

# R Notebook

## Parametros:

Measure = Matthews correlation coefficient  
Columns = sampling, weight\_space, ruspool  
Performance = holdout\_measure\_residual  
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.  
summary(ds)
```

```
##           learner      weight_space  
## classif.ksvm      :17100  Mode :logical  
## classif.randomForest:17100 FALSE:41040  
## classif.xgboost    :17100  TRUE :10260  
##                                     NA's :0  
##  
##  
##  
##           measure      sampling      ruspool  
## 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.: 0.5924  1st Qu.: 0.3114  1st Qu.: 0.1648  
## Median : 0.9624  Median : 0.8193  Median : 0.5192  
## Mean : 0.7570  Mean : 0.6469  Mean : 0.5099  
## 3rd Qu.: 0.9965  3rd Qu.: 0.9879  3rd Qu.: 0.8636  
## Max. : 1.0000  Max. : 1.0000  Max. : 1.0000  
## NA's :1761  NA's :1761  NA's :1761  
## 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 :1761 (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.xgboost    :1230  TRUE :738  
##                   NA's :0  
##  
##  
##  
##           measure      sampling      ruspool  
## Accuracy           : 0  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:3690  
##  
##  
## tuning_measure      holdout_measure      holdout_measure_residual  
## Min.      :-0.1277  Min.      :-0.2120  Min.      :-0.4571  
## 1st Qu.: 0.2902    1st Qu.: 0.0000  1st Qu.: 0.0255  
## Median : 0.7512    Median : 0.4904  Median : 0.2055  
## Mean      : 0.6188    Mean      : 0.4650  Mean      : 0.3031  
## 3rd Qu.: 0.9634    3rd Qu.: 0.8098  3rd Qu.: 0.5400  
## Max.      : 1.0000    Max.      : 1.0000  Max.      : 1.0000  
## NA's      :75       NA's      :75       NA's      :75  
## 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      :75       (Other)     :3420
```

Computando as médias das iteracoes

```
ds = group_by(ds, learner , weight_space , measure , sampling , ruspool , 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] 246    5

# Renomeando a variavel
df = df_tec_wide_residual

summary(df)

```

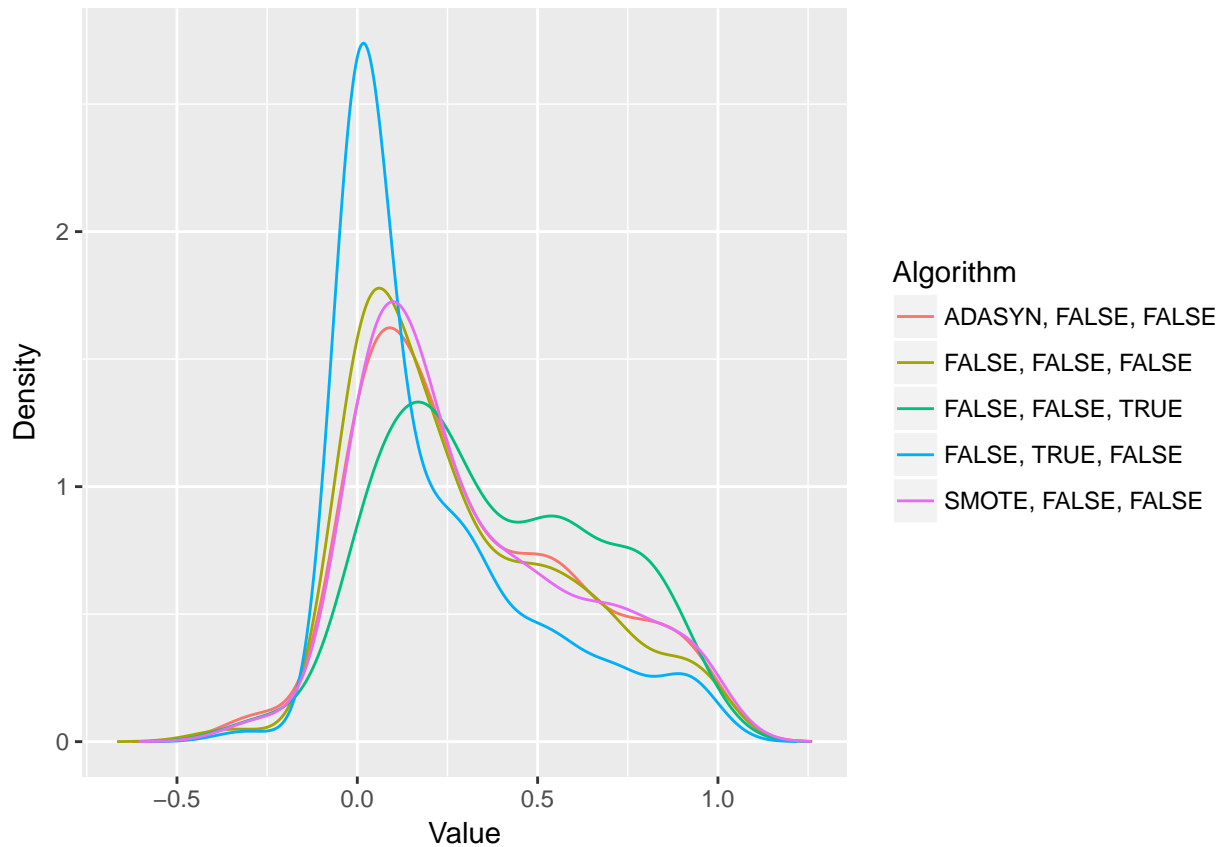
```

## ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## Min.   :-0.34981    Min.   :-0.39928    Min.   :-0.3331
## 1st Qu.: 0.06627    1st Qu.: 0.05527    1st Qu.: 0.1413
## Median : 0.21777    Median : 0.20258    Median : 0.3245
## Mean   : 0.30775    Mean    : 0.29438    Mean    : 0.3827
## 3rd Qu.: 0.53628    3rd Qu.: 0.49997    3rd Qu.: 0.6221
## Max.   : 0.98682    Max.    : 0.99489    Max.    : 0.9863
## NA's   :10          NA's     :4
## FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## Min.   :-0.34302    Min.   :-0.33229
## 1st Qu.: 0.00000    1st Qu.: 0.07465
## Median : 0.07629    Median : 0.20635
## Mean   : 0.21336    Mean    : 0.31718
## 3rd Qu.: 0.33403    3rd Qu.: 0.52932
## Max.   : 1.00000    Max.    : 0.98479
## NA's   :5          NA's     :6

```

## 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 = 124.59, 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          FALSE          TRUE
## [2,]          FALSE          FALSE          TRUE
## [3,]           TRUE          TRUE         FALSE
## [4,]           TRUE          TRUE          TRUE
## [5,]          FALSE          FALSE          TRUE
##      FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## [1,]           TRUE          FALSE
```

```
## [2,]          TRUE          FALSE
## [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) {})
```

