Universidad Loyola Andalucía

Máster Oficial en Inteligencia Artificial

Estimación probabilística de precios unitarios en presupuestos de construcción mediante modelos híbridos de aprendizaje automático

---

Trabajo Fin de Máster

Fco. Javier Cerón Contreras

Tutor: Bernardo Ronquillo Japón

Julio 2026

Resumen

La estimación de precios unitarios en presupuestos de construcción es difícil: los precios están muy sesgados, dependen del contexto de cada obra y del histórico de cada partida, y los conjuntos disponibles suelen ser pequeños y de un solo cliente. Este trabajo desarrolla y evalúa un sistema de estimación probabilística de precios unitarios que, además de un precio, entrega un intervalo de confianza y una señal de auditoría de sobrecostes.

El sistema combina tres piezas. Un modelo base CatBoost estima el precio (en escala logarítmica) y una primera incertidumbre, aprovechando el texto de las partidas y sus variables categóricas. Sobre sus residuos, una red LSTM probabilística con modulación FiLM y atención temporal aprende una corrección que explota el histórico de cada código. Una calibración por experto, con un ajuste conforme, produce intervalos fiables, y un semáforo basado en un Z-score logarítmico marca las partidas con posible sobrecoste para priorizar la revisión humana.

La evaluación se realiza sobre datos reales (25.046 partidas de presupuestos de retail) con validación cruzada temporal de ventana expansiva. El sistema logra una buena precisión de punto (WAPE en torno al 11 % y R^2 de 0,89), intervalos calibrados experto a experto y un semáforo que concentra la revisión en una minoría de partidas (en torno al 6 %). Frente a un baseline LightGBM (WAPE del 37 %), reduce el error a menos de un tercio, con una ventaja estadísticamente fundamentada. El análisis por segmentos muestra que la mayor diferencia la marca el histórico del código: las partidas con código visto se predicen mucho mejor que las de arranque en frío. La principal vía de mejora es acumular más histórico por código.

Palabras clave: estimación de costes de construcción, aprendizaje automático, estimación probabilística, CatBoost, LSTM, calibración conforme, intervalos de predicción, auditoría de sobrecostes.

Abstract

Estimating unit prices in construction budgets is hard: prices are heavily skewed, depend on the context of each project and on each line item's price history, and the available datasets are usually small and come from a single client. This work develops and evaluates a probabilistic unit-price estimation system that, besides a price, returns a confidence interval and an overcost-auditing signal.

The system combines three parts. A CatBoost base model estimates the price (on a logarithmic scale) and a first uncertainty, exploiting the line-item text and its categorical variables. On its residuals, a probabilistic LSTM with FiLM modulation and temporal attention learns a correction that leverages each code's history. A per-expert calibration, with a conformal adjustment, yields reliable intervals, and a traffic light based on a logarithmic Z-score flags line items with possible overcost to prioritise human review.

The evaluation uses real data (25,046 line items from retail budgets) with expanding-window temporal cross-validation. The system achieves good point accuracy (WAPE around 11 % and R^2 of 0.89), per-expert calibrated intervals, and a traffic light that concentrates review on a minority of line items (around 6 %). Against a LightGBM baseline (WAPE of 37 %), it cuts the error to less than a third, with a statistically supported advantage. The segment analysis shows that the largest difference comes from the code's history: line items with a previously seen code are predicted far better than cold-start ones. The main avenue for improvement is to accumulate more history per code.

Keywords: construction cost estimation, machine learning, probabilistic estimation, CatBoost, LSTM, conformal calibration, prediction intervals, overcost auditing.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Motivación

El control de costes es uno de los problemas centrales de la gestión de proyectos de construcción. Un presupuesto de obra se descompone en cientos o miles de partidas, cada una con su unidad de medida, su medición y su precio unitario. La desviación entre el coste presupuestado y el coste real es habitual en el sector, y una parte significativa de esa desviación se origina ya en la fase de oferta, cuando los precios unitarios pactados con la empresa constructora no se corresponden con los valores esperables para ese tipo de trabajo.

En un esquema de licitación, el promotor o su consultora envía a varias constructoras un presupuesto sin precios, es decir, un presupuesto con solo la descripción y la medición de cada partida, como se ve en la Figura 1.1. Cada constructora lo devuelve con sus precios, como se muestra en la Figura 1.2.

COSTES DE OBRA

Presupuesto

Código	Nat	Ud	Resumen	Comentario	N	Longitud	Anchura	Altura	Parcial	CanPres	PrPres	ImpPres
A1	Capítulo		BUILDING COST							1	0.00	0.00
COA1.1000	Capítulo		ESTRUCTURA							1.00	0.00	0.00
COA1.1000.01	Partida	Kg	ESTRUCTURA METÁLICA COMPLETA							19,030.83	0.00	0.00
			Estructuta metálica completa según planos. Vigas, pilares, correas de forjado y cubierta, estructura auxiliar en petos de cubierta para sujeción de marquesina,con acabado de pintura, dos manos y aplicación miniada, formación de huecos en cubierta, bases cartelas y cabezales, incluido el empotramiento de la estructura hasta la cota de cimentación. Segun planos de Estructura. NOTA 1: Se considerará en esta partida todos los elementos metalicos de montaje (cartelas, barras auxiliares, elementos de montaje, tubos para sujeción .) y demas elementos que, aunque no esten considerados en los Kg de esta partida se deban considerar. NOTA 2: Se deberá considerar tambien los rigidizadores entre correas El precio de la estructura deberá ser cerrado y no se aceptaran sobrecostes.								
				Barras:	1	17,369.80	0.00	0.00	17,369.80			
				Placas	1	308.25	0.00	0.00	308.25			
				Pernos	1	67.05	0.00	0.00	67.05			
				Tubos Aux. Marquesinas	1	189.75	0.00	0.00	189.75			
				Tudos Aux. Canalon	1	189.75	0.00	0.00	189.75			
				Mermas	0.05	18,124.60	0.00	0.00	906.23			
									COA1.1000.01	19,030.83	0.00	0.00

Figura 1.1: Presupuesto sin precios enviado a las constructoras.

704 URB MCD SAN FERNANDO JANER 2022

Presupuesto

Código	Nat	Ud	Resumen	Comentario	N	Longitud	Anchura	Altura	Parcial	CanPres	PrPres	ImpPres
A3	Capítulo		TECHNICAL COST							1	23,950.80	23,950.80
COA3.2001	Capítulo		FONTANERIA EXTERIOR							1.00	3,456.22	3,456.22
COA3.2001.01	Partida	UN	ARMARIO PARA CONTADOR DE AGUA							1.00	531.00	531.00
			Suministro e instalación de armario o centralización de contador de agua, para alojar contador de 40 mm i/p.p. de piezas especiales para conexión del contador, limpieza y medios auxiliares. Totalmente colocado según CTE, normativas municipales vigentes.									
					1	0.00	0.00	0.00	1.00			
									COA3.2001.01	1.00	531.00	531.00

Figura 1.2: Presupuesto con precios recibido de las constructoras.

La empresa se enfrenta entonces a un problema de revisión a gran escala: debe decidir, partida a partida y constructora a constructora, qué precios son razonables y cuáles están inflados. Hacerlo manualmente es lento y propenso a errores, porque una sola obra puede contener muchas partidas y se reciben varias ofertas por obra.

De ese problema surge este trabajo. Si se dispone de un histórico de partidas ya presupuestadas por distintas constructoras (constructoras que suelen ser las mismas) y en distintas obras, es posible aprender de esos datos cuál es el precio esperable y utilizar esa estimación como referencia objetiva para detectar sobrecostes. La dificultad no está solo en predecir un precio puntual, sino en cuantificar su incertidumbre: marcar una partida como sobrecoste solo tiene sentido si la desviación observada es estadísticamente significativa frente a lo que el modelo considera el rango normal de precios para esa partida.

1.2. Contexto industrial

Este trabajo se desarrolla en colaboración con Argenia, una consultora que gestiona un volumen relevante de obras de McDonald's España en el ámbito de retail y edificación. Dentro de su operativa, el departamento de retail de Argenia envía a varias constructoras el presupuesto de cada obra sin precios y recibe de vuelta las ofertas valoradas. El objetivo de negocio que da origen a este trabajo es detectar, dentro de la oferta de cada constructora, qué partidas presentan sobrecoste, de modo que la revisión técnica pueda priorizarse sobre las partidas realmente sospechosas en lugar de revisarlas todas de forma uniforme.

Para ello, Argenia ha cedido un conjunto de presupuestos reales correspondientes a distintas constructoras y distintas obras. Tras el proceso de extracción y estructuración descrito en el Capítulo 3, el conjunto de datos resultante comprende 25 046 partidas procedentes de 65 presupuestos, asociados a 9 constructoras, en un horizonte temporal que abarca de enero de 2024 a marzo de 2026.

Los presupuestos corresponden a dos tipologías (prototipo y locales) y tres categorías de obra (local, edificación y urbanización).

Todas las restricciones y dificultades se analizan en detalle en el Capítulo 3 y guían las decisiones metodológicas del Capítulo 4.

1.3. Definición del problema y objetivos

El problema que se aborda puede enunciarse en dos planos complementarios.

Plano predictivo. Dada una partida, se desea estimar su precio unitario esperado. Se trata de un problema de regresión sobre datos tabulares con un fuerte componente de texto libre y una marcada asimetría en la distribución del precio, que abarca varios órdenes de magnitud.

Plano de auditoría. Una vez estimado el precio esperado y su incertidumbre, se desea comparar dicho valor con el precio realmente ofertado por la constructora y decidir si la desviación constituye un sobrecoste. Esta decisión no puede basarse únicamente en la diferencia absoluta o relativa de precio: una misma desviación puede ser perfectamente normal en una partida con precios muy dispersos y claramente anómala en otra con precios muy estables. Por ello, la auditoría debe apoyarse en la incertidumbre de la estimación, marcando como sospechosas solo aquellas partidas cuyo precio ofertado se sitúa de forma significativa por encima del rango esperado.
Este doble planteamiento conlleva varios retos técnicos que recorren toda la memoria: hay poco histórico por código concreto, hay que evitar fugas de información temporal en la validación y los intervalos de confianza deben estar bien calibrados para que la auditoría sea fiable.
A partir de esta definición, el objetivo general de este trabajo es diseñar y evaluar un sistema de estimación probabilística de precios unitarios que sirva como apoyo a la auditoría de sobrecostes en presupuestos de obra. El sistema no pretende sustituir la revisión técnica; su papel es ofrecer una referencia cuantitativa que ayude a priorizar las partidas que requieren mayor atención.
De este objetivo general se derivan los siguientes objetivos específicos:

1. Desarrollar un modelo capaz de estimar el precio unitario esperado de una partida a partir de la información disponible.

2. Incorporar una estimación de incertidumbre asociada a cada predicción, de forma que la detección de sobrecostes tenga en cuenta la variabilidad esperable de cada tipo de partida.

3. Construir un mecanismo de auditoría interpretable que compare el precio ofertado por la constructora con el rango esperado por el modelo y señale las partidas con mayor riesgo de sobrecoste.

1.4. Estructura de la memoria

El resto de la memoria se organiza como sigue. El Capítulo 2 revisa el estado del arte en estimación de costes de construcción, modelos de gradient boosting y redes recurrentes, cuantificación y calibración de la incertidumbre y validación temporal. El Capítulo 3 describe el origen de los datos, el proceso de extracción de los presupuestos, el esquema del conjunto resultante, el análisis exploratorio y las decisiones de limpieza y partición temporal. El Capítulo 4 detalla la metodología: el preprocesado, el modelo base CatBoost, la corrección residual mediante la LSTM probabilística, la calibración de la incertidumbre, la validación k-fold temporal y el sistema semáforo de auditoría. El Capítulo 5 presenta las métricas de evaluación, el estudio de ablación y los resultados por banda de precio y por disponibilidad de histórico, además de la evaluación del semáforo y una comparación con un baseline. El Capítulo 6 interpreta los resultados a la luz del cumplimiento de los objetivos y analiza las limitaciones y la validez externa del trabajo. Por último, el Capítulo 7 resume las aportaciones y plantea las líneas de trabajo futuro.

Capítulo 2

Estado del arte

Este capítulo sitúa el trabajo en la literatura y justifica las decisiones metodológicas de los capítulos posteriores. La revisión recorre los componentes que integra el sistema propuesto: la estimación de costes en construcción, el aprendizaje automático sobre datos tabulares, el modelado temporal, la cuantificación y calibración de la incertidumbre y la validación temporal.

2.1. Estimación de costes en construcción y aprendizaje automático

La estimación de costes es una actividad central en la gestión de obra, con enfoques que van de lo paramétrico a lo analógico y al detalle por unidades de obra y precios unitarios. La literatura ha documentado que las desviaciones de coste son un problema estructural del sector: Flyvbjerg et al. [10] muestran, sobre proyectos públicos, que el error de estimación no es marginal, lo que sugiere que una estimación útil debe reconocer su propia incertidumbre. Las revisiones recientes de estimación de costes con inteligencia artificial [9], [15] confirman la adopción progresiva de técnicas de aprendizaje automático, a la vez que señalan sus límites: dependen de la cantidad y calidad de histórico, de una definición clara del objetivo y de una validación realista. A nivel de aplicación, los modelos de aprendizaje automático se han comparado de forma reiterada con la regresión clásica, desde los primeros trabajos que contrastan regresión, redes neuronales y razonamiento basado en casos [21] hasta los más recientes, que emplean redes neuronales [32] o gradient boosting [2] y suelen situar los métodos basados en árboles entre los más precisos. Este trabajo se distancia de buena parte de esa literatura en dos cosas: trabaja a nivel de precio unitario de partida (no de coste agregado de proyecto) y, además de estimar, audita la oferta comparándola con el precio esperado.

2.2. Modelos de boosting para datos tabulares

Los métodos de ensemble basados en árboles son la referencia para datos tabulares. Frente a los random forests [5], que reducen varianza por agregación, el gradient boosting [11] construye un modelo aditivo en el que cada estimador corrige el residuo del anterior, combinación especialmente eficaz en regresión tabular. Sus implementaciones modernas difieren en preprocesado y eficiencia: XGBoost [7] aportó regularización y escalabilidad;

LightGBM [18] optimizó la velocidad; y CatBoost [27] trata de forma nativa las variables categóricas mediante estadísticas de objetivo ordenadas y un boosting ordenado que limitan la fuga del objetivo durante el entrenamiento. Como los presupuestos contienen numerosas categóricas de alta cardinalidad y texto libre descriptivo (que la librería CatBoost admite también de forma nativa como variables de entrada), CatBoost encaja bien como modelo base. La evidencia de que los árboles siguen siendo muy competitivos frente al deep learning en datos tabulares [14], [29] respalda usar la parte neuronal de forma complementaria, no como sustituto del modelo tabular.

2.3. Modelado temporal y aprendizaje residual

Cuando un código de partida tiene histórico, sus precios pasados informan sobre el valor esperado y su variabilidad. Las redes recurrentes procesan estas secuencias: las LSTM [16] mantienen un estado interno que resume la información vista anteriormente y mitigan el problema del desvanecimiento del gradiente. La atención temporal [3] permite ponderar cada paso según su relevancia (una observación reciente puede pesar más que una antigua), y el condicionamiento mediante capas FiLM [26] modula la representación secuencial con información categórica de contexto. Sobre estas ideas, la metodología (Capítulo 4) adopta una interpretación residual: el modelo tabular da la estimación base y la red recurrente aprende una corrección solo donde hay histórico suficiente, en coherencia con la evidencia sobre datos tabulares.

2.4. Cuantificación y calibración de la incertidumbre

En auditoría no basta con una predicción puntual: una desviación de precio que es normal en una partida con precios dispersos puede ser sospechosa en otra de precios estables, así que la estimación necesita ir acompañada de una medida de incertidumbre. Suele distinguirse la incertidumbre aleatoria de la epistémica [19], ambas presentes aquí (partidas intrínsecamente variables y escasez de histórico). Para estimarla existen varias familias: modelar directamente media y varianza por verosimilitud, aproximaciones bayesianas como MC Dropout [12] o los deep ensembles [24]; métodos cuantílicos [22], [25]; y la predicción conforme [31], que ofrece garantías de cobertura bajo intercambiabilidad y que, combinada con regresión cuantílica [28] y con herramientas como MAPIE [8], produce intervalos adaptativos. Un buen intervalo equilibra cobertura y anchura, y su evaluación se apoya en reglas de puntuación propias [13] y en métricas como PICP y MPIW [20]. Cuando la cobertura nominal no coincide con la empírica, la calibración [23] la corrige; en este trabajo se aplica de forma segmentada por experto, pues la incertidumbre no se comporta igual en todos los tramos.

2.5. Validación temporal, fuga de información y detección de anomalías

El esquema de validación condiciona lo que realmente se está midiendo. Cuando los datos tienen estructura temporal, una partición aleatoria resulta demasiado optimista: la evaluación debe imitar el uso real del modelo, entrenando con el pasado y validando sobre periodos posteriores [30], con el cuidado que exigen la dependencia temporal y la no estacionariedad [4]. A ello se añade el riesgo de fuga de información [17]: si partidas de un mismo presupuesto se reparten entre entrenamiento y test, las métricas se inflan; de ahí la partición out-of-time agrupada por presupuesto del Capítulo 3. Por último, el objetivo de auditoría conecta con la detección de anomalías [6]: la anomalía de interés no es cualquier valor raro, sino la desviación al alza respecto al rango esperado, y su utilidad depende del coste de falsos positivos y negativos, lo que justifica presentar el semáforo como filtro de atención y no como auditor autónomo.

2.6. Research gap

Cada componente cuenta con literatura propia: estimación de costes, boosting tabular, modelos temporales, incertidumbre y predicción conforme. Lo que escasea es su integración en un caso operativo concreto: la auditoría de sobrecostes a nivel de partida en presupuestos reales. La literatura tiende a estimar costes agregados, a comparar algoritmos o a estudiar la incertidumbre de forma metodológica, y rara vez combina estimación tabular, corrección temporal cuando hay histórico, calibración de intervalos y una salida interpretable orientada a la revisión de ofertas. Este TFM se posiciona en esa intersección: no propone una familia algorítmica nueva; su aportación es diseñar y evaluar un pipeline híbrido adaptado al problema. El Capítulo 4 describe la arquitectura y el Capítulo 5 la evalúa.

Capítulo 3

Datos

Este capítulo describe el conjunto de datos que sustenta todo el trabajo: su origen, el proceso de extracción que transforma los presupuestos en Excel en una base de datos en .csv y un análisis exploratorio (EDA) orientado a justificar las decisiones de modelado del Capítulo 4. Se detallan también la limpieza aplicada, la partición temporal del conjunto, las garantías frente a fugas de información y la segmentación de las partidas en expertos. El capítulo se cierra con un diagnóstico de la representatividad del conjunto de validación que motiva el uso de una validación cruzada temporal.

3.1. Origen de los datos

Los datos proceden de presupuestos reales de obra cedidos por Argenia en el marco descrito en el Capítulo 1. Corresponden a obras de McDonald's España en el ámbito de retail y edificación, ofertadas por distintas constructoras entre enero de 2024 y marzo de 2026. Cada presupuesto es un fichero Excel en formato Presto en el que una constructora valora, partida a partida, la obra que el departamento de retail le había enviado sin precios.

Tras el proceso de extracción y estructuración descrito en la Sección 3.2, el conjunto resultante comprende 25 046 partidas procedentes de 65 presupuestos, asociados a 9 constructoras (ABESSIS, AITANA, AOCSA, CAAD, CAMPOS, MARAGON, RUAYA, SERCOINS y VIALTI) y a obras situadas en 5 provincias (Almería, Cádiz, Granada, Málaga y Sevilla). Una misma obra de McDonald's es ofertada habitualmente por varias constructoras, de modo que un número reducido de obras genera el conjunto de presupuestos. Las partidas se reparten en dos tipologías, Prototipo (16 321 partidas) y Locales (8 725), y tres categorías de obra: Local (8 725), Edificación (8 626) y Urbanización (7 695).

El volumen de datos es, por tanto, modesto y procede de un único cliente. Esta es una característica determinante del problema: el modelo debe aprender de partidas valoradas, con un fuerte desequilibrio entre obras y constructoras, y generalizar a ofertas futuras. Las implicaciones de esta limitación se retoman en la Sección 6.2 del Capítulo 6.

3.2. De los presupuestos sueltos a un CSV maestro

Cada constructora entrega su oferta como un fichero Excel independiente, con su propia estructura de capítulos, subcapítulos y partidas. Tener decenas de ficheros sueltos no

sirve para entrenar un modelo, así que el primer paso es juntarlos todos en una sola tabla.

Para ello hay un proceso de extracción (`src/extraction/`) que recorre todos esos Excel y los reúne en un único fichero CSV (`BD_MAESTRA_PRECIOS.csv`). En esa tabla, cada fila es una partida de obra y las columnas recogen su precio, su descripción, sus mediciones y el contexto del presupuesto al que pertenece (constructora, obra, fecha y tipología). A partir de aquí, todo el trabajo se hace sobre ese único fichero.

Hay que aclarar cómo se rellena cada fila, porque en el Excel la información de una partida no está toda en una línea. Debajo de cada partida suele haber varias líneas auxiliares: las de medición, que indican cuántas unidades hay y sus dimensiones (longitud, anchura y altura) junto con su subtotal, y la mayoria de veces la descripción técnica larga. El proceso de extracción reúne todo eso en la fila de la partida: suma las cantidades y las dimensiones de sus líneas de medición (columnas `Suma_N`, `Suma_Longitud`, `Suma_Anchura`, `Suma_Altura` y `Suma_Parcial`), cuenta cuántas líneas eran (`Num_Lineas_Medicion`), junta sus comentarios en `Textos_Mediciones` y guarda aparte la descripción. Así, cada partida acaba resumida en una sola fila con su precio, su texto y sus mediciones ya agregadas.

3.3. Esquema del CSV maestro

El fichero maestro contiene 25 columnas que combinan identificadores, contexto de la obra, jerarquía presupuestaria, texto descriptivo, magnitudes de medición y el precio objetivo. La Tabla 3.1 resume cada columna, su tipo y su porcentaje de valores ausentes. La gran mayoría de las columnas no presenta nulos; las excepciones relevantes son `Textos_Mediciones` (ausente en el 77,9 % de las partidas) y `Descripcion` (15,8 %). La escasez del campo `Textos_Mediciones` es notable y condiciona la utilidad de las features derivadas de él.

Columna	Tipo	% nulos	Descripción
`idx_original`	entero	0	Índice global de la fila.
`ID_Presupuesto`	texto	0	Identificador único del presupuesto.
`Tipología`	texto	0	Prototipo o Locales.
`Categoria_Obra`	texto	0	Local, Edificación o Urbanización.
`Constructora`	texto	0	Constructora que oferta el presupuesto.
`Fecha_Oferta`	fecha	0	Fecha del presupuesto.
`Ubicacion_Obra`	texto	0	Provincia de la obra.
`Posicion_Partida`	entero	0	Orden de la partida en el presupuesto.
`Capítulo_Padre`	texto	0	Capítulo de nivel superior.
`Subcapítulo`	texto	0	Subcapítulo de la partida.
`Código_Orig`	texto	0	Código de partida original.
`Nombre`	texto	0	Resumen o nombre corto de la partida.
`Descripcion`	texto	15,8	Texto técnico largo (0 o 1 por partida).
`Textos_Mediciones`	texto	77,9	Comentarios de las líneas de medición.
`Unidad_Medida`	texto	0,9	Unidad (m^2, ml, ud, ...).
`Cantidad`	real	0	Cantidad presupuestada.
`Precio_Unitario`	real	0	Precio unitario en EUR (objetivo).
`Importe_Total`	real	0	Precio × cantidad.
`Tiene_Desglose`	entero	0	1 si la partida tiene líneas de medición.
`Suma_N`	real	0	Suma del número de unidades de medición.
`Suma_Longitud`	real	0	Suma de longitudes de medición.
`Suma_Anchura`	real	0	Suma de anchuras de medición.
`Suma_Altura`	real	0	Suma de alturas de medición.
`Suma_Parcial`	real	0	Suma de subtotales de medición.
`Num_Lineas_Medicion`	entero	0	Número de líneas de medición.

Tabla 3.1: Esquema del CSV maestro: tipo, porcentaje de valores nulos y semántica de cada columna.

3.4. Análisis exploratorio de datos

El análisis exploratorio persigue un objetivo concreto: entender por qué el problema es difícil y qué decisiones de modelado quedan justificadas por los datos. Se centra, por tanto, en cuatro aspectos (la distribución del precio, la cobertura de histórico por código, la estabilidad del precio de cada código en el tiempo y la estructura temporal del conjunto) y no en una descripción exhaustiva de todas las variables. Cada aspecto se acompaña de su figura; el notebook de análisis exploratorio (`notebooks/EDA_ComPres.ipynb`) recoge el detalle completo.

3.4.1. Distribución del precio unitario

La variable objetivo, `Precio_Unitario`, presenta dos rasgos que marcan toda la metodología. En primer lugar, 6 955 partidas (el 27,8 %) tienen precio cero: son partidas «a justificar» o sin valorar por la constructora. No aportan información de precio y se eliminan en la limpieza (Sección 3.5), de modo que el modelado se realiza sobre las 18 091 partidas valoradas.

Ocurre algo parecido en el extremo bajo de los precios positivos. Según el departamento de retail de Argenia, algunas constructoras tampoco quieren valorar ciertas partidas pero, en lugar de dejarlas a cero, les ponen un precio mínimo, casi simbólico. Como esos valores no reflejan un precio real, en la limpieza se descarta el 1 % de precios más bajos (las partidas por debajo del percentil 1).

En segundo lugar, el precio de las partidas valoradas abarca varios órdenes de magnitud y está fuertemente sesgado a la derecha, como muestran la Figura 3.1 y la Tabla 3.2: la mediana es de 106,29 EUR pero la media asciende a 612,98 EUR, con un máximo superior a 64 000 EUR. El coeficiente de asimetría es de 10,57. La transformación logarítmica `log1p` reduce la asimetría a 0,23 y deja una distribución prácticamente normal, por lo que se modela el logaritmo del precio y se emplean métricas en escala logarítmica durante todo el pipeline.

Estadístico	Valor (EUR)
Mínimo	0,01
Percentil 1	1,93
Percentil 5	6,06
Mediana	106,29
Percentil 99	7 739,37
Máximo	64 295,40
Media	612,98
Desviación típica	2 055,62
Asimetría (precio crudo)	10,57
Asimetría (`log1p`)	0,23

Tabla 3.2: Distribución del `Precio_Unitario` sobre las 18 091 partidas valoradas (precio positivo), en EUR, y efecto de la transformación `log1p`.

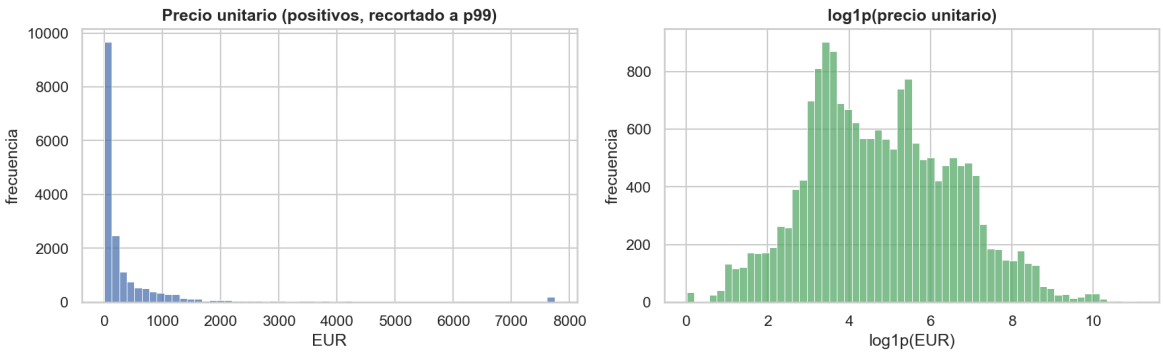

Figura 3.1: Distribución del precio unitario de las partidas valoradas: en escala cruda (izquierda); la transformación `log1p` la deja muy parecida a una normal (derecha).

3.4.2. Cobertura de histórico por código

La corrección residual mediante la red recurrente (Capítulo 4) se apoya en que cada código de partida tenga un historial de precios en distintas fechas. La mediana de fechas por código es de 5 fechas y el máximo observado es 8, que coincide con el número de fechas distintas del conjunto. Un 17,1 % de los códigos aparece en una sola fecha (cold-start puro) (Figura 3.2).

La Tabla 3.3 muestra qué fracción de códigos y de partidas dispone de al menos k fechas de histórico. Aunque solo el 24,2 % de los códigos alcanza 6 o más fechas, esos códigos concentran el 45,1 % de las partidas; y el 88,5 % de las partidas corresponde a códigos con al menos 3 fechas. Así, la mayor parte del volumen tiene histórico suficiente para construir secuencias, aunque queda una cola apreciable de códigos nuevos o casi nuevos que requieren un tratamiento aparte.

k (fechas)	% de códigos	% de partidas
≥ 1	100,0	100,0
≥ 2	82,9	96,5
≥ 3	68,1	88,5
≥ 4	62,8	84,4
≥ 6	24,2	45,1
≥ 8	16,4	32,6

Tabla 3.3: Cobertura de histórico: porcentaje de códigos y de partidas con al menos k fechas distintas.

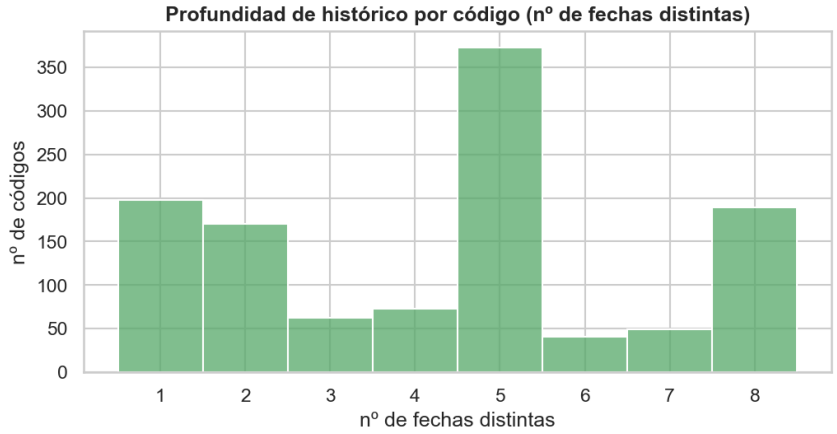

Figura 3.2: Distribución de la profundidad de histórico por código: número de fechas distintas en que aparece cada código.

3.5. Limpieza y preprocesado del dataset

Antes de modelar hay que dejar los datos limpios y comparables. Cualquier umbral o valor que dependa de los datos se calcula solo con el conjunto de entrenamiento y luego se reutiliza igual en validación y test.
Los pasos principales son:

- Tipado. Las columnas numéricas que llegan como texto (con formato europeo) se pasan a número y `Fecha_Oferta` se convierte a fecha.
- Outliers. Se descartan las partidas con precio muy bajo o muy alto (por debajo del percentil 1 o por encima del 99), con los umbrales calculados solo con el entrenamiento. Los precios más bajos son, en buena parte, partidas que la constructora no quiso valorar (Sección 3.4).
- Unidades. Se unifican las unidades de medida (por ejemplo "m^2" y "m2", o "uds" y "ud") y se descartan las partidas sin unidad o medidas por paquete (PAK), que no son comparables.
- Texto. Los campos de texto se pasan a minúsculas, sin acentos ni espacios raros, y los textos opcionales que faltan se dejan vacíos.

Además, a partir de los textos de medición y de las fechas se crean algunas variables nuevas (por ejemplo, los días transcurridos desde la primera oferta o el mes en forma cíclica). El detalle se ve en el Capítulo 4.

3.6. División temporal del dataset

La evaluación tiene que parecerse al uso real: predecir precios de presupuestos futuros. Por eso no se reparte al azar, sino por fecha. Cada presupuesto va entero a un único conjunto según su fecha más antigua: ordenados de más viejo a más nuevo, el 70 % inicial es entrenamiento, el 15 % siguiente validación y el 15 % final test. Se agrupa por `ID_Presupuesto`.

Para verlo más claro, la Tabla 3.4 muestra un ejemplo. Las dos partidas del presupuesto P1, que son de la misma oferta, van juntas a entrenamiento porque su fecha es de las más antiguas; las de los presupuestos P58 y P63, mucho más recientes, caen en validación y en test. Lo importante es que ninguna oferta se reparte entre conjuntos: un presupuesto entero va siempre al mismo lado.

Tabla 3.4: Ejemplo de la partición agrupada. Todas las partidas de un mismo presupuesto van al mismo conjunto, según la fecha más antigua del presupuesto.

Partida	Presupuesto	Fecha	Conjunto
solera de hormigón	P1	18/01/2024	Train
pintura plástica	P1	18/01/2024	Train
falso techo registrable	P2	05/06/2024	Train
...	...	...	...
carpintería de aluminio	P58	24/03/2026	Val
solera de hormigón	P58	24/03/2026	Val
micropilotes	P63	30/03/2026	Test

Sobre los presupuestos que no son test se añade además una validación cruzada temporal de ventana creciente (expanding window). Sirve para evaluar el modelo con más fiabilidad y para generar los residuos que necesita la corrección con la LSTM. Funciona así: se ordenan las fechas, se reservan las primeras como arranque y cada fold mueve la frontera una fecha hacia adelante, entrenando con todo lo anterior y validando en esa fecha, con el test siempre fijo (Figura 3.3). Es la práctica habitual para series temporales [4], [30] y se prueba con 2, 3 y 4 folds en el Capítulo 5.

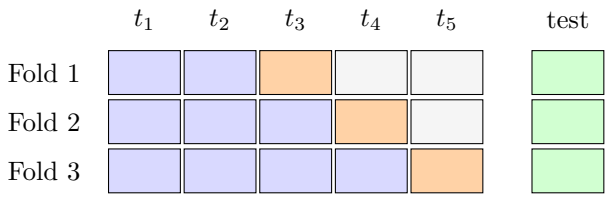

Figura 3.3: Validación cruzada temporal con ventana expansiva (aquí, tres folds). Cada fila es un fold: entrena con las fechas anteriores (azul) y valida con la siguiente (naranja); al avanzar de fold, la validación rota hacia delante y el entrenamiento crece. El test (verde) son las fechas más recientes y se mantiene apartado en todos los folds.

3.7. Análisis de fuga temporal

Una fuga de información (data leakage) ocurre cuando el modelo se entrena con datos que en la realidad no tendría al predecir; es un error fácil de cometer y difícil de detectar [17]. Aquí hay que cuidar dos casos.

El primero es que un mismo presupuesto acabe partido entre entrenamiento y test. Una misma partida puede repetirse en varios presupuestos; si dos que comparten partidas cayeran a lados distintos de la frontera, el modelo se estaría evaluando sobre algo que ya ha visto y las métricas saldrían mejores de lo real. Por eso la unidad de reparto es el presupuesto entero: un presupuesto nunca se parte entre conjuntos. Como cada presupuesto se asigna por su fecha más antigua, una misma fecha de calendario sí puede aparecer en dos conjuntos; lo que no se reparte es un presupuesto.

El segundo es calcular cualquier cosa usando datos de validación o test. Para evitarlo, como ya se dijo en la Sección 3.5, los umbrales de outliers, la fecha de referencia y las variables de medición se calculan solo con el entrenamiento y se reutilizan sin recalcular. Lo mismo vale al generar los residuos para la corrección con la LSTM (Capítulo 4).

3.8. Definición del experto (`expert_id`)

Para que el modelo pueda especializarse por tipo de trabajo, las partidas se reparten en diez grupos temáticos a los que llamamos expertos; esto permite además calibrar la incertidumbre por separado en cada uno (Capítulo 4). El reparto no es estadístico, lo hace un conjunto de reglas fijas: para cada partida se juntan su capítulo, subcapítulo, nombre y descripción y se busca la primera palabra clave de una lista temática (las claves más largas tienen prioridad, por ser más específicas). Si no encaja con ninguna, la partida va al experto 10 («Varios»). La Tabla 3.5 recoge los diez grupos. Estos grupos fueron designados por el departamento de retail.

Tabla 3.5: Definición de los diez expertos temáticos asignados por el router determinista.

`expert_id`	Dominio
1	Coste técnico / Saneamiento
2	Decoración interior (zona pública)
3	Obra exterior dentro de parcela
4	Obra exterior fuera de parcela
5	Albañilería / Ayudas de albañilería
6	Cimentación / Cubiertas
7	Carpintería metálica y de madera / Cerrajería / Vidriería
8	Aislamientos / Impermeabilización / Ignifugados
9	Demoliciones / Trabajos previos
10	Varios (sin coincidencia)

Los grupos se forman por el tipo de obra, no por el precio, y el reparto es muy desigual: unos expertos reúnen miles de partidas y otros apenas unas decenas (Figura 3.4). Esa desigualdad influye en la calibración por experto y se mide sobre el test en el Capítulo 5.

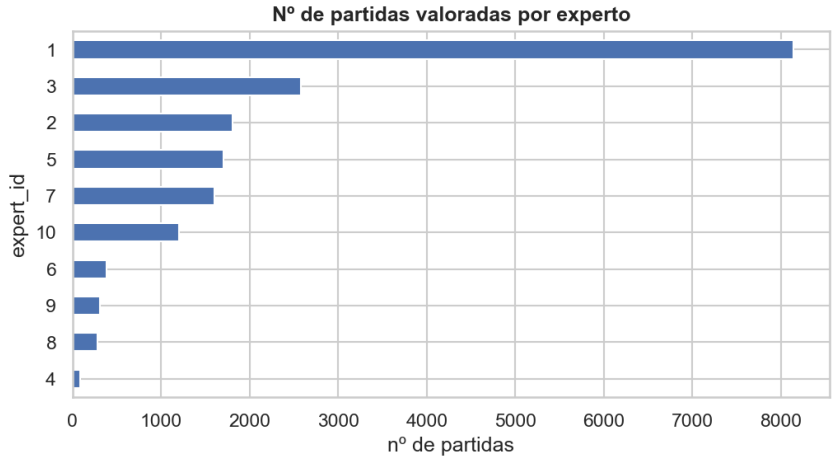

Figura 3.4: Número de partidas valoradas por cada experto (`expert_id`).

3.9. Estadística descriptiva de los splits

La Tabla 3.6 resume la partición resultante. Se indican tanto el total de partidas como las efectivamente valoradas (con precio positivo, que son las que entran en el modelado tras la limpieza), junto con el número de presupuestos, provincias, constructoras y fechas de cada conjunto, su rango temporal y la mediana de precio.

Tabla 3.6: Estadística descriptiva de la partición temporal. «Valoradas» son las partidas con precio positivo tras la limpieza. La mediana se calcula sobre las partidas valoradas.

Conjunto	Partidas	Valoradas	Presup.	Prov.	Const.	Fechas
Train	17 455	12 919	45	4	9	6
Val	3 922	2 966	10	2	6	2
Test	3 669	2 206	10	1	4	2
Total	25 046	18 091	65	5	9	8

El reparto cumple aproximadamente las proporciones 70/15/15 sobre los presupuestos.

3.10. Diagnóstico del conjunto de validación y motivación del k-fold

La partición temporal estándar, aunque es correcta, deja en este caso una validación poco representativa. Como muestra la Tabla 3.6, la validación está formada por solo 10 presupuestos repartidos en 2 fechas, comprendidas en una ventana de apenas tres días (del 24 al 27 de marzo de 2026), y el test ocupa una ventana igual de estrecha e inmediatamente posterior. A esto se suma la estructura temporal irregular descrita en la Sección 3.4, con dos campañas separadas por unos veinte meses.

Por eso una única partición train/val/test da una estimación de validación inestable: bastan unas pocas partidas de precio elevado para que las métricas globales se desplomen, lo que refleja la fragilidad del conjunto de validación más que un fallo real del modelo. Este diagnóstico motiva la decisión metodológica principal del trabajo: en lugar de fiar la evaluación a una única partición de validación, se emplea la validación cruzada temporal de ventana expansiva introducida en la Sección 3.6. Al promediar varias fronteras de validación, el k-fold da una estimación más estable de la capacidad de generalización y, de paso, genera los residuos fuera de muestra que necesita la corrección residual. La sensibilidad de los resultados al número de folds ($N \in \{2, 3, 4\}$) se examina en el Capítulo 5.

Capítulo 4

Metodología

Este capítulo describe la metodología del sistema: cómo se preparan los datos, los dos modelos que estiman el precio (CatBoost como modelo base y una LSTM que lo corrige con el histórico), cómo se calibra la incertidumbre, el esquema de validación temporal y, por último, el semáforo que convierte la estimación en una señal de auditoría de sobrecostes. Cada decisión se apoya en las características de los datos analizadas en el Capítulo 3.

4.1. Visión general del pipeline

El sistema es una cascada de modelos que parte de la base de datos maestra del Capítulo 3 y termina en una señal de auditoría para cada partida. El esquema es el siguiente: un primer modelo da una estimación rápida del precio; un segundo la afina cuando hay histórico que aprovechar, y una etapa final convierte esa estimación en intervalos fiables y en un aviso de posible sobrecoste. La Figura 4.1 resume este recorrido.

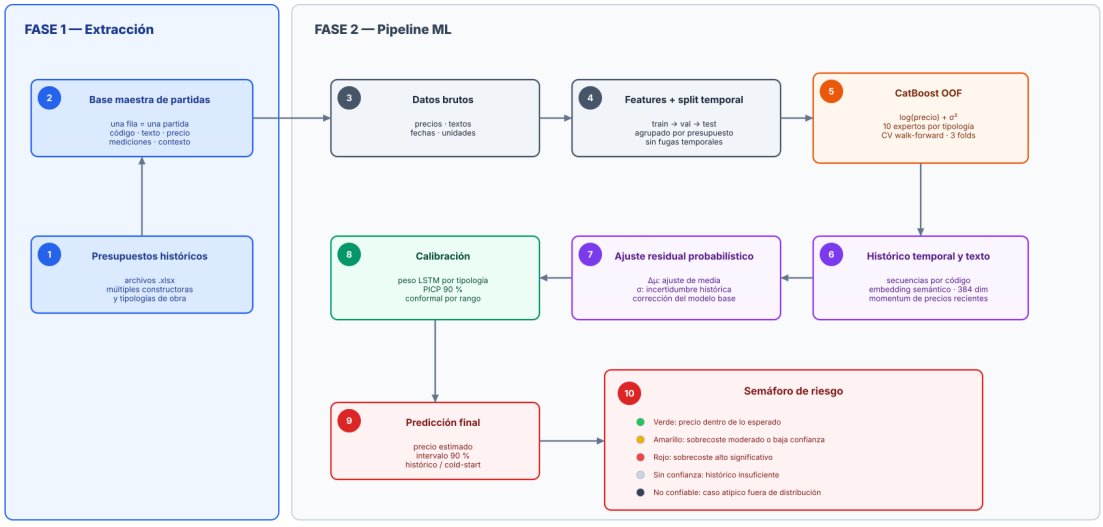

Figura 4.1: Visión general del pipeline: de los presupuestos sueltos a la señal de auditoría.

El diagrama separa dos fases. La primera (FASE 1) es la extracción descrita en el Capítulo 3: convierte los presupuestos sueltos en la base maestra de partidas. La segunda (FASE 2) es el pipeline de aprendizaje, que es lo que desarrolla este capítulo.
La segunda fase se desarrolla en seis etapas:

- Preprocesado. A partir del CSV maestro se preparan las variables (texto, categóricas, numéricas y temporales), se construyen las secuencias históricas de cada código y se transforma el precio a escala logarítmica (Sección 4.2).
- Modelo base, CatBoost. Estima, para cada partida, el precio esperado (en logaritmo) y una medida de incertidumbre. (Sección 4.3).
- Corrección residual, LSTM. Donde un código tiene suficiente histórico, una red recurrente aprende una corrección sobre la predicción de CatBoost a partir de la secuencia de precios pasados. Solo la afina, no la sustituye (Sección 4.4).
- Calibración. Ajusta cuánto pesa la corrección de la LSTM y la anchura de los intervalos, de modo que la confianza que se declara coincida con la realidad (Sección 4.5).
- Predicción final. Para cada partida se obtiene un precio estimado, un intervalo de confianza y una etiqueta de su origen (con histórico, sin histórico o solo CatBoost).
- Semáforo. Compara el precio ofertado por la constructora con el rango esperado y marca las partidas con posible sobrecoste para que el equipo de retail las revise (Sección 4.7).

Las siguientes secciones desarrollan cada etapa por orden.

4.2. Preprocesado

La limpieza y la partición temporal ya se vieron en el Capítulo 3. Aquí se explica cómo esos datos, ya limpios, se convierten en las entradas de los dos modelos. CatBoost y la LSTM no usan las variables igual: CatBoost mira cada partida por separado, mientras que la LSTM necesita además el historial del código en forma de secuencia.

4.2.1. Texto

CatBoost admite columnas de texto de forma nativa, sin vectorizarlas a mano. Se le pasan como texto el nombre de la partida, su descripción y los comentarios de medición (`Nombre`, `Descripcion` y `Textos_Mediciones`). La LSTM, que solo entiende números, convierte ese mismo texto en un vector con un modelo de embeddings de frase multilingüe (`paraphrase-multilingual-MiniLM-L12-v2`), que da 384 números por partida.

4.2.2. Variables categóricas y numéricas

CatBoost trata las variables categóricas de forma nativa, sin codificación one-hot: usa la tipología, la constructora, el capítulo, el código, la unidad de medida y la etiqueta del experto, además de varios atributos que describen la partida (material principal, marca, dimensiones, calidad y si requiere medios auxiliares). De los comentarios de medición (`Textos_Mediciones`) se derivan, aparte de un recuento de ítems, unas cuarenta banderas binarias para los términos de medición más frecuentes e informativos del conjunto de entrenamiento. La LSTM no las usa todas: convierte en un embedding solo tres, el capítulo, la constructora y el experto, con tamaños fijados en la configuración (16, 12 y 16). El capítulo y la constructora se concatenan en un único vector que, junto con las variables numéricas, condiciona la red; el del experto sirve para adaptar la predicción a cada experto. El detalle de cómo se combinan se ve en la arquitectura (Sección 4.4). Las numéricas (cantidad, días desde la primera oferta, momentum de precios recientes y las variables derivadas de las mediciones) entran directamente.

4.2.3. Secuencias por código

La LSTM no mira una partida aislada, sino el historial de su código en el tiempo. Para cada código se ordenan sus apariciones por fecha y se toma una ventana con las últimas seis (el parámetro `seq_len`, fijado en 6); si hay menos, se rellena con ceros (Figura 4.2). Solo se construye secuencia para los códigos con al menos dos fechas de histórico (`min_history` fijado en 2): los códigos nuevos no tienen historial que aprovechar y los predice CatBoost.

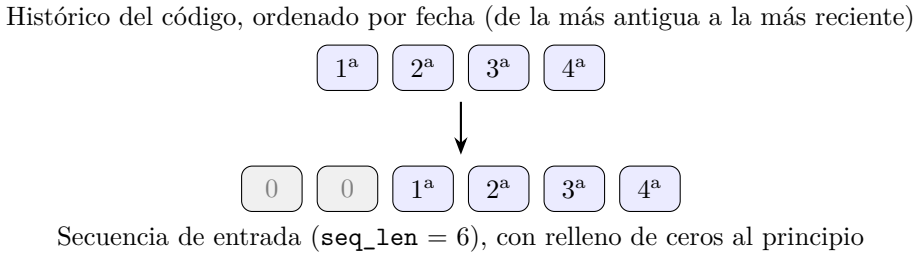

Figura 4.2: Construcción de la secuencia de un código: sus apariciones se ordenan por fecha y se toman las últimas seis. Si hay menos de seis (aquí, cuatro), las posiciones que faltan se rellenan con ceros (padding).

4.2.4. Transformación del precio

El precio está muy sesgado (Capítulo 3), así que CatBoost no predice el precio directo sino su logaritmo, `log1p`(precio), y las métricas se miden en esa escala. El LSTM no predice el precio: predice el residuo, es decir, lo que le falta o le sobra a la estimación de CatBoost en escala logarítmica. Ese residuo no mide lo mismo en todas las partidas:

en unos tipos de partida CatBoost se equivoca más y en otros menos. Para que la LSTM aprenda una corrección comparable en todos, el residuo se reescala dentro de cada experto, restándole su valor típico (la mediana) y dividiéndolo por su dispersión (el rango intercuartílico). Así los residuos de todos los expertos quedan en una escala parecida.

Ese residuo se calcula fuera de muestra (out-of-fold) para no introducir fuga de información: el train se ordena por fecha y se parte en bloques, y cada bloque se predice con un CatBoost entrenado solo con los bloques anteriores, nunca con datos posteriores (Figura 4.3). El residuo es la diferencia entre el precio real y esa predicción en escala logarítmica.

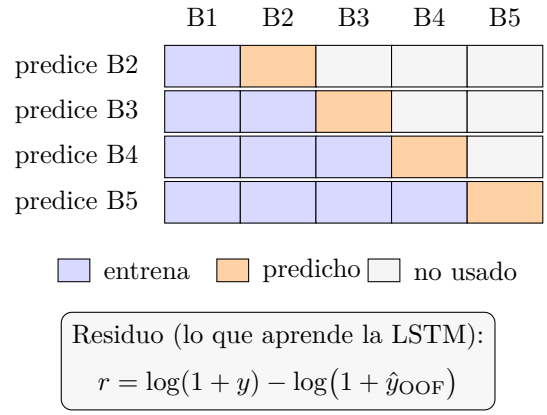

Figura 4.3: Creación de los residuos fuera de muestra (out-of-fold) sobre el entrenamiento. El train se ordena por fecha y se divide en bloques; cada bloque (en naranja) se predice con un CatBoost entrenado solo con los bloques anteriores (en azul), nunca con datos posteriores, para no introducir fuga. El residuo es la diferencia entre el precio real y y esa predicción $\hat{y}_{\mathrm{OOF}}$ en escala logarítmica, y es el objetivo que aprende a corregir la LSTM. El primer bloque queda como arranque, sin residuo. Es un esquema interno, distinto del k-fold de evaluación (Figura 3.3).

4.3. Modelo base: CatBoost

CatBoost es el modelo base: estima el precio de todas las partidas y aporta una primera medida de incertidumbre. Es rápido, trata bien las categóricas y el texto y rinde con pocos datos, que es justo lo que hay aquí (Capítulo 2).

4.3.1. Función de pérdida `RMSEWithUncertainty`

CatBoost se entrena con la pérdida `RMSEWithUncertainty`. En lugar de dar solo una predicción, aprende a la vez dos salidas por partida: la media del precio (en logaritmo) y su varianza. Así, además del precio esperado, el modelo da ya una primera estimación de cuánta incertidumbre tiene esa predicción, que más tarde servirá para crear los intervalos.

En concreto, para cada partida, siendo y el precio real en logaritmo, μ la media predicha y σ^2 la varianza predicha, la pérdida es la log-verosimilitud negativa de una distribución normal:

$$\mathcal{L}(y;\mu,\sigma^2) = \frac{1}{2}\ln\left(2\pi\sigma^2\right) + \frac{(y-\mu)^2}{2\sigma^2}.$$

En la práctica, cuando se equivoca, ese fallo le cuesta más si había dicho que estaba muy seguro (σ^2 pequeña) y le cuesta menos si había avisado de que dudaba (σ^2 grande). Pero no le sale gratis declararse siempre inseguro, porque poner una σ^2 grande tiene su propio coste. Ese tira y afloja lleva al modelo a dar una σ^2 que refleja el tamaño real de sus errores, pequeña donde acierta y grande donde falla.

4.3.2. Búsqueda de hiperparámetros con Optuna

Los hiperparámetros del modelo se ajustan con Optuna [1], que prueba muchas combinaciones y se queda con la mejor. Hay que distinguir dos cosas: CatBoost siempre se entrena con la pérdida `RMSEWithUncertainty` (la del apartado anterior), mientras que Optuna usa, por encima, una métrica distinta para puntuar cada combinación y elegir la mejor. Esa métrica de selección es el error absoluto medio en escala logarítmica (`mae_log`), calculado sobre la validación. Para que la búsqueda sea rápida se hace con una configuración ligera, sin las columnas de texto, que encarecen mucho el entrenamiento; solo el modelo definitivo se entrena ya con el texto incluido.
Se prueban 20 combinaciones (con un tope de 45 minutos) de los hiperparámetros que más influyen en CatBoost; la estructura del árbol y la función de pérdida se mantienen fijas. La Tabla 4.1 recoge cada hiperparámetro y su rango.

Hiperparámetro	Rango explorado	Qué controla
`iterations`	600–2200 (paso 200)	número de árboles
`depth`	4–7	profundidad de cada árbol
`learning_rate`	0,01–0,18 (log)	aporte de cada árbol
`l2_leaf_reg`	0,1–150 (log)	regularización L2
`random_strength`	0,01–25 (log)	aleatoriedad en los cortes
`min_data_in_leaf`	1–96 (log)	mínimo de muestras por hoja
`leaf_estimation_iterations`	1–6	ajuste del valor de las hojas
`border_count`	{64, 128, 192}	cortes al discretizar numéricas
`rsm`	0,75–1,0 (solo CPU)	fracción de variables por árbol
`bootstrap_type`	{Bayesian, Bernoulli}	tipo de muestreo de filas
`bagging_temperature`	0–5 (si Bayesian)	intensidad del muestreo
`subsample`	0,65–1,0 (si Bernoulli)	fracción de filas por árbol

Tabla 4.1: Espacio de búsqueda de Optuna para CatBoost. Los rangos marcados con "log" se muestrean en escala logarítmica.

4.3.3. Un modelo por experto

Además del modelo global se entrena un CatBoost para cada experto que tenga suficientes datos al menos 75 partidas de entrenamiento, con una búsqueda de Optuna más corta (8 pruebas).

Lo que cambia entre expertos no es la pérdida de entrenamiento, que sigue siendo `RMSEWithUncertainty`, sino esa métrica con la que Optuna elige los hiperparámetros. Por defecto es la misma que en el modelo global, el `mae_log`. El problema es que ese error, medido en logaritmo, no refleja bien el error relativo, que es el que más duele en las partidas baratas: en un precio de pocos euros, un fallo pequeño en valor absoluto es un porcentaje enorme. Por eso, en los expertos de precios bajos que además ya fallan en términos relativos, conviene un objetivo que vigile también ese error.

En concreto, un experto pasa al objetivo mixto (el `mae_log` más el WAPE con un peso de 0,35) solo si cumple las cuatro condiciones a la vez:

- tiene suficiente entrenamiento (más de 75 partidas).
- tiene suficiente validación (al menos 25 partidas), para medir el error relativo con fiabilidad.

- sus partidas son de precio bajo respecto al conjunto global, lo que se cumple cuando se da al menos una de estas comparaciones: su mediana de precio es menor o igual que la mediana global, su percentil 20 es menor o igual que el percentil 20 global, o tiene mayor proporción de precios por debajo de 10 o de 20 euros que el conjunto global.
- su error relativo en validación (WAPE o MAPE) es al menos tan alto como el del modelo global.

Si no las cumple todas, se queda con `mae_log`. Así, la búsqueda solo persigue el error relativo donde de verdad es un problema y, en el resto de expertos, no se desvía del objetivo habitual.

4.3.4. Mezcla del modelo global y el del experto

Para cada partida hay entonces dos predicciones posibles: la del modelo global y la del modelo de su experto. La predicción final es una media ponderada de las dos, en escala logarítmica:

$$\mu_{\text{final}} = w\,\mu_{\text{experto}} + (1-w)\,\mu_{\text{global}}, \qquad \sigma^2_{\text{final}} = \sigma^2_{\text{global}},$$

donde el peso w del experto depende de cuántos datos tiene:

$$w = \frac{n}{n+K}, \qquad K = 20,$$

y n es el número de partidas con las que se entrenó el modelo de ese experto.
La incertidumbre (σ^2) se toma siempre del modelo global, nunca del experto: el global se entrena con todos los datos y su estimación de la varianza es más estable, mientras que la de un experto con pocas partidas sería ruidosa.
En general, K marca el punto en que el experto y el global pesarían igual ($w = 0{,}5$ cuando $n = K$). Aquí, sin embargo, un experto solo se entrena si supera las 75 partidas, de modo que n siempre es mayor que 75 y el peso del experto nunca baja de $w \approx 0{,}8$: el experto siempre lleva la mayor parte y el global no llega a dominar. Con $K = 20$, lo que se gradúa es cuánto se adelanta un experto con mucho histórico ($w \to 1$) frente a uno recién creado ($w \approx 0{,}8$), que conserva un pequeño tirón hacia el global, más estable. Un K mayor (por ejemplo, las mismas 75 partidas del umbral) haría que los expertos recién creados se apoyaran más en el global.

4.4. Modelo residual: LSTM probabilística

4.4.1. Arquitectura

La LSTM no parte del texto en bruto, sino de la secuencia histórica del código (Sección 4.2): para cada paso temporal recibe el embedding de texto de la partida (384 números), su capítulo y su constructora, y unas variables numéricas alineadas en el tiempo. La Figura 4.4 resume cómo se combinan.

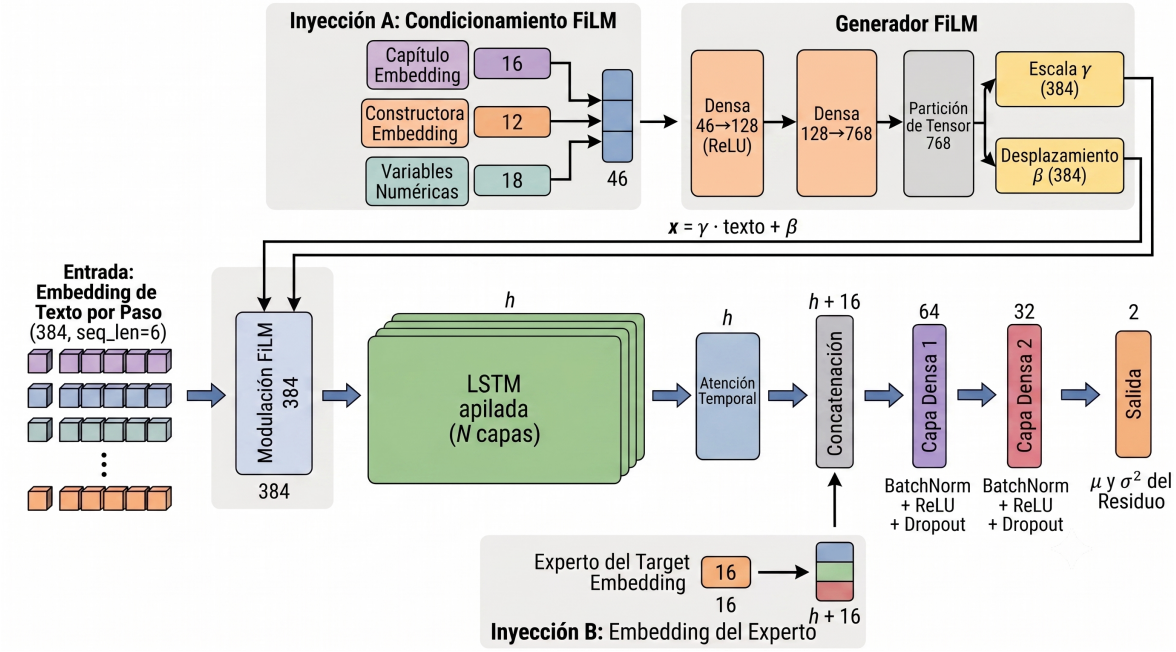

Figura 4.4: Arquitectura de la LSTM probabilística. El texto de cada paso se modula (FiLM) con el capítulo, la constructora y las variables numéricas; la secuencia pasa por una LSTM apilada y una atención temporal que la resume en un contexto; ese contexto, junto al embedding del experto, alimenta una cabeza que produce la media y la varianza del residuo escalado.

El primer bloque es una modulación FiLM (Feature-wise Linear Modulation) [26], que sirve para que el contexto de la partida ajuste el texto antes de procesarlo. A partir del capítulo, la constructora y las variables numéricas, una pequeña red produce dos vectores del tamaño del embedding de texto, una escala γ y un desplazamiento β, y el texto se ajusta multiplicando y sumando,

$$x = \gamma \cdot \text{texto} + \beta.$$

El factor γ sube o baja el peso de cada parte del texto y β lo corre, de modo que una misma descripción se interpreta distinto según el capítulo o la constructora. La modulación arranca casi sin efecto ($\gamma \approx 1$, $\beta \approx 0$) y se va aprendiendo durante el entrenamiento.

Las variables numéricas que entran en ese contexto, las meta, se resumen en la Tabla 4.2: 18 valores por paso con el residuo anterior y la inercia de los residuos, varias señales de tiempo, el tamaño y la posición de la partida, un par de banderas y un resumen de cada aparición del código.

Grupo	Variables y qué aportan
Residuo e inercia	`Residuo_Anterior`, `Momentum` y `Momentum_Codigo`: el residuo de la aparición anterior y la tendencia reciente de los residuos, tanto del mercado como del propio código.
Tiempo	`dt_same_code` (cuánto hace que no se veía el código), `year_scaled`, `years_since_start` y `month_sin/month_cos` (el mes en forma cíclica).
Partida	`Cantidad_Log` (tamaño de la medición) y `Posicion_Relativa` (lugar de la partida dentro del presupuesto).
Señales	`Es_Target_Context` y `Es_Cold_Start`: banderas de si el paso es el objetivo y de si la secuencia es de arranque en frío.
Resumen de la fecha	`Hist_Fecha_Residuo_*` (media, desviación, mínimo y máximo de los residuos) y el número de ofertas y de constructoras presentes en esa aparición.

Tabla 4.2: Variables numéricas (meta) que recibe la LSTM en cada paso, agrupadas (18 valores en total).

Con el texto ya modulado, la secuencia entra en una LSTM apilada, que la recorre paso a paso y va arrastrando memoria: cada paso actualiza un estado interno que resume el histórico hasta ese momento.

Sobre las salidas de la LSTM se aplica una atención temporal, la capa que aparece en el diagrama justo después de la parte recurrente. En lugar de fiarse solo del último paso, una pequeña capa puntúa cada paso válido del histórico; esas puntuaciones pasan por un softmax que las convierte en pesos que suman uno (los pasos de relleno se descartan), y el contexto final es la media ponderada de los pasos con esos pesos. Los pesos dejan que la red dé más importancia a las apariciones del código más informativas, por lo común las más recientes o las más parecidas a la que se quiere predecir, y no a todas por igual.

Por último, ese vector de contexto se concatena con un embedding del experto del target (16 números) y entra en la cabeza, una pequeña red densa que produce las dos salidas finales: la media μ y la varianza σ^2 del residuo, en la escala normalizada de la Sección 4.2. El experto no crea cabezas separadas: hay una sola cabeza, compartida por todos, y ese embedding es lo único que aporta el experto a la red, su contexto. Una cabeza compartida aprende de todas las partidas a la vez y no fragmenta los datos, así que los expertos con poco histórico no se quedan sin una cabeza fiable; el embedding del experto basta para que esa única cabeza se adapte a cada uno, con muchos menos

parámetros que mantener una red por experto.

4.4.2. Función de pérdida

La LSTM se entrena con una pérdida en dos partes. La principal es la log-verosimilitud negativa de una normal (NLL gaussiana), la misma idea que en CatBoost pero ahora sobre el residuo escalado: la red da una media μ y una varianza σ^2, y la pérdida mide cómo de bien esa normal explica el residuo real r de la partida:

$$\mathcal{L}_{\text{NLL}} = \tfrac{1}{2}\left(\ln \sigma^2 + \frac{(r-\mu)^2}{\sigma^2}\right).$$

Como en CatBoost, el modelo aprende a la vez la media y una varianza que se ajusta al tamaño real de sus errores.
A esa pérdida se le suma, solo en algunos expertos, un término de error relativo. El motivo es el mismo que en CatBoost (Sección 4.3): la NLL vive en el espacio del residuo y no controla bien el error porcentual, que es el que más pesa en las partidas baratas. Los expertos a los que se aplica se eligen con el mismo criterio que allí (precios bajos con error relativo alto), aquí con un mínimo de 100 partidas de entrenamiento. A ellos se les añade un sMAPE capado, calculado en euros y ponderado por un coeficiente λ:

$$\mathcal{L} = \mathcal{L}_{\text{NLL}} + \lambda \cdot \text{sMAPE}.$$

Para ese sMAPE se deshace el escalado del residuo, se suma a la predicción de CatBoost para recomponer el precio y se compara con el real; el valor se capa a 2 para que las partidas con error enorme no acaparen el entrenamiento. En el resto de expertos la pérdida es solo la NLL.

4.4.3. Regularización y parada anticipada

La LSTM tiene muchos parámetros, así que el sobreajuste se controla por varias vías:

- Dropout, aplicado en el generador FiLM, en la entrada, entre las capas de la LSTM y en la cabeza.
- Weight decay (regularización L2) de 10^{-4} en el optimizador (Adam), que penaliza los pesos grandes.
- Recorte de gradiente (gradient clipping) a norma máxima 1, para que un gradiente puntual muy grande no desestabilice el entrenamiento.

La tasa de aprendizaje no se mantiene constante durante el entrenamiento: se reduce a la mitad cuando la NLL de validación no mejora durante tres épocas (con el planificador

`ReduceLROnPlateau`), hasta un mínimo de 10^{-5}. Esto suele desatascar el entrenamiento cuando Adam con una tasa de aprendizaje fija se estanca.
La parada anticipada es doble. Por un lado, el entrenamiento se detiene si la NLL de validación no mejora durante 12 épocas. Por otro, mientras entrena se guardan dos versiones del modelo: la de mejor NLL de validación y la de mejor `business_score` (una métrica de negocio que combina el error en euros, la anchura de los intervalos y la cobertura). Al terminar, se elige entre las dos la que tenga mejor `business_score`. La NLL decide cuándo parar, pero la métrica de negocio decide qué modelo se queda.

4.4.4. Búsqueda de hiperparámetros con Optuna

Igual que CatBoost, la LSTM ajusta sus hiperparámetros con Optuna, con tres diferencias respecto a la búsqueda de aquel modelo (Sección 4.3). La primera es el objetivo: aquí no se minimiza un error a secas, sino el `business_score`, una métrica que combina el error en euros, la anchura de los intervalos y la cobertura; así la búsqueda premia a la vez acertar el precio y dar intervalos útiles. La segunda es que la búsqueda es de baja fidelidad: cada combinación se entrena solo 30 épocas (el modelo definitivo se entrena con 500 y parada anticipada) y un pruner Hyperband corta pronto las combinaciones que van mal, para que entren más pruebas en el presupuesto (hasta 80 pruebas o 1,5 horas). La tercera es que solo se exploran los seis hiperparámetros de más impacto (Tabla 4.3); el resto (weight decay, recorte de gradiente, tamaño de lote, etc.) se dejan fijos.

Hiperparámetro	Rango explorado	Qué controla
`hidden_dim`	{64, 96, 128, 192}	tamaño del estado oculto (capacidad)
`num_layers`	1–3	número de capas de la LSTM
`dropout`	0,1–0,4	regularización
`learning_rate`	0,0001–0,003	tasa de aprendizaje
`var_floor`	0–0,10	suelo de la varianza (anchura del intervalo)
`relative_loss_weight`	0,1–0,5	peso del término de error relativo

Tabla 4.3: Espacio de búsqueda de Optuna para la LSTM.

4.4.5. Manejo de cobertura temporal

La LSTM no corrige todas las partidas: solo aquellas cuyo código tiene histórico que aprovechar. Por eso, cada predicción final lleva una etiqueta de origen que indica de dónde sale:

- `lstm_temporal`: el código tiene suficiente histórico (al menos dos fechas, `min_history`; Sección 4.2), así que se construye su secuencia y la LSTM aplica su corrección.

- `catboost_fallback`: el código no tiene histórico suficiente, o es nuevo. No hay secuencia, la LSTM no interviene y la predicción es solo la de CatBoost.

4.5. Calibración de la incertidumbre

La predicción ya tiene un precio y una incertidumbre, pero dos cosas no están garantizadas: que la corrección de la LSTM convenga aplicarla entera en todos los expertos, y que la anchura de los intervalos refleje la confianza real. La calibración ajusta ambas sobre el conjunto de validación, en tres pasos encadenados: el peso de la corrección (α), la anchura de los intervalos (σ) y una capa conformal.

4.5.1. Calibración α por experto

La LSTM aporta una corrección sobre la predicción de CatBoost, pero no siempre conviene aplicarla entera: en unos expertos ayuda mucho y en otros poco. Por eso se introduce un peso α por experto, entre 0 y 1, que gradúa cuánta corrección se aplica. Todo se combina en la escala logarítmica del precio (`log1p`, Sección 4.2): la predicción en esa escala, $\hat{y}$, es la base de CatBoost más la corrección de la LSTM ponderada,

$$\hat{y} = \mu_{\text{CatBoost}} + \alpha \cdot \mu_{\text{LSTM}},$$

y el precio en euros se recupera con $\hat{p} = e^{\hat{y}} - 1$. Se usa $\log(1 + \text{precio})$, y no $\log(\text{precio})$, para que los precios muy bajos o nulos no den problemas (el logaritmo normal se dispara cerca de cero). Para cada experto se prueba α en una rejilla (de 0 a 1 en pasos de 0,1) y se elige el valor que minimiza el error absoluto medio (MAE, en euros) sobre la validación. Con $\alpha = 0$ la corrección se desactiva (queda la predicción de CatBoost) y con $\alpha = 1$ se aplica completa; si un experto no tiene datos de validación, su α es 0.

4.5.2. Calibración σ por experto

El modelo también da una incertidumbre σ, que fija la anchura del intervalo de precios. El problema es que esa σ en bruto no suele dar una cobertura fiable: si el intervalo se anuncia al 90 %, el precio real debería caer dentro el 90 % de las veces, pero a veces sale demasiado ancho o demasiado estrecho.

Para ajustarlo se multiplica σ por un factor, probando varios en una rejilla, y se elige el que deja la cobertura real (PICP) cerca del 90 % con el intervalo más estrecho posible (menor MPIW). El factor es por experto si tiene suficientes datos de validación (al menos 80 partidas); si no, se usa uno global. El intervalo se forma en la escala logarítmica, $\hat{y} \pm z\,\sigma$ (con z el valor de la normal para el 90 %), y se transforma a euros igual que la predicción:

$$\text{Lower} = e^{\hat{y} - z\,\sigma} - 1, \qquad \text{Upper} = e^{\hat{y} + z\,\sigma} - 1.$$

4.5.3. Predicción conformal

Como capa final se añade una calibración conformal [31], que da una garantía de cobertura más sólida sin suponer que los errores siguen una normal. Sobre la validación se calcula, para cada partida, una puntuación de no conformidad

$$s = \frac{|\log(1 + p_{\text{real}}) - \hat{y}|}{\sigma},$$

es decir, a cuántas sigmas está el precio real de la predicción. El umbral conformal $\hat{q}$ es el cuantil de esas puntuaciones al nivel de cobertura deseado (90 %). No son lo mismo: s es una puntuación por partida, mientras que $\hat{q}$ es un único valor (el cuantil que deja por debajo el 90 % de esas puntuaciones). Para que se adapte al rango de precios, $\hat{q}$ se calcula por bandas de precio (0–50, 50–100, ..., más de 1000 euros); si una banda tiene pocos datos (menos de 20 partidas), usa el umbral global. Este umbral no cambia la predicción: se usa en el semáforo (Sección 4.7) para que, cuando el precio ofertado caiga dentro del intervalo conformal, un aviso de sobrecoste se rebaje a la categoría "modelo no confiable", evitando falsas alarmas.

4.6. Validación cruzada temporal con ventana expansiva

El esquema de validación es la validación cruzada temporal de ventana expansiva ya descrita en el Capítulo 3 (Sección 3.6 y Figura 3.3): se prueban $N \in \{2, 3, 4\}$ folds, cada uno entrena con las fechas anteriores y valida con la siguiente, con el test siempre fijo. Esta sección no repite ese esquema, sino que explica cómo lo aprovecha el pipeline: para generar las predicciones fuera de muestra que calibran el modelo, para predecir el test con un ensemble y para ponderar los folds en la búsqueda de hiperparámetros.

4.6.1. Predicciones OOF

Cada fold predice su propia fecha de validación, que no ha visto al entrenar. Al juntar (concatenar) las predicciones de validación de todos los folds se obtiene un conjunto OOF (out-of-fold): predicciones limpias para un tramo amplio de fechas, cada una hecha por un modelo que no la usó para entrenar. La calibración (Sección 4.5) se ajusta sobre ese OOF, y no sobre el test, para no contaminar la evaluación final.

4.6.2. Predicción final y modelo de producción

Para evaluar en test no se usa un solo fold: se promedian las predicciones de los N modelos (un ensemble), tanto en CatBoost como en la LSTM, lo que da una estimación más estable que un único modelo. Sobre ese promedio se aplican las calibraciones

ajustadas en la OOF. El modelo que se guarda para producción es el del último fold, por ser el que ha entrenado con más datos.

4.7. Sistema semáforo de auditoría de sobrecostes

La última etapa convierte la estimación en una señal de auditoría. Para cada partida compara el precio ofertado por la constructora con el rango que predice el sistema y la marca con un color. Es un filtro de atención, no un auditor autónomo: prioriza qué partidas debería revisar una persona, en lugar de decidir por sí solo si hay sobrecoste. La señal base se calcula con un Z-score. En las partidas donde el precio ofertado supera al predicho (hay sobrecoste), se mide a cuántas desviaciones está ese precio por encima de la predicción:

$$Z = \frac{\log(1 + \text{ofertado}) - \hat{y}}{\sigma},$$

con $\hat{y}$ la predicción y σ su incertidumbre, en la escala logarítmica. El color depende de Z: verde si no hay sobrecoste o $Z < 1$, amarillo si $1 \leq Z < 2$ y rojo si $Z \geq 2$ (Figura 4.5).

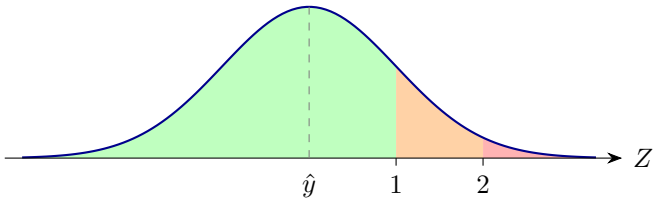

Figura 4.5: Semáforo sobre la distribución predicha. El precio se modela (en escala logarítmica) como una normal centrada en la predicción $\hat{y}$ y con desviación σ; el color depende de en qué parte de la cola cae el precio ofertado: verde hasta $+1\sigma$ ($Z < 1$), amarillo entre $+1\sigma$ y $+2\sigma$, y rojo más allá ($Z \geq 2$). Las degradaciones posteriores solo pueden hacerlo más prudente.

Sobre esa base se aplican cinco degradaciones que vuelven la señal más prudente:

- Si la incertidumbre σ es grande (confianza baja), no se permite el verde: pasa a amarillo.
- Si la predicción no tiene histórico (viene solo de CatBoost), tampoco se permite el verde.
- Si el sobrecoste relativo es muy alto (más del 50 %), se fuerza el rojo.
- Si la predicción es de arranque en frío y saldría roja, se rebaja a “sin confianza”, porque no hay señal histórica fiable detrás.
- Si el precio ofertado, pese a dar un Z alto, cae dentro del intervalo conformal (Sección 4.5), el rojo se rebaja a “modelo no confiable”: el aviso se debe más al error del modelo que a un sobrecoste real.

Los umbrales ($Z = 1$ y $Z = 2$ y el 50 % de sobrecoste relativo) se fijan en la configuración. El resultado son cinco etiquetas posibles (verde, amarillo, rojo, sin confianza y modelo no confiable) con las que el equipo de retail centra su revisión en las partidas más sospechosas.

Capítulo 5

Experimentos y resultados

5.1. Métricas de evaluación

Antes de los resultados se fijan las métricas. Se agrupan en tres familias: el error del punto estimado, la calidad de los intervalos y una métrica de negocio que resume las anteriores. En todas, N es el número de partidas, y_i el precio real, $\hat{y}_i$ el precio predicho y $[\ell_i, u_i]$ su intervalo.

5.1.1. Error del punto estimado

El error absoluto medio y la raíz del error cuadrático medio,

$$\mathrm{MAE} = \frac{1}{N}\sum_i |y_i - \hat{y}_i|, \qquad \mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_i (y_i - \hat{y}_i)^2},$$

miden el error en euros; la RMSE penaliza más los errores grandes. El WAPE divide el error absoluto total por el importe total,

$$\mathrm{WAPE} = \frac{\sum_i |y_i - \hat{y}_i|}{\sum_i |y_i|},$$

y es el error relativo del conjunto: un WAPE del 11 % quiere decir que la suma de lo estimado se desvía un 11 % de la suma real. Se prefiere al MAPE, que se dispara cuando hay precios muy bajos. El coeficiente de determinación $R^2 = 1 - \sum_i (y_i - \hat{y}_i)^2 / \sum_i (y_i - \bar{y})^2$ resume cuánta varianza del precio explica el modelo. Cuando interesa la escala logarítmica se usan las mismas MAE y RMSE sobre $\log(1 + \text{precio})$ (la RMSE en esa escala es el RMSLE).

5.1.2. Calidad de los intervalos

En los intervalos importan dos cosas: que cubran lo que prometen y que sean estrechos. La cobertura se mide con el PICP (fracción de precios reales dentro del intervalo) y la anchura con el MPIW (anchura media, en euros) y el rMPIW (anchura relativa al precio):

$$\mathrm{PICP} = \frac{1}{N}\sum_i \mathbf{1}[\ell_i \le y_i \le u_i], \qquad \mathrm{MPIW} = \frac{1}{N}\sum_i (u_i - \ell_i),$$

$$\mathrm{rMPIW} = \frac{1}{N}\sum_i \frac{u_i - \ell_i}{\max(|y_i|,\, 0{,}01)}.$$

Un buen intervalo tiene un PICP cercano al nivel nominal (aquí el 90 %) con el menor MPIW posible: cubrir de más a costa de intervalos enormes no informa. Las partidas sin intervalo (las de arranque en frío, que caen al fallback de CatBoost) se excluyen de estas tres métricas.

5.1.3. Métrica de negocio

Una última métrica resume las anteriores para elegir y comparar modelos:

$$\text{business score} = \text{MAE} + 0{,}1 \cdot \text{MPIW} + 500 \cdot \max(0,\ 0{,}90 - \text{PICP}).$$

Combina el error, la anchura de los intervalos y un castigo fuerte por quedarse por debajo de la cobertura objetivo del 90 %. Cuanto menor, mejor. Es la métrica que guía la parada anticipada y la búsqueda de hiperparámetros.

5.2. Estudio de ablación

Este apartado mide qué aporta cada pieza del sistema. Salvo que se indique lo contrario, los números son los del modelo principal (la configuración de 2 folds, justificada en la Sección 5.2.3) sobre el conjunto de test.

5.2.1. Aportación de cada etapa

La Tabla 5.1 compara tres versiones acumulativas: solo CatBoost, CatBoost más la corrección de la LSTM (sin calibrar) y el sistema final (con calibración).

Modelo	MAE (EUR)	WAPE	RMSE (EUR)	R^2
CatBoost base	34,04	11,26 %	236,52	0,8903
CatBoost + LSTM (sin calibrar)	33,99	11,24 %	237,02	0,8898
Final (calibrado)	33,88	11,21 %	236,39	0,8904

Tabla 5.1: Aportación de cada etapa, sobre el conjunto de test.

La tabla muestra que CatBoost, partiendo solo del texto y las variables categóricas, ya alcanza un buen nivel (un R^2 de 0,89 y un WAPE del 11 %), por lo que el margen de mejora en el punto es reducido. Sobre esa base, la LSTM cumple dos funciones.
La primera es corregir la estimación aprovechando el histórico del código, algo que CatBoost no puede hacer porque trata cada partida por separado. La corrección se aplica solo en los expertos donde la validación indica que es útil (peso α por experto, activo en el 49,7 % de las partidas), de modo que el sistema no empeora al modelo base: en promedio lo iguala o lo mejora en todas las métricas. En este conjunto la mejora

del error medio es pequeña (MAE de 34,04 a 33,88 euros), lo que es coherente con la limitación de los datos, pues hay pocos códigos con histórico amplio, que es justo lo que la red recurrente necesita para aportar más.
La segunda función, y la más importante en este trabajo, es sostener la parte probabilística del sistema: la corrección residual de la LSTM y su calibración por experto son las que producen los intervalos de confianza y la señal del semáforo. Esa contribución se analiza en los apartados siguientes.

5.2.2. Efecto de la calibración por experto

La calibración no busca mejorar el punto, sino que los intervalos cubran lo que prometen con la menor anchura posible, y lo hace experto a experto. La Tabla 5.2 compara, en los expertos con suficientes partidas, la cobertura y la anchura del intervalo antes (solo CatBoost, con σ global) y después (sistema final, con σ por experto y conformal).

Experto	n	PICP base	PICP final	MPIW base	MPIW final
1	784	0,98	0,94	57,8	34,7
2	102	0,86	0,99	88,5	211,7
3	702	0,82	0,99	93,6	255,0
5	106	0,88	0,93	38,0	43,1
7	126	0,92	1,00	295,0	647,2
10	66	0,86	0,95	86,1	116,7

Tabla 5.2: Calidad del intervalo por experto, antes y después de la calibración (test). PICP objetivo del 90 %; MPIW en euros (menor es mejor a igual cobertura).

La calibración corrige en las dos direcciones (Figura 5.1): donde CatBoost se quedaba corto de cobertura, ensancha el intervalo hasta alcanzar el objetivo (el experto 3 pasa de cubrir el 82 % al 99 %), y donde cubría de sobra con intervalos anchos, los estrecha (el experto 1 baja de una anchura media de 57,8 a 34,7 euros, un 40 % menos, manteniendo la cobertura). Así, cada experto queda calibrado según su propio comportamiento, algo que un único factor global no consigue. El experto 7 es un caso de sobre-cobertura: ya cubría el 92 %, por encima del objetivo, y la calibración (ajustada en validación y con el margen del conformal) lo lleva al 100 % en test ensanchando el intervalo; con solo 126 partidas y errores muy dispersos, basta poco para que el PICP suba. La tabla recoge solo los expertos con suficientes datos: deja fuera al experto 4 (sin partidas en test) y a los expertos 6, 8 y 9, que con 49, 5 y 2 partidas tienen métricas de intervalo poco fiables.

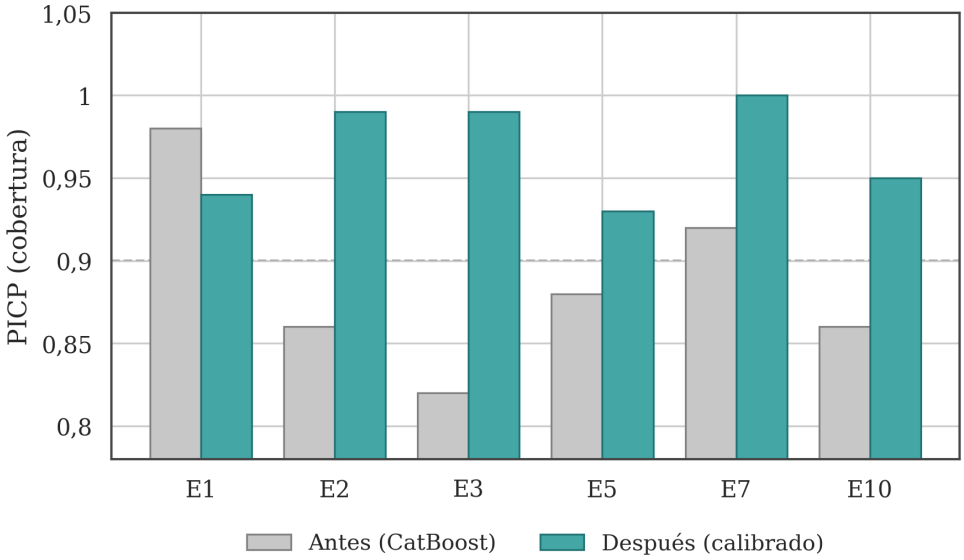

Figura 5.1: Cobertura (PICP) por experto antes y después de la calibración. La línea discontinua marca el objetivo del 90 %. La calibración acerca todos los expertos al objetivo o lo supera.

5.2.3. Sensibilidad al número de folds

El número de folds de la validación temporal ($N \in \{2, 3, 4\}$) cambia el modelo final, porque este es un ensemble de los N folds y porque, con ventana expansiva, más folds dejan menos fechas de arranque para los primeros (Sección 4.6). La Tabla 5.3 compara las tres configuraciones.

Folds	MAE (EUR)	WAPE	R^2	PICP	MPIW (EUR)	Business
2	33,88	11,21 %	0,8904	95,80 %	395,9	73,47
3	35,00	11,58 %	0,8895	89,86 %	510,3	86,73
4	36,20	11,98 %	0,8888	99,22 %	416,0	77,80

Tabla 5.3: Sensibilidad al número de folds, sobre el conjunto de test.

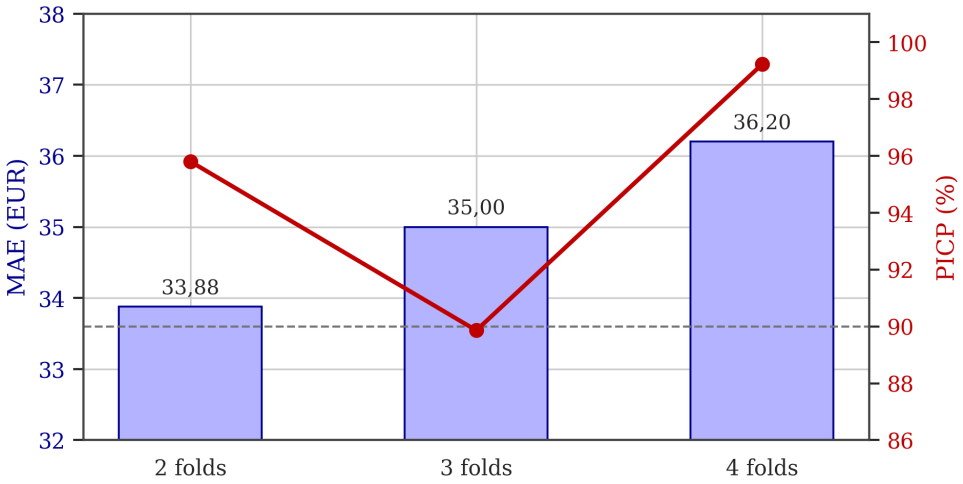

Figura 5.2: Sensibilidad al número de folds. Barras azules: MAE (eje izquierdo). Línea roja: PICP (eje derecho); la discontinua gris marca el objetivo del 90 %. El error crece de forma monótona con el número de folds, mientras que 2 folds combina el menor MAE con una cobertura por encima del objetivo.

El error crece de forma monótona con el número de folds (MAE de 33,88 con 2 folds a 36,20 con 4, Figura 5.2). La razón es el tamaño de los datos: con pocas fechas, partir en más folds deja a los primeros con muy poco histórico para entrenar, y esos modelos débiles diluyen el ensemble. Por eso se adopta la configuración de 2 folds como modelo principal, con un MAE significativamente menor que el de 3 y 4 folds (Sección 5.6). Su único pero es que sobre-cubre un poco (PICP 95,8 % frente al 90 % objetivo), es decir, intervalos algo más anchos de lo necesario, lo que es el lado seguro.

Cada configuración ejecuta su propia búsqueda de Optuna, así que los hiperparámetros elegidos cambian de una a otra. Las Tablas 5.4 y 5.5 recogen los seleccionados para CatBoost y para la LSTM en cada caso.

Hiperparámetro	2 folds	3 folds	4 folds
`iterations`	1800	1800	1800
`depth`	6	7	6
`learning_rate`	0,024	0,056	0,024
`l2_leaf_reg`	9,52	10,25	9,52
`random_strength`	0,42	0,03	0,42
`min_data_in_leaf`	1	23	1
`leaf_estimation_iterations`	5	6	5
`border_count`	128	128	128
`bootstrap_type`	Bayesian	Bernoulli	Bayesian

Tabla 5.4: Hiperparámetros de CatBoost elegidos por Optuna en cada configuración.

Hiperparámetro	2 folds	3 folds	4 folds
`hidden_dim`	128	192	96
`num_layers`	1	1	1
`dropout`	0,20	0,36	0,23
`learning_rate`	$1{,}2 \cdot 10^{-4}$	$2{,}9 \cdot 10^{-4}$	$1{,}7 \cdot 10^{-4}$
`var_floor`	0,035	0,081	0,054
`relative_loss_weight`	0,34	0,49	0,31

Tabla 5.5: Hiperparámetros de la LSTM elegidos por Optuna en cada configuración.

La LSTM elige siempre una sola capa y una tasa de aprendizaje del orden de 10^{-4} (muy por debajo del valor de partida), en las tres configuraciones; el resto de valores varía con cada búsqueda, como cabe esperar en un conjunto pequeño.
El valor del histórico disponible, que es lo que la LSTM intenta aprovechar, se analiza de forma agregada por cobertura de código en la Sección 5.3.2.

5.3. Resultados por segmento

Más allá del agregado, interesa ver dónde acierta más o menos el sistema. Se analiza por banda de precio y por disponibilidad de histórico.

5.3.1. Por banda de precio

La Tabla 5.6 desglosa el error y los intervalos por banda de precio.

Banda (EUR)	n	MAE (EUR)	WAPE	PICP	rMPIW
0–50	888	2,32	9,86 %	94,30 %	112,1 %
50–100	266	6,09	8,69 %	96,37 %	71,2 %
100–200	270	10,17	7,03 %	99,16 %	69,3 %
200–500	238	26,89	8,54 %	98,59 %	67,3 %
500–1000	130	45,93	6,26 %	94,83 %	39,1 %
1000+	150	313,33	13,89 %	94,44 %	171,4 %

Tabla 5.6: Resultados por banda de precio (test, modelo de 2 folds). Las bandas se definen según el precio real de la partida.

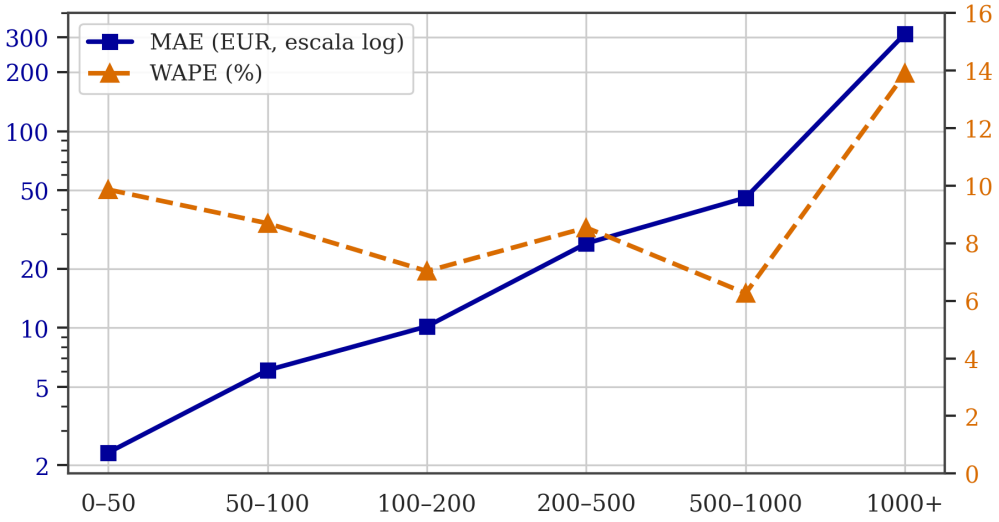

Figura 5.3: Error por banda de precio. El error absoluto (MAE, en escala logarítmica) crece con el precio, mientras que el relativo (WAPE) se mantiene entre el 6 y el 14 %.

El error absoluto crece con el precio (Figura 5.3), de unos pocos euros en las partidas más baratas a varios cientos en las de más de 1000, como es lógico; pero el error relativo (WAPE) se mantiene moderado en todas las bandas (entre el 6 y el 14 %). La cobertura es buena en todo el rango (PICP del 94 al 99 %). El punto débil es la banda de más de 1000 euros: hay pocas partidas, muy dispares, y el modelo lo refleja con intervalos muy anchos (rMPIW del 171 %), declarando mucha incertidumbre donde de verdad la hay.

5.3.2. Por disponibilidad de histórico

La mayor diferencia no la marca el precio, sino si el código ya se había visto. Las partidas cuyo código tiene histórico (las que corrige la LSTM) logran un MAE de 25,9 euros, un R^2 de 0,92 y una cobertura del 96 %. Las de código nuevo, que caen al modelo base de CatBoost sin secuencia, empeoran hasta un MAE de 122,6 euros, un R^2 de 0,37 y una cobertura del 76 % (Figura 5.4). Es el coste del arranque en frío: sin histórico ni código conocido, la estimación es mucho peor. La diferencia es estadísticamente significativa (Sección 5.6) y confirma el valor de la parte temporal donde hay datos; además, señala la principal vía de mejora: acumular más histórico por código.

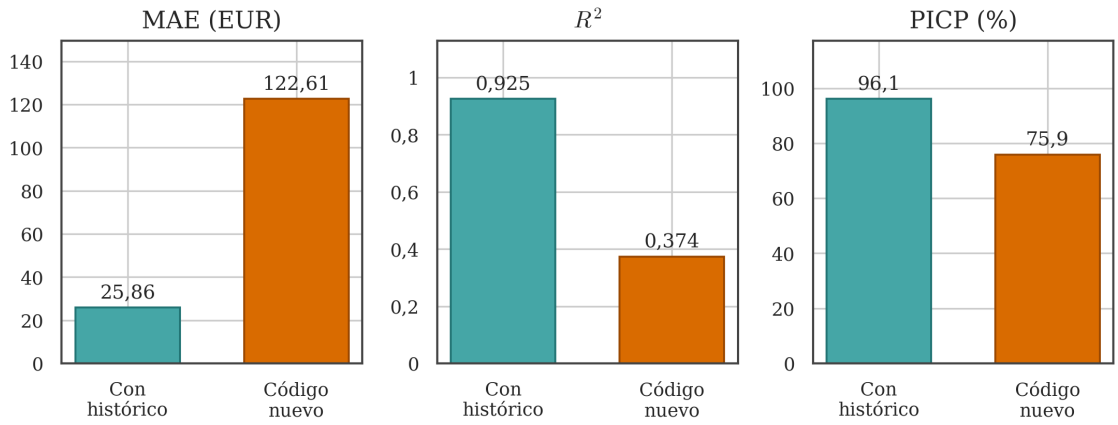

Figura 5.4: Rendimiento según haya o no histórico del código. Las partidas con código visto en el entrenamiento (verde azulado) se predicen mucho mejor que las de código nuevo (naranja): en error de punto (MAE), en ajuste (R^2) y en cobertura (PICP).

5.4. Sistema semáforo

El semáforo (Sección 4.7) se evalúa sobre las partidas de test que traen precio ofertado. De las 1784 partidas con precio para auditar, el 92 % salen verdes, el 4 % amarillas y el 2 % rojas; un 2 % adicional se marca como "modelo no confiable" (un rojo que el ajuste conformal rebaja por dudoso) y ninguna queda "sin confianza" (Figura 5.5). El semáforo concentra así la atención en una minoría (en torno al 6 % entre amarillo y rojo): el equipo de retail no tiene que revisarlo todo, solo esos casos.

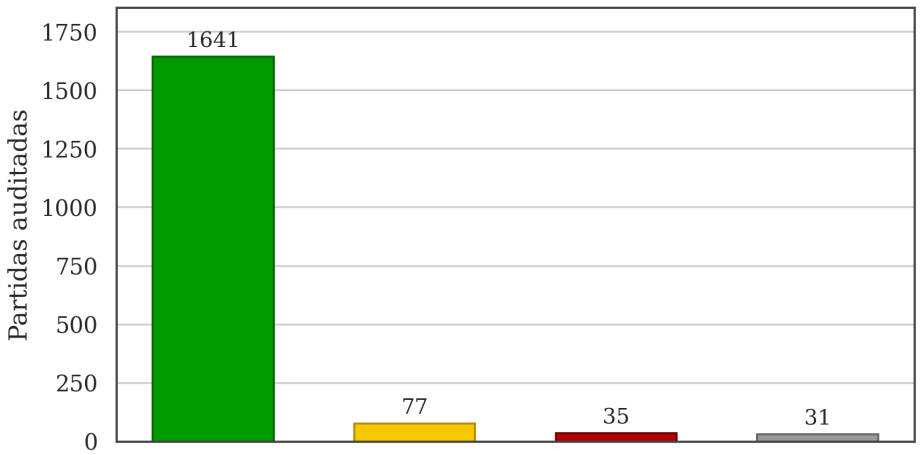

Figura 5.5: Reparto del semáforo sobre las 1784 partidas con precio ofertado: verde, amarillo, rojo y, en gris, "modelo no confiable". La gran mayoría salen en verde; el sistema concentra la revisión en el 6 % restante (amarillo y rojo).

5.5. Comparación con un baseline

Para situar el rendimiento se compara con un baseline sencillo: un LightGBM entrenado sobre el mismo CSV, sin la parte de texto, ni los expertos, ni la calibración. La Figura 5.6

enfrenta ambos modelos en las cuatro métricas de punto sobre test: el sistema propuesto gana en todas, con un margen amplio (el WAPE baja del 37 al 11 %, más del triple de error). La comparación es indicativa (el baseline no está homologado al detalle con el mismo preprocesado), pero la ventaja queda muy fuera de la variabilidad muestral del sistema (Sección 5.6), de modo que deja clara la aportación del modelo completo frente a un gradient boosting estándar.

Métrica	Sistema	LightGBM
MAE (EUR)	33,88	111,01
RMSE (EUR)	236,39	390,19
WAPE	11,21 %	36,72 %
R^2	0,890	0,701

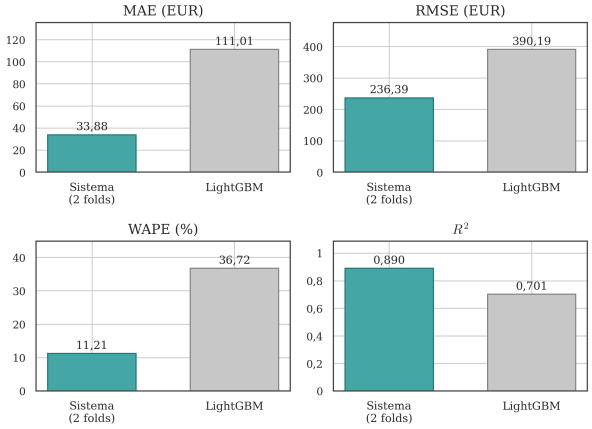

Figura 5.6: Comparación con el baseline LightGBM en las cuatro métricas de punto (test): tabla con los valores (izquierda) y su representación (derecha). En MAE, RMSE y WAPE menor es mejor; en R^2, mayor es mejor. El sistema propuesto (verde azulado) supera al baseline en todas.

5.6. Significancia estadística

Las tres afirmaciones centrales de este capítulo (el sistema bate al baseline, el código con histórico rinde mucho mejor que el de arranque en frío, y 2 folds es la mejor configuración) se apoyan en diferencias de métrica, así que hay que comprobar que no se deben al azar del muestreo. Se usan dos contrastes no paramétricos, apropiados para errores de regresión con cola pesada: un bootstrap por remuestreo de las partidas de test (10 000 réplicas) para los intervalos de confianza de las métricas y de sus diferencias, y la prueba U de Mann-Whitney para comparar dos grupos independientes. La Tabla 5.7 resume los resultados.

Afirmación	Contraste	Resultado
Sistema < baseline (WAPE)	bootstrap IC 95 %	11,2 % [8,4–14,4] vs 36,7 % (fuera)
Histórico < arranque frío (error abs.)	Mann-Whitney U	$p < 0{,}001$
2 folds < 3 folds (MAE, pareado)	bootstrap dif. IC 95 %	+1,12 [+0,25, +2,05]
2 folds < 4 folds (MAE, pareado)	bootstrap dif. IC 95 %	+2,33 [+1,07, +3,60]

Tabla 5.7: Contrastes estadísticos de las tres afirmaciones centrales (test). Los intervalos de confianza por bootstrap usan 10 000 réplicas; las diferencias de MAE entre folds son pareadas sobre las mismas partidas (en euros).

El sistema frente al baseline: el WAPE del sistema (11,2 %) tiene un intervalo de confianza del 95 % de [8,4, 14,4] %, que deja al baseline (36,7 %) muy fuera, de modo que la ventaja no es atribuible al muestreo. El histórico frente al arranque en frío: el error de las partidas con código visto es menor, con una diferencia muy significativa (Mann-Whitney, $p < 0{,}001$). La elección de 2 folds: su MAE es menor que el de 3 y 4 folds y los intervalos de confianza de la diferencia pareada excluyen el cero en ambos casos; el margen sobre 3 folds es pequeño (+1,1 euros) y la ventaja sobre 4 folds, más clara (+2,3 euros).

Capítulo 6

Discusión

6.1. Cumplimiento de los objetivos

El objetivo era estimar precios unitarios con una medida de incertidumbre y convertir eso en una ayuda a la auditoría de sobrecostes. Los resultados del Capítulo 5 muestran que el sistema lo consigue. La estimación de punto es buena (un WAPE en torno al 11 % y un R^2 de 0,89). Los intervalos quedan calibrados experto a experto, con una cobertura cercana al objetivo y una anchura razonable, y el semáforo concentra la revisión en una minoría de partidas. La corrección temporal de la LSTM, en cambio, aporta poco al error de punto en este conjunto; su valor está en la capa probabilística y en su potencial con más histórico, no en recortar el MAE hoy.

6.2. Limitaciones

El trabajo tiene varias limitaciones, casi todas ligadas al dato. La principal es que el conjunto es pequeño y procede de un único cliente (McDonald's España), con un fuerte desequilibrio entre obras y constructoras: eso acota cuánto puede generalizar el modelo y cuánto histórico hay por código. De ahí se derivan dos puntos concretos: el arranque en frío, donde los códigos nuevos, sin secuencia, se predicen bastante peor (un MAE de 122 frente a 26 euros; diferencia significativa, Sección 5.6), y la escasez de `Textos_Mediciones` (vacío en el 78 % de las partidas), que limita las variables derivadas de ese campo. En las partidas muy caras (más de 1000 euros) hay pocos datos y muy dispares, de modo que, aunque la cobertura se mantiene, los intervalos salen muy anchos y poco informativos.

6.3. Implicaciones prácticas y trabajo futuro

En la práctica, el sistema encaja como un filtro de atención para el equipo de retail: no decide por sí solo, prioriza qué partidas conviene revisar y acompaña cada estimación de un intervalo que comunica su confianza. La vía de mejora más clara, y la de mayor impacto, es acumular más histórico por código: beneficiaría a la vez a la corrección temporal de la LSTM y al problema del arranque en frío.

Capítulo 7

Conclusiones

Este trabajo ha desarrollado un sistema de estimación probabilística de precios unitarios de obra. Combina un modelo base CatBoost, una corrección residual con LSTM que aprovecha el histórico de cada código y una calibración por experto que produce intervalos de confianza fiables. Sobre esa estimación, un semáforo basado en un Z-score logarítmico señala las partidas con posible sobrecoste como apoyo a la auditoría. La evaluación sobre datos reales (25 046 partidas de presupuestos de retail) muestra una buena precisión de punto (WAPE en torno al 11 %), intervalos calibrados y un semáforo que concentra la revisión en una minoría de partidas. La lectura principal es que, con un conjunto pequeño y de un solo cliente, la mayor parte del acierto la aporta el modelo base; la parte temporal y la calibración no recortan el error medio, pero sí cuantifican bien la incertidumbre.

Bibliografía

[1] T. Akiba, S. Sano, T. Yanase, T. Ohta y M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," en *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019, págs. 2623-2631. DOI: 10.1145/3292500.3330701

[2] Z. H. Ali y A. M. Burhan, "Hybrid Machine Learning Approach for Construction Cost Estimation: An Evaluation of Extreme Gradient Boosting Model," *Asian Journal of Civil Engineering*, vol. 24, n.º 7, págs. 2427-2442, 2023. DOI: 10.1007/s42107-023-00651-z

[3] D. Bahdanau, K. Cho e Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," en *International Conference on Learning Representations*, 2015. dirección: https://arxiv.org/abs/1409.0473

[4] C. Bergmeir, R. J. Hyndman y B. Koo, "A Note on the Validity of Cross-Validation for Evaluating Autoregressive Time Series Prediction," *Computational Statistics & Data Analysis*, vol. 120, págs. 70-83, 2018. DOI: 10.1016/j.csda.2017.11.003

[5] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, n.º 1, págs. 5-32, 2001. DOI: 10.1023/A:1010933404324

[6] V. Chandola, A. Banerjee y V. Kumar, "Anomaly Detection: A Survey," *ACM Computing Surveys*, vol. 41, n.º 3, 15:1-15:58, 2009. DOI: 10.1145/1541880.1541882

[7] T. Chen y C. Guestrin, "XGBoost: A Scalable Tree Boosting System," en *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, págs. 785-794. DOI: 10.1145/2939672.2939785

[8] T. Cordier, V. Blot, L. Lacombe, T. Morzadec, A. Capitaine y N. Brunel, "Flexible and Systematic Uncertainty Estimation with Conformal Prediction via the MAPIE Library," en *Conformal and Probabilistic Prediction with Applications*, ép. Proceedings of Machine Learning Research, 2023. dirección: https://mapie.readthedocs.io/

[9] H. H. Elmousalami, "Artificial Intelligence and Parametric Construction Cost Estimate Modeling: State-of-the-Art Review," *Journal of Construction Engineering and Management*, vol. 146, n.º 1, pág. 03 119 008, 2020. DOI: 10.1061/(ASCE)CO.1943-7862.0001678

[10] B. Flyvbjerg, M. Skamris y S. Buhl, "Underestimating Costs in Public Works Projects: Error or Lie?" *Journal of the American Planning Association*, vol. 68, págs. 279-295, jun. de 2002. DOI: 10.1080/01944360208976273

[11] J. H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," *The Annals of Statistics*, vol. 29, n.º 5, págs. 1189-1232, 2001. DOI: 10.1214/aos/1013203451

[12] Y. Gal y Z. Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," en *Proceedings of the 33rd International Conference on Machine Learning*, ép. Proceedings of Machine Learning Research, vol. 48, 2016, págs. 1050-1059.

[13] T. Gneiting y A. E. Raftery, "Strictly Proper Scoring Rules, Prediction, and Estimation," *Journal of the American Statistical Association*, vol. 102, n.º 477, págs. 359-378, 2007. DOI: 10.1198/016214506000001437

[14] L. Grinsztajn, E. Oyallon y G. Varoquaux, "Why Do Tree-Based Models Still Outperform Deep Learning on Typical Tabular Data?" En *Advances in Neural Information Processing Systems*, vol. 35, 2022, págs. 507-520.

[15] S. T. Hashemi, O. M. Ebadati y H. Kaur, "Cost Estimation and Prediction in Construction Projects: A Systematic Review on Machine Learning Techniques," *SN Applied Sciences*, vol. 2, pág. 1703, 2020. DOI: 10.1007/s42452-020-03497-1

[16] S. Hochreiter y J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, n.º 8, págs. 1735-1780, 1997. DOI: 10.1162/neco.1997.9.8.1735

[17] S. Kaufman, S. Rosset, C. Perlich y O. Stitelman, "Leakage in Data Mining: Formulation, Detection, and Avoidance," *ACM Transactions on Knowledge Discovery from Data*, vol. 6, n.º 4, 15:1-15:21, 2012. DOI: 10.1145/2382577.2382579

[18] G. Ke et al., "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," en *Advances in Neural Information Processing Systems*, vol. 30, 2017, págs. 3146-3154.

[19] A. Kendall e Y. Gal, "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" En *Advances in Neural Information Processing Systems*, vol. 30, 2017, págs. 5574-5584.

[20] A. Khosravi, S. Nahavandi, D. Creighton y A. F. Atiya, "Comprehensive Review of Neural Network-Based Prediction Intervals and New Advances," *IEEE Transactions on Neural Networks*, vol. 22, n.º 9, págs. 1341-1356, 2011. DOI: 10.1109/TNN.2011.2162110

[21] G.-H. Kim, S.-H. An y K.-I. Kang, "Comparison of Construction Cost Estimating Models Based on Regression Analysis, Neural Networks, and Case-Based Reasoning," *Building and Environment*, vol. 39, n.º 10, págs. 1235-1242, 2004. DOI: 10.1016/j.buildenv.2004.02.013

[22] R. Koenker y G. Bassett, "Regression Quantiles," *Econometrica*, vol. 46, n.º 1, págs. 33-50, 1978. DOI: 10.2307/1913643

[23] V. Kuleshov, N. Fenner y S. Ermon, "Accurate Uncertainties for Deep Learning Using Calibrated Regression," en *Proceedings of the 35th International Conference on Machine Learning*, ép. Proceedings of Machine Learning Research, vol. 80, 2018, págs. 2796-2804. dirección: https://proceedings.mlr.press/v80/kuleshov18a.html

[24] B. Lakshminarayanan, A. Pritzel y C. Blundell, "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles," en *Advances in Neural Information Processing Systems*, vol. 30, 2017, págs. 6402-6413.

[25] N. Meinshausen, "Quantile Regression Forests," *Journal of Machine Learning Research*, vol. 7, n.º 35, págs. 983-999, 2006. dirección: https://www.jmlr.org/papers/v7/meinshausen06a.html

[26] E. Perez, F. Strub, H. de Vries, V. Dumoulin y A. Courville, "FiLM: Visual Reasoning with a General Conditioning Layer," en *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018. DOI: 10.1609/aaai.v32i1.11671

[27] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush y A. Gulin, "CatBoost: Unbiased Boosting with Categorical Features," en *Advances in Neural Information Processing Systems*, vol. 31, 2018, págs. 6638-6648.

[28] Y. Romano, E. Patterson y E. J. Candès, "Conformalized Quantile Regression," en *Advances in Neural Information Processing Systems*, vol. 32, 2019, págs. 3538-3548.

[29] R. Shwartz-Ziv y A. Armon, "Tabular Data: Deep Learning is Not All You Need," *Information Fusion*, vol. 81, págs. 84-90, 2022. DOI: 10.1016/j.inffus.2021.11.011

[30] L. J. Tashman, "Out-of-Sample Tests of Forecasting Accuracy: An Analysis and Review," *International Journal of Forecasting*, vol. 16, n.º 4, págs. 437-450, 2000. DOI: 10.1016/S0169-2070(00)00065-0

[31] V. Vovk, A. Gammerman y G. Shafer, *Algorithmic Learning in a Random World*. Springer, 2005. DOI: 10.1007/b106715

[32] S. Yun, "Performance Analysis of Construction Cost Prediction Using Neural Network for Multioutput Regression," *Applied Sciences*, vol. 12, n.º 19, pág. 9592, 2022. DOI: 10.3390/app12199592

Apéndice A

Reproducibilidad

El código completo del trabajo está disponible en https://github.com/JaviCeronn/Estimador-Precios-Presupuestos_Modelos.

Entorno

- Gestor de paquetes: `uv`.
- Python 3.14.3 (véase `.python-version`).
- Versiones exactas de todas las dependencias en `uv.lock`.
- GPU: NVIDIA RTX 3050 Laptop (4 GB VRAM), CUDA 12.6.

Instalación

Listing A.1: Instalación del entorno

```
# Clonar el repositorio
git clone https://github.com/JaviCeronn/Estimador-Precios-Presupuestos_Modelos.git
cd Estimador-Precios-Presupuestos_Modelos

# Instalar dependencias
uv sync

# Instalar PyTorch con soporte CUDA (opcional, solo si se dispone de GPU)
uv pip install --force-reinstall \
  --index-url https://download.pytorch.org/whl/cu126 \
  torch torchvision torchaudio
```

Reproducción del experimento principal

Listing A.2: Ejecución del pipeline completo

```
uv run python -m src.run_probabilistic_lstm
```

El resultado se escribe en `reportes/probabilistic_lstm/Resumen_Probabilistico.md`.

Semillas y determinismo

Se fija `SEED=42` en todos los puntos no-deterministas (NumPy, PyTorch, Optuna, CatBoost). El entrenamiento en GPU introduce un no-determinismo residual por las operaciones atómicas de CUDA. Eliminarlo del todo penalizaría demasiado el rendimiento, así que se asume.

Disponibilidad del dataset

El conjunto de datos contiene información comercial confidencial de un único cliente, por lo que no se publica. Para verificar la integridad de la versión utilizada en los experimentos, se incluye la huella SHA-256 del fichero `data/BD_MAESTRA_PRECIOS.csv`:

`SHA-256: 4ae05751979d8325eab8c253a1c1c8c5d2f71c16bb893881e126401edd43839d`