Challenge 2

```
# Damit setzen wir das Working Directory auf den Ordner dieser Datei
if (!is.null(parent.frame(2)$ofile)) {
  this.dir <- dirname(parent.frame(2)$ofile)
  setwd(this.dir)}
```

Get Data

```
library(data.table)

machines = fread(file = "machine_data.csv",
                 sep = ',',
                 header = TRUE,
                 stringsAsFactors = FALSE,
                 dec = '.')

products = fread(file = "product_data.csv",
                 sep = ',',
                 header = TRUE,
                 stringsAsFactors = FALSE,
                 dec = '.')

transaction = fread(file = "transactional_data.csv",
                    sep = ',',
                    header = TRUE,
                    stringsAsFactors = FALSE,
                    dec = '.')

mfailures = fread(file = "machine_failures.csv",
                  sep = ',',
                  header = TRUE,
                  stringsAsFactors = FALSE,
                  dec = '.')

mfailures = data.table(mfailures)
transaction = data.table(transaction)
products = data.table(products)
machines = data.table(machines)
```

1. Merge the transactional dataset with the machine failures data set setting failure variable to 0 when no failure is recorded

```
# merge the datasets
transactional = merge(transaction,mfailures, by = c('machine', 'timestamp', 'column'
), all.x=T)
setDT(transactional)
# set the failure value of normal records to 0
transactional[is.na(failure),failure:=0]
```

2. In the transactional data table, create a variable called “last_vend” containing the timestamp of the previous sale of each machine

```
#order by machine and date
transactional = transactional[order(machine,date)]

#create "last stand colum" with shift (one down):
transactional[, last_vend := shift(timestamp)]

# Empty first row for each machine since first selling point doesnt have a previous s
ale:
transaction[, last_vend := c(NA, timestamp[-.N]), by=machine]
summary(transactional)
```

```
##     machine       timestamp              column          date          
##  Min.   :   1   Length:1840477     Min.   : 0.00   Length:1840477    
##  1st Qu.: 880   Class :character   1st Qu.:13.00   Class :character  
##  Median :1472   Mode  :character   Median :25.00   Mode  :character  
##  Mean   :1462                      Mean   :27.03                     
##  3rd Qu.:2072                      3rd Qu.:39.00                     
##  Max.   :2724                      Max.   :69.00                     
##  product_name          failure          last_vend        
##  Length:1840477     Min.   :0.000000   Length:1840477    
##  Class :character   1st Qu.:0.000000   Class :character  
##  Mode  :character   Median :0.000000   Mode  :character  
##                     Mean   :0.006725                     
##                     3rd Qu.:0.000000                     
##                     Max.   :1.000000
```

3. Create a new variable in the transactional data table called "deltahours" containing, for every sale, the hours that passed since the last sale

```
help("difftime")
transactional[, deltahours := as.numeric(difftime(timestamp,last_vend,units = 'hours'
))]
# Only rows without NAs
transactional = transactional[complete.cases(transactional)]
head(transactional)
```

mach... <int>	timestamp <chr>	colu... <int>	date <chr>	product_name <chr>	fa
1	2017-01-02 12:32:20	54	2017-01-02	flavoured_carbonates_2	
1	2017-01-02 13:11:15	21	2017-01-02	sugar_confectionary_incl_gums_3	
1	2017-01-02 13:13:19	24	2017-01-02	sugar_confectionary_incl_gums_1	
1	2017-01-02 13:42:20	50	2017-01-02	unflavoured_carbonates_1	
1	2017-01-02 13:56:16	24	2017-01-02	sugar_confectionary_incl_gums_1	
1	2017-01-02 14:02:05	33	2017-01-02	chocolate_based_3	

6 rows | 1-6 of 8 columns

4. Create an auxiliary data table called "machine_daily_average" 2ith the average daily sales per machine. Use this auxiliary table to attach to every row of the transactional data table to the average daily sales per machine.

```
machine_daily_average = transactional[, .(daily_sales_machine=length(product_name)/un
iqueN(date)),by=machine]
transactional = merge(transactional,machine_daily_average, by = 'machine', all.x=T)
head(transactional)
```

mach... <int>	timestamp <chr>	colu... <int>	date <chr>	product_name <chr>	fa
1	2017-01-02 12:32:20	54	2017-01-02	flavoured_carbonates_2	
1	2017-01-02 13:11:15	21	2017-01-02	sugar_confectionary_incl_gums_3	
1	2017-01-02 13:13:19	24	2017-01-02	sugar_confectionary_incl_gums_1	
1	2017-01-02 13:42:20	50	2017-01-02	unflavoured_carbonates_1	
1	2017-01-02 13:56:16	24	2017-01-02	sugar_confectionary_incl_gums_1	
1	2017-01-02 14:02:05	33	2017-01-02	chocolate_based_3	

6 rows | 1-6 of 9 columns

5. Create a new variable called "delta" in the transactional data table containing a normalized version of deltahours consisting on the deltahours associated with each sale divided by the average deltahours of each machine i.e. delta = deltahours /(24/daily_sales_machine). The interpretation of delta is the amount of "missed sales" if the machine was selling at a constant rate

```
transactional$delta <- transactional[, .(delta=as.numeric(deltahours*daily_sales_mach
ine/24, units='hours'))]
head(transactional)
```

mach... <int>	timestamp <chr>	colu... <int>	date <chr>	product_name <chr>	fa
1	2017-01-02 12:32:20	54	2017-01-02	flavoured_carbonates_2	
1	2017-01-02 13:11:15	21	2017-01-02	sugar_confectionary_incl_gums_3	
1	2017-01-02 13:13:19	24	2017-01-02	sugar_confectionary_incl_gums_1	
1	2017-01-02 13:42:20	50	2017-01-02	unflavoured_carbonates_1	
1	2017-01-02 13:56:16	24	2017-01-02	sugar_confectionary_incl_gums_1	
1	2017-01-02 14:02:05	33	2017-01-02	chocolate_based_3	

6 rows | 1-6 of 10 columns

```
# Checking if high deltahours have a strong correlation with failure:
high_deltas = mean(transactional$failure[transactional$deltahours >
                                           unname(quantile(transactional$deltahours,
0.99,na.rm = TRUE))],na.rm = TRUE)

low_deltas = mean(transactional$failure[transactional$deltahours <
                                          unname(quantile(transactional$deltahours,0.
99, na.rm = TRUE))], na.rm = TRUE)

influence_deltas = data.table(high_deltas,low_deltas)
influence_deltas
```

high_deltas <dbl>	low_deltas <dbl>
0.2019126	0.004753934

1 row

Through this correlation analysis we intended to identify the relationship between the number of failures of a machine and lower sales. The higher correlation coefficient of machines with higher number of failures confirmed that they tend to have lower sales (which makes intuitively completely sense)

6. Select 30% of the machines in the transactional data for testing and 70% of the machines for training and train a linear logistic regression model called “m” to predict whether a machine has a failure as a function of variable delta. What is the value of the intercept and the coefficient accompanying variable delta?

```
set.seed(42)
idx <- sample(x=unique(transactional$machine), size=round(0.7*length(unique(transacti
onal$machine)), 0))

train <- transactional[machine %in% idx, ]
test <- transactional[!machine %in% idx, ]

m <- glm(failure ~ delta, train, family="binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(m)
```

```
##
## Call:
## glm(formula = failure ~ delta, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0331  -0.0616  -0.0482  -0.0447   3.7182
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.911617   0.023127  -298.9   <2e-16 ***
## delta        0.562050   0.003131   179.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 103681  on 1281524  degrees of freedom
## Residual deviance:  58250  on 1281523  degrees of freedom
## AIC: 58254
##
## Number of Fisher Scoring iterations: 9
```

Answer: Intercept : -6.911617; coef,delta: 0.562049

a) What's the AUC, a measure of quality, of the model you have built on the train set? and on test set?

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
pred_train = predict(m, newdata = train, type = "response")
pred_test = predict(m, newdata = test, type = "response")

auc(train$failure, pred_train)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
## Area under the curve: 0.9198
```

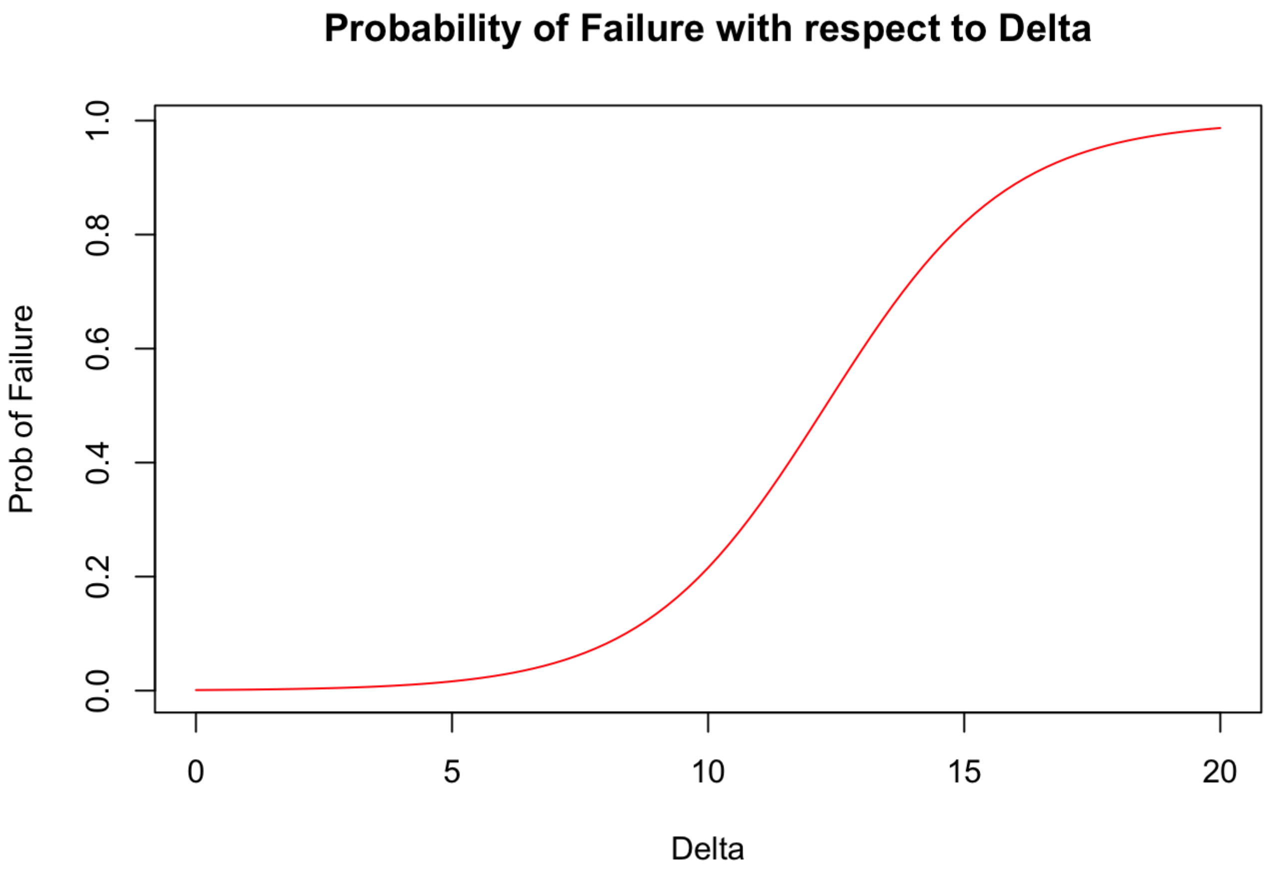
```
auc(test$failure, pred_test)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
## Area under the curve: 0.9206
```

b) Plot the function of probability of failure with respect to delta to gain intuition:

```
intercept = m$coefficients["(Intercept)"]
coefficient = m$coefficients["delta"]
f = function(x){(1/(1+exp(-(intercept + (coefficient*x)))))}
curve(f, 0, 20,
      xlab = "Delta", ylab="Prob of Failure", col="red",
      main = "Probability of Failure with respect to Delta")
```



c) Let us create alarms with two levels of priority: med-risk and high-risk. Med-risk alarms will fire when the probability of failure is >=60% and High-risk when that probability is >=80%.

i. What are the threshold deltas for each type of alarm to fire? #Hint (check the case)

```
# Can be solved using the function and log it:
log_f = function(x){
  (log((1/x)-1) + intercept) / -coefficient}

threshold_med_risk = log_f(0.6)
threshold_high_risk = log_f(0.8)

threshold_med_risk
```

```
## (Intercept)
##    13.01857
```

```
threshold_high_risk
```

```
## (Intercept)
##    14.76367
```

ii. How many of these alarms would be fired per day on average according to your model?

```
daily_alarm_med_risk = length(transactional$delta[transactional$delta >= threshold_me
d_risk]) / length(unique(transactional$date))

daily_alarm_high_risk = length(transactional$delta[transactional$delta >= threshold_h
igh_risk]) / length(unique(transactional$date))

daily_alarm_med_risk
```

```
## [1] 41.17778
```

```
daily_alarm_high_risk
```

```
## [1] 27.7
```

iii. What % of these will be “false alarms” i.e. failure variable is equal to 0, for each level of priority?

```
transactional$alarm_med_risk = 0
transactional$alarm_med_risk[transactional$delta >= threshold_med_risk] = 1

transactional$alarm_high_risk = 0
transactional$alarm_high_risk[transactional$delta >= threshold_high_risk] = 1

false_alarm_rate_med_risk = sum(transactional$alarm_med_risk[transactional$failure ==
0]) / sum(transactional$alarm_med_risk)
false_alarm_rate_high_risk = sum(transactional$alarm_high_risk[transactional$failure
 == 0]) / sum(transactional$alarm_high_risk)


sprintf("%1.2f%%", 100*false_alarm_rate_med_risk)
```

```
## [1] "20.24%"
```

```
sprintf("%1.2f%%", 100*false_alarm_rate_high_risk)
```

```
## [1] "8.90%"
```

d) In this exercise we will estimate the profit impact of our EWS system vs the current system:
i. If we set the EWS only with the med-risk alarms, what is the annual profit we will generate vs the current system as a % of the total profit?

ii. And if we set the EWS only with the high-risk alarms?

```
# Assumptions:
margin_per_item = 1.7
cost_of_checking = 10
time_to_fix = 1.5
current_false_alarms = 2.2
```

Function for calculating increase in profit based on risk threshold:

```
profit_increase = function(threshold) {
  if (threshold == threshold_med_risk){
    dt_alarms = transactional[transactional$alarm_med_risk == 1,]
  }
  else {
    dt_alarms = transactional[transactional$alarm_high_risk == 1,]
  }

## Additional costs from false alarms
  # New costs
  new_costs = nrow(dt_alarms[dt_alarms$failure == 0]) * cost_of_checking

  # Current costs
  current_costs = transactional[, .(max = max(timestamp), min = min(timestamp)), by =
machine]
  current_costs$cost = as.numeric(difftime(current_costs$max,
                                           current_costs$min,
                                           units = "days"))/ 365 * current_false_alar
ms * cost_of_checking
  # Total current costs
  current_costs = sum(current_costs$cost)

  # Net costs
  net_costs = new_costs - current_costs
#----------------------

# Additional revenue
  dt_alarms[, threshold_hours := threshold * (24 / daily_sales_machine)]
  dt_alarms[, threshold_hours_fixed := threshold_hours + time_to_fix]
  dt_alarms[, delta_fixed := threshold_hours_fixed * daily_sales_machine / 24 ]

# Additional sales:
  dt_alarms[, won_sales := failure * (delta - delta_fixed)]
  add_margin = sum(dt_alarms$won_sales) * margin_per_item

# Profit increase
  delta_profit = add_margin - net_costs
  current_profit = nrow(transactional) * margin_per_item
  profit_increase = delta_profit/current_profit

  return(profit_increase)
}
```

```
# i) increase Profit of medium risk:
profit_increase_medium_risk = profit_increase(threshold_med_risk)
sprintf("%1.2f%%", 100*profit_increase_medium_risk)
```

```
## [1] "2.39%"
```

```
# ii) Increase Profit of high risk
profit_increase_high_risk = profit_increase(threshold_high_risk)
sprintf("%1.2f%%", 100*profit_increase_high_risk)
```

```
## [1] "2.35%"
```