

# R Notebook

## Parametros:

```
Measure = G-mean
Columns = learner
Performance = tuning_measure
Filter keys = sampling, weight_space, underbagging, imba.rate
Filter values = FALSE, FALSE, FALSE, 0.03
```

```
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.csv")
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      cardiocography-10clases: 900 3rd Qu.:0.0500
## Max. :3      cardiocography-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      :198 Mode :logical
## classif.randomForest:198 FALSE:594
## classif.rusboost   : 0  NA's :0
## classif.xgboost    :198
##
##
##
##           measure      sampling  underbagging
## Accuracy           : 0  ADASYN: 0  Mode :logical
## Area under the curve : 0  FALSE :594 FALSE:594
## F1 measure          : 0  SMOTE : 0  NA's :0
## G-mean              :594
## Matthews correlation coefficient: 0
##
##
## tuning_measure  holdout_measure  holdout_measure_residual
## Min. :0.0000  Min. :0.0000  Min. :0.00000
## 1st Qu.:0.1345 1st Qu.:0.0000 1st Qu.:0.07956
## Median :0.5914 Median :0.6325 Median :0.34862
## Mean :0.5140 Mean :0.5217 Mean :0.40941
## 3rd Qu.:0.8667 3rd Qu.:0.9129 3rd Qu.:0.72637
## Max. :1.0000 Max. :1.0000 Max. :1.00000
## NA's :3 NA's :3 NA's :3
## iteration_count dataset imba.rate
## Min. :1 abalone : 9 Min. :0.03
## 1st Qu.:1 adult : 9 1st Qu.:0.03
## Median :2 annealing : 9 Median :0.03
## Mean :2 arrhythmia : 9 Mean :0.03
## 3rd Qu.:3 balance-scale: 9 3rd Qu.:0.03
## Max. :3 bank : 9 Max. :0.03
## NA's :3 (Other) :540
```

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$performance)))

# 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] 66 3
```

```
# Renomeando a variavel
df = df_tec_wide_residual

head(df)
```

```
##   classif.ksvm classif.randomForest classif.xgboost
## 1   0.23019478         0.00000000         0.0000000
## 2   0.34575254                NA         0.5292628
## 3   0.26604411         0.6540313         0.5043107
## 4   0.00000000         0.5037406         0.5408550
## 5   0.95922146         1.0000000         1.0000000
## 6   0.08803044         0.2017186         0.2888779
```

```
summary(df)
```

```
##   classif.ksvm   classif.randomForest classif.xgboost
## Min.   :0.00000   Min.   :0.0000         Min.   :0.0000
## 1st Qu.:0.01121   1st Qu.:0.2017         1st Qu.:0.4237
## Median :0.27026   Median :0.6758         Median :0.6896
## Mean   :0.33794   Mean   :0.5755         Mean   :0.6295
## 3rd Qu.:0.64915   3rd Qu.:0.8985         3rd Qu.:0.8910
## Max.   :0.96114   Max.   :1.0000         Max.   :1.0000
## NA's   :1
```

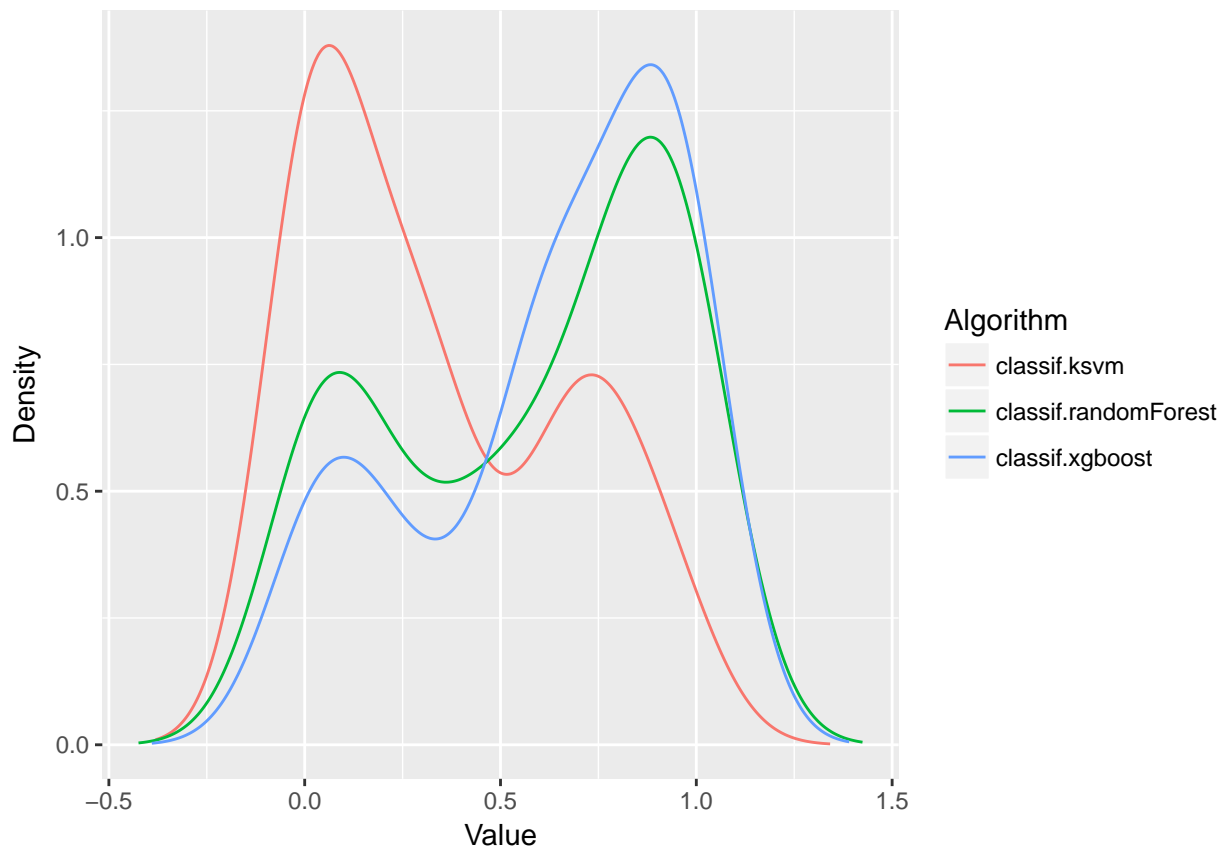
## Verificando a média de cada coluna selecionada

```
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 classif.ksvm = 0.337943434843279"
## [1] "Media da coluna classif.randomForest = 0.57549035347215"
## [1] "Media da coluna classif.xgboost = 0.629513573662504"
```

## 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 = 37.932, df = 2, p-value = 5.797e-09
```

## Testando as diferenças par a par

```
test <- nemenyiTest (df, alpha=0.05)
abs(test$diff.matrix) > test$statistic
```

```
##      classific.svm classific.randomForest classific.xgboost
## [1,]          FALSE                TRUE                TRUE
## [2,]           TRUE                FALSE                TRUE
## [3,]           TRUE                TRUE                 FALSE
```

## Plotando os ranks

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

```
##      classific.svm classific.randomForest      classific.xgboost
##      2.560606      1.946970      1.492424
```

## Plotando grafico de Critical Difference

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

