R Notebook

Parametros:

Measure = Matthews correlation coefficient
Columns = sampling, weight_space, underbagging
Performance = holdout_measure
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           :    0   FALSE :6156   FALSE:8208
##  F1 measure                     :    0   SMOTE :2052   TRUE :2052
##  G-mean                         :    0                 NA's :0
##  Matthews correlation coefficient:10260
##
##
##  tuning_measure    holdout_measure   holdout_measure_residual
##  Min.   :-0.1277   Min.   :-0.2120   Min.   :-0.46576
##  1st Qu.: 0.3307   1st Qu.: 0.0000   1st Qu.: 0.03886
##  Median : 0.8174   Median : 0.4907   Median : 0.21377
##  Mean   : 0.6548   Mean   : 0.4657   Mean   : 0.30966
##  3rd Qu.: 0.9890   3rd Qu.: 0.8152   3rd Qu.: 0.53139
##  Max.   : 1.0000   Max.   : 1.0000   Max.   : 1.00000
##  NA's   :225       NA's   :225       NA's   :225
##  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   :225     (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        -0.0144995162        -0.002681154         0.03648651
## 2        -0.0144995162        -0.002681154         0.03648651
## 3         0.0004984056        -0.007549758         0.08179207
## 4         0.0591905572         0.103707362         0.12220615
## 5        -0.0090811368         0.011520780         0.11251890
## 6        -0.0090811368         0.011520780         0.11251890
##   FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## 1       -0.004028968         -0.01502422
## 2       -0.004028968         -0.01502422
## 3        0.023040142          0.01880716
## 4        0.075853035          0.09021363
## 5        0.009851304          0.01260881
## 6        0.009851304          0.01260881
```

```
summary(df)
```

```
##  ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
##  Min.   :-0.06657     Min.   :-0.04044    Min.   :-0.06927
##  1st Qu.: 0.10500     1st Qu.: 0.00000    1st Qu.: 0.17622
##  Median : 0.47437     Median : 0.53612    Median : 0.37560
##  Mean   : 0.47796     Mean   : 0.47690    Mean   : 0.41910
##  3rd Qu.: 0.82994     3rd Qu.: 0.84465    3rd Qu.: 0.62900
##  Max.   : 1.00000     Max.   : 1.00000    Max.   : 1.00000
```

```
##  NA's   :32           NA's   :6           NA's   :6
##  FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
##  Min.   :-0.03836   Min.   :-0.0748
##  1st Qu.: 0.00000   1st Qu.: 0.1152
##  Median : 0.50398   Median : 0.4830
##  Mean   : 0.47172   Mean   : 0.4839
##  3rd Qu.: 0.83988   3rd Qu.: 0.8393
##  Max.   : 1.00000   Max.   : 1.0000
##  NA's   :11         NA's   :20
```

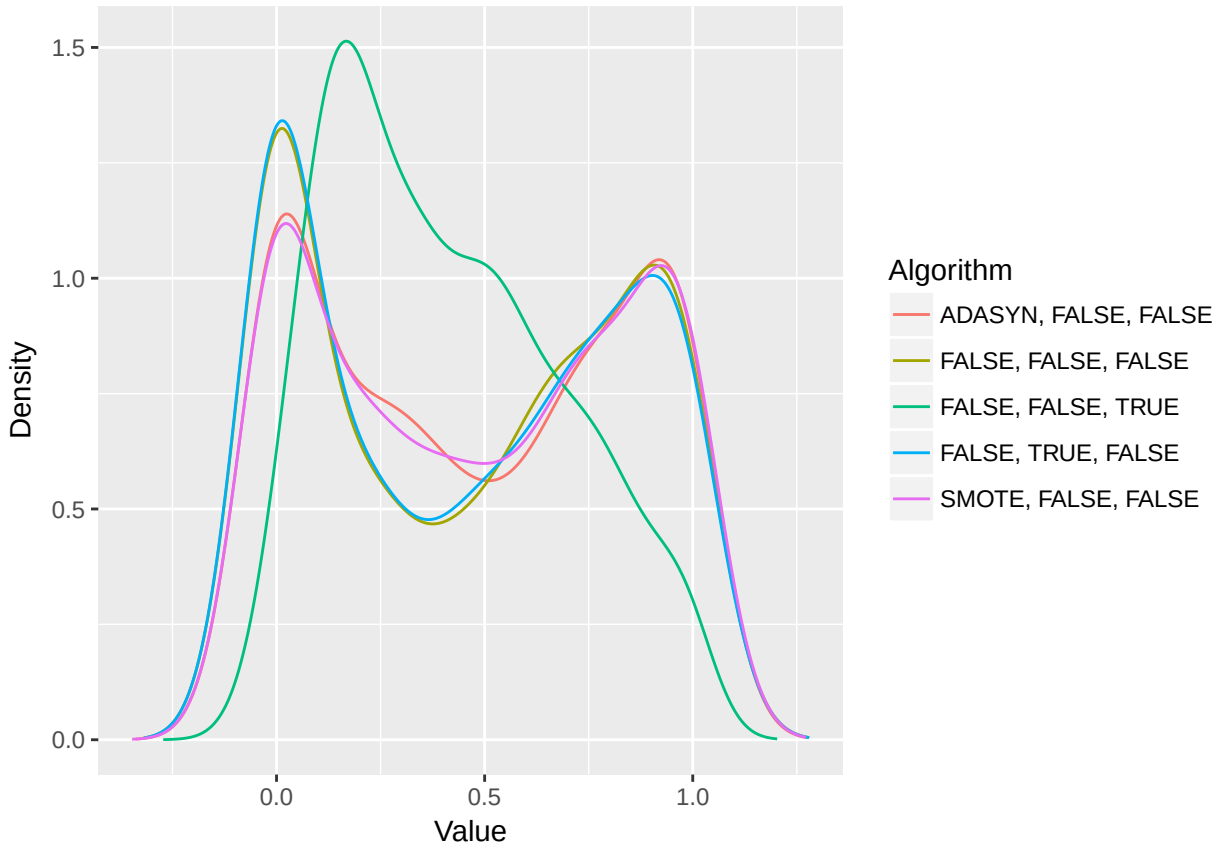
Verificando a média de cada coluna selecionada

```
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 = 0.477962273506577"
## [1] "Media da coluna FALSE, FALSE, FALSE = 0.47690369276557"
## [1] "Media da coluna FALSE, FALSE, TRUE = 0.419103377167762"
## [1] "Media da coluna FALSE, TRUE, FALSE = 0.471719912101946"
## [1] "Media da coluna SMOTE, FALSE, FALSE = 0.483939594059321"
```

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 = 23.296, df = 4, p-value = 0.0001105
```

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
##             2.913743             2.989766             3.239766
##   FALSE, TRUE, FALSE  SMOTE, FALSE, FALSE
##             2.998538             2.858187
```

Plotando grafico de Critical Diference

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