# R Notebook

## Parametros:

**Measure =** G-mean
**Columns =** sampling, weight_space, underbagging, learner
**Performance =** holdout_measure_residual
**Filter keys =** imba.rate
**Filter values =** 0.001

```
library("scmamp")
library(dplyr)
```

## Tratamento dos dados

Carregando data set compilado

```
ds = read.csv("/home/rodrigo/Dropbox/UNICAMP/IC/estudo_cost_learning/SummaryResults/summary_compilation_

ds = filter(ds, learner != "classif.rusboost")
summary(ds)
```

```
##                   learner      weight_space  
##  classif.ksvm        :17100   Mode :logical  
##  classif.randomForest:17100   FALSE:41040    
##  classif.rusboost    :    0   TRUE :10260    
##  classif.xgboost     :17100   NA's :0        
##                                              
##                                              
##                                              
##                               measure        sampling     underbagging   
##  Accuracy                        :10260   ADASYN:10260   Mode :logical  
##  Area under the curve            :10260   FALSE :30780   FALSE:41040    
##  F1 measure                      :10260   SMOTE :10260   TRUE :10260    
##  G-mean                          :10260                  NA's :0        
##  Matthews correlation coefficient:10260                                 
##                                                                         
##                                                                         
##  tuning_measure    holdout_measure   holdout_measure_residual
##  Min.   :-0.1277   Min.   :-0.2120   Min.   :-0.4658         
##  1st Qu.: 0.6911   1st Qu.: 0.4001   1st Qu.: 0.1994         
##  Median : 0.9700   Median : 0.8571   Median : 0.5581         
##  Mean   : 0.7903   Mean   : 0.6718   Mean   : 0.5298         
##  3rd Qu.: 0.9975   3rd Qu.: 0.9900   3rd Qu.: 0.8755         
##  Max.   : 1.0000   Max.   : 1.0000   Max.   : 1.0000         
##  NA's   :1077      NA's   :1077      NA's   :1077            
##  iteration_count                      dataset        imba.rate     
##  Min.   :1       abalone                  :  900   Min.   :0.0010  
##  1st Qu.:1       adult                    :  900   1st Qu.:0.0100  
##  Median :2       bank                     :  900   Median :0.0300  
##  Mean   :2       car                      :  900   Mean   :0.0286  
##  3rd Qu.:3       cardiotocography-10clases:  900   3rd Qu.:0.0500
##  Max.   :3       cardiotocography-3clases :  900   Max.   :0.0500
##  NA's   :1077    (Other)                  :45900
```

Filtrando pela metrica

```
ds = filter(ds, measure == params$measure)
```

Filtrando o data set

```
if(params$filter_keys != 'NULL' && !is.null(params$filter_keys)){
  dots = paste0(params$filter_keys," == '",params$filter_values,"'")
  ds = filter_(ds, .dots = dots)
}

summary(ds)
```

```
##                    learner    weight_space
##  classif.ksvm        :600   Mode :logical
##  classif.randomForest:600   FALSE:1440
##  classif.rusboost    :  0   TRUE :360
##  classif.xgboost     :600   NA's :0
##
##
##
##                             measure       sampling    underbagging
##  Accuracy                        :   0   ADASYN: 360   Mode :logical
##  Area under the curve            :   0   FALSE :1080   FALSE:1440
##  F1 measure                      :   0   SMOTE : 360   TRUE :360
##  G-mean                          :1800                 NA's :0
##  Matthews correlation coefficient:   0
##
##
##  tuning_measure   holdout_measure  holdout_measure_residual
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.5941   1st Qu.:0.0000   1st Qu.:0.1173
##  Median :0.9638   Median :0.7062   Median :0.4257
##  Mean   :0.7528   Mean   :0.5598   Mean   :0.4404
##  3rd Qu.:0.9988   3rd Qu.:0.9645   3rd Qu.:0.7589
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##  NA's   :48       NA's   :48       NA's   :48
##  iteration_count                      dataset       imba.rate
##  Min.   :1       abalone                  :  45   Min.   :0.001
##  1st Qu.:1       adult                    :  45   1st Qu.:0.001
##  Median :2       bank                     :  45   Median :0.001
##  Mean   :2       car                      :  45   Mean   :0.001
##  3rd Qu.:3       cardiotocography-10clases:  45   3rd Qu.:0.001
##  Max.   :3       cardiotocography-3clases :  45   Max.   :0.001
##  NA's   :48      (Other)                  :1530
```

Computando as médias das iteracoes

```
ds = group_by(ds, learner , weight_space , measure , sampling , underbagging , dataset , imba.rate)
ds = summarise(ds, tuning_measure = mean(tuning_measure), holdout_measure = mean(holdout_measure),
               holdout_measure_residual = mean(holdout_measure_residual))

ds = as.data.frame(ds)
```

Criando dataframe

```r
# Dividindo o ds em n, um para cada técnica
splited_df = ds %>% group_by_at(.vars = params$columns) %>% do(vals = as.data.frame(.)) %>% select(vals

# Juntando cada uma das partes horizontalmente em um data set
df_tec_wide = do.call("cbind", splited_df)

# Renomeando duplicacao de nomes
colnames(df_tec_wide) = make.unique(colnames(df_tec_wide))

# Selecionando apenas as medidas da performance escolhida
df_tec_wide_residual = select(df_tec_wide, matches(paste("^", params$performance, "$|", params$performan

# Renomeando colunas
new_names = NULL
for(i in (1:length(splited_df))){
  id = toString(sapply(splited_df[[i]][1, params$columns], as.character))
  new_names = c(new_names, id)
}
colnames(df_tec_wide_residual) = new_names

# Verificando a dimensao do df
dim(df_tec_wide_residual)
```

```
## [1] 40 15
```

```r
# Renomeando a variavel
df = df_tec_wide_residual

summary(df)
```

```
##  ADASYN, FALSE, FALSE, classif.ksvm
##  Min.   :0.00000
##  1st Qu.:0.00000
##  Median :0.07322
##  Mean   :0.19518
##  3rd Qu.:0.27125
##  Max.   :0.90232
##  NA's   :2
##  ADASYN, FALSE, FALSE, classif.randomForest
##  Min.   :0.0000
##  1st Qu.:0.1926
##  Median :0.3910
##  Mean   :0.4566
##  3rd Qu.:0.6744
##  Max.   :0.9999
##  NA's   :6
##  ADASYN, FALSE, FALSE, classif.xgboost FALSE, FALSE, FALSE, classif.ksvm
##  Min.   :0.08138                       Min.   :0.0000
##  1st Qu.:0.28060                       1st Qu.:0.0000
##  Median :0.54412                       Median :0.1947
##  Mean   :0.54351                       Mean   :0.2570
##  3rd Qu.:0.78987                       3rd Qu.:0.3317
##  Max.   :0.99993                       Max.   :0.8910
##
```

```
##  FALSE, FALSE, FALSE, classif.randomForest
##  Min.   :0.0000
##  1st Qu.:0.1137
##  Median :0.3131
##  Mean   :0.3840
##  3rd Qu.:0.5923
##  Max.   :0.9999
##  NA's   :2
##  FALSE, FALSE, FALSE, classif.xgboost FALSE, FALSE, TRUE, classif.ksvm
##  Min.   :0.0000                       Min.   :0.03782
##  1st Qu.:0.1438                       1st Qu.:0.47417
##  Median :0.3688                       Median :0.61368
##  Mean   :0.4181                       Mean   :0.59366
##  3rd Qu.:0.6290                       3rd Qu.:0.78184
##  Max.   :0.9999                       Max.   :0.93922
##
##  FALSE, FALSE, TRUE, classif.randomForest
##  Min.   :0.1064
##  1st Qu.:0.6769
##  Median :0.8175
##  Mean   :0.7643
##  3rd Qu.:0.9401
##  Max.   :0.9999
##  NA's   :1
##  FALSE, FALSE, TRUE, classif.xgboost FALSE, TRUE, FALSE, classif.ksvm
##  Min.   :0.1499                      Min.   :0.0000
##  1st Qu.:0.6159                      1st Qu.:0.0000
##  Median :0.8175                      Median :0.1832
##  Mean   :0.7600                      Mean   :0.2517
##  3rd Qu.:0.9319                      3rd Qu.:0.3317
##  Max.   :0.9999                      Max.   :0.8910
##
##  FALSE, TRUE, FALSE, classif.randomForest
##  Min.   :0.0000
##  1st Qu.:0.1106
##  Median :0.3118
##  Mean   :0.3752
##  3rd Qu.:0.6154
##  Max.   :0.9999
##  NA's   :2
##  FALSE, TRUE, FALSE, classif.xgboost SMOTE, FALSE, FALSE, classif.ksvm
##  Min.   :0.0000                      Min.   :0.00000
##  1st Qu.:0.1429                      1st Qu.:0.01074
##  Median :0.3658                      Median :0.12383
##  Mean   :0.4201                      Mean   :0.20003
##  3rd Qu.:0.6168                      3rd Qu.:0.26149
##  Max.   :0.9999                      Max.   :0.90233
##
##  SMOTE, FALSE, FALSE, classif.randomForest
##  Min.   :0.0000
##  1st Qu.:0.1715
##  Median :0.3643
##  Mean   :0.4437
##  3rd Qu.:0.6907
##  Max.   :1.0000
##  NA's   :3
##  SMOTE, FALSE, FALSE, classif.xgboost
##  Min.   :0.0000
##  1st Qu.:0.2602
##  Median :0.5707
##  Mean   :0.5350
##  3rd Qu.:0.8230
##  Max.   :0.9999
##
```

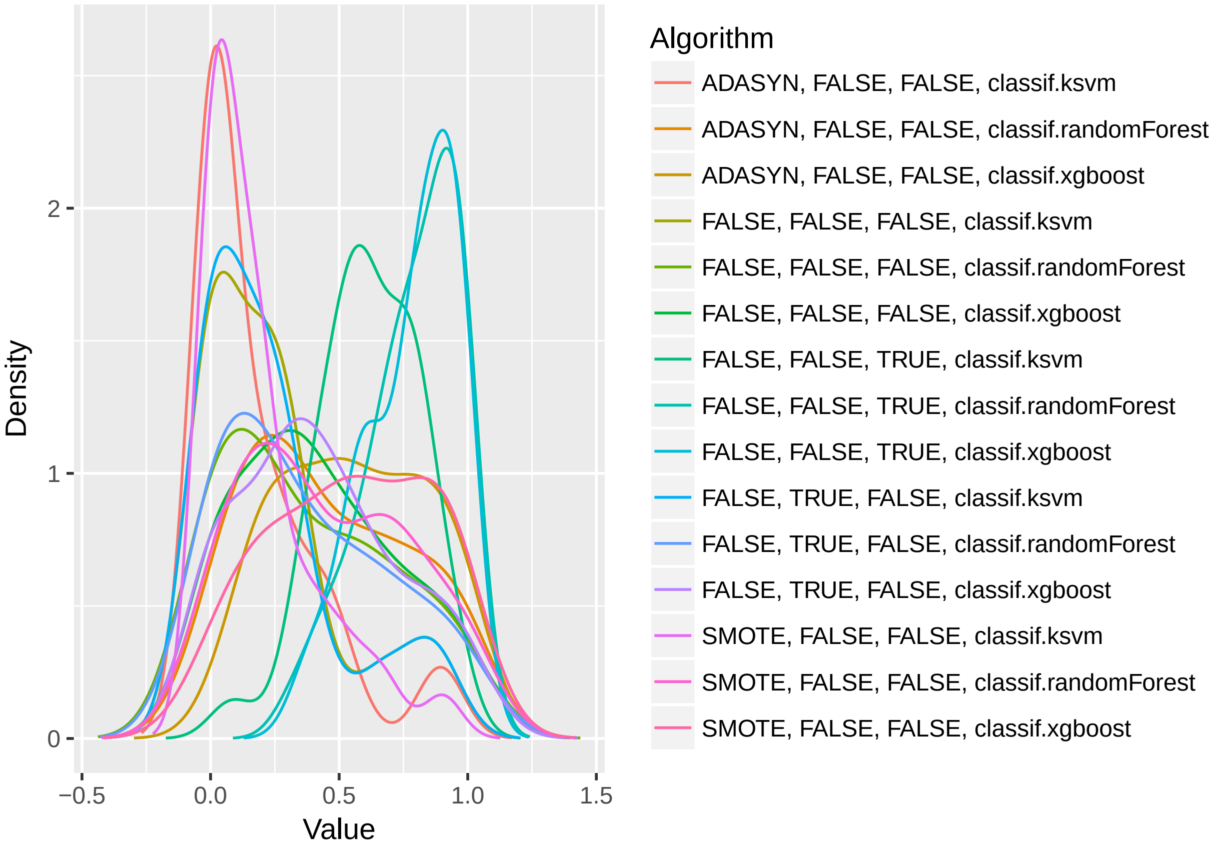
# Verificando a média de cada coluna selecionada

```
for(i in (1:dim(df)[2])){
  print(paste("Media da coluna ", colnames(df)[i], " = ", mean(df[,i], na.rm = TRUE), sep=""))
}
```

```
## [1] "Media da coluna ADASYN, FALSE, FALSE, classif.ksvm = 0.19518305573556"
## [1] "Media da coluna ADASYN, FALSE, FALSE, classif.randomForest = 0.456631299796649"
## [1] "Media da coluna ADASYN, FALSE, FALSE, classif.xgboost = 0.543509498678387"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.ksvm = 0.256956550225041"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.randomForest = 0.383976132773795"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.xgboost = 0.418074011113785"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.ksvm = 0.593658779887989"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.randomForest = 0.764260225319559"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.xgboost = 0.759991987108657"
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.ksvm = 0.251657162506451"
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.randomForest = 0.375198267533911"
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.xgboost = 0.420051794322392"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.ksvm = 0.200029031857121"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.randomForest = 0.443736015453979"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.xgboost = 0.535042975739401"
```

# Fazendo teste de normalidade

```
plotDensities(data = na.omit(df))
```



## Testando as diferencas

```
friedmanTest(df)
```

```
##
##  Friedman's rank sum test
##
## data:  df
## Friedman's chi-squared = 300.44, df = 14, p-value < 2.2e-16
```

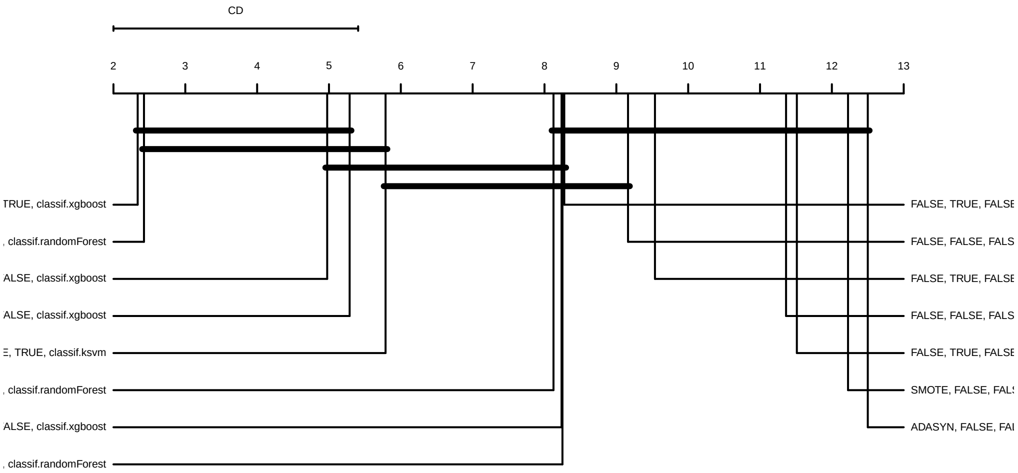
## Testando as diferencas par a par

```
test <- nemenyiTest (df, alpha=0.05)
abs(test$diff.matrix) > test$statistic
```

```
##       ADASYN, FALSE, FALSE, classif.ksvm
##  [1,]                              FALSE
##  [2,]                               TRUE
##  [3,]                               TRUE
##  [4,]                              FALSE
##  [5,]                              FALSE
##  [6,]                               TRUE
##  [7,]                               TRUE
##  [8,]                               TRUE
##  [9,]                               TRUE
## [10,]                              FALSE
## [11,]                              FALSE
## [12,]                               TRUE
## [13,]                              FALSE
## [14,]                               TRUE
## [15,]                               TRUE
##       ADASYN, FALSE, FALSE, classif.randomForest
##  [1,]                                       TRUE
##  [2,]                                      FALSE
##  [3,]                                      FALSE
##  [4,]                                      FALSE
##  [5,]                                      FALSE
##  [6,]                                      FALSE
##  [7,]                                      FALSE
##  [8,]                                       TRUE
##  [9,]                                       TRUE
## [10,]                                      FALSE
## [11,]                                      FALSE
## [12,]                                      FALSE
## [13,]                                       TRUE
## [14,]                                      FALSE
## [15,]                                      FALSE
##       ADASYN, FALSE, FALSE, classif.xgboost
##  [1,]                                  TRUE
##  [2,]                                 FALSE
##  [3,]                                 FALSE
##  [4,]                                  TRUE
##  [5,]                                  TRUE
##  [6,]                                 FALSE
##  [7,]                                 FALSE
##  [8,]                                 FALSE
##  [9,]                                 FALSE
## [10,]                                  TRUE
## [11,]                                  TRUE
## [12,]                                 FALSE
## [13,]                                  TRUE
## [14,]                                 FALSE
## [15,]                                 FALSE
##       FALSE, FALSE, FALSE, classif.ksvm
##  [1,]                             FALSE
##  [2,]                             FALSE
##  [3,]                              TRUE
##  [4,]                             FALSE
##  [5,]                             FALSE
##  [6,]                             FALSE
##  [7,]                              TRUE
##  [8,]                              TRUE
##  [9,]                              TRUE
## [10,]                             FALSE
## [11,]                             FALSE
## [12,]                             FALSE
## [13,]                             FALSE
## [14,]                             FALSE
## [15,]                              TRUE
##        FALSE, FALSE, FALSE, classif.randomForest
##  [1,]                                     FALSE
##  [2,]                                     FALSE
##  [3,]                                      TRUE
##  [4,]                                     FALSE
##  [5,]                                     FALSE
##  [6,]                                     FALSE
##  [7,]                                     FALSE
##  [8,]                                      TRUE
##  [9,]                                      TRUE
## [10,]                                     FALSE
## [11,]                                     FALSE
## [12,]                                     FALSE
## [13,]                                     FALSE
## [14,]                                     FALSE
## [15,]                                      TRUE
##        FALSE, FALSE, FALSE, classif.xgboost
##  [1,]                                 TRUE
##  [2,]                                FALSE
##  [3,]                                FALSE
##  [4,]                                FALSE
##  [5,]                                FALSE
##  [6,]                                FALSE
##  [7,]                                FALSE
##  [8,]                                 TRUE
##  [9,]                                 TRUE
## [10,]                                FALSE
## [11,]                                FALSE
## [12,]                                FALSE
## [13,]                                 TRUE
## [14,]                                FALSE
## [15,]                                FALSE
##        FALSE, FALSE, TRUE, classif.ksvm
##  [1,]                             TRUE
##  [2,]                            FALSE
##  [3,]                            FALSE
##  [4,]                             TRUE
##  [5,]                            FALSE
##  [6,]                            FALSE
##  [7,]                            FALSE
##  [8,]                            FALSE
##  [9,]                             TRUE
## [10,]                             TRUE
## [11,]                             TRUE
## [12,]                            FALSE
## [13,]                             TRUE
## [14,]                            FALSE
## [15,]                            FALSE
##        FALSE, FALSE, TRUE, classif.randomForest
##  [1,]                                     TRUE
##  [2,]                                     TRUE
##  [3,]                                    FALSE
##  [4,]                                     TRUE
##  [5,]                                     TRUE
##  [6,]                                     TRUE
##  [7,]                                    FALSE
##  [8,]                                    FALSE
##  [9,]                                    FALSE
## [10,]                                     TRUE
## [11,]                                     TRUE
## [12,]                                     TRUE
## [13,]                                     TRUE
## [14,]                                     TRUE
## [15,]                                    FALSE
##       FALSE, FALSE, TRUE, classif.xgboost FALSE, TRUE, FALSE, classif.ksvm
##  [1,]                                TRUE                            FALSE
##  [2,]                                TRUE                            FALSE
##  [3,]                               FALSE                             TRUE
##  [4,]                                TRUE                            FALSE
##  [5,]                                TRUE                            FALSE
##  [6,]                                TRUE                            FALSE
##  [7,]                                TRUE                             TRUE
##  [8,]                               FALSE                             TRUE
##  [9,]                               FALSE                             TRUE
## [10,]                                TRUE                            FALSE
## [11,]                                TRUE                            FALSE
## [12,]                                TRUE                            FALSE
## [13,]                                TRUE                            FALSE
## [14,]                                TRUE                            FALSE
## [15,]                               FALSE                             TRUE
##       FALSE, TRUE, FALSE, classif.randomForest
##  [1,]                                    FALSE
##  [2,]                                    FALSE
##  [3,]                                     TRUE
##  [4,]                                    FALSE
##  [5,]                                    FALSE
##  [6,]                                    FALSE
##  [7,]                                     TRUE
##  [8,]                                     TRUE
##  [9,]                                     TRUE
## [10,]                                    FALSE
## [11,]                                    FALSE
## [12,]                                    FALSE
## [13,]                                    FALSE
## [14,]                                    FALSE
## [15,]                                     TRUE
##       FALSE, TRUE, FALSE, classif.xgboost
##  [1,]                                TRUE
##  [2,]                               FALSE
##  [3,]                               FALSE
##  [4,]                               FALSE
##  [5,]                               FALSE
##  [6,]                               FALSE
##  [7,]                               FALSE
##  [8,]                                TRUE
##  [9,]                                TRUE
## [10,]                               FALSE
## [11,]                               FALSE
## [12,]                               FALSE
## [13,]                                TRUE
## [14,]                               FALSE
## [15,]                               FALSE
##       SMOTE, FALSE, FALSE, classif.ksvm
##  [1,]                             FALSE
##  [2,]                              TRUE
##  [3,]                              TRUE
##  [4,]                             FALSE
##  [5,]                             FALSE
##  [6,]                              TRUE
##  [7,]                              TRUE
##  [8,]                              TRUE
##  [9,]                              TRUE
## [10,]                             FALSE
## [11,]                             FALSE
## [12,]                              TRUE
## [13,]                             FALSE
## [14,]                              TRUE
## [15,]                              TRUE
##       SMOTE, FALSE, FALSE, classif.randomForest
##  [1,]                                      TRUE
##  [2,]                                     FALSE
##  [3,]                                     FALSE
##  [4,]                                     FALSE
##  [5,]                                     FALSE
##  [6,]                                     FALSE
##  [7,]                                     FALSE
##  [8,]                                      TRUE
##  [9,]                                      TRUE
## [10,]                                     FALSE
## [11,]                                     FALSE
## [12,]                                     FALSE
## [13,]                                      TRUE
## [14,]                                     FALSE
## [15,]                                     FALSE
##       SMOTE, FALSE, FALSE, classif.xgboost
##  [1,]                                 TRUE
##  [2,]                                FALSE
##  [3,]                                FALSE
##  [4,]                                 TRUE
##  [5,]                                 TRUE
##  [6,]                                FALSE
##  [7,]                                FALSE
##  [8,]                                FALSE
##  [9,]                                FALSE
## [10,]                                 TRUE
## [11,]                                 TRUE
## [12,]                                FALSE
## [13,]                                 TRUE
## [14,]                                FALSE
## [15,]                                FALSE
```

# Plotando grafico de Critical Diference

```r
result = tryCatch({
    plotCD(df, alpha=0.05, cex = 0.35)
}, error = function(e) {})
```


CD
TRUE, classif.xgboost
, classif.randomForest
ALSE, classif.xgboost
ALSE, classif.xgboost
E, TRUE, classif.ksvm
, classif.randomForest
ALSE, classif.xgboost
, classif.randomForest
FALSE, TRUE, FALSE
FALSE, FALSE, FALS
FALSE, TRUE, FALSE
FALSE, FALSE, FALS
FALSE, TRUE, FALSE
SMOTE, FALSE, FAL
ADASYN, FALSE, FAI