R Notebook

Parametros:

Measure = G-mean
Columns = sampling, weight_space, underbagging
Performance = holdout_measure
Filter keys = imba.rate
Filter values = 0.05

```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_
```

```r
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

```r
ds = filter(ds, measure == params$measure)
```

Filtrando o data set

```r
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        :1230   Mode :logical
##  classif.randomForest:1230   FALSE:2952
##  classif.rusboost    :   0   TRUE :738
##  classif.xgboost     :1230   NA's :0
##
##
##
##                                 measure        sampling     underbagging
##  Accuracy                        :   0   ADASYN: 738   Mode :logical
##  Area under the curve            :   0   FALSE :2214   FALSE:2952
##  F1 measure                      :   0   SMOTE : 738   TRUE :738
##  G-mean                          :3690                 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.:0.6329    1st Qu.:0.3162    1st Qu.:0.2321
##  Median :0.9254    Median :0.7412    Median :0.5564
##  Mean   :0.7606    Mean   :0.6130    Mean   :0.5202
##  3rd Qu.:0.9872    3rd Qu.:0.9487    3rd Qu.:0.8165
##  Max.   :1.0000    Max.   :1.0000    Max.   :1.0000
##  NA's   :39        NA's   :39        NA's   :39
##  iteration_count         dataset        imba.rate
##  Min.   :1       abalone     :  45   Min.   :0.05
##  1st Qu.:1       adult       :  45   1st Qu.:0.05
##  Median :2       annealing   :  45   Median :0.05
##  Mean   :2       arrhythmia  :  45   Mean   :0.05
##  3rd Qu.:3       balance-scale:  45   3rd Qu.:0.05
##  Max.   :3       bank        :  45   Max.   :0.05
##  NA's   :39      (Other)     :3420
```

Computando as médias das iteracoes

```r
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

```r
# 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] 246    5
```

```r
# Renomeando a variavel
df = df_tec_wide_residual

head(df)
```

```
##   ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## 1           0.36347460           0.4188068          0.6449588
## 2           0.43290439           0.4940109          0.7810530
## 3           0.57108256           0.5474844          0.8696251
## 4           0.00000000           0.0000000          0.5826260
## 5           1.00000000           1.0000000          0.9973856
## 6           0.08824582           0.3978845          0.6812398
##   FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## 1          0.3765112           0.4067937
## 2          0.4757633           0.4771356
## 3          0.5487756           0.5190879
## 4          0.0000000           0.0000000
## 5          1.0000000           0.7979490
## 6          0.3094874           0.2027097
```

```r
summary(df)
```

```
##   ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
##   Min.   :0.0000       Min.   :0.0000      Min.   :0.02351
##   1st Qu.:0.3187       1st Qu.:0.1468      1st Qu.:0.70178
##   Median :0.7211       Median :0.5801      Median :0.86219
##   Mean   :0.6042       Mean   :0.5256      Mean   :0.80371
##   3rd Qu.:0.9119       3rd Qu.:0.8777      3rd Qu.:0.95777
##   Max.   :1.0000       Max.   :1.0000      Max.   :1.00000
```

```
##  NA's   :4              NA's   :1             NA's   :3
##  FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
##  Min.   :0.0000    Min.   :0.0000
##  1st Qu.:0.1491    1st Qu.:0.3569
##  Median :0.5938    Median :0.7387
##  Mean   :0.5188    Mean   :0.6141
##  3rd Qu.:0.8710    3rd Qu.:0.9275
##  Max.   :1.0000    Max.   :1.0000
##  NA's   :1         NA's   :4
```

Verificando a média de cada coluna selecionada

```r
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 = 0.604215599895028"
## [1] "Media da coluna FALSE, FALSE, FALSE = 0.525626011227214"
## [1] "Media da coluna FALSE, FALSE, TRUE = 0.803711277358415"
## [1] "Media da coluna FALSE, TRUE, FALSE = 0.518764867981824"
## [1] "Media da coluna SMOTE, FALSE, FALSE = 0.614106740016343"
```

Fazendo teste de normalidade

```r
plotDensities(data = na.omit(df))
```
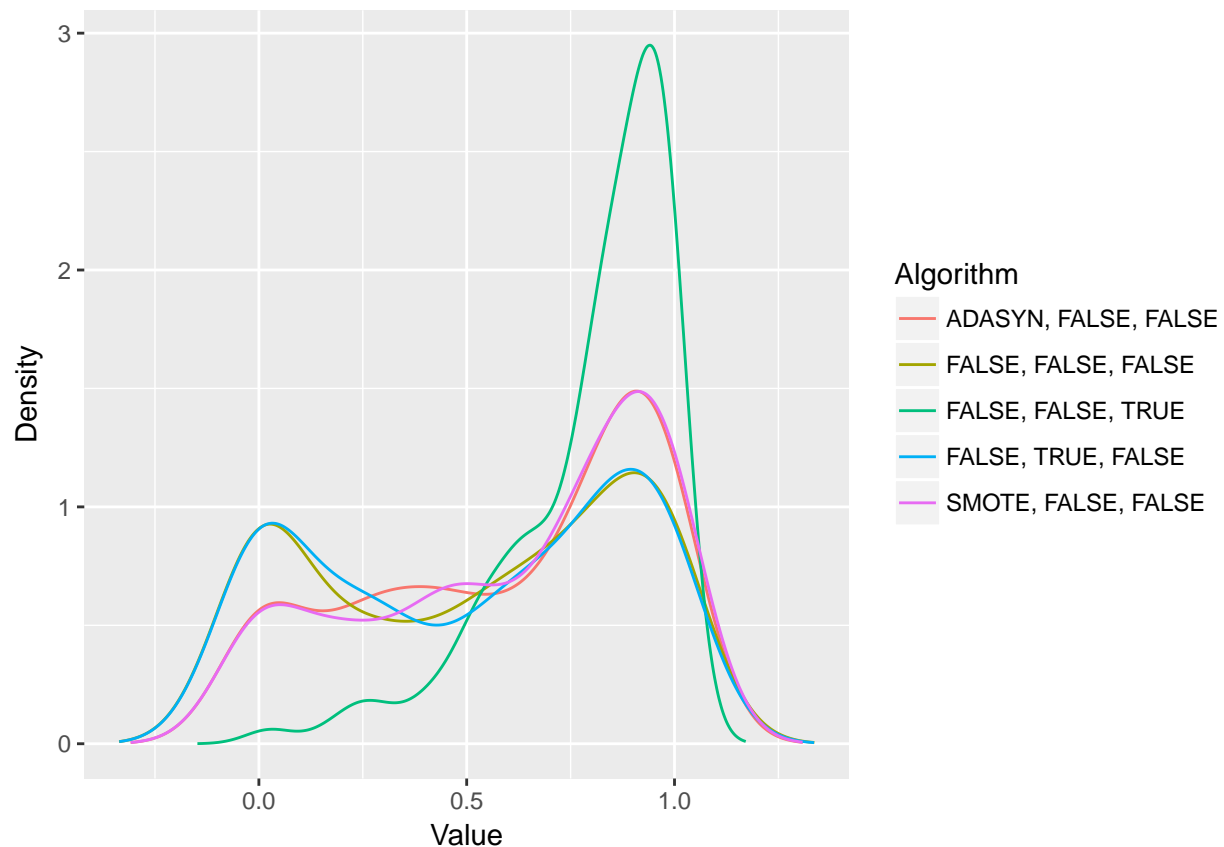
Testando as diferencas

```r
friedmanTest(df)
```

```
## 
##  Friedman's rank sum test
## 
## data:  df
## Friedman's chi-squared = 273.97, df = 4, p-value < 2.2e-16
```

Testando as diferencas par a par

```r
test <- nemenyiTest (df, alpha=0.05)
abs(test$diff.matrix) > test$statistic
```

```
##       ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## [1,]                FALSE                TRUE               TRUE
## [2,]                 TRUE               FALSE               TRUE
## [3,]                 TRUE                TRUE              FALSE
## [4,]                 TRUE               FALSE               TRUE
## [5,]                FALSE                TRUE               TRUE
##       FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## [1,]                TRUE               FALSE
```

```
## [2,]              FALSE                 TRUE
## [3,]               TRUE                 TRUE
## [4,]              FALSE                 TRUE
## [5,]               TRUE                FALSE
```

Plotando os ranks

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

```
## ADASYN, FALSE, FALSE  FALSE, FALSE, FALSE   FALSE, FALSE, TRUE
##             2.955285             3.713415             1.697154
##   FALSE, TRUE, FALSE  SMOTE, FALSE, FALSE
##             3.750000             2.884146
```

Plotando grafico de Critical Diference

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