

# R Notebook

## Parametros:

```
Measure = Accuracy
Columns = sampling, weight_space, underbagging
Performance = holdout_measure_residual
Filter keys = imba.rate
Filter values = 0.03
```

```
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)){
  dots = paste0(params$filter_keys, " == '",params$filter_values, "'")
  ds = filter_(ds, .dots = dots)
}
```

```
summary(ds)
```

```
##           learner      weight_space
## classif.ksvm      :990 Mode :logical
## classif.randomForest:990 FALSE:2376
## classif.rusboost   : 0 TRUE :594
## classif.xgboost    :990 NA's :0
##
##
##
##           measure      sampling      underbagging
## Accuracy           :2970 ADASYN: 594 Mode :logical
## Area under the curve : 0 FALSE :1782 FALSE:2376
## F1 measure           : 0 SMOTE : 594 TRUE :594
## G-mean               : 0 NA's :0
## Matthews correlation coefficient: 0
##
##
## tuning_measure      holdout_measure      holdout_measure_residual
## Min. :0.09041 Min. :0.02655 Min. :0.0346
## 1st Qu.:0.96926 1st Qu.:0.96647 1st Qu.:0.3599
## Median :0.98130 Median :0.97619 Median :0.6882
## Mean :0.95405 Mean :0.94750 Mean :0.6478
## 3rd Qu.:0.99560 3rd Qu.:0.99045 3rd Qu.:0.9438
## Max. :1.00000 Max. :1.00000 Max. :1.0000
## NA's :57 NA's :57 NA's :57
## iteration_count      dataset      imba.rate
## Min. :1 abalone : 45 Min. :0.03
## 1st Qu.:1 adult : 45 1st Qu.:0.03
## Median :2 annealing : 45 Median :0.03
## Mean :2 arrhythmia : 45 Mean :0.03
## 3rd Qu.:3 balance-scale: 45 3rd Qu.:0.03
## Max. :3 bank : 45 Max. :0.03
## NA's :57 (Other) :2700
```

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] 198 5
```

```
# Renomeando a variavel
df = df_tec_wide_residual
```

```
head(df)
```

```
## ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## 1 0.3810826 0.3311463 0.6504465
## 2 0.4831227 0.4176651 0.5304448
## 3 0.7721519 0.7074543 0.7172996
## 4 0.9807692 0.9807692 0.8301282
## 5 0.2203548 0.2819795 0.2698413
## 6 0.6854839 0.6610215 0.5295699
## FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## 1 0.3214872 0.3728814
## 2 0.4185678 0.4875267
## 3 0.7074543 0.7566807
## 4 0.9807692 0.9807692
## 5 0.2819795 0.2885154
## 6 0.6610215 0.6634409
```

```
summary(df)
```

```
## ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## Min. :0.03709 Min. :0.03709 Min. :0.1530
## 1st Qu.:0.35839 1st Qu.:0.32845 1st Qu.:0.5423
## Median :0.67244 Median :0.63108 Median :0.7948
## Mean :0.63890 Mean :0.61970 Mean :0.7163
## 3rd Qu.:0.94371 3rd Qu.:0.95116 3rd Qu.:0.9097
## Max. :0.99992 Max. :1.00000 Max. :0.9998
```

```
## NA's :9          NA's :2          NA's :2
## FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## Min. :0.03709    Min. :0.03709
## 1st Qu.:0.32156   1st Qu.:0.36088
## Median :0.64181   Median :0.66534
## Mean :0.62201     Mean :0.64194
## 3rd Qu.:0.95739   3rd Qu.:0.94905
## Max. :0.99992     Max. :1.00000
## NA's :1          NA's :5
```

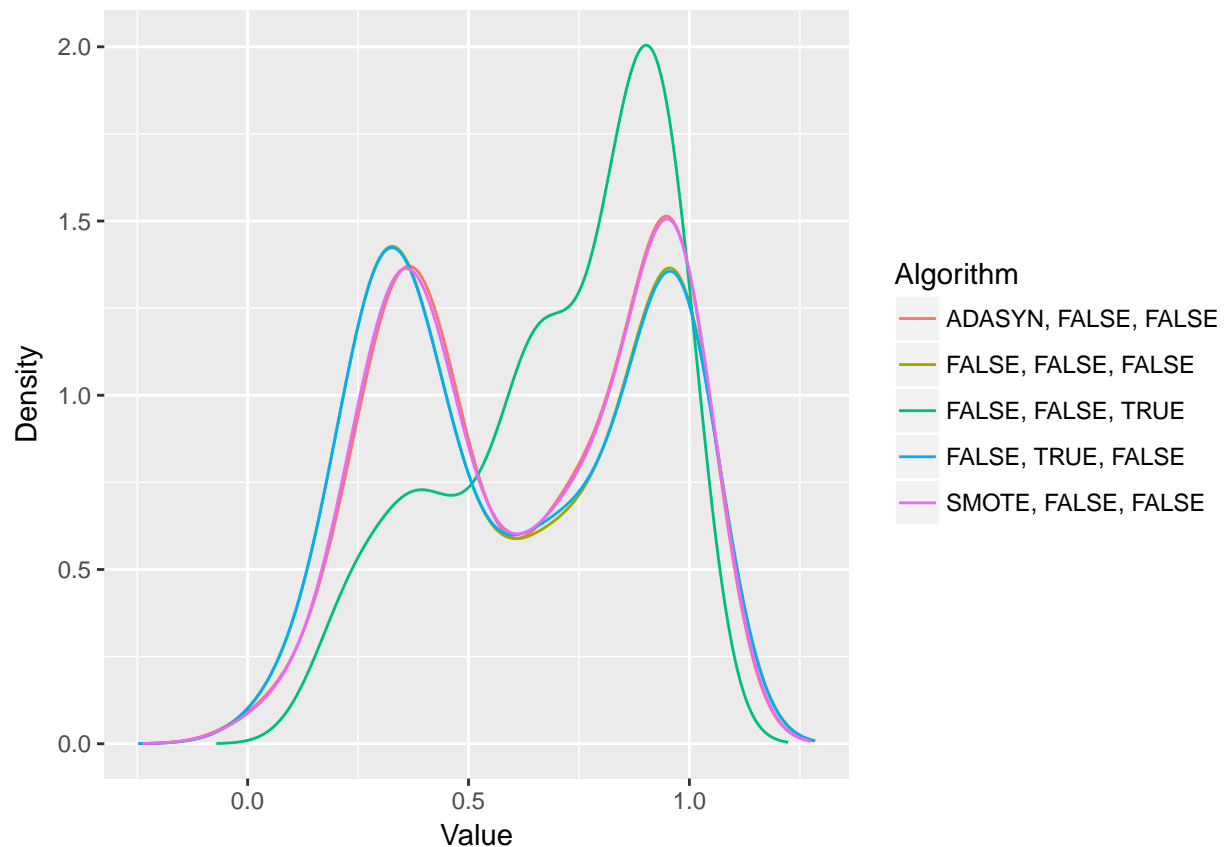
## Verificando a média de cada coluna selecionada

```
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 = 0.638897930547164"
## [1] "Media da coluna FALSE, FALSE, FALSE = 0.619701792142005"
## [1] "Media da coluna FALSE, FALSE, TRUE = 0.716250984811708"
## [1] "Media da coluna FALSE, TRUE, FALSE = 0.622011954138431"
## [1] "Media da coluna SMOTE, FALSE, FALSE = 0.64194281819364"
```

## 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 = 48.504, df = 4, p-value = 7.409e-10
```

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

```
## [2,]          FALSE          TRUE
## [3,]          TRUE          FALSE
## [4,]          FALSE          TRUE
## [5,]          TRUE          FALSE
```

## Plotando os ranks

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

```
## ADASYN, FALSE, FALSE  FALSE, FALSE, FALSE  FALSE, FALSE, TRUE
##          2.909091          3.431818          2.522727
##  FALSE, TRUE, FALSE  SMOTE, FALSE, FALSE
##          3.368687          2.767677
```

## Plotando grafico de Critical Difference

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

