Car Price Predictor - Cowlar

Muhammad Abdullah Cheema

May 1, 2024

NOTE: Some figures are not positioned correctly (they appear later in the document) due to the size constraints.

1 Data Exploration

a) Distribution of Target Feature: The distribution of the target feature (Price) was initially skewed with a long tail, indicating a non-normal distribution. To address this, the Price column was log-transformed to approximate a normal distribution. This transformation is beneficial as many machine learning models assume a normal distribution of the target variable, leading to better model performance.

b) Distribution of Other Features: The exploration revealed that almost all features were skewed or imbalanced, potentially affecting the model's performance. Note: Kindly refer to notebook to visualize all the related plots.

c) Correlation Matrix: A correlation matrix was constructed to examine the relationships between numerical features and the target feature (Price). Minor multicollinearity was observed, particularly between the mileage and Make_Year columns, which could influence model predictions.

2 Feature Engineering

a) Age Column Creation: An 'Age' column was generated from the 'Make_Year' feature to better visualize its relationship with the target feature.

b) Addressing Multicollinearity: To mitigate multicollinearity between 'Make_Year' and 'Mileage', a new feature was engineered using a weighted sum formula. This feature captured information from both variables, considering the higher correlation of 'Age' with 'Price' compared to 'Mileage'. The following formula was used (inferred from the correlation matrix); Usage Index = 0.8*Age + 0.2*Mileage

   Figure 3 shows updated correlation matrix.

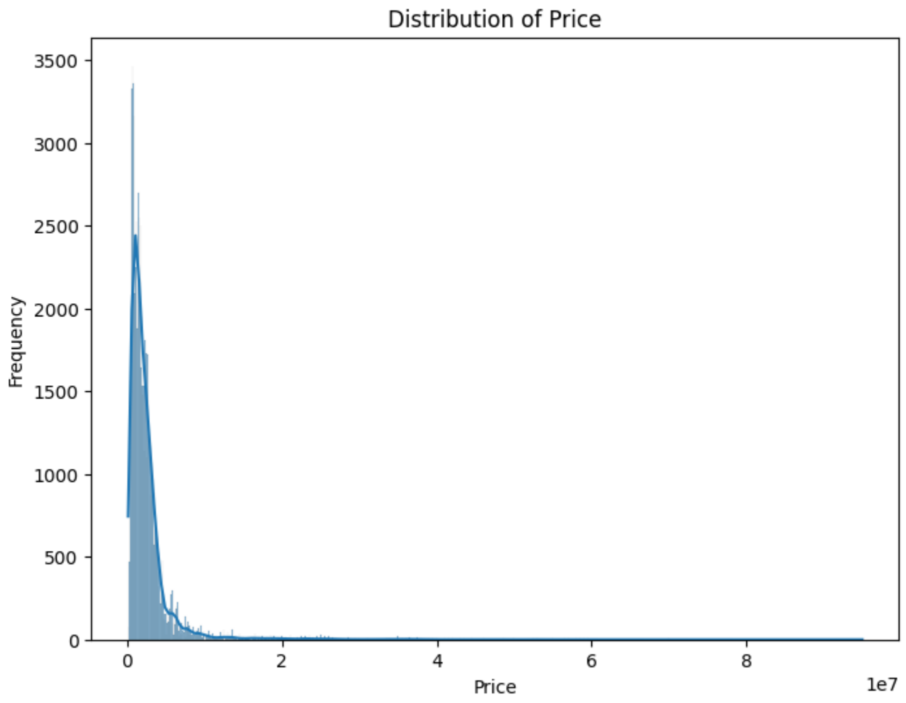

Figure 1: Distribution of Target Feature (Price)

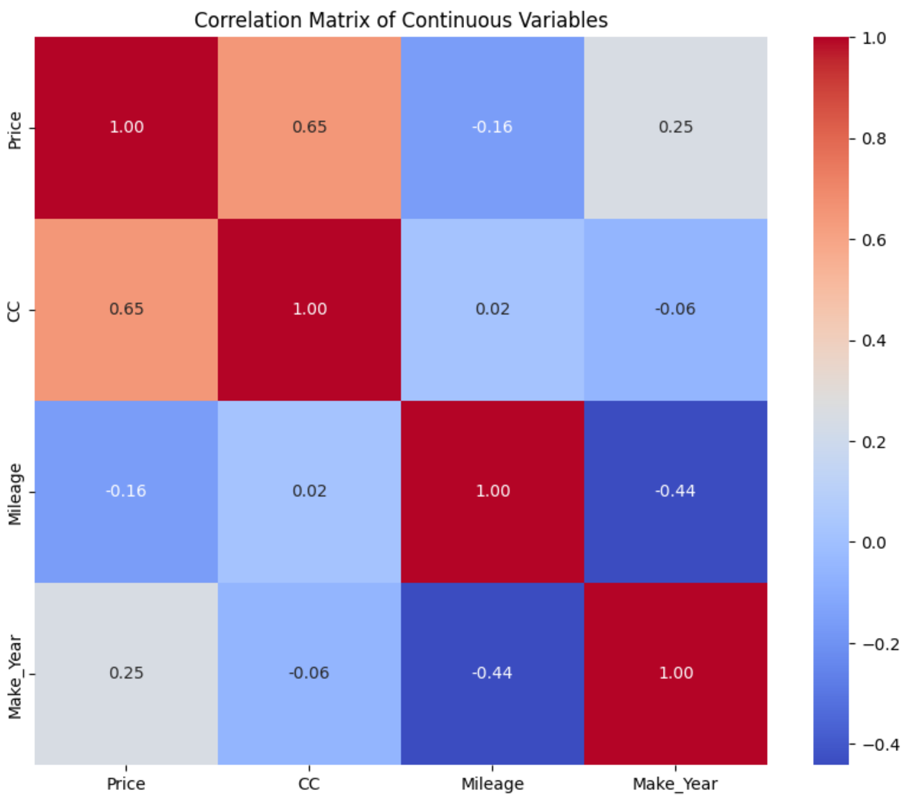

Figure 2: Initial Correlation Matrix

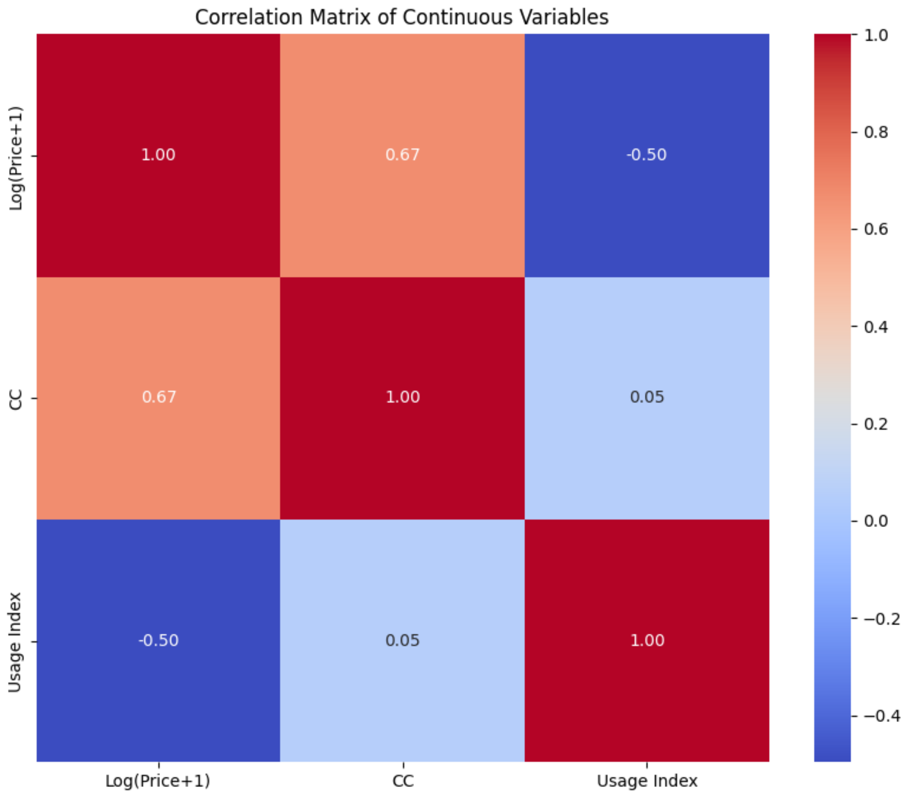

Figure 3: Final Correlation Matrix

c) Handling Low-Frequency Values: Features such as 'Make', 'Model', and 'Registered City' contained values with low frequencies. To retain meaningful information, only the top occurring values were preserved, while others were replaced with 'Other'.

Figures 4, 5, 6, and 7 represent the plots showing the distributions of 'Make', 'Model', 'Version', and 'Registered City'.

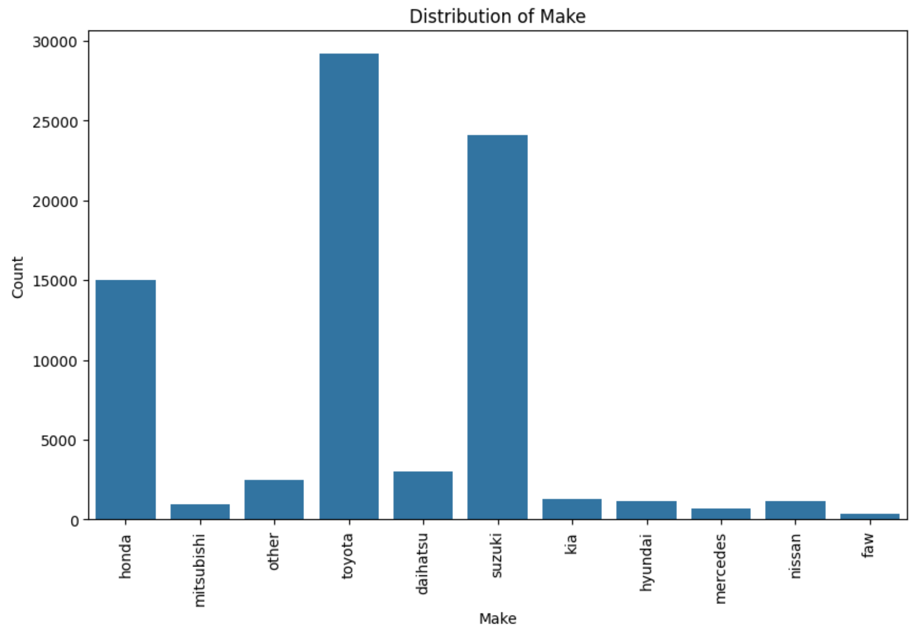

Figure 4: Engineered Make Column Distribution

d) Feature Selection: After reevaluation, the 'Version' column was dropped as it didn't contribute significant information. Retaining only informative columns helps reduce dimensionality and improves model efficiency.

3 Data Preprocessing

a) Handling Missing Values: Null values in the 'Version' column were replaced with 'unknown', and rows with 'Call For Price' in the Price column were removed due to their non-quantitative nature. Given the large dataset size, removing rows was preferable over imputation to maintain data integrity.

b) Textual Data Standardization: Textual (categorical) columns were standardized by removing extra spaces and converting to lowercase to ensure consistency. Also white spaces (' ') were replaced with underscores ('_').

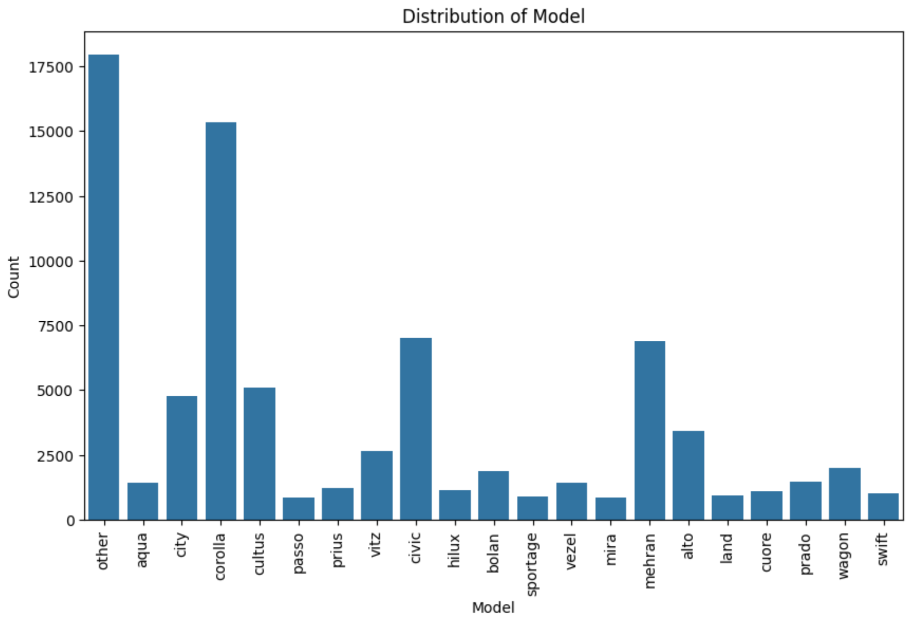

Figure 5: Engineered Model Column Distribution

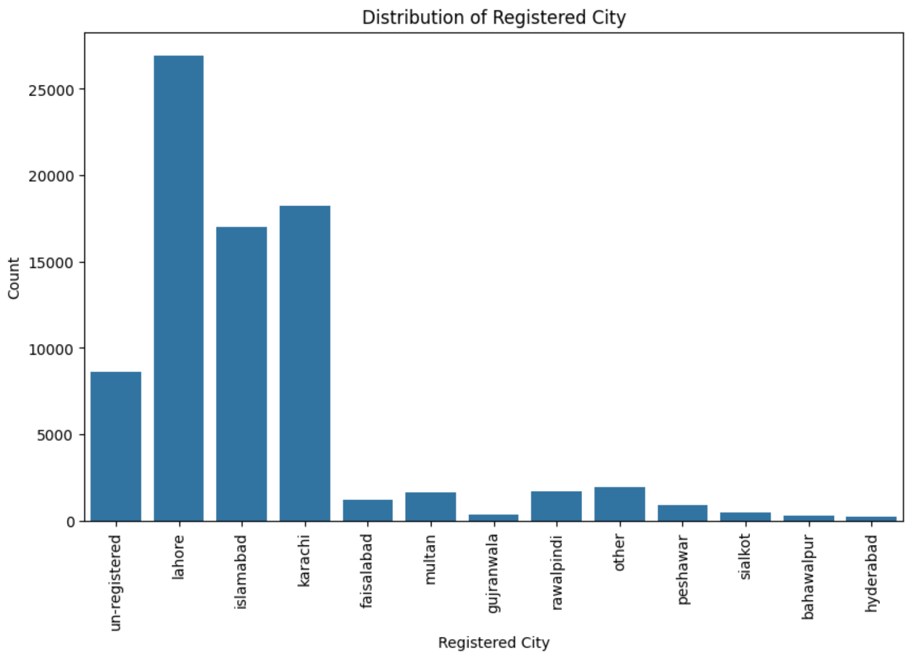

Figure 6: Engineered Registered City Column Distribution

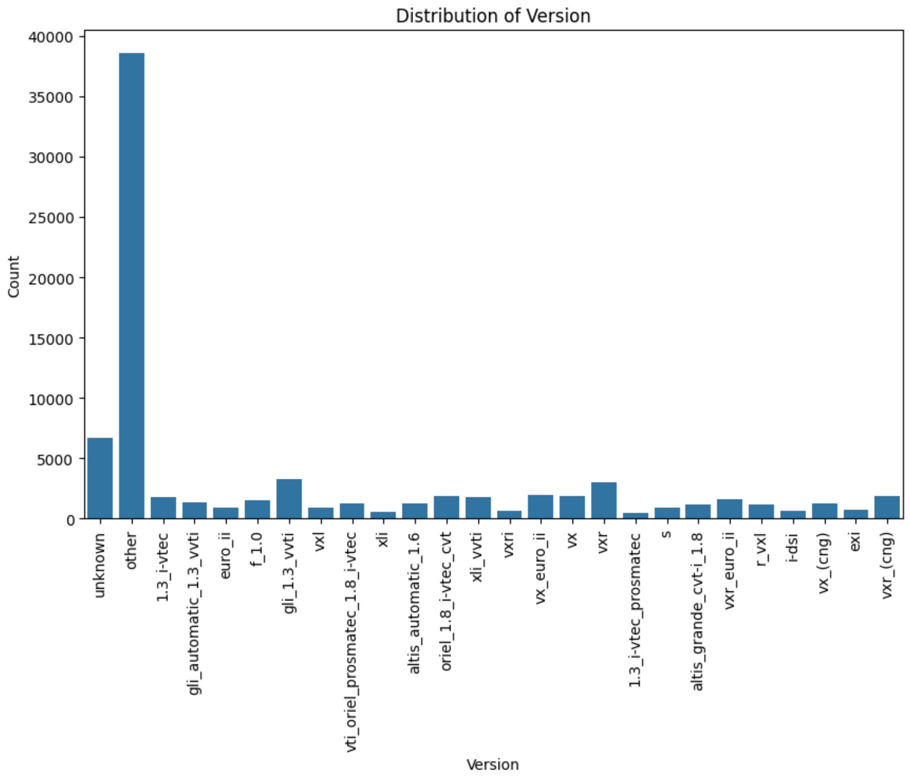

Figure 7: Engineered Version Column Distribution

c) One-Hot Encoding: Categorical features were one-hot encoded to convert them into numerical format suitable for modeling. One-hot encoding was preferred over label encoding due to the nominal nature of categorical features.

d) Numerical Feature Scaling: Numerical features were scaled to ensure uniform scales across all features, preventing any single feature from dominating the model due to its magnitude.

4 Initial Model Evaluation

a) Train/Test Split: A split of 80/20 was employed for training and testing.

b) Linear Regression: Linear Regression performed poorly due to the high dimensionality of the dataset, leading to the curse of dimensionality.

c) Lasso Regression: Lasso Regression performed well despite the curse of dimensionality, likely due to its ability to perform feature selection by shrinking less important feature coefficients to zero.

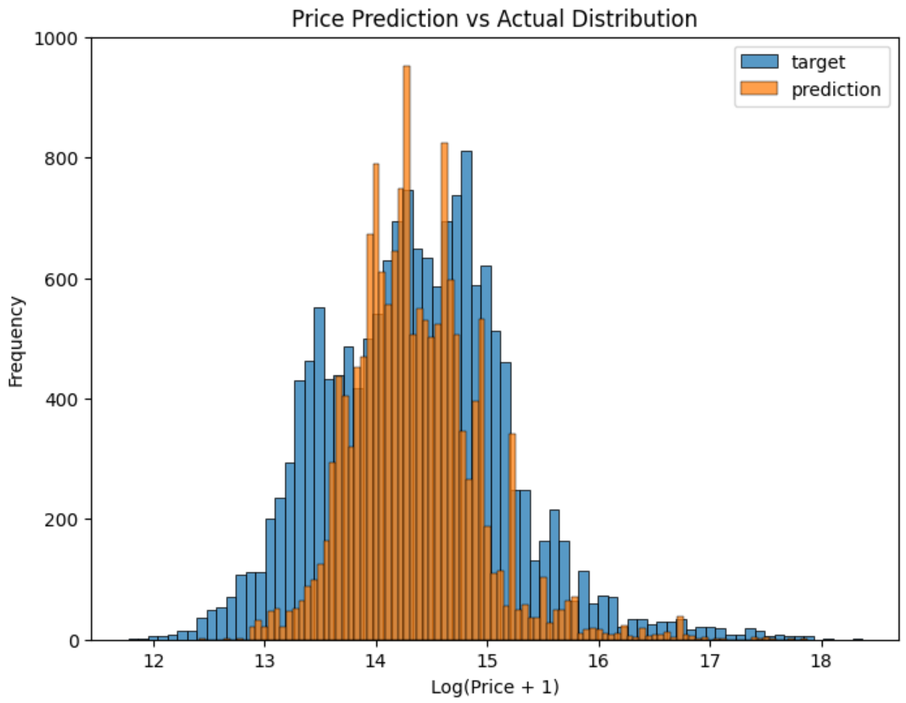

Figure 8: Initial Lasso Regression's Actual vs Predicted Performance on Test Data

d) Random Forest Regressor: Random Forest Regressor with n_estimators=15 showed exceptional performance even on un-engineered dataset.

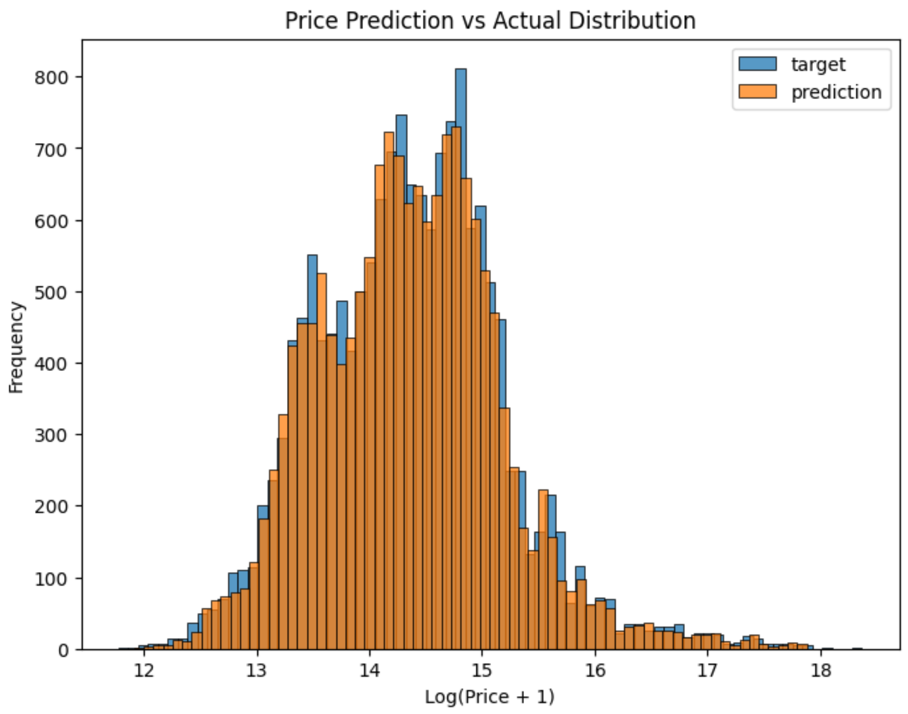

Figure 9: Initial Random Forest Regressor's Actual vs Predicted Performance on Test Data

e) Initial Results Comparison:

Table 1: Initial Model Performance Comparison

Model	**Mean Squared Error**
Linear Regression	3.9098e+18
Lasso Regression	0.1755
Random Forest Regressor	0.0423

5 Final Results Evaluation

- Feature Engineering Impact: After feature engineering and dimensionality reduction, the number of columns reduced significantly (from 1957 to just 49), leading to improved performance of linear models.

- Linear Regression: Linear Regression performed exceptionally well beating even the performance of Lasso Regression as well.

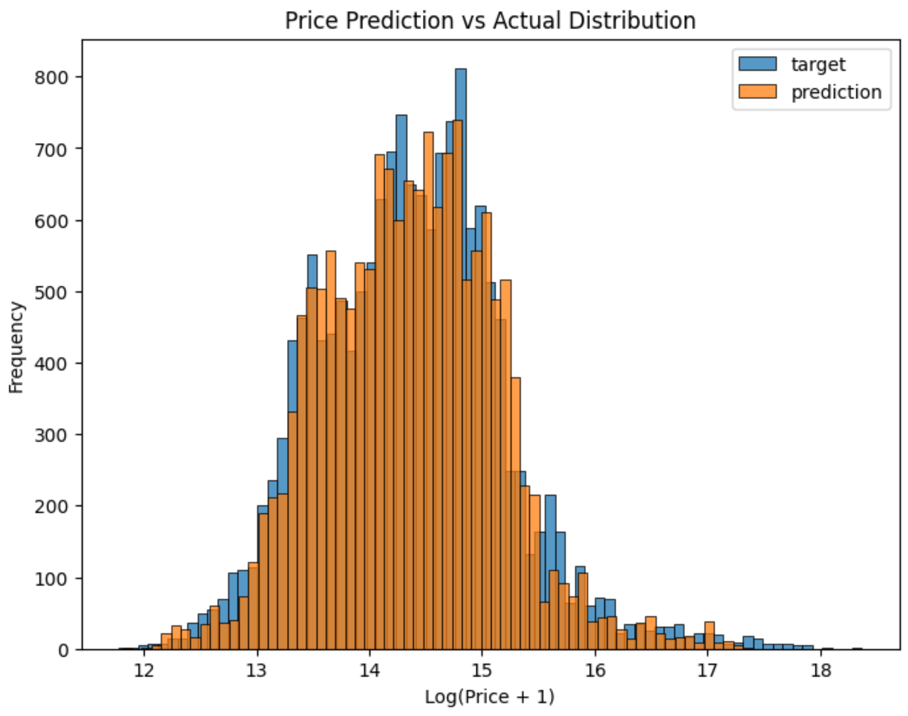

Figure 10: Linear Regression's Actual vs Predicted Performance on Test Data

- Lasso Regression: Lasso Regression showed minimal improvement due to its inherent feature selection mechanism.
- Random Forest Regressor: Random Forest Regressor exhibited slight improvement, suggesting that feature engineering mitigated but didn't entirely eliminate the curse of dimensionality.
- Decision Tree Regressor:Decision Tree Regressor showed good performance due to its ability to select the most important/relevant features.
- Gradient Boosting Regressor: Gradient Boosting Regressor showed superior performance compared to Decision Tree but was slightly inferior to Random Forest.
- Final Results Comparison:

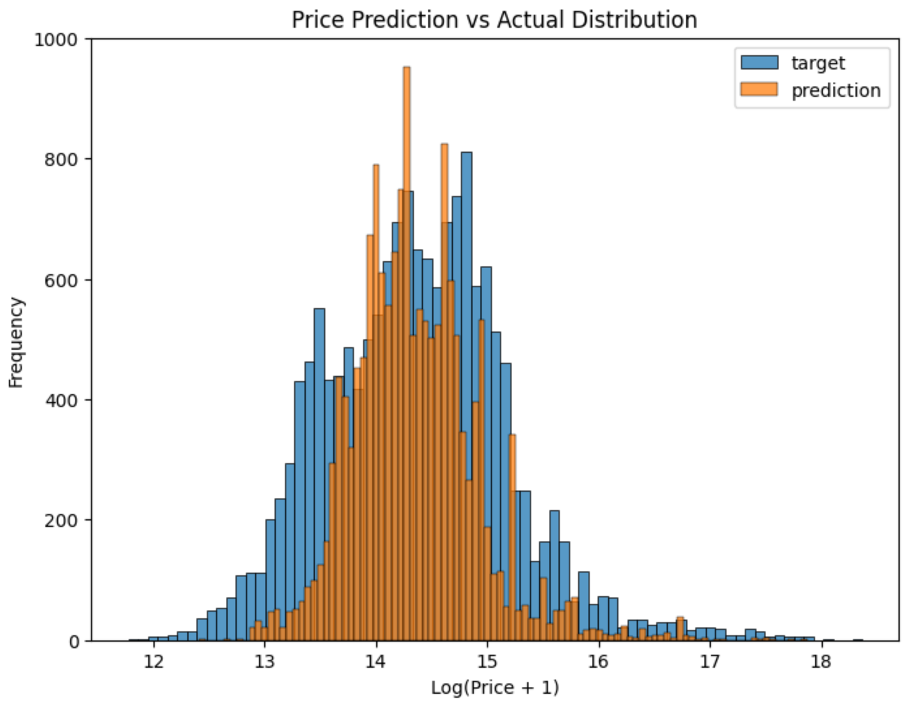

Figure 11: Lasso Regression's Actual vs Predicted Performance on Test Data

Table 2: Final Model Performance Comparison

Model	**Mean Squared Error**
Linear Regression	0.0801
Lasso Regression	0.1755
Random Forest Regressor	0.0481
Decision Tree Regressor	0.0701
Gradient Boosting Regressor	0.0586

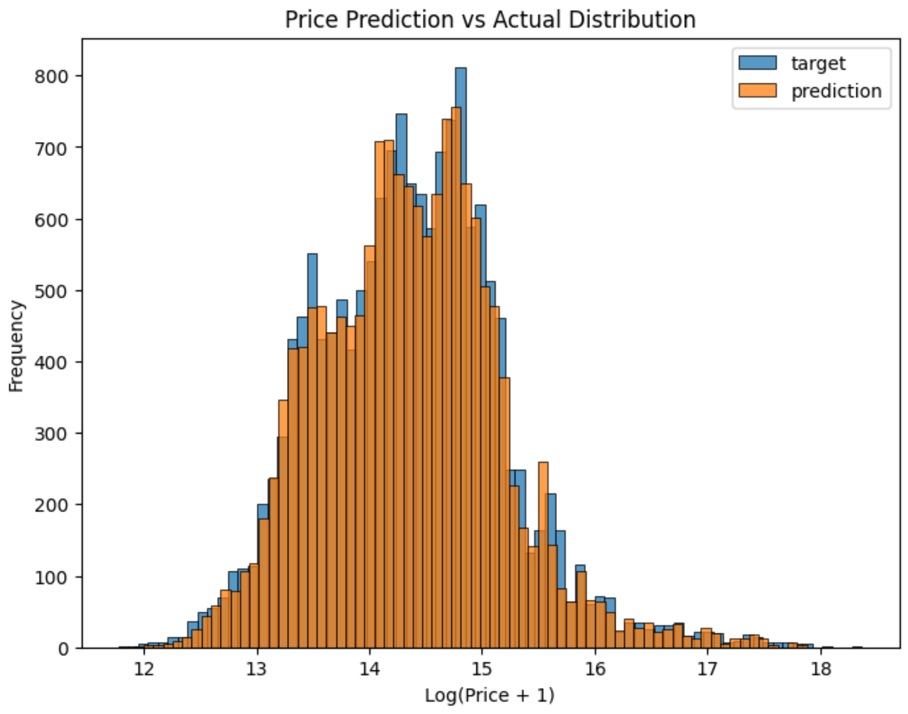

Figure 12: Random Forest Regressor's Actual vs Predicted Performance on Test Data

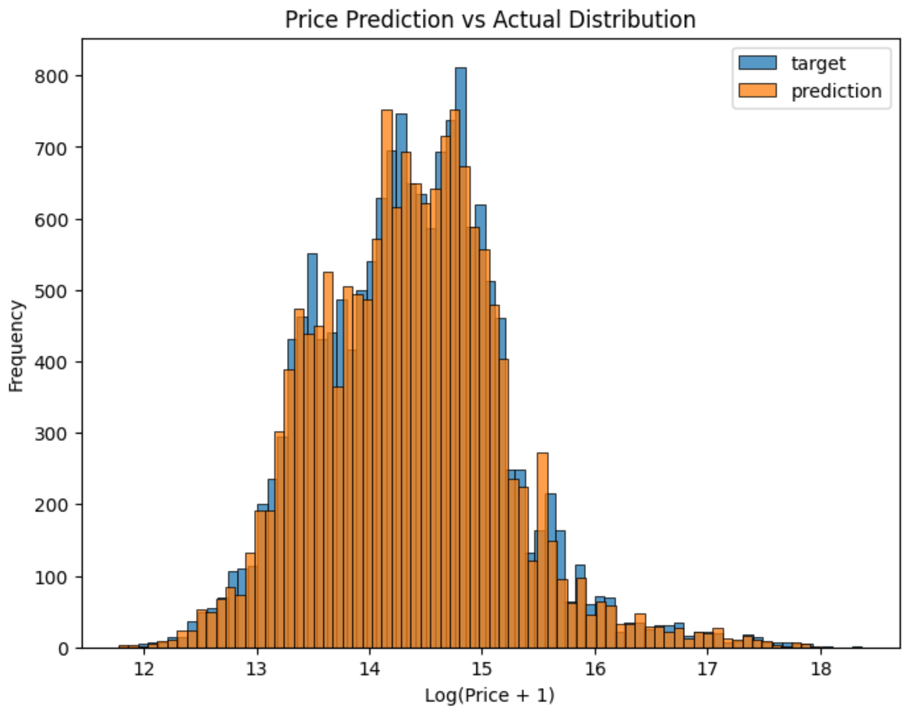

Figure 13: Decision Tree Regressor's Actual vs Predicted Performance on Test Data

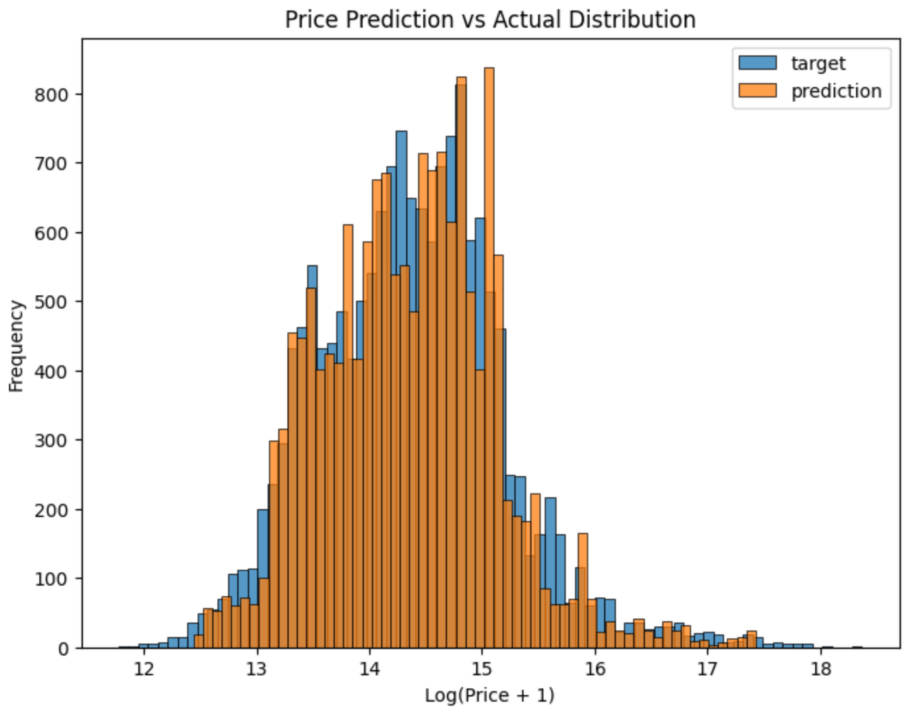

Figure 14: Gradient Boosting Regressor's Actual vs Predicted Performance on Test Data

6 Hyperparameter Tuning

Given the superior performance of Random Forest Regressor, hyperparameter tuning was conducted to optimize its performance further.

- Parameter Grid: A parameter grid was defined to explore various combinations of hyperparameters.
  - **n_estimators:** [15, 30, 45]
  - **max_depth:** [None, 5, 10]
  - **min_samples_split:** [2, 5]
  - **min_samples_leaf:** [1, 2]
- Results: The tuned model with the best parameters exhibited reduced mean squared error, indicating improved predictive performance.
  - **Best Parameters:**
    - **max_depth:** None
    - **min_samples_leaf:** 2
    - **min_samples_split:** 5
    - **n_estimators:** 45
  - **Mean Squared Error:** 0.0442

7 Conclusion

In order to really see the performance of our model in practice, we back-transformed the price column into its original scale. After testing, we found the MAE value to be 284113.9476.

In wrapping up our evaluation of the car price prediction model, it's essential to explain the Mean Absolute Error (MAE) value we obtained. While MAE is an important measure, let's focus on why it might seem higher than expected and why it's actually reasonable considering the outliers in our dataset.

We chose MAE because it's good at handling outliers and easy to understand. But let's talk about our MAE value. It might seem big at first – "284113.9476" – making us worry about our model's accuracy. But let's take a closer look. Our dataset has some really high values, outliers we call them, that can throw off our error calculations.

These outliers don't mean our model is bad. They're just really tricky to predict accurately. When we look at the big picture, our model is pretty good at making predictions for most of the data points. Even though our MAE seems large compared to the mean price of cars in our dataset, it's just a small part of the huge outliers we have.

Following are some figures showing Actual vs Predicted in Original Scale. They further prove our point by demonstrating that the errors in predictions

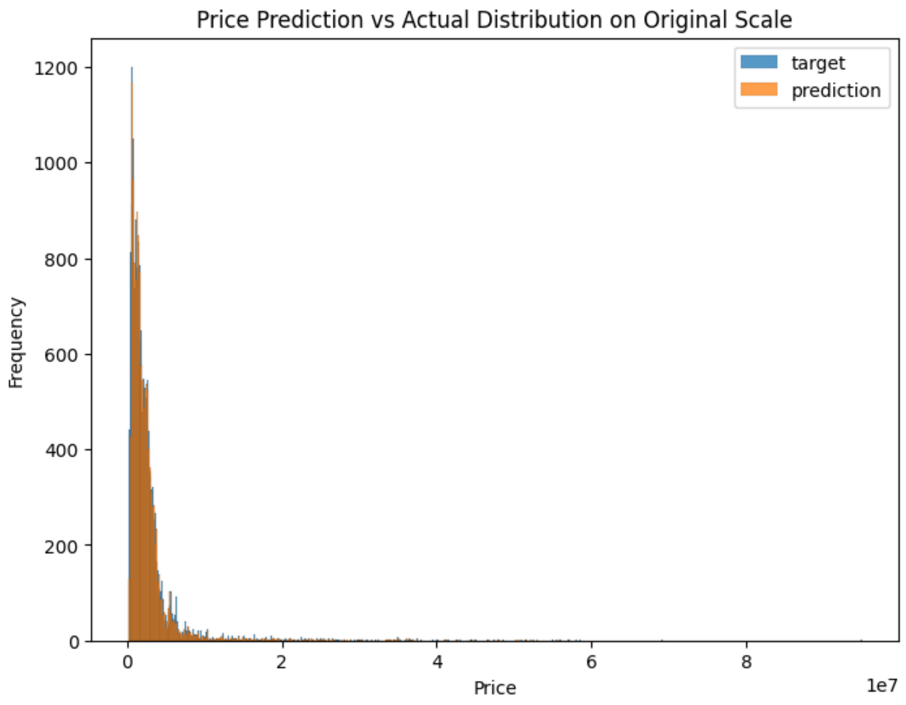

Figure 15: Actual vs Predicted in Original Scale

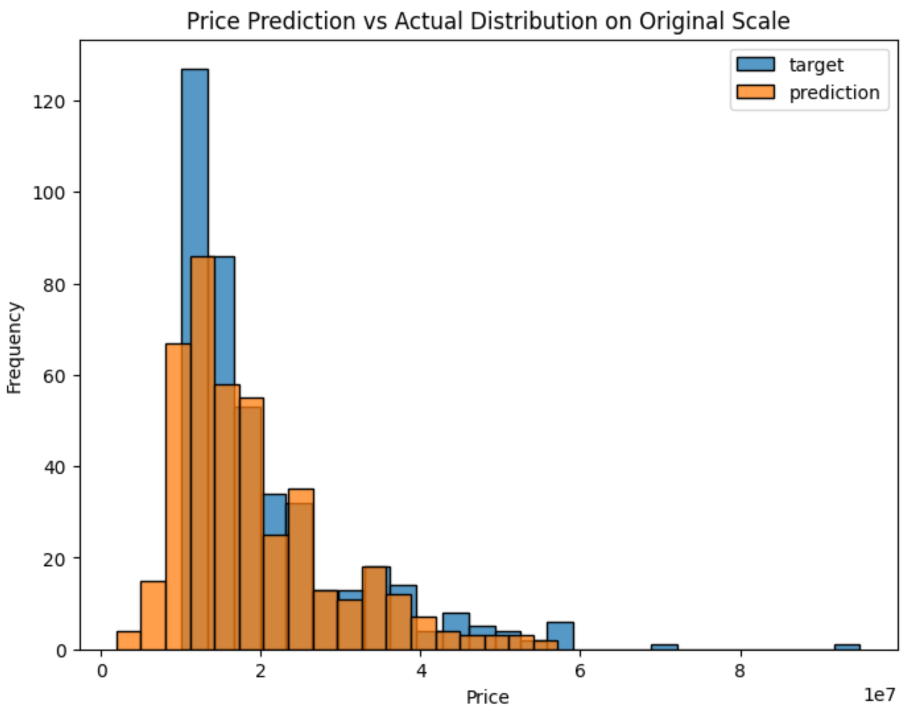

Figure 16: Actual vs Predicted for values Greater than 1e7

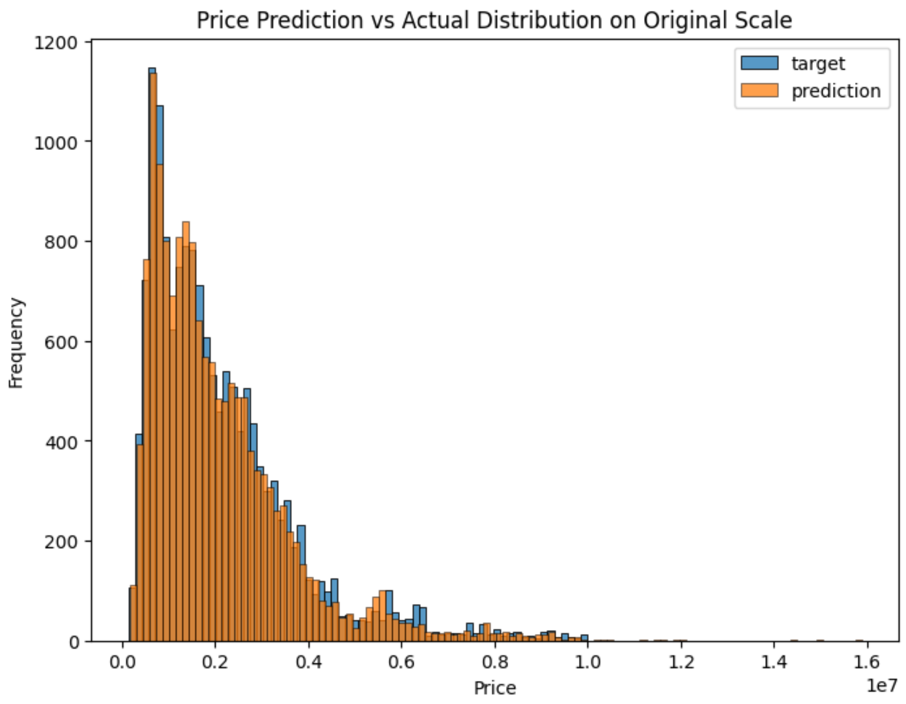

Figure 17: Actual vs Predicted for values Less than 1e7

are generally towards the tail of the distribution, while around the mean the predictions are quite accurate.

So, while our MAE might seem high, it's not because our model is bad. It's because predicting car prices is tough, especially with such extreme outliers. Overall, our model is doing a solid job handling these challenges and giving us reliable predictions for most situations.

NOTE: Neural Networks would have performed really well for our dataset capturing the complex relationships of the Predictors against the Target Feature. But since this task was related to ML so I tried to use ML specific techniques only.

Another NOTE: SVR (Support Vector Regression) is one ML technique that would excel in our situation but since it was taking a lot of time to train, so I had omit it from the code and report.