R Notebook

Parametros:

Measure = Accuracy
Columns = sampling, weight_space, underbagging
Performance = tuning_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_

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        :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.:0.38526
##  Median :0.9938   Median :0.99163   Median :0.75447
##  Mean   :0.9691   Mean   :0.96664   Mean   :0.66878
##  3rd Qu.:0.9990   3rd Qu.:0.99687   3rd Qu.:0.95350
##  Max.   :1.0000   Max.   :1.00000   Max.   :1.00000
##  NA's   :57       NA's   :57        NA's   :57
##  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   :57      (Other)                  :1530
```

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] 120   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            0.9920423           0.9891070          0.8833724
## 3            0.9987904           0.9900002          0.4496844
## 4            1.0000000           0.9976581          0.9976611
## 5            1.0000000           0.9933338          0.9895862
## 6            0.9986705           0.9943033          0.9973719
##   FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## 1          0.9883986           0.9845604
## 2          0.9896592           0.9930487
## 3          0.9900002           0.9986848
## 4          0.9976581           1.0000000
## 5          0.9933338           1.0000000
## 6          0.9943033           0.9986707
```

```r
summary(df)
```

```
##  ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
##  Min.   :0.9337       Min.   :0.9878      Min.   :0.4327
##  1st Qu.:0.9952       1st Qu.:0.9900      1st Qu.:0.7952
##  Median :0.9985       Median :0.9919      Median :0.9256
##  Mean   :0.9963       Mean   :0.9933      Mean   :0.8700
##  3rd Qu.:0.9996       3rd Qu.:0.9958      3rd Qu.:0.9808
##  Max.   :1.0000       Max.   :1.0000      Max.   :1.0000
```

```
##  NA's   :9            NA's   :1
##  FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
##  Min.   :0.9883     Min.   :0.9337
##  1st Qu.:0.9900     1st Qu.:0.9965
##  Median :0.9923     Median :0.9987
##  Mean   :0.9933     Mean   :0.9967
##  3rd Qu.:0.9962     3rd Qu.:0.9997
##  Max.   :1.0000     Max.   :1.0000
##  NA's   :2          NA's   :7
```

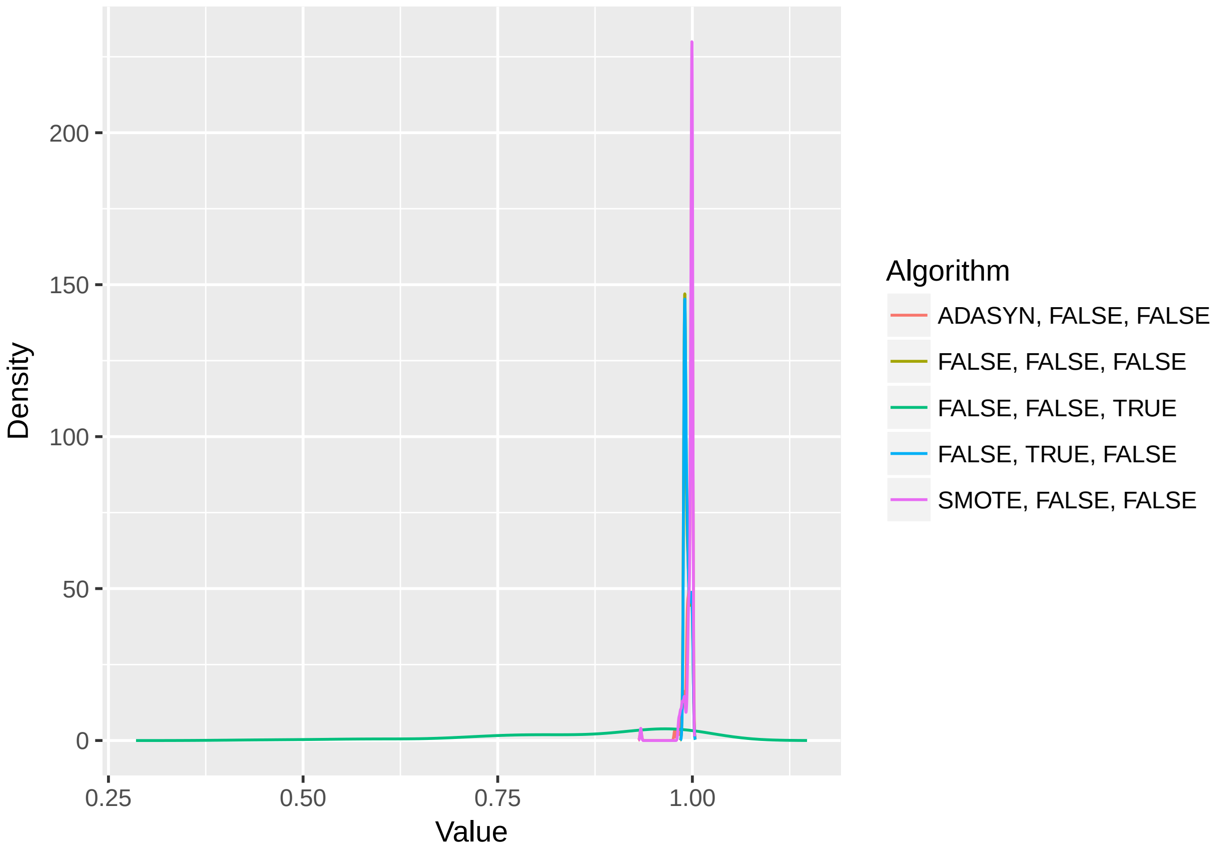
Verificando a média de cada coluna selecionada

```
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 = 0.996265103803147"
## [1] "Media da coluna FALSE, FALSE, FALSE = 0.993309113805628"
## [1] "Media da coluna FALSE, FALSE, TRUE = 0.870003088777276"
## [1] "Media da coluna FALSE, TRUE, FALSE = 0.993312454379742"
## [1] "Media da coluna SMOTE, FALSE, FALSE = 0.996666872531583"
```

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 = 208.89, 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

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

```
## ADASYN, FALSE, FALSE  FALSE, FALSE, FALSE   FALSE, FALSE, TRUE
##             2.129167             3.191667             4.545833
##   FALSE, TRUE, FALSE  SMOTE, FALSE, FALSE
##             3.195833             1.937500
```

Plotando grafico de Critical Diference

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