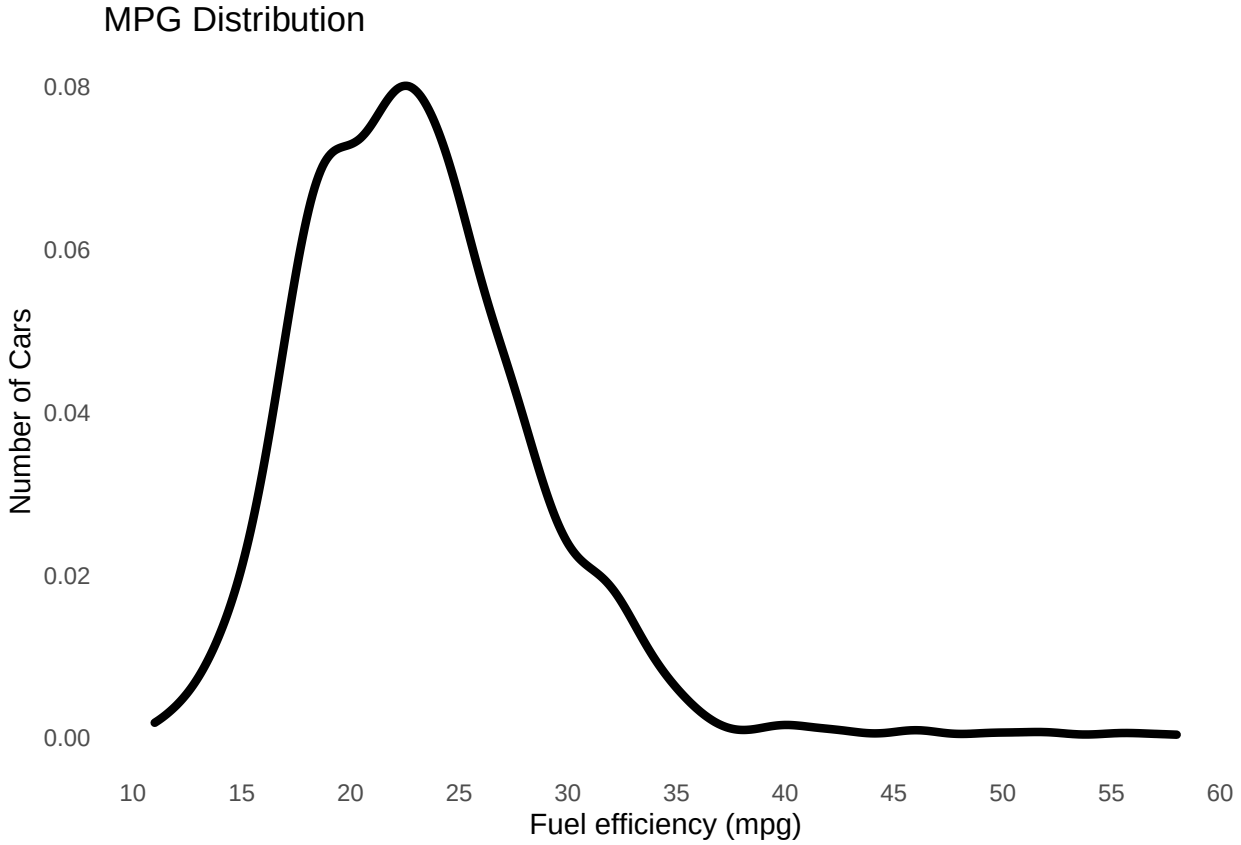
Tidymodels

Edgar Zamora

6/15/2020

MPG Distribuation

Per the density plot below, one can see that a large portion of the cars have a MPG between 20 and 25. Also we can say that the MPG data is postively skewed in that the mean is greater in value than the median which leads to assume that there might a lot of cars that poor MPG like larger trucks or SUVs.



Linear Modeling

Creating a linear model is quite easy. R offers the abilty to use the `lm()` function. The arguments that are needed is the dependent variable (left of tilda) and its features/independent variables (right of tilda). Finally the data is also feed through the data argument.

```
## # A tibble: 17 x 5
```

```
##    term                                    estimate std.error statistic   p.value
##    <chr>                                      <dbl>     <dbl>     <dbl>     <dbl>
##  1 (Intercept)                             43.5       1.10       39.5   1.10e-214
##  2 displacement                            -3.79      0.265     -14.3   9.95e- 43
##  3 cylinders                                0.520     0.162       3.22  1.34e-  3
##  4 gears                                    0.158     0.0700      2.25  2.44e-  2
##  5 transmissionCVT                          4.88      0.404      12.1   1.21e- 31
##  6 transmissionManual                      -1.07      0.366      -2.94  3.40e-  3
##  7 aspirationTurbocharged/Supercharged     -2.19      0.268      -8.19  7.24e- 16
##  8 lockup_torque_converterY                -2.62      0.381      -6.88  9.65e- 12
##  9 drive2-Wheel Drive, Rear                -2.68      0.291      -9.20  1.73e- 19
## 10 drive4-Wheel Drive                      -3.40      0.335     -10.1   3.59e- 23
## 11 driveAll Wheel Drive                    -2.94      0.257     -11.4   9.89e- 29
## 12 max_ethanol                             -0.00738   0.00590    -1.25  2.11e-  1
## 13 recommended_fuelPremium Unleaded Reco~   0.996     0.272       3.66  2.68e-  4
## 14 recommended_fuelPremium Unleaded Requ~   0.592     0.279       2.13  3.37e-  2
## 15 intake_valves_per_cyl                   -1.45      1.62       -0.892 3.72e-  1
## 16 exhaust_valves_per_cyl                  -2.47      1.55       -1.60  1.11e-  1
## 17 fuel_injectionMultipoint/sequential i~ -0.658     0.244      -2.70  7.03e-  3
```

Getting started with tidymodels

The `yardstick` package is used to evaulate how well the model is performing. Functions in this package offer metrics to measure how well models are doing.

Training and testing data

It is good practice to seperate some the original data into a training and testing data as to get a better estimate of how the model will perform on new data. Using the `rsample()` package will help to acheive the splitting of this data.

More often than not data is split 80/20 where 80% goes into training while 20% goest into testing. The reason for doing such a split is to balance charactertics in the data.

Holding out testing data allows one to assess if a model is being overfit.

Creating training/testing splits reduces overfitting. You get better estimates when you evaulate your model(s) on data that it (the model) has not seen/trained on.

Train models with tidymodels

```
## # A tibble: 17 x 5
##    term                                  estimate std.error statistic  p.value
##    <chr>                                    <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)                           3.96e+0  0.0421       94.2  0.
##  2 displacement                         -1.55e-1  0.0101      -15.4  1.16e-47
##  3 cylinders                             6.74e-3  0.00618       1.09 2.76e- 1
##  4 gears                                 1.11e-2  0.00267       4.16 3.54e- 5
##  5 transmissionCVT                       1.63e-1  0.0154       10.6  1.24e-24
##  6 transmissionManual                   -3.26e-2  0.0140       -2.33 2.00e- 2
##  7 aspirationTurbocharged/Supercharged  -9.09e-2  0.0102       -8.90 3.02e-18
```

```
##  8 lockup_torque_converterY                -8.02e-2  0.0145       -5.52 4.32e- 8
##  9 drive2-Wheel Drive, Rear                -8.40e-2  0.0112       -7.52 1.35e-13
## 10 drive4-Wheel Drive                      -1.25e-1  0.0128       -9.76 1.86e-21
## 11 driveAll Wheel Drive                    -1.04e-1  0.00985     -10.6  7.68e-25
## 12 max_ethanol                             -2.68e-4  0.000222     -1.21 2.28e- 1
## 13 recommended_fuelPremium Unleaded Recom~  3.89e-2  0.0104        3.75 1.91e- 4
## 14 recommended_fuelPremium Unleaded Requi~  2.51e-2  0.0107        2.35 1.91e- 2
## 15 intake_valves_per_cyl                   -6.61e-2  0.0568       -1.16 2.45e- 1
## 16 exhaust_valves_per_cyl                  -9.35e-2  0.0538       -1.74 8.25e- 2
## 17 fuel_injectionMultipoint/sequential ig~ -2.24e-2  0.00926      -2.42 1.55e- 2
```

```
## parsnip model object
##
## Fit time:  1000ms
##
## Call:
##  randomForest(x = as.data.frame(x), y = y)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##           Mean of squared residuals: 0.006191373
##                     % Var explained: 88.05
```

Evaluate model performance

There are several things to consider, including both what *metrics* and what *data* to use, when determine how our models did.

Regression models: Focus on evaluating use the **root mean squared error** metric. Measured in the same units as the dependent variable. A lower root mean squarted error indicates a better fit to the data. The `yardstick` package offer convenient functions.

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard      0.0990
## 2 rsq     standard      0.811
## 3 mae     standard      0.0742
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard      0.0606
## 2 rsq     standard      0.930
## 3 mae     standard      0.0440
```

Based on the above output, we can see that the random forest model has a lower **root mean squared error** lead us to prematurally say that the random forest model is a better fit for the data. However, this is premature becuase the fit and subsequent output were done on the *training* data which was used to build the model. Therefore the model is said to be optimistic in its estiamtion so we need to use the *testing* data now.

Using the testing data

We have just trained our models on the entire training set once and then proceded to evaluate the testing data. Though fine it does not offer much in terms of rebustness. Instead a better approach is to to use the method of **resampling**. The idea of resampling is to create simulated data sets that can be used to estimate performance of your model when you want to compare models. The reason for this type of approach is that using the entire training set provides an overly opitmistic result while using your testing data more than twice renders the data meaningless.

Bootstrap

Bootstrap resampling means drawing with replacement (there is a possiblity of seeing the same datapoint twice in a datset) from our original datset and then fitting on that dataset.

Essentially you draw randomly pull from the training set until you reach the same sample size as your training set, which in this case was 917. Again there is a probably of drawing the same observation mulitple times. After creating that new dataset the data is split into a training and testing dataset and fit with the model and evaluated. This process of sampling, fitting, and evaulating is repeated multiple times. After done a number of times an average of the perfromance metrics is taken.

Visualizing both models

```
## `geom_smooth()` using formula 'y ~ x'
```