

R Notebook

Parametros:

Measure = G-mean
Columns = sampling, weight_space, underbagging
Performance = holdout_measure_residual
Filter keys = imba.rate
Filter values = 0.001

```
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      :600  Mode :logical
## classif.randomForest:600 FALSE:1440
## classif.rusboost   : 0  TRUE :360
## classif.xgboost    :600 NA's :0
##
##
##
##          measure      sampling      underbagging
## Accuracy           : 0  ADASYN: 360  Mode :logical
## Area under the curve : 0  FALSE :1080 FALSE:1440
## F1 measure          : 0  SMOTE : 360  TRUE :360
## G-mean              :1800           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.:0.5941 1st Qu.:0.0000 1st Qu.:0.1173
## Median :0.9638 Median :0.7062 Median :0.4257
## Mean :0.7528  Mean :0.5598  Mean :0.4404
## 3rd Qu.:0.9988 3rd Qu.:0.9645 3rd Qu.:0.7589
## 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. :0.001
## 1st Qu.:1      adult      : 45  1st Qu.:0.001
## Median :2      bank      : 45  Median :0.001
## Mean :2      car      : 45  Mean :0.001
## 3rd Qu.:3      cardiocography-10clases: 45 3rd Qu.:0.001
## Max. :3      cardiocography-3clases : 45 Max. :0.001
## NA's :48      (Other) :1530
```

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] 120 5
```

```
# Renomeando a variavel
df = df_tec_wide_residual

head(df)
```

```
## ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## 1 0.22988211 0.1284832 0.5605192
## 2 0.20871555 0.1892378 0.6578023
## 3 0.04517624 0.0000000 0.6206804
## 4 0.10753762 0.2694472 0.7831986
## 5 0.00000000 0.2507360 0.6066726
## 6 0.09607398 0.2088070 0.4125283
## FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## 1 0.1254198 0.22399174
## 2 0.1993466 0.21728496
## 3 0.0000000 0.05765257
## 4 0.2694472 0.17055729
## 5 0.2507360 0.04134491
## 6 0.2088070 0.08997243
```

```
summary(df)
```

```
## ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## Min. :0.0000 Min. :0.00000 Min. :0.03782
## 1st Qu.:0.1167 1st Qu.:0.05798 1st Qu.:0.57537
## Median :0.3290 Median :0.27402 Median :0.77487
## Mean :0.3990 Mean :0.35248 Mean :0.70548
## 3rd Qu.:0.6557 3rd Qu.:0.57527 3rd Qu.:0.89383
## Max. :0.9999 Max. :0.99993 Max. :0.99992
```

```
## NA's :8          NA's :2          NA's :1
## FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## Min. :0.00000 Min. :0.0000
## 1st Qu.:0.07198 1st Qu.:0.1064
## Median :0.28412 Median :0.3229
## Mean :0.34852 Mean :0.3916
## 3rd Qu.:0.55555 3rd Qu.:0.6847
## Max. :0.99993 Max. :1.0000
## NA's :2          NA's :3
```

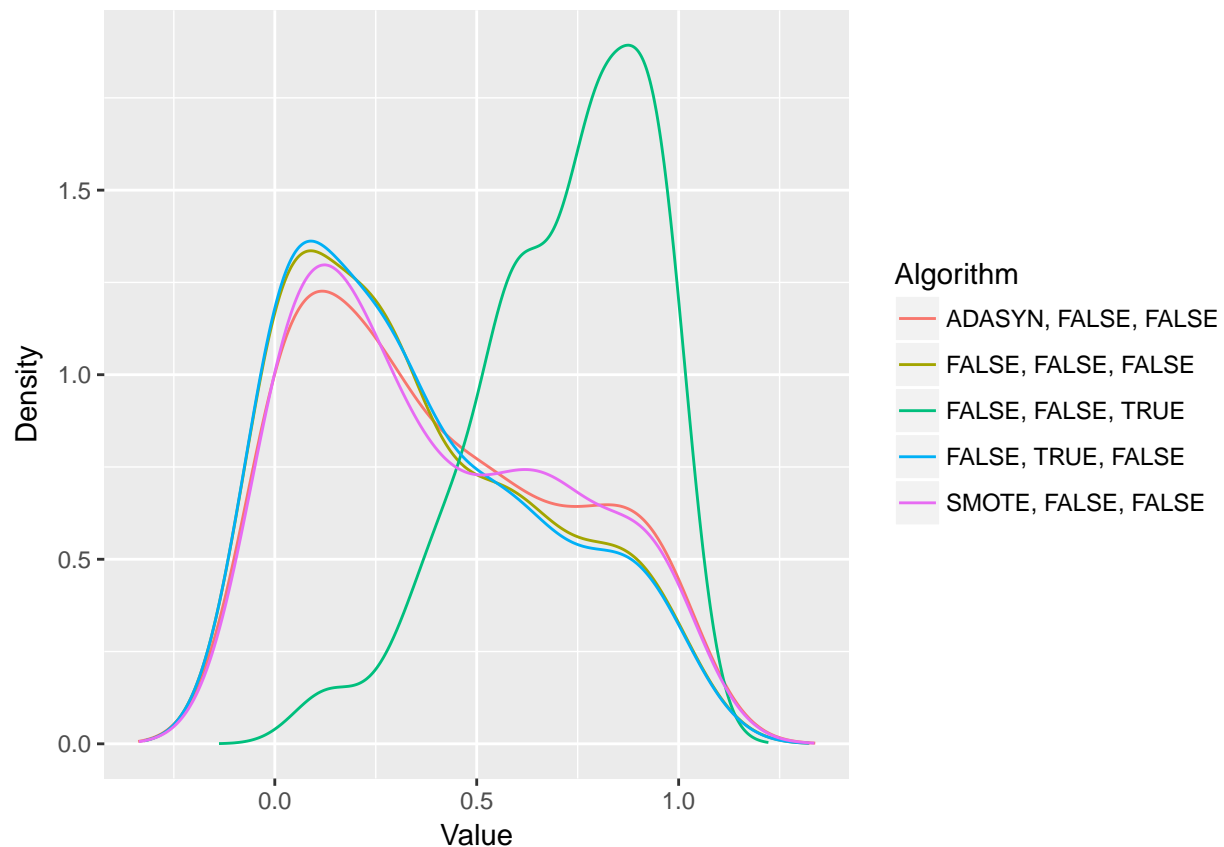
Verificando a média de cada coluna selecionada

```
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 = 0.398953573733686"
## [1] "Media da coluna FALSE, FALSE, FALSE = 0.352477249991163"
## [1] "Media da coluna FALSE, FALSE, TRUE = 0.705480499725451"
## [1] "Media da coluna FALSE, TRUE, FALSE = 0.348524512198664"
## [1] "Media da coluna SMOTE, FALSE, FALSE = 0.391633443381693"
```

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 = 188.73, df = 4, 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 FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## [1,]          FALSE          FALSE          TRUE
## [2,]          FALSE          FALSE          TRUE
## [3,]           TRUE          TRUE         FALSE
## [4,]           TRUE          FALSE          TRUE
## [5,]          FALSE          FALSE          TRUE
##      FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## [1,]           TRUE          FALSE
```

```
## [2,]          FALSE          FALSE
## [3,]          TRUE           TRUE
## [4,]          FALSE          TRUE
## [5,]          TRUE           FALSE
```

Plotando os ranks

```
print(colMeans(rankMatrix(df)))
```

```
## ADASYN, FALSE, FALSE  FALSE, FALSE, FALSE  FALSE, FALSE, TRUE
##           3.175000           3.608333           1.291667
##  FALSE, TRUE, FALSE  SMOTE, FALSE, FALSE
##           3.766667           3.158333
```

Plotando grafico de Critical Difference

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

