Katholieke
Universiteit
Leuven

Statistische Modellen & Data-analyse

Practicum 2

Tomas Fiers

Juni 2017

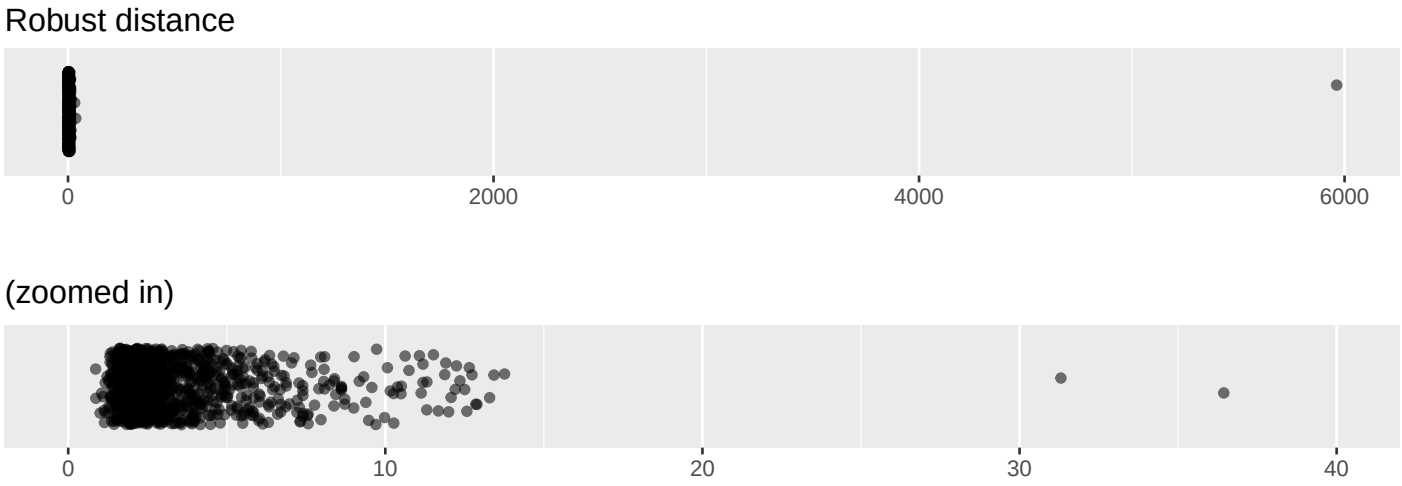


Figure 1: Robust distance-to-center of each car in the training set.

1 Linear regression

The complete dataset of 1500 cars was randomly split into a 1000-car training set and a 500-car test set. We continue this analysis with the training set.

Exploratory analysis & outlier removal

We search for outliers by calculating, for each data point, its statistical distance to the multivariate distribution – using robust estimates for the mean and the covariance matrix. Specifically, the Minimum Covariance Determinant estimator [1] was used, approximated by the `Fast MCD` algorithm [2]. Figure 1 shows the resulting robust distances.

Three strong outliers are found:

- the "Volkswagen *Jetta (from NOV 06 Wk 45 >)* 1.4 TSI (170 PS) Sport" with an abnormal `noise_level` of 0.3;
- the "Vauxhall *Signum MY2008* 3.0CDTi V6 24v with 16/17/18" wheel" with an abnormal `nox_emissions` value of 237000;
- the "MG Rover Group *Streetwise* 1.8" without any individually abnormal variables.

Note that this third outlier would not have been found had we only looked at univariate (or even bivariate) distributions of the data.

The found strong outliers are removed from the training set.

Figure 2: Uni- and bivariate distributions in the training set. The three strongest robust distance-outliers have been removed. Colours according to the future classification task.

Figure 2 shows the uni- and bivariate distributions in the data set. Note the very strong correlations between `urban_metric`, `extra_urban_metric`, `combined_metric`, and `co2` – and to a lesser extend `engine_capacity`. This is in accordance with their meanings: cars with larger engine volumes consume more liters of fuel (all three `_metrics` measure fuel consumption), and every liter of fuel corresponds to a fixed amount of CO_2. These strong pairwise correlations point to the problem of multicollinearity in the data set. We explore this more formally further on.

We also note that for the variable `noise_level`, a large subset of cars take on discrete values. (There are also cars with `noise_level`s in between). This discrete character would negatively impact a cluster analysis, as the clusters would tend to form around the disrete values; while other variables would have disproportionately less impact on the clustering. However, we decide this will probably not impact the subsequent regression and classification tasks too much, so we keep this variable, for now.

Additionally, we note that none of the continuous variables seem to be univariate normally distributed (further on we perform numeric normality tests. Also normal QQ plots have been checked; they did not indicate any normality at all). Rather, their distributions are positively skewed, with heavy right tails.

Finally, note the multimodality of `engine_capacity` and `nox_emissions`, and the fact that `fuel_type` tends to split the other variables into groups – see e.g. `Diesel` versus `Hybrid` or `Petrol` for `nox_emissions`.

Model construction

We will construct a general linear model to predict the `co2` variable. The predictor variables to include in the model will be selected via bidirectional stepwise regression. The Akaike Information Criterion (AIC) is used to compare models. The least squares estimator will always be used to estimate the coefficients $\boldsymbol{\beta}$ for a candidate model. The categorical variables `euro_standard`, `transmission_type`, and `fuel_type` are expanded into indicator variables (e.g. `fuel_type_Petrol`, taking on either a 0 or a 1).

Predictor variable transformations

The AIC assumes the residuals to be normally distributed. As they are an affine transformation of the predictor variables, the predictor variables need to be normally distributed as well. Thus, before starting the stepwise variable selection procedure, we find a Box-Cox power transformation [3] for all continuous predictor variables that maximises their normality in the maximum likelihood sense. The thusly found exponents λ are listed in table 1, along with the p-values for the Shapiro-Wilk test of normality before and after the transformation. Note that, except for `noise_level`, the normality of each variable greatly increases after the transformation. This was confirmed using normal QQ plots of the variables. We

Variable	λ_{ML}	p_{before}	p_{after}
`engine_capacity`	-0.65	4E$-$33	2E$-$12
`urban_metric`	-0.06	2E$-$23	0.164
`extra_urban_metric`	-0.64	3E$-$22	0.002
`combined_metric`	-0.37	3E$-$23	0.011
`noise_level`	7.11	2E$-$14	3E$-$12
`co_emissions`	0.29	7E$-$26	0.093
`nox_emissions`	0.04	1E$-$35	1E$-$12

Table 1: Maximum likelihood transformation exponents λ, and p-values for the Shapiro-Wilk test of normality before and after transformation, for each possible continuous predictor variable.

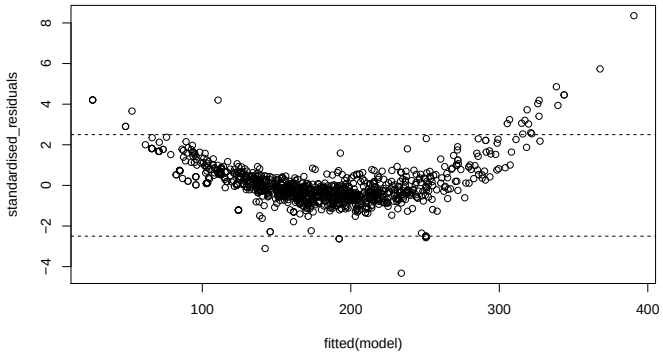


Figure 3: Standardised residuals of the linear model versus fitted values, before the response variable has been transformed.

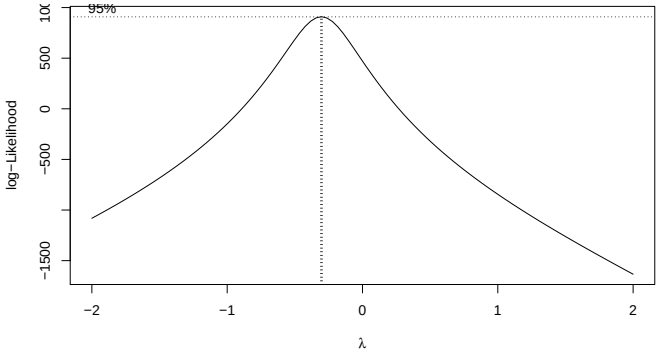


Figure 4: Log-likelihood of λ for the Box-Cox transformation of the response variable.

apply the maximum likelihood transformation to all continuous predictor variables, except for `noise_level` (as its maximum likelihood exponent falls outside the usual range, and its normality does not significantly improve after transformation).

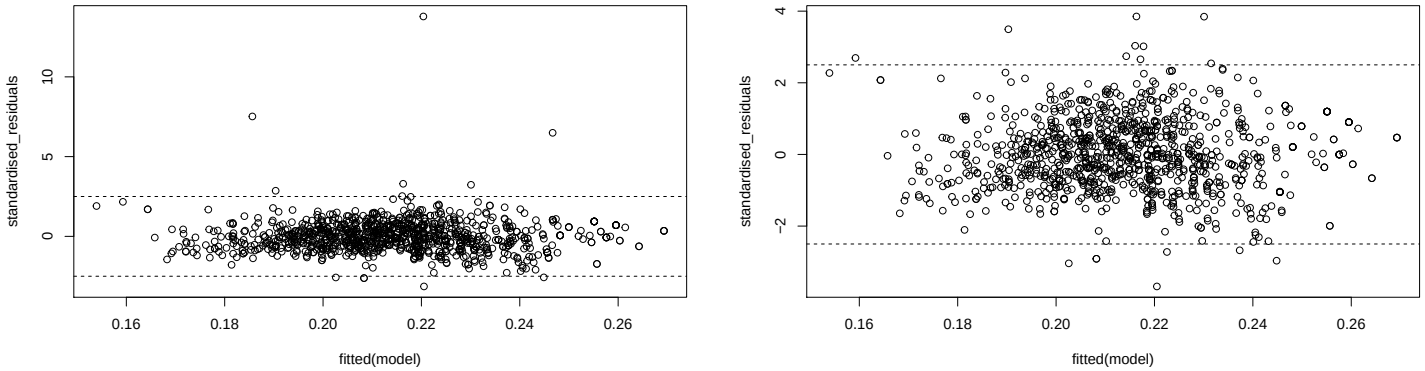
Response variable transformation

We now create a first linear model (via stepwise AIC regression starting from a model containing all predictor variables). Figure 3 shows the residuals of the resulting model. The strong (nonlinear) correlation between the predicted `co2` values and the residuals prompts us to transform the response variable. The optimal Box-Cox power transformation of the response variable [3] is found for $\lambda = -0.3$ (see fig. 4). We create a new linear model with the transformed response variable.

Removal of residual outliers

The residuals of the model with transformed response variable are shown in fig. 5a. We find three strong vertical outliers. They are:

Figure 5: Standardised residuals of the linear model versus fitted values, after the response variable has been transformed.



(a) Before removal of vertical outliers.

(b) After removal of vertical outliers.

- the "Volkswagen *LT 35 Kombi MWB - LWB* 2.8 (158 PS) TDI Axle Ratio 3.727";
- the "Honda *Insight, Model Year 2012* 1.3 IMA HS, HS-T, HX";
- and the "Renault *New Mégane Hatchback* 2.0 dCi 160 FAP".

(These cars do not have higher robust distances than other cars (as in fig. 1); they can however be spotted as outliers when the Mahalonobis distance is considered). These three additional outliers are removed from the training set. We note that removing these outliers has no effect on the maximum-likelihood λ of the Box-Cox transformation of the response variable.

Variable selection & final model

With optimal input and output transformations and the strongest outliers removed, we can now construct our final model, by selecting an appropriate subset of predictor variables. As already mentioned, bidirectional stepwise AIC regression is used. The procedure is started both from a model containing all predictor variables, and from an empty model containing only a constant term. Both procedures converge to the same model. The included variables, along with estimates for their coefficient and an analysis of variance (ANOVA), are shown in table 2. The variables that are not included, are `noise_level` and `transmission_type_Manual`. We find a final R^2 value of 0.9987, an adjusted R^2 value of 0.9987, and a mean squared error $\hat{\sigma}^2$ of 4.750×10^{-7}. The standardised residuals are shown in fig. 5b.

Predictor variable j	λ_j	$\hat{\beta}_j$	s($\hat{\beta}_j$)	t	p	SS	F	$Pr(>F)$
Intercept	–	5.3E−1	2.8E−2	18.9	3E−68	45.6	95 972 309	< 2E−16
`engine_capacity`	−0.65	−9.3E−2	1.8E−2	−5.1	5E−7	0.1663	350 125	< 2E−16
`urban_metric`	−0.06	−2.2E−2	1.2E−3	−18.4	3E−65	0.1771	372 852	< 2E−16
`extra_urban_metric`	−0.64	−5.5E−2	3.7E−3	−14.7	2E−44	0.0032	6710	< 2E−16
`combined_metric`	−0.37	−5.4E−2	4.5E−3	−12.0	7E−31	0.0001	229	< 2E−16
`co_emissions`	0.29	−2.6E−5	7.6E−6	−3.5	6E−4	0.0010	2149	< 2E−16
`nox_emissions`	0.04	7.2E−5	3.7E−5	2.0	5E−2	0.0058	12 130	< 2E−16
`euro_standard_4`	–	3.7E−4	7.0E−5	5.2	2E−7	0.0002	506	< 2E−16
`euro_standard_5`	–	1.4E−3	7.9E−5	18.0	5E−63	0.0002	391	< 2E−16
`euro_standard_6`	–	1.0E−3	1.6E−4	6.0	2E−9	0.0002	388	< 2E−16
`fuel_type_Hybrid`	–	7.5E−3	1.7E−4	44.0	2E−234	0.0002	390	< 2E−16
`fuel_type_Petrol`	–	7.5E−3	1.4E−4	54.7	2E−300	0.0014	2993	< 2E−16
Residuals	–	–	–	–	–	0.0005	–	–

Table 2: Coefficients and ANOVA of the final general linear model. λ_j's taken from table 1. Note that both the predictor variables and the response variable are transformed non-linearly. That is why some coefficients may seem counterintuitive at first – e.g. a negative slope for `engine_capacity` with respect to `co2`.

Discussion

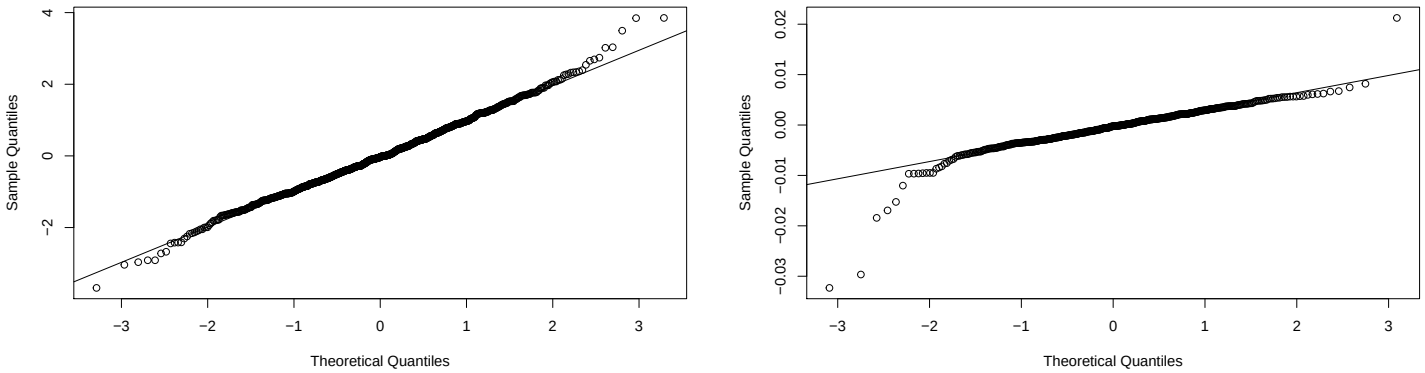
Performance on the test set

As we have a proper test set at our disposal, we can check whether our model generalises well. It does. The mean absolute relative error (defined as $\frac{1}{n-p}\sum_{i=1}^{n}\left|\frac{\hat{y}_i-y_i}{y_i}\right|$) on the test set is 2.9×10^{-3}, demonstrating excellent predictive power – the CO_2 emissions of a car can be predicted with an error of on average only 0.3%. The mean squared error on the test set is 8.00×10^{-7}, indicating that our previous estimate $\hat{\sigma}^2 = 4.75 \times 10^{-7}$ was a decent one.

Gauss-Markov conditions

We check whether our final model satisfies the Gauss-Markov conditions. The first condition, that the errors have expectation zero, is satisfied by construction. The other two conditions – that the errors have equal variance and are uncorrelated – cannot be definitely proven, but we can check whether some necessary conditions seem to be satisfied. All residual plots – residuals versus their index, residuals versus the fitted values, and residuals versus each of the independent variables – need to be free of correlation. No curvature, funnel or trend should be visible. We have checked all possible residual plots – both for the training set as well as for the test set. (One such plot can be seen in fig. 5b). No visible correlations were found, supporting the hypothesis that the Gauss-Markov conditions are reasonably well satisfied.

Figure 6: Normal QQ plots of linear model residuals



(a) Training set standardised residuals

(b) Test set residuals

Normality

Most steps in the construction and analysis of our model assume the errors to be normally distributed (e.g. the Box-Cox transformation of the response variable, the AIC, and the statistics in table 2). Based on the normal QQ plots of fig. 6, we pose that the errors are roughly normally distributed, but with tails that are too heavy. (The p values for the Shapiro-Wilk test of normality are 0.01 and $< 2\text{E}{-16}$ for the training and test set, respectively). This less-than-ideal normality does not seem to be a problem during model construction. It does have the result however that the statistics and p-values in table 2 need to be taken with a grain of salt.

Multicollinearity

In the exploratory analysis, we noted the strong pairwise correlations between `urban_metric`, `extra_urban_metric`, `combined_metric`, and `co2`. This points to the problem of multicollinearity. Indeed, when we perform more formal tests, strong evidence is found for multicollinearity in the data (cutoff values from [4]): the variance inflation factors (VIF) of these mentioned variables are all larger than 100, and the mean VIF is 456, significantly larger than 1. The condition number $\sqrt{\lambda_{max}/\lambda_{min}} = 129$ is greater than 30 (where λ are the eigenvalues of the corrrelation matrix of the data). In the preceding linear regression, we have ignored this multicollinearity. In the following exercise however, we will use a biased regression method with reduced variance to mitigate its effects.

2 Classification

For the classification task, the variable `euro_standard` is transformed into a binary factor. Former Euro Standards 3 and 4 are considered "old" (0), while former Euro Standards 5 and 6 are considered "new" (1). We will predict this binary variable by fitting a logistic model, and performing linear and quadratic determinant analyses. But before any model is fit, we turn our attention to the multicollinearity problem.

Decorrelating regressors via PCA

Instead of the regular unbiased estimator for the linear model parameters, we will choose an estimator that trades off some newly present bias for a large reduction in model variance. We choose for principal component analysis (PCA); which in this context is also sometimes called principal component regression. Our task is then to choose a suitable number k of principal components (PC) to retain.

For each possible number k of largest eigenvalues (and thus number of PC) to retain, we fitted a full logistic model predicting `euro_model` using the transformed input variables and the remaining regressing factors. This model was applied to the (identically transformed) test data. The resulting prediction error rates are shown in fig. 8.

Logistic model

Linear & quadratic discriminant analysis

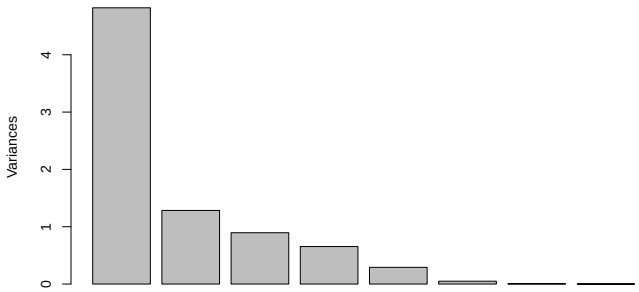


Figure 7: Eigenvalues of the correlation matrix of the training data, in descending order.

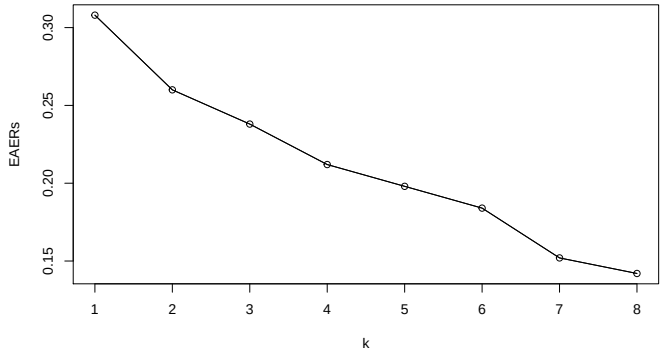


Figure 8: Prediction error rates on the test set after fitting a logistic model on the training set using only the first k PC.

Predictor variable j	$\hat{\beta}^+$	PC1	PC2	PC3	PC4	PC5
`engine_capacity`	0.89	0.355	0.192	−0.371	−0.220	0.806
`urban_metric`	−0.26	0.447	−0.021	−0.040	0.042	−0.222
`extra_urban_metric`	−0.39	0.444	−0.003	−0.065	0.073	−0.201
`combined_metric`	−0.31	0.451	−0.016	−0.052	0.054	−0.214
`noise_level`	0.01	0.182	0.337	0.757	−0.527	0.044
`co2`	−0.84	0.447	0.092	−0.052	0.112	−0.184
`co_emissions`	−0.15	0.190	−0.557	0.508	0.474	0.412
`nox_emissions`	−4.73	0.065	0.728	0.140	0.653	0.094
`transmission_type_Manual`	−0.59					
`fuel_type_Hybrid`	−4.99					
`fuel_type_Petrol`	−4.44					
Intercept	1.70					

Table 3

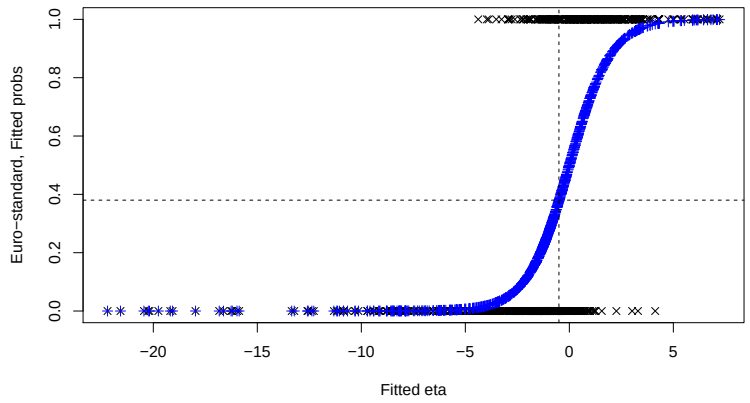


Figure 9

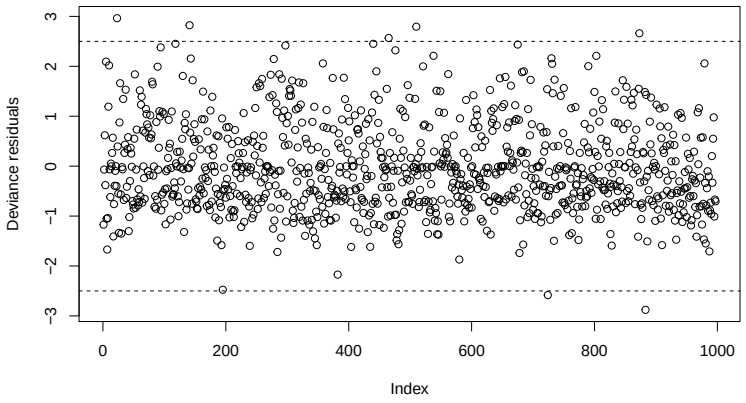


Figure 10

References

[1] Rousseeuw, P. J. and Leroy, A. M. (1987) *Robust Regression and Outlier Detection.* Wiley, 1987

[2] Rousseeuw, P. J. and van Driessen, K. (1999) *A fast algorithm for the minimum covariance determinant estimator.* Technometrics 41, 212–223.

[3] Box, G. E. P. and Cox, D. R. (1964) *An analysis of transformations (with discussion).* Journal of the Royal Statistical Society B, 26, 211–252.

[4] Hubert, M (2017) *Statistische modellen en data-analyse.* Acco. Deel II, p. 155-156.

A Code

src/init.R

```r
rm(list = ls()) # Clear environment
options(digits = 4) # Set output display precision
cat("\014") # Clear console
# Hit Ctrl+Shift+L to clear plots

X = read.table('data.txt', header=TRUE)
X$euro_standard = factor(X$euro_standard)

set.seed(0380267)
selVec <- c(sample(1:dim(X)[1],1000))
XTrain <- X[selVec,]
XTest <- X[-selVec,]

# These identifying variables will not be included in any models
id_col_names = c('manufacturer',
                 'model',
                 'description')
id_cols = which(names(X) %in% id_col_names)

factor_col_names = c('euro_standard',
                     'transmission_type',
                     'fuel_type')
factor_cols = which(names(X) %in% factor_col_names)

continuous_cols = which(!(names(X) %in% id_col_names)
                        & !(names(X) %in% factor_col_names))

# Plotting init
library(ggplot2)
univariate_plot_theme = theme(axis.title.y=element_blank(),
                              axis.title.x=element_blank(),
                              axis.text.y=element_blank(),
                              axis.ticks.y=element_blank(),
                              panel.grid.major.y=element_blank(),
                              panel.grid.minor.y=element_blank(),
                              legend.position='none')

plot_stdres_bounds = function() {
  abline(-2.5, 0, lty=2)
  abline(+2.5, 0, lty=2)
}
```

src/remove_RD_outliers.R

```r
# Find, save, and remove all outliers

library(robustbase)

data = XTrain[,continuous_cols]
mcd = covMcd(data)
d = sqrt(mahalanobis(data, mcd$center, mcd$cov))
outlier_indexes = which(d > 20)

outliers = XTrain[outlier_indexes,]
XTrain = XTrain[-outlier_indexes,]
```

src/outliers_plot.R

```
source('init.R')
source('remove_RD_outliers.R')

save = function(affix) {
  filename = paste('../multivar_outlier_', affix, '.pdf', sep='')
  ggsave(filename, width=8, height=1.25)
}

df = data.frame(robust_distance=d)

p = ggplot(df, aes(x=seq_along(robust_distance), y=robust_distance, alpha=0.2)) +
  scale_x_continuous(expand=c(0.3,0.3)) +
  geom_point() +
  coord_flip() +
  ggtitle('Robust distance') +
  univariate_plot_theme
print(p)
save('all')

p +
  coord_flip(ylim=c(0,40)) +
  ggtitle('(zoomed in)')
save('zoom')
```

src/pairs.R

```
source('init.R')
source('remove_RD_outliers.R')

library(GGally)

# Pairwise plots
ggpairs(XT, columns=c(continuous_cols, factor_cols),
        aes(alpha=0.01),
        labeller='label_parsed')
ggsave('../pairs.pdf', width=15, height=13)
```

src/transformer.R

```
# ML power transform of continuous variables

transform = function(XX, ignored_col='co2', verbose=FALSE, XSource=NULL) {
  # The maximum likelihood transforms are calculated using XSource (default: XX)
  if (is.null(XSource)) {
    XSource = XX
  }
  for (col in continuous_cols) {
    name = names(XX)[col]
    pt = powerTransform(XSource[,col])
    pt$lambda = pt$lambda
    transf_coldata = bcPower(XX[,col], pt$lambda)
    if ((abs(pt$lambda) < 3) && (name != ignored_col)) {
      XX[,col] = transf_coldata
    }
    if (verbose) {
      qqnorm(XX[,col], main=name)
      qqline(XX[,col])
      p_before = shapiro.test(XX[,col])$p
      #
```

```
        qqnorm(transf_coldata, main=name)
        qqline(transf_coldata)
        p_after = shapiro.test(transf_coldata)$p
        #
        print(name)
        print(as.numeric(pt$lambda), digits=3)
        print(p_before, digits=3)
        print(p_after, digits=3)
      }
    }
    return(XX)
}
```

src/lm.R

```
# Model selection by bidirectional stepwise regression using AIC.

source('init.R')
source('remove_RD_outliers.R')
source('transformer.R')

library(MASS)
library(car)

transform_predictors = TRUE
remove_stdres_outliers = TRUE
transform_response = TRUE

# Copy training set
XT = XTrain

# (Extra plot: MD-RD)
cont_cols = XT[,continuous_cols]
mcd = covMcd(cont_cols)
rd = sqrt(mahalanobis(cont_cols, mcd$center, mcd$cov))
md = sqrt(mahalanobis(cont_cols, colMeans(cont_cols), cov(cont_cols)))

# Remember: data got shuffled in init.R.
# First/top number is original row;
# Second/bottom number is current row / indexing number.

# ML power transform of continuous variables
if (transform_predictors) {
  XT = transform(XT, verbose=TRUE)
}

# Define, save and remove lm outliers
# (These were found with 'which(abs(stdres(lm1)) > 5))')
if (remove_stdres_outliers) {
  lm_outliers_idx = c(176, 317, 635)
  lm_outliers = XT[lm_outliers_idx,]
  XT = XT[-lm_outliers_idx,]
}

expand_factors = function(full_X) {
  data = full_X[,c(continuous_cols, factor_cols)]
  # Expand factor columns into a set of 'one hot' columns
  # eg: fuel_type -> (fuel_typeHybrid, fuel_typePetrol)
  # (fuel_type=Diesel is then encoded as (0,0))
  design_matrix = model.matrix(lm(co2~., data))
  X_encoded = cbind(design_matrix[,-1], data['co2',drop=FALSE])
```

```
  return(X_encoded)
}

XT_encoded = expand_factors(XT)

vars = colnames(XT_encoded)
vars = vars[-length(vars)] # Remove 'co2'
full_formula = as.formula(paste("~",paste(vars,collapse="+")))

if (transform_response) {
  # The power is determined by running:
  # b = boxcox(model_without_transformed_respons)
  # lambda = b$x[which.max(b$y)]
  response = 'co2^-0.3'
} else {
  response = 'co2'
}

# Start with a full linear model
full_lm = lm(as.formula(paste(response,'~.')), XT_encoded)
lm1 = stepAIC(full_lm, list(lower=~1, upper=full_formula), direction='both')

# Start with an empty model
empty_model = lm(as.formula(paste(response,'~1')), XT_encoded)
lm2 = stepAIC(empty_model, list(lower=~1, upper=full_formula), direction='both')

# (Extra: model with interaction terms)
# full2_lm = lm(as.formula(paste(response,'~.^2')), XT_encoded)
# lm1 = stepAIC(full2_lm, list(lower=~1, upper=~.^2), direction='both')

# Check whether the two seeds converged to the same model
same_vars = setequal(names(lm1$coefficients), names(lm2$coefficients))
diff = lm1$coefficients[names(XT_encoded)] - lm2$coefficients[names(XT_encoded)]
same_coeffs = all((diff < 1e-12) | (is.na(diff)))
stepAIC_convergence = same_vars & same_coeffs
```

src/evaluate_lm.R

```
# Evaluate the found linear model

source('lm.R')

model = lm1

print(summary(model))
print(model$anova)

standardised_residuals = stdres(model)

for (col in c(factor_cols,continuous_cols)) {
  plot(XT[[col]], standardised_residuals, xlab=names(X)[col])
  plot_stdres_bounds()
}

plot(standardised_residuals)
plot_stdres_bounds()

plot(fitted(model), standardised_residuals)
plot_stdres_bounds()

b = boxcox(model)
```

```
lambda = b$x[which.max(b$y)]

qqnorm(standardised_residuals)
qqline(standardised_residuals)
shapiro.test(standardised_residuals)

# Compare with robust linear model
# lmrob = ltsreg(log(co2)~., data)
# ltsPlot(lmrob, which=c('rfit'))

# Diagnostic plot
if (!remove_stdres_outliers) {
  plot(rd, stdres(lm1))
  plot_stdres_bounds()
}

# Find strong stdres outliers
which(abs(stdres(lm1)) > 5)

print(sum(model$residuals^2))
print(lambda, digits=5)
print(stepAIC_convergence)


# Test the model on the test set
# Important: needs to use same transform for unbiased predictions.
XTT = expand_factors(transform(XTest, XSource=XTrain))
y = XTT$co2^-0.3
yhat = predict(lm1, XTT)

plot(y,yhat)
abline(0,1)

e = (y - yhat)
e_rel = e/y
qqnorm(e_rel)
qqline(e_rel)
plot(e_rel)
plot(yhat, e_rel)

n = 500
p = 11
sigma = sum(e^2)/(n-p)
print(sigma)

msre = sum(e_rel^2)/(n-p)
print(msre, digits=5)

mare = sum(abs(e_rel))/(n-p)
print(mare, digits=5)

# Manual prediction
# x = c(1, as.numeric(XTT[1,-c(5,11,14)]))
# w = as.numeric(summary(lm1)$coeff[,1])
# y[1]
# yhat[1]
# w %*% x
#
# plot(XT_encoded$combined_metric, XT_encoded$co2^-0.3)
# points(XTT$combined_metric, XTT$co2^-0.3, col='blue')
```

src/multicol.R

```
source('init.R')
source('remove_RD_outliers.R')

c = cor(XTrain[continuous_cols])
I = diag(nrow=dim(c)[1])
ridge = 0
diag(solve(c+ridge*I))
ev = eigen(c)$values
sqrt(max(ev)/ev)
```

src/classification_init.R

```
source('init.R')
source('remove_RD_outliers.R')

library(MASS)

make_eurostandard_binary = function(X_in) {
  # euro_standard had been made a factor -- we must undo this.
  f = as.numeric(levels(X_in$euro_standard))[X_in$euro_standard]
  f = as.factor(f > 4)
  f = factor(f, labels=c('old', 'new'))
  X_in$euro_standard = f
  return(X_in)
}

XTrain = make_eurostandard_binary(XTrain)
XTest = make_eurostandard_binary(XTest)


# Proportion of new cars
cutoff = mean(as.numeric(XTrain$euro_standard) - 1)

get_table_glm = function(model, testing_set) {
  # Find estimated error rate on test set
  # Find predicted probabilities using reduced model
  pred.prob = predict(model, testing_set, type='response')
  # Bin predicted probabilities according to found proportion
  yhat = ifelse(pred.prob > cutoff, 1, 0)
  # Find real responses
  y = as.numeric(testing_set$euro_standard) - 1
  # Compare in table
  return(prop.table(table(y, yhat)))
}

get_table_DA = function(model, testing_set) {
  p = predict(model, testing_set)
  return(table(p$class, testing_set$euro_standard))
}

get_error_rate = function(model, testing_set, type='glm') {
  if (type=='glm') {
    table = get_table_glm(model, testing_set)
  } else {
    table = prop.table(get_table_DA(model, testing_set))
  }
  return(1 - sum(diag(table)))
}
```

src/PCA.R

```
source('classification_init.R')

library(rrcov)

XTrain_0 = scale(XTrain[continuous_cols])
XTest_0 = scale(XTest[continuous_cols])

find_PCA_k_test_error_rate = function(k) {
  pca = PcaClassic(XTrain_0, k=k)
  Z = pca@loadings
  # Todo: outliers? MD-OD plot
  XTrain_PCA_cols = XTrain_0 %*% Z
  XTest_PCA_cols = XTest_0 %*% Z
  XTrain_PCA_and_factors = cbind(XTrain_PCA_cols, XTrain[factor_cols])
  XTest_PCA_and_factors = cbind(XTest_PCA_cols, XTest[factor_cols])

  # Construct full logistic model
  fullmod = glm(euro_standard~., XTrain_PCA_and_factors, family=binomial)
  # Prune model
  redmod = stepAIC(fullmod, list(lower=~1,upper=~.), direction='both')

  return(get_error_rate(redmod, XTest_PCA_and_factors))
}

# We bepalen k zodat error rate op test set minimaal is.
EAERs = c(1:8)
for (k in 1:8) {
  EAERs[k] = find_PCA_k_test_error_rate(k)
}
EAERs
plot(EAERs)
lines(EAERs, xlab='k')

pca = PcaClassic(XTrain_0, k=8)
screeplot(pca, main='')
pca@eigenvalues

# Decision: retain k PC
k = 5
pca = PcaClassic(XTrain_0, k=k)
Z = pca@loadings
XTrain_PCA = XTrain_0 %*% Z
XTest_PCA = XTest_0 %*% Z
XTrain_PCA_and_factors = cbind(XTrain_PCA, XTrain[factor_cols])
XTest_PCA_and_factors = cbind(XTest_PCA, XTest[factor_cols])
XTrain_PCA_and_response = cbind(XTrain_PCA, XTrain['euro_standard'])
XTest_PCA_and_response = cbind(XTest_PCA, XTest['euro_standard'])
```

src/log_res.R

```
# Logistic model

source('PCA.R')

# Construct full logistic model
fullmod = glm(euro_standard~., XTrain_PCA_and_factors, family=binomial)
# Prune model
lrmod = stepAIC(fullmod, list(lower=~1,upper=~.), direction='both')

print(get_error_rate(lrmod, XTrain_PCA_and_factors))
```

```
print(get_error_rate(lrmod, XTest_PCA_and_factors))


# Analyse found model

# Iterativa algo convergance
print(lrmod$converged)
print(lrmod$iter)

summary(lrmod)
# Residual deviance: 776 on 987 dof
qchisq(0.95,987) # = 1061 > 776 --> do not reject model
# Deviance of fit: 1324.28 - 775.97 = 548.3 on 4 degrees of freedom
# So this fit explains a lot more than a constant probability.
qchisq(0.95,4) # = 10

# Deviance residuals
plot(residuals(lrmod, 'deviance'), ylab='Deviance residuals')
plot_stdres_bounds()

# Plot logistic fit.
eta = predict(lrmod, XTrain_PCA_and_factors, type="link")
fitted_probs = predict(lrmod, XTrain_PCA_and_factors, type="response")
y = as.numeric(XTrain_PCA_and_factors$euro_standard) - 1

plot(eta, y, pch=4, xlab="Fitted eta", ylab="Euro-standard, Fitted probs")
points(eta, fitted_probs, pch=3, col='blue')
abline(h=cutoff, lty=2)
abline(v=-0.5, lty=2)


# Transform model coefficients back to input variable space
#
# Get PC coefficients only
PC_coeffs = lrmod$coefficients[2:(2+k-1)]
input_coeffs = Z %*% PC_coeffs
print(input_coeffs)
```

src/LDA_QDA.R

```
source('PCA.R')

LDA_model = lda(euro_standard~., XTrain_PCA_and_response)

print(get_error_rate(LDA_model, XTrain_PCA_and_response, type='da'))
print(get_error_rate(LDA_model, XTest_PCA_and_response, type='da'))


QDA_model = qda(euro_standard~., XTrain_PCA_and_response)

print(get_error_rate(QDA_model, XTrain_PCA_and_response, type='da'))
print(get_error_rate(QDA_model, XTest_PCA_and_response, type='da'))
```