R Notebook

Parametros:

Measure = Area under the curve
Columns = sampling, weight_space, underbagging
Performance = holdout_measure
Filter keys = NULL
Filter values = NULL

```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.kknn        :17100   Mode :logical
##  classif.ksvm        :17100   FALSE:68400
##  classif.randomForest:17100   TRUE :17100
##  classif.rpart       :17100   NA's :0
##  classif.rusboost    :    0
##  classif.xgboost     :17100
##
##                             measure        sampling     underbagging
##  Accuracy                       :17100   ADASYN:17100   Mode :logical
##  Area under the curve           :17100   FALSE :51300   FALSE:68400
##  F1 measure                     :17100   SMOTE :17100   TRUE :17100
##  G-mean                         :17100                  NA's :0
##  Matthews correlation coefficient:17100
##
##
##  tuning_measure   holdout_measure  holdout_measure_residual
##  Min.   :-0.128   Min.   :-0.212   Min.   :-0.466
##  1st Qu.: 0.738   1st Qu.: 0.500   1st Qu.: 0.275
##  Median : 0.969   Median : 0.890   Median : 0.603
##  Mean   : 0.810   Mean   : 0.708   Mean   : 0.562
##  3rd Qu.: 0.996   3rd Qu.: 0.990   3rd Qu.: 0.886
##  Max.   : 1.000   Max.   : 1.000   Max.   : 1.000
##  NA's   :23802    NA's   :23802    NA's   :23802
##  iteration_count                      dataset         imba.rate
##  Min.   :1       abalone                  : 1500   Min.   :0.0010
##  1st Qu.:1       adult                    : 1500   1st Qu.:0.0100
##  Median :2       bank                     : 1500   Median :0.0300
##  Mean   :2       car                      : 1500   Mean   :0.0286
##  3rd Qu.:3       cardiotocography-10clases: 1500   3rd Qu.:0.0500
##  Max.   :3       cardiotocography-3clases : 1500   Max.   :0.0500
##  NA's   :23802   (Other)                  :76500
```

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.kknn        :3420   Mode :logical
##  classif.ksvm        :3420   FALSE:13680
##  classif.randomForest:3420   TRUE :3420
##  classif.rpart       :3420   NA's :0
##  classif.rusboost    :   0
##  classif.xgboost     :3420
##
##                                 measure        sampling     underbagging
##  Accuracy                        :    0   ADASYN: 3420   Mode :logical
##  Area under the curve            :17100   FALSE :10260   FALSE:13680
##  F1 measure                      :    0   SMOTE : 3420   TRUE :3420
##  G-mean                          :    0                  NA's :0
##  Matthews correlation coefficient:    0
##
##
##  tuning_measure   holdout_measure  holdout_measure_residual
##  Min.   :0.3023   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.8748   1st Qu.:0.7590   1st Qu.:0.6323
##  Median :0.9851   Median :0.9545   Median :0.8362
##  Mean   :0.9021   Mean   :0.8608   Mean   :0.7891
##  3rd Qu.:0.9995   3rd Qu.:0.9986   3rd Qu.:0.9670
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##  NA's   :1023     NA's   :1023     NA's   :1023
##  iteration_count                      dataset        imba.rate
##  Min.   :1       abalone                  :  300   Min.   :0.0010
##  1st Qu.:1       adult                    :  300   1st Qu.:0.0100
##  Median :2       bank                     :  300   Median :0.0300
##  Mean   :2       car                      :  300   Mean   :0.0286
##  3rd Qu.:3       cardiotocography-10clases:  300   3rd Qu.:0.0500
##  Max.   :3       cardiotocography-3clases :  300   Max.   :0.0500
##  NA's   :1023    (Other)                  :15300
```

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

```
# 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] 1140    5
```

```
# Renomeando a variavel
df = df_tec_wide_residual

head(df)
```

```
##   ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## 1            0.4720217           0.4930806          0.5865824
## 2            0.4720217           0.4930806          0.5865824
## 3            0.5707640           0.4945725          0.6588959
## 4            0.5663743           0.5347953          0.6511111
## 5            0.5729312           0.5678789          0.8082364
## 6            0.5729312           0.5678789          0.8082364
##   FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## 1                 NA           0.4717208
## 2                 NA           0.4717208
## 3                 NA           0.5192632
## 4                 NA           0.5675439
## 5                 NA           0.5530302
## 6                 NA           0.5530302
```

```
summary(df)
```

```
##  ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
##  Min.   :0.3435       Min.   :0.2924      Min.   :0.3576
##  1st Qu.:0.7669       1st Qu.:0.7198      1st Qu.:0.8291
##  Median :0.9337       Median :0.9167      Median :0.9524
##  Mean   :0.8663       Mean   :0.8444      Mean   :0.8836
##  3rd Qu.:0.9944       3rd Qu.:0.9975      3rd Qu.:0.9946
##  Max.   :1.0000       Max.   :1.0000      Max.   :1.0000
##  NA's   :48           NA's   :9           NA's   :24
##  FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
##  Min.   :0.3333     Min.   :0.2679
##  1st Qu.:0.6885     1st Qu.:0.7708
##  Median :0.9390     Median :0.9353
##  Mean   :0.8379     Mean   :0.8678
##  3rd Qu.:0.9981     3rd Qu.:0.9963
##  Max.   :1.0000     Max.   :1.0000
##  NA's   :242        NA's   :18
```

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 = 0.866349174850206"
## [1] "Media da coluna FALSE, FALSE, FALSE = 0.844350007779144"
## [1] "Media da coluna FALSE, FALSE, TRUE = 0.883594161882493"
## [1] "Media da coluna FALSE, TRUE, FALSE = 0.837889564519326"
## [1] "Media da coluna SMOTE, FALSE, FALSE = 0.867797922220711"
```

Fazendo teste de normalidade

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

Testando as diferencas

```
friedmanTest(df)
```

```
##
##  Friedman's rank sum test
##
## data:  df
## Friedman's chi-squared = 153.84, df = 4, 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 FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## [1,]                FALSE               FALSE              FALSE
## [2,]                FALSE               FALSE              FALSE
## [3,]                FALSE               FALSE              FALSE
## [4,]                 TRUE                TRUE               TRUE
## [5,]                FALSE               FALSE              FALSE
##      FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## [1,]               TRUE               FALSE
## [2,]               TRUE               FALSE
## [3,]               TRUE               FALSE
## [4,]              FALSE                TRUE
## [5,]               TRUE               FALSE
```

Plotando os ranks

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

```
## ADASYN, FALSE, FALSE  FALSE, FALSE, FALSE   FALSE, FALSE, TRUE
##             2.860526             2.942982             2.858772
##   FALSE, TRUE, FALSE  SMOTE, FALSE, FALSE
##             3.513596             2.824123
```

Plotando grafico de Critical Diference

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