DATA1030 Final Report

Burke O'Brien

December 9, 2022

https://github.com/burkeob/Car-Price-Prediction

Introduction

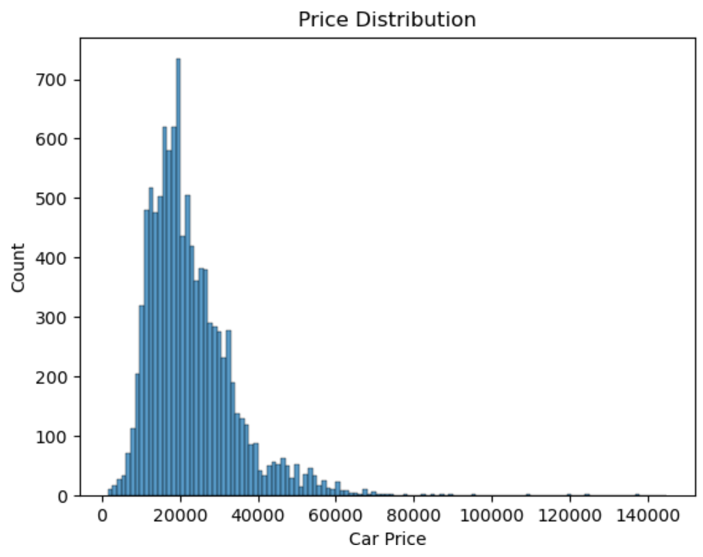
In addition to their cultural and economic importance, cars are a good topic for machine learning because they naturally have numerous features with variation to exploit. The used car market is particularly interesting because it consists of individuals evaluating their assets and setting a price according to their personal assessment of the value of their car, as opposed to a large corporation with more price-setting power and huge teams of analysts setting prices. Furthermore, I am curious whether any features play an outsized role in predicting the price of a car.

To investigate this topic, I found a Kaggle dataset called “100,000 UK Used Car Dataset”, which web-scraped used car listings in the United Kingdom during an unspecified time frame. The data are separated by car make, and to keep things simple, I only use the Audi data; any deployed models will therefore only be helpful in predicting the price of other Audis, but it’s a good start that could easily be expanded to the universe of used cars. Being a Kaggle dataset, there were some useful examples of EDA and modeling to replicate. While I found most of the EDA that others did to be ugly, I did find some interesting graphs and techniques that I will try to recreate. Most of the projects I researched found the best model to be a random forest regressor; one project that predicted the price of Volkswagens got an R2 between the predicted prices and the actual prices of 0.95, and another that used all car makes got an R2 of 0.96. A project that also used only the Audi data got an R2 of 0.96 and a mean absolute error of $1580, which at first glance seems pretty good. I will compare my model's success to these other projects.

The dataset has 10,698 rows and nine features: 1 target (list price), three categorical (model, transmission, fuel type), and five continuous (mileage, registration year, road tax, engine size, and miles per gallon (mpg)). Predicting a continuous target variable makes this a regression problem. While there are undoubtedly other features that go into the price of a car (things like car color, overall condition, optional features, etc.), the features in this data should provide enough variation to make prediction possible. Each feature is well-documented, though I was not familiar with what “road tax” means in this context. It seems that in the UK, drivers pay a one-time tax when registering their vehicle according to how much the car pollutes.

EDA Summary

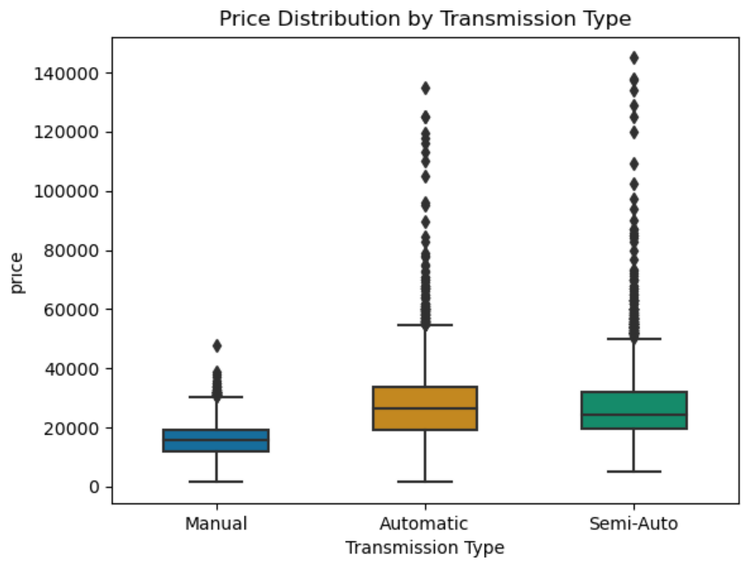
This section is almost identical to that of my midterm report; I did not learn anything about my dataset that would give me cause me to redo this section. I start by investigating the target variable, car price. As we can see in the histogram below, there is a long right tail. While the median price is around $20,200, and 90% of cars have a price of less than about $40,000, some prices are several standard deviations higher, with the most expensive car listed at $145,000.



Price	
Count	10,668
Mean	$22,896.68
Std	$11,714.84
Min	$1,490
25%	$15,130.75
50%	$40,200
75%	$27,990
Max	$145,000

Figure 1: Target variable distribution

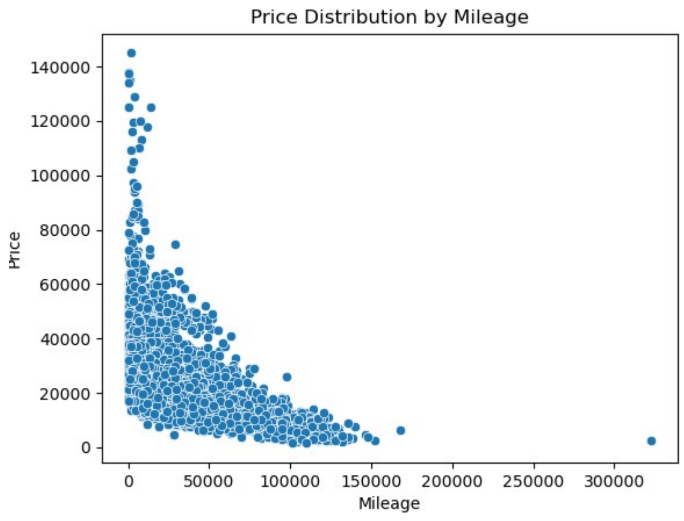
I performed similar EDA on my predicting features. For each feature, I checked for missing values, the general distribution in the form of a histogram or box plot, and how each feature varies by price. Two features worth mentioning are transmission and mileage.



Frequency	
Manual	.41
Semi-Auto	.34
Automatic	.25

Figure 2: Price distribution by vehicle transmission

I found the transmission type to be interesting for two reasons. First, it seems as though it could be a good predictor for the price, where the prices of cars with manual transmission are lower than cars with automatic or semi-auto transmissions. I also thought the relative frequency of transmission type was interesting, as I would have expected many fewer cars in the UK to be automatic. I also found the following relationship between price and mileage to be interesting.



Mileage	
Count	10,668
Mean	24,827.24
Std	23,505,26
Min	1
25%	5,968.75
50%	19,000
75%	36,464,5
Max	323,000

Figure 3: Price distribution by vehicle mileage

There are two things worth noting here. First, there are a couple of severe outliers. While the average car has around 25,000 miles, many cars have a mileage of several standard deviations higher than this. Second, while we can see that price tends to decrease with more mileage, the relationship does not appear to be linear. I therefore do some feature engineering to better capture this relationship.

Methods

Having conducted sufficient EDA, I'm prepared to fit and interpret some models. I decided to use mean absolute percent error (MAPE) as my evaluation metric; I realize that this is not the standard metric we use in class, but I wanted my preferred model to perform about as well on both less expensive cars and more expensive cars, and I was worried that this would not be the case if I used something like mean average error (MAE). To begin, I create two new variables, log mileage and log miles per gallon in an attempt to capture the apparent non-linear relationship between these features and price. I then separate 10% of my data as a test set- we will use these data solely to report the model's test score. With more than 10,000 observations my dataset is large enough to have 90% of the data used to train the model.

The rest of the ML pipeline is facilitated by scikit-learn's GridSearch. There are three steps in the pipeline. As detailed in the midterm report, for preprocessing I apply a OneHotEncoder to my categorical features and MinMax and StandardScalers to my continuous ones. After doing so, I end up with 39 features. I then apply another StandardScaler to everything so that I can interpret the magnitude of the coefficients in my linear models. Finally, I fit the model on the transformed dataset, using four KFolds as cross-validation. This cross-validation method is different from what I originally proposed, but I could not figure out how to use GridSearch with hold-out validation, and KFold is more robust anyway. I then record the model whose parameters produced the best average MAPE score across the 4 KFolds. I repeat this process across several states to capture randomness from splitting and non-deterministic models. Thus, for each model, I report the average test score across several different random states, along with the standard deviation of these different test scores.

I ended up trying all seven models we discussed in class: Lasso, Ridge, Elastic Net, KNearestNeighbor, RandomForest, Support Vector Machine, and XGBoost. Having made a general ML pipeline, it wasn't too much work to try each model. However, to facilitate early stopping with the XGBoost models, I do the cross-validation in a loop instead of having GridSearch do it for me; otherwise, the process is the exact same. I detail the hyperparameters tuned in the table below, all of which are fairly standard. As I mention in the outlook section, given more time I would tune more hyperparameters and over more extensive ranges.

Model	**Hyperparameters Tuned**
Lasso	Alpha: np.logspace(-6,0,21)
Ridge	Alpha: np.logspace(-6,0,21)
Elastic Net	Alpha: np.logspace(-6,0,21) L1 Ratio: np.logspace(-3,0,21)
KNearestNeighbor	N Neighbors: [1,3,10,15,20,30], Model Weights: ['uniform', 'distance']
RandomForestRegressor	N estimators: [1, 3, 10, 30], Max depth: [1, 3, 7, 10, 30]
Support Vector Machine	C: (1/ (np.logspace(-7,0,11))), Gamma: [1e-3, 1e-1, 1e1, 1e3, 1e5]
XGBoost	Learning Rate = np.linspace(0,1, 10) Depth = [1,3,8,10,15]

Results

I will compare my models to the baseline MAPE score from predicting the average price in the training set for every car in the test set. I compute this baseline MAPE to be .366; that is, if we were to predict the average price of the training set for every car in the test set, on average we would be 36.6% incorrect. Figure 4 below summarizes my results. We can see that every model has an average test score well below the baseline MAPE, which means that we are always able to somewhat predict the price of the car. The best model was XGBoost, which had an average MAPE score of .067. With a standard deviation of only .001, this puts the XGBoost model 299(!) standard deviations above the baseline model. My best XGBoost model has an *R2* of .93, which is slightly worse than similar models using my Kaggle dataset.

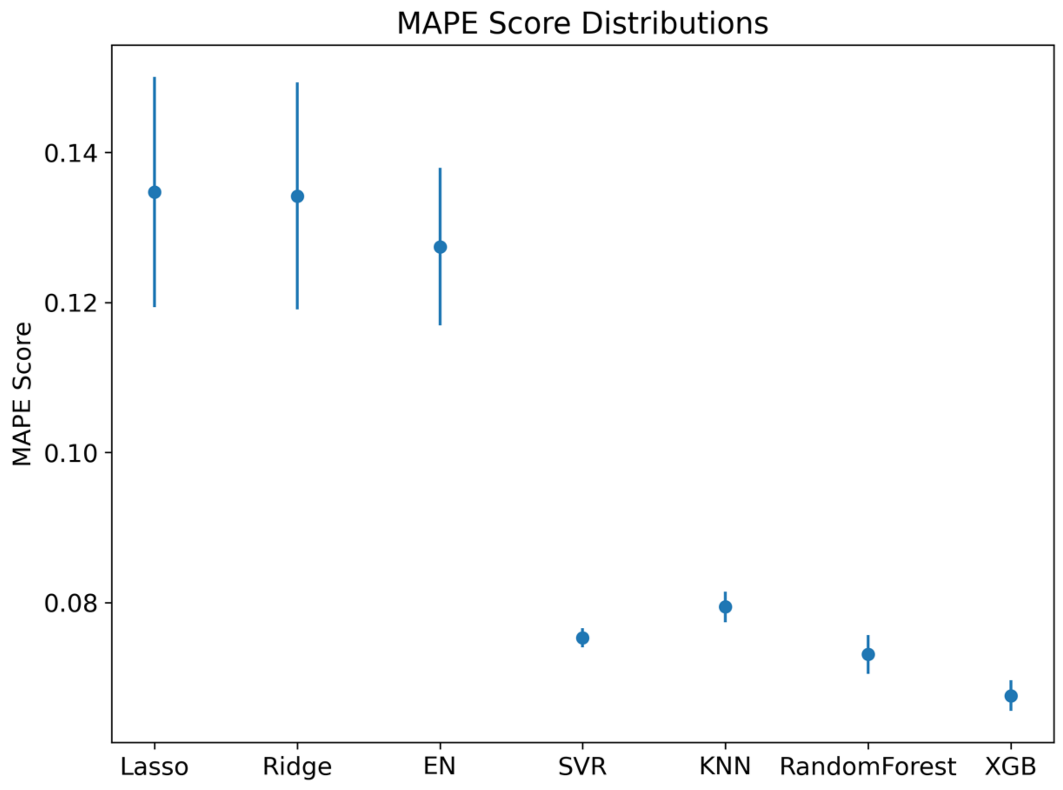


Figure 4: The non-linear models generally performed better than the linear ones

Having identified the best model, we can do additional investigation to uncover what is driving the price predictions. As the best model is XGBoost, several global feature importance metrics are at our disposal. I chose to report the average SHAP score, feature permutation importance, and XGBoost's weight metric. I chose these because I think they're the most intuitive, and the average SHAP score and feature permutation importance are model agnostic, so we can compare results to other models. I get similar results for other metrics and my other models, which is a good sign.

The ten features with the highest average SHAP value are displayed below, where each point is the local SHAP value for a car in the test set. As described in this blog post, SHAP values represent the "effort" each feature contributed toward lowering the error of a local prediction; features with higher SHAP values, or effort, contributed more to the price prediction. In this way, we can see that the year, miles per gallon, and engine size were most important in predicting price.

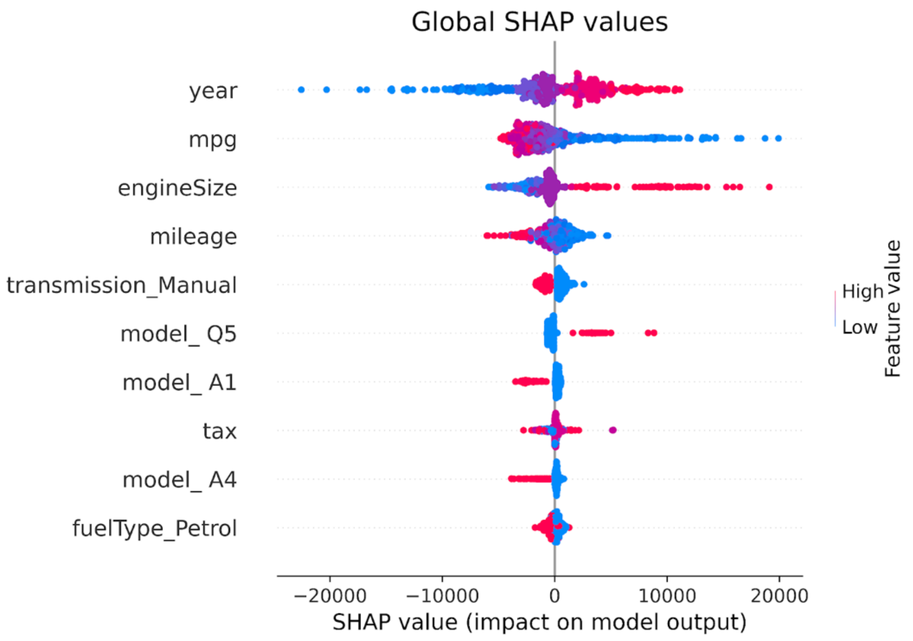


Figure 5: Some features have very high average SHAP values

Figures 6 and 7 show that we get similar features as the most important when using feature permutation and XGBoost’s weight metric. This is a good indication that these features are very important in predicting price.

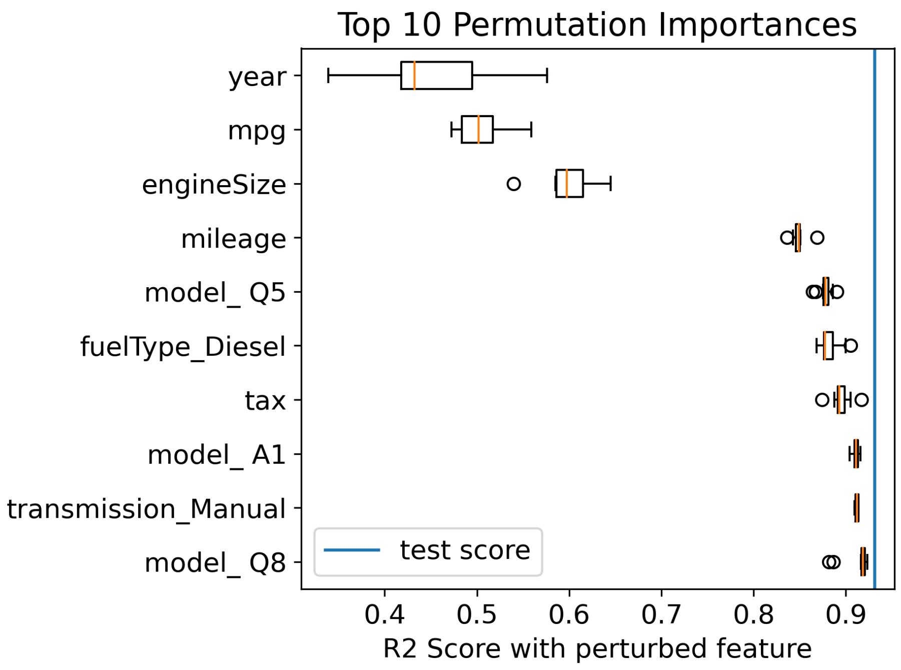


Figure 6: We again see year, mpg, and engine size as the most important features.

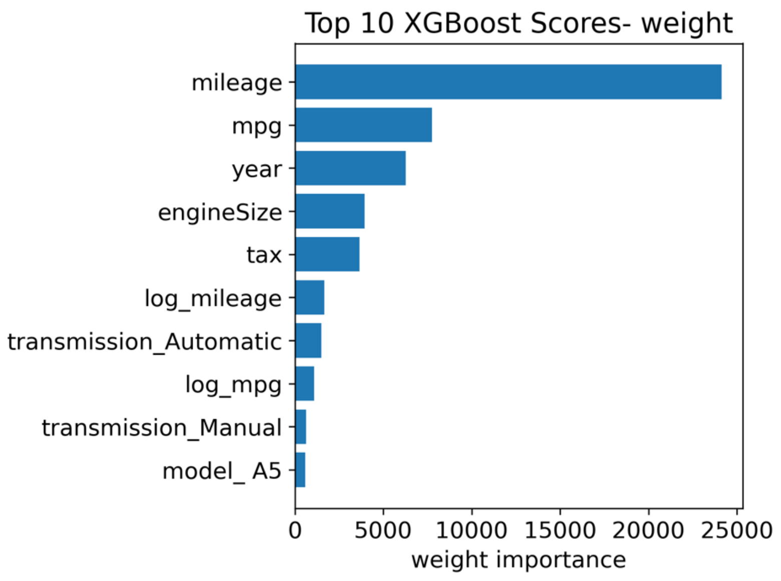


Figure 6: XGBoost's weight metric identifies mileage as the most important feature

We can analyze the SHAP values for a single car to ascertain local feature importance; that is, for a given car, which features were most important for determining the predicted price.

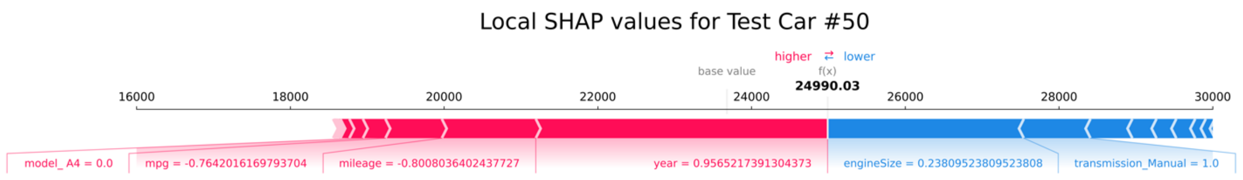


Figure 7: Being a newer car with low mileage makes this car more expensive than average

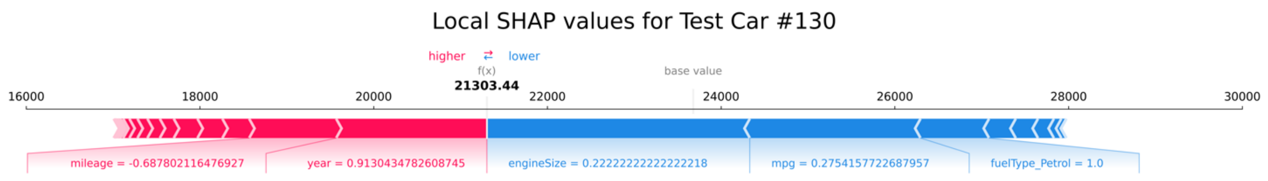


Figure 8: Higher engine size and miles per gallon lower the price of this car

I was not terribly surprised by these findings; the year a car was made, its mileage, and miles per gallon each make sense as important price determinants. I was perhaps a little surprised that specific Audi models were not more important in determining price. However, it’s likely that this information is also “baked into” the other car features. I also think it’s funny that higher mpg *decreases* the price of an Audi- people value less efficient Audis! In short, these findings give me more confidence in the validity of my models.

Outlook

Overall, machine learning models can predict a car's price much better than a baseline model (in this dataset, at least). However, there are areas that could be improved with additional time and data. One such area is feature engineering; I have a hunch that there is more we could do here to uncover relationships between the car features and, in doing so, come up with better predictions. Additionally, this analysis was conducted on only a single kind of car, Audis. Audis are a luxury brand, and the relationship I model between Audi features and price may differ for other types of cars. Expanding my models to include other car models would not be too much work. I’m also curious whether I would get similar results if I used MAE as my evaluation metric rather than MAPE. Given more time, I would run the entire process with MAE as the evaluation metric to see if I get similar results.

Lastly, several unanswered questions about this dataset make real-world deployment difficult. Additional data on the car listing, like posting date and location, are likely important in determining the price of the car, but not present in the current dataset. We are also likely to encounter missing data in the real world, and thus my pipeline would need processes to handle missing data.

References

1. Aditya, 2021, “100,000 UK Used Car Data set” Version 3, retrieved October 19 2023, link
2. Gireesh Sundaram, 2020 “Volkswagen Price Regression (R^2 = 0.9555)”, retrieved October 19 2023, link
3. Kuo, “Explain Your Model with the SHAP Values”, retrieved December 5 2023, link
4. Mahmoud Hasasn Mahmoud, 2021, “Audi Car price Regression with accuracy 97%”, retrieved October 19 2023, link
5. Barath, “Audi Car price prediction with (R^2 - 0.97)”, retrieved October 19 2023, link