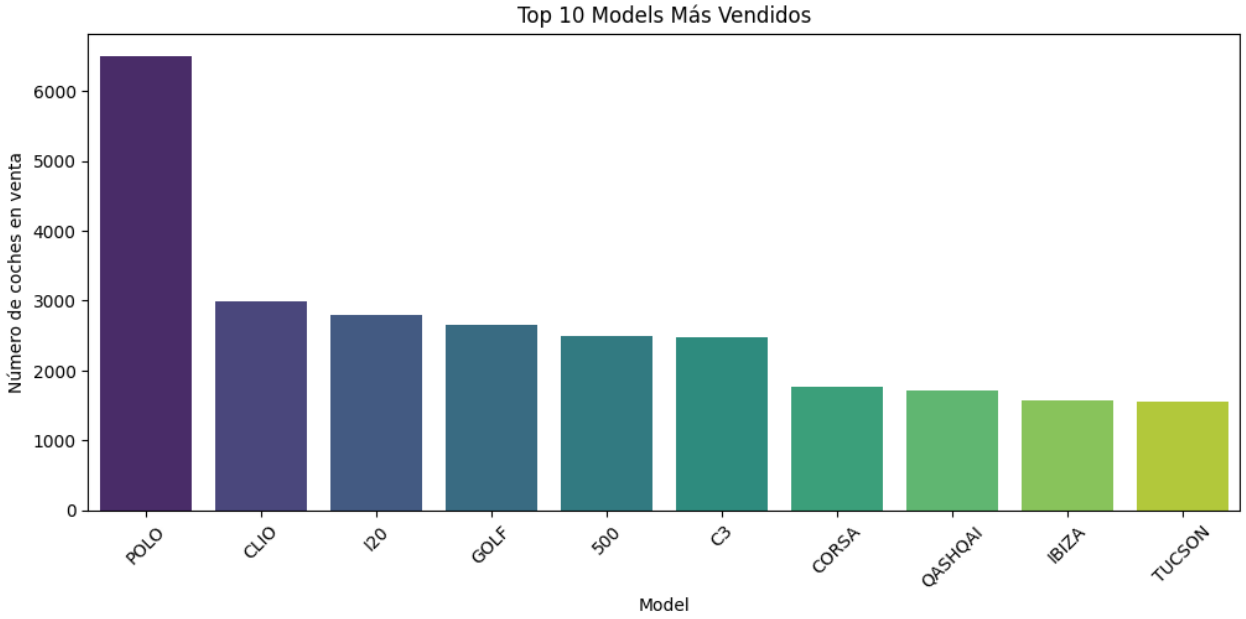

Ilustración A.1: Top 10 modelos de coche más vendidos en el conjunto de datos.

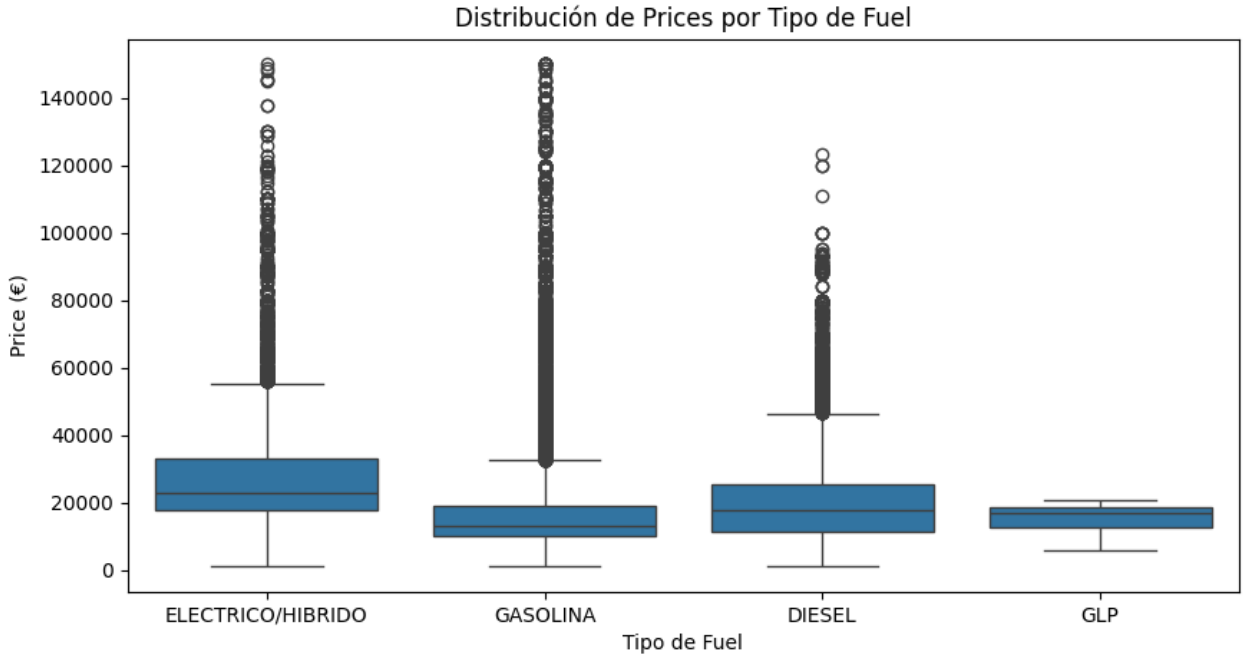

Ilustración A.2: Distribución de precios por tipo de combustible.

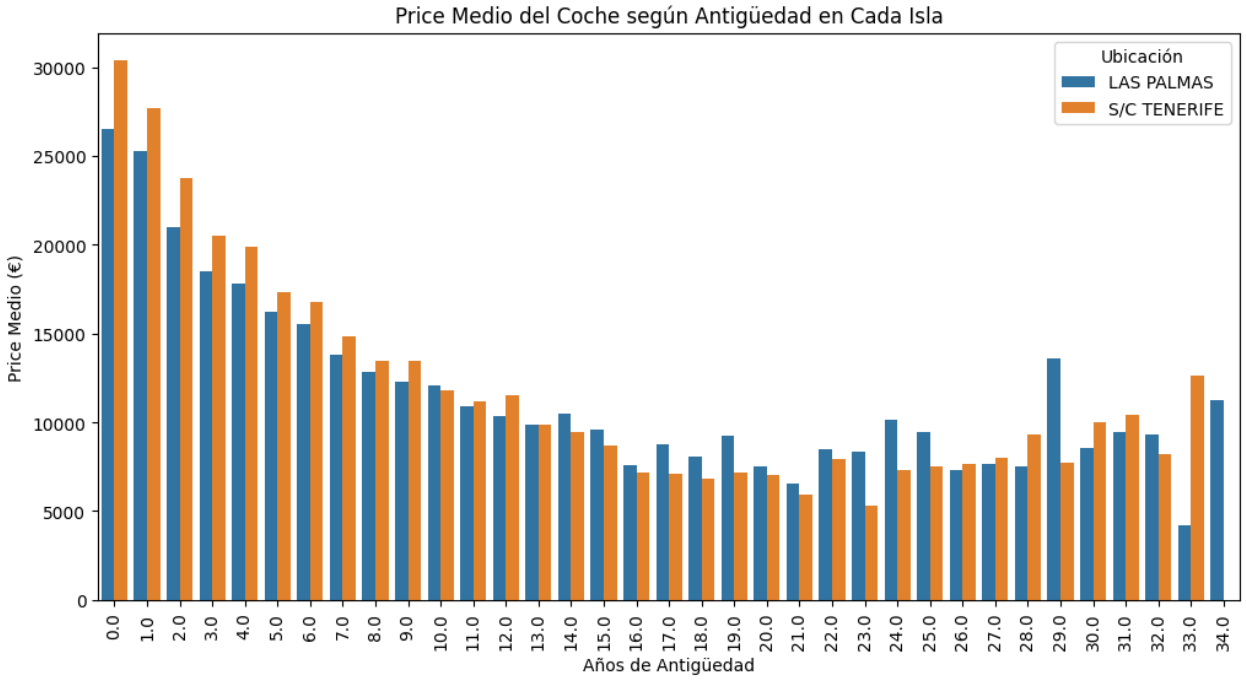

Ilustración A.3: Precio mediano del coche según antigüedad en cada isla.

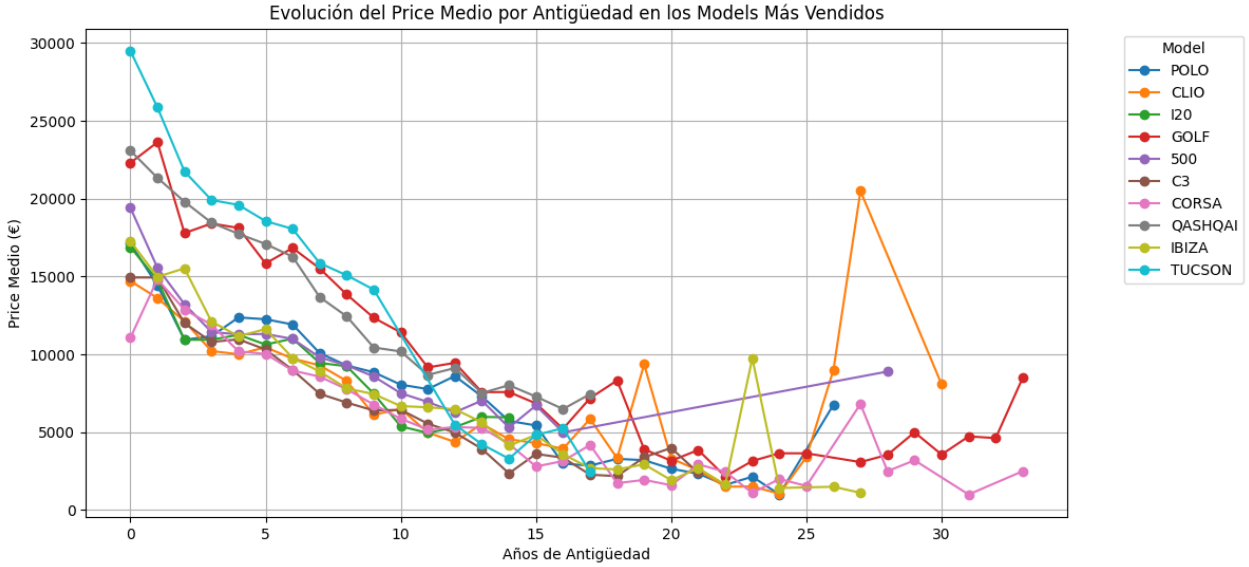

Ilustración A.4: Evolución del precio mediano por antigüedad en los modelos más vendidos.

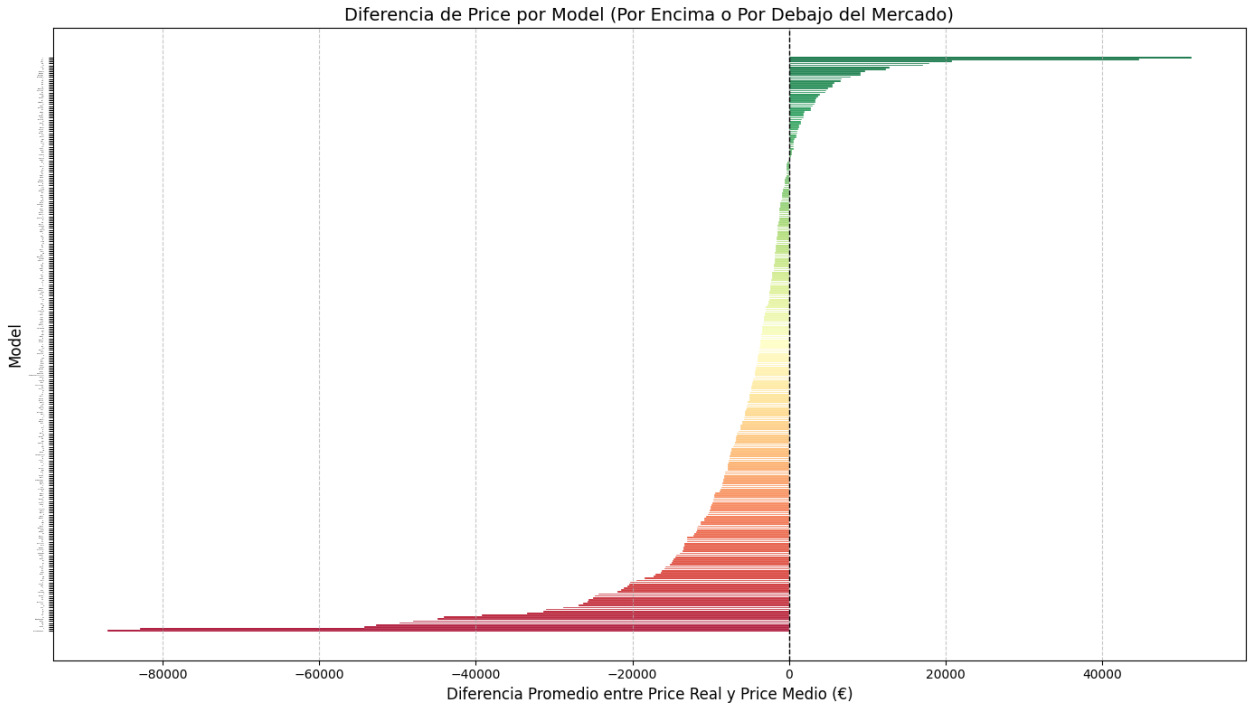

Ilustración A.5: Diferencia de precio mediano por modelo respecto al precio de mercado.

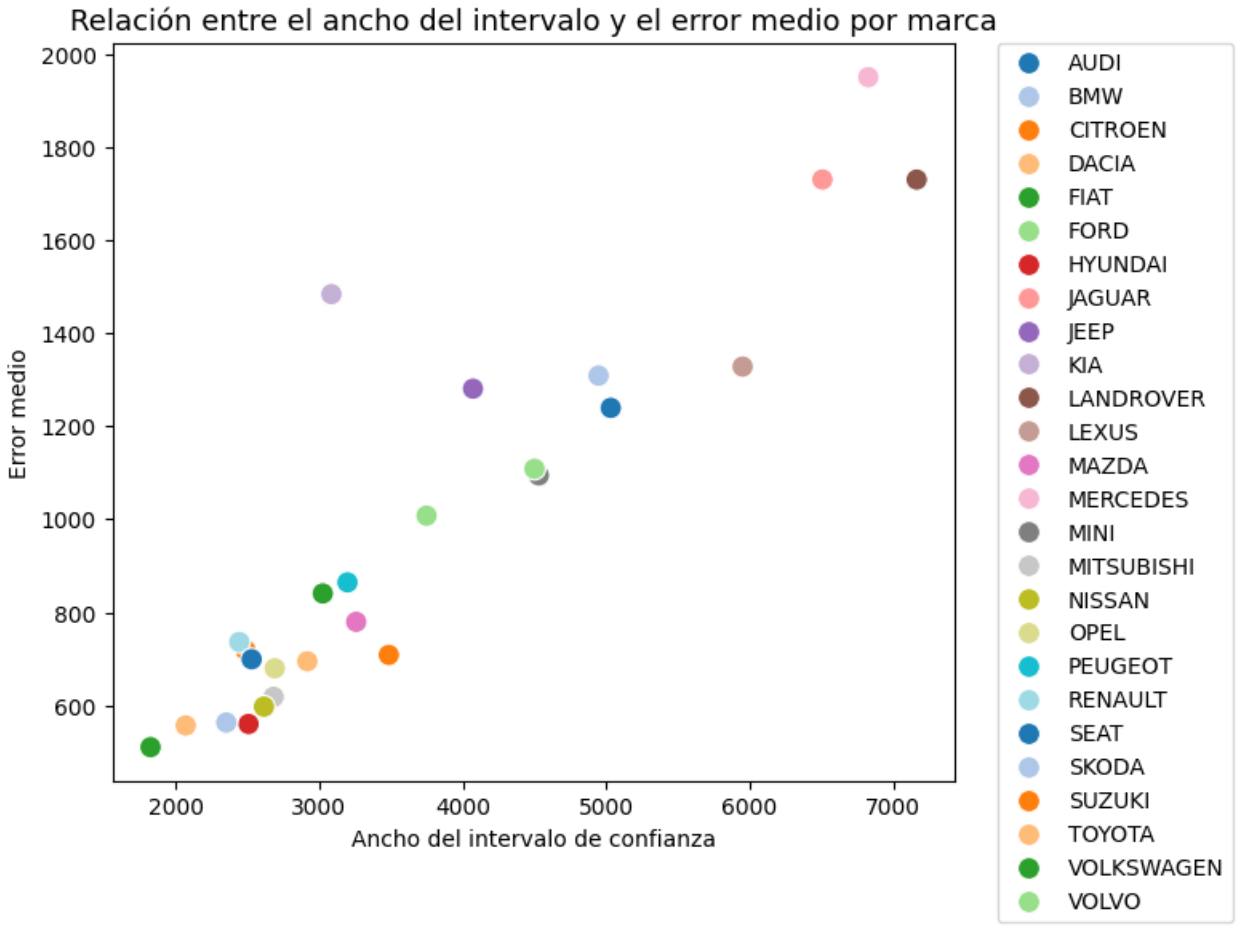

Ilustración A.6: Relación entre el ancho del intervalo de confianza y el error medio por marca.

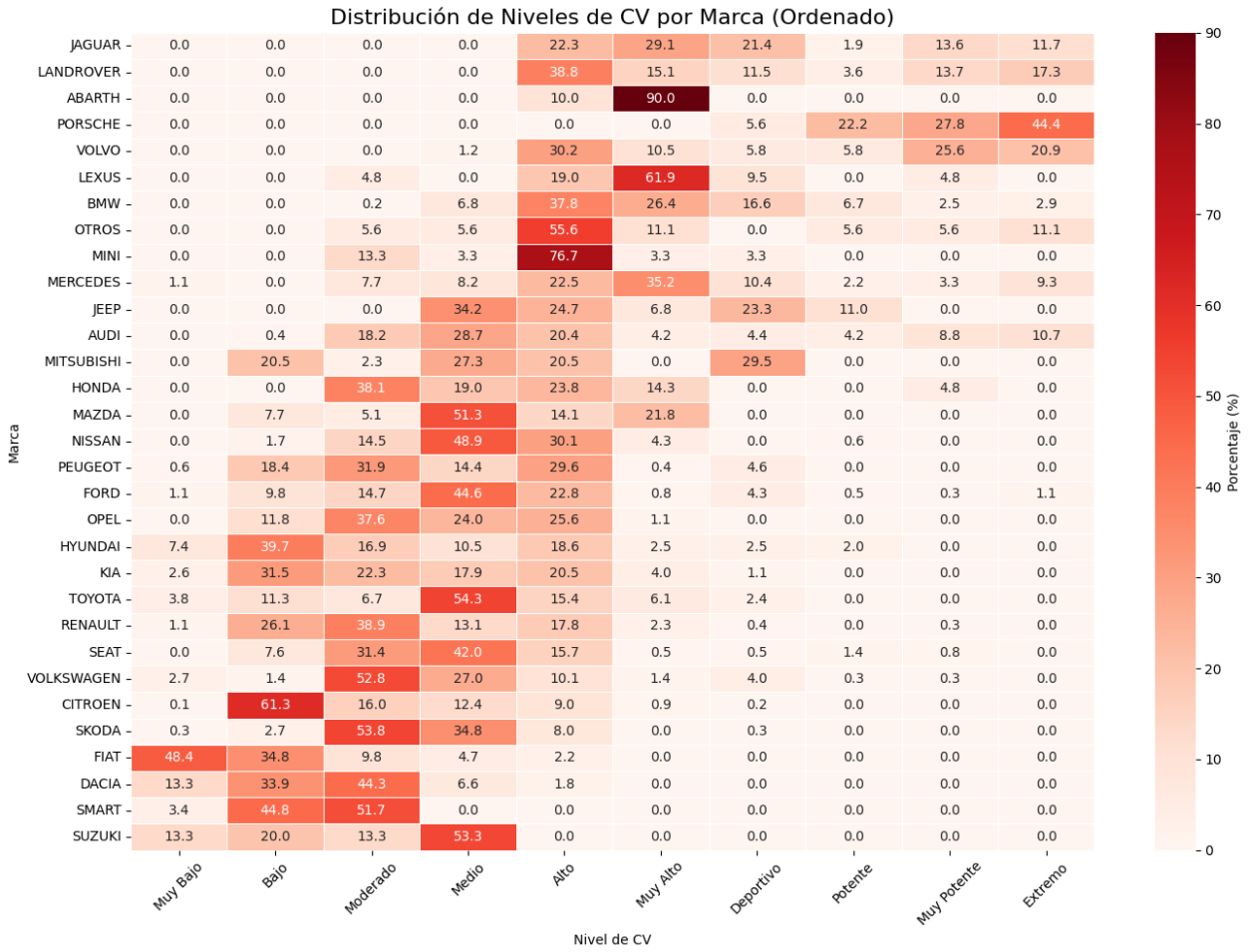

Ilustración A.7: Distribución porcentual de los niveles de caballos de vapor (CV) por marca.

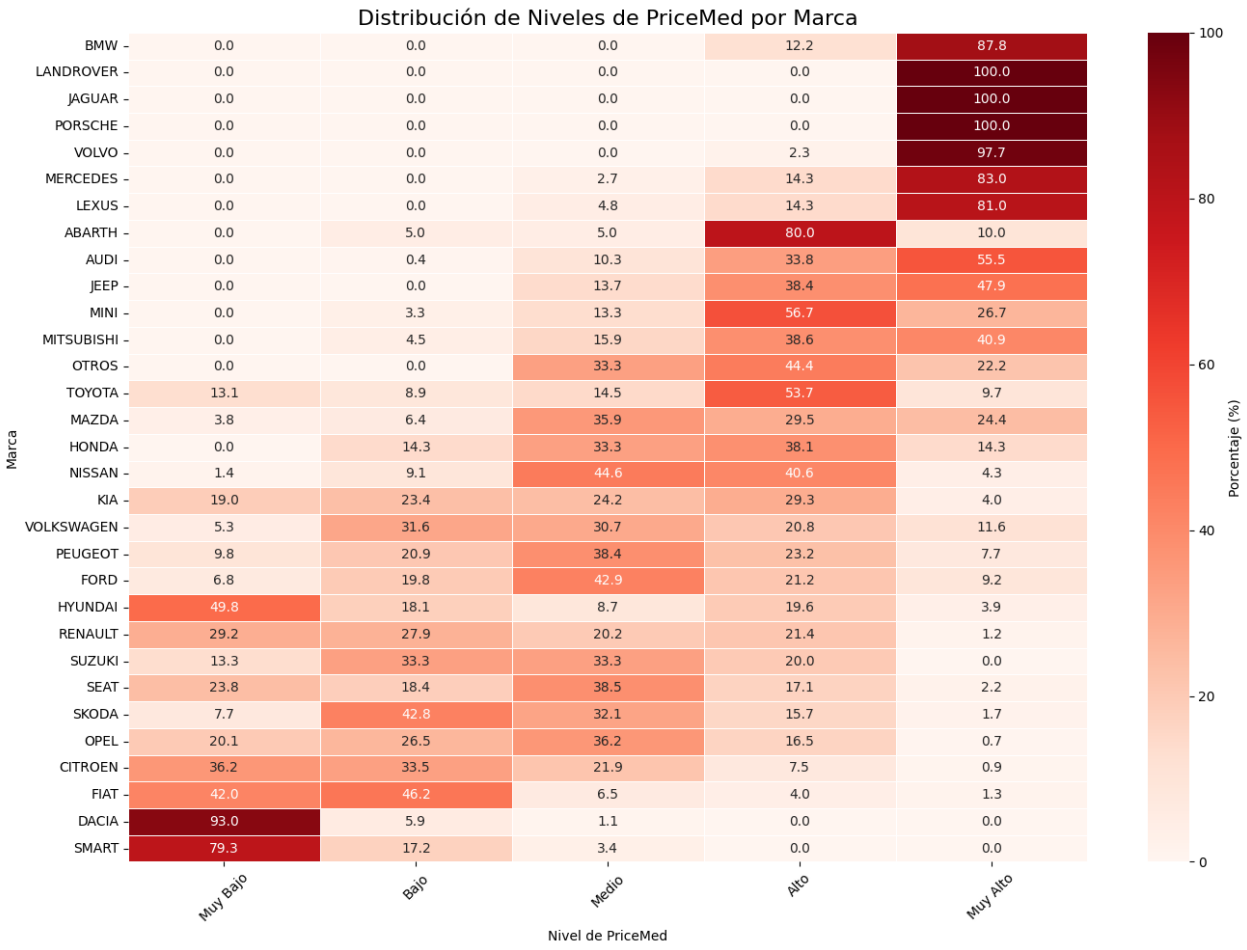

Ilustración A.8: Distribución porcentual de los niveles del precio mediano (PriceMed) por marca.

Apéndice B

Configuración de los flujos en Databricks

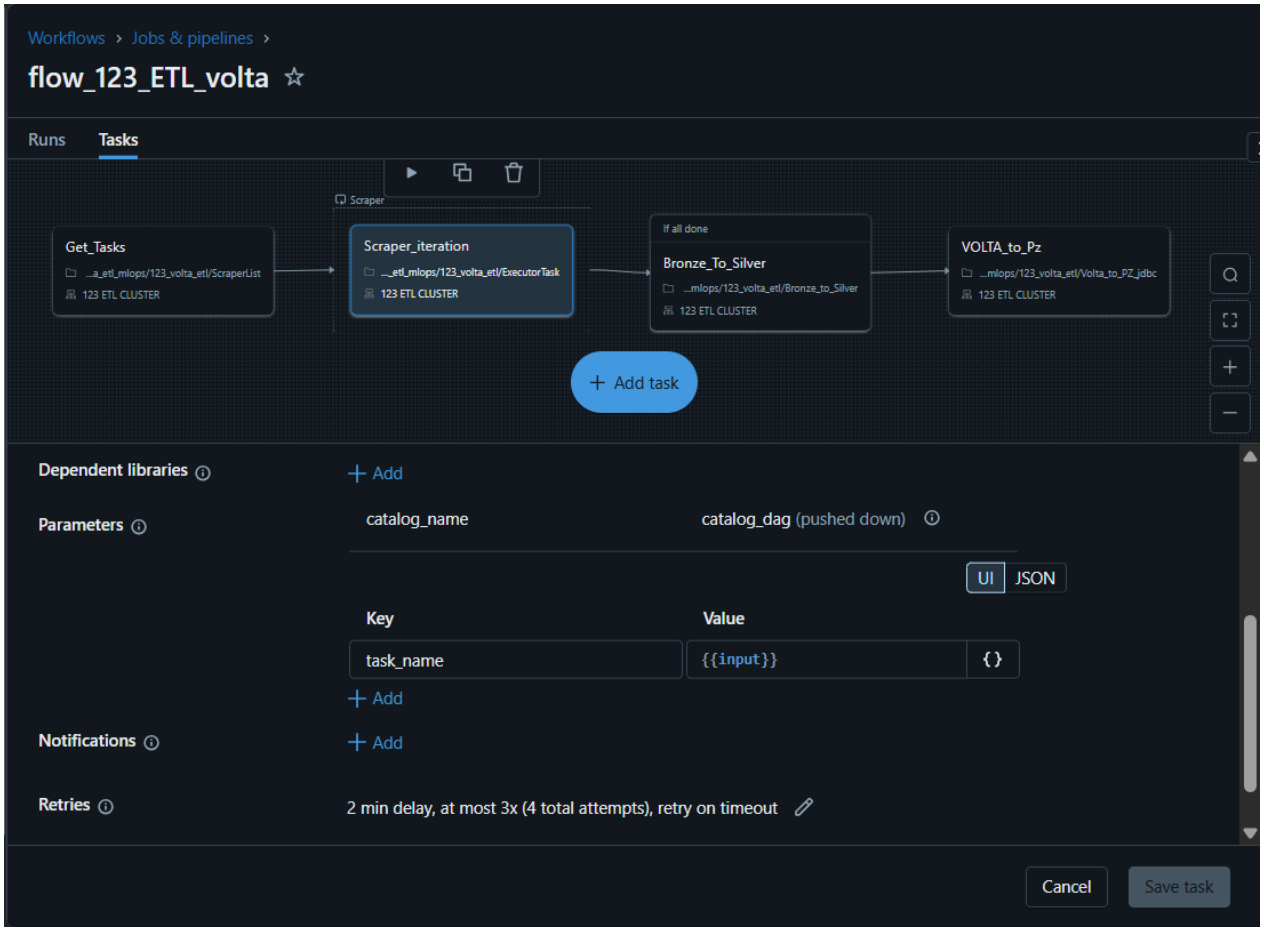

Ilustración B.1: Visualización del flujo ETL principal compuesto por las tareas: *Get_Tasks*, *Scraper_iteration*, *Bronze_To_Silver* y *VOLTA_to_Pz*. Se observa también la política de reintentos configurada para las tareas de Scraper_iteration.

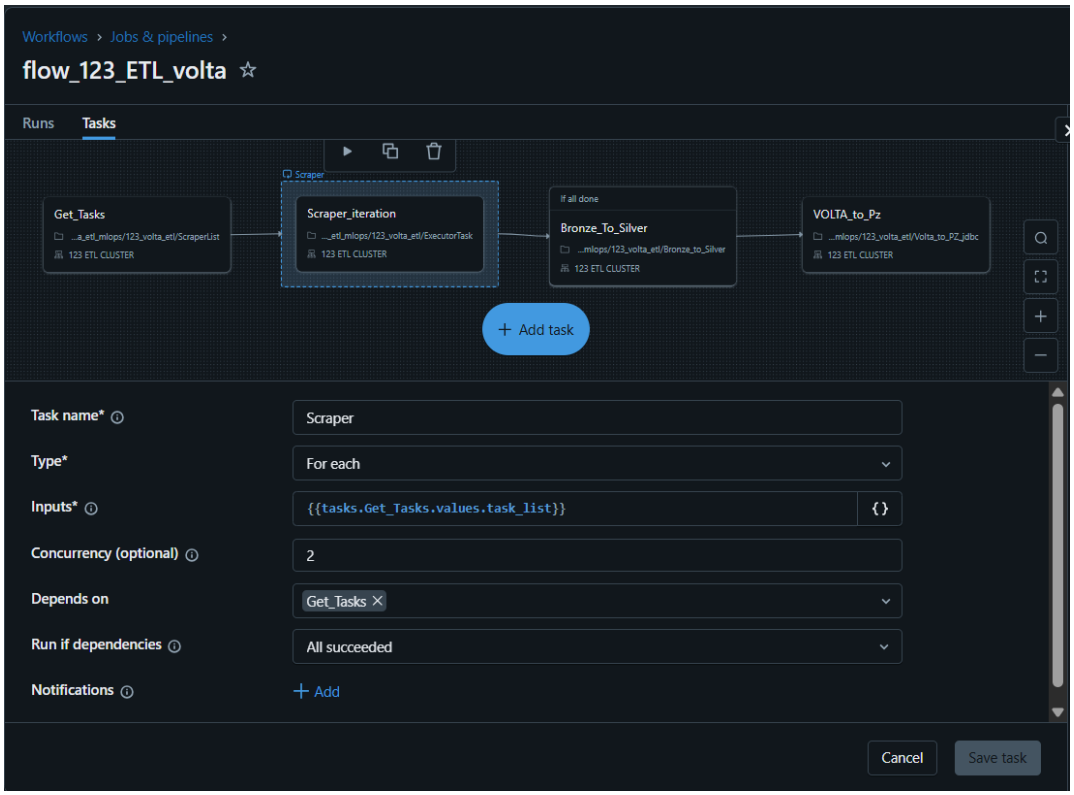

Ilustración B.2: Configuración interna de la tarea *Scraper_iteration* en modo iterativo (`for each`), utilizando como entrada la lista de tareas extraída por *Get_Tasks*.

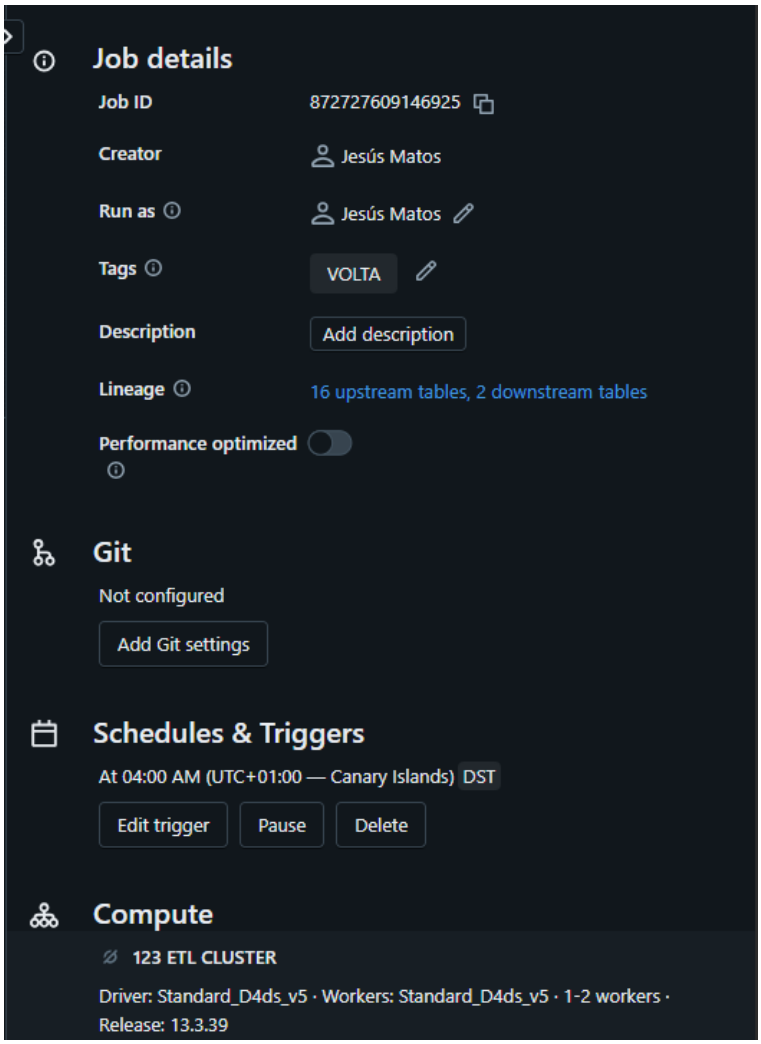

Ilustración B.3: Parámetros definidos en la tarea de scraping del flujo ETL.

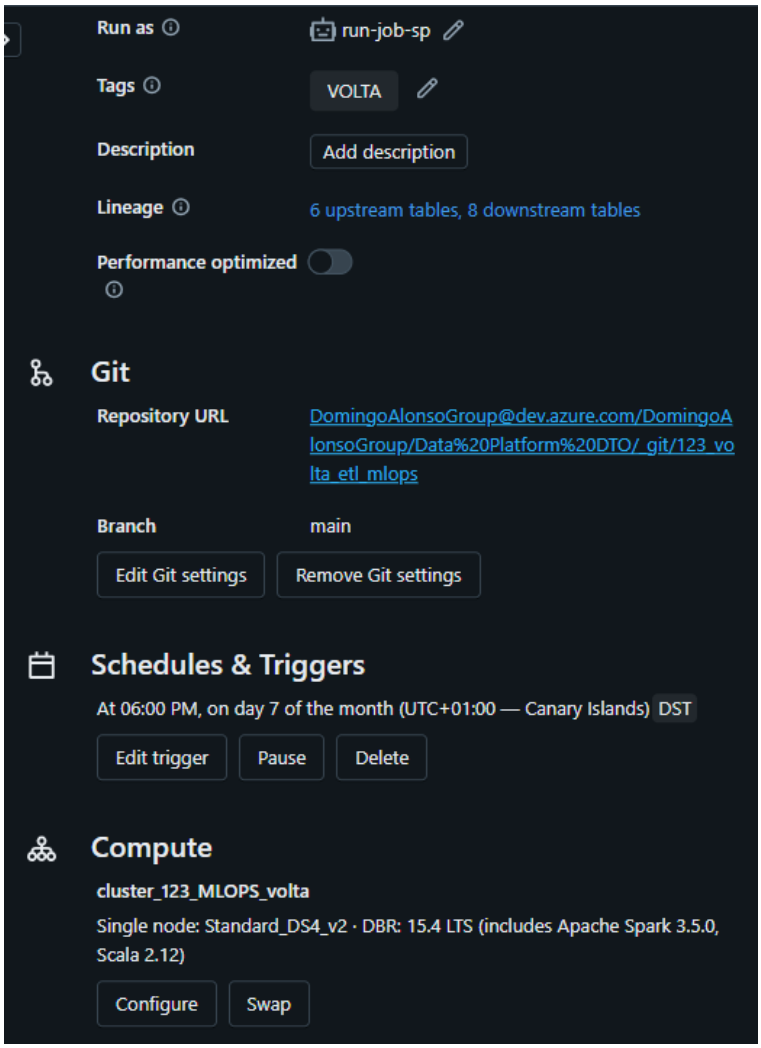

Ilustración B.4: Parámetros y opciones adicionales definidas para la tarea de ejecución del flujo MLOps.

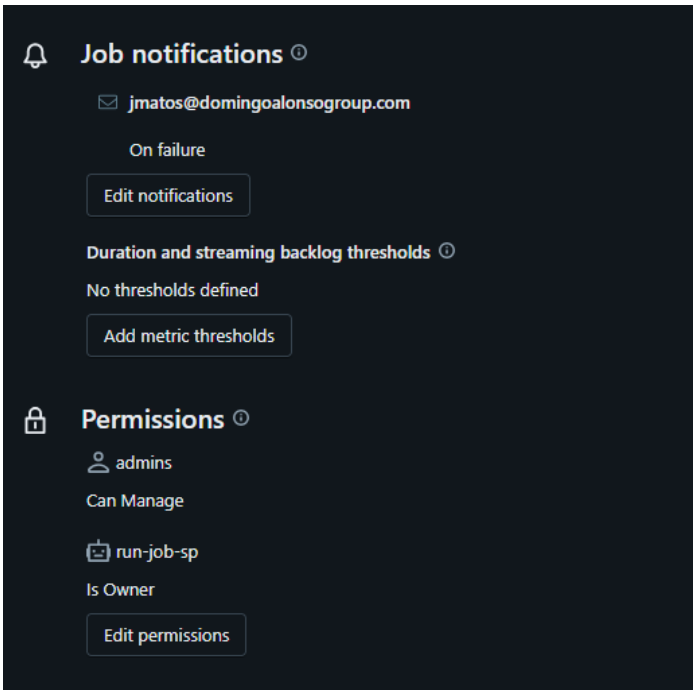

Ilustración B.5: Configuración de notificaciones por correo electrónico ante fallo, y permisos asignados para la ejecución de los flujos.