

# R Notebook

## Parametros:

Measure = Matthews correlation coefficient  
Columns = sampling, weight\_space, ruspool, learner  
Performance = tuning\_measure  
Filter keys = NULL  
Filter values = NULL

```
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      :3420  Mode :logical  
## classif.randomForest:3420 FALSE:8208  
## classif.xgboost    :3420  TRUE :2052  
##                   NA's :0  
##  
##  
##  
##           measure      sampling      ruspool  
## Accuracy           : 0  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:10260  
##  
##  
## tuning_measure  holdout_measure  holdout_measure_residual  
## Min.   :-0.1277  Min.   :-0.2120  Min.   :-0.4658  
## 1st Qu.: 0.2088  1st Qu.: 0.0000  1st Qu.: 0.0067  
## Median : 0.7306  Median : 0.4258  Median : 0.1812  
## Mean   : 0.6065  Mean   : 0.4333  Mean   : 0.2883  
## 3rd Qu.: 0.9851  3rd Qu.: 0.8098  3rd Qu.: 0.5031  
## Max.   : 1.0000  Max.   : 1.0000  Max.   : 1.0000  
## NA's   :357     NA's   :357     NA's   :357  
## 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   :357     (Other)      :9180
```

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] 228 15
```

```

# Renomeando a variavel
df = df_tec_wide_residual

summary(df)

```

```

## ADASYN, FALSE, FALSE, classif.ksvm
## Min. :0.5483
## 1st Qu.:0.9557
## Median :0.9899
## Mean :0.9620
## 3rd Qu.:0.9975
## Max. :1.0000
## NA's :7
## ADASYN, FALSE, FALSE, classif.randomForest
## Min. :0.4102
## 1st Qu.:0.9704
## Median :0.9920
## Mean :0.9713
## 3rd Qu.:0.9985
## Max. :1.0000
## NA's :37
## ADASYN, FALSE, FALSE, classif.xgboost FALSE, FALSE, FALSE, classif.ksvm
## Min. : -0.004855 Min. : -0.003311
## 1st Qu.: 0.910795 1st Qu.: 0.000000
## Median : 0.979204 Median : 0.210174
## Mean : 0.857472 Mean : 0.310733
## 3rd Qu.: 0.996005 3rd Qu.: 0.565760
## Max. : 1.000000 Max. : 1.000000
##
## FALSE, FALSE, FALSE, classif.randomForest

```

```

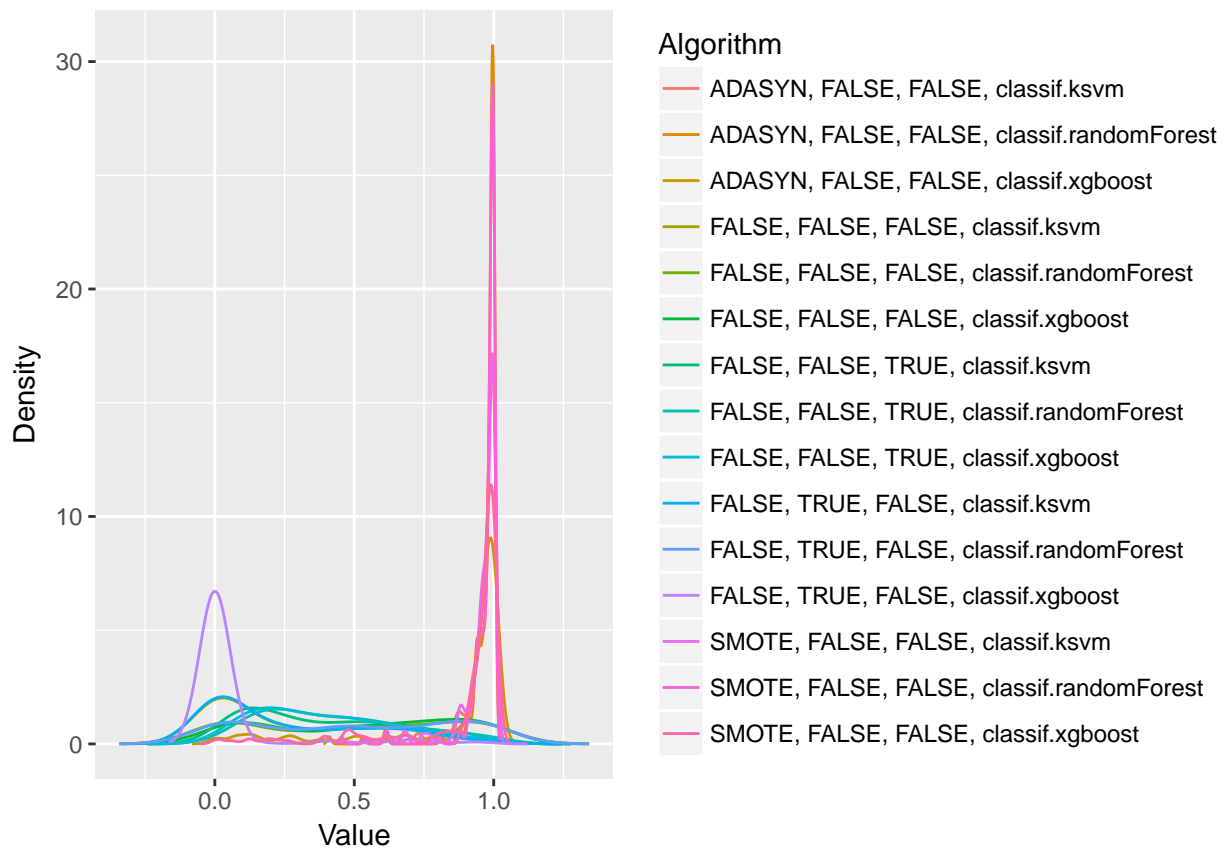
## Min.      :0.0000
## 1st Qu.:0.1826
## Median :0.5472
## Mean    :0.5299
## 3rd Qu.:0.8705
## Max.     :1.0000
## NA's     :8
## FALSE, FALSE, FALSE, classif.xgboost FALSE, FALSE, TRUE, classif.ksvm
## Min.      :-0.004855      Min.      :0.01683
## 1st Qu.: 0.242927      1st Qu.:0.16131
## Median : 0.643131      Median :0.39629
## Mean    : 0.570437      Mean    :0.41608
## 3rd Qu.: 0.855521      3rd Qu.:0.65193
## Max.     : 1.000000      Max.     :1.00000
##                      NA's      :3
## FALSE, FALSE, TRUE, classif.randomForest
## Min.      :0.006784
## 1st Qu.:0.221089
## Median :0.405446
## Mean    :0.444584
## 3rd Qu.:0.637303
## Max.     :1.000000
## NA's     :18
## FALSE, FALSE, TRUE, classif.xgboost FALSE, TRUE, FALSE, classif.ksvm
## Min.      :0.004119      Min.      :-0.002739
## 1st Qu.:0.200231      1st Qu.: 0.000000
## Median :0.368467      Median : 0.199159
## Mean    :0.409164      Mean    : 0.304707
## 3rd Qu.:0.579754      3rd Qu.: 0.536926
## Max.     :1.000000      Max.     : 1.000000
## NA's     :3
## FALSE, TRUE, FALSE, classif.randomForest
## Min.      :-0.002022
## 1st Qu.: 0.157423
## Median : 0.527450
## Mean    : 0.525408
## 3rd Qu.: 0.871424
## Max.     : 1.000000
## NA's     :13
## FALSE, TRUE, FALSE, classif.xgboost SMOTE, FALSE, FALSE, classif.ksvm
## Min.      :0.00000      Min.      :0.5423
## 1st Qu.:0.00000      1st Qu.:0.9570
## Median :0.00000      Median :0.9905
## Mean    :0.07776      Mean    :0.9616
## 3rd Qu.:0.00000      3rd Qu.:0.9988
## Max.     :0.95552      Max.     :1.0000
##
## SMOTE, FALSE, FALSE, classif.randomForest
## Min.      :0.4552
## 1st Qu.:0.9691
## Median :0.9926
## Mean    :0.9724
## 3rd Qu.:0.9989
## Max.     :1.0000

```

```
## NA's :30
## SMOTE, FALSE, FALSE, classif.xgboost
## Min. :0.0000
## 1st Qu.:0.9239
## Median :0.9803
## Mean :0.8764
## 3rd Qu.:0.9964
## Max. :1.0000
##
```

## Fazendo teste de normalidade

```
plotDensities(data = na.omit(df))
```



## Testando as diferenças

```
friedmanTest(df)
```

```
##
## Friedman's rank sum test
##
## data: df
## Friedman's chi-squared = 1811, df = 14, 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, classif.ksvm
## [1,]                                     FALSE
## [2,]                                     FALSE
## [3,]                                     FALSE
## [4,]                                     TRUE
## [5,]                                     TRUE
## [6,]                                     TRUE
## [7,]                                     TRUE
## [8,]                                     TRUE
## [9,]                                     TRUE
## [10,]                                    TRUE
## [11,]                                    TRUE
## [12,]                                    TRUE
## [13,]                                    FALSE
## [14,]                                    FALSE
## [15,]                                    FALSE
##      ADASYN, FALSE, FALSE, classif.randomForest
## [1,]                                     FALSE
## [2,]                                     FALSE
## [3,]                                     FALSE
## [4,]                                     TRUE
## [5,]                                     TRUE
## [6,]                                     TRUE
## [7,]                                     TRUE
## [8,]                                     TRUE
## [9,]                                     TRUE
## [10,]                                    TRUE
## [11,]                                    TRUE
## [12,]                                    TRUE
## [13,]                                    TRUE
## [14,]                                    FALSE
## [15,]                                    FALSE
##      ADASYN, FALSE, FALSE, classif.xgboost
## [1,]                                     FALSE
## [2,]                                     FALSE
## [3,]                                     FALSE
## [4,]                                     TRUE
## [5,]                                     TRUE
## [6,]                                     TRUE
## [7,]                                     TRUE
## [8,]                                     TRUE
## [9,]                                     TRUE
## [10,]                                    TRUE
## [11,]                                    TRUE
## [12,]                                    TRUE
## [13,]                                    TRUE
## [14,]                                    FALSE
## [15,]                                    FALSE
```

```

##      FALSE, FALSE, FALSE, classif.ksvm
## [1,]      TRUE
## [2,]      TRUE
## [3,]      TRUE
## [4,]     FALSE
## [5,]      TRUE
## [6,]      TRUE
## [7,]      TRUE
## [8,]      TRUE
## [9,]     FALSE
## [10,]    FALSE
## [11,]     TRUE
## [12,]     TRUE
## [13,]     TRUE
## [14,]     TRUE
## [15,]     TRUE
##      FALSE, FALSE, FALSE, classif.randomForest
## [1,]      TRUE
## [2,]      TRUE
## [3,]      TRUE
## [4,]      TRUE
## [5,]     FALSE
## [6,]     FALSE
## [7,]     FALSE
## [8,]     FALSE
## [9,]     FALSE
## [10,]     TRUE
## [11,]    FALSE
## [12,]     TRUE
## [13,]     TRUE
## [14,]     TRUE
## [15,]     TRUE
##      FALSE, FALSE, FALSE, classif.xgboost
## [1,]      TRUE
## [2,]      TRUE
## [3,]      TRUE
## [4,]      TRUE
## [5,]     FALSE
## [6,]     FALSE
## [7,]      TRUE
## [8,]      TRUE
## [9,]      TRUE
## [10,]     TRUE
## [11,]    FALSE
## [12,]     TRUE
## [13,]     TRUE
## [14,]     TRUE
## [15,]     TRUE
##      FALSE, FALSE, TRUE, classif.ksvm
## [1,]      TRUE
## [2,]      TRUE
## [3,]      TRUE
## [4,]      TRUE
## [5,]     FALSE

```

```

## [6,] TRUE
## [7,] FALSE
## [8,] FALSE
## [9,] FALSE
## [10,] TRUE
## [11,] FALSE
## [12,] TRUE
## [13,] TRUE
## [14,] TRUE
## [15,] TRUE
## FALSE, FALSE, TRUE, classif.randomForest
## [1,] TRUE
## [2,] TRUE
## [3,] TRUE
## [4,] TRUE
## [5,] FALSE
## [6,] TRUE
## [7,] FALSE
## [8,] FALSE
## [9,] FALSE
## [10,] TRUE
## [11,] FALSE
## [12,] TRUE
## [13,] TRUE
## [14,] TRUE
## [15,] TRUE
## FALSE, FALSE, TRUE, classif.xgboost FALSE, TRUE, FALSE, classif.ksvm
## [1,] TRUE TRUE
## [2,] TRUE TRUE
## [3,] TRUE TRUE
## [4,] FALSE FALSE
## [5,] FALSE TRUE
## [6,] TRUE TRUE
## [7,] FALSE TRUE
## [8,] FALSE TRUE
## [9,] FALSE FALSE
## [10,] FALSE FALSE
## [11,] FALSE TRUE
## [12,] TRUE TRUE
## [13,] TRUE TRUE
## [14,] TRUE TRUE
## [15,] TRUE TRUE
## FALSE, TRUE, FALSE, classif.randomForest
## [1,] TRUE
## [2,] TRUE
## [3,] TRUE
## [4,] TRUE
## [5,] FALSE
## [6,] FALSE
## [7,] FALSE
## [8,] FALSE
## [9,] FALSE
## [10,] TRUE
## [11,] FALSE

```



```

## [12,] TRUE
## [13,] TRUE
## [14,] TRUE
## [15,] TRUE
## FALSE, TRUE, FALSE, classif.xgboost
## [1,] TRUE
## [2,] TRUE
## [3,] TRUE
## [4,] TRUE
## [5,] TRUE
## [6,] TRUE
## [7,] TRUE
## [8,] TRUE
## [9,] TRUE
## [10,] TRUE
## [11,] TRUE
## [12,] FALSE
## [13,] TRUE
## [14,] TRUE
## [15,] TRUE
## SMOTE, FALSE, FALSE, classif.ksvm
## [1,] FALSE
## [2,] TRUE
## [3,] TRUE
## [4,] TRUE
## [5,] TRUE
## [6,] TRUE
## [7,] TRUE
## [8,] TRUE
## [9,] TRUE
## [10,] TRUE
## [11,] TRUE
## [12,] TRUE
## [13,] FALSE
## [14,] FALSE
## [15,] TRUE
## SMOTE, FALSE, FALSE, classif.randomForest
## [1,] FALSE
## [2,] FALSE
## [3,] FALSE
## [4,] TRUE
## [5,] TRUE
## [6,] TRUE
## [7,] TRUE
## [8,] TRUE
## [9,] TRUE
## [10,] TRUE
## [11,] TRUE
## [12,] TRUE
## [13,] FALSE
## [14,] FALSE
## [15,] FALSE
## SMOTE, FALSE, FALSE, classif.xgboost
## [1,] FALSE

```

```
## [2,] FALSE
## [3,] FALSE
## [4,] TRUE
## [5,] TRUE
## [6,] TRUE
## [7,] TRUE
## [8,] TRUE
## [9,] TRUE
## [10,] TRUE
## [11,] TRUE
## [12,] TRUE
## [13,] TRUE
## [14,] FALSE
## [15,] FALSE
```

## Plotando grafico de Critical Difference

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

