R Notebook

Parametros:

Measure = Area under the curve
Columns = sampling, weight_space, ruspool
Performance = holdout_measure
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
##  classif.ksvm        :17100   Mode :logical
##  classif.randomForest:17100   FALSE:41040
##  classif.xgboost     :17100   TRUE :10260
##                               NA's :0
##
##
##
##                              measure        sampling      ruspool
##  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.: 0.5924   1st Qu.: 0.3114   1st Qu.: 0.1648
##  Median : 0.9624   Median : 0.8193   Median : 0.5192
##  Mean   : 0.7570   Mean   : 0.6469   Mean   : 0.5099
##  3rd Qu.: 0.9965   3rd Qu.: 0.9879   3rd Qu.: 0.8636
##  Max.   : 1.0000   Max.   : 1.0000   Max.   : 1.0000
##  NA's   :1761      NA's   :1761      NA's   :1761
##  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   :1761    (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)){
  ds = filter_at(ds, .vars = params$filter_keys, .vars_predicate = any_vars(. == params$filter_values))
}

summary(ds)
```

```
##                  learner    weight_space
##  classif.ksvm        :600   Mode :logical
##  classif.randomForest:600   FALSE:1440
##  classif.xgboost     :600   TRUE :360
##                             NA's :0
##
##
##
##                              measure       sampling     ruspool
##  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.   :0.2479   Min.   :0.3092
##  1st Qu.:0.9151   1st Qu.:0.8472   1st Qu.:0.6828
##  Median :0.9986   Median :0.9848   Median :0.8893
##  Mean   :0.9230   Mean   :0.8873   Mean   :0.8247
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.9815
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##  NA's   :87       NA's   :87       NA's   :87
##  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   :87      (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
```

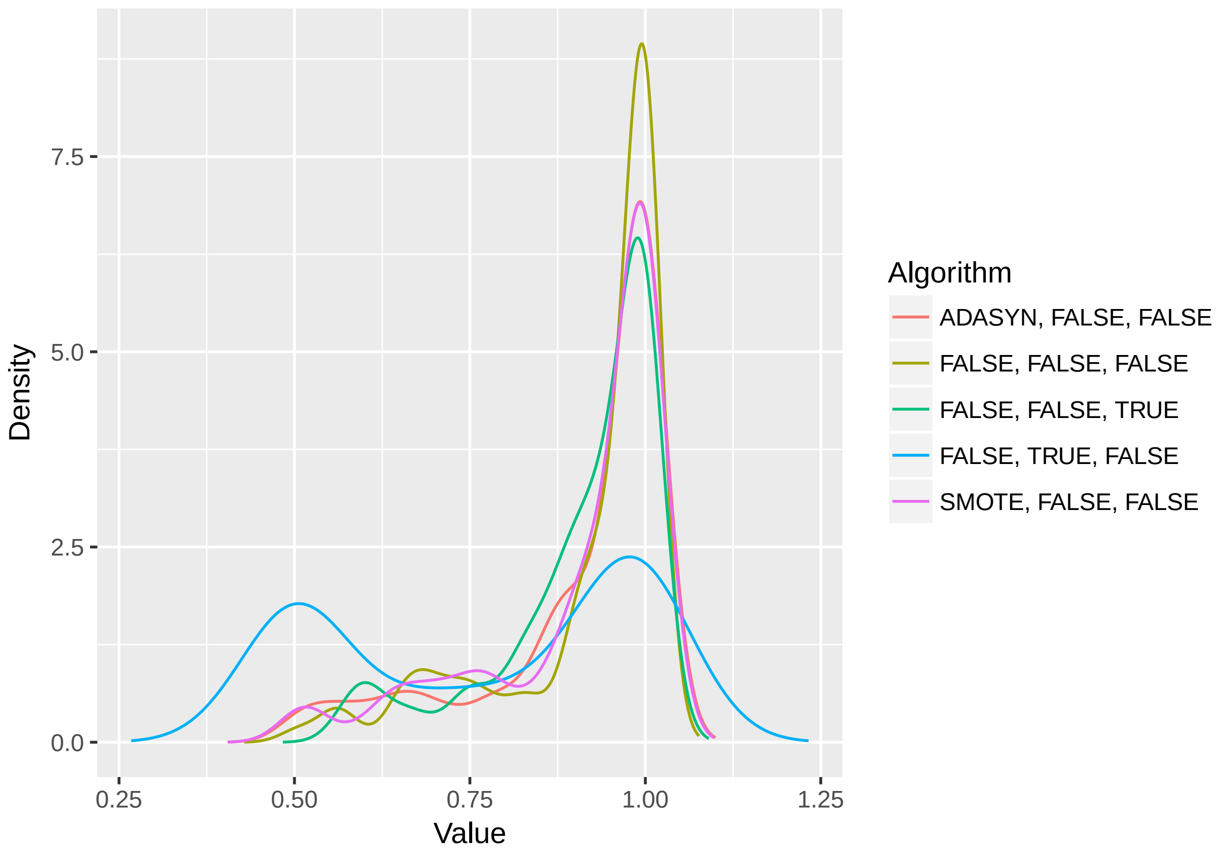
```
# Renomeando a variavel
df = df_tec_wide_residual

summary(df)
```

```
##  ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
##  Min.   :0.5048       Min.   :0.4917      Min.   :0.5739
##  1st Qu.:0.8702       1st Qu.:0.9087      1st Qu.:0.8837
##  Median :0.9783       Median :0.9906      Median :0.9535
##  Mean   :0.9070       Mean   :0.9198      Mean   :0.9136
##  3rd Qu.:0.9995       3rd Qu.:1.0000      3rd Qu.:0.9970
##  Max.   :1.0000       Max.   :1.0000      Max.   :1.0000
##  NA's   :14           NA's   :3           NA's   :4
##  FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
##  Min.   :0.5000     Min.   :0.5026
##  1st Qu.:0.5000     1st Qu.:0.8792
##  Median :0.9087     Median :0.9772
##  Mean   :0.7912     Mean   :0.9075
##  3rd Qu.:0.9984     3rd Qu.:0.9997
##  Max.   :1.0000     Max.   :1.0000
##  NA's   :3          NA's   :5
```

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 = 27.742, df = 4, p-value = 1.407e-05
```

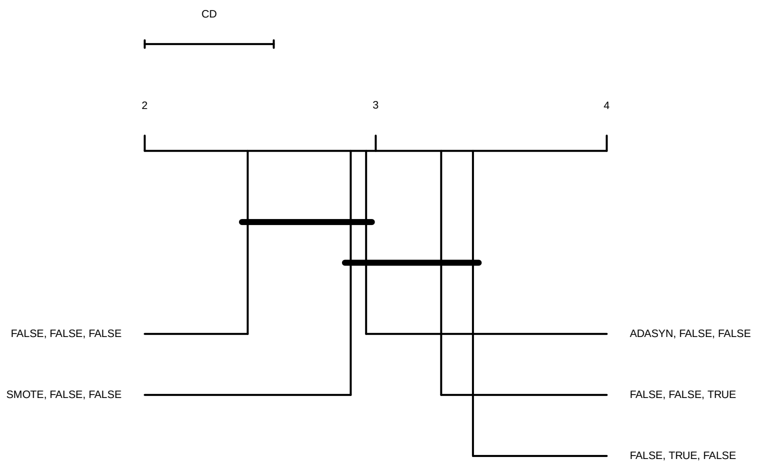
Testando as diferencas par a par

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

Plotando grafico de Critical Diference

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


CD
FALSE, FALSE, FALSE
SMOTE, FALSE, FALSE
ADASYN, FALSE, FALSE
FALSE, FALSE, TRUE
FALSE, TRUE, FALSE