

R Notebook

Parametros:

Measure = G-mean
Columns = sampling, weight_space, underbagging, learner
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.  
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           : 0  ADASYN: 594  Mode :logical
## Area under the curve : 0  FALSE :1782 FALSE:2376
## F1 measure          : 0  SMOTE : 594  TRUE :594
## G-mean              :2970           NA's :0
## Matthews correlation coefficient: 0
##
##
## tuning_measure  holdout_measure  holdout_measure_residual
## Min. :0.0000  Min. :0.0000  Min. :0.0000
## 1st Qu.:0.6338 1st Qu.:0.2132 1st Qu.:0.1828
## Median :0.9453 Median :0.7348 Median :0.4920
## Mean :0.7583  Mean :0.6032 Mean :0.4882
## 3rd Qu.:0.9933 3rd Qu.:0.9533 3rd Qu.:0.8073
## Max. :1.0000  Max. :1.0000  Max. :1.0000
## NA's :48      NA's :48      NA's :48
## 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 :48      (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] 66 15
```

```
# Renomeando a variavel
df = df_tec_wide_residual

summary(df)
```

```
## ADASYN, FALSE, FALSE, classif.ksvm
## Min. :0.00000
## 1st Qu.:0.06152
## Median :0.23505
## Mean :0.29941
## 3rd Qu.:0.43582
## Max. :0.93658
## NA's :2
## ADASYN, FALSE, FALSE, classif.randomForest
## Min. :0.0000
## 1st Qu.:0.2922
## Median :0.5020
## Mean :0.5209
## 3rd Qu.:0.7565
## Max. :0.9999
## NA's :7
## ADASYN, FALSE, FALSE, classif.xgboost FALSE, FALSE, FALSE, classif.ksvm
## Min. :0.1159 Min. :0.00000
## 1st Qu.:0.3359 1st Qu.:0.02855
## Median :0.6272 Median :0.19604
## Mean :0.5949 Mean :0.28228
## 3rd Qu.:0.8579 3rd Qu.:0.37211
## Max. :1.0000 Max. :0.97624
##
```

```

## FALSE, FALSE, FALSE, classif.randomForest
## Min.      :0.0000
## 1st Qu.:0.1440
## Median :0.4242
## Mean    :0.4676
## 3rd Qu.:0.7622
## Max.     :1.0000
## NA's     :1
## FALSE, FALSE, FALSE, classif.xgboost FALSE, FALSE, TRUE, classif.ksvm
## Min.      :0.0000      Min.      :0.09623
## 1st Qu.:0.1606      1st Qu.:0.35911
## Median :0.5247      Median :0.58534
## Mean    :0.4793      Mean    :0.57035
## 3rd Qu.:0.7284      3rd Qu.:0.75784
## Max.     :1.0000      Max.     :0.98933
##
## FALSE, FALSE, TRUE, classif.randomForest
## Min.      :0.2294
## 1st Qu.:0.6400
## Median :0.8321
## Mean    :0.7601
## 3rd Qu.:0.9465
## Max.     :0.9999
## NA's     :1
## FALSE, FALSE, TRUE, classif.xgboost FALSE, TRUE, FALSE, classif.ksvm
## Min.      :0.2111      Min.      :0.000000
## 1st Qu.:0.5867      1st Qu.:0.004324
## Median :0.8141      Median :0.193975
## Mean    :0.7329      Mean    :0.269673
## 3rd Qu.:0.9270      3rd Qu.:0.364723
## Max.     :0.9999      Max.     :0.976240
##
## FALSE, TRUE, FALSE, classif.randomForest
## Min.      :0.0000
## 1st Qu.:0.1506
## Median :0.4189
## Mean    :0.4572
## 3rd Qu.:0.7346
## Max.     :0.9999
## NA's     :2
## FALSE, TRUE, FALSE, classif.xgboost SMOTE, FALSE, FALSE, classif.ksvm
## Min.      :0.0000      Min.      :0.00000
## 1st Qu.:0.1609      1st Qu.:0.04287
## Median :0.5392      Median :0.20835
## Mean    :0.4822      Mean    :0.27166
## 3rd Qu.:0.7363      3rd Qu.:0.40696
## Max.     :1.0000      Max.     :0.98106
##
## SMOTE, FALSE, FALSE, classif.randomForest
## Min.      :0.0000
## 1st Qu.:0.2714
## Median :0.5726
## Mean    :0.5482
## 3rd Qu.:0.8389

```

```
## Max.      :1.0000
## NA's      :3
## SMOTE, FALSE, FALSE, classif.xgboost
## Min.      :0.1090
## 1st Qu.:0.3197
## Median :0.5890
## Mean      :0.5895
## 3rd Qu.:0.8641
## Max.      :1.0000
##
```

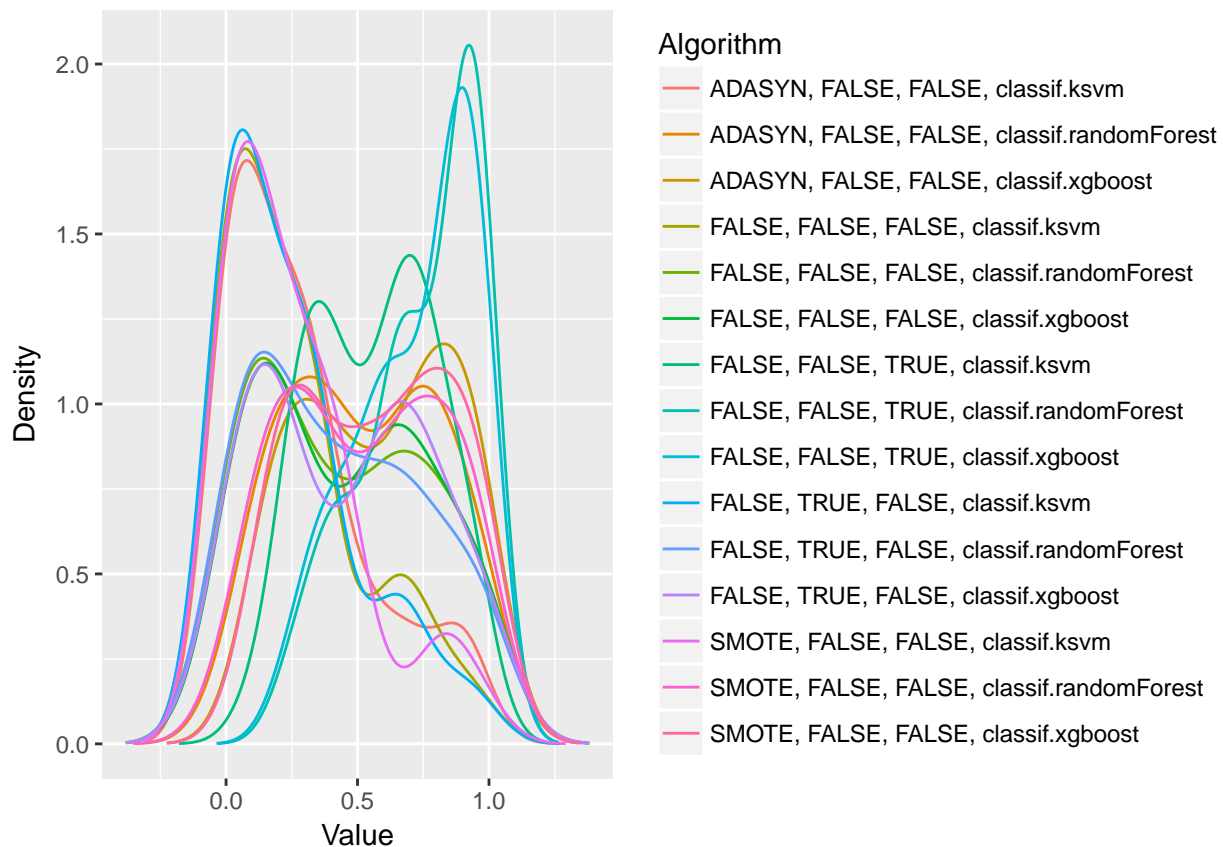
Verificando a média de cada coluna selecionada

```
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, classif.ksvm = 0.299411504550748"
## [1] "Media da coluna ADASYN, FALSE, FALSE, classif.randomForest = 0.520934760616434"
## [1] "Media da coluna ADASYN, FALSE, FALSE, classif.xgboost = 0.594929504937969"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.ksvm = 0.282279187238992"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.randomForest = 0.467572106238412"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.xgboost = 0.47925141449756"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.ksvm = 0.570345699459489"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.randomForest = 0.76014513516345"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.xgboost = 0.732852110282045"
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.ksvm = 0.269672553427394"
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.randomForest = 0.457188650737349"
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.xgboost = 0.482208738315524"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.ksvm = 0.271661823893925"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.randomForest = 0.5482213375776"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.xgboost = 0.589463437013953"
```

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 = 475.94, 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,]                                TRUE
## [3,]                                TRUE
## [4,]                                FALSE
## [5,]                                TRUE
## [6,]                                TRUE
## [7,]                                TRUE
```

```

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

```

```

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

```



```

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

```

```

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

