

R Notebook

Parametros:

```
Measure = G-mean
Columns = sampling, weight_space, ruspool, learner
Performance = holdout_measure_residual
Filter keys = NULL
Filter values = NULL
```

```
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      :3420  Mode :logical  
## classif.randomForest:3420 FALSE:8208  
## classif.xgboost    :3420  TRUE :2052  
##                   NA's :0  
##  
##  
##  
##           measure      sampling      ruspool  
## Accuracy           : 0 ADASYN:2052  Mode :logical  
## Area under the curve : 0 FALSE :6156  FALSE:8208  
## F1 measure           : 0 SMOTE :2052  TRUE :2052  
## G-mean              :10260           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.:0.4958  1st Qu.:0.0000  1st Qu.:0.1064  
## Median :0.9108  Median :0.7004  Median :0.4331  
## Mean :0.7066  Mean :0.5513  Mean :0.4500  
## 3rd Qu.:0.9931  3rd Qu.:0.9414  3rd Qu.:0.7846  
## Max. :1.0000  Max. :1.0000  Max. :1.0000  
## NA's :318    NA's :318    NA's :318  
## iteration_count      dataset      imba.rate  
## Min. :1      abalone      : 180  Min. :0.0010  
## 1st Qu.:1      adult      : 180  1st Qu.:0.0100  
## Median :2      bank      : 180  Median :0.0300  
## Mean :2      car      : 180  Mean :0.0286  
## 3rd Qu.:3      cardiotocography-10clases: 180  3rd Qu.:0.0500  
## Max. :3      cardiotocography-3clases : 180  Max. :0.0500  
## NA's :318    (Other)      :9180
```

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] 228 15

# Removendo linhas com NA's
df_tec_wide_residual = na.omit(df_tec_wide_residual)

# Renomeando a variavel
df = df_tec_wide_residual

summary(df)

## ADASYN, FALSE, FALSE, classif.ksvm
## Min. :0.00000
## 1st Qu.:0.01087
## Median :0.17978
## Mean :0.24894
## 3rd Qu.:0.42124
## Max. :0.98958
## ADASYN, FALSE, FALSE, classif.randomForest
## Min. :0.0000
## 1st Qu.:0.2250
## Median :0.4746
## Mean :0.4965
## 3rd Qu.:0.7579
## Max. :0.9999
## ADASYN, FALSE, FALSE, classif.xgboost FALSE, FALSE, FALSE, classif.ksvm
## Min. :0.0000 Min. :0.0000
## 1st Qu.:0.2460 1st Qu.:0.0000
## Median :0.4941 Median :0.1816
## Mean :0.5173 Mean :0.2270
## 3rd Qu.:0.7820 3rd Qu.:0.3183
## Max. :0.9999 Max. :0.9864
## FALSE, FALSE, FALSE, classif.randomForest

```

```

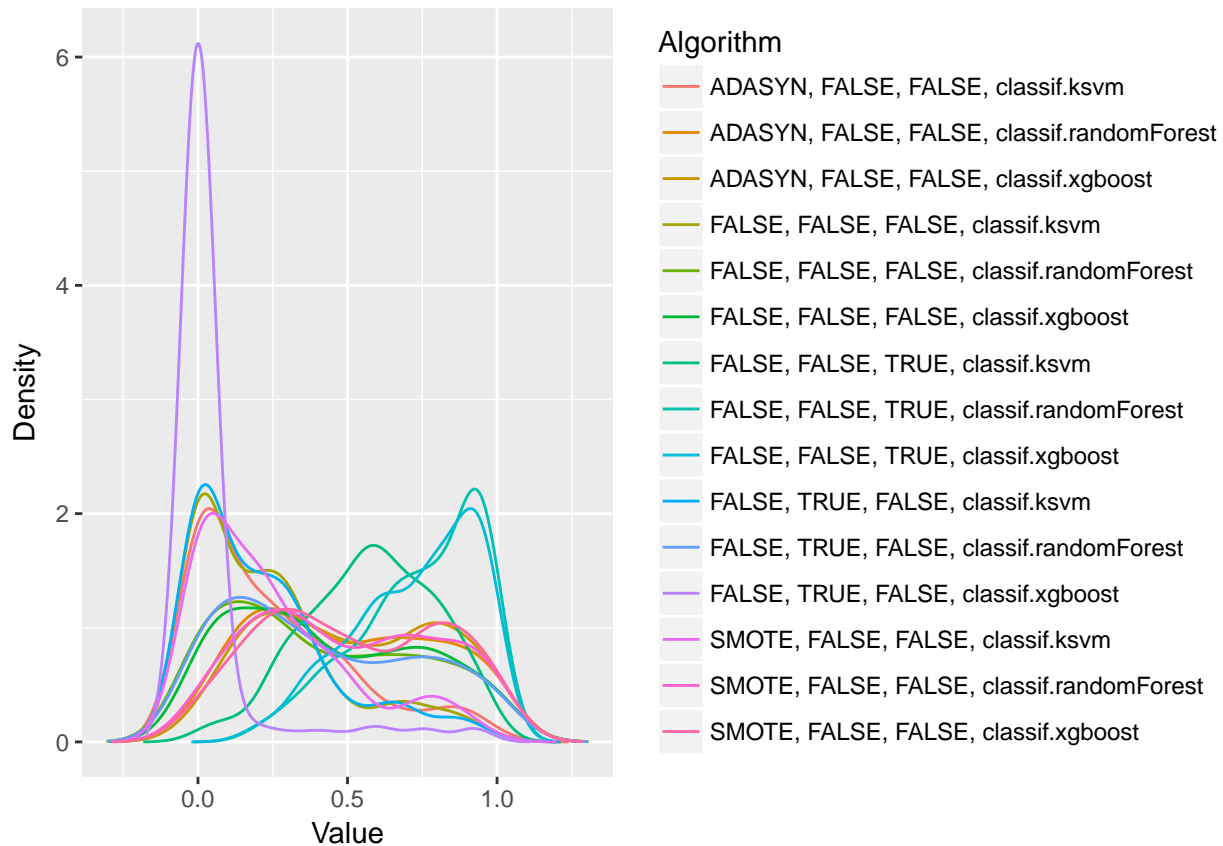
## Min.      :0.0000
## 1st Qu.:0.1362
## Median :0.3328
## Mean    :0.4154
## 3rd Qu.:0.6944
## Max.    :0.9999
## FALSE, FALSE, FALSE, classif.xgboost FALSE, FALSE, TRUE, classif.ksvm
## Min.      :0.0000      Min.      :0.02295
## 1st Qu.:0.1606      1st Qu.:0.42594
## Median :0.3670      Median :0.57920
## Mean    :0.4349      Mean    :0.57801
## 3rd Qu.:0.7114      3rd Qu.:0.74847
## Max.    :0.9999      Max.    :0.99115
## FALSE, FALSE, TRUE, classif.randomForest
## Min.      :0.1800
## 1st Qu.:0.6159
## Median :0.7779
## Mean    :0.7445
## 3rd Qu.:0.9243
## Max.    :0.9999
## FALSE, FALSE, TRUE, classif.xgboost FALSE, TRUE, FALSE, classif.ksvm
## Min.      :0.1744      Min.      :0.0000
## 1st Qu.:0.5875      1st Qu.:0.0000
## Median :0.7698      Median :0.1557
## Mean    :0.7277      Mean    :0.2170
## 3rd Qu.:0.9169      3rd Qu.:0.3158
## Max.    :0.9999      Max.    :0.9864
## FALSE, TRUE, FALSE, classif.randomForest
## Min.      :0.0000
## 1st Qu.:0.1528
## Median :0.3357
## Mean    :0.4130
## 3rd Qu.:0.7041
## Max.    :0.9999
## FALSE, TRUE, FALSE, classif.xgboost SMOTE, FALSE, FALSE, classif.ksvm
## Min.      :0.00000      Min.      :0.00000
## 1st Qu.:0.00000      1st Qu.:0.03374
## Median :0.00000      Median :0.18855
## Mean    :0.05512      Mean    :0.24978
## 3rd Qu.:0.00000      3rd Qu.:0.38897
## Max.    :0.93476      Max.    :0.95632
## SMOTE, FALSE, FALSE, classif.randomForest
## Min.      :0.0000
## 1st Qu.:0.2347
## Median :0.4744
## Mean    :0.4978
## 3rd Qu.:0.7379
## Max.    :1.0000
## SMOTE, FALSE, FALSE, classif.xgboost
## Min.      :0.0000
## 1st Qu.:0.2735
## Median :0.5072
## Mean    :0.5179
## 3rd Qu.:0.7957

```

```
## Max. :0.9999
```

Fazendo teste de normalidade

```
plotDensities(data = df)
```



Testando as diferencas

```
friedmanTest(df)
```

```
##  
## Friedman's rank sum test  
##  
## data: df  
## Friedman's chi-squared = 1406.6, 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,]                               TRUE
## [12,]                               TRUE
## [13,]                               FALSE
## [14,]                               TRUE
## [15,]                               TRUE
##      ADASYN, FALSE, FALSE, classif.randomForest
## [1,]                                TRUE
## [2,]                                FALSE
## [3,]                                FALSE
## [4,]                                TRUE
## [5,]                                TRUE
## [6,]                                TRUE
## [7,]                                FALSE
## [8,]                                TRUE
## [9,]                                TRUE
## [10,]                               TRUE
## [11,]                               TRUE
## [12,]                               TRUE
## [13,]                               TRUE
## [14,]                               FALSE
## [15,]                               FALSE
##      ADASYN, FALSE, FALSE, classif.xgboost
## [1,]                                TRUE
## [2,]                                FALSE
## [3,]                                FALSE
## [4,]                                TRUE
## [5,]                                TRUE
## [6,]                                TRUE
## [7,]                                FALSE
## [8,]                                TRUE
## [9,]                                TRUE
## [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,] TRUE
## [12,] TRUE
## [13,] FALSE
## [14,] TRUE
## [15,] TRUE
## FALSE, FALSE, FALSE, classif.randomForest
## [1,] TRUE
## [2,] TRUE
## [3,] TRUE
## [4,] TRUE
## [5,] FALSE
## [6,] FALSE
## [7,] TRUE
## [8,] TRUE
## [9,] TRUE
## [10,] TRUE
## [11,] FALSE
## [12,] TRUE
## [13,] TRUE
## [14,] TRUE
## [15,] TRUE
## FALSE, FALSE, FALSE, classif.xgboost
## [1,] TRUE
## [2,] TRUE
## [3,] TRUE
## [4,] TRUE
## [5,] FALSE
## [6,] FALSE
## [7,] TRUE
## [8,] TRUE
## [9,] TRUE
## [10,] TRUE
## [11,] FALSE
## [12,] TRUE
## [13,] TRUE
## [14,] TRUE
## [15,] TRUE
## FALSE, FALSE, TRUE, classif.ksvm
## [1,] TRUE
## [2,] FALSE
## [3,] FALSE
## [4,] TRUE
## [5,] TRUE
## [6,] TRUE
## [7,] FALSE
## [8,] TRUE
## [9,] TRUE
## [10,] TRUE
## [11,] TRUE

```

```

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

```



```

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

```

```
## [8,] TRUE
## [9,] TRUE
## [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) {})
```

