A Multiple Linear Regression Model for Predicting Car Prices

Dylan DeNicola and Natalie Cardoso

Rensselaer Polytechnic Institute

COGS 4210: Cognitive Modeling

Introduction

The problem we are trying to address is learning which variables are the most significant in predicting the price of a car and attempting to predict car prices from these variables using a multiple linear regression model. In our analysis, we compare our results with one of the top-rated analyses associated with the dataset on kaggle. In this comparison, we will see how well a Bayesian analysis using Stan can compare to an analysis using the prebuilt linear regression model from scikit-learn.

Methods

Software and Libraries

We perform our analysis in a Jupyter notebook using Python. The necessary libraries for the analysis are NumPy, pandas, Matplotlib, seaborn, scikit-learn, PyStan, and ArviZ.

Dataset and Preprocessing

The dataset we are using consists of 205 cars with 26 variables corresponding to each car. We use the car price as our dependent variable and the other 25 variables act as the features for our linear regression model. The kaggle analysis that we will be compare our results to also did some data preprocessing of their own. This involved splitting up the "CarName" feature into a "brand" and "model" feature, encoding the categorical features (all the features initially without a numerical value) into numerical values using the LabelEncoder function from scikit-learn, adding a new feature called "power_to_weight_ratio" which is the "horsepower" feature divided by the "curbweight" feature, adding a new feature for every numerical feature (all the features initially that had a numerical value, except "car_ID") that squares the value of the feature, and adding a new feature called "log_enginesize" which applies a log transform to the "enginesize" feature. The last preprocessing the kaggle analysis did was scaling all the numerical features using the StandardScaler function from scikit-learn. The kaggle analysis used all these features when performing its linear regression analysis using scikit-learn.

Baseline Model

The first model we created was a multiple linear regression model in Stan that tried to replicate the kaggle analysis as closely as possible. However, to get a working a Stan model, we needed to make some of our own preprocessing decisions. First, the price data wasn't normally distributed, instead, it was distributed like an exponential decay. To counteract this, we applied a log transformation to the price data to make it normally distributed. Next, the extra squared features and the "log_enginesize" that the kaggle analysis implemented caused the Stan model to not converge, so we didn't use them for our analysis. Lastly, we didn't include the "car_ID" as a feature because it wasn't relevant to the actual data. This resulted in our baseline model having 26 features. With this preprocessing done, we split the data into training and testing sets.

The multiple linear regression model specification for our baseline analysis is given below.

$$\sigma \sim Normal(0, 10)$$

$$\alpha \sim Normal(9.2, 2)$$

$$\beta \sim Normal(9.2, 2)$$

$$y_n \sim Normal(\alpha + \beta * x_n, \sigma)\ for\ n = 1, \dots, N$$

For σ, we restricted its value to be greater than or equal to 0, so we couldn't get a negative standard deviation for y_n. We also chose 0 for its mean and 10 for its standard deviation because we wanted the standard deviation for y_n to have varying values concentrated towards 0. For α and β, we chose 9.2 for their means and 2 for their standard deviations because we wanted to match the distribution of the log transformed price data. For y_n, we followed the standard linear regression model, although, in our actual implementation, we vectorized the Stan code. Lastly, we used 4 chains with 2000 samples and 1000 warmup steps to fit our model.

Feature Removal

After creating and assessing our baseline model, we wanted to systematically determine which features were the most important and least important for predicting car prices. To accomplish this, we analyzed the β values of our baseline model, and we kept any feature associated with a β value greater than or equal to 0.01. This left our new model with 16 features to use for the prediction. We then formulated the Stan model to be exactly the same as our baseline model.

After evaluating the results of our second model, we wanted to try removing more features, so we repeated the same process. We analyzed the β values of our second model, and we kept any feature associated with a β value greater than or equal to 0.1. This left our third model with 5 features to use for the prediction. These 5 features are "fueltype", "enginelocation", "curbweight", "compressionratio", and "power_to_weight_ratio". As before, our model's specification remained the same. We then decided that removing any more features would make the model too simple and unable to accurately predict car prices.

Testing Different Priors

Once we determined which of the previous 3 models was the best, we used only the features from the best model for our prior sensitivity analysis. We tested out different means and standard deviations for σ, α, and β, but any large change to these parameters made our results worse. Thus, for our final model we only made slight alterations. For σ, we also tested out using a different prior distribution than a normal distribution. Since we were already restricting the value of σ to be greater than or equal to 0, we decided to try an inverse gamma distribution. We ended up setting both the shape and scale parameters to 1. Other shape and scale parameters either didn't change the result significantly or had worse performance. The multiple linear regression model specification for our final analysis is given below.

$$\sigma \sim Inverse\_Gamma(1, 1)$$

$$\alpha \sim Normal(9.4, 3)$$

$$\beta \sim Normal(9.4, 3)$$

$$y_n \sim Normal(\alpha + \beta * x_n, \sigma)\ for\ n = 1, \ldots, N$$

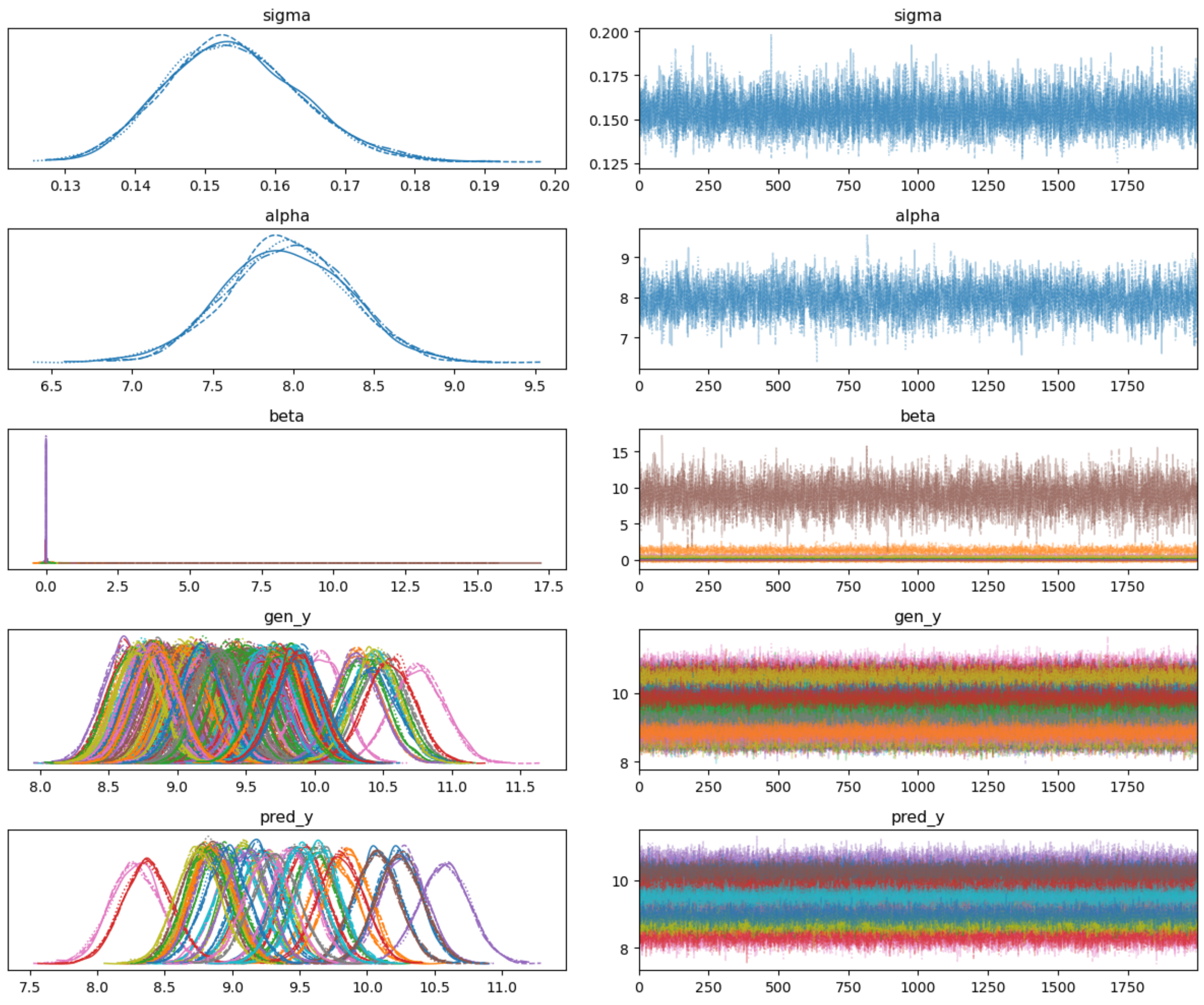
Assessing Model Fit and Performance

For each model, we first analyzed convergence and efficiency diagnostics from Stan. We also examined the convergence graphs from Stan. To check both the generative performance and predictive performance, we first compared the model's outputs to actual training and testing residuals' distributions using histograms. Then we calculated the root mean squared errors and R-squared values for both the generated output and predicted output. Lastly, we outputted the β values of each model to check which features had the most influence on the model's fit.

Results

Baseline Model Results

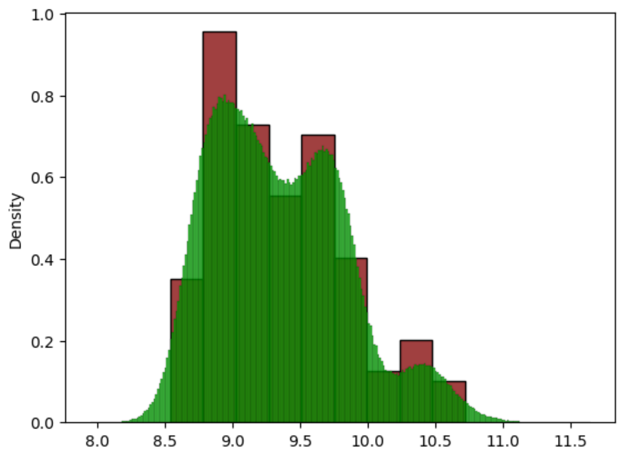
The model successfully converged as indicated by the visual diagnostic for MCMC given below. We see that there are no obvious patterns in convergence graphs, meaning that the model converged.

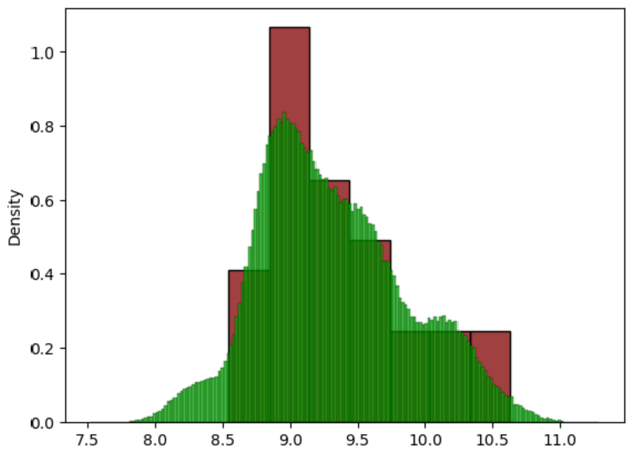


We also verified that the model converged and was efficient as indicated by the numeric diagnostic for MCMC given below. We see that "r_hat" is 1.0, meaning that the model converged and that the efficiency diagnostics are greater than 100 and less than the number of chains times the number of samples, meaning that the model was efficient.

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
sigma	0.154	0.010	0.136	0.171	0.000	0.000	7159.0	5814.0	1.0
alpha	7.959	0.377	7.290	8.709	0.006	0.004	3717.0	4623.0	1.0
beta[0]	-0.001	0.017	-0.032	0.032	0.000	0.000	6391.0	5638.0	1.0
beta[1]	1.281	0.394	0.540	2.003	0.007	0.005	3649.0	4572.0	1.0
beta[2]	0.082	0.054	-0.024	0.176	0.001	0.001	5749.0	6074.0	1.0
...	...	...	...	...	...	...	...	...	...
pred_y[36]	9.265	0.171	8.944	9.581	0.002	0.001	8280.0	7929.0	1.0
pred_y[37]	9.322	0.175	8.995	9.652	0.002	0.001	8251.0	7792.0	1.0
pred_y[38]	8.745	0.164	8.448	9.062	0.002	0.001	8005.0	7859.0	1.0
pred_y[39]	9.513	0.161	9.207	9.814	0.002	0.001	8000.0	7674.0	1.0
pred_y[40]	8.953	0.167	8.634	9.266	0.002	0.001	7729.0	7926.0	1.0

The marginal checks helped us verify that generative performance and predictive performance were relatively accurate. The generative performance is given on the left and the predictive performance is given on the right.





For the generative performance, the baseline model had a root mean squared error of 0.13995 and an R-squared score of 0.92099. This means that our model was able to reproduce the training data well.

For the predictive performance, the baseline model had a root mean squared error of 0.18430 and an R-squared score of 0.87481. This means that our model was able to predict new data pretty accurately. This is also slightly better than kaggle analysis' predictive performance result of an R-squared score of 0.861. However, the difference between the generative and predictive performance results also implies that the baseline model might be overfitting on the training data.

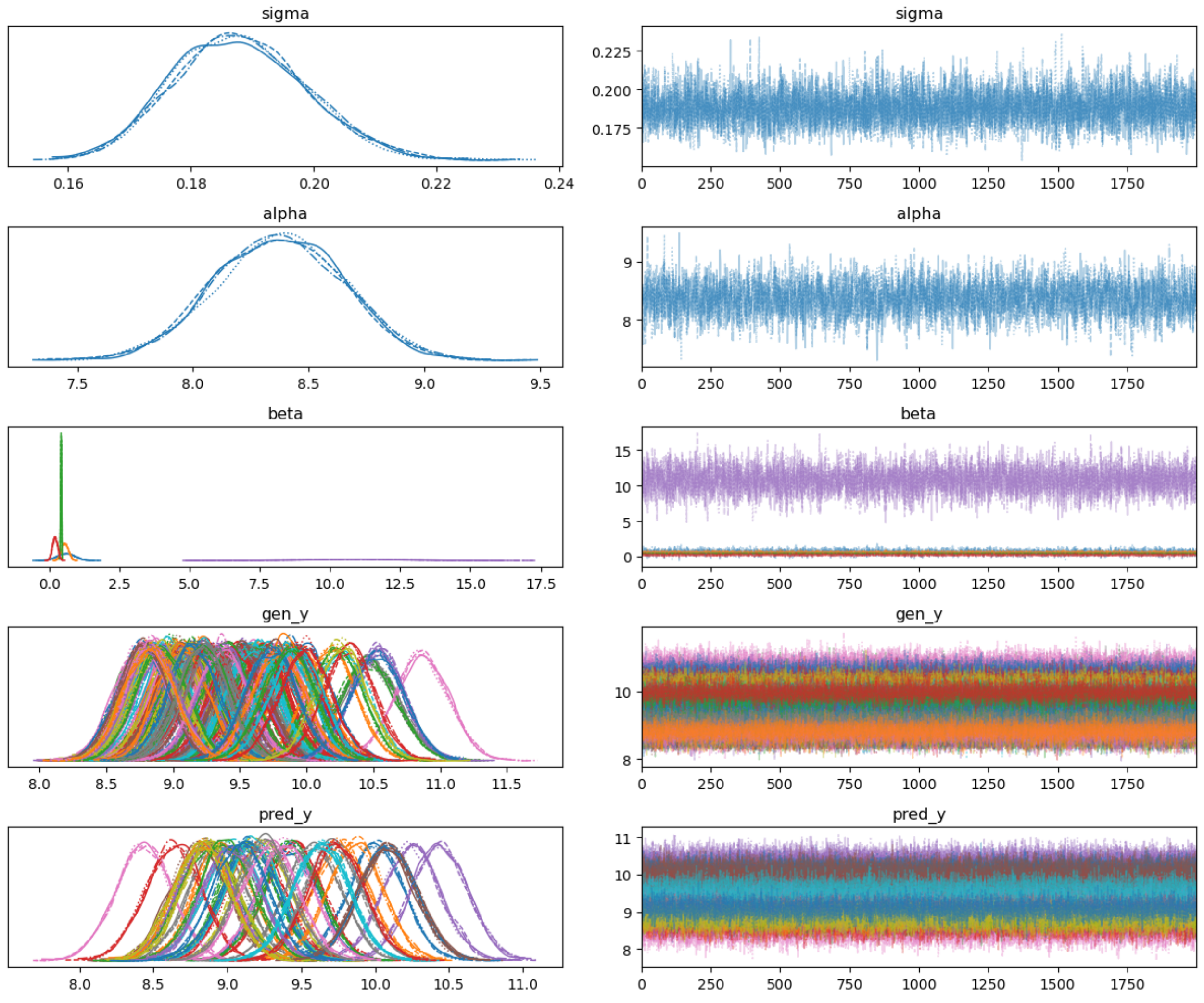
Feature Removal Results

The second model with 16 features also converged as indicated by the visual and numeric diagnostics for MCMC (see Jupyter notebook for these diagnostics). Like the baseline model, the second model had an "r_hat" of 1.0 and good efficiency diagnostics as well. The marginal checks showed us that the generative performance was slightly less accurate than the baseline model, but the predictive performance was about the same (see Jupyter notebook for these plots). To confirm this, we computed that the root mean squared error was 0.16190 and the R-squared score was 0.89427 for the generative performance. As for the predictive performance, the second model had a root mean squared error of 0.19373 and an R-squared score of 0.86167. Although these results are slightly worse than the baseline model, we see that the R-squared scores for the generative and predictive performances are closer than before. This indicates that our baseline model was in fact overfitting on the training data.

As indicated by the visual and numeric diagnostics for MCMC (see Jupyter notebook for these diagnostics), the third model with 5 features converged and was efficient. The marginal checks once again seemed to show us that the generative performance was getting worse while the predictive performance remained about the same (see Jupyter notebook for these plots). For the generative performance, the third model had a root mean squared error of 0.18193 and an R-squared score of 0.86648. For the predictive performance, the third model had a root mean squared error of 0.17560 and an R-squared score of 0.88635. We see that for this third model, the predictive performance is better than the generative performance. The predictive performance of the third model also surpasses the predictive performance of the baseline model. This suggests that the third model is more adaptable to new data, while the baseline model and second model are still overfitting on the training data.

Different Priors Results

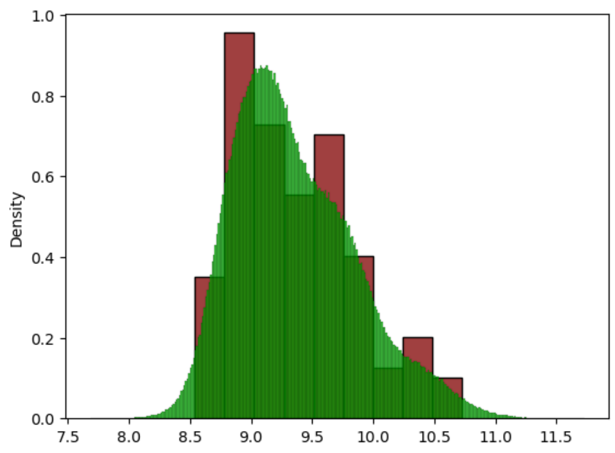
The final model with 5 features and the altered priors converged as indicated by the visual diagnostic for MCMC given below. We see that there are no visible patterns in the trace plots, so the final model converged correctly.

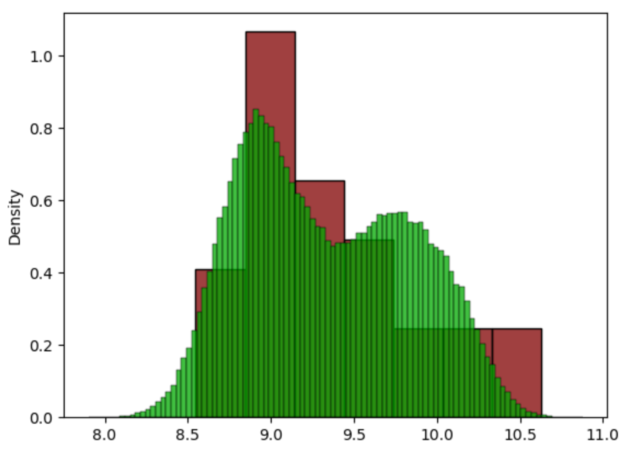


Additionally, the final model had an “r_hat” of 1.0 and had good efficiency as shown in the numeric diagnostic given below.

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
sigma	0.188	0.011	0.168	0.208	0.000	0.000	5549.0	5021.0	1.0
alpha	8.372	0.277	7.846	8.879	0.005	0.004	2953.0	3632.0	1.0
beta[0]	0.594	0.307	0.005	1.154	0.006	0.004	2896.0	3614.0	1.0
beta[1]	0.552	0.122	0.321	0.775	0.002	0.001	4506.0	4509.0	1.0
beta[2]	0.413	0.017	0.381	0.446	0.000	0.000	5079.0	5158.0	1.0
...	...	...	...	...	...	...	...	...	...
pred_y[36]	9.286	0.189	8.934	9.642	0.002	0.002	7080.0	8054.0	1.0
pred_y[37]	9.252	0.188	8.893	9.605	0.002	0.002	7803.0	7896.0	1.0
pred_y[38]	8.823	0.189	8.448	9.157	0.002	0.001	7958.0	7876.0	1.0
pred_y[39]	9.633	0.189	9.276	9.984	0.002	0.001	7960.0	7880.0	1.0
pred_y[40]	9.086	0.188	8.735	9.437	0.002	0.001	8037.0	7908.0	1.0

The marginal checks helped us quickly verify that the generative performance and predictive performance of the final model was similar to the third model. The generative performance is given on the left and the predictive performance is given on the right.





For the generative performance, the final model had a root mean squared error of 0.18116 and an R-squared score of 0.86760. For the predictive performance, the final model had a root mean squared error of 0.17385 and an R-squared score of 0.88861. We see that these results are barely better than the third model for both the generative and predictive performance. This suggests that our method of determining the initial priors was a reasonable way to do this, and that slight alterations won't significantly change the result. Lastly, we see that all 4 of our models performed better than model used in the kaggle analysis.

Discussion

Our analysis showed that a multiple linear regression Stan model can perform equally or even better than the LinearRegression function from scikit-learn. The baseline model that replicated the kaggle analysis by using all possible features performed slightly better than model from the kaggle analysis. We also discovered that using as many features as possible caused the model to overfit the training data. We were able to determine that the most influential features from the dataset for predicting the price of a car were the "fueltype", "enginelocation", "curbweight", "compressionratio", and "power_to_weight_ratio". Using only these features to predict the car price, we improved the predictive performance of the model. Lastly, our prior sensitivity analysis revealed that our initial estimates for the prior parameters were as good assumptions. Altering the mean and standard deviation of α and β didn't impactfully change the results or made the results worse. Changing the prior distribution for σ from a normal to an inverse-gamma, slightly improved the performance, but the difference was negligible.

The main challenge with creating and testing this linear regression model was figuring out which features to use to give the best predictive performance. By narrowing down the features using the estimated β values from Stan, we were able to improve the performance and prevent overfitting. This method of determining the most influential features could be applied to a multiple linear regression model for other datasets. The other biggest challenge was figuring out the best prior parameters to use, but we learned that if we chose reasonable values, the results would be consistent with each other. Based on our findings, choosing a distribution that mimics the posterior distribution was a reliable way to choose a reasonable prior.

There are a few future outlooks that could improve our analysis of this model. First, it would still be beneficial to test out other combinations of features in case our method of choosing features doesn't always lead to the best result. It would also be useful to test out more varied combinations of prior parameters and prior distributions. As mentioned, all our tests had either little changes to the result or a negative impact on the performance, but this doesn’t necessarily mean that we chose the best option. Lastly, it would be advantageous to try out the analysis on a larger dataset. The car dataset only had 205 data points, so we could be more confident in our results if our methodology was shown to work and give good predictive performance on a dataset with more data points.

References

https://www.kaggle.com/datasets/hellbuoy/car-price-prediction

https://www.kaggle.com/code/zabihullah18/car-price-prediction

https://vasishth.github.io/bayescogsci/book/ch-reg.html#sec-pupil