

R Notebook

Parametros:

```
Measure = G-mean
Columns = sampling, weight_space, underbagging, learner
Performance = holdout_measure
Filter keys = imba.rate
Filter values = 0.01
```

```
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.csv")
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      :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.:0.5895  1st Qu.:0.0000  1st Qu.:0.1104
## Median :0.9629  Median :0.7066  Median :0.4187
## Mean :0.7515  Mean :0.5605  Mean :0.4386
## 3rd Qu.:0.9987  3rd Qu.:0.9645  3rd Qu.:0.7566
## Max. :1.0000  Max. :1.0000  Max. :1.0000
## NA's :54      NA's :54      NA's :54
## iteration_count      dataset      imba.rate
## Min. :1      abalone      : 45  Min. :0.01
## 1st Qu.:1      adult      : 45  1st Qu.:0.01
## Median :2      bank      : 45  Median :0.01
## Mean :2      car      : 45  Mean :0.01
## 3rd Qu.:3      cardiocography-10clases: 45  3rd Qu.:0.01
## Max. :3      cardiocography-3clases : 45  Max. :0.01
## NA's :54      (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

```
# 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] 40 15
```

```
# Renomeando a variavel
df = df_tec_wide_residual

head(df)
```

```
## ADASYN, FALSE, FALSE, classif.ksvm
## 1 0.0000000
## 2 0.0000000
## 3 0.0000000
## 4 0.4646156
## 5 0.0000000
## 6 0.1924501
## ADASYN, FALSE, FALSE, classif.randomForest
## 1 0.0000000
## 2 NA
## 3 0.0000000
## 4 1.0000000
## 5 0.6372650
## 6 0.9388322
## ADASYN, FALSE, FALSE, classif.xgboost FALSE, FALSE, FALSE, classif.ksvm
## 1 0.0000000 0.00000000
## 2 0.4236795 0.08202566
## 3 0.0000000 0.00000000
## 4 0.9994722 0.85911676
## 5 0.7880998 0.63807119
## 6 0.9370562 0.73678114
## FALSE, FALSE, FALSE, classif.randomForest
## 1 0.0000000
## 2 NA
```

```

## 3          0.0000000
## 4          1.0000000
## 5          0.3333333
## 6          0.9995564
## FALSE, FALSE, FALSE, classif.xgboost FALSE, FALSE, TRUE, classif.ksvm
## 1          0.0000000          0.5587283
## 2          0.4769822          0.6875628
## 3          0.0000000          0.6598937
## 4          1.0000000          1.0000000
## 5          0.4997894          0.7960663
## 6          0.9991129          0.9366085
## FALSE, FALSE, TRUE, classif.randomForest
## 1          0.6129433
## 2          0.8247649
## 3          0.8307118
## 4          0.9931144
## 5          0.9476091
## 6          0.9955549
## FALSE, FALSE, TRUE, classif.xgboost FALSE, TRUE, FALSE, classif.ksvm
## 1          0.6692660          0.0000000
## 2          0.8515676          0.06713144
## 3          0.8557636          0.0000000
## 4          0.9791795          0.85911676
## 5          0.8936202          0.63807119
## 6          0.9847078          0.73678114
## FALSE, TRUE, FALSE, classif.randomForest
## 1          0.0000000
## 2          NA
## 3          0.0000000
## 4          1.0000000
## 5          0.5000000
## 6          0.9995564
## FALSE, TRUE, FALSE, classif.xgboost SMOTE, FALSE, FALSE, classif.ksvm
## 1          0.0000000          0.0000000
## 2          0.5147004          0.1144564
## 3          0.0000000          0.0000000
## 4          1.0000000          0.6054989
## 5          0.7440169          0.1666667
## 6          0.9991129          0.1924501
## SMOTE, FALSE, FALSE, classif.randomForest
## 1          0.0000000
## 2          NA
## 3          0.0000000
## 4          1.0000000
## 5          0.5686145
## 6          0.9388322
## SMOTE, FALSE, FALSE, classif.xgboost
## 1          0.0000000
## 2          0.4189175
## 3          0.0000000
## 4          1.0000000
## 5          0.6905357
## 6          0.9379451

```

```
summary(df)
```

```
## ADASYN, FALSE, FALSE, classif.ksvm
## Min. :0.0000
## 1st Qu.:0.0000
## Median :0.0000
## Mean :0.2053
## 3rd Qu.:0.3701
## Max. :0.9459
## NA's :2
## ADASYN, FALSE, FALSE, classif.randomForest
## Min. :0.0000
## 1st Qu.:0.2025
## Median :0.6525
## Mean :0.5809
## 3rd Qu.:0.9388
## Max. :1.0000
## NA's :6
## ADASYN, FALSE, FALSE, classif.xgboost FALSE, FALSE, FALSE, classif.ksvm
## Min. :0.0000 Min. :0.0000
## 1st Qu.:0.4675 1st Qu.:0.0000
## Median :0.7966 Median :0.2452
## Mean :0.6912 Mean :0.3649
## 3rd Qu.:0.9739 3rd Qu.:0.6842
## Max. :1.0000 Max. :1.0000
##
## FALSE, FALSE, FALSE, classif.randomForest
## Min. :0.0000
## 1st Qu.:0.0000
## Median :0.6571
## Mean :0.5444
## 3rd Qu.:0.9599
## Max. :1.0000
## NA's :1
## FALSE, FALSE, FALSE, classif.xgboost FALSE, FALSE, TRUE, classif.ksvm
## Min. :0.0000 Min. :0.2753
## 1st Qu.:0.1336 1st Qu.:0.6106
## Median :0.6388 Median :0.7672
## Mean :0.5514 Mean :0.7327
## 3rd Qu.:0.9446 3rd Qu.:0.9024
## Max. :1.0000 Max. :1.0000
##
## FALSE, FALSE, TRUE, classif.randomForest
## Min. :0.5538
## 1st Qu.:0.8654
## Median :0.9471
## Mean :0.8994
## 3rd Qu.:0.9851
## Max. :1.0000
##
## FALSE, FALSE, TRUE, classif.xgboost FALSE, TRUE, FALSE, classif.ksvm
## Min. :0.3532 Min. :0.0000
## 1st Qu.:0.8584 1st Qu.:0.0000
## Median :0.9099 Median :0.1853
```

```
## Mean      :0.8884                      Mean      :0.3610
## 3rd Qu.:0.9765                      3rd Qu.:0.6842
## Max.      :1.0000                      Max.      :1.0000
##
## FALSE, TRUE, FALSE, classif.randomForest
## Min.      :0.0000
## 1st Qu.:0.0000
## Median :0.5877
## Mean      :0.5306
## 3rd Qu.:0.9343
## Max.      :1.0000
## NA's      :2
## FALSE, TRUE, FALSE, classif.xgboost SMOTE, FALSE, FALSE, classif.ksvm
## Min.      :0.0000                      Min.      :0.0000
## 1st Qu.:0.1889                      1st Qu.:0.0000
## Median :0.7089                      Median :0.1353
## Mean      :0.5658                      Mean      :0.2352
## 3rd Qu.:0.9399                      3rd Qu.:0.4799
## Max.      :1.0000                      Max.      :1.0000
##
## SMOTE, FALSE, FALSE, classif.randomForest
## Min.      :0.0000
## 1st Qu.:0.2336
## Median :0.5690
## Mean      :0.5659
## 3rd Qu.:0.9388
## Max.      :1.0000
## NA's      :7
## SMOTE, FALSE, FALSE, classif.xgboost
## Min.      :0.0000
## 1st Qu.:0.5381
## Median :0.7757
## Mean      :0.6739
## 3rd Qu.:0.9756
## Max.      :1.0000
##
```

Verificando a média de cada coluna selecionada

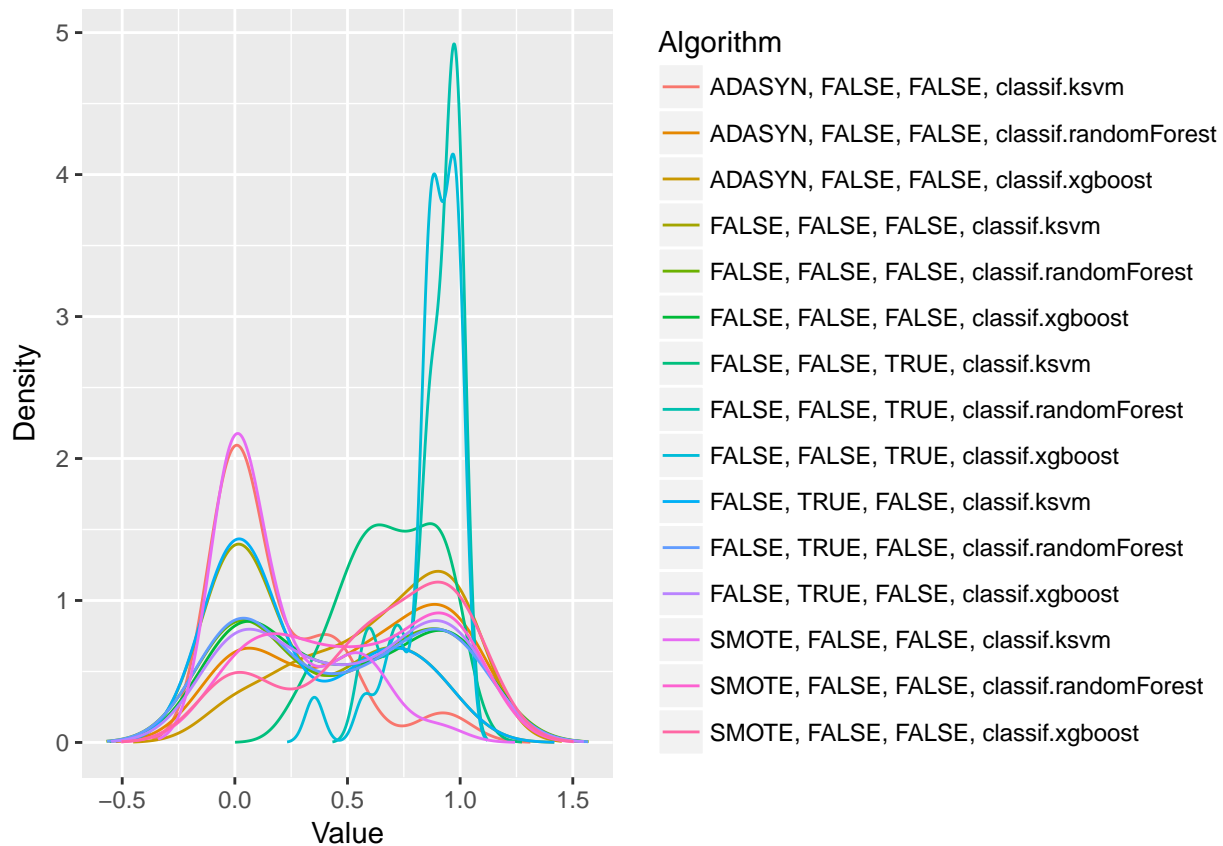
```
for(i in (1:dim(df)[2])){
  print(paste("Media da coluna ", colnames(df)[i], " = ", mean(df[,i], na.rm = TRUE), sep=""))
}
```

```
## [1] "Media da coluna ADASYN, FALSE, FALSE, classif.ksvm = 0.205303144031441"
## [1] "Media da coluna ADASYN, FALSE, FALSE, classif.randomForest = 0.58093468728437"
## [1] "Media da coluna ADASYN, FALSE, FALSE, classif.xgboost = 0.691204431259122"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.ksvm = 0.364947033827767"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.randomForest = 0.544441797113101"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.xgboost = 0.551373055070062"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.ksvm = 0.732730227110866"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.randomForest = 0.899410390473483"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.xgboost = 0.888413748602069"
```

```
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.ksvm = 0.361043106852639"
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.randomForest = 0.530568389245812"
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.xgboost = 0.565815486002984"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.ksvm = 0.235150346934879"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.randomForest = 0.565926616855987"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.xgboost = 0.673937553992975"
```

Fazendo teste de normalidade

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



Testando as diferenças

```
friedmanTest(df)
```

```
##
## Friedman's rank sum test
##
## data: df
## Friedman's chi-squared = 225.88, 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,] TRUE
## [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,] FALSE
## [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,]     TRUE
## [14,]     FALSE
## [15,]     FALSE
##      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,] FALSE
## [10,] TRUE
## [11,] FALSE
## [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,] FALSE 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,] 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, 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,] TRUE
## [6,] TRUE
## [7,] TRUE
## [8,] TRUE
## [9,] TRUE
## [10,] FALSE
## [11,] TRUE
## [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,] FALSE
## [6,] FALSE
## [7,] FALSE
## [8,] FALSE
## [9,] FALSE
## [10,] TRUE
## [11,] FALSE
## [12,] FALSE
## [13,] TRUE
## [14,] FALSE
## [15,] FALSE
```

Plotando os ranks

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

```
##      ADASYN, FALSE, FALSE, classif.ksvm
##      12.2000
## ADASYN, FALSE, FALSE, classif.randomForest
##      8.2750
##      ADASYN, FALSE, FALSE, classif.xgboost
##      5.4500
##      FALSE, FALSE, FALSE, classif.ksvm
##      10.6875
## FALSE, FALSE, FALSE, classif.randomForest
##      8.9250
##      FALSE, FALSE, FALSE, classif.xgboost
##      7.9625
##      FALSE, FALSE, TRUE, classif.ksvm
##      6.0000
## FALSE, FALSE, TRUE, classif.randomForest
##      3.1250
##      FALSE, FALSE, TRUE, classif.xgboost
##      3.4500
##      FALSE, TRUE, FALSE, classif.ksvm
##      10.6875
## FALSE, TRUE, FALSE, classif.randomForest
##      8.6375
##      FALSE, TRUE, FALSE, classif.xgboost
##      7.7750
##      SMOTE, FALSE, FALSE, classif.ksvm
##      12.3375
## SMOTE, FALSE, FALSE, classif.randomForest
##      8.7500
##      SMOTE, FALSE, FALSE, classif.xgboost
##      5.7375
```

Plotando grafico de Critical Difference

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

