Predicting Car Prices

Emmanuel Messori

01/10/2021

Objective

In this project I'll work with a 1987 data set retrieved from the UCI Machine Learning Archive which contains information about 205 cars. The objective is to predict the cars' price by a selected set of features.

The Data

> Data Set Information:
>
> This data set consists of three types of entities: (a) the specification of an auto in terms of various characteristics, (b) its assigned insurance risk rating, (c) its normalized losses in use as compared to other cars. The second rating corresponds to the degree to which the auto is more risky than its price indicates. Cars are initially assigned a risk factor symbol associated with its price. Then, if it is more risky (or less), this symbol is adjusted by moving it up (or down) the scale. Actuarians call this process "symboling". A value of +3 indicates that the auto is risky, -3 that it is probably pretty safe.
>
> The third factor is the relative average loss payment per insured vehicle year. This value is normalized for all autos within a particular size classification (two-door small, station wagons, sports/speciality, etc…), and represents the average loss per car per year.
>
> Note: Several of the attributes in the database could be used as a "class" attribute.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
# retrieving the attribute names from `colnames.txt`
colnames <- read.table("colnames.txt")
# the NA values are signaled by a question mark
cars <- read_csv("cars.data", col_names = colnames$V1, na = "?", show_col_types = FALSE)
```

```r
summary(cars)
```

```
##    symboling       normalized-losses     make            fuel-type
##  Min.   :-2.0000   Min.   : 65       Length:205         Length:205
##  1st Qu.: 0.0000   1st Qu.: 94       Class :character   Class :character
##  Median : 1.0000   Median :115       Mode  :character   Mode  :character
##  Mean   : 0.8341   Mean   :122
##  3rd Qu.: 2.0000   3rd Qu.:150
##  Max.   : 3.0000   Max.   :256
##                    NA's   :41
##   aspiration        num-of-doors        body-style        drive-wheels
##  Length:205         Length:205         Length:205         Length:205
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##  engine-location      wheel-base         length          width
##  Length:205         Min.   : 86.60   Min.   :141.1   Min.   :60.30
##  Class :character   1st Qu.: 94.50   1st Qu.:166.3   1st Qu.:64.10
##  Mode  :character   Median : 97.00   Median :173.2   Median :65.50
##                     Mean   : 98.76   Mean   :174.0   Mean   :65.91
##                     3rd Qu.:102.40   3rd Qu.:183.1   3rd Qu.:66.90
##                     Max.   :120.90   Max.   :208.1   Max.   :72.30
##
##      height       curb-weight   engine-type        num-of-cylinders
##  Min.   :47.80   Min.   :1488   Length:205         Length:205
##  1st Qu.:52.00   1st Qu.:2145   Class :character   Class :character
##  Median :54.10   Median :2414   Mode  :character   Mode  :character
##  Mean   :53.72   Mean   :2556
##  3rd Qu.:55.50   3rd Qu.:2935
##  Max.   :59.80   Max.   :4066
##
##   engine-size    fuel-system             bore          stroke
##  Min.   : 61.0   Length:205         Min.   :2.54   Min.   :2.070
##  1st Qu.: 97.0   Class :character   1st Qu.:3.15   1st Qu.:3.110
##  Median :120.0   Mode  :character   Median :3.31   Median :3.290
##  Mean   :126.9                      Mean   :3.33   Mean   :3.255
##  3rd Qu.:141.0                      3rd Qu.:3.59   3rd Qu.:3.410
##  Max.   :326.0                      Max.   :3.94   Max.   :4.170
##                                     NA's   :4      NA's   :4
##  compression-ratio   horsepower       peak-rpm       city-mpg
##  Min.   : 7.00     Min.   : 48.0   Min.   :4150   Min.   :13.00
##  1st Qu.: 8.60     1st Qu.: 70.0   1st Qu.:4800   1st Qu.:19.00
##  Median : 9.00     Median : 95.0   Median :5200   Median :24.00
##  Mean   :10.14     Mean   :104.3   Mean   :5125   Mean   :25.22
##  3rd Qu.: 9.40     3rd Qu.:116.0   3rd Qu.:5500   3rd Qu.:30.00
##  Max.   :23.00     Max.   :288.0   Max.   :6600   Max.   :49.00
##                    NA's   :2       NA's   :2
##   highway-mpg        price
##  Min.   :16.00   Min.   : 5118
##  1st Qu.:25.00   1st Qu.: 7775
```

```
##  Median :30.00   Median :10295
##  Mean   :30.75   Mean   :13207
##  3rd Qu.:34.00   3rd Qu.:16500
##  Max.   :54.00   Max.   :45400
##                  NA's   :4
```

Feature relationship

The dataset contain numerical data and categorical data, and has missing values. Before setting up a model, it is imperative to study the relationship between the potential predictors and the outcome variable. We'll use the `featurePlot()` function from the `caret` package to plot price against each numeric variable:

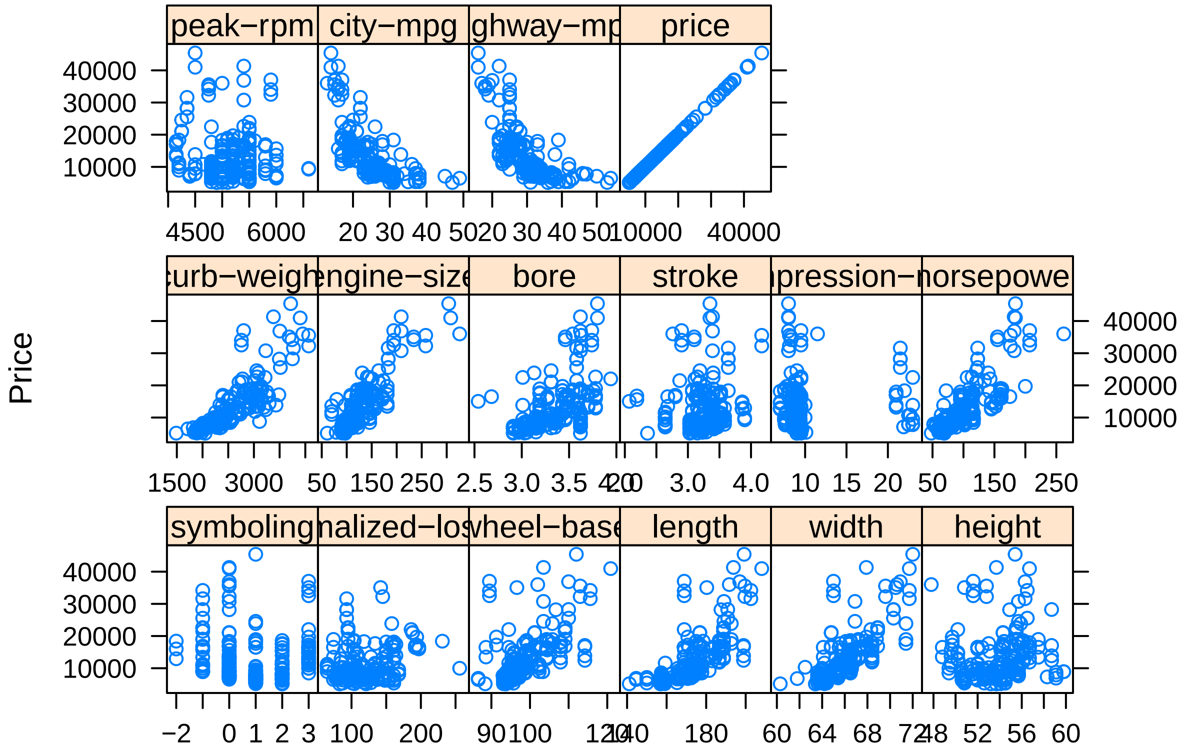
```r
library(caret)
```

```
## Le chargement a nécessité le package : lattice
```

```
##
## Attachement du package : 'caret'
```

```
## L'objet suivant est masqué depuis 'package:purrr':
##
##     lift
```

```r
# keep the numeric variables and exclude NA in the dependent variable
cars %>%
    select(where(is.numeric)) %>%
    filter(!is.na(price)) -> cnum
featurePlot(cnum, cnum$price, labels = c("", "Price"))
```

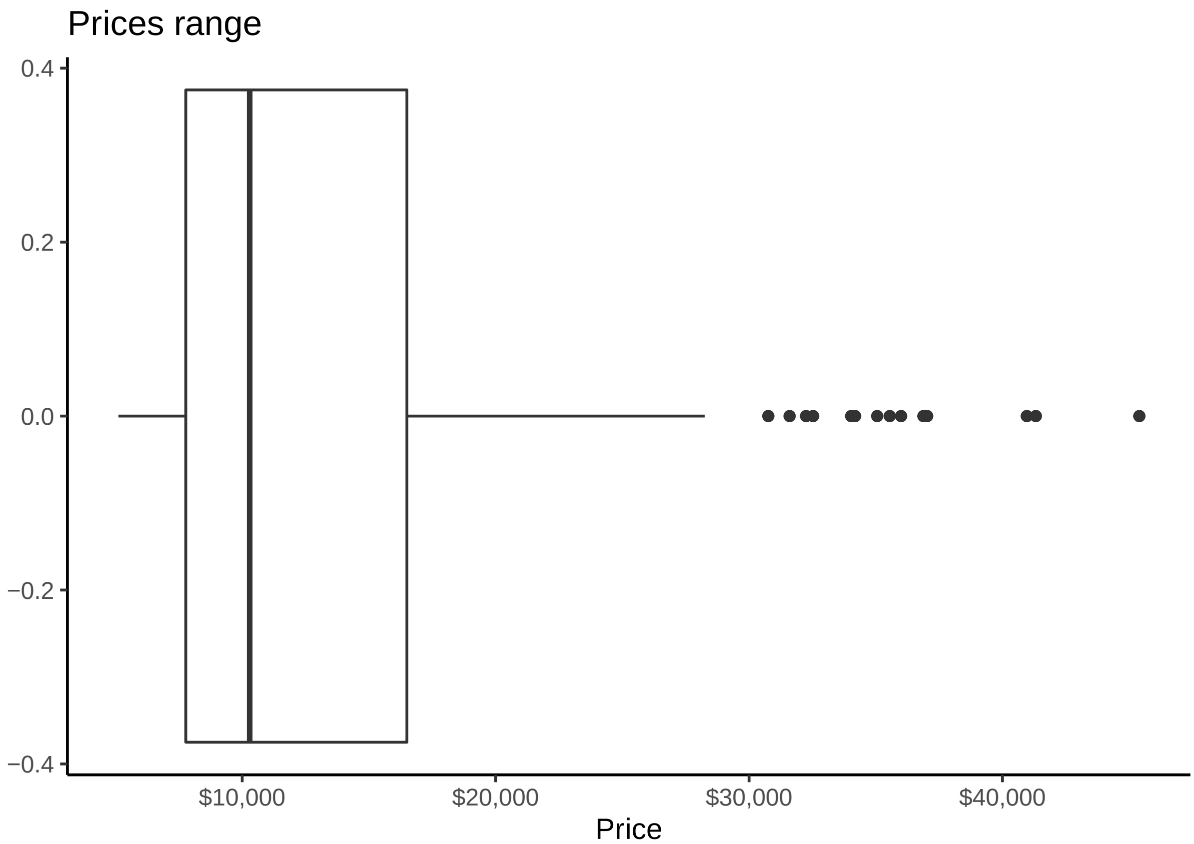


We can spot many variables that seem to have a linear relationship with price :

- Positive
  - engine-size
  - length
  - curb-weight
  - width
  - horsepower or
- Negative
  - city-mpg
  - highway-mpg

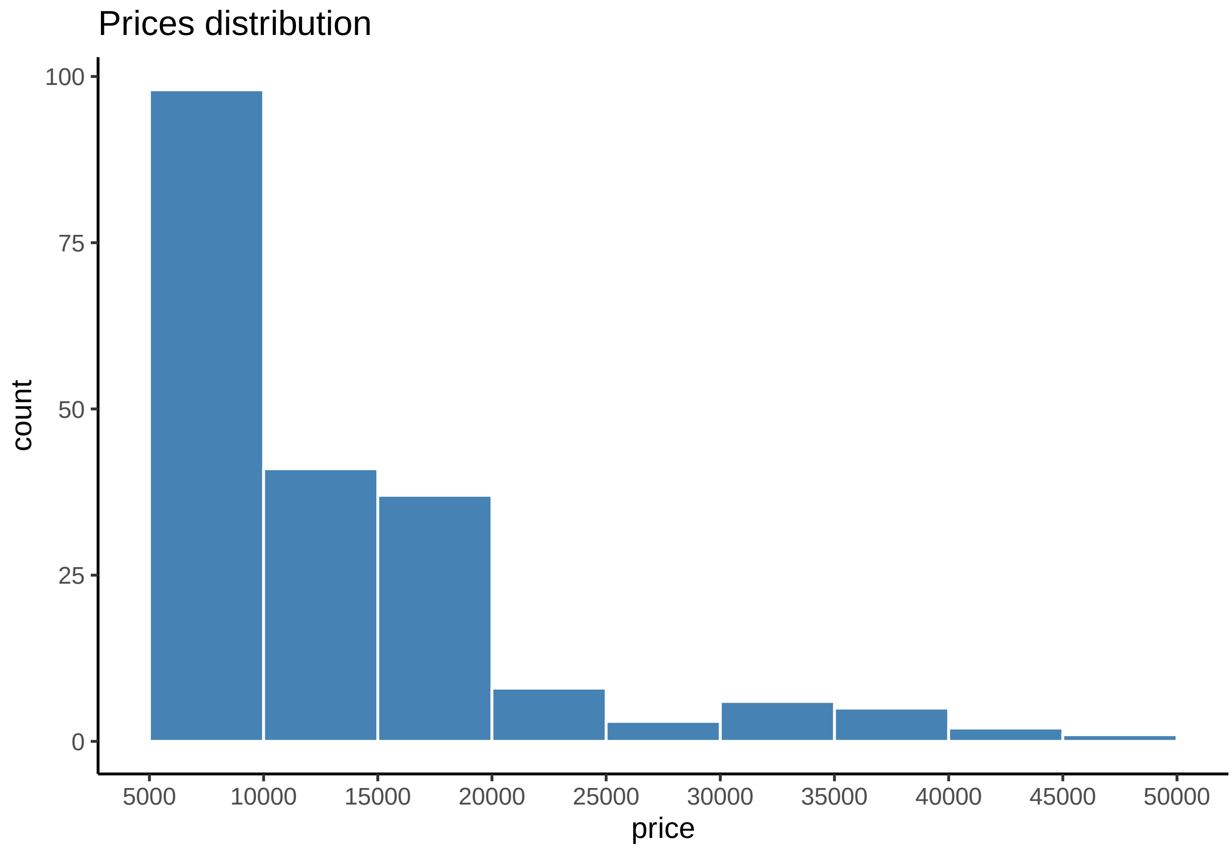
Let's now visualize the distribution of the target variable:

```
theme_set(theme_classic())
cnum %>%
    ggplot(aes(price)) + geom_boxplot() + labs(title = "Prices range", x = "Price",
    y = "") + scale_x_continuous(labels = scales::dollar)
```



The price is ranging from 5118$ to 45400$, with the vast majority of the prices to be found under 20000$.

```
cnum %>%
    ggplot(aes(price)) + geom_histogram(color = "white", fill = "steelblue", bins = 10,
    binwidth = 5000, boundary = 0.5) + labs(title = "Prices distribution") + scale_x_continuous(n.breaks
```



It's interesting to further analyse the outliers which we can easily spot in the boxplot above :

```
cars %>%
    filter(price >= 30000)
```

```
## # A tibble: 14 x 26
##    symboling `normalized-losses` make          `fuel-type` aspiration `num-of-doors`
##        <dbl>               <dbl> <chr>         <chr>       <chr>      <chr>
##  1         0                  NA bmw           gas         std        four
##  2         0                  NA bmw           gas         std        two
##  3         0                  NA bmw           gas         std        four
##  4         0                 145 jaguar        gas         std        four
##  5         0                  NA jaguar        gas         std        four
##  6         0                  NA jaguar        gas         std        two
##  7        -1                  93 mercedes-benz diesel      turbo      four
##  8        -1                  NA mercedes-benz gas         std        four
##  9         3                 142 mercedes-benz gas         std        two
## 10         0                  NA mercedes-benz gas         std        four
## 11         1                  NA mercedes-benz gas         std        two
## 12         3                  NA porsche       gas         std        two
## 13         3                  NA porsche       gas         std        two
## 14         3                  NA porsche       gas         std        two
## # ... with 20 more variables: body-style <chr>, drive-wheels <chr>,
## #   engine-location <chr>, wheel-base <dbl>, length <dbl>, width <dbl>,
## #   height <dbl>, curb-weight <dbl>, engine-type <chr>, num-of-cylinders <chr>,
## #   engine-size <dbl>, fuel-system <chr>, bore <dbl>, stroke <dbl>,
```

```
## #   compression-ratio <dbl>, horsepower <dbl>, peak-rpm <dbl>, city-mpg <dbl>,
## #   highway-mpg <dbl>, price <dbl>
```

The brands found in the outliers are bmw, jaguar, mercedes-benz and porsche. Let's see if we find the same brands in the preceding quartiles:

```
cars %>%
    filter(price <= 30000 & make %in% c("bmw", "jaguar", "mercedes-benz", "porsche"))
```

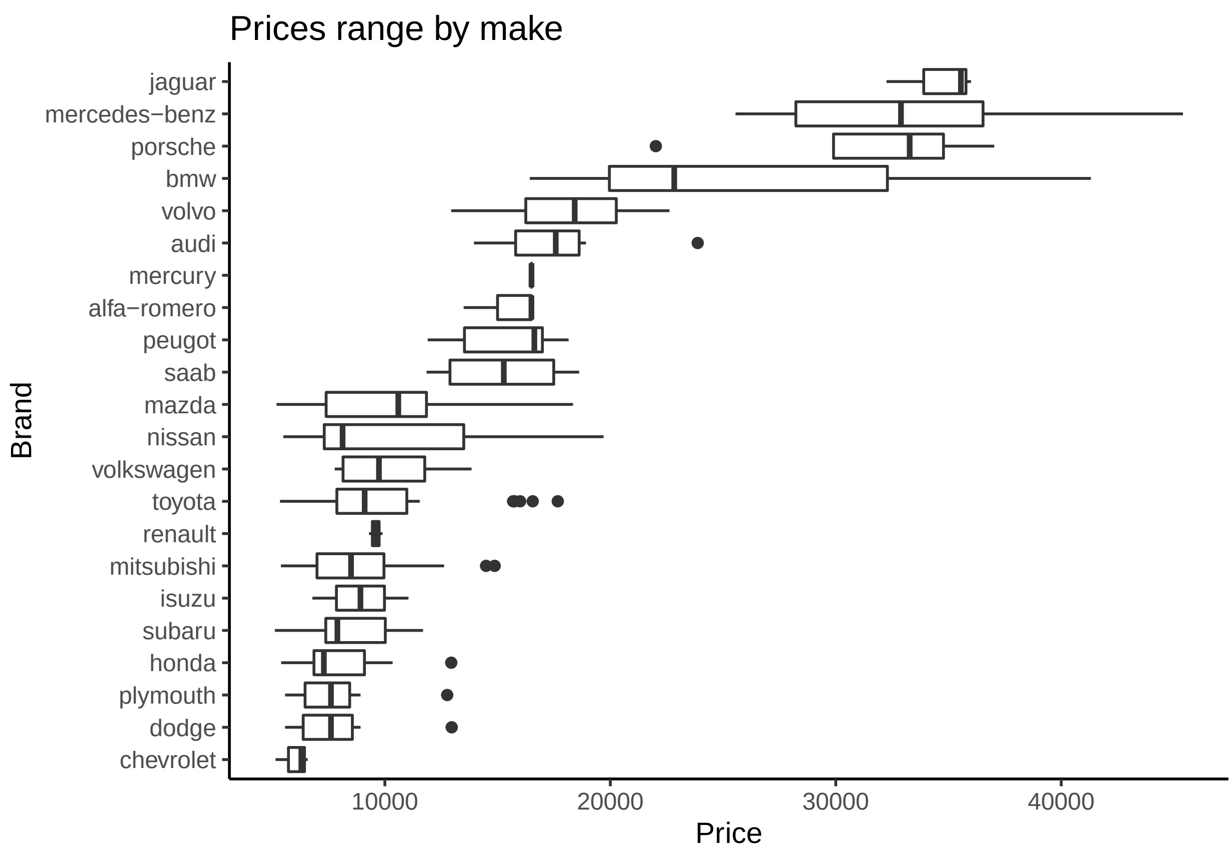
```
## # A tibble: 9 x 26
##   symboling `normalized-losses` make          `fuel-type` aspiration `num-of-doors`
##       <dbl>               <dbl> <chr>         <chr>       <chr>      <chr>
## 1         2                 192 bmw           gas         std        two
## 2         0                 192 bmw           gas         std        four
## 3         0                 188 bmw           gas         std        two
## 4         0                 188 bmw           gas         std        four
## 5         1                  NA bmw           gas         std        four
## 6        -1                  93 mercedes-benz diesel      turbo      four
## 7        -1                  93 mercedes-benz diesel      turbo      four
## 8         0                  93 mercedes-benz diesel      turbo      two
## 9         3                 186 porsche       gas         std        two
## # ... with 20 more variables: body-style <chr>, drive-wheels <chr>,
## #   engine-location <chr>, wheel-base <dbl>, length <dbl>, width <dbl>,
## #   height <dbl>, curb-weight <dbl>, engine-type <chr>, num-of-cylinders <chr>,
## #   engine-size <dbl>, fuel-system <chr>, bore <dbl>, stroke <dbl>,
## #   compression-ratio <dbl>, horsepower <dbl>, peak-rpm <dbl>, city-mpg <dbl>,
## #   highway-mpg <dbl>, price <dbl>
```

Jaguar is not present and the other models seem to differ from the outliers by a lower engine size and horsepower. There seems to be no specific reason to reject them as the higher price range matches the more expensive brands.

```
quantile(cars$price, na.rm = TRUE)
```

```
##    0%   25%   50%   75%  100%
##  5118  7775 10295 16500 45400
```

```
ggplot(cars, aes(price, reorder(make, price, mean, na.rm = TRUE))) + geom_boxplot() +
    labs(title = "Prices range by make", y = "Brand", x = "Price")
```



Model conception

We will now use the numeric attributes to build a k-nearest neighbors model to predict prices.

```
library(caret)
# create train and test sets
set.seed(1)
trindex <- createDataPartition(cnum$price, p = 0.85, list = FALSE)
train <- cnum[trindex, ]
test <- cnum[-trindex, ]
```

```
# setting up the hyperparameter grid
kneigh <- expand.grid(k = 1:20)

# setting up a 5 folds cross validation
mytrain <- trainControl(method = "cv", number = 5)

# splitting target and predictors
X <- as.data.frame(train[1:15])
target <- train[[16]]

# required package
library(RANN)

# setting up with two models, a random forest and a knn
```

```r
rf <- train(x = X, y = target, method = "ranger", trControl = mytrain, preProcess = "knnImpute")

knn <- train(x = X, y = target, method = "knn", trControl = mytrain, preProcess = "knnImpute",
    tuneGrid = kneigh)

print(knn)
```

```
## k-Nearest Neighbors
##
## 173 samples
##  15 predictor
##
## Pre-processing: nearest neighbor imputation (15), centered (15), scaled (15)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 140, 137, 138, 138, 139
## Resampling results across tuning parameters:
##
##   k  RMSE      Rsquared   MAE
##    1  3556.775  0.7738273  2048.651
##    2  3427.052  0.7977470  2074.870
##    3  3487.501  0.7940264  2174.290
##    4  3524.755  0.8010687  2168.887
##    5  3689.461  0.7923679  2255.317
##    6  3744.910  0.7825322  2326.604
##    7  3918.675  0.7698338  2436.456
##    8  3961.881  0.7718798  2461.269
##    9  3995.915  0.7784965  2463.450
##   10  4088.666  0.7701720  2509.780
##   11  4179.716  0.7633312  2509.541
##   12  4230.186  0.7597300  2521.214
##   13  4286.437  0.7571900  2554.083
##   14  4279.797  0.7613535  2540.322
##   15  4278.676  0.7663824  2511.788
##   16  4247.150  0.7823405  2497.047
##   17  4274.590  0.7831812  2504.797
##   18  4304.862  0.7816977  2531.531
##   19  4346.883  0.7776927  2535.994
##   20  4345.352  0.7820675  2499.520
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 2.
```

```r
print(rf)
```

```
## Random Forest
##
## 173 samples
##  15 predictor
##
## Pre-processing: nearest neighbor imputation (15), centered (15), scaled (15)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 139, 137, 138, 139, 139
```

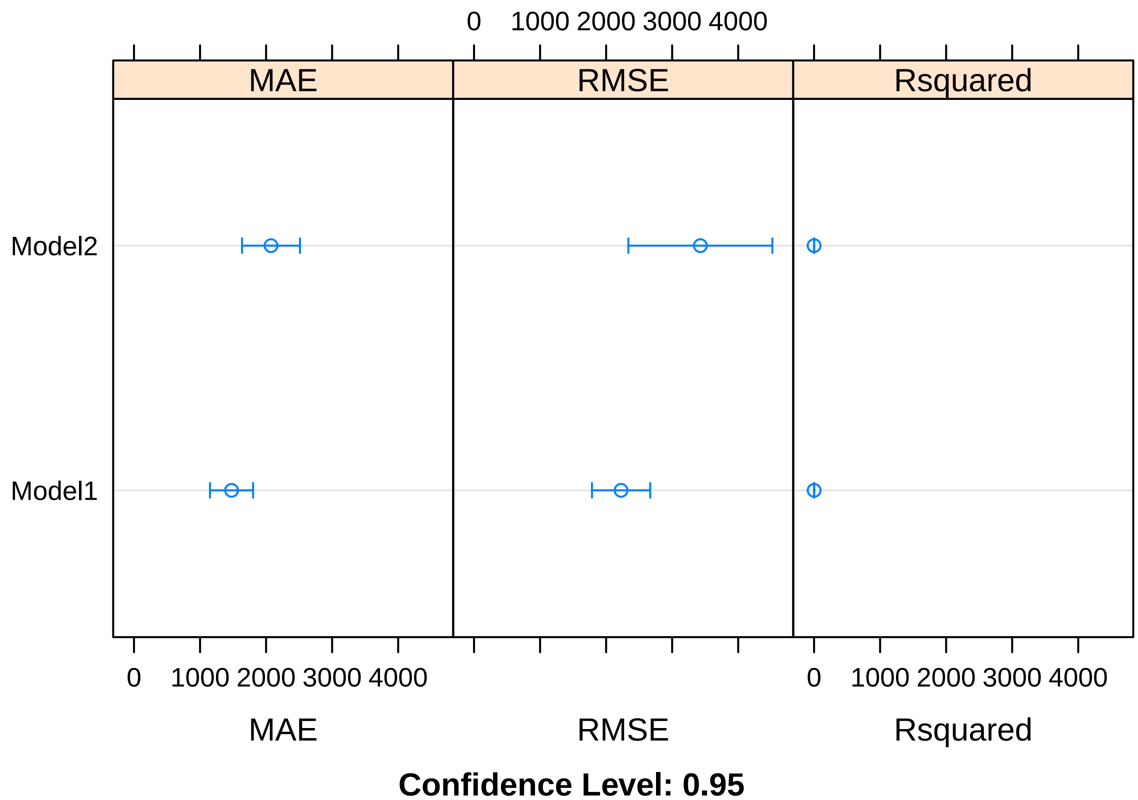
```
## Resampling results across tuning parameters:
##
##   mtry  splitrule   RMSE      Rsquared   MAE
##    2    variance    2712.456  0.8820105  1701.434
##    2    extratrees  3013.460  0.8585249  1882.648
##    8    variance    2300.137  0.9124406  1491.756
##    8    extratrees  2647.553  0.8840129  1664.698
##   15    variance    2226.738  0.9171427  1477.039
##   15    extratrees  2588.281  0.8893183  1617.642
##
## Tuning parameter 'min.node.size' was held constant at a value of 5
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were mtry = 15, splitrule = variance
##  and min.node.size = 5.
```

The random forest model outperforms the knn with a Train RMSE of 2226.74 versus 3427.05'.

```
model_list <- list(rf, knn)
rs <- resamples(model_list)
summary(rs)
```

```
##
## Call:
## summary.resamples(object = rs)
##
## Models: Model1, Model2
## Number of resamples: 5
##
## MAE
##            Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## Model1 1263.293 1298.441 1410.275 1477.039 1495.702 1917.484    0
## Model2 1748.706 1777.583 1947.990 2074.870 2389.719 2510.352    0
##
## RMSE
##            Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## Model1 1895.601 2111.340 2139.145 2226.738 2153.959 2833.643    0
## Model2 2291.938 3058.669 3213.689 3427.052 4019.596 4551.367    0
##
## Rsquared
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Model1 0.8906534 0.9090694 0.9159217 0.9171427 0.9231792 0.9468901    0
## Model2 0.6783047 0.6994616 0.8625408 0.7977470 0.8631801 0.8852476    0
```

```
dotplot(rs)
```



The random forest model shows also less variability in all the metrics.

Model evaluation

```
predictions <- predict(rf, newdata = test)
postResample(pred = predictions, obs = test$price)
```

```
##        RMSE    Rsquared         MAE
## 1727.262425    0.972398 1284.468980
```

The random forest seem to perform very well on the test data, with even lower RMSE and MAE, and a higher Rsquared. We should take these results carefully given the small amount of data.