R Notebook

Parametros:

Measure = Accuracy
Columns = sampling, weight_space, ruspool
Performance = holdout_measure_residual
Filter keys = imba.rate
Filter values = 0.01

```
library("scmamp")
library(dplyr)
```

Tratamento dos dados

Carregando data set compilado

```
ds = read.csv("/home/rodrigo/Dropbox/UNICAMP/IC/estudo_cost_learning/SummaryResults/summary_compilation_
summary(ds)
```

```
##                  learner      weight_space                                  measure        sampling
##  classif.ksvm        :17100   Mode :logical   Accuracy                          :10260   ADASYN:10260
##  classif.randomForest:17100   FALSE:41040     Area under the curve              :10260   FALSE :30780
##  classif.xgboost     :17100   TRUE :10260     F1 measure                        :10260   SMOTE :10260
##                               NA's :0         G-mean                            :10260
##                                               Matthews correlation coefficient:10260
##
##
##  tuning_measure    holdout_measure   holdout_measure_residual iteration_count                      da
##  Min.   :-0.1277   Min.   :-0.2120   Min.   :-0.4658          Min.   :1       abalone
##  1st Qu.: 0.5924   1st Qu.: 0.3114   1st Qu.: 0.1648          1st Qu.:1       adult
##  Median : 0.9624   Median : 0.8193   Median : 0.5192          Median :2       bank
##  Mean   : 0.7570   Mean   : 0.6469   Mean   : 0.5099          Mean   :2       car
##  3rd Qu.: 0.9965   3rd Qu.: 0.9879   3rd Qu.: 0.8636          3rd Qu.:3       cardiotocography-10cla
##  Max.   : 1.0000   Max.   : 1.0000   Max.   : 1.0000          Max.   :3       cardiotocography-3clas
##  NA's   :1761      NA's   :1761      NA's   :1761             NA's   :1761    (Other)
```

Filtrando pela metrica

```
ds = filter(ds, measure == params$measure)
```

Filtrando o data set

```
if(params$filter_keys != 'NULL' && !is.null(params$filter_keys)){
  ds = filter_at(ds, .vars = params$filter_keys, .vars_predicate = any_vars(. == params$filter_values))
}

summary(ds)
```

```
##                  learner    weight_space                                measure       sampling     ru
##  classif.ksvm        :600   Mode :logical   Accuracy                        :1800   ADASYN: 360   Mo
##  classif.randomForest:600   FALSE:1440      Area under the curve            :   0   FALSE :1080   FA
##  classif.xgboost     :600   TRUE :360       F1 measure                      :   0   SMOTE : 360   TR
```

```
##                              NA's :0         G-mean                          :   0                 NA
##                                              Matthews correlation coefficient:   0
##
##
##  holdout_measure   holdout_measure_residual iteration_count                     dataset       imba.
##  Min.   :0.01517   Min.   :0.03881          Min.   :1       abalone                  :  45   Min.
##  1st Qu.:0.98875   1st Qu.:0.35486          1st Qu.:1       adult                    :  45   1st Qu.
##  Median :0.99090   Median :0.74876          Median :2       bank                     :  45   Median
##  Mean   :0.96720   Mean   :0.66291          Mean   :2       car                      :  45   Mean
##  3rd Qu.:0.99632   3rd Qu.:0.95226          3rd Qu.:3       cardiotocography-10clases:  45   3rd Qu.
##  Max.   :1.00000   Max.   :1.00000          Max.   :3       cardiotocography-3clases :  45   Max.
##  NA's   :99        NA's   :99               NA's   :99      (Other)                  :1530
```

Computando as médias das iteracoes

```
ds = group_by(ds, learner , weight_space , measure , sampling , ruspool , 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$performa

# 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
```

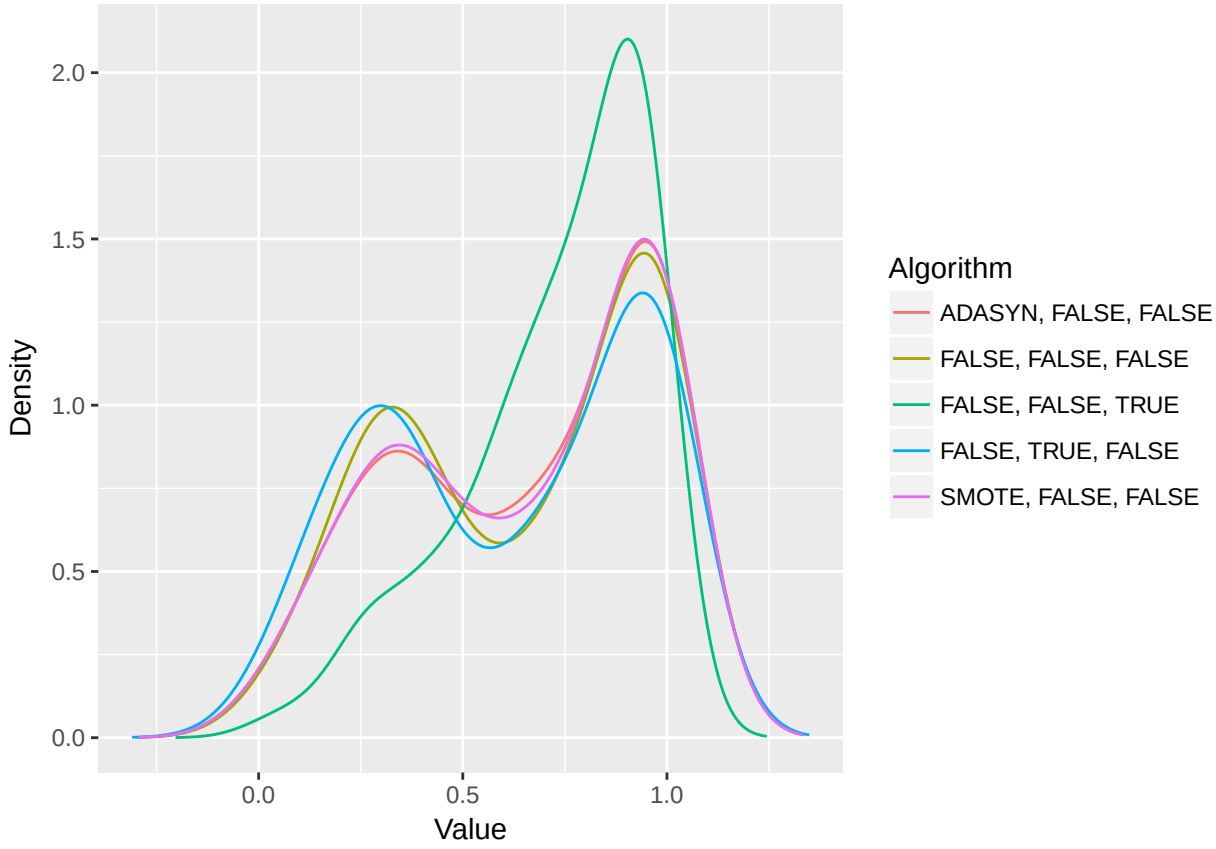
```
# Renomeando a variavel
df = df_tec_wide_residual

summary(df)
```

```
##  ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
##  Min.   :0.03881      Min.   :0.03881     Min.   :0.04134    Min.   :0.03881    Min.   :0.03881
##  1st Qu.:0.38696      1st Qu.:0.32630     1st Qu.:0.58747    1st Qu.:0.31072    1st Qu.:0.34745
##  Median :0.75989      Median :0.71457     Median :0.79015    Median :0.69068    Median :0.73045
```

```
##  Mean   :0.67283      Mean   :0.64148      Mean   :0.72943      Mean   :0.61905      Mean   :0.65197
##  3rd Qu.:0.97460      3rd Qu.:0.97165      3rd Qu.:0.92982      3rd Qu.:0.93255      3rd Qu.:0.97356
##  Max.   :0.99989      Max.   :0.99991      Max.   :0.99984      Max.   :0.99995      Max.   :0.99992
##  NA's   :13           NA's   :2            NA's   :3            NA's   :3            NA's   :12
```

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 = 31.845, df = 4, p-value = 2.058e-06
```

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

Plotando grafico de Critical Diference

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