R Notebook

Parametros:

Measure = F1 measure
Columns = sampling, weight_space, underbagging
Performance = tuning_measure
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
```

```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        :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          :  0    FALSE :1080    FALSE:1440
## F1 measure                    :1800   SMOTE : 360    TRUE :360
## 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.:0.1444   1st Qu.:0.0000   1st Qu.:0.02254
## Median :0.8072   Median :0.3333   Median :0.21133
## Mean   :0.6196   Mean   :0.4116   Mean   :0.32573
## 3rd Qu.:0.9987   3rd Qu.:0.8000   3rd Qu.:0.59294
## Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
## NA's   :60       NA's   :60       NA's   :60
## 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   :60   (Other)                  :1530
```

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)
```

2

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            0.9855714          0.00000000          0.02691361
## 2            0.9920578          0.03745141          0.06037610
## 3            0.9988447          0.00000000          0.03542331
## 4            1.0000000          0.82222222          0.82892416
## 5            1.0000000          0.50176367          0.34858986
## 6            0.9986680          0.61022928          0.85101010
##   FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## 1         0.00000000           0.9847324
## 2         0.04852524           0.9930476
## 3         0.00000000           0.9986867
## 4         0.82222222           1.0000000
## 5         0.50176367           1.0000000
## 6         0.61022928           0.9986613
```

```r
summary(df)
```

```
##  ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
##  Min.   :0.6463       Min.   :0.00000     Min.   :0.02172
##  1st Qu.:0.9955       1st Qu.:0.04444     1st Qu.:0.07650
##  Median :0.9986       Median :0.41270     Median :0.17363
##  Mean   :0.9932       Mean   :0.43227     Mean   :0.28773
##  3rd Qu.:0.9996       3rd Qu.:0.74788     3rd Qu.:0.41346
##  Max.   :1.0000       Max.   :1.00000     Max.   :1.00000
```

```
##  NA's   :9            NA's   :2           NA's   :1
##  FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
##  Min.   :0.00000    Min.   :0.9262
##  1st Qu.:0.04444    1st Qu.:0.9965
##  Median :0.40291    Median :0.9987
##  Mean   :0.43077    Mean   :0.9966
##  3rd Qu.:0.74115    3rd Qu.:0.9997
##  Max.   :1.00000    Max.   :1.0000
##  NA's   :1          NA's   :7
```

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 = 0.993179274666809"
## [1] "Media da coluna FALSE, FALSE, FALSE = 0.432274005647669"
## [1] "Media da coluna FALSE, FALSE, TRUE = 0.287725569828617"
## [1] "Media da coluna FALSE, TRUE, FALSE = 0.430767968344781"
## [1] "Media da coluna SMOTE, FALSE, FALSE = 0.996590929254598"
```

Fazendo teste de normalidade

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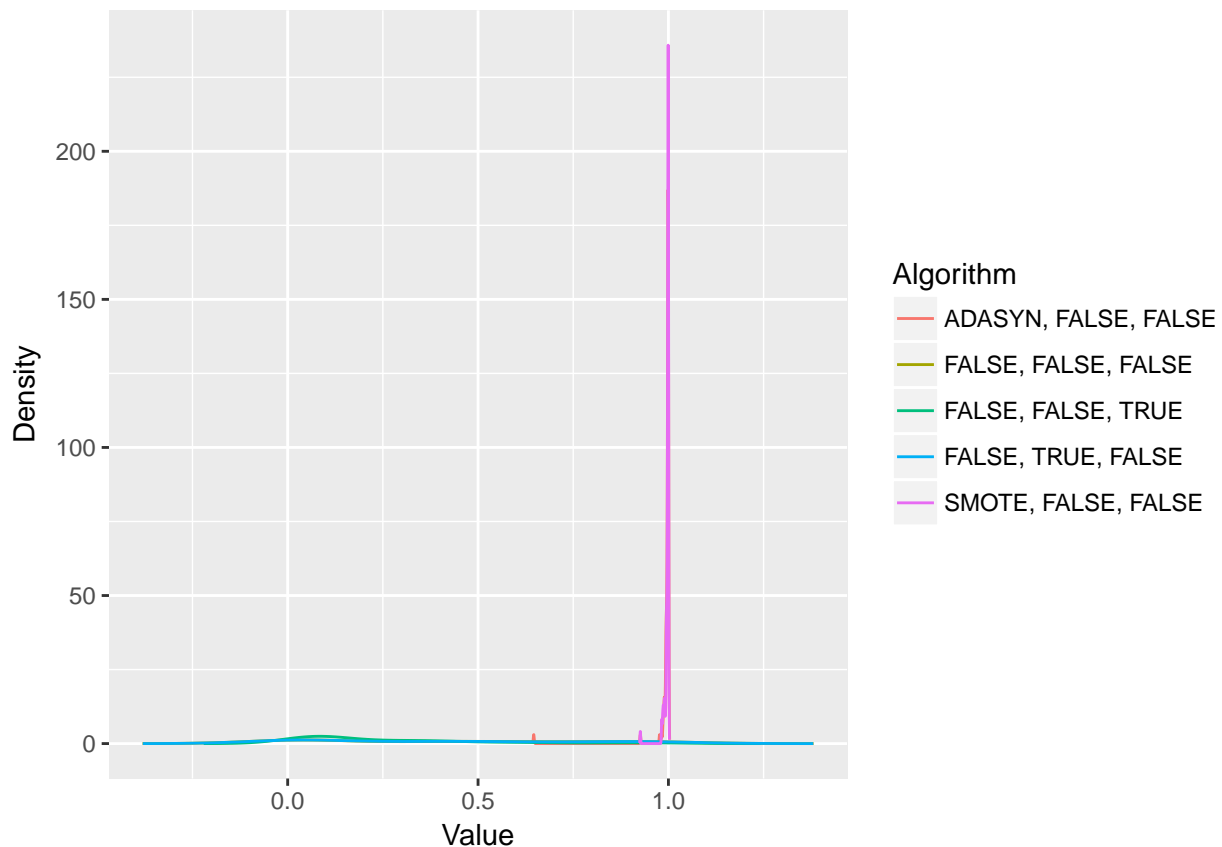
Testando as diferencas

```r
friedmanTest(df)
```

```
## 
##  Friedman's rank sum test
## 
## data:  df
## Friedman's chi-squared = 245.34, 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              FALSE
## [3,]                 TRUE               FALSE              FALSE
## [4,]                 TRUE               FALSE              FALSE
## [5,]                FALSE                TRUE               TRUE
##       FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## [1,]                TRUE               FALSE
```

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

Plotando os ranks

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

```
## ADASYN, FALSE, FALSE  FALSE, FALSE, FALSE    FALSE, FALSE, TRUE
##            1.862500              3.654167              4.062500
##   FALSE, TRUE, FALSE  SMOTE, FALSE, FALSE
##            3.733333              1.687500
```

Plotando grafico de Critical Diference

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