R Notebook

Parametros:

Measure = Area under the curve
Columns = learner
Performance = tuning_measure
Filter keys = sampling, weight_space, underbagging, imba.rate
Filter values = FALSE, FALSE, FALSE, 0.001

```
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        :120   Mode :logical
##  classif.randomForest:120   FALSE:360
##  classif.rusboost    :  0   NA's :0
##  classif.xgboost     :120
##
##
##
##                             measure      sampling   underbagging
##  Accuracy                       :  0   ADASYN:  0   Mode :logical
##  Area under the curve           :360   FALSE :360   FALSE:360
##  F1 measure                     :  0   SMOTE :  0   NA's :0
##  G-mean                         :  0
##  Matthews correlation coefficient:  0
##
##
##  tuning_measure   holdout_measure  holdout_measure_residual
##  Min.   :0.4832   Min.   :0.2847   Min.   :0.3092
##  1st Qu.:0.8892   1st Qu.:0.9113   1st Qu.:0.7475
##  Median :0.9879   Median :0.9954   Median :0.9184
##  Mean   :0.9210   Mean   :0.9189   Mean   :0.8534
##  3rd Qu.:0.9994   3rd Qu.:1.0000   3rd Qu.:0.9866
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##
##  iteration_count                      dataset      imba.rate
##  Min.   :1       abalone                  :  9   Min.   :0.001
##  1st Qu.:1       adult                    :  9   1st Qu.:0.001
##  Median :2       bank                     :  9   Median :0.001
##  Mean   :2       car                      :  9   Mean   :0.001
##  3rd Qu.:3       cardiotocography-10clases:  9   3rd Qu.:0.001
##  Max.   :3       cardiotocography-3clases :  9   Max.   :0.001
##                  (Other)                  :306
```

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] 40  3
```

```
# Renomeando a variavel
df = df_tec_wide_residual

head(df)
```

```
##   classif.ksvm classif.randomForest classif.xgboost
## 1    0.5777335            0.5577409       0.6580300
## 2    0.6022761            0.8632699       0.9166584
## 3    0.7565083            0.8534540       0.8535358
## 4    1.0000000            1.0000000       1.0000000
## 5    0.9757015            0.9877876       0.9852483
## 6    0.9996823            0.9999631       0.9997785
```

```
summary(df)
```

```
##   classif.ksvm    classif.randomForest classif.xgboost
##  Min.   :0.5777   Min.   :0.5577       Min.   :0.6580
##  1st Qu.:0.7526   1st Qu.:0.8993       1st Qu.:0.9171
##  Median :0.9372   Median :0.9905       Median :0.9873
##  Mean   :0.8739   Mean   :0.9441       Mean   :0.9449
##  3rd Qu.:0.9983   3rd Qu.:0.9995       3rd Qu.:0.9987
##  Max.   :1.0000   Max.   :1.0000       Max.   :1.0000
```

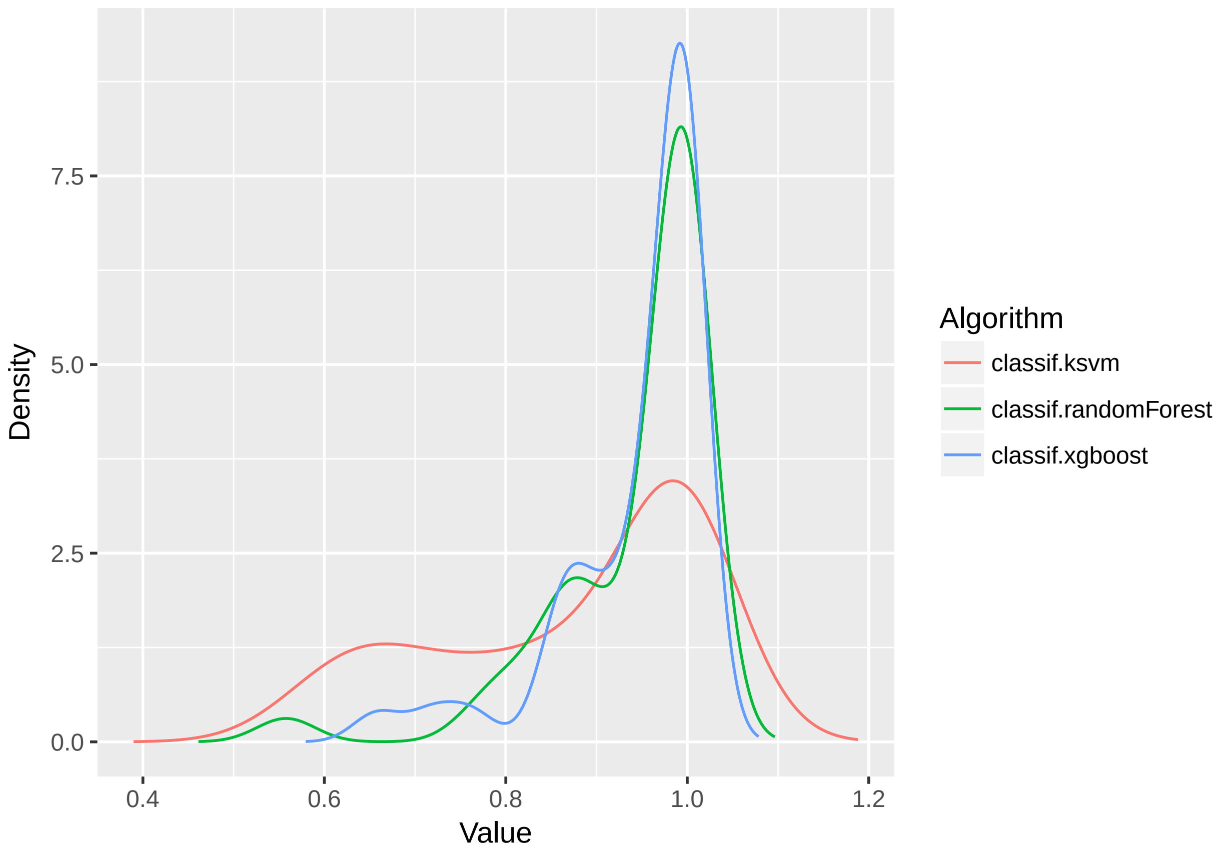
Verificando a média de cada coluna selecionada

```
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 classif.ksvm = 0.873924077627644"
## [1] "Media da coluna classif.randomForest = 0.94406271492398"
## [1] "Media da coluna classif.xgboost = 0.944902952710432"
```

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 = 15.387, df = 2, p-value = 0.0004557
```

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
```

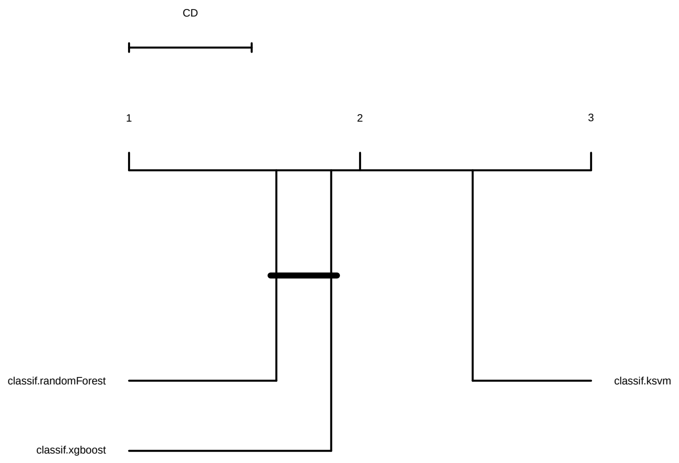
Plotando os ranks

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

```
##         classif.ksvm classif.randomForest      classif.xgboost
##               2.4875               1.6375               1.8750
```

Plotando grafico de Critical Diference

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


CD
classif.randomForest
classif.xgboost
classif.ksvm