

R Notebook

Parametros:

Measure = Matthews correlation coefficient
Columns = sampling, weight_space, underbagging, learner
Performance = holdout_measure
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      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.: 0.23511 1st Qu.: 0.0000 1st Qu.: 0.02210
## Median : 0.81997 Median : 0.3764 Median : 0.19355
## Mean : 0.64034 Mean : 0.4305 Mean : 0.29627
## 3rd Qu.: 0.99727 3rd Qu.: 0.8152 3rd Qu.: 0.49996
## Max. : 1.00000 Max. : 1.0000 Max. : 1.00000
## NA's :54 NA's :54 NA's :54
## 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 cardiocography-10clases: 45 3rd Qu.:0.001
## Max. :3 cardiocography-3clases : 45 Max. :0.001
## 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 -0.014499516
## 2 -0.009081137
## 3 -0.004917732
## 4 0.463579851
## 5 0.000000000
## 6 0.191940287
## ADASYN, FALSE, FALSE, classif.randomForest
## 1 -0.01440011
## 2 0.08263412
## 3 -0.00286235
## 4 1.00000000
## 5 0.49884085
## 6 0.93847099
## ADASYN, FALSE, FALSE, classif.xgboost FALSE, FALSE, FALSE, classif.ksvm
## 1 -0.010554999 -0.002681154
## 2 0.242989502 0.011520780
## 3 -0.009202388 0.000000000
## 4 0.954884675 0.858510613
## 5 0.600634775 0.636257514
## 6 0.794855403 0.735548937
## FALSE, FALSE, FALSE, classif.randomForest
## 1 0.0000000
## 2 0.3603999
```

```

## 3          0.0000000
## 4          1.0000000
## 5          0.3320778
## 6          0.9549577
## FALSE, FALSE, FALSE, classif.xgboost FALSE, FALSE, TRUE, classif.ksvm
## 1          0.0000000          0.03648651
## 2          0.4492768          0.11251890
## 3          0.0000000          0.09881969
## 4          1.0000000          1.00000000
## 5          0.4484451          0.00000000
## 6          0.9099153          0.46374461
## FALSE, FALSE, TRUE, classif.randomForest
## 1          0.05301455
## 2          0.14841422
## 3          0.17269359
## 4          0.68133508
## 5          0.28555458
## 6          0.70932540
## FALSE, FALSE, TRUE, classif.xgboost FALSE, TRUE, FALSE, classif.ksvm
## 1          0.07848644         -0.004028968
## 2          0.16753202          0.009851304
## 3          0.17800632          0.000000000
## 4          0.30122692          0.858510613
## 5          0.24506149          0.636257514
## 6          0.47403606          0.735548937
## FALSE, TRUE, FALSE, classif.randomForest
## 1          0.0000000
## 2          NA
## 3          0.0000000
## 4          1.0000000
## 5          0.4981168
## 6          0.9384710
## FALSE, TRUE, FALSE, classif.xgboost SMOTE, FALSE, FALSE, classif.ksvm
## 1          0.0000000         -0.015024224
## 2          0.3699468          0.012608807
## 3          0.0000000         -0.004539511
## 4          1.0000000          0.605069238
## 5          0.6894596          0.166038918
## 6          0.9099153          0.191940287
## SMOTE, FALSE, FALSE, classif.randomForest
## 1         -0.012556964
## 2          NA
## 3         -0.004333984
## 4          1.000000000
## 5          0.467843754
## 6          0.938470992
## SMOTE, FALSE, FALSE, classif.xgboost
## 1         -0.010790139
## 2          0.266952337
## 3         -0.008995889
## 4          1.000000000
## 5          0.401766327
## 6          0.848386330

```

```
summary(df)
```

```
## ADASYN, FALSE, FALSE, classif.ksvm
## Min.      :-0.0145
## 1st Qu.: 0.0000
## Median : 0.0000
## Mean     : 0.1914
## 3rd Qu.: 0.3232
## Max.     : 0.9455
## NA's     :2
## ADASYN, FALSE, FALSE, classif.randomForest
## Min.      :-0.01440
## 1st Qu.: 0.09364
## Median : 0.50975
## Mean     : 0.50236
## 3rd Qu.: 0.93837
## Max.     : 1.00000
## NA's     :6
## ADASYN, FALSE, FALSE, classif.xgboost FALSE, FALSE, FALSE, classif.ksvm
## Min.      :-0.01056      Min.      :-0.004335
## 1st Qu.: 0.29845      1st Qu.: 0.000000
## Median : 0.64815      Median : 0.225365
## Mean     : 0.59152      Mean     : 0.346678
## 3rd Qu.: 0.93094      3rd Qu.: 0.683887
## Max.     : 1.00000      Max.     : 1.000000
##
## FALSE, FALSE, FALSE, classif.randomForest
## Min.      :-0.002375
## 1st Qu.: 0.000000
## Median : 0.575532
## Mean     : 0.520199
## 3rd Qu.: 0.912029
## Max.     : 1.000000
## NA's     :1
## FALSE, FALSE, FALSE, classif.xgboost FALSE, FALSE, TRUE, classif.ksvm
## Min.      :-0.005742      Min.      :0.0000
## 1st Qu.: 0.170088      1st Qu.:0.1001
## Median : 0.612517      Median :0.1741
## Mean     : 0.538864      Mean     :0.3527
## 3rd Qu.: 0.897964      3rd Qu.:0.6027
## Max.     : 1.000000      Max.     :1.0000
##
## FALSE, FALSE, TRUE, classif.randomForest
## Min.      :0.03794
## 1st Qu.:0.17178
## Median :0.32086
## Mean     :0.38000
## 3rd Qu.:0.50063
## Max.     :1.00000
##
## FALSE, FALSE, TRUE, classif.xgboost FALSE, TRUE, FALSE, classif.ksvm
## Min.      :-0.02588      Min.      :-0.004029
## 1st Qu.: 0.17384      1st Qu.: 0.000000
## Median : 0.26571      Median : 0.184706
```

```
## Mean      : 0.32038                      Mean      : 0.344307
## 3rd Qu.: 0.39103                      3rd Qu.: 0.683887
## Max.      : 1.00000                      Max.      : 1.000000
##
## FALSE, TRUE, FALSE, classif.randomForest
## Min.      : -0.003281
## 1st Qu.: 0.000000
## Median : 0.571561
## Mean      : 0.519272
## 3rd Qu.: 0.915390
## Max.      : 1.000000
## NA's      : 2
## FALSE, TRUE, FALSE, classif.xgboost SMOTE, FALSE, FALSE, classif.ksvm
## Min.      : -0.005032                      Min.      : -0.01502
## 1st Qu.: 0.159433                      1st Qu.: 0.00000
## Median : 0.671363                      Median : 0.02320
## Mean      : 0.537200                      Mean      : 0.23050
## 3rd Qu.: 0.881911                      3rd Qu.: 0.47869
## Max.      : 1.000000                      Max.      : 1.00000
##
## SMOTE, FALSE, FALSE, classif.randomForest
## Min.      : -0.01256
## 1st Qu.: 0.16371
## Median : 0.46784
## Mean      : 0.51562
## 3rd Qu.: 0.93840
## Max.      : 1.00000
## NA's      : 7
## SMOTE, FALSE, FALSE, classif.xgboost
## Min.      : -0.01672
## 1st Qu.: 0.28440
## Median : 0.68064
## Mean      : 0.58764
## 3rd Qu.: 0.92596
## Max.      : 1.00000
##
```

Verificando a média de cada coluna selecionada

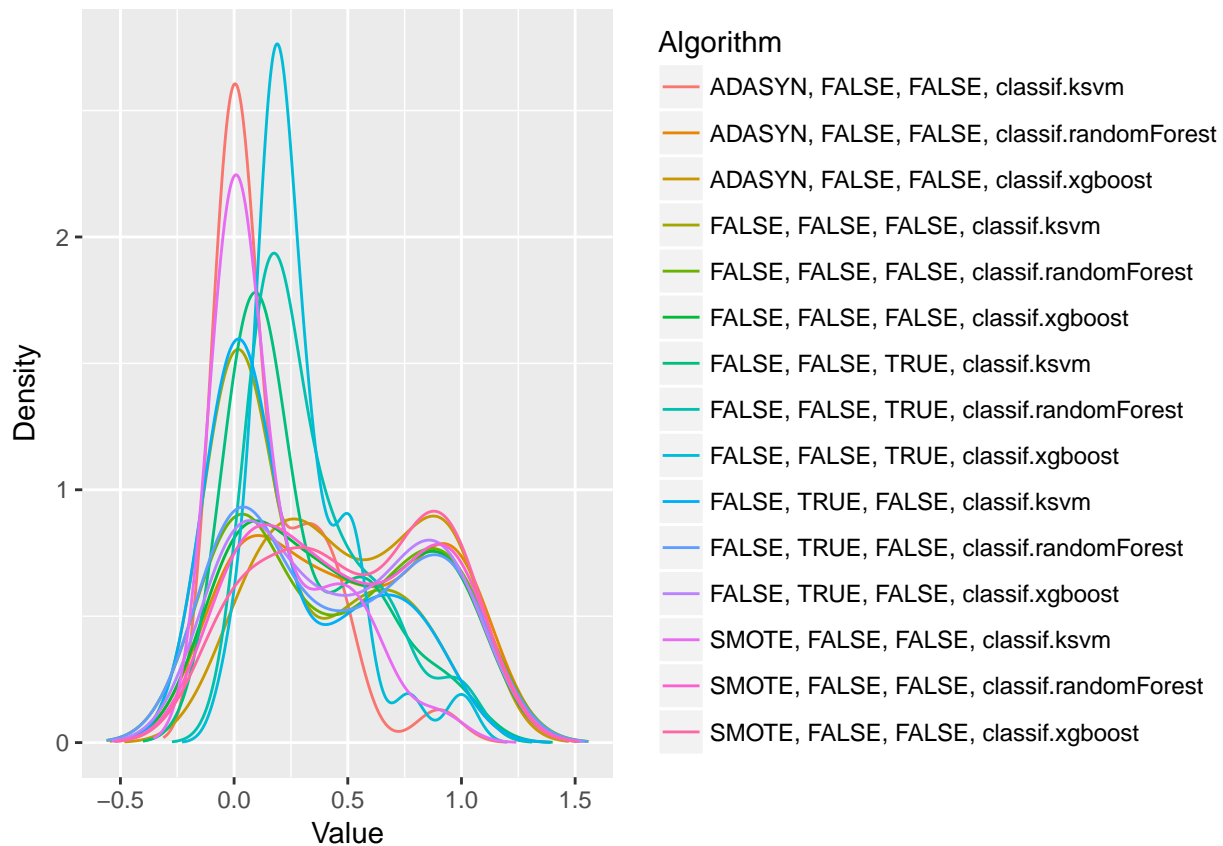
```
for(i in (1:dim(df)[2])){
  print(paste("Media da coluna ", colnames(df)[i], " = ", mean(df[,i], na.rm = TRUE), sep=""))
}
```

```
## [1] "Media da coluna ADASYN, FALSE, FALSE, classif.ksvm = 0.191372282616757"
## [1] "Media da coluna ADASYN, FALSE, FALSE, classif.randomForest = 0.502356819356604"
## [1] "Media da coluna ADASYN, FALSE, FALSE, classif.xgboost = 0.591522231454064"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.ksvm = 0.346677876313596"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.randomForest = 0.52019887229972"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.xgboost = 0.538864265752372"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.ksvm = 0.35266857312572"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.randomForest = 0.379995961885672"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.xgboost = 0.32038390509281"
```

```
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.ksvm = 0.344306687051818"
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.randomForest = 0.519271762268402"
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.xgboost = 0.53719997235341"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.ksvm = 0.230496923387896"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.randomForest = 0.515618230943551"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.xgboost = 0.587635154564366"
```

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 = 120.52, 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,]                                     FALSE
## [8,]                                     TRUE
## [9,]                                     FALSE
## [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,]                                     FALSE
## [9,]                                     FALSE
## [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,]                                     FALSE
## [6,]                                     FALSE
## [7,]                                     TRUE
## [8,]                                     TRUE
## [9,]                                     TRUE
## [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,]      FALSE
## [8,]      FALSE
## [9,]      FALSE
## [10,]     FALSE
## [11,]     FALSE
## [12,]     FALSE
## [13,]     FALSE
## [14,]     FALSE
## [15,]     TRUE
##      FALSE, FALSE, FALSE, classif.randomForest
## [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, 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
##      FALSE, FALSE, TRUE, classif.ksvm
## [1,]      FALSE
## [2,]      FALSE
## [3,]      TRUE
## [4,]      FALSE
## [5,]      FALSE

```

```

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

Plotando os ranks

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

```
##      ADASYN, FALSE, FALSE, classif.ksvm
##      11.7625
## ADASYN, FALSE, FALSE, classif.randomForest
##      7.7125
##      ADASYN, FALSE, FALSE, classif.xgboost
##      4.7375
##      FALSE, FALSE, FALSE, classif.ksvm
##      9.5000
## FALSE, FALSE, FALSE, classif.randomForest
##      6.9250
##      FALSE, FALSE, FALSE, classif.xgboost
##      6.1500
##      FALSE, FALSE, TRUE, classif.ksvm
##      8.6250
## FALSE, FALSE, TRUE, classif.randomForest
##      8.2250
##      FALSE, FALSE, TRUE, classif.xgboost
##      8.8750
##      FALSE, TRUE, FALSE, classif.ksvm
##      9.4750
## FALSE, TRUE, FALSE, classif.randomForest
##      7.1000
##      FALSE, TRUE, FALSE, classif.xgboost
##      6.1250
##      SMOTE, FALSE, FALSE, classif.ksvm
##      11.7000
## SMOTE, FALSE, FALSE, classif.randomForest
##      7.7500
##      SMOTE, FALSE, FALSE, classif.xgboost
##      5.3375
```

Plotando grafico de Critical Difference

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

