R Notebook

Parametros:

Measure = Accuracy
Columns = sampling, weight_space, underbagging
Performance = tuning_measure
Filter keys = NULL
Filter values = NULL

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

Tratamento dos dados

Carregando data set compilado

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

```r
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

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

Filtrando o data set

```r
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                        :10260   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:    0
##
##
##  tuning_measure    holdout_measure    holdout_measure_residual
##  Min.   :0.09041   Min.   :0.01517   Min.   :0.0346
##  1st Qu.:0.96185   1st Qu.:0.95349   1st Qu.:0.3809
##  Median :0.98796   Median :0.98113   Median :0.7239
##  Mean   :0.95509   Mean   :0.94933   Mean   :0.6600
##  3rd Qu.:0.99669   3rd Qu.:0.99347   3rd Qu.:0.9428
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.0000
##  NA's   :204       NA's   :204       NA's   :204
##  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   :204     (Other)                 :9180
```

Computando as médias das iteracoes

```r
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

```r
# 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$performan

# 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
```

```r
# Renomeando a variavel
df = df_tec_wide_residual

head(df)
```

```
##   ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## 1            0.9853969           0.9878042          0.5674864
## 2            0.9853969           0.9878042          0.5674864
## 3            0.9595381           0.9578372          0.6175442
## 4            0.9453798           0.9266667          0.6054167
## 5            0.9920423           0.9891070          0.8833724
## 6            0.9920423           0.9891070          0.8833724
##   FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## 1          0.9883986           0.9845604
## 2          0.9883986           0.9845604
## 3          0.9625227           0.9611646
## 4          0.9345833           0.9448830
## 5          0.9896592           0.9930487
## 6          0.9896592           0.9930487
```

```r
summary(df)
```

```
##   ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
##  Min.   :0.7035       Min.   :0.9264      Min.   :0.2896
##  1st Qu.:0.9824       1st Qu.:0.9694      1st Qu.:0.7736
##  Median :0.9951       Median :0.9874      Median :0.9066
##  Mean   :0.9834       Mean   :0.9800      Mean   :0.8504
##  3rd Qu.:0.9988       3rd Qu.:0.9931      3rd Qu.:0.9686
##  Max.   :1.0000       Max.   :1.0000      Max.   :1.0000
```

```
##  NA's   :33              NA's   :6              NA's   :5
##  FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
##  Min.   :0.9346      Min.   :0.6952
##  1st Qu.:0.9691      1st Qu.:0.9822
##  Median :0.9875      Median :0.9954
##  Mean   :0.9799      Mean   :0.9835
##  3rd Qu.:0.9930      3rd Qu.:0.9993
##  Max.   :1.0000      Max.   :1.0000
##  NA's   :6           NA's   :18
```

Verificando a média de cada coluna selecionada

```r
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 = 0.983393474568938"
## [1] "Media da coluna FALSE, FALSE, FALSE = 0.97997133888602"
## [1] "Media da coluna FALSE, FALSE, TRUE = 0.850441156011817"
## [1] "Media da coluna FALSE, TRUE, FALSE = 0.97994498777825"
## [1] "Media da coluna SMOTE, FALSE, FALSE = 0.98347667806597"
```

Fazendo teste de normalidade

```r
plotDensities(data = na.omit(df))
```
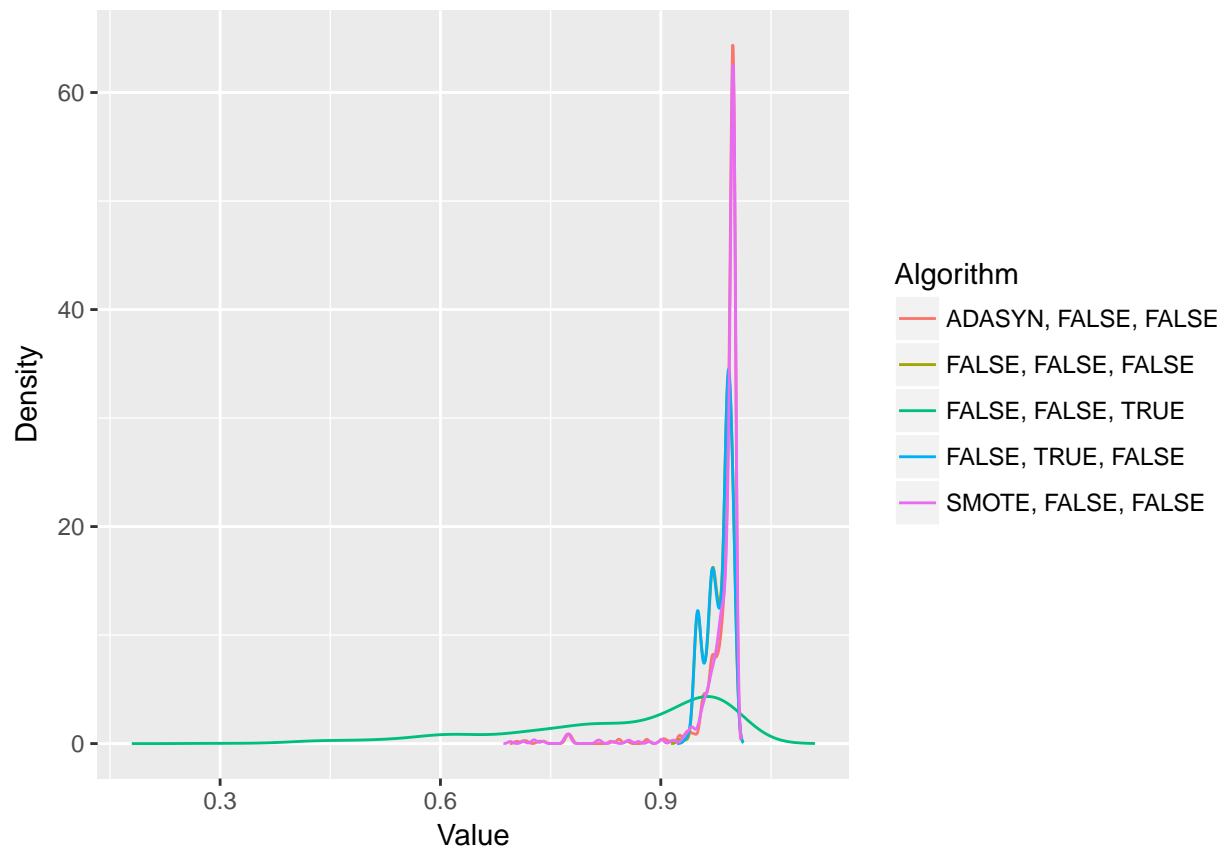
Testando as diferencas

```
friedmanTest(df)
```

```
##
##   Friedman's rank sum test
##
## data:  df
## Friedman's chi-squared = 1426.6, 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                TRUE               TRUE
## [2,]                 TRUE               FALSE               TRUE
## [3,]                 TRUE                TRUE              FALSE
## [4,]                 TRUE               FALSE               TRUE
## [5,]                FALSE                TRUE               TRUE
##       FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## [1,]                TRUE               FALSE
```

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

Plotando os ranks

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

```
## ADASYN, FALSE, FALSE  FALSE, FALSE, FALSE   FALSE, FALSE, TRUE
##             1.978070             3.228070             4.674708
##   FALSE, TRUE, FALSE  SMOTE, FALSE, FALSE
##             3.239766             1.879386
```

Plotando grafico de Critical Diference

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