Data Analytics I: Predictive Econometrics
Self-Study

Lauritz Storch

2022-02-14

Data Preparation and Package Installation

You work for a health insurance company in South America. The data file obesity.Rdata contains the current Body Mass Index (BMI) of a sample of adults from Mexico, Peru and Colombia together with characteristics about their eating habits, physical condition and other variables collected via a survey. The insurance company wants to develop a model to predict BMI based on the survey questionnaire. The BMI predictions will serve for further cost prediction purposes as patients with obesity incur significantly higher medical costs. The file obesity_predict.Rdata contains adults for whom you are asked to deliver first BMI predictions.[1]

Exercise:
Predict the BMI using any method or approach that we either studied during the course or you potentially studied by yourself. **Implement max. three different estimation approaches that you know (e.g. OLS, Lasso, Ridge, Tree, Forest, etc.).** Select one approach that you believe gives the best predictions. Save these predictions into a file firstnames_lastnames.csv (replace with your name) and use comma as separator for the csv file. Additionally, submit your R code (containing also the approaches that you have implemented but have not used in your preferred prediction) and the answers to the questions below in a PDF format. Please submit all files via Canvas.

Important: Submit only predictions for BMI from one method in one csv file. I.e. the file will contain one column with the predicted BMI (do not change the order of the observations).

The points for the self-study depend mainly on the answers to the questions below and the R code. Additionally, the accuracy of your predictions will be assessed based on the R2 of the predicted BMI and will be used only marginally for the final score.

[1] Please make sure your device is connected to the internet and you are logged into Canvas to download the data sets.

Data Preparation and Package Installation

```
# Clear R
# rm(list=ls())
# cat("\014")
# graphics.off()

# Set Seed
set.seed(2023)

## Set working directory to load data
setwd("~/Desktop/Self Assignment")


## Packages and Library
# Packages function
install_if_missing <- function(p) {
  if (p %in% rownames(installed.packages()) == F) {
    try(install.packages(p, dependencies = T))
  } else {
    cat(paste("Skipping already installed package:", p, "\n"))
  }
}

# Define packages
packages = c("gridExtra","dplyr","kableExtra","knitr","lmtest","glmnet","corrplot",
             "gglasso","caret","stargazer") # R-Markdown
# packages = c("gridExtra","dplyr","lmtest","glmnet","corrplot",
#              "gglasso","caret","stargazer") # Code

# Install
invisible(sapply(packages, install_if_missing))

# read library
for(pkg in packages){
  suppressWarnings(suppressMessages(library(pkg, character.only = TRUE)))
}


## Read in data
load("obesity.Rdata")
load("obesity_predict.Rdata")


## Create folder for prediction results (might differ in a Windows environment)
options(warn = -1)
dir.create(paste0(getwd(),"//Prediction Results"))
options(warn = 0)
```

Relevant packages were loaded for the estimation of OLS, Post-Lasso[2], and a Random Forest.

Remark: Parts of the answers, which are marked with * are commented out in the R-Markdown file for visibility reasons. The corresponding code chunks are not commented out in the additional R file submitted.

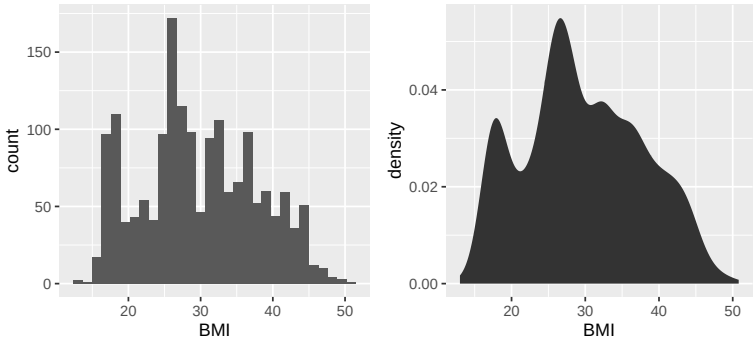
[2] I used a group Lasso for model selection for reasons that will be explained in the regarding tasks.

Question 1

> Are there any issues with the data (missing values, problematic labels, small groups in some categories, useless predictors, etc.)? Describe your data cleaning procedure.

Answer:

I started with a visual check over the data tables* and skimmed through it to catch anomalies and to become familiar with the data structure. In the next step, I checked the variable types with the *glimpse()*-function. Moreover, I plotted the histogram of the dependent variable *BMI* *.



The summary statistics of both data sets indicate that the variable *Age* is not identically distributed. It is crucial for test-training-splits that cross-sectional data is i.i.d, otherwise it is not guaranteed that cross-validation is working as a statistical learning device. Therefore, I drop data points larger than the maximum point of the test data set. NAs were not detected.

	Age train	Age test
	Min. :14.00	Min. :15.00
	1st Qu.:19.75	1st Qu.:19.75
	Median :22.96	Median :22.73
	Mean :27.65	Mean :24.74
	3rd Qu.:25.48	3rd Qu.:25.37
	Max. :61.00	Max. :43.00

Code:

```
# Data Processing ---------------------------------------------------------------
# # Visual Data check (commented out)
# View(obesity)
# View(obesity_predict)

# Data summary
glimpse(obesity)

# # Plot histogram
# options(warn = -1)
#
# # BMI predict
# discrete <- ggplot(data = obesity, aes(x = BMI)) +
#   geom_histogram(bins=30)
# continuous <- ggplot(data = obesity, aes(x = BMI)) +
#   stat_density()
# grid.arrange(arrangeGrob(discrete, continuous, ncol=2, nrow=1), heights=c(2,1))
```

```r
# Check summary
summary(obesity)
summary(obesity_predict)

# Prune variable Age as it is not iid
obesity <- obesity[c(which(obesity$Age <= max(obesity_predict$Age))),] # drop Age above 45

# Check NAs
if(all(is.na(obesity) == F)){
  print("No missing values (NAs) detected")
} else if (any(is.na(obesity)) == T) {
  print("Missing values detected")
}

if(all(is.na(obesity_predict) == F)){
  print("No missing values (NAs) detected")
} else if (any(is.na(obesity_predict)) == T) {
  print("Missing values detected")
}
```

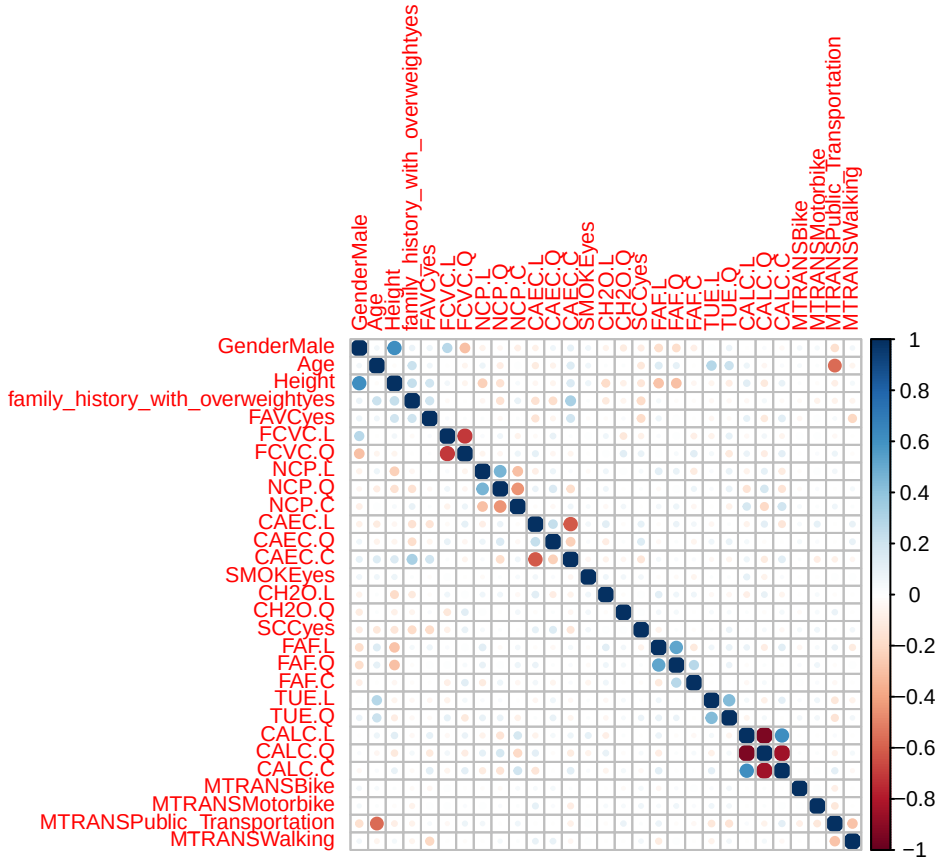
Question 2

> Describe your preferred prediction method/approach. Did you transform variables in training and test data? Did you partition the data? How did you select the tuning parameters? Which estimation method did you use? Why did you select this method/approach for your preferred specification?

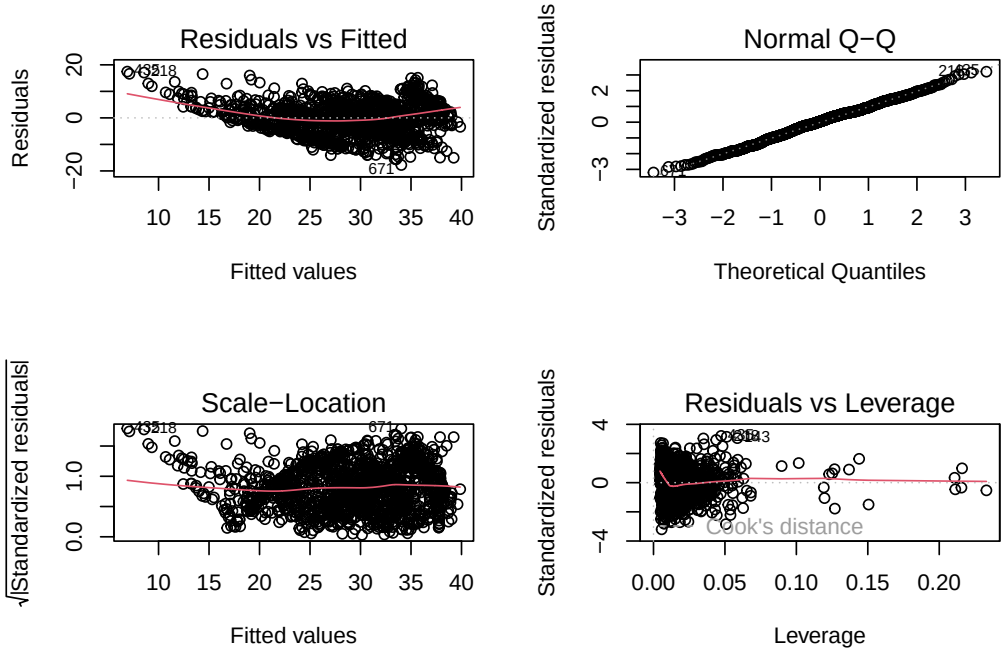
Answer:

Preference: I prefer the Random Forest (third prediction method) over the OLS (first prediction method) and Post-Lasso with a group Lasso for model selection (second prediction method). The Random Forest is a machine learning algorithm that combines multiple decision trees to create a more robust and accurate model. I use a 10 fold cross-validation to measure the optimum performance of the Random Forest model. For performance evaluation, I choose R^2. The *caret* package is used for this method.

Data transformation: I transform the data according to the information provided in VariableDescription.txt. Variables *Age* and *Height* are already properly numerically scaled. *FCVC*, *NCP*, *CH2O*, *FAF*, *TUE* are ordinal and sorted by numbers. I transform the variables into factors with ordered levels to account for their ordinal scale. Similarly, I had to transform *CAEC* and *CALC* to account for their ordinal scale with the distinction that they are sorted by word. *MTRANS* is nominal and has to be transformed into a factor and levels are preserved in words. The binary variables *Gender*, *family_history_with_overweight*, *FAVC*, *SMOKE*, *SCC* are likewise transformed.
Furthermore, I transform the variables into numeric matrix form. The *model.matrix()*-function partitions all factors into separate variables to account for their ordinal and nominal scale, respectively. Additionally, the function also transforms values of non-numeric variables into numeric values. This procedure is required to bring the data into a proper form for the Post-Lasso regression and the correlation plot. Later is used to check for multicollinearity between the predictors. The correlation plot shows that the variables *Gender* and *Height*, and *Age* and *Public_Transportation* value of *MTRANS* are highly correlated. Nevertheless, the correlation is not high enough to indicate multicollinearity. Other correlations are between the different linear, quadratic, and cubic components of variables, e.g. *CALC.L* and *CALC.C*, and can be neglected as these are grouped variables.
Tuning parameters are the number of variables that are considered as split candidates (*mtry*), which is optimized at *mtry*=15 and the number of minimum splits for each tree in the forest, which is equal to 5.



The outcome variable *BMI* is quantitatively given. I choose the two parametric and one non-parametric method as a regression model for prediction purposes. My preferred method is the Random Forest regression because the data sets face a misspecification issue, which can be seen from the Residual vs. Fitted plot of the OLS regression. The red line should be flat if the model is correctly specified.



As this misspecification will also affect the group Lasso regression, the Random Forrest is the better choice. Moreover, the Random Forest does not assume a functional form for the relationship between the dependent and independent variables and, hence, will produce a better in-sample performance[3] than models which underlie a parametric method. Even though a Random Forest tends to overfit due to recursive partitioning, this works in its favor because the test data set is a subsample of the training set.

Code:

```
# OLS ---------------------------------------------------------------------------
ols <- lm(obesity$BMI~.,as.data.frame(train)) # OLS regression
summary(ols)                                  # summary statistic

# # Plot
# options(warn = -1)
# par(mfrow= c(2,2))
# plot(ols)
# dev.off()
# options(warn = 0)

# Breusch-Pagan Test for heteroskedasticity
bptest(ols)

# Prediction OLS
ols_pred <- as.data.frame(predict(ols, newdata = as.data.frame(test)))
colnames(ols_pred) <- c("OLS prediction")

# # Write CSV (might differ in a Windows environment)
# write.csv(ols_pred, paste0(getwd(),"//Prediction Results//OLS.csv"), row.names=FALSE)


# Post-Lasso --------------------------------------------------------------------
# Group variables
colnames(train)
v.group <- c(1,2,3,4,5,6,6,7,7,7,8,8,8,9,10,10,11,12,12,12,13,13,14,14,14,15,15,15,15)

# Cross validated group Lasso
glasso.cv <- cv.gglasso(x=train,
                        y=obesity$BMI,
                        group=v.group,
                        loss="ls",
                        pred.loss="L2",
                        nfolds=10)

# Model selection
glasso <- coef(glasso.cv, s = "lambda.min") # variable selection

# Tuning Parameter (lambda with smallest in-sample MSE)
glasso.cv$lambda.min

# Post-Lasso regression
post_lasso <- lm(obesity$BMI ~. ,data=as.data.frame(train[,which(glasso!= 0)-1]))
summary(post_lasso)
```

[3] Measured via the mean squared error (MSE)

```r
# Prediction Post-Lasso
glasso_pred <- as.data.frame(predict(post_lasso, newdata = as.data.frame(test)))
colnames(glasso_pred) <- c("GL prediction")

# # Write CSV (might differ in a Windows environment)
# write.csv(ols_pred, paste0(getwd(),"//Prediction Results//GL.csv"), row.names=FALSE)


# Random forest ----------------------------------------------------------------
# Set up CV settings
forest_cv <- trainControl(method = "repeatedcv", number = 10, repeats = 1)

# Random Forest regression
forest <- train(BMI~.,
                obesity,
                method = "ranger",
                trControl = forest_cv,
                metric = "Rsquared")

# Tuning parameter
forest$bestTune # mtry: number of variables that are considered as split candidates
                # min.nodes: Minimum number of nodes within a single tree of a Forest

# Prediction Random Forest
forest_pred <- as.data.frame(predict(forest,obesity_predict))
colnames(forest_pred) <- c("RF prediction")

# Write CSV (might differ in a Windows environment)
write.table(forest_pred, paste0(getwd(),"//Prediction Results//lauritz_storch.csv"),
          row.names = F,
          sep = ",")
```

Question 3

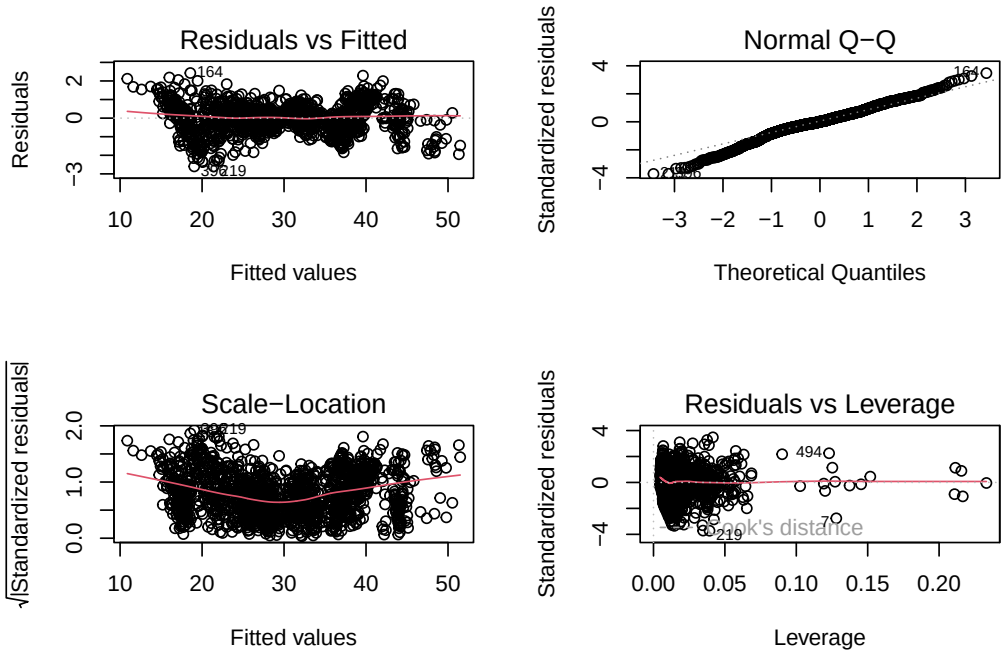
Besides your preferred specification, did you test additional variable transformations?

Answer:

Looking closer into the data, one can see that the test data set is a subsample of the training data set. Therefore, the true BMI can be reconstructed for the test data set. Secondly, the BMI is a deterministic measure and can be calculated via the formula

$$BMI = \frac{Weight}{Height^2}. \tag{1}$$

Obviously, *Weight* is not given in our data, but it can be reconstructed. Applying OLS on the reconstructed data reveals that the model is misspecified due to an omitted variable bias because *Weight* is dropped out of both data sets. The Residuals vs. Fitted plot shows a flat line now.



This conjecture is also strengthened by a Lasso regression which selects *Weight* and *Height* when the maximum number of variables to be nonzero is set to two (pmax = 2). The Post-Lasso on the variables selected gives a regression with an R^2 of 0.99.

	Dependent variable:
	BMI
Height	−32.994***
	(0.240)
Weight	0.340***
	(0.001)
Constant	56.399***
	(0.381)
Observations	1,688
R^2	0.990
Adjusted R^2	0.990
Residual Std. Error	0.812 (df = 1685)
F Statistic	81,711.630*** (df = 2; 1685)
Note:	*p<0.1; **p<0.05; ***p<0.01

Code:

```
# Appendix ------------------------------------------------------------------------
# Check values (true BMI)
row_vector <- rep(NA,length(rownames(obesity_predict)))
for (i in seq(length(rownames(obesity_predict)))){
  row_vector[i] <- which(rownames(obesity_predict)[i] == rownames(obesity))
```

```
}
true_BMI <- matrix(obesity$BMI[row_vector])

# Reconstructed BMI in test data
obesity2 <- obesity
obesity_predict2 <- obesity_predict
obesity_predict2$BMI <- true_BMI

# Reconstruct Weight for both data sets
obesity2$Weight <- obesity$BMI*(obesity$Height)^2
obesity_predict2$Weight <- obesity_predict2$BMI*(obesity_predict2$Height)^2

train2 <- model.matrix(~ .,
                       data=obesity2,
                       contrast.arg = lapply(sapply(obesity2, is.factor),
                                             contrasts,
                                             contrasts = FALSE))[,-1]

train2 <- train2[,-(ncol(train2)-1)] # Drop dependent variable BMI

test2 <- model.matrix(~ .,
                      data=obesity_predict2,
                      contrast.arg = lapply(sapply(obesity_predict2, is.factor),
                                            contrasts,
                                            contrasts = FALSE))[,-1]
test2 <- test2[,-(ncol(test2)-1)]

# OLS ------
ols2 <- lm(obesity2$BMI~.,as.data.frame(train2)) # OLS regression
summary(ols2)                                    # summary statistic

# # Plot
# par(mfrow= c(2,2))
# plot(ols2)
# dev.off()

# Post-Lasso -----
k <- 2 # number of variables to select for post-lasso regression
options(warn = -1)
lasso.cv <- cv.glmnet(train2,
                      y = obesity2$BMI,
                      alpha = 1,
                      standardize = T,
                      family = "gaussian",
                      type.measure = "mse",
                      nfolds = 10,
                      pmax = k)
options(warn = 0)
lasso <- coef(lasso.cv, s = "lambda.min") # variable selection

post_lasso2 <- lm(obesity2$BMI ~. ,data=as.data.frame(train2[,which(lasso!= 0)-1]))
summary(post_lasso2)
```

```r
report_regression <- function(model, format, ...){
  if(format == "latex"){
    require(stargazer)
    stargazer(model, ...)
  } else {
    print("This only works with latex output")
  }
}

report_regression(post_lasso2, format = "latex")
```

Question 4

> Besides your preferred specification, did you test additional ways to select the tuning parameters and/or partition the sample?

Answer:

I increase the number of folds for the CV to change the tuning parameters in the Random Forest. The increments of folds have only minor effects on the tuning parameters.
An additional tuning parameter is *lambda.min* in the group-Lasso regression, which is the λ of minimum mean cross-validated error. It is optimal at $\lambda = 0.0122$.
Reconstruction and the regarding partitioning of the data were already mentioned in task 3.

Question 5

> Besides your preferred specification, did you test additional estimation methods/approaches?

Answer:

As already explained in task 2, I used an OLS and Post-Lasso regression. I chose the OLS because it is the best linear unbiased estimator (BLUE) under the correct model specification. Considering the number of variables given, the Post-Lasso could have been a good approach to reduce the number of predictors used in the OLS regression and further improve the predictive power. However, due to the omitted variable bias and heteroskedasticity[4], both models are inappropriate because two Gauss-Markov assumptions are violated. Hence, OLS is not BLUE.

Question 6

> Did you use or test methods/approaches that were not covered in the lecture?

Answer:

I used a group Lasso instead of Lasso because the ordinal data points were split into several subcategories due to the data transformation, e.g., *CAEC.L*, *CAEC.Q*, and *CAEC.C*. Therefore, a Lasso would have been problematic because it would have selected only single subcategories of some ordinal predictors. The group Lasso addresses this issue by grouping all subcategories and either dropping or selecting variables' subcategories.

[4] It was tested with the Breusch-Pagan Test, which rejected homoskedasticity with a p-value $<$ 2.2e-16.