Pakistani Car Sales: A Machine Learning Approach

Alexander Holmes Aria Sajjad Lucas McPherron Tyler Dykes
Brennan Riddle Brooks Heath

December 14, 2023

Abstract

This research paper investigates the application of machine learning models to predict used car prices in the dynamic context of the Pakistani market, with a primary focus on everyday vehicles. The data preprocessing phase involved comprehensive steps to enhance the dataset's quality, addressing issues such as collinearity, high-level categorical features, and numeric feature normalization. The modeling section focuses on a diverse set of algorithms, including GLMNET, MLP, KNN, decision trees using CART, LightGBM, and XGBoost, each with its strengths and limitations. The models were all tuned by the use of a racing ANOVA methodology. LightGBM was our highest-performing model. The results section provides an in-depth analysis of each model's performance, offering insights into their strengths and limitations in predicting used car prices in the Pakistani market. This study contributes valuable knowledge to the intersection of machine learning and automotive commerce, offering a higher understanding of factors influencing pricing in the dynamic landscape of the Pakistani used car market.

1 Introduction

This paper presents an application of machine learning for the prediction of used car prices in Pakistan's used car market, with a specific focus on the more commonplace vehicles. While existing pricing methods are opaque, especially for everyday vehicles, our primary goal is to untangle the complexities surrounding price determinations. We want to improve the pricing of used cars by providing a framework that clarifies the process for buyers and sellers. We aim to address the unique challenges posed by the diversity of vehicles in the Pakistani used car market by employing machine learning and offering an innovative pricing approach that caters to the dynamics of the Pakistani used car market.

This study recognizes the imbalances in car pricing between buyers and sellers, especially regarding non-luxury vehicle appraisals. Our study attempts to deliver accurate and useful insights to help dealers and consumers navigate the intricate web of factors that affect car prices, such as make, model, mileage, and condition. We contribute to a conversation about leveraging machine learning to enhance transparency in the used car market by exploring the complexities of car pricing, specifically for common-place used cars. With our project, the goal was not to create a price predictor but an aid for buyers, or sellers, to utilize and leverage when negotiating the price of the vehicle they would like to purchase. This allows a buyer to feel peace knowing they paid a fair price for their "new to them" vehicle, while also allowing salesmen to know that they were not only fair but were able to get the most out of a fair range for the vehicle they sold.

This paper not only demonstrates the potential of machine learning to reduce pricing disparities, but it also envisions a future in Pakistan with a more informed and equitable pre-owned car market, benefiting all parties involved in the process of buying and selling everyday automobiles.

2 Methods

2.1 Data

The data used in this study was obtained from the Kaggle dataset 'Pakistan Used Car Prices 2023' (https://www.kaggle.com/datasets/talhabarkaatahmad/pakistan-used-car-prices-2023). The dataset contains 77,878 records and 16 columns, including variables like make, model, assembly, registered, year, mileage, and engine. The dataset was collected from the website PakWheels.com, a popular online marketplace for buying and selling cars in Pakistan. The data was scraped from the website using the Python library BeautifulSoup. The dataset was then cleaned and filtered to remove outliers and irrelevant data. The final dataset was then exported as a CSV file and uploaded to Kaggle for public use. The dataset contains a wide range of vehicles, including both luxury and non-luxury cars. Luxury cars were identified by the make and model variables, which were then filtered out to focus solely on everyday makes and models. The dataset contains a mix of categorical and numeric variables, with the price variable being the most important response variable.

2.2 Preprocessing

The data preprocessing involved several steps to enhance the quality and relevance of the 'car_sales' dataset to machine learning algorithms. The creation of a binary 'luxury' variable classifies specific makes as either luxury or non-luxury. These cars were ultimately filtered out to focus solely on every day makes and models and not high-end luxury brands. Categorical variables, such as 'engine,' underwent conversion to factors for improved model compatibility. The registered column was transformed into binary values, distinguishing between registered and unregistered vehicles. A new 'usd' column was introduced, representing the price in USD to facilitate easier comparison. All vehicles with a USD value above $80,000 were filtered out for our analysis. The rationale for this is due to some extreme outliers in the luxury car scene in Pakistan. The missing values in the 'assembly' column were handled by imputing any missing values with the value of local. This was a stipulation on the Kaggle data set. Additional filtering steps excluded records with specific identifiers and non-luxury cars, while also removing irrelevant columns such as Addref and City. Some vehicles were outliers within this market that would not normally be in the USA. This problem was addressed by the price filter. Lastly, any remaining incomplete rows were removed, resulting in a refined dataset ready for machine learning algorithms.

2.3 Testing Splits

The dataset before creating a training and testing split had an overall row count of 61,962. Once the data was split, the training set contained 46,471 records, and the testing set contained 15,491 records. Cross-validation was employed in this analysis, with five folds being utilized during the training of all models. An important aspect in the creation of the training and testing data was the use of stratification based on the usd of the vehicle to ensure that the training and testing data had similar distributions of usd values. This was done to ensure that the models were trained and tested with vehicles of similar USD values.

2.4 Preprocessing Recipe

A data preprocessing recipe was designed to prepare the training dataset for modeling the relationship between the response variable USD (used car price in USD) and all explanatory variables. The recipe, implemented using the 'recipes' package in R, consists of several key steps. Initially, it removes the price and luxury predictors to eliminate correlated and unneeded features. The next step encodes categorical variables into linear functions using Bayesian likelihood encodings. This step excluded nominal predictors that had a few levels: fuel, transmission, and assembly. One-hot encoding is then applied to all remaining nominal predictors, introducing binary dummy variables for categorical data. Zero-variance predictors are removed, and all predictors are normalized to ensure consistent scaling. Lastly, a correlation filter is applied, which

removes highly correlated numeric predictors, mitigating multicollinearity issues. Overall, this comprehensive data preprocessing enhances the suitability of the dataset for machine learning modeling.

2.5 Models

In the modeling phase of our study, we applied a diverse set of machine learning algorithms to predict used car prices. The machine learning algorithms utilized included GLMNET, MLP (Multilayer Perceptron), LightGBM, XGBoost, KNN (K-Nearest Neighbors), and decision trees utilizing the CART (Classification and Regression Trees) methodology. Each model brings unique strengths, addressing different aspects of the relationship between USD (used car price in USD) and the explanatory features. GLMNET, an elastic net regularized linear regression model, allows us to balance the trade-off between error and variance. MLP, a multi-layer perceptron, captures intricate nonlinear relationships within the explanatory variables. LightGBM and XGBoost, gradient-boosting models, excel at handling large datasets and complex interactions through the use of an ensemble of decision trees. KNN, a distance-based algorithm, provides insights into specific groupings of cars. Finally, decision trees using CART afford interpretability and uncover interactions between explanatory variables. By exploring each of these model performances, we will enhance our understanding of the factors influencing used car prices in Pakistan. See Appendix C for more information regarding model selection and decisions.

2.6 Model Tuning

The tuning process used was a racing methodology to optimize the time spent training each machine learning model with an expansive grid of possible hyper-parameters. All models were trained by the methodology. We were able to identify the most effective parameter configurations for each model, fine-tuning their abilities to capture the effect each explanatory variable had on USD.

The racing methodology randomly selects certain folds of training data to train all combinations of hyperparameters. After three iterations of training, ANOVA tables were fitted with underperforming hyperparameter models being dropped out of the training process. This process was repeated until the optimal hyperparameter configuration was identified. Once the optimal combination of hyperparameters was identified, the model was then fit to the final testing set of data.

3 Results

Each model, GLMNET, MLP, KNN, decision trees using CART, LightGBM, and XGBoost, underwent training and testing set evaluation. We selected LightGBM as the most parsimonious and best-performing model. We discuss the performance of each model in both the training and testing sets and provide calibration plots with multiple metrics to holistically understand the performance of each model type. We also provide a feature importance plot to understand how important each explanatory variable is in the LightGBM model and how it relates to USD.

3.1 GLMNET

Results of the GLMNET model were inherently mixed, as we knew the relationship between USD and the predictors was most likely nonlinear. The model was not able to capture some of the interactions within the data, as indicated by the high RMSE and low R-squared. The linear model's performance was also consistent across the training and testing sets, indicating that it was not overfitting the data. The model predictions between both training and testing are displayed in Figure 1. The model predictions were relatively inaccurate with a training RMSE of 3900.95 and standard error of 33.34 displayed in Table 1. GLMNET testing results were marginally worse with an RMSE of 3910.05 and an R-Squared of 0.84 displayed in Table 2.

Table 1: GLMNET Training Results

.metric	mean	std_err
rmse	3900.95	33
rsq	0.85	0

Table 2: GLMNET Test Results

.metric	.estimate
rmse	3910.05
rsq	0.84

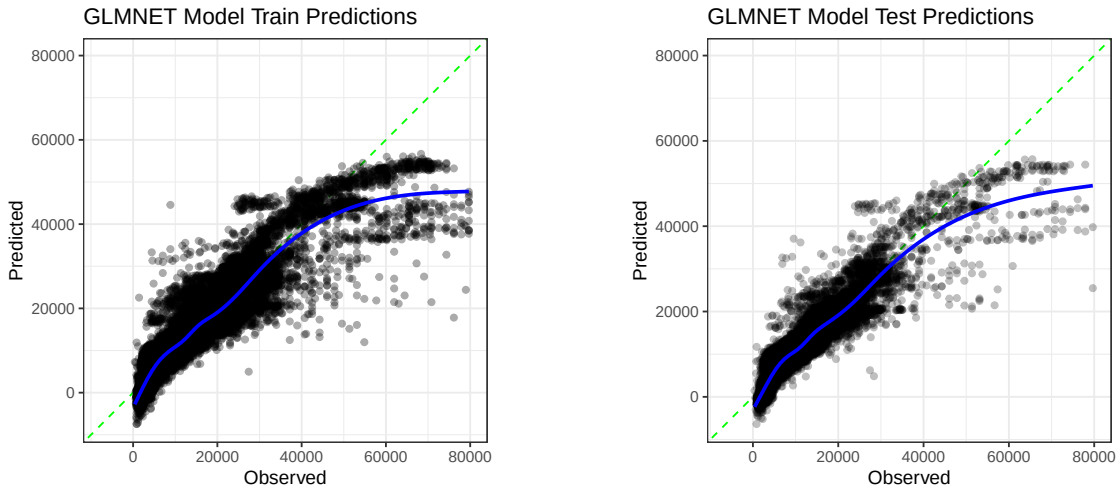

Figure 1: GLMNET Training and Testing Predictions

The testing results were also promising, with the model performing similarly to the training results. The RMSE was slightly lower and the R-squared value was slightly higher. This indicates that the model was slightly overfitting the data.

3.2 MLP

The results of the MLP trained with the BRLUEE engine were disappointing, as indicated by the high RMSE and low R-squared values. The model's performance was also consistent across the training and testing sets, indicating that it was not overfitting the data. The model predictions between both training and testing are displayed in Figure 2. The model predictions were relatively accurate with a training RMSE of 3690.64 and standard error of 66.07 displayed in Table 3. MLP testing results were marginally worse with an RMSE of 3731.25 and an R-Squared of 0.86 displayed in Table 4. The performance was better than the GLMNET model but we knew that performance could be improved by employing other models.

Table 3: MLP Training Results

.metric	mean	std_err
rmse	3690.64	66
rsq	0.86	0

Table 4: MLP Test Results

.metric	.estimate
rmse	3731.25
rsq	0.86

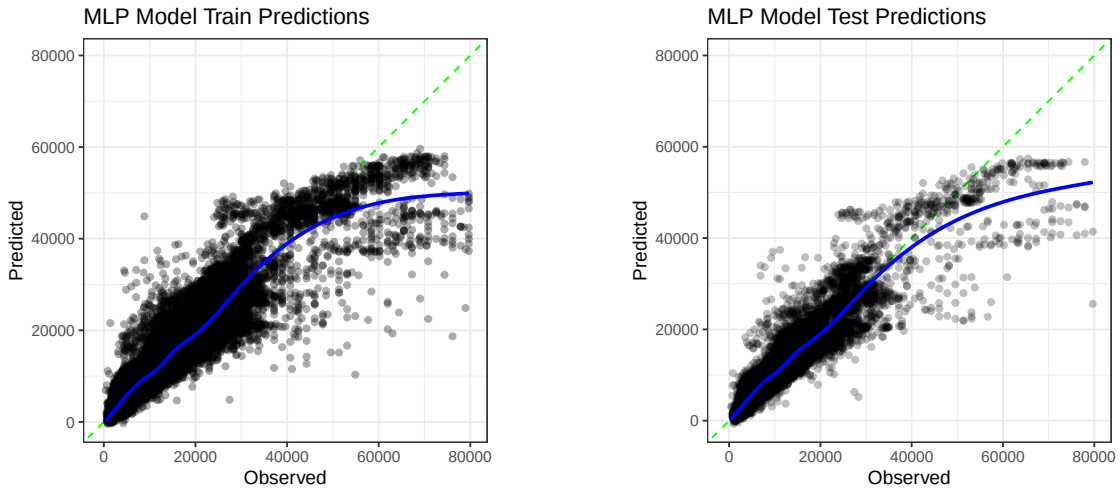

Figure 2: MLP Training and Testing Predictions

3.3 KNN

The results of the KNN model were much better than the previous two models due to low average RMSE and high R-squared values. The model's performance was also consistent across the training and testing sets, which was an indicator of the generalization of predictions. The model predictions between both training and testing are displayed in Figure 3. The model predictions were relatively accurate with a training RMSE of 2043.43 and standard error of 58.67 displayed in Table 5. KNN testing results were marginally better with an RMSE of 1946.31 and a R-Squared of 0.96 displayed in Table 6. The KNN model was a great leap forward in performance compared to the previous two models in Section 3.1 and Section 3.2.

Table 5: KNN Training Results

.metric	mean	std_err
rmse	2043.43	59
rsq	0.96	0

Table 6: KNN Test Results

.metric	.estimate
rmse	1946.31
rsq	0.96

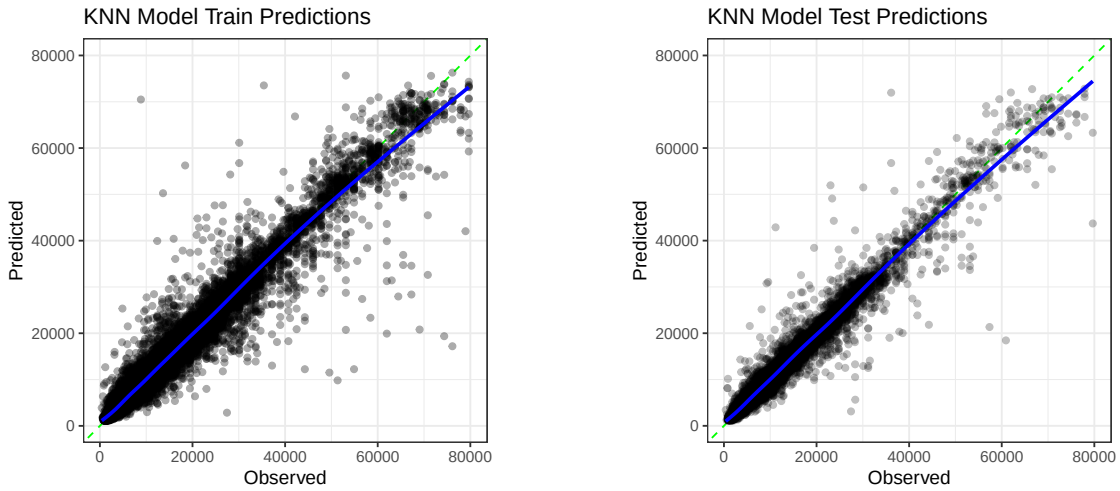

Figure 3: KNN Training and Testing Predictions

3.4 Decision Trees using CART

Results from the CART model were marginally worse than the KNN model described in Section 3.3. The model still had relatively low RMSE and high R-squared values compared to models in Sections 3.2 and 3.1. The model's performance was also consistent across the training and testing sets, indicating generalization of predictions. The model predictions between both training and testing are displayed in Figure 4. The model predictions were accurate with a training RMSE of 2058.47 and standard error of 41.96 displayed in Table 7. CART Decision Tree testing results were marginally better with an RMSE of 1945.49 and a R-Squared of 0.96 displayed in Table 8. Based on the performance of this model, we knew an ensemble of decision trees would provide even greater performance.

Table 7: Decision Tree Training Results

.metric	mean	std_err
rmse	2058.47	42
rsq	0.96	0

Table 8: Decision Tree Test Results

.metric	.estimate
rmse	1945.49
rsq	0.96

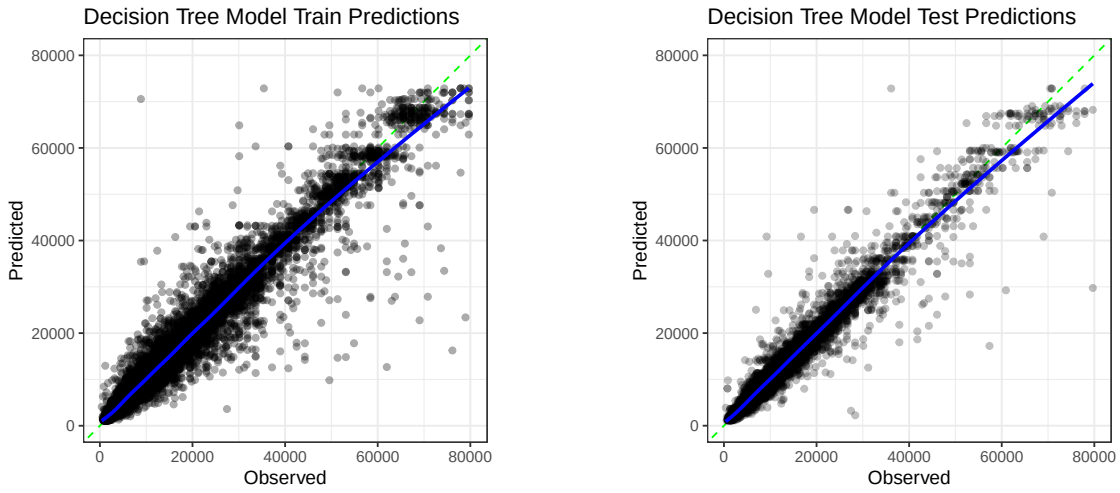

Figure 4: Decision Tree Training and Testing Predictions

3.5 LightGBM

LightGBM results were the best of all models tested, as evidenced by the extremely low RMSE and high R-squared values compared to models in Sections 3.3, 3.1, and @ref(decision trees using cart). The gradient-boosted model performance was also similar in both training and testing, which is indicative of good generalization of predictions. The model predictions between both training and testing are displayed in Figure 5. The model predictions were highly accurate with a training RMSE of 1734.89 and standard error of 47.07 displayed in Table 9. The standard error was the second-lowest of all trained models. LightGBM testing results were slightly better with an RMSE of 1636.53 and an R-Squared of 0.97 displayed in Table 10. This model was ultimately chosen as the best model out of all the models tested due to the use of the one standard deviation rule to select the best model. The RMSE was 1.1 standard deviations away from the performance of the XGBoost model which is discussed in the following subsection. Variables importance within the LightGBM model are displayed in Figure 6. Variables like model, year, and engine were the most important explanatory features when making a used car pricing prediction. Less important variables were transmission, luxury, and registration status.

Table 9: Light GBM Training Results

.metric	mean	std_err
rmse	1734.89	47
rsq	0.97	0

Table 10: Light GBM Test Results

.metric	.estimate
rmse	1636.53
rsq	0.97

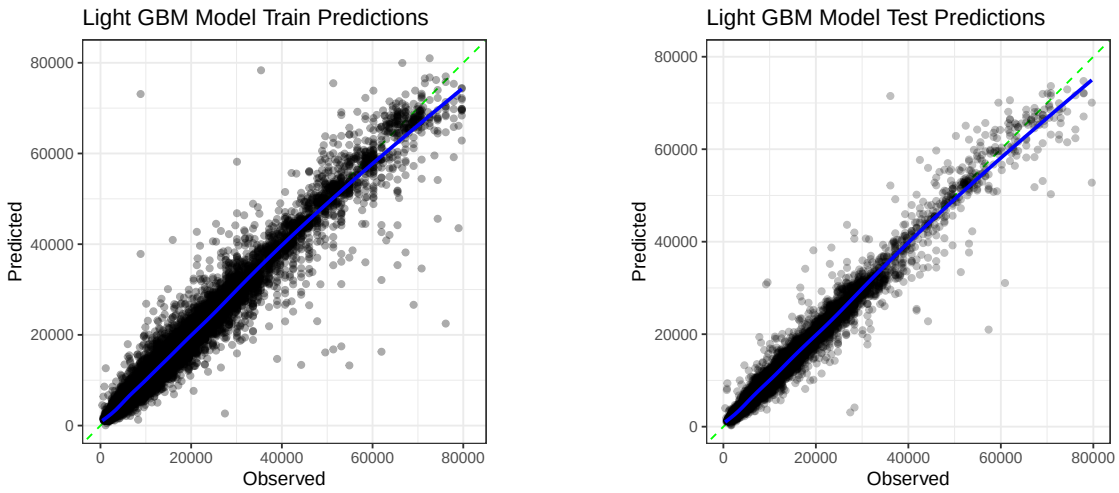

Figure 5: Light GBM Training and Testing Predictions

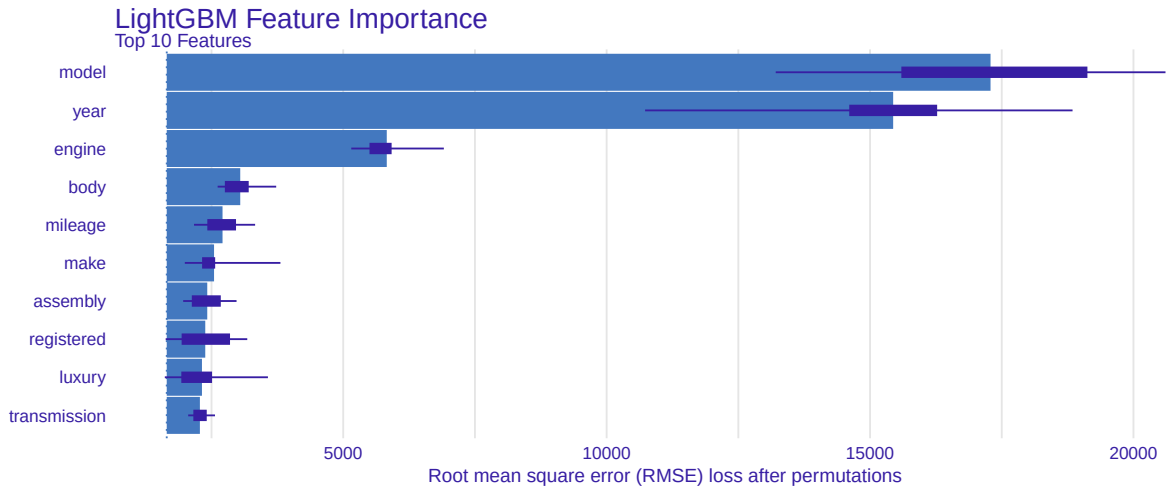

Figure 6: Variable Importance Plot for Light GBM Model

3.6 XGBoost

XGBoost was the second-best performing model in terms of RMSE and R-squared of all fitted models. The model's performance was also consistent across the training and testing sets which indicates a lack of overfitting of the data. The model predictions between both training and testing are displayed in Figure 7. The model predictions were extremely accurate with a training RMSE of 1789.6 and standard error of 49.6 displayed in Table 11. This model had the third lowest standard error of all fitted models. XGBoost testing results were similar, with an RMSE of 1698.47 and an R-Squared of 0.97 displayed in Table 12. In comparison with the previously discussed LightGBM model, the XGBoost model had a higher RMSE value, greater than one standard deviation away, indicating significant performance differences.

Table 11: XGBoost Training Results

.metric	mean	std_err
rmse	1789.60	50
rsq	0.97	0

Table 12: XGBoost Test Results

.metric	.estimate
rmse	1698.47
rsq	0.97

The XGBoost model still had some pretty big missed predictions, which are displayed in Figure 7. The model missed predictions were mostly for cars that were priced higher than 10,000 USD. The model was not able to predict these car accurately due to the sparse data of these types of cars in the testing set.

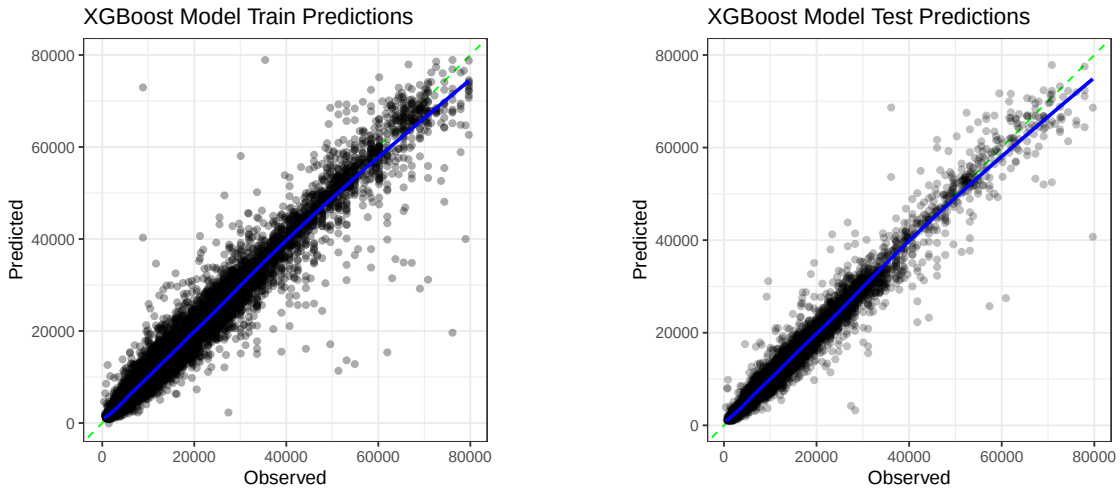

Figure 7: XGBoost Training and Testing Predictions

3.7 Model Recap

The performance of all models tested is displayed in Figure 8. The model performance was measured using RMSE, which is the square root of the average squared difference between the predicted and actual values. The model performance was also measured using R-squared, which is the proportion of the variance in the dependent variable that is explained by the explanatory variables. The model's performance was ultimately measured by the five-fold cross-validation process. The testing set was used to test the model for overfitting. The model performance was similar between the training and testing sets for most of the models tested, indicating that the models were generalizing the predictions across the data. The model performance was similar for gradient-boosted models, with KNN and decision tree coming in second. The worst performance was GLMNET and MLP models. The LightGBM model was ultimately chosen for its superior performance in pricing predictions.

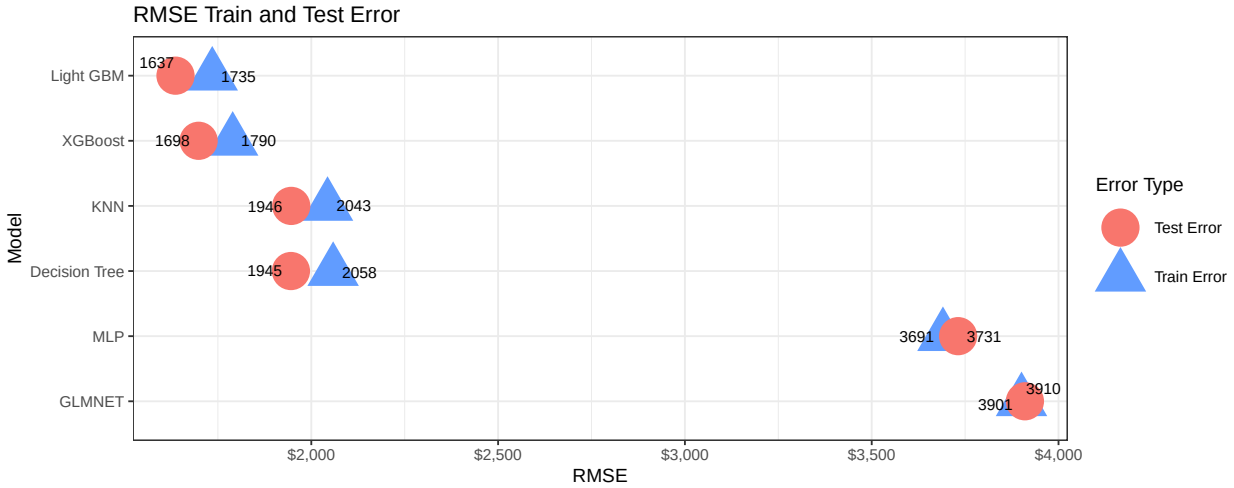

Figure 8: Model Performance

4 Discussion

In evaluating our machine learning models for predicting used car prices in the Pakistani market, we found LightGBM to be the most effective. It showed the best predictive performance, handling the dataset's complexity efficiently. XGBoost also performed well, ranking second in our tests, and offered a roughly comparable performance to LightGBM. However the performance metrics of LightGBM were slightly over one standard deviation better than that of XGBoost. These models, using boosting techniques, were effective in managing the complex relationships between various features and car prices.

Other models like GLMNET and MLP were less effective in this context. Their lower performance indicates difficulties in capturing the complex dynamics of used car pricing in Pakistan. However, understanding why GLMNET and MLP were less effective is useful for improving future models. Specifically, they struggled with the interaction and non-linearity of some features affecting car prices.

It's important to note that our models' predictive accuracy depends on the specific data used. The performance may vary with different datasets or when including features not in our study, such as accident history or recalls. These limitations are detailed in Appendix C.

Our research has practical implications for both car dealers and buyers in Pakistan. Dealers can use our models as a reference for pricing cars, but should be aware of the errors in each model. Despite efforts to minimize these errors, they still exist, with our best model, LightGBM, showing an average error of around $1,735 (testing error $1,636). For buyers, the models can help assess if a car's price is reasonable. Similar to tools like CarFax in the United States and Canada, which evaluate car pricing based on features, our models could be beneficial in markets without such tools, particularly Pakistan.

We aim for transparency in our model and advise users to exercise vigilance when using our models. Our models do not include data on prior accidents, repairs, maintenance history, or minor damages like interior wear, scratches, or rust. They also do not consider the color or type of interior. While the models are a useful tool, users should be cautious and consider these limitations when making real-world decisions.

It should also be noted that our models are tailored for common cars in Pakistan's market, not all cars. Initially, we used all 77,178 records in our dataset, leading to poor model performance. Improvements were made by adjusting features, including scaling, normalizing, and encoding both numeric and categorical data. Further enhancement came from excluding unusually expensive cars, such as Bentley models, certain Ford F-series trucks, and Toyota Land Cruisers. This still left us with 61,962 records. By limiting our analysis to cars priced under $80,000, which represent the majority of the market and are relevant to most buyers and sellers, our models' performance significantly improved. Therefore, our models are best suited for everyday cars within this price range.

For the sake of brevity, we excluded certain models like SVM and TensorFlow from this report. SVM was ineffective due to limitations in our data and the algorithm itself, failing to yield productive results. TensorFlow, while operational, resulted in extremely poor performance. Additionally, some of our feature engineering efforts had minimal impact. For instance, reducing the 372 unique color entries to 18 basic colors before encoding them as categorical variables unexpectedly worsened our models' performance slightly. Therefore, we decided to incorporate the original, more detailed color entries in the dataset.

Overall, our study provides insights into the effectiveness of various machine learning models for the Pakistani used car market. We identify the best-performing models and highlight the challenges in price prediction. Despite data and model limitations, our findings are useful for those looking to understand and navigate this market.

5 Appendix

5.1 Appendix A: Data Dictionary

Variable Name	Description	Type
assembly	The country where the car was assembled	Factor
body	The body type of the car	Factor
make	The car manufacturer	Factor
model	The car model	Factor
year	The year the car was manufactured	Factor
engine	The engine capacity of the car	Factor
transmission	The transmission type of the car	Factor
fuel	The fuel type of the car	Factor
color	The color of the car	Factor
registered	Whether the car is registered or not	Factor
mileage	The mileage of the car	Numeric
price	The price of the car in Pakistani Rupees	Numeric
luxury	Whether the car is a luxury car or not	Factor
usd	The price of the car in US Dollars	Numeric

5.2 Appendix B: Model Performance Metrics

Table 14: Model Performance Metrics

Model	Train RMSE	Train RMSE Std Error	Train Rsq	Train Rsq Std Error	Test RMSE
GLMNET	3901	33	0.85	0	3910
MLP	3691	66	0.86	0	3731
KNN	2043	59	0.96	0	1946
Decision Tree	2058	42	0.96	0	1945
Light GBM	1735	47	0.97	0	1637
XGBoost	1790	50	0.97	0	1698

5.3 Appendix C: Model Descriptions

5.3.1 Elastic Net Regression (GLMNET)

The GLMNET algorithm is used with specified tuning parameters, evaluated using a 5-fold cross-validation approach. The best model is chosen based on the one-standard-error rule for the mixture parameter and the minimum RMSE. Predictions from the optimally tuned GLMNET model are presented in a calibration plot.
Pros: Simple and interpretable, handles categorical variables effectively, and is suitable for linear relationships.
Cons: Assumes linear relationships between predictors and response, sensitive to outliers, and susceptible to multicollinearity.

5.3.2 LightGBM

LightGBM, set up with boost_tree, uses tuning parameters in a workflow. We use 5-fold cross-validation to evaluate performance, selecting the best model based on the lowest RMSE and highest R-squared. The predictions of the optimally tuned LightGBM model are displayed in a calibration plot.
Pros: Efficient gradient boosting with high performance, suitable for large datasets, and effectively handles categorical features.
Cons: Sensitive to outliers and multicollinearity.

5.3.3 XGBoost

XGBoost is set up with tuning parameters. We use Latin Hypercube sampling to test different hyperparameter combinations. The best configuration is identified through a racing process, and the final model, optimized with these parameters, is assessed using the testing set.
Pros: Effective with missing data, offers regularization to prevent overfitting, and allows feature importance analysis.
Cons: Needs precise hyperparameter tuning, computationally demanding, less interpretable due to its black-box nature, sensitive to outliers, and can overfit with complex models.

5.3.4 K-Nearest Neighbors (KNN)

The KNN model is set up with specific tuning parameters. We select the best model based on the lowest RMSE, identified through a racing process. The top-tuned KNN model's predictions are displayed in a calibration plot. The final model, optimized with the best hyper-parameters, is tested on the testing dataset.
Pros: It doesn't assume a specific data distribution, handles noisy data well, and is straightforward.
Cons: Requires significant computation for large datasets, sensitive to irrelevant features, and dependent on the distance metric used. Features must also be normalized.

5.3.5 Decision Tree

The Decision Tree model is tuned with specific parameters and evaluated through a racing process to select the model with the lowest RMSE. This final model, optimized with the best hyperparameters, is tested on the holdout dataset.
Pros: Effectively captures non-linear relationships, provides clear decision rules, and is robust to outliers.
Cons: Can overfit easily, produces non-smooth decision boundaries, and is sensitive to minor data changes.

5.3.6 Multi-Layer-Perceptron (MLP)

The MLP model, a neural network, is set up with tuning parameters and the best model is chosen based on the lowest RMSE, using a racing process. We visualize predictions from the best-tuned model in a calibration plot. The final model is assessed on a test dataset, with a scatter plot comparing predicted and actual values.
Pros: Identifies complex non-linear patterns, effective for image and text data, and learns hierarchical features.
Cons: Needs a lot of data, can overfit easily, demands high computational power, and is hard to interpret due to its black-box nature.