Regression of Used Car Prices

AI6102-GroupA-Machine Learning Methodologies & Applications

Kaggle Challenge

Eugene Ho Hong Zhuang
G2204889B
Nanyang Technological University
eugene_ho@simtech.a-star.edu.sg

Teixayavong SalineLat
G02303887A
Nanyang Technological University
salinela001@e.ntu.edu.sg

Pan Feng
G2405221K
Nanyang Technological University
feng006@e.ntu.edu.sg

https://github.com/genehhz/AI6102---Machine-Learning-Methodologies-Applications

Abstract

This study explores the application of machine learning algorithms to predict used car resale prices using a dataset of 188,533 vehicles with 13 features. The research involved extensive data preprocessing, including handling missing values, feature extraction from unstructured text descriptions of engine and transmission specifications, and recategorization of color variables. Six machine learning models were developed and compared: Ridge Regression, LightGBM, XGBoost, CatBoost, Multi-Layer Perceptron, and Gaussian Naïve Bayes. Using 5-fold cross-validation and root mean squared error (RMSE) as the evaluation metric, LightGBM emerged as the best-performing model with a test RMSE of 63,481.83, significantly outperforming the baseline Ridge Regression, RMSE: 64,743.27.

1 Introduction

The automotive resale market plays a crucial role in the economy, with buyers and sellers relying on accurate price predictions to make informed decisions. Predicting the resale value of used cars is a challenging problem due to the interplay of multiple factors, including vehicle specifications, market trends, and consumer preferences. This report explores the application of machine learning techniques to predict the resale price of used cars based on various features. The primary goal is to build a robust predictive model that can aid stakeholders in estimating the resale value in unseen data instances with high accuracy.

2 Dataset, Attributes and Outcome

The dataset used in this study is obtained from a Kaggle competition [7] focused on predicting the resale price of used cars. The dataset was synthetically generated, but it still provides a rich collection of features that comprehensively describe the vehicles and their conditions, along with the target variable, *price*. The dataset includes both numerical and categorical attributes, offering a balanced mix of quantitative and qualitative information. This structure allows for an in-depth

exploration of feature interactions and an evaluation of the effectiveness of machine learning models for supervised-learning for regression prediction.

The dataset is divided into training and test sets. The training dataset, containing **188,533 rows and 13 features**, includes the target variable *price*, making it suitable for supervised model training. In contrast, the test dataset comprises **125,690 rows and 12 features**, excluding the target variable, and is used for evaluating model performance on unseen data. The full list of variables includes `id`, `brand`, `model`, `model_year`, `milage`, `fuel_type`, `engine`, `transmission`, `ext_col`, `int_col`, `accident`, `clean_title`, and `price`. These variables represent a broad spectrum of car attributes, from technical specifications to ownership history.

The dataset consists of both numerical and categorical attributes, offering a mix of quantitative and qualitative information:

- **Numerical Features**: `model_year` and `milage`
- **Categorical Features**: `id`, `brand`, `model`, `fuel_type`, `engine`, `transmission`, `ext_col`, `int_col`, `accident`, and `clean_title`
- **Target Variable**: The target variable, `price`, is a continuous numerical variable representing the predicted resale value

The dataset offers an excellent opportunity to explore feature interactions and evaluate the effectiveness of machine learning models in a real-world regression problem.

3 Data Preprocessing

3.1 Missing or Null Values

In both the training and test sets, only three variables had missing values: `fuel_type`, `accident` and `clean_title`. The missing value rates were 1.3% for `accident`, 2.7% for `fuel_type` and 11.4% for `clean_title` in both the training and test sets.

Missing values in these variables were addressed using imputation techniques. For the `clean_title` variable, missing entries were filled with the value 'no', indicating the absence of a clean title. Similarly, the `accident` variable was imputed with the value 'None reported' for missing rows. Missing values for the `fuel_type` variable cannot be simply inferred, we employed additional steps to address missing values for this variable. We first extracted information on fuel type embedded within the `engine` variable. Any remaining unfilled missing values are imputed using the mode fuel type of the corresponding car brand.
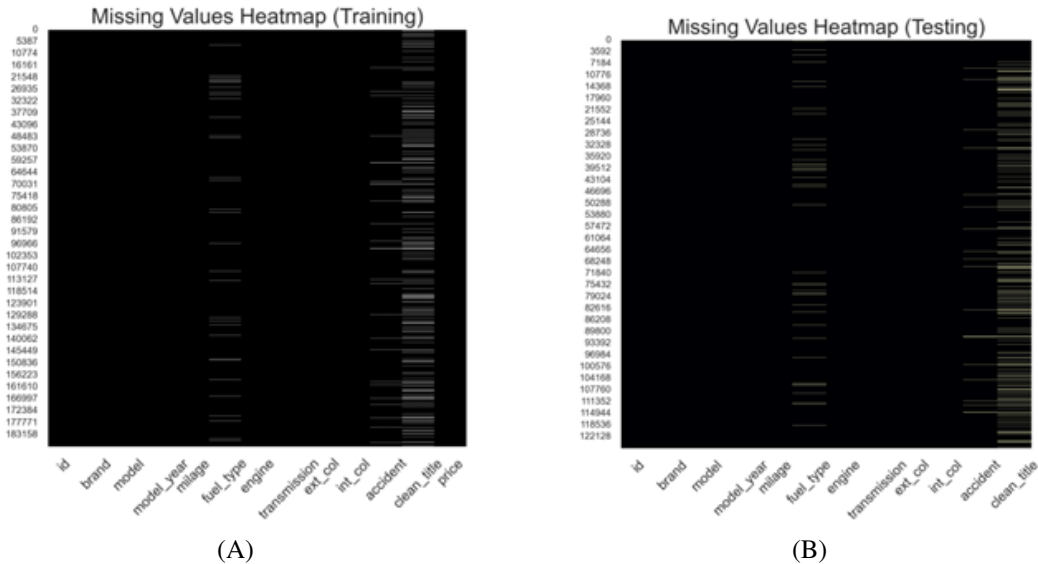

Figure 1: Heatmap illustrating the missing values in training (A) and test (B) set

3.2 Features Extractions and Imputations

The columns `engine` and `transmission` contain long texts for the detailed descriptions of the vehicle's engine specifications and attributes related to the gearbox and its functionality. Texts within `engine` and `transmission` were parsed and extracted for both the training (df) and test (dt) datasets to ensure that these descriptions were meaningful, structured and machine-readable, facilitating improved predictive modeling and analysis. Details extracted from `engine` were stored in new columns: `engine_hp`, `engine_liter`, `engine_cyl`, and `engine_fuel`.

1. **Horsepower** (`engine_hp`): The numeric horsepower value was identified using a regular expression that matches patterns such as 250 hp. Missing values were imputed with NaN.

2. **Engine Capacity in Liters** (`engine_liter`): Liter specifications, e.g., 2.0L, were extracted using a regular expression and converted to numeric values. Missing entries were also imputed with NaN.

   (a) NaN was eventually imputed with median value

3. **Number of Cylinders** (`engine_cyl`): Cylinder counts were derived from patterns like 4 cylinders or Roman numerals (e.g., IV cylinders). These were converted to integers or left as NaN for missing values.

   (a) NaN was eventually imputed with median value

4. **Fuel Type** (`engine_fuel`): Fuel types such as Gasoline, Diesel, Electric, Hybrid, E85 Flex Fuel, and Plug-In Hybrid were identified from a predefined list of options. If no match was found, the value was set to NaN. Electric and Hybrid were manually imputed using the following rules:

   (a) Electric was imputed for cars that are from electric vehicles-based brands, including Tesla, Rivian, Lucid, and Polestar.

   (b) Hybrid was imputed for cars with imputed cars with imputed Electric but also showed engine capacity.

Key extracted features from `transmission` include:

1. **Number of Gears** (`gears`): Patterns like 6-speed or 6 speed were parsed to extract the number of gears as integers. Special cases like single-speed were normalized to 1. Missing values were left as None.

2. **Transmission Type** (`transmission_type`): Transmission types were categorized into values such as Automatic, Manual, CVT, AMT, Dual_Shift and Electronically_controlled.

3. **Special Features** (`special_features`): Additional transmission features like Dual Shift Mode, Auto-Shift, Overdrive, Variable, and Electronically Controlled were identified and recorded. Missing values were left as 'missing', pertaining to not applicable.

4. **Transmission Designation** (`transmission_designation`): Common transmission abbreviations such as A/T, M/T, DCT, and CVT were extracted to create a standard designation field. Missing values were left as 'missing', pertaining to not applicable.

3.3 Other transformation and Encoding

3.3.1 Recategorizing Interior and Exterior Car Colour

The `ext_col` (exterior color) and `int_col` (interior color) variables were pre-processed by grouping specific hues or tones into broader base color categories. This step reduced the complexity of analyzing the wide variety of descriptive color names in the dataset. We matched the text of the original color description to any entries within a predefined list of base colors: [`'white'`, `'black'`, `'grey'`, `'gray'`, `'blue'`, `'red'`, `'yellow'`, `'silver'`, `'green'`, `'beige'`, `'gold'`, `'orange'`, `'brown'`, `'ebony'`, `'purple'`]. If a match was found, the function returned the corresponding base color. If no match was identified, the value was categorized as "uncommon" to capture rare or non-standard colors.

3.3.2 Creating and Dropping Variables

- The variable `model_year` was converted into `Vehicle_Age` by substituting it from current year (2024)

- The binary variable `Is_Luxury_Brand` was created. Cars were deemed to be luxuries if their brands were either "mercedes-benz", "bmw", "audi", "porsche", "lexus", "cadillac", "jaguar", "bentley", "genesis", "maserati", "lamborghini", "rolls-royce", "ferrari", "mclaren", "aston", "lotus", "bugatti", or "maybach".

- Variables that were dropped include `id`, `model`, `transmission`, `engine`, `engine_fuel`, and `model_year`.

3.3.3 One-hot encoding of categorical variables

- The OneHotEncoder function from sklearn (preprocessing) was applied on categorical variables to one-hot encode them, making them more amenable as inputs into models.

- The seven final categorical variables and their levels are:

Table 1: Categorical variable and their corresponding levels

Variable	Levels
`brand`	57
`fuel_type`	6
`ext_col`	15
`int_col`	14
`clean_title`	2
`transmission_type`	9
`Special_features`	6

The full list of features and their details after preprocessing can be seen in Appendix Table A.1.

4 Exploratory Data Analysis (EDA) and Visualisation

EDA was conducted to uncover underlying patterns, relationships, and anomalies in the dataset. Both univariate and multivariate visualizations were employed to understand the distribution of individual features and their interactions with the target variable. Insights from EDA were instrumental in shaping feature selection and subsequent model design.

4.1 Continuous Variables

4.1.1 Price Distribution

From the distribution plots above, we can see that resale price is highly skewed (right-skewed), with 94% of cars priced under $100,000. Car prices ranged from $2,000 – $2,954,083.

| (A) | (B) |
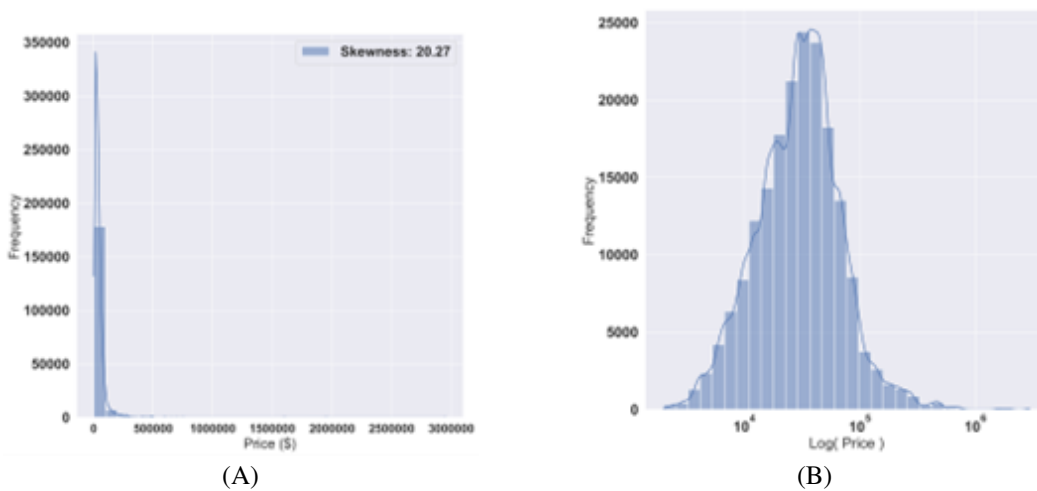
Figure 2: (A) Frequency distribution of mileage, (B) Scatterplot of Price against Mileages

4.1.2 Mileage Distribution

Mileage distribution is acceptable, and shows a negative relationship with car prices (i.e., cars with higher mileages have lower prices).

| (A) | (B) |
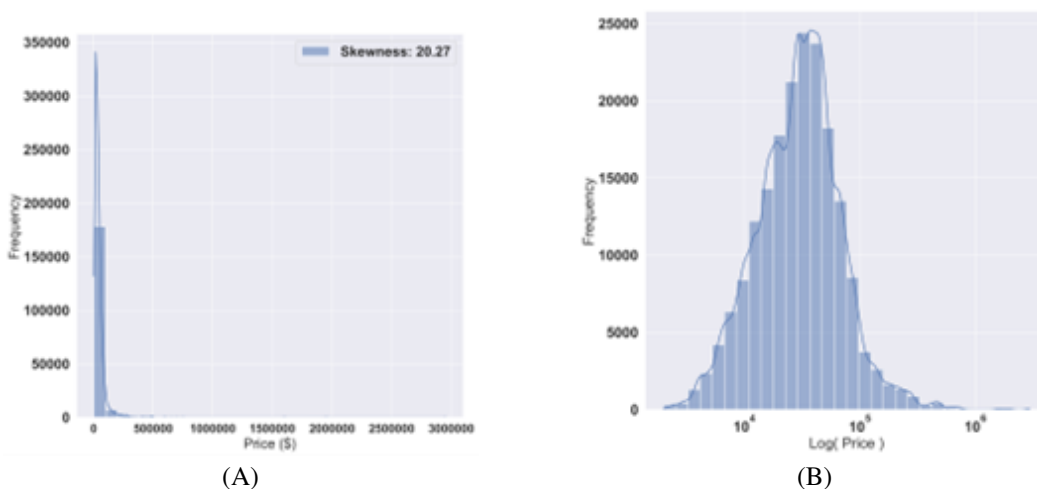
Figure 3: Frequency distribution of price (A) and log(price) (B)

4.1.3 Age of Vehicle Distribution

Vehicle age distribution is acceptable, and shows a weak negative relationship with car prices.

(A)   (B)
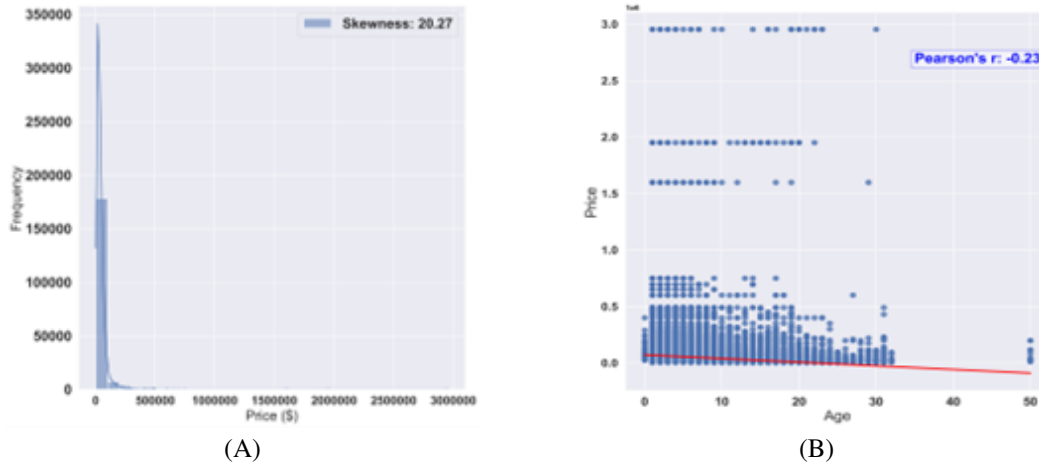
Figure 4: (A) Frequency distribution of vehicle Age, (B) Scatterplot of Price against Age with pearson correlation coefficient of these 2 variables

4.2 Categorical Variables

4.2.1 Car Brands

Five most common car brands include Ford, Mercedes-Benz, BMW, Chevrolet, and Audi, and they are priced in mid- to upper-range in terms of average car prices. Brands with the highest resale values were Bugatti, Lambhorgini, Rolls-Royce, Bentley, and McLaren, reselling at more $120,000 on average. On the other hand, brands with the lowest resale values included Scion, Hammer, Mini, Saturn, and Mitsubishi, reselling at $20,000 or less on average. Expectedly, brands carried great significance in the resale values of cars.
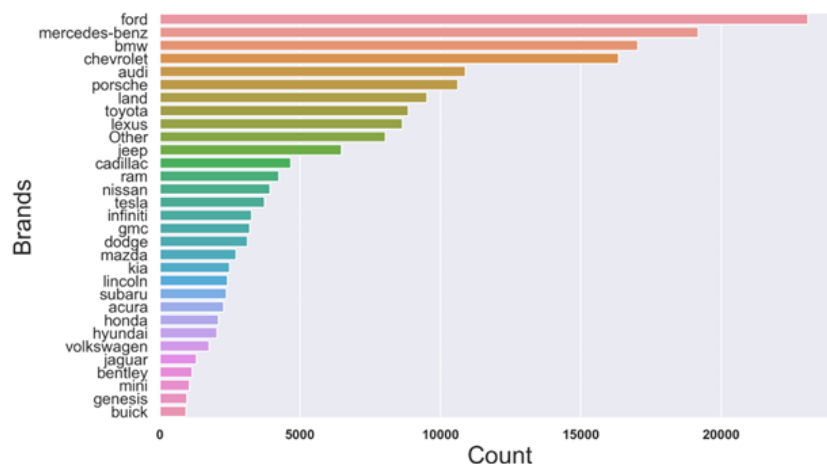
Figure 5: Frequency distribution of car brands by count
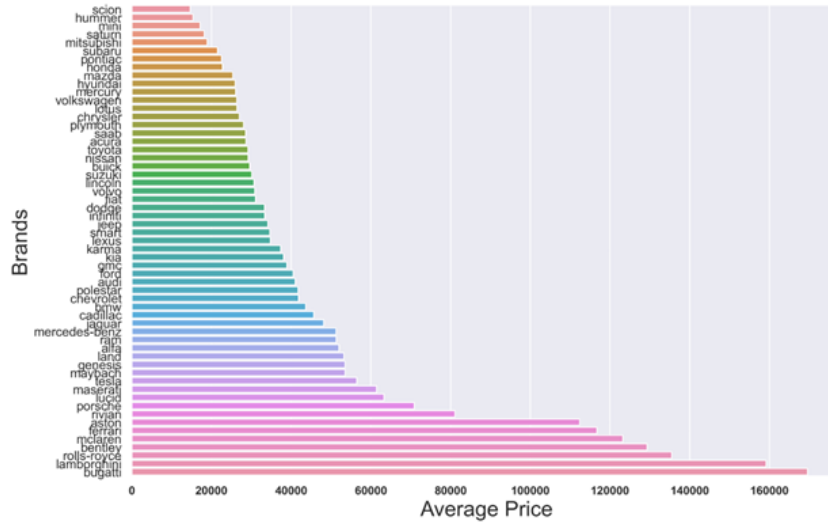
Figure 6: Average price for each car brands

4.2.2 Fuel Types

The most prevalent fuel types in the dataset was gasoline (88%), indicating that the feature shows marked skewness. Fuel types seem to also affect the average resale prices, with all-electric and hybrid cars having the two highest resale values ($50,000) while gasoline ($43,308) and flex-fuel cars ($26,698) had the lowest average resale values.

(A)   (B)
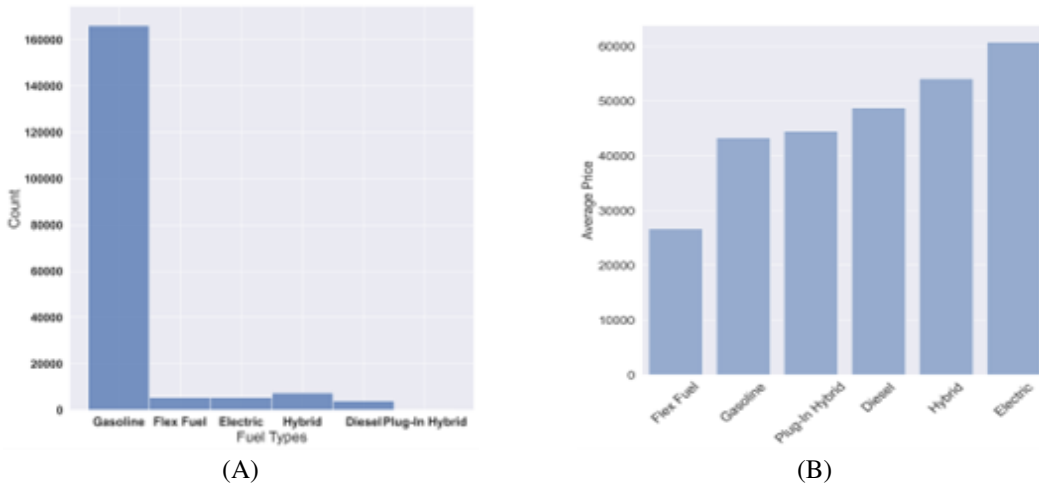
Figure 7: (A) Frequency distribution of fuel types by count, (B) Average price of different fuel types

4.2.3 Color Popularity and Price

The most common exterior car colors were black, white, gray, and silver (all >10% of all car colors). Though, among these, only gray and black cars had higher than average resale values. Other colors that managed to receive good resale prices included uncommon colors, green, orange, and yellow.

There was also significant variability in interior colors and their relationships with resale price. Black was by far the most common interior color (59% of all cars), and had higher than average resale price. Beige and gray were other two common interior colors, but they did not have a high resale price. Interestingly, yellow interior, a rare interior color, had an unusually high resale value at $106,548.

7
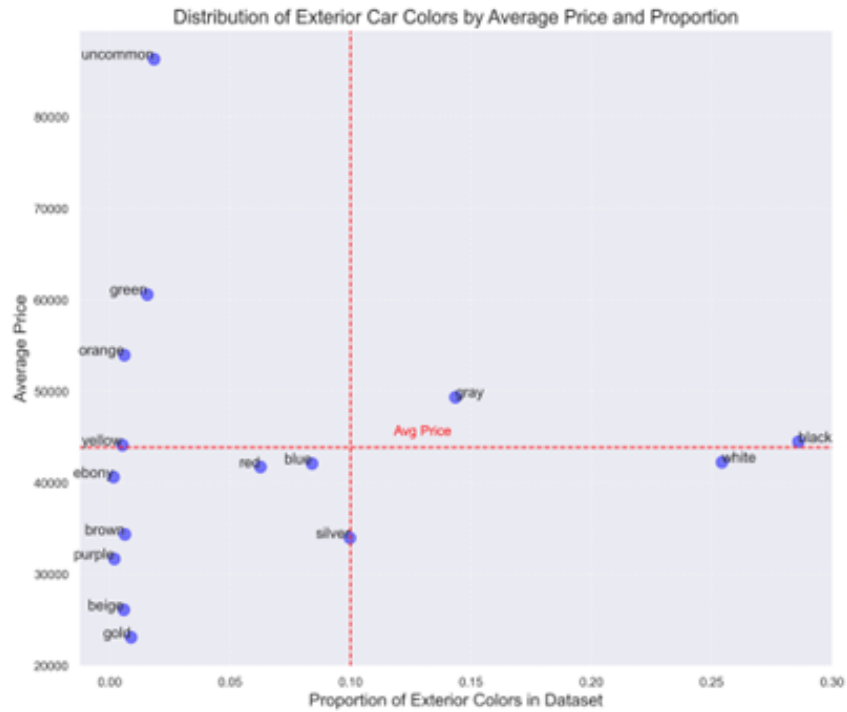
Figure 8: Scatterplot of Average Price against Exterior Colours
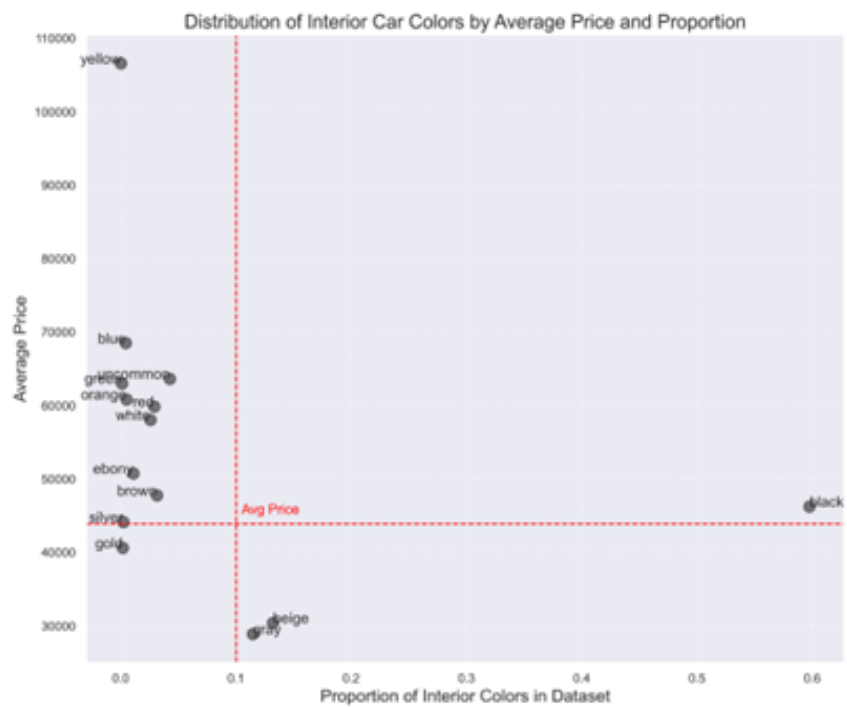
Figure 9: Scatterplot of Average Price against Interior Colors

4.2.4 Engine and Transmission Configurations

From Figure 10, there is moderate skewness for `engine_hp` and `engine_liter`, and weak relationship with price.
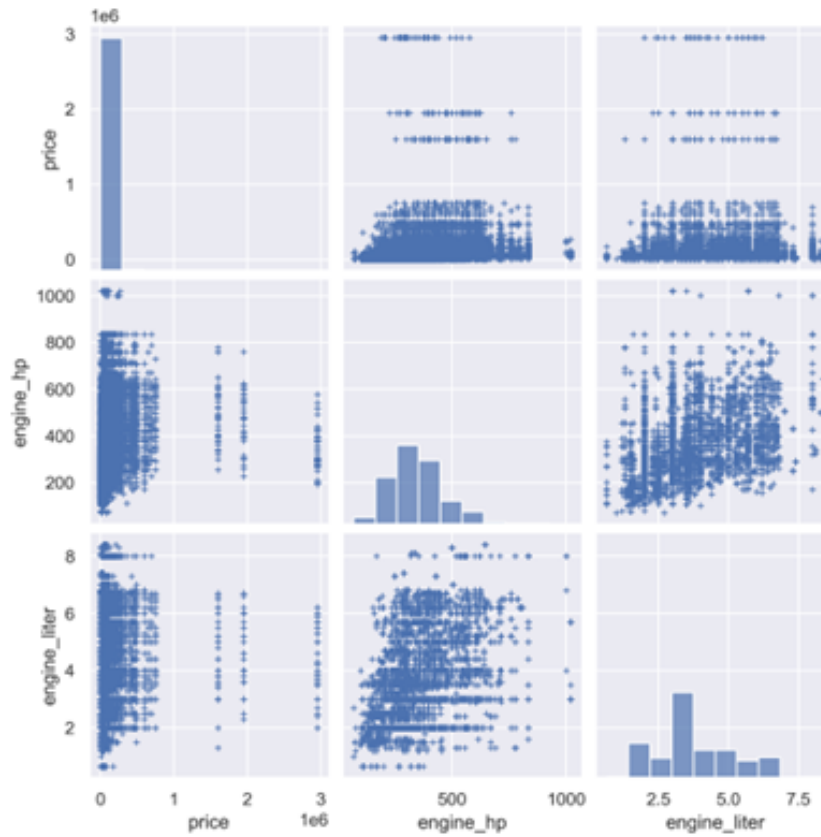

Figure 10: Pairplot illustrating the correlations of multi-variable - `price`, `engine_hp` and `engine_litre`

From Figure 11, for both `gears` and `transmission_types`, median car price did not vary across categories. However, the median price seemed to be higher for transmission with Auto-Shift feature. Price also seemed to increase with the number of engine cylinders. Across all categories of each feature, there are outlier prices to suggest skewed distributions.
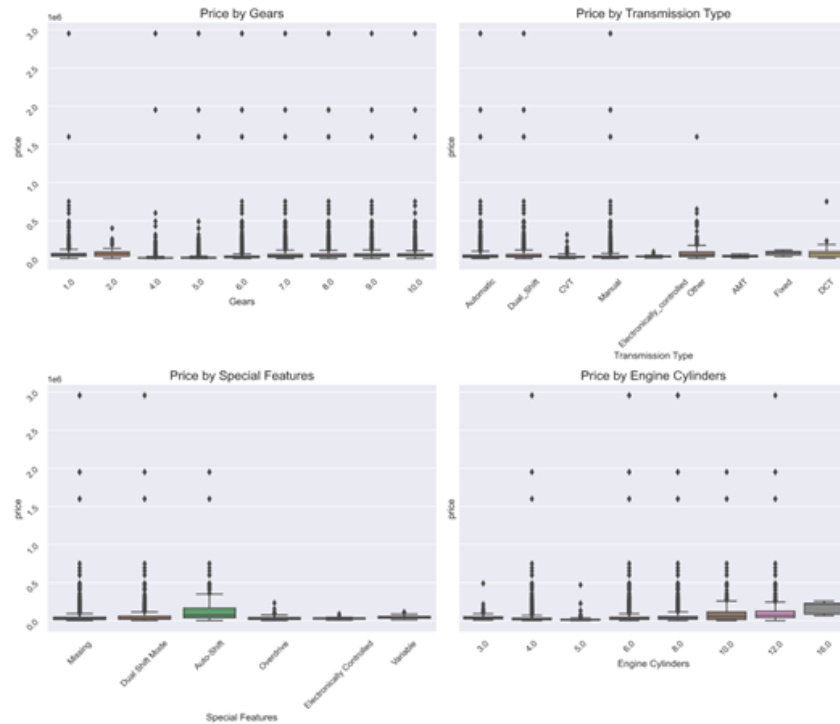
Figure 11: Boxplots of price against different variables: gears, transmission type, special features and engine cylinders

4.2.5 Other Variables

The history of reported accidents also significantly impacts a car's resale price. Vehicles with a reported history of accidents generally exhibit lower average prices compared to those with no accidents. For instance, cars with a clean accident history have an average price of **$49,122**, while those with a reported accident history are priced around **$25,334**. This difference reflects buyer preferences for cars perceived as more reliable and in better condition, emphasizing the importance of accident-free histories in the used car market.

A car's title status is a critical factor in determining its resale value. Vehicles with a clean title, indicating no major damage, theft, or salvage history, command higher prices than those without. The analysis reveals that cars with clean titles have an average price of **$63,567**, whereas cars with branded or salvage titles average around **$41,354**. This stark difference highlights the significant role of title status in buyer confidence and perceived vehicle quality.

The distinction between luxury and non-luxury brands is a pivotal variable influencing car prices. Luxury brands, such as Mercedes, BMW, and Lexus, maintain higher average prices due to their brand reputation, premium features, and perceived status. The analysis shows that luxury brands have an average price of **$52,368**, compared to **$37,929** for non-luxury brands. This disparity underscores the enduring demand and value retention of luxury vehicles in the resale market.

5 Model Development

We employed a variety of machine learning algorithms, ranging from simple classical models to more complex architectures, to predict car prices. The prediction task involves supervised regression problem, where the goal was to estimate the target variable, `price`, for unseen test instances. By employing models of varying complexity, we aimed to evaluate how incremental increases in model sophistication improved predictive performance on both the validation and test sets. As the baseline, we used linear regression with L2 regularization (i.e., ridge regression), a simple yet effective algorithm. This approach allowed us to model linear relationships within the data and served as a benchmark to understand the added value of more complex algorithms. Ridge regression provided a clear starting point for interpreting how regularization controls overfitting while retaining simplicity and computational efficiency.

Building on this, we incorporated three advanced gradient-boosting models [2]: LightGBM [4], XGBoost [1], and CatBoost [6]. These algorithms are highly adept at capturing non-linear relationships in the data and are particularly effective in handling large numbers of features, including categorical variables. Their ability to perform efficient feature selection and mitigate overfitting through ensemble techniques ensure optimal performance.

Next, we evaluated the performance of a Multi-Layer Perceptron (MLP) [5], a fundamental deep learning architecture. Unlike tree-based models, MLPs excel at learning complex, non-linear interactions between features through their multi-layered structure of fully connected neurons. This made the MLP a valuable addition to our study, particularly for exploring intricate patterns in high-dimensional data.

Lastly, we assessed the Naïve Bayes classifier for its probabilistic approach to modeling categorical data [10]. While traditionally used for classification tasks, its application in this study offered a unique perspective on how simple probabilistic assumptions could capture the relationships in categorical features and contribute to the overall predictive task. Below, we provide detailed descriptions of each model used in our analysis.

5.1 Training Procedure and Model Selection

The training of each model adheres to a consistent 5-fold cross-validation (CV) strategy to ensure robust performance evaluation [9]. In this approach, the dataset is divided into five equal-sized folds. During each iteration, four folds are used for training while the remaining fold serves as the validation set. This process is repeated five times, ensuring that every fold is used as a validation set exactly once. The average Root Mean Squared Error (RMSE) across the five validation folds is computed to evaluate the model's performance during training and hyperparameter tuning.

After the cross-validation process, each model is further evaluated on a held-out test set to calculate the test RMSE. This ensures that the model's performance is assessed on entirely unseen data, providing a reliable estimate of its generalization capability. For the final stage, the model with the best validation RMSE is re-trained on the entire training dataset (excluding the test set) to fully utilize all available data. This re-trained model is then used to generate final predictions on the test set, and the resulting RMSE is recorded as the model's definitive performance metric. The consistency in training methodology across the Ridge Regression, LightGBM, XGBoost, CatBoost, and MLP models ensures fair comparisons, while the comprehensive use of CV and test evaluation provides a thorough assessment of each model's predictive capability.

5.2 Performance Metric

In this Kaggle challenge, the scores are evaluated based on the Root Mean Square Error (RMSE) between predicted price and original price as seen in the equation below. Hence, RMSE was employed to evaluate model performance across training, validation, and testing phases, providing a consistent metric for assessing prediction accuracy.

$$RMSE = \left(\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \right)^{1/2} \tag{1}$$

5.3 Machine Learning Models

5.3.1 Ridge Regression with L2 Regularisation

Ridge regression was chosen as the baseline model due to its simplicity and interpretability. This algorithm extends linear regression by incorporating L2 regularization, which penalizes large coefficients to mitigate overfitting. The regularization strength is controlled by the hyperparameter λ, where a larger λ results in stronger penalization, and a smaller λ approaches standard linear regression. After fine-tuning, an optimal λ of 0.5 was adopted. The parameters (i.e., weights) of the model were computed using the analytical closed-form solution (using sklearn's `LinearRegression`) rather than iterative gradient descent.

5.3.2 LightGBM (LGBM)

LightGBM employs gradient boosting with a leaf-wise tree growth strategy, expanding the leaf that maximizes the reduction in loss at each step. This minimizes the objective function:

$$L(\Theta) = \sum_{i=1}^{n} \ell(y_i, \hat{y}_i) + \lambda \sum_{j=1}^{T} \|w_j\|^2 \tag{2}$$

where $\ell(y_i, \hat{y}_i)$ is the loss (e.g., root mean squared error), w_j are the leaf weights, T is the number of leaves, and λ is the regularization term.

LightGBM optimizes using histogram-based binning to accelerate split finding and uses Gradient-Based One-Side Sampling (GOSS) to retain instances with larger gradients, ensuring effective learning while reducing data size. Exclusive Feature Bundling (EFB) combines sparse features into a single feature, reducing dimensionality without loss of information. LightGBM's efficiency stems from these optimizations, enabling fast training on large datasets.

To further optimize the model, we mainly experimentally tuned `num_leaves`, `max_depth`, `learning_rate`, `min_data_in_leaf`, `feature_fraction`, `bagging_fraction` and `bagging_freq` as these parameters impact the model the most. As can seen from the photo below, different versions of lightgbm model were submitted, and the RMSE score increased from 63542.09 (version 1) to 6348.83 (version 3) after tuning the above mentioned parameters.

Table 2: Final hyperparameters used for LightGBM

Hyperparameters	Value
num_leaves	150
max_depth	15
learning_rate	0.02
n_estimators	1000
metric	RMSE
subsample	0.6
colsample_bytree	0.9
reg_alpha	0.2
reg_lambda	1
objective	regression
min_data_in_leaf	350
feature_fraction	0.75
bagging_fraction	0.6
bagging_freq	2
random_state	42
min_child_weight	0.1
verbose	-1

5.3.3 eXtreme Gradient Boosting (XGBoost)

XGBoost is a highly scalable tree-boosting system designed with regularization to control overfitting. It employs second-order gradient-based optimization, enabling more precise updates than standard gradient boosting methods. With built-in support for parallel computation, XGBoost achieves fast training times while maintaining high accuracy. Its versatility and proven performance in regression tasks make it a staple in machine learning competitions and applications. To further optimize the model, we mainly experimentally tuned `max_depth`, `subsample`, `colsample_bytree`, `gamma` and `reg_alpha`.

Table 3: Final hyperparameters used for XGBoost

Hyperparameters	Value
n_estimators	1000
max_depth	10
learning_rate	0.01
subsample	0.7
colsample_bytree	0.7
gamma	2
reg_alpha	0.1
reg_lambda	1
objective	reg:squarederror
early_stopping_rounds	50

5.3.4 CatBoost

CatBoost is a variant of the gradient boosting framework that, to some extent, is similar to LightGBM. It employs ordered boosting, a technique that prevents target leakage by training trees on permutations of the dataset. Additionally, CatBoost uses *mean target encoding* for categorical variables, where categories are encoded based on their conditional means:

$$\text{Category Value} = \frac{\sum_{i=1}^{n} y_i}{\text{Count} + \alpha}, \tag{3}$$

where α is a smoothing parameter to mitigate the effect of small categories, `Count` represents the total number of data points or occurrences belonging to the category in the dataset, and `Category Value` is the encoded value assigned to the category after the mean target encoding.

The algorithm constructs symmetric trees, ensuring consistent splits across all branches, which reduces inference time. CatBoost efficiently handles categorical features natively, supports GPU acceleration, and employs dynamic learning rates, enabling faster convergence while preserving model stability. To further optimize the model, we primarily tuned hyperparameters including learning rate, depth, random strength, L2 leaf regularization, maximum leaves, and fold permutation block through experimentation.

Table 4: Final hyperparameters used for CatBoost

Hyperparameters	Value
learning_rate	0.02
iterations	1500
depth	10
random_strength	0.01
l2_leaf_reg	0.8
task_type	CPU
max_leaves	256
fold_permutation_block	64
random_seed	42
verbose	False
grow_policy	Lossguide

5.3.5 Multi-Layer Perceptron (MLP)

The Multilayer Perceptron (MLP) is one of the fundamental deep learning models comprising multiple hidden neural layers designed to learn and capture complex, non-linear data relationships for predictive or classification purposes.

Our experiments involved fine-tuning various hyperparameters. Specifically, we adjusted the learning rates (0.01, 0.001, and 0.0001), experimented with different network architectures (4 and 6 layers), and evaluated the models with and without L2 regularization, as shown in Table 5. However, several hyperparameters were kept constant across all experiments, including 1000 epochs and an L2 regularization coefficient (λ) of 0.01 when applied. For optimization, we utilized the Adaptive Moment Estimation (Adam) optimizer, with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Adam was chosen for its ability to combine the benefits of RMSprop and momentum optimization techniques, which generally leads to faster convergence. In this study, we have referenced and drawn guidance from other Kaggle participants who evaluated a range of hyperparameters, and have further built upon their work by conducting additional experiments to refine and validate the models [3, 11].

Table 5: Hyperparameters used for MLP

Model	Regularisation	Learning Rate	Number of Layers	Validation loss (5-fold)
1	None	0.001	4	76824.47
2	L2	0.01	4	75773.10
3	L2	0.001	4	75966.84
4	L2	0.0001	4	88090.79
5*	**None**	**0.001**	**6**	**74934.64**
6	L2	0.001	6	73665.02

*Selected model and hyperparameters used for final test set evaluation and comparison with other machine learning models

Our experiments revealed that incorporating L2 regularization significantly improved the validation loss. As shown in Figure 12, learning rates of 0.01 and 0.001 exhibited comparable performance, though the 0.01 configuration showed instability around epoch 900, while a learning rate of 0.0001 underperformed. Furthermore, Figure 13 demonstrates that the 6-layer architecture achieved slightly better performance than the 4-layer model without incurring significant computational overhead. There is no indication of overfitting, as both the training and validation losses decreased consistently without diverging, shown in Figure 14. Based on these findings, the optimal configuration is a 6-layer model with L2 regularization and learning rate of 0.001.
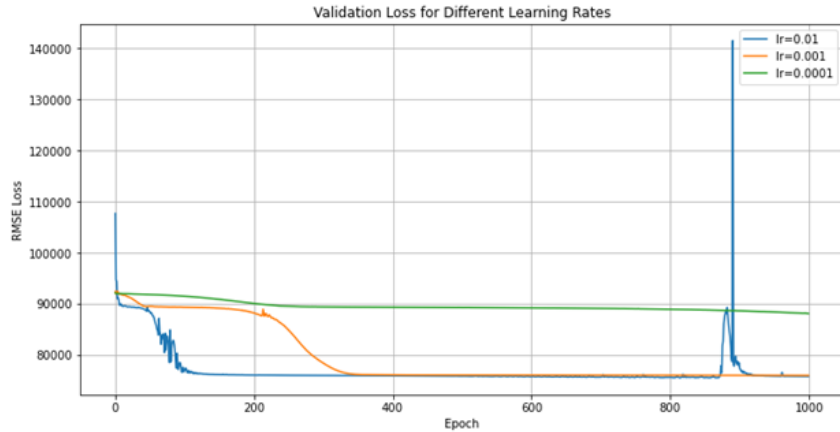
Figure 12: Validation loss across 1000 epochs for different learning rates: 0.01, 0.001, and 0.0001

14
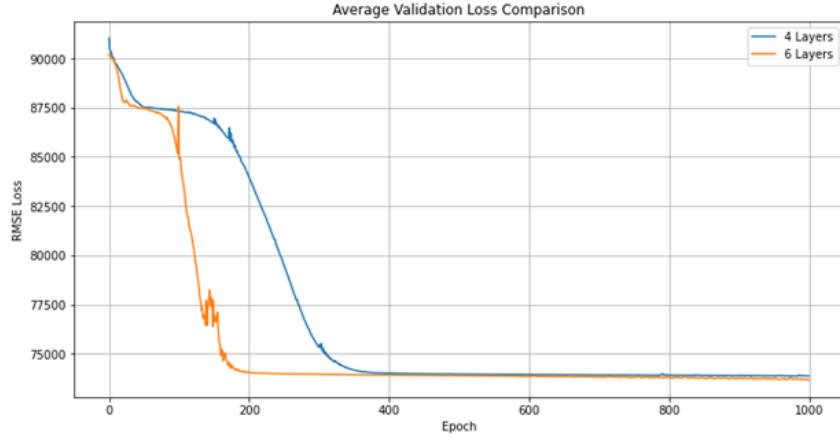
Figure 13: 5-fold Average validation loss across 1000 epochs for different layers: 4 and 6
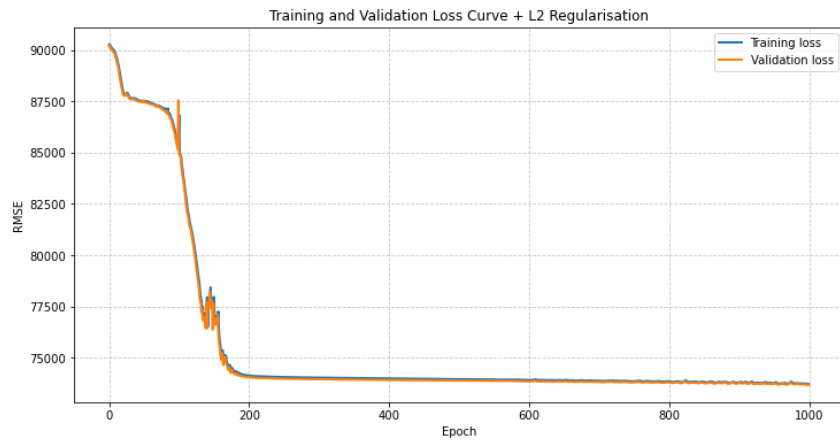

Figure 14: Training and validation losses for Model 6: 6 layers, learning rate: 0.001, with L2 regularization

5.3.6 Gaussian Naïve Bayes

Gaussian Naïve Bayes is a probabilistic classifier that extends Naïve Bayes by assuming features follow a Gaussian distribution, chosen for our analysis since our target variable (`price`) and several features are continuous. Rather than discretizing continuous features through binning, which could lose discriminative information, we maintain their continuous nature by assuming Gaussian distributions. While this handles continuous data more appropriately, the fundamental Naï ve Bayes assumption of conditional independence between features given the target variable remains.

5.4 Training Procedure and Model Selection

The training of each model adheres to a consistent 5-fold cross-validation (CV) strategy to ensure robust performance evaluation. In this approach, the dataset is divided into five equal-sized folds. During each iteration, four folds are used for training while the remaining fold serves as the validation set. This process is repeated five times, ensuring that every fold is used as a validation set exactly once. The average Root Mean Squared Error (RMSE) across the five validation folds is computed to evaluate the model's performance during training and hyperparameter tuning.

After the cross-validation process, each model is further evaluated on a held-out test set to calculate the test RMSE. This ensures that the model's performance is assessed on entirely unseen data, providing a reliable estimate of its generalization capability. For the final stage, the model with the

best validation RMSE is re-trained on the entire training dataset (excluding the test set) to fully utilize all available data. This re-trained model is then used to generate final predictions on the test set, and the resulting RMSE is recorded as the model's definitive performance metric. The consistency in training methodology across the Ridge Regression, LightGBM, XGBoost, CatBoost, and MLP models ensures fair comparisons, while the comprehensive use of CV and test evaluation provides a thorough assessment of each model's predictive capability.

6   Results and Discussion

In this study, we aimed to predict car prices using machine learning models by leveraging a rich dataset of structured and unstructured features. Preprocessing was a critical component, involving the handling of missing values, extraction of information from unstructured text (e.g., engine and transmission details), and reducing the complexity of color variables by grouping nuanced tones into base color categories. These preprocessing steps ensured the dataset was clean, interpretable, and suitable for use in machine learning models. The models' performance was evaluated using both the 5-fold cross validation and the private leaderboard score on Kaggle, which calculates the RMSE on approximately 80% of the unseen test data. The comparative results across models are presented in Table 6, and the actual Kaggle submission screenshots can be found in Appendix Figures A.2 and A.3.

Table 6: Comparison of Model Performance: 5-Fold Cross-Validation RMSE and Test Set Results.

Model	Average RMSE across validation folds	RMSE on Test Set*
LightGBM	**72,806.06**	**63,481.83**
XGBoost	73,235.15	63,671.16
Catboost	73,555.09	64,317.89
Multi-Layer Perceptron	73,665.02	64,529.35
Gaussian Naïve Bayes	97,005.27	90,914.16
Ridge Regression	**73,742.24**	**64,743.27**

Test set consists of unseen data, and the predictions are evaluated through Kaggle's platform.

Six machine learning models, ranging from simple regression to advanced gradient boosting and neural network models, were developed and compared. LightGBM emerged as the best-performing model, achieving the lowest RMSE on both validation and test sets. Its leaf-wise tree growth strategy, histogram-based learning, and ability to handle categorical variables natively allowed it to effectively model the complex, non-linear relationships in the dataset . Compared to Ridge Regression, which serves as a baseline linear model, LightGBM demonstrated superior performance, highlighting the need for more sophisticated algorithms to capture the nuances of car pricing.

The Multi-Layer Perceptron (MLP), a neural network model, exhibited a slightly higher RMSE compared to LightGBM, likely due to its increased sensitivity to hyperparameter tuning. While the current architecture does not exhibit overfitting, suggesting the potential for expanding the number of layers and nodes per layer, the performance could also be improved by incorporating additional features, such as learning rate scheduling. However, this would introduce more hyperparameters that require fine-tuning. Although these adjustments may reduce RMSE, they come with significantly higher computational resource demands and increase training duration compared to the more efficient tree-based models like LightGBM.

Gaussian Naïve Bayes, on the other hand, exhibited the highest error due to its assumption of feature conditional independence, which is unrealistic for this dataset where variables like mileage, engine size, and year are highly correlated with price. This probabilistic approach also struggles with numerical and continuous features, leading to suboptimal performance.

A key limitation of this study was the highly skewed distribution of car prices, with a long tail of high-value cars. This right skewness likely introduced systemic errors, where models tended to underpredict prices for expensive cars and overpredict for cheaper ones. The RMSE metric, while useful for comparing overall model performance, may not fully capture these systemic errors, as it aggregates errors without distinguishing their directional biases. Future work could explore alternative metrics or loss functions (e.g., mean absolute percentage error) to better account for such biases [8].

16

7 Conclusion

This study investigates the application of machine learning techniques for predicting resale prices of used cars, leveraging a comprehensive dataset comprising over 188,000 vehicle records. Following rigorous data preprocessing and feature engineering, six machine learning models were systematically evaluated. Among these, the LightGBM algorithm demonstrated superior performance, achieving a test root mean square error (RMSE) of 63,481.83, significantly outperforming the baseline Ridge Regression model (RMSE: 64,743.27). The results underscore the efficacy of advanced machine learning approaches in capturing the complexity of real-world car pricing data, characterized by non-linear relationships and intricate feature interactions. LightGBM's strengths in balancing accuracy, computational efficiency, and interpretability were particularly noteworthy. However, the study also highlights areas for further improvement, including addressing issues related to data skewness and prediction biases. These findings contribute to the growing body of research on automotive price prediction and demonstrate the practical utility of machine learning in addressing complex regression challenges.

References

[1] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, NY, USA, 2016. Association for Computing Machinery.

[2] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[3] Kazuhiro Fujimori. Prediction by deep learning regression model, 2024. Kaggle, https://www.kaggle.com/code/kazuhirofujimori/prediction-by-deeplearning-regression-model.

[4] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[5] Marius-Constantin Popescu, Valentina E. Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. Multilayer perceptron and neural networks. *WSEAS Trans. Cir. and Sys.*, 8(7):579–588, July 2009.

[6] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 6639–6649, Red Hook, NY, USA, 2018. Curran Associates Inc.

[7] Walter Reade and Ashley Chow. Regression of used car prices. `https://kaggle.com/competitions/playground-series-s4e9`, 2024. Kaggle, https://kaggle.com/competitions/playground-series-s4e9.

[8] Chris Tofallis. A better measure of relative prediction accuracy for model selection and model estimation. *Journal of the Operational Research Society*, 66(8):1352–1362, August 2015.

[9] Tzu-Tsung Wong and Po-Yang Yeh. Reliable accuracy estimates from k-fold cross validation. *IEEE Transactions on Knowledge and Data Engineering*, 32(8):1586–1594, 2019.

[10] Feng-Jen Yang. An implementation of naive bayes classifier. In *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 301–306, 2018.

[11] Danish Yousuf. Using neural network for car price prediction, 2024. Kaggle, https://www.kaggle.com/code/kazuhirofujimori/prediction-by-deeplearning-regression-model.

A Appendix / supplemental material

Table A.1: Features in the dataset, source columns, derived columns, variable types, and characteristics

Source	Column	Type	Levels	Dropped
id	id	numerical	–	Yes
model_year	model_year	numerical	–	Yes
milage	milage	numerical	–	No
accident	accident	categorical	2	No
price	price	numerical	–	No
engine	engine_hp	numerical	–	No
engine	engine_liter	numerical	–	No
engine	engine_cyl	numerical	–	No
transmission	gears	numerical	–	No
model_year	vehicle_age	numerical	–	No
brand	is_luxury_brand	categorical	55	No
model	model	categorical	1897	Yes
fuel_type	fuel_type	categorical	6	No
engine	engine	categorical	1117	Yes
transmission	transmission	categorical	52	Yes
ext_col	ext_col	categorical	15	No
int_col	int_col	categorical	16	No
clean_title	clean_title	categorical	2	No
engine	engine_fuel	categorical	6	No
transmission	transmission_type	categorical	6	No
transmission	special_features	categorical	6	No
transmission	transmission_designation	categorical	5	No

Figure A.2: Screenshots for MLP, Gaussian naïve Bayes and Ridge Regression

Figure A.3: Screenshots for LGBM, XGB, and CB