

R Notebook

Parametros:

```
Measure = Accuracy
Columns = sampling, weight_space, ruspool
Performance = tuning_measure
Filter keys = imba.rate
Filter values = 0.001
```

```
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      :600  Mode :logical  
## classif.randomForest:600 FALSE:1440  
## classif.xgboost    :600  TRUE :360  
##                   NA's :0  
##  
##  
##  
##           measure      sampling    ruspool  
## 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.:0.9897  1st Qu.:0.98917  1st Qu.:0.35692  
## Median :0.9927  Median :0.99101  Median :0.75246  
## Mean :0.9702  Mean :0.96745  Mean :0.66386  
## 3rd Qu.:0.9987  3rd Qu.:0.99624  3rd Qu.:0.95373  
## Max. :1.0000  Max. :1.00000  Max. :1.00000  
## NA's :87  NA's :87  NA's :87  
## iteration_count      dataset      imba.rate  
## Min. :1  abalone : 45  Min. :0.001  
## 1st Qu.:1  adult : 45  1st Qu.:0.001  
## Median :2  bank : 45  Median :0.001  
## Mean :2  car : 45  Mean :0.001  
## 3rd Qu.:3  cardiotocography-10clases: 45  3rd Qu.:0.001  
## Max. :3  cardiotocography-3clases : 45  Max. :0.001  
## NA's :87  (Other) :1530
```

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] 120 5
```

```

# Renomeando a variavel
df = df_tec_wide_residual

summary(df)

```

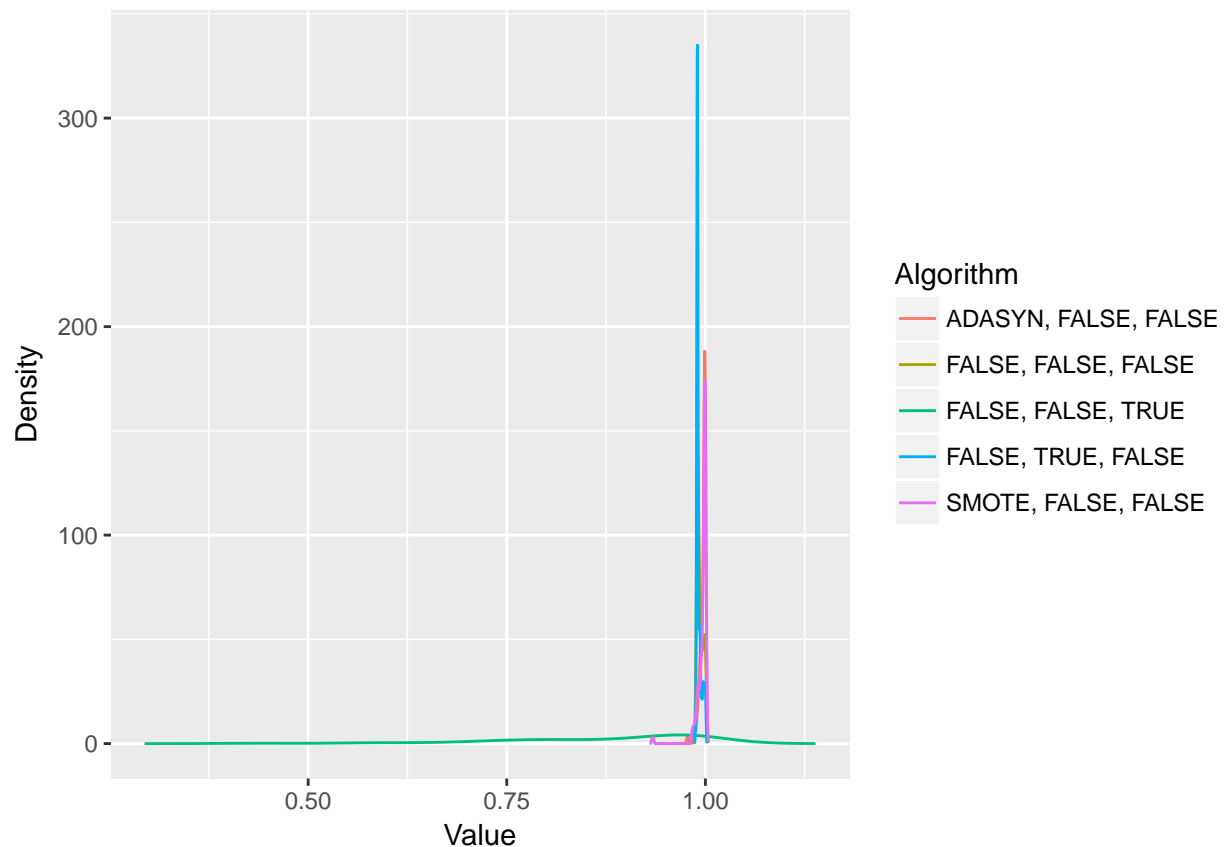
```

## ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## Min. :0.9337 Min. :0.9878 Min. :0.4327
## 1st Qu.:0.9951 1st Qu.:0.9900 1st Qu.:0.8011
## Median :0.9986 Median :0.9921 Median :0.9303
## Mean :0.9963 Mean :0.9933 Mean :0.8765
## 3rd Qu.:0.9995 3rd Qu.:0.9958 3rd Qu.:0.9821
## Max. :1.0000 Max. :1.0000 Max. :1.0000
## NA's :11 NA's :2 NA's :3
## FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## Min. :0.9883 Min. :0.9337
## 1st Qu.:0.9898 1st Qu.:0.9947
## Median :0.9900 Median :0.9986
## Mean :0.9918 Mean :0.9962
## 3rd Qu.:0.9930 3rd Qu.:0.9999
## Max. :1.0000 Max. :1.0000
## NA's :5 NA's :8

```

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 = 208.02, 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 grafico de Critical Difference

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

