R Notebook

Parametros:

Measure = Accuracy
Columns = sampling, weight_space, underbagging, learner
Performance = holdout_measure
Filter keys = imba.rate
Filter values = 0.03

```r
library("scmamp")
library(dplyr)
```

Tratamento dos dados

Carregando data set compilado

```r
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       cardiotocography-10clases:  900   3rd Qu.:0.0500
##  Max.   :3       cardiotocography-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                        :2970   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:   0
##
##
##  tuning_measure    holdout_measure   holdout_measure_residual
##  Min.   :0.09041   Min.   :0.02655   Min.   :0.0346
##  1st Qu.:0.96926   1st Qu.:0.96647   1st Qu.:0.3599
##  Median :0.98130   Median :0.97619   Median :0.6882
##  Mean   :0.95405   Mean   :0.94750   Mean   :0.6478
##  3rd Qu.:0.99560   3rd Qu.:0.99045   3rd Qu.:0.9438
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.0000
##  NA's   :57        NA's   :57        NA's   :57
##  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   :57      (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$performan

# 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                          0.9175199
## 2                          0.9477612
## 3                          0.9571429
## 4                          0.9767442
## 5                          0.9898990
## 6                          0.9584860
##   ADASYN, FALSE, FALSE, classif.randomForest
## 1                                  0.9340159
## 2                                         NA
## 3                                  0.9714286
## 4                                  0.9689922
## 5                                  1.0000000
## 6                                         NA
##   ADASYN, FALSE, FALSE, classif.xgboost FALSE, FALSE, FALSE, classif.ksvm
## 1                             0.9544937                         0.9476678
## 2                             0.9707384                         0.9658288
## 3                             0.9690476                         0.9690476
## 4                             0.9806202                         0.9767442
## 5                             1.0000000                         1.0000000
## 6                             0.9658120                         0.9711030
##   FALSE, FALSE, FALSE, classif.randomForest
## 1                                 0.9709898
## 2                                        NA
## 3                                 0.9833333
## 4                                 0.9728682
## 5                                 1.0000000
## 6                                 0.9715100
##   FALSE, FALSE, FALSE, classif.xgboost FALSE, FALSE, TRUE, classif.ksvm
## 1                            0.9709898                        0.6080774
## 2                            0.9764991                        0.9109715
## 3                            0.9690476                        0.9666667
## 4                            0.9767442                        0.8178295
## 5                            1.0000000                        0.9747475
## 6                            0.9702890                        0.3549044
##   FALSE, FALSE, TRUE, classif.randomForest
## 1                                0.6160410
## 2                                0.7896701
## 3                                0.8761905
## 4                                0.9186047
## 5                                0.9595960
## 6                                0.8156288
##   FALSE, FALSE, TRUE, classif.xgboost FALSE, TRUE, FALSE, classif.ksvm
## 1                           0.6427759                        0.9607509
## 2                           0.8133019                        0.9655669
## 3                           0.8333333                        0.9690476
## 4                           0.9108527                        0.9767442
## 5                           0.9040404                        1.0000000
## 6                           0.7952788                        0.9711030
##   FALSE, TRUE, FALSE, classif.randomForest
## 1                                0.9709898
## 2                                0.9768919
## 3                                0.9880952
## 4                                0.9806202
## 5                                1.0000000
## 6                                0.9706960
##   FALSE, TRUE, FALSE, classif.xgboost SMOTE, FALSE, FALSE, classif.ksvm
## 1                           0.9709898                         0.9124005
## 2                           0.9779393                         0.9476957
## 3                           0.9714286                         0.9642857
## 4                           0.9806202                         0.9767442
## 5                           1.0000000                         0.9949495
## 6                           0.9702890                         0.9682540
##   SMOTE, FALSE, FALSE, classif.randomForest
## 1                                 0.9379977
## 2                                 0.9579733
## 3                                 0.9690476
## 4                                 0.9689922
## 5                                 1.0000000
## 6                                 0.9601140
##   SMOTE, FALSE, FALSE, classif.xgboost
## 1                            0.9493743
## 2                            0.9725714
## 3                            0.9666667
## 4                            0.9573643
## 5                            1.0000000
## 6                            0.9584860
```

```r
summary(df)
```

```
##  ADASYN, FALSE, FALSE, classif.ksvm
##  Min.   :0.5458
##  1st Qu.:0.9637
##  Median :0.9729
##  Mean   :0.9641
##  3rd Qu.:0.9831
##  Max.   :1.0000
##  NA's   :2
##  ADASYN, FALSE, FALSE, classif.randomForest
##  Min.   :0.5839
##  1st Qu.:0.9672
##  Median :0.9817
##  Mean   :0.9714
##  3rd Qu.:0.9948
##  Max.   :1.0000
##  NA's   :7
##  ADASYN, FALSE, FALSE, classif.xgboost FALSE, FALSE, FALSE, classif.ksvm
##  Min.   :0.5861                        Min.   :0.9457
##  1st Qu.:0.9622                        1st Qu.:0.9709
##  Median :0.9860                        Median :0.9759
##  Mean   :0.9719                        Mean   :0.9777
##  3rd Qu.:0.9951                        3rd Qu.:0.9831
##  Max.   :1.0000                        Max.   :1.0000
##
##  FALSE, FALSE, FALSE, classif.randomForest
##  Min.   :0.9633
##  1st Qu.:0.9746
##  Median :0.9835
##  Mean   :0.9837
##  3rd Qu.:0.9937
##  Max.   :1.0000
##  NA's   :2
##  FALSE, FALSE, FALSE, classif.xgboost FALSE, FALSE, TRUE, classif.ksvm
##  Min.   :0.9633                       Min.   :0.05952
##  1st Qu.:0.9739                       1st Qu.:0.66525
##  Median :0.9846                       Median :0.94300
##  Mean   :0.9840                       Mean   :0.79717
##  3rd Qu.:0.9937                       3rd Qu.:0.97562
##  Max.   :1.0000                       Max.   :1.00000
##
##  FALSE, FALSE, TRUE, classif.randomForest
##  Min.   :0.3923
##  1st Qu.:0.8055
##  Median :0.9158
##  Mean   :0.8631
##  3rd Qu.:0.9658
##  Max.   :1.0000
##  NA's   :2
##  FALSE, FALSE, TRUE, classif.xgboost FALSE, TRUE, FALSE, classif.ksvm
##  Min.   :0.3274                      Min.   :0.9496
##  1st Qu.:0.7733                      1st Qu.:0.9708
##  Median :0.9107                      Median :0.9752
##  Mean   :0.8508                      Mean   :0.9778
##  3rd Qu.:0.9586                      3rd Qu.:0.9832
##  Max.   :1.0000                      Max.   :1.0000
##
##  FALSE, TRUE, FALSE, classif.randomForest
##  Min.   :0.9596
##  1st Qu.:0.9747
##  Median :0.9817
##  Mean   :0.9838
##  3rd Qu.:0.9934
##  Max.   :1.0000
##  NA's   :1
##  FALSE, TRUE, FALSE, classif.xgboost SMOTE, FALSE, FALSE, classif.ksvm
##  Min.   :0.9633                      Min.   :0.5447
##  1st Qu.:0.9747                      1st Qu.:0.9698
##  Median :0.9827                      Median :0.9726
##  Mean   :0.9840                      Mean   :0.9649
##  3rd Qu.:0.9936                      3rd Qu.:0.9818
##  Max.   :1.0000                      Max.   :0.9990
##
##  SMOTE, FALSE, FALSE, classif.randomForest
##  Min.   :0.5534
##  1st Qu.:0.9664
##  Median :0.9808
##  Mean   :0.9708
##  3rd Qu.:0.9937
##  Max.   :1.0000
##  NA's   :5
##  SMOTE, FALSE, FALSE, classif.xgboost
##  Min.   :0.5534
##  1st Qu.:0.9600
##  Median :0.9839
##  Mean   :0.9714
##  3rd Qu.:0.9961
##  Max.   :1.0000
##
```

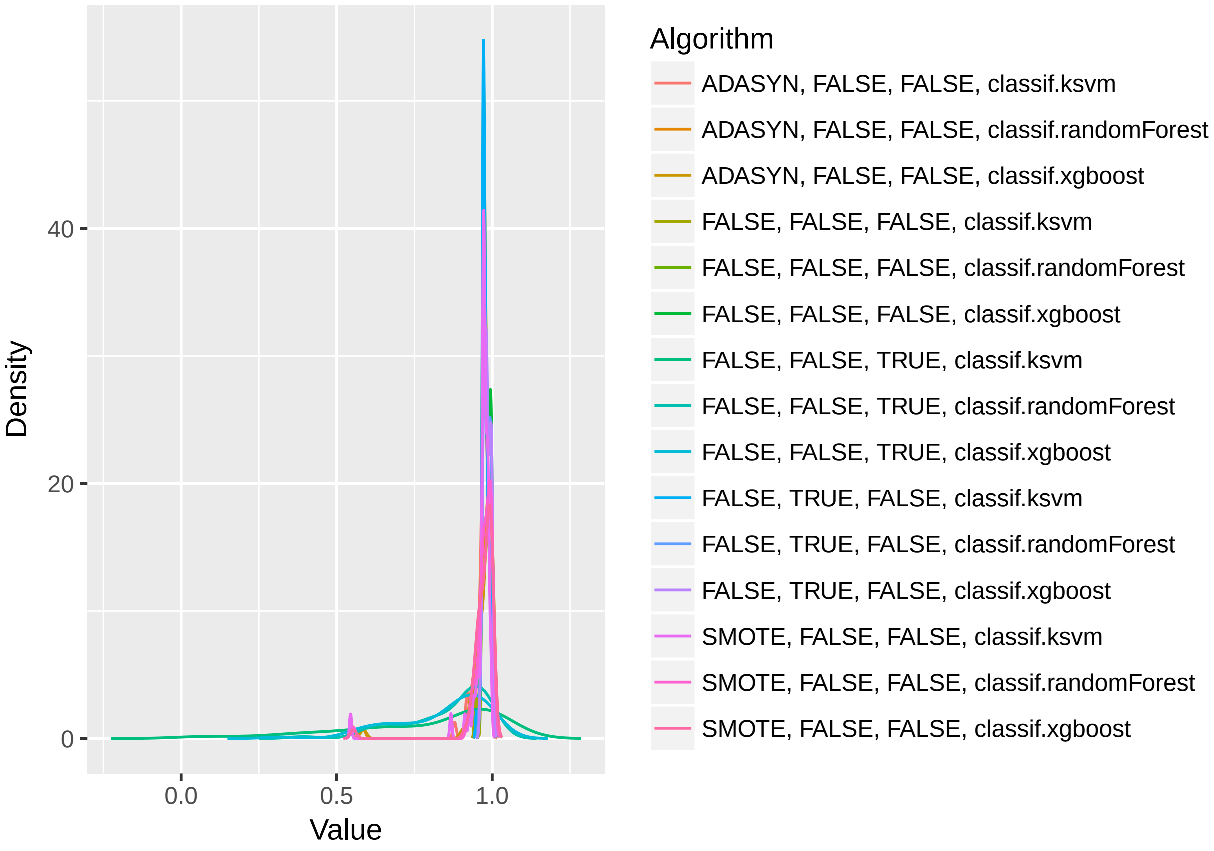
Verificando a média de cada coluna selecionada

```
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 = 0.964062348299677"
## [1] "Media da coluna ADASYN, FALSE, FALSE, classif.randomForest = 0.971404284902596"
## [1] "Media da coluna ADASYN, FALSE, FALSE, classif.xgboost = 0.971895708302867"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.ksvm = 0.97772953557104"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.randomForest = 0.983664863584591"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.xgboost = 0.984003603270289"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.ksvm = 0.797170131347226"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.randomForest = 0.863073202585973"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.xgboost = 0.850793979840824"
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.ksvm = 0.977762884889045"
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.randomForest = 0.983770098293553"
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.xgboost = 0.98402308460922"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.ksvm = 0.964911242264743"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.randomForest = 0.970753550509565"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.xgboost = 0.971398212090727"
```

Fazendo teste de normalidade

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



Testando as diferencas

```r
friedmanTest(df)
```

```
##
##  Friedman's rank sum test
##
## data:  df
## Friedman's chi-squared = 403.4, 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,]                               TRUE
##  [6,]                               TRUE
##  [7,]                              FALSE
##  [8,]                               TRUE
##  [9,]                               TRUE
## [10,]                              FALSE
## [11,]                               TRUE
## [12,]                               TRUE
## [13,]                              FALSE
## [14,]                              FALSE
## [15,]                               TRUE
##       ADASYN, FALSE, FALSE, classif.randomForest
##  [1,]                                      FALSE
##  [2,]                                      FALSE
##  [3,]                                      FALSE
##  [4,]                                      FALSE
##  [5,]                                      FALSE
##  [6,]                                       TRUE
##  [7,]                                       TRUE
##  [8,]                                       TRUE
##  [9,]                                       TRUE
## [10,]                                      FALSE
## [11,]                                      FALSE
## [12,]                                       TRUE
## [13,]                                      FALSE
## [14,]                                      FALSE
## [15,]                                      FALSE
##       ADASYN, FALSE, FALSE, classif.xgboost
##  [1,]                                  TRUE
##  [2,]                                 FALSE
##  [3,]                                 FALSE
##  [4,]                                 FALSE
##  [5,]                                 FALSE
##  [6,]                                 FALSE
##  [7,]                                  TRUE
##  [8,]                                  TRUE
##  [9,]                                  TRUE
## [10,]                                 FALSE
## [11,]                                 FALSE
## [12,]                                 FALSE
## [13,]                                  TRUE
## [14,]                                 FALSE
## [15,]                                 FALSE
##        FALSE, FALSE, FALSE, classif.ksvm
##  [1,]                             FALSE
##  [2,]                             FALSE
##  [3,]                             FALSE
##  [4,]                             FALSE
##  [5,]                             FALSE
##  [6,]                              TRUE
##  [7,]                              TRUE
##  [8,]                              TRUE
##  [9,]                              TRUE
## [10,]                             FALSE
## [11,]                             FALSE
## [12,]                              TRUE
## [13,]                             FALSE
## [14,]                             FALSE
## [15,]                             FALSE
##        FALSE, FALSE, FALSE, classif.randomForest
##  [1,]                                      TRUE
##  [2,]                                     FALSE
##  [3,]                                     FALSE
##  [4,]                                     FALSE
##  [5,]                                     FALSE
##  [6,]                                     FALSE
##  [7,]                                      TRUE
##  [8,]                                      TRUE
##  [9,]                                      TRUE
## [10,]                                     FALSE
## [11,]                                     FALSE
## [12,]                                     FALSE
## [13,]                                      TRUE
## [14,]                                     FALSE
## [15,]                                     FALSE
##        FALSE, FALSE, FALSE, classif.xgboost
##  [1,]                                 TRUE
##  [2,]                                 TRUE
##  [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, TRUE, classif.ksvm
##  [1,]                            FALSE
##  [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                            FALSE
##  [2,]                                TRUE                            FALSE
##  [3,]                                TRUE                            FALSE
##  [4,]                                TRUE                            FALSE
##  [5,]                                TRUE                            FALSE
##  [6,]                                TRUE                             TRUE
##  [7,]                               FALSE                             TRUE
##  [8,]                               FALSE                             TRUE
##  [9,]                               FALSE                             TRUE
## [10,]                                TRUE                            FALSE
## [11,]                                TRUE                            FALSE
## [12,]                                TRUE                             TRUE
## [13,]                                TRUE                            FALSE
## [14,]                                TRUE                            FALSE
## [15,]                                TRUE                            FALSE
##       FALSE, TRUE, FALSE, classif.randomForest
##  [1,]                                     TRUE
##  [2,]                                    FALSE
##  [3,]                                    FALSE
##  [4,]                                    FALSE
##  [5,]                                    FALSE
##  [6,]                                    FALSE
##  [7,]                                     TRUE
##  [8,]                                     TRUE
##  [9,]                                     TRUE
## [10,]                                    FALSE
## [11,]                                    FALSE
## [12,]                                    FALSE
## [13,]                                     TRUE
## [14,]                                    FALSE
## [15,]                                    FALSE
##       FALSE, TRUE, FALSE, classif.xgboost
##  [1,]                                TRUE
##  [2,]                                TRUE
##  [3,]                               FALSE
##  [4,]                                TRUE
##  [5,]                               FALSE
##  [6,]                               FALSE
##  [7,]                                TRUE
##  [8,]                                TRUE
##  [9,]                                TRUE
## [10,]                                TRUE
## [11,]                               FALSE
## [12,]                               FALSE
## [13,]                                TRUE
## [14,]                                TRUE
## [15,]                               FALSE
##       SMOTE, FALSE, FALSE, classif.ksvm
##  [1,]                             FALSE
##  [2,]                             FALSE
##  [3,]                              TRUE
##  [4,]                             FALSE
##  [5,]                              TRUE
##  [6,]                              TRUE
##  [7,]                              TRUE
##  [8,]                              TRUE
##  [9,]                              TRUE
## [10,]                             FALSE
## [11,]                              TRUE
## [12,]                              TRUE
## [13,]                             FALSE
## [14,]                             FALSE
## [15,]                              TRUE
##       SMOTE, FALSE, FALSE, classif.randomForest
##  [1,]                                     FALSE
##  [2,]                                     FALSE
##  [3,]                                     FALSE
##  [4,]                                     FALSE
##  [5,]                                     FALSE
##  [6,]                                     FALSE
##  [7,]                                      TRUE
##  [8,]                                      TRUE
##  [9,]                                      TRUE
## [10,]                                     FALSE
## [11,]                                     FALSE
## [12,]                                      TRUE
## [13,]                                     FALSE
## [14,]                                     FALSE
## [15,]                                     FALSE
##       SMOTE, FALSE, FALSE, classif.xgboost
##  [1,]                                 TRUE
##  [2,]                                FALSE
##  [3,]                                FALSE
##  [4,]                                FALSE
##  [5,]                                FALSE
##  [6,]                                FALSE
##  [7,]                                 TRUE
##  [8,]                                 TRUE
##  [9,]                                 TRUE
## [10,]                                FALSE
## [11,]                                FALSE
## [12,]                                FALSE
## [13,]                                 TRUE
## [14,]                                FALSE
## [15,]                                FALSE
```

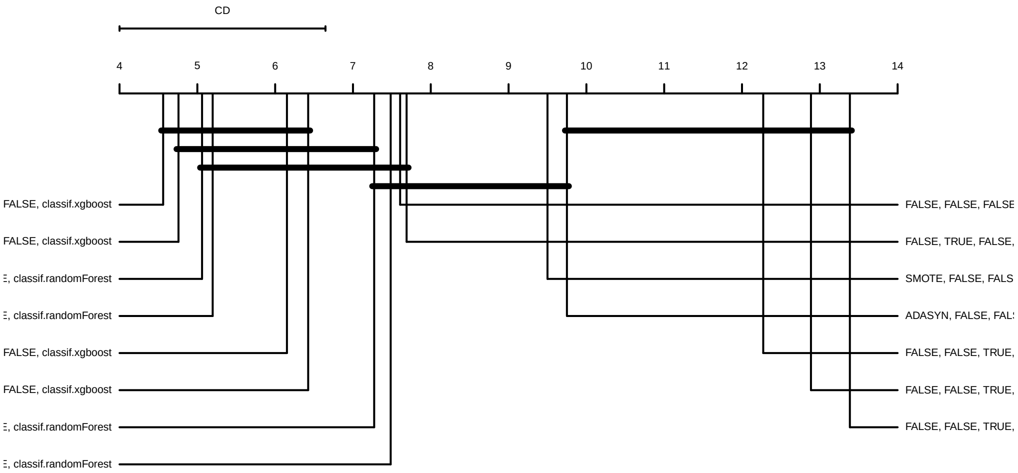
Plotando os ranks

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

```
##         ADASYN, FALSE, FALSE, classif.ksvm 
##                                   9.750000 
## ADASYN, FALSE, FALSE, classif.randomForest 
##                                   7.484848 
##      ADASYN, FALSE, FALSE, classif.xgboost 
##                                   6.151515 
##          FALSE, FALSE, FALSE, classif.ksvm 
##                                   7.606061 
##  FALSE, FALSE, FALSE, classif.randomForest 
##                                   5.196970 
##       FALSE, FALSE, FALSE, classif.xgboost 
##                                   4.757576 
##           FALSE, FALSE, TRUE, classif.ksvm 
##                                  12.272727 
##   FALSE, FALSE, TRUE, classif.randomForest 
##                                  12.886364 
##        FALSE, FALSE, TRUE, classif.xgboost 
##                                  13.386364 
##           FALSE, TRUE, FALSE, classif.ksvm 
##                                   7.689394 
##   FALSE, TRUE, FALSE, classif.randomForest 
##                                   5.060606 
##        FALSE, TRUE, FALSE, classif.xgboost 
##                                   4.560606 
##          SMOTE, FALSE, FALSE, classif.ksvm 
##                                   9.500000 
##  SMOTE, FALSE, FALSE, classif.randomForest 
##                                   7.272727 
##       SMOTE, FALSE, FALSE, classif.xgboost 
##                                   6.424242
```

Plotando grafico de Critical Diference

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


CD
FALSE, classif.xgboost
FALSE, classif.xgboost
Ξ, classif.randomForest
Ξ, classif.randomForest
FALSE, classif.xgboost
FALSE, classif.xgboost
Ξ, classif.randomForest
Ξ, classif.randomForest
FALSE, FALSE, FALSE
FALSE, TRUE, FALSE,
SMOTE, FALSE, FALS
ADASYN, FALSE, FAL!
FALSE, FALSE, TRUE,
FALSE, FALSE, TRUE,
FALSE, FALSE, TRUE,