

R Notebook

Parametros:

Measure = Area under the curve
Columns = sampling, weight_space, underbagging
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.  
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      :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 :10260 FALSE :6156  FALSE:8208
## F1 measure          :    0  SMOTE :2052  TRUE :2052
## G-mean              :    0              NA's :0
## Matthews correlation coefficient: 0
##
##
## tuning_measure  holdout_measure  holdout_measure_residual
## Min. :0.3023  Min. :0.0000  Min. :0.0000
## 1st Qu.:0.9325  1st Qu.:0.8620  1st Qu.:0.7067
## Median :0.9967  Median :0.9831  Median :0.8932
## Mean :0.9380  Mean :0.8972  Mean :0.8310
## 3rd Qu.:1.0000  3rd Qu.:0.9999  3rd Qu.:0.9819
## Max. :1.0000  Max. :1.0000  Max. :1.0000
## NA's :243  NA's :243  NA's :243
## 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 :243  (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] 684 5
```

```
# Renomeando a variavel
df = df_tec_wide_residual

head(df)
```

```
## ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## 1 0.9965924 0.5777335 0.6095754
## 2 0.9965924 0.5777335 0.6095754
## 3 0.9879838 0.5950041 0.6592267
## 4 0.9815223 0.6418531 0.6658918
## 5 NA 0.6022761 0.8041582
## 6 NA 0.6022761 0.8041582
## FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## 1 0.5776046 0.9977386
## 2 0.5776046 0.9977386
## 3 0.5958792 0.9897728
## 4 NA 0.9803999
## 5 0.6355048 0.9974304
## 6 0.6355048 0.9974304
```

```
summary(df)
```

```
## ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## Min. :0.7697 Min. :0.4564 Min. :0.4610
## 1st Qu.:0.9989 1st Qu.:0.8535 1st Qu.:0.8513
## Median :0.9999 Median :0.9706 Median :0.9483
## Mean :0.9945 Mean :0.9029 Mean :0.8985
## 3rd Qu.:1.0000 3rd Qu.:0.9978 3rd Qu.:0.9915
## Max. :1.0000 Max. :1.0000 Max. :1.0000
```

```
## NA's :34          NA's :9          NA's :6
## FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## Min. :0.4652      Min. :0.7440
## 1st Qu.:0.8428    1st Qu.:0.9990
## Median :0.9704    Median :1.0000
## Mean :0.9016      Mean :0.9952
## 3rd Qu.:0.9979    3rd Qu.:1.0000
## Max. :1.0000      Max. :1.0000
## NA's :14          NA's :18
```

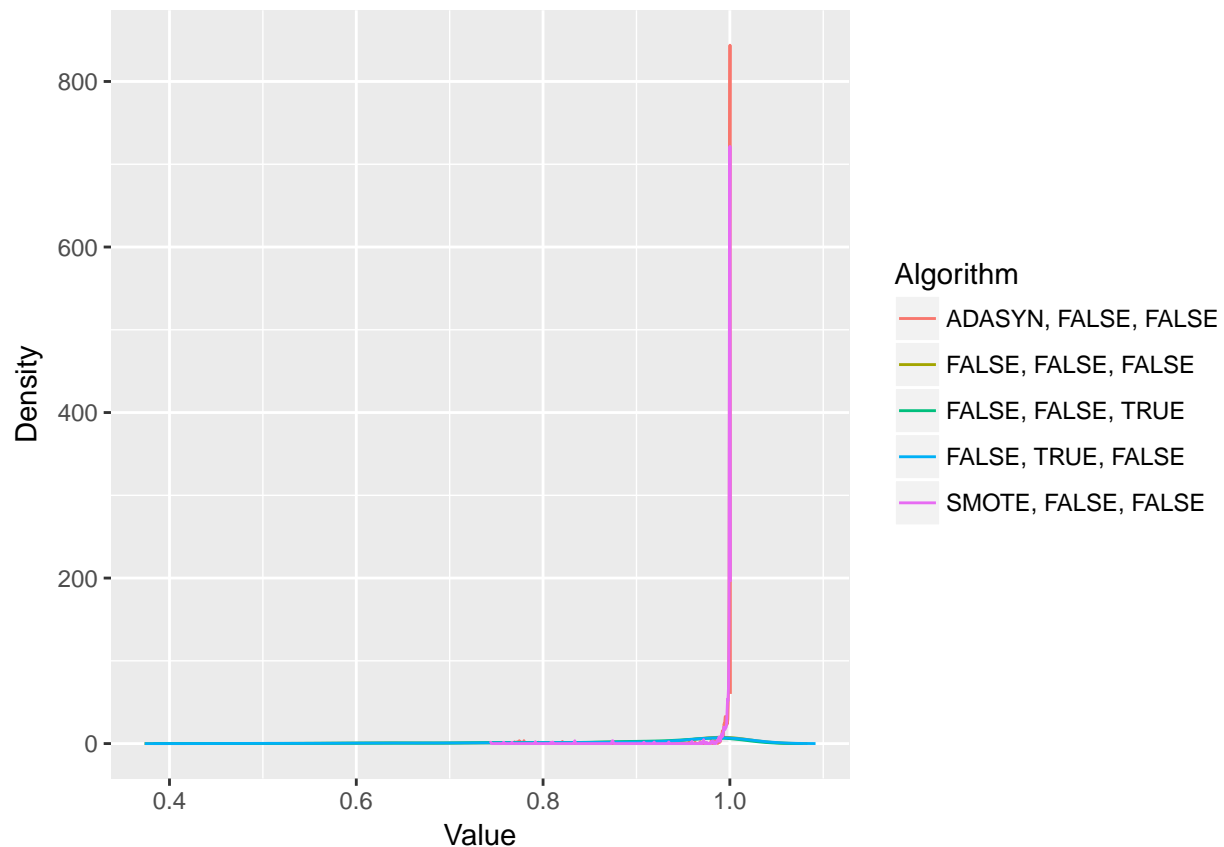
Verificando a média de cada coluna selecionada

```
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 = 0.994475716536219"
## [1] "Media da coluna FALSE, FALSE, FALSE = 0.902918453297002"
## [1] "Media da coluna FALSE, FALSE, TRUE = 0.898493166428649"
## [1] "Media da coluna FALSE, TRUE, FALSE = 0.901629902394603"
## [1] "Media da coluna SMOTE, FALSE, FALSE = 0.995166271984517"
```

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 = 1488.7, 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          TRUE          TRUE
## [2,]          TRUE          FALSE          TRUE
## [3,]          TRUE          TRUE          FALSE
## [4,]          TRUE          FALSE          TRUE
## [5,]          FALSE          TRUE          TRUE
##      FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## [1,]          TRUE          FALSE
```

```
## [2,]          FALSE          TRUE
## [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
##          1.795322          3.639620          4.155702
##  FALSE, TRUE, FALSE  SMOTE, FALSE, FALSE
##          3.722222          1.687135
```

Plotando grafico de Critical Difference

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

