R Notebook

Parametros:

Measure = Matthews correlation coefficient
Columns = learner
Performance = tuning_measure
Filter keys = sampling, weight_space, underbagging, imba.rate
Filter values = FALSE, FALSE, FALSE, 0.05

```
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         :246    Mode :logical
##  classif.randomForest:246    FALSE:738
##  classif.rusboost    :  0    NA's :0
##  classif.xgboost     :246
##
##
##
##                                measure       sampling    underbagging
##  Accuracy                         :  0   ADASYN:  0   Mode :logical
##  Area under the curve             :  0   FALSE :738   FALSE:738
##  F1 measure                       :  0   SMOTE :  0   NA's :0
##  G-mean                           :  0
##  Matthews correlation coefficient:738
##
##
##  tuning_measure     holdout_measure     holdout_measure_residual
##  Min.   :-0.02216   Min.   :-0.05263   Min.   :-0.45710
##  1st Qu.: 0.12022   1st Qu.: 0.00000   1st Qu.: 0.01115
##  Median : 0.51609   Median : 0.56796   Median : 0.19268
##  Mean   : 0.48425   Mean   : 0.48350   Mean   : 0.29438
##  3rd Qu.: 0.80345   3rd Qu.: 0.86038   3rd Qu.: 0.52028
##  Max.   : 1.00000   Max.   : 1.00000   Max.   : 1.00000
##
##  iteration_count          dataset         imba.rate
##  Min.   :1       abalone      :  9   Min.   :0.05
##  1st Qu.:1       adult        :  9   1st Qu.:0.05
##  Median :2       annealing    :  9   Median :0.05
##  Mean   :2       arrhythmia   :  9   Mean   :0.05
##  3rd Qu.:3       balance-scale:  9   3rd Qu.:0.05
##  Max.   :3       bank         :  9   Max.   :0.05
##                  (Other)      :684
```

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] 82  3
```

```r
# Renomeando a variavel
df = df_tec_wide_residual

head(df)
```

```
##   classif.ksvm classif.randomForest classif.xgboost
## 1   0.08104013            0.0000000      0.02330986
## 2   0.24483618            0.4854582      0.48961416
## 3   0.29346248            0.7292073      0.68320251
## 4   0.00000000            0.7804829      0.67382183
## 5   1.00000000            1.0000000      1.00000000
## 6   0.10956287            0.1939237      0.21903979
```

```r
summary(df)
```

```
##   classif.ksvm      classif.randomForest classif.xgboost
##  Min.   :-0.00263   Min.   :0.0000       Min.   :0.0000
##  1st Qu.: 0.00000   1st Qu.:0.2241       1st Qu.:0.2768
##  Median : 0.22396   Median :0.6288       Median :0.6695
##  Mean   : 0.30194   Mean   :0.5636       Mean   :0.5872
##  3rd Qu.: 0.49531   3rd Qu.:0.8255       3rd Qu.:0.8344
##  Max.   : 1.00000   Max.   :1.0000       Max.   :1.0000
```

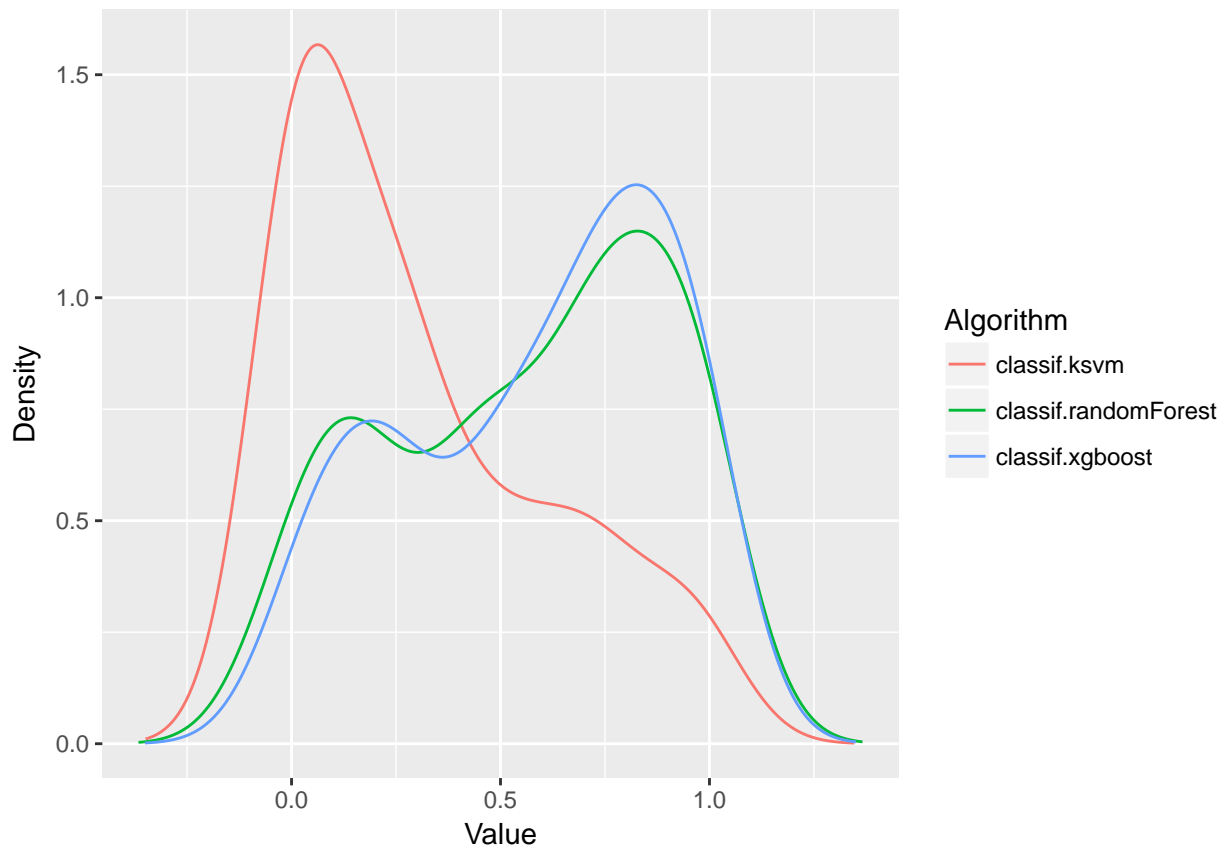## Verificando a média de cada coluna selecionada

```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 classif.ksvm = 0.301939372449071"
## [1] "Media da coluna classif.randomForest = 0.563594360772488"
## [1] "Media da coluna classif.xgboost = 0.587222752029455"
```

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 = 54.591, df = 2, p-value = 1.398e-12
```

Testando as diferencas par a par

```
test <- nemenyiTest (df, alpha=0.05)
abs(test$diff.matrix) > test$statistic
```

```
##      classif.ksvm classif.randomForest classif.xgboost
## [1,]        FALSE                 TRUE            TRUE
## [2,]         TRUE                FALSE           FALSE
## [3,]         TRUE                FALSE           FALSE
```

Plotando os ranks

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

```
##        classif.ksvm classif.randomForest      classif.xgboost
##            2.634146             1.859756             1.506098
```

Plotando grafico de Critical Diference

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