

R Notebook

Parametros:

Measure = Area under the curve
Columns = sampling, weight_space, underbagging
Performance = tuning_measure
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.  
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)){
  ds = filter_at(ds, .vars = params$filter_keys, .vars_predicate = any_vars(. == params$filter_values))
}
```

```
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           : 0 ADASYN: 594 Mode :logical
## Area under the curve :2970 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. :0.3023 Min. :0.0000 Min. :0.00057
## 1st Qu.:0.9338 1st Qu.:0.8603 1st Qu.:0.69645
## Median :0.9963 Median :0.9835 Median :0.89271
## Mean :0.9356 Mean :0.8947 Mean :0.82476
## 3rd Qu.:0.9999 3rd Qu.:0.9998 3rd Qu.:0.98444
## Max. :1.0000 Max. :1.0000 Max. :1.00000
## NA's :66 NA's :66 NA's :66
## 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 :66 (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

summary(df)
```

```
## ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## Min. :0.7697 Min. :0.4574 Min. :0.4610
## 1st Qu.:0.9985 1st Qu.:0.8460 1st Qu.:0.8298
## Median :0.9999 Median :0.9689 Median :0.9461
## Mean :0.9946 Mean :0.8990 Mean :0.8937
## 3rd Qu.:1.0000 3rd Qu.:0.9973 3rd Qu.:0.9917
## Max. :1.0000 Max. :1.0000 Max. :1.0000
## NA's :8 NA's :2 NA's :2
## FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## Min. :0.4652 Min. :0.8739
## 1st Qu.:0.8328 1st Qu.:0.9986
## Median :0.9664 Median :0.9999
## Mean :0.8968 Mean :0.9963
## 3rd Qu.:0.9970 3rd Qu.:1.0000
## Max. :1.0000 Max. :1.0000
## NA's :5 NA's :5
```

Verificando a média de cada coluna selecionada

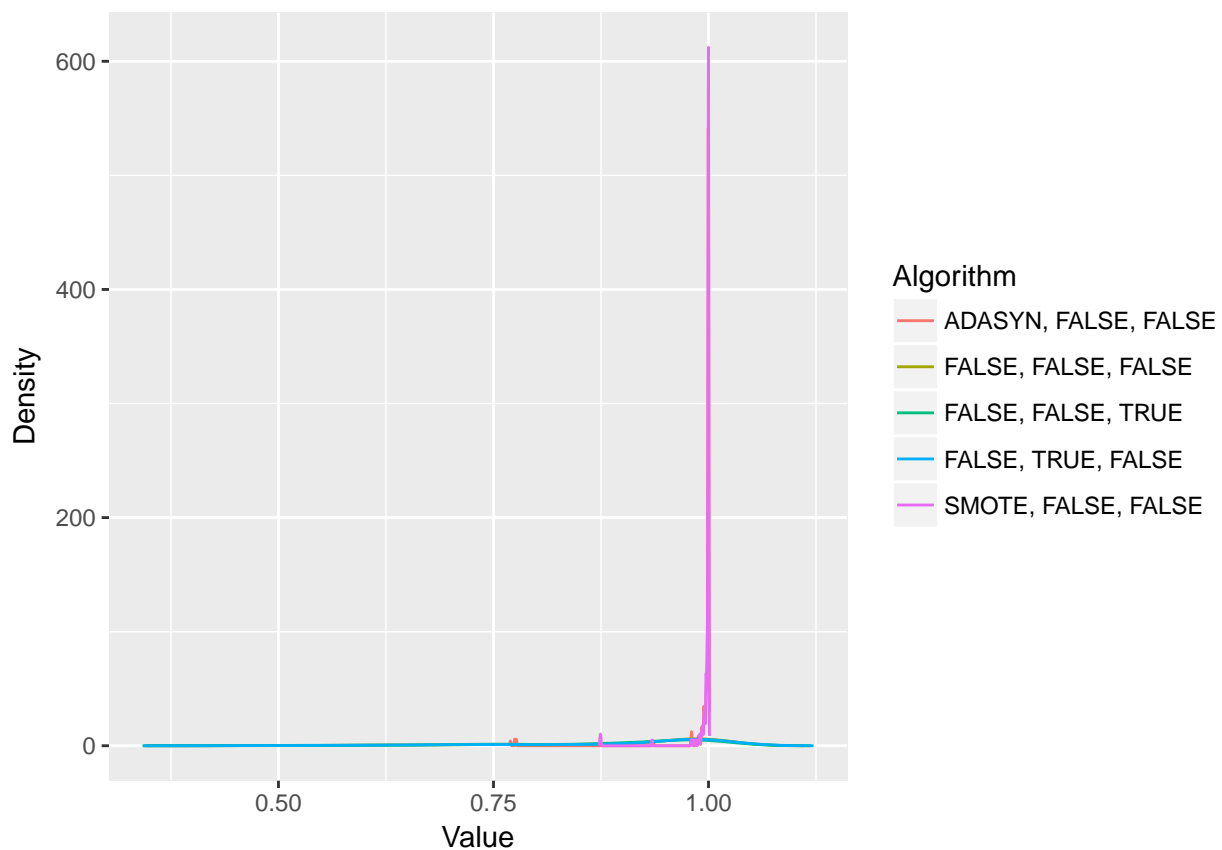
```
for(i in (1:dim(df)[2])){
  #print(df[,i])
}
```

```
print(paste("Media da coluna ", colnames(df)[i], " = ", mean(df[,i], na.rm = TRUE), sep=""))
}

## [1] "Media da coluna ADASYN, FALSE, FALSE = 0.994581095091789"
## [1] "Media da coluna FALSE, FALSE, FALSE = 0.898997602638118"
## [1] "Media da coluna FALSE, FALSE, TRUE = 0.893676966104373"
## [1] "Media da coluna FALSE, TRUE, FALSE = 0.896801058736894"
## [1] "Media da coluna SMOTE, FALSE, FALSE = 0.996284109522124"
```

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 = 453.48, df = 4, p-value < 2.2e-16
```

Testando as diferenças 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) {})
```

