Deliverable 3

Statistical model building

Pere Arnau Alegre & Andrés Jiménez González

May 29, 2022

Contents

1 Data description

- Description https://www.kaggle.com/datasets/adityadesai13/used-car-dataset-ford-and-mercedes
- Data Dictionary - Scraped data of used cars, which have been separated into files corresponding to each car manufacturer (only Mercedes, BMW, Volkswagen and Audi cars are to be considered).

1.1 Variables

- Model
  - A string indicating the model of the car.
- Year
  - A discrete numeric variable to indicate the year the car was sold
- Price
  - Continuous variable indicating the price at which the car was sold
- Transmission
  - Categorical variable that indicates the type of transmission of the car
  - Values:
    - Automatic
    - Manual
    - Semi-Automatic
    - Other
- Mileage
  - A discrete numeric variable to indicate the number of miles the car had when it was sold
- Fuel Type
  - Categorical variable that indicates the type of fuel of the car

  - Values:
    * Diesel
    * Electric
    * Hybrid
    * Petrol
    * Other

- Tax
  - A discrete numeric variable to indicate the road tax of the vehicle.
- MPG
  - Continuous variable indicating the fuel consumption of the car
- Engine Size
  - Continuous variable indicating the size of the engine
- Manufacturer
  - Categorical variable that indicates the manufacturer brand of the car.
  - Values:
    * Mercedes
    * Audi
    * Volkswagen
    * BMW

2 Loading of Required Packages for the deliverable

We load the necessary packages and set the working directory

```
# setwd('C:/Users/TOREROS-II/Documents/GitHub/adei/adei/deliverable2')
setwd("C:/Users/Arnau/Desktop/adei/deliverable3")
# Load Required Packages
options(contrasts = c("contr.treatment", "contr.treatment"))
requiredPackages <- c("missMDA", "chemometrics", "mvoutlier", "effects",
    "FactoMineR", "car", "factoextra", "RColorBrewer", "dplyr", "ggmap",
    "ggthemes", "knitr", "corrplot", "moments")
missingPackages <- requiredPackages[!(requiredPackages %in% installed.packages()[,
    "Package"])]
if (length(missingPackages)) install.packages(missingPackages)
lapply(requiredPackages, require, character.only = TRUE)
```

```
if (!is.null(dev.list())) dev.off()  # Clear plots
rm(list = ls())  # Clean workspace
```

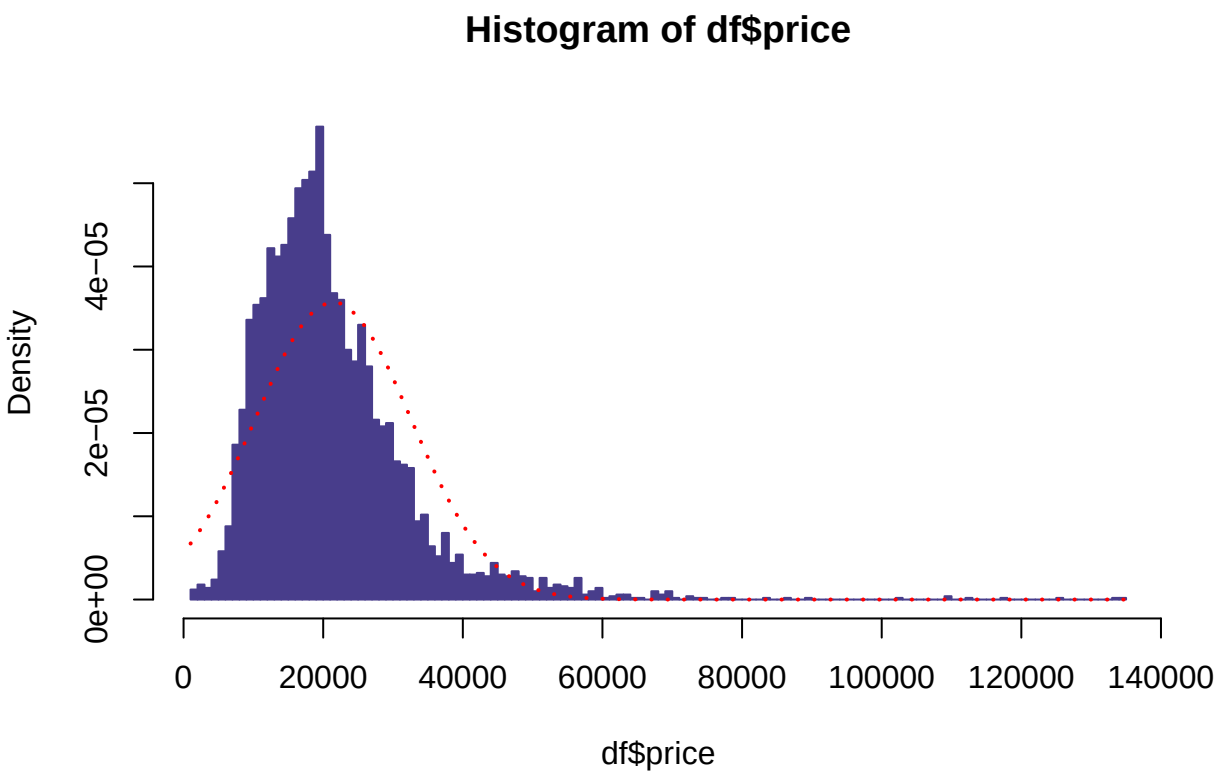
```
# filepath<-'C:/Users/TOREROS-II/Documents/GitHub/adei/adei/'
filepath <- "C:/Users/Arnau/Desktop/adei/deliverable3/"
load(paste0(filepath, "5000cars_clean.RData"))
```

3 Quantitative Logistics Regression

Before we begin to see correlations with our target (variable price), we should consider the normality of this.

3.1 Normality

```
hist(df$price, 100, freq = F, col = "darkslateblue", border = "darkslateblue")
mm <- mean(df$price)
ss <- sd(df$price)
curve(dnorm(x, mean = mm, sd = ss), col = "red", lwd = 2, lty = 3, add = T)
```



```r
shapiro.test(df$price)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df$price
## W = 0.85049, p-value < 2.2e-16
```

Although the histogram of the variable price is not really symmetric, we see that the target price is normally distributed because in the shapiro test we get a p-value smaller than 0.01 so we can reject the null hypothesis. We can see that our target variable of price is not normally distributed, as the histogram of the variable price is not symmetric and we get a p-value smaller than 0.01 in the shapiro wilk test so we can reject the null hypothesis of normality.

3.1.1 Symmetry

```r
# moments library
skewness(df$price)
```

```
## [1] 2.316528
```

Normal data should have 0 skewness: we see that our data is right skewed (2.31).

3.1.2 Kurtosis

```r
kurtosis(df$price)
```

```
## [1] 15.06607
```

Normal data should be 3. We have 15.1, so, in this case, our data is not normal.

3.2 Some transformations of numeric & quantitative variables

```r
ll <- which(df$years_after_sell == 0)
df$years_after_sell[ll] <- 0.5

ll <- which(df$tax == 0)
df$tax[ll] <- 0.5

ll <- which(df$mileage == 0)
df$mileage[ll] <- 0.5

ll <- which(df$mpg == 0)
df$mpg[ll] <- 0.5
```

3.3 Start to find models

Steps to follow:

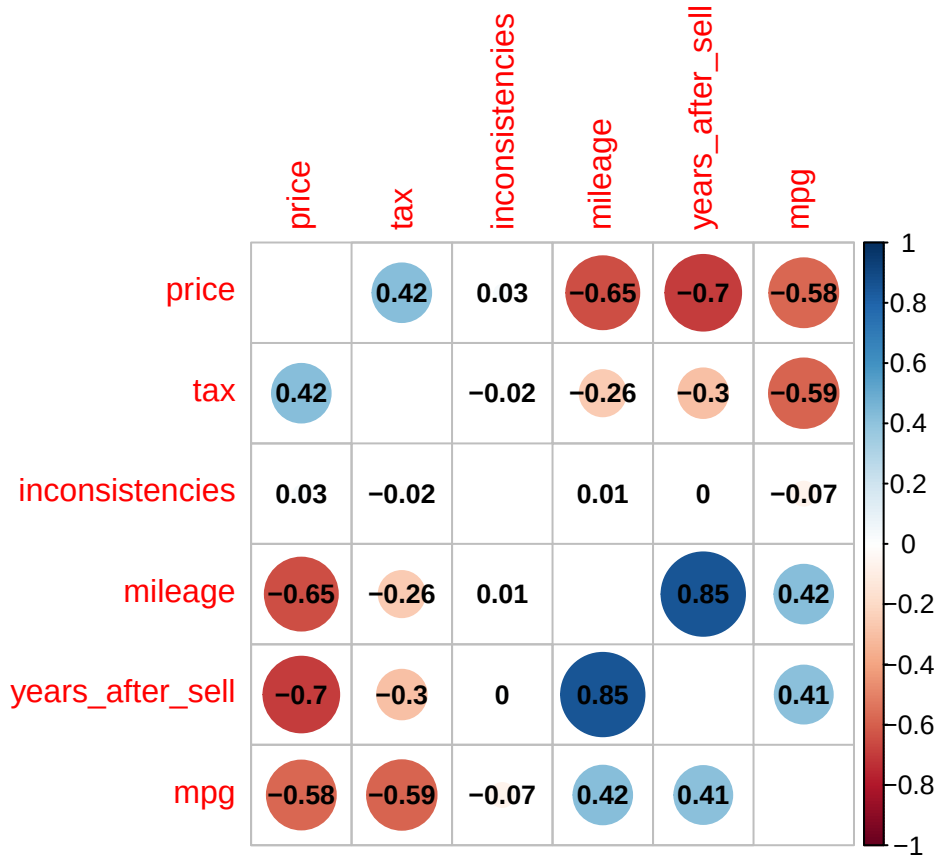
1. Enter all relevant numerical variables in the model
2. See if you need to replace a number with its equivalent factor
3. Add to the best model of step 2, the main effects of the factors and retain the significant net effects.
4. Add interactions: between factor-factor and between factor-numeric (doubles).
5. Diagnosis of waste and observations. Lack of adjustment and / or influential.

A good model not only needs to fit data well, it also needs to be parsimonious. That is, a good model should be only be as complex as necessary to describe a dataset. If we are choosing between a very simple model with 1 IV, and a very complex model with, say, 10 IVs, the very complex model needs to provide a much better fit to the data in order to justify its increased complexity. If it can't, then the more simpler model should be preferred.

3.3.1 Model 1

We will take the most correlated numerical variables We use spearman's method since out target is not normally distributed

```r
M = round(cor(df[, c("price", vars_con)], method = "spearman"), dig = 2)
corrplot(M, method = "circle", insig = "blank", addCoef.col = "black",
    number.cex = 0.8, order = "AOE", diag = FALSE)
```



After seeing the correlation, to make an initial model, we should select the ones that are most correlated, which are:

- tax
- mileage
- years_after_sell
- mpg

```
model_1 <- lm(price ~ mileage + tax + mpg + years_after_sell, data = df)
summary(model_1)
```

```
##
## Call:
## lm(formula = price ~ mileage + tax + mpg + years_after_sell,
##     data = df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -37638  -4977   -260   3415  95570
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       5.741e+04  8.880e+02  64.655  < 2e-16 ***
## mileage          -2.603e-02  8.501e-03  -3.062  0.00221 **
## tax              -6.665e+00  2.166e+00  -3.077  0.00210 **
## mpg              -4.630e+02  1.278e+01 -36.226  < 2e-16 ***
## years_after_sell -2.097e+03  9.038e+01 -23.200  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7842 on 4995 degrees of freedom
## Multiple R-squared:  0.5076, Adjusted R-squared:  0.5072
## F-statistic:  1287 on 4 and 4995 DF,  p-value: < 2.2e-16
```

Model_1 explains 50.76% of the variability of the target.

```
vif(model_1)  #Variance inflation factor: multicorrelation
```

```
##          mileage              tax              mpg years_after_sell
##         2.693407         1.652226         1.787447         2.638829
```

None of the variables of our model have a vif greater than 5 so they are independent with each other.

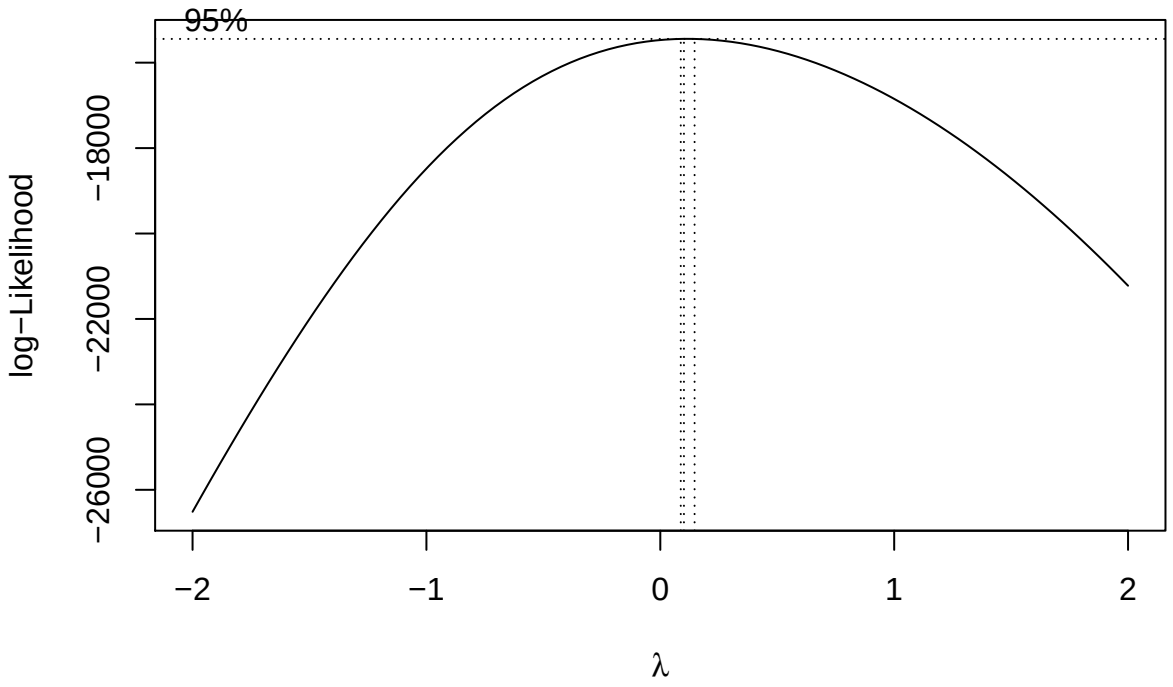
3.3.2 Model 2

With this second model we will start applying transformations to our numerical variables and response variable

```
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
# Target variable transformation?
boxcox(price ~ mileage + tax + mpg + years_after_sell, data = df)
```



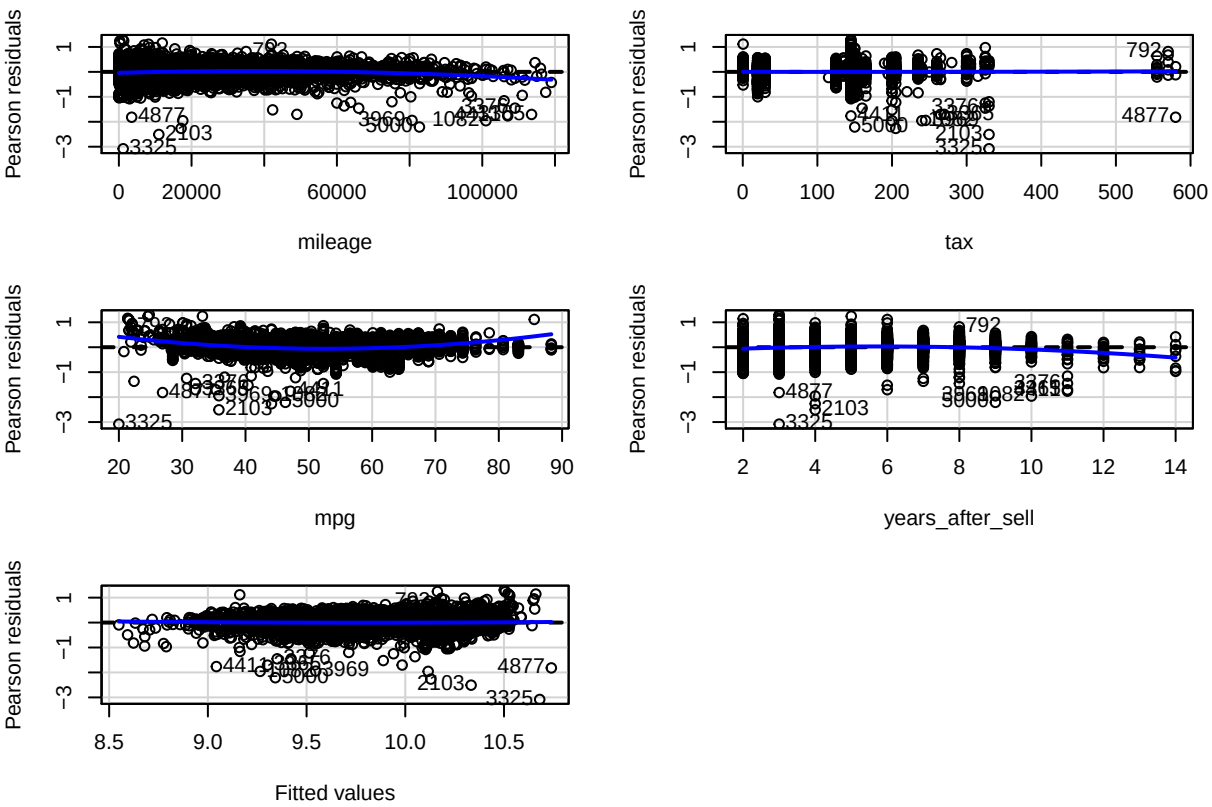

```
# Lambda = 0 - log transformation is needed, as we predicted while
# testing for normality

# New model:
m2 <- lm(log(price) ~ mileage + tax + mpg + years_after_sell, data = df)
summary(m2)
```

```
##
## Call:
## lm(formula = log(price) ~ mileage + tax + mpg + years_after_sell,
##     data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.08244 -0.16765  0.02261  0.19621  1.30586
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       1.108e+01  3.535e-02 313.497  < 2e-16 ***
## mileage          -2.434e-06  3.384e-07  -7.193 7.26e-13 ***
## tax               6.266e-04  8.622e-05   7.267 4.25e-13 ***
## mpg              -1.334e-02  5.088e-04 -26.215  < 2e-16 ***
## years_after_sell -1.129e-01  3.598e-03 -31.367  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3122 on 4995 degrees of freedom
## Multiple R-squared:  0.5881, Adjusted R-squared:  0.5877
## F-statistic:  1783 on 4 and 4995 DF,  p-value: < 2.2e-16
```

```
vif(m2)  #Not changed because explanatory variables have not changed
```

```
##          mileage              tax              mpg years_after_sell
##         2.693407         1.652226         1.787447         2.638829
```

```
# Transformations to my regresors
residualPlots(m2, id = list(method = cooks.distance(m2), n = 10))
```



```
##                  Test stat Pr(>|Test stat|)
## mileage            -8.0911       7.363e-16 ***
## tax                 0.3594          0.7193
## mpg                17.7611       < 2.2e-16 ***
## years_after_sell   -9.4941       < 2.2e-16 ***
## Tukey test          1.2121          0.2255
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the residual plots, we see that some variables show a non linear pattern with the residuals and therefore should be applyed a transformation.

```
boxTidwell(log(price) ~ mileage + tax + years_after_sell, data = df)
```

```
##                  MLE of lambda Score Statistic (z) Pr(>|z|)
## mileage                0.64324              0.8664 0.386290
## tax                    0.86366             -1.7427 0.081387 .
## years_after_sell       1.52621             -3.0451 0.002326 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## iterations =  5
```

```
boxTidwell(log(price) ~ mpg + years_after_sell + mileage, data = df)
```

```
##                  MLE of lambda Score Statistic (z)  Pr(>|z|)
## mpg                   -1.1968             17.7818 < 2.2e-16 ***
## years_after_sell       1.3792             -5.4298 5.641e-08 ***
## mileage                3.5454             -2.6022  0.009262 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## iterations =  7
```

```
# Power transformations of the predictors in a linear model

# Suggested by BoxTidwell
m3 <- lm(log(price) ~ I(mileage^3) + tax + I(mpg^-1) + I(years_after_sell^2),
    data = df)
m3_aux <- step(m3, k = log(nrow(df)))  #to see if we can reduce the model
```

```
## Start:  AIC=-11989.2
## log(price) ~ I(mileage^3) + tax + I(mpg^-1) + I(years_after_sell^2)
##
##                         Df Sum of Sq    RSS    AIC
## <none>                               450.72 -11989
## - I(mileage^3)           1     7.685 458.40 -11913
## - tax                    1     9.447 460.16 -11894
## - I(mpg^-1)              1   121.133 571.85 -10808
## - I(years_after_sell^2)  1   176.303 627.02 -10347
```

```
# step shows that no reduction is needed
summary(m3)
```

```
##
## Call:
## lm(formula = log(price) ~ I(mileage^3) + tax + I(mpg^-1) + I(years_after_sell^2),
##     data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7567 -0.1601  0.0207  0.1903  1.1119
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            9.289e+00  1.926e-02 482.259   <2e-16 ***
## I(mileage^3)          -3.214e-16  3.482e-17  -9.229   <2e-16 ***
## tax                    8.075e-04  7.892e-05  10.232   <2e-16 ***
## I(mpg^-1)              3.779e+01  1.031e+00  36.639   <2e-16 ***
## I(years_after_sell^2) -9.930e-03  2.247e-04 -44.202   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3004 on 4995 degrees of freedom
## Multiple R-squared:  0.6186, Adjusted R-squared:  0.6183
## F-statistic:  2025 on 4 and 4995 DF,  p-value: < 2.2e-16
```

Checking if the transformations suggested by boxTidwell are actually relevant.

```
anova(m2, m3)
```

```
## Analysis of Variance Table
##
## Model 1: log(price) ~ mileage + tax + mpg + years_after_sell
## Model 2: log(price) ~ I(mileage^3) + tax + I(mpg^-1) + I(years_after_sell^2)
##   Res.Df    RSS Df Sum of Sq F Pr(>F)
## 1   4995 486.82
## 2   4995 450.72  0    36.104
```

```r
abs(AIC(m2, m3))  #model 3 offers a better fit
```

```
##    df      AIC
## m2  6 2554.883
## m3  6 2169.598
```

```r
abs(BIC(m2, m3))  #model 3 offers a better fit
```

```
##    df      BIC
## m2  6 2593.986
## m3  6 2208.701
```

The anova test shows that the 2 models are equal and from the AIC and the BIC we get that model 3 offers a better fit.

Let's try to remove multivariate outliers from the model and perform boxTidwell again.

```r
df_clean = df[!df$mout == "MvOut.Yes", ]
m3 <- lm(log(price) ~ mileage + tax + mpg + years_after_sell, data = df_clean)
summary(m3)
```

```
##
## Call:
## lm(formula = log(price) ~ mileage + tax + mpg + years_after_sell,
##     data = df_clean)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.2834 -0.1625  0.0217  0.1879  0.8648
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       1.093e+01  3.478e-02 314.324  < 2e-16 ***
## mileage          -1.749e-06  3.586e-07  -4.878 1.11e-06 ***
## tax               1.001e-03  8.713e-05  11.485  < 2e-16 ***
## mpg              -1.228e-02  4.874e-04 -25.194  < 2e-16 ***
## years_after_sell -1.060e-01  3.759e-03 -28.196  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.287 on 4831 degrees of freedom
## Multiple R-squared:  0.5898, Adjusted R-squared:  0.5895
## F-statistic:  1737 on 4 and 4831 DF,  p-value: < 2.2e-16
```

We can see from the summary that the mileage has lost some relevancy in the model due to the removal of multivariate outliers.

```r
boxTidwell(log(price) ~ mileage + tax + mpg + years_after_sell, data = df_clean)
```

```
##                   MLE of lambda Score Statistic (z)  Pr(>|z|)
## mileage                 0.54225              0.9015    0.3673
## tax                     0.49939             -7.7898 6.712e-15 ***
## mpg                    -2.25671             19.6587 < 2.2e-16 ***
## years_after_sell        1.97966             -5.9550 2.600e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## iterations =  11
```

```r
# we see from the boxtidwell output that the mileage does not need
# transformation and that the tax should be square rooted
boxTidwell(log(price) ~ mileage + sqrt(tax) + I(mpg^-2) + I(years_after_sell^2),
    data = df_clean)
```

```
##                         MLE of lambda Score Statistic (z) Pr(>|z|)
## mileage                       0.54225              1.7694  0.07683 .
## sqrt(tax)                     0.99825             -0.0889  0.92918
## I(mpg^-2)                     1.12862              1.8946  0.05814 .
## I(years_after_sell^2)         0.98983              0.2636  0.79207
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## iterations =  8
```

```r
m3_aux <- lm(log(price) ~ mileage + sqrt(tax) + I(mpg^-2) + I(years_after_sell^2),
    data = df_clean)
summary(m3_aux)
```

```
##
## Call:
## lm(formula = log(price) ~ mileage + sqrt(tax) + I(mpg^-2) + I(years_after_sell^2),
##     data = df_clean)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.25844 -0.15068  0.01796  0.18314  0.83659
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)              9.597e+00  1.677e-02 572.388  < 2e-16 ***
```

```
## mileage               -2.172e-06  3.310e-07  -6.564 5.78e-11 ***
## sqrt(tax)              2.188e-02  1.327e-03  16.487  < 2e-16 ***
## I(mpg^-2)              7.622e+02  2.222e+01  34.296  < 2e-16 ***
## I(years_after_sell^2) -8.930e-03  3.184e-04 -28.046  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2765 on 4831 degrees of freedom
## Multiple R-squared:  0.6193, Adjusted R-squared:  0.619
## F-statistic:  1965 on 4 and 4831 DF,  p-value: < 2.2e-16
```

```
m3 <- m3_aux
```

We will keep the new model3 as it offers a greater r-squared than model 2 and has a better fit with residuals and AIC score.

```
# Validation and effects consideration:
Anova(m3)  #Net effect test
```

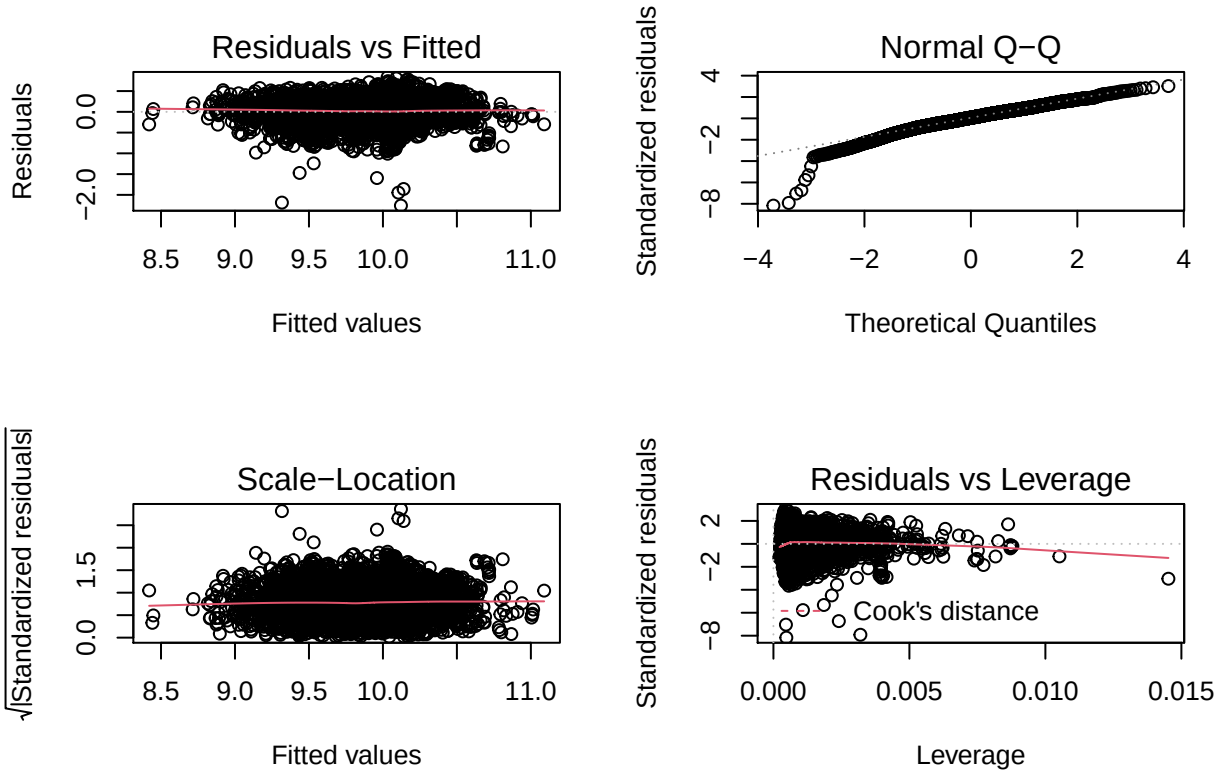
3.3.2.1 Small validation of our model before adding factors

```
## Anova Table (Type II tests)
##
## Response: log(price)
##                       Sum Sq   Df  F value    Pr(>F)
## mileage                 3.29    1   43.088 5.784e-11 ***
## sqrt(tax)              20.78    1  271.816 < 2.2e-16 ***
## I(mpg^-2)              89.93    1 1176.222 < 2.2e-16 ***
## I(years_after_sell^2)  60.14    1  786.574 < 2.2e-16 ***
## Residuals             369.36 4831
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
vif(m3)
```

```
##                mileage             sqrt(tax)             I(mpg^-2)
##               2.584157              1.432182              1.382442
## I(years_after_sell^2)
##               2.532646
```

```
par(mfrow = c(2, 2))
plot(m3, id.n = 0)
```




```
par(mfrow = c(1, 1))
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

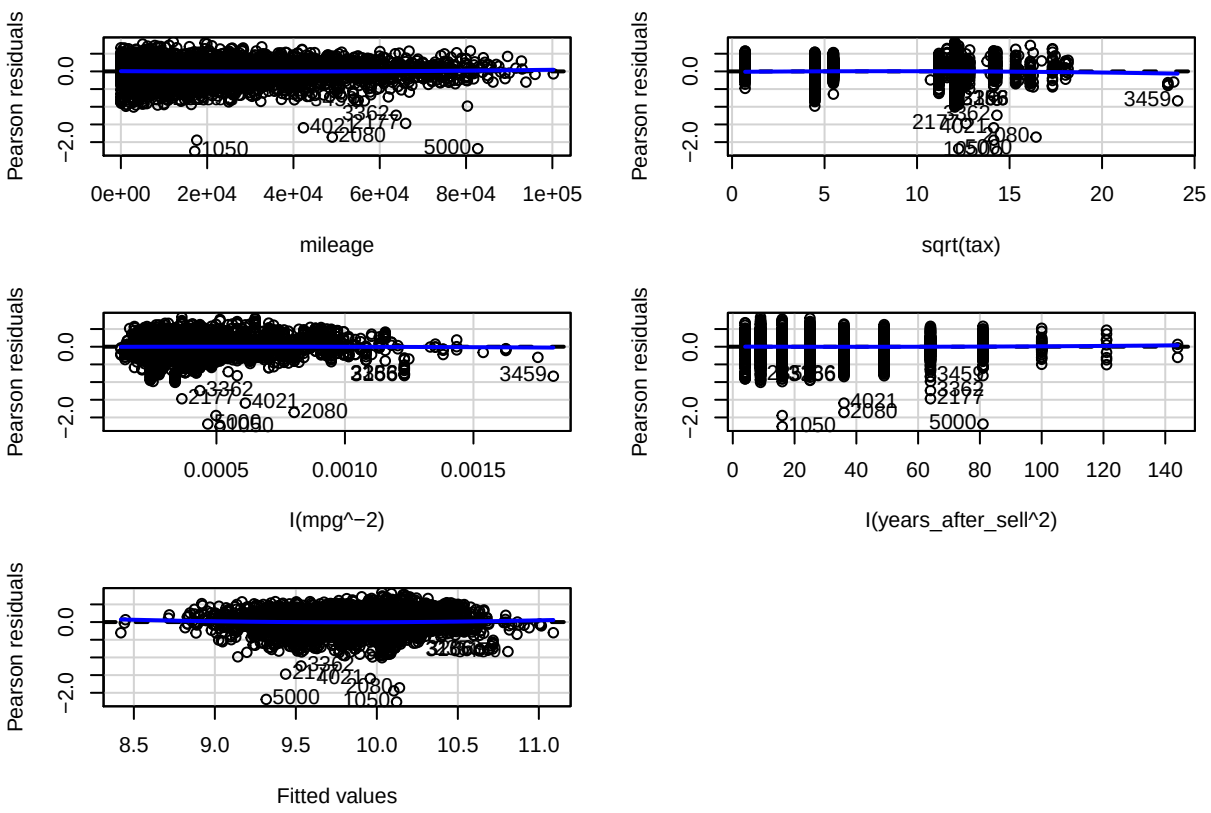
```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
bptest(m3)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  m3
## BP = 5.8267, df = 4, p-value = 0.2125
```

In the residuals vs fitted models we can observe that the mean residual doesn't change with the fitted values (and so is doesn't change with x), and the spread of the residuals (and hence of the y's about the fitted line) is consistent as the fitted values (or x) changes. That is, the spread is constant and so we have homoscedasticity.With the Breusch-Pagan test we get a p-value of 0.2125 so with a lot of confidence we don't reject the hypothesis of homoscedasticity. The model fits the linear hypothesis. About the test: The bp test statistic approximately follows a chi-square distribution. The null hypothesis for this test is that the error variances are all equal. More specifically, as Y increases, the variances increase (or decrease). In the Normal QQ plot we can see that values tend to drop in value in the left tail, this effect happens because our variable log(price) is left skewed. But the effect is not too drastic so we could accept that our distribution of residuals is normal.

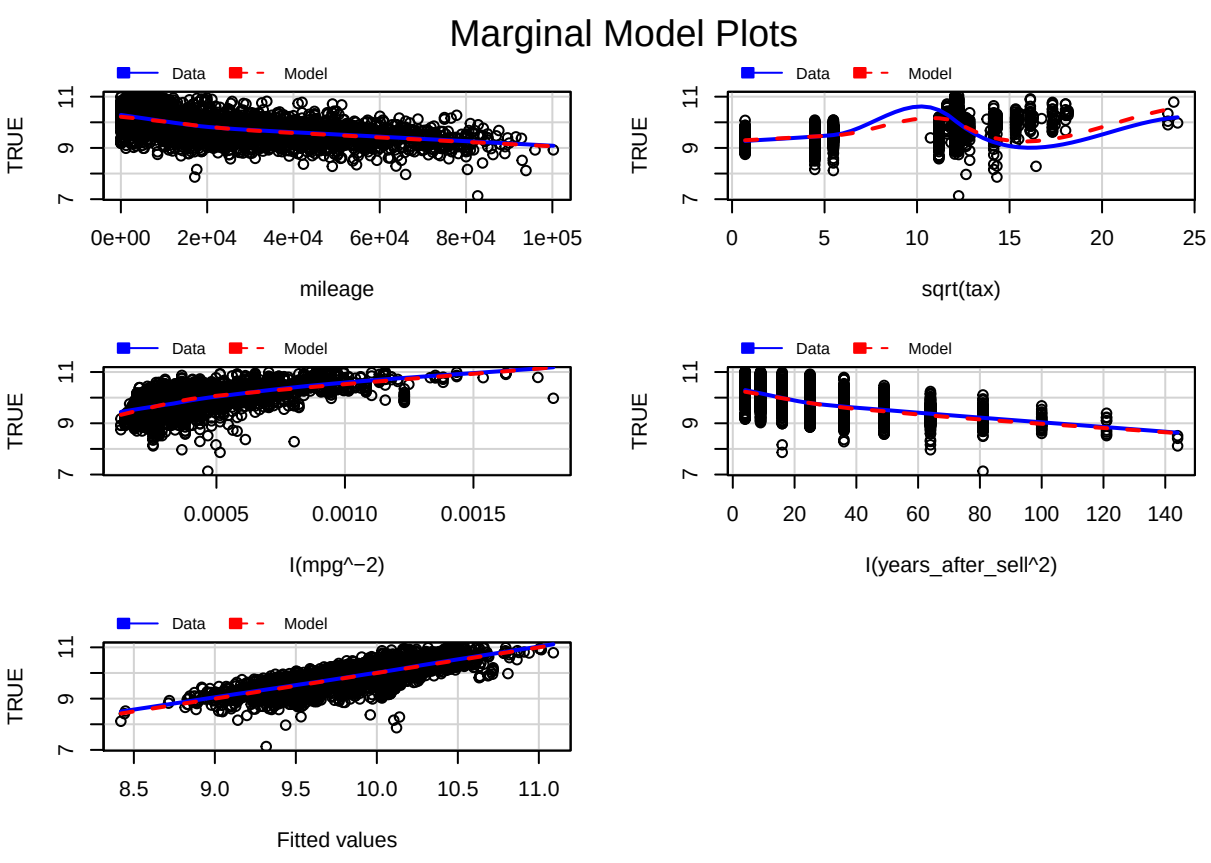
```
residualPlots(m3, id = list(method = cooks.distance(m3), n = 10))
```




```
##                         Test stat Pr(>|Test stat|)
## mileage                    1.2978         0.194407
## sqrt(tax)                 -1.0514         0.293141
## I(mpg^-2)                 -0.3641         0.715821
## I(years_after_sell^2)      0.6778         0.497962
## Tukey test                 2.6909         0.007126 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

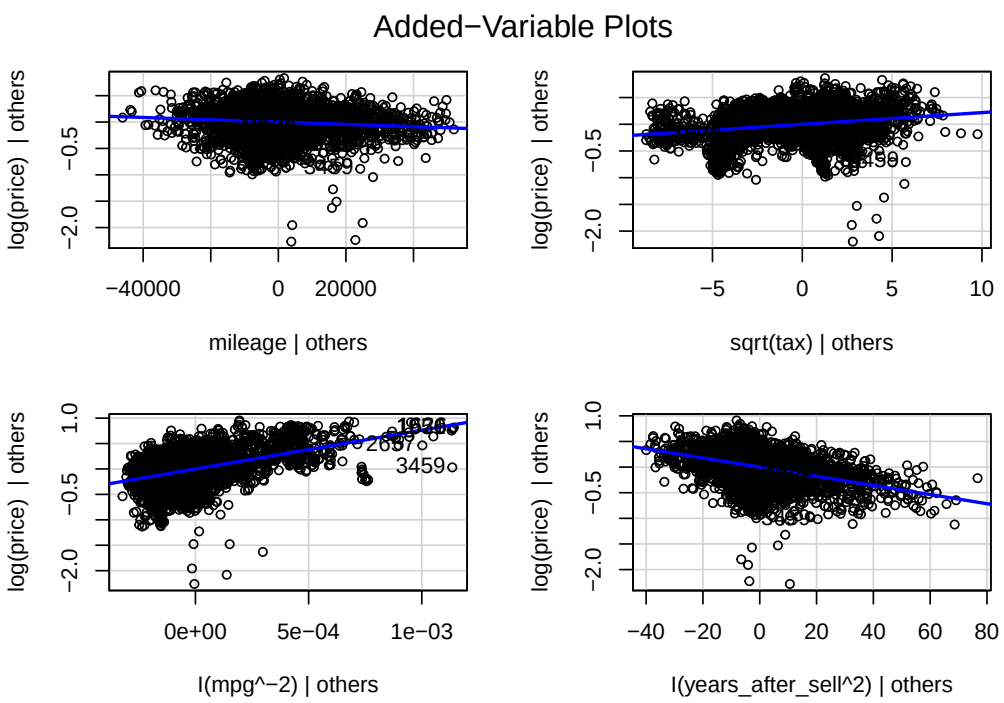
We can see that all the variables from the model show linear residuals because we have applied the according transformation to all the variables.

```
marginalModelPlots(m3)
```

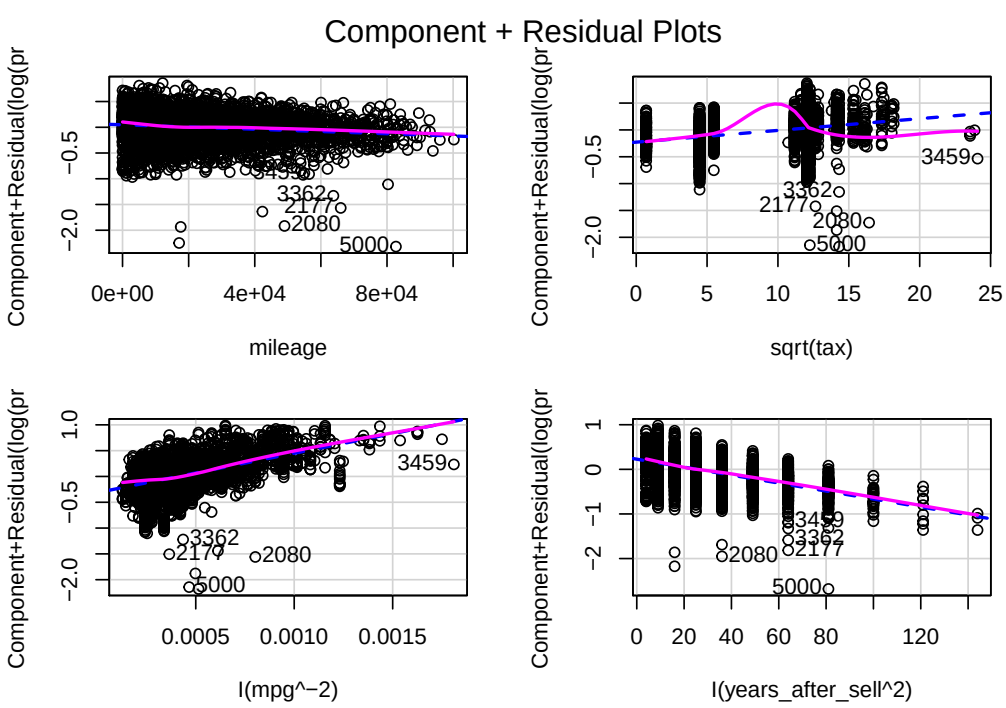


Marginal model plots display the marginal relationship between the response and each predictor in a regression model. Marginal model plots display the dependent variable on each vertical axis and each independent variable on a horizontal axis. There is one marginal model plot for each independent variable and one additional plot that displays the predicted values on the horizontal axis. Each plot contains a scatter plot of the two variables, a smooth fit function for the variables in the plot (labeled "Data"), and a function that displays the predicted values as a function of the horizontal axis variable (labeled "Model"). When the two functions are similar in each of the graphs, there is evidence that the model fits well. When the two functions differ in at least one of the graphs, there is evidence that the model does not fit well. We can see that our model fits well with the data.

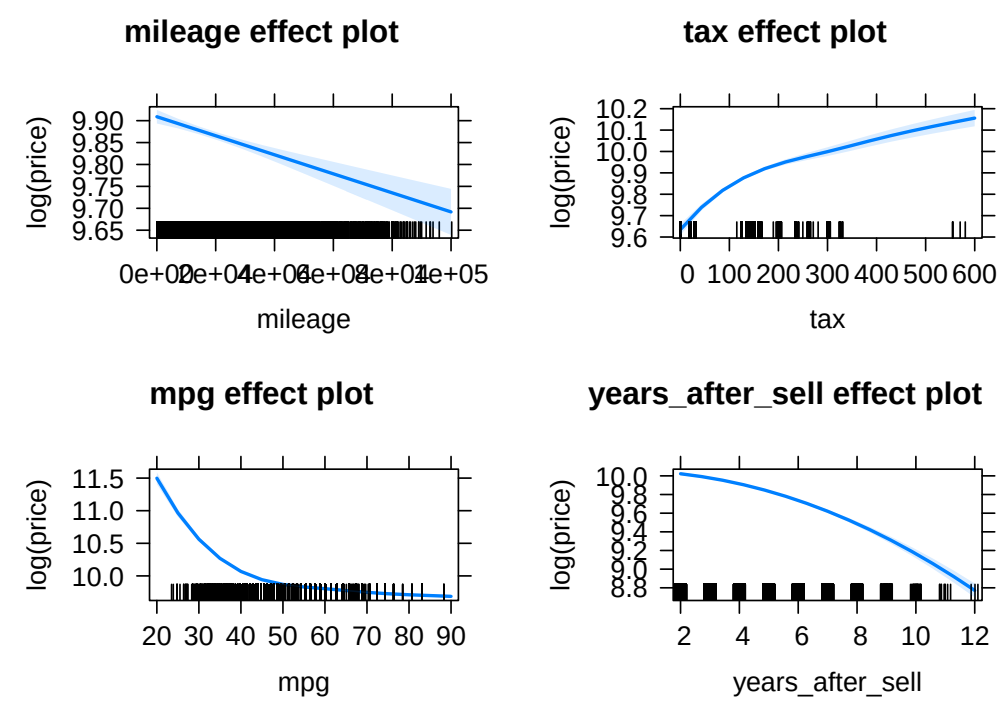
```
avPlots(m3, id = list(method = hatvalues(m3), n = 5))
```




```
crPlots(m3, id = list(method = cooks.distance(m3), n = 5))
```




```
library(effects)
plot(allEffects(m3))
```



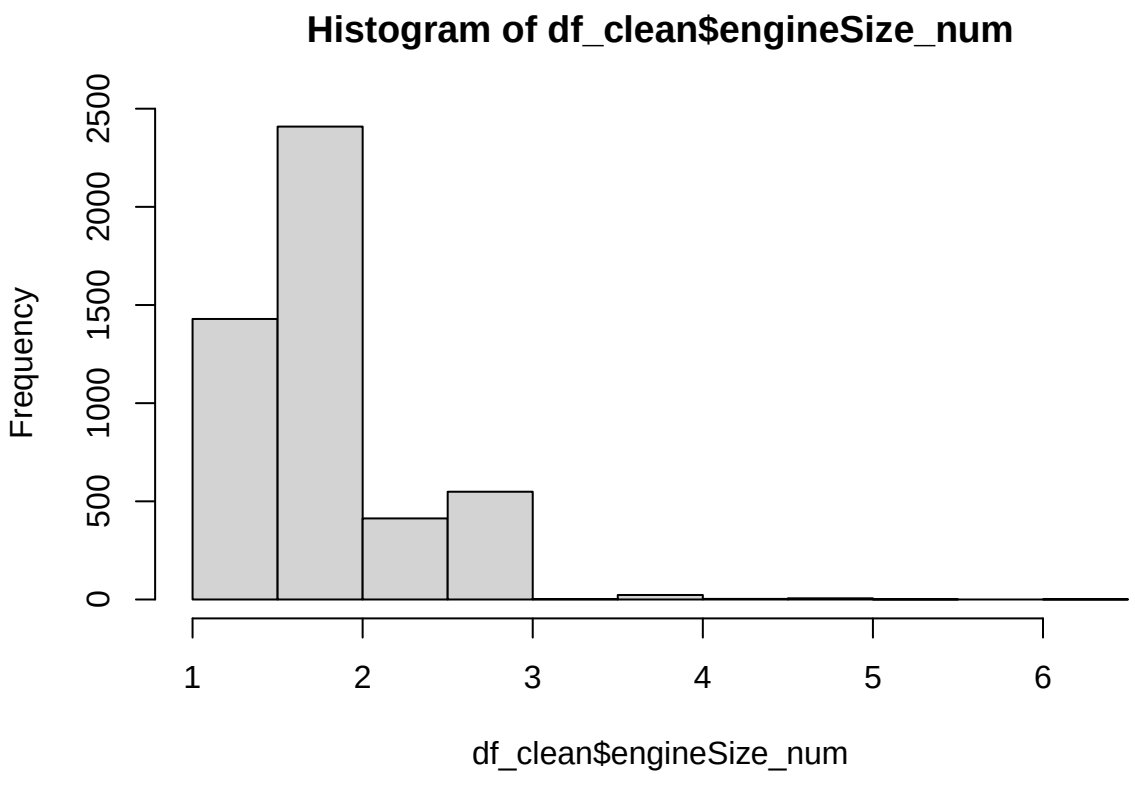
Form the effect plots We see that our model defines the following:

- the log(price) decreases in a linear way as the mileage grows
- the log(price) increases in a logarithmic way as the tax grows
- the log(price) decreases in a exponential way as the mpg grows
- the log(price) decreases in a quadratic way (-x^2) as the years_after_sell grows

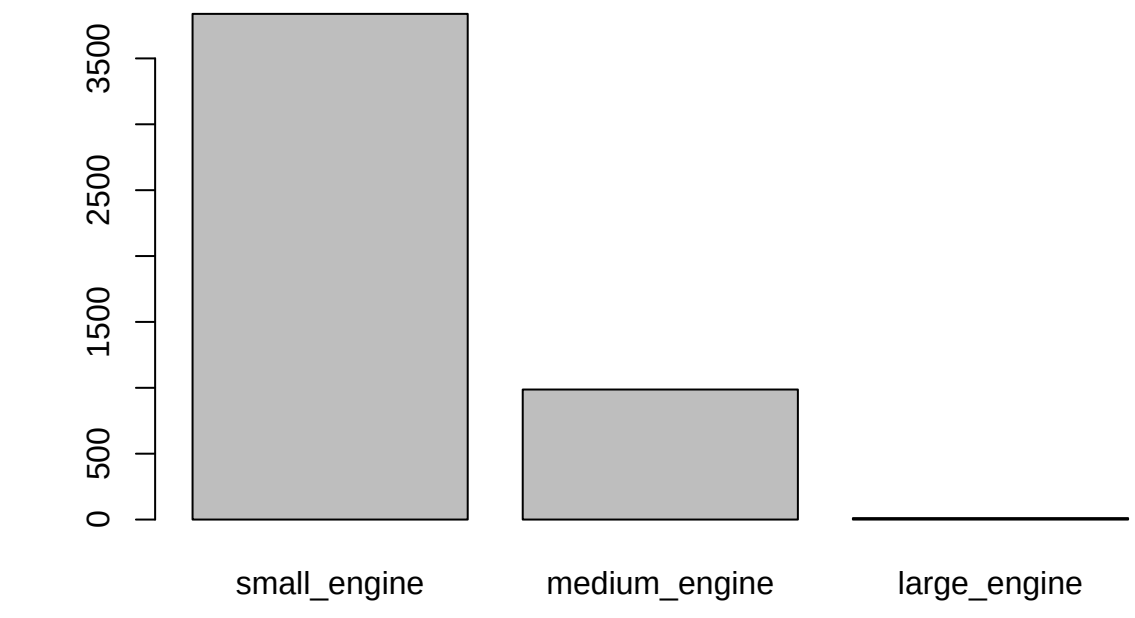
3.4 Adding factors to the model

Refactor of engineSize to include it in the model

```r
df_clean$engineSize_num <- as.numeric(as.character(df_clean$engineSize))
hist(df_clean$engineSize_num)
```




```r
s_aux <- summary(df_clean$engineSize_num)
df_clean$engineSize <- factor(cut(df_clean$engineSize_num, breaks = c(s_aux[1],
    2, 4, s_aux[6]), include.lowest = T), labels = c("small_engine", "medium_engine",
    "large_engine"))
barplot(table(df_clean$engineSize))
```




```r
m4 <- update(m3, ~. + fuelType + transmission + engineSize + manufacturer,
    data = df_clean)
vif(m4)
```

```
##                           GVIF Df GVIF^(1/(2*Df))
## mileage                2.695658  1        1.641846
## sqrt(tax)              1.542217  1        1.241860
## I(mpg^-2)              2.000604  1        1.414427
## I(years_after_sell^2) 2.607387  1        1.614740
## fuelType               1.505837  2        1.107757
## transmission           1.589752  2        1.122877
## engineSize             1.592377  2        1.123341
## manufacturer           1.561948  3        1.077154
```

```r
# we do not have dependency among variables
summary(m4)
```

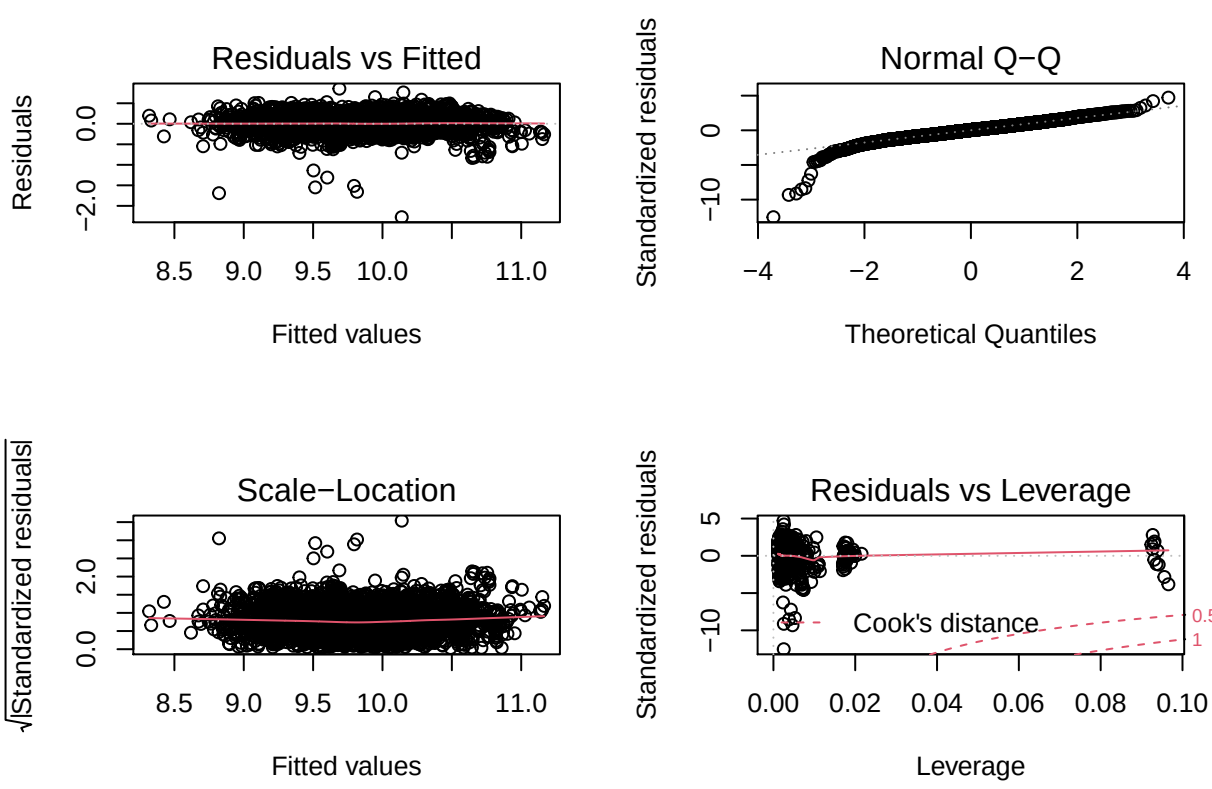
```
##
## Call:
## lm(formula = log(price) ~ mileage + sqrt(tax) + I(mpg^-2) + I(years_after_sell^2) +
##     fuelType + transmission + engineSize + manufacturer, data = df_clean)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.2756 -0.1072  0.0074  0.1100  0.8560
```

```
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   9.768e+00  1.317e-02 741.647  < 2e-16 ***
## mileage                      -4.873e-06  2.223e-07 -21.917  < 2e-16 ***
## sqrt(tax)                     9.264e-03  9.058e-04  10.227  < 2e-16 ***
## I(mpg^-2)                     8.603e+02  1.758e+01  48.927  < 2e-16 ***
## I(years_after_sell^2)        -7.572e-03  2.125e-04 -35.638  < 2e-16 ***
## fuelTypef.Fuel-Petrol        -2.395e-01  6.403e-03 -37.405  < 2e-16 ***
## fuelTypef.Fuel-Hybrid         1.215e-01  2.364e-02   5.139 2.87e-07 ***
## transmissionf.Trans-SemiAuto  1.740e-01  7.277e-03  23.914  < 2e-16 ***
## transmissionf.Trans-Automatic 1.555e-01  8.026e-03  19.372  < 2e-16 ***
## engineSizemedium_engine       9.299e-02  7.991e-03  11.637  < 2e-16 ***
## engineSizelarge_engine       -3.498e-02  5.675e-02  -0.616    0.538
## manufacturerBMW              -6.515e-02  8.254e-03  -7.892 3.64e-15 ***
## manufacturerMercedes         -3.407e-02  8.384e-03  -4.063 4.91e-05 ***
## manufacturerVW               -2.056e-01  7.544e-03 -27.258  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1819 on 4822 degrees of freedom
## Multiple R-squared:  0.8356, Adjusted R-squared:  0.8352
## F-statistic:  1886 on 13 and 4822 DF,  p-value: < 2.2e-16
```

```r
Anova(m4)
```

```
## Anova Table (Type II tests)
##
## Response: log(price)
##                          Sum Sq   Df  F value    Pr(>F)
## mileage                  15.885    1  480.358 < 2.2e-16 ***
## sqrt(tax)                 3.459    1  104.597 < 2.2e-16 ***
## I(mpg^-2)                79.165    1 2393.844 < 2.2e-16 ***
## I(years_after_sell^2)    42.000    1 1270.032 < 2.2e-16 ***
## fuelType                 48.346    2  730.958 < 2.2e-16 ***
## transmission             20.020    2  302.688 < 2.2e-16 ***
## engineSize                4.633    2   70.045 < 2.2e-16 ***
## manufacturer             27.932    3  281.545 < 2.2e-16 ***
## Residuals               159.464 4822
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# we see from the Anova output that all variables greatly contribute
# to the regression model

par(mfrow = c(2, 2))
plot(m4, id.n = 0)
```




```r
par(mfrow = c(1, 1))
```

We can see that with this new model is fully homoscesdastic, and it is also normally distributed.

3.4.1 Adding interactions

We now have built a very strong model with high predictive power, but we could exploit the interactions between the factor variables to increase the accuracy of the model.

```r
m5 <- update(m3, ~. + fuelType * (engineSize + transmission + manufacturer)^2,
    data = df_clean)
# summary(m5) we will perform a step to remove the insignificant
# interactions from our model
m5_aux <- step(m5, k = log(nrow(df_clean)))
```

```
## Start:  AIC=-16326.18
## log(price) ~ mileage + sqrt(tax) + I(mpg^-2) + I(years_after_sell^2) +
##     fuelType + engineSize + transmission + manufacturer + engineSize:transmission +
##     engineSize:manufacturer + transmission:manufacturer + fuelType:engineSize +
##     fuelType:transmission + fuelType:manufacturer + fuelType:engineSize:transmission +
##     fuelType:engineSize:manufacturer + fuelType:transmission:manufacturer
##
##                                      Df Sum of Sq    RSS    AIC
## - fuelType:engineSize:transmission    3     0.029 150.93 -16351
## - fuelType:transmission:manufacturer  8     1.675 152.58 -16341
## <none>                                            150.91 -16326
## - fuelType:engineSize:manufacturer    2     1.123 152.03 -16307
## - sqrt(tax)                           1     3.328 154.23 -16229
## - mileage                             1    16.216 167.12 -15841
## - I(years_after_sell^2)               1    39.970 190.88 -15198
## - I(mpg^-2)                           1    56.320 207.22 -14801
##
## Step:  AIC=-16350.71
## log(price) ~ mileage + sqrt(tax) + I(mpg^-2) + I(years_after_sell^2) +
##     fuelType + engineSize + transmission + manufacturer + engineSize:transmission +
##     engineSize:manufacturer + transmission:manufacturer + fuelType:engineSize +
##     fuelType:transmission + fuelType:manufacturer + fuelType:engineSize:manufacturer +
##     fuelType:transmission:manufacturer
##
##                                      Df Sum of Sq    RSS    AIC
## - engineSize:transmission             3     0.036 150.97 -16375
## - fuelType:transmission:manufacturer  8     1.673 152.61 -16365
## <none>                                            150.93 -16351
## - fuelType:engineSize:manufacturer    2     1.149 152.08 -16331
## - sqrt(tax)                           1     3.321 154.25 -16254
## - mileage                             1    16.196 167.13 -15866
## - I(years_after_sell^2)               1    40.151 191.09 -15218
## - I(mpg^-2)                           1    56.305 207.24 -14826
##
## Step:  AIC=-16375.02
## log(price) ~ mileage + sqrt(tax) + I(mpg^-2) + I(years_after_sell^2) +
##     fuelType + engineSize + transmission + manufacturer + engineSize:manufacturer +
##     transmission:manufacturer + fuelType:engineSize + fuelType:transmission +
##     fuelType:manufacturer + fuelType:engineSize:manufacturer +
##     fuelType:transmission:manufacturer
##
##                                      Df Sum of Sq    RSS    AIC
## - fuelType:transmission:manufacturer  8     1.875 152.84 -16383
## <none>                                            150.97 -16375
## - fuelType:engineSize:manufacturer    2     1.227 152.20 -16353
## - sqrt(tax)                           1     3.300 154.27 -16279
## - mileage                             1    16.220 167.19 -15890
## - I(years_after_sell^2)               1    40.191 191.16 -15242
## - I(mpg^-2)                           1    56.468 207.44 -14847
##
## Step:  AIC=-16383.21
## log(price) ~ mileage + sqrt(tax) + I(mpg^-2) + I(years_after_sell^2) +
##     fuelType + engineSize + transmission + manufacturer + engineSize:manufacturer +
##     transmission:manufacturer + fuelType:engineSize + fuelType:transmission +
##     fuelType:manufacturer + fuelType:engineSize:manufacturer
##
##                                     Df Sum of Sq    RSS    AIC
## - transmission:manufacturer          6     0.802 153.65 -16409
## - fuelType:transmission              3     0.220 153.06 -16402
## <none>                                           152.84 -16383
## - fuelType:engineSize:manufacturer   2     1.178 154.02 -16363
## - sqrt(tax)                          1     3.398 156.24 -16285
## - mileage                            1    16.255 169.10 -15903
## - I(years_after_sell^2)              1    40.452 193.30 -15256
## - I(mpg^-2)                          1    56.148 208.99 -14879
##
## Step:  AIC=-16408.8
## log(price) ~ mileage + sqrt(tax) + I(mpg^-2) + I(years_after_sell^2) +
##     fuelType + engineSize + transmission + manufacturer + engineSize:manufacturer +
##     fuelType:engineSize + fuelType:transmission + fuelType:manufacturer +
##     fuelType:engineSize:manufacturer
##
##                                     Df Sum of Sq    RSS    AIC
## - fuelType:transmission              3     0.176 153.82 -16429
## <none>                                           153.65 -16409
## - fuelType:engineSize:manufacturer   2     1.220 154.87 -16388
## - sqrt(tax)                          1     3.357 157.00 -16313
## - mileage                            1    16.671 170.32 -15919
## - I(years_after_sell^2)              1    40.435 194.08 -15288
## - I(mpg^-2)                          1    56.099 209.75 -14912
##
## Step:  AIC=-16428.7
## log(price) ~ mileage + sqrt(tax) + I(mpg^-2) + I(years_after_sell^2) +
##     fuelType + engineSize + transmission + manufacturer + engineSize:manufacturer +
##     fuelType:engineSize + fuelType:manufacturer + fuelType:engineSize:manufacturer
##
##                                     Df Sum of Sq    RSS    AIC
## <none>                                           153.82 -16429
## - fuelType:engineSize:manufacturer   2     1.169 154.99 -16409
## - sqrt(tax)                          1     3.308 157.13 -16334
## - mileage                            1    16.694 170.52 -15939
## - transmission                       2    19.069 172.89 -15880
## - I(years_after_sell^2)              1    40.456 194.28 -15308
```

```
## - I(mpg^-2)                          1    58.181 212.00 -14886
```

```
# from the step function we are left with this model
m5 <- lm(log(price) ~ mileage + sqrt(tax) + I(mpg^-2) + I(years_after_sell^2) +
    fuelType + engineSize + transmission + manufacturer + engineSize:manufacturer +
    fuelType:engineSize + fuelType:manufacturer + fuelType:engineSize:manufacturer,
    data = df_clean)
Anova(m5)
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison

## Anova Table (Type II tests)
##
## Response: log(price)
##                                  Sum Sq   Df   F value    Pr(>F)
## mileage                          16.694    1  521.6988 < 2.2e-16 ***
## sqrt(tax)                         3.308    1  103.3679 < 2.2e-16 ***
## I(mpg^-2)                        58.181    1 1818.1609 < 2.2e-16 ***
## I(years_after_sell^2)            40.456    1 1264.2754 < 2.2e-16 ***
## fuelType                         47.099    2  735.9206 < 2.2e-16 ***
## engineSize                        5.786    2   90.4136 < 2.2e-16 ***
## transmission                     19.069    2  297.9581 < 2.2e-16 ***
## manufacturer                     27.784    3  289.4159 < 2.2e-16 ***
## engineSize:manufacturer           1.530    4   11.9544 1.148e-09 ***
## fuelType:engineSize               0.011    2    0.1759    0.8387
## fuelType:manufacturer             2.512    6   13.0811 9.763e-15 ***
## fuelType:engineSize:manufacturer  1.169    2   18.2735 1.241e-08 ***
## Residuals                       153.823 4807
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

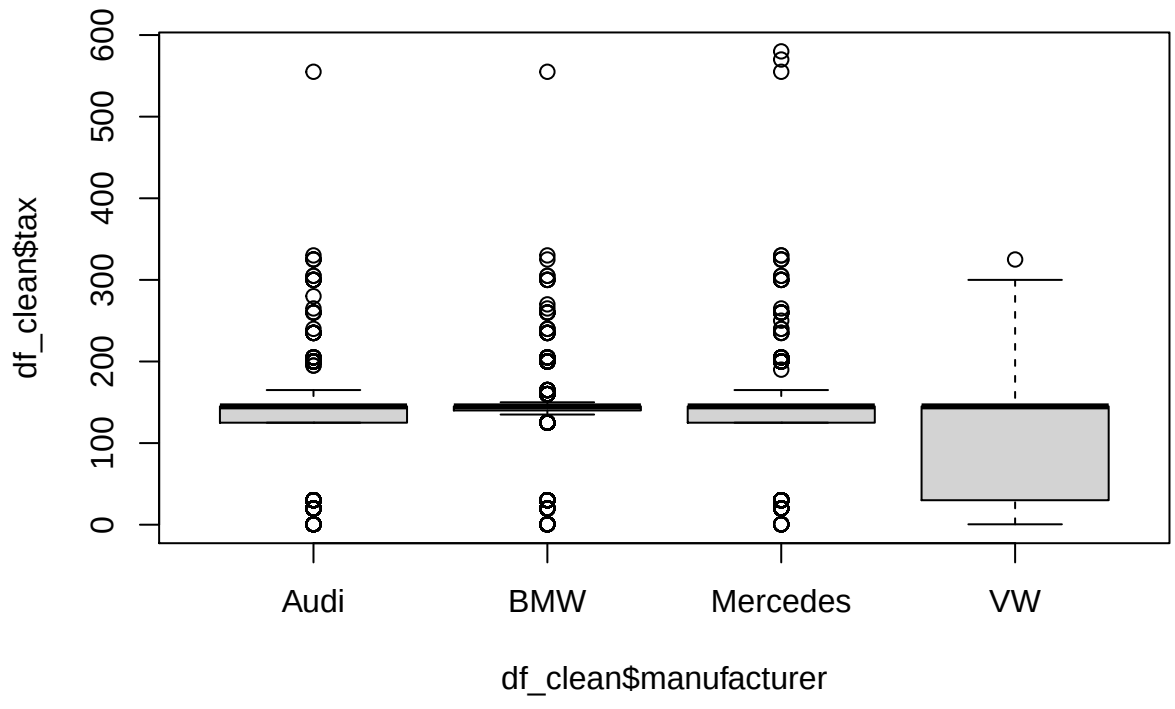
```
# we see from the Anova that the interaction between fuelType and
# engineSize has no relevancy, so we will remove it from our model
m5 <- lm(log(price) ~ mileage + sqrt(tax) + I(mpg^-2) + I(years_after_sell^2) +
    fuelType + engineSize + transmission + manufacturer + engineSize:manufacturer +
    fuelType:manufacturer + fuelType:engineSize:manufacturer, data = df_clean)
```

We will now consider adding an interaction between a factor and a covariate. To choose the best pairing, we should join the couple of variables with the worst correlation in order to create a new sort of variable that will give greater information to our model. On the other hand, if we join 2 variables with strong correlation, we would be just adding redundant information because 1 variable explaing the other. From our previous analyses, we found that tax and manufacturer were not very correlated so we will pair those 2 variables.

```
kruskal.test(df_clean$tax ~ df_clean$manufacturer)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  df_clean$tax by df_clean$manufacturer
## Kruskal-Wallis chi-squared = 47.859, df = 3, p-value = 2.281e-10
```

```
boxplot(df_clean$tax ~ df_clean$manufacturer)  #not so strong correlation
```




```
m6 <- update(m5, ~. + tax:manufacturer, data = df_clean)
# summary(m6) summary output: Residual standard error: 0.1781 on 4803
# degrees of freedom Multiple R-squared: 0.843, Adjusted R-squared:
# 0.842 F-statistic: 805.9 on 32 and 4803 DF, p-value: < 2.2e-16
```

We will now compare model 4 and model 6 to see if the upgrade is worth it.

```
# let's see if our model with improved interactions offers some
# difference from our previous model
anova(m6, m4)  #it does
```

```
## Analysis of Variance Table
##
## Model 1: log(price) ~ mileage + sqrt(tax) + I(mpg^-2) + I(years_after_sell^2) +
##     fuelType + engineSize + transmission + manufacturer + engineSize:manufacturer +
##     fuelType:manufacturer + manufacturer:tax + fuelType:engineSize:manufacturer
## Model 2: log(price) ~ mileage + sqrt(tax) + I(mpg^-2) + I(years_after_sell^2) +
##     fuelType + transmission + engineSize + manufacturer
##   Res.Df    RSS  Df Sum of Sq      F    Pr(>F)
## 1   4803 152.33
## 2   4822 159.46 -19   -7.1314 11.834 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# let's compare the AIC of both models to see which has the better
# fit
abs(AIC(m6, m4))  #model 4 offers a better fit
```

```
##    df      AIC
## m6 34 2929.833
## m4 15 2746.579
```

```
abs(BIC(m6, m4))  #model 4 offers a better fit
```

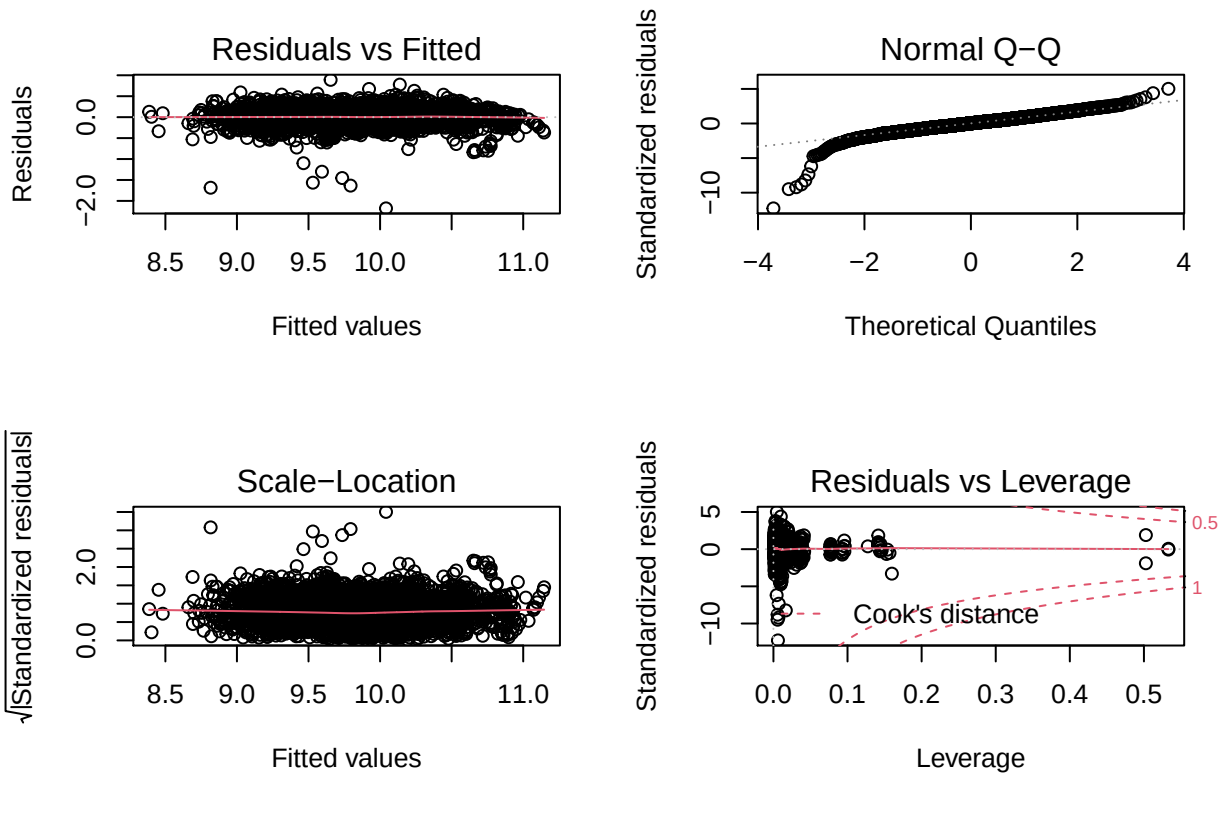
```
##    df      BIC
## m6 34 2709.383
## m4 15 2649.321
```

```
# model 6 = 0.843 model 4 = 0.8356
```

Model 4 also offers a greater overall fit as the AIC & BIC scores are greater, but model 6 offers a greater residual squared score, so we will stay with model 6 just for performance reasons.

3.5 Treatment of influential data

```
par(mfrow = c(2, 2))
plot(m6, id.n = 0)
```




```
par(mfrow = c(1, 1))
```

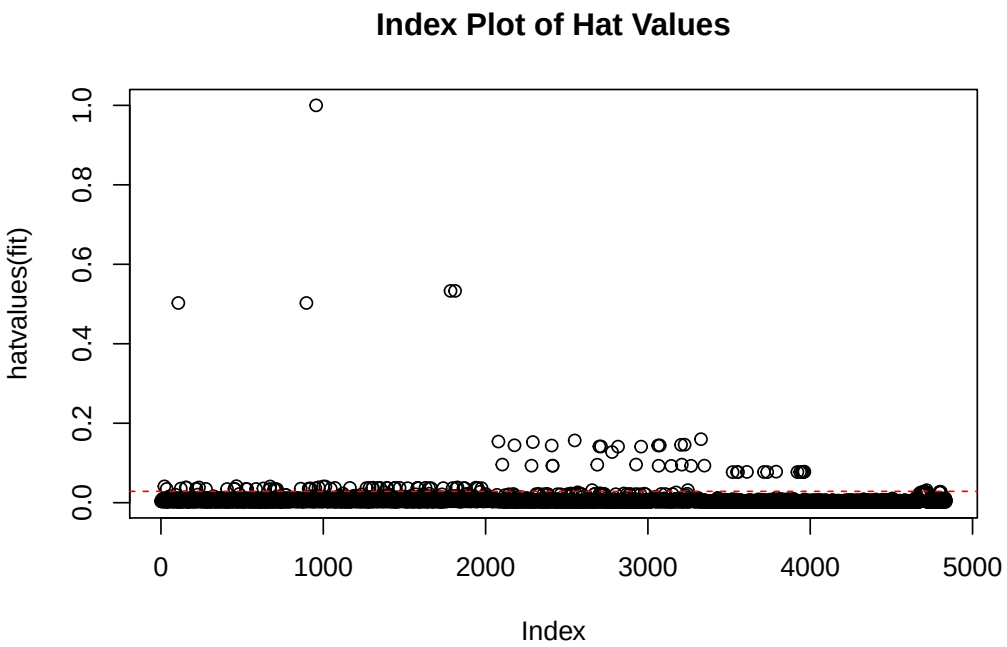
In the residual plots we can see the same patterns as with our previous models: the model checks the linear hypothesis, presents homoscedasticity and normality. This suggests a level of remarkable consistency with the predictions of the model. We can also see some high residuals, despite having treated outlying observations during preprocessing and having eliminated multivariate outliers.

In order to the fit of our model we will treat influential observations. Influence is the combination of the leverage and outlierness of an observation. * Leverage: An observation with an extreme value on a predictor variable is a point with high leverage. Leverage is a measure of how far an independent variable deviates from its mean. High leverage points can have a great amount of effect on the estimate of regression coefficients. * Influence: An observation is said to be influential if removing the observation substantially changes the estimate of the regression coefficients. Influence can be thought of as the product of leverage and outlierness.

```
library(olsrr)
```

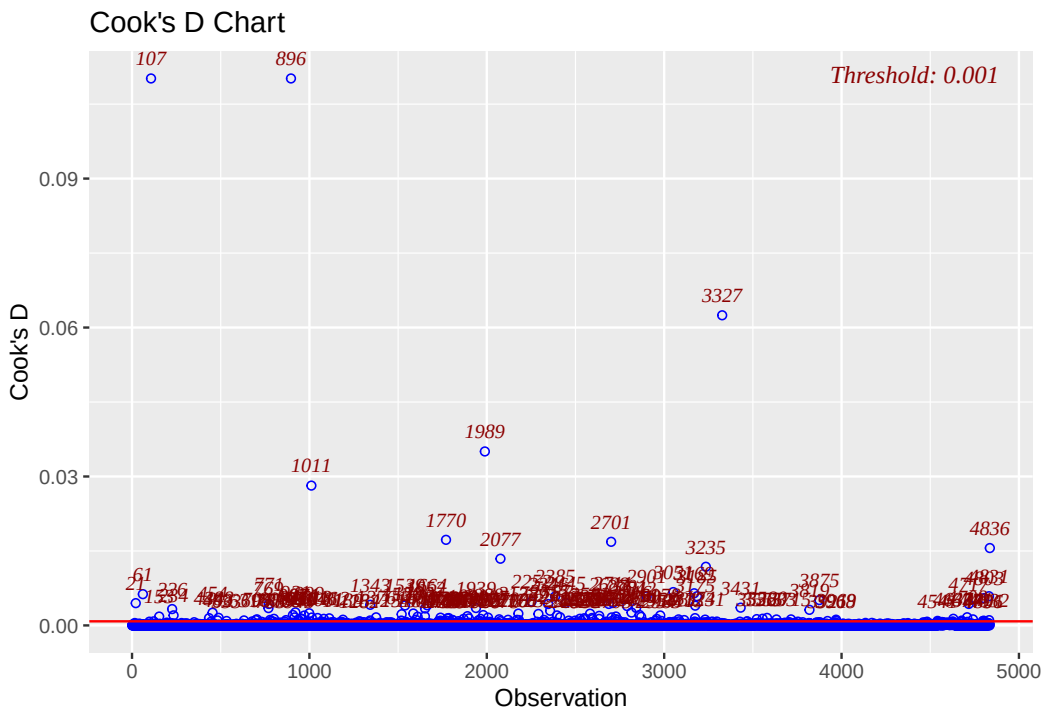
```
##
## Attaching package: 'olsrr'

## The following object is masked from 'package:MASS':
##
##     cement

## The following object is masked from 'package:datasets':
##
##     rivers
```

```r
hat.plot <- function(fit) {
    p <- length(coefficients(fit))
    n <- length(fitted(fit))
    plot(hatvalues(fit), main = "Index Plot of Hat Values")
    abline(h = 3 * p/n, col = "red", lty = 2)
}
hat.plot(m6)
```



As a priori observation of influence we will use the hat value that is the most common measure of leverage. From the hat plot generated we can see we have a lot of observations with high leverage (the points outside the red line).

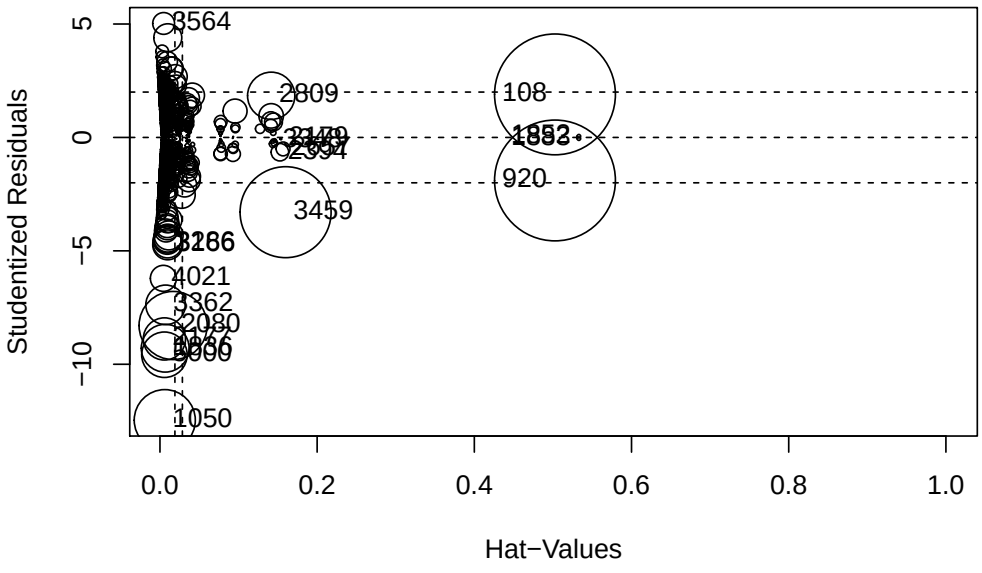
```r
ols_plot_cooksd_chart(m6)
```



As a posteriori observation of influence we will use the cook's distance because it is the most common measure of influence. From the cook's D chart generated we can see we have a lot of observations with high influential level.

Finally to choose the influential observations we will use Cook's distance because it offers a consistent method for detecting them. We could use a mixture of hat values (leverage obs.) and rstudent residuals (outlying obs.) to predict influential observations, but it is not as efficient and reliable.

```r
# Define initial parameters to calculate obs. with high leverage:
p <- length(m6$coefficients)
n <- length(m6$fitted.values)
hat_param <- 3
# we use as a threshold to label the leverage of the obs 3*p/n
# because we have a large dataset

# A priori influential observation
influencePlot(m6, id = list(n = 10))
```

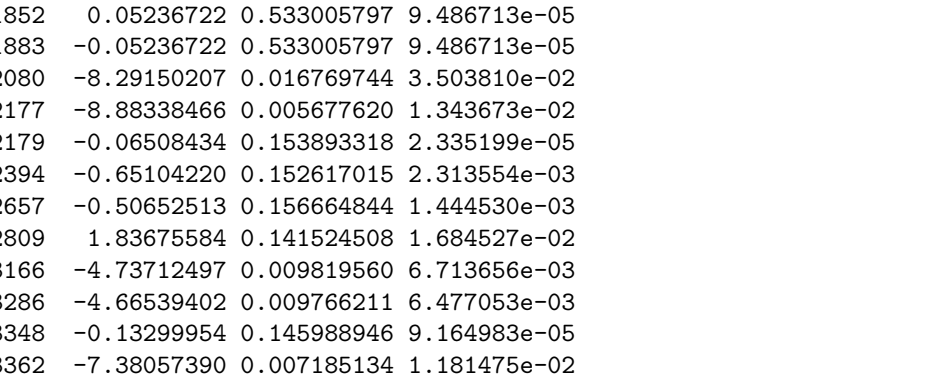



```
##           StudRes         Hat        CookD
## 108    1.89726458 0.502616724 1.101673e-01
## 920   -1.89726458 0.502616724 1.101673e-01
## 986           NaN 1.000000000          NaN
## 1050 -12.46413383 0.006137122 2.816509e-02
## 1836  -9.29183475 0.006661950 1.724029e-02
## 1852   0.05236722 0.533005797 9.486713e-05
## 1883  -0.05236722 0.533005797 9.486713e-05
## 2080  -8.29150207 0.016769744 3.503810e-02
## 2177  -8.88338466 0.005677620 1.343673e-02
## 2179  -0.06508434 0.153893318 2.335199e-05
## 2394  -0.65104220 0.152617015 2.313554e-03
## 2657  -0.50652513 0.156664844 1.444530e-03
## 2809   1.83675584 0.141524508 1.684527e-02
## 3166  -4.73712497 0.009819560 6.713656e-03
## 3286  -4.66539402 0.009766211 6.477053e-03
## 3348  -0.13299954 0.145988946 9.164983e-05
## 3362  -7.38057390 0.007185134 1.181475e-02
## 3459  -3.29491780 0.159853184 6.246705e-02
## 3564   5.02538321 0.004679440 3.579874e-03
## 4021  -6.21211027 0.004383399 5.108548e-03
## 5000  -9.57966983 0.005678736 1.558767e-02
```

```
ll_priori_influential <- which(abs(hatvalues(m6)) > hat_param * (p/n))
length(ll_priori_influential)
```

```
## [1] 108
```

```
boxplot(abs(rstudent(m6)))
quants_rst <- calcQ(abs(rstudent(m6)))
abline(h = quants_rst$souts, col = "red")
```



```
quants_rst$souts
```

```
##  3rd Qu.
## 3.249514
```

```
# let's choose as rstudent threshold the severe outlier threshold
ll_rst <- which(abs(rstudent(m6)) > quants_rst$souts)
length(ll_rst)
```

```
## [1] 27
```

```r
ll_hat_rst <- unique(c(ll_priori_influential, ll_rst))
length(ll_hat_rst)
```

```
## [1] 134
```

```r
# A posteriori influential observation:
ll_posteriori_influential <- which(abs(cooks.distance(m6)) > (4/(n - p)))
length(ll_posteriori_influential)
```

```
## [1] 148
```

```r
# a general rule of thumb is that D(i) > 4/(n-p) is a good threshold
# for determining highly influential points for cook's distance

m7_hat_rst <- update(m6, data = df_clean[-ll_hat_rst, ])
# summary(m7_hat_rst) Summary output: Residual standard error: 0.1609
# on 4678 degrees of freedom Multiple R-squared: 0.867, Adjusted
# R-squared: 0.8663 F-statistic: 1326 on 23 and 4678 DF, p-value: <
# 2.2e-16

m7_cook <- update(m6, data = df_clean[-ll_posteriori_influential, ])
# summary(m7_cook) summary output: Residual standard error: 0.1536 on
# 4656 degrees of freedom Multiple R-squared: 0.8745, Adjusted
# R-squared: 0.8737 F-statistic: 1047 on 31 and 4656 DF, p-value: <
# 2.2e-16

m7 <- m7_cook
```

As expected, the model with the removal of influential data using cook's distance has a better fit.

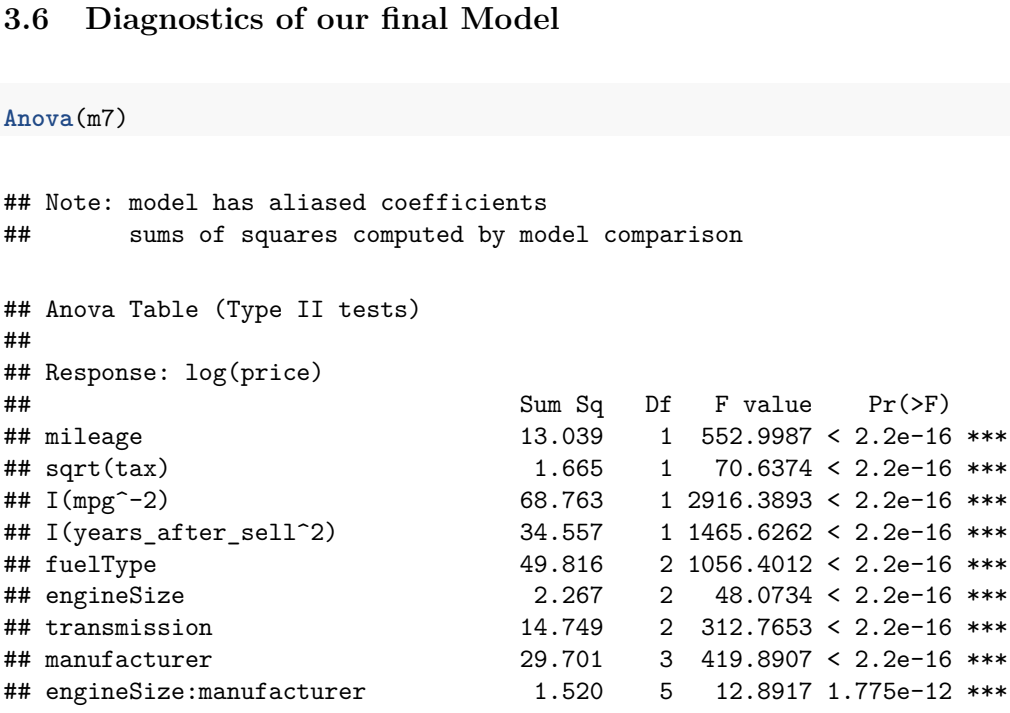
3.6 Diagnostics of our final Model

```r
Anova(m7)
```

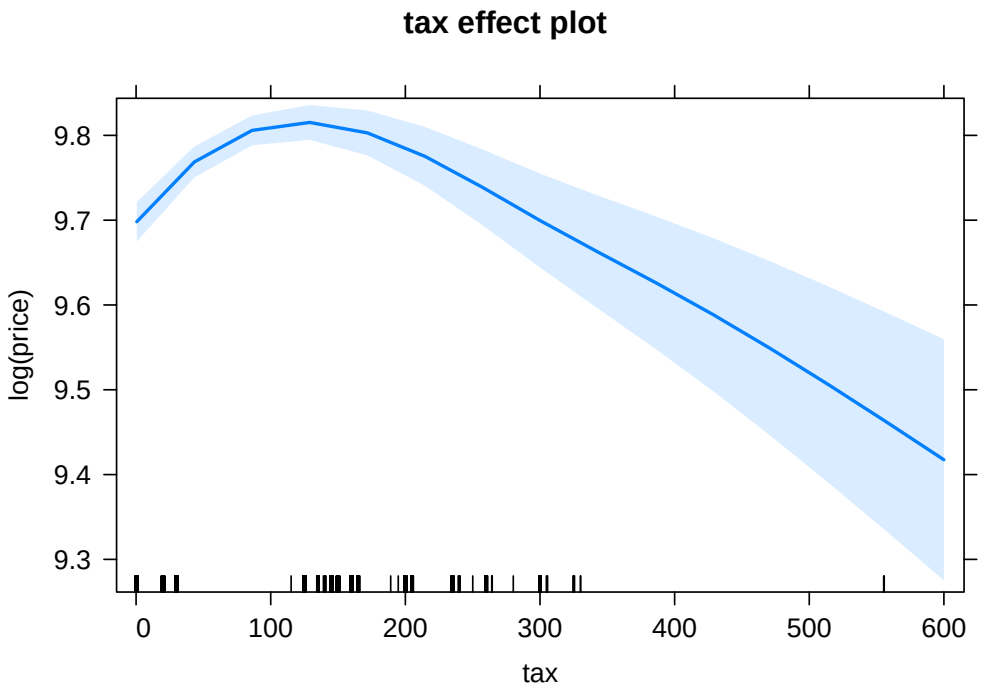
```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
## Anova Table (Type II tests)
##
## Response: log(price)
##                                 Sum Sq   Df   F value    Pr(>F)
## mileage                         13.039    1  552.9987 < 2.2e-16 ***
## sqrt(tax)                        1.665    1   70.6374 < 2.2e-16 ***
## I(mpg^-2)                       68.763    1 2916.3893 < 2.2e-16 ***
## I(years_after_sell^2)           34.557    1 1465.6262 < 2.2e-16 ***
## fuelType                        49.816    2 1056.4012 < 2.2e-16 ***
## engineSize                       2.267    2   48.0734 < 2.2e-16 ***
## transmission                    14.749    2  312.7653 < 2.2e-16 ***
## manufacturer                    29.701    3  419.8907 < 2.2e-16 ***
## engineSize:manufacturer          1.520    5   12.8917 1.775e-12 ***
## fuelType:manufacturer            2.299    5   19.5019 < 2.2e-16 ***
## manufacturer:tax                 1.415    4   15.0029 3.451e-12 ***
## fuelType:engineSize:manufacturer 0.748    4    7.9337 2.272e-06 ***
## Residuals                      109.780 4656
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

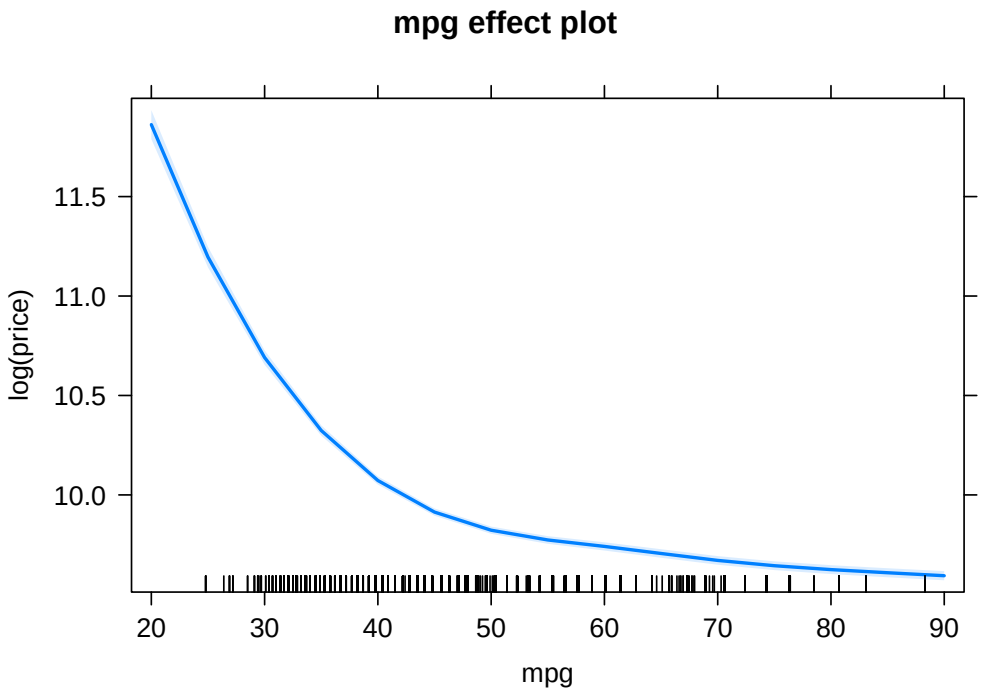
```r
# all variables and interactions are relevant
par(mfrow = c(1, 1))
plot(allEffects(m7), selection = 1)
```



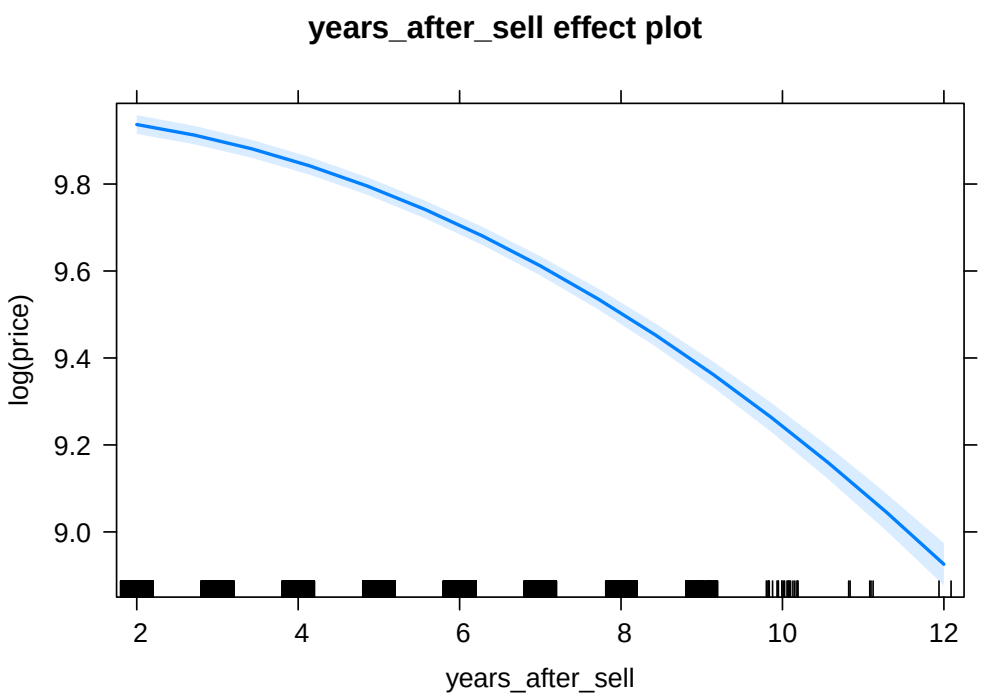

```r
plot(allEffects(m7), selection = 2)
```



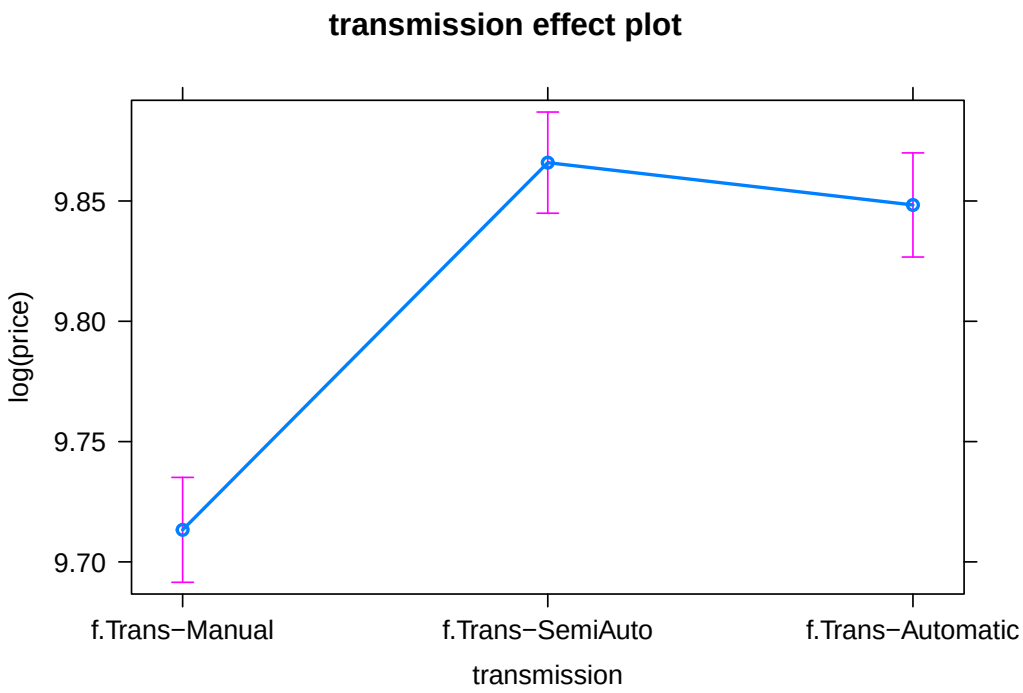

```
plot(allEffects(m7), selection = 3)
```



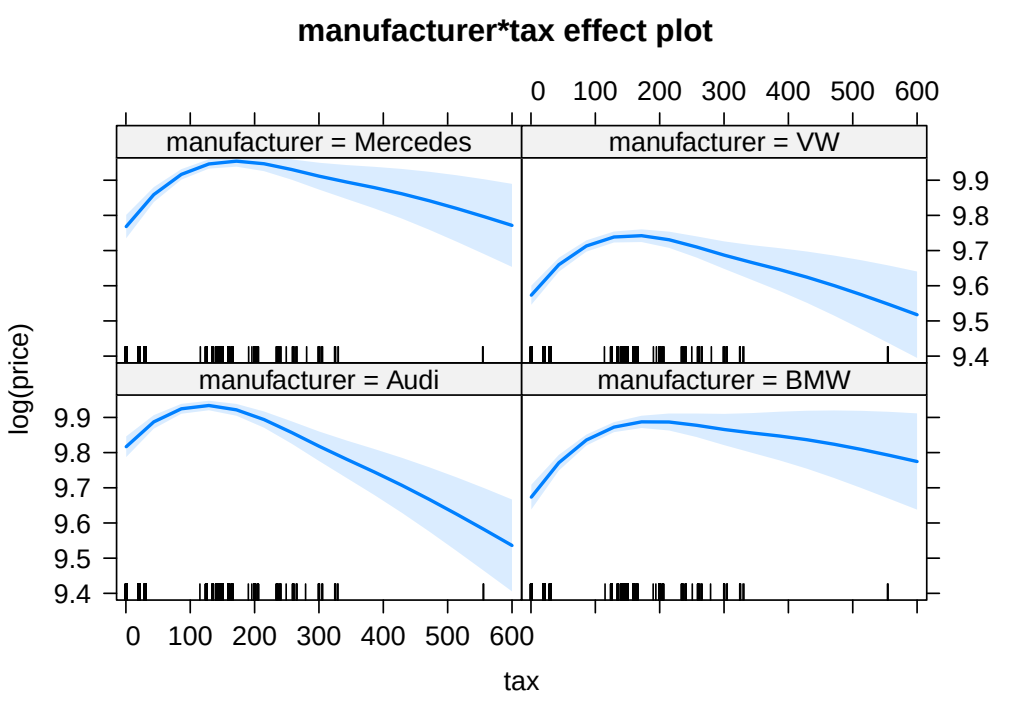

```
plot(allEffects(m7), selection = 4)
```



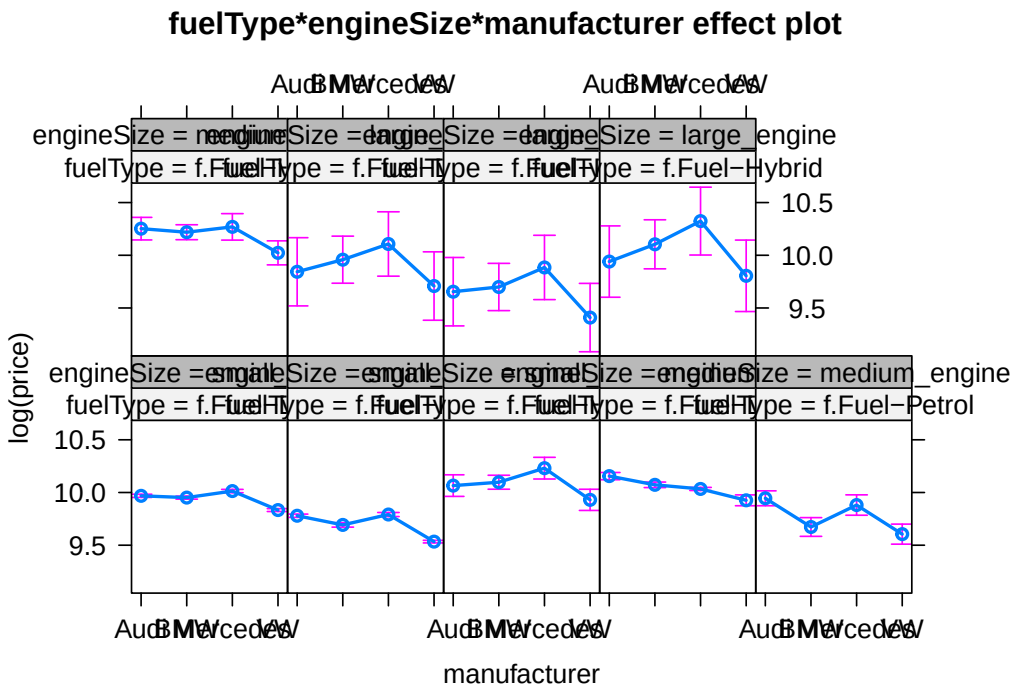

```
plot(allEffects(m7), selection = 5)
```




```
plot(allEffects(m7), selection = 6)
```



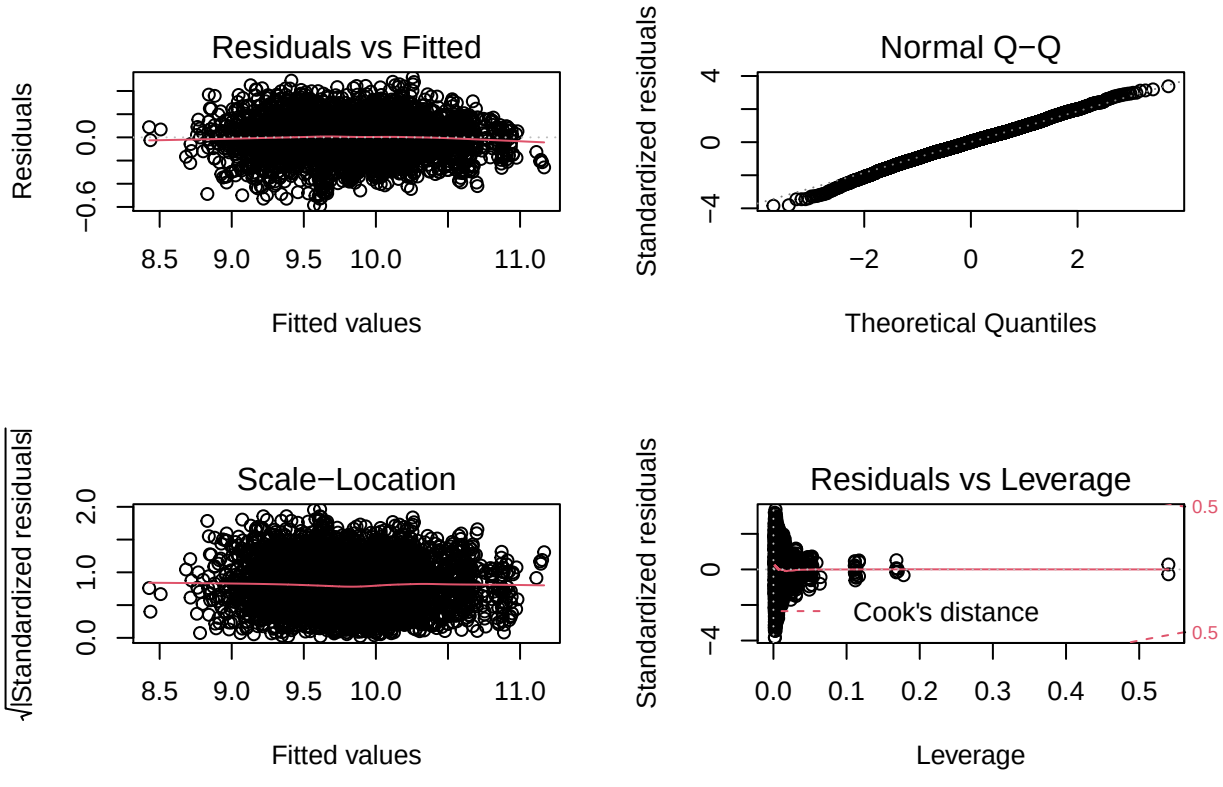

```
plot(allEffects(m7), selection = 7)
```



We see that our model defines the following:

- the log(price) decreases in a linear way as the mileage grows
- the log(price) increases as tax increases until tax=140, then log(price) decreases as tax increases
- the log(price) decreases in an exponential way as the mpg grows
- the log(price) decreases in a quadratic way (-xˆ2) as the years_after_sell grows
- the log(price) is higher with BMW, Merc and Audi cars than VW cars
- the log(price) is higher with diesel and hybrid cars than petrol cars
- the log(price) is higher with automatic and semi-automatic cars than manual cars
- the log(price) is higher with larger engines

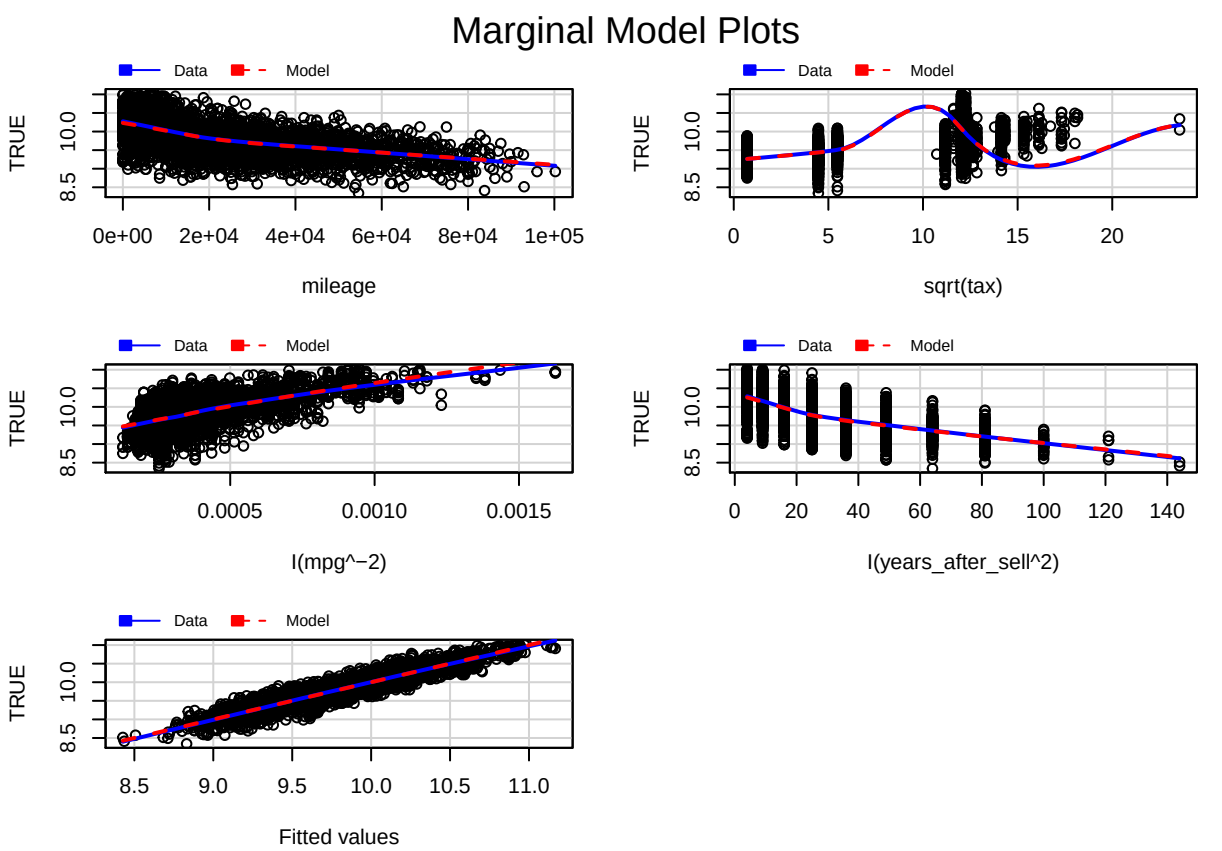
```
par(mfrow = c(2, 2))
plot(m7, id.n = 0)
```




```
par(mfrow = c(1, 1))
```

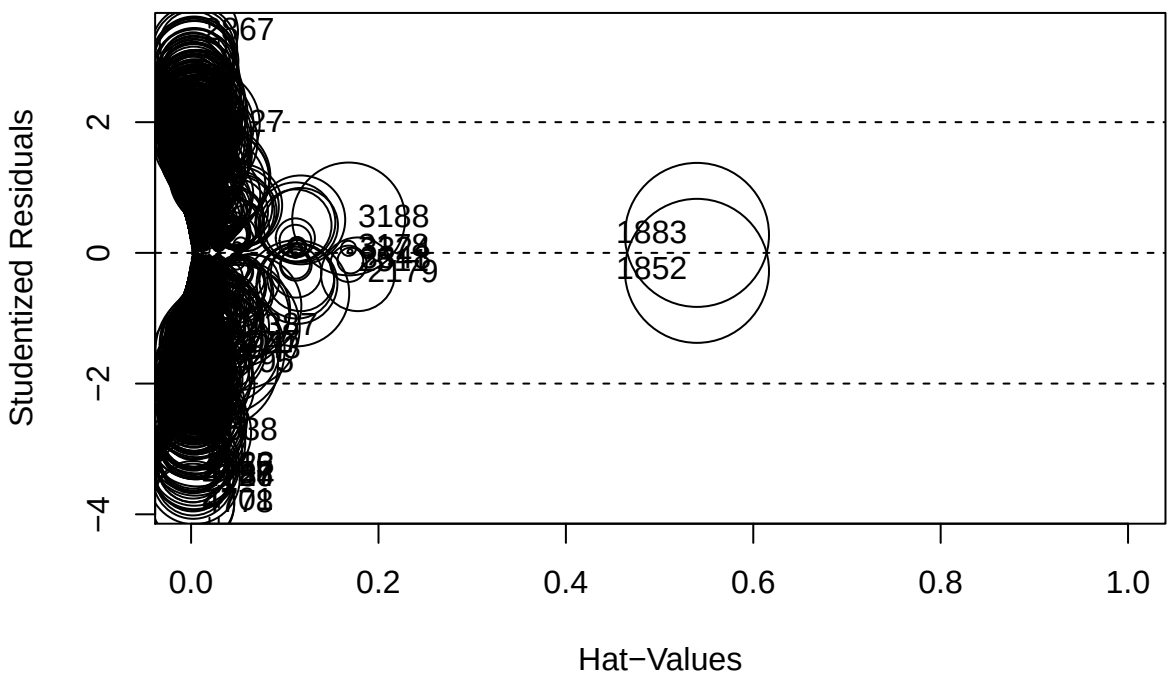
In the residual plots we can see the same patter as before, the model presents homoscedasticity and normality and fits the linear hypothesis. This suggests a level of remarkable consistency with the predictions of the model. We can also see the effect of the deletion of infuential data as there are not outlying residuals.

```
marginalModelPlots(m7)
```



We can see from our plots that the model fits perfectly with the data.

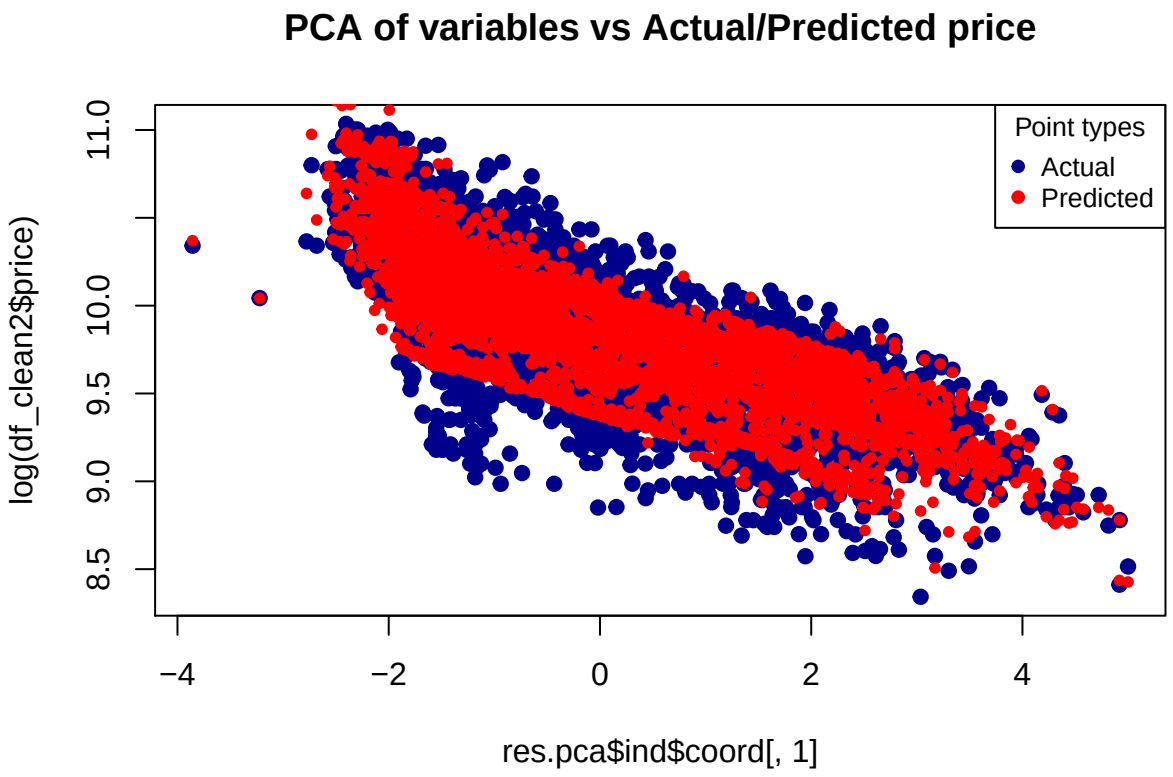
```
# avPlots(m7) We will abstain from plotting the added variable plots
# because they do not give us additional information and we would
# also need to analyze 45 graphs
par(mfrow = c(1, 1))
influencePlot(m7, id = list(n = 10))
```




```
##          StudRes         Hat        CookD
## 986          NaN 1.000000000          NaN
## 1027  1.97238659 0.013551413 1.669071e-03
## 1387 -1.14249518 0.049006920 2.101892e-03
## 1670 -1.41301780 0.023922058 1.528855e-03
## 1795 -1.70947885 0.023871181 2.232364e-03
## 1852 -0.27501741 0.539904773 2.774125e-03
## 1883  0.27501741 0.539904773 2.774125e-03
## 2179 -0.32758152 0.177976013 7.261880e-04
## 2511 -0.16709009 0.167988010 1.761939e-04
## 2843 -1.49722583 0.030842740 2.228782e-03
## 2867  3.38205879 0.002914921 1.042640e-03
## 2889         NaN 1.000000000          NaN
## 3047 -1.39848358 0.031064244 1.959032e-03
## 3178  0.06950025 0.167985457 3.048290e-05
## 3188  0.51906983 0.168192027 1.702755e-03
## 3324  0.02162597 0.169716619 2.988081e-06
## 3348 -0.11754572 0.170050111 8.848726e-05
## 4638 -2.74941765 0.006536627 1.552108e-03
## 4701 -3.81823263 0.002011635 9.156591e-04
## 4715 -3.28605486 0.001909636 6.442692e-04
## 4723 -3.26201056 0.002980444 9.919727e-04
## 4727 -3.45452211 0.002313612 8.627874e-04
## 4746 -3.45464717 0.001939676 7.231212e-04
## 4756 -3.28359766 0.001920424 6.469494e-04
## 4757 -3.39732387 0.001982121 7.147161e-04
## 4778 -3.85434330 0.001948349 9.035956e-04
## 4784 -3.45073935 0.003024189 1.126114e-03
```

From a look at the influence plot, we can see a clar difference from the influence plot of model6. We can see now that we do not have any influential observation. We only have some obs. with some leverage but without being outliers and vice versa.

```
df_clean2 = df_clean[-ll_posteriori_influential, ]
res.pca <- PCA(df_clean2[c(5, 7, 8, 13)], graph = FALSE)
plot(res.pca$ind$coord[, 1], log(df_clean2$price), pch = 19, col = "darkblue",
    main = "PCA of variables vs Actual/Predicted price")
points(res.pca$ind$coord[, 1], m7$fitted.values, col = "red", pch = 20)
legend("topright", legend = c("Actual", "Predicted"), col = c("darkblue",
    "red"), pch = 19, cex = 0.8, title = "Point types")
```




```
# plot(x=c(1:length(m7$fitted.values)), log(df_clean2$price),
# col='darkgreen', pch=20) points(x=c(1:length(m7$fitted.values)),
# y=m7$fitted.values, col='red', pch=19)
```

As an experiment to see how the model was behaving, we performed a pca with just numeric variables and we plotted the first dimension of the pca in the x-axis and the actual price and predicted price on the y-axis. We can see from our graph that our model does very good predictions as it almost fits all the data. With a perfect model, we would have a perfect overlapping of the 2 points.

As a final experiment to see a practical usage of linear regression, we will try to predict the price of a used Audi Q3 from the UK as our dataset is from the UK. The used car can be found here: https://www.autotrader.co.uk/car-details/202205296279195 with its specifications being imputed in the predict funcion.

```
pre <- predict(m7, newdata = data.frame(transmission = "f.Trans-Automatic",
    mileage = 66.579, fuelType = "f.Fuel-Petrol", tax = 220, mpg = 42.8,
    engineSize = "small_engine", manufacturer = "Audi", years_after_sell = 6),
    interval = "prediction")

# we transform the prediction from log(price) to normal price
I(exp(1)^pre)
```

```
##        fit     lwr      upr
## 1 21141.53 15625.2 28605.34
```

```
error <- (18950 - 21141.53)/18950 * 100
error
```

```
## [1] -11.5648
```

We get as a result a 95% confidence interval of [15625.2 28605.34] with the actual predicted price being 21141.53 real price being £18,950 so our model has an error of 11.5%, this is more or less the expected error because of our model expains 87.45% of the variability.

4 Binary Logistics Regression

4.1 Split into train and test:

```
# clean workspace and load again
rm(list = ls())
filepath <- "C:/Users/Arnau/Desktop/adei/deliverable3/"
load(paste0(filepath, "5000cars_clean.RData"))

# 80% train sample and 20% test sample
llwork <- sample(1:nrow(df), round(0.8 * nrow(df), 0))

dfall <- df
df_train <- dfall[llwork, ]
df_test <- dfall[-llwork, ]
```

We split our data in to parts, an 80% will be used to train the model and the other 20% to test it.

We are gonna follow the next iterations to get the best model to predict the binary variable Audi:

1. Using numerical explanatory variables
2. Introducing transformations
3. Excluding multivariate outliers
4. Excluding not contributory variables
5. Adding factors
6. Introducing interactions
7. Eliminating influential individuals

4.2 Using numerical explanatory variables

First of all we will introduce all the numeric variables to our model.

```r
vars_con
```

```
## [1] "mileage"          "tax"              "mpg"              "years_after_sell"
## [5] "inconsistencies"
```

```r
ll <- which(df_train$tax == 0)
ll
```

```
##   [1]   67   83  117  190  238  240  247  320  324  369  460  535  542  575  620
##  [16]  621  639  646  669  704  745  754  765  781  787  817  828  839  841  890
##  [31]  900  923  934  959 1001 1024 1049 1092 1142 1202 1256 1284 1286 1289 1305
##  [46] 1330 1345 1413 1445 1505 1551 1563 1743 1755 1790 1815 1831 1852 1877 1903
##  [61] 1933 1934 1980 1981 1999 2013 2029 2052 2067 2071 2074 2081 2102 2106 2111
##  [76] 2129 2140 2160 2200 2216 2232 2258 2314 2318 2338 2358 2362 2372 2398 2401
##  [91] 2405 2478 2503 2506 2541 2555 2566 2630 2639 2640 2652 2678 2714 2734 2764
## [106] 2818 2872 2893 2912 2986 3018 3019 3061 3071 3082 3115 3135 3182 3186 3215
## [121] 3222 3227 3293 3296 3304 3310 3324 3326 3373 3441 3443 3473 3507 3538 3590
## [136] 3611 3624 3628 3680 3758 3776 3777 3787 3803 3811 3866 3941 3944 3951 3956
```

```r
df$tax[ll] <- 0.5

m1 <- glm(Audi ~ price + mileage + tax + mpg + years_after_sell, family = "binomial",
    data = df_train)
summary(m1)
```

```
##
## Call:
## glm(formula = Audi ~ price + mileage + tax + mpg + years_after_sell,
##     family = "binomial", data = df_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7117  -0.7297  -0.6432  -0.4945   2.1578
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)       1.343e-01  4.231e-01   0.317 0.750982
## price             1.735e-05  4.575e-06   3.794 0.000148 ***
## mileage           1.287e-05  2.783e-06   4.625 3.75e-06 ***
## tax              -3.461e-03  7.648e-04  -4.525 6.04e-06 ***
## mpg              -3.355e-02  5.255e-03  -6.383 1.74e-10 ***
## years_after_sell  1.978e-02  3.191e-02   0.620 0.535391
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4232.9  on 3999  degrees of freedom
## Residual deviance: 4112.8  on 3994  degrees of freedom
## AIC: 4124.8
##
## Number of Fisher Scoring iterations: 4
```

```r
vif(m1)
```

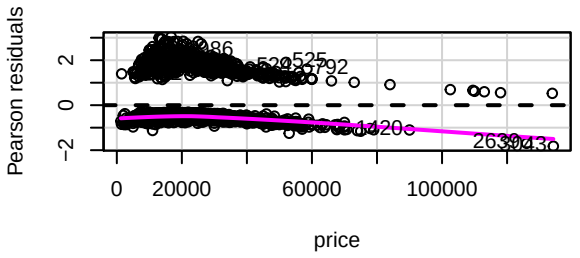
```
##            price          mileage              tax              mpg
##         2.169826         2.751626         1.742452         2.568469
## years_after_sell
##         2.967046
```

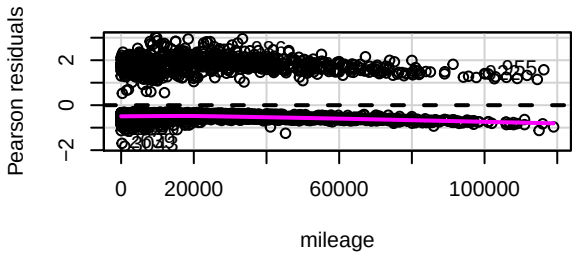
As we can see all variables contribute to the model except from years_after_sell, but we won't exclude it from the model because maybe including factors or interactions to the model it gains relevance. The reduction of residual deviance from the null model is 4232.9-4112.8=120.1.

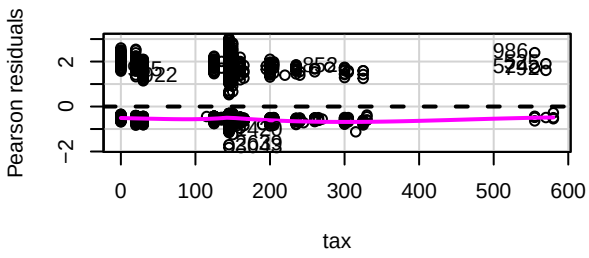
Interpreting the estimate of the explanatory variables we can see that when price, mileage and years_after_sell increase the probability of a car of being Audi increases. While tax and mpg have the opposite effect.

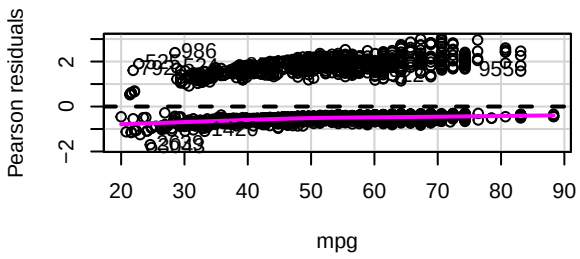
None of the variables of our model have a vif greater than 5 so they are independent with each other.

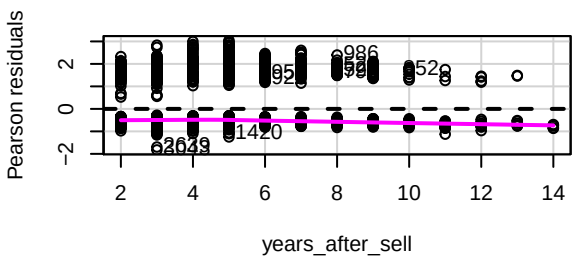
```r
residualPlots(m1, id = list(method = cooks.distance(m1), n = 10))
```

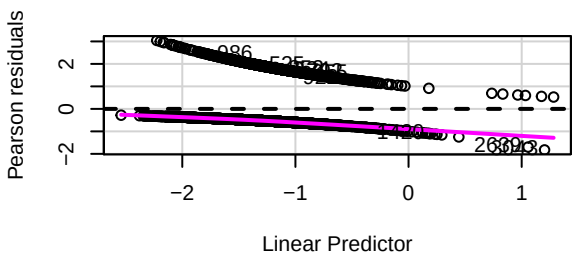






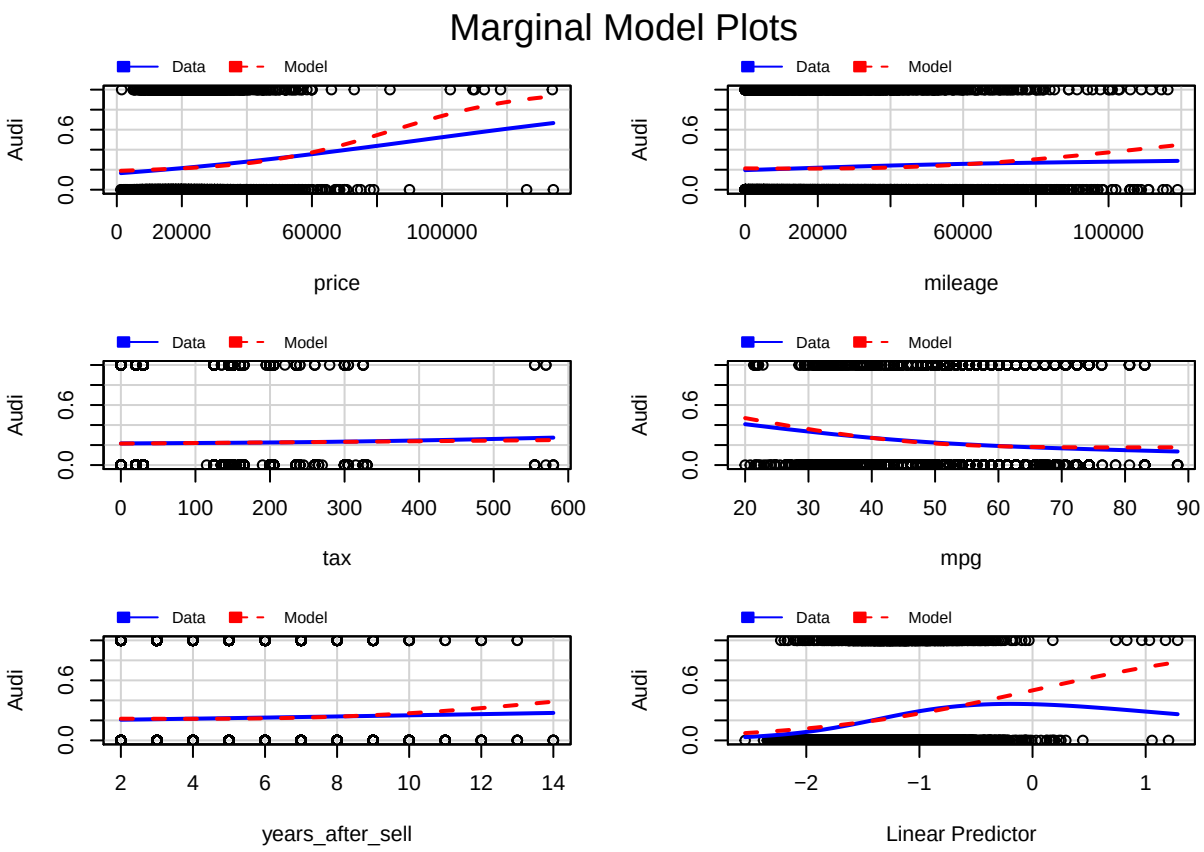






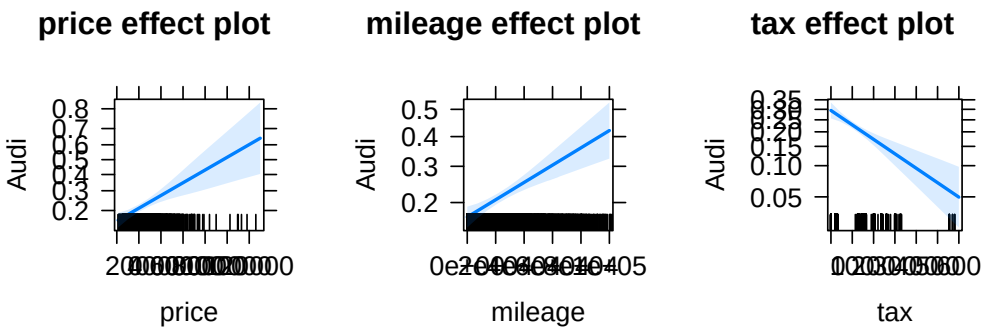

```
##                   Test stat Pr(>|Test stat|)
## price               14.7961        0.0001198 ***
## mileage             16.3351        5.307e-05 ***
## tax                  1.6081        0.2047553
## mpg                  1.2637        0.2609517
## years_after_sell     9.5592        0.0019895 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
marginalModelPlots(m1)
```

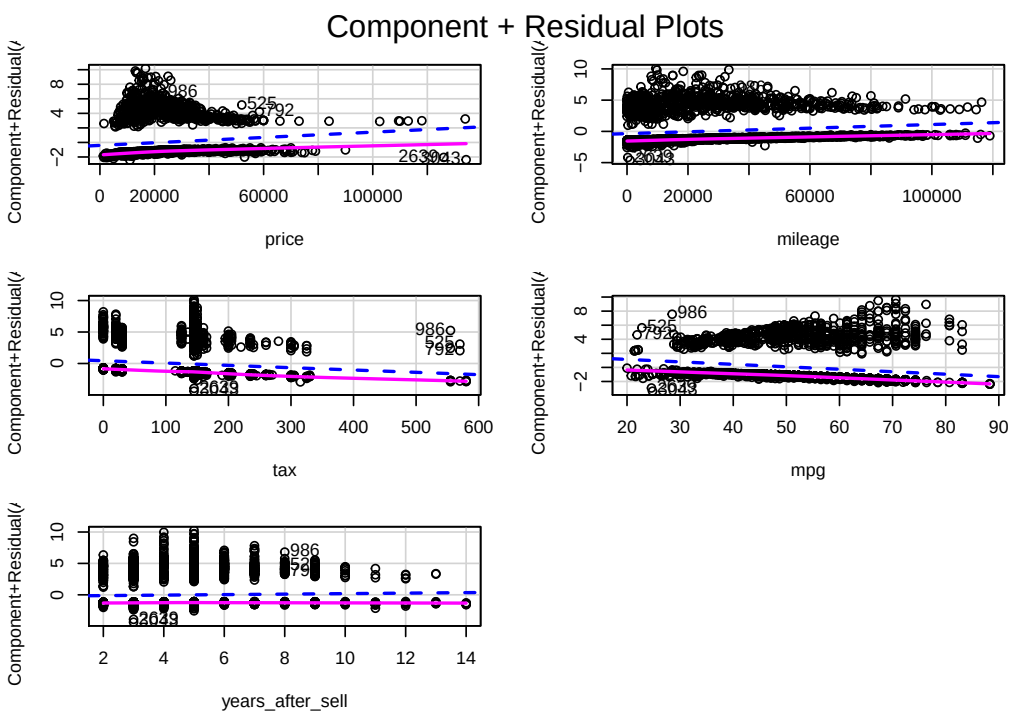


The variables price, mileage and years_after_sell doesn't fit well the data, so we will have to apply transformations.

```
library(effects)
plot(allEffects(m1))
```




```
crPlots(m1, id = list(method = cooks.distance(m1), n = 5))
```



Form the effect plots we see that our model defines the following:

- the probability of being Audi increases as the price grows
- the probability of being Audi increases as the mileage grows
- the probability of being Audi decreases as the tax grows
- the probability of being Audi decreases as the mpg grows
- the probability of being Audi increases as the years_after_sell grows

This corroborates the information seen in the summary about the estimate of the variables.

4.3 Introducing transformations

For the 2nd model we will introduce the polynomial transformation to all the variables and keep the contributory ones.

```
m2 <- glm(Audi ~ poly(price, 2) + poly(mileage, 2) + poly(tax, 2) + poly(mpg,
    2) + poly(years_after_sell, 2), family = "binomial", data = df_train)
summary(m2)
```

```
##
## Call:
## glm(formula = Audi ~ poly(price, 2) + poly(mileage, 2) + poly(tax,
##     2) + poly(mpg, 2) + poly(years_after_sell, 2), family = "binomial",
##     data = df_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5296  -0.7460  -0.6321  -0.4337   2.2983
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  -1.31998    0.04024 -32.802  < 2e-16 ***
## poly(price, 2)1              19.73732    3.90686   5.052 4.37e-07 ***
## poly(price, 2)2              -9.29265    2.80746  -3.310 0.000933 ***
## poly(mileage, 2)1            24.15912    4.37367   5.524 3.32e-08 ***
## poly(mileage, 2)2            -8.55838    3.22098  -2.657 0.007882 **
## poly(tax, 2)1               -18.00671    3.38670  -5.317 1.06e-07 ***
## poly(tax, 2)2                 4.44646    2.79207   1.593 0.111265
## poly(mpg, 2)1               -27.24737    4.06197  -6.708 1.97e-11 ***
## poly(mpg, 2)2                -3.84880    3.15053  -1.222 0.221845
## poly(years_after_sell, 2)1    1.84850    4.90427   0.377 0.706235
## poly(years_after_sell, 2)2   -2.24316    3.20818  -0.699 0.484427
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4232.9  on 3999  degrees of freedom
## Residual deviance: 4079.4  on 3989  degrees of freedom
## AIC: 4101.4
##
## Number of Fisher Scoring iterations: 4
```

The transformations in the variables tax, mpg and years_after_sell are not contributory to the model, we will exclude them according to the output of a step. The reduction of residual deviance from the null model is 4232.9-4079.4=153.5.

```
anova(m2, m1, test = "LR")
```

```
## Analysis of Deviance Table
##
## Model 1: Audi ~ poly(price, 2) + poly(mileage, 2) + poly(tax, 2) + poly(mpg,
##     2) + poly(years_after_sell, 2)
## Model 2: Audi ~ price + mileage + tax + mpg + years_after_sell
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1      3989     4079.4
## 2      3994     4112.8 -5  -33.342 3.218e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
AIC(m1, m2)
```

```
##    df      AIC
## m1  6 4124.791
## m2 11 4101.448
```

Nevertheless, the output of the anova test indicates us that the reduction of residuals deviance is significant because Pr(>Chi) = 3.218e-06 < 0.05.

Moreover, the AIC has been reduced.

Before excluding the not influential transformations, we will use the data_train without the multivariate outliers to modelize because they could affecting negatively to our model.

4.4 Excluding multivariate outliers

```
m3 <- glm(Audi ~ poly(price, 2) + poly(mileage, 2) + poly(tax, 2) + poly(mpg,
    2) + poly(years_after_sell, 2), family = "binomial", data = df_train[!df_train$mout ==
    "MvOut.Yes", ])
summary(m3)
```

```
##
## Call:
## glm(formula = Audi ~ poly(price, 2) + poly(mileage, 2) + poly(tax,
##     2) + poly(mpg, 2) + poly(years_after_sell, 2), family = "binomial",
##     data = df_train[!df_train$mout == "MvOut.Yes", ])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3939  -0.7551  -0.6145  -0.3942   2.4477
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 -1.34980    0.04188 -32.230  < 2e-16 ***
## poly(price, 2)1             27.25231    4.21016   6.473 9.61e-11 ***
## poly(price, 2)2            -16.41107    3.05502  -5.372 7.79e-08 ***
## poly(mileage, 2)1           24.98576    4.56350   5.475 4.37e-08 ***
## poly(mileage, 2)2           -9.68566    3.26440  -2.967  0.00301 **
## poly(tax, 2)1              -23.74830    3.67567  -6.461 1.04e-10 ***
## poly(tax, 2)2                2.40986    2.75953   0.873  0.38251
## poly(mpg, 2)1              -30.22291    4.01396  -7.529 5.10e-14 ***
## poly(mpg, 2)2               -0.87950    3.05973  -0.287  0.77377
## poly(years_after_sell, 2)1   8.34801    5.10234   1.636  0.10182
## poly(years_after_sell, 2)2   1.05555    3.15775   0.334  0.73817
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4076.3  on 3862  degrees of freedom
## Residual deviance: 3891.3  on 3852  degrees of freedom
## AIC: 3913.3
##
## Number of Fisher Scoring iterations: 4
```

Despite of excluding the multivariate outliers the transformations without significance continue to not contribute to the model. The reduction of residual deviance from the null model is 4076.3-3891.3=185.

```
AIC(m1, m2, m3)
```

```
##    df      AIC
## m1  6 4124.791
## m2 11 4101.448
## m3 11 3913.333
```

The AIC has been reduced, so excluding the multivariate outliers has a good effect on the model.

Now we will use function step to improve our model. The function step select a formula-based model by the best AIC.

```
m4 <- step(m3)
```

```
## Start:  AIC=3913.33
## Audi ~ poly(price, 2) + poly(mileage, 2) + poly(tax, 2) + poly(mpg,
##     2) + poly(years_after_sell, 2)
##
##                             Df Deviance    AIC
## - poly(years_after_sell, 2)  2   3894.7 3912.7
## <none>                           3891.3 3913.3
## - poly(mileage, 2)           2   3922.6 3940.6
## - poly(tax, 2)               2   3935.6 3953.6
## - poly(price, 2)             2   3946.8 3964.8
## - poly(mpg, 2)               2   3952.2 3970.2
##
## Step:  AIC=3912.71
## Audi ~ poly(price, 2) + poly(mileage, 2) + poly(tax, 2) + poly(mpg,
##     2)
##
##                     Df Deviance    AIC
## <none>                   3894.7 3912.7
## - poly(tax, 2)       2   3938.3 3952.3
## - poly(price, 2)     2   3948.7 3962.7
## - poly(mpg, 2)       2   3960.1 3974.1
## - poly(mileage, 2)   2   3982.5 3996.5
```

```
summary(m4)
```

```
##
## Call:
## glm(formula = Audi ~ poly(price, 2) + poly(mileage, 2) + poly(tax,
##     2) + poly(mpg, 2), family = "binomial", data = df_train[!df_train$mout ==
##     "MvOut.Yes", ])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4321  -0.7549  -0.6135  -0.3967   2.4308
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -1.34709    0.04176 -32.255  < 2e-16 ***
## poly(price, 2)1    24.46199    3.90577   6.263 3.78e-10 ***
## poly(price, 2)2   -14.69967    2.86901  -5.124 3.00e-07 ***
## poly(mileage, 2)1  29.60146    3.34493   8.850  < 2e-16 ***
## poly(mileage, 2)2 -10.51593    2.62197  -4.011 6.05e-05 ***
## poly(tax, 2)1     -23.54489    3.64277  -6.463 1.02e-10 ***
## poly(tax, 2)2       3.19875    2.64787   1.208    0.227
## poly(mpg, 2)1     -30.99420    3.97775  -7.792 6.60e-15 ***
## poly(mpg, 2)2      -1.29141    3.05041  -0.423    0.672
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4076.3  on 3862  degrees of freedom
## Residual deviance: 3894.7  on 3854  degrees of freedom
## AIC: 3912.7
##
## Number of Fisher Scoring iterations: 4
```

The function step has eliminated the variables poly(years_after_sell,2)1 and poly(years_after_sell,2)2.

```
anova(m3, m4, test = "LR")
```

```
## Analysis of Deviance Table
##
## Model 1: Audi ~ poly(price, 2) + poly(mileage, 2) + poly(tax, 2) + poly(mpg,
##     2) + poly(years_after_sell, 2)
## Model 2: Audi ~ poly(price, 2) + poly(mileage, 2) + poly(tax, 2) + poly(mpg,
##     2)
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      3852     3891.3
## 2      3854     3894.7 -2  -3.3771   0.1848
```

```
AIC(m1, m2, m3, m4)
```

```
##    df      AIC
## m1  6 4124.791
## m2 11 4101.448
## m3 11 3913.333
## m4  9 3912.710
```

The residual deviance has increased (it's normal having less explanatory variables) but is not significant. We will keep m4 because it's AIC is lower than m2 and it is better to have less explanatory variables.

4.5 Excluding not contributory variables

We will extract from the transformations that are not contributory: poly(tax,2)2 and poly(mpg,2)2.

```
m5 <- glm(formula = Audi ~ poly(price, 2) + poly(mileage, 2) + tax + mpg,
    family = "binomial", data = df_train[!df_train$mout == "MvOut.Yes",
        ])
summary(m5)
```

```
##
## Call:
## glm(formula = Audi ~ poly(price, 2) + poly(mileage, 2) + tax +
##     mpg, family = "binomial", data = df_train[!df_train$mout ==
##     "MvOut.Yes", ])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4514  -0.7521  -0.6140  -0.4043   2.4120
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         1.704e+00  3.660e-01   4.656 3.22e-06 ***
## poly(price, 2)1     2.373e+01  3.704e+00   6.407 1.49e-10 ***
## poly(price, 2)2    -1.454e+01  2.825e+00  -5.148 2.63e-07 ***
## poly(mileage, 2)1   3.026e+01  3.287e+00   9.206  < 2e-16 ***
## poly(mileage, 2)2 -1.091e+01  2.600e+00  -4.196 2.72e-05 ***
## tax                -5.897e-03  9.208e-04  -6.404 1.51e-10 ***
## mpg                -4.383e-02  5.623e-03  -7.795 6.46e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4076.3  on 3862  degrees of freedom
## Residual deviance: 3896.0  on 3856  degrees of freedom
## AIC: 3910
##
## Number of Fisher Scoring iterations: 4
```

Now all the variables of the model are contributory. The residual deviance has increased again but now we will check if it is significant.

```
anova(m5, m4, test = "LR")
```

```
## Analysis of Deviance Table
##
## Model 1: Audi ~ poly(price, 2) + poly(mileage, 2) + tax + mpg
## Model 2: Audi ~ poly(price, 2) + poly(mileage, 2) + poly(tax, 2) + poly(mpg,
##     2)
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      3856     3896.0
## 2      3854     3894.7  2   1.3281   0.5148
```

```
AIC(m1, m2, m3, m4, m5)
```

```
##    df      AIC
## m1  6 4124.791
## m2 11 4101.448
## m3 11 3913.333
## m4  9 3912.710
## m5  7 3910.038
```

This increase of residual deviance isn't significant because Pr(>Chi) = 0.5148 > 0.05, so we can eliminate this variables without a major effect. Moreover the AIC has been reduced so we will keep m5.

4.6 Adding factors

The next model is gonna introduce factor variables.

```
table(df_train$engineSize)
```

```
##
##    1  1.2  1.3  1.4  1.5  1.6  1.8  1.9    2  2.1  2.2  2.3  2.5  2.9    3  3.2
##  300  104   58  282  415  293   18    2 1678  312   11    4    4   11  444    2
##  3.5  3.7    4  4.1  4.2  4.4  4.7  5.2  5.5  6.2  6.6
##    3    1   35    1    1    6    5    5    3    1    1
```

The factor variable engineSize has too many categories so we will reduce them to 3.

```
df_train$engineSize_f <- as.numeric(levels(df_train$engineSize))[df_train$engineSize]
par(mfrow = c(1, 1))
hist(df_train$engineSize_f)
```


```
quantile(df_train$engineSize_f, c(0.3333333, 0.6666666, 1))
```

```
## 33.33333% 66.66666%      100%
##       1.6       2.0       6.6
```

```
df_train$engineSize_f <- factor(cut(df_train$engineSize_f, breaks = c(0,
    1.6, 2, 10)))
df_test$engineSize_f <- as.numeric(levels(df_test$engineSize))[df_test$engineSize]
df_test$engineSize_f <- factor(cut(df_test$engineSize_f, breaks = c(0,
    1.6, 2, 10)))
table(df_train$engineSize_f)
```

```
##
## (0,1.6] (1.6,2]  (2,10]
##    1452    1698     850
```

By this way we create a new factor variable grouping the values of the variable engineSize.

```
m6 <- update(m5, ~. + fuelType + transmission + engineSize_f, data = df_train[!df_train$mout ==
    "MvOut.Yes", ])
vif(m6)
```

```
##                      GVIF Df GVIF^(1/(2*Df))
## poly(price, 2)   8.794658  2        1.722086
## poly(mileage, 2) 3.475861  2        1.365418
## tax              2.165002  1        1.471395
## mpg              4.281182  1        2.069102
## fuelType         2.418899  2        1.247109
## transmission     1.764955  2        1.152613
## engineSize_f     3.196572  2        1.337122
```

```
summary(m6)
```

```
##
## Call:
## glm(formula = Audi ~ poly(price, 2) + poly(mileage, 2) + tax +
##     mpg + fuelType + transmission + engineSize_f, family = "binomial",
##     data = df_train[!df_train$mout == "MvOut.Yes", ])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8757  -0.7361  -0.4987  -0.2144   3.0097
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    2.714317   0.523482   5.185 2.16e-07 ***
## poly(price, 2)1               86.286978   6.078472  14.196  < 2e-16 ***
## poly(price, 2)2              -28.822615   3.434645  -8.392  < 2e-16 ***
## poly(mileage, 2)1             68.568896   4.486559  15.283  < 2e-16 ***
## poly(mileage, 2)2            -21.608035   2.920445  -7.399 1.37e-13 ***
## tax                           -0.005099   0.001013  -5.036 4.75e-07 ***
## mpg                           -0.040781   0.007499  -5.438 5.38e-08 ***
## fuelTypef.Fuel-Petrol         -0.115366   0.129271  -0.892 0.372161
## fuelTypef.Fuel-Hybrid         -2.449760   0.734075  -3.337 0.000846 ***
## transmissionf.Trans-SemiAuto  -1.002030   0.120486  -8.317  < 2e-16 ***
## transmissionf.Trans-Automatic -0.891389   0.129521  -6.882 5.89e-12 ***
## engineSize_f(1.6,2]           -0.634448   0.124455  -5.098 3.44e-07 ***
## engineSize_f(2,10]            -2.351232   0.204360 -11.505  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 4076.3  on 3862  degrees of freedom
## Residual deviance: 3581.2  on 3850  degrees of freedom
## AIC: 3607.2
##
## Number of Fisher Scoring iterations: 5
```

All factor variables contribute to the model except for the category fuelTypef.Fuel-Petrol of the factor variable fuelType, but as the other categories contribute we should keep it. Besides, we will check if the factor variables are contributory with the function Anova. The reduction of residual deviance from the null model is 4076.3-3581.2=495.1.

- The baseline of the variable fuelType is the category fuelTypef.Fuel-Diesel, and fuelTypef.Fuel-Petrol and fuelTypef.Fuel-Hybrid contribute negatively.
- The baseline of the variable transmission is the category transmissionf.Trans-Manual, and transmissionf.Trans-Automatic and transmissionf.Trans-SemiAuto contribute negatively.
- The baseline of the variable engineSize_f is the category engineSize_f(0,1.4] , and engineSize_f(1.4,2] and engineSize_f(2,10] contribute negatively.

None of the variables of our model have a vif greater than 5 so they are independent with each other.

```
Anova(m6, test = "LR")
```

```
## Analysis of Deviance Table (Type II tests)
##
## Response: Audi
##                  LR Chisq Df Pr(>Chisq)
## poly(price, 2)    235.577  2  < 2.2e-16 ***
## poly(mileage, 2)  284.492  2  < 2.2e-16 ***
## tax                26.414  1  2.755e-07 ***
## mpg                30.435  1  3.453e-08 ***
## fuelType           22.540  2  1.275e-05 ***
## transmission       75.834  2  < 2.2e-16 ***
## engineSize_f      175.301  2  < 2.2e-16 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
```

All variables contribute to the model.

```
anova(m6, m5, test = "LR")
```

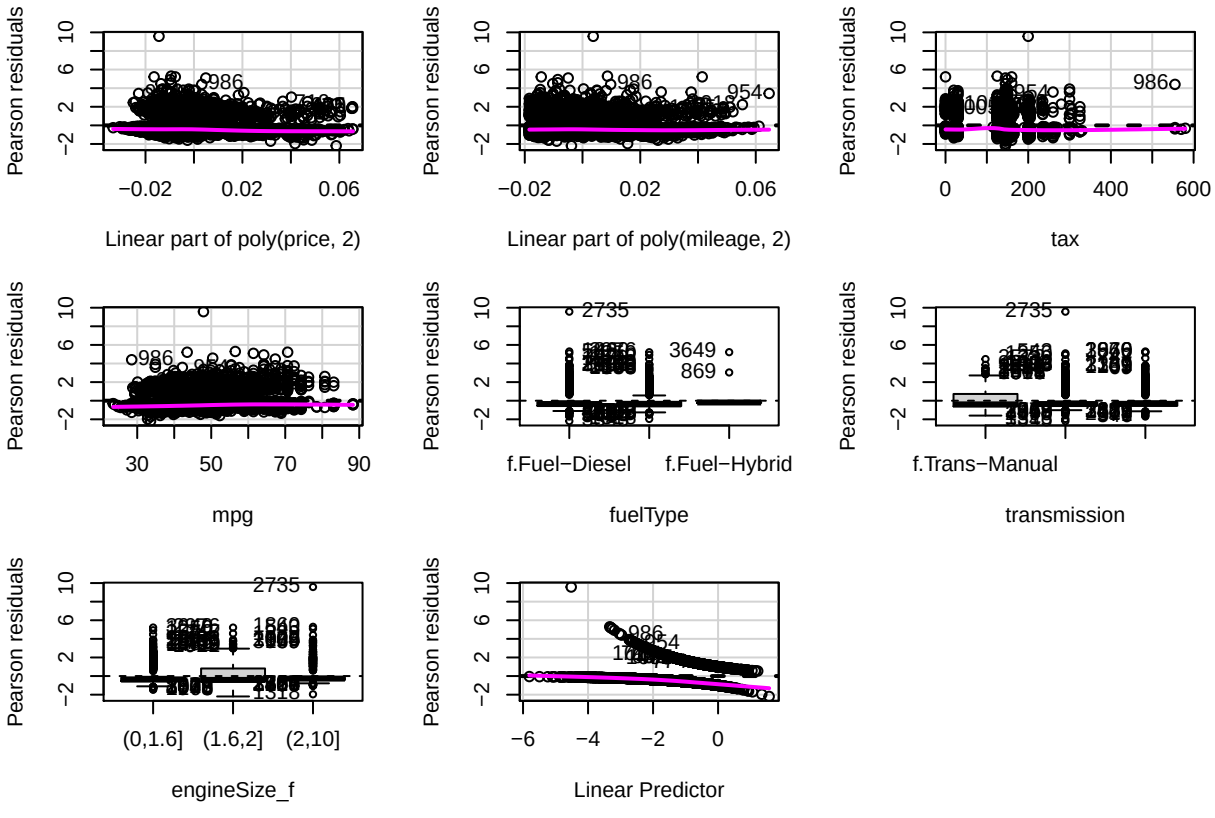
```
## Analysis of Deviance Table
##
## Model 1: Audi ~ poly(price, 2) + poly(mileage, 2) + tax + mpg + fuelType +
##     transmission + engineSize_f
## Model 2: Audi ~ poly(price, 2) + poly(mileage, 2) + tax + mpg
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1      3850     3581.2
## 2      3856     3896.0 -6  -314.84 < 2.2e-16 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
```

```
AIC(m1, m2, m3, m4, m5, m6)
```

```
##    df      AIC
## m1  6 4124.791
## m2 11 4101.448
## m3 11 3913.333
## m4  9 3912.710
## m5  7 3910.038
## m6 13 3607.197
```

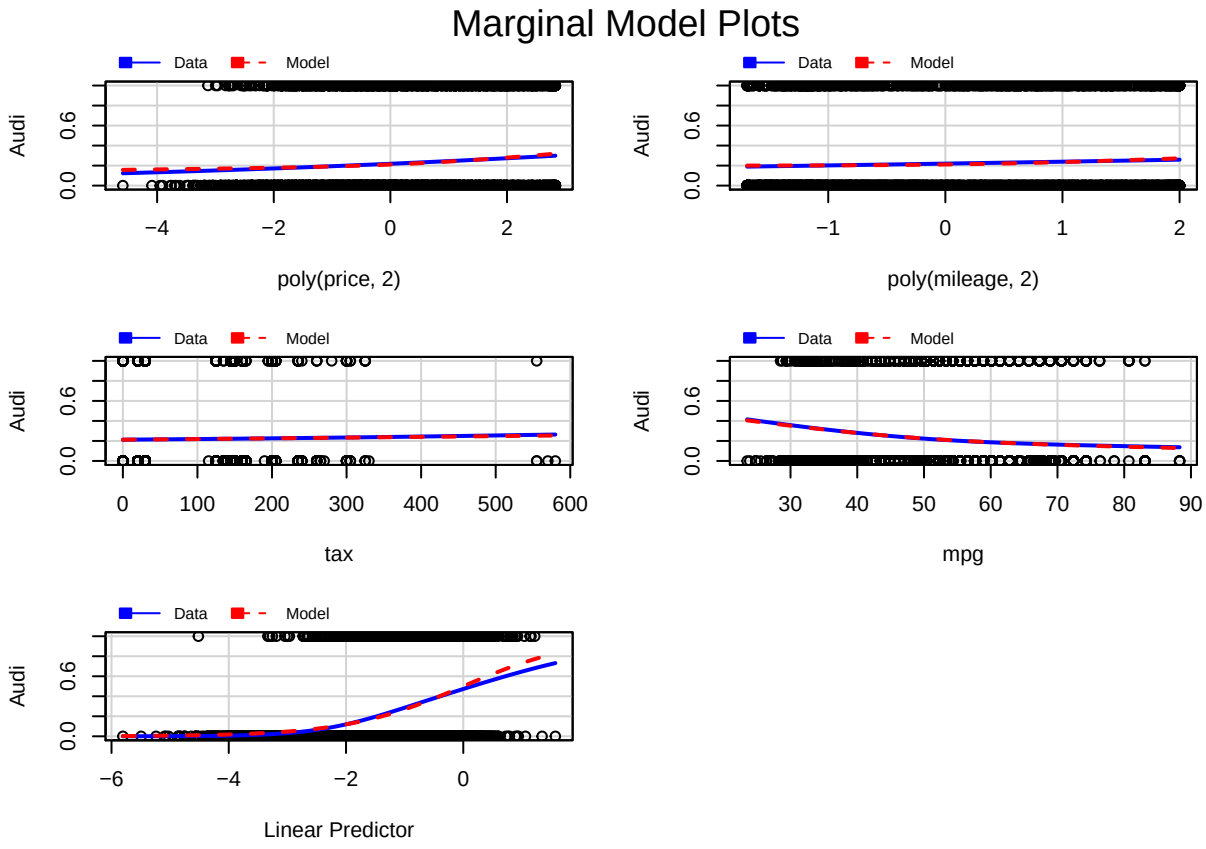
The reduction of the residual deviance is significant so we will keep m6. The AIC has also been reduced.

```
residualPlots(m6, id = list(method = cooks.distance(m5), n = 10))
```



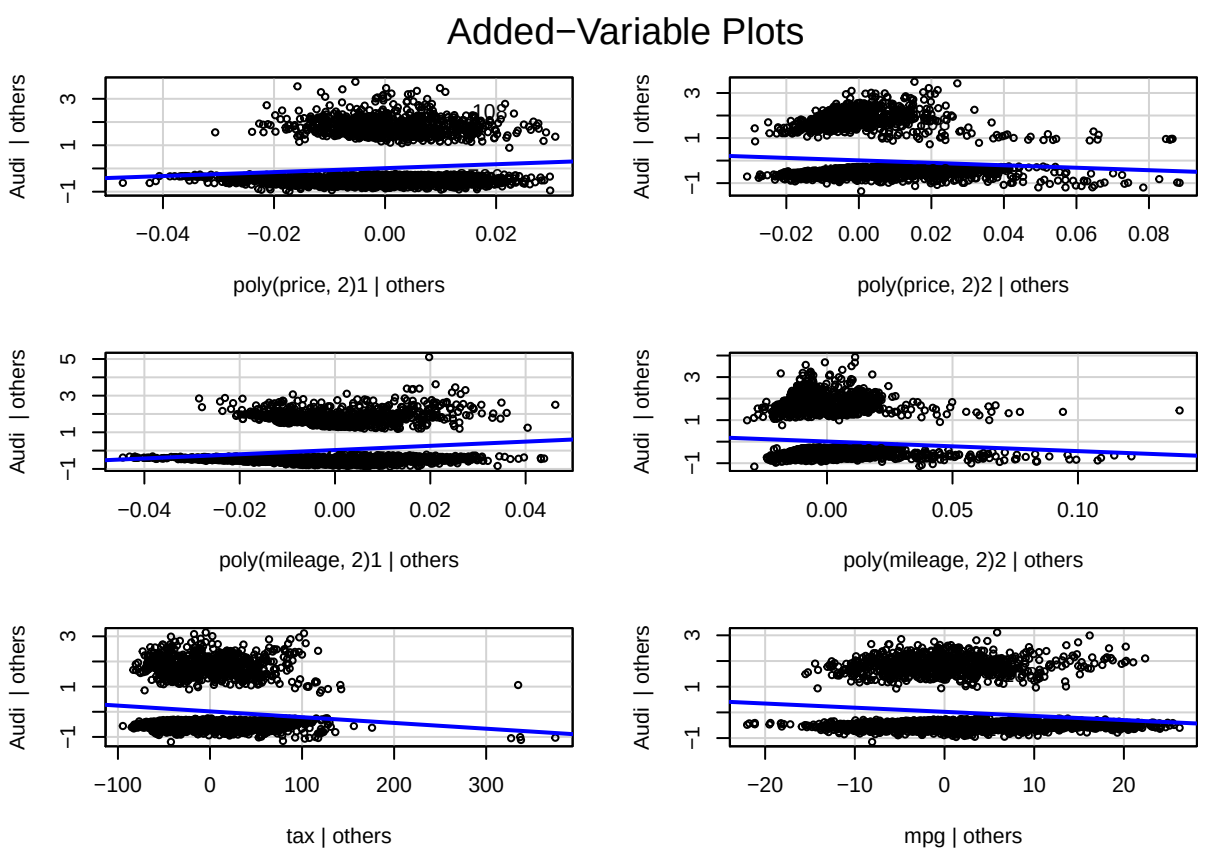
```
##                  Test stat Pr(>|Test stat|)
## poly(price, 2)
## poly(mileage, 2)
## tax                 5.5455          0.01853 *
## mpg                 0.0602          0.80622
## fuelType
## transmission
## engineSize_f
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
```

```
marginalModelPlots(m6)
```

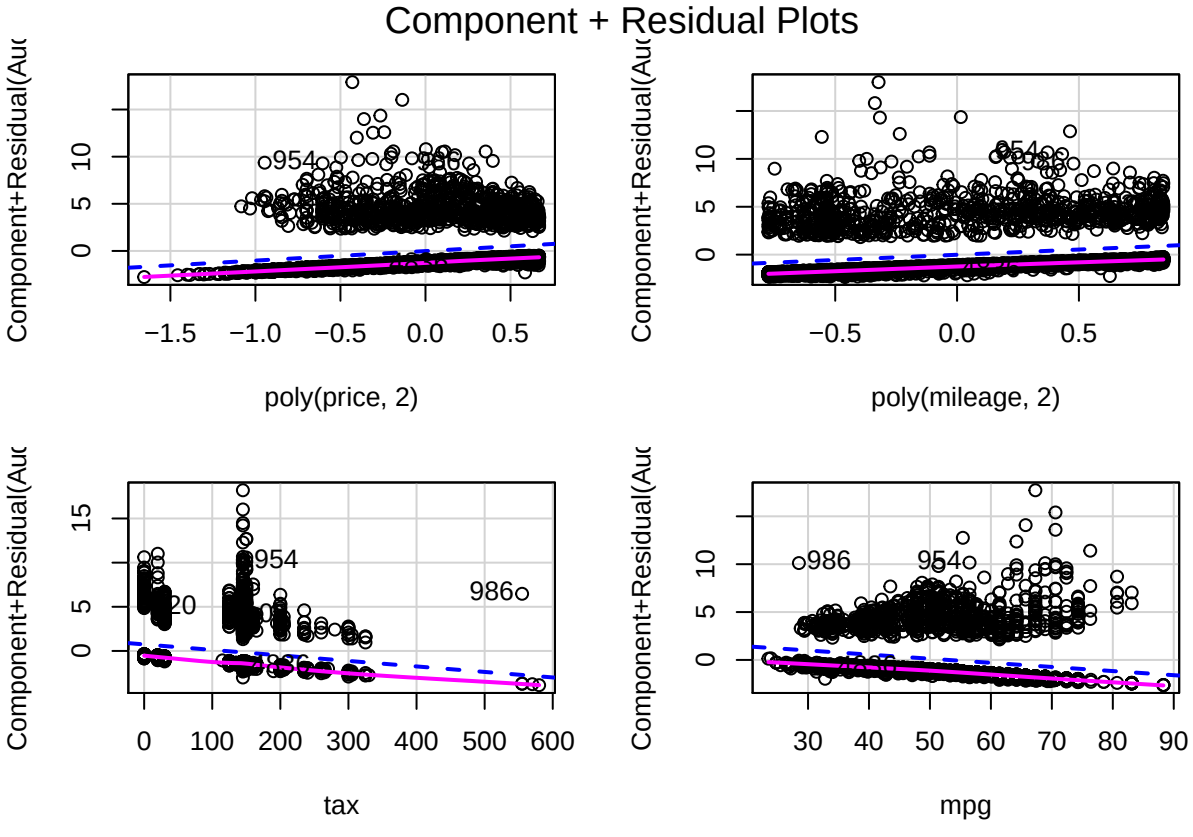


The variables price and mileage are now better adapted to the model.

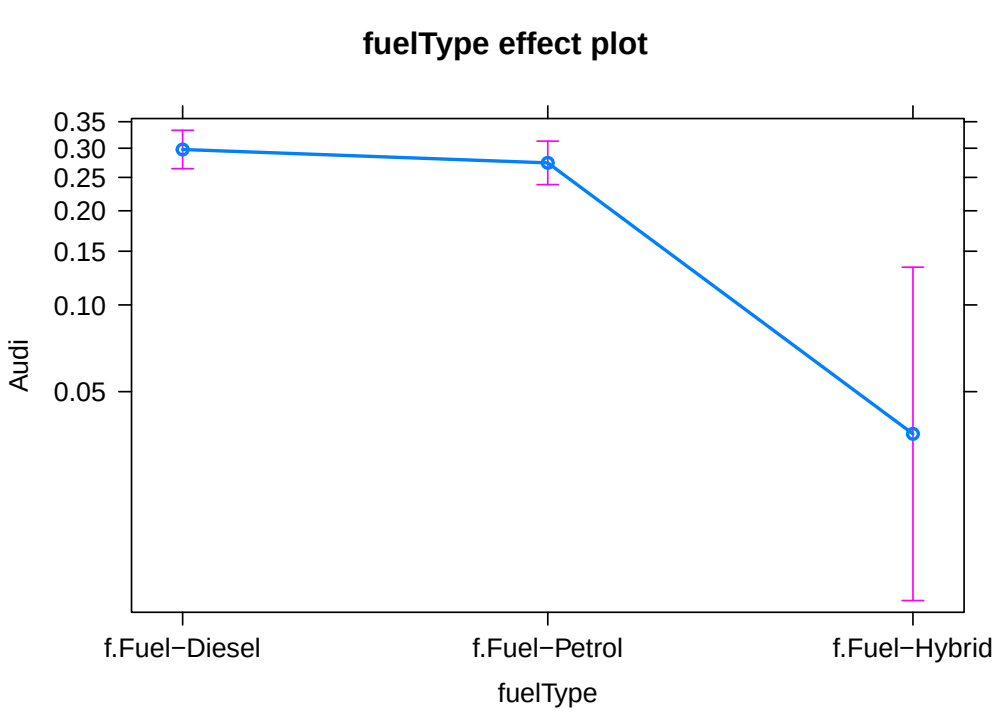
```
avPlots(m5, id = list(method = hatvalues(m6), n = 5))
```




```
crPlots(m5, id = list(method = cooks.distance(m6), n = 5))
```

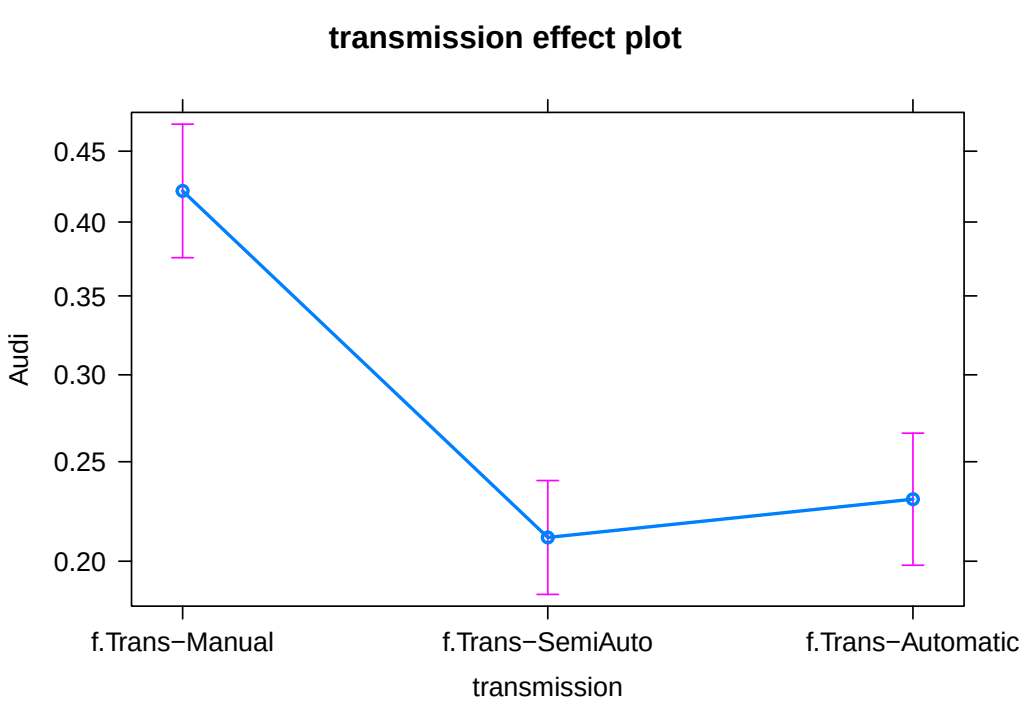



```
plot(allEffects(m6), selection = 5)
```



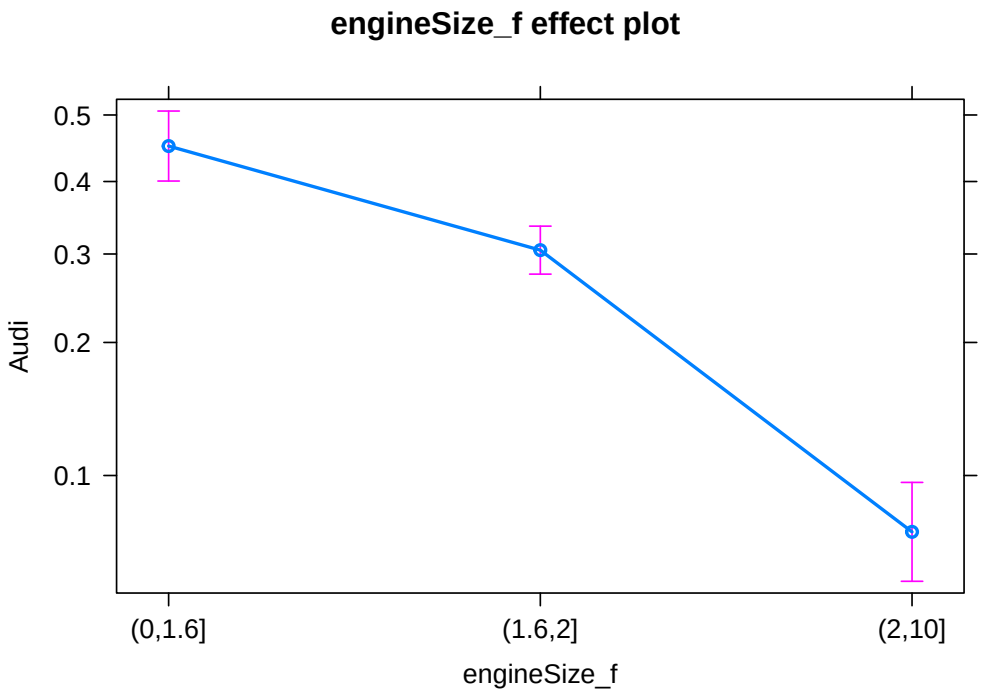
There is more probability of a car being Audi if it uses diesel or petrol than being Hybrid.

```
plot(allEffects(m6), selection = 6)
```



There is more probability of a car being Audi if it has manual transmission.

```
plot(allEffects(m6), selection = 7)
```



There is more probability of a car being Audi if it has a smaller engine.

4.7 Introducing interactions

We are gonna introduce all the possible interactions and then use the step function to improve that model. After doing that we are gonna eliminate the not contributory variables that are yet in the model returned by the step function.

```
m7 <- update(m6, ~. * (fuelType + transmission + engineSize_f)^2, data = df_train[!df_train$mout ==
    "MvOut.Yes", ])
# summary(m7) Summary output Signif. codes: 0 '***' 0.001 '**' 0.01
# '*' 0.05 '.' 0.1 ' ' 1 (Dispersion parameter for binomial family
# taken to be 1) Null deviance: 4076.3 on 3862 degrees of freedom
# Residual deviance: 3194.0 on 3735 degrees of freedom AIC: 3450
# Number of Fisher Scoring iterations: 16
```

The reduction of residual deviance from the null model is 4076.3-3194.0=882.3.

```
anova(m7, m6, test = "LR")
```

```
## Analysis of Deviance Table
##
## Model 1: Audi ~ poly(price, 2) + poly(mileage, 2) + tax + mpg + fuelType +
##     transmission + engineSize_f + fuelType:transmission + fuelType:engineSize_f +
##     transmission:engineSize_f + poly(price, 2):fuelType + poly(price,
##     2):transmission + poly(price, 2):engineSize_f + poly(mileage,
##     2):fuelType + poly(mileage, 2):transmission + poly(mileage,
##     2):engineSize_f + tax:fuelType + tax:transmission + tax:engineSize_f +
##     mpg:fuelType + mpg:transmission + mpg:engineSize_f + poly(price,
##     2):fuelType:transmission + poly(price, 2):fuelType:engineSize_f +
##     poly(price, 2):transmission:engineSize_f + poly(mileage,
##     2):fuelType:transmission + poly(mileage, 2):fuelType:engineSize_f +
##     poly(mileage, 2):transmission:engineSize_f + tax:fuelType:transmission +
##     tax:fuelType:engineSize_f + tax:transmission:engineSize_f +
##     mpg:fuelType:transmission + mpg:fuelType:engineSize_f + mpg:transmission:engineSize_f +
##     fuelType:transmission:engineSize_f
## Model 2: Audi ~ poly(price, 2) + poly(mileage, 2) + tax + mpg + fuelType +
##     transmission + engineSize_f
##   Resid. Df Resid. Dev   Df Deviance  Pr(>Chi)
## 1      3735     3194.0
## 2      3850     3581.2 -115  -387.22 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This reduction is significant.

```
Anova(m7, test = "LR")
```

```
## Analysis of Deviance Table (Type II tests)
##
## Response: Audi
##                                        LR Chisq Df Pr(>Chisq)
## poly(price, 2)                              237  2  < 2.2e-16 ***
## poly(mileage, 2)                            234  2  < 2.2e-16 ***
## tax                                          24  1  9.542e-07 ***
## mpg                                          20  1  7.228e-06 ***
## fuelType                                     13  2   0.001746 **
## transmission                                 73  2  < 2.2e-16 ***
## engineSize_f                                139  2  < 2.2e-16 ***
## fuelType:transmission                        10  3   0.020910 *
## fuelType:engineSize_f                         1  4   0.965987
## transmission:engineSize_f                     7  4   0.163694
## poly(price, 2):fuelType                   35611  4  < 2.2e-16 ***
## poly(price, 2):transmission                   3  4   0.532494
## poly(price, 2):engineSize_f                  34  4  7.100e-07 ***
## poly(mileage, 2):fuelType                    13  4   0.009335 **
## poly(mileage, 2):transmission                 7  4   0.114758
## poly(mileage, 2):engineSize_f                 3  4   0.497080
```

```
## tax:fuelType                                  7  2   0.027700 *
## tax:transmission                              2  2   0.307577
## tax:engineSize_f                              2  2   0.361780
## mpg:fuelType                                  9  2   0.011421 *
## mpg:transmission                             18  2  9.991e-05 ***
## mpg:engineSize_f                              5  2   0.068908 .
## poly(price, 2):fuelType:transmission          2  6   0.894825
## poly(price, 2):fuelType:engineSize_f         19  6   0.003499 **
## poly(price, 2):transmission:engineSize_f     14  8   0.081895 .
## poly(mileage, 2):fuelType:transmission        9  6   0.187189
## poly(mileage, 2):fuelType:engineSize_f       19  6   0.003475 **
## poly(mileage, 2):transmission:engineSize_f    4  8   0.895139
## tax:fuelType:transmission                     0  3   0.951490
## tax:fuelType:engineSize_f                     0  3   1.000000
## tax:transmission:engineSize_f                 7  4   0.131248
## mpg:fuelType:transmission                     3  3   0.432301
## mpg:fuelType:engineSize_f                    14  3   0.003159 **
## mpg:transmission:engineSize_f                 2  4   0.746662
## fuelType:transmission:engineSize_f            4  5   0.575652
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Not all the interactions contribute to the model so we are gonna execute a step.

```
m8 <- step(m7)
summary(m8)
# summary output: (Dispersion parameter for binomial family taken to
# be 1) Null deviance: 4076.3 on 3862 degrees of freedom Residual
# deviance: 3258.7 on 3795 degrees of freedom AIC: 3394.7 Number of
# Fisher Scoring iterations: 17
```

```
anova(m8, m7, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: Audi ~ poly(price, 2) + poly(mileage, 2) + tax + mpg + fuelType +
##     transmission + engineSize_f + fuelType:transmission + fuelType:engineSize_f +
##     transmission:engineSize_f + poly(price, 2):fuelType + poly(price,
##     2):engineSize_f + poly(mileage, 2):fuelType + poly(mileage,
##     2):engineSize_f + tax:fuelType + tax:engineSize_f + mpg:fuelType +
##     mpg:transmission + mpg:engineSize_f + poly(price, 2):fuelType:engineSize_f +
##     poly(mileage, 2):fuelType:engineSize_f + mpg:fuelType:engineSize_f
## Model 2: Audi ~ poly(price, 2) + poly(mileage, 2) + tax + mpg + fuelType +
##     transmission + engineSize_f + fuelType:transmission + fuelType:engineSize_f +
##     transmission:engineSize_f + poly(price, 2):fuelType + poly(price,
##     2):transmission + poly(price, 2):engineSize_f + poly(mileage,
##     2):fuelType + poly(mileage, 2):transmission + poly(mileage,
##     2):engineSize_f + tax:fuelType + tax:transmission + tax:engineSize_f +
##     mpg:fuelType + mpg:transmission + mpg:engineSize_f + poly(price,
##     2):fuelType:transmission + poly(price, 2):fuelType:engineSize_f +
##     poly(price, 2):transmission:engineSize_f + poly(mileage,
##     2):fuelType:transmission + poly(mileage, 2):fuelType:engineSize_f +
##     poly(mileage, 2):transmission:engineSize_f + tax:fuelType:transmission +
##     tax:fuelType:engineSize_f + tax:transmission:engineSize_f +
##     mpg:fuelType:transmission + mpg:fuelType:engineSize_f + mpg:transmission:engineSize_f +
##     fuelType:transmission:engineSize_f
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      3795     3258.7
## 2      3735     3194.0 60    64.74   0.3148
```

```
AIC(m1, m2, m3, m4, m5, m6, m7, m8)
```

```
##     df      AIC
## m1   6 4124.791
## m2  11 4101.448
## m3  11 3913.333
## m4   9 3912.710
## m5   7 3910.038
## m6  13 3607.197
## m7 128 3449.975
## m8  68 3394.716
```

As it is normal when we eliminate explanatory variables the residual deviance has grown but that increase of in not significant because $\Pr(>\text{Chi}) = 0.3148 > 0.05$. Moreover, the AIC has been reduced so we will keep m8.

```
Anova(m8, test = "LR")
```

```
## Analysis of Deviance Table (Type II tests)
##
## Response: Audi
##                                      LR Chisq Df Pr(>Chisq)
## poly(price, 2)                        244.017  2  < 2.2e-16 ***
## poly(mileage, 2)                      235.183  2  < 2.2e-16 ***
## tax                                    29.136  1  6.747e-08 ***
## mpg                                    20.238  1  6.837e-06 ***
## fuelType                               13.757  2   0.001030 **
## transmission                           73.451  2  < 2.2e-16 ***
## engineSize_f                          128.801  2  < 2.2e-16 ***
## fuelType:transmission                   8.634  3   0.034578 *
## fuelType:engineSize_f                   1.394  4   0.845228
```

```
## transmission:engineSize_f                11.959  4   0.017662 *
## poly(price, 2):fuelType                   15.620  4   0.003575 **
## poly(price, 2):engineSize_f               67.220  4  8.763e-14 ***
## poly(mileage, 2):fuelType                 12.569  4   0.013588 *
## poly(mileage, 2):engineSize_f              5.220  4   0.265493
## tax:fuelType                               7.781  2   0.020436 *
## tax:engineSize_f                           6.306  2   0.042722 *
## mpg:fuelType                              11.824  2   0.002706 **
## mpg:transmission                          35.785  2  1.696e-08 ***
## mpg:engineSize_f                           9.827  2   0.007345 **
## poly(price, 2):fuelType:engineSize_f      29.072  6  5.896e-05 ***
## poly(mileage, 2):fuelType:engineSize_f    17.777  6   0.006814 **
## mpg:fuelType:engineSize_f                 32.018  3  5.187e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

But as we still have there are some interactions that doesn't contribute to the model so we are gonna eliminate them.

According to the summary we are going to eliminate the interactions transmission:engineSize_f, poly(price, 2):engineSize_f and mpg:engineSize_f. We can not eliminate, for example the variable engineSize_f despite of not being contributory to the model, because it is included in the interaction fuelType:engineSize_f which is contributory. It happens the same with other variables.

```r
m9 <- glm(formula = Audi ~ poly(price, 2) + poly(mileage, 2) + tax + mpg +
    fuelType + transmission + engineSize_f + fuelType:transmission + fuelType:engineSize_f +
    poly(price, 2):fuelType + poly(mileage, 2):fuelType + poly(mileage,
    2):engineSize_f + tax:fuelType + tax:engineSize_f + mpg:fuelType +
    mpg:transmission + poly(price, 2):fuelType:engineSize_f + poly(mileage,
    2):fuelType:engineSize_f + mpg:fuelType:engineSize_f, family = "binomial",
    data = df_train[!df_train$mout == "MvOut.Yes", ])

# summary(m9) Summary output: Null deviance: 4076.3 on 3862 degrees
# of freedom Residual deviance: 3270.7 on 3799 degrees of freedom
# AIC: 3398.7 Number of Fisher Scoring iterations: 17

Anova(m9, test = "LR")
```

```
## Analysis of Deviance Table (Type II tests)
##
## Response: Audi
##                                          LR Chisq Df Pr(>Chisq)
## poly(price, 2)                            246.402  2  < 2.2e-16 ***
## poly(mileage, 2)                          234.198  2  < 2.2e-16 ***
## tax                                        28.257  1  1.063e-07 ***
## mpg                                        20.706  1  5.356e-06 ***
## fuelType                                   18.375  2  0.0001023 ***
## transmission                               73.451  2  < 2.2e-16 ***
## engineSize_f                              128.801  2  < 2.2e-16 ***
## fuelType:transmission                       8.882  3  0.0309074 *
## fuelType:engineSize_f                       1.612  4  0.8066943
## poly(price, 2):fuelType                    57.698  4  8.830e-12 ***
## poly(mileage, 2):fuelType                  12.746  4  0.0125851 *
## poly(mileage, 2):engineSize_f               5.319  4  0.2561158
## tax:fuelType                                8.404  2  0.0149679 *
## tax:engineSize_f                            7.647  2  0.0218537 *
## mpg:fuelType                               25.765  2  2.542e-06 ***
## mpg:transmission                           32.499  2  8.767e-08 ***
## poly(price, 2):fuelType:engineSize_f       95.232 10  4.884e-16 ***
## poly(mileage, 2):fuelType:engineSize_f     18.040  6  0.0061339 **
## mpg:fuelType:engineSize_f                  41.521  5  7.361e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
anova(m9, m8, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: Audi ~ poly(price, 2) + poly(mileage, 2) + tax + mpg + fuelType +
##     transmission + engineSize_f + fuelType:transmission + fuelType:engineSize_f +
##     poly(price, 2):fuelType + poly(mileage, 2):fuelType + poly(mileage,
##     2):engineSize_f + tax:fuelType + tax:engineSize_f + mpg:fuelType +
##     mpg:transmission + poly(price, 2):fuelType:engineSize_f +
##     poly(mileage, 2):fuelType:engineSize_f + mpg:fuelType:engineSize_f
## Model 2: Audi ~ poly(price, 2) + poly(mileage, 2) + tax + mpg + fuelType +
##     transmission + engineSize_f + fuelType:transmission + fuelType:engineSize_f +
##     transmission:engineSize_f + poly(price, 2):fuelType + poly(price,
##     2):engineSize_f + poly(mileage, 2):fuelType + poly(mileage,
##     2):engineSize_f + tax:fuelType + tax:engineSize_f + mpg:fuelType +
##     mpg:transmission + mpg:engineSize_f + poly(price, 2):fuelType:engineSize_f +
##     poly(mileage, 2):fuelType:engineSize_f + mpg:fuelType:engineSize_f
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      3799     3270.7
## 2      3795     3258.7  4   11.959  0.01766 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
AIC(m1, m2, m3, m4, m5, m6, m7, m8, m9)
```

```
##     df      AIC
## m1   6 4124.791
## m2  11 4101.448
## m3  11 3913.333
```

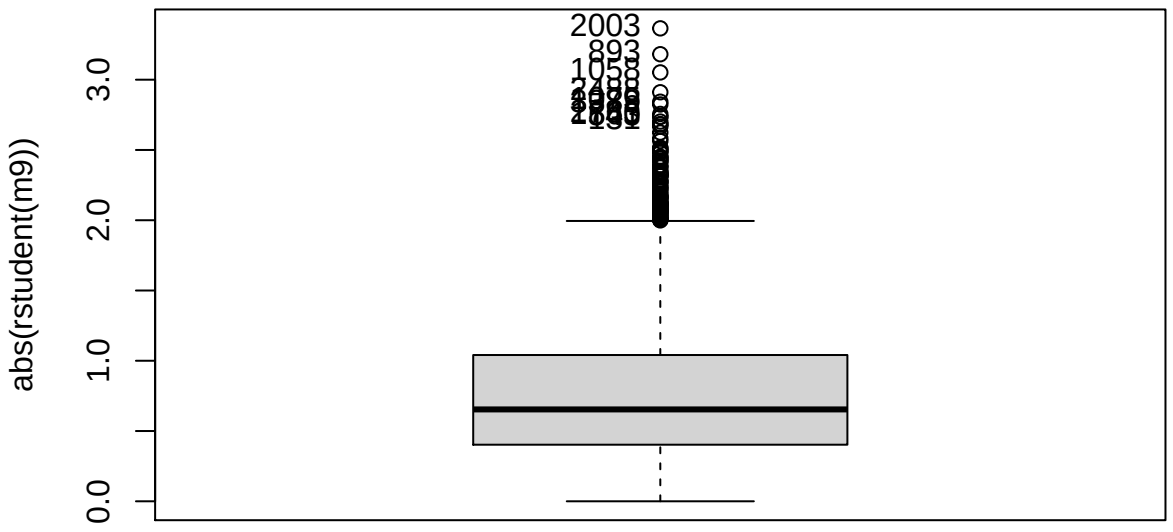
```
## m4   9 3912.710
## m5   7 3910.038
## m6  13 3607.197
## m7 128 3449.975
## m8  68 3394.716
## m9  64 3398.674
```

The residual deviance increases as it is normal but in this case it is significant. The AIC also increases so we could consider m8 a worse model than m9 but we have to consider that when the model has a lot of variables AIC can have a strange behaviour, and we are more interested in reducing the numbre of explanatory variables. We will keep m9.

4.8 Eliminating influential individuals

Finally, we will eliminate the influential individuals and give a diagnostic of the model.

```
dfwork <- df_train[!df_train$mout == "MvOut.Yes", ]

Boxplot(abs(rstudent(m9)))
```



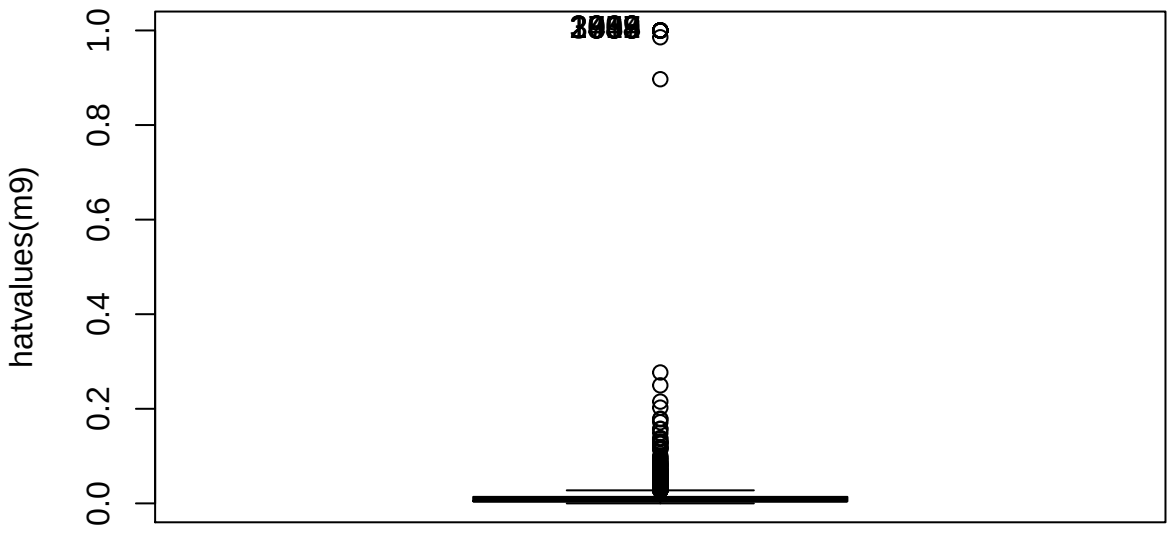

```
##  [1] 2003  893 1058 2488 1075 3389 2735 1860  543  131
```

```
llres <- which(abs(rstudent(m9)) > 2.6)
llres
```

```
##    4  795  191 3957 2516  748  935  300 1687  854  436   62  998  545
##  131  376  543  893 1058 1075 1860 1945 2003 2149 2488 2735 3385 3389
```

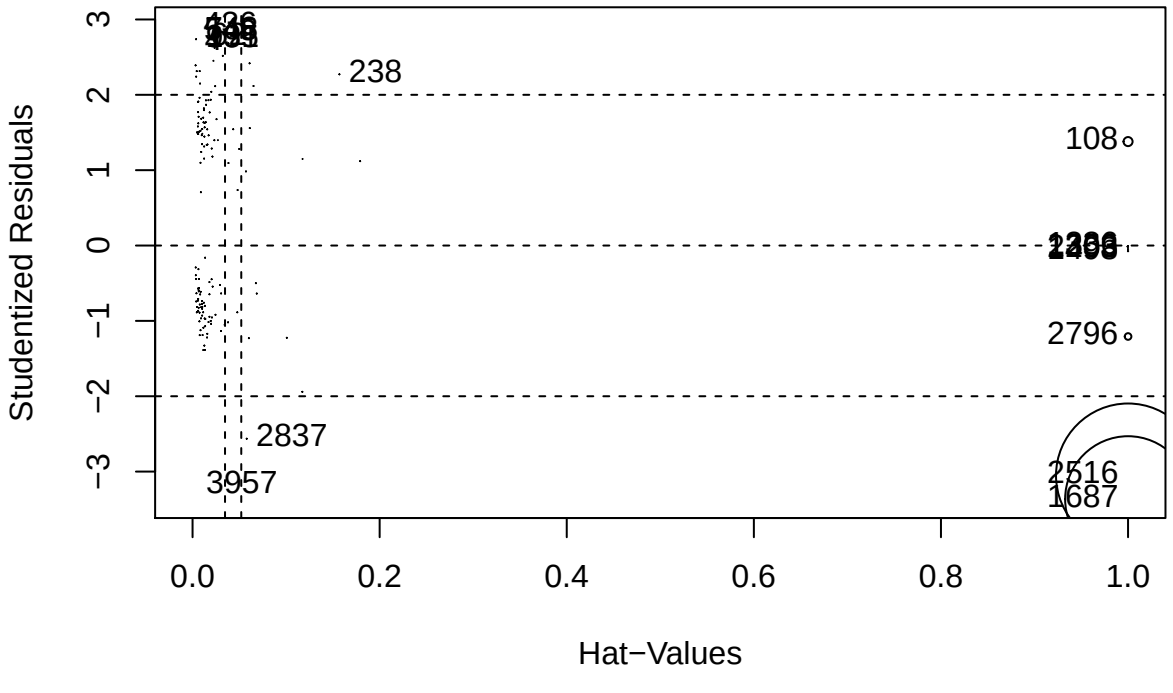
We will consider influential individuals those with an rstudent bigger than 2.6.

```
Boxplot(hatvalues(m9))
```




```
##  [1]  754 1497 3052 3468 1058 2003  869  345 2039 2913
```

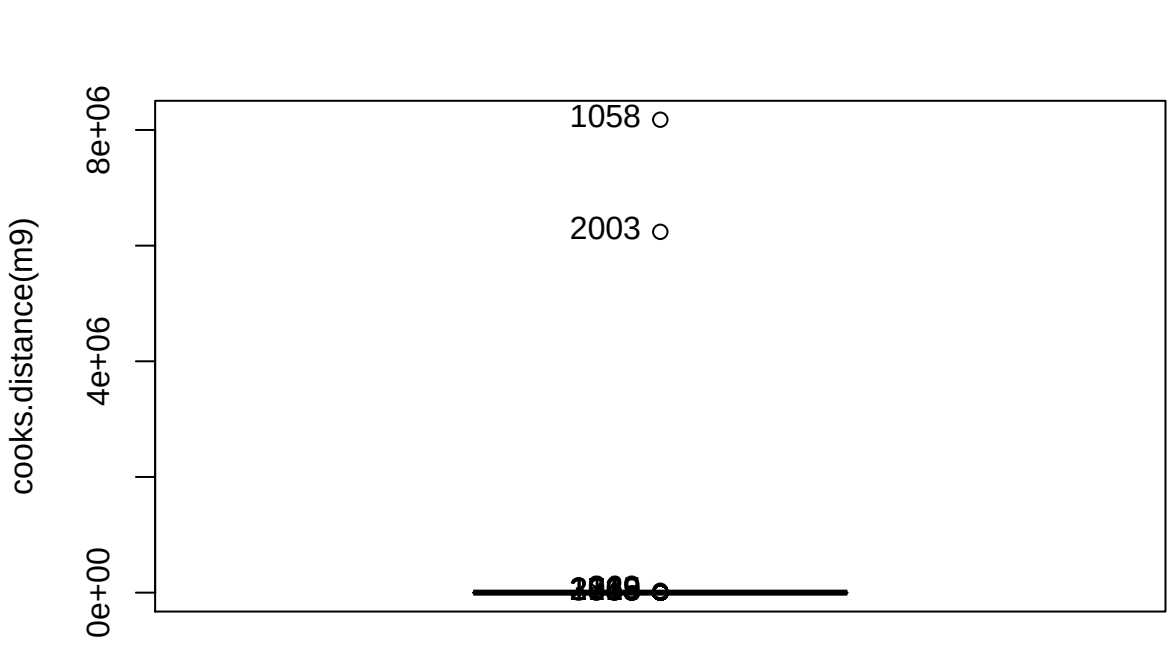
```r
influencePlot(m9, id = list(n = 10))
```




```
##           StudRes         Hat        CookD
## 4     2.702000884 0.002837135 1.586702e-03
## 2796 -1.207174006 0.999998691 1.739892e+04
## 191   2.734478961 0.003909510 2.340764e-03
## 3178          NaN 1.000000000          NaN
## 108   1.379345252 0.999999082 3.238714e+04
## 3957 -3.181348533 0.004422702 8.354164e-03
## 2516 -3.051956880 0.999999982 8.177348e+06
## 748   2.843378221 0.003061770 2.484786e-03
## 2837 -2.563497103 0.057882458 1.586368e-02
## 2511          NaN 1.000000000          NaN
## 1336 -0.005007622 0.999982804 2.278488e-02
## 935   2.737378052 0.003787121 2.289285e-03
## 1687 -3.365346688 0.999999972 6.238877e+06
## 1498 -0.074575465 0.999997315 3.236796e+01
## 436   2.910646669 0.001589574 1.611037e-03
## 62    2.756482143 0.008968526 5.251921e-03
## 2203 -0.055190910 0.999994392 8.486948e+00
## 2179          NaN 1.000000000          NaN
## 238   2.271910293 0.156980959 2.183892e-02
## 545   2.823850166 0.002040795 1.607297e-03
## 3188          NaN 1.000000000          NaN
## 1353 -0.033773963 0.999981682 9.729362e-01
```

A priori influential individuals.

```r
Boxplot(cooks.distance(m9))
```




```
##  [1] 1058 2003  869  345 2039 2913 3735 1768 3169 1318
```

```r
llout <- which(abs(cooks.distance(m9)) > 5000)
llout
```

```
## 2796  108 2516 1687
##  345  869 1058 2003
```

```
llrem <- unique(c(llout, llres))
llrem
```

```
##  [1]  345  869 1058 2003  131  376  543  893 1075 1860 1945 2149 2488 2735 3385
## [16] 3389
```

We will also consider influential individuals those with an cook distance bigger than 5000.

```
m10 <- glm(formula = Audi ~ poly(price, 2) + poly(mileage, 2) + tax + mpg +
    fuelType + transmission + engineSize_f + fuelType:transmission + fuelType:engineSize_f +
    poly(price, 2):fuelType + poly(mileage, 2):fuelType + poly(mileage,
    2):engineSize_f + tax:fuelType + tax:engineSize_f + mpg:fuelType +
    mpg:transmission + poly(price, 2):fuelType:engineSize_f + poly(mileage,
    2):fuelType:engineSize_f + mpg:fuelType:engineSize_f, family = "binomial",
    data = dfwork[-llrem, ])
```

4.9 Diagnostic

```
# summary(m10) summary output: Null deviance: 4037.9 on 3846 degrees
# of freedom Residual deviance: 3173.7 on 3783 degrees of freedom
# AIC: 3301.7 Number of Fisher Scoring iterations: 14
AIC(m1, m2, m3, m4, m5, m6, m7, m8, m9, m10)
```

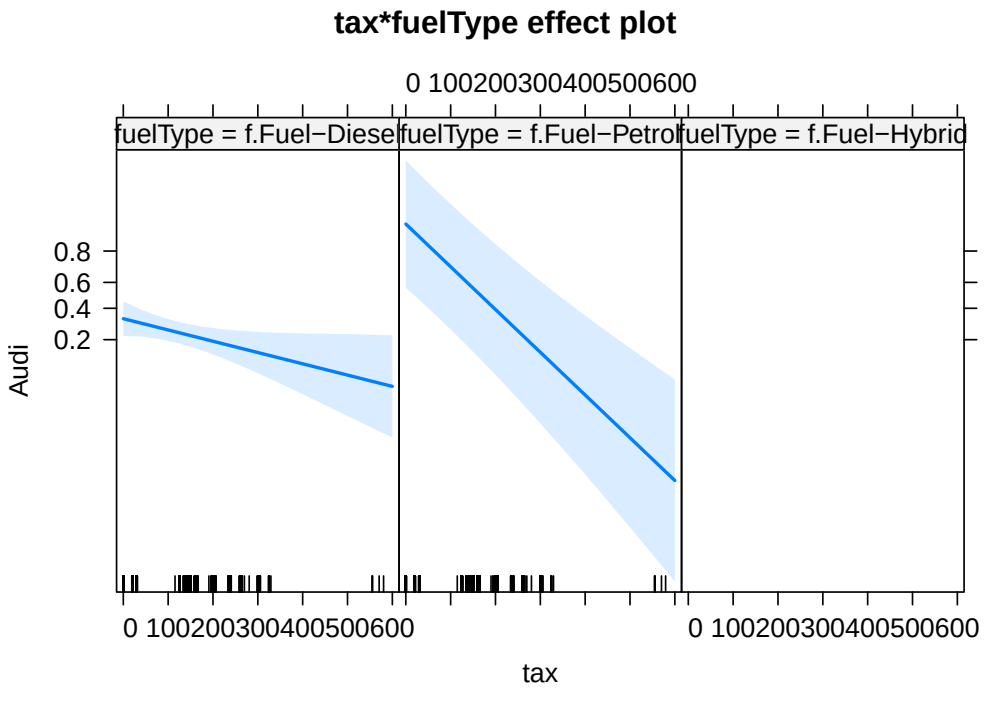
```
##      df      AIC
## m1    6 4124.791
## m2   11 4101.448
## m3   11 3913.333
## m4    9 3912.710
## m5    7 3910.038
## m6   13 3607.197
## m7  128 3449.975
## m8   68 3394.716
## m9   64 3398.674
## m10  64 3301.732
```

The reduction of residual deviance from the null model is 4037.9-3173.7=864.2. The AIC is 3301.7.

```
Anova(m10)
```

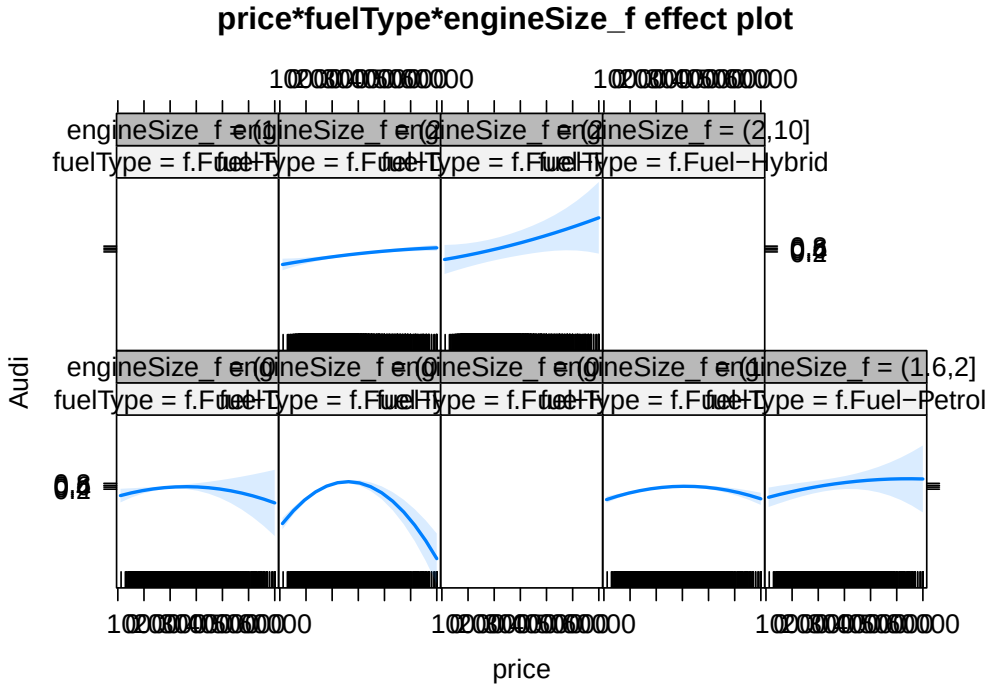
```
## Analysis of Deviance Table (Type II tests)
##
## Response: Audi
##                                       LR Chisq Df Pr(>Chisq)
## poly(price, 2)                         263.406  2  < 2.2e-16 ***
## poly(mileage, 2)                       246.303  2  < 2.2e-16 ***
## tax                                     33.821  1  6.041e-09 ***
## mpg                                     21.445  1  3.642e-06 ***
## fuelType                                19.882  2  4.816e-05 ***
## transmission                            76.053  2  < 2.2e-16 ***
## engineSize_f                           136.755  2  < 2.2e-16 ***
## fuelType:transmission                    8.975  3  0.0296222 *
## fuelType:engineSize_f                    2.050  4  0.7266093
## poly(price, 2):fuelType                 56.584  4  1.513e-11 ***
## poly(mileage, 2):fuelType               13.289  4  0.0099488 **
## poly(mileage, 2):engineSize_f            4.695  4  0.3200228
## tax:fuelType                             9.645  2  0.0080463 **
## tax:engineSize_f                         8.133  2  0.0171398 *
## mpg:fuelType                            30.913  2  1.938e-07 ***
## mpg:transmission                        37.743  2  6.371e-09 ***
## poly(price, 2):fuelType:engineSize_f   119.442 10  < 2.2e-16 ***
## poly(mileage, 2):fuelType:engineSize_f  22.485  6  0.0009888 ***
## mpg:fuelType:engineSize_f               45.897  5  9.531e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
plot(allEffects(m10), selection = 2)
```



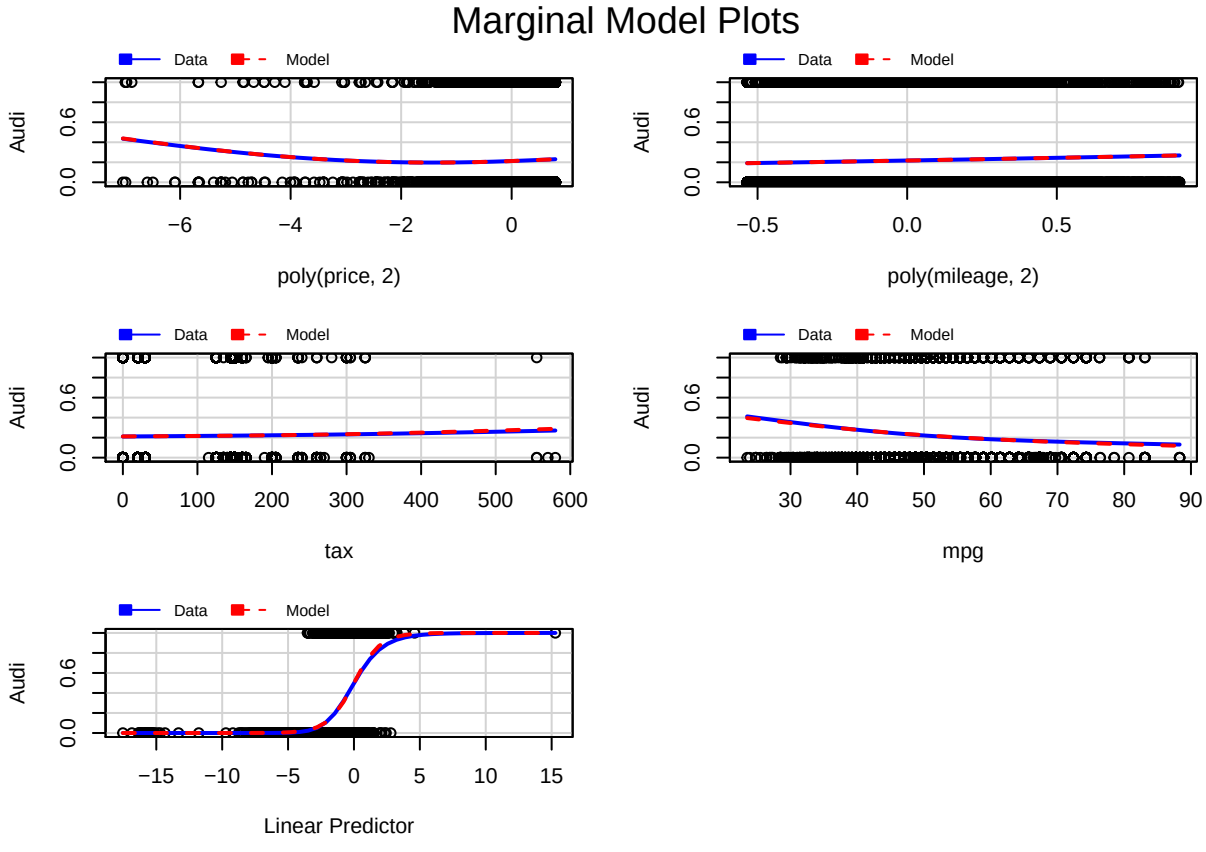
The probability of being audi es reduced when the cars use diesel or petrol and the tax increases.

```
plot(allEffects(m10), selection = 5)
```



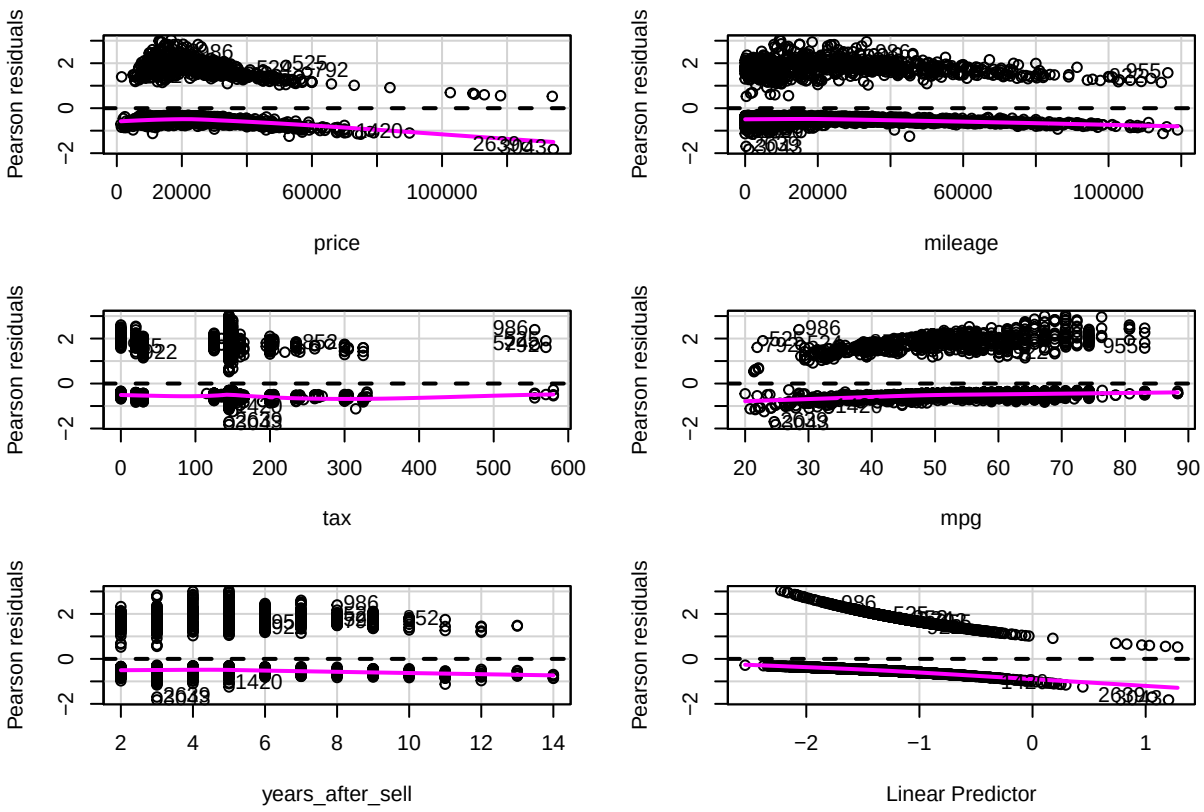
The probability of being audi es increases when the cars has a small engine, use petrol or diesel and the price increases.

```
marginalModelPlots(m10)
```



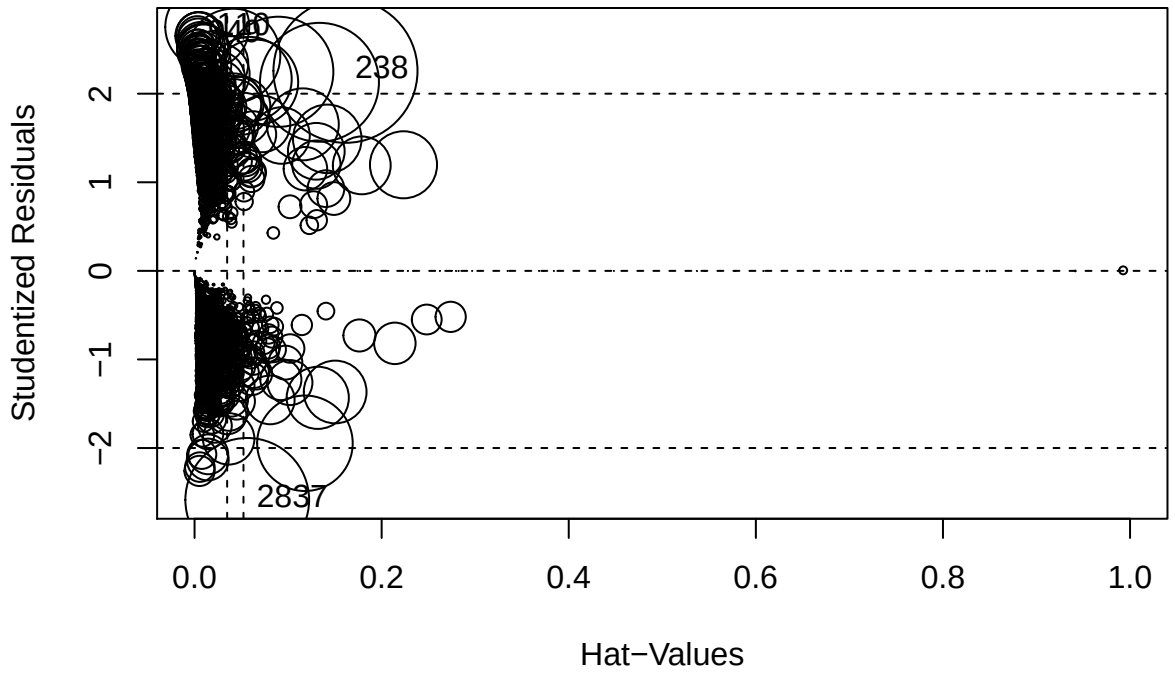
The model is very well adapted to the data.

```
residualPlots(m1, id = list(method = cooks.distance(m1), n = 10))
```




```
##                  Test stat Pr(>|Test stat|)
## price              14.7961        0.0001198 ***
## mileage            16.3351        5.307e-05 ***
## tax                 1.6081        0.2047553
## mpg                 1.2637        0.2609517
## years_after_sell    9.5592        0.0019895 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
influencePlot(m10)
```




```
##        StudRes         Hat       CookD
## 3178       NaN 1.000000000         NaN
## 110   2.753780 0.013600458 0.007437622
## 2837 -2.584879 0.056344873 0.016154281
## 2511       NaN 1.000000000         NaN
## 949   2.672090 0.004028939 0.002048147
## 238   2.257975 0.161334008 0.021898727
```

```
outlierTest(m10)
```

```
## No Studentized residuals with Bonferroni p < 0.05
## Largest |rstudent|:
##     rstudent unadjusted p-value Bonferroni p
## 110  2.75378          0.0058911           NA
```

The influence plot show us that has been a reduction of the influential individuals.

4.10 Goodness of fit and Predictive Capacity

```
# H0: Model fits data
pchisq(m10$null.deviance - m10$deviance, m10$df.null - m10$df.residual,
    lower.tail = FALSE)
```

```
## [1] 3.919645e-141
```

The model fits well the data because 3.919645e-141 < 0.05 and we can accept H0.

```
X2m10 <- sum((resid(m10, "pearson")^2))
X2m10
```

```
## [1] 3398.006
```

```
1 - pchisq(X2m10, m10$df.res)
```

```
## [1] 0.9999977
```

```
library(DescTools)
```

```
##
## Attaching package: 'DescTools'

## The following object is masked from 'package:car':
##
##     Recode
```

```
PseudoR2(m10, which = "all")
```

```
##        McFadden     McFaddenAdj        CoxSnell      Nagelkerke   AldrichNelson
##       0.2140181       0.1823186       0.2011958       0.3095643       0.1834331
## VeallZimmermann           Efron McKelveyZavoina             Tjur             AIC
##       0.3581932       0.2131215       0.5997898       0.2163923    3301.7323600
##             BIC          logLik          logLik0              G2
##    3702.0554898   -1586.8661800   -2018.9601035     864.1878470
```

This model obtains a PseudoR2 of McFadden of 0.214. According to this factor a value between 0.2 and 0.4 represents a model that fits excellently the data.

```
library(ResourceSelection)
```

```
## ResourceSelection 0.3-5   2019-07-22
```

```
ht <- hoslem.test(m10$y, fitted(m10), g = 10)
ht
```

```
##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
## data:  m10$y, fitted(m10)
## X-squared = 5.4865, df = 8, p-value = 0.7045
```

This gives p-value=0.7045 > 0.05, indicating no evidence of poor fit.

```
pred_test <- predict(m10, newdata = df_test, type = "response")
ht <- hoslem.test(df_test$Audi, pred_test)
ht
```

```
##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
## data:  df_test$Audi, pred_test
## X-squared = 1000, df = 8, p-value < 2.2e-16
```

```
cbind(ht$observed, ht$expected)
```

```
##                    y0 y1    yhat0      yhat1
## [1.21e-13,0.0119]  0  0 99.47956  0.5204383
## (0.0119,0.0435]    0  0 97.51412  2.4858835
## (0.0435,0.0815]    0  0 93.64907  6.3509288
## (0.0815,0.127]     0  0 89.78087 10.2191312
## (0.127,0.181]      0  0 84.42001 15.5799901
## (0.181,0.24]       0  0 79.20903 20.7909735
## (0.24,0.296]       0  0 73.18741 26.8125850
## (0.296,0.371]      0  0 67.11081 32.8891872
## (0.371,0.508]      0  0 56.91703 43.0829675
## (0.508,1]          0  0 33.85308 66.1469226
```

As p-value = 2.2e-16 < 0.04 there is no evidence that m10 doesn't predict well the variable value of the variable Audi of the cars on df_test.

```
# ROC Curve
library("pROC")
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
par(pty = "s")
roc(df_test$Audi, pred_test, plot = TRUE, legacy.axes = TRUE, percent = TRUE,
    col = "#377eb8", xlab = "False Positive Percentage", ylab = "True Positive Percentage",
    print.auc = TRUE)
```

```
## Setting levels: control = No, case = Yes

## Setting direction: controls < cases

##
## Call:
## roc.default(response = df_test$Audi, predictor = pred_test, percent = TRUE,     plot = TRUE, legacy.axes = TRUE, col = "#377eb8", xlab = "False Positive
##
## Data: pred_test in 798 controls (df_test$Audi No) < 202 cases (df_test$Audi Yes).
## Area under the curve: 77.61%
```

```r
pred_test_m9 <- predict(m9, newdata = df_test, type = "response")

plot.roc(df_test$Audi, pred_test_m9, percent = TRUE, col = "#4daf4a", print.auc = TRUE,
    add = TRUE, print.auc.y = 45)
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```r
pred_test_m8 <- predict(m8, newdata = df_test, type = "response")

plot.roc(df_test$Audi, pred_test_m8, percent = TRUE, col = "#ff0000", print.auc = TRUE,
    add = TRUE, print.auc.y = 40)
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```r
pred_test_m7 <- predict(m7, newdata = df_test, type = "response")

plot.roc(df_test$Audi, pred_test_m7, percent = TRUE, col = "#00EFFF", print.auc = TRUE,
    add = TRUE, print.auc.y = 35)
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```r
pred_test_m6 <- predict(m6, newdata = df_test, type = "response")

plot.roc(df_test$Audi, pred_test_m6, percent = TRUE, col = "#F700FF", print.auc = TRUE,
    add = TRUE, print.auc.y = 30)
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```r
pred_test_m5 <- predict(m5, newdata = df_test, type = "response")

plot.roc(df_test$Audi, pred_test_m5, percent = TRUE, col = "#FFCD00", print.auc = TRUE,
    add = TRUE, print.auc.y = 25)
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```r
pred_test_m4 <- predict(m4, newdata = df_test, type = "response")

plot.roc(df_test$Audi, pred_test_m4, percent = TRUE, col = "#00FF89", print.auc = TRUE,
    add = TRUE, print.auc.y = 20)
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```r
pred_test_m3 <- predict(m3, newdata = df_test, type = "response")

plot.roc(df_test$Audi, pred_test_m3, percent = TRUE, col = "#00BCFF", print.auc = TRUE,
    add = TRUE, print.auc.y = 15)
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

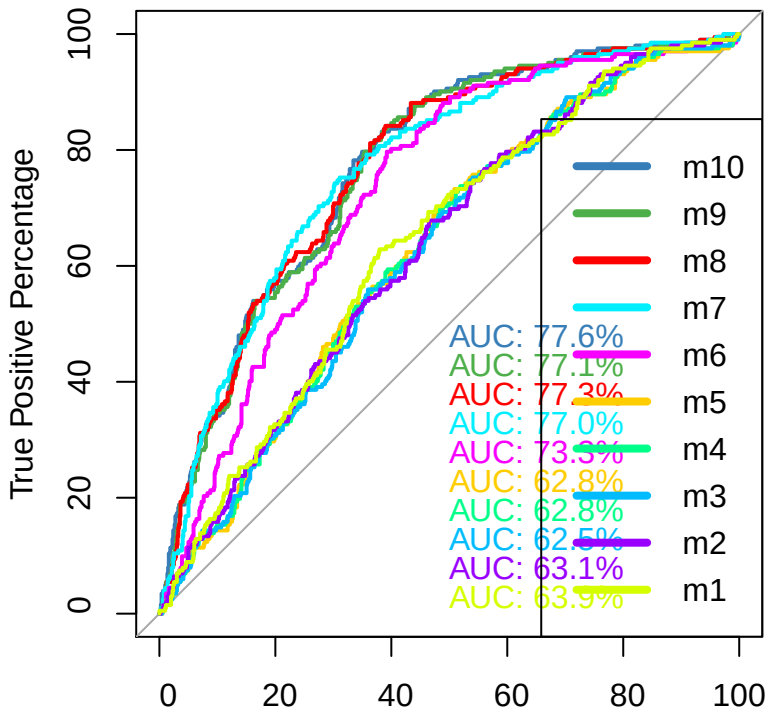
```r
pred_test_m2 <- predict(m2, newdata = df_test, type = "response")

plot.roc(df_test$Audi, pred_test_m2, percent = TRUE, col = "#9A00FF", print.auc = TRUE,
    add = TRUE, print.auc.y = 10)
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```r
pred_test_m1 <- predict(m1, newdata = df_test, type = "response")

plot.roc(df_test$Audi, pred_test_m1, percent = TRUE, col = "#D5FF00", print.auc = TRUE,
    add = TRUE, print.auc.y = 5)
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```
legend("bottomright", legend = c("m10", "m9", "m8", "m7", "m6", "m5", "m4",
    "m3", "m2", "m1"), col = c("#377eb8", "#4daf4a", "#ff0000", "#00EFFF",
    "#F700FF", "#FFCD00", "#00FF89", "#00BCFF", "#9A00FF", "#D5FF00"),
    lwd = 4)
```



With this graph we can be seen the ROC curve of all the models we have created.

The ROC curve create a graph were the value of the x represent the false positive rate (in our case expressed as a percentage), and the value of the y the true positive rate. These values change accordingly to the variation of the threshold. The best possible ROC curve would have a point where True Positive Rate = 1 and False Positive Rate = 0.

From these type of graph we can calculate the AUC that is based on the area under the ROC curve. When the AUC is bigger the model predicts better.

The best model based on the AUC is the last one, m10, with an AUC of 77.6%.

```
# Confusion Table Analysis
treshold <- 0.5
audi.est <- ifelse(pred_test < treshold, 0, 1)
tt <- table(audi.est, df_test$Audi)
tt
```

```
##
## audi.est  No Yes
##        0 748 149
##        1  50  53
```

We can see that our model predicts well when a car is not an Audi but not when it is with a threshold of 0.5. The thershold can be modified depending on our interests, if we are more interested in getting a better True Positive Rate or a lower False Positive Rate.

```
100 * sum(diag(tt))/sum(tt)
```

```
## [1] 80.1
```

Our model has an accuracy of 80.1%.