R Notebook

Parametros:

Measure = Area under the curve
Columns = sampling, weight_space, underbagging
Performance = holdout_measure_residual
Filter keys = NULL
Filter values = NULL

```
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        :3420   Mode :logical
##  classif.randomForest:3420   FALSE:8208
##  classif.rusboost    :   0   TRUE :2052
##  classif.xgboost     :3420   NA's :0
##
##
##
##                             measure        sampling    underbagging
##  Accuracy                       :    0   ADASYN:2052   Mode :logical
##  Area under the curve           :10260   FALSE :6156   FALSE:8208
##  F1 measure                     :    0   SMOTE :2052   TRUE :2052
##  G-mean                         :    0                 NA's :0
##  Matthews correlation coefficient:    0
##
##
##  tuning_measure   holdout_measure  holdout_measure_residual
##  Min.   :0.3023   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.9325   1st Qu.:0.8620   1st Qu.:0.7067
##  Median :0.9967   Median :0.9831   Median :0.8932
##  Mean   :0.9380   Mean   :0.8972   Mean   :0.8310
##  3rd Qu.:1.0000   3rd Qu.:0.9999   3rd Qu.:0.9819
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##  NA's   :243      NA's   :243      NA's   :243
##  iteration_count                      dataset       imba.rate
##  Min.   :1       abalone                  : 180   Min.   :0.0010
##  1st Qu.:1       adult                    : 180   1st Qu.:0.0100
##  Median :2       bank                     : 180   Median :0.0300
##  Mean   :2       car                      : 180   Mean   :0.0286
##  3rd Qu.:3       cardiotocography-10clases: 180   3rd Qu.:0.0500
##  Max.   :3       cardiotocography-3clases : 180   Max.   :0.0500
##  NA's   :243     (Other)                  :9180
```

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] 684   5
```

```
# Renomeando a variavel
df = df_tec_wide_residual

head(df)
```

```
##   ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## 1            0.5136216           0.5133720          0.6259505
## 2            0.5136216           0.5133720          0.6259505
## 3            0.5814637           0.6445602          0.6536723
## 4            0.5948389           0.6479328          0.6628553
## 5                   NA           0.4549243          0.7943273
## 6                   NA           0.4549243          0.7943273
##   FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## 1          0.5137510           0.5167415
## 2          0.5137510           0.5167415
## 3          0.6538361           0.5561325
## 4                 NA           0.5862406
## 5          0.5371265           0.6257369
## 6          0.5371265           0.6257369
```

```
summary(df)
```

```
##  ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
##  Min.   :0.3050       Min.   :0.3577      Min.   :0.2777
##  1st Qu.:0.7250       1st Qu.:0.7277      1st Qu.:0.6722
##  Median :0.8836       Median :0.8941      Median :0.8635
##  Mean   :0.8361       Mean   :0.8385      Mean   :0.8115
##  3rd Qu.:0.9795       3rd Qu.:0.9785      3rd Qu.:0.9661
##  Max.   :1.0000       Max.   :1.0000      Max.   :1.0000
```

```
##  NA's   :34           NA's   :9           NA's   :6
##  FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
##  Min.   :0.2916     Min.   :0.2778
##  1st Qu.:0.7274     1st Qu.:0.7228
##  Median :0.8927     Median :0.8762
##  Mean   :0.8370     Mean   :0.8325
##  3rd Qu.:0.9779     3rd Qu.:0.9801
##  Max.   :1.0000     Max.   :1.0000
##  NA's   :14         NA's   :18
```

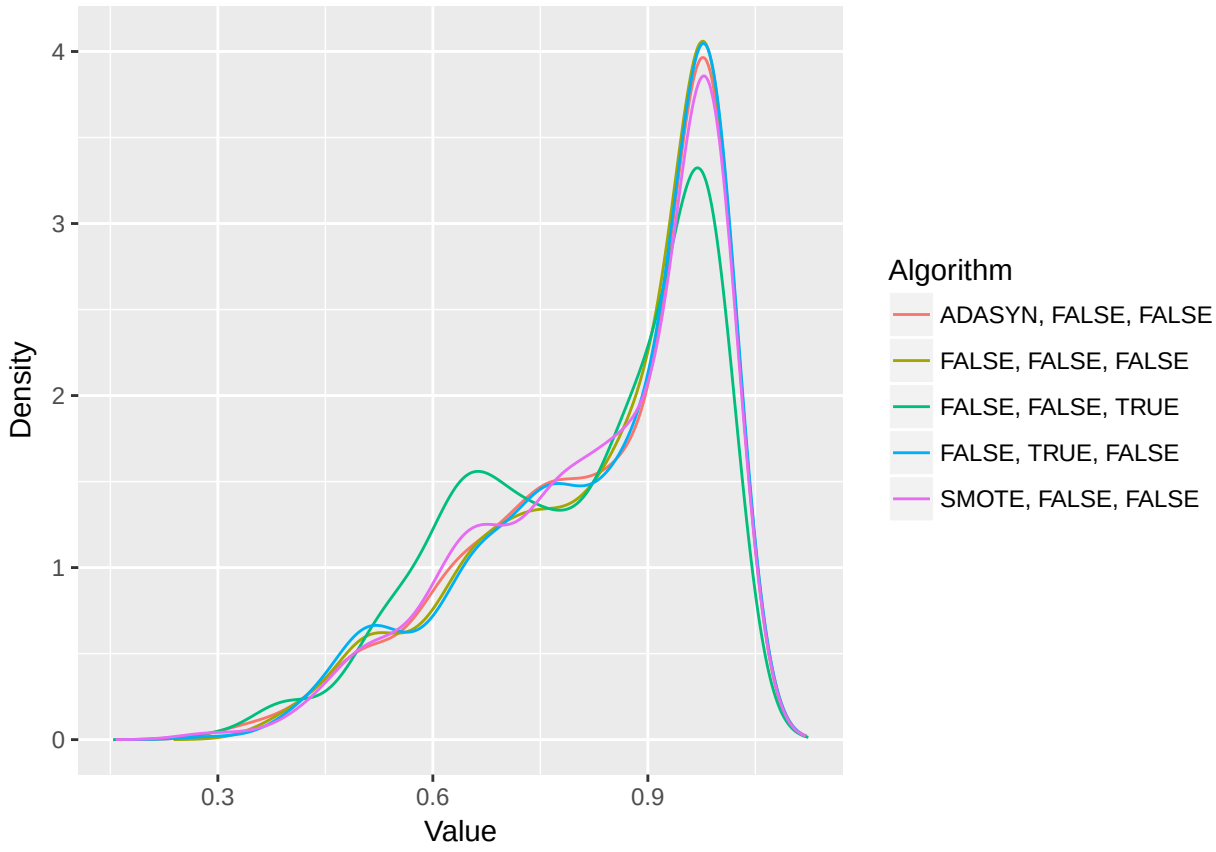
Verificando a média de cada coluna selecionada

```
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 = 0.836068721740555"
## [1] "Media da coluna FALSE, FALSE, FALSE = 0.838499008679809"
## [1] "Media da coluna FALSE, FALSE, TRUE = 0.811510294383748"
## [1] "Media da coluna FALSE, TRUE, FALSE = 0.836984836601512"
## [1] "Media da coluna SMOTE, FALSE, FALSE = 0.832505690338521"
```

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 = 129.73, 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                FALSE                TRUE
## [2,]                 FALSE                FALSE                TRUE
## [3,]                  TRUE                 TRUE               FALSE
## [4,]                 FALSE                FALSE                TRUE
## [5,]                 FALSE                FALSE                TRUE
##       FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## [1,]               FALSE               FALSE
```

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

Plotando os ranks

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

```
## ADASYN, FALSE, FALSE  FALSE, FALSE, FALSE   FALSE, FALSE, TRUE
##             2.954678             2.750731             3.597953
##   FALSE, TRUE, FALSE  SMOTE, FALSE, FALSE
##             2.791667             2.904971
```

Plotando grafico de Critical Diference

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