Data Analyisis for Business - Final Project - Group: Mission 42

Data Analysis For Business - Final Project

Group: Mission 42

- Yaseen Abdulmahdi (277021)
- Maria Chiara Lischi (271281)
- Matteo Spadaccia (277141)
- Eyad Walid (273821)

Introduction and data description

> In this project, we will be working with a data-set that contains information about automobiles. Each of the row 205 of the data-set represents a different car model, and for each observation, we have details about the car features and price. Our goal is to build a predictive model for the sale prices of cars based on their characteristics, after that we will explore the possibility of clustering car types using the available covariates.

Preliminary code

> In this very first section [code section: 1] we loaded packages and libraries we exploited to empower our analyis

[code section: 1]

Data cleaning and encoding

> In this first part [code sections 2 -> 5] our aim is to to gain a general understanding of the structure and contents of the data-set; we will also prepare it for the following analysis by cleaning it and encoding the categorical variables.

[code section: 2]

```
# IMPORTING CAR PRICES DATASET

# imports data-set as data-frame from .csv file with relative path
car_prices = read.csv("data/CarPrices.csv", sep = ",", dec = ".", header = T, colClasses = "c
haracter")

print("Cars dataframe:")
```

```
## [1] "Cars dataframe:"
```

```
#str(car_prices)
```

We started by checking for any issues with data quality, such as duplicates, missing and unary values, which were not found; we performed data cleaning steps, by transforming quantitative variables in the proper data type.

[code section: 3]

```
cleaned_car_prices = car_prices
numericalVariables <- c('wheelbase', 'carlength', 'carwidth', 'carheight', 'curbweight', 'eng
inesize', 'boreratio', 'stroke', 'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'hig
hwaympg', 'price')
for (tempVar in numericalVariables) {
  cleaned_car_prices[[tempVar]] <- as.numeric(cleaned_car_prices[[tempVar]])}
```

We decided to concentrate on the data-set general structure, by handling the issue with the column “carCompany”, which was present in the data-set description but not in the actual data-set. We created this column by extracting information from the “CarName” column.

[code section: 4]

```
# Creating a column for the company name
cleaned_car_prices$carCompany <- word(cleaned_car_prices$CarName, 1)

# Correcting some typos
cleaned_car_prices$carCompany[cleaned_car_prices$carCompany == "porcshce"] <- "porsche"
cleaned_car_prices$carCompany[cleaned_car_prices$carCompany == "vokswagen"] <- "volkswagen"
cleaned_car_prices$carCompany[cleaned_car_prices$carCompany == "vw"] <- "volkswagen"
cleaned_car_prices$carCompany[cleaned_car_prices$carCompany == "toyouta"] <- "toyota"
cleaned_car_prices$carCompany[cleaned_car_prices$carCompany == "Nissan"] <- "nissan"
cleaned_car_prices$carCompany[cleaned_car_prices$carCompany == "maxda"] <- "mazda"
cleaned_car_prices$carCompany[cleaned_car_prices$carCompany == "alfa-romero"] <- "alfaromeo"
cleaned_car_prices$fuelsystem[cleaned_car_prices$fuelsystem == "mfi"] <- "mpfi"
```

Once the data were cleaned, we proceeded by encoding the categorical variables we considered useful for the regression task (hence, we excluded CarModel and car_ID).

[code section: 5]

```
encoded_car_prices <- cleaned_car_prices
categoricalVariables <- c('symboling', 'fueltype', 'aspiration', 'doornumber', 'carbody', 'dr
ivewheel', 'enginelocation', 'enginetype', 'cylindernumber', 'fuelsystem', 'carCompany')

fact_car_prices <- cleaned_car_prices
for (tempVar in categoricalVariables) {
  fact_car_prices[[tempVar]] <- as.factor(cleaned_car_prices[[tempVar]])}

encoded_car_prices <- one_hot(as.data.table(fact_car_prices))
colnames(encoded_car_prices)[which(names(encoded_car_prices) == "symboling_-1")] <- "symbolin
g_minus_1"
colnames(encoded_car_prices)[which(names(encoded_car_prices) == "symboling_-2")] <- "symbolin
g_minus_2"
```

Exploratory data analysis

> To achieve the goals of building a predictive model for car sale prices and performing cluster analysis of car types, in this part [coding sections: 6 -> 14] we conducted an exploratory data analysis to gain insights and to understand the variables and their correlations.

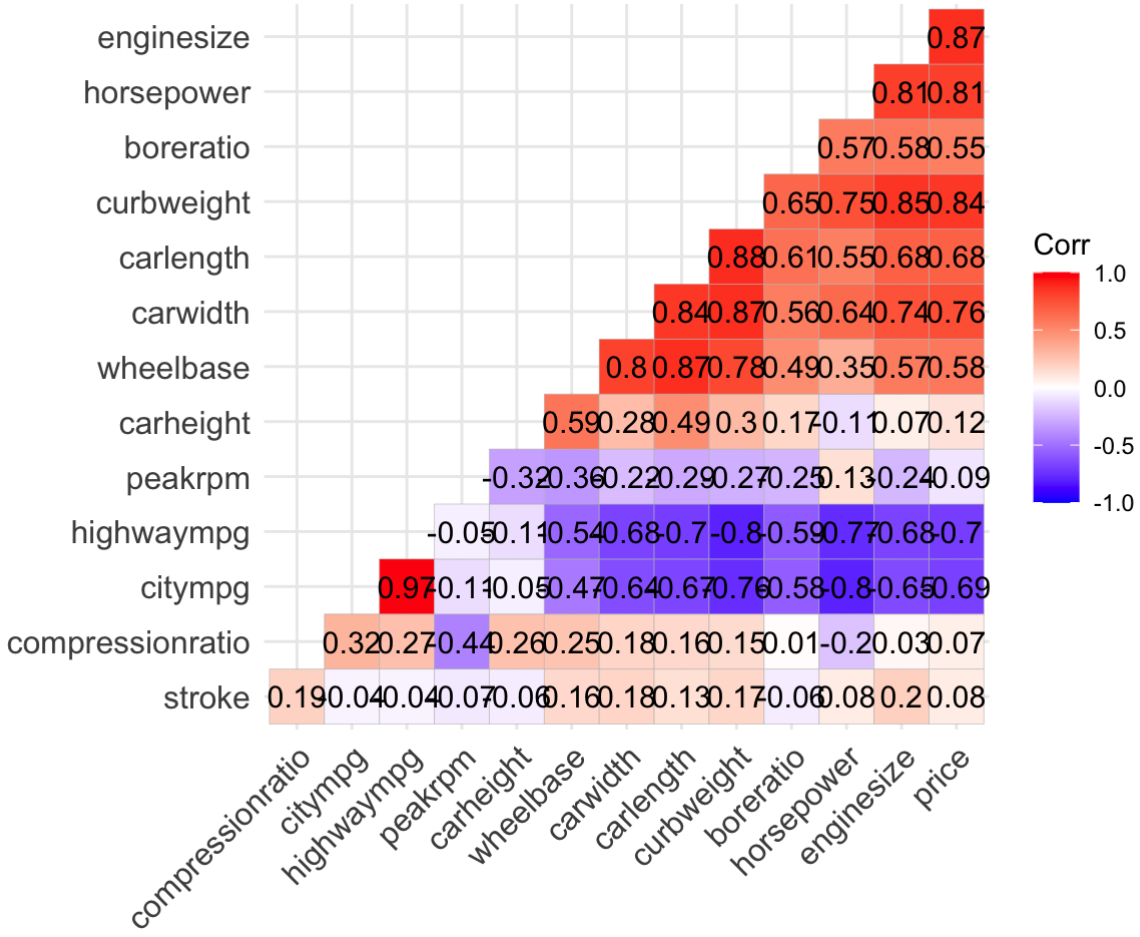
Firstly, we performed a correlation analysis over the quantitative variable, to understand correlation among them, specifically, we concentrated in the correlation that variables have with price, which is the continuous variable we aim to predict in the following task.

[code section: 6]

```
corr_matrix = round(cor(subset(encoded_car_prices %>% select(numericalVariables))), 2)
```

```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## ℹ Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(numericalVariables)
##
##   # Now:
##   data %>% select(all_of(numericalVariables))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
ggcorrplot(corr_matrix, hc.order = TRUE, type = "lower", lab = TRUE)
```



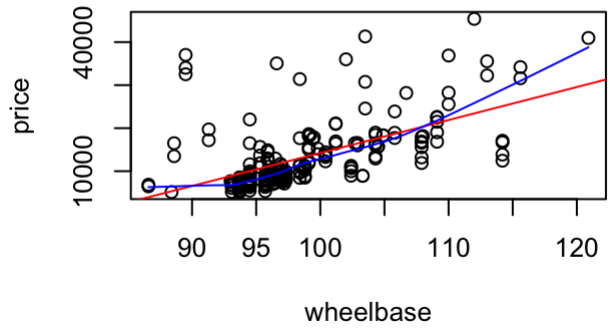
We noticed that there is an very high correlation (0.87) between engine size and price. This is expected, as larger engines are intuitively associated with higher performance and luxury. Also horsepower has a significant correlation with price (0.81). Furthermore, we observed significant correlations between curb weight (0.84), car length (0.68), car width (0.76), and wheelbase (0.58) with car price. These variables represent different dimensions of a car, reflecting its size and proportions. The positive correlations indicate that heavier and larger cars tend to be associated with higher prices. In contrast, city mpg and highway mpg shows strong negative correlations with car price (respectively -0.71 and -0.79). We assume that is because cars with higher fuel efficiency have lower operating costs, which brings to lower prices.

We also gained some insights on the relationship among the quantitative variables and the price, plotting scatter-plots (and fitted lines over them).

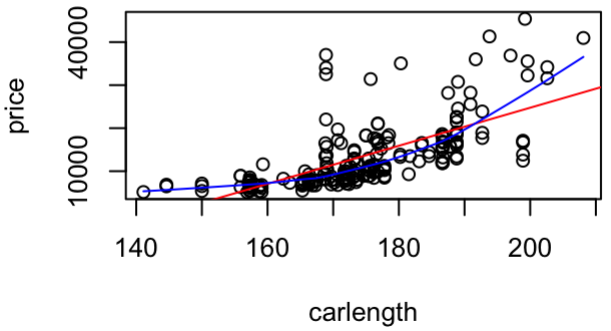
[code section: 7]

```
par(mfrow=c(2,2))
numericalRegressors = numericalVariables[1:(length(numericalVariables) - 1)]
for (tempVar in numericalRegressors){
  plot(fact_car_prices[,tempVar], fact_car_prices$price, main=paste("Scatterplot: price ove
r", tempVar), xlab=tempVar, ylab="price", pch=1)
  abline(lm(fact_car_prices$price~fact_car_prices[,tempVar]), col="red")
  lines(lowess(fact_car_prices[,tempVar],fact_car_prices$price), col="blue")}
```

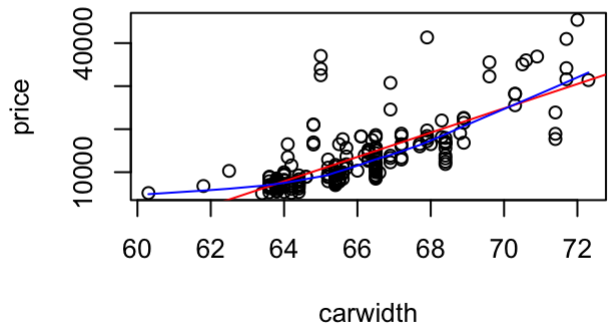
Scatterplot: price over wheelbase



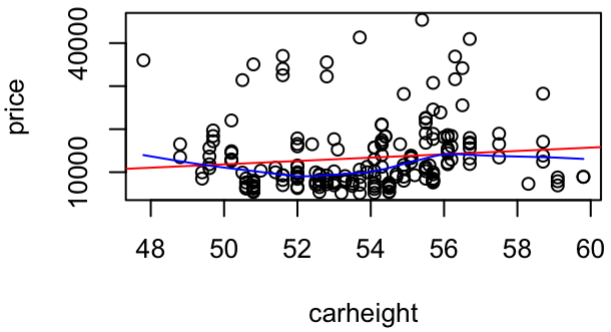
Scatterplot: price over carlength



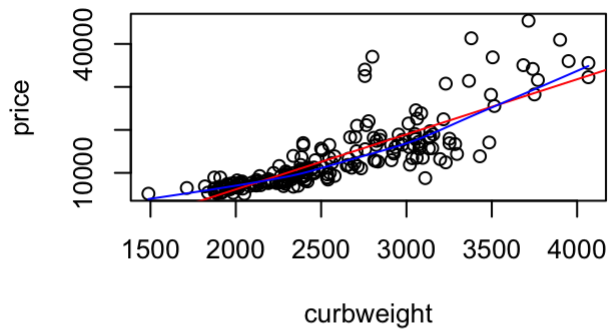
Scatterplot: price over carwidth



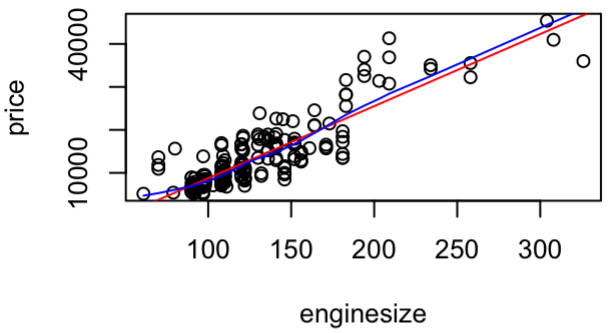
Scatterplot: price over carheight



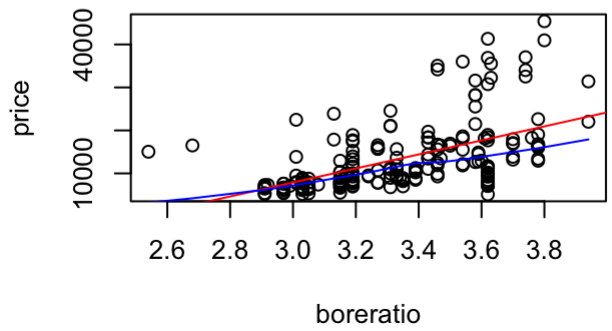
Scatterplot: price over curbweight



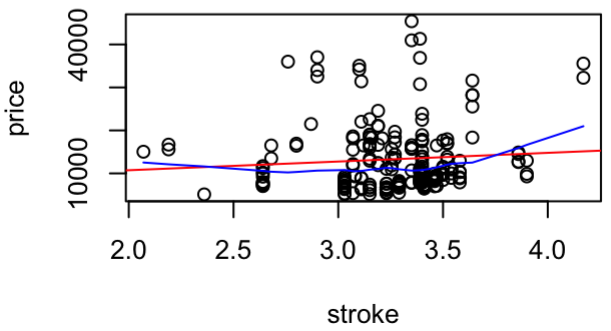
Scatterplot: price over enginesize

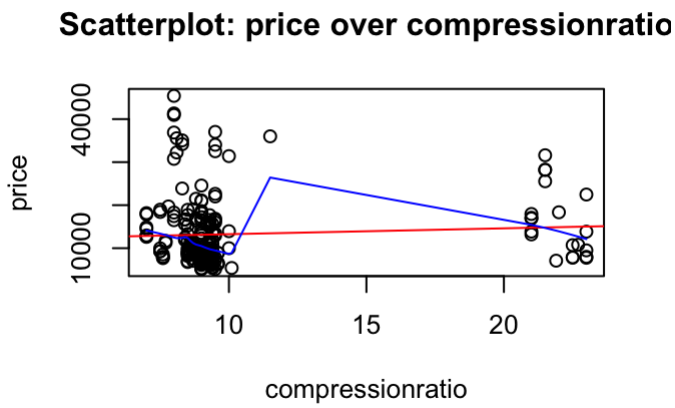


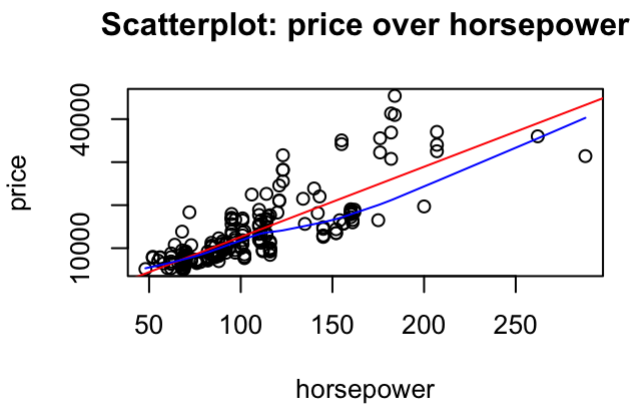
Scatterplot: price over boreratio

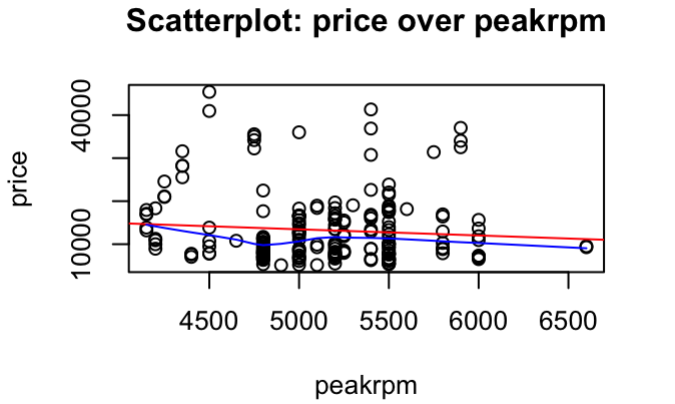


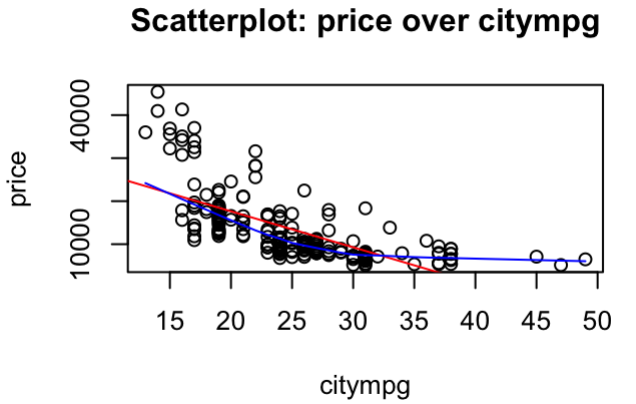
Scatterplot: price over stroke

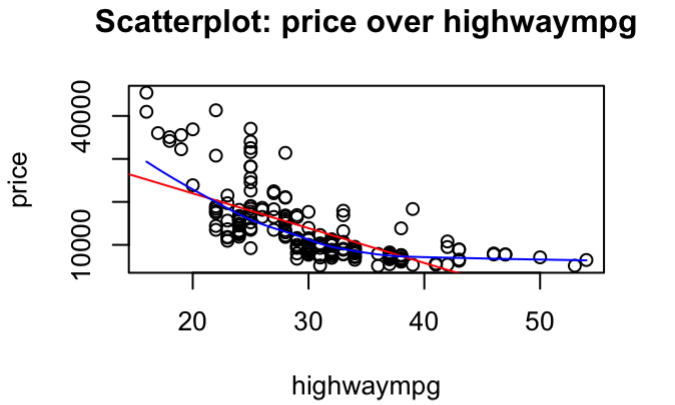












We noticed that many variables seem to have an almost linear relationship with price (such as curb-weight, engine size, bore-ratio, horse power), while other seem to have an almost quadratic relationship with price (such as car length)

Since our goal for the regression task is to predict the car prices, which is a continuous variable, we plotted a box-plot to visualize the distribution of prices over our data-set.

[code section: 8] !REMOVED FROM SUMMARY!

One notable observation is the presence of outliers in the higher price range.

Our data-set is also rich of categorical variables, hence we decided to gain insights about each of them, by exploring the frequency of the different values they assume. We choose pie-charts for this kind of analysis.

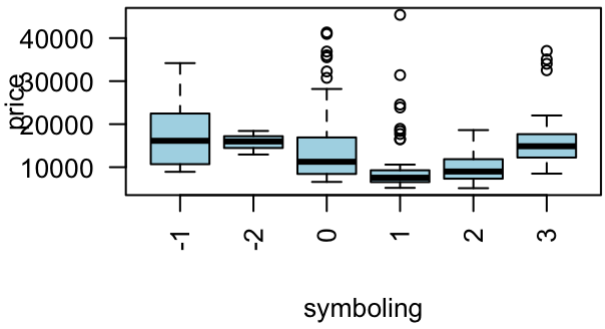
[code section: 8] !REMOVED FROM SUMMARY!

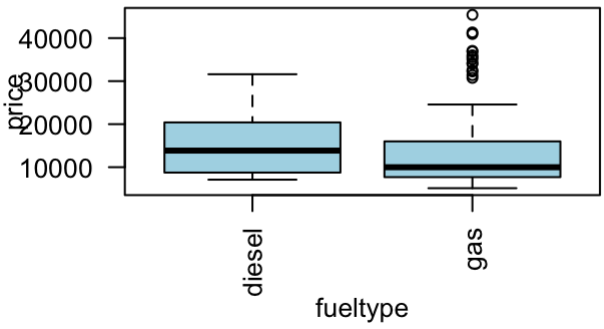
Notable findings are that the most frequent engine type is ohc, that the most frequent car producing company is Toyota, and the most common cilinder number is four.

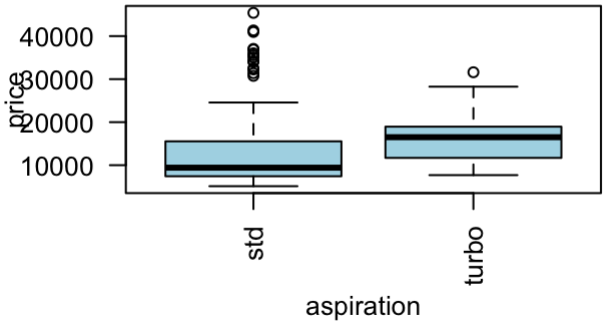
For gaining useful insights for our regression task, we decided to plot box-plots to visualize the distribution of car prices across different car companies. In the graphs below, the x-axis denotes the various categorical variables, and the y-axis represents the car prices.

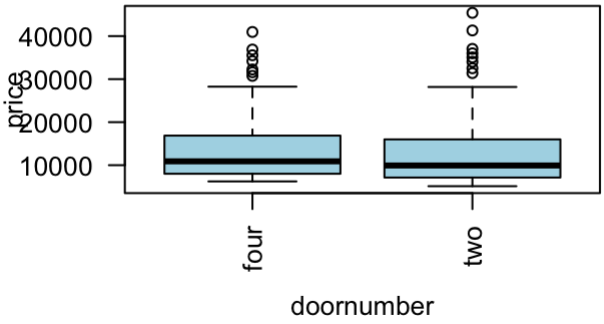
[code section: 9]

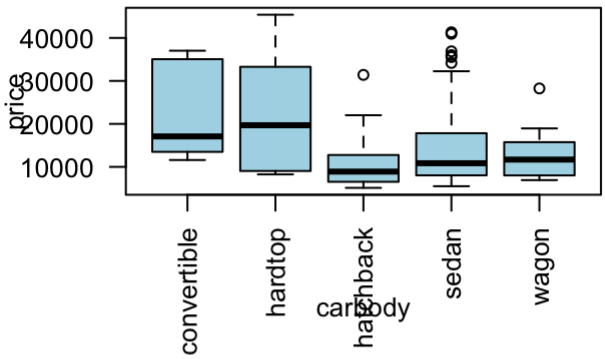
```
par(mfrow=c(2,2))
for (tempVar in categoricalVariables){
  boxplot(fact_car_prices$price ~ fact_car_prices[,tempVar], col = c("lightblue"), xlab = tem
pVar, ylab = "price", tick = FALSE, las = 2)}
```

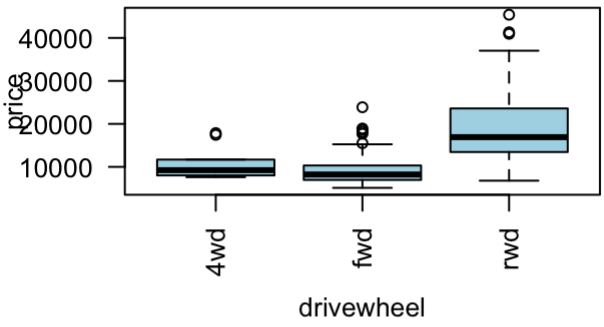

price
40000
30000
20000
10000
-1
-2
0
1
2
3
symboling

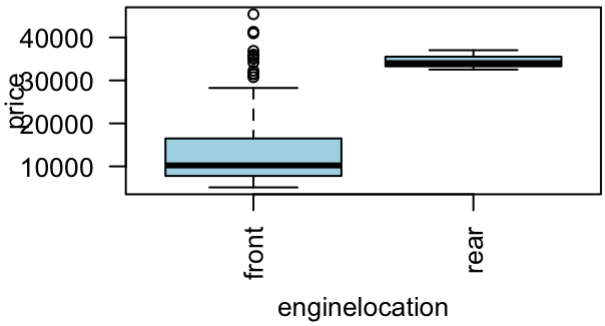

price
40000
30000
20000
10000
diesel
gas
fueltype

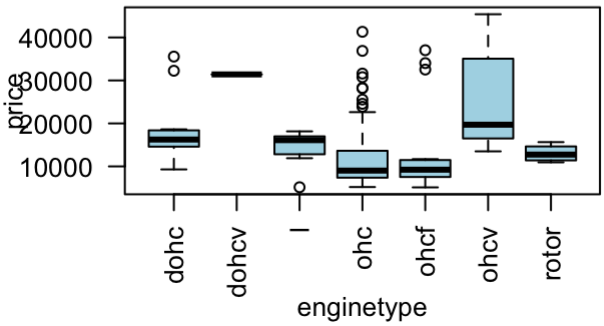

price
40000
30000
20000
10000
std
turbo
aspiration

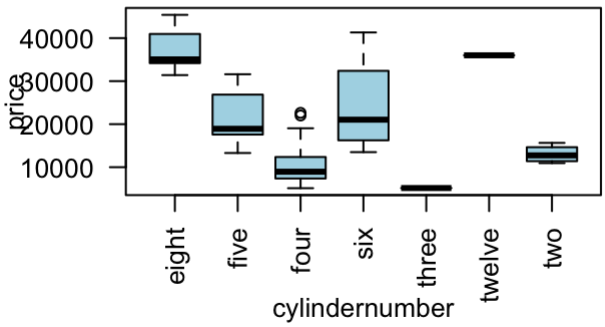

price
40000
30000
20000
10000
four
two
doornumber

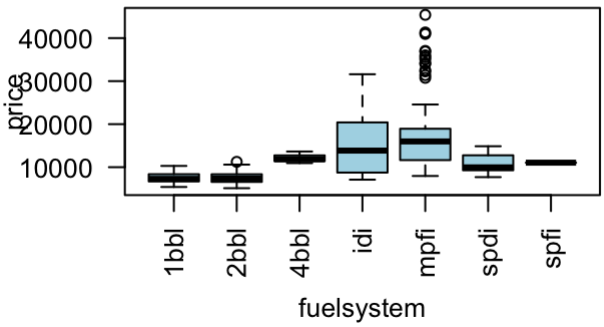

price
40000
30000
20000
10000
convertible
hardtop
hatchback
sedan
wagon
carbody

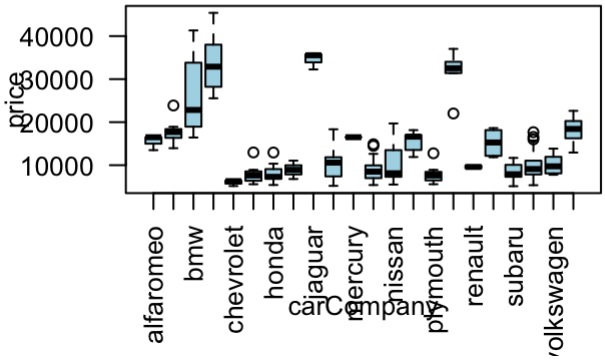

price
40000
30000
20000
10000
4wd
fwd
rwd
drivewheel


price
40000
30000
20000
10000
front
rear
enginelocation


price
40000
30000
20000
10000
dohc
dohcv
l
ohc
ohcf
ohcv
rotor
enginetype







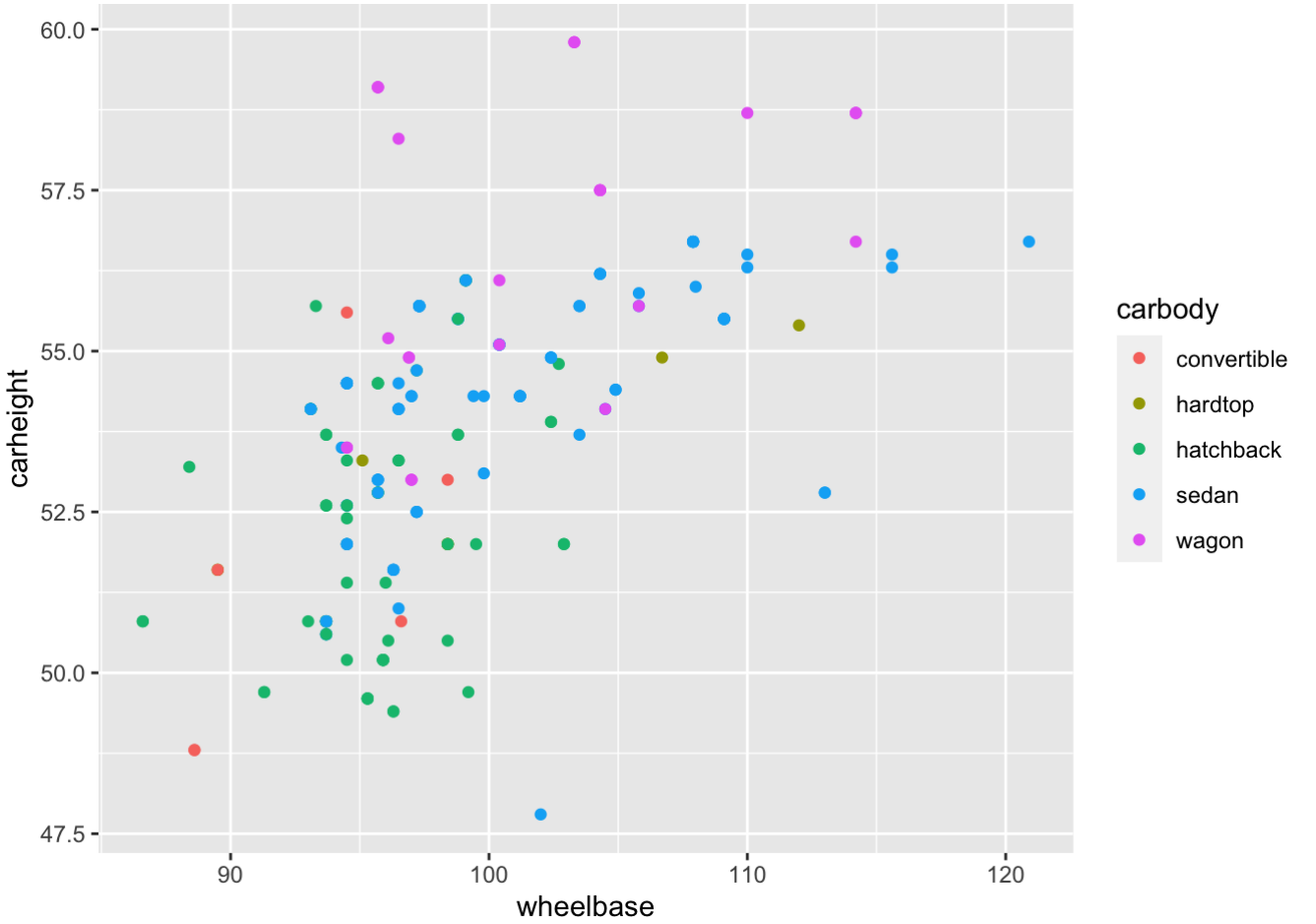
Among all the box-plots, we consider some of them worth analysing:

- the box-plot that visualizes the distribution of car prices across different car companies. We observe that each car company has a distinct range of prices. For example, Porsche, BMW, Jaguar, and Buick are associated with higher prices, while Plymouth, Mazda, Nissan, Dodge, and Chevrolet tend to have comparatively lower prices. This disparity in price ranges suggests that certain car companies position their vehicles in the premium or luxury segment, while others focus on more affordable options.
- the box-plot that visualizes the distribution of car prices across different engine locations: the engine location rear is associated with high prices, while the location front is associated with relatively higher prices. We observe that the “rear” engine location has a higher median and smaller interquartile range (IQR), and whiskers compared to the “front” engine location. This suggests that cars with rear engine locations tend to have higher prices.
- Also the box-plots relatives to engine type, cylinder number and fuel system are interesting, since different values appears to be associated with different price ranges.

The following scatter plots showcases the distribution of car lengths based on their wheel base The color of the points differentiates between car body types.

[code section: 10]

```
ggplot(cleaned_car_prices, aes(x = wheelbase, y = carheight, col = carbody)) + geom_point()
```



We noticed that different car-bodies are well recognisable in the above plot. We used this graph as a starting point for the clustering section.

Feature engineering

We focused in feature engineering to uncover valuable insights from the data-set.

- we have created an engineered feature called “ledge” by subtracting the “wheelbase” variable from the “carlength”. This new feature represents the overhang of the car beyond its wheelbase, which provides valuable insights into the proportions and design characteristics of the cars in our data-set.
- we have identified a high correlation between the “highwaympg” and “citympg” variables, which was expected since both variables capture similar information regarding fuel efficiency. Including both of them in our models might introduce redundancy. We have decided to create an engineered variable called “mpg”, which is computed as the mean of the “highwaympg” and “citympg” variables.

We also plotted scatter-plots that visualize the relationship between the two engineered features and the car prices, grouped by car body type.

[code section: 11] !PARTIALLY REMOVED FROM VIEW! *plots were removed*

```
eng_car_prices <- fact_car_prices
eng_car_prices$ledge = fact_car_prices$carlength-fact_car_prices$wheelbase
encoded_car_prices$ledge = eng_car_prices$ledge
cor(eng_car_prices$price, eng_car_prices$ledge)
```

```
## [1] 0.6465369
```

```
eng_car_prices$mpg <- rowMeans(eng_car_prices[,c("highwaympg", "citympg")], na.rm=TRUE)
encoded_car_prices$mpg = eng_car_prices$mpg
cor(eng_car_prices$price, eng_car_prices$mpg)
```

```
## [1] -0.6968352
```

```
numericalVariables <- append(numericalVariables, c("ledge", "mpg"), after = length(numericalV
ariables))
numericalRegressors <- append(numericalRegressors, c("ledge", "mpg"), after = length(numerica
lVariables))
numericalRegressors <- numericalRegressors[numericalRegressors != "highwaympg" & numericalReg
ressors != "citympg"]
```

We noticed that there is a significant correlation between price and ledge (0.65), and their relationship appears to be almost exponential Instead, between price and mpg we observed a negative correlation (-0.70), also they relationship appears to be non-linear. Regarding the distribution of different carbodies on the plot, we observe that they appear to be scattered with some patterns, however barely distinguishable.

Normalization, scaling and outlier removal of the data

To uderstand if our data is skewed, we plotted box-plots of the numerical variables.

[code section: 12] !REMOVED FROM SUMMARY!

The different variables appear to have very few outliers, but since by a deeper manual analysis they don't appear to be mistakes, nor they represent a threat for modelling quality, we decided not to remove them from our already small data-set.

We also implemented a mathematical analysis the check for outliers, by implementing the function "out_id_iqr", which identifies them basing on Inter-Quantile Ranges. We run the function for each of our numerical variables, and confirmed the decision to not remove outliers.

[code section: 13] !REMOVED FROM SUMMARY!

By analysing the plots of [code section: 12], we concluded that scaling the data could be useful to perform some analysis. We saved the scaled data in a tailored data-frame, that we used only for specific modelling algorithms, naming neural networks and clustering.

[code section: 14]

```
norm_car_prices <- eng_car_prices[,c(numericalRegressors, "price", "ledge", "mpg")]
norm_car_prices <- as.data.frame(scale(norm_car_prices, center = apply(norm_car_prices, 2, mi
n), scale = apply(norm_car_prices, 2, max) - apply(norm_car_prices, 2, min)))

priceNormCoeff1 <- (max(eng_car_prices$price) - min(eng_car_prices$price))
priceNormCoeff2 <- min(eng_car_prices$price)

norm_car_prices <- cbind(norm_car_prices, select(encoded_car_prices, -c(numericalVariables,
"price", "car_ID", "CarName", "highwaympg", "citympg")))
NNregressors <- colnames(norm_car_prices)
```

Task 1: Regression

> Our aim is to predict the price of the new-designed car according to its specifics, to do so, we are going to build a predictive model for the sale prices of cars based on their characteristics. We will reach this goal by trying different approaches [code sections: 15 -> 26], starting with some lower-dimensional models and then moving to more complex ones.

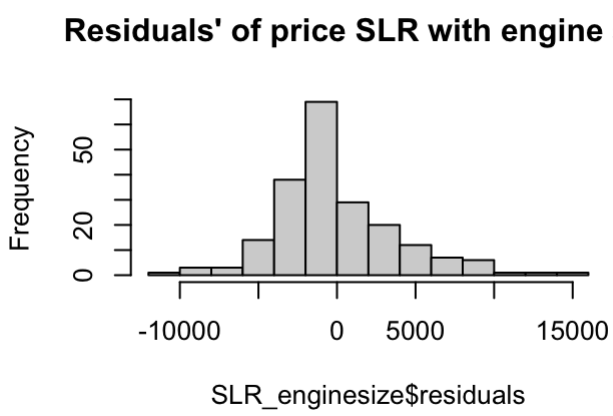
Lower-diensional linear models

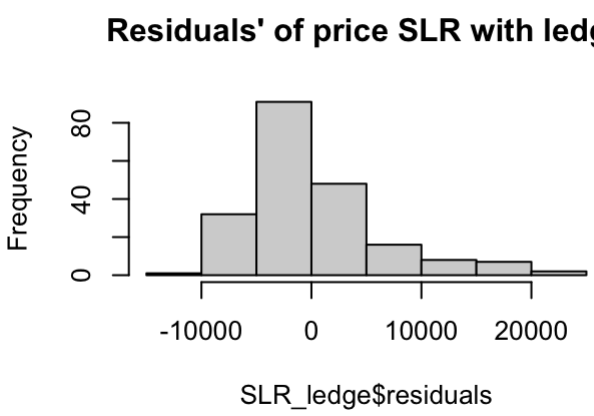
In the code below, we trained single linear regression models for different variables of our data-set in order to investigate correlation and gain insights. We considered:

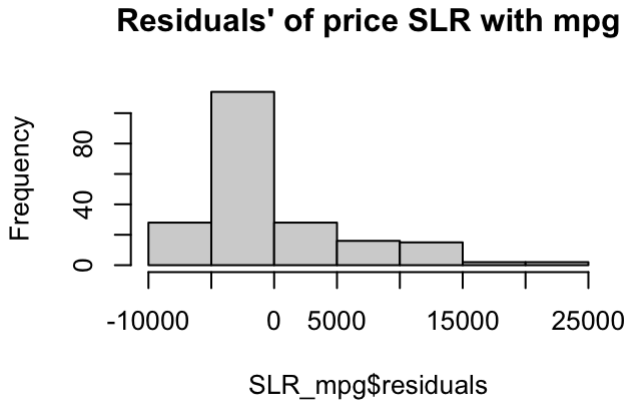
- the numerical ones that have a correlation with price > |0.75|: enginesize, curbweight, horsepower, carwidth.
- the engineered variables: ledge and mpg.
- the categorical variables that resulted to be more interesting from the box-plots analysis: carCompany, enginelocation, enginetype, cylindernumber, and fuelsystem.

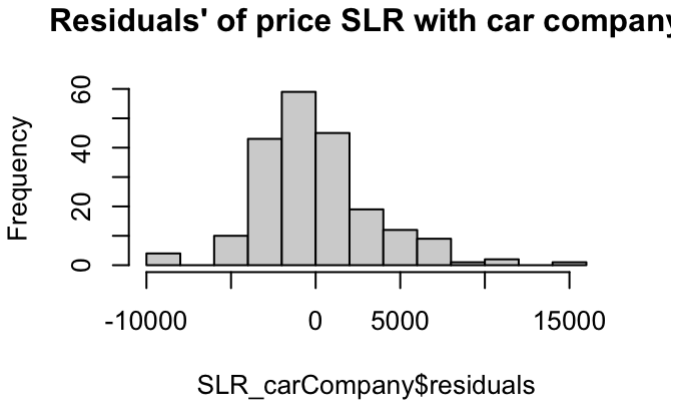
[code section: 15] !PARTIALLY REMOVED FROM SUMMARY! (only interesting plots were kept)

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

SLR_enginesize = lm(formula = price ~ enginesize, data = eng_car_prices)
SLR_enginesize_hist <- hist(SLR_enginesize$residuals, main = paste("Residuals' of price SLR w
ith engine size"))

SLR_ledge = lm(formula = price ~ ledge, data = eng_car_prices)
SLR_ledge_hist <-hist(SLR_ledge$residuals, main = paste("Residuals' of price SLR with ledg
e"))

SLR_mpg = lm(formula = price ~ mpg, data = eng_car_prices)
SLR_mpg_hist <-hist(SLR_mpg$residuals, main = paste("Residuals' of price SLR with mpg"))

SLR_carCompany = lm(formula = price ~ carCompany, data = eng_car_prices)
SLR_carCompany_hist <- hist(SLR_carCompany$residuals, main = paste("Residuals' of price SLR w
ith car company"))
```









Almost every single linear regression's residuals distribution appear to be quite normal (with the exception of the ones based on fuel system and engine location), but the skewness is always more accentuated to the right.

Data split into train and test

We split the data-set randomly into train (containing 80% of observations) and test sets (containing 20% of observations). This approach of splitting the data allows for independent evaluation of the model's performance on unseen data: the training data is used to fit and optimize the model, while the test data serves as an unbiased measure of the model's performance.

[code section: 16]

```
training_samples <- eng_car_prices$price %>% createDataPartition(p = 0.8, list = FALSE)
train_data <- eng_car_prices[training_samples,]
test_data <- eng_car_prices[-training_samples,]

norm_train_data <- norm_car_prices[training_samples,]
norm_test_data <- norm_car_prices[-training_samples,]
```

Initial selection of regressors and creation of performance dataframe

We created a list of the regressors that will be employed in the following models. We are using all the provided variables except from:

- car_ID and CarName, since it does not make sense to use them for forecasting price
- highwaympg and citympg to avoid redundacy, and used instead the engineerd feature which capture similar information regarding fuel efficiency.

- we kept the variables carlength and wheelbase, even if they were used to create the engineerd feature ledge, because of the hierarchical principle, which states that if we include an interaction in a model, we should also include the main effects, even if the p-values associated with their coefficients are not significant.

[code section: 17]

```
regressorsList <- colnames(eng_car_prices)
regressorsList <- regressorsList[!regressorsList == "car_ID" & !regressorsList == "CarName" &
!regressorsList == "price" & !regressorsList == "highwaympg" & !regressorsList == "citympg"]
```

We created also a data-frame to keep track of the best models we created and theirs characteristics.

[code section: 18]

```
bestPerfDFCols = c("Record",  "ModelName", "RegressorsNum", "AIC", "BIC", "lambda", "alpha",
"TreesNum", "LayersNum",  "MSEtrain", "RMSEtrain", "MSEtest", "RMSEtest", "R2", "Model")

best_perf_df = data.frame(matrix(nrow = 0, ncol = length(bestPerfDFCols)))
colnames(best_perf_df) = bestPerfDFCols
```

AIC and BIC stepwise selection for multivariate linear models

We started by constructing multiple linear regression models using step-selection based on the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC); we provided the different functions: forward, backward, and forward/backward (both), for both AIC and BIC, with the entire set of variables (including numerical, engineered, and categorical; excluding the Car ID and Car Name), allowing them to autonomously determine the optimal predictors to retain.

[code section: 19] !PARTIALLY REMOVED FROM SUMMARY! *only one AIC model and two BIC models were saved since models were the same*

```
AICBICModelsList <- list()
intercept_only <- lm(price ~ 1, data = train_data[,c("price", regressorsList)])
all <- lm(price ~ ., data = train_data[,c("price", regressorsList)])

AIC_forward <- step(intercept_only, direction = "forward", scope = formula(all), trace = 0)
AIC_backward<- step(all, direction = "backward", scope = formula(all), trace = 0)
AIC_both <- step(intercept_only, direction = "both", scope = formula(all), trace = 0)

new_row <- data.frame(Record = NA, ModelName = "AIC forward - backward - forward/backward sel
ection", RegressorsNum = length(coef(AIC_forward)) - 1, AIC = AIC(AIC_forward), BIC = BIC(AIC
_forward), lambda = NA, alpha = NA, TreesNum = NA, LayersNum = NA, MSEtrain = mse(AIC_forwar
d, train_data), RMSEtrain = rmse(AIC_forward, train_data), MSEtest = mse(AIC_forward, test_da
ta), RMSEtest = rmse(AIC_forward, test_data), R2 = summary(AIC_forward)$r.squared, Model = N
A)
```

```
## Warning in predict.lm(model, data): prediction from a rank-deficient fit may be
## misleading

## Warning in predict.lm(model, data): prediction from a rank-deficient fit may be
## misleading

## Warning in predict.lm(model, data): prediction from a rank-deficient fit may be
## misleading

## Warning in predict.lm(model, data): prediction from a rank-deficient fit may be
## misleading
```

```
best_perf_df <- rbind(best_perf_df, new_row)
AICBICModelsList[[length(AICBICModelsList)+1]] <- serialize(AIC_forward,NULL)

BIC_forward <- step(intercept_only, direction = "forward", scope = formula(all), trace = 0, k
= log(nrow(train_data)))

BIC_backward <- step(all, direction = "backward", scope = formula(all), trace = 0, k = log(nr
ow(train_data)))
new_row <- data.frame(Record = NA, ModelName = "BIC backward selection", RegressorsNum = leng
th(coef(BIC_backward)) - 1, AIC = AIC(BIC_backward), BIC = BIC(BIC_backward), lambda = NA, al
pha = NA, TreesNum = NA, LayersNum = NA, MSEtrain = mse(BIC_backward, train_data), RMSEtrain
= rmse(BIC_backward, train_data), MSEtest = mse(BIC_backward, test_data), RMSEtest = rmse(BIC
_backward, test_data), R2 = summary(BIC_backward)$r.squared, Model = NA)
best_perf_df <- rbind(best_perf_df, new_row)
AICBICModelsList[[length(AICBICModelsList)+1]] <- serialize(BIC_backward,NULL)

BIC_both <- step(intercept_only, direction = "both", scope = formula(all), trace = 0, k = log
(nrow(train_data)))
new_row <- data.frame(Record = NA, ModelName = "BIC forward - backward selection", Regressors
Num = length(coef(BIC_forward)) - 1, AIC = AIC(BIC_forward), BIC = BIC(BIC_forward), lambda =
NA, alpha = NA, TreesNum = NA, LayersNum = NA, MSEtrain = mse(BIC_forward, train_data), RMSEt
rain = rmse(BIC_forward, train_data), MSEtest = mse(BIC_forward, test_data), RMSEtest = rmse
(BIC_forward, test_data), R2 = summary(BIC_forward)$r.squared, Model = NA)
best_perf_df <- rbind(best_perf_df, new_row)
AICBICModelsList[[length(AICBICModelsList)+1]] <- serialize(BIC_forward,NULL)

best_perf_df$Model <- AICBICModelsList
```

Looking at the results, we notice that models selected by the three selection methods using AIC have more regressors with respect to the models selected using AIC. This is expected, because the BIC method tends to choose models with fewer regressors than the AIC because it applies a more substantial penalty for model complexity. The best R^2 (0.97) is reached by the model choosen by the AIC forward selection method.

We proceeded by implementing penalized approach algorithms to find models with good generalization capabilities. We will implement the function that performs Elastic Net regression, performed by the formula:

$$\frac{\sum_{i=1}^{n}(y_i - x_i^J\hat{\beta})^2}{2n} + \lambda\left(\frac{1-\alpha}{2}\underbrace{\sum_{j=1}^{m}\hat{\beta}_j^2}_{ridge} + \alpha\underbrace{\sum_{j=1}^{m}|\beta_j|}_{lasso}\right)$$

Elastic Net regression is a hybrid approach that blends both penalizations of the lasso and ridge methods, for this reason, will tune the alpha parameter of the formula above (where alpha = 0 corresponds to ridge regression and alpha = 1 corresponds to lasso regression), choosing an alpha value between 0 and 1 to optimize the Elastic Net, to find a good lambda hyper-parameter.

[code section: 20]

```
EN_perf_df = data.frame(matrix(nrow = 0, ncol = length(bestPerfDFCols)))
colnames(best_perf_df) = bestPerfDFCols
ENModelsList <- list()

for (alpha in seq(0, 1, by=0.05)) {
  tempCVModel <- cv.glmnet(data.matrix(train_data[,regressorsList]), train_data$price, alpha
= alpha)
  tempModel <- glmnet(data.matrix(train_data[,regressorsList]), train_data$price, alpha = alp
ha, lambda = tempCVModel$lambda.min)

  ENModelsList[[length(ENModelsList)+1]] <- serialize(tempModel,NULL)
  tempMSEtrain = sum((train_data$price - predict(tempModel, newx = data.matrix(train_data[,re
gressorsList])))^2) / nrow(train_data)
  tempMSEtest = sum((test_data$price - predict(tempModel, newx = data.matrix(test_data[,regre
ssorsList])))^2) / nrow(test_data)
  predicted <- predict(tempModel, newx = data.matrix(train_data[,regressorsList]), s = "lambd
a.min", type = "response")
  tempR2 <- (cor(predicted, train_data$price))^2

  EN_perf_df <- rbind(EN_perf_df, data.frame(Record = NA, ModelName = "EL", RegressorsNum = N
A, AIC = NA, BIC = NA, lambda = tempCVModel$lambda.min, alpha = alpha, TreesNum = NA, LayersN
um = NA, MSEtrain = tempMSEtrain, RMSEtrain = sqrt(tempMSEtrain), MSEtest = tempMSEtest, RMSE
test = sqrt(tempMSEtrain), R2 = tempR2, Model = NA))}

EN_perf_df$Model <- ENModelsList

newRow = EN_perf_df[which.min(EN_perf_df$MSEtrain),]
newRow$Record <- "Elastic Network lower MSE error on train"
best_perf_df <- rbind(best_perf_df, newRow)

newRow = EN_perf_df[which.min(EN_perf_df$MSEtest),]
newRow$Record <- "Elastic Network lower MSE error on test"
best_perf_df <- rbind(best_perf_df, newRow)
```

The errors of the best Elastic Net model still result a bit higher with respect to the ones choosen with the previous methods (BIC and AIC).

Random forests

We kept going with our research of the best model to forecast car prices basing on car characteristics; this time we employed a Random Forest algorithm.

We selected a list of regressors to include in the Random Forest model, by choosing:

- the most correlated quantitative regressors that were not highly correlated with each other or used in feature engineering;
- the categorical variables that showed strong relationships with the target variable, following the same criteria mentioned in the Lower-dimensional linear models section;
- the engineered variables.

[code section: 21]

```
RFregressors = c("enginesize", "horsepower", "curbweight", "ledge", "mpg", "carCompany", "eng
inelocation", "enginetype", "cylindernumber", "fuelsystem")
```

We developed an algorithm that generates and compares the possible Random Forest models considering all the combinations of aforementioned regressors with a maximum of 30 trees. As for the previous methods, we saved in the best performance data-frame the best performing models in terms of train and test residuals.

[code section: 22]

```
RF_perf_df = data.frame()
RFModelsList <- list()

for (tempVarNum in c(1:length(RFregressors))){
  tempVarCombinations <- (combn(RFregressors, tempVarNum, fun=NULL, simplify=FALSE))
  for (tempVarComb in tempVarCombinations){
    for (tempTreeNum in c(1:30)){
      tempModel = randomForest(formula = as.formula(paste("price ~ ", paste(unlist(tempVarCom
b), sep ="", collapse = " + "), sep = "")),
      data = train_data, ntree = tempTreeNum, keep.forest = TRUE, importance = TRUE)

      RFModelsList[[length(RFModelsList)+1]] <- serialize(tempModel,NULL)
      tempTSS <- sum((mean(test_data$price) - predict(tempModel, test_data))^2)
      tempRSS <- sum((test_data$price - predict(tempModel, test_data))^2)
      tempR2 <- 1 - (tempRSS / tempTSS)
      RF_perf_df <- rbind(RF_perf_df, data.frame(Record = NA, ModelName = paste("RF: ", paste
(unlist(tempVarComb), sep = "", collapse = ", "), sep = ""), RegressorsNum = tempVarNum, AIC
= NA, BIC = NA, lambda = NA, alpha = NA, TreesNum = tempTreeNum, LayersNum = NA, MSEtrain = m
se(tempModel, train_data), RMSEtrain = rmse(tempModel, train_data), MSEtest = mse(tempModel,
test_data), RMSEtest = rmse(tempModel, test_data), R2 = tempR2, Model = NA))}}}
RF_perf_df$Model <- RFModelsList
newRow = RF_perf_df[which.min(RF_perf_df$MSEtrain),]
newRow$Record <- "Random Forest lower MSE error on train"
best_perf_df <- rbind(best_perf_df, newRow)
newRow = RF_perf_df[which.min(RF_perf_df$MSEtest),]
newRow$Record <- "Random Forest lower MSE error on test"
best_perf_df <- rbind(best_perf_df, newRow)
```

We want to print and evaluate the residuals of the best performing models both in the train and test data-sets.

[code section: 23] ! REMOVED FROM SUMMARY!

Even with different run, causing slightly different random forests, the residuals appear to be distributed almost normally. However, the residuals' distribution computed over the test data-set appears to be less normal, due to the fact that only 40 observations belong to this data-set.

Neural networks

In this section, we employed a neural network model to address our predictive modeling task.

[code section: 24]

```
NN_perf_df = data.frame()
NNModelsList <- list()
maxLayers = 2
maxNodesperLayer = 18

for (tempLayersNum in c(1:maxLayers)){
  tempHiddenCombinations <- permutations(n=maxNodesperLayer,r=maxLayers,v=c(1:maxNodesperLaye
r),repeats.allowed=T)

  for (tempHidden in tempHiddenCombinations){
    tempModel <- neuralnet(formula = as.formula(paste("price ~ ", paste(unlist(NNregressors),
sep ="", collapse = " + "), sep = "")), data = norm_train_data, hidden = tempHidden, linear.o
utput = TRUE)

    NNModelsList[[length(NNModelsList)+1]] <- serialize(tempModel,NULL)
    tempMSEtrain = sum((train_data$price - (compute(tempModel, norm_train_data)$net.result *
priceNormCoeff1 + priceNormCoeff2))^2)/nrow(norm_test_data)
    tempMSEtest = sum((test_data$price - (compute(tempModel, norm_test_data)$net.result * pri
ceNormCoeff1 + priceNormCoeff2))^2)/nrow(norm_test_data)
    tempTSS <- sum((mean(test_data$price) - (compute(tempModel, norm_test_data)$net.result *
priceNormCoeff1 + priceNormCoeff2))^2)
    tempRSS <- sum((test_data$price - (compute(tempModel, norm_test_data)$net.result * priceN
ormCoeff1 + priceNormCoeff2))^2)
    tempR2 <- 1 - (tempRSS / tempTSS)
    NN_perf_df <- rbind(NN_perf_df, data.frame(Record = NA, ModelName = "NN: all regressors",
RegressorsNum = 25, AIC = NA, BIC = NA, lambda = NA, alpha = NA, TreesNum = NA, LayersNum = t
empLayersNum, MSEtrain = tempMSEtrain, RMSEtrain = sqrt(tempMSEtrain), MSEtest = tempMSEtest,
RMSEtest = sqrt(tempMSEtest), R2 = tempR2, Model = NA))}}
NN_perf_df$Model <- NNModelsList
newRow = NN_perf_df[which.min(NN_perf_df$MSEtrain),]
newRow$Record <- "Neural Network lower MSE error on train"
best_perf_df <- rbind(best_perf_df, newRow)
newRow = NN_perf_df[which.min(NN_perf_df$MSEtest),]
newRow$Record <- "Neural Network lower MSE error on test"
best_perf_df <- rbind(best_perf_df, newRow)
```

We printed the residuals of the best performing neural networks both in the train and test data-sets. We also plotted a visualization of the best performing neural network on test data.

[code section: 25] !PARTIALLY REMOVED FROM SUMMARY! *only the plot of the neural network was kept*

```
plot(tempModel)
plot(test_data$price, compute(tempModel, norm_test_data)$net.result * priceNormCoeff1 +priceN
ormCoeff2, col = "red", main = 'Real vs Predicted')
abline(0, 1, lwd = 2, col = "red")
```

The residuals appear to be distributed quite normally, and particularly concentrated around zero. In the plot of the neural network we clearly observe that is made by one layer and one neuron.

To conclude this regression task, we printed the data-frame containing all of our best performing models and their characteristics.

[code section: 26] !REMOVED FROM SUMMARY!

Conclusions

Looking at the data-frame, we drow the following conclusions:

- following the AIC metric, the best performing models are the three models selected with the AIC step-wise forward, backward, and forward/backward selections. The three models, however resulting from three different kind of selection, are the same one. They are multivariate linear regressions which exploits 46 different regressors.
- basing on the BIC metric, the best performing models are the two models selected with the BIC step-wise forward and forward/backward selections. The two models are actually the same one. They are multivariate linear regressions which exploits 31 different regressors. It is worth noting that the BIC backward step-wise selection method resulted in a worse model, containing more regressors (34) and an higher BIC value: this is because the algorithm got stuck in a local minima of the BIC value, and did not proceed in looking for best performing models.
- looking at the R^2, the best model appear to be a neural network, specifically the one with lowest mse on the test data, with a score > 0.99 (definitely good!). At the second place we find the multivariate linear regression choose by the AIC step-wise methods.
- basing on the mse/rmse on test, the best performing model appears to be again the neural network, with an approximate error of only 600 dollars. The second best model is the random forest (obviously, the one with the lowest mse error): the exact parameters of this model changes each time with run the script, since it has a random component, but we can say that it will always have a number of trees that ranges at most between 20 and 30, and contains only few regressors (probably choosing the combinations with more of them results in overfitting the train data).

The penalised-approach models (elastic networks) did not achieved best scoring performance in any of the fields we took under consideration, but it’s worth noting that, among all the elastic networks models we tried, the best performing ones are a lasso regression (best mse on train) and another one with a relatively high alpha value of 0.7 (best mse on test)

To summarize, the best model that we found is a neural network, made of layer and one neuron (Steve). We are also happy about the results of the Random Forests model and the AIC selected model performances.

Clustering

In this task we experimented various clusterizations of cars basing on all the available covariates excluding model and carbody. The objective of the proposed models is to give advice to a company about marketing and policy making, as car market is strongly segmented and it is crucial to understand how a car would be categorized.

We compared both k-means and hierarchical methods, also focusing on visualizing the resulting clusterizations in lower dimensional spaces and its statistics, in order to compare the obtained divisions with already available labels, therefore understanding the main variable influencing the automatic segmentation.

[code section: 27]

```
cluster_car_prices <- cbind(norm_car_prices[,c(numericalVariables[numericalVariables != "city
mpg" & numericalVariables != "highwaympg"]),], cleaned_car_prices[categoricalVariables])
```

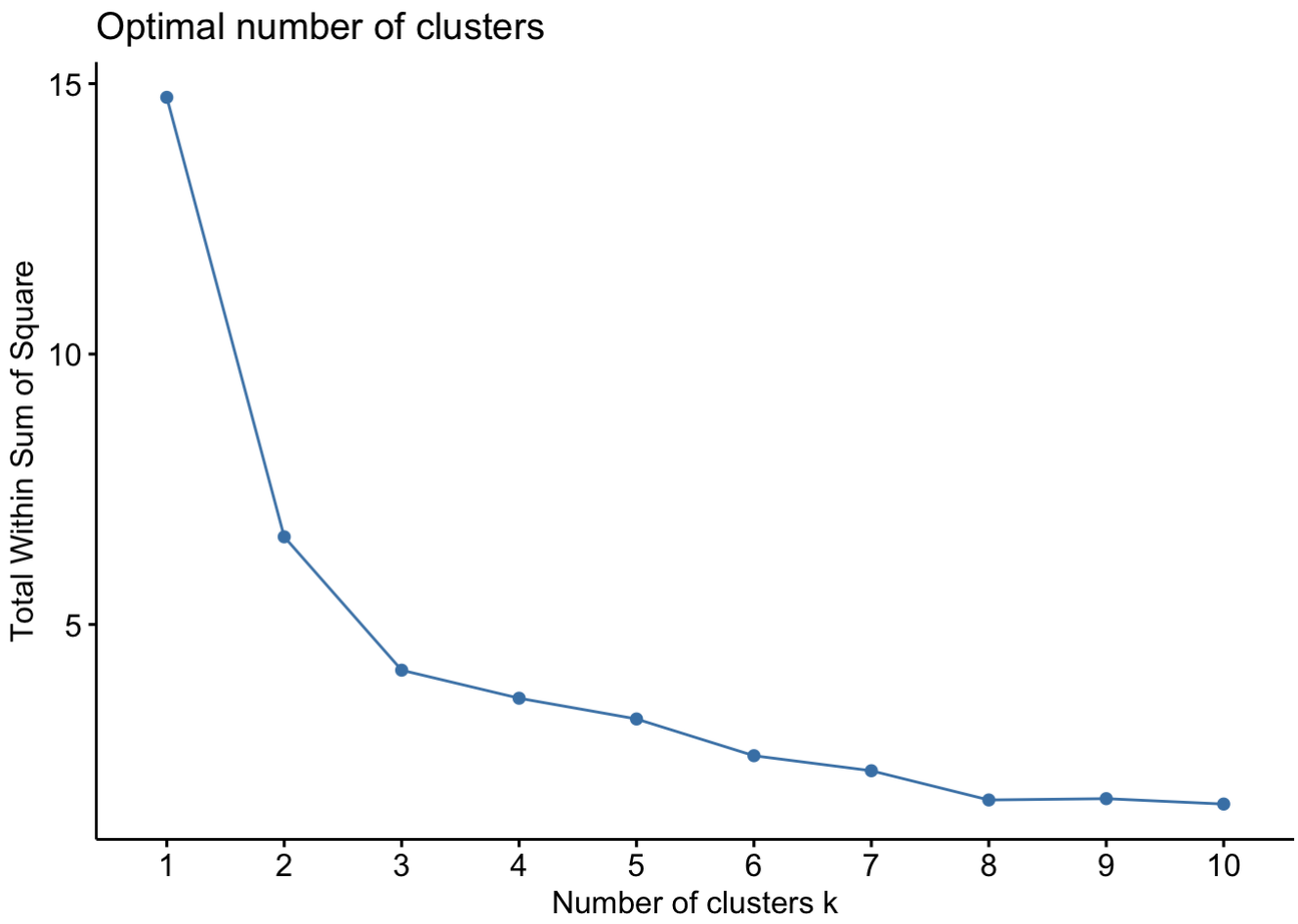
k-Means clustering

For k-means models, we always picked the best number of clusters basing on WSS, Silouhette and Gap-Statistic analyses' outputs.

Firstly, we focused on lower-dimensional models and found particularly interesting the simple clustering basing on wheel base and car height, logically appearing as good indicators of the car size.

[code section: 28] !PARTIALLY REMOVED FROM VIEW!

```
fviz_nbclust(cluster_car_prices[,c("wheelbase", "carheight")], FUNcluster = kmeans, method =
"wss", iter.max=30)
```

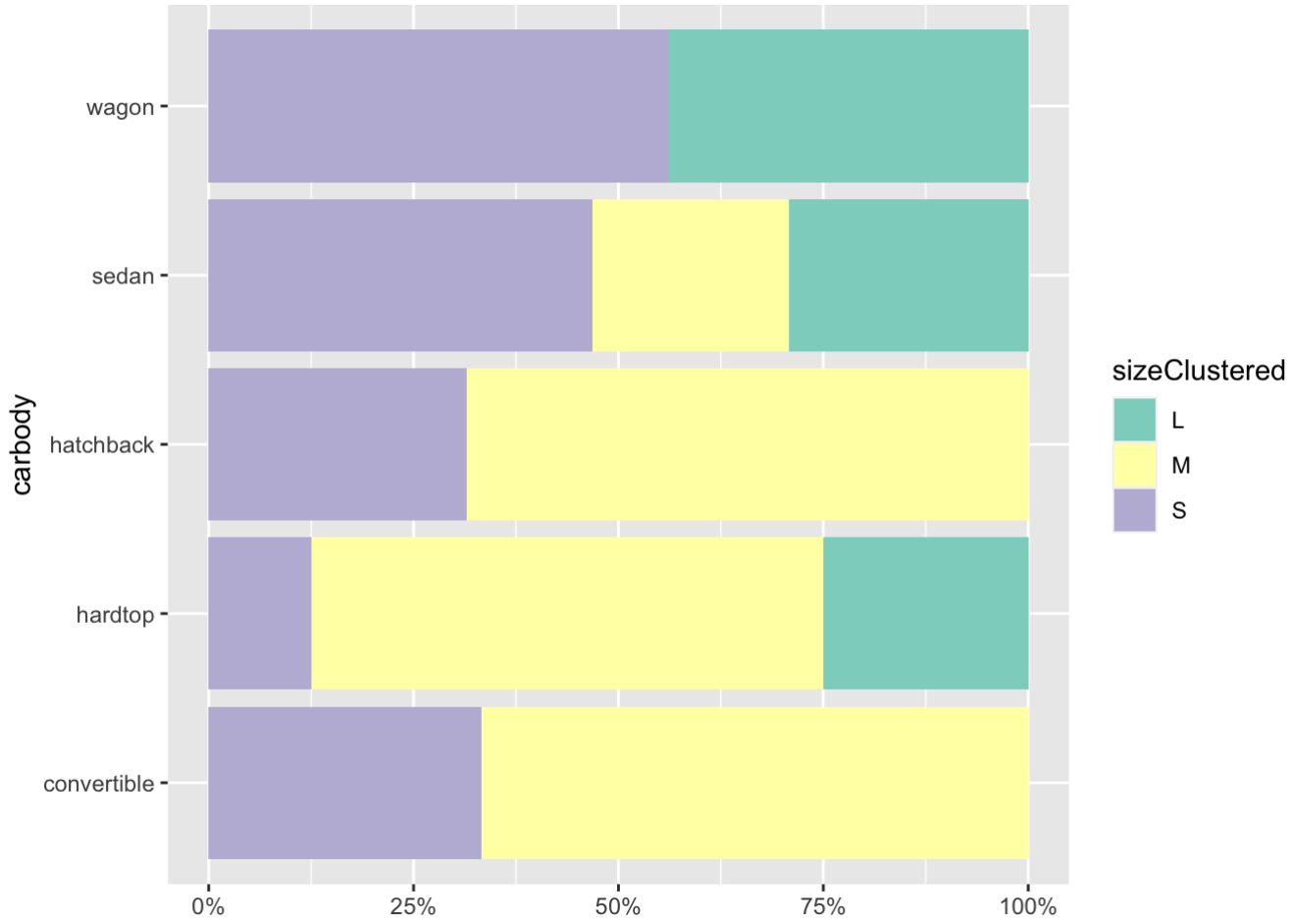



```
tempClustersNum = 3

kMeans1 <- kmeans(cluster_car_prices[,c("wheelbase", "carheight")], centers = tempClustersNu
m, nstart=50, iter.max=100)
tab_kMeans1 = table(cluster_car_prices$carbody, kMeans1$cluster)
pander(prop.table(tab_kMeans1, 1), caption = "Table")
```

Table

	1	2	3
convertible	0.3333	0	0.6667
hardtop	0.125	0.25	0.625
hatchback	0.3143	0	0.6857
sedan	0.4688	0.2917	0.2396
wagon	0.56	0.44	0

```
cluster_car_prices$sizeClustered = kMeans1$cluster
levels(cluster_car_prices$sizeClustered) <- c(levels(cluster_car_prices$sizeClustered), "S",
"M", "L")
cluster_car_prices$sizeClustered[cluster_car_prices$sizeClustered == '1'] <- "S"
cluster_car_prices$sizeClustered[cluster_car_prices$sizeClustered == '2'] <- "L"
cluster_car_prices$sizeClustered[cluster_car_prices$sizeClustered == '3'] <- "M"

cluster_car_prices %>% select(carbody, sizeClustered) %>% group_by(carbody, sizeClustered) %
>% count() %>% ggplot(aes(x = carbody, y = n, fill = sizeClustered)) + geom_col(position = "f
ill") + scale_fill_brewer(palette = "Set3") + scale_y_continuous(labels = scales::percent) +
labs(y = NULL) + coord_flip()
```

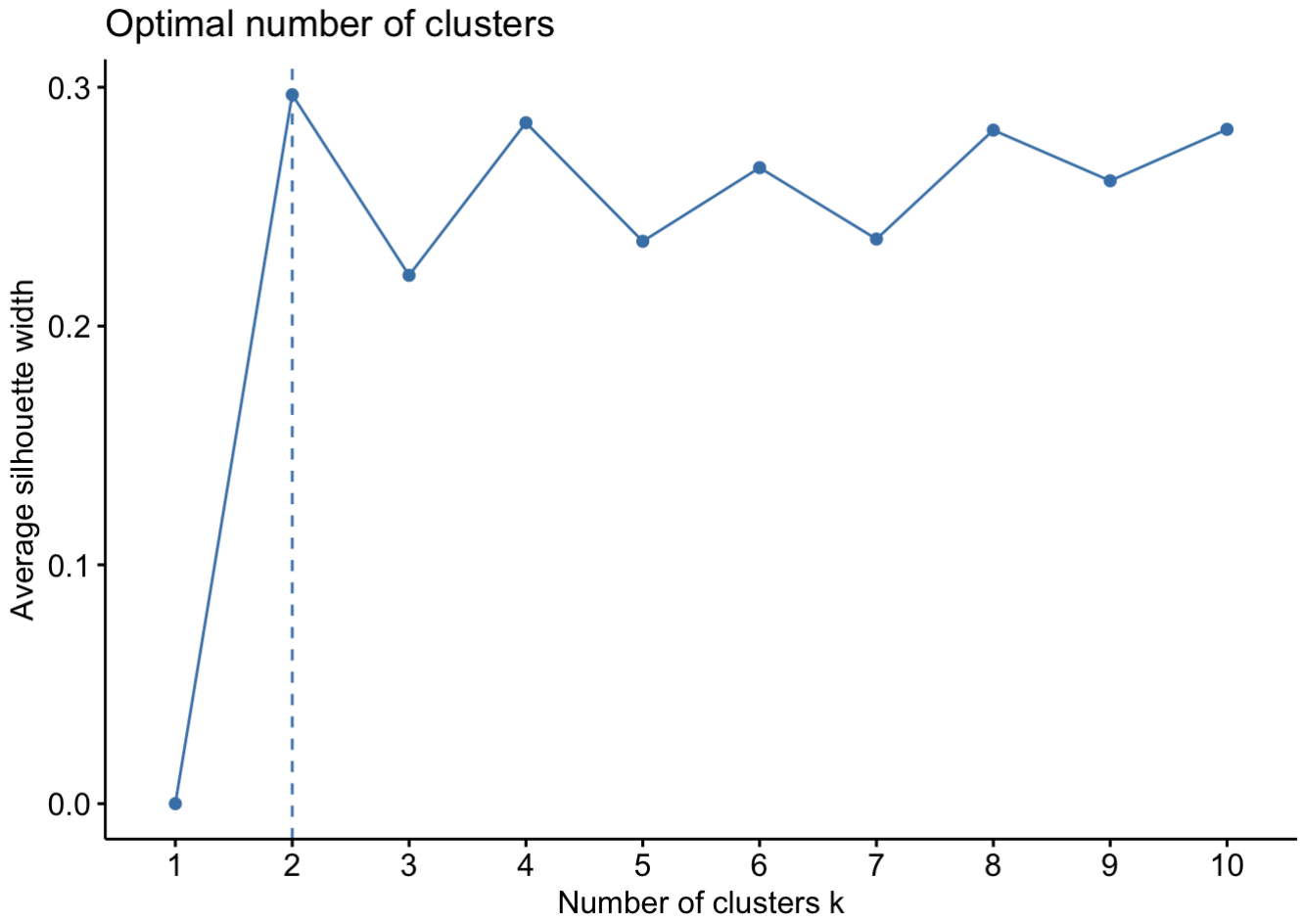


As expected, we obtained a good clusterization of car size, higly correlated with the car body label.

Now, we focus on a k-means clustering basing on all the numerical variables and on the best clusters' number we esteemed with the three main tools we knew.

[code section: 29] !PARTIALLY REMOVED FROM VIEW!

```
fviz_nbclust(cluster_car_prices[numericalRegressors], FUNcluster = kmeans, method = "silhouet
te", iter.max=30 )
```



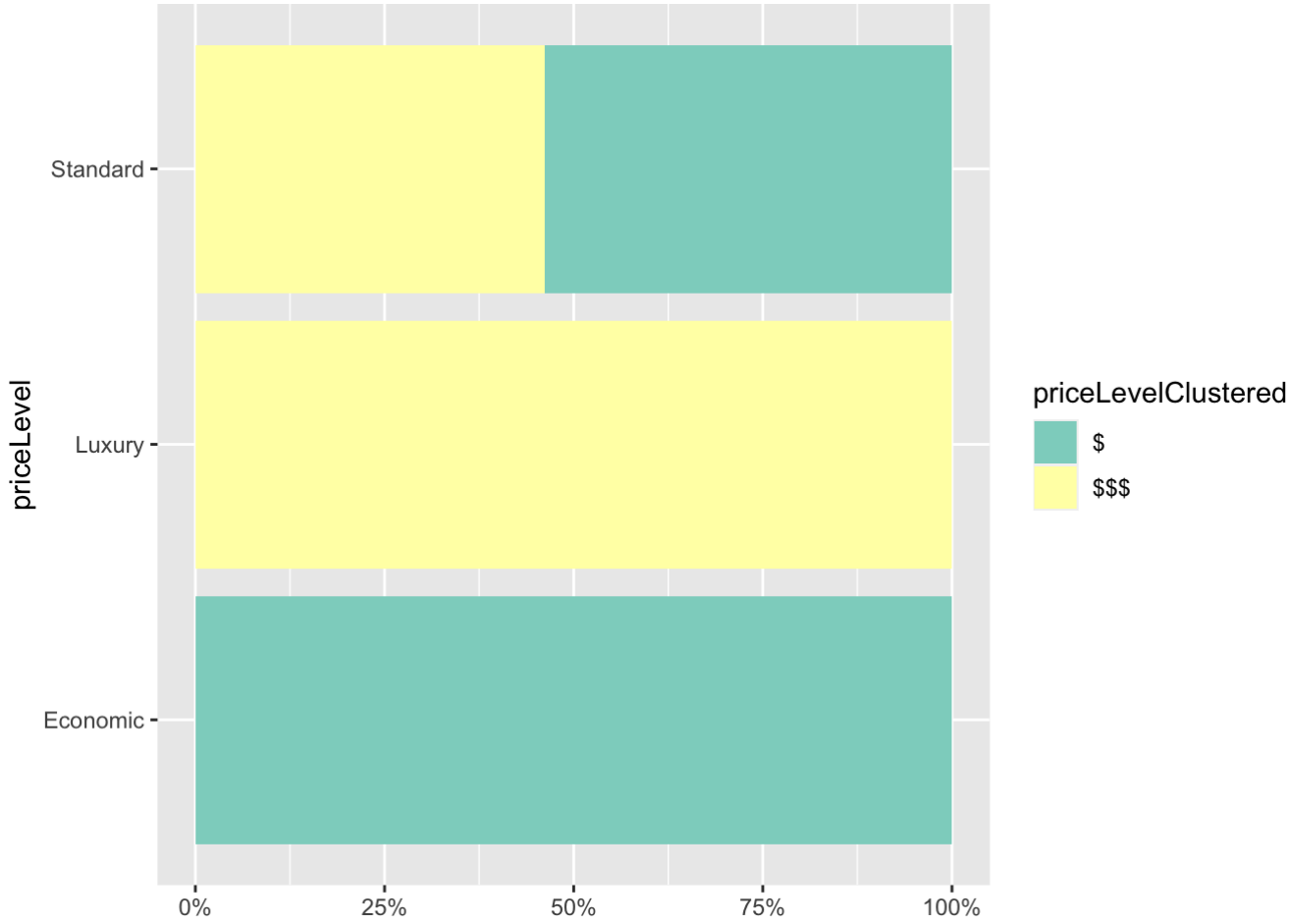

```
tempClustersNum = 2

kMeans2 <- kmeans(cluster_car_prices[numericalRegressors], centers = tempClustersNum, nstart
= 10, iter.max=100)

cluster_car_prices$priceLevel[cluster_car_prices$price >= 0.33] <- "Luxury"
cluster_car_prices$priceLevel[cluster_car_prices$price < 0.33 & cluster_car_prices$price >=
0.08] <- "Standard"
cluster_car_prices$priceLevel[cluster_car_prices$price < 0.08] <- "Economic"

fviz_cluster(kMeans2, data = cluster_car_prices[numericalRegressors]) + theme_minimal() + ggt
itle("Title")
```


```
tab_kMeans2 = table(cluster_car_prices$priceLevel, kMeans2$cluster)
pander(prop.table(tab_kMeans2, 1), caption = "Table")
```

Table

	1	2
Economic	0	1
Luxury	1	0
Standard	0.4623	0.5377

```
cluster_car_prices$priceLevelClustered = kMeans2$cluster

levels(cluster_car_prices$priceLevelClustered) <- c(levels(cluster_car_prices$priceLevelClust
ered), "$$$", "$")
cluster_car_prices$priceLevelClustered[cluster_car_prices$priceLevelClustered == '1'] <-
"$$$"
cluster_car_prices$priceLevelClustered[cluster_car_prices$priceLevelClustered == '2'] <- "$"

cluster_car_prices %>% select(priceLevel, priceLevelClustered) %>% group_by(priceLevel, price
LevelClustered) %>% count() %>%
  ggplot(aes(x = priceLevel, y = n, fill = priceLevelClustered)) +
  geom_col(position = "fill") +
  scale_fill_brewer(palette = "Set3") +
  scale_y_continuous(labels = scales::percent) +
  labs(y = NULL) +
  coord_flip()
```



We obtained a clusterization mainly related with the price level of the cars.

Hierarchical clustering

For hierarchical models, we used Euclidean, Correlation-based and Gower distances, always exploiting Ward.D2 and Complete methods for the actual clusterization in a number of groups we chose basing on the dendogram's heights.

Firstly, we obtained an Euclidean-distance-based hierarchical clusterization.

[code section: 30] !PARTIALLY REMOVED FROM VIEW!

```
distanceMatrixEuclidean <- dist(cluster_car_prices[numericalRegressors], method = "euclidea
n")

dendogramWardD2_Euclidean <- hclust(distanceMatrixEuclidean, method = "ward.D2")
tempClustersNum = 4
tempGroups <- cutree(dendogramWardD2_Euclidean, k = tempClustersNum)

cluster_car_prices$companyClustered1 = tempGroups
tab_hier_eucl_wd2 = table(cluster_car_prices$carCompany, cluster_car_prices$companyClustered
1)
pander(prop.table(tab_hier_eucl_wd2, 1), caption = "Table")
```

Table

	1	2	3	4
alfaromeo	0.6667	0.3333	0	0
audi	0.4286	0.5714	0	0
bmw	0.625	0.375	0	0
buick	0	0.5	0	0.5
chevrolet	0	0	1	0
dodge	0.1111	0.1111	0.7778	0
honda	0.6154	0	0.3846	0
isuzu	0.25	0	0.75	0
jaguar	0	1	0	0
mazda	0.5294	0.05882	0.2941	0.1176
mercury	0	1	0	0
mitsubishi	0.4615	0.2308	0.3077	0
nissan	0.1111	0.3333	0.5	0.05556
peugeot	0	0.5455	0	0.4545
plymouth	0.1429	0.1429	0.7143	0
porsche	0	1	0	0
renault	1	0	0	0
saab	0	1	0	0
subaru	1	0	0	0
toyota	0.2812	0.3125	0.3125	0.09375
volkswagen	0.6667	0	0	0.3333
volvo	0	0.9091	0	0.09091

```
dendogramComplete_Euclidean <- hclust(distanceMatrixEuclidean, method = "complete")
tempClustersNum = 3
tempGroups <- cutree(dendogramComplete_Euclidean, k = tempClustersNum)

cluster_car_prices$companyClustered2 = tempGroups
tab_hier_eucl_comp = table(cluster_car_prices$carCompany, cluster_car_prices$companyClustered
2)
pander(prop.table(tab_hier_eucl_comp, 1), caption = "Table")
```

Table We obtained a clusterization mainly related to the car company label.

	1	2	3
alfaromeo	1	0	0
audi	1	0	0
bmw	0.625	0.375	0
buick	0	1	0
chevrolet	0	0	1
dodge	0.2222	0	0.7778
honda	0.2308	0	0.7692
isuzu	0.5	0	0.5
jaguar	0	1	0
mazda	0.6471	0	0.3529
mercury	1	0	0
mitsubishi	0.6923	0	0.3077
nissan	0.6111	0.3333	0.05556
peugeot	0.5455	0.4545	0
plymouth	0.2857	0	0.7143
porsche	0	1	0
renault	1	0	0
saab	1	0	0
subaru	1	0	0
toyota	0.8438	0	0.1562
volkswagen	0.5	0	0.5
volvo	0.9091	0.09091	0

Then, we obtained a Correlation-distance-based hierarchical clusterization.

[code section: 31] !PARTIALLY REMOVED FROM VIEW!

```
distanceMatrixCor = as.dist(1 - cor(t(cluster_car_prices[numericalRegressors])))

dendogramWardD2_Cor = hclust(distanceMatrixCor, method="ward.D2")
tempClustersNum = 4
tempGroups <- cutree(dendogramWardD2_Cor, k = tempClustersNum)

cluster_car_prices$fuelSystemClustered1 = tempGroups
tab_hier_cor_wd2 = table(cluster_car_prices$fuelsystem, cluster_car_prices$fuelSystemClustere
d1)
pander(prop.table(tab_hier_cor_wd2, 1), caption = "Table")
```

Table

	1	2	3	4
1bbl	0	1	0	0
2bbl	0.06061	0.6212	0.3182	0
4bbl	1	0	0	0
idi	0	0	0	1
mpfi	0.5263	0.1053	0.3684	0
spdi	0.7778	0.2222	0	0
spfi	1	0	0	0

```
dendogramComplete_Cor = hclust(distanceMatrixCor, method="complete")
tempClustersNum = 4
tempGroups <- cutree(dendogramComplete_Cor, k = tempClustersNum)

cluster_car_prices$fuelSystemClustered2 = tempGroups
tab_hier_cor_comp = table(cluster_car_prices$fuelsystem, cluster_car_prices$fuelSystemCluster
ed2)
pander(prop.table(tab_hier_cor_comp, 1), caption = "Table")
```

Table We obtained a clusterization mainly related to the fuel system label.

	1	2	3	4
1bbl	0	0.1818	0.8182	0
2bbl	0	0.4697	0.5303	0
4bbl	0	0	1	0
idi	0	0	0	1
mpfi	0.3789	0.5368	0.08421	0
spdi	0	0.4444	0.5556	0
spfi	0	1	0	0

Then, we obtained a Gower-distance-based hierarchical clusterization.

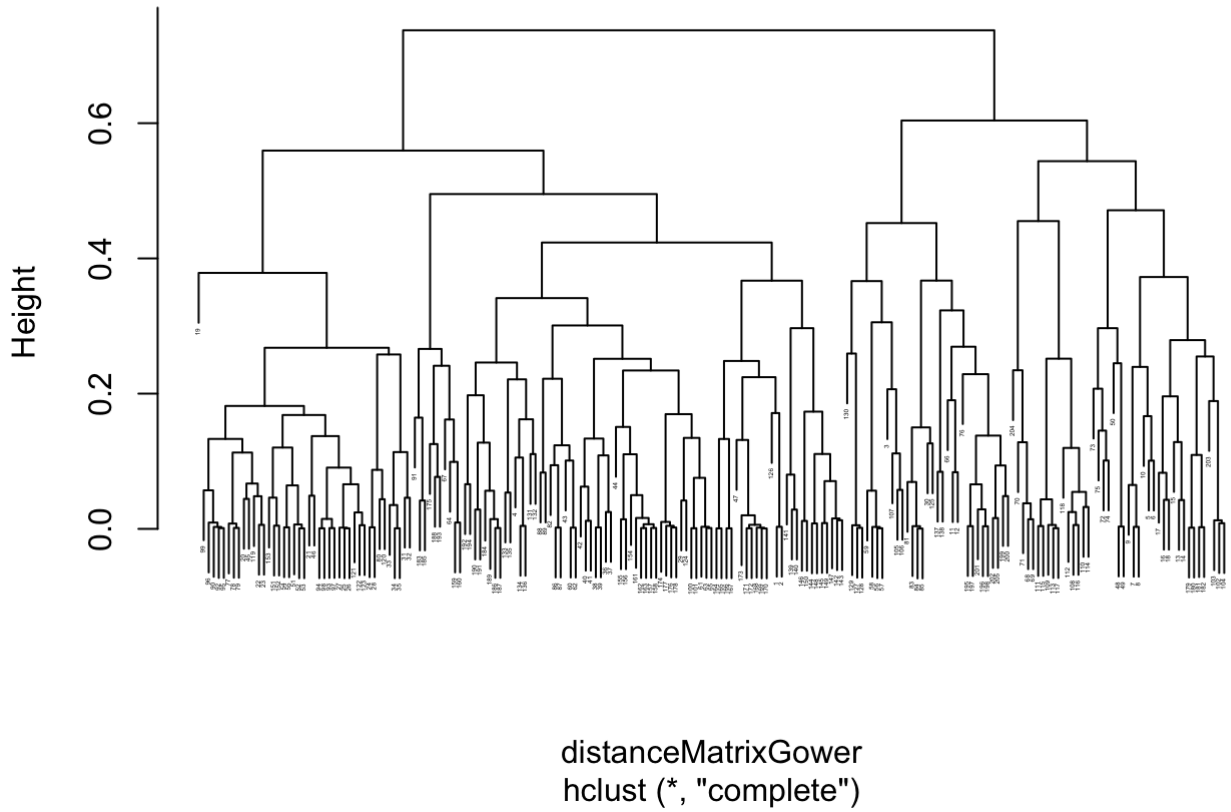
[code section: 32] !PARTIALLY REMOVED FROM VIEW!

```
distanceMatrixGower <- cluster::daisy(select(eng_car_prices, -c("car_ID", "CarName", "carbod
y", "citympg", "highwaympg")), metric = "gower")

dendogramWardD2_Gow = hclust(distanceMatrixGower, method="ward.D2")
tempClustersNum = 3

tempGroups <- cutree(dendogramWardD2_Gow, k = tempClustersNum)

cluster_car_prices$fuelSystemClustered3 = tempGroups

tab_hier_gow_wd2 = table(cluster_car_prices$fuelsystem, cluster_car_prices$fuelSystemClustere
d3)
pander(prop.table(tab_hier_gow_wd2, 1), caption = "Table")
```

Table

	1	2	3
1bbl	0	1	0
2bbl	0.0303	0.9697	0
4bbl	1	0	0
idi	0	0	1
mpfi	0.9263	0.07368	0
spdi	0.5556	0.4444	0
spfi	1	0	0

```
dendogramComplete_Gow = hclust(distanceMatrixGower, method="complete")
plot(dendogramComplete_Gow, cex=0.2)
```




```
tempClustersNum = 2

tempGroups <- cutree(dendogramComplete_Gow, k = tempClustersNum)

cluster_car_prices$cylindernumberClustered = tempGroups
tab_hier_gow_comp = table(cluster_car_prices$cylindernumber, cluster_car_prices$cylindernumbe
rClustered)
pander(prop.table(tab_hier_gow_comp, 1), caption = "Table")
```

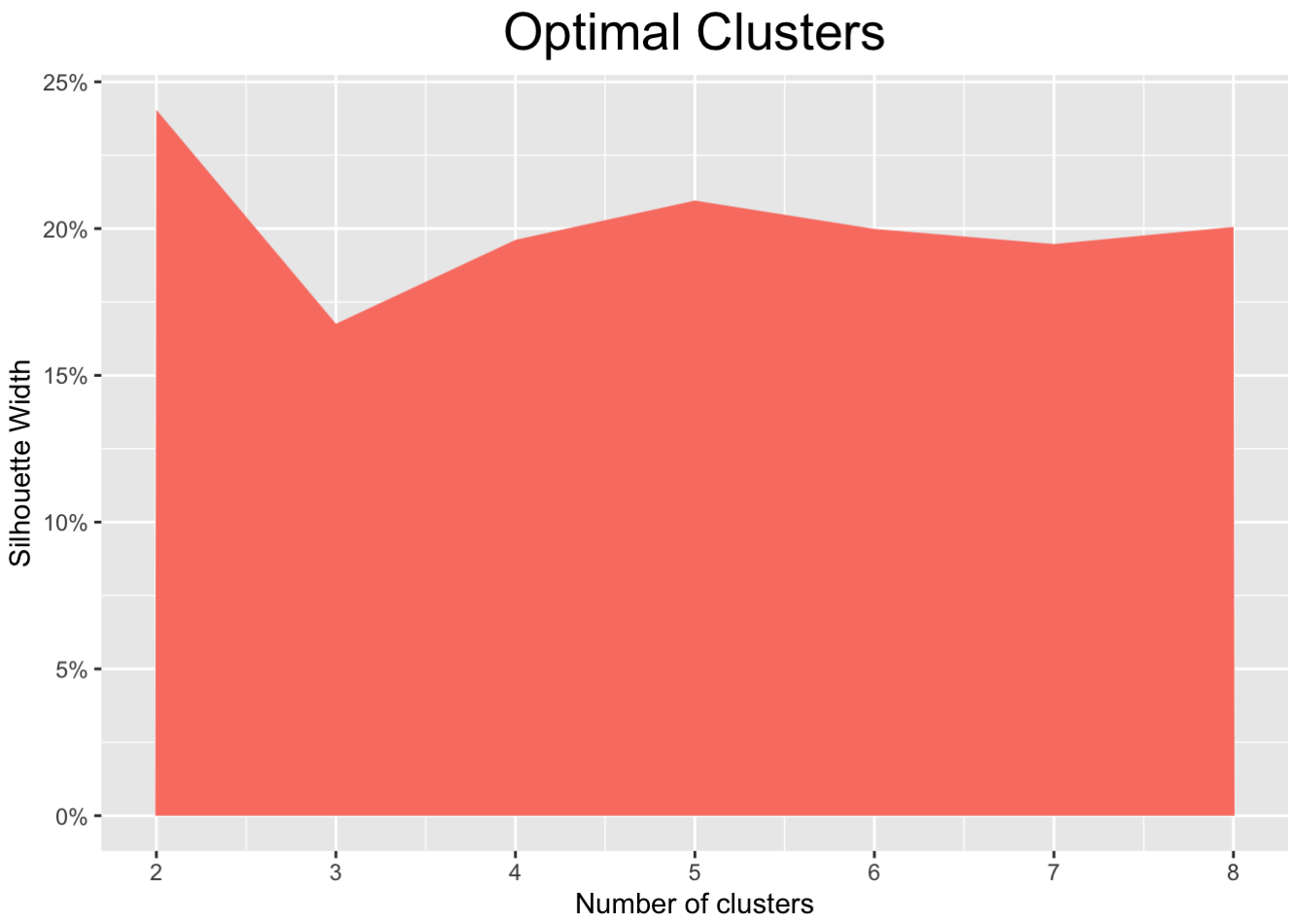
Table We obtained a clusterization mainly related to the cylinder number label.

	1	2
eight	0	1
five	0.09091	0.9091
four	0.7987	0.2013
six	0	1
three	1	0
twelve	0	1
two	0	1

Finally, we used the Silhouette method with the Gower-distance-based hierarchical clusterization.

[code section: 33] !PARTIALLY REMOVED FROM VIEW!

```
# Visualize the optimal number of clusters using Silhouette method
n_clusters <- c(2:8)
lapply(n_clusters, function(x){
  pam_Fit <- pam(distanceMatrixGower, diss = TRUE, k = x)
  n_SW <- pam_Fit$silinfo$avg.width
  return(n_SW)
}) %>%
  as.numeric() %>%
  as_tibble() %>%
  ggplot(aes(x = n_clusters, y = value)) +
  geom_area(fill = "salmon") +
  ggtitle("Optimal Clusters") +
  labs(x = "Number of clusters", y = "Silhouette Width") +
  theme(plot.title = element_text(hjust = 0.5, size = 20)) +
  scale_y_continuous(labels = scales::percent) +
  scale_x_continuous(breaks = n_clusters)
```




```
tempClustersNum <- 2

cluster_car_prices <- cbind(cluster_car_prices, TotalClusterization1 = as.factor(pam(distance
MatrixGower, diss = TRUE, tempClustersNum)$clustering))

tempClustersNum <- 5

cluster_car_prices <- cbind(cluster_car_prices, TotalClusterization2 = as.factor(pam(distance
MatrixGower, diss = TRUE, tempClustersNum)$clustering))
```

We tried the two most interesting clusters numbers, but e did not found great correlations with any label.