

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



“PREDICCIÓN DE VALORES DE INTERCAMBIO DE AUTOMÓVILES MEDIANTE ESTRATEGIAS DE TRANSFER LEARNING”

DIEGO ALEJANDRO QUEZADA CAMPOS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Carlos Valle Vidal
Profesor Correferente: Ricardo Ñanculef

Octubre - 2023

DEDICATORIA

A mis padres, Jeannette y Raúl: Por brindarme todo lo necesario durante más de dos décadas para que hoy pueda estar aquí.

Agradecimientos

La realización de esta investigación ha sido un camino lleno de desafíos y aprendizajes. Quiero reconocer y agradecer a todos los que me acompañaron en este proceso. En primer lugar, agradezco nuevamente a mi familia por su apoyo incondicional, especialmente a mis padres, Jeannette y Raúl, por brindarme todo lo necesario durante más de dos décadas para que hoy pueda estar aquí. Agradezco a Scarlett, mi compañera de vida, por su apoyo y comprensión durante este proceso. Agradezco a Benjamín Sánchez, Braulio Fuentes, Diego Barboza, Jesús Oyanedel, Diego Cattarinich, Bruno Prieto, Felipe Cisternas, y a todos mis amigos de la universidad por los momentos compartidos y por su apoyo en los momentos en que el término de esta investigación parecía lejano. Agradezco a Germán Treimún por aconsejarme y estimular mi interés en tópicos avanzados desde mi segundo año de universidad. Agradezco a Cristián Serpell por enseñarme en cada conversación. Agradezco a Carlos Valle por introducirme en el mundo del aprendizaje de máquinas y por su apoyo durante el desarrollo de esta investigación. Agradezco al Departamento de Informática de la Universidad Técnica Federico Santa María y sus profesores por formarme como Ingeniero Civil Informático. Finalmente, agradezco a Dimitrios Lazarou y Andrew Kabrit por su confianza y por brindarme todo lo necesario para desarrollar esta investigación.

Resumen

Resumen— Predecir el valor de *trade-in* de un automóvil es una tarea compleja, más aún cuando no se puede analizar en detalle por un profesional pues se busca una evaluación rápida, pero también, robusta. Desarrollar una solución basada en aprendizaje de máquinas es axiomático, sin embargo, lidiar con atributos de entrada de alta dimensionalidad e insuficientes datos para entrenar modelos que aprendan directamente a predecir valores de *trade-in* hacen de esta tarea un gran desafío. En esta memoria, se propone un modelo que predice valores de *trade-in* mediante la transferencia del conocimiento adquirido para predecir precios de lista.

Palabras Clave— Aprendizaje de Máquinas, Aprendizaje Profundo, Transferencia de Conocimiento, Aprendizaje Ensamblado

Abstract

Abstract— Predicting the *trade-in* value of an automobile is a complex problem, even more so when it cannot be analyzed in detail by a professional as a quick, but also robust, evaluation is sought. Developing a machine learning based solution is axiomatic, however, dealing with high dimensionality input attributes and lack of data to train models that directly learn to predict *trade-in* values makes this task a great challenge. In this paper, we propose a model that predicts *trade-in* values by transferring the knowledge acquired to predict list prices.

Keywords— Machine Learning, Deep Learning, Transfer Learning, Ensemble Learning

Glosario

DI: Departamento de Informática.

IA: Inteligencia Artificial.

ML: Machine Learning.

TSPO: Two-stage prediction-oriented.

TSPOS: Two-stage prediction-oriented Stacking.

TSRO: Two-stage representation-oriented.

TSROS: Two-stage representation-oriented Stacking.

UTFSM: Universidad Técnica Federico Santa María.

XAI: Explainable Artificial Intelligence.

$\mathcal{S}$: Conjunto de datos.

$\mathcal{X}$: Espacio de atributos.

$\mathcal{Y}$: Espacio de salida.

$\mathcal{H}$: Espacio de hipótesis.

ℓ : Función de pérdida.

J : Función de costo.

I : Número de atributos.

M : Número de datos de entrenamiento.

N : Número de modelos base en un ensamblado.

FNN: *Feedforward Neural Network*.

Índice de contenidos

Resumen	IV
Abstract	IV
Glosario	V
Índice de figuras	VIII
Índice de tablas	VIII
Introducción	1
Capítulo 1: Definición del problema	2
1.1 Contexto	2
1.2 Situación actual	3
1.3 Problemáticas	3
1.4 Objetivos	4
1.4.1 Objetivo general	4
1.4.2 Objetivos específicos	5
1.5 Alcance	5
Capítulo 2: Marco conceptual	6
2.1 Ciencia de datos	6
2.2 Sistemas de Aprendizaje de Máquinas	7
2.3 Aprendizaje de máquinas	8
2.4 Aprendizaje supervisado	9
2.4.1 Definiciones	9
2.4.2 Regresión lineal	10
2.4.3 Descenso del gradiente	11
2.4.4 Árboles de decisión	11
2.5 Aprendizaje por ensambles	12
2.6 CatBoost	15
2.7 Redes neuronales de propagación hacia adelante	17
2.8 Transferencia de Aprendizaje	18
2.9 Generalización apilada	19
2.10 Inteligencia artificial explicable	19
2.10.1 LIME	20
2.10.2 SHAP	20
2.11 Predicción de precios de automóviles	21
Capítulo 3: Propuesta de solución	25
3.1 Metodología	25
3.2 Fuentes de datos	26

3.3	Análisis exploratorio de datos	27
3.4	Preprocesamiento	33
3.5	Modelamiento	38
3.5.1	2SPO	40
3.5.2	2SRO	41
3.5.3	3SPO	42
3.5.4	3SRO	43
3.6	Entrenamiento	44
3.6.1	Modelo primera etapa	44
3.6.2	Modelo segunda etapa	46
3.6.3	Modelo tercera etapa	47
Capítulo 4: Validación de la solución		49
4.1	Indicadores clave de desempeño	49
4.2	Conjunto de prueba	49
4.3	Proceso de evaluación	50
4.4	Validación hipótesis central	51
4.5	Explicaciones con XAI	52
Capítulo 5: Conclusiones		57

Índice de figuras

1	Proceso para obtener un valor de <i>trade-in</i>	3
2	Árbol del problema	5
3	Fases del proceso CRISP-DM	6
4	Componentes de un sistema de ML	8
5	Metodología de la investigación	25
6	Ooyyo: Distribución de atributos numéricos	30
7	Ooyyo: Distribución de atributos categóricos	31
8	Ooyyo: Distribución del precio	32
9	SeezPAD: Distribución de atributos numéricos	32
10	SeezPAD: Distribución de atributos categóricos	33
11	Listing: Proceso de construcción	34
12	Trade-in: Proceso de construcción	35
13	Listing: Matriz de correlación lineal	37
14	Trade-in: Matriz de correlación lineal	38
15	<i>Modelamiento 2SPO</i>	41
16	<i>Modelamiento 2SRO</i>	42
17	<i>Modelamiento 3SPO</i>	43
18	<i>Modelamiento 3SRO</i>	44
19	Primera etapa: Evolución de la pérdida por época	46
20	Segunda etapa: Evolución de la pérdida por época	47
21	Distribución del precio en el conjunto de prueba de <i>Trade-in</i>	50
22	Correlación predicción de precios de lista y valores de <i>trade-in</i>	52
23	Explicación: Peugeot 208	54
24	Explicación: Mitsubishi Outlander	54
25	Explicación: Hyundai i20	55
26	Valores <i>SHAP</i>	55
27	Proceso de decisión explicado con valores <i>SHAP</i>	56

Índice de tablas

1	Resumen de fuentes de datos	27
2	Ooyyo: Ejemplos de datos sin preprocesar	27
3	SeezPAD: Ejemplos de datos sin preprocesar	27
4	Atributos de la fuente de datos Ooyyo	28
5	Atributos de la fuente de datos SeezPAD	29
6	Atributos obtenidos desde la API de <i>Bilstatistik</i>	29
7	Atributos obtenidos desde el sitio web <i>Nummerplade</i>	30
8	Conjuntos de datos	36
9	Técnicas de modelamiento	39
10	Hiperparámetros primera etapa	44
11	Arquitecturas primera etapa	45

12	Hiperparámetros segunda etapa	46
13	Hiperparámetros tercera etapa	48
14	Resultados validación cruzada 3SPO	48
15	Resultados validación cruzada 3SRO	48
16	Modelo base: Métricas de evaluación	50
17	Modelo de ajuste: Métricas de evaluación	51
18	Instancias de prueba	52
19	Coeficientes modelo de <i>Stacking</i>	53

Introducción

Con el aumento continuo del número de automóviles particulares y el crecimiento del mercado de automóviles de segunda mano la necesidad de modelos matemáticos para predecir el precio de automóviles es cada vez más importante para empresas del rubro automovilístico. El precio de un automóvil de segunda mano es un aspecto clave para el éxito de una transacción, tanto para compradores como para vendedores. Para los compradores, conocer el precio de los automóviles usados permite negociar con tranquilidad; para los vendedores, evaluar el valor residual de los automóviles usados puede ayudarles a fijar precios razonables.

Como consecuencia, el *trade-in*, i.e. intercambio de automóviles, se ha posicionado como una modalidad popular para adquirir un nuevo automóvil. Esta alternativa permite ofrecer un automóvil como parte del medio de pago para adquirir uno nuevo. En este contexto, toma relevancia el **valor de *trade-in***: el precio que se le asigna al automóvil que se ofrece como parte del medio de pago de **manera acordada** entre un concesionario y un cliente. En Seez, una empresa que vende automóviles de manera 100 % *online* en Dinamarca y que está en transición a ser un proveedor de servicios *SaaS*, el proceso que determina el valor de *trade-in* se realiza de forma manual, lo que requiere una gran dedicación del equipo de ventas para estudiar el mercado y negociar con los concesionarios.

El uso de **aprendizaje de máquinas** para predecir precios es una aplicación de larga data. A pesar de su simple formulación como un problema de regresión, lidiar con la incertidumbre intrínseca del mercado es un desafío colosal que implica definir minuciosamente los atributos de entrada y mantener los modelos actualizados con datos vigentes. En esta investigación, se propone un modelo de aprendizaje de máquinas para predecir valores de *trade-in* utilizando técnicas de **transferencia de aprendizaje**.

La presente memoria pertenece al área de **Inteligencia Computacional** del Departamento de Informática de la Universidad Técnica Federico Santa María y consta de 5 capítulos. En el capítulo 1 se define exhaustivamente el problema. En el capítulo 2 se discute el marco teórico. En el capítulo 3 se propone la solución. En el capítulo 4 se valida la solución. Finalmente, en el capítulo 5 se plantean las conclusiones de la presente memoria considerando los alcances y limitaciones de la solución planteada.

Capítulo 1

Definición del problema

En esta sección se definen las componentes fundamentales del problema abordado en la presente investigación. Se comienza explicando el contexto del problema, el cual queda en gran parte definido por la empresa Seez y su servicio de *trade-in* que permite a los usuarios ofrecer sus automóviles actuales como parte de pago para adquirir uno nuevo. Luego, se describe el funcionamiento actual del proceso que permite obtener un valor de *trade-in*, para posteriormente, identificar las principales dificultades en él. Finalmente, se definen los objetivos y alcance de la presente investigación.

1.1. Contexto

Seez es una empresa de Dubai que está revolucionando la industria automotriz cambiando la forma en que las personas buscan y compran automóviles de manera 100 % *online*. Actualmente, su *Marketplace* funciona en Dinamarca y fue la primera plataforma *online* para adquirir automóviles en tal país. Es una empresa joven y en rápido crecimiento, actualmente tiene más de 5.000 usuarios por día y ha digitalizado a los dos más grandes concesionarios automotrices de Dinamarca y más de 40 en total, vendiendo y entregando más de 100 automóviles hasta la fecha. A pesar de esto, su modelo de negocios ha evolucionado con el tiempo y la empresa está en transición a ofrecer servicios como **Software-as-a-Service** (SaaS) para la industria automotriz enfocándose en su producto SeezPAD (Platform Admin Dashboard): un centro de control desde donde un vendedor puede gestionar, editar y cerrar las órdenes de sus clientes. SeezPAD fue diseñado para apoyar a los concesionarios automotrices en su transición a un enfoque digital, es una extensión digital de la oficina trasera de cada sala de exhibición y un facilitador para cada vendedor.

Mediante su modalidad de *trade-in*, i.e., intercambio, Seez permite a los usuarios ofrecer sus automóviles actuales como parte de pago para adquirir uno nuevo. Este servicio permite a los usuarios actualizar sus automóviles sin la necesidad de gestionar ni la venta ni la entrega de los mismos, pues Seez también se encarga del retiro del automóvil. El proceso actual para obtener un valor de *trade-in* es **manual** y es realizado por el equipo de ventas de Seez, es decir, para cada nueva solicitud de *trade-in* se invierte considerable tiempo en estudiar el mercado y en negociar con los diferentes concesionarios automotrices.

El servicio de *trade-in* es crucial para Seez; el 72 % de las ventas totales han sido completadas utilizando este servicio. De esta forma, es de esperarse el gran interés por desarrollar una solución basada en aprendizaje de máquinas que permita predecir el valor de *trade-in* para un automóvil y así acelerar el estudio de mercado llevado a cabo por el equipo de ventas y, por ende, comunicar la oferta a los usuarios más rápidamente. Además, en el contexto SaaS, la predicción de valores de *trade-in* destaca como una de las características más deseables

de *SeezPAD* para los concesionarios y vendedores de automóviles.

1.2. Situación actual

Tal como se mencionó en la sección Introducción, el proceso actual para generar un valor de *trade-in* es manual. La figura 1 detalla el proceso como un diagrama de flujo:

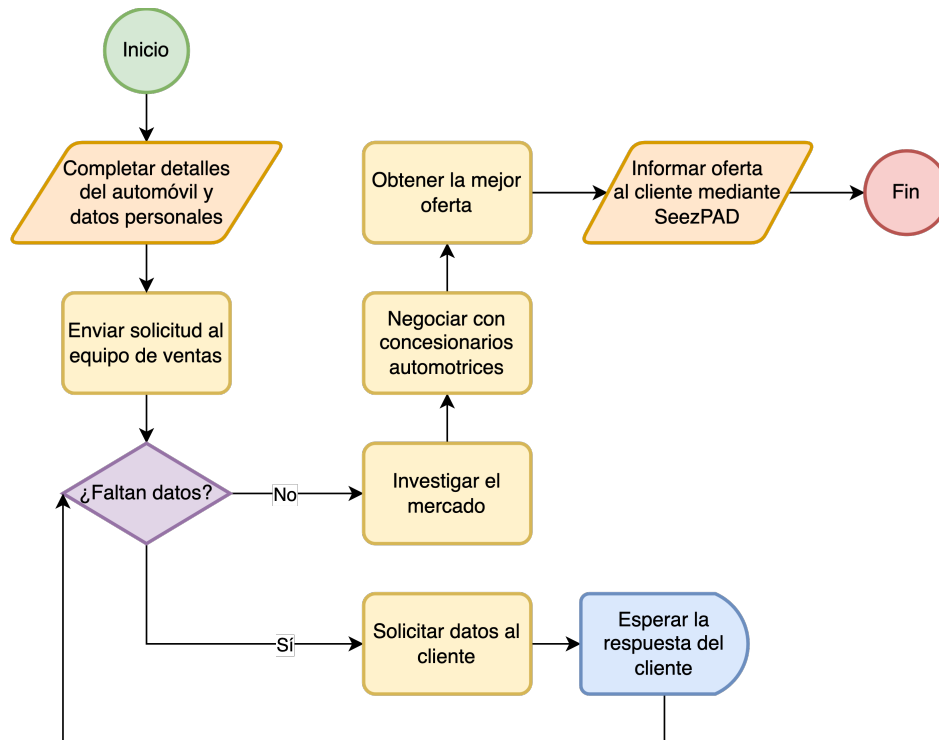


Figura 1: Proceso para obtener un valor de *trade-in*
Fuente: Elaboración propia

Si bien el diagrama de flujo no representa visualmente un proceso extenso, es importante considerar que las etapas *Investigar el mercado* y *Negociar con concesionarios automotrices* requieren una cantidad significativa de tiempo.

1.3. Problemáticas

El tiempo invertido en estudiar el mercado para cada solicitud de *trade-in* es elevado e indeterminado, como consecuencia el tiempo de espera para que un cliente obtenga una oferta a su solicitud de *trade-in* es incierto. Además, el proceso actual depende de terceros ya que el automóvil intercambiado es finalmente comprado por un concesionario automotriz y se

debe negociar para obtener el mejor precio. Finalmente, este proceso no escala bien considerando un tamaño fijo para el equipo de ventas. Así, quedan en evidencia las **principales dificultades del flujo actual** para obtener un valor de *trade-in*.

Diseñar un modelo de aprendizaje de máquinas supervisado utilizando un conjunto de datos etiquetado parece ser un enfoque evidente y prometedor para predecir valoraciones de *trade-in*, sin embargo, en el contexto actual de Seez esta solución no está exenta de desafíos. El volumen de datos disponible de valores de *trade-in* no es suficiente para entrenar directamente un modelo de aprendizaje de máquinas para predecir estos valores de manera precisa considerando las distintas marcas y modelos de automóviles presentes en el mercado de Dinamarca. De hecho, actualmente este conjunto de datos no supera los 500 datos. Por otro lado, se encuentra disponible una variedad de conjuntos de datos de gran volumen y de distintos sitios web de Dinamarca con **precios de lista de automóviles**, i.e., precios fijados por los dueños para la venta de sus automóviles. Si bien el precio de lista y el valor de *trade-in* siguen distribuciones distintas, el precio de lista aporta una valiosa información a la hora de predecir valores de *trade-in*.

Los datos de marcas y modelos de automóviles tienen una **gran dimensionalidad** y son **jerárquicos**. En el mercado automotriz de Dinamarca conviven más de 40 marcas de automóviles, en donde cada una tiene 25 modelos en promedio y cada modelo tiene por lo general 3 o más configuraciones (el mismo modelo con diferente transmisión, potencia del motor o tipo de combustible por ejemplo). Así, el número de atributos categóricos necesarios para modelar cada uno de los vehículos son al menos 3000. Esta simple observación tiene una importante consecuencia en el contexto de aprendizaje de máquinas: a medida que consideramos más marcas, modelos y configuraciones de automóviles, el **espacio de atributos** se vuelve cada vez más disperso y, por ende, el número de datos necesarios para obtener un modelo con buen rendimiento aumenta exponencialmente. Este clásico problema es conocido como la **maldición de la dimensionalidad** y ha sido ampliamente estudiado en la literatura.

En el siguiente árbol se sintetiza la definición inicial del problema mediante la identificación del problema central, sus causas y sus efectos:

1.4. Objetivos

1.4.1. Objetivo general

Diseñar y evaluar modelos de aprendizaje de máquinas para la predicción de valores de *trade-in* mediante técnicas de transferencia de aprendizaje.

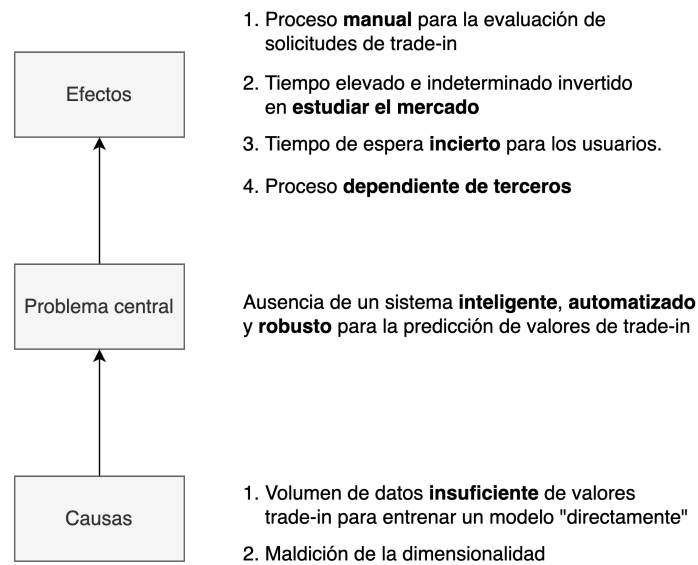


Figura 2: Árbol del problema
Fuente: Elaboración propia

1.4.2. Objetivos específicos

1. Proponer y evaluar diferentes técnicas de modelamiento para la predicción de valores de *trade-in* según indicadores clave del negocio.
2. Proveer interpretabilidad a los modelos de aprendizaje con *XAI* para entender sus decisiones y determinar la confiabilidad de sus predicciones.
3. Garantizar la reproducibilidad de los experimentos mediante una documentación adecuada y un código legible y modular.

1.5. Alcance

El alcance de la presente memoria abarca desde el entendimiento del negocio de Seez hasta el despliegue de los diferentes modelos de aprendizaje de máquinas, pasando por las etapas de entendimiento de los datos, preparación de los datos, modelamiento y evaluación. El resultado esperado de la presente investigación es un conjunto de modelos bien documentado que predicen valores de *trade-in* de manera precisa utilizando técnicas de transferencia de aprendizaje para aprovechar los datos disponibles. Además, se enfatiza la importancia de desplegar al menos uno de estos modelos, poniéndolo en marcha y haciéndolo accesible para los usuarios, lo que permitirá la aplicación práctica de los resultados obtenidos y proporcionará un entorno en el que se puedan validar y utilizar las predicciones generadas en situaciones del mundo real.

Capítulo 2

Marco conceptual

En esta sección se plantea el marco conceptual de la presente investigación definiendo exhaustivamente los conceptos fundamentales en los cuales se sostiene la solución planteada en el capítulo 3.

2.1. Ciencia de datos

El objetivo de la ciencia de datos es mejorar la toma de decisiones basándose en conocimientos extraídos de grandes conjuntos de datos. Como un campo multidisciplinario, engloba un conjunto de principios, definiciones de problemas, algoritmos y procesos para extraer patrones no obvios y útiles a partir de grandes conjuntos de datos [Kelleher y Tierney, 2018].

CRISP-DM [Wirth y Hipp, 2000] es un método estándar, iterativo y flexible para orientar proyectos de minería de datos. La principal ventaja de CRISP-DM es que fue diseñado de manera independiente a cualquier *software*, empresa o técnica de análisis de datos. El modelo genérico contempla 6 fases:

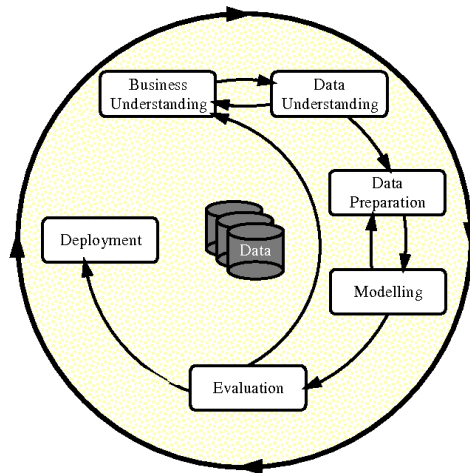


Figura 3: Fases del proceso CRISP-DM
Fuente: [Wirth y Hipp, 2000]

A continuación, se describe cada fase brevemente:

- Entendimiento del negocio: La fase inicial se enfoca en comprender los objetivos y requerimientos del proyecto desde un punto de vista del negocio para luego definirlos como un problema de minería de datos.

- Entendimiento de los datos: La segunda fase se enfoca en recolectar, describir, explorar y analizar los datos de distintas fuentes con el objetivo de descubrir los primeros patrones en ellos e identificar problemas en la calidad de los datos.
- Preparación de los datos: La tercera fase se enfoca en construir los conjuntos de datos finales a partir de los conjuntos de datos sin procesar recolectados en la fase anterior.
- Modelamiento: La cuarta fase se enfoca en diseñar, seleccionar y aplicar distintas técnicas de modelamiento.
- Evaluación: La quinta fase se enfoca en evaluar minuciosamente la calidad de los modelos, revisar el proceso y determinar los siguientes pasos.
- Despliegue: La última fase se enfoca en implementar el modelo en un entorno de producción, establecer un plan para el monitoreo y mantenimiento del proceso, y elaborar una documentación.

Una alternativa moderna a CRISP-DM es *Team Data Science Process* [Microsoft, 2021], sin embargo, no se ajusta al contexto del presente proyecto.

2.2. Sistemas de Aprendizaje de Máquinas

En un reciente libro [Huyen, 2022], se propone una perspectiva integral y sistémica en relación al ML, que va más allá del enfoque puramente algorítmico y matemático. La figura 4 ilustra un diagrama detallado que muestra los componentes clave de un sistema de ML, así como los elementos externos que interactúan con dicho sistema. Entre estos componentes externos, se destacan los requerimientos del negocio que generan la necesidad y fundamentan el desarrollo de proyectos basados en ML, así como los usuarios finales que interactúan con el sistema a través de diversas interfaces diseñadas específicamente para facilitar la comunicación y el intercambio de información.

La importancia de esta visión sistémica en un sistema de ML radica en la necesidad de considerar de manera holística todos los componentes esenciales para el éxito de un proyecto de ML, incluyendo infraestructura, datos, ingeniería de atributos, algoritmos de ML, evaluación, despliegue, monitoreo y actualización de lógica. Adoptar este enfoque integral permite abordar de manera eficiente los desafíos y problemáticas asociadas a cada componente, y reconocer las interdependencias que existen entre ellos.

Una visión sistémica facilita la identificación de cuellos de botella, ineficiencias y posibles mejoras en cada etapa del proceso de desarrollo e implementación del sistema de ML. Además, permite un mayor entendimiento de cómo las decisiones tomadas en un área pueden afectar el rendimiento y la funcionalidad del sistema en su totalidad. Por ejemplo, la calidad y diversidad de los datos pueden tener un impacto significativo en la efectividad del algoritmo

de ML seleccionado, y a su vez, la evaluación y despliegue adecuados del sistema pueden asegurar que el mismo cumpla con los requerimientos del negocio y las expectativas de los usuarios.

En última instancia, adoptar una visión sistémica en un sistema de ML puede mejorar significativamente la calidad y eficacia del proyecto, al garantizar que todos los componentes y procesos involucrados estén alineados y funcionen de manera eficiente y coherente para cumplir con los objetivos establecidos. Esta perspectiva holística también promueve una mayor transparencia y responsabilidad, ya que permite a los interesados comprender mejor cómo cada componente contribuye al éxito del proyecto y asegurar que se aborden las preocupaciones éticas y legales asociadas al ML.

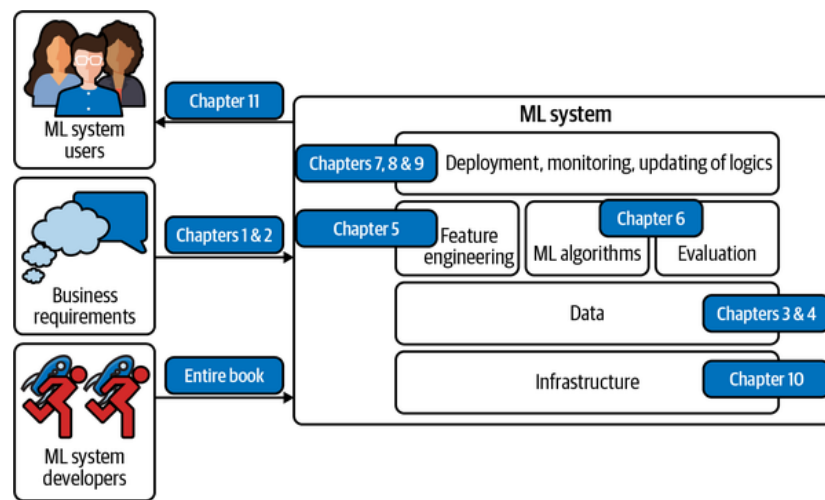


Figura 4: Componentes de un sistema de ML
Fuente: [Huyen, 2022]

2.3. Aprendizaje de máquinas

El aprendizaje de máquinas se define como un **conjunto de métodos** que pueden automáticamente detectar **patrones en datos** y luego utilizar esos patrones descubiertos para predecir datos futuros o para realizar otro tipo de toma de decisiones bajo cierta incertidumbre. En este contexto, el **aprendizaje** es clave y en [Mitchell, 1997] se provee una específica definición: "Se dice que un programa de computador aprende de una experiencia E con respecto a una clase de tareas T y medida de desempeño P , si su desempeño en las tareas T , medido mediante P , mejora con la experiencia E ". Extendiendo esta definición, el "programa de computador" es un modelo M que aprende mediante un **algoritmo de aprendizaje** $\mathcal{A}$.

La tarea T define **qué** aprenderá a hacer M . Por ejemplo, podríamos estar interesados en desarrollar un modelo que clasifique un *tweet* como ofensivo o no ofensivo, o bien, en un modelo que estime el número de reproducciones que tendrá un video de *Youtube* en el día

que es publicado. "La medida de desempeño P indica **cuán** hábil es M para realizar T . Siguiendo el ejemplo anterior para el modelo que clasifica un *tweet*, podríamos definir P como el porcentaje de *tweets* clasificados correctamente, de esta forma si el porcentaje es cercano al 100 % diríamos que M es un modelo con buen desempeño. La medida de desempeño P depende de la tarea T , sin embargo, para una misma tarea T se podrían utilizar distintas medidas de desempeño.

La experiencia E define **cómo** M aprenderá a realizar T . De manera general, E se puede clasificar como aprendizaje supervisado o aprendizaje no supervisado.

2.4. Aprendizaje supervisado

2.4.1. Definiciones

El aprendizaje supervisado es el tipo de aprendizaje más usado en la práctica. En este marco, un algoritmo aprende una **hipótesis** $f: \mathcal{X} \rightarrow \mathcal{Y}$ utilizando un conjunto $\mathcal{S} = \{(\mathbf{x}_m, y_m)\}_{m=1}^M$ de datos entrenamiento, en donde $\mathcal{X}$ es el **espacio de atributos** e $\mathcal{Y}$ el **espacio de salida**. Cada instancia del conjunto de entrenamiento $(\mathbf{x}_m, y_m)$ está compuesta por un vector de atributos de entrada $\mathbf{x}_m$ y una salida y_m . Generalmente, las entradas $\mathbf{x}_m$ corresponden a un vector numérico y las salidas y_m son categóricas para un **problema de clasificación**, o bien, reales para un **problema de regresión**.

El **espacio de hipótesis** $\mathcal{H}$ es el conjunto de posibles hipótesis que un algoritmo de aprendizaje $\mathcal{A}$ podría obtener a partir de un conjunto de entrenamiento. Considerando una familia de hipótesis $f(\mathbf{x}; \mathbf{w})$ parametrizadas por un vector de parámetros $\mathbf{w} \in \mathcal{W}$, el espacio de hipótesis se puede definir como $\mathcal{H} = \{f(\mathbf{x}; \mathbf{w}); \mathbf{w} \in \mathcal{W}\}$.

El principal desafío del aprendizaje supervisado es aprender una hipótesis $y = f(\mathbf{x}; \mathbf{w})$ a partir del conjunto de entrenamiento $\mathcal{S}$ de tal forma que f tenga la capacidad de predecir correctamente entradas $\mathbf{x}_i$ futuras, es decir, que no han sido vistas antes. Esta propiedad clave se llama **generalización**.

La medida de desempeño P en el contexto de aprendizaje supervisado recibe el nombre de **función de pérdida**. De manera general, se puede definir como una función $\ell: \mathbb{R} \times \mathcal{Y} \rightarrow \mathbb{R}$ que asigna una penalización a una predicción $f(\mathbf{x}; \mathbf{w})$ en función de cuánto difiere respecto a la verdadera salida y . Intuitivamente, si $f(\mathbf{x}; \mathbf{w}) = y$ entonces $\ell(y, f(\mathbf{x}; \mathbf{w})) = 0$ pues la predicción fue exacta. Al aplicar ℓ a un conjunto de entrenamiento obtenemos la **función de costo** $J(\mathbf{w})$:

$$J(\mathbf{w}) = \sum_{m=1}^M \ell(y_m, f(\mathbf{x}_m; \mathbf{w})). \quad (1)$$

2.4.2. Regresión lineal

Nuestra definición actual de un algoritmo de aprendizaje como un algoritmo que es capaz de mejorar el rendimiento de un programa de computador en una tarea T a través de una experiencia E es un tanto abstracta. Para materializar estos conceptos se presenta un algoritmo de aprendizaje fundamental: la **regresión lineal**. Tal como su nombre lo sugiere, este algoritmo resuelve un problema de regresión, i.e. $y \in \mathbb{R}$, y su salida es una función lineal de la entrada $\mathbf{x} \in \mathbb{R}^I$:

$$f(\mathbf{x}; \mathbf{w}) = b + \sum_{i=1}^I w_i x_i. \quad (2)$$

Alternativamente, $f(\mathbf{x}; \mathbf{w})$ se puede escribir como $f(\mathbf{x}; \mathbf{w}) = \mathbf{w}^\top \mathbf{x}$ considerando $\mathbf{w}$ y $\mathbf{x}$ como vectores columnas y $x_0 = 1$ para considerar el parámetro b en el primer elemento de $\mathbf{w}$.

Los **parámetros** w_i y b son los valores que definen a un modelo de regresión lineal, es decir, los que instancian una hipótesis desde $\mathcal{H}$. Notar que un modelo de regresión lineal siempre tiene $n + 1$ parámetros, donde n es el número de atributos de la entrada. Intuitivamente, estos parámetros se deben aprender desde un conjunto de entrenamiento. La función de pérdida estándar para este algoritmo es la pérdida cuadrática $\ell(y, f(\mathbf{x}; \mathbf{w})) = (y - f(\mathbf{x}; \mathbf{w}))^2$ y en el entrenamiento se minimiza el **error cuadrático medio** sobre S :

$$J(\mathbf{w}) = \frac{1}{M} \sum_{m=1}^M (y_m - f(\mathbf{x}_m; \mathbf{w}))^2 = \frac{1}{m} \sum_{m=1}^M (y_m - \mathbf{w}^\top \mathbf{x}_m)^2. \quad (3)$$

Para minimizar J basta resolver $\nabla_{\mathbf{w}} J = 0$. La siguiente demostración se basa en [Goodfellow et al., 2016] y descompone S en una matriz $\mathbf{X}$ cuyas filas son las entradas $\mathbf{x}_m$ y un vector $\mathbf{y}$ cuyos elementos son las salidas y_m .

$$\begin{aligned} \nabla_{\mathbf{w}} J &= 0 \\ \nabla_{\mathbf{w}} \frac{1}{m} \|\hat{\mathbf{y}} - \mathbf{y}\|_2^2 &= 0 \\ \frac{1}{m} \nabla_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 &= 0 \\ \nabla_{\mathbf{w}} (\mathbf{X}\mathbf{w} - \mathbf{y})^\top (\mathbf{X}\mathbf{w} - \mathbf{y}) &= 0 \\ \nabla_{\mathbf{w}} \mathbf{w}^\top \mathbf{X}^\top \mathbf{X} \mathbf{w} - 2\mathbf{w}^\top \mathbf{X}^\top \mathbf{y} + \mathbf{y}\mathbf{y}^\top &= 0 \\ 2\mathbf{X}^\top \mathbf{X} \mathbf{w} - 2\mathbf{X}^\top \mathbf{y} &= 0 \\ (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} &= \mathbf{w}. \end{aligned} \quad (4)$$

El sistema de ecuaciones cuya solución viene dada por 4 es conocido como **ecuaciones normales**.

De esta forma, solo basta computar 4 para instanciar el modelo de regresión lineal óptimo para $\mathbf{X}$. Por supuesto, esta computación no es trivial y además no todos los algoritmos de aprendizaje tienen una expresión analítica para sus ecuaciones normales. Es necesario un algoritmo más general para solucionar $\nabla_w J = 0$.

2.4.3. Descenso del gradiente

El gradiente descendiente es un algoritmo de optimización que utiliza la información de la derivada para obtener un óptimo. De manera general, podemos minimizar una función $f(x)$ moviendo x en pequeños pasos en la dirección contraria al signo de $f'(x)$. En el contexto de aprendizaje supervisado, cada paso se calcula considerando el gradiente $\nabla_w J$ y una tasa de aprendizaje α que controla el “tamaño de cada paso”.

Algorithm 1 Gradiente descendente

Require: epochs

```
 $i \leftarrow 1$   
while  $i \leq \text{epochs}$  do  
     $\mathbf{w}_i \leftarrow \mathbf{w}_{i-1} - \alpha \nabla_{\mathbf{w}_{i-1}} J$   
     $i \leftarrow i + 1$   
end while
```

Es importante notar que cada actualización de $\mathbf{w}$ es proporcional al error cometido en cada iteración. De esta forma, a medida que $\mathbf{w}$ converge al óptimo $\mathbf{w}^*$ los elementos de $\mathbf{w}$ cambiarán cada vez menos.

Por lo general este algoritmo obtiene mínimos locales, sin embargo, si la función J es convexa solo existe un óptimo global. Además, el algoritmo 1 es la versión tradicional de gradiente descendente.

2.4.4. Árboles de decisión

Los Árboles de Clasificación y Regresión, conocidos por sus siglas en inglés como CART o simplemente como árboles de decisión, pertenecen a una familia de modelos no paramétricos. Estos se caracterizan por estructurarse a través de un conjunto de reglas de decisión anidadas, facilitando así la interpretación y análisis de los datos. La operatividad de estos árboles se define de manera recursiva, dividiendo el espacio de atributos, denotado como $\mathcal{X}$, en distintas regiones y estableciendo un modelo local en cada una de ellas.

En el proceso de construcción del árbol, cada nodo j juega un papel crucial, en este, un atributo específico d_j se compara con un valor umbral u_j . Dependiendo del resultado de esta comparación, la instancia se dirige hacia la izquierda o hacia la derecha, es decir, si d_j es menor o mayor que u_j , respectivamente. Al alcanzar las hojas del árbol, se asigna una salida única para todas las entradas que convergen en esa región particular de $\mathcal{X}$, estableciendo así la predicción final.

Formalmente, podemos definir un árbol de regresión como:

$$f(\mathbf{x}; \boldsymbol{\theta}) = \sum_{t=1}^T p_t \mathbb{1}(\mathbf{x} \in R_t). \quad (5)$$

Donde R_t representa la región determinada por el nodo terminal t y p_t su predicción correspondiente. Definimos el conjunto de parámetros $\boldsymbol{\theta}$ como $\boldsymbol{\theta} = \{(R_t, p_t) : t = 1, \dots, T\}$, siendo T el número total de nodos terminales. Es esencial destacar que la determinación de la región R_t involucra una serie de decisiones acumulativas tomadas a lo largo del camino desde el nodo raíz hasta alcanzar el nodo hoja t . Además, p_t se suele definir como:

$$p_t = \sum_{m=1}^M \frac{y_m \mathbb{1}(\mathbf{x}_m \in R_t)}{\mathbb{1}(\mathbf{x}_m \in R_t)}. \quad (6)$$

Como se ha señalado en secciones anteriores, la formulación de una función de costo es un paso crucial para orientar el proceso de entrenamiento. En el contexto de los árboles de decisión, el objetivo es minimizar la siguiente función de costo:

$$J(\boldsymbol{\theta}) = \sum_{t=1}^T \sum_{(\mathbf{x}, y) \in R_t} (y - p_t)^2. \quad (7)$$

No obstante, esta función presenta una considerable limitación: no es diferenciable. Esto se debe a que el aprendizaje involucra la determinación de una estructura discreta, la del árbol, lo que introduce discontinuidades en la función. Esta particularidad conduce a la adopción de estrategias alternativas en la práctica estándar, recurriendo a algoritmos voraces que construyen el árbol de manera iterativa, añadiendo un nodo en cada paso. Algunos algoritmos clásicos que emplean esta estrategia son CART, C4.5 e ID3.

2.5. Aprendizaje por ensambles

El aprendizaje por ensambles es una estrategia en aprendizaje automático donde, en lugar de utilizar un único modelo $f(\mathbf{x}; \boldsymbol{\theta})$ para predecir la salida y , se emplea un conjunto de

N modelos base de baja precisión $\{f_1, f_2, \dots, f_N\}$. En términos generales, un modelo de ensamble se define como:

$$F(\mathbf{x}; \boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^N f_n(\mathbf{x}; \boldsymbol{\theta}_n). \quad (8)$$

El principio fundamental de esta estrategia es que al combinar múltiples modelos **heterogéneos** de baja precisión, es posible obtener un modelo compuesto de alta precisión. Los modelos base f_n se denominan comúnmente como aprendices débiles o *weak learners*, caracterizados por su incapacidad para aprender patrones complejos, ofreciendo un rendimiento solo marginalmente mejor que una asignación aleatoria. A pesar de esto, estos aprendices débiles son computacionalmente eficientes y fáciles de entrenar. El modelo compuesto F es conocido como el modelo *meta*.

Uno de los conceptos centrales en el aprendizaje por ensambles es la **descomposición de la ambigüedad** [Audhkhasi et al., 2013], o *ambiguity decomposition* en inglés. Considerando un par único $(\mathbf{x}, y)$ y percibiendo $F(\mathbf{x})$ como una variable aleatoria de la cual tomamos una muestra de tamaño N , donde cada instancia f_n tiene una probabilidad w_n de ocurrir, podemos expresar el error cuadrático medio de $F(\mathbf{x})$ como:

$$E[y - F(\mathbf{x})]^2 = E[(y - F(\mathbf{x}))^2] - V[F(\mathbf{x})]. \quad (9)$$

Dado que y es constante y la varianza no se altera con traslaciones, podemos reformular la ecuación anterior como:

$$E[y - F(\mathbf{x})]^2 = E[(y - F(\mathbf{x}))^2] - V[F(\mathbf{x})]. \quad (10)$$

Finalmente, aplicando el teorema conocido como ley del estadístico inconsciente, o *LOTUS* por sus siglas en inglés, logramos expresar $E[(y - F(\mathbf{x}))^2]$ como el valor esperado de una función $g(x) = (y - x)^2$, aplicada a la variable aleatoria $F(\mathbf{x})$:

$$\begin{aligned}
 E[y - F(\mathbf{x})]^2 &= E[g(F(\mathbf{x}))] - V[F(\mathbf{x})] \\
 &= \sum_{n=1}^N w_n g(f_n(\mathbf{x})) - V[F(\mathbf{x})] \\
 &= \sum_{n=1}^N w_n (y - f_n(\mathbf{x}))^2 - V[F(\mathbf{x})] \\
 &= \sum_{n=1}^N w_n (y - f_n(\mathbf{x}))^2 - \sum_{n=1}^N w_n (f_n(\mathbf{x}) - F(\mathbf{x}))^2.
 \end{aligned}
 \tag{11}$$

A partir del segundo término, conocido como término de ambigüedad, observamos que es factible reducir el error cuadrático medio de $F(\mathbf{x})$ incrementando la varianza individual de los aprendices débiles f_n .

Los principales métodos para construir modelos de ensamblado son *Bagging*, *Boosting* y *Stacking*. Brevemente, describiremos cada uno de ellos:

- **Bagging:** Este método involucra generar múltiples subconjuntos del conjunto de datos de entrenamiento utilizando muestreo con reemplazo, y luego entrenar un modelo base en cada uno de estos subconjuntos. Las predicciones finales se obtienen mediante votación para problemas de clasificación o promediando las predicciones para problemas de regresión. Este enfoque ayuda a reducir la varianza y evitar el sobreajuste.
- **Boosting:** En esta técnica, los modelos se entrenan secuencialmente, con cada modelo intentando corregir los errores de sus predecesores. Esto da como resultado un modelo compuesto que es una suma ponderada de modelos débiles, enfocándose dinámicamente en las instancias más difíciles de predecir a medida que el entrenamiento progresa.
- **Stacking:** Aquí, varios modelos base son entrenados inicialmente y luego se utiliza un modelo meta para combinar sus predicciones y generar predicciones finales. Este modelo meta aprende cómo asignar pesos óptimos a las predicciones de los modelos base para obtener predicciones finales más precisas.

Cada una de estas estrategias brinda una manera distinta de conjuntar modelos más simples para formar uno más preciso y robusto. En los capítulos subsiguientes, desglosaremos con más detalle las particularidades y beneficios de algunos de estos métodos.

2.6. CatBoost

CatBoost [Prokhorenkova et al., 2019], una librería desarrollada por Yandex, define el estado del arte de *gradient boosting*. Sus autores proponen dos principales contribuciones: una mejora al algoritmo **estadísticos de salida**, *target statistics (TS)* en inglés, y una mejora al algoritmo *gradient boosting*. Ambas técnicas utilizan una permutación para evitar un tipo especial de *target leakage*, identificado por los autores como *shift prediction*, presente en todas las técnicas de *gradient boosting* e implementaciones de TS.

Target statistic Un estadístico de salida es una técnica para codificar un atributo categórico $x_m^{(i)}$ mediante un atributo numérico $\hat{x}_m^{(i)}$ que representa una estadística de la salida y asociada a las instancias del conjunto de entrenamiento $\mathcal{S}$ con valor $x_m^{(i)}$ en el atributo i . Una elección común es simplemente utilizar el valor esperado de la salida: $\hat{x}_m^{(i)} = \mathbb{E}[y | x^{(i)} = x_m^{(i)}]$. Este estimador es ruidoso para categorías de baja frecuencia [Prokhorenkova et al., 2019], normalmente se suaviza utilizando un apriori p :

$$\hat{x}_m^{(i)} = \frac{\sum_{j=1}^M \mathbb{1}(x_m^{(i)} = x_j^{(i)} \cdot y_j) + \alpha p}{\sum_{j=1}^M \mathbb{1}(x_m^{(i)} = x_j^{(i)}) + \alpha}. \quad (12)$$

Considerando un parámetro $\alpha > 0$. Por simple inspección, es evidente la **fuga en la salida** al computar $\hat{x}_m^{(i)}$ utilizando y_m . Una solución simple sería no utilizar y_m cuando se está calculando $\hat{x}_m^{(i)}$, es decir, computar el estadístico de prueba utilizando $\mathcal{S} \setminus x_m$, sin embargo, sorprendentemente esta medida no previene la fuga [Prokhorenkova et al., 2019].

CatBoost plantea una estrategia más efectiva, denominada *Ordered TS*, para calcular un estadístico de salida sin cometer una fuga en la salida al tomar inspiración del aprendizaje en línea introduciendo un tiempo artificial mediante una permutación σ de los datos de entrenamiento $\mathcal{S}$. De esta forma, se utilizan todos los datos históricos $\mathcal{S}_m = \{x_j : \sigma(j) < \sigma(m)\}$ para computar el estadístico de prueba para una instancia de entrenamiento x_m y todos los datos $\mathcal{S}$ para una instancia de prueba.

Gradient boosting La formulación estándar de *gradient boosting* construye iterativamente una secuencia de aproximaciones F_n de manera voraz. F_n se obtiene a partir de una aproximación anterior F_{n-1} de manera aditiva: $F_n = F_{n-1} + \alpha h_n$, donde α es el tamaño de salto y la función h_n es elegida a partir de una familia de funciones H :

$$h_n = \arg \min_{h \in H} \mathcal{L}(F_{n-1} + h) = \arg \min_{h \in H} \mathbb{E}[\ell(y, F_{n-1}(\mathbf{x}) + h(\mathbf{x}))]. \quad (13)$$

El problema de minimización se puede resolver dando un paso en la dirección del gradiente negativo. Considerando $g_n(\mathbf{x}, y) = \frac{\partial \ell(y, F_{n-1}(\mathbf{x}))}{\partial F_{n-1}(\mathbf{x})}$, h_n es elegido de tal forma que aproxime $-g(\mathbf{x}, y)$. Usualmente, una aproximación de mínimos cuadrados es utilizada:

$$h_n = \arg \min_{h \in H} \sum_{m=1}^M (-g_n(\mathbf{x}_m, y_m) - h(\mathbf{x}_m))^2. \quad (14)$$

Los autores identifican un tipo especial de *target leakage* en esta formulación la cual denominan *prediction shift*. La distribución condicional del gradiente $g_n(\mathbf{x}_m, y|\mathbf{x}_m)$ para una instancia de entrenamiento $\mathbf{x}_m$ está desplazada respecto a esa distribución en una instancia de prueba $g_n(\mathbf{x}, y|\mathbf{x})$, por ende, la función h_n está sesgada desde la solución de la ecuación 13, lo que afecta finalmente la capacidad de generalización del modelo F_n . Esto se debe a un *target leakage* al estimar los gradientes g_n utilizando las salidas de las mismas instancias en las que el modelo F_{n-1} fue construido. La distribución condicional $F_{n-1}(\mathbf{x}_m)|\mathbf{x}_m$ para una instancia de entrenamiento $\mathbf{x}_m$ está por lo general desplazada de la distribución $F_{n-1}(\mathbf{x})|x$ para una instancia de prueba $\mathbf{x}$.

CatBoost plantea una estrategia para evitar este desplazamiento en la predicción, denominada *Ordered boosting 2*. Su formulación considera como función de pérdida el error cuadrático $\ell(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$ y, por ende, el gradiente g considerado en 13 es reemplazado por el residuo $r = y - f(\mathbf{x})$. La parte clave de este algoritmo es el cálculo del residuo r_m utilizando un modelo que no ha sido entrenado en la instancia $\mathbf{x}_m$. Para hacer esto, se utiliza la permutación σ y M modelos distintos de tal forma que el modelo F_m es aprendido utilizando las primeras m instancias de la permutación. De esta forma para calcular el residuo r_j se utiliza el modelo F_{j-1} .

Algorithm 2 Ordered Boosting

Input: $\mathcal{S}, N$;
 $\sigma \leftarrow$ Permutación de $[1, 2, \dots, M]$;
 $F_m \leftarrow 0$ for $m = 1, 2, \dots, M$;
for $n = 1$ to N **do**
 for $m = 1$ to M **do**
 $r_m \leftarrow y_m - F_{\sigma(m)-1}(\mathbf{x}_m)$;
 end for
 for $m = 1$ to M **do**
 $\Delta F \leftarrow \text{LearnModel}((\mathbf{x}_j, r_j) : \sigma(j) \leq m)$;
 $F_m \leftarrow F_m + \Delta F$;
 end for
end for
return F_M ;

Lamentablemente, el algoritmo 2 no es factible en la mayoría de los problemas prácticos pues requiere entrenar M modelos. *CatBoost* implementa una modificación de este algoritmo considerando árboles de decisión como modelos base [Prokhorenkova et al., 2019].

2.7. Redes neuronales de propagación hacia adelante

Las redes neuronales de propagación hacia adelante, o *feedforward neural networks* (FNN) en inglés, representan un pilar esencial en el ámbito del aprendizaje profundo. Su objetivo principal es aproximar una función objetivo f^* a través de una función parametrizada $f(x; \theta)$, donde θ es el conjunto de parámetros que define la red. En el caso de una FNN de cuatro capas, la función $f(x; \theta)$ se puede descomponer como $f^{(4)} \circ f^{(3)} \circ f^{(2)} \circ f^{(1)}$, donde $f^{(1)}$ es la capa de entrada y $f^{(4)}$ es la capa de salida.

Para una FNN con L capas, la operación que cada capa realiza sobre su entrada se puede definir mediante una función de activación $\sigma^{(l)}$ y una matriz de pesos $\mathbf{W}^{(l)}$. En cada capa, se lleva a cabo una transformación lineal, conocida como preactivación $\mathbf{Z}^{(l)}$, a la cual se le aplica una función de activación no lineal $\sigma^{(l)}$ para obtener la activación $\mathbf{a}^{(l)}$ de la capa l .

$$\mathbf{a}^{(l)} = \sigma^{(l)}(\mathbf{Z}^{(l)}) = \sigma^{(l)}(\mathbf{W}^{(l)} \mathbf{a}^{(l-1)}).$$

Es importante notar que la activación de la primera capa se considera como los atributos de entrada, es decir, $\mathbf{a}^{(1)} = \mathbf{x}$.

Al observar el entrenamiento de una FNN desde una perspectiva de alto nivel, el proceso comparte similitudes con el entrenamiento de otros modelos de aprendizaje automático que emplean descenso de gradiente, como se detalla en la sección 2.4.3. Sin embargo, una distinción crucial reside en la no linealidad inherente a las FNN, que resulta en una función de pérdida no convexa. Durante el entrenamiento, las entradas $\mathbf{x}$ se propagan hacia adelante a través de la red para generar la salida $\hat{\mathbf{y}}$. Para optimizar los parámetros de la red, es esencial llevar a cabo un proceso de **propagación hacia atrás**, también conocido como **retropropagación** [Rumelhart *et al.*, 1986]. Este algoritmo calcula el gradiente de la función de pérdida con respecto a los parámetros, permitiendo ajustes efectivos durante el aprendizaje.

Aunque las redes neuronales de propagación hacia adelante son ampliamente reconocidas por su eficacia en áreas como el reconocimiento de imágenes, el procesamiento del lenguaje natural, la predicción de series temporales y la toma de decisiones en sistemas autónomos, su aplicabilidad se extiende de manera notable a la manipulación de datos tabulares, especialmente cuando se cuenta con un gran volumen de datos. En este contexto, a pesar de que se suelen preferir métodos más tradicionales como la regresión lineal o los árboles de decisión, las FNN ofrecen la capacidad única de modelar interacciones no lineales y complejas entre las variables. Esta versatilidad las convierte en una herramienta poderosa para tareas de clasificación y regresión en conjuntos de datos tabulares de gran tamaño o alta dimensionalidad. En estos casos, la habilidad de las FNN para aprender representaciones abstractas y capturar relaciones intrincadas entre las características puede llevar a un rendimiento significativamente mejorado, superando las limitaciones de modelos más convencionales que no pueden abordar la complejidad inherente en los datos.

2.8. Transferencia de Aprendizaje

La transferencia de aprendizaje, o *transfer learning* en inglés, es una prometedora metodología de aprendizaje de máquinas que tiene por objetivo mejorar el rendimiento de una hipótesis en un **dominio objetivo** mediante la transferencia del conocimiento contenido en un diferente pero relacionado **dominio fuente** [Zhuang *et al.*, 2020].

A continuación, se definen una serie de conceptos fundamentales sobre transferencia de aprendizaje:

- **Dominio:** Un dominio $\mathcal{D} = \{\mathcal{X}, P(X)\}$ consiste en dos partes: un espacio de atributos $\mathcal{X}$ y una distribución de probabilidad $P(X)$.
- **Tarea:** Una tarea $T = \{\mathcal{Y}, f\}$ consiste en dos partes: un espacio de salida y una hipótesis f . En este caso, la función f es la que se busca aprender.
- **Transferencia de aprendizaje:** Considerando $m^{(S)} \in \mathbb{N}^+$ dominios fuente y tareas, como así también, algunas observaciones de $m^{(T)} \in \mathbb{N}^+$ dominios objetivo y tareas, la transferencia de aprendizaje utiliza el conocimiento implícito en los dominios fuente para mejorar el rendimiento de las hipótesis en los dominios objetivo.

La definición anterior de transferencia de aprendizaje es general para $m^{(S)}$ dominios fuente y $m^{(T)}$ dominios objetivo, sin embargo, por lo general se considera el escenario en donde $m^{(S)} = m^{(T)} = 1$.

Según [Zhuang *et al.*, 2020] existen distintos tipos de categorizaciones para los problemas de transferencia de aprendizaje. Una primera categorización se basa en la disponibilidad de las salidas y en los distintos dominios, cuando las salidas están disponibles en ambos dominios el escenario se categoriza como transferencia de aprendizaje **inductiva**, cuando las salidas están solo disponibles en los dominios fuente se categoriza como transferencia de aprendizaje **transductiva** y cuando las salidas no están disponibles en ningún dominio se categoriza como transferencia de aprendizaje **no supervisado**. Otra categorización se basa en la consistencia en los espacios de atributos y de salida entre los dominios fuente y objetivo, cuando $\mathcal{X}^{(S)} = \mathcal{X}^{(T)}$ y $\mathcal{Y}^{(S)} = \mathcal{Y}^{(T)}$ el escenario se categoriza como transferencia de aprendizaje **homogénea**, de lo contrario, como transferencia de aprendizaje **heterogénea**.

De acuerdo con [Pan y Yang, 2010], los diferentes enfoques en aprendizaje automático pueden clasificarse en cuatro categorías: basados en instancias, basados en atributos, basados en parámetros y basados en relaciones. Un estudio más reciente [Zhuang *et al.*, 2020] examina varios enfoques de transferencia de aprendizaje desde dos perspectivas distintas: **basadas en datos** y **basadas en modelos**. Por un lado, las estrategias basadas en datos se centran en transferir conocimiento mediante la manipulación y transformación de los datos. Ejemplos de estas estrategias incluyen la **ponderación de instancias** y la **transformación de atributos**, que ajustan la importancia o las características de los datos de entrenamiento para adaptarse

mejor al problema a resolver. Por otro lado, las estrategias basadas en modelos se enfocan en mejorar la precisión de las predicciones en el dominio objetivo a través de diversas técnicas, como la regularización de modelos, el ajuste de parámetros y la combinación de múltiples modelos en ensamblados.

2.9. Generalización apilada

En [Wolpert, 1992], se introduce una estrategia para minimizar el error de generalización de uno o más predictores. Esta técnica, conocida como *stacked generalization* o simplemente *stacking*, busca mejorar el rendimiento predictivo al combinar múltiples modelos base y aprovechar su conocimiento adquirido en un dominio fuente para resolver una tarea en un dominio objetivo [Ting y Witten, 2011].

Aunque el *stacking* comparte cierta similitud con la transferencia de aprendizaje en términos de la utilización de conocimiento previo, existe una diferencia fundamental en la forma en que ambos enfoques abordan este concepto. En la transferencia de aprendizaje, se emplea el conocimiento previamente adquirido en el dominio fuente, como las representaciones latentes aprendidas en las capas ocultas de una red neuronal, en el caso del aprendizaje profundo. Un ejemplo de esto es tomar un modelo preentrenado en una tarea de clasificación de imágenes y ajustar sus últimas capas para adaptarlo a una tarea diferente en el dominio objetivo, conservando las representaciones de características aprendidas en las capas anteriores.

Por otro lado, el *stacking* se centra en combinar las predicciones de varios modelos base entrenados en el dominio fuente como entradas a un modelo *meta*. En lugar de depender de las representaciones ocultas aprendidas, el *stacking* permite que el modelo *meta* aprenda cómo asignar pesos adecuados a las predicciones de los modelos base, mejorando así el rendimiento predictivo general en el dominio objetivo.

2.10. Inteligencia artificial explicable

La *eXplainable Artificial Intelligence* (XAI) es un área emergente de investigación en la intersección entre la inteligencia artificial (IA) y la interpretación de modelos, que busca abordar el desafío de la falta de transparencia en los sistemas de aprendizaje automático [Barredo Arrieta *et al.*, 2020]. A medida que los algoritmos de IA se vuelven más complejos y se aplican en una variedad creciente de campos, se hace cada vez más necesario que los sistemas de aprendizaje automático sean comprensibles y justificables en sus procesos de toma de decisiones [Guidotti *et al.*, 2018].

La XAI aborda esta preocupación al desarrollar técnicas y métodos que permiten a los usuarios y partes interesadas entender, confiar e interactuar de manera efectiva con los modelos

de aprendizaje automático. Estos enfoques incluyen métodos de interpretación de modelos locales y globales, visualización de características y datos, y la creación de modelos comprensibles por diseño [Barredo Arrieta *et al.*, 2020, Holzinger, 2018].

Al mejorar la transparencia y responsabilidad de los sistemas de IA, la XAI facilita su adopción en entornos sensibles como medicina, finanzas y justicia, donde las decisiones pueden tener consecuencias significativas para las personas y las organizaciones involucradas [Guidotti *et al.*, 2018]. Además, la XAI puede contribuir al desarrollo de sistemas de IA más robustos y éticos al proporcionar mecanismos para detectar y abordar problemas como el sesgo en los datos y en los modelos [Holzinger, 2018].

2.10.1. LIME

Uno de los métodos fundamentales en XAI es el método LIME [Ribeiro *et al.*, 2016]. LIME es un modelo agnóstico que explica de manera local las predicciones de un modelo **caja negra** f mediante la aproximación de f con un modelo **explicable** g en una **vecindad** de la instancia de interés x . LIME utiliza una **representación interpretable** $x' \in \{0, 1\}^{d'}$ de la instancia $x \in \mathbb{R}^d$. De esta forma, el dominio de la explicación g es $\{0, 1\}^{d'}$ y, por lo tanto, g actúa sobre la presencia o ausencia de *componentes interpretables* [Ribeiro *et al.*, 2016]. Para generar la vecindad se utiliza una función $\pi_x(z)$ que mide la distancia entre dos puntos x y z en el espacio de atributos interpretable. Adicionalmente, dado que la idea es aproximar f mediante un modelo interpretable, se considera una penalización Ω que mide la complejidad de g . Finalmente, y considerando una función de pérdida $\mathcal{L}(f, g, \pi_x)$ que mide la imprecisión del modelo g al aproximar f en la vecindad de x de acuerdo a la función de distancia π , la explicación producida por LIME que asegura **interpretabilidad** y **fidelidad local** se obtiene según la siguiente fórmula:

$$\xi(x) = \arg \min_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g). \quad (15)$$

Donde G es el conjunto de todos los modelos interpretables considerados [Ribeiro *et al.*, 2016], como por ejemplo modelos lineales o árboles de decisión.

2.10.2. SHAP

SHAP (SHapley Additive exPlanations) [Lundberg y Lee, 2017] es otro método prominente en el campo de la Inteligencia Explicable (XAI) que también busca explicar las predicciones de un modelo de aprendizaje automático **caja negra** f . A diferencia de LIME, SHAP se basa en la teoría de juegos cooperativos para distribuir de manera "justa" la contribución de cada atributo a una predicción específica. En particular, utiliza los valores de Shapley, una solución

conceptuada en la teoría de juegos, para asignar un valor a cada atributo de entrada que explica cómo ese atributo contribuye, en promedio, a la predicción para una instancia x .

Al igual que LIME, SHAP utiliza una representación interpretable x' de la instancia x .

Para calcular los valores de Shapley, se consideran todas las posibles combinaciones de atributos y se evalúa cómo la inclusión de un atributo específico cambia la predicción del modelo. Matemáticamente, el valor de Shapley ϕ_i para el atributo i -ésimo se calcula como sigue:

$$\phi_i(x) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} [f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S)] . \quad (16)$$

Donde N es el conjunto de todos los atributos y S es un subconjunto de N que no incluye el atributo i . $f_{S \cup \{i\}}$ y f_S son las predicciones del modelo cuando se incluye o se excluye el atributo i , respectivamente.

La suma de todos los valores de Shapley para una instancia específica x es igual a la diferencia entre la predicción del modelo para esa instancia y la predicción media del modelo para todas las instancias. Esto asegura una propiedad de consistencia en las explicaciones.

$$f(x) - E[f(\mathbf{X})] = \sum_{i=1}^n \phi_i(x). \quad (17)$$

En resumen, mientras que LIME se enfoca en la aproximación local utilizando un modelo interpretable, SHAP proporciona una descomposición aditiva de la predicción del modelo en términos de contribuciones de atributos individuales, basada en una fundamentación teórica sólida proveniente de la teoría de juegos.

2.11. Predicción de precios de automóviles

En la literatura académica, se han explorado numerosos algoritmos de aprendizaje automático con el objetivo de predecir los precios de vehículos, abordando desde modelos clásicos, como la regresión lineal y polinómica, árboles de decisión, máquinas de soporte vectorial y ensamblados, hasta enfoques más contemporáneos, como las redes neuronales profundas. No obstante, **la falta de un conjunto de datos estándar** para la predicción de precios de automóviles **dificulta la comparación de los resultados obtenidos en cada investigación**. Por lo tanto, resulta esencial evaluar las conclusiones en el contexto específico del conjunto de datos utilizado.

En [Lessmann y Voß, 2017] se investiga exhaustivamente el potencial de distintos modelos de aprendizaje de máquinas para apoyar la toma de decisiones en el contexto de fijación de

precios, en inglés *pricing*, de automóviles de segunda mano. Los autores guiaron el desarrollo de su investigación a partir de las siguientes preguntas:

1. ¿Cuáles modelos predicen los precios de reventa con mayor precisión?
2. ¿Cómo influyen las diferentes formas de abordar la **no linealidad** en la precisión de las predicciones de precios de reventa?
3. ¿Qué tan efectivos son los diferentes enfoques para enfrentar la alta dimensionalidad en la predicción de precios de reventa?
4. ¿La combinación de predicciones aumenta la precisión en la predicción de precios de reventa?
5. ¿La información privada ayuda a predecir los precios de reventa de mejor forma?
6. ¿En qué medida la **especificidad** del modelo afecta la precisión de las predicciones de precios de reventa? Entendiendo el nivel de especificidad de un modelo como una medida de la cantidad de modelos de automóviles distintos que se consideran en su entrenamiento.

Las preguntas 2, 3 y 6 son particularmente interesantes para la presente investigación, razón por la cual serán abordadas a continuación en detalle. Los autores plantean que la relación entre el precio de reventa y los atributos de entrada es probablemente no lineal. Un simple ejemplo es la depreciación de los automóviles: los automóviles nuevos pierden valor rápidamente en los primeros años de su vida útil, pero luego la depreciación se ralentiza. Es por esto, que en la investigación experimentan distintas formas de abordar la no linealidad y su impacto en la calidad de las predicciones. Por otro lado, identifican inmediatamente la **alta dimensionalidad** de los datos como un desafío, y por ende, examinan cómo los distintos modelos se comportan en este escenario. Finalmente, definen un importante concepto: la **heterogeneidad**. Los autores indican que la heterogeneidad ocurre cuando los efectos de los atributos de entrada en la salida difieren entre subgrupos de observaciones. Además, indican que es una característica presente en la predicción de precios de reventa de automóviles. Para esto, dan un simple ejemplo: el color rojo en autos deportivos es un atributo que probablemente aumenta el precio de reventa, mientras que en autos familiares es un atributo que disminuye el precio de reventa. Es por esto, que los autores comparan la capacidad de generalización de modelos específicos (aquellos construidos para predecir el precio de reventa de un solo modelo de automóvil) y modelos generales (aquellos construidos para predecir el precio de reventa de distintos modelos de automóviles).

El conjunto de datos consta de aproximadamente 450.000 ventas realizadas por un concesionario automotriz Alemán. Las 450.000 observaciones abarcan 6 distintos modelos de automóvil, todos de la misma marca, con 5 variaciones *premium* y una estándar. Tanto el nombre del concesionario como los atributos exactos utilizados en la investigación no son revelados

por los autores, sin embargo, un subconjunto de los atributos de entrada utilizados en la investigación son: modelo, antigüedad del automóvil, kilometraje, año, tipo de motor y tipo de transmisión. La salida es modelada como la fracción entre el precio de reventa del automóvil y su precio de lista, es decir, como un **valor residual**.

Respecto a los resultados, podemos resumirlos en los siguientes puntos:

1. Los métodos de ensamblado (selección de ensamblado y bosques aleatorios) son los que logran el mejor rendimiento en general. Esto sugiere la eficacia de combinar distintas predicciones para predecir el precio de reventa de automóviles.
2. La calidad de las predicciones varía según el modelo de automóvil. Esto puede indicar que el precio de reventa de algunos automóviles es considerablemente más difícil de predecir.
3. Modelos avanzados que capturan la no linealidad generalmente superan a los modelos lineales.
4. La mayoría de los modelos comete un MAE de un 4 % al predecir los valores residuales. Esto puede sugerir que la alta dimensionalidad no fue un problema significativo.
5. Agregar información personal como atributos de entrada, en general, mejora la precisión de las predicciones.
6. Al comparar modelos específicos y generales, sugieren que la heterogeneidad es la principal razón por la cual los modelos específicos son más precisos que los modelos generales.
7. Modelos de regresión no lineales no son afectados considerablemente por la heterogeneidad.

Los resultados de esta investigación **deben ser tomados con precaución**, pues si bien las métricas reportadas son robustas, el conjunto de datos solo considera automóviles de una única marca y solo 6 modelos distintos. Este escenario contrasta enormemente con el abarcado en la presente memoria, en donde se tiene un objetivo mucho más ambicioso al considerar más de 40 marcas y más de 1000 modelos. Por ende, es probable que los problemas de alta dimensionalidad y heterogeneidad sean mucho más significativos en el escenario de la presente memoria.

En [Collard, 2022], se explora la aplicación de algoritmos de aprendizaje automático para predecir los precios de autos usados. El estudio compara el rendimiento de varios algoritmos, incluidos Regresión Lineal, Regresión Ridge, Regresión Lasso y Regresión de Bosque Aleatorio, utilizando un gran conjunto de datos públicos. Entre los algoritmos evaluados, la Regresión de Bosque Aleatorio mostró el menor error absoluto porcentual medio (MAPE) en la fase de pruebas, con un valor de 37.65. El conjunto de datos utilizado proviene de Kaggle y

consta de 122144 registros, 464 marcas únicas y 4289 modelos distintos. Sin embargo, es importante señalar que la calidad de los datos tiene ciertas limitaciones. Por ejemplo, algunos valores que toma el atributo marca son en realidad modelos de automóviles, y además hay inconsistencias en cómo se nombra a las mismas marcas en diferentes registros. El paper no ofrece detalles sobre el preprocesamiento de los datos, lo que genera incertidumbre acerca de la cantidad real de marcas y modelos que los algoritmos evaluados pueden soportar.

En un artículo científico reciente [Liu *et al.*, 2022], se establece el estado del arte en la predicción de precios de vehículos usados mediante un modelo de red neuronal poco profunda de tres capas y el algoritmo de aprendizaje de retropropagación, optimizado con la técnica de optimización por enjambre de partículas (PSO, por sus siglas en inglés). La investigación se llevó a cabo utilizando un conjunto de datos que contiene 10.260 transacciones del principal mercado de comercio electrónico de vehículos usados en China, iautos. Este conjunto de datos incluye 12 atributos: marca, tipo de tracción, tipo de transmisión, potencia del motor, tipo de carrocería, kilometraje, tiempo de uso, cilindrada, consumo de combustible, normas de emisiones, región y precio del automóvil nuevo. Es importante destacar que, en el conjunto de datos mencionado, **no se proporciona información específica sobre la cantidad de marcas y modelos de automóviles** presentes. Utilizando el análisis relacional gris (GRA, por sus siglas en inglés) se seleccionaron los siguientes atributos en orden de importancia para alimentar los modelos de aprendizaje automático: precio del automóvil nuevo, cilindrada, kilometraje, tipo de transmisión, consumo de combustible y tiempo de uso. En cuanto a los resultados, el modelo de red neuronal poco profunda (BPNN) obtuvo un error porcentual absoluto medio (MAPE) del 9.891 % al utilizar todos los atributos del conjunto de datos, mientras que al emplear únicamente los atributos seleccionados mediante GRA (GRA-BPNN), se alcanzó un MAPE del 7.118 %. Al incluir la optimización con PSO (PSO-GRA-BPNN), se logró un MAPE del 3.936 %. Al comparar PSO-GRA-BPNN con otros modelos clásicos de aprendizaje automático, como bosques aleatorios, regresión lineal múltiple y máquinas de vectores de soporte, PSO-GRA-BPNN demostró ser ampliamente superior en términos de MAPE.

En [Chen *et al.*, 2018] se plantean una serie de modelos de aprendizaje de máquinas para predecir el precio de productos en base a imágenes. El conjunto de datos de automóviles consistía en 1400 imágenes, cada una con su precio. En esta investigación el precio utilizado como *target* para el conjunto de datos de automóviles es el **precio de venta sugerido por el fabricante** (MSRP, por sus siglas en inglés) y el modelamiento se llevó a cabo mediante dos clásicos enfoques: regresión y clasificación. Para la clasificación, los datos se etiquetaron utilizando segmentos basados en percentiles, es decir, a cada automóvil se le asignó un segmento considerando su precio en relación a los percentiles 20, 40, 60, 80 y 100 de la distribución de precios total. Los modelos de redes neuronales convolucionales propuestos aprovechan el conocimiento de las redes VGG-16 y SqueezeNet. Se lograron resultados del estado del arte en la predicción de precios de automóviles basada en atributos visuales mediante el uso de arquitecturas modernas de aprendizaje profundo, como así también, técnicas avanzadas de aprendizaje de máquinas como *transfer learning* en redes neuronales convolucionales.

Capítulo 3

Propuesta de solución

En este capítulo se describe la propuesta de solución para el problema planteado en el primer capítulo, desde la adquisición de datos hasta el entrenamiento. Se detallan las fuentes de datos y sus atributos con sus respectivas unidades, número de valores únicos y número de valores faltantes, para luego realizar un análisis de datos exploratorio. Luego se explica el preprocesamiento, una tarea fundamental que incluye la limpieza, la transformación y la corrección de inconsistencias, con el fin de garantizar la calidad y la confiabilidad de los datos. En la misma sección se describe cómo se abordaron los valores atípicos, la estandarización de los datos, la codificación de atributos categóricos y la imputación de valores faltantes. Luego, en la sección de modelamiento, se presentan las diferentes técnicas de modelamiento empleadas para resolver la predicción de valores de *trade-in*. Finalmente, en la sección de entrenamiento, se detallan los diferentes experimentos realizados para encontrar el modelo más adecuado.

3.1. Metodología

La metodología utilizada en la presente investigación está fuertemente basada en CRISP-DM [Wirth y Hipp, 2000].

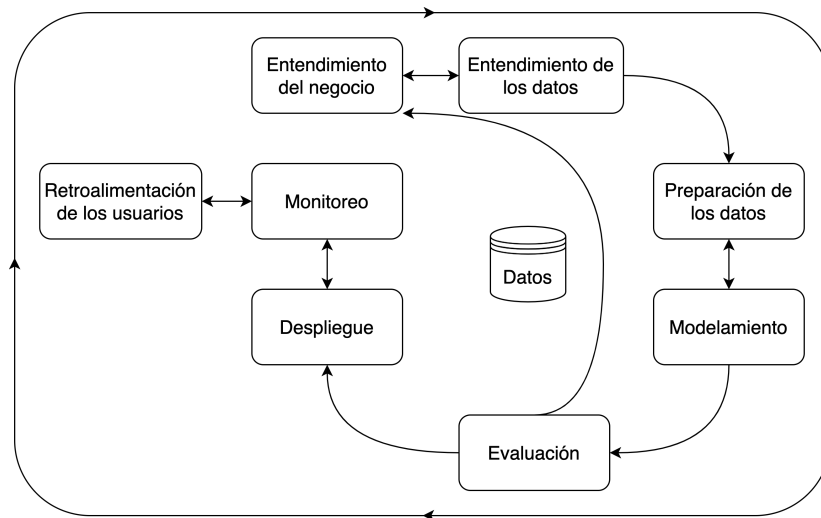


Figura 5: Metodología de la investigación
Fuente: Elaboración propia

En esta adaptación del marco CRISP-DM, se incorporan dos etapas adicionales: el monitoreo y la retroalimentación por parte de los usuarios. El monitoreo implica supervisar el desem-

peño del modelo en el entorno de producción con el propósito de identificar y corregir rápidamente cualquier inconveniente en su implementación. Por otro lado, la retroalimentación de los usuarios permite obtener información valiosa sobre la percepción que los usuarios del sistema de *ML* tienen respecto a las predicciones y tiempo de respuesta proporcionados por el modelo.

El **entendimiento del negocio** es abordado en el capítulo 1, en donde se describió el contexto de la investigación, el proceso actual para obtener un valor de *trade-in* y las problemáticas asociadas a este proceso. El **entendimiento de los datos** es abordado en las secciones 3.2 y 3.3, mientras que la **preparación de los datos** es abordada en la sección 3.4. La etapa **modelamiento** es abordada en la sección 3.5. Finalmente, la etapa **evaluación** será abordada en el siguiente capítulo.

3.2. Fuentes de datos

Las fuentes de datos son un elemento crítico en cualquier proyecto de ciencia de datos, ya que proporcionan la información base para el análisis y la creación de modelos. Un entendimiento sólido de estas es vital, ya que impacta directamente en la calidad de los resultados obtenidos. Este entendimiento no solo implica tener una idea clara de la estructura y formato de los datos, sino también del contexto en el que se recogen y utilizan. Comprender la procedencia de los datos, la metodología de recolección y las posibles limitaciones o sesgos de los datos, es crucial para realizar análisis precisos y construir modelos robustos. Además, este conocimiento ayuda a identificar posibles desafíos en la etapa de preprocesamiento, como el tratamiento de datos faltantes o la eliminación de ruido, así como a guiar la estrategia de ingeniería de atributos, a fin de extraer las características más informativas para el modelo.

Los datos utilizados en la presente investigación provienen principalmente de dos fuentes de datos: Ooyyo y SeezPAD. Por un lado, los datos provenientes del sitio web Danés Ooyyo consisten en *listings* de automóviles, es decir, **publicaciones de venta** de automóviles realizadas por sus dueños. Por el otro, los datos provenientes de SeezPAD consisten en automóviles valorados por el equipo de ventas de Seez para un posible *trade-in*. Además, se utiliza la API de *Bilstatistik* y el sitio web *Nummerplade* para enriquecer los datos. Estas últimas dos fuentes de datos proporcionan detalles técnicos de automóviles de Dinamarca a partir de su VIN o patente.

Respecto a los sesgos y alcances de los datos se puede mencionar, por ejemplo, que las valoraciones provenientes desde la fuente de datos SeezPAD son realizadas por el equipo de ventas de Seez. Por ende, estas valoraciones pueden estar sesgadas por la subjetividad de los evaluadores. De manera similar, los precios de lista provenientes de Ooyyo pueden estar sesgados por la subjetividad de los vendedores. Estos sesgos, son particularmente relevantes para aquellos modelos de vehículos poco frecuentes en los datos disponibles.

En la tabla 1 se resumen las fuentes de datos utilizadas en la presente investigación.

Fuente de datos	Descripción	Número de atributos	Número de datos
Ooyyo	Precios de lista	11	3,394,168
SeezPAD	Valores de <i>trade-in</i>	9	869
Bilstatistik	Detalles de automóviles	20	-
Nummerplade	Detalles de automóviles	7	-

Tabla 1: Resumen de fuentes de datos
Fuente: Elaboración propia

En la tabla 2 se presentan 5 registros sin preprocesar provenientes de la fuente de datos Ooyyo. Cada registro está asociado a una publicación de venta de un automóvil en el sitio web Ooyyo. Los valores faltantes están representados mediante el valor *NaN*.

Marca	Modelo	Variante	Año	Kilometraje	Combustible	Carrocería	Potencia	Cilindrada	Transmisión	Precio de lista
Mercedes Benz	Glb-Class	250 4Matic	2020	18000	Petrol	Nore	227	NaN	Automatic	629900
Renault	Twingo	1.0 SCe	2015	86000	Petrol	Mini tsi	71	1.0	Manual	61900
Nissan	Juke	1.6 Nismo	2014	113000	Petrol	Crossover	203	1.6	Manual	169900
Peugeot	2008	1.2 vti	2017	34000	Petrol	Crossover	112	1.2	Manual	149500
Opel	Karl	1.0	2017	82000	Petrol	Hatchback	75	1.0	Automatic	79900

Tabla 2: Ooyyo: Ejemplos de datos sin preprocesar
Fuente: Elaboración propia

Analogamente, en la tabla 3 se presentan 5 registros sin preprocesar provenientes de la fuente de datos SeezPAD. Cada registro está asociado a una valoración de *trade-in* realizada por el equipo de ventas de Seez. Los valores faltantes están representados mediante el valor *NaN*.

Marca	Modelo	Variante	Año	Kilometraje	Combustible	Carrocería	Transmisión	Valor trade-in
Skoda	Superb	1.4 TSI Hybrid 218 DSG	2019	68000	Plug-in Hybrid (Benzin)	SUV	Automatisk	95500
Volkswagen	Polo	1.4 TDI	2007	153000	Diesel	Sedan	Automatisk	25000
Citroen	C4 Cactus	1.2 PureTech 110 HK	2018	80000	Benzin	SUV	Automatisk	60000
Peugeot	307	1.6 HDI MPV	2009	280000	Diesel	Hatchback	NaN	50500
Mercedes	C-Klasse	C 220 D Stationcar 9G-Tronic	2018	132000	Diesel	Stationcar	Automatisk	230000

Tabla 3: SeezPAD: Ejemplos de datos sin preprocesar
Fuente: Elaboración propia

Como se puede apreciar, los datos provenientes de SeezPAD no contienen información sobre la potencia o cilindrada del motor.

3.3. Análisis exploratorio de datos

En esta sección se realiza un análisis exploratorio de los datos sin preprocesar provenientes de las fuentes Ooyyo y SeezPAD. El objetivo de esta sección es comprender mejor los datos, identificar patrones y tendencias, así como detectar valores atípicos y anomalías.

En el contexto de transferencia de aprendizaje es clave analizar exhaustivamente la diferencia entre el dominio fuente $\mathcal{D}_s$ definido por los datos provenientes de la fuente de datos Ooyyo y el dominio objetivo $\mathcal{D}_t$ definido en parte por los datos provenientes de la fuente de datos SeezPAD. Un buen análisis puede guiar hacia un valioso pronóstico de la factibilidad de la transferencia de aprendizaje, como así también, de la cantidad de ajustes requeridos para su implementación exitosa.

En la tabla 4 se detallan los atributos de los datos provenientes de la fuente de datos Ooyyo. Cada atributo se presenta con su respectiva unidad de medida (si aplica), su escala de medición subyacente, el número de valores únicos y el porcentaje de valores faltantes. Tomemos como ejemplo el atributo "potencia del motor", que se expresa en caballos de fuerza (hp) y se encuentra en una escala de medición de razón. Este atributo cuenta con 496 valores únicos y presenta valores faltantes en el 23.72 % de los registros recopilados de Ooyyo. Es evidente que varios atributos, como el tipo de carrocería, potencia del motor, cilindrada del motor y el tipo de transmisión, muestran un porcentaje significativo de valores faltantes. Esto subraya la necesidad de **realizar un proceso adecuado de limpieza o imputación de datos** antes de emprender el proceso de entrenamiento.

Atributo	Unidad	Escala de medición	Nº valores únicos	Valores faltantes
Marca	-	Nominal	115	-
Modelo	-	Nominal	1270	-
Variante	-	Nominal	1363	18.65 %
Año	-	Intervalo	98	-
Kilometraje	km	Razón	-	-
Combustible	-	Nominal	9	4.50 %
Tipo de carrocería	-	Nominal	11	50.33 %
Potencia del motor	hp	Razón	496	23.72 %
Cilindrada del motor	L	Razón	69	29.13 %
Transmisión	-	Nominal	3	13.10 %
Precio	DKK	Razón	-	-

Tabla 4: Atributos de la fuente de datos Ooyyo
Fuente: Elaboración propia

En la tabla 5 se proporciona una descripción exhaustiva de los atributos presentes en los datos provenientes de la fuente de datos SeezPAD. Una vez más, se incluye información sobre la unidad de medida aplicable a cada atributo, su escala de medición subyacente, la cantidad de valores únicos y el porcentaje de valores faltantes. Para ilustrar este punto, consideremos el atributo "tipo de carrocería", el cual se encuentra en una escala de medición nominal, presenta 8 categorías únicas y registra valores faltantes en el 24.14 % de las observaciones recopiladas de SeezPAD.

Atributo	Unidad	Escala de medición	Nº valores únicos	Valores faltantes
Marca	-	Nominal	64	-
Modelo	-	Nominal	223	-
Variante	-	Nominal	539	-
Año	-	Intervalo	30	-
Kilometraje	km	Razón	-	0.12 %
Combustible	-	Nominal	15	-
Tipo de carrocería	-	Nominal	8	24.14 %
Transmisión	-	Nominal	2	24.14 %
Valor de Trade-in	DKK	Razón	217	-

Tabla 5: Atributos de la fuente de datos *SeezPAD*

Fuente: Elaboración propia

En la tabla 6 se presenta una lista de los atributos recopilados a través de la API de *Bilstatistik*, incluyendo información sobre su unidad de medida y escala de medición correspondiente.

Atributo	Unidad	Escala de medición
Segmento	-	Ordinal
Alto	mm	Razón
Largo	mm	Razón
Ancho	mm	Razón
Número de asientos	-	Razón
Número de puertas	-	Razón
Combustible	-	Nominal
Consumo de combustible	l/km	Razón
<i>Plugin hybrid</i>	-	Nominal
Potencia del motor	hp	Razón
Cilindrada del motor	cm3	Razón
Número de cilindros	-	Razón
Máxima velocidad	km/h	Razón
Normativa europea de emisión	-	Ordinal
Emisión de CO2	g/km	Razón
Emisión de CO	g/km	Razón
Emisión de NOx	mg/km	Razón
Máxima carga	kg	Razón
Número de dueños	-	Razón
Número de airbags	-	Razón

Tabla 6: Atributos obtenidos desde la API de *Bilstatistik*

Fuente: Elaboración propia

En la tabla 7 se presenta una lista de los atributos recopilados desde el sitio web *Nummerplade*, incluyendo información sobre su unidad de medida y escala de medición correspon-

diente.

Atributo	Unidad	Escala de medición
Potencia del motor	hp	Razón
Cilindrada del motor	cm3	Razón
Número de cilindros	-	Razón
Consumo de combustible	l/km	Razón
Peso	kg	Razón
Máxima velocidad	km/h	Razón
Plugin hybrid	-	Nominal

Tabla 7: Atributos obtenidos desde el sitio web *Nummerplade*
Fuente: Elaboración propia

La figura 6 muestra la distribución de los atributos numéricos provenientes de la fuente de datos Ooyyo utilizando diagrama de cajas. Al respecto, es importante señalar que:

- Los valores atípicos se han omitido con el fin de mejorar la visualización.
- La línea discontinua indica la media de los datos.
- La línea continua indica la mediana de los datos.

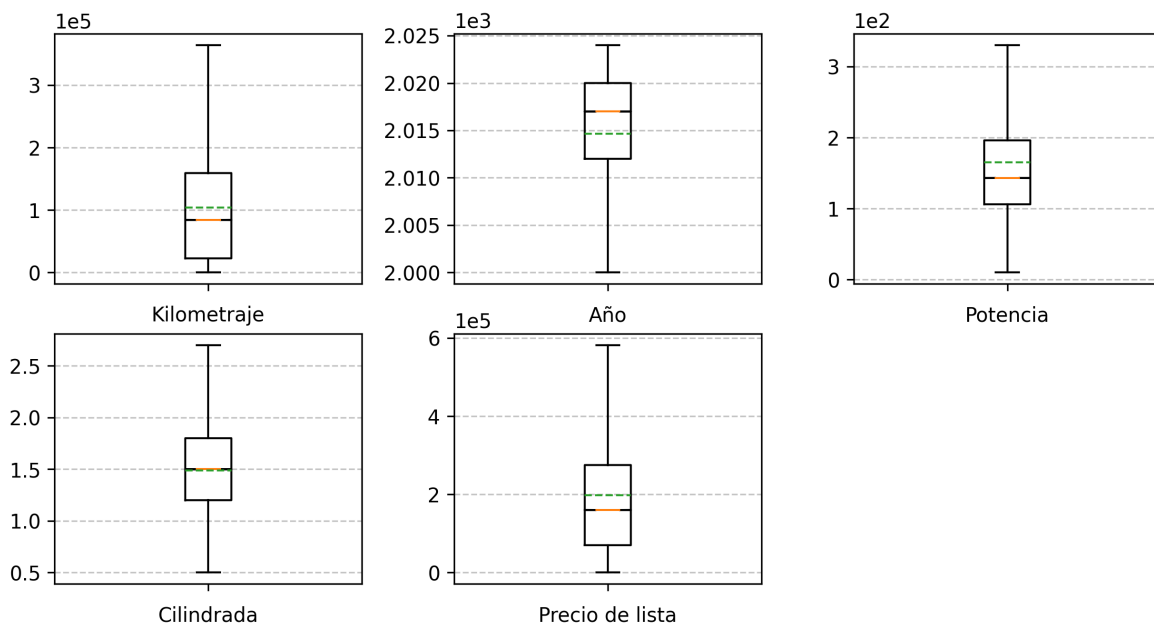


Figura 6: Ooyyo: Distribución de atributos numéricos
Fuente: Elaboración propia

Se observa que el kilometraje tiende a estar cerca de los 100,000 km, el año suele situarse alrededor del 2017, la potencia del motor tiende a rondar los 150 hp y la cilindrada del motor suele encontrarse en torno a 1.5 L. Por otra parte, el precio de lista generalmente se encuentra cerca de los 200,000 DKK, lo que equivale aproximadamente a 25,000,000 CLP. Además, es importante destacar que tanto el kilometraje como el precio de lista muestran distribuciones asimétricas positivas.

La figura 7 ilustra la distribución de los atributos categóricos provenientes de la fuente de datos Ooyo centrándose en los 15 valores más frecuentes. Entre las marcas de automóviles más destacadas se encuentran Volkswagen, Ford, Peugeot y Mercedes Benz. En cuanto al tipo de combustible preferido, el petróleo ocupa el primer lugar, seguido de cerca por el diésel. Además, en lo que respecta al tipo de carrocería, los modelos mini o *city car* lideran las preferencias, seguidos por las categorías de *crossover*, *hatchback* y *SUV*. Por último, la transmisión automática es el tipo de transmisión más común.

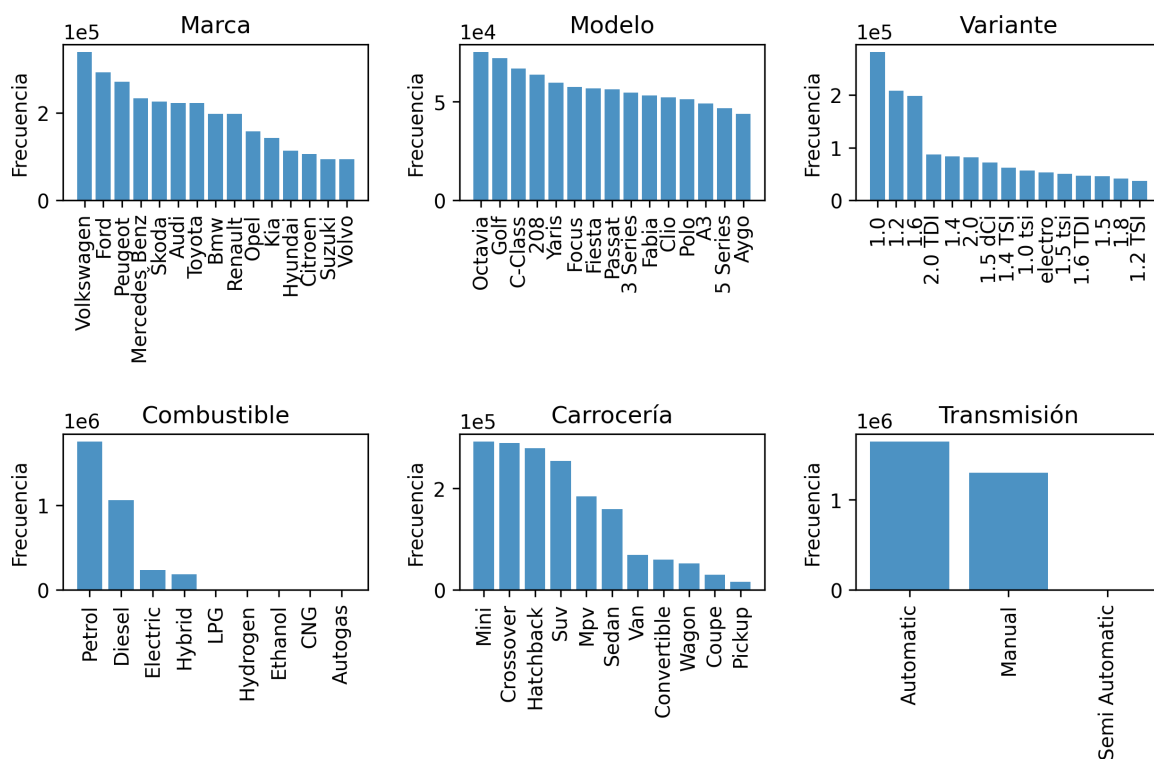


Figura 7: Ooyo: Distribución de atributos categóricos
Fuente: Elaboración propia

La figura 8 ilustra la distribución del precio de lista de los automóviles provenientes de la fuente de datos Ooyo. Para poder visualizar mejor la distribución, los datos se han limitado al intervalo $[0, 1,000,000]$ y todos los valores mayores a 1,000,000 DKK son representados en el gráfico como 1,000,000 DKK.

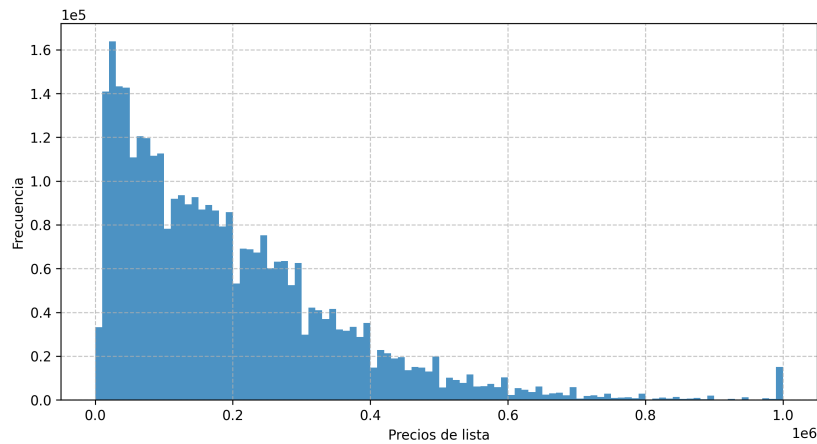


Figura 8: Ooyyo: Distribución del precio
Fuente: Elaboración propia

La figura 9 presenta la distribución de los atributos numéricos extraídos de la fuente de datos *SeezPAD*. Al comparar estos datos con los obtenidos de la fuente de datos *Ooyyo*, se observa que la distribución del kilometraje muestra un desplazamiento leve hacia valores más altos, lo cual es coherente con la naturaleza de los datos provenientes de *SeezPAD*. En cuanto al año, la distribución se inclina ligeramente hacia valores más bajos. Por último, la distribución del valor de *trade-in* exhibe un sesgo hacia valores menores con una mediana cercana a 100,000 DKK, lo que equivale aproximadamente a 13,000,000 CLP, lo cual también se alinea con la naturaleza de los datos de *SeezPAD*.

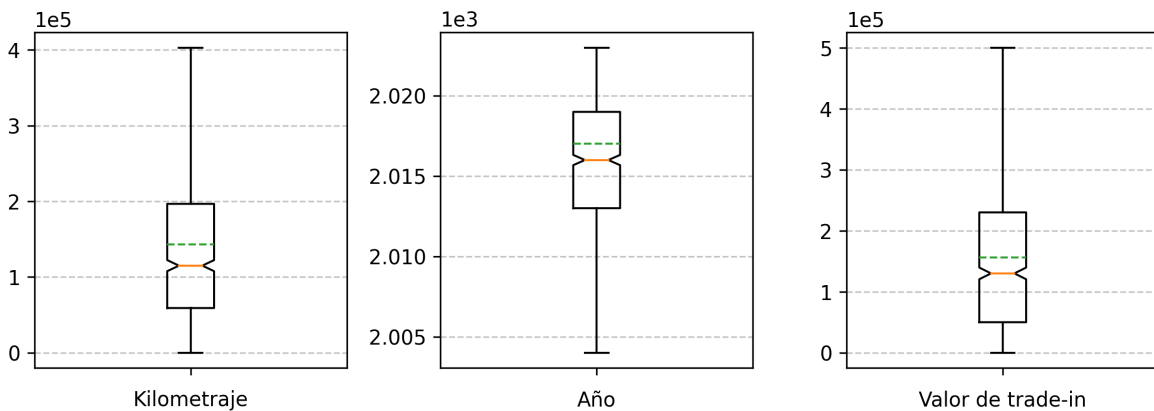


Figura 9: SeezPAD: Distribución de atributos numéricos
Fuente: Elaboración propia

La figura 10 exhibe la distribución de los atributos categóricos extraídos de la fuente de datos *SeezPAD*. Al contrastar estos datos con los obtenidos de la fuente de datos *Ooyyo*, se destaca que una gran proporción de los automóviles pertenecen a la marca Mercedes Benz. No obstante, es notable que las marcas presentes en el top 15 son prácticamente idénticas.

En cuanto al tipo de carrocería y al tipo de combustible, se mantienen distribuciones similares entre ambas fuentes de datos. Sin embargo, es evidente que el tipo de transmisión está significativamente sesgado hacia la transmisión automática en el conjunto de datos de SeezPAD.

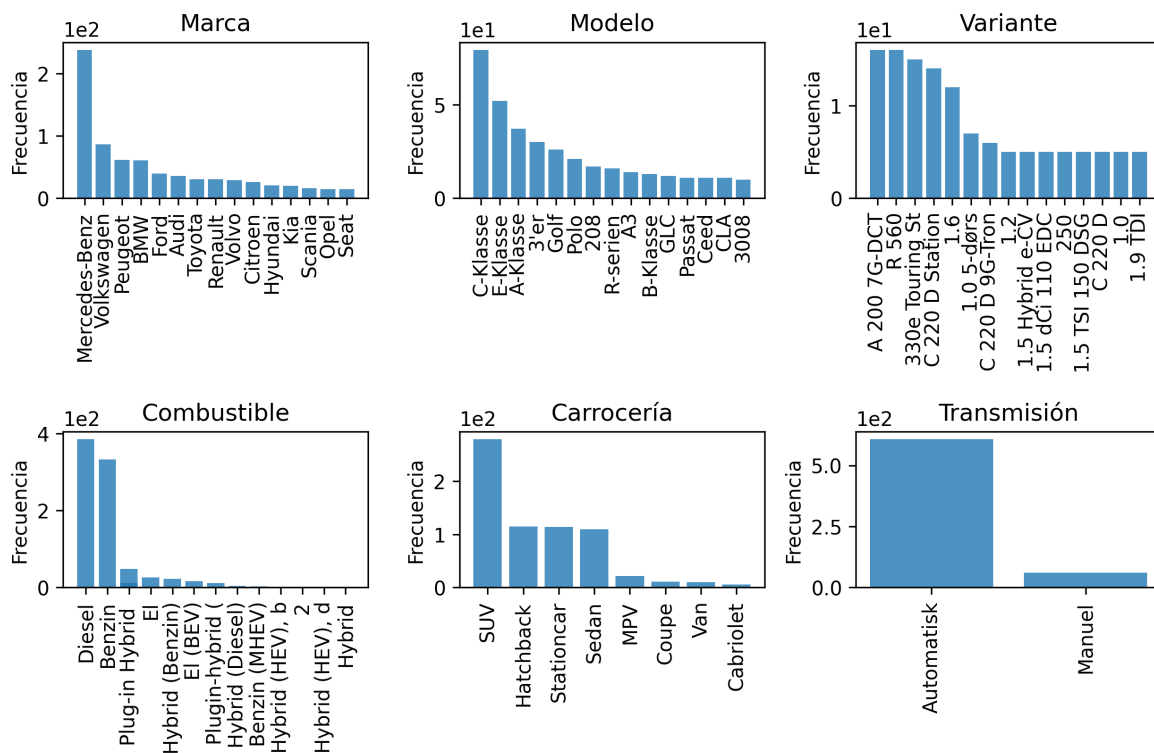


Figura 10: SeezPAD: Distribución de atributos categóricos

Fuente: Elaboración propia

3.4. Preprocesamiento

El preprocesamiento de datos y la ingeniería de atributos son etapas críticas en cualquier proyecto de ciencia de datos y aprendizaje automático. Estas etapas pueden ser, en muchos casos, las más intensivas en tiempo de todo el proyecto. El motivo de esto radica en su importancia fundamental: sin datos de calidad, los modelos que se construyen a partir de ellos estarán inherentemente limitados en su capacidad para producir resultados confiables y precisos.

El objetivo principal del preprocesamiento de datos es mejorar la calidad y fiabilidad de los datos a través de una serie de operaciones que incluyen la limpieza de datos, la transformación de datos y la corrección de inconsistencias. La limpieza de datos implica la identificación y manejo de valores atípicos y la eliminación de ruido y errores presentes en los datos. La transformación de datos incluye operaciones como la normalización y la discretización, que

pueden hacer que los datos sean más manejables y significativos para los modelos de aprendizaje automático. La corrección de inconsistencias, por otro lado, implica asegurarse de que los datos sean coherentes y no contengan contradicciones que puedan sesgar los resultados.

En el contexto de esta investigación, las tareas de preprocesamiento son esenciales para preparar los datos sin preprocesar provenientes de las fuentes *SeezPAD* y *Ooyyo* para su uso en la etapa de modelamiento. Estos datos sin preprocesar son así el punto de partida, y el objetivo es transformarlos en los nuevos conjuntos de datos *Listing* y *Trade-in*.

Estos nuevos conjuntos de datos serán la base para entrenar los modelos predictivos que se utilizan para predecir los precios de lista y los valores de *trade-in*. En otras palabras, estos nuevos conjuntos de datos son la entrada de los modelos que producen las salidas deseadas. La creación de estos conjuntos de datos es un proceso cuidadoso que implica una comprensión profunda de los datos en sí y de cómo se utilizarán en el modelado.

En la figura 11 se indica la *pipeline* de preprocesamiento que construye el conjunto de datos *Listing* a partir de los datos provenientes de *Ooyyo*.

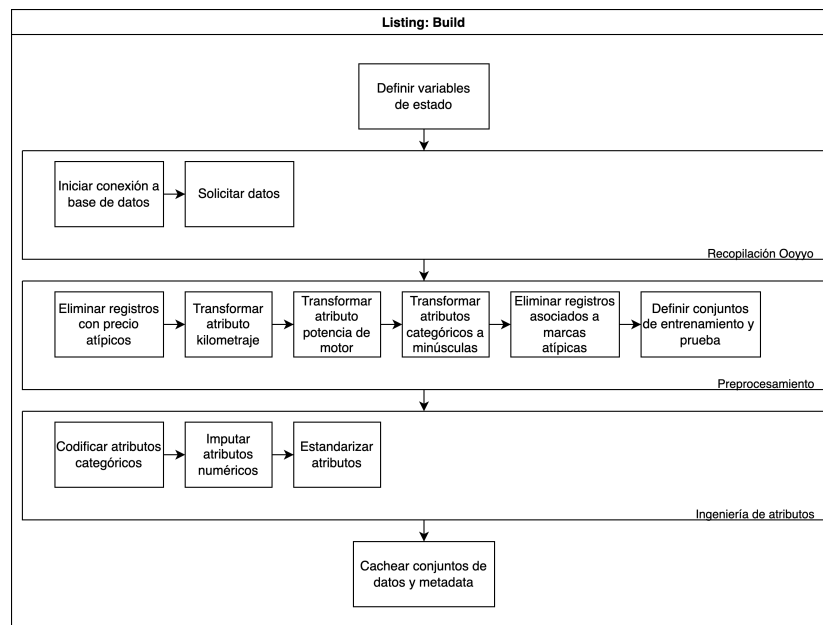


Figura 11: Listing: Proceso de construcción
Fuente: Elaboración propia

En el proceso de depuración de los datos, se abordó la eliminación de registros con precios atípicos, definiéndolos como aquellos cuyos precios de lista se encuentran por debajo del percentil 5 o por encima del percentil 95. Asimismo, se eliminaron registros asociados a marcas consideradas atípicas, es decir, aquellas que cuentan con menos de 50 registros en la base de datos.

Posteriormente, para la generación de los conjuntos de datos que serán utilizados en el en-

trenamiento, validación y prueba del modelo, se aplicó una proporción de 80 %, 10 % y 10 %, respectivamente. Durante esta partición, se prestó especial atención a mantener la **proporción original de registros por marca en cada uno de los conjuntos de datos**, garantizando así una representación uniforme y equitativa de todas las marcas a lo largo de todo el proceso de modelado. Este enfoque contribuye a que el modelo sea imparcial y capaz de generalizar de manera efectiva, al no favorecer desproporcionadamente a ninguna marca en particular durante su entrenamiento y evaluación.

En la figura 12 se indica la *pipeline* de preprocesamiento de los datos que construye el conjunto de datos *Trade-in* a partir de los datos provenientes de *SeezPAD*.

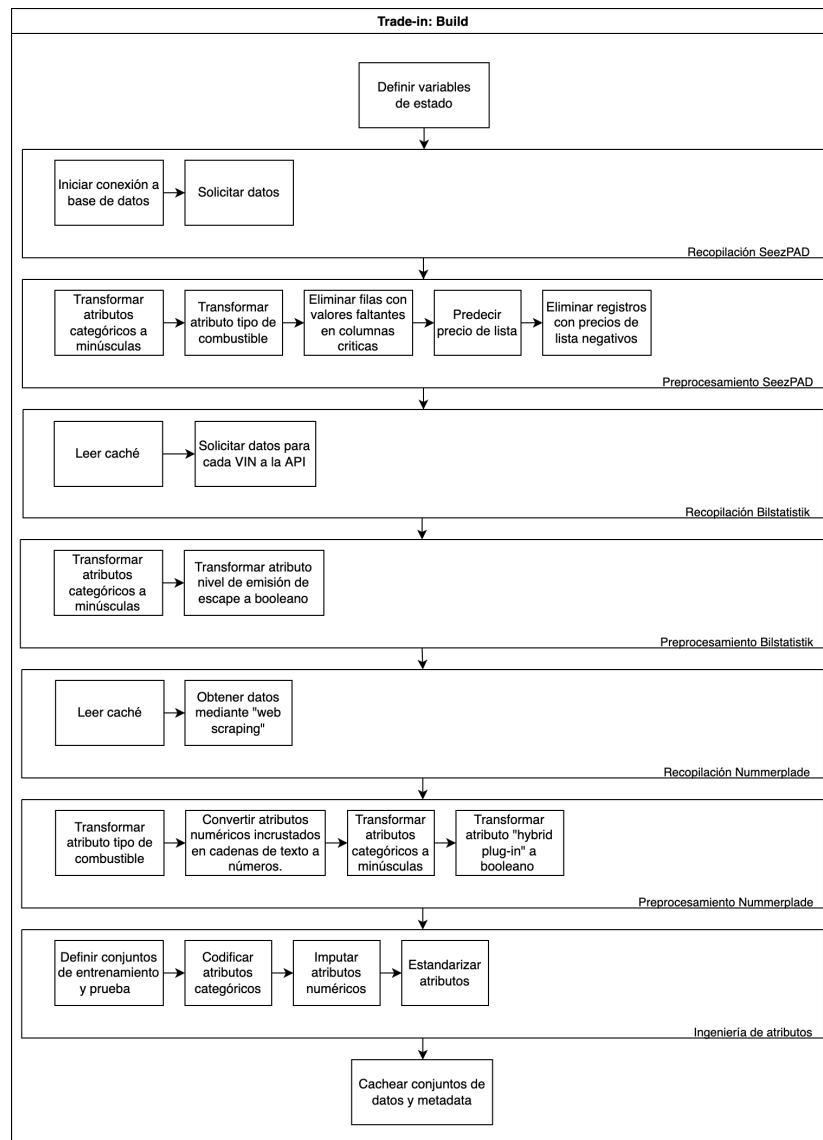


Figura 12: Trade-in: Proceso de construcción
Fuente: Elaboración propia

Como es evidente, esta *pipeline* de preprocesamiento es más compleja que la anterior. Esto se debe principalmente a que el conjunto de datos *Trade-in* utiliza fuentes adicionales de datos para enriquecer los datos provenientes de *SeezPAD*.

El proceso de ingeniería de atributos para ambos conjuntos de datos se detalla a continuación. En primer lugar, se codificaron los atributos categóricos utilizando vectores *one-hot* de las clases presentes en el conjunto de datos de entrenamiento. A continuación, se imputaron los valores numéricos faltantes utilizando las medias del conjunto de entrenamiento. Por último, se estandarizaron los atributos utilizando las medias y desviaciones estándar del conjunto de entrenamiento. Este proceso de preprocesamiento de datos permitió obtener conjuntos de datos más completos y normalizados, listos para su posterior análisis y modelado en la etapa de aprendizaje automático.

Como resultado de este proceso, se obtuvieron los conjuntos de datos *Listing* y *Trade-in* mencionados en la tabla 8.

Conjunto de datos	Tipo	Número de datos
<i>Listing</i>	Entrenamiento	2,437,400
<i>Listing</i>	Validación	304,673
<i>Listing</i>	Prueba	304,623
<i>Trade-in</i>	Entrenamiento	691
<i>Trade-in</i>	Validación	86
<i>Trade-in</i>	Prueba	87

Tabla 8: Conjuntos de datos
Fuente: Elaboración propia

La figura 13 muestra la correlación de *Pearson* en los atributos numéricos del conjuntos de datos *Listing*. Es evidente una considerable una correlación positiva con valor 0.63 entre el precio de lista y la potencia del motor, como así también, una correlación negativa con valor -0.47 entre el precio de lista y el kilometraje y una correlación positiva con valor de 0.47 entre el precio de lista y el año.

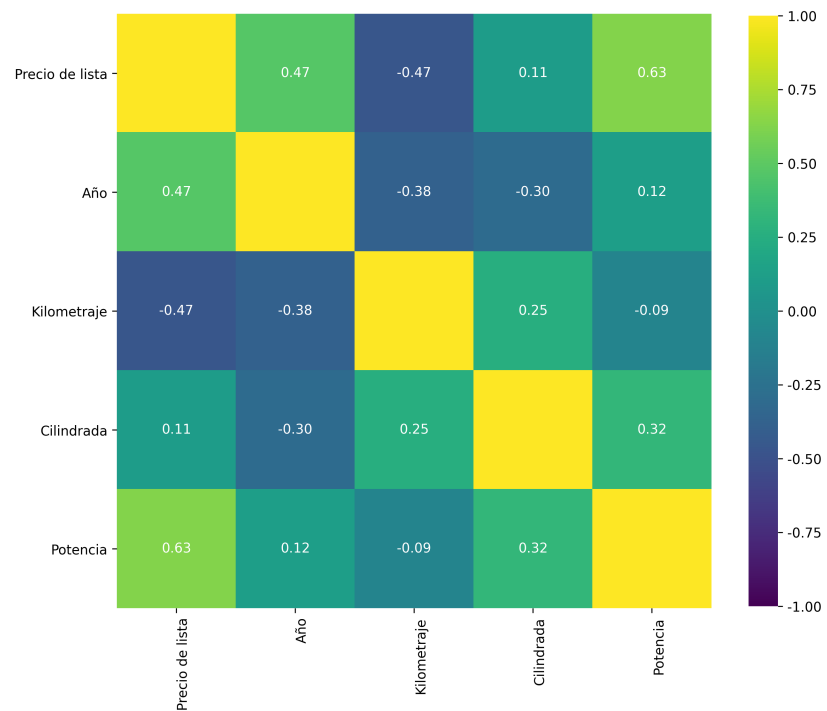


Figura 13: Listing: Matriz de correlación lineal
Fuente: Elaboración propia

De manera similar, la figura 14 presenta una importante correlación positiva con valor 0.63 entre el valor de *trade-in* y el año del automóvil. Esta misma correlación está presente entre el valor de *trade-in* y el ancho del automóvil.

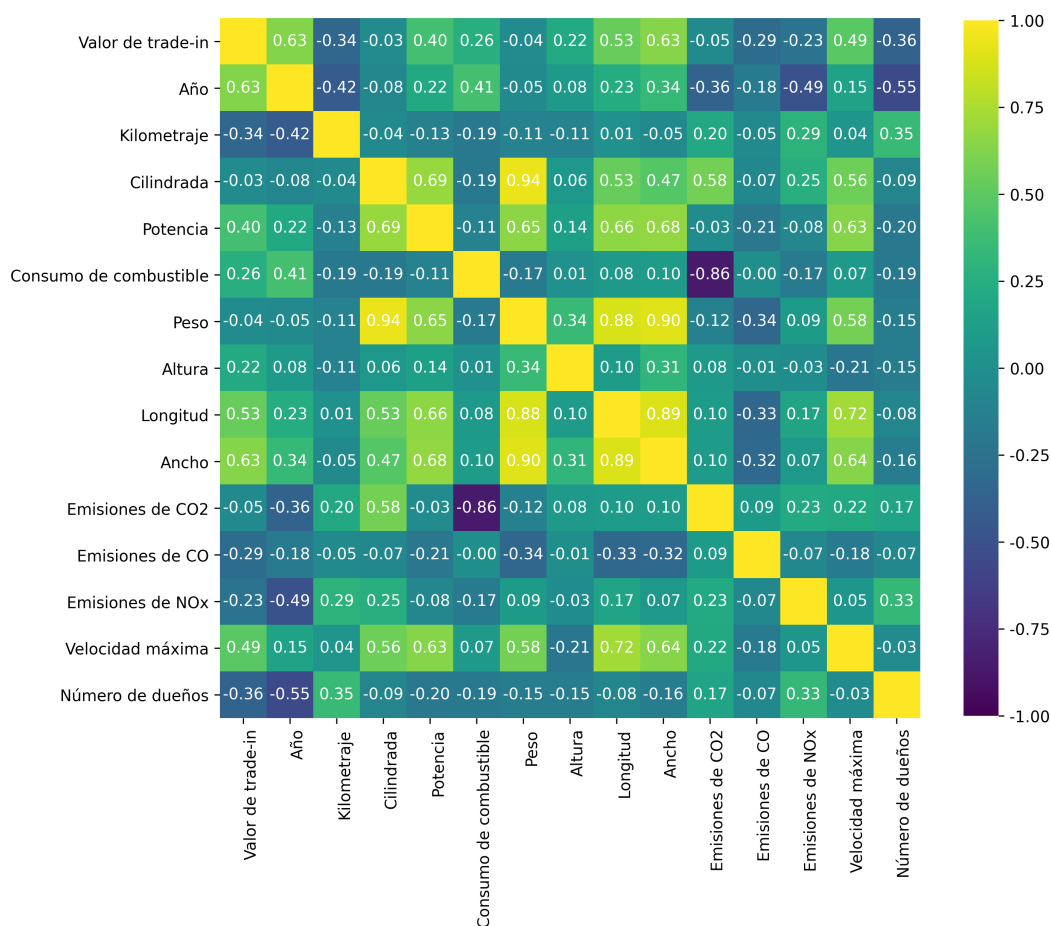


Figura 14: Trade-in: Matriz de correlación lineal
Fuente: Elaboración propia

Es interesante notar que el año del automóvil está más correlacionado con el valor de *trade-in* que con el precio de lista.

3.5. Modelamiento

La fase de modelamiento busca aprovechar al máximo la información disponible para diseñar modelos predictivos capaces de generalizar hacia datos no observados. En esta sección, se detallan las técnicas de aprendizaje automático empleadas para predecir tanto los precios de lista como los valores de *trade-in*.

En la siguiente tabla se resumen las cuatro diferentes técnicas de modelamiento empleadas en la presente investigación:

Técnica	Descripción
<i>Two-stage prediction-oriented (2SPO)</i>	Primero se entrena un modelo base para predecir precios de lista y luego se entrena un modelo de ajuste para predecir valores de <i>trade-in</i> considerando la predicción del modelo base como uno de los atributos de entrada.
<i>Two-stage representation-oriented (2SRO)</i>	Primero se entrena una red neuronal como modelo base para predecir precios de lista y luego se entrena un modelo de ajuste para predecir valores de <i>trade-in</i> considerando las representaciones latentes de la última capa escondida del modelo base como parte de los atributos de entrada.
<i>Three-stage prediction-oriented (3SPO)</i>	Primero se entrenan independientemente distintos modelos 2SPO considerando el mismo modelo base y luego se entrena un modelo de <i>Stacking</i> para predecir valores de <i>trade-in</i> considerando las predicciones de los modelos 2SPO como atributos de entrada.
<i>Three-stage representation-oriented (3SRO)</i>	Primero se entrenan independientemente distintos modelos 2SRO considerando el mismo modelo base y luego se entrena un modelo <i>Stacking</i> para predecir valores de <i>trade-in</i> considerando las predicciones de los modelos 2SRO como atributos de entrada.

Tabla 9: Técnicas de modelamiento
Fuente: Elaboración propia

Estas técnicas de modelamiento toman como inspiración la forma en la que el equipo de ventas de Seez lleva a cabo el proceso actual descrito en la figura 1. La estrategia 2SPO se divide en dos fases: en la primera, se realiza un estudio de mercado para obtener una primera aproximación del precio de lista del automóvil; en la segunda, un experto ajusta esta primera estimación basándose en su conocimiento especializado y en la información proporcionada por el usuario. En contraste, la estrategia 3SPO convoca a un comité de expertos que, inicialmente, llevan a cabo un estudio de mercado de manera conjunta y, posteriormente, ajustan individualmente el precio de lista hacia un valor de *trade-in*.

Adicionalmente, se proponen las técnicas 2SRO y 3SRO, que son extensiones de las anteriores y que incorporan un proceso de ajuste más sofisticado. En estas, el experto realiza un estudio de mercado más profundo para adquirir **conocimiento** detallado sobre los precios de lista actuales del automóvil, y luego ajusta el precio de lista hacia un valor de *trade-in* teniendo en cuenta este conocimiento adquirido.

3.5.1. 2SPO

La técnica 2SPO consta de dos etapas y constituye el componente fundamental de esta investigación. En primera instancia, el modelo base realiza una estimación del precio de lista promedio de un automóvil teniendo en cuenta atributos de entrada generales. Luego, el modelo de ajuste busca, valga la redundancia, ajustar la predicción del precio de lista hacia un valor de *trade-in* considerando atributos de mayor resolución sobre el automóvil.

Intuitivamente, el modelo de ajuste debería considerar la marca y el modelo del automóvil como atributos de entrada. De esta manera, la primera etapa consistiría en obtener el precio de lista del automóvil, mientras que la segunda sería ajustar este precio de lista hacia un valor de *trade-in*, teniendo en cuenta, entre otros atributos, la marca y el modelo del automóvil. Sin embargo, es importante destacar que el modelo de ajuste **no considera la marca ni el modelo del automóvil como atributos de entrada**. La razón, en principio, es de índole práctica: la inclusión de estos atributos generaría una **alta dimensionalidad** en el espacio de atributos. Dado el número limitado de datos disponibles para entrenar el modelo de ajuste (869 registros), esto podría resultar en un sobreajuste del modelo. Además, en los 869 registros, la cantidad de marcas y modelos únicos presentes es muy baja. En consecuencia, en la práctica, el modelo de ajuste no sería capaz de realizar predicciones cuando se le presenten marcas y modelos que no hayan sido observados durante el entrenamiento. Además, y más importante aún, se considera como hipótesis el poder ajustar un precio de lista a un valor de *trade-in* independientemente de la marca y el modelo del automóvil considerando atributos de mayor resolución sobre el automóvil.

En resumen, el modelo de ajuste realiza una estimación del valor de *trade-in* de un automóvil considerando diversos aspectos, como el motor, la carrocería, el impacto ambiental y el estado actual del automóvil además de la estimación del precio de lista realizada por el modelo base.

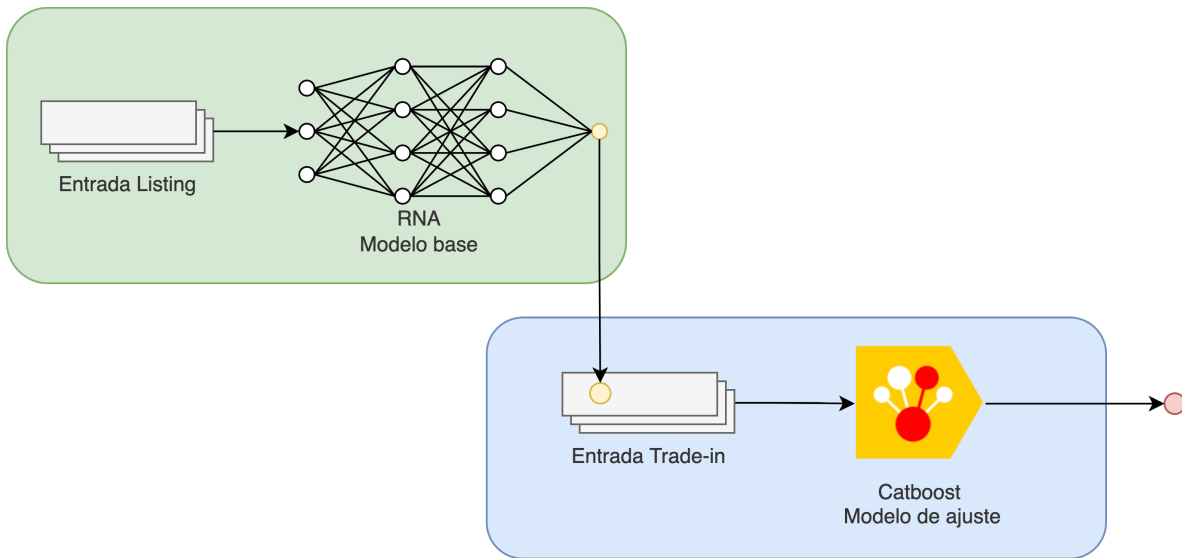


Figura 15: Modelamiento 2SPO
Fuente: Elaboración propia

3.5.2. 2SRO

La técnica 2SRO, al igual que el enfoque de 2SPO, se compone de dos etapas. En la primera etapa, el modelo base, que en este caso es una red neuronal artificial, realiza un *forward pass* para obtener las representaciones latentes aprendidas que capturan patrones útiles para la tarea de predicción de precios de lista. Luego, el modelo de ajuste utiliza estas representaciones aprendidas como parte de los atributos de entrada para predecir un valor de *trade-in*.

La principal diferencia entre esta técnica de modelamiento y 2SPO yace en **cómo se transfiere el conocimiento** aprendido en la tarea de predicción de precios de lista hacia la tarea de predicción de valores de *trade-in*. En 2SPO, el conocimiento se transfiere de manera explícita considerando la predicción del modelo base como un atributo de entrada del modelo de ajuste. En la técnica 2SRO, el conocimiento se transfiere de manera implícita considerando las representaciones latentes aprendidas por el modelo base como atributos de entrada del modelo de ajuste.

Es importante destacar que esta no es la clásica aplicación de transferencia de aprendizaje en la que se elimina la última capa de una red neuronal artificial, se congelan sus pesos y se realiza un *fine-tuning* de la red. En este caso, se utiliza la red neuronal artificial como un modelo base para obtener representaciones latentes, las cuales se utilizan posteriormente como atributos de entrada en un modelo de ajuste. La razón que justifica esta decisión es nuevamente de carácter práctico: el conjunto de datos *Listing* tiene muchos menos atributos que el conjunto de datos *Trade-in*, por ende, el *fine-tuning* de la red neuronal de ajuste no

sería posible a menos que se cumplan las siguientes dos condiciones:

1. No se consideren la mayoría de los atributos del conjunto de datos *Trade-in* para forzar un *match* con los atributos de entrada de la red neuronal artificial entrenada en el conjunto de datos *Listing*.
2. Se agreguen los atributos de entrada **marca** y **modelo**.

La primera condición implicaría una importante pérdida de información, por ejemplo, no se considerarían los atributos de entrada que permiten caracterizar precisamente la variante de un automóvil, pues, a pesar de que el conjunto de datos *Listing* cuenta con el atributo **modelo**, este atributo no es lo suficientemente descriptivo como para caracterizar la variante de un automóvil como lo son los atributos presentes en el conjunto de datos *Trade-in* asociados al motor y la carrocería del automóvil. La segunda condición, por otro lado, implicaría una alta dimensionalidad en el espacio de atributos para el conjunto de datos *Trade-in*, lo cual podría resultar en un sobreajuste del modelo de ajuste debido a la limitada cantidad de datos disponibles para entrenar el modelo de ajuste.

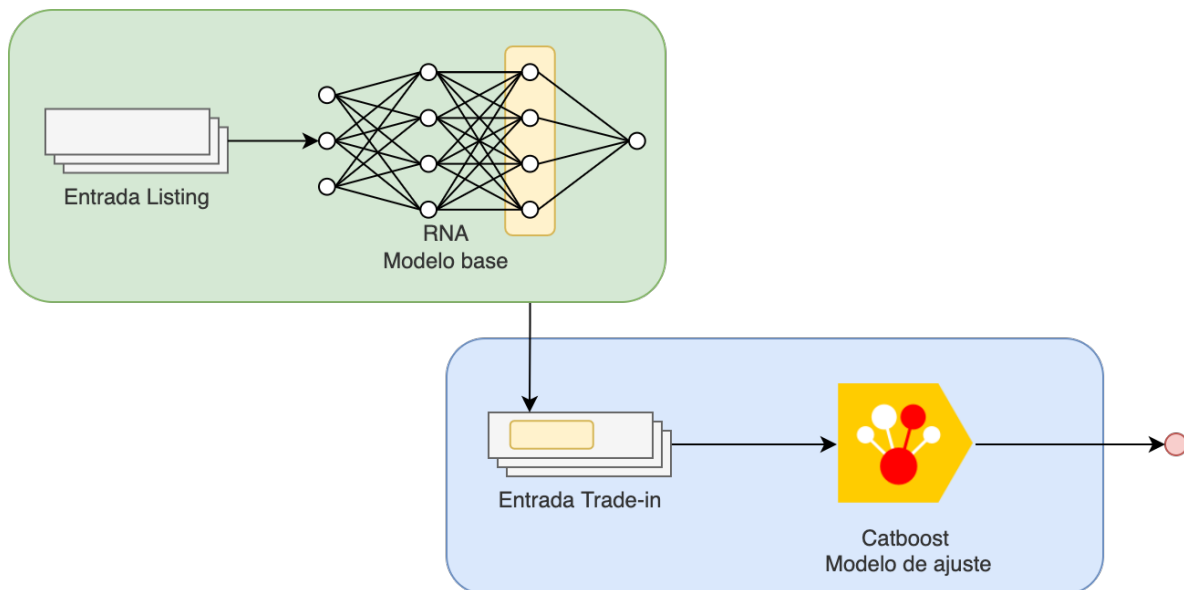


Figura 16: Modelamiento 2SRO
Fuente: Elaboración propia

3.5.3. 3SPO

La técnica 3SPO agrega una etapa adicional al proceso de modelamiento al considerar un modelo *meta* que utiliza las predicciones de un comité de modelos 2SPO como atributos

de entrada. El objetivo de esta técnica es mejorar la predicción de valores de *trade-in* al considerar las predicciones de distintos modelos *2SPO* como atributos de entrada.

Esta técnica de modelamiento busca aprovechar la **diversidad** del comité de modelos *2SPO* para mejorar la predicción de los valores de *trade-in*. Este punto es clave. Si bien los distintos modelos *2SPO* comparten el mismo modelo de primera etapa, **los modelos de segunda etapa son distintos**. Por ende, cada modelo de segunda etapa aprende distintos patrones a partir de los atributos de entrada. En consecuencia, cada modelo de segunda etapa realiza una predicción distinta.

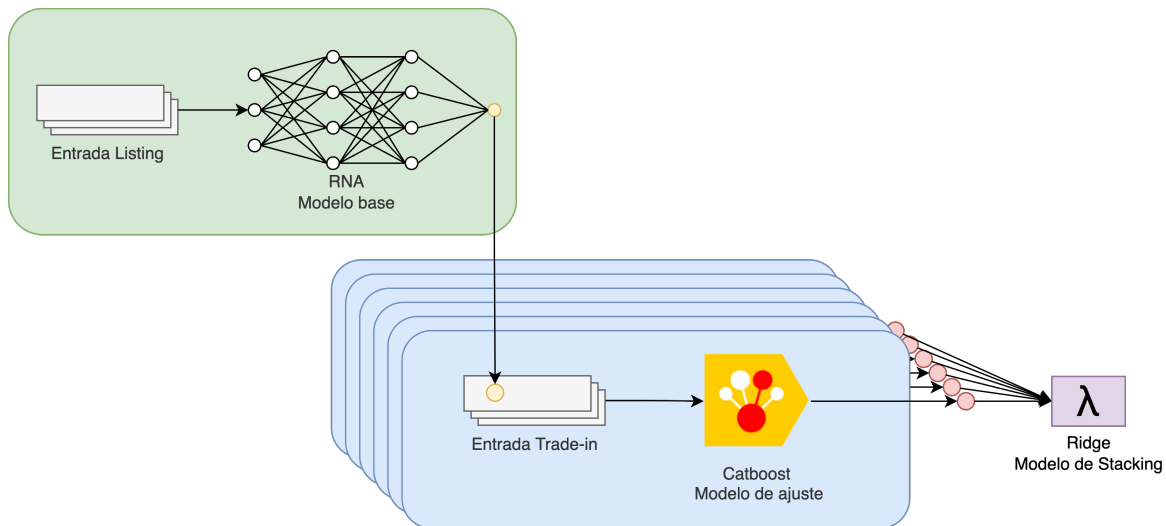


Figura 17: Modelamiento 3SPO
Fuente: Elaboración propia

3.5.4. 3SRO

La técnica *3SRO*, al igual que el enfoque *3SPO*, considera tres etapas en donde un modelo *meta* utiliza las predicciones de distintos modelos *2SRO* como atributos de entrada.

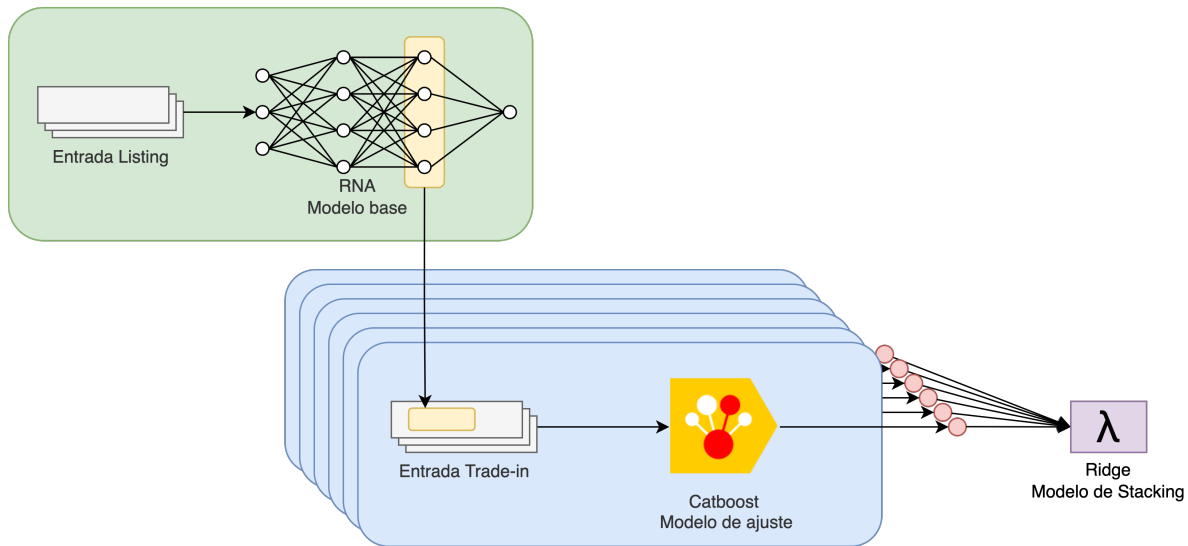


Figura 18: Modelamiento 3SRO
Fuente: Elaboración propia

3.6. Entrenamiento

3.6.1. Modelo primera etapa

Como se mencionó en el capítulo 3.5, el modelo base de las cuatro técnicas de modelamiento propuestas es una red neuronal artificial. Para todos los entrenamientos realizados en la primera etapa se consideró la siguiente configuración base de hiperparámetros fijos:

Hiperparámetro	Valor
Función de pérdida	Error absoluto medio
Optimizador	Adam
Taza de aprendizaje	0.001
Decaimiento de pesos	0.0001
Función de activación	ReLU
Tamaño de lote	512

Tabla 10: Hiperparámetros primera etapa

La adopción de Adam como optimizador se basa en su eficacia comprobada en la optimización de redes neuronales. Adam incorpora estrategias de optimización estocástica y adaptación de tasas de aprendizaje, lo que lo hace apto para converger de manera eficiente hacia soluciones óptimas. Además, la elección de una tasa de aprendizaje de 0.001 se ajusta a valores típicamente efectivos en problemas de regresión y clasificación, equilibrando una convergencia ágil con la estabilidad del proceso de entrenamiento. El decaimiento de pesos,

establecido en 0.0001, se utiliza con el propósito de regularizar el modelo y prevenir el sobreajuste al limitar el crecimiento excesivo de los coeficientes de peso. Por último, la función de activación ReLU se elige para las capas ocultas debido a su capacidad probada para introducir no linealidad en el modelo, permitiendo la captura de relaciones complejas en los datos.

El valor del tamaño de lote puede llamar la atención pues, en general, se considera un tamaño de lote 64. Sin embargo, en este caso, se consideró un tamaño de lote más grande debido a la gran cantidad de datos disponibles para entrenar el modelo y a la limitada capacidad de cómputo disponible. La elección de un tamaño de lote de 512 se basó en encontrar un equilibrio práctico entre las limitaciones de recursos computacionales y la necesidad de obtener gradientes estables durante el entrenamiento. Es relevante destacar que optar por tamaños de lote más pequeños habría incrementado significativamente el tiempo de entrenamiento debido a las operaciones de lectura principalmente.

A partir de esta configuración estándar, se realizaron distintos experimentos variando principalmente el número de capas y el número de neuronas por capa de la red neuronal. En la tabla 11 se detallan las distintas arquitecturas consideradas:

Modelo	Parámetros	Arquitectura
<i>Tiny</i>	0.7M	Lineal + ReLU: 1024 x 512 x 256 x 8 x 1
<i>Base</i>	1.4M	Lineal + ReLU: 1024 x 512 x 256 x 128 x 64 x 32 x 16 x 8 x 1

Tabla 11: Arquitecturas primera etapa
Fuente: Elaboración propia

En la figura 19a se muestra la evolución de la pérdida promedio por época evaluada en el conjunto de entrenamiento para cada arquitectura. Como se puede observar, la arquitectura *Base* destaca por su mejor desempeño durante el proceso de entrenamiento, lo que puede atribuirse a su capacidad superior para comprender y representar las relaciones complejas presentes en los datos.

Análogamente, en la figura 19b se representa la evolución de la pérdida promedio por época en el conjunto de validación para cada una de las arquitecturas. Como era de anticipar, la arquitectura *Base* exhibe un rendimiento superior durante el proceso de validación. Es importante destacar que, al examinar esta visualización, no se detecta ningún indicio de sobreajuste en ninguna de las arquitecturas que se han considerado.

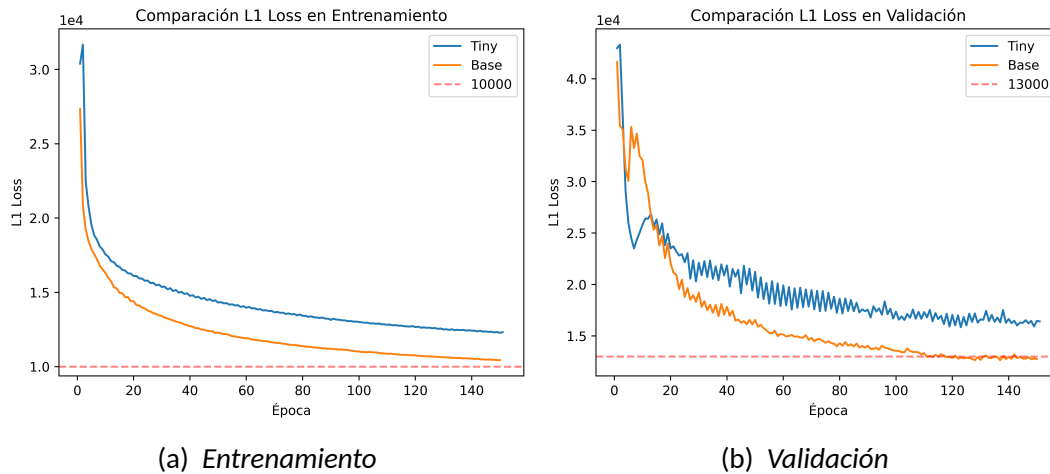


Figura 19: Primera etapa: Evolución de la pérdida por época
Fuente: Elaboración propia

3.6.2. Modelo segunda etapa

Como se menciona en el capítulo 3.5, el modelo de ajuste empleado para las técnicas 2SPO y 2SRO es el algoritmo *Catboost*. Para todos los procesos de entrenamiento, se utilizó el error absoluto medio como función de pérdida y se aplicó una estrategia de detención temprana, conocida como *early stopping* en inglés, con un número fijo de rondas. Posteriormente, se llevaron a cabo ajustes en los hiperparámetros más relevantes del modelo mediante una búsqueda aleatoria. La tabla 12 proporciona detalles sobre los hiperparámetros considerados:

Hiperparámetro	Valor
Número máximo de árboles	[300, 400, 500, 600, 800]
Profundidad de los árboles	[3,4,5,6,7]
Taza de aprendizaje	[0.01, 0.05, 0.1, 0.2, 0.3]
Regularización L2 en las hojas	[1, 2, 3]

Tabla 12: Hiperparámetros segunda etapa
Fuente: Elaboración propia

Se optó por emplear una estrategia de validación cruzada con 5 particiones con el propósito de exhaustivamente evaluar el rendimiento de cada posible combinación de hiperparámetros. Esta técnica de validación cruzada garantiza una evaluación rigurosa y robusta de los modelos, al dividir el conjunto de datos de entrenamiento en cinco partes distintas y, en cada iteración, entrenar el modelo en cuatro de ellas y evaluar el modelo en la parte restante, repitiendo este proceso de forma iterativa para obtener resultados confiables y generalizables.

Luego de encontrar los hiperparámetros óptimos a través de la búsqueda aleatoria de hiperparámetros para cada modelo *Tiny + 2SPO*, *Tiny + 2SRO*, *Base + 2SPO* y *Base + 2SRO*, se entrenaron los modelos con estos hiperparámetros fijos utilizando el conjunto de datos de entrenamiento completo y el conjunto de datos de validación.

En la figura 20a se muestra la evolución de la pérdida promedio por iteración evaluada en el conjunto de entrenamiento para cada modelo. Como se puede observar, el modelo *Tiny + 2SRO* destaca por su mejor desempeño durante el entrenamiento.

Análogamente, en la figura 20b se representa la evolución de la pérdida promedio por iteración en el conjunto de validación para cada modelo. A pesar de que el modelo *Tiny + 2SRO* muestra un considerable mejor desempeño respecto al modelo *Base 2SRO* en el conjunto de entrenamiento, en el conjunto de validación estos modelos exhiben un rendimiento similar.

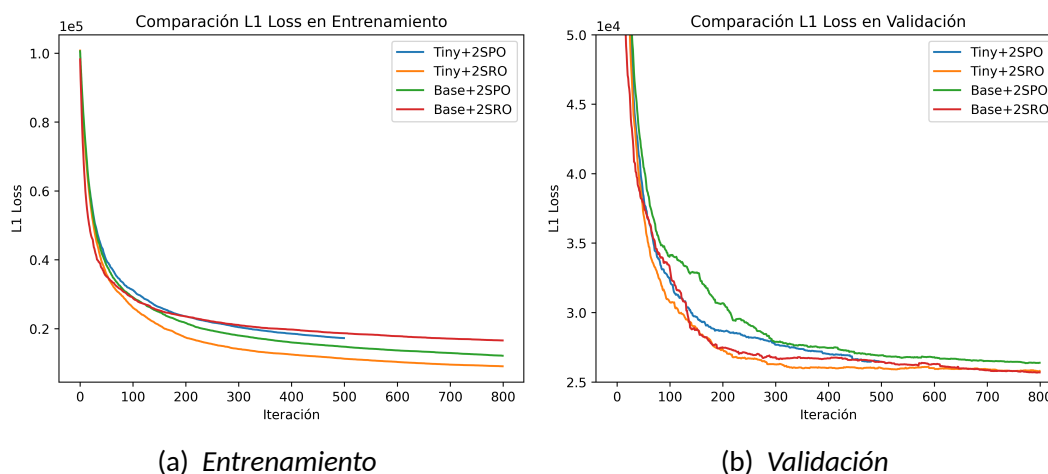


Figura 20: Segunda etapa: Evolución de la pérdida por época
Fuente: Elaboración propia

Como se mencionó en la sección 2.6, en cada iteración n de entrenamiento para un modelo *Catboost* se obtiene el modelo h_n y se agrega al comité de modelos $F_n = F_{n-1} + \alpha h_n$ para posteriormente medir su desempeño en el conjunto de validación.

Si bien para las técnicas *2SPO* y *2SRO* se consideró el algoritmo *Catboost* como el único modelo de ajuste, en las técnicas *3SPO* y *3SRO* se utilizaron distintos modelos de ajuste con sus hiperparámetros por defecto para construir el comité de modelos *2SPO* y *2SRO*, respectivamente.

3.6.3. Modelo tercera etapa

Como se menciona en el capítulo 3.5, el modelo *meta* empleado para las técnicas *3SPO* y *3SRO* es una regresión *Ridge*. La tabla 13 proporciona detalles sobre los hiperparámetros

considerados:

Hiperparámetro	Valor
Regularización L2	[0.1, 1, 10]

Tabla 13: Hiperparámetros tercera etapa
Fuente: Elaboración propia

Nuevamente, se optó por considerar una estrategia de validación cruzada con 5 particiones con el propósito de exhaustivamente evaluar el rendimiento de cada posible combinación de hiperparámetros. Si bien no se puede obtener una gráfica de visualización que muestre cómo progresa la pérdida a través de las épocas, en las tablas 14 y en la tabla 15 se muestra como ejemplo los resultados de la validación cruzada de las técnicas 3SPO y 3SRO, respectivamente, utilizando el modelo base *Tiny*.

Modelo	MSE Promedio (Negativo)	Desviación Estándar del MSE
CatBoost	$-2,87 \times 10^9$	$2,84 \times 10^8$
XGBoost	$-3,13 \times 10^9$	$4,56 \times 10^8$
Extra Trees	$-3,01 \times 10^9$	$3,80 \times 10^8$
k-NN	$-6,97 \times 10^9$	$1,23 \times 10^9$

Tabla 14: Resultados validación cruzada 3SPO
Fuente: Elaboración propia

Modelo	MSE Promedio (Negativo)	Desviación Estándar del MSE
CatBoost	$-2,95 \times 10^9$	$4,05 \times 10^8$
XGBoost	$-3,28 \times 10^9$	$3,18 \times 10^8$
Extra Trees	$-2,99 \times 10^9$	$3,47 \times 10^8$
k-NN	$-6,72 \times 10^9$	$1,27 \times 10^9$

Tabla 15: Resultados validación cruzada 3SRO
Fuente: Elaboración propia

Capítulo 4

Validación de la solución

En este capítulo se describe el proceso de validación de la solución propuesta. Para ello, se describen los indicadores clave de desempeño provenientes del entendimiento del negocio y el proceso de evaluación de los distintos modelos de aprendizaje automático propuestos.

4.1. Indicadores clave de desempeño

Los indicadores clave de desempeño provenientes del entendimiento del negocio para el sistema de Aprendizaje Automático propuesto son:

1. Error porcentual absoluto medio (MAPE): Debido a que el rango de precios de los vehículos es muy amplio al considerar un número elevado de marcas y modelos, una métrica que no sea porcentual no es adecuada para medir el error de predicción pues no permite comparar el desempeño de los modelos en distintos rangos de precios. Por ejemplo, un error absoluto de 500,000 CLP en un vehículo de 5,000,000 CLP es mucho más significativo que un error absoluto de 500,000 CLP en un vehículo de 50,000,000 CLP. Por lo tanto, se utiliza el error porcentual absoluto medio (MAPE) para medir el error de predicción de los modelos.
2. Cobertura: La cobertura es la proporción de vehículos para la cual se logra predecir un valor de *trade-in* en producción. La cobertura es un indicador clave de desempeño pues permite medir el impacto de la solución propuesta en el negocio. Si la cobertura fuera baja, la solución propuesta no tendría un impacto significativo en el negocio a pesar de que el error de predicción sea bajo pues no se lograría predecir un valor de *trade-in* para la mayoría de los vehículos que se reciben en producción. Es importante mencionar que la cobertura del sistema de Aprendizaje Automático propuesto no depende de los modelos de aprendizaje automático sino de los datos de entrada y de los procesos de preprocesamiento.

4.2. Conjunto de prueba

Tal como se destacó en la sección 3.4, el conjunto de prueba de *Trade-in* se compone de 87 instancias únicas. Este conjunto no es solo notable por su tamaño, sino también por su diversidad: abarca vehículos de 20 marcas distintas y más de 60 modelos diferentes. Esta diversidad no es un detalle menor; de hecho, juega un papel crucial en la evaluación del rendimiento del modelo. Al incluir una amplia gama de marcas y modelos, el conjunto de prueba

asegura que el modelo se enfrente a una variedad de escenarios. Esto no solo permite una mejor evaluación de la capacidad de generalización del modelo, sino que también minimiza el riesgo de sobreajuste a un subconjunto específico de datos. En resumen, la diversidad en el conjunto de prueba hace que este sea más representativo de las condiciones del mundo real, fortaleciendo así la confiabilidad y aplicabilidad del modelo en situaciones prácticas.

La figura 21 muestra la distribución de los precios de *trade-in* en el conjunto de prueba. Como se puede observar, la distribución de los precios de *trade-in* no está sesgada a valores altos o bajos.

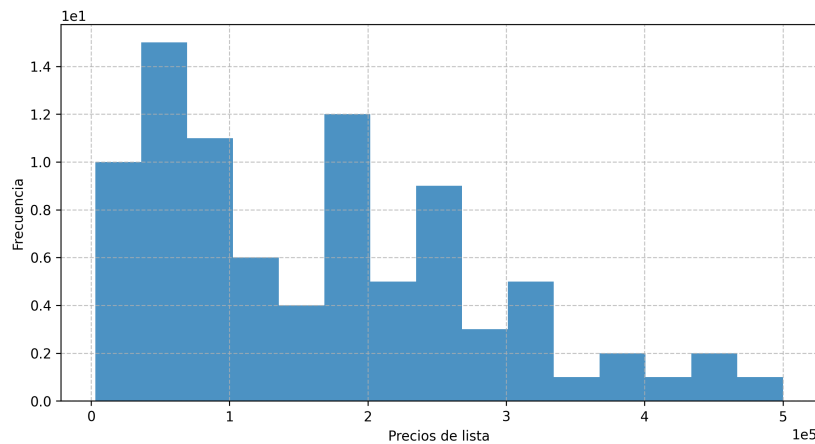


Figura 21: Distribución del precio en el conjunto de prueba de *Trade-in*
Fuente: Elaboración propia

4.3. Proceso de evaluación

Debido a las técnicas de modelamiento propuestas, el proceso de evaluación consta de dos etapas. Primero, se evalúa la capacidad de generalización de los modelos propuestos para el modelo base, es decir, el modelo que predice precios de lista de vehículos. Luego, se evalúa la capacidad de generalización de las técnicas de modelamiento *2SPO*, *2SRO*, *3SPO* y *3SRO* para la predicción de valores de *trade-in* de vehículos.

En la tabla 16 se presentan las métricas de evaluación del modelo base, las cuales han sido calculadas en el conjunto de prueba del conjunto de datos *Listing* mencionado en la sección 3.4.

Modelo	MAE [DKK]	MDAE [DKK]	MAPE	MDAPE
<i>Tiny</i>	16,414.42	10,684.94	0.11	0.072
<i>Base</i>	12,706.73	7,506.21	0.083	0.051

Tabla 16: Modelo base: Métricas de evaluación
Fuente: Elaboración propia

En la tabla 17 se presentan las métricas de evaluación del modelo de ajuste, las cuales han sido calculadas en el conjunto de prueba del conjunto de datos *Trade-in* mencionado en la sección 3.4. Se ha considerado como **baseline** un modelo *Catboost* con sus hiperparámetros por defecto y entrenado **sin la información del modelo base**.

Modelo	MAE [DKK]	MDAE [DKK]	MAPE	MDAPE
Baseline	25,622.67	15,801.06	0.40	0.17
<i>Tiny</i> + 2SPO	25,859.14	16,641.42	0.31	0.16
<i>Tiny</i> + 3SPO	21,154.62	13,126.50	0.31	0.12
<i>Tiny</i> + 2SRO	25,544.97	19,175.67	0.39	0.17
<i>Tiny</i> + 3SRO	19,120.87	12,361.17	0.29	0.10
Base + 2SPO	25,434.13	22,085.87	0.30	0.15
Base + 3SPO	23,712.78	17,902.40	0.28	0.16
Base + 2SRO	26,754.81	19,914.57	0.28	0.16
Base + 3SRO	23,990.76	19,565.48	0.33	0.16

Tabla 17: Modelo de ajuste: Métricas de evaluación
Fuente: Elaboración propia

En cada una de las tablas y por cada columna asociada a una métrica de evaluación, se resalta en color verde la casilla i, j asociada al modelo en la fila i que obtiene la mejor métrica de evaluación en la columna j . De manera análoga, se resalta en color rojo la casilla i, j asociada al modelo en la fila i que obtiene la peor métrica de evaluación en la columna j .

Como se puede observar en la tabla 17, los mejores resultados para 3 de las 4 métricas están asociados a el modelo *Tiny* + 3SRO.

4.4. Validación hipótesis central

A pesar de los datos mostrados en la tabla 17, es imperativo analizar críticamente la metodología adoptada en esta investigación. Las estrategias de modelado propuestas, 2SPO, 2SRO, 3SPO y 3SRO, se fundamentan en la hipótesis de que **el precio de lista de un vehículo es esencial para estimar su valor de *trade-in***. Dicha hipótesis, no obstante, aún no ha sido corroborada.

Cabe la duda de que los modelos de la segunda etapa estén determinando el valor de *trade-in* sin dar el debido peso al precio de lista predicho en la primera etapa, lo que sugeriría que la aplicación de técnicas de modelado en dos y tres etapas no era necesaria desde el principio. Esto nos lleva a cuestionar la relevancia real de las predicciones del precio de lista del modelo base para estimar el valor de *trade-in* de un vehículo.

Para abordar esta cuestión, se puede considerar la correlación entre las predicciones de pre-

cios de lista generadas por el modelo base y los valores de *trade-in* presentes en el conjunto de datos *Trade-in*. Este análisis se visualiza en la figura 22, donde se muestra un gráfico de dispersión que relaciona las predicciones del modelo base *Tiny* con los valores de *trade-in* del mencionado conjunto de datos.

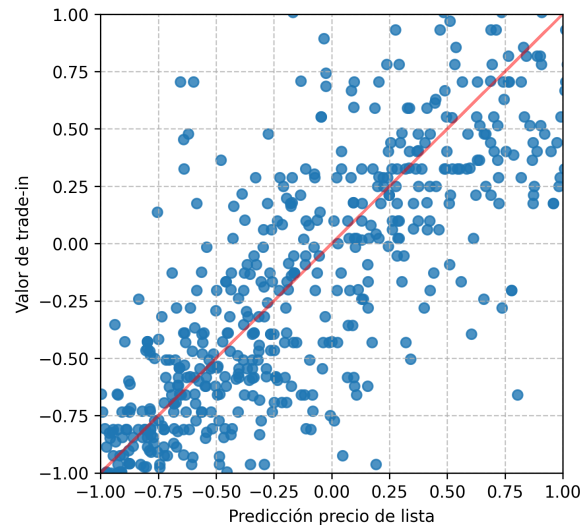


Figura 22: Correlación predicción de precios de lista y valores de *trade-in*

Fuente: Elaboración propia

Adicionalmente, al determinar el coeficiente de correlación de *Pearson*, se registra un valor de 0.79, lo que indica una correlación positiva y significativa. Esto permite inferir que las predicciones de precios de lista realizadas por el modelo base *Tiny* desempeñan un papel informativo relevante en la predicción del valor de *trade-in* de un vehículo, validando así la utilidad de estas predicciones en el proceso de modelado.

4.5. Explicaciones con XAI

En la tabla 18 se presentan 3 instancias pertenecientes al conjunto de datos de prueba *Trade-in*. Para cada instancia, solo se considera un subconjunto de los atributos disponibles en el conjunto de datos *Trade-in* para una mejor visualización.

Marca	Modelo	Año	Kilometraje	Transmisión	Valor de trade-in [DKK]
Peugeot	208	2016	120,000	Automática	62,000
Mitsubishi	Outlander	2019	37,000	Automática	235,000
Hyundai	i20	2017	52,000	Automática	70,500

Tabla 18: Instancias de prueba

Fuente: Elaboración propia

Para las explicaciones utilizaremos los valores *SHAP* y el modelo *Tiny + 3SPO* pues es el modelo **orientado a predicciones** que obtiene el mejor desempeño considerando las métricas de evaluación presentadas en la tabla 17. La elección de un modelo orientado a predicciones se debe a que es más simple de explicar en la segunda etapa.

Como se mencionó en la sección 3.5, el modelo de *stacking* es una regresión Ridge. Debido a lo simple de este modelo, sus coeficientes se pueden interpretar como la importancia relativa de cada modelo base en la predicción final. En la tabla 19 se presentan los coeficientes del modelo de *stacking*.

Modelos base	Coeficientes
Catboost	0.6072
XGB	0.3317
Extra Trees	0.1052
Knn	0.0004

Tabla 19: Coeficientes modelo de *Stacking*

Dado que el modelo base con mayor coeficiente es el modelo *Catboost*, se utilizará este para explicar las predicciones de la segunda etapa pues su predicción es la que define con mayor importancia la predicción final del modelo 3SPO.

En la figura 23 se explican las predicciones de primera y segunda etapa para la instancia de prueba asociada a el automóvil Peugeot 209 de la tabla 18. La predicción del modelo *Tiny* para el precio de lista es de 99,000 DKK aproximadamente. Dentro de los atributos más importantes para esta predicción se encuentra la marca, la transmisión y la cilindrada del motor. Además, es importante destacar cómo los atributos booleanos asociados a otras marcas impactan fuertemente en el proceso de decisión, como por ejemplo, el valor 0 para el atributo *make_alfa_romeo* disminuye la predicción del precio de lista en 15,000 DKK aproximadamente. La predicción del modelo *Catboost* para el valor de *trade-in* es de 62,770 DKK. El atributo más importante por lejos es la predicción del precio de lista realizada por el modelo *Tiny*, seguido por la potencia del motor, el peso máximo de carga, la normativa europea de emisión y las emisiones de dióxido de carbono. El error porcentual para esta predicción es de 1.2 %.



Figura 23: Explicación: Peugeot 208
Fuente: Elaboración propia

En la figura 24 se explican las predicciones de primera y segunda etapa para la instancia de prueba asociada a el automóvil Mitsubishi Outlander de la tabla 18. La predicción del modelo *Tiny* para el precio de lista es de 253,576 DKK aproximadamente. Dentro de los atributos más importantes para esta predicción se encuentra el año del automóvil, seguido por el modelo y el tipo de combustible. La predicción del modelo *Catboost* para el valor de *trade-in* es de 249,107 DKK. El atributo más importante es el año del automóvil, seguido por la predicción del precio de lista realizada por el modelo *Tiny*, el kilometraje, el peso máximo de carga y las emisiones de dióxido de carbono. El error porcentual para esta predicción es de 6.4 %.



Figura 24: Explicación: Mitsubishi Outlander
Fuente: Elaboración propia

En la figura 25 se explican las predicciones de primera y segunda etapa para la instancia de prueba asociada a el automóvil Hyundai i20 de la tabla 18. La predicción del modelo *Tiny* para el precio de lista es de 138,000 DKK aproximadamente. Dentro de los atributos más importantes para esta predicción se encuentra el año del automóvil seguido por la marca y el modelo del mismo. La predicción del modelo *Catboost* para el valor de *trade-in* es de 81,200 DKK aproximadamente. El atributo más importante es la predicción del precio de lista realizada por el modelo *Tiny*, seguido por la potencia del motor, el peso máximo de carga, el kilometraje y el año del automóvil. El error porcentual para esta predicción es de 15.2 %.

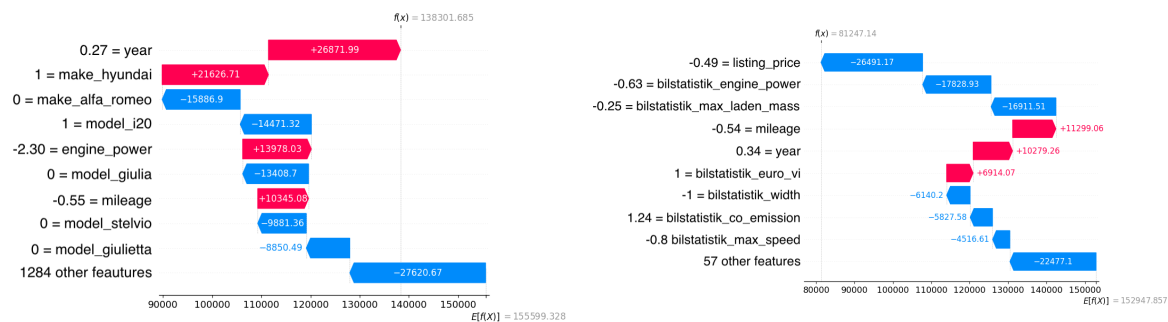


Figura 25: Explicación: Hyundai i20
Fuente: Elaboración propia

En la figura 26 se presentan los valores *SHAP* promedio asociados a las predicciones realizadas en todo el conjunto de prueba. De esta forma, y para el conjunto de prueba elegido aleatoriamente, se confirma la hipótesis de que el precio de lista es esencial para estimar el valor de *trade-in* de un vehículo, considerando como precio de lista las predicciones del modelo base *Tiny*.

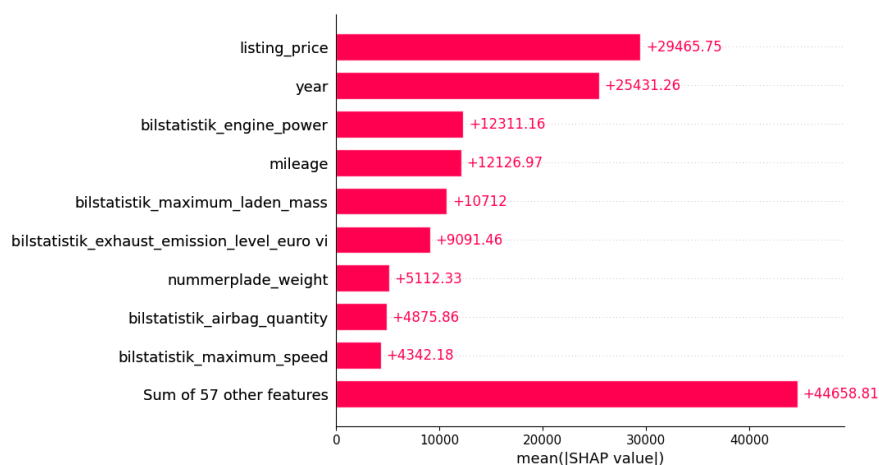


Figura 26: Valores *SHAP*
Fuente: Elaboración propia

Finalmente, en la figura 27 se presenta cómo cada atributo aporta en la predicción del valor de *trade-in*. El eje x corresponde a las predicciones del modelo y el eje y a los atributos. Los atributos están ordenados por la importancia de manera descendente calculada considerando todo el conjunto de prueba del conjunto de datos *Trade-in*. Cada predicción está representada por una línea que en la cima alcanza su valor. Interpretando el gráfico desde abajo hacia arriba, los valores *SHAP* de cada atributo son agregados a la predicción base del modelo.

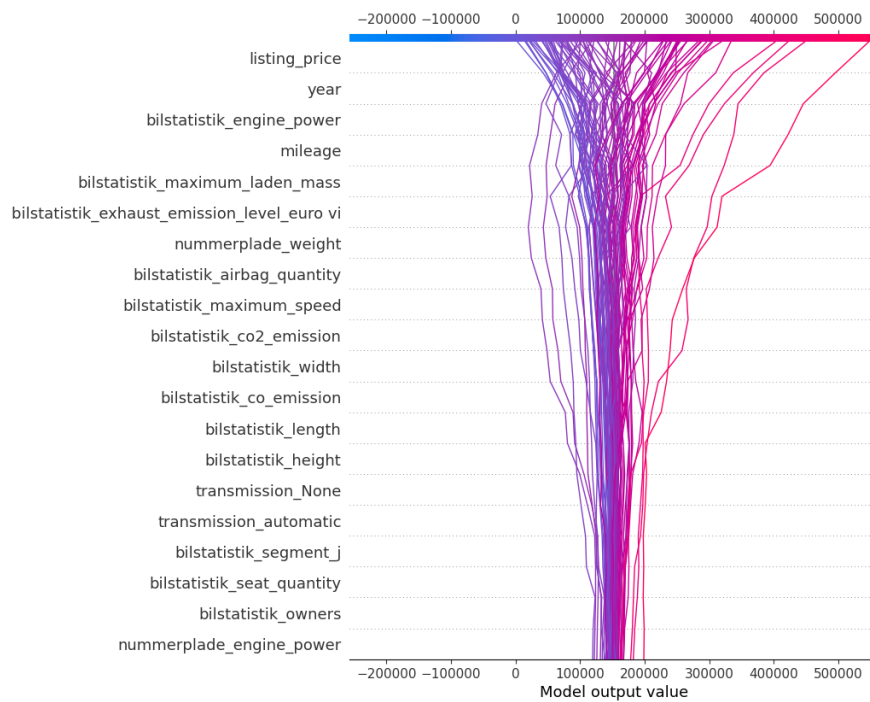


Figura 27: Proceso de decisión explicado con valores SHAP
Fuente: Elaboración propia

Capítulo 5

Conclusiones

En el transcurso de esta investigación, abordamos el desafío de predecir valores de *trade-in* de automóviles en el contexto de Seez mediante el análisis de diversas técnicas de modelamiento. Inspiradas en las prácticas manuales del equipo de ventas de Seez, estas técnicas, tras un riguroso análisis, evidenciaron un desempeño superior al modelo **baseline**, siendo especialmente notable la técnica 3SRO con un error absoluto porcentual promedio de 0.29 y un error absoluto porcentual mediano de 0.10. En un esfuerzo complementario, se brindaron interpretaciones a través de SHAP para las predicciones de precios de lista y valores de *trade-in* de diferentes automóviles. Estos resultados reafirmaron la capacidad de los modelos, tanto en la primera como en la segunda etapa, para asimilar reglas de negocio y ponderar atributos conforme a los fundamentos esenciales de la industria automovilística.

El alcance y las limitaciones de la propuesta de solución surgen como consecuencia de las fuentes de datos consideradas y de los entrenamientos *off-line* de los modelos propuestos. En primer lugar, el alcance de la solución planteada para la predicción de valores de *trade-in* se limita al mercado automotriz Danés. Además, si bien los datos obtenidos desde el sitio web Ooyyo son representativos para la tarea de predecir precios de lista en el mercado automotriz Danés, los datos obtenidos desde SeezPAD para la tarea de predecir valores de *trade-in* son generados por el equipo de ventas de Seez y, por lo tanto, contienen un sesgo inherente. En segundo lugar, los modelos son entrenados de forma *off-line* y, por ende, no captan la dinámica del mercado automotriz en tiempo real. En consecuencia, los modelos pueden presentar un desfase entre los datos de entrenamiento y los datos de prueba por lo que es importante monitorear el desempeño de los modelos en producción y actualizarlos periódicamente.

Como trabajo futuro, una primera línea de acción es investigar diferentes métodos para codificar atributos de alta dimensionalidad, como la marca y el modelo, evitando la codificación *one-hot*. Este punto se puede abordar desde el conocimiento del dominio y la ingeniería de atributos, como así también, desde la implementación de PCA, *Autoencoders* o los ya mencionados estadísticos de salida en la sección 2.6. Como segunda línea de acción se propone adoptar aprendizaje *online* para actualizar continuamente ambos modelos, permitiendo así reflejar la dinámica en tiempo real del mercado automotriz y minimizar discrepancias entre los datos de entrenamiento y prueba. Como tercera línea de acción, se propone incorporar los datos no etiquetados provenientes de SeezPAD utilizando técnicas de aprendizaje semi-supervisado, buscando enriquecer la representación y mejorar la robustez de los modelos propuestos. Como cuarta y última línea de acción, se propone expandir el alcance de la solución a otros mercados automotrices, para lo cual axiomático contar con nuevos conjuntos de datos y, más importante aún, un algoritmo para identificar las marcas y modelos de automóviles en distintos formatos pues distintos concesionarios utilizan diferentes convenciones para nombrar a los automóviles.

Referencias

- [Audhkhasi *et al.*, 2013] Audhkhasi, K., Sethy, A., Ramabhadran, B., y Narayanan, S. S. (2013). Generalized ambiguity decomposition for understanding ensemble diversity.
- [Barredo Arrieta *et al.*, 2020] Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., y Herrera, F. (2020). Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115.
- [Chen *et al.*, 2018] Chen, S., Chou, E., y Yang, R. R. (2018). The price is right: Predicting prices with product images. *CoRR*, abs/1803.11227.
- [Collard, 2022] Collard, M. (2022). Price prediction for used cars: A comparison of machine learning regression models.
- [Goodfellow *et al.*, 2016] Goodfellow, I., Bengio, Y., y Courville, A. (2016). *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>.
- [Guidotti *et al.*, 2018] Guidotti, R., Monreale, A., Turini, F., Pedreschi, D., y Giannotti, F. (2018). A survey of methods for explaining black box models. *CoRR*, abs/1802.01933.
- [Holzinger, 2018] Holzinger, A. (2018). From machine learning to explainable ai. En *2018 World Symposium on Digital Intelligence for Systems and Machines (DISA)*, pp. 55–66.
- [Huyen, 2022] Huyen, C. (2022). *Designing Machine Learning Systems*. O'Reilly Media, USA.
- [Kelleher y Tierney, 2018] Kelleher, J. D. y Tierney, B. (2018). *Data science*. MIT Press.
- [Lessmann y Voß, 2017] Lessmann, S. y Voß, S. (2017). Car resale price forecasting: The impact of regression method, private information, and heterogeneity on forecast accuracy. *International Journal of Forecasting*, 33(4):864–877.
- [Liu *et al.*, 2022] Liu, E., Li, J., Zheng, A., Liu, H., y Jiang, T. (2022). Research on the prediction model of the used car price in view of the pso-gra-bp neural network. *Sustainability*, 14:8993.
- [Lundberg y Lee, 2017] Lundberg, S. M. y Lee, S.-I. (2017). A unified approach to interpreting model predictions. En Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., y Garnett, R., editores, *Advances in Neural Information Processing Systems* 30, pp. 4765–4774. Curran Associates, Inc.
- [Microsoft, 2021] Microsoft (2021). Team data science process. Accessed: 2023-04-24.
- [Mitchell, 1997] Mitchell, T. (1997). *Machine Learning*. McGraw-Hill Education.
- [Pan y Yang, 2010] Pan, S. J. y Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

- [Prokhorenkova *et al.*, 2019] Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., y Gulin, A. (2019). Catboost: unbiased boosting with categorical features.
- [Ribeiro *et al.*, 2016] Ribeiro, M. T., Singh, S., y Guestrin, C. (2016). "why should I trust you?": Explaining the predictions of any classifier. *CoRR*, abs/1602.04938.
- [Rumelhart *et al.*, 1986] Rumelhart, D. E., Hinton, G. E., y Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.
- [Ting y Witten, 2011] Ting, K. M. y Witten, I. H. (2011). Issues in stacked generalization. *CoRR*, abs/1105.5466.
- [Wirth y Hipp, 2000] Wirth, R. y Hipp, J. (2000). Crisp-dm: Towards a standard process model for data mining. En *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, volumen 1, pp. 29–39. Manchester.
- [Wolpert, 1992] Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, 5(2):241–259.
- [Zhuang *et al.*, 2020] Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., y He, Q. (2020). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76.