SNS Brownbook - Car Price Predictions

Nyasha Gwaza, Sufi Hossain, Saumili Chakravarty

ABSTRACT

Buying or selling a used car can be a frustrating experience, especially with the risk of scams and dishonest pricing from dealerships or individual sellers. This project aims to ease that process by building a machine learning model that can estimate the fair value of a used car based on factors like the make and model, mileage, damage history, parts value, and overall roadworthiness. With this tool, sellers can better decide whether to sell the car as-is, invest in repairs first, or sell it for parts. Buyers, on the other hand, get a realistic idea of the car's worth and any likely repair costs—helping both sides make smarter, more informed decisions.

INTRODUCTION

The main goal of this project was to build a model that can accurately predict the price of a used car based on its various features. This kind of problem is a good fit for supervised learning, especially regression models, since we're predicting a continuous value (price). It's also a real-world scenario that people care about—buyers want to know if a car is fairly priced, and sellers want to make informed decisions.

Previous works on car price prediction often start with basic linear regression models due to their simplicity and interpretability. Linear regression is useful for understanding general trends, like how price might increase with car age or decrease with mileage. However, because car pricing depends on many factors that can interact in complex ways—like brand, model, fuel type, or transmission—more advanced models like Random Forest are often used for better accuracy.

We took a similar approach by testing both linear regression and Random Forest to see how well each one could handle our dataset. We found that while linear regression gave us a reasonable baseline, Random Forest performed better at capturing the nonlinear relationships between features and price. Our work also ties into what many existing studies highlight: the importance of preprocessing and feature engineering. Before training the models, we cleaned the data and created new features like car age and mileage bands to help the model make better predictions. We also limited outlier values to make sure the model wasn't skewed by unusually high prices or extreme mileage.

This project builds on those common practices, combining a clear goal—predicting car prices—with tested methods like regression and tree-based models, all supported by thoughtful data preparation.

METHOD

For this project, we scraped data from AutoTempest [1] using the Instant Data Scraper Google Chrome extension, which resulted in a dataset containing 1,138 car listings. The data we collected included key details about each vehicle, such as the car's brand, model year, mileage, transmission type, fuel type, and price. Once the data was in hand, we began by loading it into a Pandas DataFrame for easy manipulation and analysis.

The first step was to clean the data. We noticed that some important fields, like trim and mileage, had missing values, which could affect the accuracy of our predictions. To address this, we filled in missing values where possible. For instance, missing mileage values were replaced with the mean mileage of cars within the same model year. Similarly, we made sure that any cars with missing trim information were handled appropriately. Next, we removed unnecessary columns that wouldn't help in predicting the price, such as the listing URL and some other data related to the vehicle. These didn't provide any useful information for our models, so it was important to clean up the dataset and keep only the features relevant to the task.

We also did some feature engineering to make the dataset more useful for our models. One notable change was splitting the city/highway MPG combined column into two separate columns, one for city MPG and one for highway MPG. This division provided more granularity and allowed the models to better understand how these two factors could influence a car's price.

Another key step was to handle the geographic data—specifically, the location of the car. The dataset had the car's location as a combined city/state field, so we extracted just the state information. This state was treated as a categorical feature, as it could influence car prices (e.g., prices might differ based on location). We then applied one-hot encoding to this state column, as well as to other categorical features like the brand and drivetrain type. This transformed those variables into numerical data, making them usable by the machine learning models.

For the numerical features, such as mileage and year, we used a StandardScaler to standardize the values. Standardizing these features ensures that no variable dominates the others because of differences in scale. We also used a ColumnTransformer to apply the correct transformations to different types of features—one-hot encoding for categorical variables and scaling for numerical variables.

Once the data was preprocessed, we split it into training and testing sets, with 60% of the data used for training, 20% for validation and 20% for testing. This split helps ensure that we have an unbiased evaluation of how well our models generalize to unseen data. We then built two machine learning models to predict car prices: a Linear Regression model and a Random Forest Regressor. Both models were implemented in scikit-learn Pipelines, which allowed us to streamline the process by ensuring that the same preprocessing steps were applied to both the training and

test data consistently. This also ensured that our models were trained on correctly formatted data.

After training the models, we evaluated their performance using two key metrics: Root Mean Squared Error (RMSE) and the R^2 score. RMSE gives us an idea of the average error in the predictions, while the R^2 score helps us understand how well the model is fitting the data. These two metrics together provided a good indication of the accuracy of our predictions and helped us compare the performance of the two models.

EXPERIMENT

After setting up our models, we moved on to testing and comparing how well Linear Regression and Random Forest could predict car prices. Right off the bat, Random Forest performed a bit better than Linear Regression. But to push the model further, we tuned some of the key parameters like the number of estimators, max depth, and how the data was split within the trees. After running a grid search, the best combination turned out to be 300 estimators, a max depth of 30, with specific values for how the data splits and what features are considered at each split.

The original Random Forest model gave us an RMSE of about $13,593.79 and an R^2 score of 0.589. After tuning, the RMSE slightly increased to $13,685.90 and the R^2 dropped just a bit to 0.584—so not a massive difference. It seems the model was already well-optimized to begin with, and tuning gave us only a marginal improvement.

Linear Regression, while simpler, held up decently with an RMSE of $13,948.88 and an R^2 of 0.568. It's not far off, but it clearly didn't capture the non-linear patterns in the data as well as Random Forest did, especially when it came to edge cases or outliers.

To visualize how the models were performing, we plotted two graphs. Figure 1 shows actual vs. predicted prices for the Linear Regression model, and Figure 2 does the same for the Random Forest model. Both show a clear upward trend, which is what you'd hope for—predicted prices generally increase as actual prices increase. But you can tell from the scatter of the points that the predictions from Random Forest were more closely aligned to the ideal red line (where predictions perfectly match actual values). The dots are more tightly packed along that line in Figure 2, whereas Figure 1 shows a bit more spread, especially for higher-priced cars.

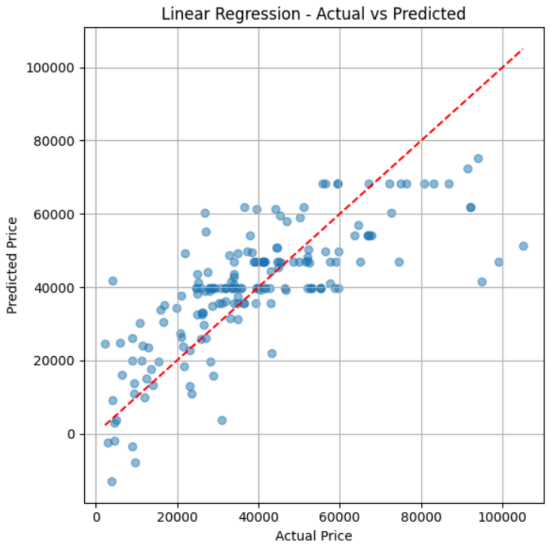

Figure 1

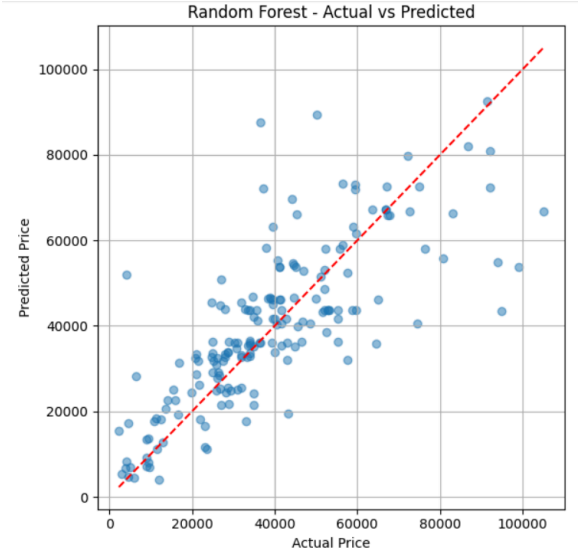

Figure 2

We also created a feature importance plot (Figure 3) to get a better sense of which features the Random Forest model leaned on the most. Unsurprisingly, mileage came out as the top predictor—cars with higher mileage typically go for less, so it makes sense the model picked up on that trend. Car age was the second most important feature, reinforcing the idea that newer cars generally fetch higher prices. Other influential factors included brand and drivetrain, but they had noticeably less weight compared to mileage and age. Overall, the model seemed to capture key real-world pricing patterns pretty well, which added confidence to the predictions we were seeing.

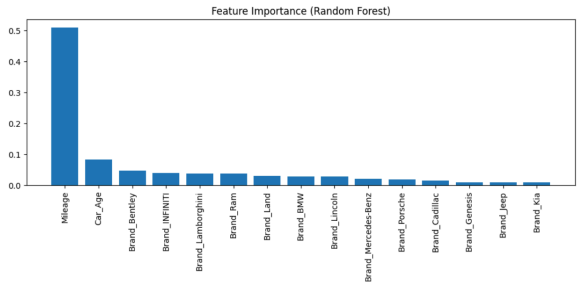

Figure 3

All in all, Random Forest came out on top. While both models performed respectably, Random Forest was clearly better at handling the complexity and subtle interactions in the dataset. The visualizations helped confirm that its predictions were not just statistically better, but also more consistent across the board.

CONCLUSION

Our work demonstrates that a well-tuned Random Forest model, supported by solid feature engineering, can deliver strong performance in predicting used car prices. Compared to Linear Regression, the Random Forest was slightly more accurate and proved more capable of capturing complex patterns and interactions between variables.

Through our analysis, we found that mileage was the most influential factor in determining price, followed closely by the car's age. These results make sense given how vehicles typically depreciate over time and usage. The model's predictions followed the overall trend of actual prices, and visualizations showed good alignment between predicted and actual values, reinforcing that the model was learning meaningful patterns.

Another improvement could be exploring simpler methods, like decision trees or k-nearest neighbors, to see if they offer competitive results with fewer computational resources. Additionally, incorporating regularization techniques like Ridge or Lasso could help avoid overfitting and improve generalization. Finally, incorporating external data sources, such as market trends or regional pricing data, might enhance the model's accuracy in a real-world setting.

Reference

[1] *AutoTempest.com: All the cars. One search.* (n.d.). AutoTempest. https://www.autotempest.com/

[2] Scikit-learn – Machine learning library used for model implementation (Linear Regression, Random Forest, MLPRegressor), pipelines, preprocessing, and metrics.
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830. https://scikit-learn.org/

[3] Pandas – Data manipulation library used to clean and preprocess the car dataset.
McKinney, W. (2010). Data Structures for Statistical Computing in Python. Proceedings of the 9th Python in Science Conference.

[4] Matplotlib – Used for generating all visualizations (loss curves, actual vs predicted scatter plots, feature importance).
Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering, 9(3), 90–95.

[5] Google Chrome Instant Data Scraper – Browser extension used to collect raw car data from AutoTempest.
Instant Data Scraper. Chrome Web Store. https://chrome.google.com/webstore/detail/instant-data-scraper

[6] StandardScaler & ColumnTransformer – Preprocessing methods from scikit-learn used for feature scaling and pipeline abstraction.
See scikit-learn documentation: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

[7] Random Forest & Feature Importance – Ensemble method used for non-linear regression, and analysis of key predictors.
Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5–32.

[8] Multiple Epoch Training in MLPRegressor – Method used to simulate deep learning-like iterative training using warm start and manually looping epochs.
scikit-learn MLP documentation: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html

[9] Course Material: ECE 408: The Art of Machine Learning
Zhiyao Duan, University of Rochester. Topics include supervised learning, regression, regularization, ensemble models, and neural networks.
https://hajim.rochester.edu/ece/sites/zduan/teaching/ece408/calendar.html