A Multiple Linear Regression Model for Predicting Car Prices

Dylan DeNicola and Natalie Cardoso

Rensselaer Polytechnic Institute

COGS 4210: Cognitive Modeling

**Introduction**

What is the problem setting and the research question? Is there some interesting previous work done on the topic?

**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. Every other remaining feature was used for our baseline model. 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.

For , we restricted its value to be greater than or equal to 0, so we couldn’t get a negative standard deviation for . We also chose 0 for its mean and 10 for its standard deviation because we wanted the standard deviation for to have varying values concentrated towards 0. For and , we chose for their means and 2 for their standard deviations because we wanted to match the distribution of the log transformed price data. For , 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**

**Testing Different Priors**

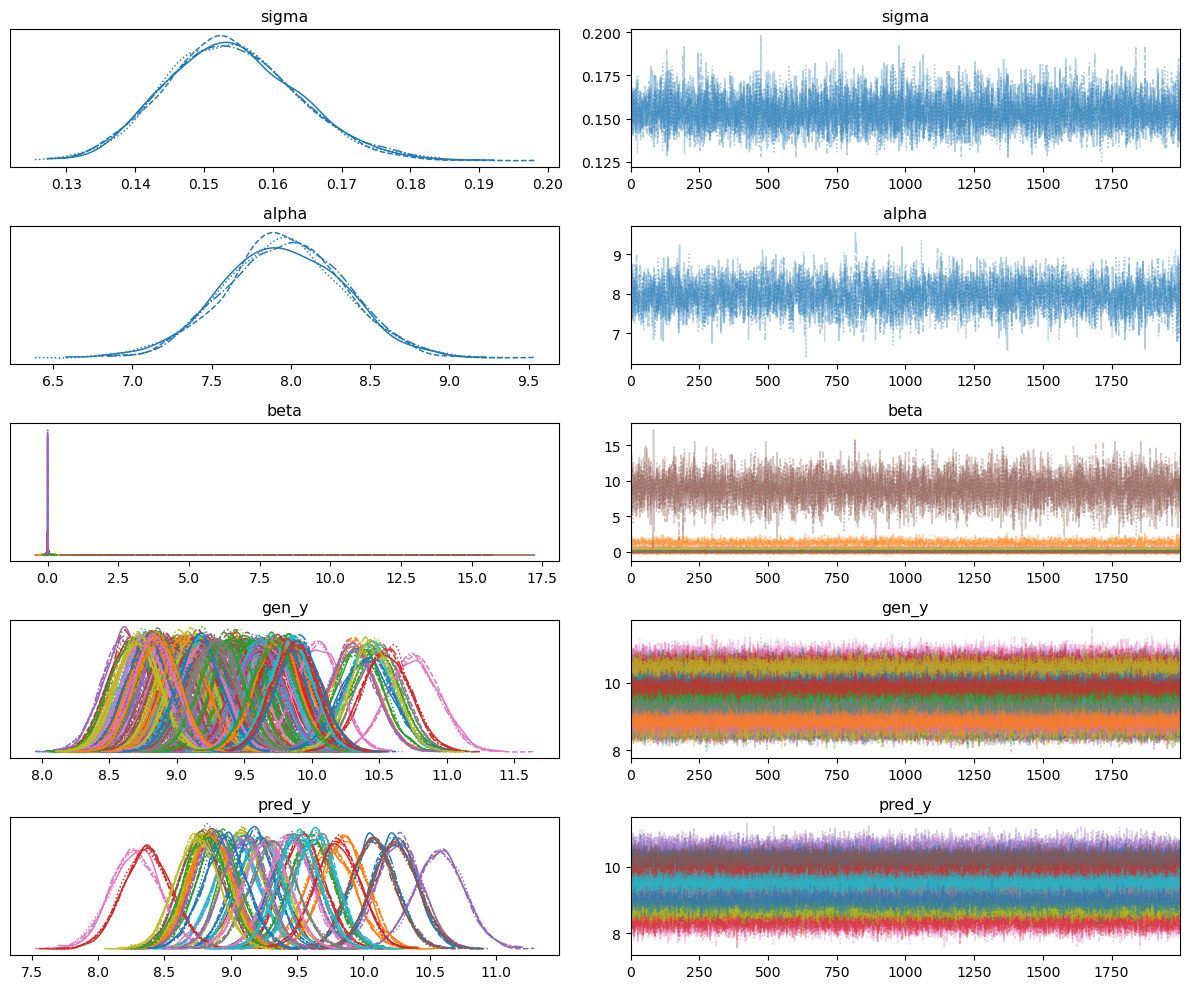
**Assessing Model Fit and Performance**

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 values of each model to check which features had the most influence on the model’s fit.

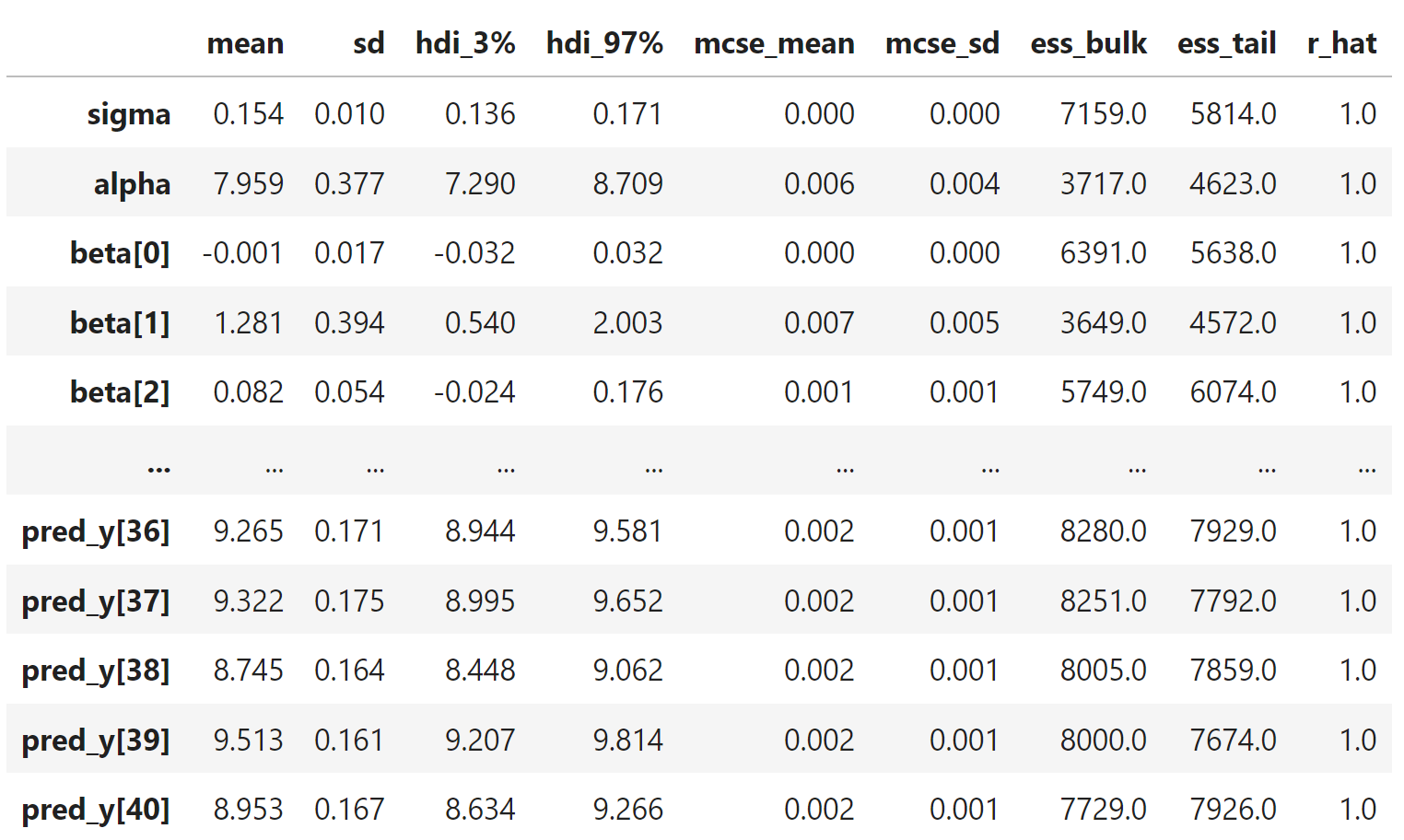
**Results**

**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.



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.

A graph of a number of data

Description automatically generated with medium confidence

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**

**Different Priors Results**

**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 and didn’t impactfully change the results or made the results worse. Changing the prior distribution for 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 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>