

R Notebook

Parametros:

Measure = Matthews correlation coefficient
Columns = sampling, weight_space, underbagging
Performance = holdout_measure_residual
Filter keys = imba.rate
Filter values = 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.  
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      :600 Mode :logical
## classif.randomForest:600 FALSE:1440
## classif.rusboost   : 0 TRUE :360
## classif.xgboost    :600 NA's :0
##
##
##
##          measure      sampling      underbagging
## Accuracy          : 0 ADASYN: 360 Mode :logical
## Area under the curve : 0 FALSE :1080 FALSE:1440
## F1 measure          : 0 SMOTE : 360 TRUE :360
## G-mean              : 0 NA's :0
## Matthews correlation coefficient:1800
##
##
## tuning_measure      holdout_measure      holdout_measure_residual
## Min. : -0.00646 Min. : -0.1370 Min. : -0.06817
## 1st Qu.: 0.23261 1st Qu.: 0.0000 1st Qu.: 0.02011
## Median : 0.82014 Median : 0.3764 Median : 0.19200
## Mean : 0.64070 Mean : 0.4285 Mean : 0.29498
## 3rd Qu.: 0.99730 3rd Qu.: 0.8152 3rd Qu.: 0.49996
## Max. : 1.00000 Max. : 1.0000 Max. : 1.00000
## NA's :69 NA's :69 NA's :69
## iteration_count      dataset      imba.rate
## Min. :1 abalone : 45 Min. :0.01
## 1st Qu.:1 adult : 45 1st Qu.:0.01
## Median :2 bank : 45 Median :0.01
## Mean :2 car : 45 Mean :0.01
## 3rd Qu.:3 cardiocography-10clases: 45 3rd Qu.:0.01
## Max. :3 cardiocography-3clases : 45 Max. :0.01
## NA's :69 (Other) :1530
```

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] 120 5
```

```
# Renomeando a variavel
df = df_tec_wide_residual
```

```
head(df)
```

```
## ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## 1 0.070168439 0.03701282 0.17619019
## 2 0.101553966 0.09710987 0.37634653
## 3 -0.002295506 0.00000000 0.37090560
## 4 0.100755082 0.06878240 0.70821704
## 5 0.000000000 0.23602639 0.05484309
## 6 0.082764035 0.18013252 0.14171779
## FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## 1 0.002457572 0.06360859
## 2 0.108197202 0.10857900
## 3 0.000000000 0.01349113
## 4 0.068782404 0.10462939
## 5 0.236026391 0.03874297
## 6 0.180132517 0.07725384
```

```
summary(df)
```

```
## ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## Min. : -0.03745 Min. : -0.03888 Min. : -0.003467
## 1st Qu.: 0.04247 1st Qu.: 0.01367 1st Qu.: 0.135466
## Median : 0.17548 Median : 0.15391 Median : 0.370906
## Mean : 0.28256 Mean : 0.25189 Mean : 0.407780
## 3rd Qu.: 0.47091 3rd Qu.: 0.44963 3rd Qu.: 0.642368
## Max. : 0.95929 Max. : 0.90474 Max. : 0.916350
```

```
## NA's :10      NA's :3      NA's :1
## FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## Min. : -0.04521 Min. : -0.06687
## 1st Qu.: 0.01333 1st Qu.: 0.04024
## Median : 0.14537 Median : 0.14985
## Mean : 0.25229 Mean : 0.27704
## 3rd Qu.: 0.46253 3rd Qu.: 0.43881
## Max. : 0.89582 Max. : 0.93840
## NA's :4      NA's :5
```

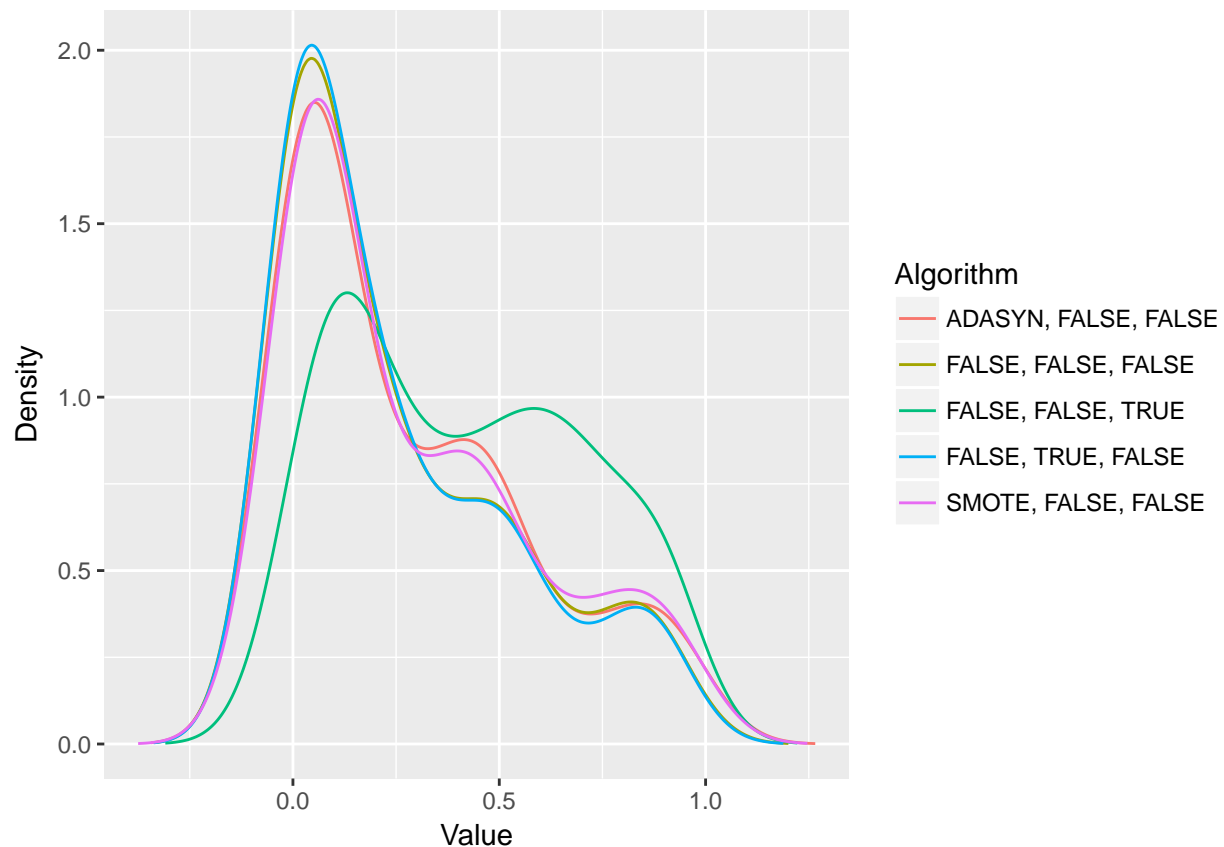
Verificando a média de cada coluna selecionada

```
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 ADASYN, FALSE, FALSE = 0.282558052756018"
## [1] "Media da coluna FALSE, FALSE, FALSE = 0.251891555237523"
## [1] "Media da coluna FALSE, FALSE, TRUE = 0.407780260024908"
## [1] "Media da coluna FALSE, TRUE, FALSE = 0.252292258604788"
## [1] "Media da coluna SMOTE, FALSE, FALSE = 0.27703812343907"
```

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 = 64.953, df = 4, p-value = 2.632e-13
```

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          FALSE          TRUE
## [2,]          FALSE          FALSE          TRUE
## [3,]           TRUE          TRUE         FALSE
## [4,]          FALSE          FALSE          TRUE
## [5,]          FALSE          FALSE          TRUE
##      FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## [1,]          FALSE          FALSE
```

```
## [2,]          FALSE          FALSE
## [3,]           TRUE           TRUE
## [4,]          FALSE          FALSE
## [5,]          FALSE          FALSE
```

Plotando os ranks

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

```
## ADASYN, FALSE, FALSE  FALSE, FALSE, FALSE  FALSE, FALSE, TRUE
##           3.033333          3.408333          2.041667
##  FALSE, TRUE, FALSE  SMOTE, FALSE, FALSE
##           3.516667          3.000000
```

Plotando grafico de Critical Difference

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

