R Notebook

Parametros:

Measure = F1 measure
Columns = sampling, weight_space, underbagging
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

```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        :990   Mode :logical
##  classif.randomForest:990   FALSE:2376
##  classif.rusboost    :  0   TRUE :594
##  classif.xgboost     :990   NA's :0
##
##
##
##                                 measure      sampling    underbagging
##  Accuracy                        :  0   ADASYN: 594   Mode :logical
##  Area under the curve            :  0   FALSE :1782   FALSE:2376
##  F1 measure                      :2970  SMOTE : 594   TRUE :594
##  G-mean                          :  0                 NA's :0
##  Matthews correlation coefficient:  0
##
##
##  tuning_measure    holdout_measure   holdout_measure_residual
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.2788   1st Qu.:0.0481   1st Qu.:0.04815
##  Median :0.8296   Median :0.4840   Median :0.28571
##  Mean   :0.6542   Mean   :0.4646   Mean   :0.37464
##  3rd Qu.:0.9927   3rd Qu.:0.8000   3rd Qu.:0.70061
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
##  NA's   :51       NA's   :51       NA's   :51
##  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   :51     (Other)      :2700
```

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] 198    5
```

```r
# Renomeando a variavel
df = df_tec_wide_residual

head(df)
```

```
##    ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## 1           0.03849057          0.04164767         0.08464646
## 2           0.11510026          0.16774773         0.17717435
## 3           0.16190476          0.30158730         0.30875817
## 4           0.00000000          0.00000000         0.07407407
## 5           0.66666667          1.00000000         0.68888889
## 6           0.11827957          0.02222222         0.07426864
##    FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## 1          0.06740848          0.05409754
## 2          0.17763202          0.13195365
## 3          0.30158730          0.20634921
## 4          0.00000000          0.00000000
## 5          1.00000000          0.88888889
## 6          0.02222222          0.00000000
```

```r
summary(df)
```

```
##   ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
##   Min.   :0.0000      Min.   :0.00000     Min.   :0.0441
##   1st Qu.:0.1273      1st Qu.:0.06061     1st Qu.:0.1436
##   Median :0.5277      Median :0.52381     Median :0.3369
##   Mean   :0.5003      Mean   :0.47288     Mean   :0.3924
##   3rd Qu.:0.8667      3rd Qu.:0.83333     3rd Qu.:0.6119
##   Max.   :1.0000      Max.   :1.00000     Max.   :1.0000
```

```
## NA's   :8              NA's   :1             NA's   :2
## FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## Min.   :0.00000    Min.   :0.0000
## 1st Qu.:0.06061    1st Qu.:0.1249
## Median :0.47248    Median :0.5333
## Mean   :0.46572    Mean   :0.4932
## 3rd Qu.:0.81824    3rd Qu.:0.8233
## Max.   :1.00000    Max.   :1.0000
## NA's   :1          NA's   :5
```

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 = 0.500291926747327"
## [1] "Media da coluna FALSE, FALSE, FALSE = 0.472884421683797"
## [1] "Media da coluna FALSE, FALSE, TRUE = 0.392438224809691"
## [1] "Media da coluna FALSE, TRUE, FALSE = 0.465719213873104"
## [1] "Media da coluna SMOTE, FALSE, FALSE = 0.493163354586734"
```

Fazendo teste de normalidade

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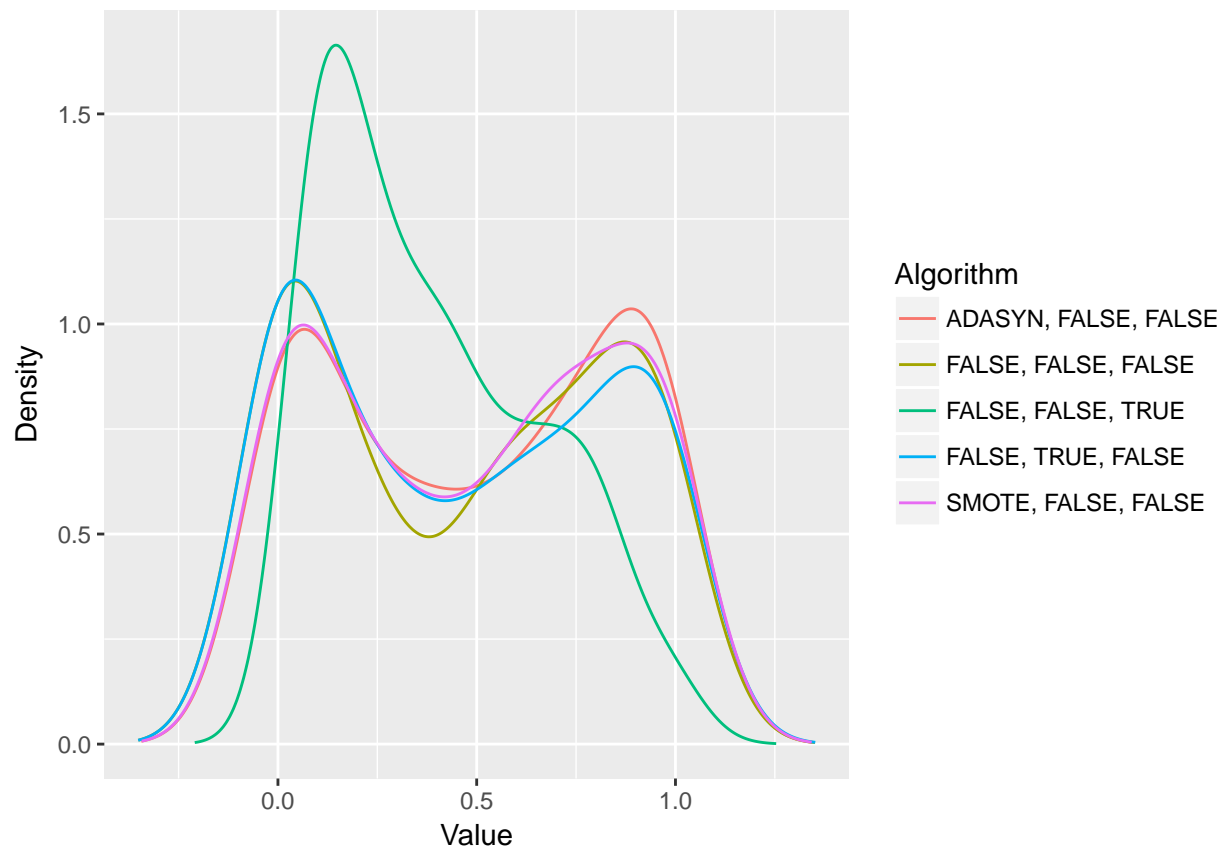
Testando as diferencas

```r
friedmanTest(df)
```

```
##
##  Friedman's rank sum test
##
## data:  df
## Friedman's chi-squared = 23.28, df = 4, p-value = 0.0001113
```

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

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

Plotando os ranks

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

```
## ADASYN, FALSE, FALSE  FALSE, FALSE, FALSE   FALSE, FALSE, TRUE
##           2.782828             3.078283             3.414141
##   FALSE, TRUE, FALSE  SMOTE, FALSE, FALSE
##           2.987374             2.737374
```

Plotando grafico de Critical Diference

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