R Notebook

Parametros:

Measure = Accuracy
Columns = sampling, weight_space, underbagging
Performance = holdout_measure_residual
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       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        :3420   Mode :logical
##  classif.randomForest:3420   FALSE:8208
##  classif.rusboost    :   0   TRUE :2052
##  classif.xgboost     :3420   NA's :0
##
##
##
##                               measure        sampling    underbagging
##  Accuracy                        :10260   ADASYN:2052   Mode :logical
##  Area under the curve            :    0   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.   :0.09041   Min.   :0.01517   Min.   :0.0346
##  1st Qu.:0.96185   1st Qu.:0.95349   1st Qu.:0.3809
##  Median :0.98796   Median :0.98113   Median :0.7239
##  Mean   :0.95509   Mean   :0.94933   Mean   :0.6600
##  3rd Qu.:0.99669   3rd Qu.:0.99347   3rd Qu.:0.9428
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.0000
##  NA's   :204       NA's   :204       NA's   :204
##  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       cardiotocography-10clases: 180   3rd Qu.:0.0500
##  Max.   :3       cardiotocography-3clases : 180   Max.   :0.0500
##  NA's   :204     (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$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] 684   5
```

```
# Renomeando a variavel
df = df_tec_wide_residual

head(df)
```

```
##   ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## 1            0.3572658           0.3329890          0.6153581
## 2            0.3572658           0.3329890          0.6153581
## 3            0.3810826           0.3311463          0.6504465
## 4            0.3933596           0.3513412          0.6178953
## 5            0.4186973           0.4313027          0.5874736
## 6            0.4186973           0.4313027          0.5874736
##   FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## 1          0.3297176           0.3553719
## 2          0.3297176           0.3553719
## 3          0.3214872           0.3728814
## 4          0.3425249           0.4019884
## 5          0.3964926           0.4211234
## 6          0.3964926           0.4211234
```

```
summary(df)
```

```
##  ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
##  Min.   :0.03682      Min.   :0.0367      Min.   :0.04134
##  1st Qu.:0.38896      1st Qu.:0.3327      1st Qu.:0.58083
##  Median :0.72316      Median :0.6794      Median :0.78629
##  Mean   :0.65921      Mean   :0.6325      Mean   :0.72168
##  3rd Qu.:0.94404      3rd Qu.:0.9499      3rd Qu.:0.91828
##  Max.   :0.99992      Max.   :1.0000      Max.   :0.99985
##  NA's   :33           NA's   :6            NA's   :5
##  FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
##  Min.   :0.0367     Min.   :0.03682
##  1st Qu.:0.3325     1st Qu.:0.38002
##  Median :0.6721     Median :0.71339
##  Mean   :0.6308     Mean   :0.65560
##  3rd Qu.:0.9487     3rd Qu.:0.94968
##  Max.   :1.0000     Max.   :1.00000
##  NA's   :6          NA's   :18
```

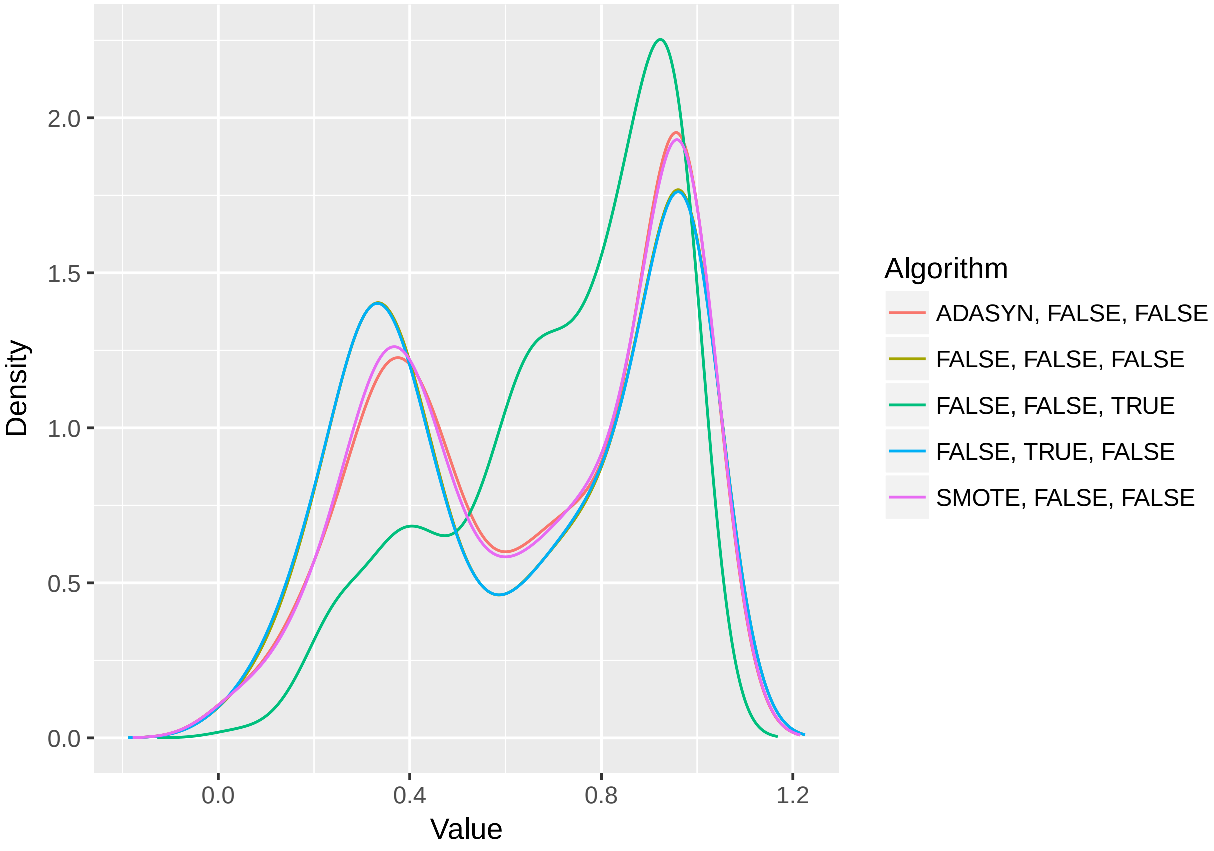
Verificando a média de cada coluna selecionada

```r
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 = 0.659211626562193"
## [1] "Media da coluna FALSE, FALSE, FALSE = 0.632486883725381"
## [1] "Media da coluna FALSE, FALSE, TRUE = 0.721681494090847"
## [1] "Media da coluna FALSE, TRUE, FALSE = 0.630764193944798"
## [1] "Media da coluna SMOTE, FALSE, FALSE = 0.655600329487413"
```

Fazendo teste de normalidade

```r
plotDensities(data = na.omit(df))
```



Testando as diferencas

```
friedmanTest(df)
```

```
##
##  Friedman's rank sum test
##
## data:  df
## Friedman's chi-squared = 155.43, 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
```

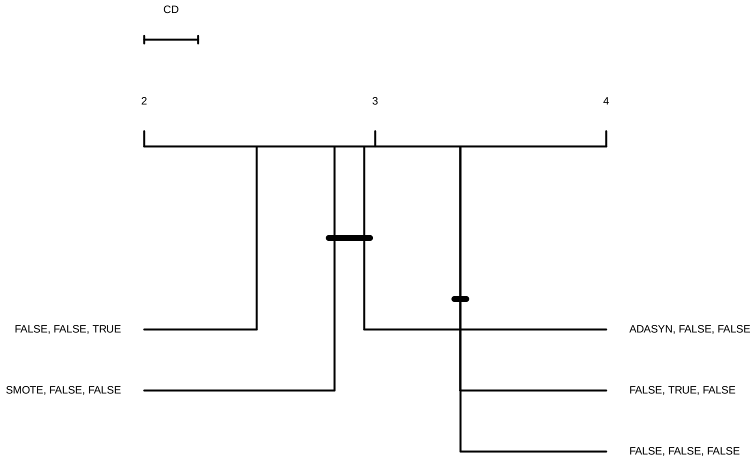
Plotando os ranks

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

```
## ADASYN, FALSE, FALSE  FALSE, FALSE, FALSE   FALSE, FALSE, TRUE
##             2.952485             3.369152             2.486842
##   FALSE, TRUE, FALSE  SMOTE, FALSE, FALSE
##             3.367690             2.823830
```

Plotando grafico de Critical Diference

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


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