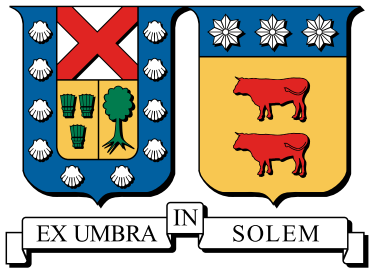
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



Vehicle Trade-in Appraisal via Multi-Source Heterogeneous Inductive Transfer Learning and RAG-Based Schema Normalization

Diego Alejandro Quezada Campos

Tesis para optar al Grado de Magíster en Ciencias de la Ingeniería Informática

Advisor:	Ricardo Ñanculef A., Ph.D.
Co-advisor:	Carlos Valle V., Ph.D.
Internal national member:	Mauricio Solar F., Ph.D.
External national member:	Marcelo Mendoza R., Ph.D.
Committee Chair:	Mauricio Solar F., Ph.D.

July 31, 2026

CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☐ Memoria o trabajo de título; ☒ Tesis de Postgrado;

Título del trabajo: Predicting Vehicle Trade-in Values by Integrating Knowledge from Multiple Heterogeneous Sources Via LLMs

Nombre del candidato(a): Diego Alejandro Quezada Campos

Carrera / Grado: Magíster en Ciencias de la Ingeniería Informática

Campus: Casa Central

Departamento: Departamento de Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Juan Ricardo Ñanculef Alegría, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo ☒**NO contiene información que amerite confidencialidad** y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

☐ 6 meses; ☐ 12 meses; ☐ 2 años; ☐ 3 años; ☐ 5 años; ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 15/06/2026 ; **Firma:** 

Estudiante o Candidato(a):

Fecha: 15/06/2026 ; **Firma:** 

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

To my mother and father, to whom I owe everything.

Acknowledgments

I would like to thank my advisor, Ricardo Ñanculef, and my co-advisor, Carlos Valle, for their guidance and the time they devoted to this work. Their questions consistently made it better. I am also grateful to the members of my committee, Mauricio Solar and Marcelo Mendoza, whose comments improved the final version of this thesis.

I thank Seez for providing the data and the industrial context that motivated this research, and my colleagues there for the many conversations that shaped how I came to understand the problem.

Finally, I thank my family and friends for their support and encouragement throughout these years. My deepest thanks go to Scarlett, my love, who supported me through every stage of this process. None of this would have been possible without them.

Resumen

El intercambio de vehículos es un proceso central de la industria automotriz: permite a los clientes reducir el costo de una nueva compra y, a la vez, ayuda a los concesionarios a gestionar su inventario. Aunque investigaciones previas han estudiado la predicción de precios de lista y de reventa mediante diversas técnicas de aprendizaje automático (ML), la estimación de los valores de intercambio apenas se ha explorado. Los valores de intercambio difieren de los precios de lista, ya que los propietarios suelen publicar sus vehículos a precios más altos, anticipando una negociación. Del mismo modo, difieren de los precios de reventa, pues los concesionarios deben tener en cuenta los costos de reacondicionamiento, las estrategias de inventario y los márgenes de ganancia. Se necesita un modelo especializado, pero los datos de intercambio etiquetados siguen siendo escasos. Un enfoque prometedor para abordar esta escasez es el aprendizaje por transferencia a partir de los abundantes datos de lista, pero la representación inconsistente de los vehículos en los distintos conjuntos de datos dificulta su aplicación. Para abordar estos desafíos, presentamos un novedoso sistema de ML para la tasación de vehículos de intercambio (VTA), integrado por dos componentes principales: un método de normalización de vehículos basado en generación aumentada por recuperación (RAG) y un método basado en *stacking* que transfiere predicciones desde múltiples dominios de lista al dominio de intercambio. Los experimentos muestran que, en VTA, nuestro enfoque supera tanto a los modelos de lista como a los modelos de intercambio. Experimentos adicionales sobre la diversidad de vehículos muestran que la práctica habitual de entrenar modelos de ML con conjuntos de datos de menos de 40 marcas o menos de 400 modelos de vehículos produce resultados demasiado optimistas que no generalizan en condiciones reales, donde la diversidad es mucho mayor. Nuestro método supera esta limitación al emplear 181 marcas y 1.940 modelos, lo que excede al trabajo previo más diverso en 100 marcas y 891 modelos, y alcanza un MdAPE de 14,14%.

Palabras clave: Tasación de Vehículos de Intercambio, Predicción de Precios de Vehículos, Ensambles, Aprendizaje por Transferencia, LLM, RAG.

Abstract

Vehicle trade-in is a central process in the automotive industry, enabling customers to offset the cost of a new purchase while helping dealerships manage their inventory. Although previous research has examined listing and resale price prediction using diverse machine learning (ML) techniques, the estimation of trade-in values remains largely unexplored. Trade-in values differ from listing prices as vehicle owners typically list their vehicles at higher prices, anticipating negotiation. Similarly, trade-in values differ from resale prices as dealerships must account for reconditioning costs, inventory strategies, and profit margins. A dedicated model is needed, but labeled trade-in data remains scarce. Transfer learning from abundant listing data is promising, but inconsistent vehicle representations across datasets complicate its application. To address these challenges, we introduce a novel ML system for vehicle trade-in appraisal (VTA) composed of two main components: a retrieval-augmented generation (RAG) based framework for vehicle normalization, and a stacking-based method that transfers predictions from multiple listing domains into the trade-in domain. Experiments show that our approach outperforms both listing models and trade-in models on VTA. Further experiments on vehicle diversity show that the common practice of training models on datasets with fewer than 40 brands or 400 models produces over-optimistic results that fail to generalize under real-world conditions, where diversity is far higher. Our method overcomes this limitation by using 181 brands and 1,940 models, exceeding the most diverse prior work by 100 brands and 891 models, and achieving a MdAPE of 14.14%.

Keywords: Vehicle Trade-in Appraisal, Vehicle Price Prediction, Ensembles, Transfer Learning, LLM, RAG.

Contents

Acknowledgments **II**

Resumen **III**

Abstract **IV**

Contents **VI**

List of Figures **VII**

List of Tables **VIII**

List of Algorithms **IX**

1 Introduction **1**
1.1 Hypotheses 6
1.2 Objectives 6
1.3 Outline 7

2 Background **8**
2.1 Ensemble methods 8
2.1.1 Bagging 8
2.1.2 Gradient boosting 9
2.1.3 Stacking 9
2.2 Large language models 10
2.3 Information extraction 10
2.4 Entity resolution 11
2.5 Information retrieval 11
2.6 Retrieval-augmented generation 12

3 Related work **13**
3.1 Vehicle price prediction 13
3.2 Vehicle trade-in appraisal 16
3.3 Information extraction 17
3.4 Entity resolution 17

3.5 Information retrieval 18

4 Method **19**
4.1 Problem formulation 19
4.2 Vehicle normalization 20
4.3 Vehicle trade-in appraisal 24
4.4 Computational complexity 28

5 Experiments **31**
5.1 Experimental setup 31
5.2 Datasets 33
5.3 Data preparation 40
5.4 Performance metric 42
5.5 Hyperparameter tuning 42
5.6 Results 43
5.7 Error analysis 45
5.8 Vehicle diversity 50
5.9 Discussion 51

6 Conclusions **56**
6.1 Limitations 56
6.2 Future work 57

References **59**

A Notation **65**

B Prompts **68**

C Scalability **73**

D Domain mismatch **75**

E Statistical significance **77**
E.1 Listing domain 77
E.2 Trade-in domain 78

F Software **81**

G Hardware **82**

List of Figures

1.1 Vehicle trade-in appraisal process at Seez. 2
1.2 Granularity levels of a vehicle model and a propulsion system. 5

4.1 Vehicle normalization pipeline. 21
4.2 Vehicle trade-in appraisal pipeline. 26
4.3 Inference pipeline. 28

5.1 Most popular brands by market. 35
5.2 Cumulative brand and model coverage. 36
5.3 Estimated price density. 37
5.4 Median price by vehicle age. 37
5.5 Correlation among numerical attributes. 38
5.6 Listing and trade-in price densities. 39
5.7 Most frequent variant, transmission, and fuel type values. 39
5.8 Effect of transfer learning in the trade-in domain. 46
5.9 Trade-in error by brand and by price band. 48
5.10 Effect of vehicle brand and model diversity. 52
5.11 Top 10 most important features for CatBoost+ (DK trade-in). 54
5.12 Most important features across trade-in and listing models. 55
5.13 Prediction bias for pricing and appraisal. 55

C.1 Computational efficiency over a 10,000-record stream. 74

List of Tables

1.1 Mercedes A-Class 200 representations. 3
1.2 AWD PHEV representations. 4
1.3 BMW 3 Series Touring representations. 4

3.1 Summary of selected studies. 16

4.1 Computational characteristics of the proposed algorithms. 30

5.1 Proposed schema. 32
5.2 Domain of closed-domain attributes. 32
5.3 Summary of datasets. 34
5.4 Missing data across datasets. 36
5.5 Normalized representation of a serialized vehicle. 41
5.6 Hyperparameter search space for the evaluated algorithms. 43
5.7 Performance of appraisal models in the trade-in domain. 44
5.8 Performance of pricing models in the listing domain. 45
5.9 Trade-in error by vehicle brand. 47
5.10 Effect of re-balancing the brand distribution. 51

A.1 Method notation. 65

C.1 Open-source datasets used for the scalability evaluation. 73

D.1 Performance of pricing models in the trade-in domain. 76

E.1 Statistical significance of normalization in the listing domain. 78
E.2 Statistical significance of the trade-in results. 80

F.1 Software environment used for all experiments. 81

G.1 Hardware configuration used for all experiments. 82

List of Algorithms

1 Vehicle Normalization 25
2 Transfer Learning 28

Chapter 1

Introduction

The global used-car market, valued at approximately USD 1.90 trillion in 2024, is projected to grow at a compound annual growth rate (CAGR) of 9.4% through 2027 (Hu et al., 2025). Several factors drive this growth: rising new vehicle prices, better financing options, and the spread of online platforms (Hu et al., 2025; Zacharof et al., 2025). These platforms make it easier to find cars and to consult information that builds buyer confidence, such as reviews and ratings (Zhang, Minshun, 2025).

With the growing and rapidly aging fleet of passenger vehicles (Zacharof et al., 2025), trade-in programs have become an increasingly common industry practice, allowing consumers to exchange old cars when purchasing new ones (Hu et al., 2025). The trade-in process serves a dual purpose: it simplifies transactions for customers and supports dealerships in meeting their inventory demands for the used-car market. As part of this process, dealerships conduct a vehicle trade-in appraisal (VTA) to determine the value of the customer's used vehicle, which is then applied toward the purchase of another vehicle. Trade-in valuations are systematically lower than listing prices because dealerships must account for reconditioning costs, inventory holding risks, and profit margins before resale (Israeli et al., 2021). While listing prices often represent the upper bound of market value, trade-in offers are strategically set to ensure that the vehicle can be resold competitively in the used-car market.

VTA demands significant expertise and dedication from automotive industry professionals. Figure 1.1 illustrates the trade-in process at Seez, a leader in digitalizing the automotive industry. This manual process, which depends completely on human experts, is not scalable in a rapidly evolving global automotive market, where trade-in requests are growing (Hu et al., 2025). This motivates the development of VTA systems that assist experts, optimizing decision-making and improving efficiency. The digitalization of the automotive industry has fundamentally transformed the trade-in process. Online platforms now enable customers to initiate trade-in requests remotely, submitting vehicle details and receiving preliminary valuations without visiting a dealership. This shift toward digital trade-in channels, driven by the same e-commerce dynamics reshaping the broader used-car market, amplifies the scalability challenge: as the volume of online trade-in requests grows, the reliance on human experts becomes an increasingly critical bottleneck. Automating VTA

is therefore not merely an operational concern for dealerships, but a core requirement for platforms seeking to deliver responsive and scalable pricing in the digital automotive marketplace.

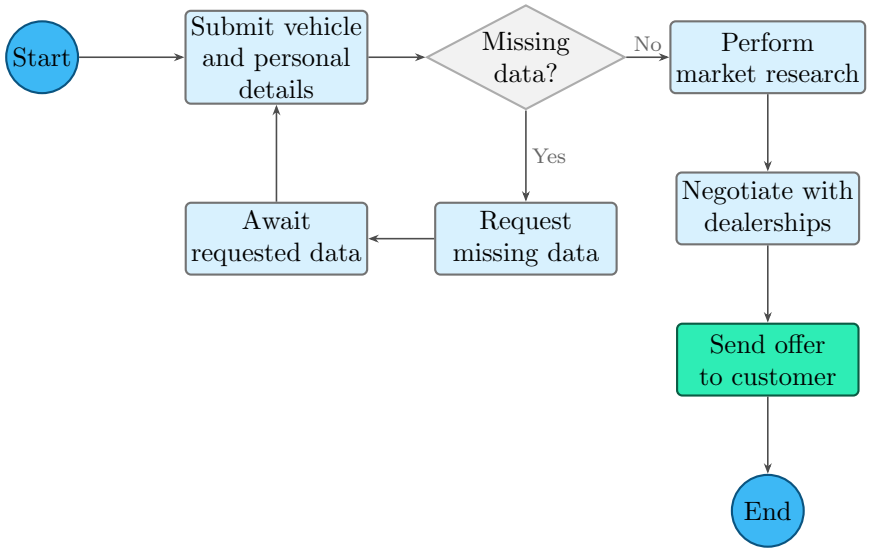


Figure 1.1: Vehicle trade-in appraisal process at Seez.

Machine learning (ML) for vehicle price prediction (VPP) is a well-established practice that has made significant progress in the listing and resale domains. Despite this success, VTA remains underexplored and constrained by specific challenges. Access to trade-in data is severely limited, as valuations depend on expert-driven assessments and are not readily available from public sources. An appealing approach to overcoming this challenge is to perform transfer learning from a data-rich source domain, such as listings. However, the absence of a publicly accessible standard for vehicle representation in the automotive industry impedes data integration across different data sources and automotive markets, hindering transfer learning and the identification of global pricing patterns in the listing domain.

The vehicle identification number (VIN) is often assumed to be the ideal universal standard for vehicle representation. Comprising a unique 17-character code assigned by manufacturers, the VIN is standardized under ISO 3779 (International Organization for Standardization, 2009) and includes three sections: the world manufacturer identifier (WMI), the vehicle descriptor section (VDS), and the vehicle identifier section (VIS). However, this standard is not publicly accessible, and different manufacturers use various systems to encode the VDS, making decoding difficult. Furthermore, open-source datasets often lack the VIN, and different countries may follow slightly different standards. As a result, the promise of the VIN as a standardized identifier is not realized in practice. The VIN is not alone in this shortfall: several other efforts have sought to prescribe how a vehicle should be described, yet none has achieved adoption in open datasets. The NHTSA vPIC database (National Highway Traffic Safety Administration, 2025) decodes VINs into approximately 130 attributes

but is restricted to US-market vehicles and excludes condition, usage history, and equipment. The ACES and PIES standards (Auto Care Association, 2026) provide structured fitment data for aftermarket parts in North America rather than whole-vehicle attributes. The W3C Automotive Ontology and its schema.org extension (W3C Automotive Ontology Community Group, 2024) define a minimal set of vehicle properties for web markup, but accept free-form text for key fields and omit trim, equipment, and condition entirely. Finally, COVESA's Vehicle Signal Specification (COVESA, 2025b) standardizes in-vehicle telemetry signals rather than vehicle-as-product attributes, and its own gap analysis acknowledges persistent interoperability issues across manufacturers (COVESA, 2025a). A standard that is not adopted at scale offers no interoperability in practice. Without a unified standard, dealerships and e-commerce platforms adopt diverse schemas to describe vehicles with varying attributes and naming conventions. These schemas typically exhibit three problems: inconsistent naming, where identical entities are represented by different string values across data sources; inconsistent granularity, where identical attributes vary in granularity across data sources; and scope inconsistency, where attribute values mix distinct vehicle characteristics. These problems are illustrated in Tables 1.1, 1.2, and 1.3. Left unaddressed, these inconsistencies propagate through downstream ML pipelines, undermining feature extraction, similarity computation, and ultimately model performance.

Table 1.1: Mercedes A-Class 200 representations. The brand attribute exhibits inconsistent naming by using "Mercedes-Benz" and "Mercedes" to represent the same entity. The model attribute exhibits inconsistent granularity by representing the same vehicle model at two different granularity levels: "A200" and "A Class". The transmission attribute exhibits both problems, recording the same automatic gearbox as a language-generic label ("Automatik"), a technology code ("DCT"), and a manufacturer trade name with gear count ("7G-DCT").

Marketplace	**Brand**	**Model**	**Transmission**
mobile.de	Mercedes-Benz	A 200	Automatik
bilbasen.dk	Mercedes	A200	DCT
autoscout24.it	Mercedes-Benz	A 200	Automatico
coches.net	Mercedes-Benz	Clase-A 200	7G-DCT
autotrader.co.uk	Mercedes-Benz	A Class	Automatic

The performance of ML algorithms largely depends on the choice of data representation (Bengio et al., 2013). To illustrate the importance of this within the automotive industry, consider the fuel type column in Table 1.2. Experts observe that hybrid electric vehicles (HEVs) are generally more cost-effective than plug-in hybrid electric vehicles (PHEVs). Representing both HEV and PHEV as a single category prevents the model from capturing the price difference between the two, limiting its expressiveness and likely reducing its predictive accuracy. On the other hand, representing HEV and PHEV as separate categories allows the model to

Table 1.2: AWD PHEV representations. The drive type attribute exhibits inconsistent naming by using "Integral" and "All wheel drive" to represent the same entity and the fuel type entries exhibit scope inconsistency by representing the energy source hybrid and the propulsion system PHEV as fuel types. The entry "Petrol Plug-in Hybrid" further conflates the fuel type petrol with the energy source and the propulsion system in a single label.

Marketplace	**Drive type**	**Fuel type**
mobile.de	All wheel drive	Plug-in-Hybrid (Benzin)
bilbasen.dk	Firehjulstræk	Plug-in petrol
autoscout24.it	Trazione integrale	Ibrida
coches.net	Integral	Híbrido enchufable
autotrader.co.uk	Four wheel drive	Petrol Plug-in Hybrid

Table 1.3: BMW 3 Series Touring representations. The model attribute exhibits inconsistent naming through localization, recording the same model as "3er", "3-serie", "Serie 3", and "3 Series"; the body type attribute likewise exhibits inconsistent naming, using five regional terms for the same estate body. The brand attribute, by contrast, is consistent across all marketplaces, illustrating that not every attribute is inconsistent for every entity.

Marketplace	**Brand**	**Model**	**Body type**
mobile.de	BMW	3er	Kombi
bilbasen.dk	BMW	3-serie	Stationcar
autoscout24.it	BMW	Serie 3	Station wagon
coches.net	BMW	Serie 3	Familiar
autotrader.co.uk	BMW	3 Series	Estate

capture their price differences, but it fails to encode their common characteristics. More importantly, this approach implicitly considers the distance between HEV and PHEV as equivalent to that between HEVs and internal combustion engine vehicles (ICEVs), or similarly, between PHEVs and ICEVs, which misrepresents their relationship. A hierarchical representation that preserves multiple granularity levels, as shown in Figure 1.2, along with a method for mapping any given representation to this standard, is essential for effective knowledge integration and learning across different automotive markets and data sources.

There are multiple commercial solutions for VTA. Edmunds and Kelley Blue Book are two popular choices for the US market, while Bilbasen serves as a comparable product for the Danish market. However, the methodologies, data sources, and performance metrics of these solutions are not publicly disclosed. As a result, these tools cannot be independently replicated or verified, making them unsuitable as baselines for academic research.

In this thesis, we present the first ML approach for VTA based on vehicle normalization and transfer learning. Unlike previous studies on VPP in the listing and

Figure 1.2: Hierarchical representation of a vehicle model (left) and a propulsion system (right), each with three levels of granularity. At each level, the highlighted node lies on the canonical path, while the dotted nodes denote alternative entities at the same granularity.

resale domains, which typically focus on limited datasets with few brands or models (Table 3.1), our method addresses a significantly larger variety of brands and models, better reflecting real-world conditions. Working under this setting, we made a discovery relevant to both VTA and VPP: models trained and evaluated on restricted sets of vehicle brands and models produce over-optimistic results that fail to generalize under real-world conditions of diversity.

The contributions of this work are as follows:

1. We introduce a novel framework for normalizing vehicle representations that enables fusing multiple heterogeneous datasets. Following the retrieval-augmented generation (RAG) paradigm, it combines large language models (LLMs) with dense information retrieval over a knowledge catalog to map any raw vehicle description to a normalized, domain-consistent representation. This component, together with its evaluation against extraction-only and nearest-neighbor retrieval baselines, is reported in (Quezada et al., 2026).

2. We present the first ML approach to vehicle trade-in appraisal (VTA). Inspired by how experts evaluate a trade-in request, our method transfers knowledge from cross-market listing data into the trade-in domain, using the normalizer to facilitate the learning of a model over heterogeneous data sources by mapping them into a shared representation.

3. We systematically assess the effect of vehicle diversity on predictive performance evaluation, demonstrating that evaluating on datasets with restricted diversity can lead to results that underestimate the complexity of VPP on more diverse datasets that reflect real-world conditions.

1.1 Hypotheses

The main hypothesis of this work is stated as follows:

> *If transfer learning is applied from the vehicle listing domain to the vehicle trade-in domain, then the performance of models on VTA will improve, because the patterns learned in the source domain provide information that is relevant and adaptable to the target domain.*

This hypothesis is broken down into the following sub-hypotheses:

1. Vehicle listing data contains transferable patterns that are relevant to VTA.
2. The standardization of heterogeneous data sources increases the effectiveness of transfer learning.

1.2 Objectives

General Objective

The general objective of this work is to develop a machine learning (ML) system for vehicle trade-in appraisal (VTA) that optimizes the accuracy and efficiency of the appraisal process.

Specific Objectives

The specific objectives are to:

1. Design and implement a vehicle normalization method based on large language models (LLMs) and retrieval-augmented generation (RAG) that transforms textual vehicle representations from diverse sources into structured, standardized data.
2. Develop a transfer learning model that adapts patterns learned from the vehicle listing domain to the vehicle trade-in domain in order to improve accuracy in VTA under data-scarce scenarios.
3. Evaluate the performance of the proposed system in terms of accuracy and efficiency, analyzing the impact of each component of the system.

1.3 Outline

The remainder of this thesis is organized as follows. Chapter 2 provides the technical foundation for this study, offering an overview of ensemble methods, LLMs, and RAG. Chapter 3 reviews the literature on vehicle price prediction (VPP). Chapter 4 formalizes the method proposed in this work. Chapter 5 describes the experimental setup, the data, and the performance metrics, presents the results in the listing and trade-in domains, analyzes how the error distributes across the vehicle population, examines the effect of vehicle diversity on predictive performance evaluation, and discusses the findings. Chapter 6 acknowledges the limitations of our study, summarizes the findings, and suggests directions for future work. Appendix A summarizes the notation used throughout the thesis. Appendix B details the prompts used in the vehicle normalization framework. Appendix C empirically evaluates the scalability of the normalization framework on open-source datasets. Appendix D evaluates the pricing models in the trade-in domain. Appendix E reports the statistical significance of the experimental results. Finally, Appendices F and G describe the software and hardware used in the experiments.

Chapter 2

Background

This chapter introduces the concepts that underpin the methods developed in this thesis. We begin with the predictive models used for appraisal: ensemble methods. We then turn to large language models and the three text-centric tasks our normalization pipeline builds on, presented in the order they are applied: information extraction, entity resolution, and information retrieval. We close with retrieval-augmented generation, which combines retrieval and language modeling and frames how these components interact. Throughout, each task is presented in its general, conceptual form; the most relevant prior applications are reviewed in Chapter 3.

2.1 Ensemble methods

Ensemble methods combine multiple base learners to produce predictions that often surpass those of any individual model. This section reviews three foundational ensemble paradigms that underpin the methods employed in this work.

2.1.1 Bagging

Bagging, short for bootstrap aggregating, is a parallel ensemble technique introduced by Breiman (1996). Unlike gradient boosting, which builds models sequentially, bagging trains multiple base learners independently on different bootstrap samples of the training data and combines their predictions through averaging (for regression) or voting (for classification). Formally, given a training set $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, bagging generates B bootstrap samples $D_1, D_2, \ldots, D_B$ by sampling n instances with replacement from D. A base learner h_b is trained on each bootstrap sample D_b, and the final prediction is computed as:

$$\hat{f}(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^{B} h_b(\mathbf{x}). \tag{2.1}$$

Random forest (RF) (Breiman, 2001) is a widely used bagging method whose base learners h_b are decision trees. Beyond training each tree on its own bootstrap sample,

RF introduces a second source of randomness: at each split, only a random subset of the features is considered as a candidate. This feature subsampling decorrelates the individual trees and reduces variance further than bagging alone, making RF robust to overfitting while remaining competitive on tabular data.

2.1.2 Gradient boosting

Gradient boosting is a powerful sequential ensemble technique that has long delivered state-of-the-art results on tabular data, especially on tasks with heterogeneous features and noisy data (Friedman, 2000). Popular implementations include XGBoost (Chen and Guestrin, 2016), LightGBM (Ke et al., 2017), and CatBoost (Prokhorenkova et al., 2018). The algorithm constructs a sequence of models $F_t : \mathbb{R}^m \to \mathbb{R}$, for $t = 0, 1, \ldots$, additively: at each step, the model is updated as $F_t = F_{t-1} + \alpha h_t$, where α is a step size and h_t is a base predictor selected from a function family $\mathcal{H}$. Let $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ denote the loss function measuring the discrepancy between a prediction and the true target value. The predictor h_t is then selected as:

$$h_t = \arg\min_{h \in \mathcal{H}} \mathbb{E}_{(\mathbf{x},y)\sim p_{\text{data}}}[(r_t(\mathbf{x}, y) - h(\mathbf{x}))^2], \tag{2.2}$$

where

$$r_t(\mathbf{x}, y) = -\frac{\partial \ell(y, F_{t-1}(\mathbf{x}))}{\partial F_{t-1}(\mathbf{x})}. \tag{2.3}$$

Binary decision trees are a common choice for base learners. A decision tree recursively divides the feature space $\mathbb{R}^m$ into J disjoint regions $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_J$ according to conditions on the feature values (Breiman et al., 1984). At each node, the tree splits the feature space into two subspaces based on a condition like $\mathbf{x}_i \leq t$ or $\mathbf{x}_i > t$ where t is a threshold chosen to minimize an impurity or loss function computed on the data. For regression tasks, variance reduction is typically used as the splitting criterion. The tree grows recursively until a stopping condition (such as a maximum depth or a minimum node size) is met. Once the tree has fully grown, it can be expressed as:

$$h(\mathbf{x}) = \sum_{j=1}^{J} b_j \mathbf{1}[\mathbf{x} \in \mathcal{R}_j], \tag{2.4}$$

where b_j is usually the average of the target values in the region $\mathcal{R}_j$.

2.1.3 Stacking

Stacking, also known as stacked generalization (Wolpert, 1992), is a meta-learning ensemble technique that combines the predictions of multiple base models using a higher-level meta-learner. Unlike bagging and boosting, which use homogeneous base

learners, stacking typically combines heterogeneous models to exploit their complementary strengths. The method operates in two stages: in the first stage, K base models $\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_K$ are trained on the original training data; in the second stage, a meta-learner g is trained on a new feature set constructed from the base model predictions. For a given instance $\mathbf{x}$, the stacked prediction is:

$$\hat{f}_{\text{stack}}(\mathbf{x}) = g\left(\hat{f}_1(\mathbf{x}), \hat{f}_2(\mathbf{x}), \ldots, \hat{f}_K(\mathbf{x})\right). \tag{2.5}$$

To prevent overfitting, the base model predictions used to train the meta-learner are typically obtained through cross-validation, ensuring that the meta-learner is trained on out-of-fold predictions.

2.2 Large language models

Large language models (LLMs), such as Claude Opus 4.6, GPT-5.4, or Gemini 3.1 Pro, are large autoregressive models based on the decoder-only Transformer architecture (Vaswani et al., 2017; Radford et al., 2019), trained on massive text corpora. Their training objective is next-token prediction: given the preceding text, the model learns to predict the following token. This factorizes the probability of a token sequence $w_1, \ldots, w_T$ into a product of conditional probabilities,

$$p(w_1, \ldots, w_T) = \prod_{t=1}^{T} p(w_t \mid w_1, \ldots, w_{t-1}), \tag{2.6}$$

and at inference time text is generated autoregressively, sampling one token at a time from this distribution.

For our purposes, this generative process can be abstracted as a function that maps an input context to a generated text output. Both are sequences of tokens (words or sub-words) drawn from a fixed vocabulary: the context c is the text the model conditions on, and the output y is the text it produces in response,

$$y = \text{LLM}(c). \tag{2.7}$$

The context c bundles whatever text the model is given, such as an instruction, optional examples, and the input to be processed. A notable strength of LLMs, first explored by Brown et al. (2020), is *in-context learning*: the ability to perform new tasks by conditioning on a few examples or instructions provided directly in the context c, without additional training or fine-tuning. This ability allows the models to generalize beyond their training data, making them especially effective in scenarios with limited or non-existing labeled examples.

2.3 Information extraction

Information extraction (IE) is the task of turning unstructured text into a structured form that downstream systems can query and reason over (Cowie and Lehnert, 1996;

Sarawagi, 2008). In its most general textbook form, IE is posed as sequence labeling: each word x_i of the input $x = (x_1, \ldots, x_n)$ is assigned a label y_i marking the type of information it carries (Sarawagi, 2008),

$$(x_1, \ldots, x_n) \longmapsto (y_1, \ldots, y_n). \tag{2.8}$$

The variant relevant to this work is attribute-value extraction (also called slot filling), where the goal is to assign each attribute in a fixed set $a_1, \ldots, a_m$ a value drawn from the text:

$$\text{extract}(x) = (v_1, \ldots, v_m), \tag{2.9}$$

where v_j is the value of attribute a_j found in x, or empty if it is not mentioned. Classical systems learn this from annotated data, whereas modern systems instruct an LLM (Section 2.2) to emit the values directly, using a context $c = (\text{instruction}, x)$ that pairs the extraction instruction with the input text, so that $(v_1, \ldots, v_m) = \text{LLM}(c)$ with few or no labeled examples.

2.4 Entity resolution

Entity resolution (ER), also known as record linkage, is the task of deciding whether two records refer to the same real-world entity (Fellegi and Sunter, 1969; Elmagarmid et al., 2007). Given a pair of records (r_a, r_b), ER assigns a binary label that classifies the pair as a match or a non-match:

$$\text{match}(r_a, r_b) = \begin{cases} 1 & \text{if } r_a \text{ and } r_b \text{ denote the same entity,} \\ 0 & \text{otherwise.} \end{cases} \tag{2.10}$$

Since comparing every pair is too costly on large collections, a blocking step first narrows the comparisons to likely candidates (Elmagarmid et al., 2007). Classical methods score agreement across record fields (Fellegi and Sunter, 1969), while modern ones fine-tune or instruct LLMs to make this decision.

2.5 Information retrieval

Information retrieval (IR) is the task of finding, within a collection, the items most relevant to a query (Manning et al., 2008). Given a query q and a collection of items $\{d_1, \ldots, d_N\}$, a scoring function ranks the items by relevance and the top-k are returned. In the standard vector space model (Salton et al., 1975), the query and each item are represented as vectors $\mathbf{q}$ and $\mathbf{d}$ and scored by their cosine similarity:

$$s(q, d) = \frac{\langle \mathbf{q}, \mathbf{d} \rangle}{\|\mathbf{q}\| \, \|\mathbf{d}\|}. \tag{2.11}$$

Sparse methods build these vectors from word counts, whereas dense methods use learned embeddings that capture meaning rather than exact wording.

2.6 Retrieval-augmented generation

Retrieval-augmented generation (RAG) is a framework, first introduced by Lewis et al. (2020), that enhances LLMs by integrating them with a retrieval system that supplies external knowledge. This combination allows the LLM to generate responses based on both learned patterns and up-to-date or domain-specific information. Concretely, RAG enriches the LLM context of Equation (2.7) with external evidence: given a query q and a corpus $\mathcal{C} = \{d_1, \ldots, d_N\}$, it first retrieves the top-k relevant items $\mathcal{R} = \text{top-k}_q(\mathcal{C})$ (Section 2.5) and then builds the context from the query together with the retrieved items, $c = (q, \mathcal{R})$, so that

$$y = \text{LLM}(q, \mathcal{R}). \tag{2.12}$$

Whereas a plain LLM is confined to the context supplied at input, retrieving relevant documents at inference time grounds generation in external knowledge, mitigating hallucination and outdated knowledge. This architecture is especially effective in knowledge-intensive tasks such as open-domain question answering, where the quality of generation is directly improved by grounding it in retrieved evidence. It is also essential in settings where the task depends on evolving data sources, such as entity resolution over a dynamic product catalog.

Chapter 3

Related work

Estimating the value of a vehicle has been studied extensively, but almost exclusively in the form of vehicle price prediction (VPP) over listing or resale prices. This chapter reviews that body of work (Section 3.1) and the comparatively unexplored problem of vehicle trade-in appraisal (VTA) (Section 3.2). Because our framework first normalizes heterogeneous vehicle records before learning, we further review the three techniques that underpin our normalization pipeline: information extraction (Section 3.3), entity resolution (Section 3.4), and information retrieval (Section 3.5). The conceptual foundations of these techniques were introduced in Chapter 2; here we focus on the most relevant prior applications.

3.1 Vehicle price prediction

Extensive research has been conducted to estimate vehicle prices, employing a range of learning algorithms, from linear regression (LR) to self-attention-based neural networks, exploring various markets and analyzing different vehicle representations and data modalities (tabular, textual, visual) for listing and resale prices. We organize this section around the dimensions along which these works differ: the markets and datasets they use, how vehicles are represented, how features are encoded and engineered, how the prediction target is defined, and which learning algorithms are employed. We refer the reader to Table 3.1 for the per-study details.

Markets, scale, and data availability. The reviewed studies span a wide range of markets and scales. Geographically, they cover the German (Lessmann and Voß, 2017; Dutulescu et al., 2023), US (Carvalho et al., 2023; Nandan and Ghosh, 2023; Fayyaz et al., 2025), Swiss (Bergmann and Feuerriegel, 2025), and Moroccan (Mhamedi et al., 2026) markets, as well as several multi-country settings (Dutulescu et al., 2023; Madhusudhanan et al., 2024), while a few works do not disclose their market (Khotimah et al., 2025; Lin and Wang, 2026). Dataset sizes differ by three orders of magnitude, from a few thousand listings (Fayyaz et al., 2025) to the two-million-record corpus of Madhusudhanan et al. (2024). Most datasets are propri-

etary; the publicly available ones are hosted on Kaggle (Nandan and Ghosh, 2023; Khotimah et al., 2025; Fayyaz et al., 2025), Ali Tianchi (Lin and Wang, 2026), or Mendeley Data (Mhamedi et al., 2026), although their original provenance is often uncited. Vehicle diversity also varies widely, from the single brand and six models of Lessmann and Voß (2017) to the 81 brands and 1,049 models of Mhamedi et al. (2026), the most diverse prior work. Several authors explicitly note the absence of a large-scale benchmark for used VPP (Madhusudhanan et al., 2024) and call for standardized datasets across markets and vehicle attributes (Dutulescu et al., 2023).

Vehicle representation and data modalities. Vehicles are predominantly represented by tabular numerical and categorical features, and the tabular signal consistently dominates. Some works enrich this core with additional modalities: textual descriptions integrated through a co-attention mechanism (Carvalho et al., 2023), visual features extracted with EfficientNet and Swin Transformer backbones (Dutulescu et al., 2023), and temporal attributes transformed into sine-cosine pairs to capture cyclical patterns such as seasons or daily cycles (Madhusudhanan et al., 2024). The added value of non-tabular modalities is limited, however: Dutulescu et al. (2023) found that visual features had little impact, while *year*, *mileage*, and vehicle add-ons were the key predictors. Equipment and add-on information likewise yields consistent gains; Bergmann and Feuerriegel (2025) report a 3.27% reduction in MAE with LightGBM when 18 binary equipment attributes (such as *head up display*, *park assistance*, or *keyless access*) are included. Khotimah et al. (2025) even predict prices from a minimal schema of five attributes (*brand*, *model*, *variant*, *year*, and *mileage*) and identify the *variant* as the most influential feature; notably, this attribute exhibits *scope inconsistency*, conflating the vehicle submodel, engine size, energy source, and trim level. This is precisely the kind of representational heterogeneity that our normalization component targets.

Feature encoding and engineering. Studies differ markedly in how they encode and engineer features. Categorical encoding spans a spectrum from one-hot encoding (Lessmann and Voß, 2017; Nandan and Ghosh, 2023; Bergmann and Feuerriegel, 2025), through learned entity embeddings (Guo and Berkhahn, 2016) as used by Carvalho et al. (2023); Dutulescu et al. (2023), to target encoding (Fayyaz et al., 2025) and simple integer encoding (Khotimah et al., 2025). Beyond encoding, several works show that feature engineering drives substantial gains. Fayyaz et al. (2025) engineer derived attributes such as vehicle age and power-to-weight ratio, lifting the R^2 of a stacking regressor from 0.1486 to 0.8899; Lin and Wang (2026) construct multiplicative, additive, and difference-based feature interactions whose removal significantly degrades a stacking model across all metrics; and Mhamedi et al. (2026) replace four raw attributes (brand, model, year, and equipment) with a single LLM-generated "Vehicle Desirability Score" designed to emulate an expert's holistic evaluation, achieving an 18.4% reduction in MAE and improving R^2 from 0.901 to 0.945. This reliance on custom, dataset-specific preprocessing, discussed

further below, is a recurring theme.

Target definition and loss. The prediction target follows three main conventions. Most studies regress the price directly, either as the raw value (Nandan and Ghosh, 2023; Khotimah et al., 2025) or, more commonly, log-transformed to stabilize its skewed distribution (Carvalho et al., 2023; Fayyaz et al., 2025). A second convention, introduced by Lessmann and Voß (2017) and followed by Bergmann and Feuerriegel (2025), defines the target as the ratio of the actual price to the new-vehicle list price. The loss function is predominantly MSE, with the notable exception of Madhusudhanan et al. (2024), who optimize a Negative Log-Likelihood (NLL) to quantify predictive uncertainty. Across all of these works, however, the target is invariably a listing or resale price; none addresses the trade-in valuation that is the subject of this thesis.

Modeling approaches. The modeling approaches range from linear regression and decision trees (DT) to ensembles, neural networks, and transformers. Gradient-boosted tree ensembles (XGBoost, LightGBM, and CatBoost), often combined through stacking, are the most frequently top-performing family (Nandan and Ghosh, 2023; Fayyaz et al., 2025; Lin and Wang, 2026; Mhamedi et al., 2026), as summarized in Table 3.1. Neural architectures are also well represented, from multilayer perceptrons (MLP) over entity embeddings (Dutulescu et al., 2023; Khotimah et al., 2025) to the artificial neural network (ANN) of Carvalho et al. (2023), whose co-attention branches fuse textual and tabular modalities and whose learned embeddings, when fed to random forest (RF) or LightGBM regressors, outperformed an end-to-end regression layer, and the SAINT-based transformer of Madhusudhanan et al. (2024) (Somepalli et al., 2021). Two conceptual findings stand out. Lessmann and Voß (2017) distinguish *heterogeneity* (the varying impact of features across dataset subgroups) from *specificity* (how tailored a model is to a particular vehicle model), finding that, of 19 algorithms, only 7 benefit from a high-specificity setup while 12 worsen, with linear methods benefiting most; this suggests that heterogeneity, rather than specificity, is the primary driver of the observed gains. Separately, Madhusudhanan et al. (2024) move beyond point estimates to quantify predictive uncertainty, an aspect largely absent from the rest of the literature.

Taken together, the studies in Table 3.1 demonstrate substantial progress in VPP research across diverse markets, modalities, and modeling approaches. Nonetheless, most efforts have concentrated on listing and resale prices, leaving VTA largely unexplored. Moreover, data-centric perspectives remain underdeveloped, with prior work underscoring the absence of a large-scale benchmark dataset for used VPP (Madhusudhanan et al., 2024) and emphasizing the need for standardized datasets across markets and vehicle attributes (Dutulescu et al., 2023).

Beyond modeling choices, a recurring obstacle in this literature is that each study builds its own dataset-specific preprocessing pipeline, which does not transfer across markets, languages, or platforms. Bergmann and Feuerriegel (2025) mapped

Table 3.1: Summary of selected studies.

Study	Market	N	M	Type	N_b	N_m	Best Algorithm
Lessmann and Voß (2017)	DE	450,000	9	Resale	1	6	Ensembles
Carvalho et al. (2023)	US	57,285	7	Listing	-	-	Ensembles
Dutulescu et al. (2023)	DE, RO	322,262	13	Listing	> 10	-	ANN
Nandan and Ghosh (2023)	US	374,136	16	Resale	-	-	Ensembles
Madhusudhanan et al. (2024)	CZ, DE, ES, IT	2,000,000	-	Resale	-	-	ANN
Bergmann and Feuerriegel (2025)	CH	92,239	16	Resale	4	65	Ensembles
Fayyaz et al. (2025)	US	4,009	11	Listing	57	-	Ensembles
Khotimah et al. (2025)	-	14,657	15	Listing	5	5	ANN
Lin and Wang (2026)	-	150,000	30	Resale	40	248	Ensembles
Mhamedi et al. (2026)	MA	101,896	14	Listing	81	1,049	Ensembles

N: dataset size; M: number of attributes; N_b: number of brands; N_m: number of models

equipment strings onto manually curated categories through keyword and regular-expression rules co-designed with car dealers, a recipe explicitly tailored to their partner's market, while Fayyaz et al. (2025) relied on regular-expression extraction and per-field standardization to improve downstream regression. Such specialization impairs transfer: Dutulescu et al. (2023) trained on German listings and tested on Romanian ones, found that car-model embeddings failed to transfer for infrequent models, and explicitly called for a method for dataset standardization across car markets. The same limitation appears at scale, where Madhusudhanan et al. (2024) reported that the high cardinality of unstandardized categorical features hindered state-of-the-art tabular models on their two-million-record corpus. The closest existing attempt at systematic vehicle data structuring is the ontology-based recommender of Le et al. (2022), which organizes vehicle types, brands, and specifications for semantic matching; however, it targets recommendation rather than ML preprocessing and does not address the normalization of heterogeneous listings across markets. These observations motivate the vehicle normalization component of this thesis, whose building blocks we review in Sections 3.3–3.5.

3.2 Vehicle trade-in appraisal

In contrast to the extensive VPP literature, VTA has received little attention. To the best of our knowledge, no prior work has applied machine learning methods to the vehicle trade-in process, and the few contributions that reference trade-in do so only tangentially, falling short of providing a rigorous foundation. Besner (2024a) highlights the potential for automotive dealerships to enhance trade-in and inventory strategies through automated solutions, and in a subsequent white paper, Besner (2024b) introduces a proprietary trade-in acquisition model aimed at optimizing lead generation, claiming superior performance over traditional VTA systems. However, the white paper lacks essential details required to reproduce or use the method as a baseline, such as the vehicle representation, dataset diversity, learning algorithm, and evaluation metrics. Similarly, Kumar et al. (2023) claims to predict trade-in values, yet their experiments rely on a listing dataset rather than genuine trade-in

data. In contrast, Yi et al. (2021) addresses trade-in strategies from an operations management standpoint, using analytical and economic modeling to determine optimal pricing and production decisions for manufacturers; their work neither employs nor discusses machine learning approaches and is, therefore, not directly relevant to this study.

3.3 Information extraction

Information extraction (IE) from product descriptions, introduced in Section 2.3, has evolved from task-specific sequence taggers to general-purpose LLM-based extractors. OpenTag (Zheng et al., 2018) formalized product attribute-value extraction (AVE) as sequence tagging with a BiLSTM-CRF, and Wang et al. (2020) later recast AVE as question answering over a shared encoder. More recently, Brinkmann et al. (2024) broadened AVE to include attribute-value *normalization*, introducing the WDC-PAVE benchmark and reporting that GPT-4 reaches 91% F1 with only ten in-context demonstrations; they further found that LLMs are particularly strong at name expansion and string canonicalization, precisely the capability our extraction step exploits. In a complementary direction, GoLLIE (Sainz et al., 2024) showed that encoding annotation guidelines as code with docstrings enables high-quality zero-shot IE, a role analogous to that of the structured vehicle schema in our framework.

3.4 Entity resolution

Entity resolution (ER), discussed in Section 2.4, determines whether two surface forms refer to the same real-world entity (e.g., *mercedes* versus *mercedes-benz*). Pre-LLM approaches fine-tuned pre-trained language models: Ditto (Li et al., 2020) adapted BERT and RoBERTa with domain-knowledge injection and data augmentation, improving over the prior state of the art by up to 29% F1, while Narayan et al. (2022) were the first to cast entity matching as a prompting task for large language models. Subsequent work has shown that zero-shot LLMs can substantially outperform fine-tuned models on previously unseen entities, with Peeters et al. (2025) reporting gains of 40–68% F1 in this regime, a setting directly analogous to vehicle normalization, where new brands, models, and trims appear continuously. Wang et al. (2025) further compare matching, comparing, and selecting strategies for LLM-based entity matching and find that *selecting* the correct entity from a set of candidates is the most cost-effective, mirroring our resolver, which selects the first matching candidate from the retrieved set. Together, these results motivate our use of an LLM resolver over the retrieved candidates rather than a fixed similarity threshold.

3.5 Information retrieval

Information retrieval (IR), discussed in Section 2.5, has shifted from sparse lexical matching toward dense, learned representations. Reimers and Gurevych (2019) established the bi-encoder paradigm now standard in dense retrieval, encoding texts independently into embeddings compared by cosine similarity and reducing the cost of finding the most similar pair in a collection of 10,000 sentences from roughly 65 hours to about five seconds, which makes large-scale semantic search practical through offline indexing. Building on this paradigm, Karpukhin et al. (2020) showed that learned dual-encoder embeddings (Dense Passage Retrieval) outperform sparse lexical baselines such as BM25 by 9–19% in top-20 retrieval accuracy, and Izacard et al. (2022) later showed that fully unsupervised dense retrieval via contrastive learning can match or exceed BM25 without labeled query-passage pairs, a regime directly analogous to vehicle matching, where labeled query-passage pairs are scarce. More recently, general-purpose embedding models trained with large-scale contrastive pre-training, such as OpenAI's `text-embedding-3-small` (Neelakantan et al., 2022), provide strong off-the-shelf representations accessible through a simple API. To keep dense retrieval tractable over large, growing collections, approximate nearest-neighbor search, such as that provided by the FAISS library (Johnson et al., 2021), makes similarity search efficient at scale.

Chapter 4

Method

This chapter presents our method for vehicle trade-in appraisal (VTA). We begin by formulating the problem and stating our objective. Building on this formulation, we introduce the two components of our approach: a vehicle normalization framework, which establishes a common representation across heterogeneous domains, and a transfer learning scheme that builds on this representation to transfer knowledge from cross-market listing domains to the trade-in domain. We then analyze the computational complexity of the resulting method. The notation used throughout this chapter is summarized in Appendix A.

4.1 Problem formulation

Formally, vehicle trade-in appraisal (VTA) can be defined as the task $f_T : \mathcal{X}_T \to \mathcal{Y}_T$ where $\mathcal{X}_T \subseteq \mathbb{R}^{d_T}$ denotes the input feature space and $\mathcal{Y}_T \subseteq \mathbb{R}^+$ denotes the output value space. In our application, domains correspond to distinct geographic car markets, each characterized by its own data distribution and feature representation. To learn this task in the target domain, we are given a labeled dataset

$$D_T = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n_T}, \quad \mathbf{x}_i \in \mathcal{X}_T \subseteq \mathbb{R}^{d_T},\, y_i \in \mathcal{Y}_T, \tag{4.1}$$

where y_i is the trade-in value for $\mathbf{x}_i$. To enhance learning performance, we use a collection of source domains $r \in \{\mathrm{DK}, \mathrm{KSA}, \mathrm{UAE}\}$, each with a labeled dataset

$$D_S^{(r)} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n_S^{(r)}}, \quad \mathbf{x}_i \in \mathcal{X}_S^{(r)} \subseteq \mathbb{R}^{d_S^{(r)}},\, y_i \in \mathcal{Y}_S^{(r)} \subseteq \mathbb{R}^+, \tag{4.2}$$

where labeled data correspond to a related task $f_S^{(r)} : \mathcal{X}_S^{(r)} \to \mathcal{Y}_S^{(r)}$ of vehicle price prediction (VPP). The key advantage is the abundance of source domain data, that is, $n_S^{(r)} \gg n_T$ for all r.

This setting exhibits multiple forms of distribution shift (Tamang et al., 2025). The marginal input distributions differ across domains: $\mathcal{P}_T(\mathbf{X}) \neq \mathcal{P}_S^{(r)}(\mathbf{X})$ for all r. Moreover, due to differences in both task objectives and feature representations, the joint distributions necessarily differ: $\mathcal{P}_T(\mathbf{X}, Y) \neq \mathcal{P}_S^{(r)}(\mathbf{X}, Y)$. Following the

taxonomy of Pan and Yang (2010), this scenario constitutes an instance of multi-source heterogeneous inductive transfer learning.

Our objective is to learn f_T by exploiting the abundant labeled data from the source domains. Our approach proceeds in two stages. First, we train source models $\hat{f}_S^{(r)}$ for $r \in \{\text{DK}, \text{KSA}, \text{UAE}\}$ using only their respective labeled datasets. Second, we adopt a stacking-based transfer learning strategy: for each target instance $\mathbf{x} \in \mathcal{X}_T$, we compute predictions from all source models and concatenate these with the original feature vector to form an augmented representation. A critical prerequisite is that all source models $\hat{f}_S^{(r)}$ must accept target domain instances as input, despite the heterogeneous feature spaces $\mathcal{X}_S^{(r)} \neq \mathcal{X}_T$. This requirement also applies when transferring knowledge between source domains, necessitating a feature space normalization scheme to establish a common representation across all domains.

4.2 Vehicle normalization

We frame vehicle normalization as a three-component pipeline (Figure 4.1). A raw record $\mathbf{x}$ is first serialized and passed to an LLM that jointly extracts candidate attribute values $\hat{\mathbf{x}} = (\hat{a}_1, \ldots, \hat{a}_m)$. Closed-domain attributes are emitted directly. Each open-domain attribute $\hat{a}_j$ is encoded and matched against a knowledge catalog $\mathcal{K}_j$ to retrieve its top-k canonical candidates $\mathcal{R}_j$. Finally, an LLM resolver decides whether any candidate $c_i \in \mathcal{R}_j$ refers to the same real-world entity as $\hat{a}_j$. If one does, it replaces $\hat{a}_j$; otherwise $\hat{a}_j$ is kept as a novel entity and appended to $\mathcal{K}_j$. The remainder of this section formalizes each component.

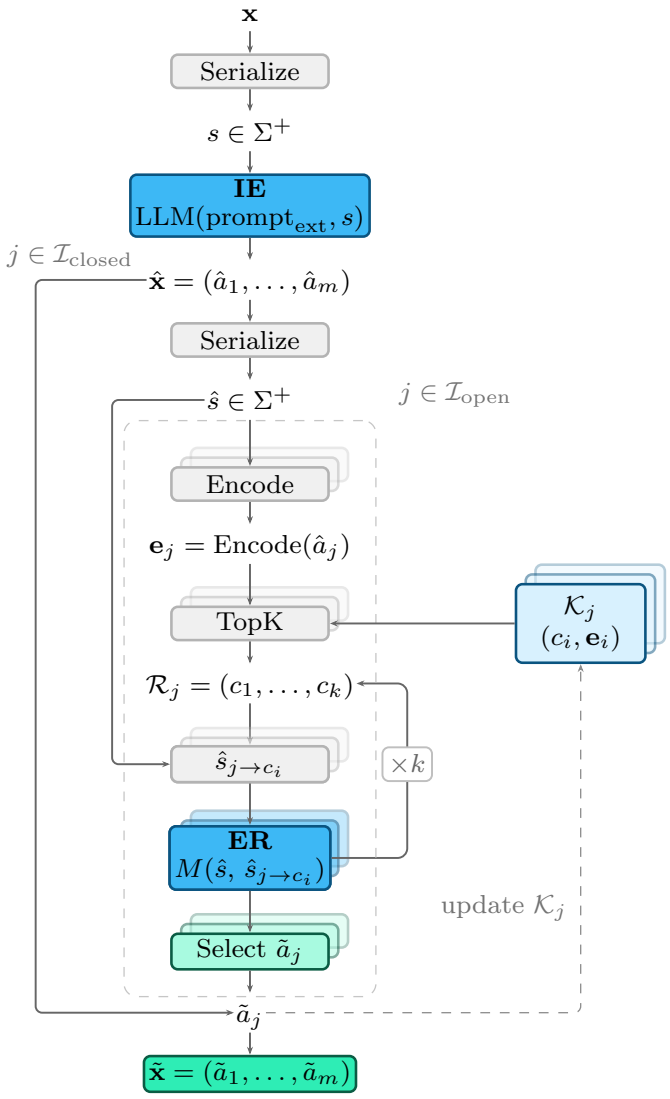


Figure 4.1: Vehicle normalization pipeline. A raw record $\mathbf{x}$ is serialized and fed to an LLM that jointly extracts candidate attribute values $\hat{\mathbf{x}}$ (IE). Closed-domain attributes ($j \in \mathcal{I}_{\text{closed}}$) bypass the loop and are emitted as-is. Each open-domain attribute ($j \in \mathcal{I}_{\text{open}}$, dashed box) is encoded into $\mathbf{e}_j$, its top-k canonical candidates $\mathcal{R}_j$ are retrieved from catalog $\mathcal{K}_j$, and each candidate c_i is substituted into $\hat{s}_{j \to c_i}$ and tested by an LLM entity resolver $M(\cdot, \cdot)$. Stacked blocks denote one instance per open-domain attribute, and the $\times k$ arrow indicates that the substitution-and-resolution step is repeated for each retrieved candidate. Select returns the first matching c_{i^*}; otherwise $\hat{a}_j$ is kept as a novel entity and the catalog is expanded (dashed feedback). See Algorithm 1.

Let the normalized vehicle schema consist of m attributes $A_1, A_2, \ldots, A_m$, where each attribute A_j is associated with a domain $\mathcal{V}_j$ of valid values. According to their value range, attributes are either numerical or categorical, so that $m = m_{\text{num}} + m_{\text{cat}}$.

Numerical attributes have a continuous domain

$$\mathcal{V}_j = \mathbb{R}^+, \quad j \in \mathcal{I}_{\text{num}}, \tag{4.3}$$

while categorical attributes have a discrete domain $\mathcal{V}_j = \mathcal{C}_j$ for $j \in \mathcal{I}_{\text{cat}}$.

For normalization, the relevant property is instead whether an attribute's domain is fixed in advance or grows over time. We therefore partition the attributes into closed- and open-domain:

$$\mathcal{V}_j = \begin{cases} \mathcal{V}_j^{\text{closed}} & \text{if } j \in \mathcal{I}_{\text{closed}}, \\ \mathcal{V}_j^{\text{open}} & \text{if } j \in \mathcal{I}_{\text{open}}. \end{cases} \tag{4.4}$$

A *closed-domain* attribute has a domain that is fully specified in advance and does not change during normalization. This includes all numerical attributes (numerical ranges, e.g., mileage $\in \mathbb{R}^+$) and finite categorical sets. An *open-domain* attribute (e.g., brand, model) is categorical with a domain that grows as the system encounters previously unseen entities. Hence $\mathcal{I}_{\text{num}} \subseteq \mathcal{I}_{\text{closed}}$ and $\mathcal{I}_{\text{open}} \subseteq \mathcal{I}_{\text{cat}}$.

The normalized vehicle space is the Cartesian product of all attribute domains:

$$\tilde{\mathcal{X}} = \prod_{j=1}^{m} \mathcal{V}_j. \tag{4.5}$$

Vehicle normalization is formulated as a mapping

$$\phi : \Sigma^+ \to \tilde{\mathcal{X}}, \tag{4.6}$$

that takes a serialized vehicle representation $s = \text{Serialize}(\mathbf{x}) \in \Sigma^+$, where $\mathbf{x}$ is the raw input record, Σ denotes the character alphabet, and $\text{Serialize} : \mathcal{X} \to \Sigma^+$ renders a vehicle representation as a string. The mapping produces $\phi(s) = \tilde{\mathbf{x}} \in \tilde{\mathcal{X}}$. Since the datasets in this study are tabular, the serialization step concatenates each column name with its value (e.g., *col1: val1, col2: val2, ...*) into a single string. The same operator renders the extracted representation $\hat{\mathbf{x}}$ and the candidate replacements $(\hat{a}_1, \ldots, c_i, \ldots, \hat{a}_m)$ used in the entity-resolution step. In practice, when the input is already textual (e.g., open-text descriptions), this step may be unnecessary.

Note that the normalized vehicle space $\tilde{\mathcal{X}}$ is unique. However, as illustrated in Figure 4.2, the normalized instance sets $\tilde{\mathbf{X}}_S^{(r)}$ and $\tilde{\mathbf{X}}_T$, with dimensions $(n_S^{(r)}, \tilde{d}_S)$ and $(n_T, \tilde{d}_T)$ respectively, lie in different feature spaces $\tilde{\mathcal{X}}_S = \tilde{\mathcal{X}}$ and $\tilde{\mathcal{X}}_T \subseteq \tilde{\mathcal{X}}$ as $d_T > d_S^{(r)}$. Furthermore, neither $\tilde{d}_S$ nor $\tilde{d}_T$ equals m, since the encoding of categorical attributes increases the overall dimensionality.

Having formalized the normalized vehicle schema and the normalization mapping ϕ, we now describe the process by which ϕ operates.

Knowledge catalog For each open-domain attribute A_j with $j \in \mathcal{I}_{\text{open}}$, we maintain a knowledge catalog:

$$\mathcal{K}_j = \{(c_1, \mathbf{e}_1), (c_2, \mathbf{e}_2), \ldots, (c_{|\mathcal{K}_j|}, \mathbf{e}_{|\mathcal{K}_j|})\}, \tag{4.7}$$

where each pair $(c_i, \mathbf{e}_i)$ consists of a canonical entity value $c_i \in \mathcal{V}_j^{\text{open}}$ and its corresponding embedding $\mathbf{e}_i = \text{Encode}(c_i) \in \mathbb{R}^{d_{\text{emb}}}$. The encoding function is defined as

$$\text{Encode} : \Sigma^* \to \mathbb{R}^{d_{\text{emb}}}. \tag{4.8}$$

In this study, we use a pre-trained embedding model, such as OpenAI's *text-embedding-3-small*. In practice, however, any suitable text encoder may be employed. Examples include classic methods like TF-IDF or Word2Vec, as well as modern transformer-based models such as BERT or RoBERTa.

Information extraction Given a serialized vehicle description s, a single LLM call extracts and infers candidate values for all attributes simultaneously:

$$\hat{\mathbf{x}} = (\hat{a}_1, \ldots, \hat{a}_m) = \text{LLM}(\text{prompt}_{\text{ext}}, s). \tag{4.9}$$

For closed-domain attributes ($j \in \mathcal{I}_{\text{closed}}$), the extracted value is used directly as $\tilde{a}_j = \hat{a}_j$, since the prompt constrains the LLM output to $\mathcal{V}_j^{\text{closed}}$. For open-domain attributes ($j \in \mathcal{I}_{\text{open}}$), the candidate $\hat{a}_j$ is passed to the retrieval and resolution steps described below. The information extraction prompt is detailed in Appendix B.

Information retrieval For each open-domain attribute A_j, the extracted candidate is embedded as $\mathbf{e}_j = \text{Encode}(\hat{a}_j)$, and the top-$k$ most similar canonical entities are retrieved from catalog $\mathcal{K}_j$:

$$\mathcal{R}_j = \text{top-k}_{(c_i, \mathbf{e}_i) \in \mathcal{K}_j} \ \text{sim}(\mathbf{e}_j, \mathbf{e}_i), \quad \text{sim}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u}^\top \mathbf{v}}{\|\mathbf{u}\|_2 \|\mathbf{v}\|_2}, \tag{4.10}$$

with candidates $\mathcal{R}_j = (c_1, \ldots, c_k)$ ranked in decreasing order of similarity.

Cosine similarity quantifies the angular proximity between two vectors, where $\|\cdot\|_2$ denotes the Euclidean (L2) norm. This metric is scale-invariant and produces values in $[-1, 1]$, where 1 indicates parallel vectors (maximal similarity), 0 indicates orthogonal vectors, and -1 indicates antiparallel vectors. Cosine similarity is particularly well-suited for high-dimensional embedding spaces, as it focuses on semantic orientation rather than magnitude (Levy et al., 2015).

Entity resolution Given the extracted attributes $\hat{\mathbf{x}}$ and the ranked candidates $\mathcal{R}_j = (c_1, \ldots, c_k)$ for an open-domain attribute A_j, the resolver compares two serialized vehicle descriptions. Let $\hat{s} = \text{Serialize}(\hat{\mathbf{x}})$ be the serialization of the extracted attributes. For each candidate $c_i \in \mathcal{R}_j$, we construct a variant in which the value of attribute A_j is replaced by c_i:

$$\hat{s}_{j \to c_i} = \text{Serialize}(\hat{a}_1, \ldots, c_i, \ldots, \hat{a}_m). \tag{4.11}$$

An LLM then evaluates whether both descriptions refer to the same vehicle:

$$M(\hat{s},\ \hat{s}_{j \to c_i}) = \text{LLM}(\text{prompt}_{\text{res}},\ \hat{s},\ \hat{s}_{j \to c_i}) \in \{0, 1\}, \tag{4.12}$$

where 1 indicates a match. The entity resolution prompt is detailed in Appendix B.

Catalog update All k candidates are evaluated and the normalized value is:

$$\tilde{a}_j = \begin{cases} c_{i^*}, & i^* = \min\{i : \delta_i = 1\}, \\ \hat{a}_j & \text{otherwise (novel entity)}, \end{cases} \tag{4.13}$$

where $\delta_i = M(\hat{s},\, \hat{s}_{j \to c_i})$. In the novel-entity case, the catalog is expanded:

$$\mathcal{K}_j \leftarrow \mathcal{K}_j \cup \{(\hat{a}_j, \text{Encode}(\hat{a}_j))\}. \tag{4.14}$$

Nearest-neighbor alternative A conventional alternative to the entity resolver is to omit it entirely and accept the top-ranked retrieved candidate whenever its similarity exceeds a fixed threshold τ, that is, a nearest-neighbor matcher over the catalog:

$$\tilde{a}_j = \begin{cases} c_1, & \text{if } \text{sim}(\mathbf{e}_j, \mathbf{e}_1) \geq \tau, \\ \hat{a}_j & \text{otherwise.} \end{cases} \tag{4.15}$$

We evaluated this variant against the full pipeline in previous work (Quezada et al., 2026), sweeping τ over $[0.50, 0.95]$ on two human-annotated listing datasets. No fixed cutoff is competitive on both axes of the problem at once: raising τ makes the matcher conservative and improves the detection of novel entities, but it stops merging genuine synonyms, while lowering it merges aggressively and absorbs new entities into existing catalog entries. Even at the threshold most favorable to each metric individually, the nearest-neighbor variant remains below the full pipeline both in normalization quality (macro F1 of 0.846 and 0.870 against 0.890 and 0.885) and in novelty detection (F1 of 0.588 and 0.795 against 0.623 and 0.878). The gap concentrates precisely where the embedding neighborhood is ambiguous, which is the case the resolver is designed for: a single scalar cannot separate a synonym of a known entity from a genuinely new one, whereas comparing the two serialized descriptions in context can. This thesis therefore instantiates the full configuration.

Complete pipeline The full normalization function maps a serialized vehicle description to its normalized representation:

$$\phi(s) = \tilde{\mathbf{x}} = (\tilde{a}_1, \ldots, \tilde{a}_m), \tag{4.16}$$

where each $\tilde{a}_j$ is the value emitted directly by extraction for closed-domain attributes ($j \in \mathcal{I}_{\text{closed}}$) and the resolved (or novel) value for open-domain attributes ($j \in \mathcal{I}_{\text{open}}$). Algorithm 1 summarizes the vehicle normalization framework.

4.3 Vehicle trade-in appraisal

We frame vehicle trade-in appraisal (VTA) as a two-stage pipeline (Figure 4.2) that realizes the stacking-based transfer learning strategy of Section 4.1. We first train a

Algorithm 1 Vehicle Normalization

Require: Raw vehicle record $\mathbf{x}$, Catalogs $\{\mathcal{K}_j\}_{j\in\mathcal{I}_{\text{open}}}$
Ensure: Normalized representation $\tilde{\mathbf{x}} = (\tilde{a}_1, \ldots, \tilde{a}_m)$

$s \leftarrow \text{Serialize}(\mathbf{x})$
$\hat{\mathbf{x}} \leftarrow \text{LLM}(\text{prompt}_{\text{ext}}, s)$
for $j \in \mathcal{I}_{\text{closed}}$ **do**
    $\tilde{a}_j \leftarrow \hat{a}_j$
end for
$\hat{s} \leftarrow \text{Serialize}(\hat{\mathbf{x}})$
for $j \in \mathcal{I}_{\text{open}}$ **do**
    $\mathbf{e}_j \leftarrow \text{Encode}(\hat{a}_j)$
    $\mathcal{R}_j \leftarrow \text{TopK}_{(c_i,\mathbf{e}_i)\in\mathcal{K}_j} \text{sim}(\mathbf{e}_j, \mathbf{e}_i)$
    for $c_i \in \mathcal{R}_j$ in ranked order **do**
        $\hat{s}_{j\to c_i} \leftarrow \text{Serialize}(\hat{a}_1, \ldots, c_i, \ldots, \hat{a}_m)$
        $\delta_i \leftarrow M(\hat{s},\ \hat{s}_{j\to c_i})$
    end for
    if $\exists\, i : \delta_i = 1$ **then**
        $i^* \leftarrow \min\{i : \delta_i = 1\}$
        $\tilde{a}_j \leftarrow c_{i^*}$
    else
        $\tilde{a}_j \leftarrow \hat{a}_j$ ▷ novel entity
        $\mathcal{K}_j \leftarrow \mathcal{K}_j \cup \{(\hat{a}_j, \text{Encode}(\hat{a}_j))\}$
    end if
end for
return $\tilde{\mathbf{x}} = (\tilde{a}_1, \ldots, \tilde{a}_m)$

source model $\hat{f}_S^{(r)}$ on each market's labeled data to perform vehicle price prediction (VPP), then augment every normalized target instance (Section 4.2) with the source predictions and train the target model $\hat{f}_T$ on the result. The remainder of this section formalizes each component.

Source models For each market $r \in \{\text{DK}, \text{KSA}, \text{UAE}\}$, we train a CatBoost model:

$$\hat{f}_S^{(r)} : \tilde{\mathcal{X}} \to \mathbb{R}, \tag{4.17}$$

to perform VPP on the normalized dataset $\tilde{D}_S^{(r)}$ by minimizing the empirical risk:

$$\hat{f}_S^{(r)} = \arg\min_{f\in\mathcal{F}} \sqrt{\frac{1}{n_S^{(r)}} \sum_{i=1}^{n_S^{(r)}} \ell(\tilde{y}_i^{(r)}, f(\tilde{\mathbf{x}}_i^{(r)}))}, \tag{4.18}$$

where

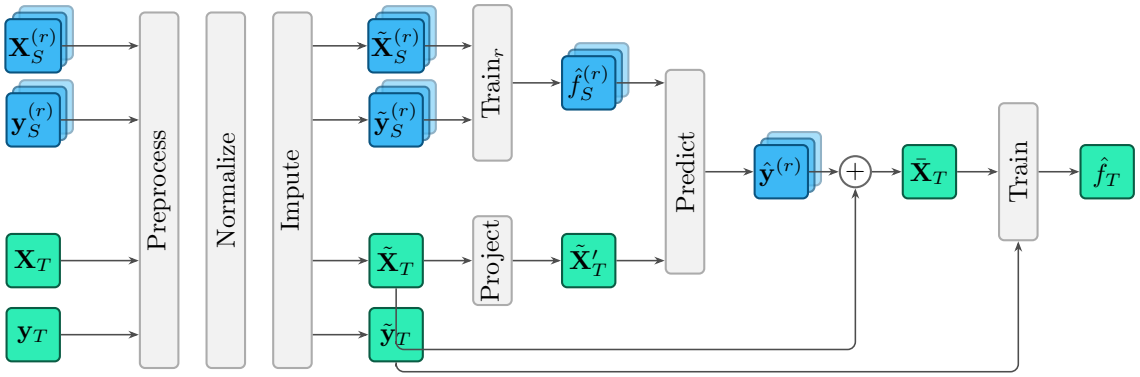


Figure 4.2: Vehicle trade-in appraisal pipeline. The source datasets $\mathbf{X}_S^{(r)}$ (blue) and the target dataset $\mathbf{X}_T$ (mint) are preprocessed, normalized, and imputed into the common normalized space, with target values log-transformed to $\tilde{\mathbf{y}}$. Each normalized source dataset $\tilde{\mathbf{X}}_S^{(r)}$ trains a VPP model $\hat{f}_S^{(r)}$. The normalized target dataset $\tilde{\mathbf{X}}_T$ is projected to $\tilde{\mathbf{X}}_T'$ and scored by the source models to produce predictions $\hat{\mathbf{y}}^{(r)}$, which augment the target features into $\bar{\mathbf{X}}_T$. Finally, the VTA model $\hat{f}_T$ is trained on the augmented target dataset.

- $\tilde{y}_i^{(r)} = \log(y_i^{(r)})$ is the log-transformed target price[1]
- $\ell(\tilde{y}, \hat{y}) = (\tilde{y} - \hat{y})^2$ is the squared error loss
- $\mathcal{F}$ is the hypothesis space of gradient boosted oblivious decision trees

CatBoost incorporates regularization through leaf value smoothing. When computing the value for each leaf node, the algorithm applies:

$$v_{\text{leaf}} = -\frac{\sum_{i \in \text{leaf}} g_i}{\sum_{i \in \text{leaf}} h_i + \lambda}, \tag{4.19}$$

where g_i and h_i are the first and second derivatives of the loss with respect to the prediction for instance i, and λ is the `l2_leaf_reg` hyperparameter. This smoothing mechanism prevents overfitting by shrinking leaf values toward zero, particularly for leaves with few training instances.

Feature augmentation For each instance $j = 1, \ldots, n_T$ in the normalized target dataset $\tilde{D}_T$, we augment its feature vector $\tilde{\mathbf{x}}_j$ with listing price predictions from the source models:

$$\bar{\mathbf{x}}_j = \left[\tilde{\mathbf{x}}_j, \hat{f}_S^{(\text{UAE})}(\tilde{\mathbf{x}}_j'), \hat{f}_S^{(\text{KSA})}(\tilde{\mathbf{x}}_j'), \hat{f}_S^{(\text{DK})}(\tilde{\mathbf{x}}_j')\right], \tag{4.20}$$

[1]The inverse transformation $\hat{y} = \exp(\hat{f}(x))$ introduces a bias in expectation due to Jensen's inequality. However, we do not apply bias correction methods (e.g., Duan's smearing estimator) as preliminary experiments showed minimal impact on the evaluation metrics, with differences below 0.5% in MAPE.

where $\tilde{\mathbf{x}}'_j = \pi(\tilde{\mathbf{x}}_j) \in \tilde{\mathcal{X}}$ denotes the projection into the normalized vehicle space. The source model predictions are in log-space, maintaining consistency with the target transformation. This yields an augmented dataset:

$$\bar{D}_T = \{(\bar{\mathbf{x}}_j, y_j)\}_{j=1}^{n_T}, \quad \bar{\mathbf{x}}_j \in \bar{\mathcal{X}}_T, \tag{4.21}$$

where $\bar{\mathcal{X}}_T$ denotes the augmented feature space in the target domain.

Target model We train a CatBoost model:

$$\hat{f}_T : \bar{\mathcal{X}}_T \to \mathbb{R}, \tag{4.22}$$

to perform VTA on the augmented dataset $\bar{D}_T$ by minimizing the empirical risk:

$$\hat{f}_T = \arg\min_{f \in \mathcal{F}} \frac{1}{n_T} \sum_{j=1}^{n_T} \ell(\tilde{y}_j, f(\bar{\mathbf{x}}_j)), \tag{4.23}$$

where

- $\tilde{y}_j = \log(y_j)$ is the log-transformed target
- $\ell(\tilde{y}, \hat{\tilde{y}}) = |\tilde{y} - \hat{\tilde{y}}|$ is the absolute error loss
- $\mathcal{F}$ is the hypothesis space of gradient boosted oblivious decision trees

Regularization is applied through the same leaf value smoothing mechanism described for the source models (see Equation 4.19).

Interpretation The target model implicitly learns a composite function:

$$\hat{f}_T(\bar{\mathbf{x}}_j) = g\left(\tilde{\mathbf{x}}_j, \left\{\hat{f}_S^{(r)}(\tilde{\mathbf{x}}'_j)\right\}_{r=1}^{3}\right), \tag{4.24}$$

where

- $\tilde{\mathbf{x}}_j \in \tilde{\mathcal{X}}$ provides the consistent, cross-domain representation on which transfer learning operates
- $\left\{\hat{f}_S^{(r)}(\tilde{\mathbf{x}}'_j)\right\}_{r=1}^{3}$ materializes transfer learning by embedding knowledge from the listing domain into the trade-in domain, yielding cross-market signals that reflect the outcomes of the market research conducted by automotive experts

Algorithm 2 summarizes the transfer learning approach, and Figure 4.3 describes the inference pipeline.

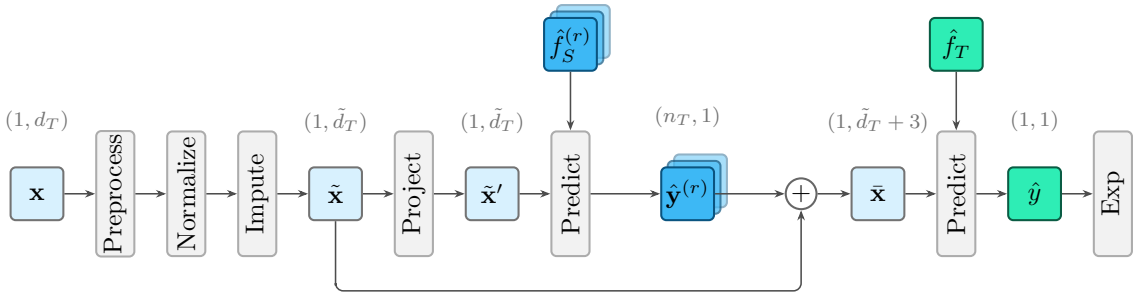


Figure 4.3: Inference pipeline. The input vehicle $\mathbf{x}$ is preprocessed, normalized, and imputed to yield $\tilde{\mathbf{x}}$. This vector is then projected into the normalized feature space and augmented with predictions $\hat{\mathbf{y}}^{(r)}$ from the VPP models $\hat{f}_S^{(r)}$. Finally, the VTA model $\hat{f}_T$ generates a trade-in value $\hat{y}$, which the exponential function transforms back to the original target space.

Algorithm 2 Transfer Learning

Require: Source datasets $\{D_S^{(r)}\}_{r=1}^{3}$, target dataset D_T
Ensure: Trained model $\hat{f}_T$

for $r \in \{\text{UAE}, \text{KSA}, \text{DK}\}$ **do**
  $\hat{f}_S^{(r)} \leftarrow \text{CatBoost}(D_S^{(r)})$
end for
for $(\mathbf{x}_j, y_j) \in D_T$ **do**
  $\mathbf{p}_j \leftarrow [\hat{f}_S^{(\text{UAE})}(\tilde{\mathbf{x}}_j'), \hat{f}_S^{(\text{KSA})}(\tilde{\mathbf{x}}_j'), \hat{f}_S^{(\text{DK})}(\tilde{\mathbf{x}}_j')]$
  $\bar{\mathbf{x}}_j \leftarrow [\tilde{\mathbf{x}}_j, \mathbf{p}_j]$
end for
$\bar{D}_T \leftarrow \{(\bar{\mathbf{x}}_j, y_j)\}_{j=1}^{n_T}$
$\hat{f}_T \leftarrow \text{CatBoost}(\bar{D}_T)$
return $\hat{f}_T$

4.4 Computational complexity

Algorithm 1: Vehicle Normalization We analyze the inference complexity, assuming catalogs have been populated during training. For each vehicle, the normalization pipeline performs: (i) one embedding API call with cost T_{emb} to generate the vector representations for retrieval, (ii) one LLM call to extract all attributes (both closed-domain and open-domain), (iii) up to $k \cdot m_{\text{open}}$ LLM calls for entity resolution (up to k per open-domain attribute, one per retrieved candidate), and (iv) m_{open} nearest-neighbor retrievals. Each open-domain attribute j maintains a separate catalog $\mathcal{K}_j$, and retrievals are performed independently per attribute. A brute-force retrieval over $\mathcal{K}_j$ computes cosine similarity between the query embedding and all $|\mathcal{K}_j|$ entries, each requiring $O(d_{\text{emb}})$ operations. The per-vehicle complexity under

brute-force is:

$$O\left(T_{\text{emb}} + (1 + k \cdot m_{\text{open}}) \cdot T_{\text{LLM}} + d_{\text{emb}} \sum_{j \in \mathcal{I}_{\text{open}}} |\mathcal{K}_j|\right), \tag{4.25}$$

where T_{LLM} denotes the LLM inference cost. Due to the hybrid nature of the algorithm, which combines external API calls with local computation, this expression mixes time and operation complexity. Specifically, T_{emb} and T_{LLM} represent API latency costs that are constant with respect to catalog size, while the final term captures the computational cost of retrieval operations that scales with catalog cardinality.

Each catalog starts empty ($|\mathcal{K}_j| = 0$) and grows as novel entities are discovered during training. Catalog cardinalities $|\mathcal{K}_j|$ vary by attribute: *brand* and *model* are practically bounded by the finite set of manufacturers and their product lines, whereas *submodel* and *trim level* may grow substantially depending on vehicle representation. In this work, we use the Chroma vector database, which employs Hierarchical Navigable Small World (HNSW) indexing (Malkov and Yashunin, 2020). HNSW reduces retrieval to $O(d_{\text{emb}} \cdot \log |\mathcal{K}_j|)$ per query, lowering the retrieval term to $O(d_{\text{emb}} \sum_{j \in \mathcal{I}_{\text{open}}} \log |\mathcal{K}_j|)$. With $m_{\text{open}} = 4$ and $k = 5$, each vehicle requires one embedding call, one extraction call, and up to $k \cdot m_{\text{open}} = 20$ resolution calls in the worst case. This worst case is rarely reached: attribute values already present in a catalog bypass resolution entirely, and processing only unique serialized representations (39,381 in our case) avoids redundant work, since vehicles sharing identical representations yield identical normalizations. As reported in Appendix C, the amortized number of resolution calls per vehicle is therefore far below this bound. The dominant costs are LLM and embedding API inference, making the algorithm I/O-bound.

Algorithm 2: Transfer learning Let R denote the number of source domains. The algorithm consists of three stages: (i) training R source models, (ii) augmenting target instances with R predictions, and (iii) training the target model on the augmented dataset. Let $C^{(r)}_{\text{train}}(n, d)$ and $C^{(r)}_{\text{pred}}(n, d)$ denote the cost of training and predicting for the model in domain r, and $C^{(T)}_{\text{train}}(n, d)$ denote the cost of training the target model. The total complexity is:

$$\begin{aligned} O\Bigg(& \sum_{r=1}^{R} C^{(r)}_{\text{train}}(n_S^{(r)}, \tilde{d}_S) + \sum_{r=1}^{R} C^{(r)}_{\text{pred}}(n_T, \tilde{d}_S) \\ & + C^{(T)}_{\text{train}}(n_T, \tilde{d}_T + R) \Bigg). \end{aligned} \tag{4.26}$$

Considering a CatBoost model with T trees of depth D, training costs $O(n \cdot d \cdot T \cdot D)$ and prediction costs $O(n \cdot T \cdot D)$ due to the symmetric structure of oblivious decision

trees. Substituting these costs yields:

$$O\left(\tilde{d}_S \cdot T \cdot D \sum_{r=1}^{R} n_S^{(r)} + n_T \cdot (\tilde{d}_T + R) \cdot T \cdot D\right). \tag{4.27}$$

With $n_S^{(\mathrm{DK})} \approx 10^6$ dominating the source data, training the largest source model constitutes the computational bottleneck. Unlike Algorithm 1, this algorithm is compute-bound rather than I/O-bound.

Summary Table 4.1 summarizes the key characteristics. Algorithm 1 scales with unique vehicle representations and is limited by API throughput, while Algorithm 2 scales with dataset size and is limited by training time on the largest source domain.

Table 4.1: Computational characteristics of the proposed algorithms.

Aspect	**Algorithm 1**	**Algorithm 2**
Bottleneck	LLM inference	Source model training
Bound type	I/O-bound	Compute-bound
Scales with	Unique vehicles	Dataset size
Parallelizable	Per vehicle	Per source domain

Chapter 5

Experiments

This chapter describes the experimental evaluation of the proposed method. It first presents the experimental setup, the data, the performance metrics, and the hyperparameter tuning procedure. It then reports the results in the listing and trade-in domains, analyzes how the error distributes across the vehicle population, and examines the effect of vehicle diversity on predictive performance evaluation, before closing with a discussion of the findings.

5.1 Experimental setup

We instantiate the vehicle normalization framework described in Section 4.2 as detailed below.

Schema The proposed schema comprises $m = 19$ attributes, of which $m_{\text{num}} = 6$ are numerical and $m_{\text{cat}} = 13$ are categorical, with $m_{\text{open}} = 4$ designated as open-domain. Table 5.1 lists the full schema with per-attribute descriptions, and Table 5.2 specifies the domains of the closed-domain categorical attributes.

Models Information extraction and entity resolution both use the LLM *gpt-4.1-nano*, while candidate embeddings for the open-domain attributes are computed with the embedding model *text-embedding-3-small* ($d_{\text{emb}} = 1536$).

Retrieval configuration Open-domain attributes are stored and retrieved using the Chroma vector database, with $k = 5$ candidates retrieved per query. This depth follows the retrieval ablation reported in (Quezada et al., 2026): normalization quality saturates between $k = 3$ and $k = 5$, below which the resolver has too few candidates to compare against and tends to over-merge, and above which the number of LLM calls grows linearly without further gains. Since four attributes are open-domain (*brand*, *model*, *submodel*, and *trim level*), we maintain four distinct collections, one per attribute, to store the literal values and their embeddings.

Table 5.1: Proposed schema.

Attribute	**Description**
Brand	Vehicle manufacturer or marque
Model	Base vehicle model name
Submodel	Specific variant of the vehicle model
Trim level	Package of features, styling, and equipment
Year	Model year of the vehicle
Mileage	Total distance the vehicle has traveled
Body type	Physical structure and shape of the vehicle
Transmission type	Type of transmission system
Transmission technology	Specific transmission mechanism
Transmission gears	Number of gear ratios in the transmission
Energy source	Main energy source powering the vehicle
Propulsion system	Configuration of the propulsion system
Fuel type	Specific fuel used by the vehicle
Drive type	Wheels receiving power from the drivetrain
Engine size	Total volume of all cylinders in the engine
Engine power	Maximum power output of the engine
Engine aspiration	Air intake method
Injection type	Fuel injection method
Cylinders	Number of cylinders in the engine

Table 5.2: Domain of closed-domain attributes.

Attribute	**Domain**
Body type	sedan, hatchback, suv, crossover, coupe, convertible, wagon, pickup, van, minivan, roadster, other, unknown
Drive type	fwd, rwd, awd, 4wd, other, unknown
Transmission type	manual, automatic, semi-automatic, other, unknown
Transmission technology	tc, cvt, sct, dct, other, unknown
Energy source	fossil, hybrid, electric, other, unknown
Propulsion system	icev, ev, pev, bev, hev, mhv, phev, erev, fcev, pfcev, other, unknown
Fuel type	petrol, diesel, cng, lpg, lng, methanol, propane, hydrogen, other, unknown
Engine aspiration	natural, turbo, supercharger, other, unknown
Injection type	direct, indirect, dual, other, unknown

Serialization Based on the minimal common schema shared across all datasets, the serialized vehicle representation concatenates the *brand*, *model*, *variant*, *transmission*, and *fuel type* attributes. The *year* and *mileage* attributes are consistently represented across all source datasets and can therefore be excluded from the system instructions provided to the LLM.

All experiments employ four ensemble algorithms (Random Forest, XGBoost,

LightGBM, and CatBoost). Algorithms supporting early stopping (CatBoost, XGBoost, LightGBM) are configured with a patience of 5 rounds. With the exception of the CatBoost appraisal model, which minimizes absolute error (Section 4.3), all models minimize squared error loss in log-space:

$$\mathrm{SE}_{\log} = \frac{1}{n}\sum_{i=1}^{n}(\tilde{y}_i - \hat{f}(\mathbf{x}_i))^2. \tag{5.1}$$

To account for variability introduced by different train-test splits, we generate 10 distinct training and testing pairs for each dataset listed in Table 5.3, using different random seeds. For each algorithm (RF, XGBoost, LightGBM, CatBoost) and each dataset, we evaluate both a version without normalization and a version with normalization, indicated by the '+' postfix.

For VPP, we evaluate RF, XGBoost, LightGBM, and CatBoost. Each algorithm is trained with and without normalization, yielding 8 configurations per dataset. We apply a hold-out validation strategy with 10% of the data reserved for validation, relying on early stopping for regularization.

For VTA, we use the same algorithms with identical early stopping and patience configurations. For each trade-in algorithm, we evaluate 8 transfer learning configurations (4 listing algorithms $\times$ 2 normalization settings) plus 2 baselines without transfer (with and without normalization), yielding 10 configurations per algorithm.

To demonstrate the effectiveness of vehicle normalization, we establish baselines for both tasks. For VPP, we consider the non-normalized version of each algorithm as a baseline. For VTA, we consider both the non-normalized version and the non-normalized and non-transfer version of each algorithm as baselines. Additionally, Appendix D presents the performance of VPP models applied directly to the trade-in domain by excluding trade-in-specific attributes and using only attributes from either the minimal common schema or the proposed schema. Their poor performance confirms that pricing and appraisal are fundamentally different tasks, necessitating a transfer learning approach to exploit the knowledge embedded in pricing models.

5.2 Datasets

The datasets used in this work are summarized in Table 5.3. The listings are sourced from established automotive e-commerce marketplaces operating across Europe and the Middle East, where vehicles are listed and traded online by both dealers and private sellers. As is common across such platforms, each marketplace independently defines its own vehicle schema, resulting in heterogeneous naming conventions and granularity inconsistencies, as illustrated in Tables 1.1 and 1.2.

For practical purposes, all prices are converted to euros. Before normalization, all data sources share a minimal common schema of 12 attributes, with categorical attributes including *brand*, *model*, *variant*, *body type*, *transmission type*, and *fuel type*, and numerical attributes including *year*, *mileage*, *engine power*, *engine size*,

cylinders, and *price*. The trade-in dataset extends this minimal common schema with 3 additional categorical attributes and 13 numerical attributes. The categorical attributes include *accident* (indicating whether the vehicle has been involved in an accident), *exhaust emission level* (based on European standards), and *segment*. The numerical attributes include *fuel consumption*, *maximum speed*, *number of previous owners*, emissions (CO_2, CO, and NOx), physical dimensions (*height*, *length*, *width*, and *maximum laden mass*), *number of airbags*, *number of seats*, and *number of doors*.

Despite having a minimal common schema, these datasets are not free from inconsistent naming, inconsistent granularity, scope inconsistency, and substantial missing data (detailed below). Vehicle normalization is therefore essential.

Table 5.3: Summary of datasets.

Market	**Type**	N	N_b	N_m	μ_{price}	σ_{price}
DK	Listing	972.4K	126	1,433	31.7K	51.9K
UAE	Listing	101.4K	149	1,096	51.6K	14.7K
KSA	Listing	19.0K	65	467	23.0K	19.1K
DK	Trade-in	1.2K	56	250	20.4K	16.5K

N: dataset size; N_b: number of brands; N_m: number of models

N_b and N_m are raw counts; normalization merges synonymous entries, reducing them.

Figure 5.1 shows the most popular brands within each data source. The leaderboards differ sharply across markets: Volkswagen, Peugeot, and Ford lead the European DK listings, whereas Toyota is the most frequent brand in both the UAE and KSA markets, followed there by Mercedes and Hyundai, respectively. The trade-in domain is dominated by Mercedes, which alone accounts for 31.9% of its records. Several premium and regional brands (e.g. Land Rover and Porsche in the UAE, Changan in KSA) appear in only a single market, underscoring the heterogeneity of vehicle supply across regions.

Beyond the most frequent brands, the listing datasets exhibit a pronounced long tail, where a small fraction of brands and models accounts for the majority of listings. Figure 5.2 quantifies this concentration through cumulative coverage curves. In the DK listings, the 15 most frequent brands and the 132 most frequent models (out of 96 and 1,082 after normalization, respectively) already account for 80% of the data, with the remaining categories scattered across a sparse tail; the same skew holds across markets. This concentration has direct implications for generalization: a model exposed predominantly to the head of the distribution may appear accurate while failing on the long tail, an effect we examine explicitly in Section 5.8.

Table 5.4 summarizes the missing data for each attribute across the datasets. The *brand* and *model* attributes have no missing values. However, several attributes, including *body type*, *transmission*, and *engine power*, are entirely missing in some datasets. This extensive missing data in critical attributes highlights the fragmentation of the minimal common schema of 12 attributes, described above, and accentuates the need for a normalization framework that ensures compliance with the proposed schema of 19 attributes, described in Table 5.1, inferring any missing data.

Figure 5.1: The eight most frequent brands within each data source, as a share of that market's records. Each market is shown in its own color; the sharply different leaderboards reflect the heterogeneous brand mix across regions.

Figure 5.3 displays the estimated price distribution across data sources (left panel), derived using kernel density estimation (KDE), with all distributions clustering around 20K € and exhibiting right skewness. Notably, the UAE listings data exhibits a heavier tail, indicating a wider range of higher-priced vehicles in this market. This trend is more pronounced when grouping the data by *brand*, as shown in the right panel of Figure 5.3. Luxury brands such as Pagani, Koenigsegg, and Bugatti drive the heavier tail in the UAE market.

Vehicle age is another dominant determinant of price. Figure 5.4 shows the median price as a function of vehicle age across all datasets. All markets exhibit the expected monotonic depreciation, with DK listings losing roughly 37% of their value within the first five years. Disaggregating the DK listings by energy source reveals

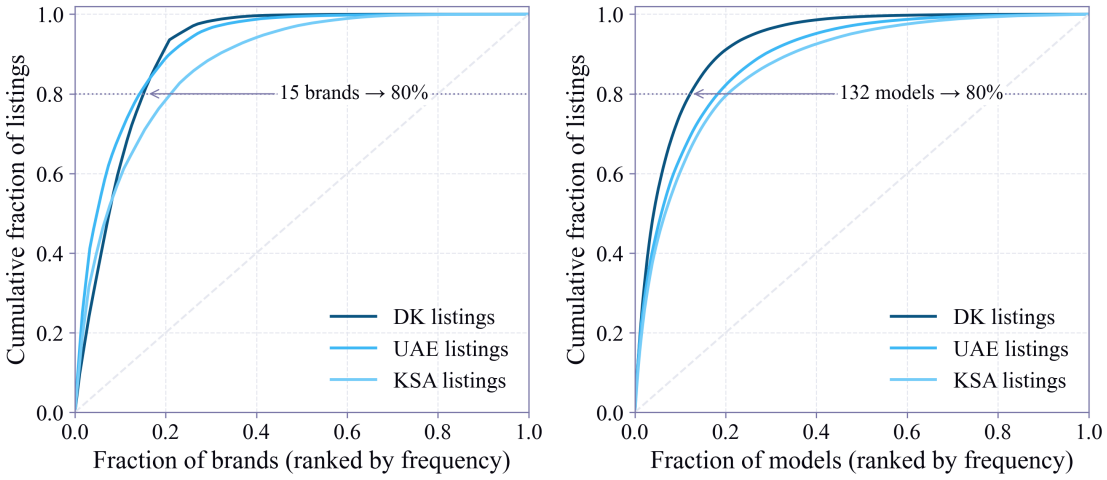


Figure 5.2: Cumulative fraction of listings covered as brands (left) and models (right) are added in order of decreasing frequency. The diagonal denotes a uniform distribution; the dotted line marks 80% coverage.

Table 5.4: Missing data across datasets.

Attribute	DK Listings	UAE Listings	KSA Listings	DK Trade-in
brand	0.00	0.00	0.00	0.00
model	0.00	0.00	0.00	0.00
variant	18.24	16.39	2.49	0.00
body type	46.42	12.65	100.00	100.00
transmission	15.10	0.00	0.00	100.00
fuel type	5.79	3.19	0.02	0.00
year	0.02	0.01	0.00	0.00
mileage	1.58	8.42	10.42	0.11
engine power	26.10	100.00	100.00	1.67
engine size	31.20	44.70	0.03	4.92
cylinders	100.00	4.45	0.03	14.64
accident	-	-	-	0.00
segment	-	-	-	7.48
height	-	-	-	24.80
length	-	-	-	24.80
width	-	-	-	24.80
owners	-	-	-	0.33
CO emissions	-	-	-	30.61
CO2 emissions	-	-	-	24.47
NOx emissions	-	-	-	30.61
maximum speed	-	-	-	23.46
seat quantity	-	-	-	2.12
door quantity	-	-	-	8.94
airbag quantity	-	-	-	35.42
fuel consumption	-	-	-	4.92
maximum laden mass	-	-	-	0.33
exhaust emission level	-	-	-	14.25

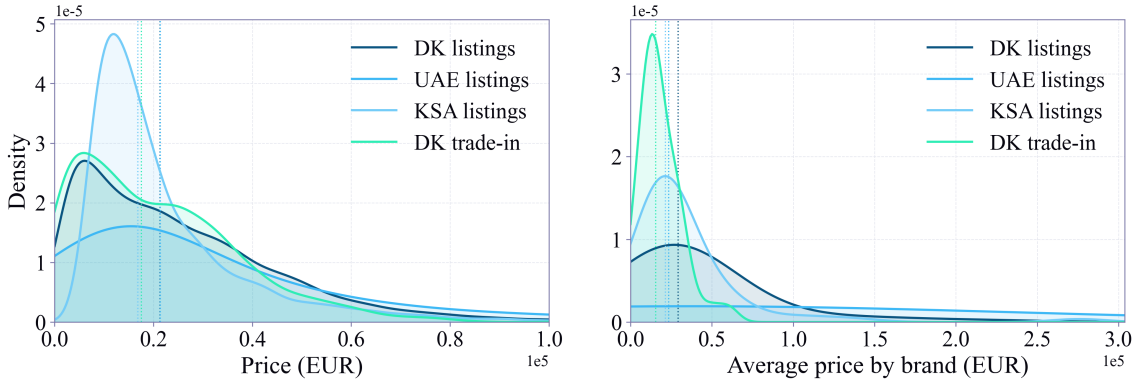


Figure 5.3: Estimated price density across data sources (left) and estimated average-price-by-brand density (right), with dotted lines marking each market's median.

that electric vehicles depreciate more steeply in their early years than fossil-fuel vehicles, while the smaller and more recent hybrid segment yields a noisier curve.

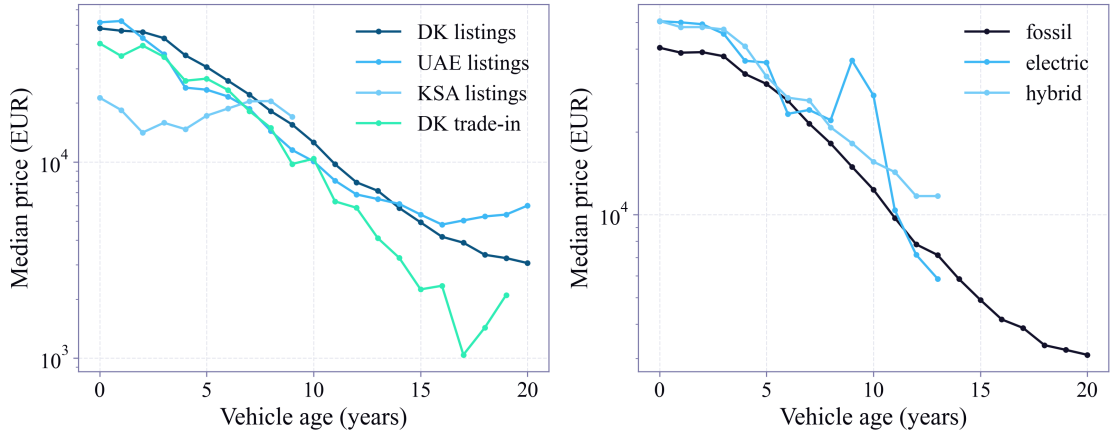


Figure 5.4: Median price as a function of vehicle age, by market (left) and by energy source for the DK listings (right). The vertical axis is logarithmic.

Figure 5.5 summarizes the linear relationships among the numerical attributes. In both domains, price correlates positively with model year and engine power and negatively with mileage, as expected, while engine size and engine power are themselves strongly correlated. The richer trade-in schema exposes additional physically meaningful structure, such as the association between maximum speed and engine power. These dependencies justify treating the attributes jointly rather than in isolation during modeling.

Finally, because the DK listing and DK trade-in datasets share the same market, the relationship between the two valuations can be examined directly. Figure 5.6 overlays the densities of the median listing price and the median trade-in valuation across the 199 brand–model pairs present in both datasets. The trade-in distribution is shifted markedly toward lower prices: the median trade-in-to-listing ratio is 0.63,

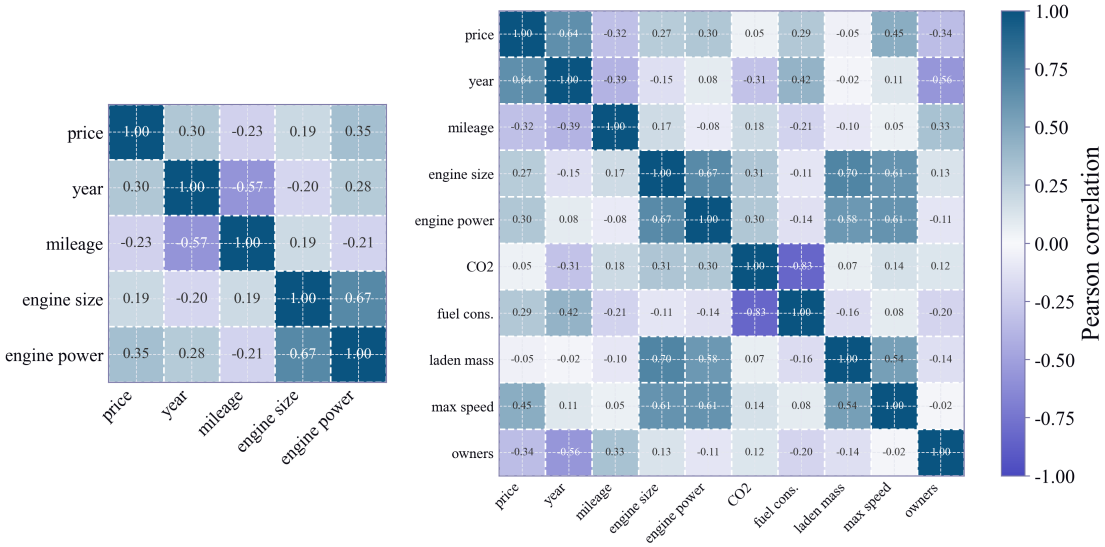


Figure 5.5: Pearson correlation among numerical attributes for the DK listings (left) and the DK trade-in dataset (right).

that is, trade-in valuations are roughly 37% below comparable listings. This gap reflects standard industry practice, where dealerships offer below-market valuations to absorb reconditioning costs, inventory risk, and margin. At the same time, the close tracking of the two distributions indicates that listing prices carry strong predictive signal for trade-in appraisal, motivating the transfer learning approach developed in Section 4.3.

Figure 5.7 shows the most frequent *variant*, *transmission*, and *fuel type* across the listing datasets. The *variant* attribute exhibits scope inconsistency, as it simultaneously represents engine size (e.g., 1.0) and engine characteristics such as turbocharging and direct injection (e.g., tsi). The *transmission* attribute exhibits inconsistent granularity by treating cvt and automatic as distinct transmissions. Finally, the *fuel type* attribute exhibits inconsistent naming by distinguishing between petrol and gasoline as separate fuel types. It also presents scope inconsistency by classifying energy sources such as hybrid and electric under the same *fuel type* attribute.

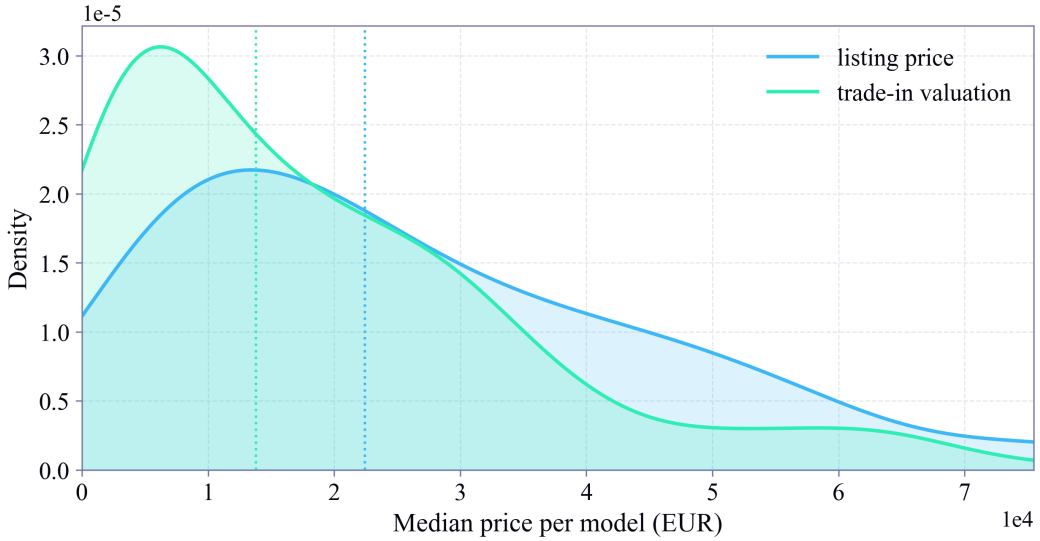


Figure 5.6: Estimated density of the median listing price (sky) and median trade-in valuation (mint) across the 199 brand–model pairs shared between the DK listing and DK trade-in datasets; dotted lines mark each median. The trade-in distribution sits markedly lower, with a median trade-in-to-listing ratio of 0.63.

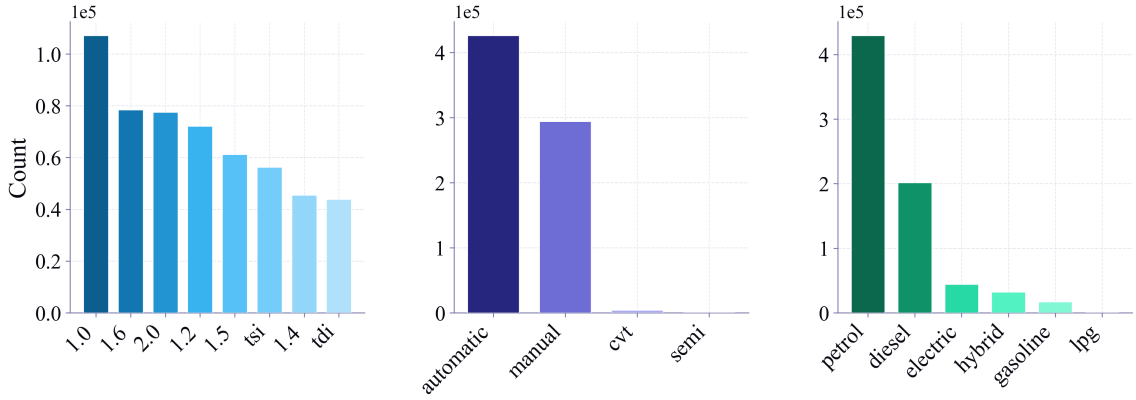


Figure 5.7: Most frequent variant, transmission, and fuel type values across listing data.

5.3 Data preparation

Preprocessing. Before normalization, all categorical attributes are converted to lowercase to ensure uniformity. Numerical attributes, such as *engine power* and *engine size*, are standardized to horsepower and liters, respectively. Categorical values are encoded using target encoding. To address the right-skewed price distribution, the target variable is log-transformed. Beyond stabilizing the distribution, this transform turns absolute error into relative error, which prevents the most expensive brands and models from dominating the gradient and aligns the training objective with MAPE (Section 5.7).

Missing values are consistently represented across all datasets, avoiding placeholders like "unknown" or "missing" for categorical attributes and arbitrary values such as -1 or 0 for numerical ones.

The datasets are split using stratified sampling based on *model* to account for significant price variations, with 80% allocated to the training sets and 20% to the test sets, ensuring all reflect real-world brand distributions. Repeated appraisals of the same vehicle, identified by its VIN, are collapsed into a single record so that no vehicle can appear in both the training and the test set, and records whose *model* occurs only once are excluded, as stratification requires at least two instances per stratum.

Normalization. Following the configuration described in Section 5.1, we apply the vehicle normalization framework to the four datasets. Starting from four empty open-domain attribute catalogs $\mathcal{K}_j$, the system normalizes 39,381 unique serialized vehicle representations, generating 181 brands, 1,940 models, 1,760 submodels, and 205 trim levels. Table 5.5 presents the normalized representation of a BMW 6-Series as an example output.

After normalization, all listing datasets conform exactly to the proposed schema of 19 attributes (Table 5.1), while the trade-in dataset retains its additional 16 attributes (3 categorical and 13 numerical).

Imputation. After vehicle normalization, a classic imputation method fills the attributes that the LLM could not resolve. The imputation statistics are derived from the training sets, aggregating across all four datasets. For an instance $\mathbf{x}_i$, let $b(\mathbf{x}_i)$ and $m(\mathbf{x}_i)$ denote its brand and model. We form two nested groups of training records, one sharing both brand and model and one sharing only the brand:

$$\mathcal{G}_{bm}(\mathbf{x}_i) = \{\, i' : b(\mathbf{x}_{i'}) = b(\mathbf{x}_i),\ m(\mathbf{x}_{i'}) = m(\mathbf{x}_i) \,\}, \tag{5.2}$$

$$\mathcal{G}_{b}(\mathbf{x}_i) = \{\, i' : b(\mathbf{x}_{i'}) = b(\mathbf{x}_i) \,\}, \tag{5.3}$$

with $i' \in \{1, \ldots, n_{\text{train}}\}$, where n_{train} is the total number of training instances across all datasets. For attribute j and a group g, let $S_g = \{\, \mathbf{x}_{i',j} : i' \in g,\ \mathbf{x}_{i',j} \neq \text{missing} \,\}$ collect its non-missing values, and let agg_j denote the median for numerical attributes

Table 5.5: Normalized representation for the serialized vehicle representation "bmw 6-series 640i gran coupe excellence coupe automatic petrol".

Attribute	**Value**
Brand	BMW
Model	6-Series
Submodel	640i
Trim level	Excellence
Year	-
Mileage	-
Body type	Coupe
Transmission type	Automatic
Transmission technology	-
Transmission gears	-
Energy source	Fossil
Propulsion system	ICEV
Fuel type	Petrol
Drive type	-
Engine size	3.0
Engine power	320
Engine aspiration	Turbo
Injection type	Direct
Cylinders	6
Price	-

and the mode for categorical ones. Missing values are imputed through a hierarchical fallback that prefers the most specific group with sufficient support:

$$\hat{\mathbf{x}}_{i,j} = \begin{cases} \mathrm{agg}_j(S_{\mathcal{G}_{bm}}) & \text{if } |S_{\mathcal{G}_{bm}}| \geq \tau_j, \\ \mathrm{agg}_j(S_{\mathcal{G}_b}) & \text{if } |S_{\mathcal{G}_{bm}}| < \tau_j \leq |S_{\mathcal{G}_b}|, \\ \mathrm{agg}_j(S_{\text{train}}) & \text{otherwise,} \end{cases} \tag{5.4}$$

where S_{train} collects the non-missing values of attribute j over the entire training set, and τ_j is a minimum group size, set to 1 for numerical attributes and 5 for categorical ones to avoid unreliable estimates from sparsely populated groups. The brand-and-model statistic captures model-specific characteristics, the brand-only statistic provides a coarser estimate when a model is rare, and the global statistic guarantees a value even for previously unseen brand–model combinations.

Mileage is not imputed, as it reflects individual vehicle usage patterns that vary by owner and location rather than by brand or model characteristics. Similarly, the attributes *accident* and *number of owners* are not imputed, as they depend on vehicle history rather than intrinsic vehicle properties.

5.4 Performance metric

Given predictions $\{\hat{y}_i\}_{i=1}^n$ and true values $\{y_i\}_{i=1}^n$, we report three primary metrics for model evaluation: Mean Absolute Percentage Error (MAPE), Median Absolute Percentage Error (MdAPE), and Mean Bias Error (MBE).

$$\text{MAPE} = \frac{100}{n} \sum_{i=1}^{n} \left| \frac{\exp(\hat{y}_i) - y_i}{y_i} \right|, \tag{5.5}$$

$$\text{MdAPE} = 100 \cdot \text{median} \left(\left| \frac{\exp(\hat{y}_i) - y_i}{y_i} \right| \right)_{i=1}^{n}, \tag{5.6}$$

$$\text{MBE} = \frac{1}{n} \sum_{i=1}^{n} \left(\exp(\hat{y}_i) - y_i \right). \tag{5.7}$$

During inference, predictions are exponentiated, and evaluation metrics are computed on the original price scale.

Both MAPE and MdAPE provide intuitive, scale-independent measures of model accuracy, expressed as percentages. MAPE is known to yield disproportionately large errors for small actual values; however, this is not a concern in our case, as the cheapest vehicle in our dataset is priced at 600 €. Still, MAPE can be sensitive to outliers, so MdAPE is included to offer a more robust measure of the typical error. MBE, on the other hand, quantifies systematic bias, revealing whether the model consistently overestimates or underestimates the target variable.

5.5 Hyperparameter tuning

For the listing task (VPP), due to the large scale of the listing datasets and computational constraints, we do not perform a hyperparameter search. Given that listing datasets are two orders of magnitude larger than trade-in data (approximately 100K to 1M vs. 1.2K samples) and we train each dataset-algorithm pair 20 times to compute robust statistics (10 runs with and 10 without normalization), exhaustive hyperparameter tuning would require prohibitive computation time across all algorithms and datasets. We therefore use default model parameters, an approach justified by the fact that larger datasets are generally less sensitive to hyperparameter choices, and our primary focus is on the trade-in task where limited data makes optimal hyperparameters more critical.

For the trade-in task (VTA), the hyperparameter search space described in Table 5.6 was defined through preliminary experiments. Tree depth was limited to 3 to 5 levels after observing validation performance degradation with deeper trees on the small trade-in dataset. Learning rates were bounded at 0.01 to 0.1 based on convergence stability in preliminary runs, and regularization values followed library documentation recommendations for datasets of similar size. Hyperparameter tuning is performed via exhaustive grid search combined with 3-fold cross-validation. For each hyperparameter configuration, the training data is partitioned into three

equally-sized folds, with the model trained on two folds and evaluated on the remaining fold. This process repeats three times so that each fold serves as the validation set exactly once. Performance is measured by averaging the MAPE (Section 5.4) across the three folds. The configuration yielding the lowest average MAPE is selected, and the final model is retrained on the entire training set using these optimal hyperparameters.

Table 5.6: Hyperparameter search space for the evaluated algorithms.

Algorithm	**Hyperparameter**	**Search space**
Random Forest	max_depth	10, 20
	n_estimators	50, 100
XGBoost	max_depth	3, 4, 5
	learning_rate	0.01, 0.05, 0.1
LightGBM	max_depth	3, 4, 5
	learning_rate	0.01, 0.05, 0.1
CatBoost	depth	3, 4, 5
	l2_leaf_reg	1, 3, 5

5.6 Results

Table 5.8 presents the performance of the pricing models in the listing domain across all datasets. Table 5.7 presents the performance of the models in the trade-in domain, where a non-empty "Listing" column indicates that transfer learning was applied from listing models trained with the specified algorithm; an empty entry indicates the model was trained exclusively on trade-in data. Results are expressed as mean (standard deviation), computed over ten independent runs with different random seeds for the train–test splits. Because the same splits are reused across configurations, the differences between them can be tested for statistical significance; Appendix E reports these tests for both domains using paired Wilcoxon signed-rank tests. Figure 5.8 summarizes the trade-in results as the relative MAPE reduction that each algorithm obtains over its own base configuration, separating the differences the tests resolve from those they do not.

Table 5.7: Performance of appraisal models in the trade-in domain. The "Listing" column indicates the listing model used for transfer learning. Bold denotes best performance per metric.

Model		**MAPE**	**MdAPE**	**MBE**
Trade-in	**Listing**			
Random Forest	-	27.44 ± 3.90	16.88 ± 1.53	-1574 ± 285
Random Forest+	-	27.62 ± 3.44	17.21 ± 1.25	-1624 ± 308
Random Forest	CatBoost	23.44 ± 2.57	14.51 ± 1.04	-1133 ± 269
Random Forest+	CatBoost+	23.17 ± 2.65	15.26 ± 1.44	-1111 ± 352
Random Forest	LightGBM	24.05 ± 2.36	15.97 ± 1.25	-1062 ± 327
Random Forest+	LightGBM+	23.74 ± 1.47	15.94 ± 1.24	-1122 ± 375
Random Forest	Random Forest	23.58 ± 1.83	16.30 ± 1.39	-1119 ± 289
Random Forest+	Random Forest+	23.54 ± 2.64	15.89 ± 1.12	-1092 ± 392
Random Forest	XGBoost	24.84 ± 2.73	16.48 ± 1.31	-1087 ± 323
Random Forest+	XGBoost+	24.79 ± 1.75	16.19 ± 1.18	-1133 ± 363
XGBoost	-	25.04 ± 2.90	16.67 ± 1.39	-1043 ± 453
XGBoost+	-	24.89 ± 2.19	15.95 ± 1.15	-1081 ± 476
XGBoost	CatBoost	23.07 ± 2.12	14.62 ± 1.35	-896 ± 315
XGBoost+	CatBoost+	22.70 ± 2.13	15.11 ± 1.50	-1025 ± 279
XGBoost	LightGBM	23.46 ± 2.61	15.22 ± 1.09	-1013 ± 414
XGBoost+	LightGBM+	23.55 ± 1.85	15.80 ± 1.23	-997 ± 332
XGBoost	Random Forest	23.20 ± 1.48	15.69 ± 1.65	-986 ± 394
XGBoost+	Random Forest+	23.86 ± 2.26	15.87 ± 1.31	-1087 ± 393
XGBoost	XGBoost	24.26 ± 2.60	15.46 ± 1.16	-975 ± 318
XGBoost+	XGBoost+	24.26 ± 2.27	15.72 ± 1.06	-1113 ± 329
LightGBM	-	25.13 ± 2.58	16.32 ± 1.36	-969 ± 415
LightGBM+	-	24.73 ± 2.47	16.06 ± 1.47	-964 ± 496
LightGBM	CatBoost	23.11 ± 2.60	14.64 ± 1.28	-892 ± 387
LightGBM+	CatBoost+	22.68 ± 1.69	15.36 ± 1.24	-892 ± 339
LightGBM	LightGBM	22.92 ± 2.07	15.46 ± 1.12	-724 ± 324
LightGBM+	LightGBM+	23.20 ± 1.86	15.91 ± 1.22	-1013 ± 374
LightGBM	Random Forest	22.69 ± 1.72	15.37 ± 1.46	-917 ± 411
LightGBM+	Random Forest+	23.18 ± 2.23	15.58 ± 0.99	-919 ± 426
LightGBM	XGBoost	24.19 ± 2.67	16.00 ± 1.98	-901 ± 494
LightGBM+	XGBoost+	24.47 ± 1.96	15.62 ± 1.08	-957 ± 364
CatBoost	-	23.67 ± 2.25	15.83 ± 1.65	-867 ± 376
CatBoost+	-	23.45 ± 1.79	15.17 ± 1.55	-830 ± 438
CatBoost	CatBoost	22.01 ± 1.83	14.23 ± 1.50	-834 ± 299
CatBoost+	CatBoost+	$\mathbf{21.69 \pm 1.74}$	$\mathbf{14.14 \pm 1.34}$	$\mathbf{-617 \pm 286}$
CatBoost	LightGBM	23.00 ± 2.63	15.20 ± 1.36	-700 ± 307
CatBoost+	LightGBM+	22.24 ± 2.05	15.31 ± 0.95	-746 ± 458
CatBoost	Random Forest	22.37 ± 1.85	14.66 ± 1.17	-643 ± 359
CatBoost+	Random Forest+	22.32 ± 2.25	14.25 ± 0.96	-701 ± 411
CatBoost	XGBoost	23.02 ± 2.25	15.25 ± 1.48	-728 ± 260
CatBoost+	XGBoost+	23.00 ± 1.74	15.56 ± 1.25	-762 ± 369

Table 5.8: Performance of pricing models in the listing domain. Bold denotes best performance per dataset and metric.

Dataset	Model	MAPE	MdAPE	MBE
DK	Random Forest	24.29 ± 0.26	12.27 ± 0.06	$\mathbf{-4882 \pm 127}$
	Random Forest+	$\mathbf{24.00 \pm 0.28}$	$\mathbf{12.07 \pm 0.11}$	-4933 ± 134
	XGBoost	26.03 ± 0.19	14.70 ± 0.16	-4963 ± 224
	XGBoost+	26.16 ± 0.34	14.74 ± 0.19	-4932 ± 128
	LightGBM	26.89 ± 0.32	15.28 ± 0.08	-5222 ± 100
	LightGBM+	26.87 ± 0.27	15.18 ± 0.09	-5199 ± 107
	CatBoost	25.95 ± 0.81	14.60 ± 0.45	-5092 ± 118
	CatBoost+	25.90 ± 0.38	14.51 ± 0.27	-5072 ± 144
UAE	Random Forest	17.17 ± 0.16	12.20 ± 0.08	$\mathbf{-2019 \pm 405}$
	Random Forest+	$\mathbf{16.55 \pm 0.14}$	$\mathbf{11.52 \pm 0.12}$	-2077 ± 394
	XGBoost	18.37 ± 0.25	13.67 ± 0.17	-2311 ± 373
	XGBoost+	17.77 ± 0.16	13.18 ± 0.10	-2168 ± 373
	LightGBM	19.67 ± 0.17	14.85 ± 0.06	-2644 ± 384
	LightGBM+	19.29 ± 0.14	14.44 ± 0.12	-2611 ± 425
	CatBoost	18.04 ± 0.24	13.45 ± 0.19	-3471 ± 350
	CatBoost+	17.47 ± 0.29	13.00 ± 0.24	-3225 ± 431
KSA	Random Forest	8.85 ± 0.11	6.45 ± 0.12	-368 ± 91
	Random Forest+	8.30 ± 0.13	$\mathbf{5.95 \pm 0.17}$	-343 ± 87
	XGBoost	8.60 ± 0.10	6.48 ± 0.15	-222 ± 118
	XGBoost+	$\mathbf{8.08 \pm 0.17}$	6.09 ± 0.21	$\mathbf{-191 \pm 89}$
	LightGBM	8.74 ± 0.10	6.60 ± 0.13	-273 ± 81
	LightGBM+	8.30 ± 0.10	6.31 ± 0.12	-246 ± 82
	CatBoost	8.91 ± 0.18	6.76 ± 0.18	-378 ± 115
	CatBoost+	8.14 ± 0.24	6.12 ± 0.19	-305 ± 96

5.7 Error analysis

The results of Section 5.6 are aggregate. Because a trade-in request may arrive for any vehicle, it is equally relevant to know how the error distributes across the vehicle population, and whether the imbalance of that population penalizes the segments that are least represented in the data. This section examines both questions on the best configuration of Table 5.7, the CatBoost+ trade-in model fed by the CatBoost+ listing models, reusing the ten seeded splits of Section 5.1. Test predictions are pooled across the ten splits, so every reported value rests on the same evidence as the aggregate results.

Performance by brand and model Table 5.9 reports the error for the brands that account for at least 1% of the test records, which together cover 98.3% of them. Performance varies substantially across brands: MAPE ranges from 10.44% for Mazda to 36.01% for Kia, and MdAPE from 6.16% for Citroën to 32.14% for Suzuki. The dispersion is comparable at model level, where MAPE ranges from 7.6% to 59.5% across the 29 vehicle models with enough test predictions to be measured.

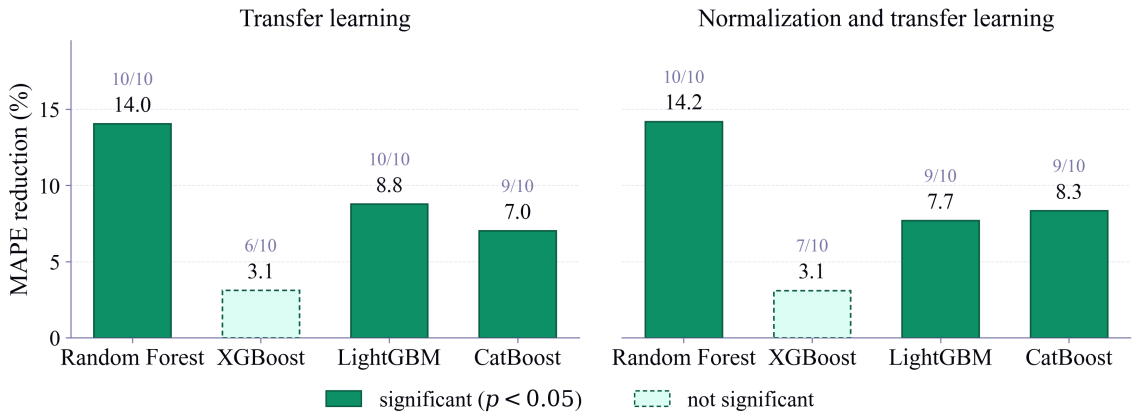


Figure 5.8: Relative MAPE reduction of each trade-in algorithm over its own base configuration (neither normalization nor transfer), with transfer learning alone (left) and with normalization and transfer learning combined (right). Each algorithm transfers from the listing models trained with that same algorithm. A solid bar denotes a reduction that a paired two-sided Wilcoxon signed-rank test over the ten seeded splits resolves as significant ($p < 0.05$); a dashed outline denotes one it does not. The fraction above each bar is the number of splits on which the configuration improves over its base. Every algorithm reduces MAPE; for XGBoost the reduction is the smallest of the four and the test does not resolve it.

Part of this spread reflects a few large errors rather than a systematic offset, and the two metrics separate the two cases. Ford is the clearest example, with a MAPE of 34.55% against a MdAPE of 14.47%: the error on a typical Ford is close to the one on a typical vehicle, while a minority of records are badly missed. Skoda shows the opposite profile, with the two metrics close together at 23.05% and 20.15%, indicating an error that is moderate but consistent.

The natural hypothesis is that the least represented brands are the worst served. The data does not support it. Across the brands of Table 5.9, the rank correlation between support and MAPE is +0.26 ($p = 0.29$), not significant and with the sign opposite to the expected one; at model level it is +0.05 ($p = 0.82$). The left panel of Figure 5.9 makes this visible. The cloud is flat, and brands separated by more than an order of magnitude in support occupy the same range of error. Mercedes, which alone contributes 35.5% of the test records, is appraised with a MAPE of 17.80%, but so are Volvo at 4.2% support, Nissan at 1.3% and Mazda at 1.3%, with 16.92%, 17.71% and 10.44% respectively. Conversely, Audi and BMW are among the five most frequent brands and sit above the overall error.

What does track the error is the price level of the segment. MAPE falls monotonically from 36.45% in the cheapest quartile of the test set to 22.28%, 14.80% and 12.91% in the following ones. This gradient is partly structural, since a given absolute error is a larger fraction of a cheap vehicle and MAPE is a relative metric, and partly substantive, as older and cheaper vehicles carry more condition-dependent

and idiosyncratic value, which Section 6.1 identifies as the component the tabular features cannot capture. It also accounts for the one aggregate gap that does appear between frequent and infrequent brands: the five most frequent brands reach a MAPE of 20.30% against 24.24% for the remaining ones, but their mean prices are 26.5K € and 15.7K €. Once price is held fixed the gap closes. Within each price quartile the two groups are indistinguishable, as the right panel of Figure 5.9 shows, except in the second quartile, where the frequent brands are in fact the worse of the two. In this dataset the support of a segment therefore does not determine how well it is appraised; its price level does. For deployment this suggests organizing error monitoring by price band rather than by brand popularity, with the cheapest segment as the one requiring the widest tolerance.

Table 5.9: Trade-in appraisal error by vehicle brand, for the best configuration (CatBoost+ trade-in model with CatBoost+ listing models). Brands accounting for at least 1% of the test records are listed individually and the remainder are pooled. *Share* is the fraction of test records contributed by the brand. Values are computed over the test predictions pooled across the ten seeded splits, so the aggregate MdAPE differs marginally from Table 5.7, which averages the per-split value.

Brand	**Share (%)**	**Mean price** (K€)	**MAPE**	**MdAPE**	**MBE (€)**
Mercedes	35.5	32.9	17.80	11.36	−1079
Volkswagen	9.6	19.2	18.98	15.39	−495
Peugeot	7.4	12.9	22.71	16.87	−55
BMW	6.5	22.5	26.09	18.40	−1526
Audi	5.6	21.1	28.41	19.81	−257
Ford	4.7	14.8	34.55	14.47	1213
Volvo	4.2	30.3	16.92	13.69	−2620
Toyota	3.5	15.9	22.99	15.26	−1127
Renault	3.5	9.4	30.34	16.32	131
Citroën	2.8	12.0	20.08	6.16	134
Skoda	2.2	19.5	23.05	20.15	−1115
Hyundai	2.2	8.8	23.54	18.53	990
Kia	2.0	11.2	36.01	16.88	941
Tesla	1.7	28.3	12.66	10.67	425
Opel	1.7	8.2	18.36	16.08	−76
Nissan	1.3	15.1	17.71	11.65	−1562
Suzuki	1.3	3.1	34.88	32.14	261
Mazda	1.3	18.5	10.44	9.41	−232
Fiat	1.1	4.2	13.93	13.15	−265
Other brands (8)	1.7	19.4	30.72	29.80	918
All	100.0	22.7	21.69	14.22	−617

Imbalanced categorical distribution The distribution of brands and models in the trade-in domain is strongly skewed, as Figures 5.1 and 5.2 show, with Mercedes

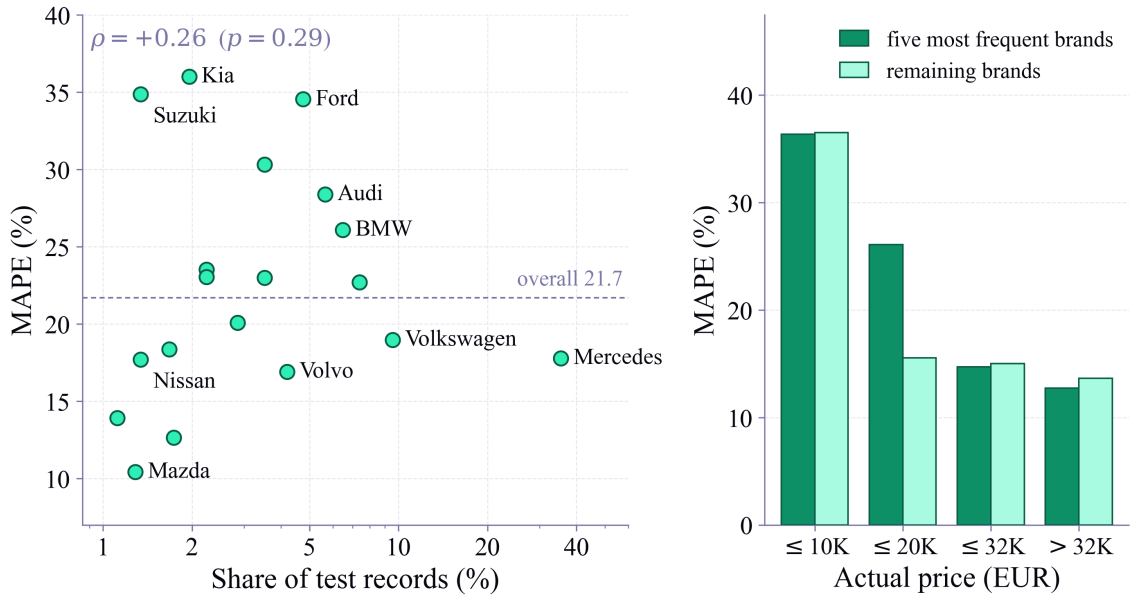


Figure 5.9: Trade-in appraisal error of the best configuration. Left: MAPE of each brand against the share of test records it contributes, on a logarithmic scale; the dashed line marks the overall MAPE and ρ is the rank correlation between the two axes. Labelled points are the brands discussed in the text, and Table 5.9 lists every value. Right: MAPE by price quartile, separating the five most frequent brands from the remaining ones. The aggregate gap between the two groups is a price-mix effect and closes within quartiles.

alone accounting for roughly a third of the records. This skew is not an artifact of data collection but a property of the market the system serves, since the principal Danish client of the platform is a dealership specialized in Mercedes vehicles. The distribution observed in training is therefore the distribution the system operates under in production, and re-weighting it toward uniform brand representation would optimize a population that does not occur at inference time. No re-sampling or re-weighting was consequently applied in the experiments of Section 5.6.

Two distinct effects nevertheless follow from an imbalanced categorical distribution, and both are addressed elsewhere in the pipeline. The first is an imbalance in the scale of the target: because expensive brands are also the ones with large absolute prices, they would dominate a squared-error gradient computed on the price scale. The log transform of Section 5.3 removes this effect by turning absolute error into relative error, which additionally aligns the training objective with MAPE. The second is an imbalance in support: categorical statistics estimated from few records are unreliable. The smoothed target encoding of Section 5.3, and the ordered target statistics that CatBoost applies natively, shrink low-support categories toward the global mean in proportion to how little evidence supports them, while the hierarchical imputation of the same section falls back from the brand-and-model group to the brand group and finally to the global statistic whenever a group is too small.

Whether an explicit re-balancing would nevertheless help is an empirical question, and we answer it directly, along the two axes on which such a technique can act. Re-weighting leaves the training set intact and changes how much each record contributes to the loss; resampling leaves the loss intact and changes the training set. We repeat the best configuration on the same ten splits under both, always applying the intervention to the trade-in model alone, the listing models being the same frozen ones used throughout, so that the comparison isolates the effect of re-balancing the target domain. On the weighting side we evaluate inverse-frequency weighting, $w_i \propto 1/n_{b(i)}$, which equalizes the contribution of every brand, and its square-root variant, $w_i \propto 1/\sqrt{n_{b(i)}}$, which only halves the imbalance in the exponent; weights are normalized to unit mean so the scale of the loss is unchanged. Besides MAPE, Table 5.10 reports the three metrics that re-balancing is meant to improve. Two of them summarize the per-brand MAPE, computing one MAPE per brand and reporting its mean, the macro-averaged MAPE, which weights every brand equally regardless of its frequency, and its standard deviation, which measures how uniform performance is across brands. The third is the MAPE restricted to the records outside the five most frequent brands.

Neither weighting scheme improves any of the metrics reported. Full inverse-frequency weighting degrades MAPE from 21.69% to 23.02% ($p = 0.004$, better on only 1 of the 10 splits) and MdAPE from 14.14% to 15.53% ($p = 0.014$); the milder scheme degrades less, roughly in proportion to how much it re-balances. The degradation also reaches the metrics that motivate the technique. The mean per-brand MAPE moves from 23.59% to 24.42% and the tail MAPE from 24.22% to 24.86%, neither significantly, but neither in the intended direction. Re-weighting does not create information about infrequent brands; it discards effective sample size from the frequent ones, and on a trade-in training set this small that cost is not recovered. Re-weighting does narrow the gap between the five most frequent brands and the rest, from 3.92 to 2.87 percentage points under full inverse weighting ($p = 0.004$), but it does so by levelling down: neither segment improves, the error rising by 1.69 points on the frequent brands and by 0.64 on the remaining ones.

Resampling reaches the same conclusion. Here each brand contributes a target number of records, drawn with replacement when the target exceeds what is available, so that infrequent brands are duplicated, and without replacement otherwise, so that frequent brands are subsampled. Three schemes span the range, and they correspond to the reference points that the long-tailed learning literature defines through $p_b \propto n_b^q$, here applied to brand frequency rather than to class labels. Square-root sampling ($q = 0.5$) leaves brand counts proportional to $\sqrt{n_b}$, reproducing the imbalance that $1/\sqrt{n_b}$ weighting induces; random oversampling to a balanced distribution ($q = 0$) raises every brand to the size of the largest, matching $1/n_b$ weighting; and random undersampling caps every brand at the median count, the only variant that discards records rather than duplicating them. The unbalanced model of the previous paragraphs is instance-balanced sampling ($q = 1$), the natural distribution. None of the three improves any metric either. Duplicating up to the largest brand

degrades MAPE to 23.31% and MdAPE to 16.86% ($p = 0.002$ for both), the partial scheme is neutral (21.84%, $p = 0.43$), and discarding down to the median is the worst of every configuration tried, at 24.91% MAPE and 18.56% MdAPE.

Neither family dominates the other: the partial scheme is marginally better under resampling and the full one marginally better under re-weighting, differences of about 0.3 percentage points against a split-to-split spread five times larger. Their agreement matters more than their ranking, since it shows the conclusion does not depend on whether the distribution is rebalanced through the loss or through the sample. The dispersion of the per-brand error behaves the same way. No scheme moves it significantly; the largest reduction comes from the mildest scheme and is not resolved by the test (13.25 to 11.92 points), and the scheme that flattens the brand distribution most aggressively leaves it slightly higher than the unbalanced model. Making the distribution of brands uniform does not make performance uniform across them. The six configurations order instead by how much effective sample size each one destroys. The three that balance completely leave the brand distribution equally flat, yet differ by 1.9 percentage points of MAPE, and the worst is the one left with 236 training records instead of 716. The disaggregated analysis above already explained why frequency is the wrong axis to act on, since the error of a segment tracks its price level rather than its frequency.

5.8 Vehicle diversity

As highlighted in Chapter 1, this study uses extensive datasets with a wide diversity of vehicle brands and models, as detailed in Table 5.3. This diversity is crucial given the practical context, where users may request a trade-in for any vehicle. To understand and demonstrate its significance, we conduct an experiment with a 70/30 train-test split using the RF+ model on UAE listings. We select this model because it provides an optimal balance between dataset size and diversity, encompassing 109 brands and 927 models after normalization. Brands are ordered from most to least frequent, and the model is sequentially trained on an increasing number of brands. We evaluate the model's performance on two test sets: the complete test set and a partial test set limited to the brands used in training, allowing us to assess the impact of the diversity of vehicle brands on model generalization. As shown in the left panel of Figure 5.10, the model's performance begins to converge once the 40 most frequent brands, which represent over 90% of the train set, are included during training. The same experiment is conducted to assess the impact of vehicle model diversity. As shown in the right panel, studies that consider 400 or fewer models may report favorable experimental results, but their real-world performance can still be suboptimal.

It is important to distinguish between specificity, as defined by Lessmann and Voß (2017), and diversity. Specificity is a model property that captures modeling depth, whereas diversity is a dataset property that captures data breadth. Unlike the specificity experiments in Lessmann and Voß (2017), we do not vary the degree of

Table 5.10: Effect of re-balancing the brand distribution of the trade-in training set, for the best configuration (CatBoost+ trade-in model with CatBoost+ listing models). Re-weighting changes the contribution of each record to the loss; resampling changes the training set itself. Both apply only to the trade-in model; the listing models are unchanged. n is the resulting number of training records. The last three columns are the ones re-balancing is meant to improve. *Tail MAPE* is restricted to the records outside the five most frequent brands. The *per-brand MAPE* block computes one MAPE per brand and reports the mean and the standard deviation of those values. The mean is the macro-averaged MAPE, which gives every brand the same weight regardless of its frequency, in contrast with the micro-averaged MAPE of the third column, where frequent brands dominate; the standard deviation measures how uniform performance is across brands. Both are computed within each split over the brands with at least three test records. Values are mean (standard deviation) over the ten seeded splits. No scheme improves any metric significantly; a marker (†) denotes a significant *degradation* with respect to the unbalanced model, under a paired two-sided Wilcoxon signed-rank test ($p < 0.05$). MdAPE, omitted here for space, degrades in step with MAPE and is reported in the text.

				Per-brand MAPE	
Scheme	n	**MAPE**	**Tail MAPE**	**Mean (macro)**	**SD**
None	716	$\mathbf{21.69 \pm 1.74}$	$\mathbf{24.22 \pm 2.89}$	$\mathbf{23.59 \pm 2.45}$	13.25 ± 4.16
Re-weighting					
$w_i \propto 1/\sqrt{n_b}$	716	$22.10 \pm 1.94^\dagger$	24.29 ± 3.05	23.68 ± 2.82	11.92 ± 2.84
$w_i \propto 1/n_b$	716	$23.02 \pm 1.67^\dagger$	24.86 ± 3.19	24.42 ± 2.78	12.76 ± 4.45
Resampling					
Partial, duplicating	1,837	21.84 ± 1.47	24.10 ± 2.36	23.00 ± 1.70	12.28 ± 3.47
Full, duplicating	7,210	$23.31 \pm 1.93^\dagger$	$25.87 \pm 3.38^\dagger$	24.34 ± 2.71	13.95 ± 4.97
Full, discarding	236	$24.91 \pm 2.03^\dagger$	$26.16 \pm 3.31^\dagger$	25.43 ± 2.85	13.08 ± 5.84

model specialization; all estimators are trained on the complete training set, making specificity a fixed property of the dataset rather than an experimental variable. The specificity experiment evaluates how model specialization affects performance within a given dataset, whereas the diversity experiment examines how the breadth of the dataset influences model generalization to real-world scenarios. Our findings are therefore supplementary to the specificity results, as each addresses a different dimension of the problem.

5.9 Discussion

The vehicle normalization framework improves VPP performance across datasets and models, with the gains concentrated in the UAE and KSA markets, where the reduction in MAPE is statistically significant for every algorithm (Appendix E). On

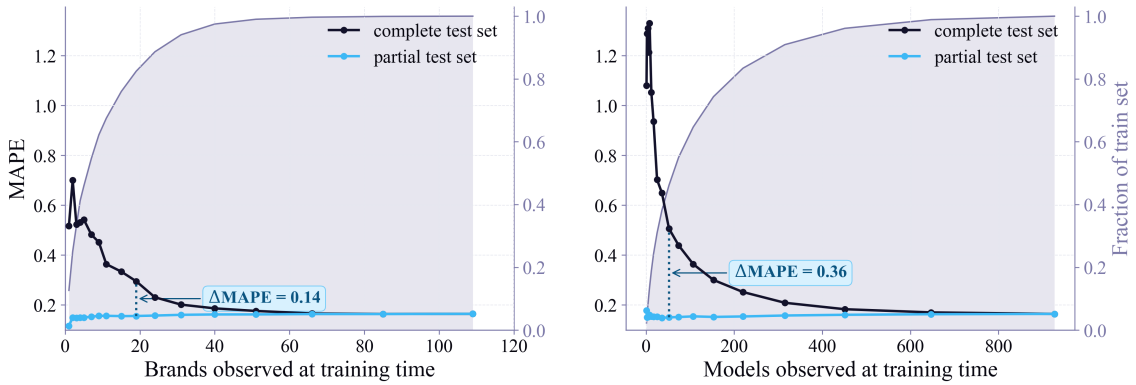


Figure 5.10: Effect of vehicle brand (left) and model (right) diversity. In each panel the two curves show MAPE on the complete and partial test sets (left axis; see legend), and the shaded band shows the fraction of the train set used at training time (right axis). The highlighted dotted line marks the performance gap at $x = 19$ brands and $x = 52$ models, respectively.

the DK listings the effect is negligible, and marginally negative for XGBoost, whose MAPE increases by 0.14 percentage points; this reflects that DK records are already largely complete (Table 5.4) and therefore leave little for normalization to repair. Across the three listing markets, RF+ emerges as the best-performing model on both the DK and UAE datasets, achieving MAPE values of 24.00% and 16.55%, respectively. On the KSA dataset, XGBoost+ achieves the lowest MAPE of 8.08%, closely followed by CatBoost+ at 8.14%. Notably, CatBoost+ consistently ranks as the second-best model across all datasets, demonstrating robust performance regardless of market dynamics. The improvements from normalization are particularly pronounced in the KSA dataset, where XGBoost+ reduces MAPE by 6.0% relative to XGBoost, and CatBoost+ reduces MAPE by 8.6% relative to CatBoost. These gains can be attributed to the KSA dataset's substantial missing data in critical attributes such as body type and engine power (Table 5.4), which the normalization framework effectively addresses through LLM-based extraction and the enhanced imputation enabled by standardized representations.

The transfer learning method improves VTA performance for all trade-in algorithms, as Figure 5.8 summarizes. The reduction in MAPE is statistically significant for Random Forest, LightGBM, and CatBoost, which improve by 14.0%, 8.8%, and 7.0% relative to their own base configuration. XGBoost also improves, from a MAPE of 25.04% to 24.26% with transfer learning and to the same value when normalization is added, but the paired test resolves neither difference ($p = 0.63$ and $p = 0.32$). Two aspects of this result deserve to be made explicit, because a non-significant entry is easily misread as an absence of effect. First, the improvement is present in the sample and consistent in direction: transfer learning lowers MAPE on 6 of the 10 splits and normalization with transfer on 7 of them, and the same configuration reduces the MdAPE of XGBoost from 16.67% to 15.46%, a difference the test does

resolve ($p = 0.01$). What the ten splits fail to establish is therefore the effect on the mean error, not the existence of an effect. Second, the reason lies in the size of the effect rather than in a property of the algorithm: the 3.1% relative gain of XGBoost is the smallest of the four, between two and four times smaller than the others, while its split-to-split variability is comparable to theirs, so it is the configuration for which ten paired splits afford the least statistical power. With ten splits the exact two-sided Wilcoxon signed-rank test reaches $p < 0.05$ only when the improvement is close to uniform, as a single reversal among the two largest per-split differences already pushes the statistic past the critical value, and the smallest attainable p value is 0.002. Non-significance here should accordingly be read as the ten splits being unable to resolve an effect of this size, not as evidence that transfer learning does not benefit XGBoost. The best overall performance is achieved by the configuration with CatBoost+ as both the trade-in and listing model, achieving a MAPE of 21.69% and a MdAPE of 14.14%; this improvement over the non-transfer baseline is statistically significant, although the limited size of the trade-in dataset constrains the power of these tests. This configuration reduces MAPE by 8.4% and MdAPE by 10.7% compared to the CatBoost baseline without normalization or transfer learning (MAPE of 23.67%, MdAPE of 15.83%). As shown in Figure 5.11, the listing price predictions from the three CatBoost+ source models rank among the top three most important features, confirming that transfer learning effectively bridges the gap between listing and trade-in domains. Figure 5.12 further reveals that while the CatBoost+ listing models rely heavily on *brand* and *model* as primary predictors, the trade-in model abstracts away from these high-level identifiers and instead relies on complementary attributes such as *exhaust emission level* and *maximum laden mass*. Feature importances are computed using permutation importance and averaged across 10 models trained on different training splits to ensure stable estimates. These results demonstrate that the combination of vehicle normalization and transfer learning from listing domains provides complementary benefits: normalization enables consistent feature representations across heterogeneous data sources, while the listing price predictions capture market signals that inform trade-in valuations.

The MBE metric reveals a consistent pattern of systematic underestimation across both VPP and VTA tasks. All models exhibit negative MBE values, indicating that predictions fall below actual prices on average. Notably, the best-performing models also tend to achieve the lowest MBE. In the listing domain, XGBoost+ on KSA attains an MBE of -190.63 €, while in the trade-in domain, the fully normalized transfer learning configuration exhibits an MBE of -616.83 €. This underpricing tendency is more pronounced in trade-in predictions, which aligns with industry practices where dealerships systematically offer below-market values to account for reconditioning costs, inventory risks, and profit margins. The models effectively learn these patterns from the training data, which reflects historical valuations made by automotive industry experts. As illustrated in the left panel of Figure 5.13, the XGBoost+ model on KSA stays close to calibration with a tight error band, while the right panel shows the same under-pricing tendency in the trade-in domain but

markedly more pronounced and with a far wider error spread, reflecting its smaller and more heterogeneous sample. From an industry perspective, this conservative bias may be acceptable or even desirable, as it reduces the risk of overvaluing vehicles that may require unforeseen repairs or face market depreciation. However, practitioners should monitor MBE to ensure that algorithmic predictions do not systematically disadvantage certain customer segments or vehicle categories. Section 5.7 disaggregates the error along precisely these axes and indicates that the relevant one is the price of the vehicle rather than the popularity of its brand.

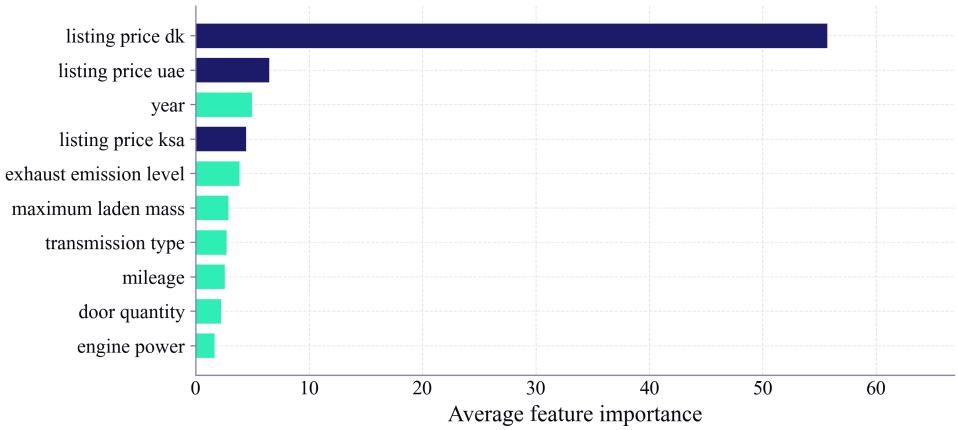


Figure 5.11: Top 10 most important features for CatBoost+ (DK trade-in).

Taken together, these results allow us to revisit the hypotheses posed in Section 1.1. The main hypothesis is supported: transferring knowledge from the listing domain improves VTA performance, with the best configuration achieving a statistically significant 8.4% relative reduction in MAPE over the non-transfer baseline. The first sub-hypothesis, that listing data contains transferable patterns relevant to VTA, is likewise supported, as the listing-price predictions rank among the most important features of the trade-in model (Figure 5.12) and the transfer gain is statistically significant for three of the four algorithms. The second sub-hypothesis, that standardization increases the effectiveness of transfer learning, is not supported on the trade-in metric. Transfer operates over the minimal common schema shared by all data sources (Section 5.2) and yields significant gains for most algorithms regardless of normalization, and once transfer is in place normalization adds no significant marginal improvement (Appendix E). Its value lies elsewhere: normalization significantly improves vehicle price prediction in the listing markets where vehicle representations are most heterogeneous and incomplete, namely UAE and KSA, whereas on the larger but already complete DK listings its effect is negligible. Moreover, in the general case of integrating genuinely heterogeneous sources (Tables 1.1 and 1.2), normalization is what makes a shared representation attainable in the first place. Standardization is therefore better characterized as improving the source models and enabling data integration than as amplifying the transfer effect itself.

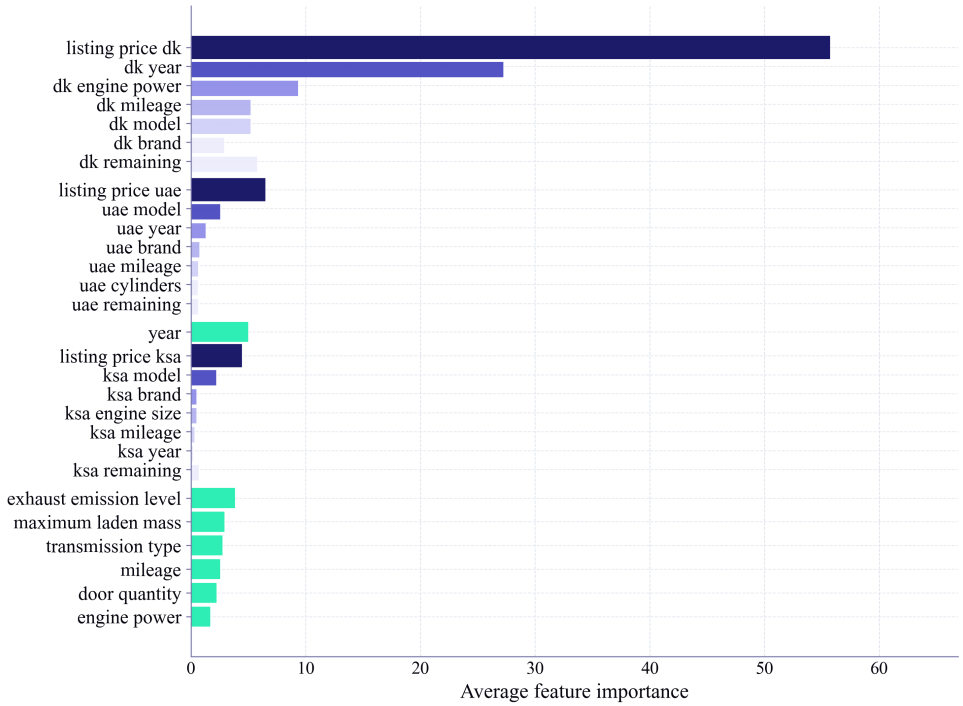


Figure 5.12: Top 10 most important features for CatBoost+ (DK trade-in) and top 5 most important features from each CatBoost+ listing model (DK, UAE, KSA).

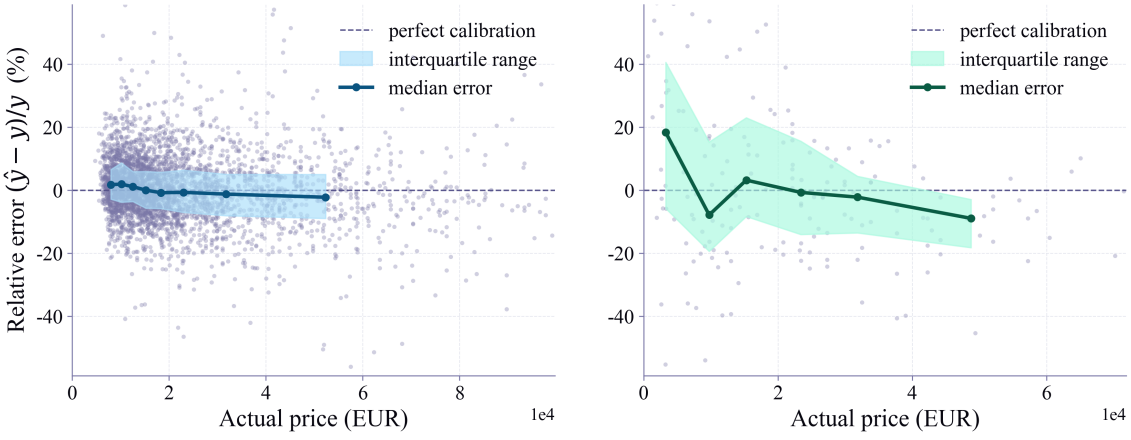


Figure 5.13: Prediction bias as relative error $(\hat{y} - y)/y$ versus actual price, for XGBoost+ on KSA listings (left) and the CatBoost+/CatBoost+ transfer configuration on DK trade-in (right). The line is the median error per price bin and the band its interquartile range. Both panels show systematic under-pricing; it is larger and noisier in the trade-in domain, whose sample is far smaller.

Chapter 6

Conclusions

In this study, we introduced a novel machine learning system for VTA supported by a vehicle normalization framework and a transfer learning approach. Experimental results demonstrate that vehicle normalization significantly improves vehicle price prediction in the markets where vehicle data is most heterogeneous and incomplete, namely UAE and KSA, while enabling robust data integration on which transfer learning depends. The transfer learning approach, incorporating listing price predictions as features, improves VTA performance for all evaluated algorithms, with the best configuration achieving a statistically significant 8.4% relative improvement in MAPE compared to non-transfer baselines. Furthermore, our experiments on brand and model diversity highlight that diverse datasets with many vehicle brands and models are essential to assess performance in realistic conditions, as training and evaluating on a dataset with limited diversity (fewer than 40 brands or 400 models) may produce over-optimistic results that fail to reflect real-world applications.

6.1 Limitations

This study presents three limitations:

1. The scarcity of trade-in data, which amounts to around 1.2K records spanning 56 brands and 250 models (Table 5.3), is consequential in two ways. First, it restricts the methodology to classical ensemble algorithms, as deep learning approaches typically require $O(10^4)$ samples to outperform gradient boosting methods on tabular regression tasks. Second, it limits the diversity of the dataset: since the model count falls below the threshold identified in Section 5.8, the reported performance may be optimistic for the long tail of vehicle models absent from the data, and validation on a larger and more diverse trade-in dataset remains an important direction for future work.

2. The exclusive reliance on tabular features prevents the modeling of condition-dependent factors, including paint quality, mechanical wear, and body damage, which constitute substantial drivers of trade-in value variance. The coarse

binary representation of accident history cannot capture the spectrum from minor cosmetic damage to structural compromise. More broadly, the trade-in market exhibits variability driven by factors that remain fundamentally difficult to quantify, including dealership-specific considerations such as inventory levels and sales targets, macroeconomic conditions, negotiation dynamics, and seasonal patterns. Section 5.7 localizes where this limitation is paid: relative error is concentrated in the cheapest quartile of the trade-in data, at a MAPE of 36.45% against 12.91% in the dearest one, which is where condition accounts for the largest share of value and where a given absolute deviation represents the largest percentage. Despite these irreducible complexities, our system achieves a MdAPE of 14.14%, suggesting that the supervised signals provided by automotive industry experts implicitly encode much of this contextual knowledge through their valuations. This highlights the value of learning from expert assessments, as their appraisals serve as a proxy for latent market factors that are otherwise difficult to capture explicitly.

3. The consistent negative MBE values across all models indicate systematic underpricing relative to actual values. While some underestimation in trade-in models may reflect standard industry practice, as dealerships set trade-in offers below market value to cover reconditioning costs, inventory risks, and profit margins, the presence of similar bias in listing price models suggests additional factors such as model conservatism or data skewness. Organizations deploying such systems should regularly audit MBE across vehicle segments and price ranges to detect and correct systematic biases.

6.2 Future work

To advance this research, several directions merit exploration. First, future studies should investigate the integration of condition-related information. While the existing literature has experimented with vehicle images to capture details such as paint quality and damage, the benefits of these approaches have not been consistently reported. Incorporating proxy information, such as damage records or detailed accident history, could offer a more reliable method for assessing vehicle condition. Second, a promising avenue involves augmenting vehicle-level features with contextual attributes that capture macroeconomic and market dynamics. As highlighted in our limitations, trade-in valuations are influenced by factors beyond intrinsic vehicle characteristics, including interest rates, fuel prices, consumer confidence indices, regional demand patterns, and seasonal cycles. Incorporating such contextual features would enable easier adaptation to evolving market conditions. This could include dealership-level attributes such as inventory turnover rates or regional market indicators such as average days-on-lot for comparable vehicles. Third, the vehicle normalization framework proposed in this work has implications that extend beyond VTA. Automotive e-commerce platforms routinely aggregate inventory

from multiple sources, each with its own schema and naming conventions. Applying the framework to this broader data integration challenge could support a range of downstream tasks, including cross-market search, inventory deduplication, and demand forecasting, contributing to more coherent and interoperable digital automotive ecosystems. Fourth, the framework's reliance on schema-guided extraction, retrieval over self-growing catalogs, and entity resolution is not specific to vehicles, and could be extended to product normalization more broadly. Online marketplaces aggregate heterogeneous product listings that exhibit the same naming, granularity, and scope inconsistencies observed in vehicle data, where the same item appears under divergent strings, attributes are described at varying levels of detail, and single fields conflate distinct dimensions. Because the normalized schema is user-defined and the per-attribute catalogs grow as new entities are encountered, adapting the framework to other product categories would require redefining the target attributes rather than redesigning the pipeline, opening a path toward general-purpose product attribute normalization for e-commerce. Finally, several of the comparisons in Appendix E are limited by the number of paired splits rather than by the size of the effect they measure. Extending the evaluation beyond ten seeded splits would increase the resolution of the paired tests and settle whether the smaller transfer learning gains, such as the one observed for XGBoost, are genuine or attributable to split-to-split variation.

Bibliography

Auto Care Association, 2026. ACES and PIES data standards. URL: https://www.autocare.org/data-standards. ACES 5.0 and PIES 8.0 released March 2026.

Bengio, Y., Courville, A., Vincent, P., 2013. Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence 35, 1798–1828. doi:https://doi.org/10.1109/TPAMI.2013.50.

Bergmann, S., Feuerriegel, S., 2025. Machine learning for predicting used car resale prices using granular vehicle equipment information. Expert Systems with Applications 263, 125640. doi:https://doi.org/10.1016/j.eswa.2024.125640.

Besner, P., 2024a. Case study: Ai-driven vehicle appraisal and inventory management in the north american automotive retail market. SSRN Electronic Journal doi:https://doi.org/10.2139/ssrn.5144707.

Besner, P., 2024b. Predictive trade-in acquisition modeling for automotive retail using multivariate ai targeting. SSRN Electronic Journal doi:https://doi.org/10.2139/ssrn.5211849.

Breiman, L., 1996. Bagging predictors. Mach. Learn. 24, 123–140. doi:https://doi.org/10.1023/A:1018054314350.

Breiman, L., 2001. Random forests. Mach. Learn. 45, 5–32. doi:https://doi.org/10.1023/A:1010933404324.

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J., 1984. Classification and Regression Trees. 1st ed., Chapman and Hall/CRC. doi:https://doi.org/10.1201/9781315139470.

Brinkmann, A., Baumann, R., Bizer, C., 2024. Using LLMs for the extraction and normalization of product attribute values, in: Proc. ADBIS, Springer. pp. 217–230.

Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D., 2020. Language models are

few-shot learners, in: Proceedings of the 34th International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA.

Carvalho, G., Viana, D., Barbosa, L., Ren, T.I., 2023. Combining structured and unstructured data using co-attention for car price prediction. Procedia Computer Science 222, 646–655. doi:https://doi.org/10.1016/j.procs.2023.08.202.

Chen, T., Guestrin, C., 2016. Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, New York, NY, USA. p. 785–794. doi:https://doi.org/10.1145/2939672.2939785.

COVESA, 2025a. Vehicle data model requirements and proposed solutions. URL: https://wiki.covesa.global/display/WIK4/Vehicle+Data+Model+Requirements+and+Proposed+Solutions. accessed: 2025-04-06.

COVESA, 2025b. Vehicle signal specification (VSS). URL: https://covesa.global/vehicle-signal-specification/. accessed: 2025-04-06.

Cowie, J., Lehnert, W., 1996. Information extraction. Communications of the ACM 39, 80–91. doi:https://doi.org/10.1145/234173.234209.

Dutulescu, A., Catruna, A., Ruseti, S., Iorga, D., Ghita, V., Neagu, L.M., Dascalu, M., 2023. Car price quotes driven by data-comprehensive predictions grounded in deep learning techniques. Electronics 12. doi:https://doi.org/10.3390/electronics12143083.

Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S., 2007. Duplicate record detection: A survey. IEEE Transactions on Knowledge and Data Engineering 19, 1–16. doi:https://doi.org/10.1109/TKDE.2007.250581.

Fayyaz, I., Ali, G.G.M.N., Khairunnesa, S.S., 2025. Advanced feature engineering and machine learning techniques for high accurate price prediction of heterogeneous pre-own cars. Vehicles 7. doi:https://doi.org/10.3390/vehicles7030094.

Fellegi, I.P., Sunter, A.B., 1969. A theory for record linkage. Journal of the American Statistical Association 64, 1183–1210. doi:https://doi.org/10.1080/01621459.1969.10501049.

Friedman, J., 2000. Greedy function approximation: A gradient boosting machine. The Annals of Statistics 29. doi:https://doi.org/10.1214/aos/1013203451.

Guo, C., Berkhahn, F., 2016. Entity embeddings of categorical variables. CoRR abs/1604.06737. doi:https://doi.org/10.48550/arXiv.1604.06737.

Hu, S., Zhu, S.X., Fu, K., 2025. The manufacturer's resale strategy for trade-ins. European Journal of Operational Research 323, 583–598. doi:https://doi.org/10.1016/j.ejor.2024.12.017.

International Organization for Standardization, 2009. Road vehicles – vehicle identification number (vin) – content and structure. ISO 3779:2009. URL: https://www.iso.org/standard/52200.html. accessed: December 8, 2025.

Israeli, A., Scott-Morton, F., Silva-Risso, J., Zettelmeyer, F., 2021. How market power affects dynamic pricing: Evidence from inventory fluctuations at car dealerships. Management Science 68, 895–916. doi:https://doi.org/10.1287/mnsc.2021.3967.

Izacard, G., Caron, M., Hosseini, L., Riedel, S., Bojanowski, P., Joulin, A., Grave, E., 2022. Unsupervised dense information retrieval with contrastive learning. Transactions on Machine Learning Research .

Johnson, J., Douze, M., Jégou, H., 2021. Billion-scale similarity search with GPUs. IEEE Transactions on Big Data 7, 535–547.

Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., tau Yih, W., 2020. Dense passage retrieval for open-domain question answering, in: Proc. EMNLP, pp. 6769–6781.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y., 2017. Lightgbm: a highly efficient gradient boosting decision tree, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA. p. 3149–3157.

Khotimah, E.K., Swasono, D.I., Fajarianto, G.W., 2025. Forecasting used car prices using machine learning. IT Journal Research and Development 9, 123–139. doi:https://doi.org/10.25299/itjrd.2025.18031.

Kumar, G., Kumar Pandey, S., Yadav, D.P., Singh, K.U., Singh, T., Kumar, A., 2023. Artificial intelligence-enabled regression model for used car price prediction, in: 2023 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES), pp. 88–94. doi:https://doi.org/10.1109/CISES58720.2023.10183593.

Le, N.L., Abel, M.H., Gouspillou, P., 2022. Towards an ontology-based recommender system for the vehicle sales area, in: Proc. ICDLAIR, Springer. pp. 126–136. doi:10.1007/978-3-030-98531-8_13.

Lessmann, S., Voß, S., 2017. Car resale price forecasting: The impact of regression method, private information, and heterogeneity on forecast accuracy. International Journal of Forecasting 33, 864–877. doi:https://doi.org/10.1016/j.ijforecast.2017.04.003.

Levy, O., Goldberg, Y., Dagan, I., 2015. Improving distributional similarity with lessons learned from word embeddings. Transactions of the Association for Computational Linguistics 3, 211–225. doi:https://doi.org/10.1162/tacl_a_00134.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., Riedel, S., Kiela, D., 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks, in: Proceedings of the 34th International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA.

Li, Y., Li, J., Suhara, Y., Doan, A., Tan, W.C., 2020. Deep entity matching with pre-trained language models. Proc. VLDB Endowment 14, 50–60.

Lin, X., Wang, L., 2026. A stacked fusion model for used car price prediction, in: Proceedings of the 2025 2nd International Conference on Digital Economy and Computer Science, Association for Computing Machinery, New York, NY, USA. p. 290–296. doi:https://doi.org/10.1145/3785706.3785752.

Madhusudhanan, K., Behrens, G., Stubbemann, M., Schmidt-Thieme, L., 2024. Probsaint: Probabilistic tabular regression for used car pricing, in: 2024 IEEE International Conference on Big Data (BigData), pp. 2179–2187. doi:https://doi.org/10.1109/BigData62323.2024.10825740.

Malkov, Y.A., Yashunin, D.A., 2020. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE Transactions on Pattern Analysis and Machine Intelligence 42, 824–836. doi:https://doi.org/10.1109/TPAMI.2018.2889473.

Manning, C.D., Raghavan, P., Schütze, H., 2008. Introduction to Information Retrieval. Cambridge University Press, Cambridge, UK.

Mhamedi, A., Mghari, M., Hibaoui, A.E., 2026. Semantic feature engineering: Predicting used car prices by modeling multi-criteria human decisions with llms. SN Operations Research Forum 7, 1–15. doi:https://doi.org/10.1007/s43069-025-00600-3.

Nandan, M., Ghosh, D., 2023. Pre-owned car price prediction by employing machine learning techniques. Journal of Decision Analytics and Intelligent Computing 3, 167–184. doi:https://doi.org/10.31181/jdaic10008102023n.

Narayan, A., Chami, I., Orr, L., Ré, C., 2022. Can foundation models wrangle your data? Proc. VLDB Endowment 16, 738–746.

National Highway Traffic Safety Administration, 2025. Product information catalog and vehicle listing (vPIC). URL: https://vpic.nhtsa.dot.gov/. accessed: 2025-04-06.

Neelakantan, A., Xu, T., Puri, R., Radford, A., Han, J.M., Tworek, J., Yuan, Q., Tezak, N., Kim, J.W., Hallacy, C., Heidecke, J., Shyam, P., Power, B., Nekoul, T.E., Sastry, G., Krueger, G., Schnurr, D., Such, F.P., Hsu, K., Thompson, M., Khan, T., Sherbakov, T., Jang, J., Welinder, P., Weng, L., 2022. Text and code embeddings by contrastive pre-training. arXiv preprint arXiv:2201.10005 .

Pan, S.J., Yang, Q., 2010. A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering 22, 1345–1359. doi:https://doi.org/110.1109/TKDE.2009.191.

Peeters, R., Steiner, R., Bizer, C., 2025. Entity matching using large language models, in: Proc. EDBT, pp. 529–541.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A.V., Gulin, A., 2018. Catboost: unbiased boosting with categorical features, in: Proceedings of the 32nd International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA. p. 6639–6649.

Quezada, D., Valle, C., Ñanculef, R., 2026. Vehicle data normalization with retrieval-augmented LLMs, in: Proc. Latin American Computing Conference (CLEI), IEEE.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., 2019. Language models are unsupervised multitask learners. OpenAI Technical Report .

Reimers, N., Gurevych, I., 2019. Sentence-BERT: Sentence embeddings using siamese BERT-networks, in: Proc. EMNLP-IJCNLP, pp. 3982–3992.

Sainz, O., García-Ferrero, I., Agerri, R., de Lacalle, O.L., Rigau, G., Agirre, E., 2024. GoLLIE: Annotation guidelines improve zero-shot information-extraction, in: Proc. ICLR.

Salton, G., Wong, A., Yang, C.S., 1975. A vector space model for automatic indexing. Communications of the ACM 18, 613–620. doi:https://doi.org/10.1145/361219.361220.

Sarawagi, S., 2008. Information extraction. Foundations and Trends in Databases 1, 261–377. doi:https://doi.org/10.1561/1900000003.

Somepalli, G., Goldblum, M., Schwarzschild, A., Bruss, C.B., Goldstein, T., 2021. SAINT: improved neural networks for tabular data via row attention and contrastive pre-training. CoRR abs/2106.01342. doi:https://doi.org/10.48550/arXiv.2106.01342.

Tabarnoust, E., Mghari, M., Zaz, Y., 2025. Moroccan used cars dataset: Insights into the used car market. Data in Brief 63, 112087. doi:https://doi.org/10.1016/j.dib.2025.112087.

Tamang, L., Bouadjenek, M.R., Dazeley, R., Aryal, S., 2025. Handling out-of-distribution data: A survey. IEEE Transactions on Knowledge and Data Engineering 37, 5948–5966. doi:https://doi.org/10.1109/TKDE.2025.3592614.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA. pp. 6000–6010.

W3C Automotive Ontology Community Group, 2024. Automotive extension—Schema.org. URL: https://schema.org/docs/automotive.html. accessed: 2025-04-06.

Wang, Q., Yang, L., Kanagal, B., Sanghai, S., Sivakumar, D., Shu, B., Yu, Z., Elsas, J., 2020. Learning to extract attribute value from product via question answering: A multi-task approach, in: Proc. ACM SIGKDD, pp. 47–55.

Wang, T., Lin, H., Han, X., Sun, L., Chen, X., 2025. Match, compare, or select? An investigation of large language models for entity matching, in: Proc. COLING, pp. 96–109.

Wolpert, D.H., 1992. Stacked generalization. Neural Networks 5, 241–259. URL: https://www.sciencedirect.com/science/article/pii/S0893608005800231, doi:https://doi.org/10.1016/S0893-6080(05)80023-1.

wspirat, 2023. Germany used cars dataset 2023. https://www.kaggle.com/datasets/wspirat/germany-used-cars-dataset-2023. License: CC0 Public Domain.

Yi, Y., Chen, J., Feng, J., 2021. Optimizing prices in trade-in strategies for vehicle manufacturers. Computers & Industrial Engineering 157, 107338. doi:https://doi.org/10.1016/j.cie.2021.107338.

Zacharof, N., Nur, J., Kourtesis, D., Krause, J., Fontaras, G., 2025. A review of the used car market in the european union. doi:https://doi.org/10.2760/3237256.

Zhang, Minshun, 2025. Decoding consumer behavior in the used car market: A machine learning approach to key decision factors. ITM Web Conf. 70, 04033. doi:https://doi.org/10.1051/itmconf/20257004033.

Zheng, G., Mukherjee, S., Dong, X.L., Li, F., 2018. OpenTag: Open attribute value extraction from product profiles, in: Proc. ACM SIGKDD, pp. 1049–1058.

Appendix A

Notation

Table A.1 summarizes the notation used throughout the method (Chapter 4), grouped by raw data, schema, normalization, and transfer learning.

Table A.1: Method notation.

Symbol	Description
Raw data	
r	Market index $\in$ {DK, KSA, UAE}
R	Number of source domains
$\mathbf{x}$	Input vehicle record (feature vector)
y	Trade-in value (target label)
$D_S^{(r)}$	Labeled dataset of source domain r
D_T	Labeled dataset of target domain
$\mathbf{X}_S^{(r)}$	Feature matrix of source domain r
$\mathbf{y}_S^{(r)}$	Label vector of source domain r
$\mathbf{X}_T$	Feature matrix of target domain
$\mathbf{y}_T$	Label vector of target domain
$n_S^{(r)}$	Number of samples in source domain r
n_T	Number of samples in target domain
$\mathcal{X}_S^{(r)}$	Feature space of source domain r
$\mathcal{X}_T$	Feature space of target domain
$\mathcal{Y}_S^{(r)}$	Output (value) space of source domain r
$\mathcal{Y}_T$	Output (value) space of target domain
$d_S^{(r)}$	Dimension of source domain r features

continued on next page

Table A.1 (continued)

Symbol	Description
d_T	Dimension of target domain features
$f_S^{(r)}$	VPP task of source domain r
f_T	VTA task of target domain
$\tilde{\mathbf{y}}$	Log-transformed target values
Schema	
A_j	Attribute j of proposed schema
m	Number of attributes of proposed schema
m_{num}	Number of numerical attributes
m_{cat}	Number of categorical attributes
$\mathcal{I}_{\text{num}}$	Numerical attribute indices
$\mathcal{I}_{\text{cat}}$	Categorical attribute indices
$\mathcal{C}_j$	Discrete (categorical) domain of attribute j
Normalization	
$\mathcal{I}_{\text{closed}}$	Closed-domain attribute indices
$\mathcal{I}_{\text{open}}$	Open-domain attribute indices
m_{open}	Number of open-domain attributes
$\mathcal{V}_j$	Domain of attribute j
$\mathcal{V}_j^{\text{closed}}$	Closed domain of attribute j
$\mathcal{V}_j^{\text{open}}$	Open domain of attribute j
s	Serialized vehicle representation
ϕ	Vehicle normalization function
$\hat{\mathbf{x}}$	Extracted attribute values $(\hat{a}_1, \ldots, \hat{a}_m)$
$\hat{a}_j$	Extracted candidate value for attribute j
$\mathbf{e}_j$	Embedding of extracted candidate $\hat{a}_j$
d_{emb}	Embedding dimension
$\mathcal{K}_j$	Knowledge catalog of open-domain attribute j
c_i	Canonical entity value (candidate) for attribute j
$\mathbf{e}_i$	Embedding of canonical entity c_i
k	Number of retrieved candidates (top-k)
$\mathcal{R}_j$	Retrieved top-k candidates for attribute j
$\hat{s}$	Serialization of extracted attributes

continued on next page

Table A.1 (continued)

Symbol	Description
$\hat{s}_{j\to c_i}$	Serialization with A_j replaced by candidate c_i
M	Entity-resolution matcher (LLM)
$\tilde{a}_j$	Normalized value for attribute j
$\tilde{\mathbf{x}}$	Normalized vehicle representation
$\tilde{\mathbf{X}}_S^{(r)}$	Normalized feature matrix of source domain r
$\tilde{\mathbf{X}}_T$	Normalized feature matrix of target domain
$\tilde{\mathbf{y}}_S^{(r)}$	Log-transformed label vector of source domain r
$\tilde{\mathbf{y}}_T$	Log-transformed label vector of target domain
$\tilde{\mathcal{X}}$	Normalized vehicle space
$\tilde{\mathcal{X}}_S$	Normalized feature space of source domains
$\tilde{\mathcal{X}}_T$	Normalized feature space of target domain
$\tilde{d}_S$	Dimension of normalized source domain features
$\tilde{d}_T$	Dimension of normalized target domain features
Transfer learning	
$\tilde{D}_S^{(r)}$	Normalized labeled dataset of source domain r
$\tilde{D}_T$	Normalized labeled dataset of target domain
$\hat{f}_S^{(r)}$	VPP model for market r
$\hat{\mathbf{y}}^{(r)}$	Source-model (VPP) predictions
π	Projection function
$\tilde{\mathbf{x}}'$	Projected normalized vehicle representation
$\tilde{\mathbf{X}}'_T$	Projected normalized target feature matrix
$\bar{\mathbf{x}}$	Augmented feature vector
$\bar{\mathbf{X}}_T$	Augmented target feature matrix
$\bar{D}_T$	Augmented labeled dataset of target domain
$\bar{\mathcal{X}}_T$	Augmented feature space of target domain
$\mathcal{F}$	Hypothesis space (gradient-boosted trees)
ℓ	Loss function
$\hat{f}_T$	VTA model

Appendix B

Prompts

We use the few-shot prompting technique to perform information extraction and entity resolution in the vehicle normalization process. Full prompt templates and two representative examples are provided below for each system instruction.

Information Extraction Prompt The information extraction prompt instructs the LLM to map a serialized vehicle representation to the proposed schema described in Table 5.1. Beyond copying values stated explicitly in the input, the model is asked to *infer* the remaining attributes from its automotive world knowledge, as a domain expert would. The prompt uses 29 diverse examples to demonstrate this extract-then-infer behavior.

Entity Resolution Prompt The entity resolution prompt frames matching as a pairwise decision: two serialized vehicle descriptions x and y that differ only in a single attribute are compared, and the LLM decides whether the two values of that attribute are surface forms of the same real-world entity. This handles synonymy, abbreviations, hyphenation, localized translations, and the catalog-canonical versus listing-written forms of a name. The prompt uses 26 examples.

```
# Role
You are a vehicle data enrichment system. Given a vehicle description, you
both extract information explicitly stated and infer anything you can
determine with confidence from your knowledge of the specific make, model,
and year -- as a car expert would.

# Task
Parse the input (a marketplace listing, CSV row, or free-form text in any
language) and fill every field in the Vehicle schema. Leave a field None
only when it is genuinely unknowable given the input and your world
knowledge.

# Rules
1. Lowercase all string values.
2. Extract then infer -- first pull what is explicit, then use your
   automotive knowledge to fill what remains: body type, drive type,
   cylinders, aspiration, transmission gears, energy source, etc. A car
   expert reading this description would not leave these blank if the model
   is well-known.
3. energy_source is always derivable from fuel type:
   diesel/petrol/lpg/cng -> fossil; electric -> electric; hybrid -> hybrid.
4. Units: preserve original values and units -- do not convert. Prefer kW
   over PS when both are present.
5. Model/submodel split for alphanumeric designations: when a model name is
   a letter-code followed by a number (e.g. Mercedes GLA 220, C 350; BMW
   320, 530; Audi A3, Q5), put the letter part as model and the
   number+suffix as submodel. Examples: GLA 220 d -> model: gla,
   submodel: 220 d; 320d -> model: 3 series, submodel: 20d.

# Examples
input: volvo v60 2.4 d6 awd wagon automatic
output: {"brand": "volvo", "model": "v60", "engine_size": 2.4, ...,
    "drive_type": "awd"}
...
# (29 examples total, 2 shown below)

# Input
Vehicle description: {{x}}
```

Information Extraction Prompt Template

```
input: audi Audi Q3 silver 2012 155 211 Automatic Petrol 267200.0 2.0 TFSI
    quattro SHZ PDC
output:
{
  "brand": "audi",
  "model": "q3",
  "year": 2012,
  "mileage": 267200,
  "body_type": "suv",
  "transmission_type": "automatic",
  "transmission_gears": 7,
  "energy_source": "fossil",
  "fuel_type": "petrol",
  "propulsion_system": "icev",
  "drive_type": "awd",          # inferred from "quattro"
  "engine_size": 2.0,
  "engine_power": 155,
  "engine_aspiration": "turbo", # inferred from "TFSI"
  "injection_type": "direct",   # inferred from "TFSI"
  "cylinders": 4,               # inferred from the 2.0 TFSI engine
  ... (other fields: null)
}
```

Example 1: Audi Q3 (fossil)

```
input: tesla model 3 long range sedan automatic electric
output:
{
  "brand": "tesla",
  "model": "model 3",
  "submodel": "long range",
  "body_type": "sedan",
  "transmission_type": "automatic",
  "transmission_gears": 1,      # single-speed, inferred for a BEV
  "energy_source": "electric",  # derived from the electric fuel type
  "propulsion_system": "bev",   # inferred
  "drive_type": "rwd",          # inferred for the base Model 3
  ... (other fields: null)
}
```

Example 2: Tesla Model 3 (electric)

```
# Role
You are an expert in vehicle brand and model naming conventions across
manufacturers and markets.

# Task
Two vehicle descriptions x and y are given below, serialized as
"col1: val1, col2: val2, ...". They are constructed so that they differ
only in the value of the {{attribute}} field. Determine whether the two
{{attribute}} values refer to the same real-world {{attribute}} entity --
that is, whether they are two surface forms of a single entity.

# Criterion
Return is_match: true ONLY when the two values are lexical variants of the
same real-world entity:
- Abbreviations:          "vw" <-> "volkswagen", "mb" <-> "mercedes-benz".
- Hyphenation / spacing:  "c 350" <-> "c350", "t-roc" <-> "t roc".
- Localized translations: "serie 3" <-> "3 series", "classe e" <-> "e class".
- Catalog-canonical vs listing-written form of the same entity.
Return is_match: false in every other case, including:
- Different brands, even within one group (GMC != Chevrolet; Skoda != VW).
- Different models with no shared root (Sierra != Silverado).
- Body/size/performance variants of a shared root (Golf GTI != Golf;
  XC40 != XC60; Grand Cherokee L != Grand Cherokee).

# Important
The other attributes (year, mileage, color, body_type, ...) may match or
differ. They are irrelevant to this decision. Base your answer ONLY on
whether the two {{attribute}} values are surface forms of the same entity.

# Inputs
x: "{{x}}"
y: "{{y}}"

# Examples (output: reasoning + is_match)
# (26 examples total, 2 shown below)
```

Entity Resolution Prompt Template

```
input:
  attribute: "model"
  x: "serie 3"
  y: "3 series"
output:
  reasoning: "Spanish/French name for the BMW '3 series' -- same model line."
  is_match: true
```

Example 1: Match

```
input:
  attribute: "model"
  x: "brand: gmc, model: sierra, year: 2012, body_type: pickup, cylinders: 8"
  y: "brand: gmc, model: silverado, year: 2012, body_type: pickup, cylinders:
    8"
output:
  reasoning: "Sierra and Silverado are platform siblings but distinct model
              lines. Identical year, body_type, and cylinders are irrelevant."
  is_match: false
```

Example 2: No Match

Appendix C

Scalability

Beyond predictive quality, a normalization framework intended for production must scale to large catalogs without prohibitive latency or cost. The marketplace datasets used in the main experiments (Table 5.3) are proprietary and cannot be redistributed, which limits the reproducibility of any profiling conducted on them. To enable a reproducible assessment, we evaluate scalability on two publicly available open-source datasets, AutoScout24 (wspirat, 2023) and MuCars (Tabarnoust et al., 2025), summarized in Table C.1. The two datasets represent distinct markets, a mature European one and an emerging North African one, and exhibit the same naming, granularity, and scope inconsistencies that the framework is designed to resolve. This empirical study complements the theoretical complexity analysis of Section 4.4.

Table C.1: Open-source datasets used for the scalability evaluation.

Dataset	**Market**	N	N_b	N_m
AutoScout24	Germany	251.1K	47	1,312
MuCars	Morocco	101.9K	81	1,049

N: dataset size; N_b: number of brands; N_m: number of models

AutoScout24 (wspirat, 2023) is a German used-car dataset with fully structured fields, publicly released on Kaggle under a CC0 public-domain license. MuCars (Tabarnoust et al., 2025) documents the Moroccan used-car market and is publicly available through Mendeley Data; it includes several market-specific attributes and a broader variety of model names than AutoScout24.

We profile the full extraction–retrieval–resolution pipeline on a stream of 10,000 unique records per dataset, processed with up to 50 concurrent requests, on the hardware described in Appendix G (Apple M4, 16 GB) under the configuration of Section 5.1. Starting from an initially seeded catalog, we record per-record latency, throughput, and the number of API calls as the catalog grows. Figure C.1 reports the results.

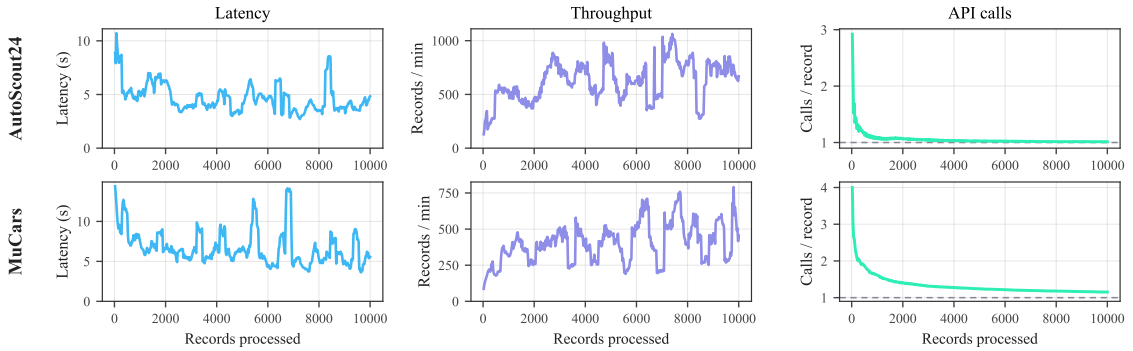


Figure C.1: Computational efficiency over a 10,000-record stream for each dataset. Rows: AutoScout24 (top) and MuCars (bottom). Columns: per-record latency (left), throughput (center), and API calls per record (right; the dashed line marks the single-extraction floor).

Latency Per-record latency is dominated by the API calls and decreases as the catalog saturates and fewer records trigger entity resolution. The median per-record latency is 3.8 s on AutoScout24 and 5.4 s on MuCars, with 95th percentiles of 8.4 and 12.7 s, respectively, corresponding to records that still require resolution.

Throughput A single process sustains roughly 566 records per minute on AutoScout24 and 364 on MuCars. Because local computation is negligible, throughput is bounded by API latency rather than by hardware, and scales with additional request concurrency up to the provider's quota.

API calls Each record incurs exactly one extraction call; only attribute values absent from the catalog require an additional embedding and one or more entity-resolution calls, while values already canonical require none. As the cache-hit rate climbs to 96% on AutoScout24 and 92% on MuCars, the number of API calls per record falls toward the single-extraction floor, settling at 1.00 and 1.06, respectively. Across the 10,000-record streams, entity resolution accounts for only 124 and 1,523 calls and embedding for a further 176 and 521. The higher MuCars value reflects the greater variety of novel models in its data, each of which must be resolved against retrieved candidates.

Local footprint Locally, the pipeline performs little computation: approximate nearest-neighbor (HNSW) search runs on CPU in under 0.3 ms per record, uses no GPU, and consumes under 0.8 GB of memory regardless of corpus size. This confirms the complexity analysis of Section 4.4: the constant-cost API calls dominate, while the only catalog-dependent term, retrieval, remains negligible at deployment scale, keeping latency and cost bounded as the corpus grows.

Appendix D

Domain mismatch

To demonstrate that vehicle price prediction (VPP) models cannot be directly applied to the trade-in domain, we evaluate listing models trained on the DK, KSA, and UAE datasets by testing them on the DK trade-in test set. For this evaluation, we exclude trade-in-specific attributes and use only the features available in the listing datasets. As shown in Table D.1, all configurations exhibit substantially higher error rates compared to the dedicated VTA models presented in Table 5.7, with MAPE values ranging from 46.55% to 127.99% versus 21.69% for our best transfer learning approach. These results confirm that listing prices and trade-in values represent fundamentally different prediction tasks, necessitating a dedicated transfer learning approach rather than direct model application.

Table D.1: Performance of pricing models in the trade-in domain. Bold denotes best performance per dataset and metric.

Dataset	**Model**	**MAPE**	**MdAPE**	**MBE**
DK	CatBoost	62.75 ± 5.37	38.93 ± 5.00	416 ± 882
	CatBoost+	64.90 ± 5.42	40.29 ± 3.15	6466 ± 499
	LightGBM	$\mathbf{57.00 \pm 5.26}$	$\mathbf{30.87 \pm 1.42}$	1414 ± 579
	LightGBM+	63.87 ± 5.63	37.57 ± 2.44	5957 ± 468
	Random Forest	60.72 ± 5.32	37.09 ± 1.37	2657 ± 300
	Random Forest+	65.06 ± 5.80	40.57 ± 2.43	6415 ± 405
	XGBoost	59.67 ± 5.75	35.89 ± 3.19	$\mathbf{670 \pm 596}$
	XGBoost+	66.13 ± 4.98	39.86 ± 2.46	6534 ± 359
KSA	CatBoost	87.54 ± 14.59	32.99 ± 2.27	-2211 ± 1215
	CatBoost+	$\mathbf{66.78 \pm 12.51}$	$\mathbf{28.28 \pm 2.27}$	-2702 ± 584
	LightGBM	102.29 ± 17.53	44.80 ± 1.67	-6492 ± 663
	LightGBM+	104.29 ± 18.78	38.26 ± 2.43	-3935 ± 812
	Random Forest	120.25 ± 24.11	45.07 ± 1.39	-5304 ± 833
	Random Forest+	127.99 ± 27.90	40.47 ± 3.66	$\mathbf{-2583 \pm 1651}$
	XGBoost	85.51 ± 11.64	41.77 ± 3.20	-6498 ± 810
	XGBoost+	85.41 ± 17.09	32.90 ± 3.36	-3243 ± 1467
UAE	CatBoost	49.13 ± 3.41	39.69 ± 2.14	-7948 ± 533
	CatBoost+	$\mathbf{46.55 \pm 3.91}$	$\mathbf{37.27 \pm 2.44}$	-7567 ± 480
	LightGBM	59.93 ± 6.16	39.82 ± 1.17	-7300 ± 461
	LightGBM+	57.17 ± 6.55	39.30 ± 2.06	$\mathbf{-6219 \pm 324}$
	Random Forest	56.04 ± 4.64	38.62 ± 0.97	-6809 ± 349
	Random Forest+	57.55 ± 4.48	38.37 ± 1.63	-6682 ± 411
	XGBoost	56.63 ± 6.53	42.66 ± 2.50	-8259 ± 502
	XGBoost+	52.20 ± 6.69	38.86 ± 1.45	-6424 ± 724

Appendix E

Statistical significance

The results reported in Chapter 5 are averaged over ten seeded train–test splits, with the same splits reused across configurations. Because the configurations are therefore paired by split, we can assess whether the differences between them are statistically significant rather than attributable to split-to-split variation. Throughout this appendix we use a paired two-sided Wilcoxon signed-rank test, a non-parametric test that compares two configurations across the ten shared splits without assuming a particular distribution of the differences. We examine the listing domain first and the trade-in domain second.

E.1 Listing domain

Table E.1 tests, for each market and algorithm, whether the normalized model (+) significantly improves over its non-normalized counterpart. Normalization yields a statistically significant reduction in error for every algorithm on the UAE and KSA markets, on both MAPE and MdAPE, with relative MAPE gains of up to 8.6% (CatBoost on KSA). These are precisely the markets in which missing data is most severe (Table 5.4): engine power, for instance, is entirely absent from both, so normalization has the most to repair through LLM-based extraction and the imputation that standardized representations enable.

The DK listings are the informative exception: despite being by far the largest dataset, they exhibit the smallest and mostly non-significant normalization effect. This is not a question of statistical power, which is greatest for this dataset, but of effect size: DK records are comparatively complete, leaving little for normalization to correct. The contrast makes the underlying mechanism explicit: normalization helps where representations are heterogeneous or incomplete, and the test confirms an effect only where one genuinely exists, regardless of dataset size.

Table E.1: Significance of normalization in the listing domain. For each market and algorithm, Δ is the reduction in error (percentage points) of the normalized model (+) relative to its non-normalized counterpart, averaged over ten seeded train–test splits; a positive value favors normalization. The p column reports a paired two-sided Wilcoxon signed-rank test between the two configurations. A marker (†) denotes a significant improvement ($p < 0.05$). Absolute error values are reported in Table 5.8.

	MAPE		**MdAPE**	
Model	Δ	p	Δ	p
DK listings (972.4K records)				
Random Forest	$0.29^\dagger$	0.004	$0.20^\dagger$	0.002
XGBoost	−0.14	0.193	−0.04	0.922
LightGBM	0.02	0.557	$0.10^\dagger$	0.014
CatBoost	0.05	0.695	0.09	0.922
UAE listings (101.4K records)				
Random Forest	$0.62^\dagger$	0.002	$0.68^\dagger$	0.002
XGBoost	$0.59^\dagger$	0.002	$0.49^\dagger$	0.002
LightGBM	$0.38^\dagger$	0.002	$0.41^\dagger$	0.002
CatBoost	$0.57^\dagger$	0.004	$0.45^\dagger$	0.004
KSA listings (19.0K records)				
Random Forest	$0.55^\dagger$	0.002	$0.50^\dagger$	0.002
XGBoost	$0.52^\dagger$	0.002	$0.39^\dagger$	0.004
LightGBM	$0.44^\dagger$	0.002	$0.29^\dagger$	0.002
CatBoost	$0.76^\dagger$	0.002	$0.64^\dagger$	0.002

E.2 Trade-in domain

Table E.2 reports, for each trade-in algorithm, a test of each configuration against its base (raw trade-in features, no transfer learning). The base and the two single-component configurations are already reported in Table 5.7; here we restate their values alongside the corresponding p-values to make the comparison explicit. Transfer learning is the dominant driver of the MAPE reduction and is significant for every algorithm except XGBoost, whereas normalization on its own most clearly improves the median error (MdAPE).

One caveat guides the reading of Table E.2. The trade-in dataset comprises only about 1,200 samples (Table 5.3), two to three orders of magnitude smaller than the listing datasets, so the test has limited statistical power: a non-significant entry should be read as the data being unable to resolve an effect of that size, not as evidence that the component has no effect. This is a direct consequence of the small trade-in sample discussed in Section 6.1, which inflates the split-to-split variance and widens the confidence intervals. Significant improvements, such as those of the

transfer learning configurations, are therefore established despite this limited power, not because of it.

The number of splits on which a configuration improves over its base, reported in parentheses next to each p-value, makes the limits of the test concrete. The Wilcoxon statistic sums the ranks of the splits that move against the treatment, so it weighs reversals by their magnitude rather than merely counting them. With ten splits the critical value at $p < 0.05$ is a rank sum of 8, which means a single reversal among the two largest per-split differences is enough to leave a configuration non-significant, while the smallest attainable p-value is 0.002, reached only when all ten splits agree. The number of favorable splits alone therefore does not determine the outcome. On MAPE, LightGBM with normalization and transfer improves on 9 of 10 splits and is significant, its single reversal being the third smallest difference in the set and yielding a rank sum of 3, whereas XGBoost with the same components improves on 7 of 10 and is not, because two of its three reversals rank among the four largest differences and the rank sum reaches 17. The same mechanism explains the pattern discussed in Section 5.9, where the MdAPE improvement of XGBoost under transfer learning is significant on 8 of 10 splits while its MAPE improvement, driven by a small number of large errors, is not.

Table E.2: Paired significance tests for the trade-in configurations of Table 5.7, grouped by trade-in algorithm. When transfer learning is enabled, the listing model is of the same algorithm. Values are mean (standard deviation) over ten seeded train–test splits; the p column reports a paired two-sided Wilcoxon signed-rank test of each configuration against the *Base* (raw features, no transfer) of the same algorithm, followed in parentheses by the number of splits on which the configuration improves over that base. A marker (†) denotes a significant improvement over the base ($p < 0.05$). Bold denotes the best value per algorithm and metric.

Component		**MAPE**		**MdAPE**	
Norm.	**Transfer**	**Value**	p **(wins)**	**Value**	p **(wins)**
Random Forest					
		27.44 ± 3.90	–	16.88 ± 1.53	–
✓		27.62 ± 3.44	0.43 (3/10)	17.21 ± 1.25	0.56 (5/10)
	✓	$23.58 \pm 1.83^\dagger$	0.002 (10/10)	16.30 ± 1.39	0.56 (5/10)
✓	✓	$\mathbf{23.54 \pm 2.64}^\dagger$	0.002 (10/10)	$\mathbf{15.89 \pm 1.12}$	0.19 (6/10)
XGBoost					
		25.04 ± 2.90	–	16.67 ± 1.39	–
✓		24.89 ± 2.19	1.00 (4/10)	$15.95 \pm 1.15^\dagger$	0.04 (8/10)
	✓	$\mathbf{24.26 \pm 2.60}$	0.63 (6/10)	$\mathbf{15.46 \pm 1.16}^\dagger$	0.01 (8/10)
✓	✓	$\mathbf{24.26 \pm 2.27}$	0.32 (7/10)	15.72 ± 1.06	0.11 (7/10)
LightGBM					
		25.13 ± 2.58	–	16.32 ± 1.36	–
✓		24.73 ± 2.47	0.19 (7/10)	16.06 ± 1.47	0.63 (6/10)
	✓	$\mathbf{22.92 \pm 2.07}^\dagger$	0.002 (10/10)	$\mathbf{15.46 \pm 1.12}^\dagger$	0.04 (8/10)
✓	✓	$23.20 \pm 1.86^\dagger$	0.010 (9/10)	15.91 ± 1.22	0.28 (6/10)
CatBoost					
		23.67 ± 2.25	–	15.83 ± 1.65	–
✓		23.45 ± 1.79	0.43 (6/10)	$15.17 \pm 1.55^\dagger$	0.006 (9/10)
	✓	$22.01 \pm 1.83^\dagger$	0.010 (9/10)	$14.23 \pm 1.50^\dagger$	0.02 (8/10)
✓	✓	$\mathbf{21.69 \pm 1.74}^\dagger$	0.010 (9/10)	$\mathbf{14.14 \pm 1.34}^\dagger$	0.01 (8/10)

Appendix F

Software

All components were implemented in Python 3.10. Table F.1 lists the main libraries used in the experiments, along with their versions. The ensemble baselines rely on their respective open-source libraries (scikit-learn for Random Forest, together with XGBoost, LightGBM, and CatBoost). Data handling and numerical computation are based on pandas and NumPy. The vehicle normalization framework is built on top of large language model APIs and a retrieval component, as described in Chapter 4.

Table F.1: Software environment used for all experiments.

Library	**Version**
Python	3.10
NumPy	1.26.4
pandas	2.2.2
scikit-learn	1.5.1
XGBoost	2.1.1
LightGBM	4.5.0
CatBoost	1.2.5

Appendix G

Hardware

All experiments reported in this thesis were conducted on a single Apple Mac mini (Mac16,10). Table G.1 summarizes the hardware configuration. The machine is powered by the Apple M4 system-on-chip, which integrates a 10-core CPU (4 performance and 6 efficiency cores) and 16 GB of unified memory shared between the CPU and GPU.

Table G.1: Hardware configuration used for all experiments.

Component	**Specification**
Model	Apple Mac mini (Mac16,10)
System-on-chip	Apple M4
CPU	10 cores (4 performance, 6 efficiency)
Memory	16 GB unified
Operating system	macOS 15.7