

R Notebook

Parametros:

Measure = Matthews correlation coefficient
Columns = learner
Performance = holdout_measure
Filter keys = sampling, weight_space, underbagging, imba.rate
Filter values = FALSE, FALSE, FALSE, 0.03

```
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      cardiocography-10classes: 900 3rd Qu.:0.0500
## Max. :3      cardiocography-3classes : 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      :198 Mode :logical
## classif.randomForest:198 FALSE:594
## classif.rusboost   : 0  NA's :0
## classif.xgboost    :198
##
##
##
##           measure      sampling  underbagging
## Accuracy           : 0  ADASYN: 0  Mode :logical
## Area under the curve : 0  FALSE :594 FALSE:594
## F1 measure           : 0  SMOTE : 0  NA's :0
## G-mean               : 0
## Matthews correlation coefficient:594
##
##
## tuning_measure      holdout_measure      holdout_measure_residual
## Min. : -0.02717      Min. : -0.03872      Min. : -0.3218
## 1st Qu.: 0.04560      1st Qu.: 0.00000      1st Qu.: 0.0000
## Median : 0.52172      Median : 0.57178      Median : 0.1669
## Mean : 0.47487      Mean : 0.48068      Mean : 0.2837
## 3rd Qu.: 0.82008      3rd Qu.: 0.84929      3rd Qu.: 0.4780
## Max. : 1.00000      Max. : 1.00000      Max. : 1.0000
## NA's :6      NA's :6      NA's :6
## iteration_count      dataset      imba.rate
## Min. :1      abalone : 9      Min. :0.03
## 1st Qu.:1      adult : 9      1st Qu.:0.03
## Median :2      annealing : 9      Median :0.03
## Mean :2      arrhythmia : 9      Mean :0.03
## 3rd Qu.:3      balance-scale: 9      3rd Qu.:0.03
## Max. :3      bank : 9      Max. :0.03
## NA's :6      (Other) :540
```

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] 66 3
```

```
# Renomeando a variavel
df = df_tec_wide_residual

head(df)
```

```
##   classif.ksvm classif.randomForest classif.xgboost
## 1 -0.007549758      0.000000000      -0.003371752
## 2  0.146444566      0.469422453      0.459268050
## 3  0.184077857      0.648010158      0.179229084
## 4  0.000000000      0.666666667      0.264961794
## 5  1.000000000      1.000000000      1.000000000
## 6  0.023322357     -0.006915521      0.061768996
```

```
summary(df)
```

```
##   classif.ksvm      classif.randomForest classif.xgboost
## Min.      :-0.02749   Min.      :-0.01694   Min.      :-0.02329
## 1st Qu.: 0.00000     1st Qu.: 0.18891   1st Qu.: 0.19820
## Median : 0.19991     Median : 0.65240   Median : 0.66636
## Mean    : 0.33782     Mean    : 0.53893   Mean    : 0.56706
## 3rd Qu.: 0.62881     3rd Qu.: 0.86172   3rd Qu.: 0.89310
## Max.    : 1.00000     Max.    : 1.00000   Max.    : 1.00000
## NA's    :2
```

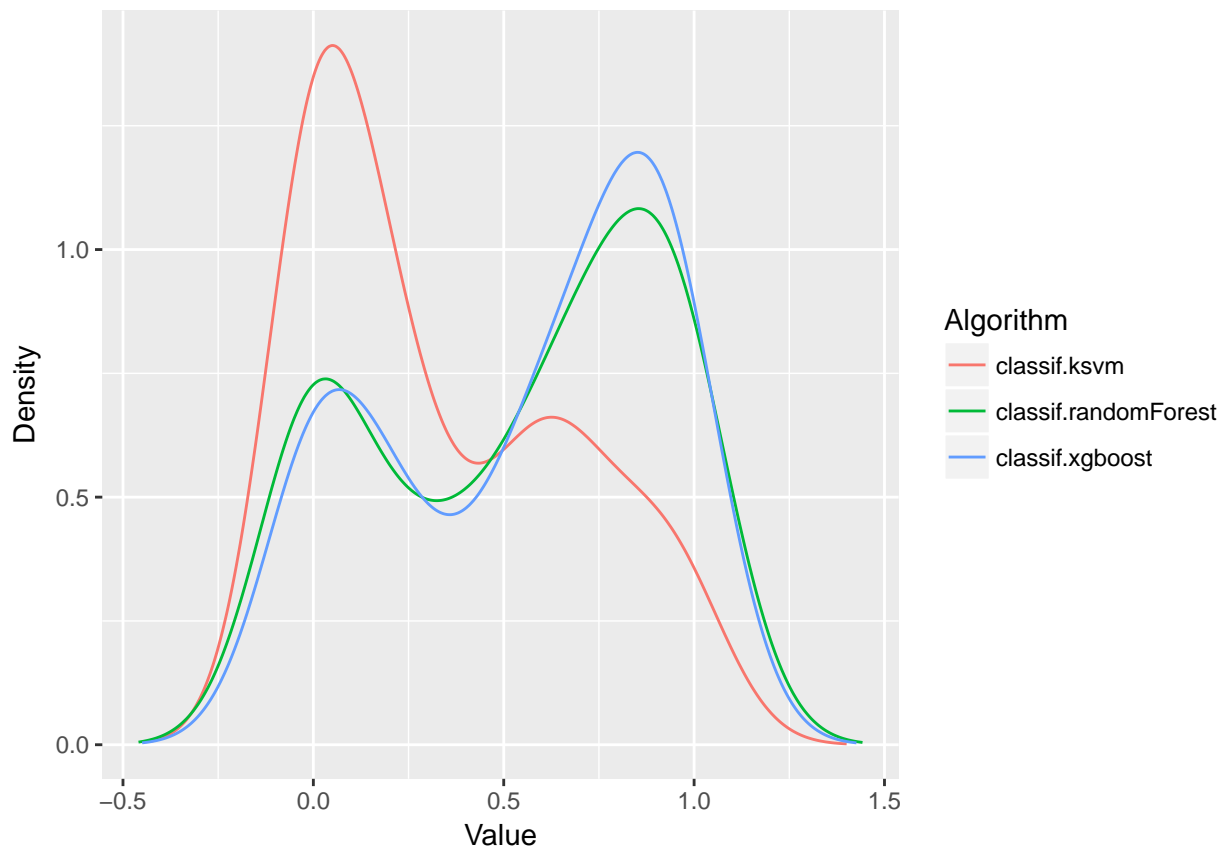
Verificando a média de cada coluna selecionada

```
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 = 0.33782362074705"
## [1] "Media da coluna classif.randomForest = 0.538933940072554"
## [1] "Media da coluna classif.xgboost = 0.567056380341611"
```

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 = 30.068, df = 2, p-value = 2.956e-07
```

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
##      2.545455      1.795455      1.659091
```

Plotando grafico de Critical Difference

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

