

R Notebook

Parametros:

Measure = Area under the curve
Columns = learner
Performance = holdout_measure_residual
Filter keys = sampling, weight_space, underbagging, imba.rate
Filter values = FALSE, FALSE, FALSE, 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.csv")  
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      cardiocography-10clases: 900 3rd Qu.:0.0500
## Max. :3      cardiocography-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.:0.8848 1st Qu.:0.9110 1st Qu.:0.7618
## Median :0.9872 Median :0.9959 Median :0.9187
## Mean :0.9204  Mean :0.9183  Mean :0.8560
## 3rd Qu.:0.9994 3rd Qu.:1.0000 3rd Qu.:0.9870
## Max. :1.0000  Max. :1.0000  Max. :1.0000
## NA's :3      NA's :3      NA's :3
## iteration_count      dataset      imba.rate
## Min. :1      abalone      : 9  Min. :0.01
## 1st Qu.:1      adult      : 9  1st Qu.:0.01
## Median :2      bank      : 9  Median :0.01
## Mean :2      car      : 9  Mean :0.01
## 3rd Qu.:3      cardiocography-10clases: 9 3rd Qu.:0.01
## Max. :3      cardiocography-3clases : 9 Max. :0.01
## NA's :3      (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$performance)))

# 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    0.5133720         0.6224639         0.6375083
## 2    0.4549243         0.8601656         0.9023521
## 3    0.7469161         0.8316542         0.8204698
## 4    0.8590001         0.7703462         0.7190811
## 5    0.9703766         0.9830413         0.9625229
## 6    0.8672659         0.7798361         0.8821091
```

```
summary(df)
```

```
##   classif.ksvm   classif.randomForest classif.xgboost
## Min.   :0.4549   Min.   :0.4953         Min.   :0.5704
## 1st Qu.:0.7339   1st Qu.:0.8222         1st Qu.:0.8118
## Median :0.8486   Median :0.9411         Median :0.9225
## Mean   :0.8197   Mean   :0.8726         Mean   :0.8761
## 3rd Qu.:0.9742   3rd Qu.:0.9808         3rd Qu.:0.9855
## Max.   :1.0000   Max.   :1.0000         Max.   :1.0000
## NA's   :1
```

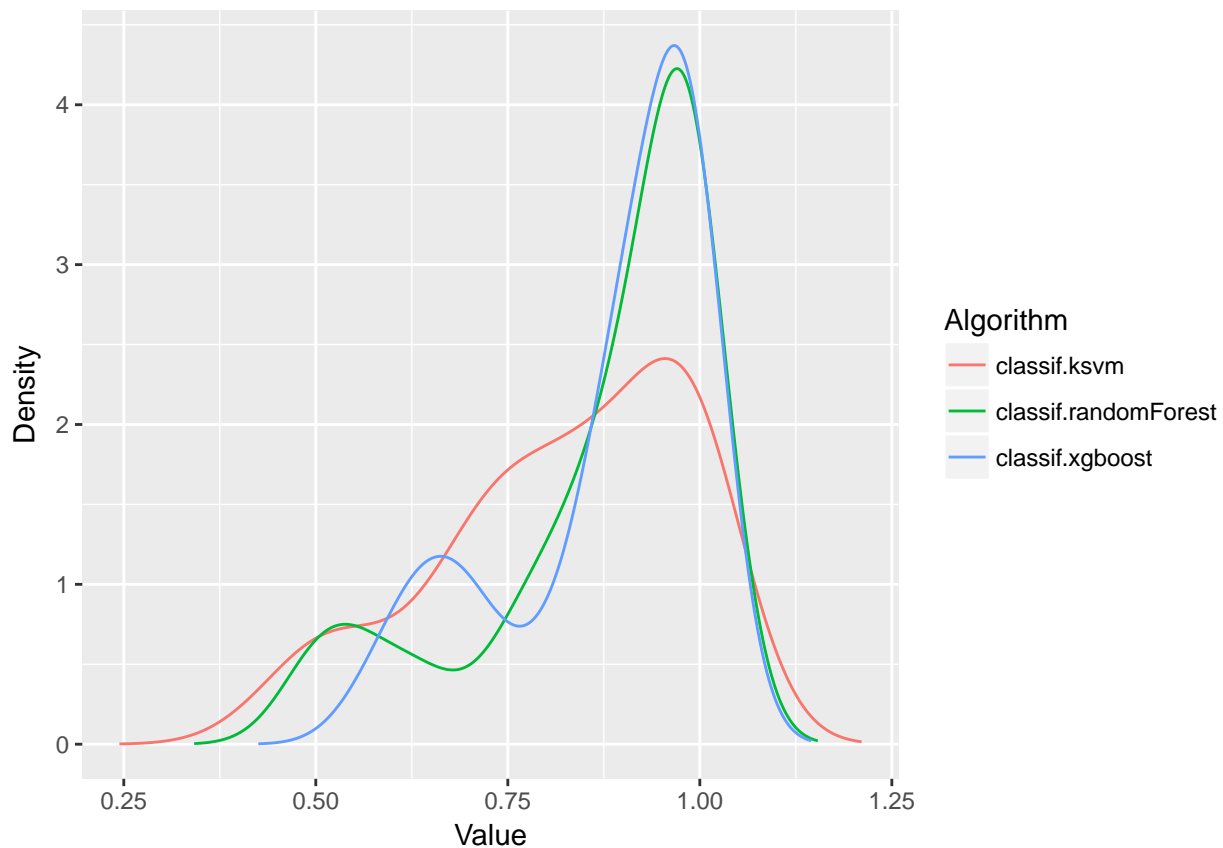
Verificando a média de cada coluna selecionada

```
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 = 0.819684267348074"
## [1] "Media da coluna classif.randomForest = 0.872551115921721"
## [1] "Media da coluna classif.xgboost = 0.876095504804726"
```

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 = 6.8625, df = 2, p-value = 0.03235
```

Testando as diferenças par a par

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

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

Plotando os ranks

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

```
##      classif.ksvm classif.randomForest      classif.xgboost
##              2.3375              1.8125              1.8500
```

Plotando grafico de Critical Difference

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

