R Notebook

Parametros:

Measure = Area under the curve
Columns = sampling, weight_space, underbagging
Performance = holdout_measure_residual
Filter keys = imba.rate
Filter values = 0.001

```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        :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          :1800  FALSE :1080   FALSE:1440
## F1 measure                    :  0   SMOTE : 360   TRUE :360
## G-mean                        :  0                 NA's :0
## Matthews correlation coefficient:  0
##
##
## tuning_measure    holdout_measure   holdout_measure_residual
## Min.   :0.3866   Min.   :0.2139   Min.   :0.3092
## 1st Qu.:0.9553   1st Qu.:0.8921   1st Qu.:0.7377
## Median :0.9993   Median :0.9916   Median :0.9069
## Mean   :0.9502   Mean   :0.9126   Mean   :0.8460
## 3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.9840
## Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
## NA's   :42       NA's   :42       NA's   :42
## 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      cardiotocography-10clases: 45   3rd Qu.:0.001
## Max.   :3      cardiotocography-3clases : 45   Max.   :0.001
## NA's   :42     (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

```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] 120    5
```

```r
# Renomeando a variavel
df = df_tec_wide_residual

head(df)
```

```
##   ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## 1            0.5136216           0.5133720          0.6259505
## 2                   NA           0.4549243          0.7943273
## 3            0.7583631           0.7469161          0.7440871
## 4            0.9337170           0.8590001          0.8088235
## 5            0.9224985           0.9703766          0.6252906
## 6            0.8857046           0.8672659          0.6192346
##   FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## 1          0.5137510           0.5167415
## 2          0.5371265           0.6257369
## 3          0.7469161           0.7616514
## 4          0.8560772           0.9305505
## 5          0.9703766           0.9551922
## 6          0.8672659           0.8325538
```

```r
summary(df)
```

```
##   ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
##   Min.   :0.4882       Min.   :0.4549      Min.   :0.5009
##   1st Qu.:0.7540       1st Qu.:0.7435      1st Qu.:0.6788
##   Median :0.9012       Median :0.9023      Median :0.8722
##   Mean   :0.8544       Mean   :0.8534      Mean   :0.8269
##   3rd Qu.:0.9799       3rd Qu.:0.9833      3rd Qu.:0.9675
##   Max.   :1.0000       Max.   :1.0000      Max.   :1.0000
```

```
##  NA's   :9
##  FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
##  Min.   :0.4988     Min.   :0.4858
##  1st Qu.:0.7536     1st Qu.:0.7096
##  Median :0.9037     Median :0.8837
##  Mean   :0.8570     Mean   :0.8393
##  3rd Qu.:0.9857     3rd Qu.:0.9791
##  Max.   :1.0000     Max.   :1.0000
##  NA's   :3          NA's   :2
```

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 = 0.854354200830927"
## [1] "Media da coluna FALSE, FALSE, FALSE = 0.853390336521365"
## [1] "Media da coluna FALSE, FALSE, TRUE = 0.826880532245444"
## [1] "Media da coluna FALSE, TRUE, FALSE = 0.857004086650482"
## [1] "Media da coluna SMOTE, FALSE, FALSE = 0.839300193536231"
```

Fazendo teste de normalidade

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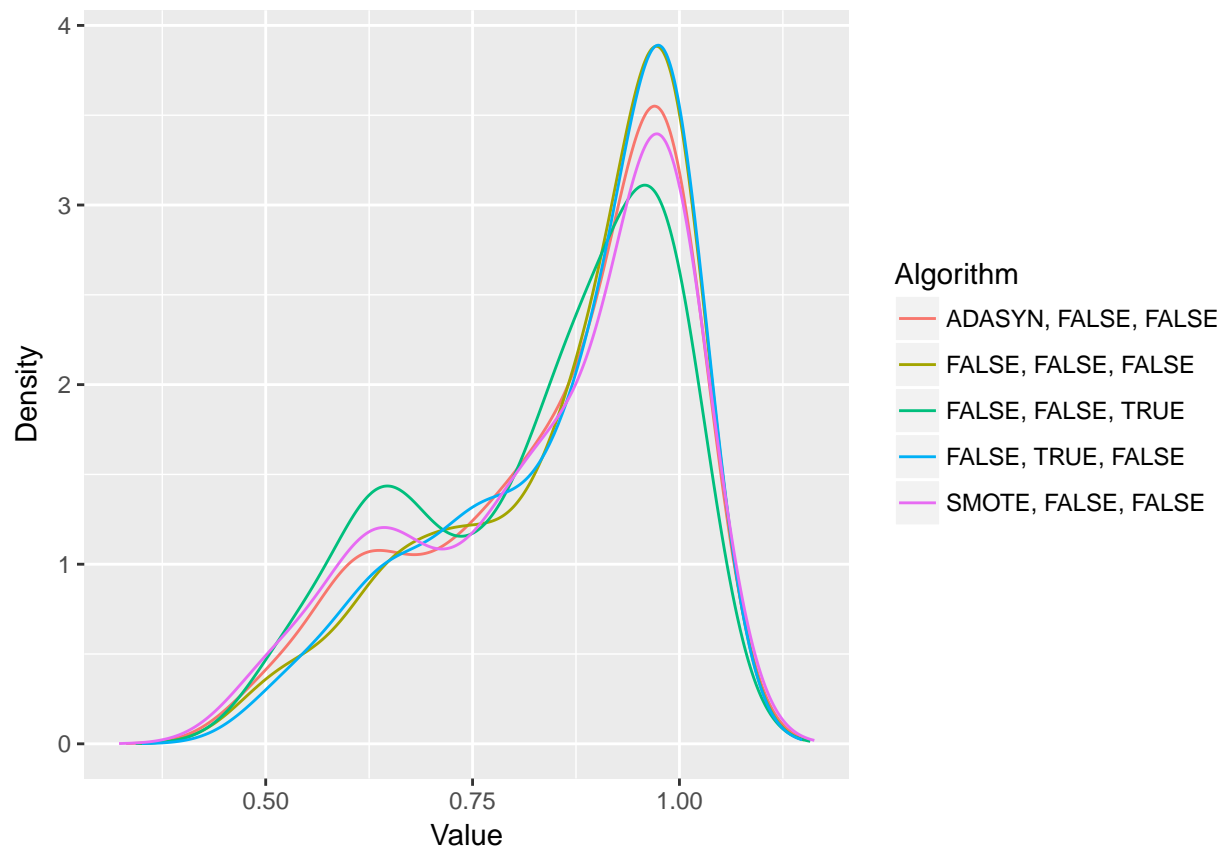
Testando as diferencas

```
friedmanTest(df)
```

```
##
##  Friedman's rank sum test
##
## data:  df
## Friedman's chi-squared = 27.392, df = 4, p-value = 1.656e-05
```

Testando as diferencas par a par

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

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

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

Plotando os ranks

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

```
## ADASYN, FALSE, FALSE  FALSE, FALSE, FALSE    FALSE, FALSE, TRUE
##            3.095833             2.625000              3.550000
##   FALSE, TRUE, FALSE  SMOTE, FALSE, FALSE
##            2.662500             3.066667
```

Plotando grafico de Critical Diference

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