

R Notebook

Parametros:

Measure = Matthews correlation coefficient
Columns = sampling, weight_space, underbagging, learner
Performance = holdout_measure_residual
Filter keys = imba.rate
Filter values = 0.03

```
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      cardiocography-10clases: 900 3rd Qu.:0.0500
## Max. :3      cardiocography-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      :990  Mode :logical
## classif.randomForest:990 FALSE:2376
## classif.rusboost   : 0  TRUE :594
## classif.xgboost    :990  NA's :0
##
##
##
##           measure      sampling      underbagging
## Accuracy           : 0  ADASYN: 594  Mode :logical
## Area under the curve : 0  FALSE :1782 FALSE:2376
## F1 measure          : 0  SMOTE : 594  TRUE :594
## G-mean              : 0              NA's :0
## Matthews correlation coefficient:2970
##
##
## tuning_measure      holdout_measure      holdout_measure_residual
## Min.      :-0.05673  Min.      :-0.1757  Min.      :-0.4658
## 1st Qu.: 0.33347    1st Qu.: 0.0000  1st Qu.: 0.0391
## Median : 0.83196    Median : 0.5030  Median : 0.2116
## Mean      : 0.66187    Mean      : 0.4753  Mean      : 0.3111
## 3rd Qu.: 0.98596    3rd Qu.: 0.8126  3rd Qu.: 0.5286
## Max.      : 1.00000    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.      :0.03
## 1st Qu.:1          adult       : 45  1st Qu.:0.03
## Median :2          annealing    : 45  Median :0.03
## Mean      :2          arrhythmia   : 45  Mean      :0.03
## 3rd Qu.:3          balance-scale: 45  3rd Qu.:0.03
## Max.      :3          bank         : 45  Max.      :0.03
## NA's      :48        (Other)      :2700
```

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

```
# 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$performance)))

# 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] 66 15
```

```
# Renomeando a variavel
df = df_tec_wide_residual

head(df)
```

```
## ADASYN, FALSE, FALSE, classif.ksvm
## 1 0.10164023
## 2 0.19876940
## 3 0.40497948
## 4 0.00000000
## 5 0.02921214
## 6 0.19270119
## ADASYN, FALSE, FALSE, classif.randomForest
## 1 0.114890993
## 2 0.343344986
## 3 0.658916689
## 4 -0.004599123
## 5 0.068520072
## 6 0.230395606
## ADASYN, FALSE, FALSE, classif.xgboost FALSE, FALSE, FALSE, classif.ksvm
## 1 0.07341364 0.05529460
## 2 0.37951623 0.21298883
## 3 0.52656367 0.21324335
## 4 0.63677523 0.00000000
## 5 0.09971571 0.07742168
## 6 0.24953590 0.03690155
## FALSE, FALSE, FALSE, classif.randomForest
## 1 0.00000000
## 2 0.35680785
```

```

## 3          0.64555550
## 4          0.66666667
## 5          0.10146479
## 6          0.08408873
##  FALSE, FALSE, FALSE, classif.xgboost FALSE, FALSE, TRUE, classif.ksvm
## 1          0.02154409          0.24865988
## 2          0.34869882          0.51275491
## 3          0.50996235          0.46011997
## 4          0.26622897          0.05829541
## 5          0.08528029          0.03390066
## 6          0.17036951          0.31011260
##  FALSE, FALSE, TRUE, classif.randomForest
## 1          0.2697597
## 2          0.6337675
## 3          0.7841658
## 4          0.3915370
## 5          0.1302082
## 6          0.5841238
##  FALSE, FALSE, TRUE, classif.xgboost FALSE, TRUE, FALSE, classif.ksvm
## 1          0.2477941          0.05469679
## 2          0.6394134          0.22354712
## 3          0.5869880          0.21324335
## 4          0.4971331          0.00000000
## 5          0.1097417          0.07742168
## 6          0.5456770          0.03690155
##  FALSE, TRUE, FALSE, classif.randomForest
## 1          0.005292689
## 2          0.356913154
## 3          0.645692484
## 4          0.233380014
## 5          0.090026560
## 6          0.101487369
##  FALSE, TRUE, FALSE, classif.xgboost SMOTE, FALSE, FALSE, classif.ksvm
## 1          0.005382185          0.07731173
## 2          0.344829690          0.20595422
## 3          0.498132564          0.35644982
## 4          0.234555284          0.00000000
## 5          0.085280287          0.10463026
## 6          0.131656113          0.07178468
##  SMOTE, FALSE, FALSE, classif.randomForest
## 1          0.120423160
## 2          0.367090343
## 3          0.653373236
## 4         -0.009198246
## 5          0.086945510
## 6          0.248187787
##  SMOTE, FALSE, FALSE, classif.xgboost
## 1          0.08605067
## 2          0.37503965
## 3          0.55474001
## 4          0.26737670
## 5          0.09971571
## 6          0.24197625

```

```
summary(df)
```

```
## ADASYN, FALSE, FALSE, classif.ksvm
## Min.      :-0.20862
## 1st Qu.: 0.01075
## Median : 0.10248
## Mean     : 0.18704
## 3rd Qu.: 0.28810
## Max.     : 0.93063
## NA's     :2
## ADASYN, FALSE, FALSE, classif.randomForest
## Min.      :-0.1721
## 1st Qu.: 0.1019
## Median : 0.2486
## Mean     : 0.3317
## 3rd Qu.: 0.5080
## Max.     : 0.9458
## NA's     :4
## ADASYN, FALSE, FALSE, classif.xgboost FALSE, FALSE, FALSE, classif.ksvm
## Min.      :-0.3218                      Min.      :-0.044918
## 1st Qu.: 0.1083                      1st Qu.: 0.001888
## Median : 0.3472                      Median : 0.089378
## Mean     : 0.3852                      Mean     : 0.191604
## 3rd Qu.: 0.5832                      3rd Qu.: 0.256238
## Max.     : 0.9974                      Max.     : 0.975899
##
## FALSE, FALSE, FALSE, classif.randomForest
## Min.      :-0.2333
## 1st Qu.: 0.0536
## Median : 0.2190
## Mean     : 0.3194
## 3rd Qu.: 0.5030
## Max.     : 1.0000
## NA's     :2
## FALSE, FALSE, FALSE, classif.xgboost FALSE, FALSE, TRUE, classif.ksvm
## Min.      :-0.3169                      Min.      :-0.1825
## 1st Qu.: 0.0739                      1st Qu.: 0.1113
## Median : 0.3356                      Median : 0.2531
## Mean     : 0.3412                      Mean     : 0.3103
## 3rd Qu.: 0.4905                      3rd Qu.: 0.5097
## Max.     : 0.9974                      Max.     : 0.9193
##
## FALSE, FALSE, TRUE, classif.randomForest
## Min.      :-0.2723
## 1st Qu.: 0.1477
## Median : 0.4069
## Mean     : 0.4403
## 3rd Qu.: 0.7769
## Max.     : 0.9620
## NA's     :2
## FALSE, FALSE, TRUE, classif.xgboost FALSE, TRUE, FALSE, classif.ksvm
## Min.      :-0.3331                      Min.      :-0.044918
## 1st Qu.: 0.1351                      1st Qu.: 0.001888
## Median : 0.3933                      Median : 0.081689
```

```
## Mean      : 0.4130                      Mean      : 0.182077
## 3rd Qu.: 0.7141                      3rd Qu.: 0.247412
## Max.      : 0.9630                      Max.      : 0.975899
##
## FALSE, TRUE, FALSE, classif.randomForest
## Min.      : -0.12254
## 1st Qu.: 0.04784
## Median : 0.21627
## Mean      : 0.32352
## 3rd Qu.: 0.52199
## Max.      : 0.99475
## NA's      : 2
## FALSE, TRUE, FALSE, classif.xgboost SMOTE, FALSE, FALSE, classif.ksvm
## Min.      : -0.31686                      Min.      : -0.2097
## 1st Qu.: 0.06474                      1st Qu.: 0.0000
## Median : 0.34591                      Median : 0.0885
## Mean      : 0.33965                      Mean      : 0.1744
## 3rd Qu.: 0.49930                      3rd Qu.: 0.2195
## Max.      : 0.99743                      Max.      : 0.9737
##
## SMOTE, FALSE, FALSE, classif.randomForest
## Min.      : -0.3169
## 1st Qu.: 0.1182
## Median : 0.2789
## Mean      : 0.3500
## 3rd Qu.: 0.5398
## Max.      : 0.9974
## NA's      : 4
## SMOTE, FALSE, FALSE, classif.xgboost
## Min.      : -0.3169
## 1st Qu.: 0.1274
## Median : 0.2950
## Mean      : 0.3811
## 3rd Qu.: 0.5575
## Max.      : 1.0000
##
```

Verificando a média de cada coluna selecionada

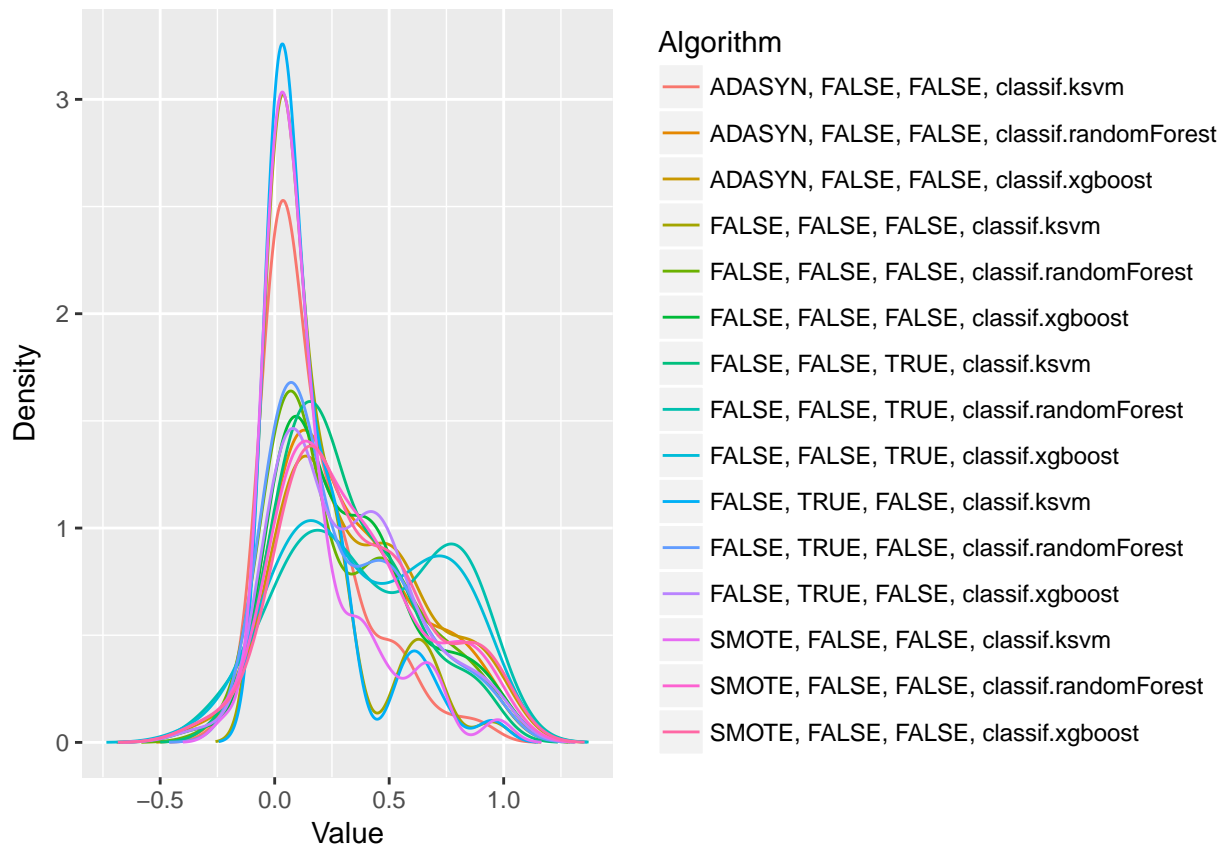
```
for(i in (1:dim(df)[2])){
  print(paste("Media da coluna ", colnames(df)[i], " = ", mean(df[,i], na.rm = TRUE), sep=""))
}
```

```
## [1] "Media da coluna ADASYN, FALSE, FALSE, classif.ksvm = 0.187035785419041"
## [1] "Media da coluna ADASYN, FALSE, FALSE, classif.randomForest = 0.331662914340239"
## [1] "Media da coluna ADASYN, FALSE, FALSE, classif.xgboost = 0.385206990231299"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.ksvm = 0.191603984233634"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.randomForest = 0.3194269697797"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.xgboost = 0.341222552581581"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.ksvm = 0.310319234311459"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.randomForest = 0.440326682922079"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.xgboost = 0.412950676243985"
```

```
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.ksvm = 0.18207697753926"
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.randomForest = 0.32351729479891"
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.xgboost = 0.339645660473032"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.ksvm = 0.17443656511907"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.randomForest = 0.34996250688299"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.xgboost = 0.381120263102655"
```

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 = 226.13, 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,]                                TRUE
## [6,]                                TRUE
## [7,]                                TRUE
## [8,]                                TRUE
## [9,]                                TRUE
## [10,]                               FALSE
## [11,]                               TRUE
## [12,]                               TRUE
## [13,]                               FALSE
## [14,]                               TRUE
## [15,]                               TRUE
##      ADASYN, FALSE, FALSE, classif.randomForest
## [1,]                                TRUE
## [2,]                                FALSE
## [3,]                                FALSE
## [4,]                                TRUE
## [5,]                                FALSE
## [6,]                                FALSE
## [7,]                                FALSE
## [8,]                                FALSE
## [9,]                                FALSE
## [10,]                               TRUE
## [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,]                                TRUE
## [8,]                                FALSE
## [9,]                                FALSE
## [10,]                               TRUE
## [11,]                               TRUE
## [12,]                               FALSE
## [13,]                               TRUE
## [14,]                               FALSE
## [15,]                               FALSE
```



```

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

```

```

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

Plotando os ranks

```
print(colMeans(rankMatrix(df)))
```

```
##      ADASYN, FALSE, FALSE, classif.ksvm
##      11.477273
## ADASYN, FALSE, FALSE, classif.randomForest
##      7.098485
##      ADASYN, FALSE, FALSE, classif.xgboost
##      5.340909
##      FALSE, FALSE, FALSE, classif.ksvm
##      10.810606
## FALSE, FALSE, FALSE, classif.randomForest
##      8.401515
##      FALSE, FALSE, FALSE, classif.xgboost
##      7.090909
##      FALSE, FALSE, TRUE, classif.ksvm
##      8.257576
## FALSE, FALSE, TRUE, classif.randomForest
##      5.439394
##      FALSE, FALSE, TRUE, classif.xgboost
##      5.848485
##      FALSE, TRUE, FALSE, classif.ksvm
##      10.977273
## FALSE, TRUE, FALSE, classif.randomForest
##      8.689394
##      FALSE, TRUE, FALSE, classif.xgboost
##      7.598485
##      SMOTE, FALSE, FALSE, classif.ksvm
##      11.022727
## SMOTE, FALSE, FALSE, classif.randomForest
##      6.856061
##      SMOTE, FALSE, FALSE, classif.xgboost
##      5.090909
```

Plotando grafico de Critical Difference

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

