November 2, 2023

The results below are generated from an R script.

```r
# 1. Lectura y preparación de datos

# Lectura de datos
# Strings como factores
library(readr)
library(Hmisc)
ACMETelephoneABT <- read_csv("./datos/ACMETelephoneABT.csv", na = c("", " "))

## Rows:  10000 Columns:  33
## - Column specification --------------------------------
## Delimiter:  ","
## chr  (5):  occupation, regionType, marriageStatus, creditRating, creditCard
## dbl (24):  customer, age, income, numHandsets, handsetAge, currentHandsetPrice, avgBill...
## lgl  (4):  children, smartPhone, homeOwner, churn
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.

# Corregir NAs y unificar valores en regionType
ACMETelephoneABT$regionType[which(ACMETelephoneABT$regionType == "unknown")] <- NA
ACMETelephoneABT$regionType[which(ACMETelephoneABT$regionType == "r")] <- "RURAL"
ACMETelephoneABT$regionType[which(ACMETelephoneABT$regionType == "s")] <- "SUBURBAN"
ACMETelephoneABT$regionType[which(ACMETelephoneABT$regionType == "t")] <- "TOWN"
ACMETelephoneABT$regionType = factor(ACMETelephoneABT$regionType, levels = c("RURAL", "SUBURBAN", "TOWN"

# Corregir NAs en marriageStatus
ACMETelephoneABT$marriageStatus[which(ACMETelephoneABT$marriageStatus == "unknown")] <- NA
ACMETelephoneABT$marriageStatus = factor(ACMETelephoneABT$marriageStatus, levels = c("YES", "NO"))

# Corregir NAs y unificar valores en creditCard
ACMETelephoneABT$creditCard[which(ACMETelephoneABT$creditCard == "f")] <- "FALSE"
ACMETelephoneABT$creditCard[which(ACMETelephoneABT$creditCard == "no")] <- "FALSE"
ACMETelephoneABT$creditCard[which(ACMETelephoneABT$creditCard == "t")] <- "TRUE"
ACMETelephoneABT$creditCard[which(ACMETelephoneABT$creditCard == "yes")] <- "TRUE"
ACMETelephoneABT$creditCard = factor(ACMETelephoneABT$creditCard, levels = c("TRUE", "FALSE"))

# Asignar NAs a casos con edad = 0
ACMETelephoneABT$age[which(ACMETelephoneABT$age == 0)] <- NA

# Asumimos casos de income = 0 como NAs
ACMETelephoneABT$income[which(ACMETelephoneABT$income == 0)] <- NA

levels(ACMETelephoneABT$creditCard)
```

```
## [1] "TRUE"  "FALSE"

levels(ACMETelephoneABT$regionType)

## [1] "RURAL"    "SUBURBAN" "TOWN"

levels(ACMETelephoneABT$marriageStatus)

## [1] "YES" "NO"

ACMETelephoneABT$churn = ifelse(ACMETelephoneABT$churn == "TRUE", 1, 0)
ACMETelephoneABT$churn = factor(ACMETelephoneABT$churn, levels = c(1,0))
summary(ACMETelephoneABT$churn)

##    1    0
## 5000 5000

levels(ACMETelephoneABT$churn)

## [1] "1" "0"

# 2. División de datos
library(caret)
library(dplyr)

set.seed(12345)
inTraining <- createDataPartition(pull(ACMETelephoneABT, churn),
                                  p = .7, list = FALSE, times = 1)
acme_training <- slice(ACMETelephoneABT, inTraining)
acme_testing <- slice(ACMETelephoneABT, -inTraining)

# 3. Modelo 1. Regresión Logística
min_overbundlemins = min(acme_training$avgOverBundleMins)
min_handsetAge = min(acme_training$handsetAge)
acme_training <- acme_training %>%
  mutate(binary_billAmountChangePct = ifelse(billAmountChangePct > 0, "positive","negative"))
acme_training <- acme_training %>%
  mutate(creditRating_DE = ifelse(creditRating %in% c("D", "E"), "yes","no"))
acme_training$creditRating_DE = as.factor(acme_training$creditRating_DE )

acme_training$creditRating = as.factor(acme_training$creditRating)
acme_training$binary_billAmountChangePct = as.factor(acme_training$binary_billAmountChangePct)
acme_training$homeOwner = as.factor(acme_training$homeOwner)
acme_training$smartPhone = as.factor(acme_training$smartPhone)

glm_model_train = glm(churn ~
                        log(lastMonthCustomerCareCalls + 1) +
                        log(avgrecurringCharge + 1) + log(peakOffPeakRatio + 1) +
                        log(avgBill + 1) + log(avgReceivedMins + 1) +
                        creditRating_DE + binary_billAmountChangePct + smartPhone,
                      data=acme_training, family= binomial)
summary(glm_model_train)

##
## Call:
## glm(formula = churn ~ log(lastMonthCustomerCareCalls + 1) + log(avgrecurringCharge +
```

```
##     1) + log(peakOffPeakRatio + 1) + log(avgBill + 1) + log(avgReceivedMins +
##     1) + creditRating_DE + binary_billAmountChangePct + smartPhone,
##     family = binomial, data = acme_training)
##
## Coefficients:
##                                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)                        -1.03816    0.17953  -5.783 7.36e-09 ***
## log(lastMonthCustomerCareCalls + 1)  0.10947    0.03471   3.154  0.00161 **
## log(avgrecurringCharge + 1)          0.49064    0.06612   7.421 1.16e-13 ***
## log(peakOffPeakRatio + 1)            0.06740    0.04217   1.598  0.10997
## log(avgBill + 1)                   -0.39265    0.06696  -5.864 4.52e-09 ***
## log(avgReceivedMins + 1)             0.02763    0.01592   1.735  0.08273 .
## creditRating_DEyes                   0.24202    0.06106   3.964 7.38e-05 ***
## binary_billAmountChangePctpositive   0.11028    0.05200   2.121  0.03394 *
## smartPhoneTRUE                       0.48571    0.08493   5.719 1.07e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 9704.1  on 6999  degrees of freedom
## Residual deviance: 9542.9  on 6991  degrees of freedom
## AIC: 9560.9
##
## Number of Fisher Scoring iterations: 4
```

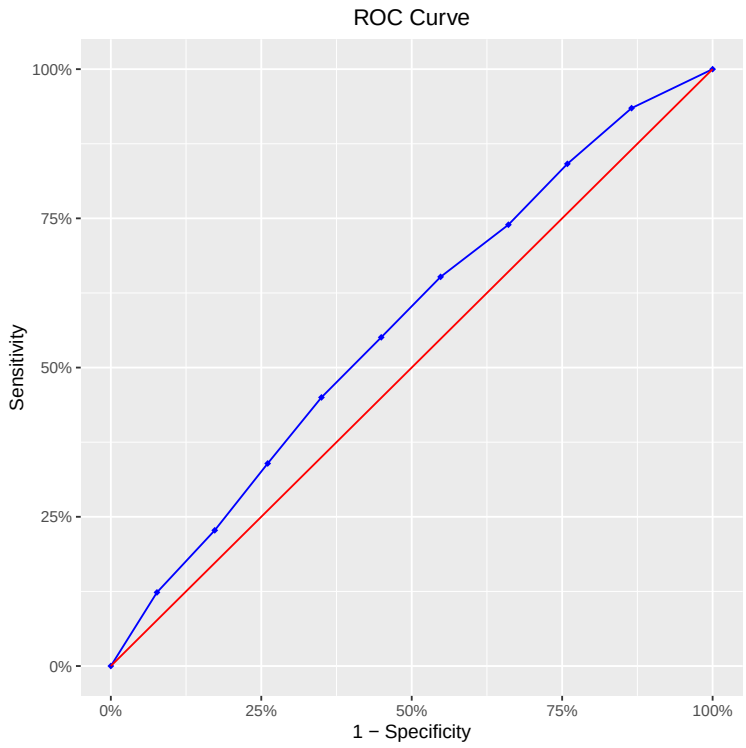
```
# 3.1. Predicción sobre datos de test. Evaluación del modelo
min_overbundlemins = min(acme_testing$avgOverBundleMins)
min_handsetAge = min(acme_testing$handsetAge)
acme_testing <- acme_testing %>%
  mutate(binary_billAmountChangePct = ifelse(billAmountChangePct > 0, "positive","negative"))

acme_testing$income = as.factor(acme_testing$income)
acme_testing$binary_billAmountChangePct = as.factor(acme_testing$binary_billAmountChangePct)
acme_testing$homeOwner = as.factor(acme_testing$homeOwner)
acme_testing$smartPhone = as.factor(acme_testing$smartPhone)
acme_testing <- acme_testing %>%
  mutate(creditRating_DE = ifelse(creditRating %in% c("D", "E"), "yes","no"))
acme_testing$creditRating_DE = as.factor(acme_testing$creditRating_DE )

glm_probs = predict(glm_model_train, newdata = acme_testing, type = "response")

umbral_dec = 0.46
glm_probs <- ifelse(glm_probs >= umbral_dec, 1, 0)
glm_probs <- factor(glm_probs, levels = c(1,0))

tabla_conf <- table(glm_probs, acme_testing$churn)
tabla_conf
```

```
##
## glm_probs    1    0
##         1 1075 1200
##         0  425  300
```

```
caret::confusionMatrix(tabla_conf, positive = '1')
```

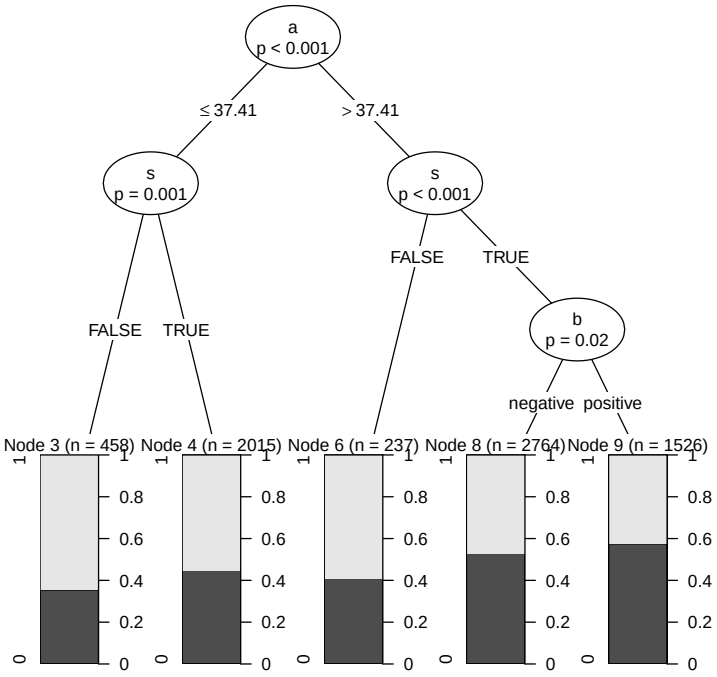
```
## Confusion Matrix and Statistics
##
##
## glm_probs    1    0
##         1 1075 1200
##         0  425  300
##
##                Accuracy : 0.4583
##                  95% CI : (0.4404, 0.4764)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.0833
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7167
##             Specificity : 0.2000
##          Pos Pred Value : 0.4725
##          Neg Pred Value : 0.4138
##              Prevalence : 0.5000
##          Detection Rate : 0.3583
##    Detection Prevalence : 0.7583
##       Balanced Accuracy : 0.4583
##
##        'Positive' Class : 1
##
```

```
# Curva ROC
logistic_gains_table <- blr_gains_table(glm_model_train, data = acme_testing)
blr_roc_curve(logistic_gains_table)
```

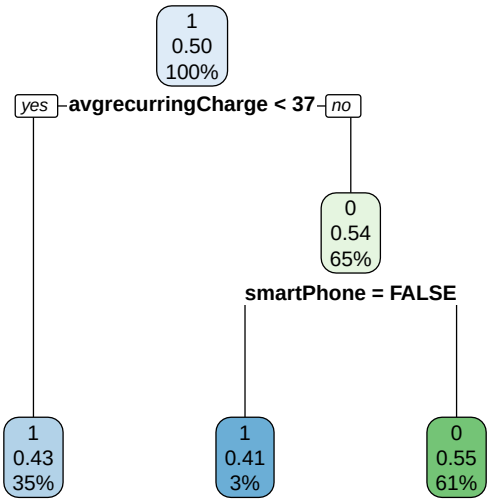



```
# 4. Modelo 2. Árbol de decisión
library(partykit)
ctree_acme = ctree(churn ~ avgOverBundleMins +
                     lastMonthCustomerCareCalls +
                     avgrecurringCharge + peakOffPeakRatio +
                     binary_billAmountChangePct + smartPhone,
                   data=acme_training)
ctree_acme

##
## Model formula:
## churn ~ avgOverBundleMins + lastMonthCustomerCareCalls + avgrecurringCharge +
##     peakOffPeakRatio + binary_billAmountChangePct + smartPhone
##
## Fitted party:
## [1] root
## |   [2] avgrecurringCharge <= 37.41
## |   |   [3] smartPhone in FALSE: 1 (n = 458, err = 35.4%)
## |   |   [4] smartPhone in TRUE: 1 (n = 2015, err = 44.8%)
## |   [5] avgrecurringCharge > 37.41
## |   |   [6] smartPhone in FALSE: 1 (n = 237, err = 40.9%)
## |   |   [7] smartPhone in TRUE
## |   |   |   [8] binary_billAmountChangePct in negative: 0 (n = 2764, err = 47.1%)
## |   |   |   [9] binary_billAmountChangePct in positive: 0 (n = 1526, err = 42.5%)
##
## Number of inner nodes:    4
## Number of terminal nodes: 5

plot(ctree_acme, gp = gpar(fontsize = 10),
     inner_panel=node_inner,
```

```
    ip_args=list(
      abbreviate = TRUE,
      id = FALSE)
)
```



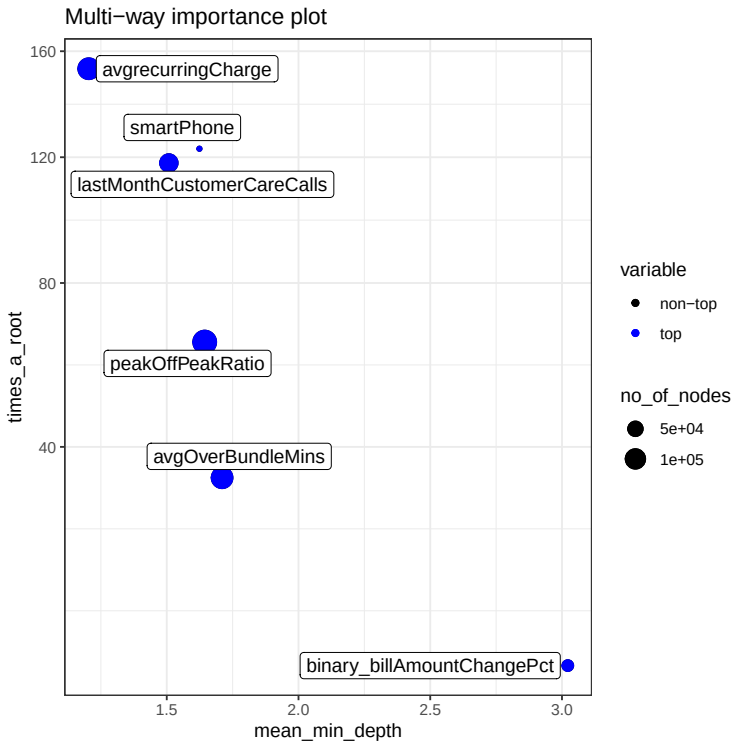

```
library(rpart)
library(rpart.plot)
rpart_acme = rpart(churn ~ avgOverBundleMins +
                     lastMonthCustomerCareCalls +
                     avgrecurringCharge + peakOffPeakRatio +
                     binary_billAmountChangePct + smartPhone,
                   data=acme_training)
rpart_acme

## n= 7000
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 7000 3500 1 (0.5000000 0.5000000)
##   2) avgrecurringCharge< 37.43 2473 1064 1 (0.5697533 0.4302467) *
##   3) avgrecurringCharge>=37.43 4527 2091 0 (0.4618953 0.5381047)
##     6) smartPhone=FALSE 237   97 1 (0.5907173 0.4092827) *
##     7) smartPhone=TRUE 4290 1951 0 (0.4547786 0.5452214) *

rpart.plot(rpart_acme)
```




```
# 4.2 Predicción sobre datos de test
ctree_pred <- predict(ctree_acme, newdata=acme_testing, type='response')
confusionMatrix(ctree_pred, acme_testing$churn)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   0
##          1 629 522
##          0 871 978
##
##                Accuracy : 0.5357
##                  95% CI : (0.5176, 0.5536)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 5.005e-05
##
##                   Kappa : 0.0713
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4193
##             Specificity : 0.6520
##          Pos Pred Value : 0.5465
##          Neg Pred Value : 0.5289
##              Prevalence : 0.5000
##          Detection Rate : 0.2097
##    Detection Prevalence : 0.3837
##       Balanced Accuracy : 0.5357
##
##        'Positive' Class : 1
```

```
##

rpart_pred <- predict(rpart_acme, newdata=acme_testing, type='class')
confusionMatrix(rpart_pred, acme_testing$churn)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   0
##          1 629 522
##          0 871 978
##
##                Accuracy : 0.5357
##                  95% CI : (0.5176, 0.5536)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 5.005e-05
##
##                   Kappa : 0.0713
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4193
##             Specificity : 0.6520
##          Pos Pred Value : 0.5465
##          Neg Pred Value : 0.5289
##              Prevalence : 0.5000
##          Detection Rate : 0.2097
##    Detection Prevalence : 0.3837
##       Balanced Accuracy : 0.5357
##
##        'Positive' Class : 1
##

# 5. Modelo 3: Random Forest
library(randomForest)

forest_acme = randomForest(churn ~ avgOverBundleMins +
                             lastMonthCustomerCareCalls +
                             avgrecurringCharge + peakOffPeakRatio +
                             binary_billAmountChangePct + smartPhone,
                           data=acme_training)
forest_acme

##
## Call:
##  randomForest(formula = churn ~ avgOverBundleMins + lastMonthCustomerCareCalls +      avgrecurringCha
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 45.7%
## Confusion matrix:
##      1    0 class.error
## 1 1797 1703   0.4865714
## 0 1496 2004   0.4274286
```

```
library(randomForestExplainer)
importance_frame <- measure_importance(forest_acme)

## [1] "Warning: your forest does not contain information on local importance so 'accuracy_decrease' mea

save(importance_frame, file = "importance_frame.rda")
load("importance_frame.rda")
plot_multi_way_importance(importance_frame, size_measure = "no_of_nodes")
```



The R session information (including the OS info, R version and all packages used):

```
sessionInfo()

## R version 4.3.1 (2023-06-16)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.6 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/atlas/libblas.so.3.10.3
## LAPACK: /usr/lib/x86_64-linux-gnu/atlas/liblapack.so.3.10.3;  LAPACK version 3.9.0
##
## locale:
##  [1] LC_CTYPE=es_ES.UTF-8       LC_NUMERIC=C               LC_TIME=es_ES.UTF-8
##  [4] LC_COLLATE=es_ES.UTF-8     LC_MONETARY=es_ES.UTF-8    LC_MESSAGES=es_ES.UTF-8
##  [7] LC_PAPER=es_ES.UTF-8       LC_NAME=C                  LC_ADDRESS=C
## [10] LC_TELEPHONE=C             LC_MEASUREMENT=es_ES.UTF-8 LC_IDENTIFICATION=C
##
## time zone: Europe/Madrid
## tzcode source: system (glibc)
##
## attached base packages:
```

```
## [1] grid      stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] randomForestExplainer_0.10.1 partykit_1.2-20
##  [3] mvtnorm_1.2-3                libcoin_1.0-10
##  [5] blorr_0.3.0                  Hmisc_5.1-1
##  [7] readr_2.1.4                  caretEnsemble_2.0.3
##  [9] DALEX_2.4.3                  ROCR_1.0-11
## [11] randomForest_4.7-1.1         arulesViz_1.5-2
## [13] arules_1.7-6                 Matrix_1.6-1.1
## [15] liver_1.15                   ggfortify_0.4.16
## [17] factoextra_1.0.7             mlbench_2.1-3.1
## [19] readxl_1.4.3                 caret_6.0-94
## [21] lattice_0.21-9               ggplot2_3.4.3
## [23] rpart.plot_3.1.1             rpart_4.1.19
## [25] caTools_1.18.2               dplyr_1.1.3
## [27] ISLR2_1.3-2
##
## loaded via a namespace (and not attached):
##   [1] RColorBrewer_1.1-3   rstudioapi_0.15.0    jsonlite_1.8.7       magrittr_2.0.3
##   [5] farver_2.1.1         rmarkdown_2.25       vctrs_0.6.3          base64enc_0.1-3
##   [9] iBreakDown_2.0.1     tinytex_0.47         htmltools_0.5.6.1    cellranger_1.1.0
##  [13] Formula_1.2-5        pROC_1.18.4          parallelly_1.36.0    htmlwidgets_1.6.2
##  [17] plyr_1.8.9           lubridate_1.9.3      igraph_1.5.1         lifecycle_1.0.3
##  [21] iterators_1.0.14     pkgconfig_2.0.3      R6_2.5.1             fastmap_1.1.1
##  [25] future_1.33.0        digest_0.6.33        reshape_0.8.9        GGally_2.1.2
##  [29] colorspace_2.1-0     labeling_0.4.3       fansi_1.0.5          timechange_0.2.0
##  [33] abind_1.4-5          polyclip_1.10-6      compiler_4.3.1       proxy_0.4-27
##  [37] bit64_4.0.5          withr_2.5.1          htmlTable_2.4.1      backports_1.4.1
##  [41] carData_3.0-5        viridis_0.6.4        highr_0.10           ggforce_0.4.1
##  [45] MASS_7.3-60          lava_1.7.2.1         ModelMetrics_1.2.2.2 tools_4.3.1
##  [49] foreign_0.8-85       future.apply_1.11.0  nnet_7.3-19          glue_1.6.2
##  [53] inum_1.0-5           nlme_3.1-163         checkmate_2.2.0      cluster_2.1.4
##  [57] reshape2_1.4.4       generics_0.1.3       recipes_1.0.8        gtable_0.3.4
##  [61] tzdb_0.4.0           class_7.3-22         tidyr_1.3.0          data.table_1.14.8
##  [65] hms_1.1.3            car_3.1-2            tidygraph_1.2.3      utf8_1.2.3
##  [69] ggrepel_0.9.3        foreach_1.5.2        pillar_1.9.0         stringr_1.5.0
##  [73] vroom_1.6.4          splines_4.3.1        tweenr_2.0.2         survival_3.5-7
##  [77] bit_4.0.5            tidyselect_1.2.0     pbapply_1.7-2        knitr_1.44
##  [81] gridExtra_2.3        stats4_4.3.1         xfun_0.40            graphlayouts_1.0.1
##  [85] hardhat_1.3.0        timeDate_4022.108    DT_0.30              visNetwork_2.1.2
##  [89] stringi_1.7.12       yaml_2.3.7           evaluate_0.22        codetools_0.2-19
##  [93] ggraph_2.1.0         tibble_3.2.1         cli_3.6.1            munsell_0.5.0
##  [97] Rcpp_1.0.11          globals_0.16.2       parallel_4.3.1       ellipsis_0.3.2
## [101] gower_1.0.1          bitops_1.0-7         listenv_0.9.0        viridisLite_0.4.2
## [105] ipred_0.9-14         scales_1.2.1         prodlim_2023.08.28   e1071_1.7-13
## [109] purrr_1.0.2          crayon_1.5.2         rlang_1.1.1
```

```
Sys.time()
```

```
## [1] "2023-11-02 21:54:13 CET"
```