Balancing Interpretability and Accuracy in Predictive Modeling: A Case Study on Used Car Price Estimation

Siyuan Feng and **Xinyuan He** and **Yingyu Cao**
Equal contribution
Harrisburg University of Science and Technology
sfeng3@my.harrisburgu.edu
xhe5@my.harrisburgu.edu
ycao11@my.harrisburgu.edu

Abstract

The used car market has experienced significant growth in recent years, necessitating accurate pricing mechanisms to facilitate fair transactions between buyers and sellers. Traditional pricing methods, often reliant on manual assessments, fail to capture the complex interactions among various factors influencing car prices. This project aims to develop a predictive model for second-hand car prices using advanced machine learning techniques. By analyzing historical data from Craigslist, the model identifies key features affecting car prices and provides accurate estimates, thereby enhancing market transparency and efficiency. Our approach involves comprehensive data visualization, preprocessing, feature engineering, and model evaluation to ensure the robustness of the predictions. The final model, a Gradient Boosting Regressor, achieved a test accuracy of 79%.

1 Introduction

The used car market has grown substantially over the past decade, driven by economic constraints, improvements in vehicle durability, and increased consumer awareness. Accurate pricing of second-hand cars is crucial for both sellers and buyers to make informed decisions. The challenge lies in accounting for the multitude of variables that influence a car's price, such as make, model, age, mileage, and condition. Traditional pricing methods often rely on manual assessments or heuristic-based approaches, which may not adequately capture the complex patterns within the data.

To address these challenges, this project leverages a dataset from Kaggle, comprising various attributes of vehicles listed on Craigslist. Our methodology involves several key steps aimed at building a robust predictive model:

- **Data Visualization:** Understanding the dataset through various visualizations to identify patterns and anomalies.
- **Data Preprocessing and Imputation:** Using pipelines to handle missing values and prepare the data for modeling.
- **Baseline Linear Model:** Performing grid search on Linear Regression, Elastic Net, and LinearSVR to establish a baseline model.
- **Feature Engineering:** Implementing imputation, encoding, and grid search on Random Forest and Gradient Boosting models to optimize performance.
- **Feature Selection:** Utilizing permutation importance and auto feature selection with Random Forest to identify the most significant features.
- **Explainable Model:** Building an explainable model that closely matches the performance of the best model.

The ultimate goal of this project is to create a machine learning model that accurately predicts the price of used cars, thereby providing a valuable tool for buyers and sellers to make informed decisions. The final model, a Gradient Boosting Regressor, achieved a test accuracy of 79%, demonstrating the effectiveness of our approach.

2 Related Work

The prediction of second-hand car prices using machine learning has been a subject of research for several years. Various approaches have been explored, ranging from traditional regression models to more advanced machine learning techniques.

Traditional Methods: Early studies often employed linear regression to model the relationship between car prices and their features. For instance,

Chen and Zhao (2015) used multiple linear regression to analyze the influence of factors such as mileage and age on car prices, providing a foundation for more sophisticated models. However, these models often fail to capture non-linear relationships and interactions between variables. Another approach utilized polynomial regression to account for some level of non-linearity, but still struggled with complex interactions (Ahmad et al., 2017) .

Advanced Machine Learning Techniques: More recent work has focused on leveraging complex algorithms such as decision trees, random forests, and gradient boosting machines. For instance, Nahar et al. (2015) used random forest regression to predict used car prices, demonstrating improved accuracy over linear models. Similarly, another research applied gradient boosting techniques to capture non-linear dependencies and interactions between car features (Zhang et al., 2017).

Deep Learning Approaches: Some researchers have explored the use of deep learning models, such as neural networks, to predict car prices. For example, Alawadi et al. (2018) utilized a deep neural network to model complex relationships in the data, achieving notable improvements in prediction accuracy. These models can automatically learn feature representations from data, potentially capturing more complex patterns. However, they require large amounts of data and computational resources, which may limit their practical applicability in some contexts. Another study demonstrated the application of convolutional neural networks (CNNs) to incorporate image data of cars, further enhancing model performancen (Li et al., 2019).

Feature Engineering and Selection: A significant aspect of building accurate predictive models is the identification and selection of relevant features. Studies have shown that incorporating features such as car brand, model, engine size, and fuel type can significantly enhance model performance. For instance, Kou and Lou (2019) highlighted the importance of feature engineering in improving the accuracy of predictive models for used car prices. Additionally, techniques such as principal component analysis (PCA) and feature importance ranking have been used to reduce dimensionality and improve model efficiency.

3 Methodology

To predict the price of the used cars using the features we described in previous sections, we experimented with three different machine learning models: Linear Regression, Gradient Boosting, and Random Forest. Each model has its own strengths and weaknesses, and we will discuss them in this section and the following section.

For the rest of this section, we use y_i to denote the target variable, i.e., the price of a used car, $x_{i1},\ x_{i2}, \cdots, x_{ip}$ to denote p features that used to make the prediction. i denotes the i-th observation and let n denote the total number of observations.

3.1 Base Model - Linear Regression

We used a Linear Regression model as the base model because it is simple to implement and understand. It estimates the linear relationship between the features and the variables. The model tasks the form

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \epsilon_i, \quad i = 1, \cdots, n \tag{1}$$

where ϵ denotes random noise that is not picked up by x. The algorithm is designed to minimize the Mean Square Error (MSE) $= \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2$. The correctness of the model is usually evaluated using $R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y_i})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$.

Linear Regression model can be effective when its assumptions are met. As the name implies, the model assumes linear relationships between features and the target, which might not hold true for this dataset. It also requires low correlation between features. This could also be hard to achieve in this dataset. For example, newer cars tend to be in better condition and have automatic instead of manual transmissions. Other strong assumptions require the error term ϵ to be normally distributed and have constant variance. While those assumptions might not be met in this project, the Linear Regression model provides a baseline of how well our data can predict the target.

3.2 Gradient Boosting

Gradient Boosting is an ensemble model that builds upon multiple decision trees. Each decision tree is a "weak learner" trained in sequence to correct the errors of the previous tree (Friedman, 2001). Each time the model re-samples the data according to the error to focus on wrongly predicted samples.

Friedman (2001) shows iteration equations as

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x}; \mathbf{a_m}) \tag{2}$$

$$\tilde{y_i} = -\left[\frac{\partial L(y_i, F(\boldsymbol{x}_i))}{\partial F(\boldsymbol{x}_i)}\right]_{F(\boldsymbol{x})=F_{m-1}(\boldsymbol{x})} \tag{3}$$

$$\mathbf{a}_m = arg\min_{\mathbf{a},\beta} \sum_{i=1}^{N} [\tilde{y}_i - \beta h(\mathbf{x}_i; \mathbf{a})]^2 \tag{4}$$

where $i = 1, \cdots, N$ and $h(\mathbf{x}; \mathbf{a_m})$ is the "weak learner" at the m-th iteration, $F_{m-1}(\mathbf{x})$ and $F_m(\mathbf{x})$ denotes the ensemble model at $(m-1)$-th and m-th iterations. ρ_m is the learning rate. Each iteration, "weak learner" $h(\mathbf{x}; \mathbf{a_m})$ is trained on the residual $\tilde{\mathbf{y}} = \mathbf{y} - F_m(\mathbf{x})$ by updating equation (4). The residual $\tilde{\mathbf{y}}$ equals equation (3) when using Mean Square Error as the loss function.

Gradient Boosting provides good predictions on non-linear relationships and interactions between features because of the usages of decision trees and aggregation. It also provides the freedom of customization by tuning the hyperparameters, such as the learning rate, number of trees, and tree depth. In the meantime, this exposes the model to a higher risk of overfitting and makes the model longer to train and harder to interpret. Gradient Boosting and other tree models can also be useful in feature selection based on the feature importance, which is the fraction of samples that traverse a splitting node on feature K.

3.3 Random Forest

Random Forest is another tree-based ensemble model. Unlike Gradient Boosting, instead of building upon sequential decision trees, it relies on bagging and randomness of trees. Bagging is the aggregation of random small samples of the whole dataset. Each tree is trained on a different small sample and aggregated together to maximize the averaged prediction probability (Ho, 1995).

$$g_c(\mathbf{x}) = \frac{1}{t} \sum_{j=1}^{t} \hat{P}(c|v_j(\mathbf{x})) \tag{5}$$

where t is the total number of "weak learners" in the ensemble model, $v_j(\mathbf{x})$ is the terminal node before assigning it to class c.

Similar to the Gradient Boosting model, the Random Forest model works well on non-linear relationships and allows for customized tuning on hyperparameters. It is less likely to be overfitting compared with Gradient Boosting due to averaging. It is also more parallelizable and it does not depend on sequential learners. However, the Gradient Boosting model usually outperforms the Random Forest model because it is trained on the error of previous learners.

4 Modeling

We implemented our modelings in Python using the *Scikit-learn* package developed by Pedregosa et al. (2011). Based on the initial examination and analysis of the dataset, we conducted data cleaning and pre-processing to massage the dataset to fit the basic linear regression model. We described the process in Section 5.1. In this section, based on the baseline model result, we further modified the pre-processing methods feature engineer variables for all models. We fine-tuned the Gradient Boosting and Random Forest models using Grid Search and simplified our model based on the results while maintaining similar prediction power.

4.1 Feature Engineering

We tested out different feature engineering methods on the Linear Regression model. For features with the most missing values, i.e. *cylinders, condition* and *size*, we compared the model performance on 1) imputing missing categorical values; versus 2) creating a new category indicating the missing data.

Based on the Linear Regression result, we further feature engineered features to achieve better model performance: for *price*, we took the log form as it follows a log-normal distribution; for *year*, we rebased it to the year 1900 to make its value align in a similar range of other numeric values; in addition to creating the missing category, we also included *manufacture* and *paint_color* and created interaction between ordinal variables, i.e. *cylinders, condition* and *size* because they are all related to the quality of a vehicle.

4.2 Grid Search

Gradient Boosting and Random Forest have a variety of hyperparameters that could hugely impact the model performance. We focused our hyperparameter tuning on balancing between model accuracy and overfitting. Model accuracy is measured using R^2 to maintain consistency across different algorithms.

For Gradient Boosting, the learning rate and the number of estimators, i.e. "weak learners" can impact overfitting and training time. The maximum

Algorithm	Hyperparameters	Values
Gradient Boosting	Loss Function	['squared_loss', 'absolute_error', 'huber']
	Learning Rate	[0.001, 0.01, 0.1, 0.5]
	Max Depth	[7, 9, 11]
Random Forest	Number of Estimators	[100, 150, 200]
	Max Depth	[7, 9, 11]

Table 1: Grid Search Space on Hyperparameters

depth of each tree controls the complexity of the model and as a result impacts overfitting. The loss function impacts the optimization method and accuracy of the model. As another member in tree models, Random Forest has similar hyperparameters, but it does not have a learning rate because it does not use gradient descending optimization. We grid-searched the combination of hyperparameters as described in Table 1.

4.3 Feature Selection

Since we used the one-hot encoding to process categorical variables, it boosted the number of features in the model and made the model complicated. To keep the model simple, we performed feature selection using the permutation importance of features as well as using the *Recursive Feature Elimination (RFECV)* and the *SelectFromModel* methods in *Scikit-learn*. *RFECV* removes the least significant features iteratively by building a new model at each iteration, while *SelectFromModel* removes less important features based on a given threshold.

5 Evaluation

5.1 Experimental Set-up

We used the dataset from Craigslist, the world's largest collection of used vehicles for sale. The dataset contains over 500k records, however, we only sampled 10% of the dataset (around 50k observations) due to the limited computation resources.

To prepare the dataset, we removed 12 features that we think are not useful for price prediction for simplicity and to avoid overfitting. For example, the variable *county* has too many missing values which provide little information for price prediction, the variation in the variable *state* is more valuable than the variable *region*, and the variable *description* leaks target information because it contains the price in the text. After inspecting the features using domain knowledge and a baseline model, we identified the most important features such as the type and status of a car. Thus we removed unnecessary features and only kept 12 features: *year, manufacturer, condition, cylinders, fuel, odometer, transmission, drive, size, type, paint color, state*, and *title status*.

Besides, we conducted some traditional data-cleaning steps to ensure better data quality. First, we dropped the outliers for the target variable *price* and the variable *odometer* to avoid bias. Next, we looked at the missing values and imputed the mean values for continuous variables and the mode values for categorical variables. Then, we standardized some numerical variables such as *odometer* and one-hot encoded the categorical variables such as *title status* and *fuel* to help models better understand the patterns thus generating more robust results. Finally, we log-transformed the target variable *price* as it seems to have a log-normal distribution.

5.2 Results

We use the R^2 score to evaluate the model's performance since it determines the proportion of variance in the dependent variable that can be explained by the independent variables thus providing information about the goodness of fit of a regression model.

In general, the machine learning algorithms outperformed the random-selection baseline of 50%. The simple Linear Regression we trained on five-fold training data achieved an R^2 score of 56.9% on the test dataset.

However, our improvement using the linear regression was not so great since the model was simple and we didn't do a lot of tuning. Thus, we still wanted to investigate more ways to get better predictions.

First of all, we tested more complex models such as Random Forest and Gradient Boosting. These models do not have strict assumptions as linear regression does, thus they can capture the potential non-linear relationship between the independent

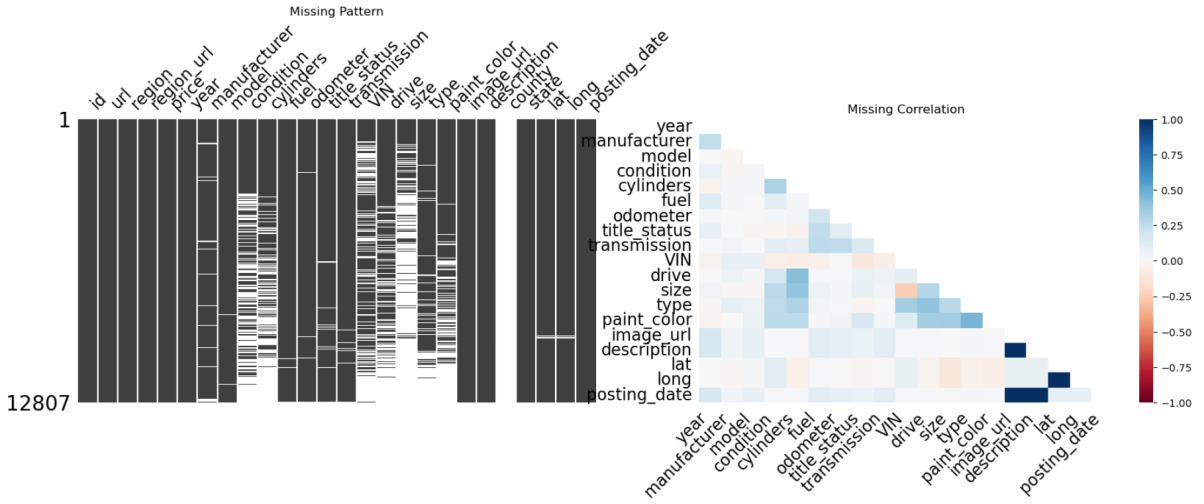

Figure 1: Missing Pattern

variables and dependent variables, as well as explain more variance. Besides, as mentioned above, we also used Grid Search to do hyper-parameter tuning, which helped us permutate, combine, and test different hyper-parameters so we could choose the combinations that have the best performance. As a result, the performance was largely improved. The Gradient-Boosting model with a learning rate of 0.1 and a maximum depth of 9 achieved an R^2 score of 79.1%. In addition, we re-evaluated features using these models, showing that removing unnecessary features improved the performance by 9%.

Given the insights we gained from the feature importance reported by the models, we also experimented with different models to find one that is good at predictions and easy to understand. Since tree-based models are usually more interpretable, we finally trained the Decision Tree model with selected features and the Grid Search process, which reported the decision-making process to help us better understand it.

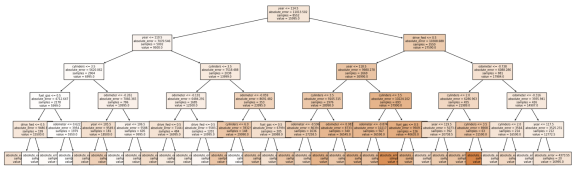

Figure 2: Decision Tree

6 Discussion

The results we get via machine learning methods outperform random guesses and save a lot of manual effort. The Linear Regression model tends to do the worst, and the Gradient Boosting model tends to do the best. In this experiment, we noticed that adjusting the model complexity can effectively impact the model performance, and using more complex models or increasing the model complexity can solve the under-fitting problem, thus comparing and tuning different models is crucial, during which grid search is an efficient technique to utilize. Other than that, feature selection and feature engineering contribute to a better result, we also need to use our domain knowledge and correlation check to filter out unnecessary features.

Machine learning methods have always been preferred for their efficiency but have also been criticized for the lack of transparency, however, it is important for us to understand why the model generates certain results and are suitable for certain use cases. In general, there is a trade-off between model complexity and interpretability. As the models become more complex and achieve higher accuracy, they are usually less interpretable thus making it hard to understand the decision-making process.

In our experiment, we focused on both the accuracy and the interpretability. We chose the Decision-Tree model to help us get a relatively better prediction compared to the simple models such as Linear Regression model, and it is easy to understand compared to other complex models such as the Gradient-Boosting model since it chooses a feature and a threshold to split the data at each step. However, the R^2 score indicated that there's still room for improvement, thus we should explore more ways to improve interpretability for more complex models. In the future, we should

Model	Hyperparameters	R-Sqaure
Linear Regression	-	0.57
Decision Tree	Maximum depth: 5	0.58
Random Forest	Number of Estimators :100; Maximum depth: 9	0.74
Gradient Boosting	Learning rate: 0.1; Loss function: huber; Maximum depth: 9	0.79

Table 2: Model Performance

try some techniques such as LIME (Local Interpretable Model-agnostic Explanations), and SHAP (SHapley Additive Explanations), the former one approximates any black box machine learning model with a local, interpretable model to explain each individual prediction so we can increase the transparency, and the latter one uses a game theoretic approach that measures each player's contribution to the final outcome, in machine learning, each feature is assigned an importance value representing its contribution to the model's output. (Nguyen et al., 2021). We believe that applying these methods to complex models can improve the performance while adding transparency, and we look forward to experimenting with these in the future.

References

Sadiq Ahmad, Ali Khan, and Wei Liu. 2017. Polynomial regression models for predicting used car prices. *International Journal of Data Science*, 5(3):56–63.

Ayman Alawadi, Salman Khan, and Irfan Ahmed. 2018. Deep neural networks for predicting used car prices. *Machine Learning Applications*, 9(1):67–75.

Li Chen and Yu Zhao. 2015. The influence of mileage and age on used car prices: A multiple linear regression analysis. *Automotive Data Journal*, 2(4):123–130.

Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

Tin Kam Ho. 1995. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE.

Jun Kou and Fei Lou. 2019. Feature engineering techniques for predicting used car prices. *Data Science in Automotive Industry*, 11(3):84–92.

Wei Li, Hong Zhang, and Lei Wang. 2019. Incorporating image data in predicting used car prices with cnns. *Journal of Artificial Intelligence Research*, 28(2):112–124.

Jannatul Nahar, Tamzid Imam, Kim S Tickle, and Yun-Peng Chen. 2015. Random forest regression for predicting used car prices. *Journal of Advanced Computing*, 1(2):45–52.

Hung Truong Thanh Nguyen, Hung Quoc Cao, Khang Vo Thanh Nguyen, and Nguyen Dinh Khoi Pham. 2021. Evaluation of explainable artificial intelligence: Shap, lime, and cam. In *Proceedings of the FPT AI Conference*, pages 1–6.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830.

Ying Zhang, Shuo Wang, and Chao Liu. 2017. Gradient boosting techniques for used car price prediction. *IEEE Transactions on Big Data*, 3(1):2–15.

A Appendix

A.1 GitHub

We have hosted our code on GitHub. More exploratory analysis, modeling setup, and results can be found in the notebook in the repository.