

R Notebook

Parametros:

Measure = Matthews correlation coefficient
Columns = sampling, weight_space, underbagging, learner
Performance = holdout_measure_residual
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.  
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               : 0              NA's :0
## Matthews correlation coefficient:1800
##
##
## tuning_measure      holdout_measure      holdout_measure_residual
## Min. : -0.00646      Min. : -0.1370      Min. : -0.06817
## 1st Qu.: 0.23261      1st Qu.: 0.0000      1st Qu.: 0.02011
## Median : 0.82014      Median : 0.3764      Median : 0.19200
## Mean : 0.64070      Mean : 0.4285      Mean : 0.29498
## 3rd Qu.: 0.99730      3rd Qu.: 0.8152      3rd Qu.: 0.49996
## Max. : 1.00000      Max. : 1.0000      Max. : 1.00000
## NA's :69            NA's :69            NA's :69
## 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 :69      (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 0.070168439
## 2 0.101553966
## 3 -0.002295506
## 4 0.100755082
## 5 0.000000000
## 6 0.082764035
## ADASYN, FALSE, FALSE, classif.randomForest
## 1 0.044557657
## 2 0.193687964
## 3 0.055661869
## 4 -0.002843659
## 5 0.120759278
## 6 0.420069262
## ADASYN, FALSE, FALSE, classif.xgboost FALSE, FALSE, FALSE, classif.ksvm
## 1 0.02455997 0.03701282
## 2 0.30955469 0.09710987
## 3 0.12388679 0.00000000
## 4 -0.03745364 0.06878240
## 5 0.41555449 0.23602639
## 6 0.46115069 0.18013252
## FALSE, FALSE, FALSE, classif.randomForest
## 1 0.005300235
## 2 NA
```

```

## 3          0.000000000
## 4          0.049944066
## 5          0.249552466
## 6          0.448111378
## FALSE, FALSE, FALSE, classif.xgboost FALSE, FALSE, TRUE, classif.ksvm
## 1          0.005300235          0.17619019
## 2          0.302563452          0.37634653
## 3          0.016930843          0.37090560
## 4          -0.038884630          0.70821704
## 5          0.252523439          0.05484309
## 6          0.464094499          0.14171779
## FALSE, FALSE, TRUE, classif.randomForest
## 1          0.21508257
## 2          0.62588263
## 3          0.54025870
## 4          0.05490573
## 5          0.71427085
## 6          0.61945579
## FALSE, FALSE, TRUE, classif.xgboost FALSE, TRUE, FALSE, classif.ksvm
## 1          0.2024389          0.002457572
## 2          0.6245448          0.108197202
## 3          0.5081803          0.000000000
## 4          0.1101324          0.068782404
## 5          0.7047332          0.236026391
## 6          0.4962446          0.180132517
## FALSE, TRUE, FALSE, classif.randomForest
## 1          0.005300235
## 2          NA
## 3          0.000000000
## 4          -0.024568018
## 5          0.276159001
## 6          0.404666469
## FALSE, TRUE, FALSE, classif.xgboost SMOTE, FALSE, FALSE, classif.ksvm
## 1          0.005300235          0.06360859
## 2          0.302725524          0.10857900
## 3          0.023934756          0.01349113
## 4          -0.045210001          0.10462939
## 5          0.231905081          0.03874297
## 6          0.462178209          0.07725384
## SMOTE, FALSE, FALSE, classif.randomForest
## 1          0.06250968
## 2          0.22879628
## 3          0.08388880
## 4          0.02958237
## 5          0.10195342
## 6          0.41339027
## SMOTE, FALSE, FALSE, classif.xgboost
## 1          0.02420618
## 2          0.29725558
## 3          0.11009171
## 4          -0.06686697
## 5          0.38060645
## 6          0.47912151

```

```
summary(df)
```

```
## ADASYN, FALSE, FALSE, classif.ksvm
## Min.      :-0.002393
## 1st Qu.: 0.000000
## Median : 0.058874
## Mean     : 0.138420
## 3rd Qu.: 0.196052
## Max.     : 0.851072
## NA's     :2
## ADASYN, FALSE, FALSE, classif.randomForest
## Min.      :-0.005847
## 1st Qu.: 0.056921
## Median : 0.247470
## Mean     : 0.320701
## 3rd Qu.: 0.518985
## Max.     : 0.950347
## NA's     :8
## ADASYN, FALSE, FALSE, classif.xgboost FALSE, FALSE, FALSE, classif.ksvm
## Min.      :-0.03745      Min.      :-0.0008605
## 1st Qu.: 0.09532      1st Qu.: 0.0000000
## Median : 0.39846      Median : 0.0786579
## Mean     : 0.38898      Mean     : 0.1900848
## 3rd Qu.: 0.52592      3rd Qu.: 0.2433956
## Max.     : 0.95928      Max.     : 0.8585106
##
## FALSE, FALSE, FALSE, classif.randomForest
## Min.      :-0.01023
## 1st Qu.: 0.01367
## Median : 0.17754
## Mean     : 0.26425
## 3rd Qu.: 0.49381
## Max.     : 0.89881
## NA's     :3
## FALSE, FALSE, FALSE, classif.xgboost FALSE, FALSE, TRUE, classif.ksvm
## Min.      :-0.03888      Min.      :-0.003467
## 1st Qu.: 0.05539      1st Qu.: 0.095746
## Median : 0.25404      Median : 0.324663
## Mean     : 0.30227      Mean     : 0.316728
## 3rd Qu.: 0.48833      3rd Qu.: 0.525981
## Max.     : 0.90474      Max.     : 0.914968
##
## FALSE, FALSE, TRUE, classif.randomForest
## Min.      :0.006653
## 1st Qu.:0.147313
## Median :0.471366
## Mean     :0.456152
## 3rd Qu.:0.735894
## Max.     :0.916350
## NA's     :1
## FALSE, FALSE, TRUE, classif.xgboost FALSE, TRUE, FALSE, classif.ksvm
## Min.      :0.01815      Min.      :-0.0008605
## 1st Qu.:0.18818      1st Qu.: 0.0000000
## Median :0.46926      Median : 0.0758148
```

```
## Mean :0.45167 Mean : 0.1866649
## 3rd Qu.:0.67906 3rd Qu.: 0.2433956
## Max. :0.89511 Max. : 0.8585106
##
## FALSE, TRUE, FALSE, classif.randomForest
## Min. :-0.02457
## 1st Qu.: 0.02226
## Median : 0.15128
## Mean : 0.26963
## 3rd Qu.: 0.49383
## Max. : 0.88606
## NA's :4
## FALSE, TRUE, FALSE, classif.xgboost SMOTE, FALSE, FALSE, classif.ksvm
## Min. :-0.04521 Min. :-0.002561
## 1st Qu.: 0.05402 1st Qu.: 0.001628
## Median : 0.24338 Median : 0.063074
## Mean : 0.30232 Mean : 0.142549
## 3rd Qu.: 0.49302 3rd Qu.: 0.178964
## Max. : 0.89582 Max. : 0.709685
##
## SMOTE, FALSE, FALSE, classif.randomForest
## Min. :-0.004377
## 1st Qu.: 0.070937
## Median : 0.228796
## Mean : 0.299965
## 3rd Qu.: 0.460087
## Max. : 0.938403
## NA's :5
## SMOTE, FALSE, FALSE, classif.xgboost
## Min. :-0.06687
## 1st Qu.: 0.10196
## Median : 0.38049
## Mean : 0.39147
## 3rd Qu.: 0.59879
## Max. : 0.93815
##
```

Verificando a média de cada coluna selecionada

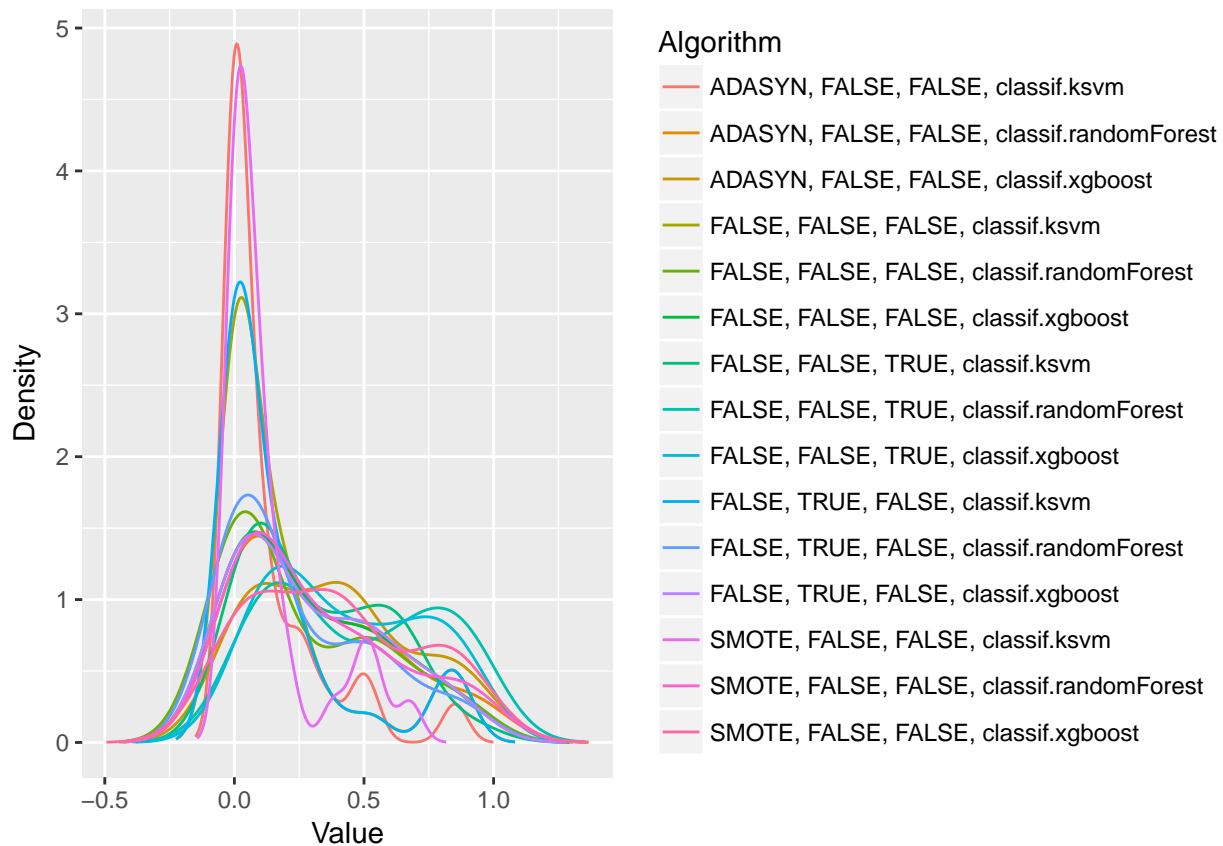
```
for(i in (1:dim(df)[2])){
  print(paste("Media da coluna ", colnames(df)[i], " = ", mean(df[,i], na.rm = TRUE), sep=""))
}
```

```
## [1] "Media da coluna ADASYN, FALSE, FALSE, classif.ksvm = 0.13841968187075"
## [1] "Media da coluna ADASYN, FALSE, FALSE, classif.randomForest = 0.320701152073292"
## [1] "Media da coluna ADASYN, FALSE, FALSE, classif.xgboost = 0.388975025643202"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.ksvm = 0.190084830988913"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.randomForest = 0.264245987227245"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.xgboost = 0.302270429895642"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.ksvm = 0.316727703199023"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.randomForest = 0.456151856986097"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.xgboost = 0.451670509813634"
```

```
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.ksvm = 0.186664904350653"
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.randomForest = 0.269628341608909"
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.xgboost = 0.302317138155214"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.ksvm = 0.142549319028629"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.randomForest = 0.299964908751718"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.xgboost = 0.391465990700945"
```

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 = 172.65, 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,]                                     FALSE
## [3,]                                     TRUE
## [4,]                                     FALSE
## [5,]                                     FALSE
## [6,]                                     TRUE
## [7,]                                     TRUE
## [8,]                                     TRUE
## [9,]                                     TRUE
## [10,]                                    FALSE
## [11,]                                    FALSE
## [12,]                                    TRUE
## [13,]                                    FALSE
## [14,]                                    FALSE
## [15,]                                    TRUE
##      ADASYN, 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,]                                    FALSE
##      ADASYN, FALSE, FALSE, classif.xgboost
## [1,]                                     TRUE
## [2,]                                     TRUE
## [3,]                                     FALSE
## [4,]                                     TRUE
## [5,]                                     TRUE
## [6,]                                     FALSE
## [7,]                                     FALSE
## [8,]                                     FALSE
## [9,]                                     FALSE
## [10,]                                    TRUE
## [11,]                                    TRUE
## [12,]                                    FALSE
## [13,]                                    TRUE
## [14,]                                    TRUE
## [15,]                                    FALSE
```



```

##      FALSE, FALSE, FALSE, classif.ksvm
## [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.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,]      FALSE
## [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,]      FALSE
## [5,]      FALSE

```

```

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

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

```
##      ADASYN, FALSE, FALSE, classif.ksvm
##      11.7500
## ADASYN, FALSE, FALSE, classif.randomForest
##      8.3625
##      ADASYN, FALSE, FALSE, classif.xgboost
##      4.5375
##      FALSE, FALSE, FALSE, classif.ksvm
##      10.4500
## FALSE, FALSE, FALSE, classif.randomForest
##      8.7875
##      FALSE, FALSE, FALSE, classif.xgboost
##      7.6500
##      FALSE, FALSE, TRUE, classif.ksvm
##      7.5375
## FALSE, FALSE, TRUE, classif.randomForest
##      4.4500
##      FALSE, FALSE, TRUE, classif.xgboost
##      4.2125
##      FALSE, TRUE, FALSE, classif.ksvm
##      10.6250
## FALSE, TRUE, FALSE, classif.randomForest
##      9.3375
##      FALSE, TRUE, FALSE, classif.xgboost
##      7.5375
##      SMOTE, FALSE, FALSE, classif.ksvm
##      11.1000
## SMOTE, FALSE, FALSE, classif.randomForest
##      8.4125
##      SMOTE, FALSE, FALSE, classif.xgboost
##      5.2500
```

Plotando grafico de Critical Difference

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

