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

scatter.smooth(x=voorbeeld\_regressie$bestelling, y=voorbeeld\_regressie$doorlooptijd, main="Bestelling ~ Tijd") # scatterplot

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

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

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

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

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

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

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

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

cor(voorbeeld\_regressie$bestelling, voorbeeld\_regressie$doorlooptijd) # calculate correlation between bestelling and tijd

linearMod <- lm(doorlooptijd ~ bestelling, 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 bestelling

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

# get std.error for bestelling

std.error <- modelCoeffs["bestelling", "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(doorlooptijd ~ bestelling, data=trainingData) # build the model

doorlooptijdPred <- predict(lmMod, testData) # predict doorlooptijdance

summary (lmMod) # model summary

actuals\_preds <- data.frame(cbind(actuals=testData$doorlooptijd, predicteds=doorlooptijdPred)) # 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=doorlooptijd ~ bestelling, 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