R Notebook

Parametros:

Measure = F1 measure
Columns = sampling, weight_space, underbagging
Performance = holdout_measure
Filter keys = imba.rate
Filter values = 0.01

```
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       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        :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                      :1800   SMOTE : 360   TRUE :360
##  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.:0.1475   1st Qu.:0.0000   1st Qu.:0.02254
##  Median :0.8030   Median :0.3333   Median :0.20700
##  Mean   :0.6194   Mean   :0.4107   Mean   :0.32309
##  3rd Qu.:0.9986   3rd Qu.:0.8000   3rd Qu.:0.58363
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
##  NA's   :54       NA's   :54       NA's   :54
##  iteration_count                      dataset       imba.rate
##  Min.   :1       abalone                   :  45   Min.   :0.01
##  1st Qu.:1       adult                     :  45   1st Qu.:0.01
##  Median :2       bank                      :  45   Median :0.01
##  Mean   :2       car                       :  45   Mean   :0.01
##  3rd Qu.:3       cardiotocography-10clases:  45   3rd Qu.:0.01
##  Max.   :3       cardiotocography-3clases :  45   Max.   :0.01
##  NA's   :54      (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$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] 120   5
```

```
# Renomeando a variavel
df = df_tec_wide_residual

head(df)
```

```
##   ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## 1            0.0000000           0.00000000         0.02321195
## 2            0.0000000           0.01886792         0.06800533
## 3            0.0000000           0.00000000         0.04408668
## 4            0.4333333           0.83333333         1.00000000
## 5            0.0000000           0.57777778         0.00000000
## 6            0.1666667           0.70000000         0.43333333
##   FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## 1         0.00000000          0.00000000
## 2         0.01626016          0.02145474
## 3         0.00000000          0.00000000
## 4         0.83333333          0.60000000
## 5         0.57777778          0.13333333
## 6         0.70000000          0.16666667
```

```
summary(df)
```

```
##  ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
##  Min.   :0.0000       Min.   :0.0000      Min.   :0.00000
##  1st Qu.:0.0000       1st Qu.:0.0000      1st Qu.:0.06472
##  Median :0.3333       Median :0.4418      Median :0.16285
##  Mean   :0.4199       Mean   :0.4494      Mean   :0.28679
##  3rd Qu.:0.7876       3rd Qu.:0.8478      3rd Qu.:0.43803
##  Max.   :1.0000       Max.   :1.0000      Max.   :1.00000
##  NA's   :11           NA's   :1
##  FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
##  Min.   :0.0000     Min.   :0.00000
##  1st Qu.:0.0000     1st Qu.:0.01609
##  Median :0.4389     Median :0.39286
##  Mean   :0.4560     Mean   :0.44455
##  3rd Qu.:0.8412     3rd Qu.:0.84900
##  Max.   :1.0000     Max.   :1.00000
##  NA's   :2          NA's   :4
```

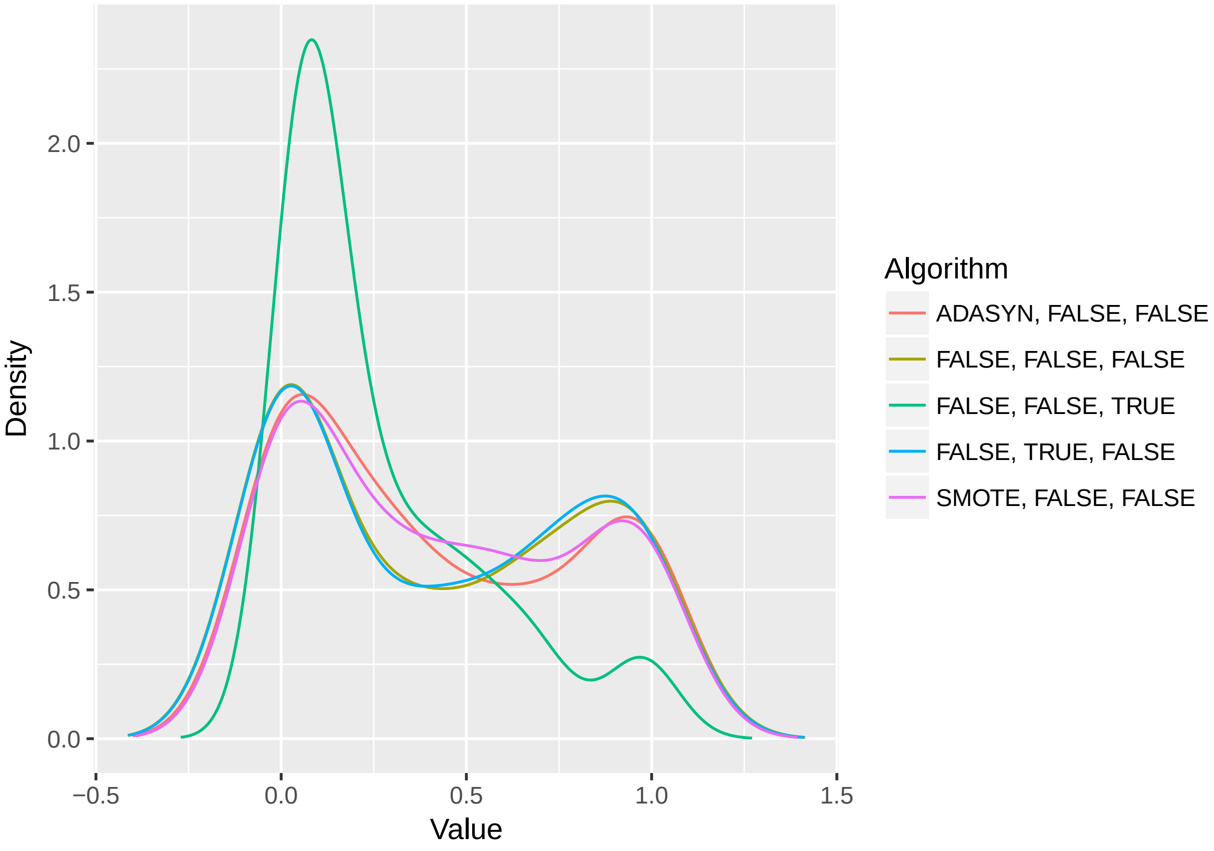
Verificando a média de cada coluna selecionada

```
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 = 0.419914822443638"
## [1] "Media da coluna FALSE, FALSE, FALSE = 0.449424167133309"
## [1] "Media da coluna FALSE, FALSE, TRUE = 0.286791659850835"
## [1] "Media da coluna FALSE, TRUE, FALSE = 0.456048038713067"
## [1] "Media da coluna SMOTE, FALSE, FALSE = 0.444548128986889"
```

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 = 6.0867, df = 4, p-value = 0.1928
```

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

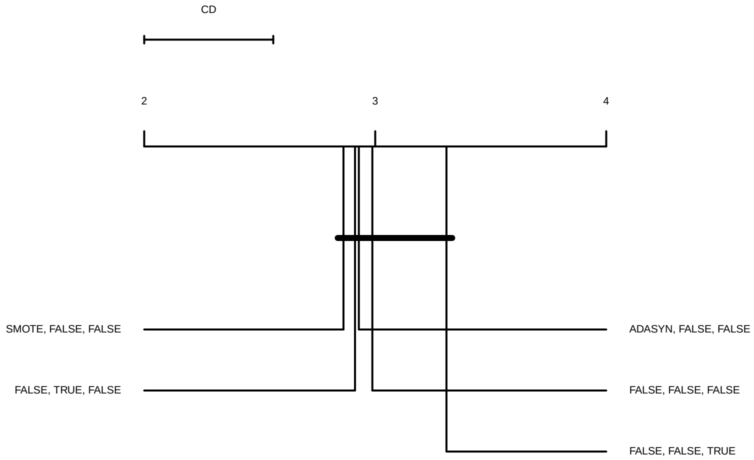
Plotando os ranks

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

```
## ADASYN, FALSE, FALSE  FALSE, FALSE, FALSE   FALSE, FALSE, TRUE
##             2.929167             2.987500             3.308333
##   FALSE, TRUE, FALSE  SMOTE, FALSE, FALSE
##             2.912500             2.862500
```

Plotando grafico de Critical Diference

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


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