

R Notebook

Parametros:

```
Measure = F1 measure
Columns = sampling, weight_space, underbagging, learner
Performance = holdout_measure_residual
Filter keys = imba.rate
Filter values = 0.05
```

```
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      :1230  Mode :logical
## classif.randomForest:1230 FALSE:2952
## classif.rusboost   :  0  TRUE :738
## classif.xgboost    :1230  NA's :0
##
##
##
##          measure      sampling  underbagging
## Accuracy           :  0  ADASYN: 738  Mode :logical
## Area under the curve :  0  FALSE :2214 FALSE:2952
## F1 measure          :3690  SMOTE : 738  TRUE :738
## G-mean              :  0              NA's :0
## Matthews correlation coefficient:  0
##
##
## tuning_measure  holdout_measure  holdout_measure_residual
## Min. :0.0000  Min. :0.0000  Min. :0.00000
## 1st Qu.:0.3333  1st Qu.:0.1000  1st Qu.:0.07022
## Median :0.8198  Median :0.5000  Median :0.32530
## Mean :0.6671  Mean :0.4905  Mean :0.39891
## 3rd Qu.:0.9848  3rd Qu.:0.8333  3rd Qu.:0.73016
## Max. :1.0000  Max. :1.0000  Max. :1.00000
## NA's :51      NA's :51      NA's :51
## iteration_count      dataset      imba.rate
## Min. :1      abalone : 45  Min. :0.05
## 1st Qu.:1      adult : 45  1st Qu.:0.05
## Median :2      annealing : 45  Median :0.05
## Mean :2      arrhythmia : 45  Mean :0.05
## 3rd Qu.:3      balance-scale: 45  3rd Qu.:0.05
## Max. :3      bank : 45  Max. :0.05
## NA's :51      (Other) :3420
```

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] 82 15
```

```
# Renomeando a variavel
df = df_tec_wide_residual

summary(df)
```

```
## ADASYN, FALSE, FALSE, classif.ksvm
## Min. :0.0000
## 1st Qu.:0.0254
## Median :0.1450
## Mean :0.2594
## 3rd Qu.:0.3977
## Max. :0.9895
## NA's :1
## ADASYN, FALSE, FALSE, classif.randomForest
## Min. :0.0000
## 1st Qu.:0.1695
## Median :0.4212
## Mean :0.4415
## 3rd Qu.:0.7133
## Max. :0.9812
## NA's :5
## ADASYN, FALSE, FALSE, classif.xgboost FALSE, FALSE, FALSE, classif.ksvm
## Min. :0.0000 Min. :0.0000
## 1st Qu.:0.1643 1st Qu.:0.0000
## Median :0.4248 Median :0.1292
## Mean :0.4712 Mean :0.2189
## 3rd Qu.:0.7880 3rd Qu.:0.3049
## Max. :0.9899 Max. :0.9949
##
```

```

## FALSE, FALSE, FALSE, classif.randomForest
## Min.      :0.00000
## 1st Qu.:0.08468
## Median :0.30383
## Mean    :0.37823
## 3rd Qu.:0.67164
## Max.    :0.97237
## NA's    :1
## FALSE, FALSE, FALSE, classif.xgboost FALSE, FALSE, TRUE, classif.ksvm
## Min.      :0.00000      Min.      :0.0003612
## 1st Qu.:0.08164      1st Qu.:0.2422053
## Median :0.29218      Median :0.4740756
## Mean    :0.40056      Mean    :0.4949795
## 3rd Qu.:0.72131      3rd Qu.:0.7567327
## Max.    :0.98600      Max.    :0.9895300
##
## FALSE, FALSE, TRUE, classif.randomForest
## Min.      :0.00463
## 1st Qu.:0.30767
## Median :0.67030
## Mean    :0.58081
## 3rd Qu.:0.85519
## Max.    :0.98381
## NA's    :3
## FALSE, FALSE, TRUE, classif.xgboost FALSE, TRUE, FALSE, classif.ksvm
## Min.      :0.001115      Min.      :0.0000
## 1st Qu.:0.297551      1st Qu.:0.0000
## Median :0.712325      Median :0.1219
## Mean    :0.575417      Mean    :0.2083
## 3rd Qu.:0.849056      3rd Qu.:0.3022
## Max.    :0.982204      Max.    :0.9949
##
## FALSE, TRUE, FALSE, classif.randomForest
## Min.      :0.00000
## 1st Qu.:0.07677
## Median :0.32566
## Mean    :0.38785
## 3rd Qu.:0.67472
## Max.    :1.00000
## NA's    :3
## FALSE, TRUE, FALSE, classif.xgboost SMOTE, FALSE, FALSE, classif.ksvm
## Min.      :0.00000      Min.      :0.00000
## 1st Qu.:0.09852      1st Qu.:0.02247
## Median :0.30378      Median :0.15839
## Mean    :0.39343      Mean    :0.25228
## 3rd Qu.:0.69440      3rd Qu.:0.37444
## Max.    :1.00000      Max.    :0.96181
##
## SMOTE, FALSE, FALSE, classif.randomForest
## Min.      :0.0000
## 1st Qu.:0.1649
## Median :0.4002
## Mean    :0.4553
## 3rd Qu.:0.7570

```

```
## Max.      :0.9832
## NA's      :4
## SMOTE, FALSE, FALSE, classif.xgboost
## Min.      :0.006122
## 1st Qu.:0.176907
## Median :0.448104
## Mean      :0.475252
## 3rd Qu.:0.782271
## Max.      :0.988373
##
```

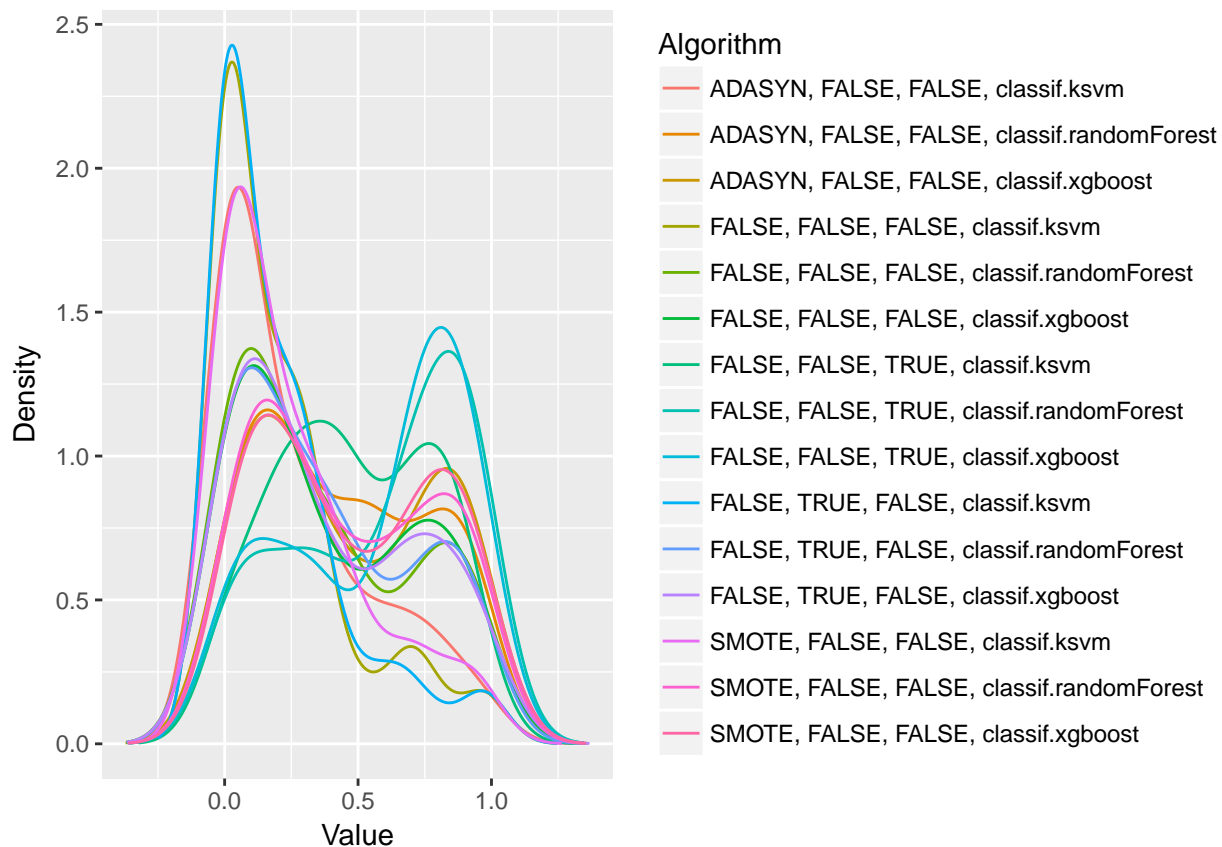
Verificando a média de cada coluna selecionada

```
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 = 0.259448788538673"
## [1] "Media da coluna ADASYN, FALSE, FALSE, classif.randomForest = 0.441474157964024"
## [1] "Media da coluna ADASYN, FALSE, FALSE, classif.xgboost = 0.471167010966672"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.ksvm = 0.218865427361255"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.randomForest = 0.378232764365048"
## [1] "Media da coluna FALSE, FALSE, FALSE, classif.xgboost = 0.400556109264134"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.ksvm = 0.494979456445285"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.randomForest = 0.580806118393993"
## [1] "Media da coluna FALSE, FALSE, TRUE, classif.xgboost = 0.575416728305805"
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.ksvm = 0.208308011677194"
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.randomForest = 0.387847506789137"
## [1] "Media da coluna FALSE, TRUE, FALSE, classif.xgboost = 0.393430237712404"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.ksvm = 0.252281727176568"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.randomForest = 0.455285475909701"
## [1] "Media da coluna SMOTE, FALSE, FALSE, classif.xgboost = 0.475252067555195"
```

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 = 313.09, 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,]                                FALSE
## [6,]                                TRUE
## [7,]                                TRUE
```

```

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

```

```

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

```



```

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

```

```

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

