Exercise 1

We consider the dataset diabetes (Efron, Hastie, Johnstone and Tibshirani (2003) "Least Angle Regression" Annals of Statistics) from the package `lars`. Ten baseline variables age, sex, body mass index (bmi), average blood pressure (map) and six blood serum measurements (tc, ldl, hdl, tch, ltg, glu), as well as disease progression one year after baseline (y), were obtained for n=442 diabetes patients. The baseline data is stored in `x` while a model matrix including interactions between baseline measurements is stored in `x2`. Here, we aim to predict the disease progression, one year after baseline based on the matrix x2. You can access the data via

```
# install.packages("lars")
library(lars)

## Warning:  package 'lars' was built under R version 3.4.4
## Loaded lars 1.2

data("diabetes")
```

(a) Split the data set into a training and a test set. Sample 70% of the data to the training set, 30% to the test set. Set the seed to 100 (`set.seed(100)`).

```
set.seed(100)

# get index for 70% of the data
train_idx = sample(1:nrow(diabetes), 0.7*nrow(diabetes))
train <- diabetes[train_idx,c("y","x2")]
test <- diabetes[-train_idx,c("y","x2")]

# We split the data into a train and test set because we aim to
# develop a model which is good for prediction. The training data is
# used to fit a model. Now, if we would predict the training data with
# this model, we would get an overly optimistic prediction performance.
# Instead, to decide how good the model is, we predict new, unseen test
# data. Since this data is not used for the fitting process, we know how
# good the model performes for prediction.
```

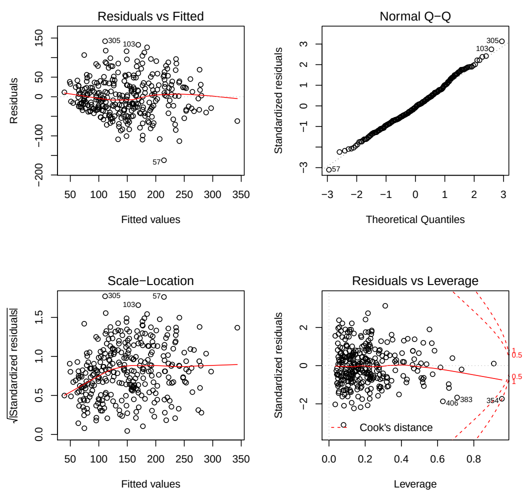
(b) Fit a linear regression model based on the training data and check the model assumptions. Is it important that all the assumptions are met? Now, use the model for prediction on the test data. Calculate the test error in terms of the mean squared error (MSE) and the mean absolute percentage error (MAPE) using OLS. Consider the predicted vs. the observed values on the test data.

Email: hezo@zhaw.ch

```
# x2 contains interactions etc (you should
# consider train$x2 to see how it looks like). Giving x2 is the
# same as giving a formula like y~age+sex+...+age:sex+age:bmi...

# We fit the linear regression on the train data
mod <- lm(y~x2, data=train)
summary(mod)
```

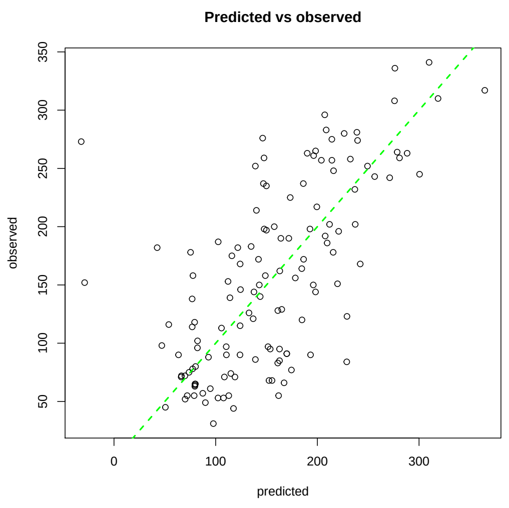
```
##
## Call:
## lm(formula = y ~ x2, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -162.396  -31.078   -3.148   29.643  141.968
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    151.285      3.211  47.109  < 2e-16 ***
## x2age           12.901     85.143   0.152  0.87969
## x2sex         -190.338     82.425  -2.309  0.02177 *
## x2bmi          377.872    115.199   3.280  0.00119 **
## x2map          231.733    100.174   2.313  0.02154 *
## x2tc        -19648.891  64386.998  -0.305  0.76050
## x2ldl        17211.983  56584.458   0.304  0.76125
## x2hdl         7061.042  24070.722   0.293  0.76951
## x2tch          -16.251    345.102  -0.047  0.96248
## x2ltg         7243.646  21169.619   0.342  0.73252
## x2glu           35.088     90.743   0.387  0.69933
## x2age^2         99.230     86.208   1.151  0.25084
## x2bmi^2         98.643    122.428   0.806  0.42119
## x2map^2         28.890     90.247   0.320  0.74915
## x2tc^2         136.230   9376.141   0.015  0.98842
## x2ldl^2       -971.065   6818.355  -0.142  0.88687
## x2hdl^2       -847.828   2429.442  -0.349  0.72740
## x2tch^2       1310.289    780.017   1.680  0.09427 .
## x2ltg^2       1158.445   1846.271   0.627  0.53095
## x2glu^2         90.027    110.335   0.816  0.41533
## x2age:sex      156.478     94.522   1.655  0.09912 .
## x2age:bmi      -90.200    107.401  -0.840  0.40182
## x2age:map       38.504     98.186   0.392  0.69529
## x2age:tc      -712.407    845.810  -0.842  0.40046
## x2age:ldl      338.366    677.871   0.499  0.61812
## x2age:hdl      439.196    383.092   1.146  0.25273
## x2age:tch      308.114    283.590   1.086  0.27834
## x2age:ltg      329.231    298.426   1.103  0.27102
## x2age:glu       61.207    100.420   0.610  0.54275
## x2sex:bmi       49.579    109.161   0.454  0.65010
## x2sex:map       76.282    100.337   0.760  0.44783
## x2sex:tc      1732.571    959.500   1.806  0.07220 .
## x2sex:ldl    -1461.008    768.291  -1.902  0.05840 .
## x2sex:hdl     -615.455    424.122  -1.451  0.14803
## x2sex:tch     -192.215    258.983  -0.742  0.45869
## x2sex:ltg     -527.721    335.377  -1.574  0.11689
## x2sex:glu       88.975     91.273   0.975  0.33061
## x2bmi:map       76.129    124.727   0.610  0.54219
## x2bmi:tc       563.706    997.144   0.565  0.57238
## x2bmi:ldl     -320.287    848.540  -0.377  0.70616
## x2bmi:hdl     -463.675    492.374  -0.942  0.34727
## x2bmi:tch     -366.729    348.217  -1.053  0.29331
## x2bmi:ltg     -158.937    374.380  -0.425  0.67155
## x2bmi:glu      -18.519    131.696  -0.141  0.88828
## x2map:tc      -692.624   1176.219  -0.589  0.55650
## x2map:ldl      660.140   1001.921   0.659  0.51060
## x2map:hdl      231.736    503.491   0.460  0.64574
## x2map:tch      -39.519    280.077  -0.141  0.88791
## x2map:ltg      257.768    423.731   0.608  0.54354
## x2map:glu     -152.860    114.685  -1.333  0.18382
## x2tc:ldl      1253.551  15388.076   0.081  0.93514
## x2tc:hdl      1649.567   5575.221   0.296  0.76758
## x2tc:tch     -1198.123   2457.628  -0.488  0.62633
## x2tc:ltg      2561.147  14638.499   0.175  0.86126
## x2tc:glu        81.902    919.932   0.089  0.92913
## x2ldl:hdl    -2201.100   4634.229  -0.475  0.63524
## x2ldl:tch       51.932   2093.967   0.025  0.98023
## x2ldl:ltg    -2395.625  12116.436  -0.198  0.84343
## x2ldl:glu     -240.020    798.622  -0.301  0.76402
## x2hdl:tch     1257.690   1301.771   0.966  0.33493
## x2hdl:ltg    -1503.305   5243.996  -0.287  0.77461
## x2hdl:glu      231.672    417.732   0.555  0.57968
## x2tch:ltg      -51.091    840.475  -0.061  0.95158
## x2tch:glu      486.218    299.345   1.624  0.10561
## x2ltg:glu       52.390    380.447   0.138  0.89059
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 54.58 on 244 degrees of freedom
## Multiple R-squared:  0.5908,Adjusted R-squared:  0.4835
## F-statistic: 5.506 on 64 and 244 DF,  p-value: < 2.2e-16
```

```
# check the model assumptions
par(mfrow=c(2,2))
plot(mod)
```



```
# The model assumptions are not violated. However, even if
# they were, it wouldn't be so important because we aim to
# use the model for prediction. If it predicts the test data
# well we don't are about the violations on the training data

# predict the data on the test set
predLM <- predict(mod, newdata = test)

# MSE and MAPE for the test data
mean((test$y - predLM)^2)

## [1] 3896.808

mean(abs((predLM - test$y))/test$y)
```

```
## [1] 0.3926821

# The MSE = mean absolut error.
# The MAPE is a percentage indicating the percentage of the error
# --> In both cases, lower values are better.

# plot the observed vs the fitted
par(mfrow=c(1,1))
plot(predLM, test$y, main="Predicted vs observed",
     xlab="predicted", ylab="observed")
abline(0,1, col='green', lty=2, lwd=2)
```

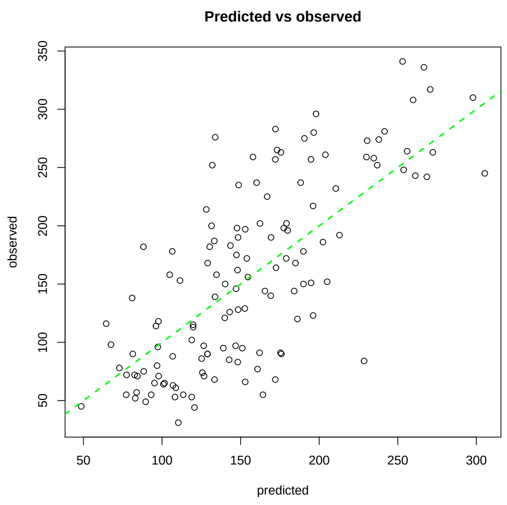


```
# The linear model seems to fit the data already quite well.
# green line=main diagonal
```

(c) Now, we aim to predict the test data using ridge regression. Recall what ridge regression is doing and what's the impact of λ. Perform a cross validation to find the best parameter λ based on the training data. Is the model, fitted with the optimal parameter λ, a better prediction model for the test data compared to the linear regression model? Plot the predicted vs. the observed values.

```
library(glmnet)

## Loading required package:  Matrix

## Loading required package:  foreach

## Loaded glmnet 2.0-16
```

```
set.seed(100)

# fit a ridge regression on the training data
mod_ridge = cv.glmnet(x=train$x2, y=train$y, alpha=0)

# In ridge regression we extend the optimization objective using a
# penalty term for large coefficients, given by the sum of squared
# coefficients. Lambda is a tuning parameter that determines the
# contribution of the penalty term to the equation. The penalty leads
# to coefficients, shrinked towards zero. These coefficients are less
# optimal on the training data but they lead to better predictions on
# new, unseen test data.

# get the best lambda
lambda_ridge <- mod_ridge$lambda.min
lambda_ridge

## [1] 30.85116

# predict the data in the test set
predRidge <- predict(mod_ridge, newx = test$x2, s = lambda_ridge)

# MSE and MAPE on the test data
mean((test$y - predRidge)^2)

## [1] 2904.235

mean(abs((predRidge - test$y))/test$y)

## [1] 0.3840076

# The MSE and MAPE decrease compared to the linear model
# indicating an improved prediction performance.

# plot the observed vs the fitted
par(mfrow=c(1,1))
plot(predRidge, test$y, main="Predicted vs observed",
     xlab="predicted", ylab="observed")
abline(0,1, col='green', lty=2, lwd=2)
```

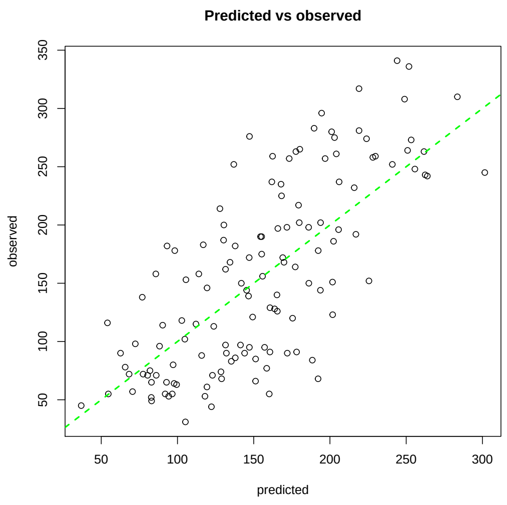


```
# The slope of the predicted data seems to be close to one
# indicating a good prediction on the test data
```

(d) Now, we aim to predict the test data using a lasso regression. What's the difference to the ridge regression? Perform a cross validation on the training data to find the best parameter λ (`cv.glmnet(..., alpha=1)`). Is the model, fitted with the optimal parameter λ, a better prediction model than the linear and the ridge regression? Plot the predicted vs the observed values.

```
set.seed(100)

# fit a lasso regression
mod_lasso <- cv.glmnet(x=train$x2, y=train$y, alpha=1)

# In Lasso and ridge regression we extend the optimization objective
# using a penalty term for large coefficients. Opposed to ridge
# regression, the penalty term is not the L2 but the L1 norm of the
# coefficients. This leads to shrinked coefficient estimates. These
# estimates are, in contrast to the ridge regression estimates, more often
# exactly equal to zero meaning, that the respective predictor is
# removed from the model.

# find the best lambda
lambda_lasso <- mod_lasso$lambda.min
lambda_lasso
```

```
## [1] 2.944893

# predict the data in the test set
predLasso <- predict(mod_lasso, newx = test$x2, s=lambda_lasso)

# MSE and MAPE
mean((test$y - predLasso)^2)

## [1] 2891.84

mean(abs((predLasso - test$y))/test$y)

## [1] 0.3764373

# The MSE and MAPE decrease comapared to the linear and the ridge regression
# model indicating improved predictions.

# plot the observed vs the fitted
par(mfrow=c(1,1))
plot(predLasso, test$y, main="Predicted vs observed",
     xlab="predicted", ylab="observed")
abline(0,1, col='green', lty=2, lwd=2)
```



(e) Calculate the predictions on the training data for each of the three models. Which model fits best? Do the results make sense?

```r
# predict the results on the training data
predLM_train <- predict(mod, train)
predRidge_train <- predict(mod_ridge, newx = train$x2, s = lambda_ridge)
predLasso_train <- predict(mod_lasso, newx = train$x2, s = lambda_lasso)

# get the MSE
mean((train$y - predLM_train)^2)

## [1] 2352.198

mean((train$y - predRidge_train)^2)

## [1] 2716.099

mean((train$y - predLasso_train)^2)

## [1] 2791.764

# get the MAPE
mean(abs((predLM_train - train$y))/train$y)

## [1] 0.3385566

mean(abs((predRidge_train - train$y))/train$y)

## [1] 0.3815519

mean(abs((predLasso_train - train$y))/train$y)

## [1] 0.3842834

# Based on the training data, the linear model is better than the ridge
# regression which fits slightly better than the lasso regression.
# This makes sense because the least square estimates lead to the best
# and unbiased model w.r.t to the data used for fitting the model. By
# extending the optimization objective with the penalty term for large
# coefficients, we get shrinked coefficient estimates which are less optimal
# on the training data. However, they lead to better prediction performance
# on new test data as seen in the previous exercises.
```