Bulgarian Diploma Thesis

User Car Predictor with Machine Learning

Aleksandar Ivanov, ID#200231727

Student: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ,

Date: \_\_\_\_\_\_\_\_\_\_\_

Supervisor: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ,

Date: \_\_\_\_\_\_\_\_\_\_\_

Department of Computer Science AUBG

Blagoevgrad 2025

Table of Contents

[1. Introduction 3](#_Toc211705323)

[1.1 Motivation 3](#_Toc211705324)

[1.2 Project Overview 4](#_Toc211705325)

[2. Specification of the Software Requirements and their Analysis 5](#_Toc211705326)

[2.1 Functional requirements 5](#_Toc211705327)

[2.2. Non-functional Requirements 7](#_Toc211705328)

[2.3. Requirements Analysis 7](#_Toc211705329)

[2.3.1. Analysis of Functional Requriements 7](#_Toc211705330)

[2.3.2 Analysis of Non-functional Requirements 12](#_Toc211705331)

[3. Design of Software Solution 12](#_Toc211705332)

[3.1. Main algorithms 12](#_Toc211705333)

[3.1.1. Linear regression 12](#_Toc211705334)

[3.1.2. Ridge Regression 13](#_Toc211705335)

[3.1.3. Random Forest Regression 14](#_Toc211705336)

[3.1.4. Gradient Boosting Regression 15](#_Toc211705337)

[3.2. Algorithm Complexity 16](#_Toc211705338)

[3.2.1. Linear Regression Complexity 16](#_Toc211705339)

[3.2.2. Ridge Regression Complexity 16](#_Toc211705340)

[3.2.3. Random Forest Complexity 17](#_Toc211705341)

[3.2.4. Gradient Boosting Complexity 17](#_Toc211705342)

[3.3. Software Architecture 18](#_Toc211705343)

[References 18](#_Toc211705344)

# 1. Introduction

## 1.1 Motivation

One of the fastest-moving parts of the global automotive industry is the used car market. Each year, millions of cars are sold and bought in the second-hand market and their prices are influenced by a lot of factors like brand, model, mileage, year of production, engine, transmission and fuel type (gasoline, diesel, hybrid, electric). For buyers who are not aware of the details in the pricing of vehicles it can lead to overpaying. On the other side, for sellers, inaccurate research on how they should price their vehicle may lead to losses or overpricing it which will cause delays in selling.

Usually, the valuation of vehicles is handled manually or by online platforms that rely on historical average (Amik et al.). However, these methods are not the most optimal as of today. Expert-based evaluations can be subjective, time-consuming and difficult to scale across large markets (Alnajim et al.). Using online platforms, while they are automated way to predict prices, they often fail to make the right prediction (Pal et al.; Amik et al.). These inaccuracies can cause a huge financial damage to the sellers because the market is very competitive.

Machine learning (ML) gives us a new and improved way to predict prices for vehicles. By using large datasets from previously sold vehicles, ML can find patterns. ML can find relations in data that is non-linear like such as brand, model, mileage, year of production which will lead to calculations which are more precise. Using supervised algorithms such as linear regression, gradient boosting, random forest accuracy is significantly improved compared to manual and rule-based techniques (Amik et al.; Alnajim et al.; Pal et al.).

The motivation behind this project is to make a working ML-powered web application that can give correct predictions to prices on used vehicles given their key attributes. Both buyers and sellers can benefit from such system because it will introduce a lot more transparency and reducing the risk of overvaluation or undervaluation. It will make dealerships more competitive as they are now going to be able to adapt to the price changes much quicker. This project can also be of use of academic purpose as it shows how ML techniques can be applied to help a real-world business problem. In this way, the project how data-centric approaches can be used to modernize an industry which relies on subjective judgement.

## 1.2 Project Overview

The project is a data-driven web application which is going to be predicting price of used vehicles. The backend is implemented with .NET which is consisting of all the prediction algorithms and endpoints which are going to be hit by the frontend which is implemented in TypeScript with React. The purpose of the web-app is to demonstrate how machine learning algorithms can be used to generate accurate car valuations which is going avoid the current method of judging which rely on humans and old rigid rule-based calculators.

In its core the application integrates several machine learning algorithms, including Linear Regression, Gradient Boosting and Random Forest. These algorithms will be trained on publicly available used-cars datasets, which contain the following information: brand, model, year of production, transmission, engine type and fuel type. The system also integrates preprocessing components like a FeatureScaler and LabelScaler, which were implemented specifically for this project. The FeatureScaler is responsible for normalizing the numeric input features (e.g., mileage, production year) into a standardized range before training. The LabelScaler performs the inverse operation by rescaling the model’s predicted values back to the original monetary scale. These components form part of the backend preprocessing pipeline and are integrated into the application’s overall architecture as independent, reusable modules.

The project has two main goals. On the practical side, it is a tool which can be used by dealerships/sellers and buyers to make the process of buying a used car more transparent and aligned with the current market. And on the academical side it can be used as a case-study for what machine learning is most suitable by enabling direct comparison between the models in terms of their accuracy and computational cost and time cost.

In addition, the project aims to provide the end user the full pipeline not only a command line interface app by implementing user-friendly web experience. Which will demonstrate how each of the algorithms perform under the same conditions and the results will be shown to the end-user in a way that a non-expert can use them. By combining everything mentioned the project will provide a robust solution for used car predictions.

# 2. Specification of the Software Requirements and their Analysis

## 2.1 Functional requirements

* Algorithms: Train and serve predictions from Linear Regression, Ridge, Random Forest, and Gradient Boosting.
* Single-car query: The user selects price, odometer, year, manufacturer, model, fuel, transmission and the planned purchase year. The app returns a predicted price from each algorithm, plus an optional ensemble average.
* Multi-year projection: For a configured car, the user specifies a horizon *n* years (e.g., 1–5). The app produces a year-by-year price projection for the next *n* years and displays a chart (line plot) of the predicted prices, with the option to overlay prediction intervals and historical prices for context.
* Batch predictions: Users can upload a CSV to obtain per-row predictions from all algorithms.
* Data preprocessing pipeline:
  + handles missing values (impute/drop with a summary report),
  + **Outlier handling. Outliers are detected per numeric feature using Tukey’s IQR rule on the training split only.**
  + encodes categorical features,
  + scales numeric features where needed (z-score for linear models; trees receive the same columns),
  + enforces a rare-category policy to remove categories with insufficient data representation in the training set, ensuring model consistency across all dataset splits.
* Dataset ingestion: Load historical vehicle listings from CSV files with the following columns: price, odometer, year, manufacturer, model, fuel, transmission. Allow importing additional CSVs that conform to the documented schema.
* Training & validation: Support k-fold cross-validation and hold-out evaluation (default 80/20). Fit all preprocessing on the training split only and apply on the validation/test to avoid leakage.
* Model comaprison UI:
  + User view: for a configured vehicle, show each algorithm’s predicted price and (if the user specifies a horizon *n* years) render a multi-year projection chart of predicted prices for the next *n* years. Allow overlaying projections from multiple algorithms, toggling prediction intervals, and exporting the chart.
  + Evaluation view (internal/reporting): show diagnostic plots (predicted vs. held-out targets and residuals) and a metrics table with MAE, RMSE, R² reported as mean ± stdev across cross-validation folds.
* **Two-vehicle comparison UI (same algorithm):** Enable a **comparison mode** where the user selects **one algorithm** and **two vehicle configurations** (price, odometer, year, manufacturer, model, fuel, transmission, purchase year/horizon n). The app renders an **overlay chart** of the two **multi-year price projections** (same horizon, same y-axis), with a legend and tooltips, and provides a **diff table** (per-year Δ and average Δ over the horizon).

## 2.2. Non-functional Requirements

1. The application should be scalable, allowing users to upload used-car datasets in CSV format that follow the defined schema.
2. The application should be configurable in terms of feature selection, enabling users to include or exclude attributes such as fuel type, transmission, or purchase year.
3. The system should achieve a reasonable level of accuracy, with at least one algorithm reaching an 𝑅² ≥ 0.7 on validation data.
4. The web interface should be intuitive and easy to use for non-expert users, presenting predictions and comparison charts in a clear and accessible way.
5. The web application should be deployable on common operating systems (Windows, macOS, Linux) and run in modern browsers without additional plugins.

## 2.3. Requirements Analysis

### 2.3.1. Analysis of Functional Requriements

The core of the application lies in its machine learning algorithms. These were selected to provide a balance between interpretability, accuracy, and computational efficiency.

**Linear Regression** was included because of its simplicity and transparency. It provides an interpretable baseline against which more advanced models can be compared. However, linear regression is often insufficient for capturing the nonlinear relationships that exist in car price data, such as the interaction between mileage and production year (J. I. Kang et al.).

**Ridge and Lasso Regression** extend linear regression by introducing regularization, which penalizes overly complex models and reduces the risk of overfitting. This makes them more stable for datasets that include correlated variables, a common situation in real-world car listings (Ting Tin Tin et al.).

**Random Forest Regression** was chosen due to its ability to handle nonlinear interactions and its robustness to noisy features. Research consistently reports that Random Forest achieves strong performance in car price prediction tasks (J. I. Kang et al.; Gegic et al.).

**Gradient Boosting** was selected because it incrementally corrects errors of weaker learners, often producing the most accurate results. While it requires more computation and careful tuning, boosting methods are reported to achieve state-of-the-art performance in car price prediction tasks (J. I. Kang et al.).

To demonstrate the implementation requirements, the algorithms can be summarized in pseudocode:

{pseudo code here}

Another essential requirement is the **data preprocessing pipeline**. Raw vehicle listings often contain missing values, outliers, and inconsistent formats. Without cleaning, models may learn spurious relationships or perform poorly on unseen data. To address this, the pipeline will include:

* Handling missing values (forward fill or row removal when thresholds are exceeded).
* Detecting and removing outliers using statistical measures such as the z-score.
* Encoding categorical variables (e.g., transmission, fuel type).
* Scaling numerical features for models sensitive to feature magnitude.

These preprocessing steps are consistently identified as key to improving generalization and stability in predictive modeling (Gegic et al.; Haque et al.). The figure below summarizes this process:

{ Preprocessing pipeline diagram}

A well-designed user interface is what distinguishes a good software package from an average one. A simple, intuitive, and comprehensive dashboard drastically improves user experience and accessibility. The dashboard in this application is designed to hide the complexities of the underlying machine learning algorithms, allowing the user to obtain predictions and comparisons without needing to understand the technical implementation details.

The interface will provide:

* A dataset upload module (CSV).
* A drop-down menu to select algorithms, with Linear Regression as the default.
* Panels for setting hyperparameters.
* Tools for running single-car predictions and batch predictions.
* Visual aids such as scatter plots of predicted vs. actual values, residual plots, and charts of error metrics (MAE, RMSE, R²).
* A comparison module to display differences between models or between two cars under the same model.

These elements transform the project into a valuable tool for both practitioners and students, as it enables experimentation and fosters understanding of supervised learning algorithms (J. I. Kang et al.; Ting Tin Tin et al.).

{Use Case Diagram of UI interactions}

The workflow of the system ties together preprocessing, training, evaluation, and visualization. Once the user has uploaded a dataset, the application preprocesses the data by handling missing values and outliers. It then splits the dataset into training and testing subsets before passing the data to the selected regression algorithms. Each algorithm is trained, evaluated on unseen test data, and its predictions are visualized through charts and error metrics. Finally, the system provides facilities to compare models under identical conditions.

This sequence ensures reproducibility and transparency while aligning with established practices in machine learning pipelines (Haque et al.).

{Activity Diagram of system workflow}

Finally, the performance of the regression algorithms must be evaluated using well-established metrics. The most common in regression tasks are the coefficient of determination (R²), the mean absolute error (MAE), and the root mean squared error (RMSE). The R² value reflects how closely predictions fit real data, with values near 1 indicating a strong fit. RMSE provides an error measure but is sensitive to outliers, while MAE is more robust to extreme values. Together, these three metrics provide a balanced assessment of predictive performance.

Based on prior research in car price prediction, ensemble models such as Random Forest and Gradient Boosting are expected to achieve the highest R² values, typically above 0.8 on large datasets (J. I. Kang et al.; Gegic et al.). Linear Regression and its regularized variants generally produce lower but still useful scores, often around 0.6–0.7 (Ting Tin Tin et al.). These thresholds are not guarantees, as actual performance will depend on dataset quality, preprocessing, and parameter tuning, but they provide reasonable targets for evaluation.

Example Use Case Narratives

i. Name: Choose Prediction Algorithm

* Actor: User
* Entry Condition: The user has uploaded a valid dataset or selected a built-in one.
* Flow: The dashboard displays a drop-down list of available algorithms (Linear Regression, Random Forest, Gradient Boosting). The default is Linear Regression.
* Exit Condition: A specific algorithm is chosen and the system is ready for training.
* Exceptions: The default prevents null selection.

ii. Name: Upload Dataset (CSV)

* Actor: User
* Entry Condition: The user starts the application and has a dataset file prepared.
* Flow: The user selects a CSV file with car attributes (price, year, mileage, fuel, transmission, etc.). The system validates schema, checks for missing values, and preprocesses data.
* Exit Condition: Dataset is validated and stored for model training.
* Exceptions: If the file is invalid or missing required columns, the system displays an error and prompts re-upload.

iii. Name: Run Single-Car Prediction

* Actor: User
* Entry Condition: The system has trained models available.
* Flow: The user specifies vehicle attributes (year, odometer, transmission, fuel type). The system applies the chosen model(s) and outputs predicted price(s).
* Exit Condition: Predictions are displayed numerically and optionally visualized.
* Exceptions: If required attributes are missing, the system prompts the user to complete the form.

iv. Name: Compare Models

* Actor: User
* Entry Condition: Two or more models have been trained.
* Flow: The user selects which models to compare. The system displays diagnostic plots (predicted vs. actual, residuals) and a table of metrics (MAE, RMSE, R²).
* Exit Condition: Results are shown side-by-side for evaluation.

### 2.3.2 Analysis of Non-functional Requirements

# 3. Design of Software Solution

## 3.1. Main algorithms

### 3.1.1. Linear regression

**Linear Regression** was chosen because of its simplicity and transparency. It provides an interpretable baseline against which more advanced models can be compared. However, linear regression is often insufficient for capturing the nonlinear relationships that exist in car price data, such as the interaction between mileage and production year (J. I. Kang et al.).

However, the strict assumption of linear relationships is often unrealistic in the used car domain. Real-world depreciation curves tend to be nonlinear: for instance, mileage might reduce value steeply initially, but the marginal effect slows as the car ages. Some researchers have proposed S-curve or logistic models precisely because linear fits lead to increasing errors at extreme values (Salim and Abu).

Moreover, linear regression is vulnerable to **outliers**. In used car datasets, extreme prices or rare models can disproportionately skew the regression line, reducing robustness (Amik et al.). Also, when features like age and mileage are highly correlated, coefficient estimates can become unstable or misleading (Ting Tin Tin et al.).

Despite these limitations, linear regression is still very valuable as a baseline model. In comparative studies, it often achieves very good error rates around 80–85% before more advanced algorithms are applied. Its speed, ease of training, and minimal hyperparameter tuning make it useful especially when rapid prototyping.

### 3.1.2. Ridge Regression

**Ridge Regression** is a regularized linear model that balances interpretability and predictive performance by introducing a penalty term to reduce overfitting. Specifically, it applies L2 regularization, which adds a squared penalty to the magnitude of coefficients. This discourages large weights and stabilizes estimates when predictors are highly correlated, effectively reducing variance at the cost of a small bias.

Recent studies have shown that Ridge Regression remains a competitive method for price prediction tasks, offering a reliable balance between accuracy and stability (Zhang and Politis). It performs particularly well when most predictors contribute moderately to the outcome, making it well suited for datasets where relationships between features and price are distributed across multiple variables.

In used car price prediction, Ridge Regression has demonstrated greater consistency than standard linear regression, particularly in the presence of multicollinearity and noisy features (Bergmann and Feuerriegel). Despite its limitation of retaining all predictors, Ridge offers computational efficiency and improved generalization, positioning it as a strong intermediate model between basic linear methods and more complex machine learning approaches.

### 3.1.3. Random Forest Regression

Random Forest Regression builds a large number of decision trees, each trained on a different sample of the data, and then combines their outputs to generate consistent and accurate predictions (Breiman). This approach reduces overfitting, which is a problem in single decision trees. By averaging the results of many trees, Random Forest captures complex patterns and interactions between features that models like linear regression, often cannot see.

In used car price prediction, Random Forest has proven to be one of the most effective and practical algorithms. It performs well because it can naturally model the nonlinear relationships between features such as mileage, age, brand, and engine size without requiring heavy preprocessing or feature transformation (Pal et al.; Bergmann and Feuerriegel). Research has shown that Random Forest models consistently outperform basic regression techniques in accuracy and robustness across various vehicle datasets (Pal et al.).

A key advantage of Random Forest is its ability to remain stable and reliable even when data contains noise or outliers. Since predictions are averaged across many independent trees, individual anomalies—such as mispriced listings—have a limited effect on the final prediction (Mentch and Zhou). Another important feature is the algorithm’s ability to estimate feature importance, which helps determine which variables most strongly influence the predicted price. Studies commonly find that mileage and model year have the highest importance scores, while other attributes like fuel type or number of doors play smaller roles (Molnar).

However, Random Forest also has limitations. It requires more computational resources than simpler algorithms, and its internal logic is less interpretable because predictions are produced by averaging hundreds of trees. When the dataset contains many weak or redundant features, its performance can also decrease slightly unless preprocessing and feature selection are applied carefully (Scornet). Despite these drawbacks, Random Forest remains a reliable and versatile model for this project. It offers an excellent balance between accuracy, robustness, and practicality, making it one of the core algorithms.

### 3.1.4. Gradient Boosting Regression

Gradient Boosting Regression is a machine learning algorithm that builds models sequentially. Each new model focuses on correcting the errors made by the previous ones. It can be epxlained with an golf example. Where a person shoots, makes a mistake analyses their mistake and shoots again in an attempt to make a better shot.

It optimizes a loss function by fitting to the residuals of prior predictions, which allows it to improve performance over time (Friedman). This process enables Gradient Boosting to learn nonlinear relationships and feature interactions which is an advantage that is especially useful in car price prediction, where variables such as mileage, age, etc.

In practice, one of the most widely used implementations of Gradient Boosting is Extreme Gradient Boosting (XGBoost), known for its computational efficiency, regularization, and scalability. Research applying XGBoost to vehicle pricing has demonstrated high predictive accuracy when paired with proper preprocessing and parameter tuning (Qian). Similarly, Gradient Boosting is one of the best performing algorithms for estimating used car resale values across diverse datasets (Bergmann and Feuerriegel).

A major strength of Gradient Boosting is the control over bias and variance through parameters such as the learning rate, tree depth, and number of estimators. Each tree in the sequence attempts to reduce the residual errors of the previous ones, enabling the model to capture data patterns that simpler algorithms often overlook. However, this flexibility also makes the algorithm more prone to overfitting if parameters are not carefully tuned. Multiple studies highlight the importance of hyperparameter optimization and early stopping to prevent overfitting and ensure stable results (Zheng).

Gradient Boosting is also useful for interpretability tools such as feature importance. This allow users to understand how individual features influence price predictions. When compared to Random Forest, Gradient Boosting requires more resources and a more complex training process. In cases where datasets are small or contain high levels of noise, simpler models can sometimes achieve comparable results with less tuning.

## 3.2. Algorithm Complexity

The algorithms chosen for this senior project were chosen for their accuracy when it comes to used car prediction systems and also their efficiency which is proven by the researchers.

### 3.2.1. Linear Regression Complexity

Linear Regression serves as the base model due to its analytical simplicity and low cost for computation. Training a linear model on a dataset with samples and features requires solving the normal equation . Which has a time complexity of . But we need to compute the matrix inverse which can be both computationally expensive and unstable for large feature matrices.

To overcome this, QR decomposition is used to solve the problem more efficiently. What QR decomposition does is, it factorizes the feature matrix into an orthogonal matrix and an upper triangular matrix such that . Substituting this into the normal equation gives . This approach avoids matrix inversion and improves both stability and efficiency (Amik et al.; Salim and Abu). The decomposition itself has a computational complexity of

### 3.2.2. Ridge Regression Complexity

Ridge Regression extends the linear regression model by introducing an regularization. Which is used to penalize large coefficients, reducing overfitting and improving stability in the presence of multicollinearity (Zhang and Politis; Ting Tin Tin et al.). Ridge Regression uses the same mathematical foundation as the linear regression algorithm, but it adds : , with  having the role of regularization parameter. The inclusion of this penalty term adds minimal computational overhead, and its complexity remains .

### 3.2.3. Random Forest Complexity

**Random Forest Regression** has strong predictive performance which is achieved by averaging the outputs of multiple decision trees. The training complexity of a single decision tree is . For a group with trees, the total training cost is (Scornet; Mentch and Zhou). But the advantage of Random Forests is that they can be trained in parallel, which has a big benefit when it comes to reducing runtime and efficiency with complexity of .

### 3.2.4. Gradient Boosting Complexity

**Gradient Boosting Regression** enhances predictive accuracy by building trees sequentially, with each new tree errors are corrected in an attempt to make a better prediction (Friedman). The training complexity of Gradient Boosting is , where represents the number of boosting iterations. While this makes boosting models more computationally demanding comapred to Random Forests, they generally achieve higher accuracy when tuned appropriately. Research has demonstrated that boosting techniques, particularly XGBoost(which is a more complex expansion on gradient boosting), outperform traditional models in capturing nonlinear relationships between vehicle attributes and price(Qian; Zheng; Bergmann and Feuerriegel).

## 3.3. Software Architecture

The application is developed using the **Model-View-Controller (MVC)** pattern. This structure ensures separation of the responsibilities of data management, user interface, and central logic. The backend, developed with **ASP.NET Core**, which implements the **Model** and **Controller** components from the MVC pattern. And the **View part of the pattern** is handled by a **React and TypeScript in a** single-page application (SPA).

In this architecture, the **Model** is responsble for the machine learning and data processing logic which responsible for training, evaluating, and serving predictions for used car prices. And it implements the following algorithms - Linear Regression, Ridge Regression, Random Forest, and Gradient Boosting - along with the data preprocessing pipeline that handles tasks such as feature encoding, scaling, and dataset validation.

The **Controller** acts as the middleman between the UI and the backend. Its main functionallity is to expose RESTful endpoints which establish the communication between the frontend and the backend. Each controller receives input data from the user, invokes the corresponding code and returns structured JSON responses containing what was requested.

The **View** component is implemented as a React-based web interface. It provides a user-friendly UI where users can and visualize prediction results, compare the performance of different algorithms and comapre different car prices throughout the years. The React application communicates asynchronously with the backend through HTTP requests.

Supporting modules are also added in addition to the main MVC components. They provide additional services such as feature scaling, label scaling, data validation, and model serialization. These utility classes maintain the consistency and reusability of the system across different operations. The interactions between all layers of the project are showcased in the illustration:

{interactions img}

The MVC design pattern was selected because it enables a clear separation between the frontend and the backend, which is essential for a web-based architecture that combines a React frontend and a .NET backend. Another way this can be developed is with monolithic designs but in this situation all UI elements and business logic coexists in the same codebase which makes everything much more couples. On the other hand, MVC ensures that the data processing and machine learning components remain fully independent from the UI. This separation facilitates parallel development, easier debugging, and improved scalability when deploying new algorithms or updating the frontend. The modular organization between the controller and model layers make MVC practical for applications that expose RESTful APIs and require consistent communication with an external UI.

### 3.3.1. Model(s)

There are four main algorithms - Linear Regression, Ridge Regression, Random Forest, and Gradient Boosting. They provide the core predictive functionality of the application. They form the backend and have been implemented according to object-oriented programming (OOP) principles.

At the highest level, an abstract class named Regressor. This class defines the common interface and functionality which is shared between all the models models. It includes methods for model training and prediction (Fit() and Predict()). Also there is a set of evaluation utilities for computing key performance metrics: coefficient of determination (), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE).

The LinearRegression class implements the core regression logic and serves as the base implementation for linear modeling. The **RidgeRegression** class inherits from *LinearRegression* and overrides the training logic by adding regularization to the weight updates.

The DecisionTreeRegressor class serves as a reusable component rather than a standalone predictive model. It defines the logic for recursive tree construction, which is reused by more complex ensemble algorithms. Both the RandomForestRegressor and GradientBoostingRegressor classes inherit from this component internally. Each of them inherits from *IRegressor* and uses multiple instances of *DecisionTreeRegressor* to accommodate their needs.

The implemented hierarchy ensures high modularity and reusability and scalability. All regression models share a common structure defined by *IRegressor*, while tree-based algorithms extend their functionality by combining multiple decision trees. With this approach there is code duplication and simplifies future expansion, as new regression types can be added by implementing or extending the same base interface.

The high-level class hierarchy, with the relevant inheritance relationships and shared interfaces, is illustrated in{figure}.

{figure}

### 3.3.2. View(s)

### 3.3.3. Controller(s)

# 4. Implementation

# References

Alnajim, Thuraya, et al. “An Intelligent Vehicle Price Estimation Approach Using a Deep Neural Network Model.” *World Electric Vehicle Journal*, vol. 15, no. 8, 2024, https://doi.org/10.3390/wevj15080345.

Amik, Fahad R., et al. “Application of Machine Learning Techniques to Predict the Price of Pre-Owned Cars in Bangladesh.” *Information*, vol. 12, no. 12, 2021, https://doi.org/10.3390/info12120514.

Bergmann, Svenja, and Stefan Feuerriegel. “Machine Learning for Predicting Used Car Resale Prices Using Granular Vehicle Equipment Information.” *Expert Systems with Applications*, vol. 263, Mar. 2025, p. 125640, https://doi.org/10.1016/j.eswa.2024.125640.

Friedman, Jerome H. “Greedy Function Approximation: A Gradient Boosting Machine.” *Annals of Statistics*, vol. 29, no. 5, 2001, pp. 1189–232, https://doi.org/10.1214/aos/1013203451.

Gegic, Enis, et al. “Car Price Prediction Using Machine Learning Techniques.” *TEM Journal*, vol. 8, no. 1, Feb. 2019, pp. 113–18. Academic Search Ultimate, 135048125, *EBSCOhost*, https://doi.org/10.18421/TEM81-16.

Haque, Alimul, et al. “Implication of Different Data Split Ratio on the Performance of Model in Price Prediction of Used Vehicles Using Regression Analysis.” *Data & Metadata*, vol. 3, Jan. 2024, pp. 1–14. Academic Search Ultimate, 181529805, *EBSCOhost*, https://doi.org/10.56294/dm2024425.

J. I. Kang, et al. “Comparing Regression Models Predicting the Price of Used Cars in Big Data.” *2022 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, 2022, pp. 01–04, https://doi.org/10.1109/ICCE-Asia57006.2022.9954633.

Mentch, Lucas, and Siyu Zhou. “Randomization as Regularization: A Degrees of Freedom Explanation for Random Forest Success.” *Journal of Machine Learning Research*, vol. 21, no. 171, 2020, pp. 1–36.

Molnar, Christoph. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. Edited by 2, Christoph Molnar, 2023, https://christophm.github.io/interpretable-ml-book/feature-importance.html. Feature Importance.

Pal, Nabarun, et al. *How Much Is My Car Worth? A Methodology for Predicting Used Cars Prices Using Random Forest*. arXiv preprint. 2017, https://arxiv.org/abs/1711.06970.

Qian, Tingyu. “Used Car Price Prediction by Using XGBoost.” *BCP Business & Management*, vol. 44, Apr. 2023, pp. 62–68, https://doi.org/10.54691/bcpbm.v44i.4794.

Salim, Fadzilah, and Nur Abu. “Used Car Price Estimation: Moving from Linear Regression towards a New S-Curve Model.” *International Journal of Business and Society*, vol. 22, Dec. 2021, pp. 1174–87, https://doi.org/10.33736/ijbs.4293.2021.

Scornet, Erwan. *Trees, Forests, and Impurity-Based Variable Importance*. 2021, https://arxiv.org/abs/2001.04295.

Ting Tin Tin, et al. “Car Depreciation Price Prediction Using Multiple Machine Learning Algorithms.” *Pakistan Journal of Life & Social Sciences*, vol. 22, no. 2, July 2024, pp. 19239–51. Academic Search Ultimate, 185141745, *EBSCOhost*, https://doi.org/10.57239/PJLSS-2024-22.2.001411.

Zhang, Yunyi, and Dimitris N. Politis. *Ridge Regression Revisited: Debiasing, Thresholding and Bootstrap*. 2021, https://arxiv.org/abs/2009.08071.

Zheng, Yufan. “Machine Learning Optimization and Challenges in Used Car Price Prediction.” *ITM Web of Conferences*, vol. 70, Jan. 2025, https://doi.org/10.1051/itmconf/20257004032.