R Notebook

Parametros:

Measure = Area under the curve
Columns = sampling, weight_space, underbagging
Performance = tuning_measure
Filter keys = imba.rate
Filter values = 0.01

```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

```
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.:0.9529    1st Qu.:0.8909    1st Qu.:0.7392
##  Median :0.9993    Median :0.9916    Median :0.9067
##  Mean   :0.9498    Mean   :0.9120    Mean   :0.8469
##  3rd Qu.:1.0000    3rd Qu.:1.0000    3rd Qu.:0.9842
##  Max.   :1.0000    Max.   :1.0000    Max.   :1.0000
##  NA's   :51        NA's   :51        NA's   :51
##  iteration_count                        dataset       imba.rate
##  Min.   :1      abalone                   : 45   Min.   :0.01
##  1st Qu.:1      adult                     : 45   1st Qu.:0.01
##  Median :2      bank                      : 45   Median :0.01
##  Mean   :2      car                       : 45   Mean   :0.01
##  3rd Qu.:3      cardiotocography-10clases: 45   3rd Qu.:0.01
##  Max.   :3      cardiotocography-3clases : 45   Max.   :0.01
##  NA's   :51     (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            0.9965924           0.5777335          0.6095754
## 2                   NA           0.6022761          0.8041582
## 3            0.9999865           0.7565083          0.7367271
## 4            1.0000000           1.0000000          0.9634617
## 5            1.0000000           0.9757015          0.8600168
## 6            1.0000000           0.9996823          0.9310388
##    FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## 1           0.5776046           0.9977386
## 2           0.6355048           0.9974304
## 3           0.7565083           0.9999889
## 4           1.0000000           1.0000000
## 5           0.9757015           1.0000000
## 6           0.9996823           1.0000000
```

```r
summary(df)
```

```
##   ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
##   Min.   :0.9966       Min.   :0.5577      Min.   :0.4862
##   1st Qu.:0.9999       1st Qu.:0.8717      1st Qu.:0.8714
##   Median :1.0000       Median :0.9849      Median :0.9546
##   Mean   :0.9998       Mean   :0.9204      Mean   :0.9142
##   3rd Qu.:1.0000       3rd Qu.:0.9991      3rd Qu.:0.9942
##   Max.   :1.0000       Max.   :1.0000      Max.   :1.0000
```

```
## NA's  :9              NA's  :1            NA's  :1
## FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## Min.  :0.5350    Min.  :0.9974
## 1st Qu.:0.8726    1st Qu.:0.9999
## Median :0.9880    Median :1.0000
## Mean  :0.9191    Mean  :0.9998
## 3rd Qu.:0.9989    3rd Qu.:1.0000
## Max.  :1.0000    Max.  :1.0000
## NA's  :2          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 = 0.999838628793152"
## [1] "Media da coluna FALSE, FALSE, FALSE = 0.920362357147077"
## [1] "Media da coluna FALSE, FALSE, TRUE = 0.914221993957975"
## [1] "Media da coluna FALSE, TRUE, FALSE = 0.9191376935576"
## [1] "Media da coluna SMOTE, FALSE, FALSE = 0.999835425433277"
```

Fazendo teste de normalidade

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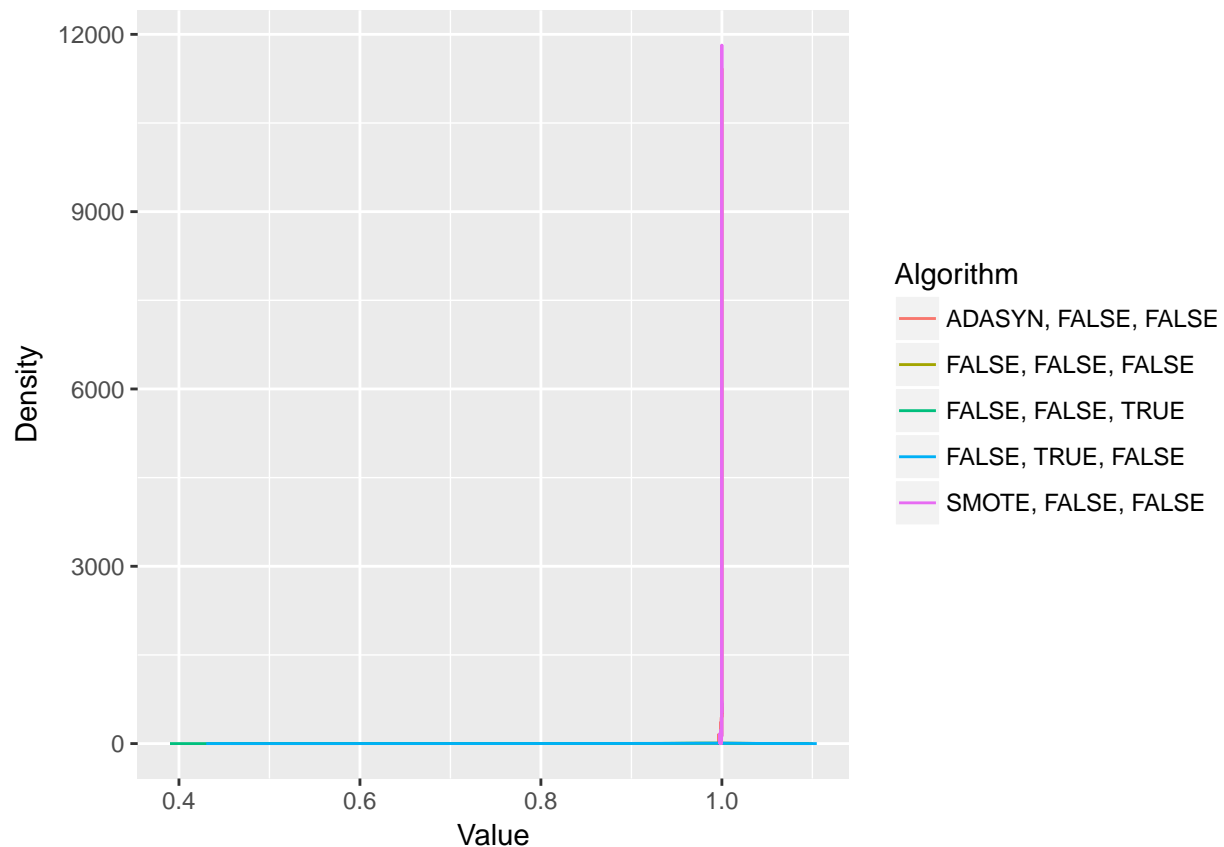
Testando as diferencas

```
friedmanTest(df)
```

```
##
##  Friedman's rank sum test
##
## data:  df
## Friedman's chi-squared = 223.83, 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                TRUE               TRUE
## [2,]                 TRUE               FALSE               TRUE
## [3,]                 TRUE                TRUE              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

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

```
## ADASYN, FALSE, FALSE  FALSE, FALSE, FALSE   FALSE, FALSE, TRUE
##           1.941667             3.520833             4.112500
##   FALSE, TRUE, FALSE  SMOTE, FALSE, FALSE
##           3.679167             1.745833
```

Plotando grafico de Critical Diference

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