

R Notebook

Parametros:

```
Measure = Accuracy
Columns = sampling, weight_space, underbagging, learner
Performance = tuning_measure
Filter keys = NULL
Filter values = NULL
```

```
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      :3420  Mode :logical
## classif.randomForest:3420 FALSE:8208
## classif.rusboost   :    0  TRUE :2052
## classif.xgboost    :3420  NA's :0
##
##
##
##          measure      sampling  underbagging
## Accuracy          :10260  ADASYN:2052  Mode :logical
## Area under the curve :    0  FALSE :6156  FALSE:8208
## F1 measure          :    0  SMOTE :2052  TRUE :2052
## G-mean              :    0              NA's :0
## Matthews correlation coefficient:    0
##
##
## tuning_measure  holdout_measure  holdout_measure_residual
## Min. :0.09041  Min. :0.01517  Min. :0.0346
## 1st Qu.:0.96185 1st Qu.:0.95349 1st Qu.:0.3809
## Median :0.98796 Median :0.98113 Median :0.7239
## Mean :0.95509  Mean :0.94933  Mean :0.6600
## 3rd Qu.:0.99669 3rd Qu.:0.99347 3rd Qu.:0.9428
## Max. :1.00000  Max. :1.00000  Max. :1.0000
## NA's :204      NA's :204      NA's :204
## iteration_count      dataset      imba.rate
## Min. :1      abalone      : 180  Min. :0.0010
## 1st Qu.:1      adult      : 180  1st Qu.:0.0100
## Median :2      bank      : 180  Median :0.0300
## Mean :2      car      : 180  Mean :0.0286
## 3rd Qu.:3      cardiocography-10clases: 180 3rd Qu.:0.0500
## Max. :3      cardiocography-3clases : 180 Max. :0.0500
## NA's :204      (Other)      :9180
```

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] 228 15
```

```
# Renomeando a variavel
df = df_tec_wide_residual

head(df)
```

```
## ADASYN, FALSE, FALSE, classif.ksvm
## 1 0.9853969
## 2 0.9853969
## 3 0.9595381
## 4 0.9453798
## 5 0.9920423
## 6 0.9920423
## ADASYN, FALSE, FALSE, classif.randomForest
## 1 0.9818597
## 2 0.9818597
## 3 0.9566567
## 4 0.9386682
## 5 NA
## 6 0.9921523
## ADASYN, FALSE, FALSE, classif.xgboost FALSE, FALSE, FALSE, classif.ksvm
## 1 0.9864477 0.9878042
## 2 0.9864477 0.9878042
## 3 0.9671889 0.9578372
## 4 0.9573086 0.9266667
## 5 0.9941462 0.9891070
## 6 0.9941462 0.9891070
## FALSE, FALSE, FALSE, classif.randomForest
## 1 0.9897371
## 2 0.9897371
```

```

## 3          0.9697619
## 4          0.9500000
## 5          NA
## 6          0.9913157
## FALSE, FALSE, FALSE, classif.xgboost FALSE, FALSE, TRUE, classif.ksvm
## 1          0.9897373          0.5674864
## 2          0.9897373          0.5674864
## 3          0.9697619          0.6175442
## 4          0.9500000          0.6054167
## 5          0.9915333          0.8833724
## 6          0.9915333          0.8833724
## FALSE, FALSE, TRUE, classif.randomForest
## 1          0.5220884
## 2          0.5220884
## 3          0.5934099
## 4          0.6205556
## 5          0.7848836
## 6          0.7848836
## FALSE, FALSE, TRUE, classif.xgboost FALSE, TRUE, FALSE, classif.ksvm
## 1          0.5574877          0.9883986
## 2          0.5574877          0.9883986
## 3          0.6314521          0.9625227
## 4          0.6205556          0.9345833
## 5          0.8011477          0.9896592
## 6          0.8011477          0.9896592
## FALSE, TRUE, FALSE, classif.randomForest
## 1          0.9897373
## 2          0.9897373
## 3          0.9697619
## 4          0.9500000
## 5          0.9913659
## 6          0.9913659
## FALSE, TRUE, FALSE, classif.xgboost SMOTE, FALSE, FALSE, classif.ksvm
## 1          0.9897373          0.9845604
## 2          0.9897373          0.9845604
## 3          0.9699038          0.9611646
## 4          0.9500000          0.9448830
## 5          0.9916671          0.9930487
## 6          0.9916671          0.9930487
## SMOTE, FALSE, FALSE, classif.randomForest
## 1          0.9821508
## 2          0.9821508
## 3          0.9550077
## 4          0.9364035
## 5          0.9921162
## 6          NA
## SMOTE, FALSE, FALSE, classif.xgboost
## 1          0.9872722
## 2          0.9872722
## 3          0.9676123
## 4          0.9538012
## 5          0.9947695
## 6          0.9947695

```

```
summary(df)
```

```
## ADASYN, FALSE, FALSE, classif.ksvm
## Min. :0.7408
## 1st Qu.:0.9775
## Median :0.9947
## Mean :0.9798
## 3rd Qu.:0.9988
## Max. :1.0000
## NA's :7
## ADASYN, FALSE, FALSE, classif.randomForest
## Min. :0.7035
## 1st Qu.:0.9856
## Median :0.9967
## Mean :0.9856
## 3rd Qu.:0.9992
## Max. :1.0000
## NA's :26
## ADASYN, FALSE, FALSE, classif.xgboost FALSE, FALSE, FALSE, classif.ksvm
## Min. :0.7120 Min. :0.9264
## 1st Qu.:0.9849 1st Qu.:0.9609
## Median :0.9948 Median :0.9766
## Mean :0.9850 Mean :0.9751
## 3rd Qu.:0.9987 3rd Qu.:0.9901
## Max. :1.0000 Max. :1.0000
##
## FALSE, FALSE, FALSE, classif.randomForest
## Min. :0.9442
## 1st Qu.:0.9702
## Median :0.9897
## Mean :0.9821
## 3rd Qu.:0.9939
## Max. :1.0000
## NA's :6
## FALSE, FALSE, FALSE, classif.xgboost FALSE, FALSE, TRUE, classif.ksvm
## Min. :0.9440 Min. :0.2896
## 1st Qu.:0.9716 1st Qu.:0.7244
## Median :0.9898 Median :0.9320
## Mean :0.9827 Mean :0.8491
## 3rd Qu.:0.9951 3rd Qu.:0.9836
## Max. :1.0000 Max. :1.0000
##
## FALSE, FALSE, TRUE, classif.randomForest
## Min. :0.4222
## 1st Qu.:0.7803
## Median :0.8951
## Mean :0.8564
## 3rd Qu.:0.9651
## Max. :1.0000
## NA's :5
## FALSE, FALSE, TRUE, classif.xgboost FALSE, TRUE, FALSE, classif.ksvm
## Min. :0.3600 Min. :0.9346
## 1st Qu.:0.7856 1st Qu.:0.9611
## Median :0.8871 Median :0.9761
```

```
## Mean      :0.8459                      Mean      :0.9752
## 3rd Qu.:0.9508                      3rd Qu.:0.9902
## Max.      :1.0000                      Max.      :1.0000
##
## FALSE, TRUE, FALSE, classif.randomForest
## Min.      :0.9442
## 1st Qu.:0.9700
## Median :0.9897
## Mean      :0.9820
## 3rd Qu.:0.9932
## Max.      :1.0000
## NA's      :6
## FALSE, TRUE, FALSE, classif.xgboost SMOTE, FALSE, FALSE, classif.ksvm
## Min.      :0.9440                      Min.      :0.7394
## 1st Qu.:0.9720                      1st Qu.:0.9780
## Median :0.9898                      Median :0.9952
## Mean      :0.9827                      Mean      :0.9795
## 3rd Qu.:0.9948                      3rd Qu.:0.9994
## Max.      :1.0000                      Max.      :1.0000
##
## SMOTE, FALSE, FALSE, classif.randomForest
## Min.      :0.7259
## 1st Qu.:0.9852
## Median :0.9964
## Mean      :0.9860
## 3rd Qu.:0.9994
## Max.      :1.0000
## NA's      :18
## SMOTE, FALSE, FALSE, classif.xgboost
## Min.      :0.6952
## 1st Qu.:0.9850
## Median :0.9949
## Mean      :0.9851
## 3rd Qu.:0.9987
## Max.      :1.0000
##
```

Verificando a média de cada coluna selecionada

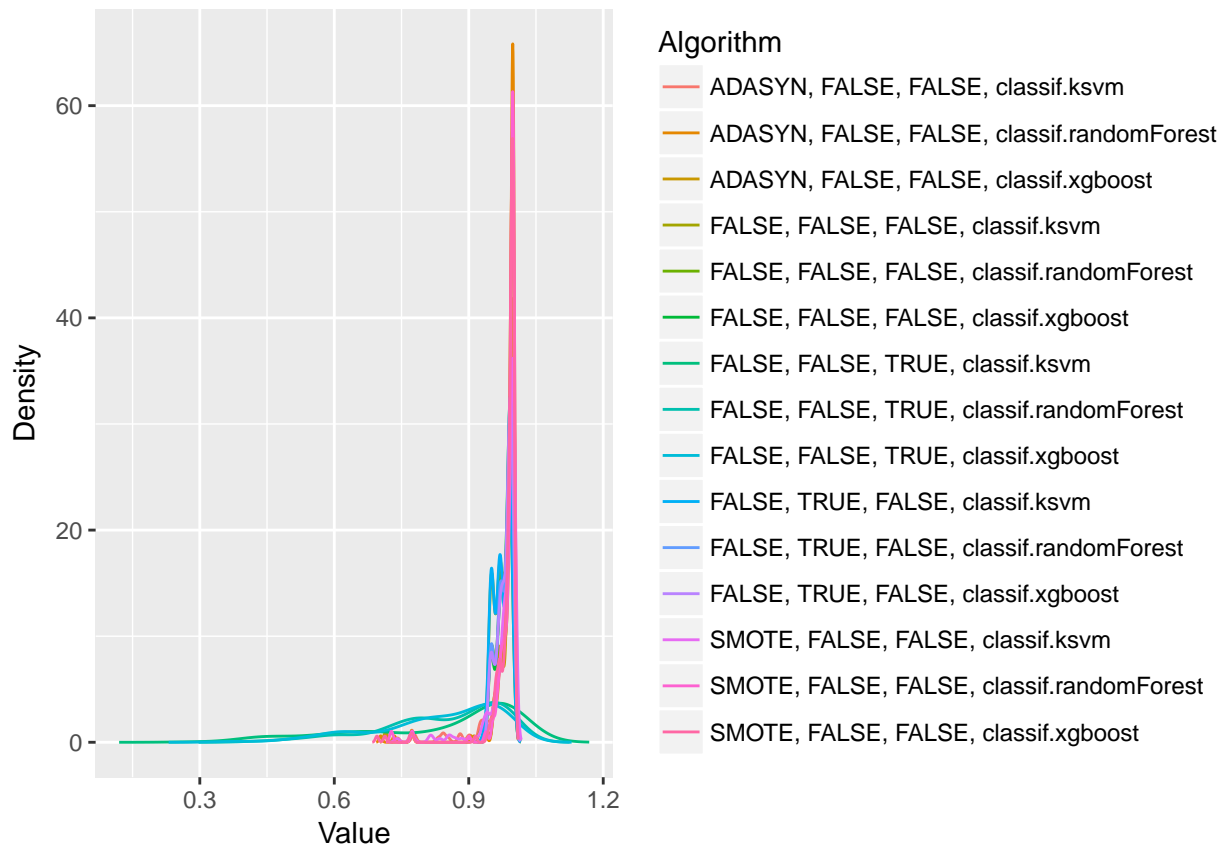
```
for(i in (1:dim(df)[2])){
  print(paste("Media da coluna ", colnames(df)[i], " = ", mean(df[,i], na.rm = TRUE), sep=""))
}
```

```
## [1] "Media da coluna ADASYN, FALSE, FALSE, classif.ksvm = 0.979794708481056"
## [1] "Media da coluna ADASYN, FALSE, FALSE, classif.randomForest = 0.985572124397908"
## [1] "Media da coluna ADASYN, FALSE, FALSE, classif.xgboost = 0.984951544919685"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.ksvm = 0.975133744239946"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.randomForest = 0.982104793887645"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.xgboost = 0.982731622083143"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.ksvm = 0.84906001351275"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.randomForest = 0.856448743496149"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.xgboost = 0.845946456366123"
```

```
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.ksvm = 0.975176544455616"
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.randomForest = 0.981998634989551"
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.xgboost = 0.98271382723725"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.ksvm = 0.979458695568733"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.randomForest = 0.986035995107693"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.xgboost = 0.985137394866884"
```

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 = 1669.7, 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,]                                     FALSE
## [4,]                                     TRUE
## [5,]                                     TRUE
## [6,]                                     TRUE
## [7,]                                     TRUE
## [8,]                                     TRUE
## [9,]                                     TRUE
## [10,]                                    TRUE
## [11,]                                    TRUE
## [12,]                                    TRUE
## [13,]                                    FALSE
## [14,]                                    FALSE
## [15,]                                    FALSE
##      ADASYN, FALSE, FALSE, classif.randomForest
## [1,]                                     FALSE
## [2,]                                     FALSE
## [3,]                                     FALSE
## [4,]                                     TRUE
## [5,]                                     TRUE
## [6,]                                     TRUE
## [7,]                                     TRUE
## [8,]                                     TRUE
## [9,]                                     TRUE
## [10,]                                    TRUE
## [11,]                                    TRUE
## [12,]                                    TRUE
## [13,]                                    FALSE
## [14,]                                    FALSE
## [15,]                                    FALSE
##      ADASYN, FALSE, FALSE, classif.xgboost
## [1,]                                     FALSE
## [2,]                                     FALSE
## [3,]                                     FALSE
## [4,]                                     TRUE
## [5,]                                     TRUE
## [6,]                                     TRUE
## [7,]                                     TRUE
## [8,]                                     TRUE
## [9,]                                     TRUE
## [10,]                                    TRUE
## [11,]                                    TRUE
## [12,]                                    TRUE
## [13,]                                    FALSE
## [14,]                                    FALSE
## [15,]                                    FALSE
```



```

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

```

```

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

```

```

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

```

```
## [2,] FALSE
## [3,] FALSE
## [4,] TRUE
## [5,] TRUE
## [6,] TRUE
## [7,] TRUE
## [8,] TRUE
## [9,] TRUE
## [10,] TRUE
## [11,] TRUE
## [12,] TRUE
## [13,] FALSE
## [14,] FALSE
## [15,] FALSE
```

Plotando os ranks

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

```
##          ADASYN, FALSE, FALSE, classif.ksvm
##                               5.535088
## ADASYN, FALSE, FALSE, classif.randomForest
##                               5.061404
##          ADASYN, FALSE, FALSE, classif.xgboost
##                               4.769737
##          FALSE, FALSE, FALSE, classif.ksvm
##                               10.280702
## FALSE, FALSE, FALSE, classif.randomForest
##                               7.936404
##          FALSE, FALSE, FALSE, classif.xgboost
##                               7.278509
##          FALSE, FALSE, TRUE, classif.ksvm
##                               12.372807
## FALSE, FALSE, TRUE, classif.randomForest
##                               13.252193
##          FALSE, FALSE, TRUE, classif.xgboost
##                               13.620614
##          FALSE, TRUE, FALSE, classif.ksvm
##                               10.260965
## FALSE, TRUE, FALSE, classif.randomForest
##                               8.061404
##          FALSE, TRUE, FALSE, classif.xgboost
##                               7.309211
##          SMOTE, FALSE, FALSE, classif.ksvm
##                               4.973684
## SMOTE, FALSE, FALSE, classif.randomForest
##                               4.563596
##          SMOTE, FALSE, FALSE, classif.xgboost
##                               4.723684
```

Plotando grafico de Critical Difference

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

