R Notebook

Parametros:

Measure = Accuracy
Columns = sampling, weight_space, underbagging
Performance = holdout_measure_residual
Filter keys = imba.rate
Filter values = 0.001

```
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_

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                        :1800   ADASYN: 360   Mode :logical
##   Area under the curve            :   0   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.   :0.1269   Min.   :0.01517   Min.   :0.03881
##   1st Qu.:0.9898   1st Qu.:0.98750   1st Qu.:0.38633
##   Median :0.9938   Median :0.99163   Median :0.76435
##   Mean   :0.9692   Mean   :0.96680   Mean   :0.67114
##   3rd Qu.:0.9990   3rd Qu.:0.99687   3rd Qu.:0.95470
##   Max.   :1.0000   Max.   :1.00000   Max.   :1.00000
##   NA's   :48       NA's   :48        NA's   :48
##   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   :48      (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

```
# 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

head(df)
```

```
##   ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## 1            0.3572658           0.3329890          0.6153581
## 2            0.4186973           0.4313027          0.5874736
## 3            0.6295736           0.6298259          0.6235175
## 4            0.8769575           0.8337062          0.9418345
## 5            0.8764259           0.8852978          0.8776933
## 6            0.7399449           0.7471074          0.7432507
##   FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## 1          0.3297176           0.3553719
## 2          0.3964926           0.4211234
## 3          0.6298259           0.6298259
## 4          0.8337062           0.8724832
## 5          0.8852978           0.8770596
## 6          0.7471074           0.7377410
```

```
summary(df)
```

```
##  ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
##  Min.   :0.03881      Min.   :0.03881     Min.   :0.04134
##  1st Qu.:0.39542      1st Qu.:0.32630     1st Qu.:0.58672
##  Median :0.75019      Median :0.72813     Median :0.78900
##  Mean   :0.67685      Mean   :0.64527     Mean   :0.72918
##  3rd Qu.:0.97498      3rd Qu.:0.97555     3rd Qu.:0.93068
##  Max.   :0.99989      Max.   :0.99991     Max.   :0.99984
```

```
##  NA's   :10           NA's   :2
##  FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
##  Min.   :0.03881    Min.   :0.03881
##  1st Qu.:0.32656    1st Qu.:0.36062
##  Median :0.72777    Median :0.74023
##  Mean   :0.64075    Mean   :0.66303
##  3rd Qu.:0.97533    3rd Qu.:0.97405
##  Max.   :0.99991    Max.   :0.99992
##  NA's   :2          NA's   :2
```

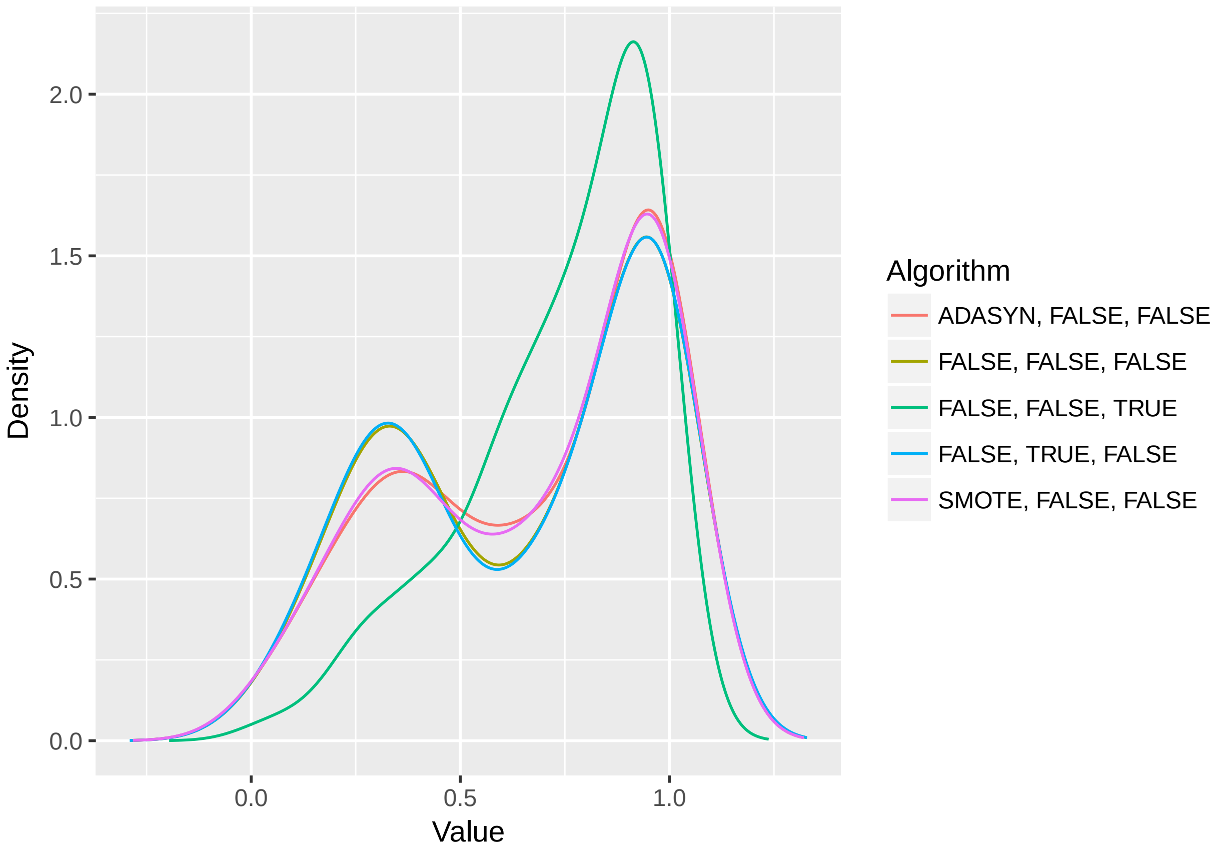
Verificando a média de cada coluna selecionada

```
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 = 0.676851883039613"
## [1] "Media da coluna FALSE, FALSE, FALSE = 0.645272174989147"
## [1] "Media da coluna FALSE, FALSE, TRUE = 0.729178795240191"
## [1] "Media da coluna FALSE, TRUE, FALSE = 0.64075122460108"
## [1] "Media da coluna SMOTE, FALSE, FALSE = 0.663026754446335"
```

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 = 25.772, df = 4, p-value = 3.518e-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               TRUE
## [2,]                FALSE               FALSE               TRUE
## [3,]                 TRUE                TRUE              FALSE
## [4,]                FALSE               FALSE               TRUE
## [5,]                FALSE               FALSE              FALSE
##      FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## [1,]              FALSE               FALSE
```

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

Plotando os ranks

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

```
## ADASYN, FALSE, FALSE  FALSE, FALSE, FALSE   FALSE, FALSE, TRUE
##             3.133333             3.283333             2.408333
##   FALSE, TRUE, FALSE  SMOTE, FALSE, FALSE
##             3.279167             2.895833
```

Plotando grafico de Critical Diference

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