

# R Notebook

## Parametros:

```
Measure = Accuracy
Columns = sampling, weight_space, underbagging
Performance = tuning_measure
Filter keys = imba.rate
Filter values = 0.05
```

```
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)){
  ds = filter_at(ds, .vars = params$filter_keys, .vars_predicate = any_vars(. == params$filter_values))
}
```

```
summary(ds)
```

```
##           learner      weight_space
## classif.ksvm      :1230  Mode :logical
## classif.randomForest:1230 FALSE:2952
## classif.rusboost   :  0  TRUE :738
## classif.xgboost    :1230  NA's :0
##
##
##
##           measure      sampling      underbagging
## Accuracy           :3690  ADASYN: 738  Mode :logical
## Area under the curve :  0  FALSE :2214  FALSE:2952
## F1 measure           :  0  SMOTE : 738  TRUE :738
## G-mean               :  0              NA's :0
## Matthews correlation coefficient:  0
##
##
## tuning_measure  holdout_measure  holdout_measure_residual
## Min. :0.2470  Min. :0.04739  Min. :0.0367
## 1st Qu.:0.9494  1st Qu.:0.94505  1st Qu.:0.3902
## Median :0.9688  Median :0.96078  Median :0.7223
## Mean :0.9425  Mean :0.93413  Mean :0.6602
## 3rd Qu.:0.9908  3rd Qu.:0.98413  3rd Qu.:0.9315
## Max. :1.0000  Max. :1.00000  Max. :1.0000
## NA's :42      NA's :42      NA's :42
## iteration_count      dataset      imba.rate
## Min. :1      abalone : 45  Min. :0.05
## 1st Qu.:1      adult : 45  1st Qu.:0.05
## Median :2      annealing : 45  Median :0.05
## Mean :2      arrhythmia : 45  Mean :0.05
## 3rd Qu.:3      balance-scale: 45  3rd Qu.:0.05
## Max. :3      bank : 45  Max. :0.05
## NA's :42      (Other) :3420
```

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))

# Seleccionando 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] 246 5
```

```
# Renomeando a variavel
df = df_tec_wide_residual

summary(df)
```

```
## ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## Min. :0.7035 Min. :0.9264 Min. :0.3600
## 1st Qu.:0.9680 1st Qu.:0.9500 1st Qu.:0.7655
## Median :0.9869 Median :0.9623 Median :0.8939
## Mean :0.9709 Mean :0.9668 Mean :0.8374
## 3rd Qu.:0.9962 3rd Qu.:0.9814 3rd Qu.:0.9622
## Max. :1.0000 Max. :1.0000 Max. :1.0000
## NA's :5 NA's :1 NA's :3
## FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## Min. :0.9346 Min. :0.6952
## 1st Qu.:0.9500 1st Qu.:0.9671
## Median :0.9623 Median :0.9858
## Mean :0.9669 Mean :0.9702
## 3rd Qu.:0.9823 3rd Qu.:0.9966
## Max. :1.0000 Max. :1.0000
## NA's :1 NA's :4
```

## Verificando a média de cada coluna seleccionada

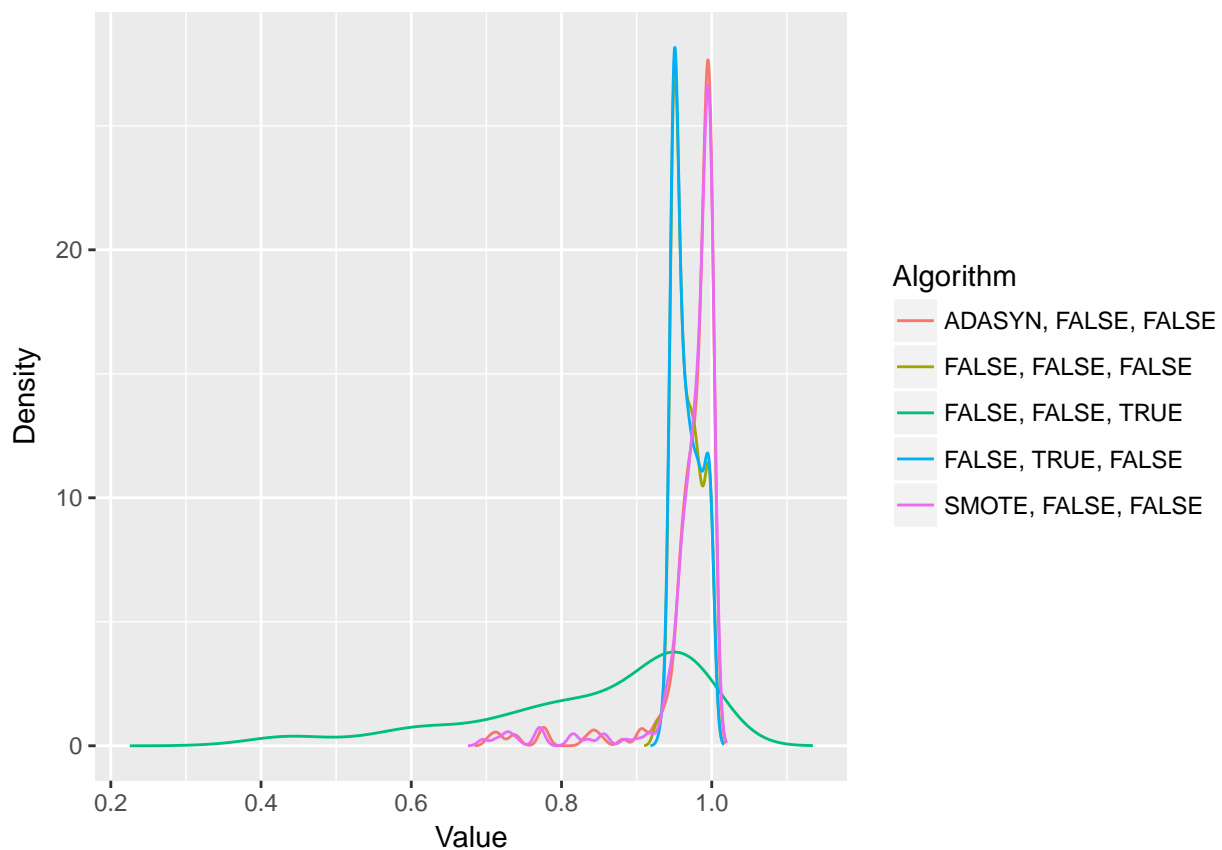
```
for(i in (1:dim(df)[2])){
  #print(df[,i])
}
```

```
print(paste("Media da coluna ", colnames(df)[i], " = ", mean(df[,i], na.rm = TRUE), sep=""))
}
```

```
## [1] "Media da coluna ADASYN, FALSE, FALSE = 0.970901888643716"
## [1] "Media da coluna FALSE, FALSE, FALSE = 0.966845067110671"
## [1] "Media da coluna FALSE, FALSE, TRUE = 0.837433538445808"
## [1] "Media da coluna FALSE, TRUE, FALSE = 0.96686737646468"
## [1] "Media da coluna SMOTE, FALSE, FALSE = 0.970195956041825"
```

## 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 = 558.99, df = 4, p-value < 2.2e-16
```

## Testando as diferenças 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 grafico de Critical Difference

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

