Used Cars: What to look out for?

Tara Mooney and Corbin Lubianski

Introduction

Cars are a central part of American life. Almost 90% of American households own at least one car, over 75% of commuters take these cars to work each day, and Americans as a whole drive around 3.2 billion miles each year. Given their necessity, cars are some of the most important purchases that Americans make throughout their lifetimes. Interestingly, the vast majority of these cars are not bought directly from the companies who make them: around 75% of cars purchased in the United States are previously owned.

From the oft-referenced "Market for Lemons," to its present use as an economic indicator, the used car market has been the subject of public and academic scrutiny for decades. Recent years have seen this market transformed by the introduction of online retail— not only in the purchases made, but in the data now publicly available. Despite the vast amount of information about this market, both buyers and sellers of used cars struggle to understand how they are priced.

In this paper, we explore how used cars are priced given their history and attributes. We hope to answer the question of the best way to predict the price of a used car given the modelling tools we currently have available, and determine which elements of a vehicle are the most explanatory for its price in the used car market.

Data Description

The data used in this analysis comes from a Kaggle dataset of "U.S. Used Cars." The data was collected using a user-made web crawler on the used car inventory of automotive shopping website CarGurus.[1] The original dataset contains 66 unique columns with 146298 observations.[2]

To keep our analysis tractable, we removed features with a significant amount of missing data, and features that are, by design, highly correlated with others. For example, the front and back legroom in a car are both strongly related to the overall length of the car's body, so we removed these variables and kept only length.

During our EDA, we found that, like most financial data, our response variable price was strongly right-skewed. Using a logarithm transformation with base 2, this variable gained a Normal-like distribution. Upon investigating other predictors, we also found that a logarithm transformation with base 2 also improved the symmetry of the variable horsepower, so we work with this variable in its transformed state, as well.

Before building our models, we decided to keep only the complete cases of our remaining data. After confirming our transformations were still valid, this brought us to a total of 55020 observations for analysis.

From here, we combined the highly correlated predictors of city and highway fuel economy into one average metric, and began our analysis including the following predictors from our dataset:

- `age`: Age of the car from its model year (ex. 2020, 2021) subtracted from 2021 since cars are normally released a year prior for marketing since data is from September 2020. Example: a 2019 model has an age of 2.

[1] We attempted to webcrawl ourselves for freedom to collect certain attributes, but ultimately favored the Kaggle dataset. For assistance with resources on webcrawling, we would like to thank fellow classmate Dora Ivkovich.

[2] Note that if you choose to download the dataset, the webpage has two downloadable files. Click on "download" on the top right which is 2 GB. The other dataset is far too large and was not choosen for optimizing run time and performance.

- `avg_fuel_economy`: Fuel economy averaged in city and highway traffic in miles per gallon (after transformation, originally in km/liter)
- `body_type`: Body Type of the vehicle. Categories include Convertible, Coupe, Hatchback, Minivan, Pickup Truck, Sedan, SUV/Crossover, Van, and Wagon.
- `daysonmarket`: number of days since the vehicle was first listed on the website.
- `engine_displacement`: The measure of the cylinder volume swept by all of the pistons of a piston engine, excluding the combustion chambers.
- `engine_type`: The engine's configuration. Ex: H4, V6, I2.
- `fleet`: Whether the vehicle was previously part of a fleet previously.
- `franchise_dealer`: Whether the dealer is a franchise dealer.
- `fuel_tank_volume`: fuel tank's filling capacity in gallons.
- `fuel_type`: Dominant type of fuel ingested by the vehicle. Categories include Biodiesel, Compressed Natural Gas, Diesel, Electric, Flex Fuel, Gasoline, and Hybrid.
- `has_accidents`: if True then vehicle has registered accidents
- `height, length, and width`: three separate predictors of each metric in inches.
- `is_new`: If True means the vehicle was launched less than 2 years ago.
- `listing_color`: Dominant color group from the exterior color.
- `log_horsepower`: Horsepower is the unit for the power produced by an engine. Logarithm transformed with a base of 2.
- `luxury`: if True then the vehicle's manufacturer is considered "luxury". Determination made by judgement from resources and opinions online.
- `make_name`: The manufacturer of the vehicle. Ex: Toyota, BMW, smart.
- `maximum_seating`: The maximum number of people who can fit in the vehicle
- `mileage`: Total number of miles traveled or covered by the vehicle.
- `owner_count`: how many registered owners the vehicle has had prior
- `rpm`: Revolutions per minute of the internal combustion engine
- `torque`: The torque, or "maximum twisting force," of the vehicle's engine. Measured in Newton Meters.
- `transmission`: the mechanism by which power is transmitted from an engine to the wheels of a motor vehicle. Categories include A (Automatic), CVT (Continuously Variable Automatic), Dual, Clutch, and M (Manual).
- `wheel_system`: The wheel system of the vehicle. Categories include Four-Wheel Drive, Two-Wheel Drive, All-Wheel Drive, Front-Wheel Drive, and Rear-Wheel Drive.

Methods

Sequential Variable Selection

To begin our analysis, we performed sequential variable selection procedures to determine which of our predictors were relevant to our response variable of log price. Additionally, we were interested in seeing if these procedures returned different overall models. For splitting our dataset into a training and testing set, we opted for a 70%-30% split respectively to favor a larger training data set to improve the test training set.

To begin, we fit three simple linear regression models: one intercept-only, one with all main effects, and one "baseline" model that we theorized to have the likeliest chance of capturing the most predictive power for price. Again, ideally we would've fit a complete interaction effects model to use in this analysis, but the large size of our dataset (and the computing power of laptops) made this infeasible in the time we had. From here, we completed backwards variable selection procedures using the main effects. Then, we did the same thing using forward variable selection starting with the intercept-only model and using the full main effects model as the upper scope. Lastly, we did step-wise variable selection using the full main effects model as the upper scope where the model started at our theorized baseline model. In total, we fit six models using sequential variable selection.

Interestingly, each of our variable selection procedures returned the same model. That is, backwards, forwards, and step-wise variable selection returned the same set of predictors in the cases where we used the same initial models:

Response	Predictors
log(price, 2)	mileage, log_horsepower, transmission
	body_type, fuel_type, avg_fuel_economy, age, is_new
	make_name, torque, franchise_dealer, wheel_system, engine_type
	width, fleet, height, has_accidents, owner_count, engine_displacement
	maximum_seating, listing_color, daysonmarket, fuel_tank_volume, rpm

Note the predictors luxury and length were excluded from the sequential models. Upon investigating the summary of the initial models, it was found that length is highly correlated with other predictors like width and height such that length's predictive power was shared in width and height. The exclusion of the created predictor luxury was surprising, considering that with the Welch's two-sample t-test, the inclusion of luxury in the model found to share significant prediction power confirming the alternative hypothesis that luxury used cars share different pricing distributions with non-luxury used cars ($T = -33.819, p < 2.2e^{-16}$).

Before our analysis, we hypothesized that mileage would be the predictor with the greatest effect on log price. The main effects model returned by our step procedures reports that, when holding other predictors constant, mileage has the most extreme t-statistic, and therefore the most extreme p-value, though R does not report this directly. Our beta coefficient for mileage in this model is $-6.949 \times e^{-6}$, meaning that, keeping other predictors constant, for every additional mile driven on a car, the log price of the car decreases by 6.949×10^{-6}, and the untransformed price decreases by approximately \$1.

Decision Trees and Random Forest

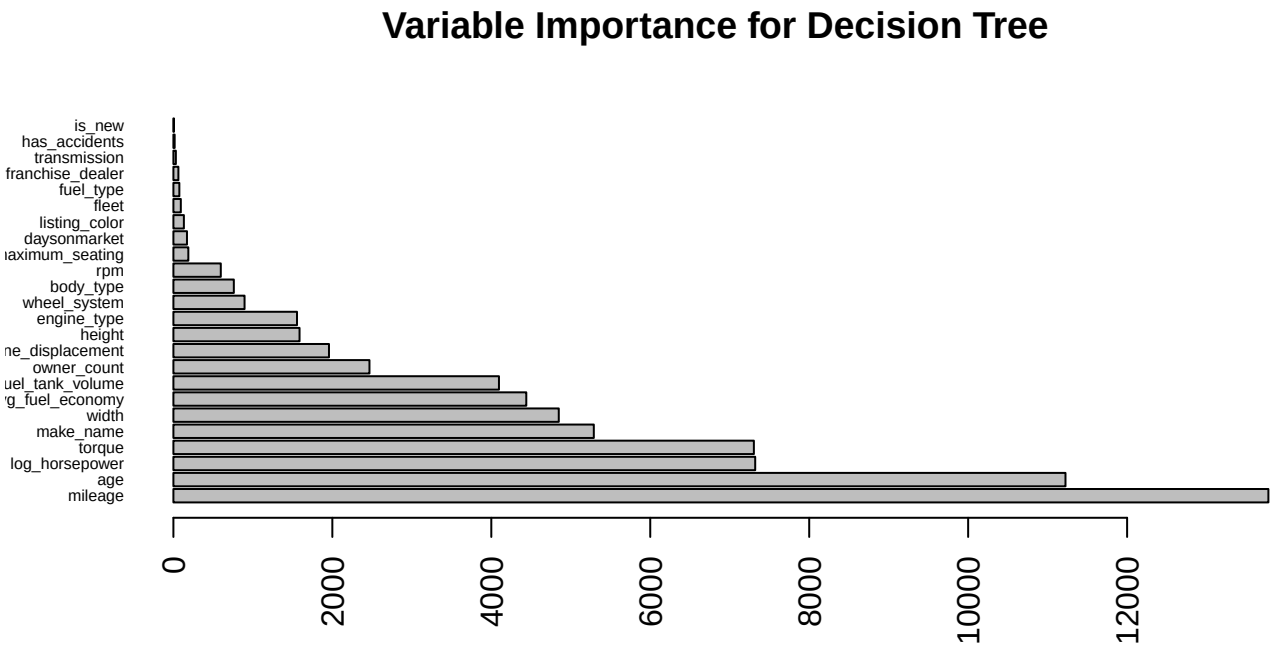
To continue our analysis from this initial fitted model given by the sequential variable selection procedures, we wanted to see whether using tools such as decision trees and random forests improved predictive accuracy (by RMSE), but also what insights these types of models provide to the most important predictors in our model.

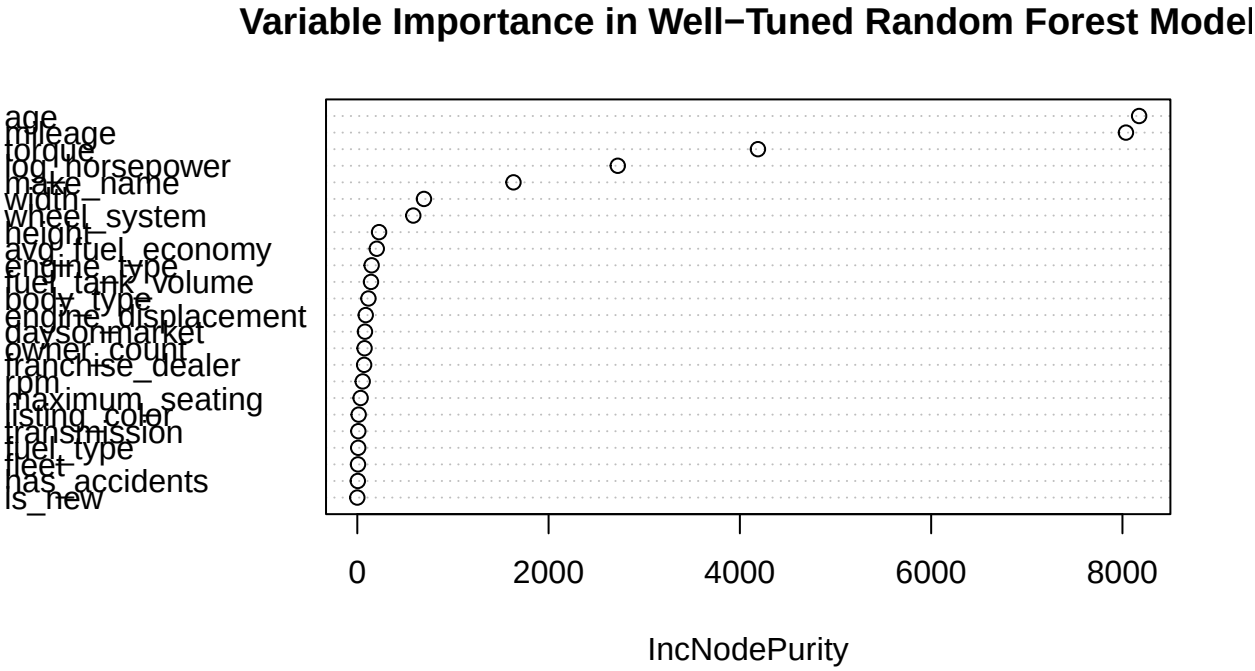
We began by fitting a well-pruned decision tree of our model. To do this, we used the formula returned by the sequential variable selection models. Decision trees in R cannot handle interaction effects, so we used the model with only main effects to do this. We kept this same approach when fitting the random forest model for consistency. Another important caveat to make here is that we fit our random forest uaing our training data set which we will compare it to the test set and entire data set. Through our tuning, we found that a random forest model that sampled 15 predictors and included 1000 leaf nodes was the best model by RMSE on average.

While it is typically the case that random forests perform better than decision trees, it is likely that our need to sample to fit the random forest affected our prediction accuracy. Nonetheless, our primary reason for fitting these models was to get a sense of the most important predictors in our model, which we are still able to do. Decision trees and random forests have the benefit of clear reporting of "variable importance." In decision trees, variable importance measures the level of improvement in our error metric (SSE or sum of squared estimate of error) every time a variable is used to define a split. In random forests, the default in R for variable importance is the sum of improvements in mean squared error each time a variable is involved in a split. Generally, "variable importance" gives us a measure of how much including a predictor in a tree improves error.

The plots below show the reported variable importance given these two approaches. Both of these models identify mileage and age as our most important predictors for the price of a used car — these make sense as both are a proxy for just how much a car has been "used." These are followed by torque and log horsepower, though their relative importance is flipped in the two models. This also makes sense, given these are both measures of the car's performance. Interestingly, they seem to be closer in importance in the decision tree

than the random forest. Both models also identify the manufacturer as the 5th most important predictor. It does seem that these are the metrics the "average car buyer" (someone who is buying the car for utilitarian reasons and is not necessarily doing much research) would care about, so it makes sense they are most indicative of price. Both the decision tree and random forest models identify whether a vehicle is new, has accidents, and its transmission type as the least important predictor variables.





Random Intercepts and Random Slopes

It's reasonable to assume that different makes' prices respond differently to their mileage. We can adapt this into a model by using Mixed Effects modeling such that we condition our intercepts and slopes based on the make of the used car. Before we dive into making the models, we should first justify a mixed effects model using make name on price. We can do this by conducting an analysis of variance test that determines whether using make names as a random intercepts component will add any explanatory power. Remarkably, make names do come explanatory power on the model so we should construct mixed effects models with random intercepts ($F = 691.9, p < 2e^{-16}$). However, we theorized earlier than different makes have various relationships with mileage such that one makes mileage will affect its pricing than other makes. We can do a similar analysis of variance test which showcases that we gain significant explanatory power with the inclusion of make names having random intercepts and random mileage slopes. For the models, we are using the formulas from the sequential model.

While we know that doing mixed effects modeling with random slopes with regards to mileage is justifiable, we do not know if the explanatory power from the analysis of variance test could be otherwise explained through other predictors in the model. We can determine whether the mixed effects model with random intercepts with regards to makes has any explanatory power difference than the mixed effects model with random intercepts and random mileage slopes by conducting another analysis of variance test. Observe below on the comparative statistics:

Statistic	Random Intercepts	Random Intercepts and Slopes
Predictors	87	89
AIC	8498.1	5466.2
BIC	9273.7	6259.7
Log Likelihood	-4162.0	-2644.1
Deviance	8324.1	5288.2
Chi-sq Statistic	–	3035.9
Degrees of Freedom	–	2
p-value	–	$2.2e^{-16}$

Notice that with the inclusion of two additional predictors with the random intercepts and slopes mixed effects model, the AIC and BIC did not increase as would be expected if the random slopes had no explanatory power since the penalty factor would increase the statistic. Therefore, we have good authority to use the random intercepts and slopes model. Lastly, we can justify the use of mixed effects model if we calculate a moderately strong intra-class correlation (ICC) between the make names. The intra-class correlation of 0.4837459 is a relatively strong correlation value which suggests that make names are quite a large source of variability in our response variable price.

Results

With all of our models constructed, observe the table below that displays the train, test, and overall RMSE for the models.

We want to get the best model with the lowest test RMSE while maintaining a small gap between train and test. The model with the biggest RMSE gap between train and test is the Decision Tree with a RMSE gap of 3598.47, yet it has the lowest test RMSE. This suggests that while the Decision Tree is overfitted, the relative simplicity of the model fits the test data set well. The baseline linear model RMSE are relatively high which suggests that our theorized best-fit model was far from ideal. It is not surprising that the intercept-only linear model is poorly since it can only use the sample mean of price to minimize the mean squared error by definition. While we already noted that the forward, backward, and step sequential models were identical, note how similar the RMSE for train, test, and whole for the sequential models and the all-predictors linear model. This suggests that our sequential models were very close to including all predictors (all but two) which also indicates that it preserved AIC and BIC since the other two predictors marginally improved

Models	RMSE.train	RMSE.test	RMSE.whole
Baseline Linear	8250.46	10569.37	9009.02
Intercept-Only Linear	14556.22	16101.06	15036.34
All-Predictors Linear	4971.94	7965.59	6028.21
Forward Sequential	4972.00	7965.67	6028.28
Backward Sequential	4972.00	7965.67	6028.28
Forward-Backward Sequential	4972.00	7965.67	6028.28
Decision Tree	3506.09	7104.56	4873.12
Random Forest	4336.52	7360.23	4981.36
Ridge	5236.01	8342.89	6318.63
LASSO	4973.03	7979.05	6036.34
Random Intercepts	5038.12	7982.84	6073.35
Random Intercepts, Random Slopes	4806.02	7793.95	5864.49

RMSE. With regards to the random forest model, we reduce the overfitting considerably, but still increase the test RMSE substantially. We expect for ridge and LASSO to have higher RMSE since their estimates are biased in hopes of reducing the slope coefficient estimates' variance. Lastly, the similarity between the mixed effects models' RMSE to the sequential models is surprising as the mixed effects models' slope coefficient estimates are biased by definition. This suggests that controlling mileage slopes and intercepts by make names allowed to break correlation between makes while maintaining much of the accuracy.

Conclusion

Among the many attributes that are supplied while trying to buy a used car, the fact that models with more predictors achieved lower RMSE indicates that buyers should be paying more attention to other characteristics than merely the mileage and make. However, we found that the most significant used car characteristics are the age, mileage, torque, horsepower, and make. How much you should weight that is not clear as our Decision Tree and Random Forest models had different rankings for age and mileage. A clear result is that someone should be asking for as much information as possible when buying used cars as RMSE generally decreased when more predictors were added even considering penalty costs as depicted in the AIC and BIC statistics.

Moving forward, it would be interesting to determine how these relationships have changed from September 2020 which was the date when the Kaggle dataset was collected considering how used car markets have changed over the last few years during the pandemic and the subsequent economic recovery. Regardless, much is to be learned from exploring used car data as people continue to harvest it for predicts and to maximize economic payoff.

Appendix

Data Wrangling

Removal Justification	Variable Names
Mostly Empty	bed, bed_length, bed_height, combined_fuel_economy, cabin, is_certified
Unsuitable for Regression	main_picture_url, major_options, savings_amount, listing_id, trim_id, sp_id, seller_rating, sp_none
Only contains NA and FALSE	frame_data, vehicle_damage_category, is_Cab, frame_data, salvage, theft_title
Only contains NA and TRUE	is_oempco, is_cpo
Duplicate of another variable	engine_cylinders, franchise_make, transmission_display
Not factorizable (too many differences)	exterior_color, interior_color, model_name, trim_name
Edited or Transformed	horsepower, year, listed_date, power, city_fuel_economy, highway_fuel_economy

```
# libraries
library(tidyverse)
library(rpart)
library(randomForest)
library(stringr)
library(glmnet)
library(lme4)
library(dplyr)
library(knitr)
library(xtable)
library(DT)
```

```
# Read dataset from https://www.kaggle.com/datasets/ananaymital/us-used-cars-dataset
used_cars = read.csv("data/vehicles_cargurus.csv")

# Removing redundant predictors (see Appendix for rationale)
used_cars = subset(used_cars, select = -c(bed, bed_height, bed_length,
                              combine_fuel_economy, cabin, engine_cylinders,
                              is_certified, is_cpo, is_oemcpo, interior_color,
                              frame_damaged, isCab, salvage, theft_title,
                              main_picture_url, major_options, wheelbase,
                              savings_amount, seller_rating, sp_id, sp_name,
                              vehicle_damage_category, listing_id, trimId, vin,
                              dealer_zip, exterior_color, franchise_make, latitude,
                              longitude, transmission_display, wheel_system_display,
                              model_name, front_legroom, back_legroom, trim_name, city))

# There are blanks and empty spaces
used_cars[used_cars == "" | used_cars == " "] = NA
```

```r
# Change string to numeric while removing units
used_cars$fuel_tank_volume = as.numeric(str_extract_all(used_cars$fuel_tank_volume,
                                                        "\\(?[0-9,.]+\\)?"))
used_cars$height = as.numeric(str_extract_all(used_cars$height,
                                              "\\(?[0-9,.]+\\)?"))
used_cars$length = as.numeric(str_extract_all(used_cars$length,
                                              "\\(?[0-9,.]+\\)?"))
used_cars$maximum_seating = as.numeric(str_extract_all(used_cars$maximum_seating,
                                                       "\\(?[0-9,.]+\\)?"))
used_cars$width = as.numeric(str_extract_all(used_cars$width,
                                             "\\(?[0-9,.]+\\)?"))
used_cars$highway_fuel_economy=used_cars$highway_fuel_economy * 2.35215 # mph conversion
used_cars$age = 2021 - used_cars$year

# Convert power variable from string into separate horsepower and rpm
power = str_extract_all(used_cars$power,"\\(?[0-9,.]+\\)?")
horsepower = rpm = rep(0, nrow(used_cars))
for (i in 1:nrow(used_cars)) {
  horsepower[i] = as.numeric(power[[i]][1])
  rpm[i] = as.numeric(gsub(",", "", power[[i]][2]))
}
used_cars$horsepower = horsepower; used_cars$rpm = rpm

# Convert torque to get first value (multiple numbers in string like power)
extracted_torque = str_extract_all(used_cars$torque,"\\(?[0-9,.]+\\)?")
torque = rep(0, nrow(used_cars))
for (i in 1:nrow(used_cars)) {
  torque[i] = as.numeric(extracted_torque[[i]][1])
}
used_cars$torque = torque

# city and highway fuel economy are highly correlated, get their average
used_cars$avg_fuel_economy = (used_cars$city_fuel_economy +
                                used_cars$highway_fuel_economy) / 2

# Have listed date as an as.Date predictor
used_cars$listed_date = as.Date(used_cars$listed_date, format = "%m/%d/%Y")

# log transform horsepower and remove original variable, better for Normality assumption
used_cars$log_horsepower = log(used_cars$horsepower, 2)

# Luxury cars have different price distributions (see analysis using Welch's t-Test)
used_cars = used_cars %>%
  mutate(luxury = case_when(make_name %in% c("Ferrari", "Lamborghini", "McLaren",
                                             "Aston Martin", "Alfa Romeo", "Bentley",
                                             "Porsche", "Rolls-Royce", "Mercedes-Benz",
                                             "Maybach") ~ 1, TRUE ~ 0))

# Factoring variables that were simply strings before
used_cars$body_type = as.factor(used_cars$body_type)
used_cars$engine_type = factor(substr(used_cars$engine_type, 1, 2))
used_cars$fuel_type = as.factor(used_cars$fuel_type)
used_cars$make_name = droplevels(as.factor(used_cars$make_name))
```

```r
used_cars$transmission = as.factor(used_cars$transmission)
used_cars$wheel_system = as.factor(used_cars$wheel_system)

# Removing edited or transformed variables
used_cars = subset(used_cars, select = -c(horsepower, year, listed_date, power,
                                          city_fuel_economy, highway_fuel_economy))

# Dataset is very large and has large amounts of NA, remove for better run-time for models
used_cars = used_cars %>%
  na.omit() %>% filter(listing_color != "UNKNOWN")
```

```r
# Should keep it since there are different price distributions
t.test(used_cars$price ~ used_cars$luxury)
```

```
##
##  Welch Two Sample t-test
##
## data:  used_cars$price by used_cars$luxury
## t = -34.047, df = 3701.5, p-value < 2.2e-16
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0
## 95 percent confidence interval:
##  -19160.59 -17074.00
## sample estimates:
## mean in group 0 mean in group 1
##        21321.79        39439.09
```

```r
set.seed(139)
# Issue when splitting into train/test that test has a new make_name that
# was not in train. Filter out all make_names with less than 10 observations
used_cars = used_cars %>% group_by(make_name) %>%
  filter(n() >= 10)

used_cars = used_cars %>% group_by(engine_type) %>%
  filter(n() >= 10)

used_cars$make_name = droplevels(used_cars$make_name)
used_cars$engine_type = droplevels(used_cars$engine_type)

# Do 70%-30% split for train and test respectively
samp_rows <- sample(1:nrow(used_cars), 0.7 * nrow(used_cars), replace=F)
used_cars.train <- used_cars[samp_rows,]
used_cars.test <- used_cars[-samp_rows,]
```

```r
RMSE = function(y,yhat){
  SSE = sum((y-yhat)^2)
  return(sqrt(SSE/length(y)))
}
```

```r
full_model = lm(log(price, 2) ~ ., used_cars.train)
intercept_model = lm(log(price, 2) ~ 1, used_cars.train)
baseline_model = lm(log(price, 2) ~ mileage + log_horsepower + transmission +
                    body_type + fuel_type + avg_fuel_economy +
                    age + is_new, used_cars.train)
```

```r
backward_model = step(full_model, direction = "backward", trace = 0)
formula(backward_model)
```

```r
forward_model = step(intercept_model, scope = list(upper = formula(full_model)),
                     direction = "forward", trace = 0)
formula(forward_model)
```

```r
step_model = step(baseline_model, scope = list(lower = intercept_model,
                                               upper = full_model),
                  direction = "both", trace = 0)
formula(step_model)
```

```r
set.seed(139)
tree.in <- rpart(formula(forward_model), data = used_cars.train,
                 control = list(minsplit = 1, cp = 0, maxdepth = 30))
cp.best <- tree.in$cptable[,"CP"][which.min(tree.in$cptable[,"xerror"])]
tree1 <- prune(tree.in, cp.best)

barplot(tree1$variable.importance, horiz = T, las = 2, cex.names = 0.5,
        main = "Variable Importance for Decision Tree")
```

```r
set.seed(139)
mtry <- c(10, 15, 21)
maxnodes <- c(500, 1000)
pars = expand.grid(maxnodes = maxnodes, mtry = mtry)
rf.RMSEs.in <- rep(NA, nrow(pars))
rf.RMSEs.out <- rep(NA, nrow(pars))
rf.RMSEs.tot <- rep(NA, nrow(pars))
bestRMSE.in <- sd(used_cars.train$price)
bestRMSE.out <- sd(used_cars.test$price)
bestRMSE.tot <- sd(used_cars$price)
for (i in 1:nrow(pars)) { # ntree = 20 so run-time is not too long
  rf <- randomForest(formula(forward_model), data = used_cars.train, mtry = pars[i,2],
                     maxnodes = pars[i,1], ntree = 20)
  rf.RMSEs.in[i] <- RMSE(log(used_cars.train$price, 2), rf$predicted)
  rf.RMSEs.out[i] <- RMSE(log(used_cars.test$price, 2),
                          predict(rf, newdata = used_cars.test))
  rf.RMSEs.tot[i] <- RMSE(log(used_cars$price, 2),
                          predict(rf, newdata = used_cars))
  if (rf.RMSEs.tot[i] < bestRMSE.tot) {
    bestRMSE.tot = rf.RMSEs.tot[i]
    best.mtry.rf = pars[i,2]
    best.maxnodes.rf = pars[i,1]
  }
}
cbind(pars, rf.RMSEs.in, rf.RMSEs.out, rf.RMSEs.tot)
```

```r
rf1 <- randomForest(formula(forward_model),
                    data = used_cars.train, mtry = best.mtry.rf,
                    maxnodes = best.maxnodes.rf, ntree = 200)
varImpPlot(rf1, main = "Variable Importance in Well-Tuned Random Forest Model")
```

Ridge, LASSO, and Random Intercepts

```
X = model.matrix(step_model)[,-1]
X = scale(X)

ridges = cv.glmnet(X, log(used_cars.train$price,2), alpha = 0)
ridge1 = glmnet(X, log(used_cars.train$price,2), alpha = 0,
                lambda = ridges$lambda.min)
```

```
lassos = cv.glmnet(X, unlist(step_model$model["log(price, 2)"]),
                   alpha = 1, lambda = exp(seq(-10,10,0.1)))
lasso = glmnet(X, unlist(step_model$model["log(price, 2)"]),
               alpha = 1, lambda = lassos$lambda.min)
```

```
lmer1 = lmer(log(price, 2) ~ mileage + log_horsepower + make_name + age +
    body_type + engine_type + wheel_system + torque + width +
    fleet + has_accidents + height + transmission + owner_count +
    maximum_seating + avg_fuel_economy + is_new + daysonmarket +
    rpm + fuel_tank_volume + length + (1 | make_name), used_cars)

variances = c((summary(lmer1)$varcor)$make_name[[1]], summary(lmer1)$sigma^2)
icc = variances[1]/sum(variances); icc # quite strong
```

```
## [1] 0.4837459
```

```
lmer2 = lmer(log(price, 2) ~ mileage + log_horsepower + make_name + age +
    body_type + engine_type + wheel_system + torque + width +
    fleet + has_accidents + height + transmission + owner_count +
    maximum_seating + avg_fuel_economy + is_new + daysonmarket +
    rpm + fuel_tank_volume + length + (1 + mileage| make_name), used_cars)

anova(lmer1, lmer2)
```

```
## Data: used_cars
## Models:
## lmer1: log(price, 2) ~ mileage + log_horsepower + make_name + age + body_type + engine_type + wheel_s
## lmer2: log(price, 2) ~ mileage + log_horsepower + make_name + age + body_type + engine_type + wheel_s
##       npar    AIC    BIC  logLik deviance  Chisq Df Pr(>Chisq)
## lmer1   87 8498.1 9273.7 -4162.0   8324.1
## lmer2   89 5466.2 6259.7 -2644.1   5288.2 3035.9  2  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
aov1 = aov(price ~ make_name, used_cars)
summary(aov1)
```

```
##                Df    Sum Sq   Mean Sq F value Pr(>F)
## make_name      44 4.194e+12 9.532e+10   691.9 <2e-16 ***
## Residuals   54975 7.573e+12 1.378e+08
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
aov2 = aov(price ~ make_name * mileage, used_cars)
summary(aov2)
```

```
##                       Df    Sum Sq   Mean Sq F value Pr(>F)
## make_name             44 4.194e+12 9.532e+10  1121.5 <2e-16 ***
## mileage                1 2.375e+12 2.375e+12 27946.6 <2e-16 ***
## make_name:mileage     44 5.295e+11 1.204e+10   141.6 <2e-16 ***
## Residuals          54930 4.668e+12 8.499e+07
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
X.train = scale(model.matrix(step_model)[,-1])
X.test = scale(model.matrix(step_model, data = used_cars.test)[,-1])
X.whole = scale(model.matrix(step_model, data = used_cars)[,-1])
Models = c("Baseline Linear", "Intercept-Only Linear",
           "All-Predictors Linear", "Forward Sequential",
           "Backward Sequential", "Forward-Backward Sequential",
           "Decision Tree", "Random Forest", "Ridge", "LASSO",
           "Random Intercepts", "Random Intercepts, Random Slopes")

RMSE.train = c(RMSE(used_cars.train$price, 2^predict(baseline_model)),
               RMSE(used_cars.train$price, 2^predict(intercept_model)),
               RMSE(used_cars.train$price, 2^predict(full_model)),
               RMSE(used_cars.train$price, 2^predict(forward_model)),
               RMSE(used_cars.train$price, 2^predict(backward_model)),
               RMSE(used_cars.train$price, 2^predict(step_model)),
               RMSE(used_cars.train$price, 2^predict(tree1)),
               RMSE(used_cars.train$price, 2^predict(rf1)),
               RMSE(used_cars.train$price, 2^predict(ridge1, X.train)),
               RMSE(used_cars.train$price, 2^predict(lasso, newx = X.train)),
               RMSE(used_cars.train$price, 2^predict(lmer1, used_cars.train)),
               RMSE(used_cars.train$price, 2^predict(lmer2, used_cars.train)))

RMSE.test = c(RMSE(used_cars.test$price, 2^predict(baseline_model, used_cars.test)),
              RMSE(used_cars.test$price, 2^predict(intercept_model, used_cars.test)),
              RMSE(used_cars.test$price, 2^predict(full_model, used_cars.test)),
              RMSE(used_cars.test$price, 2^predict(forward_model, used_cars.test)),
              RMSE(used_cars.test$price, 2^predict(backward_model, used_cars.test)),
              RMSE(used_cars.test$price, 2^predict(step_model, used_cars.test)),
              RMSE(used_cars.test$price, 2^predict(tree1, used_cars.test)),
              RMSE(used_cars.test$price, 2^predict(rf1, used_cars.test)),
              RMSE(used_cars.test$price, 2^predict(ridge1, X.test)),
              RMSE(used_cars.test$price, 2^predict(lasso, newx = X.test)),
              RMSE(used_cars.test$price, 2^predict(lmer1, used_cars.test)),
              RMSE(used_cars.test$price, 2^predict(lmer2, used_cars.test)))

RMSE.whole = c(RMSE(used_cars$price, 2^predict(baseline_model, used_cars)),
               RMSE(used_cars$price, 2^predict(intercept_model, used_cars)),
               RMSE(used_cars$price, 2^predict(full_model, used_cars)),
               RMSE(used_cars$price, 2^predict(forward_model, used_cars)),
               RMSE(used_cars$price, 2^predict(backward_model, used_cars)),
               RMSE(used_cars$price, 2^predict(step_model, used_cars)),
               RMSE(used_cars$price, 2^predict(tree1, used_cars)),
```

```
                RMSE(used_cars$price, 2^predict(rf1, used_cars)),
                RMSE(used_cars$price, 2^predict(ridge1, X.whole)),
                RMSE(used_cars$price, 2^predict(lasso, newx = X.whole)),
                RMSE(used_cars$price, 2^predict(lmer1, used_cars)),
                RMSE(used_cars$price, 2^predict(lmer2, used_cars)))

RMSE_data = data.frame(Models, RMSE.train, RMSE.test, RMSE.whole)

# Create the LaTeX table output
print(xtable(RMSE_data), comment = FALSE)
```

```
par(mfrow = c(2,2))
plot(baseline_model)
```