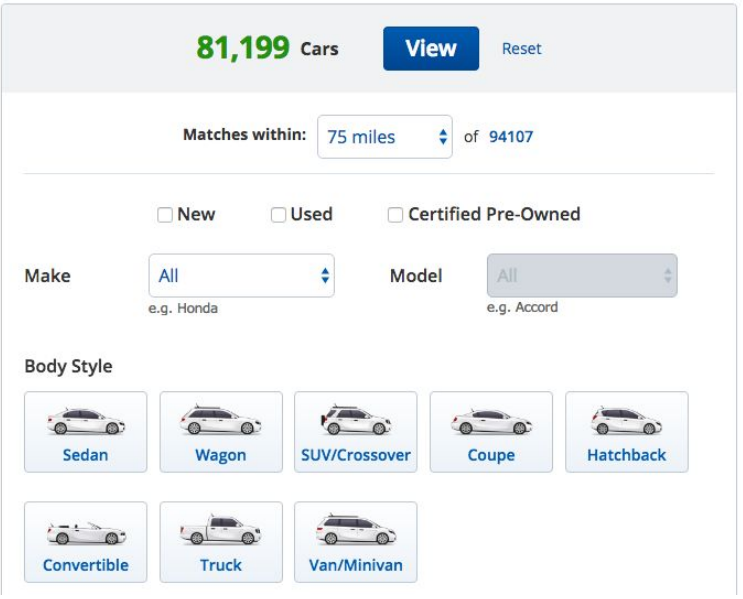
Project Luther: How not to get ripped off on Craig's List

Problem: when buying a car, how do you know whether you are being charged a fair price? In particular, the market for specialised vehicles (which are wheelchair accessible) is particularly poorly served (low supply, through monopoly dealerships). Support for non-specialised vehicles in predicting price is available (e.g. Kelley's Blue Book (see Figure 1) but search functions do not allow the specification of specialised vehicles (Figure 2). (The motivation for this is that I'm looking to buy a car at the moment and will need a specialist vehicle).

Figure 1 (left): Kelley's Blue Book: estimation of a fair price
Figure 2 (right): Kelly's Blue Book Search Function



Solution: build a model to predict price based on known features of the car, and compare this price to the sale price.

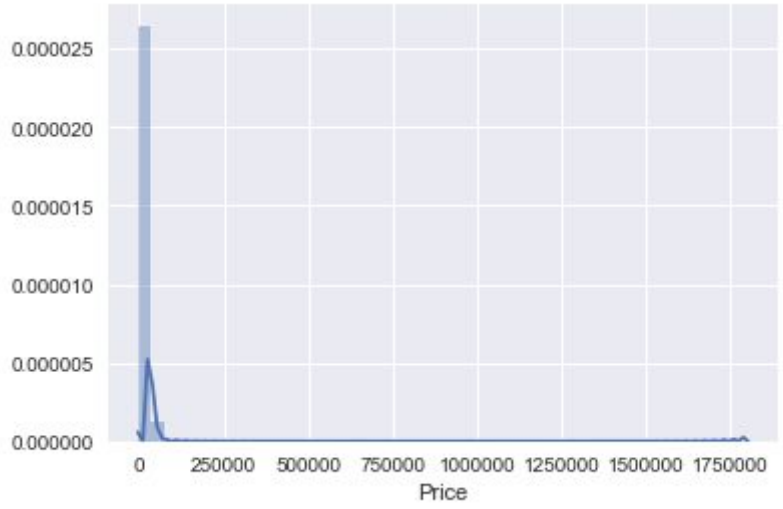
Scraping

- Scraped Craig's List for all geographic locations in the US, looped through all of the listings pages for all of the locations to get URLs for the detailed listings. Looped through each detailed listing (N = c. 80,000).
- Scraped a specialist site, MobilityWorks (N = c. 600)
- Tried scraping Kelley's Blue Book but they only allow you to go up to 10 pages of search items. Tried scraping Ebay but they only let you search up to 49 pages. I did not pursue scraping these websites further because it would have a limited impact on my sample size.
- After data cleaning, I had a sample size of c.39,000.

Data Cleaning

- Combined the datasets and selected features in common
- Clean title status only (no crashed cars)
- Restricted price to $100 - $200,000. The pricing data was very highly skewed (see Graph 1). Having looked at some of the listings at the extremes of the values, they were often input errors or deliberately confusticating the true price. I was also more interested in building an accurate model for reasonably priced cars rather than a less accurate one over the entire distribution.
- Some very old cars were listed as 1900 - it seemed as if this was because the actual date was unknown. I dropped these.
- Dropped vehicles with missing data.

Graph 1: Distribution of Prices of Cars

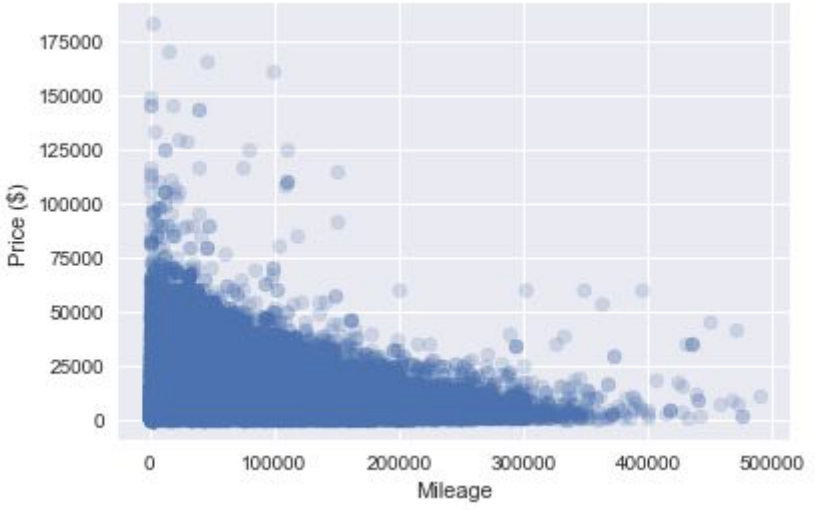


Modelling

I split my data into a 70% test set and 30% training set. The RMSEs reported are on the test set.

Base Model

- I created a base model so that I'd have a "base case" to compare my model to.
- Intuitively, mileage felt like a good place to start in that increasing the distance driven by the vehicle would increase the wear and tear on the vehicle, likely to lead to a depreciation in price.
- The plot of mileage against price also indicated that

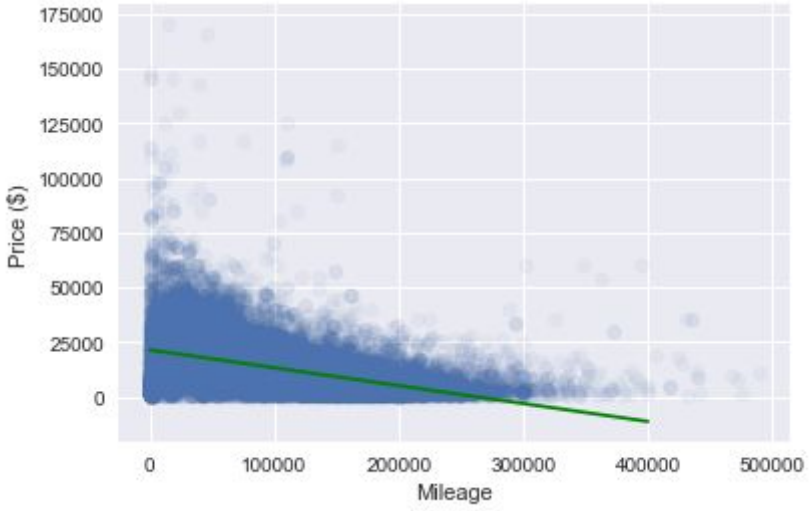


there was a pattern in the relationship (see Graph 2).

Graph 2: Mileage against Price ($)

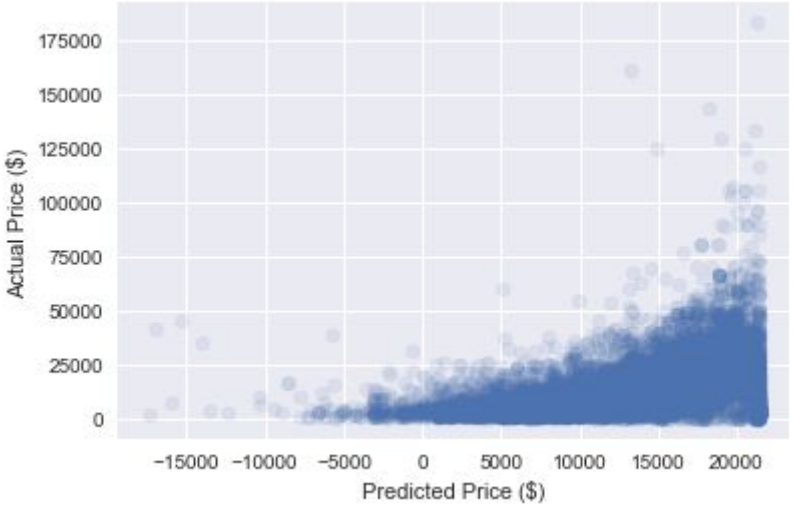
- This gives an intercept of $21,410 and a coefficient on mileage of -0.08. So for every mile, the price decreases by 8 cents. The intercept and negative coefficient means that we don't get any predictions for values above the intercept value, and also get negative price predictions - the seller would be paying you to take it! The RMSE on this model is $10,948. A large error considering that the mean is $12,190!

Graph 3: Base Model of Mileage (green line)

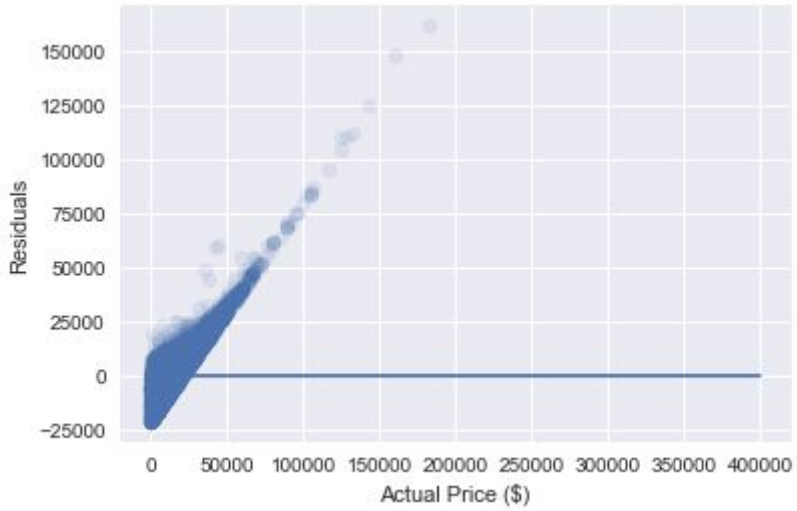


- You can also see that the actual vs predicted price for the base model does not follow well the x = y line expected of a well-fitted model (Graph 4).

Graph 4: Predicted prices plotted against actual prices



Graph 5: Residual plot of base model



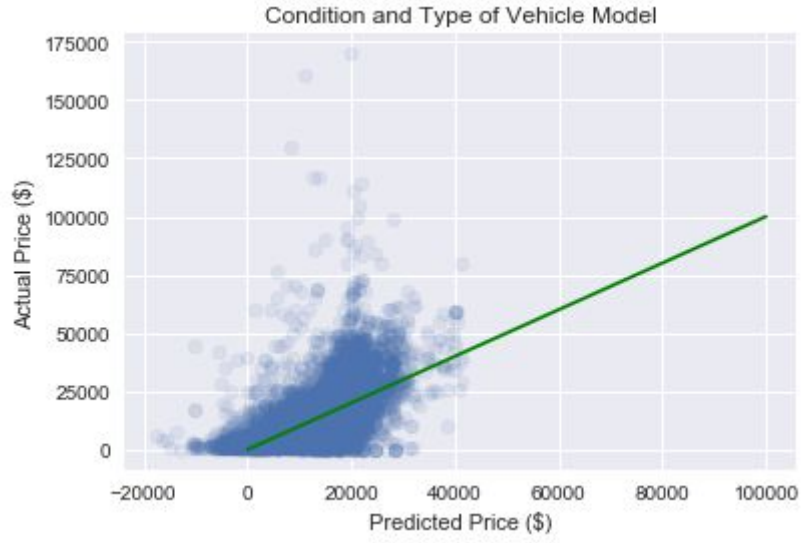
- This model produces strongly heteroskedastic errors. My predictions get worse at price increases. I'm missing signal and have lots of other variables to add in! But I also know that price has a very skewed distribution and this may account for the distribution of residuals.

Model 2

I iteratively add groups of categoricals and see which ones have the most explanatory power.

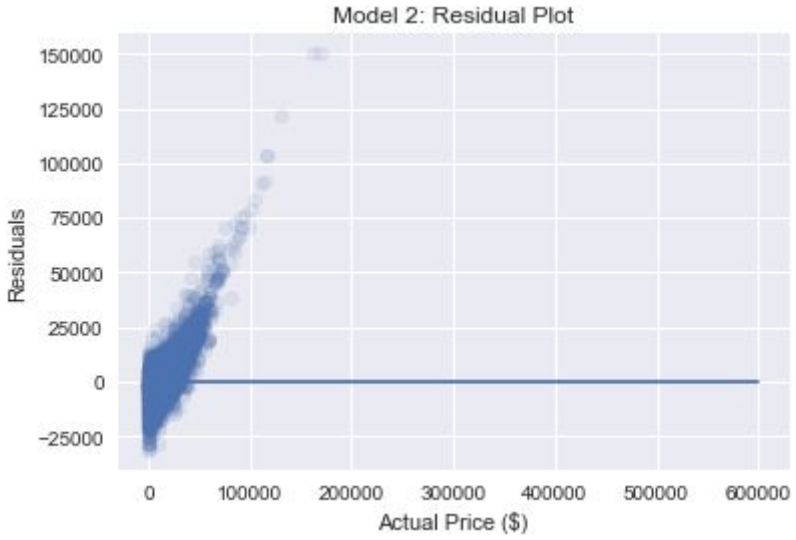
Model 2: mileage and age, with condition and type of vehicle dummies

Graph 6: Predicted prices plotted against actual prices (model 2)



- A slightly better RMSE ($9223) but still plenty of heteroskedasticity (see Graph 7)

Graph 7: Residual plot (Model 2)



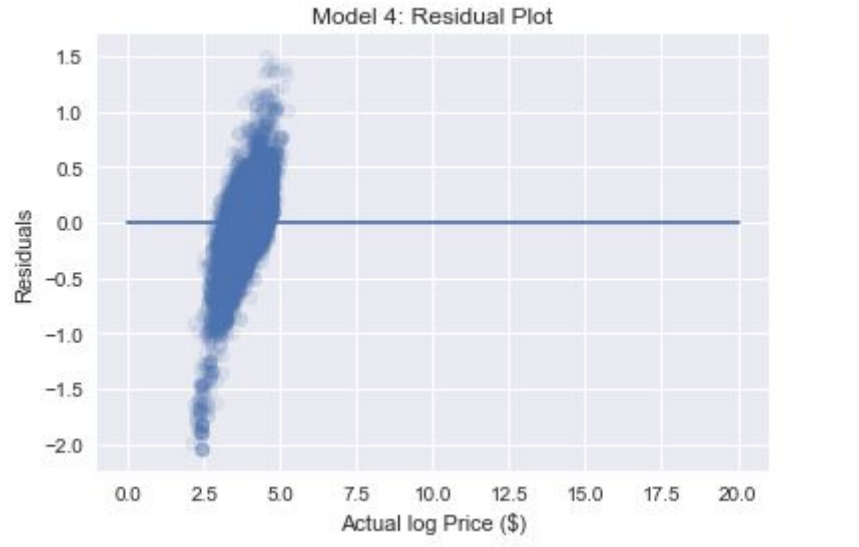
Model 3

- Use all of the data which is shared between the two datasets. This produces a marginally better RMSE but still not great: $8886.

Graph 8: Model with all shared features between datasets (green line)

There was still much heteroskedasticity and so I tried transforming the target variable, price, using a log transformation. But this did not help much (see Graph 9).

Graph 9: Heteroskedastic errors with logged price model



The transformation doesn't reduce the heteroskedasticity much, and the prediction of the transformed price is more difficult to interpret, and less useful (whilst the magnitude of the price is interesting in itself, I really want to be able to tell car buyers how much they should expect to pay).

Model 5 (polynomial model)

So I go back to model 3 (price rather than logged price), and add polynomial and interaction terms. This gave OLS regression with polynomial terms val RMSE: $8042. However, at this point, I have a very sparse dataset and I wonder whether there's the opportunity to get rid of some of the less useful dummy variables. So I use LASSO to help with feature selection. This RMSE of $8880 is higher than the model with all the features shared between the datasets with their polynomial terms without any regularisation ($8042).

I tried values of lambda between 10^2 and 10^6. Regularisation over these values actually increased the RMSE, as can be seen in Graph 10. Values of lambda between 10^-2 and 10^2 did not converge. I think this may be to do with the sparsity of my datasets.

Graph 10: The RMSE on training and test data for regularisation using values of lambda 10^2 - 10^6

Next Steps

- The next step would be to take steps to see what more signal I could derive from the data. From the regularisation failing to reduce the error, I infer that I am currently underfitting and still need to work on reducing the bias in my model. I could do this by using a different algorithm, such as random forest, which would allow me to include features not shared between the datasets. I could also include higher order polynomials.
- I would also set up a pipeline involving cross-validation before testing the model on the test dataset (I simply ran out of time to do this).