The aim of this exercise is to build a simple regression model that you can use to predict Distance (totale\_productiekosten).

Een regressie voor de voorspelling van de totale\_productiekosten

scatter.smooth(x=voorbeeld\_regressie$productiehoeveelheid, y=voorbeeld\_regressie$totale\_productiekosten, main="productiehoeveelheid ~ totale\_productiekosten") # scatterplot

par(mfrow=c(1, 2)) # divide graph area in 2 columns

boxplot(voorbeeld\_regressie$productiehoeveelheid, main="productiehoeveelheid", sub=paste("Outlier rows: ", boxplot.stats(voorbeeld\_regressie$productiehoeveelheid)$out)) # box plot for 'productiehoeveelheid'

boxplot(voorbeeld\_regressie$totale\_productiekosten, main="totale\_productiekosten", sub=paste("Outlier rows: ", boxplot.stats(voorbeeld\_regressie$totale\_productiekosten)$out)) # box plot for 'totale\_productiekosten'

par(mfrow=c(1, 2)) # divide graph area in 2 columns

plot(density(voorbeeld\_regressie$productiehoeveelheid), main="Density Plot: productiehoeveelheid", ylab="Frequency", sub=paste("Skewness:", round(e1071::skewness(voorbeeld\_regressie$productiehoeveelheid), 2))) # density plot for 'productiehoeveelheid'

polygon(density(voorbeeld\_regressie$productiehoeveelheid), col="red")

plot(density(voorbeeld\_regressie$totale\_productiekosten), main="Density Plot: Distance", ylab="Frequency", sub=paste("Skewness:", round(e1071::skewness(voorbeeld\_regressie$totale\_productiekosten), 2))) # density plot for 'totale\_productiekosten'

polygon(density(voorbeeld\_regressie$totale\_productiekosten), col="red")

cor(voorbeeld\_regressie$productiehoeveelheid, voorbeeld\_regressie$totale\_productiekosten) # calculate correlation between productiehoeveelheid and tijd

linearMod <- lm(totale\_productiekosten ~ productiehoeveelheid, data=voorbeeld\_regressie) # build linear regression model on full data

print(linearMod)

# capture model summary as an object

modelSummary <- summary(linearMod)

# model coefficients

modelCoeffs <- modelSummary$coefficients

# get beta estimate for productiehoeveelheid

beta.estimate <- modelCoeffs["productiehoeveelheid", "Estimate"]

# get std.error for productiehoeveelheid

std.error <- modelCoeffs["productiehoeveelheid", "Std. Error"]

# calc t statistic

t\_value <- beta.estimate/std.error

# calc p Value

p\_value <- 2\*pt(-abs(t\_value), df=nrow(voorbeeld\_regressie)-ncol(voorbeeld\_regressie))

# fstatistic

f\_statistic <- linearMod$fstatistic[1]

# parameters for model p-value calc

f <- summary(linearMod)$fstatistic

model\_p <- pf(f[1], f[2], f[3], lower=FALSE)

AIC(linearMod)

#=> 419.1569

BIC(linearMod)

#=> BIC => 424.8929

# Create Training and Test data -

set.seed(100) # setting seed to reproduce results of random sampling

trainingRowIndex <- sample(1:nrow(voorbeeld\_regressie), 0.8\*nrow(voorbeeld\_regressie)) # row indices for training data

trainingData <- voorbeeld\_regressie[trainingRowIndex, ] # model training data

testData <- voorbeeld\_regressie[-trainingRowIndex, ] # test data

# Build the model on training data

lmMod <- lm(totale\_productiekosten ~ productiehoeveelheid, data=trainingData) # build the model

totale\_productiekostenPred <- predict(lmMod, testData) # predict totale\_productiekostenance

summary (lmMod) # model summary

actuals\_preds <- data.frame(cbind(actuals=testData$totale\_productiekosten, predicteds=totale\_productiekostenPred)) # make actuals\_predicteds dataframe.

correlation\_accuracy <- cor(actuals\_preds) # 82.7%

head(actuals\_preds)

# Min-Max Accuracy Calculation

min\_max\_accuracy <- mean(apply(actuals\_preds, 1, min) / apply(actuals\_preds, 1, max))

# => 38.00%, min\_max accuracy

# MAPE Calculation

mape <- mean(abs((actuals\_preds$predicteds - actuals\_preds$actuals))/actuals\_preds$actuals)

# => 69.95%, mean absolute percentage deviation

DMwR::regr.eval(actuals\_preds$actuals, actuals\_preds$predicteds)

library(DAAG)

cvResults <- suppressWarnings(CVlm(df=voorbeeld\_regressie, form.lm=totale\_productiekosten ~ productiehoeveelheid, m=5, dots=FALSE, seed=29, legend.pos="topleft", printit=FALSE, main="Small symbols are predicted values while bigger ones are actuals.")); # performs the CV

attr(cvResults, 'ms')

# => 251.2783 mean squared error