Regression Analysis Framework

Giovanni De Francesco

December 2024

Introduction

The purpose of this assignment is to develop a predictive model for the variable *car price* and interpret its results.
The dataset, sourced from Kaggle, aims to uncover the factors influencing car pricing. From now on, this dataset will be referred to as *<car>*. It consists of both numerical and categorical variables. Before beginning the analysis, it's essential to understand the meaning of each variable. The *Data Dictionary* provides valuable clarification. To simplify the distinction, variables are categorized as either categorical or numerical.

Categorical Data:

- Symboling: Insurance risk rating ranging from -3 (very safe) to 3 (high risk)
- CarName: Manufacturer and model name
- Fueltype: Type of fuel used (e.g., petrol, diesel)
- Aspiration: Type of aspiration system employed
- Doornumber: Number of doors
- Carbody: Type of car body (e.g., sedan, hatchback)
- Drivewheel: Drivetrain configuration (e.g., front-wheel drive)
- Enginelocation: Placement of the engine within the car (e.g., front, rear)
- Enginetype: Engine configuration (e.g., inline, V-type)
- Cylindernumber: Number of engine cylinders
- Fuelsystem: Fuel delivery system (e.g., injection, carburetor)

Numerical Data:

- Car_ID: Unique identifier for each record
- Wheelbase: Distance between the front and rear axles
- Carlength: Overall length of the car
- Carwidth: Width of the car
- Carheight: Height of the car
- Curbweight: Weight of the car excluding passengers and cargo
- Enginesize: Engine displacement or size
- Boreratio: Ratio of the engine cylinder bore diameter to its stroke
- Stroke: Length of the engine cylinder stroke
- Compressionratio: Ratio of maximum to minimum cylinder volume during compression
- Horsepower: Engine power output in horsepower
- Peakrpm: Maximum revolutions per minute (RPM)
- Citympg: Fuel efficiency in urban driving conditions (miles per gallon)
- Highwaympg: Fuel efficiency on highways (miles per gallon)
- Price: Market price of the car

Categorical data in Colab is initially read as *Objects*. For more efficient processing and precise handling, they can be explicitly converted into *categories*.

Exploratory Data Analysis

To summarize, the dataset <*car*> consists of 25 variables and 205 rows. The EDA (Exploratory Data Analysis) aims to identify potential errors within the variables, understand their relationships with the target variable, and help select the most relevant features for the model.
To analyze these relationships with the target variable, it is essential to first understand the nature of the target itself.

Target Variable

	price
count	205.00
mean	13276.71
std	7988.85
min	5118.00
25%	7788.00
50%	10295.00
75%	16503.00
max	45400.00

Figure 1: Characteristics of the Target Variable

car price is a numerical variable with a mean of approximately $13,277 and a standard deviation of approximately $7,989. Its distribution can be analyzed to gain a deeper understanding of its characteristics.

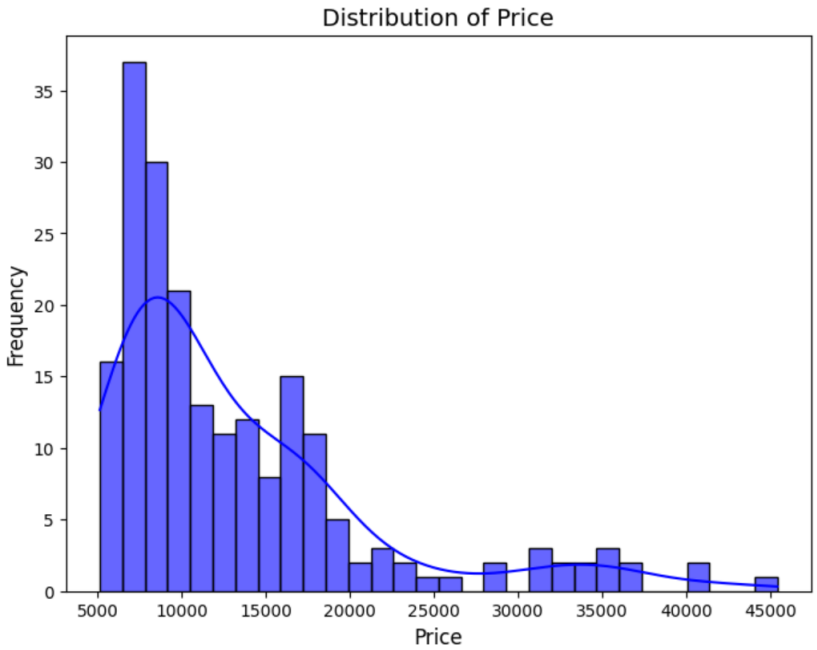
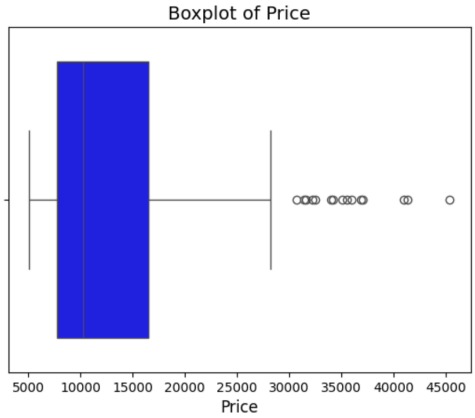


Figure 2: Distribution of Car Prices

The chart shows a right-skewed distribution of prices, where most values are concentrated between 5,000$ and 15,000$, with frequencies decreasing as prices increase.
The histogram highlights the frequency of prices in intervals, while the density curve smooths out the distribution, peaking in the lower price range. The long tail towards higher prices indicates a smaller number of premium or outliers values. This distribution suggests that lower prices dominate the dataset, with higher prices being less common.

A box-plot can complements the histogram and density plot by summarizing the distribution's key statistical features in a compact way, enabling deeper insights.



The box-plot shows a significant concentration of data within the interquartile range, while several points outside this range indicate the presence of outliers. These outliers correspond to prices significantly higher than the rest of the data. The plot suggests a right-skewed distribution, possibly requiring a logarithmic transformation for normalization.

Figure 3: Box-plot of Target Variable

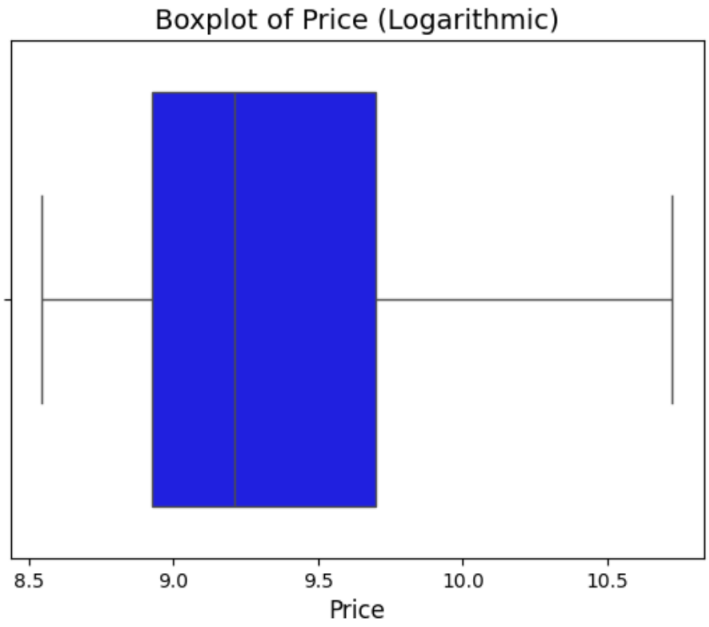


Figure 4: Box-plot with Logarithmic Transformation

The transformation has notably altered the distribution's characteristics compared to the original data. After the log transformation, the spread of values appears more balanced, as the transformation compresses the higher values while maintaining the structure of the lower ones. This adjustment makes the distribution more suitable for statistical analyses that assume normality or reduced skewness. As a result, the data now reflects its central tendencies more clearly without being overly influenced by the largest prices.
Overall, the logarithmic transformation has improved the representation of the data, mitigating the influence of outliers and skewness, and making the dataset more sensitive to further analysis and modeling.

Categorical Variable Study

As stated previously, there are 11 categorical variables in the dataset. A useful way to analyze their influence on the target variable (price) is to compare the mean price for each category within a variable against the overall mean price of the dataset.

When building a car price prediction model, it is advisable to remove the *CarName* variable because it serves as an identifier for specific car models rather than a meaningful feature that can explain the price. Including this variable in the model could lead to overfitting. This is because the model might simply memorize associations between car names and prices, rather than learning the underlying factors that influence price.

After dropping the *CarName* variable, a table with a grid of plots that examine the mean prices for each category of the remaining variables is made.

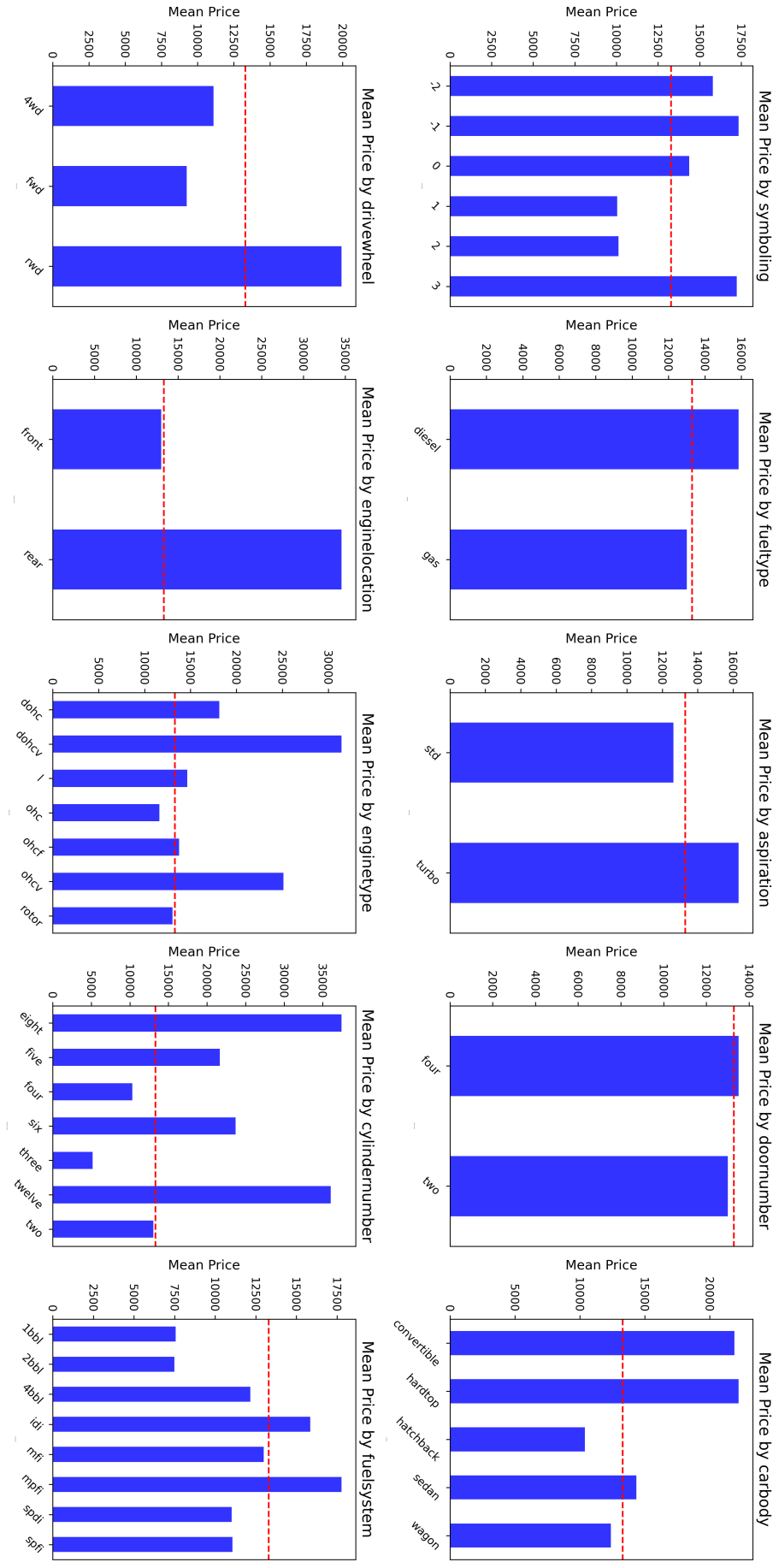


Figure 5: Matrix of Plots

For each variable, the greater the difference between the mean prices of the categories and the overall mean price of the dataset, the more informative that variable is likely to be for predicting the target variable.

In this case, all variables are informative, except for the *doornumber*, which has only two categories, and both categories have mean prices that are very similar to the overall mean price of the dataset, hence it's dropped.

Numerical Variable Study

As was done with the target variable, it is important to study the distribution of the numerical variables to determine whether a transformation, such as a logarithmic transformation, is appropriate. This step helps identify skewness, outliers, or other characteristics that may affect the interpretability or performance of statistical models and visualizations.

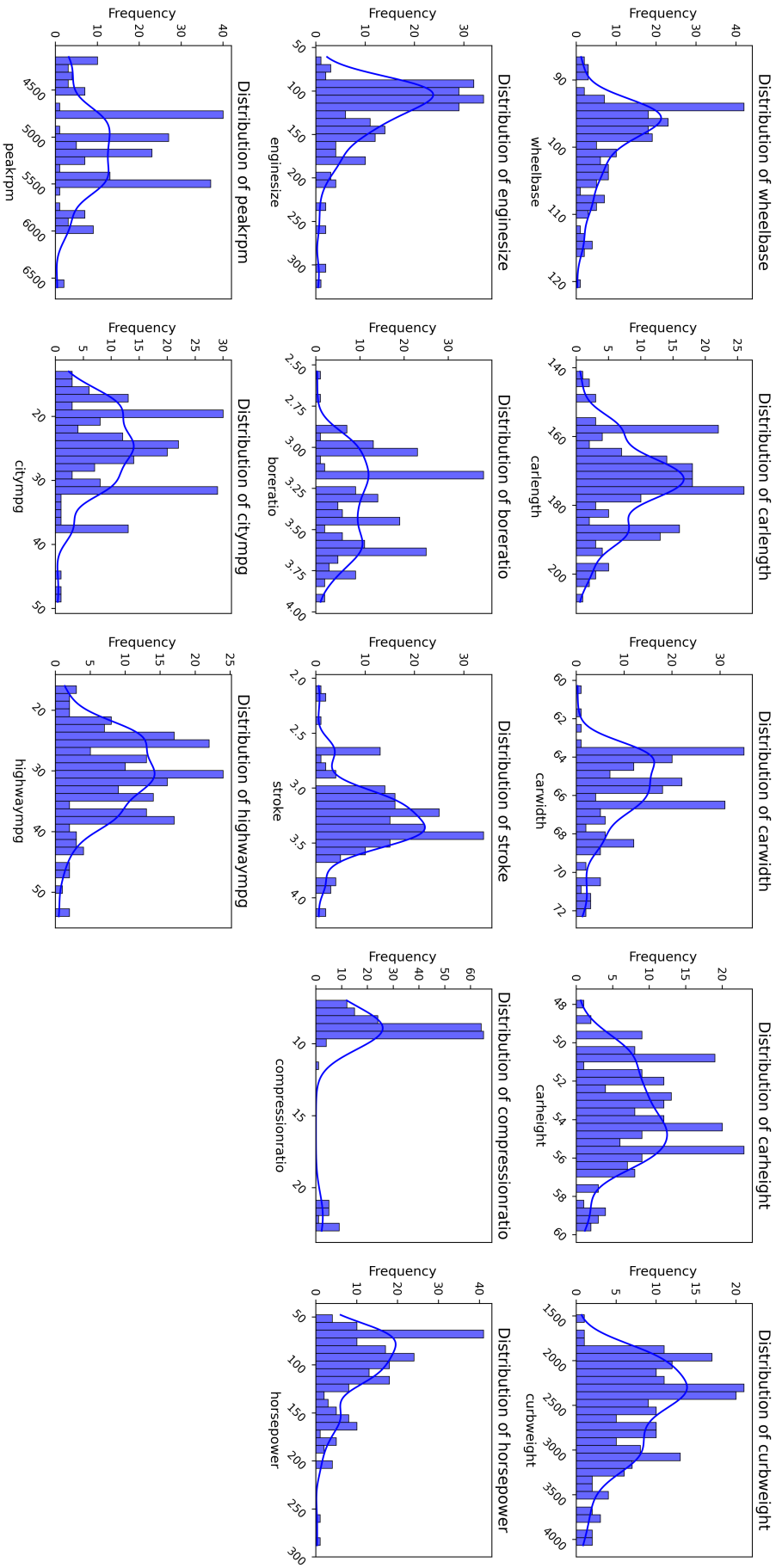


Figure 6: Matrix of Plots

Variables that exhibit right skewness and contain many outliers can benefit from a logarithmic transformation. After analyzing *Figure 6*, this transformation can be applied to the following variables: *enginesize*, *curbweight*, *horsepower*,*compressionrate*, *peakrpm* and *citympg* .

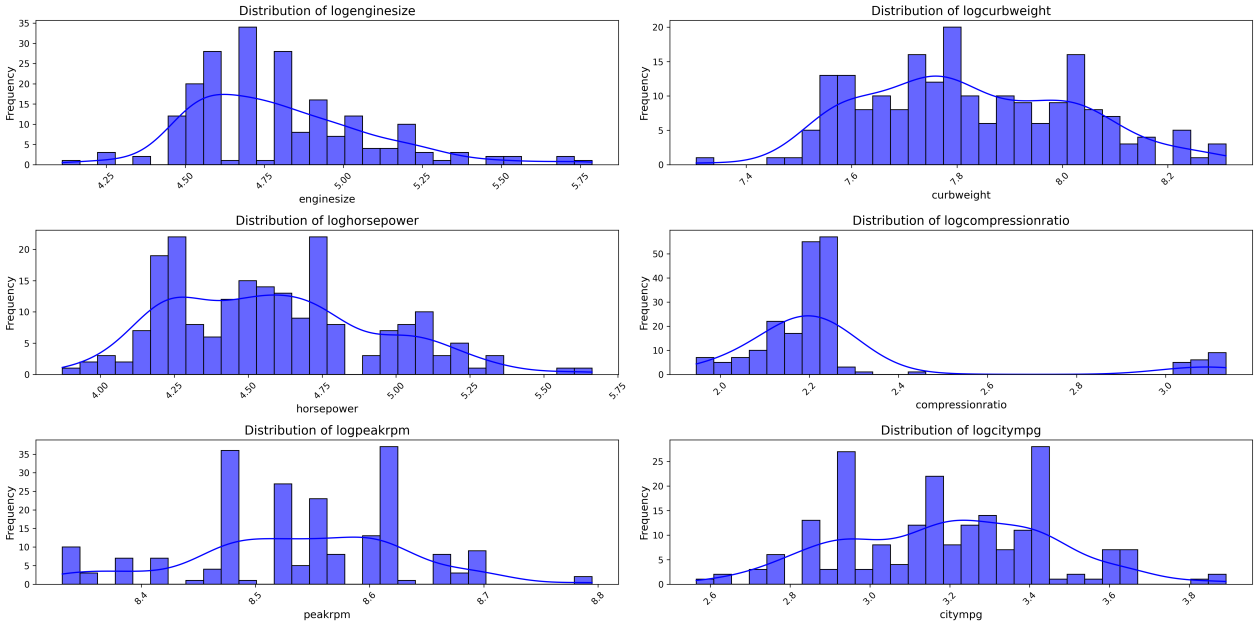


Figure 7: Matrix of Transformed Variables

Certainly, the distribution has improved, but some doubts remain regarding the presence of outliers. A useful approach to further investigate this is by computing and analyzing the corresponding boxplots.

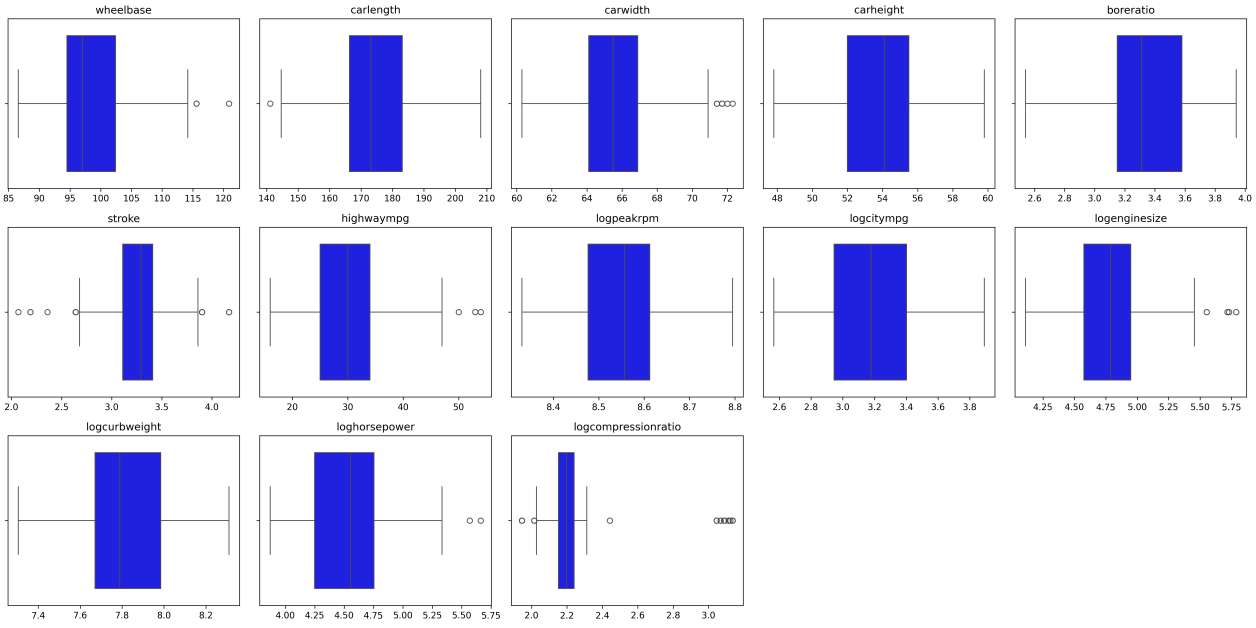


Figure 8: Matrix of Boxplots

There are still some IQR outliers present in both the transformed and non-transformed variables, although, in general, the transformation has improved the distribution.

The issue with outliers is that, without input from a technician, we cannot determine whether the units with outliers are actual errors or if they represent extreme values for particular cars (e.g., luxury cars, sports cars, etc.). Some units even have multiple outliers. If a unit has more than one outlier, it may indicate that the car has special characteristics. To understand whether these outliers are valid, we will mark them with a $-$ if the value is below the lower bound and with a $*$ if the value is above the upper bound.

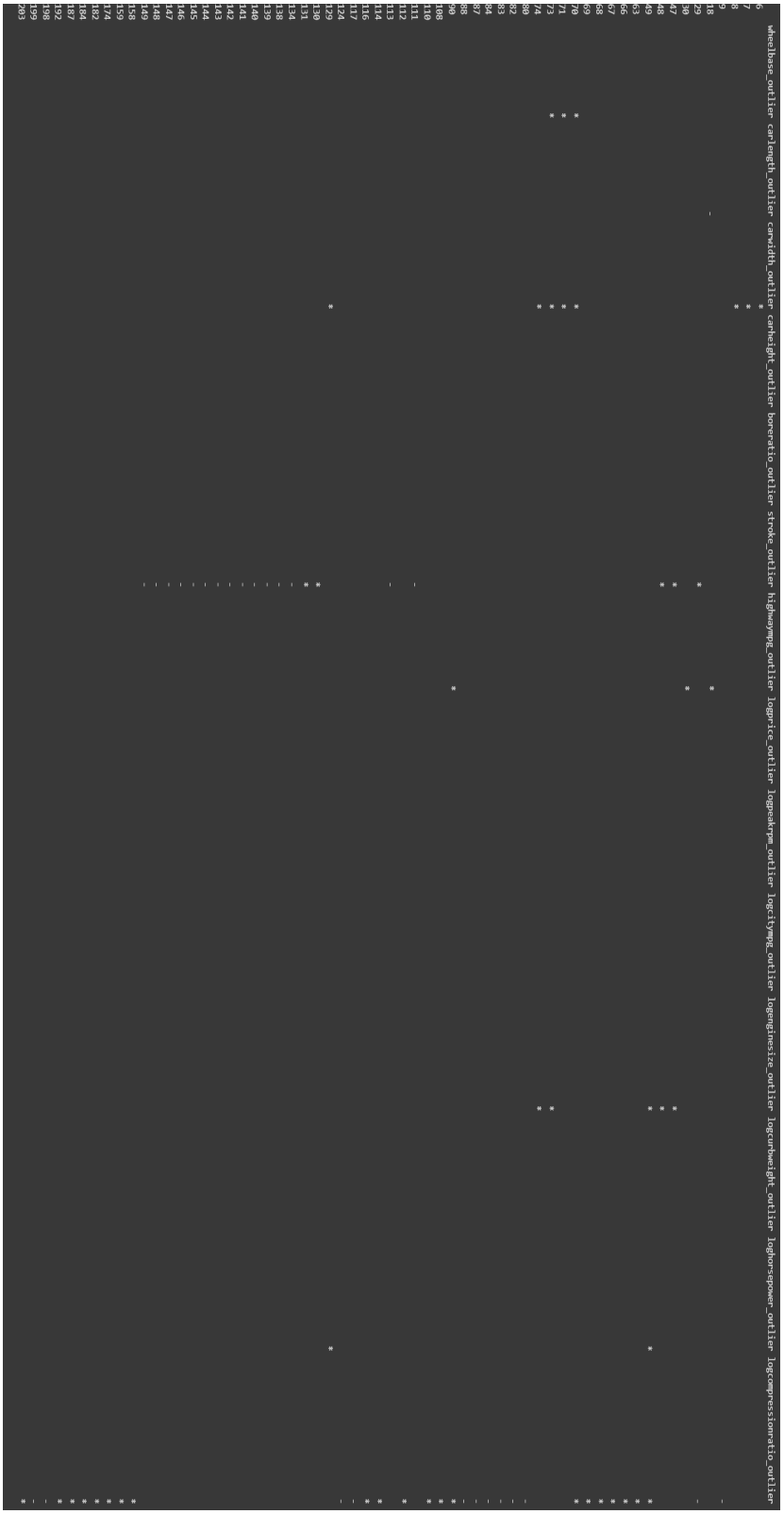


Figure 9: Outliers Display

Fifty units have only one outlier, which represents almost 25% of the dataset. However, there is a possibility that many of these units correspond to the same *CarName*, suggesting that they may not be errors but rather characteristics of specific car models. For these reasons, none of the outliers will be dropped.

Figure 10: Car with one outlier

From *figure 10*, it can be seen that many outliers correspond to the same vehicle.

Based on the scatter plots and the correlation matrix, variables that are not linearly dependent on the target can be considered for removal.

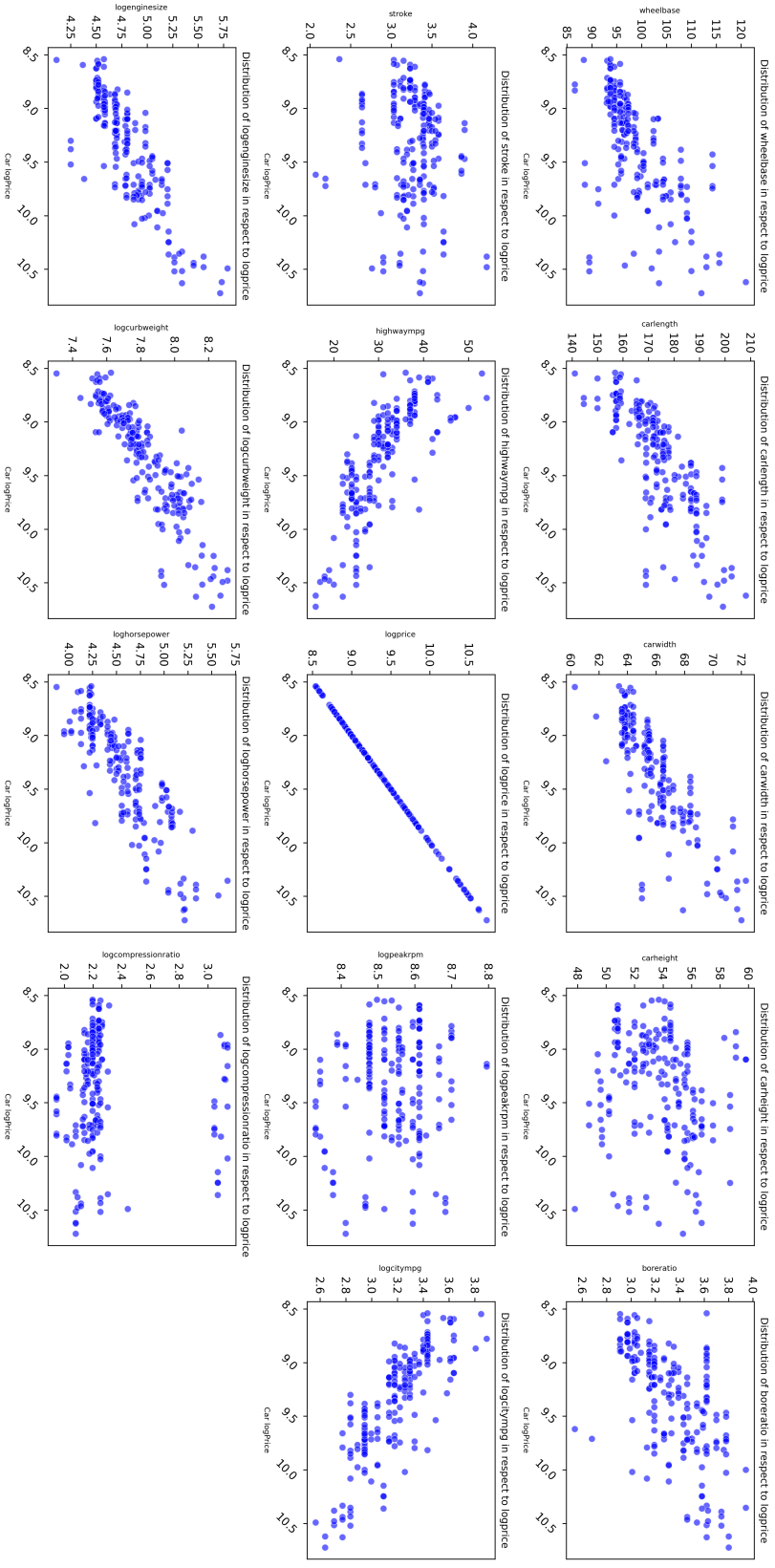


Figure 11: Scatter-plot Matrix

	wheelbase	carlength	carwidth	carheight	boreratio	stroke	highwaympg	logprice	logpeakrpm	logcitympg	logenginesize	logcurbweight	loghorsepower	logcompressionratio
wheelbase	1.000000	0.874587	0.795144	0.589435	0.488750	0.160959	-0.544082	0.629401	-0.368427	-0.476808	0.593839	0.763385	0.412677	0.224971
carlength	0.874587	1.000000	0.841118	0.491029	0.606454	0.129533	-0.704662	0.767864	-0.288465	-0.672877	0.731821	0.889075	0.615988	0.132891
carwidth	0.795144	0.841118	1.000000	0.279210	0.559150	0.182942	-0.677218	0.802544	-0.223528	-0.670986	0.755994	0.858690	0.666306	0.153960
carheight	0.589435	0.491029	0.279210	1.000000	0.171071	-0.055307	-0.107358	0.162798	-0.324267	-0.023249	0.119669	0.301473	-0.071621	0.255928
boreratio	0.488750	0.606454	0.559150	0.171071	1.000000	-0.055909	-0.587012	0.610565	-0.258658	-0.578175	0.637616	0.670269	0.609800	-0.010601
stroke	0.160959	0.129533	0.182942	-0.055307	-0.055909	1.000000	-0.043931	0.097992	-0.075529	-0.042852	0.224142	0.159479	0.110743	0.150266
highwaympg	-0.544082	-0.704662	-0.677218	-0.107358	-0.587012	-0.043931	1.000000	-0.775197	-0.050940	0.960792	-0.702646	-0.818116	-0.848605	0.297894
logprice	0.629401	0.767864	0.802544	0.162798	0.610565	0.097992	-0.775197	1.000000	-0.097356	-0.812608	0.845738	0.890683	0.842740	0.060110
logpeakrpm	-0.368427	-0.288465	-0.223528	-0.324267	-0.258658	-0.075529	-0.050940	-0.097356	1.000000	-0.121075	-0.275747	-0.261355	0.128955	-0.435180
logcitympg	-0.476808	-0.672877	-0.670986	-0.023249	-0.578175	-0.042852	0.960792	-0.812608	-0.121075	1.000000	-0.719217	-0.804476	-0.898624	0.347250
logenginesize	0.593839	0.731821	0.755994	0.119669	0.637616	0.224142	-0.702646	0.845738	-0.275747	-0.719217	1.000000	0.864911	0.813047	0.020233
logcurbweight	0.763385	0.889075	0.858690	0.301473	0.670269	0.159479	-0.818116	0.890683	-0.261355	-0.804476	0.864911	1.000000	0.792661	0.107443
loghorsepower	0.412677	0.615988	0.666306	-0.071621	0.609800	0.110743	-0.848605	0.842740	0.128955	-0.898624	0.813047	0.792661	1.000000	-0.266127
logcompressionratio	0.224971	0.132891	0.153960	0.255928	-0.010601	0.150266	0.297894	0.060110	-0.435180	0.347250	0.020233	0.107443	-0.266127	1.000000

Figure 12: Correlation Matrix

Based on both *Figure 11 and Figure 12*:

- *carheight*
- *stroke*
- *logpeakrpm*
- *logcompressionrate*

Can be dropped from the model because of their relation with the target.

Multicollinearity

Multicollinearity can lead to significant issues in a model because it becomes challenging to isolate the individual effects of each variable on the dependent variable. When multicollinearity is present, the model may struggle to determine which variable is truly influencing the target, resulting in unreliable or unstable coefficient estimates. To identify variables with high correlation, we extract from the previous correlation matrix only the values less than -0.8 or greater than 0.8, focusing on the strongest relationships.

	wheelbase	carlength	carwidth	boreratio	highwaympg	logprice	logcitympg	logenginesize	logcurbweight	loghorsepower
wheelbase	1.00	0.87	nan	nan	nan	nan	nan	nan	nan	nan
carlength	0.87	1.00	0.84	nan	nan	nan	nan	nan	0.89	nan
carwidth	nan	0.84	1.00	nan	nan	0.80	nan	nan	0.86	nan
boreratio	nan	nan	nan	1.00	nan	nan	nan	nan	nan	nan
highwaympg	nan	nan	nan	nan	1.00	nan	0.96	nan	-0.82	-0.85
logprice	nan	nan	0.80	nan	nan	1.00	-0.81	0.85	0.89	0.84
logcitympg	nan	nan	nan	nan	0.96	-0.81	1.00	nan	-0.80	-0.90
logenginesize	nan	nan	nan	nan	nan	0.85	nan	1.00	0.86	0.81
logcurbweight	nan	0.89	0.86	nan	-0.82	0.89	-0.80	0.86	1.00	nan
loghorsepower	nan	nan	nan	nan	-0.85	0.84	-0.90	0.81	nan	1.00

Figure 13: Multicollinearity Pre-transformation

Among the independent variables that exhibit strong correlations with each other, *logcurbweight and loghorsepower* are excluded. Both of these variables show a high correlation with *logenginesize*, which can effectively replace them, providing similar information without introducing redundancy.

	wheelbase	carlength	carwidth	boreratio	highwaympg	logprice	logcitympg	logenginesize
wheelbase	1.00	0.87	nan	nan	nan	nan	nan	nan
carlength	0.87	1.00	0.84	nan	nan	nan	nan	nan
carwidth	nan	0.84	1.00	nan	nan	0.80	nan	nan
boreratio	nan	nan	nan	1.00	nan	nan	nan	nan
highwaympg	nan	nan	nan	nan	1.00	nan	0.96	nan
logprice	nan	nan	0.80	nan	nan	1.00	-0.81	0.85
logcitympg	nan	nan	nan	nan	0.96	-0.81	1.00	nan
logenginesize	nan	nan	nan	nan	nan	0.85	nan	1.00

Figure 14: Multicollinearity Post-transformation

Model Creation

After performing the variable selection, let's take an initial look at the final dataset.

	wheelbase	carlength	carwidth	boreratio	highwaympg	logprice	logcitympg	logenginesize	symboling	fueltype	aspiration	carbody	drivewheel	enginelocation	enginetype	cylindernumber	fuelsystem
0	88.6	168.8	64.1	3.47	27	9.510075	3.044522	4.867534	3	gas	std	convertible	rwd	front	dohc	four	mpfi
1	88.6	168.8	64.1	3.47	27	9.711116	3.044522	4.867534	3	gas	std	convertible	rwd	front	dohc	four	mpfi
2	94.5	171.2	65.5	2.68	26	9.711116	2.944439	5.023881	1	gas	std	hatchback	rwd	front	ohcv	six	mpfi
3	99.8	176.6	66.2	3.19	30	9.543235	3.178054	4.691348	2	gas	std	sedan	fwd	front	ohc	four	mpfi
4	99.4	176.6	66.4	3.19	22	9.767095	2.890372	4.912655	2	gas	std	sedan	4wd	front	ohc	five	mpfi

Figure 15: Final Dataset

Regression models do not accept categorical data directly, making encoding a necessary step. However, there are various methods to encode categorical variables, with two common techniques being one-hot encoding and label encoding. Label encoding is applied to *symboling*, as it is the only variable with a clear ordinal relationship. For the remaining categorical variables, such as *fueltype, aspiration, carbody, drivewheel, enginetype, and fuelsystem*, one-hot encoding is applied to ensure that the model treats them as separate, non-ordinal categories.
Subsequently the data is divided in training (80%) and test (20%), using a random state.
After doing this, both test and training are scaled, having cure of excluding the one-hot encoded variables.
Now the regression model can be performed.

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
model = LinearRegression()
model.fit(X_train_scaled, y_train)
y_pred = model.predict(X_test_scaled)
r2= round(r2_score(y_test, y_pred),3)
mae= round(mean_absolute_error(y_test, y_pred),3)
rmse= round(np.sqrt(mean_squared_error(y_test, y_pred)),3)
print("\nRegression Performance:")
print("Mean Absolute Error:", mae)
print("Root Mean Square Error:", rmse)
print("R-squared:", r2)
```

```
Regression Performance:
Mean Absolute Error: 0.137
Root Mean Square Error: 0.17
R-squared: 0.893
```

The problem is that the results appear in logarithmic scale, which isn't the best way to interpret results, for these reason data is converted in it's original scale.

```
y_test_exp = np.exp(y_test)
y_pred_exp = np.exp(y_pred)
r2= round(r2_score(y_test_exp, y_pred_exp),3)
mae= round(mean_absolute_error(y_test_exp, y_pred_exp),3)
rmse= round(np.sqrt(mean_squared_error(y_test_exp, y_pred_exp)),3)
print("\nRegression Performance:")
print("Mean Absolute Error:", mae)
```

```
print("Root Mean Square Error:", rmse)
print("R-squared:", r2)
```

```
Regression Performance:
Mean Absolute Error: 2026.219
Root Mean Square Error: 3248.745
R-squared: 0.866
```

Lasso & Ridge

Lasso and Ridge can be used to improve to regression model.

```
from sklearn.linear_model import Lasso, Ridge

lasso = Lasso()
ridge = Ridge()

param_grid = {'alpha': [0.01, 0.001, 0.0001, 0.0001]}
param_grid1 = {'alpha': [0.01, 0.1, 1, 10, 100]}

grid_search_lasso = GridSearchCV(lasso, param_grid, scoring='neg_mean_squared_error', cv
    =5)
grid_search_ridge = GridSearchCV(ridge, param_grid1, scoring='neg_mean_squared_error', cv
    =5)

grid_search_lasso.fit(X_train_scaled, y_train)
best_lasso = grid_search_lasso.best_estimator_

grid_search_ridge.fit(X_train_scaled, y_train)
best_ridge = grid_search_ridge.best_estimator_

print(f"Miglior alpha per Lasso: {grid_search_lasso.best_params_['alpha']}")
print(f"Miglior alpha per Ridge: {grid_search_ridge.best_params_['alpha']}")

y_pred_lasso = best_lasso.predict(X_test_scaled)
mae_lasso = mean_absolute_error(y_test, y_pred_lasso)
rmse_lasso = np.sqrt(mean_squared_error(y_test, y_pred_lasso))
r2_lasso = r2_score(y_test, y_pred_lasso)

y_pred_ridge = best_ridge.predict(X_test_scaled)
mae_ridge = mean_absolute_error(y_test, y_pred_ridge)
rmse_ridge = np.sqrt(mean_squared_error(y_test, y_pred_ridge))
r2_ridge = r2_score(y_test, y_pred_ridge)

print("\nLasso Regression Performance:")
print(f"MAE: {mae_lasso}")
print(f"RMSE: {rmse_lasso}")
print(f"R-squared: {r2_lasso}")

print("\nRidge Regression Performance:")
print(f"MAE: {mae_ridge}")
print(f"RMSE: {rmse_ridge}")
print(f"R-squared: {r2_ridge}")
```

```
Miglior alpha per Lasso: 0.0001
Miglior alpha per Ridge: 0.1

Lasso Regression Performance:
MAE: 0.1386215941147492
RMSE: 0.1716715516044555
R-squared: 0.8913809055861984

Ridge Regression Performance:
MAE: 0.13623177108638576
RMSE: 0.16896947829855327
R-squared: 0.8947732785786506
```

As in the linear regression, the results are presented in logarithmic scale. They need to be transformed in their original scale.

```
y_pred_lasso_exp = np.exp(y_pred_lasso)
y_pred_ridge_exp = np.exp(y_pred_ridge)

mae_lasso = round(mean_absolute_error(y_test_exp, y_pred_lasso_exp),3)
rmse_lasso = round(np.sqrt(mean_squared_error(y_test_exp, y_pred_lasso_exp)),3)
r2_lasso = round(r2_score(y_test_exp, y_pred_lasso_exp),3)

mae_ridge = round(mean_absolute_error(y_test_exp, y_pred_ridge_exp),3)
rmse_ridge = round(np.sqrt(mean_squared_error(y_test_exp, y_pred_ridge_exp)),3)
r2_ridge = round(r2_score(y_test_exp, y_pred_ridge_exp),3)

print("\nLasso Regression Performance:")
print(f"MAE: {mae_lasso}")
print(f"RMSE: {rmse_lasso}")
print(f"R-squared: {r2_lasso}")

print("\nRidge Regression Performance:")
print(f"MAE: {mae_ridge}")
print(f"RMSE: {rmse_ridge}")
print(f"R-squared: {r2_ridge}")
```

```
Lasso Regression Performance:
MAE: 2005.803
RMSE: 3186.913
R-squared: 0.871

Ridge Regression Performance:
MAE: 1955.303
RMSE: 3081.787
R-squared: 0.88
```

Performance Evaluation

The coefficients assigned by each model can be compared.

	Lasso Coefficients	Ridge Coefficients	Model Coefficients
wheelbase	0.057250	0.051132	0.052976
carlength	0.052458	0.050968	0.057001
carwidth	0.083005	0.084396	0.080651
boreratio	0.003354	0.003161	0.015175
highwaympg	0.111978	0.116478	0.125034
logcitympg	-0.270269	-0.277044	-0.287164
logenginesize	0.045269	0.049617	0.025686
symboling	0.013001	0.012046	0.011672
fueltype_gas	-0.151105	-0.077330	-0.077585
aspiration_turbo	0.088760	0.088847	0.090604
enginelocation_rear	0.704442	0.652667	0.707338
carbody_hardtop	-0.345932	-0.326401	-0.362340
carbody_hatchback	-0.252149	-0.244640	-0.252557
carbody_sedan	-0.195427	-0.187653	-0.203879
carbody_wagon	-0.279229	-0.268717	-0.287614
drivewheel_fwd	-0.159933	-0.157226	-0.174500
drivewheel_rwd	-0.000000	0.002601	-0.020349
enginetype_dohcv	-0.006646	-0.042772	-0.143096
enginetype_l	-0.192387	-0.185271	-0.187824
enginetype_ohc	0.061608	0.062078	0.061701
enginetype_ohcf	-0.036481	-0.029917	-0.061183
enginetype_ohcv	-0.039631	-0.054267	-0.072690
enginetype_rotor	-0.027353	-0.122167	-0.199275
fuelsystem_2bbl	-0.103629	-0.105190	-0.114920
fuelsystem_4bbl	-0.110854	-0.111898	-0.134217
fuelsystem_idi	0.000000	0.077330	0.077585
fuelsystem_mfi	-0.069193	-0.086259	-0.095609
fuelsystem_mpfi	0.000000	-0.001205	-0.009703
fuelsystem_spdi	-0.144770	-0.150653	-0.158278
fuelsystem_spfi	-0.029845	-0.043881	-0.056840
cylindernumber_five	-0.279807	-0.281806	-0.386616
cylindernumber_four	-0.323921	-0.328063	-0.444992
cylindernumber_six	-0.181610	-0.183163	-0.262014
cylindernumber_three	0.000000	0.000000	0.000000
cylindernumber_twelve	-0.089212	-0.106075	-0.135253
cylindernumber_two	-0.212158	-0.122167	-0.199275

Ridge and Lasso actually improve the model performance in respect to the simple linear regression. In particular it appears that *Ridge Regression* is the best model.

```
Regression Performance:
Mean Absolute Error: 2026.219
Root Mean Square Error: 3248.745
R-squared: 0.866

Lasso Regression Performance:
MAE: 2005.803
RMSE: 3186.913
R-squared: 0.871

Ridge Regression Performance:
MAE: 1955.303
RMSE: 3081.787
R-squared: 0.88
```

Once the best model has been selected (in this case, Ridge Regression), it is important to assess whether the results were due to chance based on the specific train-test split. To address this, the model is iterated 10 times, with a different train-test data split each time while maintaining the same $80\%-20\%$ ratio. The results are then averaged to ensure the stability and reliability of the model's performance.

```
Ridge Regression Performance (after 10 iterations with different train-test splits):
MAE: 1742.612 ± 184.418
RMSE: 2710.273 ± 409.707
R-squared: 0.896 ± 0.023
```

Ridge Regression Performance appears to be stable.