

# R Notebook

## Parametros:

Measure = Matthews correlation coefficient  
Columns = sampling, weight\_space, underbagging  
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.  
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      :1230  Mode :logical
## classif.randomForest:1230 FALSE:2952
## classif.rusboost   :    0  TRUE :738
## classif.xgboost    :1230  NA's :0
##
##
##
##           measure      sampling  underbagging
## 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.   :-0.21201  Min.   :-0.45710
## 1st Qu.: 0.3764  1st Qu.: 0.06131  1st Qu.: 0.05637
## Median : 0.8057  Median : 0.55190  Median : 0.23378
## Mean   : 0.6629  Mean   : 0.49274  Mean   : 0.32193
## 3rd Qu.: 0.9728  3rd Qu.: 0.82456  3rd Qu.: 0.56442
## Max.   : 1.0000  Max.   : 1.00000  Max.   : 1.00000
## NA's   :54      NA's   :54      NA's   :54
## 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    :54      (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))

# 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

head(df)
```

```
## ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## 1 0.09430275 0.08009145 0.20313757
## 2 0.23594212 0.29868169 0.55327083
## 3 0.39737200 0.31222405 0.37539770
## 4 0.00000000 0.00000000 0.07136273
## 5 0.14032488 0.14801251 0.11577975
## 6 0.05815156 0.19892100 0.36690665
## FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## 1 0.07628792 0.11483245
## 2 0.29457094 0.24240609
## 3 0.36193425 0.41403016
## 4 0.00000000 0.00000000
## 5 0.14801251 0.09045042
## 6 0.14294212 0.11319614
```

```
summary(df)
```

```
## ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## Min. :-0.34981 Min. :-0.39928 Min. :-0.3331
## 1st Qu.: 0.07436 1st Qu.: 0.05527 1st Qu.: 0.1419
## Median : 0.22571 Median : 0.20258 Median : 0.3327
## Mean : 0.31790 Mean : 0.29438 Mean : 0.3839
## 3rd Qu.: 0.54785 3rd Qu.: 0.49997 3rd Qu.: 0.6253
## Max. : 0.98682 Max. : 0.99489 Max. : 0.9863
```

```
## NA's      :8                      NA's      :3
## FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## Min.      :-0.39928    Min.      :-0.33229
## 1st Qu.: 0.04571    1st Qu.: 0.08111
## Median : 0.21210    Median : 0.22498
## Mean      : 0.29105    Mean      : 0.32267
## 3rd Qu.: 0.49564    3rd Qu.: 0.54083
## Max.      : 1.00000    Max.      : 0.98479
## NA's      :3          NA's      :4
```

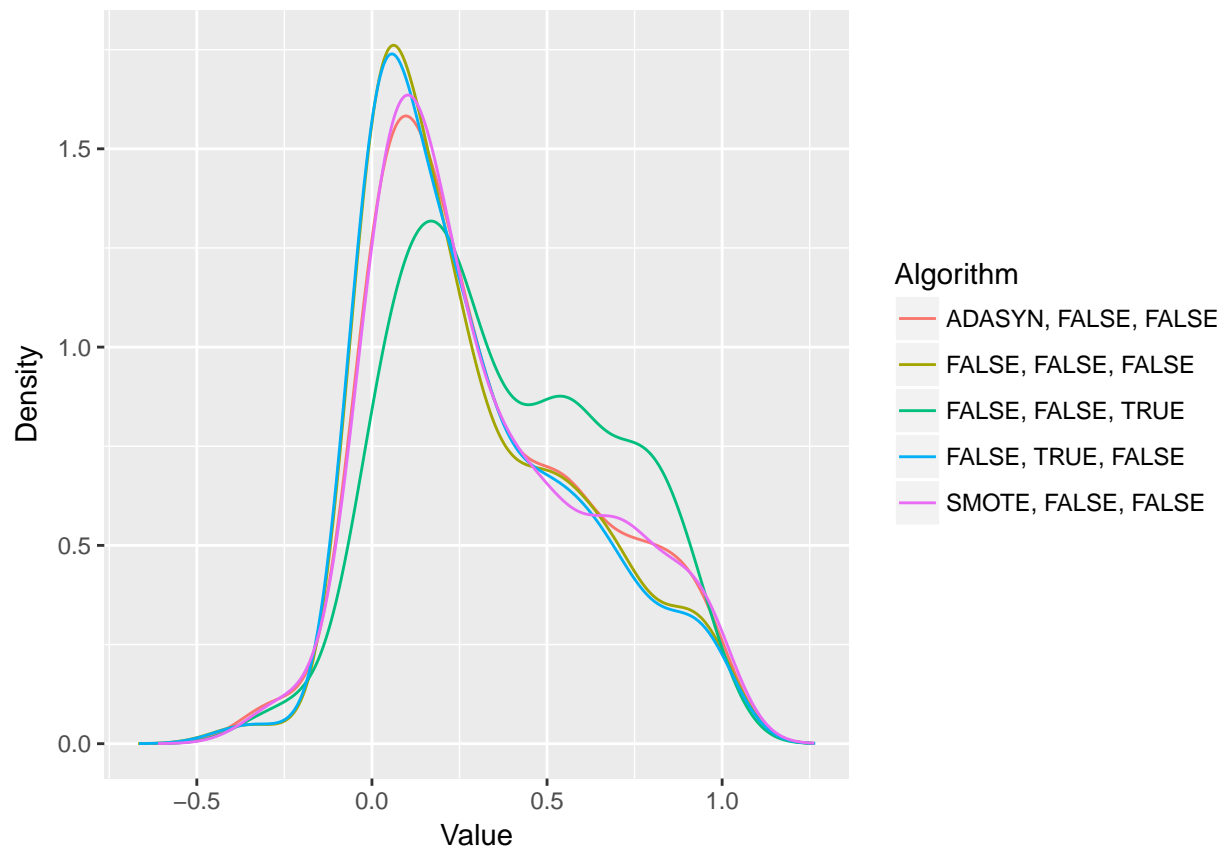
## Verificando a média de cada coluna selecionada

```
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 ADASYN, FALSE, FALSE = 0.31790269979045"
## [1] "Media da coluna FALSE, FALSE, FALSE = 0.294377299065821"
## [1] "Media da coluna FALSE, FALSE, TRUE = 0.383919063724294"
## [1] "Media da coluna FALSE, TRUE, FALSE = 0.291054167402681"
## [1] "Media da coluna SMOTE, FALSE, FALSE = 0.322667075906413"
```

## 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 = 84.95, 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 os ranks

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

```
## ADASYN, FALSE, FALSE  FALSE, FALSE, FALSE  FALSE, FALSE, TRUE
##          2.928862          3.382114          2.341463
##  FALSE, TRUE, FALSE  SMOTE, FALSE, FALSE
##          3.504065          2.843496
```

## Plotando grafico de Critical Difference

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

