

R Notebook

Parametros:

```
Measure = G-mean
Columns = sampling, weight_space, underbagging, learner
Performance = tuning_measure
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.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)){
  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.rusboost   :  0  TRUE :2052
## classif.xgboost    :3420  NA's :0
##
##
##
##          measure      sampling  underbagging
## 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.:0.6205  1st Qu.:0.0000  1st Qu.:0.1683
## Median :0.9426  Median :0.7071  Median :0.4879
## Mean :0.7570  Mean :0.5918  Mean :0.4829
## 3rd Qu.:0.9950  3rd Qu.:0.9547  3rd Qu.:0.7996
## Max. :1.0000  Max. :1.0000  Max. :1.0000
## NA's :189  NA's :189  NA's :189
## 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      cardiocography-10clases: 180  3rd Qu.:0.0500
## Max. :3      cardiocography-3clases : 180  Max. :0.0500
## NA's :189  (Other) :9180
```

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] 228 15
```

```
# Renomeando a variavel
df = df_tec_wide_residual

summary(df)
```

```
## ADASYN, FALSE, FALSE, classif.ksvm
## Min. :0.7000
## 1st Qu.:0.9772
## Median :0.9948
## Mean :0.9786
## 3rd Qu.:0.9987
## Max. :1.0000
## NA's :7
## ADASYN, FALSE, FALSE, classif.randomForest
## Min. :0.6918
## 1st Qu.:0.9853
## Median :0.9966
## Mean :0.9849
## 3rd Qu.:0.9992
## Max. :1.0000
## NA's :22
## ADASYN, FALSE, FALSE, classif.xgboost FALSE, FALSE, FALSE, classif.ksvm
## Min. :0.6674 Min. :0.0000
## 1st Qu.:0.9840 1st Qu.:0.0000
## Median :0.9940 Median :0.2765
## Mean :0.9796 Mean :0.3487
## 3rd Qu.:0.9987 3rd Qu.:0.6449
## Max. :1.0000 Max. :1.0000
##
```

```

## FALSE, FALSE, FALSE, classif.randomForest
## Min.      :0.0000
## 1st Qu.:0.2082
## Median :0.6063
## Mean    :0.5619
## 3rd Qu.:0.8944
## Max.    :1.0000
## NA's    :5
## FALSE, FALSE, FALSE, classif.xgboost FALSE, FALSE, TRUE, classif.ksvm
## Min.      :0.0000      Min.      :0.3521
## 1st Qu.:0.3202      1st Qu.:0.6169
## Median :0.6876      Median :0.7665
## Mean    :0.6082      Mean    :0.7403
## 3rd Qu.:0.9031      3rd Qu.:0.8711
## Max.    :1.0000      Max.    :1.0000
##
## FALSE, FALSE, TRUE, classif.randomForest
## Min.      :0.4493
## 1st Qu.:0.8136
## Median :0.9107
## Mean    :0.8758
## 3rd Qu.:0.9769
## Max.    :1.0000
## NA's    :5
## FALSE, FALSE, TRUE, classif.xgboost FALSE, TRUE, FALSE, classif.ksvm
## Min.      :0.4794      Min.      :0.0000
## 1st Qu.:0.8014      1st Qu.:0.0000
## Median :0.9021      Median :0.2637
## Mean    :0.8613      Mean    :0.3385
## 3rd Qu.:0.9629      3rd Qu.:0.6124
## Max.    :1.0000      Max.    :1.0000
##
## FALSE, TRUE, FALSE, classif.randomForest
## Min.      :0.0000
## 1st Qu.:0.2314
## Median :0.5616
## Mean    :0.5600
## 3rd Qu.:0.9101
## Max.    :1.0000
## NA's    :7
## FALSE, TRUE, FALSE, classif.xgboost SMOTE, FALSE, FALSE, classif.ksvm
## Min.      :0.0000      Min.      :0.7097
## 1st Qu.:0.3007      1st Qu.:0.9777
## Median :0.6804      Median :0.9952
## Mean    :0.6061      Mean    :0.9783
## 3rd Qu.:0.8990      3rd Qu.:0.9994
## Max.    :1.0000      Max.    :1.0000
##
## SMOTE, FALSE, FALSE, classif.randomForest
## Min.      :0.6760
## 1st Qu.:0.9866
## Median :0.9971
## Mean    :0.9858
## 3rd Qu.:0.9995

```

```
## Max.      :1.0000
## NA's      :17
## SMOTE, FALSE, FALSE, classif.xgboost
## Min.      :0.6787
## 1st Qu.:0.9851
## Median :0.9948
## Mean      :0.9848
## 3rd Qu.:0.9987
## Max.      :1.0000
##
```

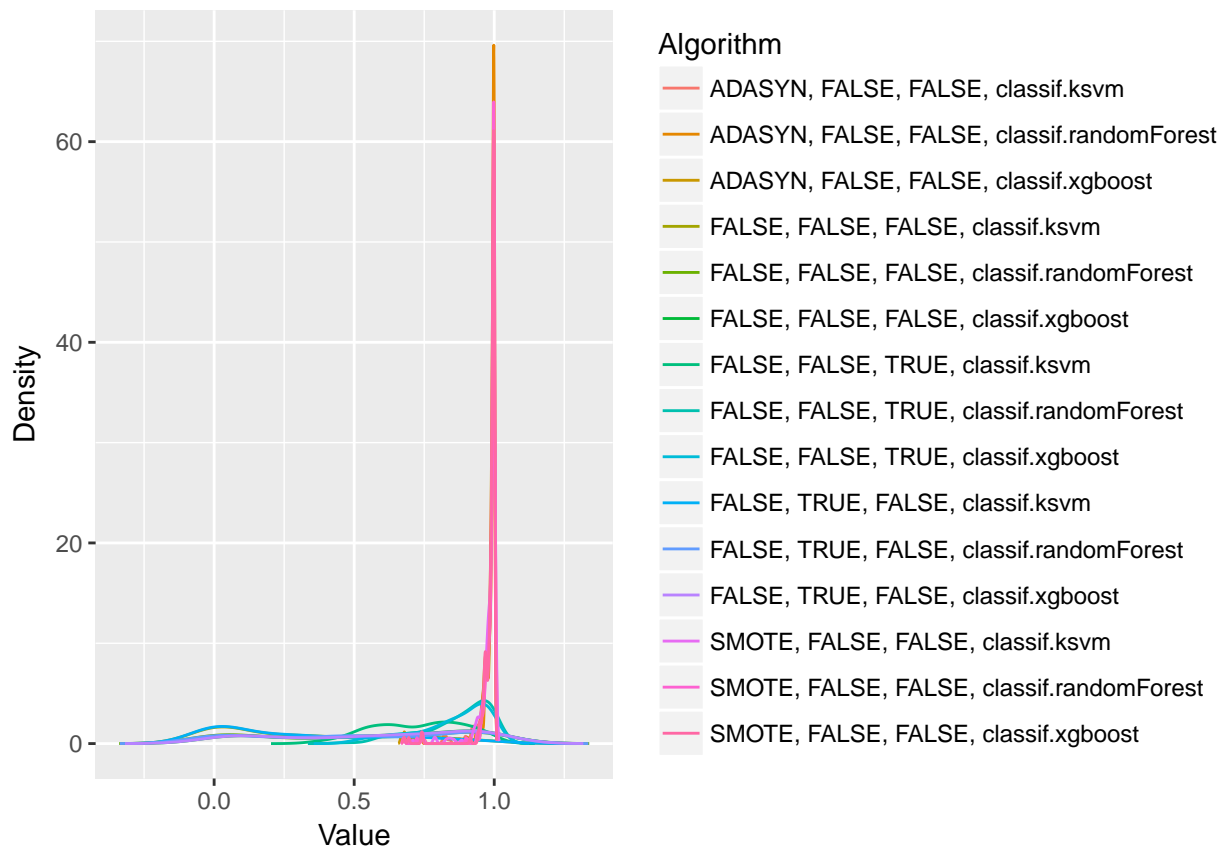
Verificando a média de cada coluna selecionada

```
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, classif.ksvm = 0.978567943379877"
## [1] "Media da coluna ADASYN, FALSE, FALSE, classif.randomForest = 0.984877155968814"
## [1] "Media da coluna ADASYN, FALSE, FALSE, classif.xgboost = 0.979586724290639"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.ksvm = 0.348723686105921"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.randomForest = 0.561931298896259"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.xgboost = 0.608150448036422"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.ksvm = 0.740325217225188"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.randomForest = 0.875791183504656"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.xgboost = 0.861338451020602"
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.ksvm = 0.338506105906646"
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.randomForest = 0.560002840503865"
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.xgboost = 0.606070183340777"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.ksvm = 0.978280106172245"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.randomForest = 0.985825082661221"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.xgboost = 0.984807584020767"
```

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 = 2104.8, 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,] FALSE
## [3,] FALSE
## [4,] TRUE
## [5,] TRUE
## [6,] TRUE
## [7,] TRUE
```

```

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

```

```

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

```

```

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

```

Plotando grafico de Critical Difference

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

