R Notebook

Parametros:

Measure = Matthews correlation coefficient
Columns = sampling, weight_space, underbagging
Performance = tuning_measure
Filter keys = imba.rate
Filter values = 0.03

```r
library("scmamp")
library(dplyr)
```

Tratamento dos dados

Carregando data set compilado

```r
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       cardiotocography-10clases:  900   3rd Qu.:0.0500
##  Max.   :3       cardiotocography-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        :990   Mode :logical
##  classif.randomForest:990   FALSE:2376
##  classif.rusboost    :  0   TRUE :594
##  classif.xgboost     :990   NA's :0
##
##
##
##                              measure       sampling    underbagging
##  Accuracy                        :   0   ADASYN: 594   Mode :logical
##  Area under the curve            :   0   FALSE :1782   FALSE:2376
##  F1 measure                      :   0   SMOTE : 594   TRUE :594
##  G-mean                          :   0                 NA's :0
##  Matthews correlation coefficient:2970
##
##
##  tuning_measure     holdout_measure   holdout_measure_residual
##  Min.   :-0.05673   Min.   :-0.1757   Min.   :-0.4658
##  1st Qu.: 0.33347   1st Qu.: 0.0000   1st Qu.: 0.0391
##  Median : 0.83196   Median : 0.5030   Median : 0.2116
##  Mean   : 0.66187   Mean   : 0.4753   Mean   : 0.3111
##  3rd Qu.: 0.98596   3rd Qu.: 0.8126   3rd Qu.: 0.5286
##  Max.   : 1.00000   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.   :0.03
##  1st Qu.:1       adult        :  45   1st Qu.:0.03
##  Median :2       annealing    :  45   Median :0.03
##  Mean   :2       arrhythmia   :  45   Mean   :0.03
##  3rd Qu.:3       balance-scale:  45   3rd Qu.:0.03
##  Max.   :3       bank         :  45   Max.   :0.03
##  NA's   :48      (Other)      :2700
```

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

```r
# 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$performan

# 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] 198   5
```

```r
# Renomeando a variavel
df = df_tec_wide_residual

head(df)
```

```
##   ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## 1            0.9202676          0.02820236         0.11727958
## 2            0.9501846          0.11635362         0.20937678
## 3            0.9664565          0.17638584         0.29672610
## 4            0.8624084          0.00000000         0.11395088
## 5            1.0000000          0.95818163         0.77180093
## 6            0.9705260          0.05285858         0.09049135
##   FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## 1         0.01746777           0.9235585
## 2         0.14047268           0.9528485
## 3         0.17638584           0.9648315
## 4         0.00000000           0.7663273
## 5         0.95818163           1.0000000
## 6         0.05285858           0.9688113
```

```r
summary(df)
```

```
##  ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
##  Min.   :0.6105       Min.   :-0.004855   Min.   :0.006784  
##  1st Qu.:0.9610       1st Qu.: 0.086965   1st Qu.:0.222084  
##  Median :0.9896       Median : 0.516278   Median :0.421194  
##  Mean   :0.9690       Mean   : 0.474868   Mean   :0.432180  
##  3rd Qu.:0.9965       3rd Qu.: 0.812064   3rd Qu.:0.645409  
##  Max.   :1.0000       Max.   : 1.000000   Max.   :1.000000  
##  NA's   :6            NA's   :2           NA's   :2
##  FALSE, TRUE, FALSE  SMOTE, FALSE, FALSE
##  Min.   :-0.002485   Min.   :0.6164
##  1st Qu.: 0.096142   1st Qu.:0.9635
##  Median : 0.493988   Median :0.9898
##  Mean   : 0.472091   Mean   :0.9706
##  3rd Qu.: 0.825359   3rd Qu.:0.9974
##  Max.   : 1.000000   Max.   :1.0000
##  NA's   :2           NA's   :4
```

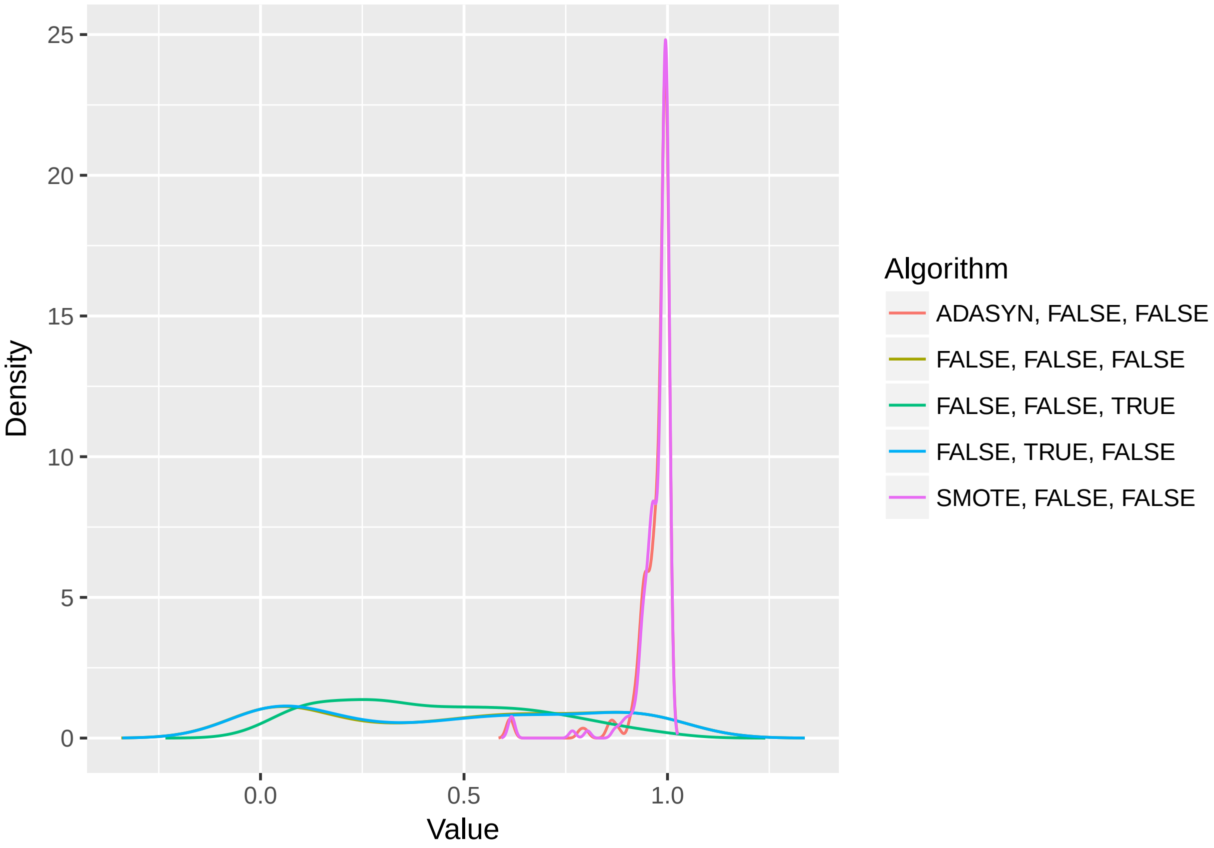
Verificando a média de cada coluna selecionada

```
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 = 0.968984422721258"
## [1] "Media da coluna FALSE, FALSE, FALSE = 0.474868280000896"
## [1] "Media da coluna FALSE, FALSE, TRUE = 0.432180390603151"
## [1] "Media da coluna FALSE, TRUE, FALSE = 0.472090883367927"
## [1] "Media da coluna SMOTE, FALSE, FALSE = 0.97062535664133"
```

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 = 486.01, 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              FALSE
## [3,]                 TRUE               FALSE              FALSE
## [4,]                 TRUE               FALSE              FALSE
## [5,]                FALSE                TRUE               TRUE
##      FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## [1,]               TRUE               FALSE
## [2,]                FALSE                TRUE
## [3,]                FALSE                TRUE
## [4,]                FALSE                TRUE
## [5,]                 TRUE               FALSE
```

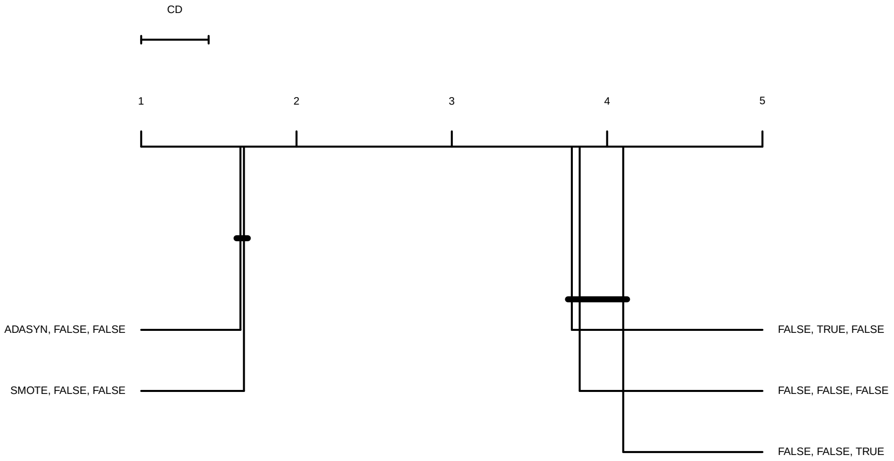
Plotando os ranks

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

```
## ADASYN, FALSE, FALSE  FALSE, FALSE, FALSE   FALSE, FALSE, TRUE
##             1.638889             3.823232             4.103535
##   FALSE, TRUE, FALSE  SMOTE, FALSE, FALSE
##             3.772727             1.661616
```

Plotando grafico de Critical Diference

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


CD
ADASYN, FALSE, FALSE
SMOTE, FALSE, FALSE
FALSE, TRUE, FALSE
FALSE, FALSE, FALSE
FALSE, FALSE, TRUE