Used Car Price Prediction

Team Elite Squad
Madhav Gupta (2022EE11737), Arunim Garg (2022EE32002), Vatsal Jain (2022EE11672)
Department of Electrical Engineering, IIT Delhi

November 24, 2024

Abstract

Accurately predicting the price of used cars is a critical problem in the automotive marketplace. This report addresses this challenge using machine learning techniques applied to a dataset of 50,000 car listings. Various preprocessing steps, feature engineering techniques, and model architectures were evaluated to enhance prediction accuracy. The best models achieved promising results, demonstrating the potential of data-driven approaches in aiding price estimation for used cars.

1 Problem Statement

The used car market is highly dynamic, with prices influenced by factors such as mileage, fuel type, accident history, and more. Buyers and sellers often face challenges in determining fair market value. This study aims to build a predictive model capable of estimating car prices based on these attributes, providing an objective tool for decision-making.

2 Proposed Solution

To tackle this problem, a machine learning pipeline was developed comprising:

1. **Dataset Preparation**: Handling missing values, encoding categorical variables, and scaling numerical features.
2. **Feature Engineering**: Transforming features such as mileage and accident history for improved model interpretability.
3. **Model Training and Selection**: Evaluating algorithms including Linear Regression, Random Forest, XGBoost, and Neural Networks.
4. **Model Evaluation**: Using Root Mean Squared Error (RMSE) as the evaluation metric for performance comparison.

3 Experiments Conducted

3.1 Dataset Description

The dataset contains several entries with features such as brand, model year, mileage, fuel type, engine specifications, transmission, exterior and interior colors, accident history, clean title status, and price.

4 Data Preprocessing

Data preprocessing is a crucial step in preparing the dataset for analysis and modeling. Below are the detailed steps taken to clean, transform, and encode the data.

4.1 Introduction to the Dataset

The dataset utilized in this project provides a comprehensive overview of various used cars, offering insights into the factors influencing their pricing. It consists of several records, each detailing the features of a specific car. Below is a summary of the key attributes:

- **Brand**: Manufacturer of the vehicle (e.g., Ford, BMW, Jaguar).
- **Model**: Specific model of the car (e.g., F-150 Lariat, XF Luxury).
- **Model Year**: Year of manufacture, indicating the car's age.
- **Mileage**: Distance traveled by the car (in miles).
- **Fuel Type**: Type of fuel used (e.g., Gasoline, Hybrid).
- **Engine**: Specifications of the car's engine, including horsepower and cylinder configuration.
- **Transmission**: Type of gearbox (e.g., Automatic, Manual, Dual Shift).
- **Exterior Color**: Color of the car's exterior.
- **Interior Color**: Color of the car's interior.
- **Accident History**: Reports of any accidents involving the car.

- **Clean Title**: Indicates whether the car has a clean ownership title.
- **Price**: Selling price of the car (target variable for prediction).

4.2 Data Cleaning

- **Handling Missing Values:** Missing values represented by `None`, `-`, and `-1` were replaced with `NaN`. Numeric columns were filled with the mean, and categorical columns were filled with the mode.
- **Duplicate Removal:** Duplicate rows were identified and removed, reducing the dataset size.
- **Outlier Removal:** Rows with *model year* earlier than 1990 were removed as they were considered outliers.
- **Unnecessary Columns:** Columns such as *id* and *clean_title*, which were irrelevant or constant, were dropped.

4.3 Feature Engineering

- **Engine Details:** The *engine* column was parsed to extract specifications such as:
  - *Horsepower (HP):* Power output in horsepower.
  - *Displacement (L):* Engine size in liters.
  - *Cylinders:* Configuration such as *I4*, *V6*, or numeric values like *6*.
  - Binary features indicating the presence of *DOHC*, *Turbo*, and *Twin Turbo*.

  This extraction process leveraged regular expressions to parse the *engine* column and created new columns for each feature, significantly enriching the dataset with granular engine-related details.
- **Age Calculation:** The age of the car was calculated by subtracting the *model year* from the current year. The *model year* column was replaced with the calculated *age*.

4.4 Encoding Categorical Variables

- **Target Encoding:** Categories in *brand*, *model*, and *transmission* were encoded based on their mean price.
- **Label Encoding:** The *accident* column was encoded as follows:
  - `At least 1 accident or damage reported` → 1.
  - `None reported` → 0.
- **One-Hot Encoding:** Binary columns were created for unique categories in the *fuel type* column.

4.5 Final Data Preparation and Transformations

1. Final List of Features

After preprocessing and feature engineering, the final DataFrame includes the following features:

1. Core features: **brand**, **model**, **age**, **mileage**, **transmission**, **ext_col**, **int_col**, **accident**, **price**.
2. Engine-related features: **HP**, **Litre**, **Cylinders**, **DOHC**, **Turbo**, **Twin Turbo**.
3. One-hot encoded fuel types: **fuel_type_E85 Flex Fuel**, **fuel_type_Gasoline**, **fuel_type_Hybrid**, **fuel_type_Plug-In Hybrid**, **fuel_type_not supported**.

2. Logarithmic Transformation of the Target Variable

To manage skewness and enhance model performance, the **price** column was transformed using a base-10 logarithm. This transformation helps reduce the impact of outliers and ensures better model fitting, especially when the data spans several orders of magnitude.

3. Scaling Numerical Features

Numerical features with more than two unique values were standardized using **StandardScaler**, which adjusts each feature to have:

- Mean = 0.
- Standard deviation = 1.

This scaling is critical for models sensitive to feature magnitudes (e.g., SVMs, Neural Networks). Table 1 shows a sample of the scaled DataFrame.

Table 1: Sample of Scaled DataFrame After Standardization

Brand	**Age (scaled)**	**Price (log)**
BMW	-0.52	4.56
Ford	0.87	3.78
Tesla	-1.25	4.98

4. Removing Low-Correlation Features

Columns with weak correlations to the target variable (**price**) were identified and removed. The following procedure was used:

1. Compute the correlation matrix.
2. Retain only features with absolute correlation above a threshold of 0.008.
3. Drop features with low correlation values.

This step ensures only features contributing meaningfully to the prediction remain.

The final dataset, streamlined after these transformations, forms the basis for subsequent model training and evaluation.

4.6 Model Training

- Linear Regression was implemented as a baseline model.
- Ensemble models like Random Forest and XGBoost were tuned for hyperparameter optimization.
- A feedforward Neural Network was constructed with three dense layers, ReLU activation, and dropout for regularization.

4.7 Model Training: Stacking Model

By comparing different configurations, we observed that the best results were achieved using a stacking model comprising the following components:

Purpose

The stacking model integrates deep learning, ensemble methods, and regularization techniques to balance the strengths of flexibility, robustness, and simplicity. Its primary goal is to achieve accurate predictions for used car prices.

Stacking Model Workflow

1. **Train Base Models:** Train the Neural Network, RF, and ElasticNet independently using the training data.
2. **Generate Predictions:** Obtain predictions from each base model on the training and validation datasets.
3. **Train Meta Model:** Use the predictions from the base models as features to train the Linear Regression meta-model. The meta-model learns to weight the base models' predictions optimally.
4. **Final Prediction:** For unseen test data, predictions from the base models are passed to the meta-model for the final output.

4.8 Model Performance Summary

This section provides an overview of the evaluation results for different models, highlighting their strengths and limitations based on the metrics of Root Mean Squared Error (RMSE) and R-squared (R^2).

Stacking Model Performance Summary

Key Observations: Among the evaluated stacking models, the combination **Neural Network + Random Forest + ElasticNet** (Stacking Model 4) achieved the best performance, with the lowest RMSE of **48,787.41**. However, its R-squared value (**0.6744**) was slightly lower compared to other stacking combinations. Other models demonstrated competitive performance, but none surpassed the RMSE of Stacking Model 4.

- **Best RMSE:** 48,787.41 (Neural Network + RF + ElasticNet)
- **Best R-squared:** 0.6801 (LightGBM + Extra Trees + Linear)
- Stacking models leveraging a mix of boosting, linear, and ensemble methods show promise, with trade-offs between RMSE and R-squared.

Note: Models combined using "+" in the table represent stacking models evaluated.

Individual Model Performance Summary

Key Takeaways:

- **Best Individual Performance:** Gradient Boosting achieved the highest R-squared (**0.6594**) and a relatively low RMSE (**48,068.26**), making it the top-performing individual model.
- **Worst Performance:** Lasso Regression significantly underperformed, with an RMSE of **55,333.60** and a negative R-squared, indicating extreme underfitting due to excessive regularization.
- **Ensemble Strengths:** Models like Random Forest, Gradient Boosting, and XGBoost consistently outperformed simpler models, showing the advantage of ensemble methods in this dataset.
- **Neural Networks:** While not the best performer, the Neural Network exhibited competitive performance (**RMSE: 48,521.12, R-squared: 0.6526**) and complements ensemble methods effectively in stacking.

Conclusion: Ensemble-based methods and stacking combinations that blend different types of models (e.g., Neural Network, ElasticNet, Random Forest) balance flexibility and predictive accuracy effectively. This approach is particularly valuable for improving RMSE while retaining interpretability in regression tasks.

Stacking Model with All Learners: Performance Summary

This stacking model leverages the predictions of a diverse set of base learners, encompassing linear models (Linear Regression, Ridge, Lasso, ElasticNet), ensemble techniques (Random Forest, Gradient Boosting, XGBoost), and others (K-Nearest Neighbors, Support Vector Regressor). The predictions from these models are combined and used as input features for a meta-model, which in this case is a Linear Regression model.

Workflow Overview:

- **Base Models:** The stacking framework includes nine base learners, ensuring a diverse mix of simple, interpretable models and complex, non-linear learners.
- **Meta-Model:** Linear Regression serves as the meta-model, tasked with learning optimal weights to combine predictions from the base learners.
- **Process:**
    1. Out-of-fold predictions from base learners are generated using **K-Fold Cross-validation**.
    2. These predictions form a new feature set, which is used to train the meta-model.
    3. The final predictions are obtained by applying the meta-model to the aggregated outputs of the base learners.

Results: The stacking model achieved the following metrics:

- **Root Mean Squared Error (RMSE): 48,054.24**, indicating a low average error in predictions.
- **R-squared (R^2): 0.6642**, showing that approximately 66.42% of the variance in the target variable is explained by the model.

Note: The combinations of models joined by "+" indicate those used in stacking. For instance:

- **NN + RF + ElasticNet**: Neural Network, Random Forest, and ElasticNet.
- **RF + GB + Ridge**: Random Forest, Gradient Boosting, and Ridge Regression.
- **CatBoost + XGB + Lasso**: CatBoost, XGBoost, and Lasso Regression.
- **LightGBM + Extra Trees + Linear**: LightGBM, Extra Trees, and Linear Regression.
- **XGB + CatBoost + Linear**: XGBoost, CatBoost, and Linear Regression (Voting Model).

Table 2: Model Performance Comparison

Model	**RMSE**	**R^2**
Linear Regression	47,934.80	0.6282
Ridge Regression	47,935.37	0.6281
Lasso Regression	55,333.60	-0.0000
ElasticNet Regression	54,363.55	0.1811
Random Forest	48,467.45	0.6346
Gradient Boosting	48,068.26	0.6594
K-Nearest Neighbors	49,066.62	0.5922
Support Vector Regressor	48,969.46	0.6532
Neural Network	48,521.12	0.6526
XGBoost	48,275.64	0.6506
NN + RF + ElasticNet	**48,787.41**	**0.6744**
RF + GB + Ridge	51,018.34	0.6794
CatBoost + XGB + Lasso	51,276.95	0.6787
LightGBM + Extra Trees + Linear	50,771.66	0.6801
XGB + CatBoost + Linear	49,378.54	0.6783

5 Conclusion

This study successfully applied machine learning techniques to predict used car prices, achieving notable results through rigorous preprocessing, feature engineering, and model evaluation. You can find the analysis in the **GitHub Repository containing analysis**.

5.1 Key Findings

- Ensemble methods such as Gradient Boosting and XGBoost outperformed simpler models like Ridge and ElasticNet, achieving higher R-squared values and lower RMSE.
- Stacking models were particularly effective, with the combination of **Neural Network, Random Forest, and ElasticNet** achieving the lowest RMSE of **48,787.41**, and **LightGBM + Extra Trees + Linear Regression** yielding the highest R-squared of **0.6801**.
- Logarithmic transformation of the target variable and feature extraction significantly improved model performance, while removing low-correlation features enhanced prediction accuracy.

5.2 References

- Taeef Najib. (2022). Used Car Price Prediction Dataset. `https://www.kaggle.com/datasets/taeefnajib/used-car-price-prediction-dataset`.
- Li, Y., Li, Y., and Liu, Y. (2022). Research on used car price prediction based on random forest and LightGBM. *2022 IEEE 2nd International Conference on Data Science and Computer Application (ICDSCA)*, Dalian, China, 539-543. 10.1109/ICDSCA56264.2022.9988116.