

R Notebook

Parametros:

```
Measure = Accuracy
Columns = sampling, weight_space, underbagging
Performance = holdout_measure
Filter keys = imba.rate
Filter values = 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.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      :600 Mode :logical
## classif.randomForest:600 FALSE:1440
## classif.rusboost   : 0 TRUE :360
## classif.xgboost    :600 NA's :0
##
##
##
##           measure      sampling      underbagging
## Accuracy           :1800 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: 0
##
##
## tuning_measure  holdout_measure  holdout_measure_residual
## Min. :0.1269 Min. :0.01517 Min. :0.03881
## 1st Qu.:0.9898 1st Qu.:0.98750 1st Qu.:0.38633
## Median :0.9938 Median :0.99163 Median :0.76435
## Mean :0.9692 Mean :0.96680 Mean :0.67114
## 3rd Qu.:0.9990 3rd Qu.:0.99687 3rd Qu.:0.95470
## Max. :1.0000 Max. :1.00000 Max. :1.00000
## NA's :48 NA's :48 NA's :48
## iteration_count      dataset      imba.rate
## Min. :1 abalone : 45 Min. :0.001
## 1st Qu.:1 adult : 45 1st Qu.:0.001
## Median :2 bank : 45 Median :0.001
## Mean :2 car : 45 Mean :0.001
## 3rd Qu.:3 cardiocography-10clases: 45 3rd Qu.:0.001
## Max. :3 cardiocography-3clases : 45 Max. :0.001
## NA's :48 (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 0.9677996 0.9898629 0.5086464
## 2 0.9819241 0.9892883 0.8815023
## 3 0.9875000 0.9900000 0.4887500
## 4 0.9937304 0.9979101 1.0000000
## 5 0.9900000 0.9941667 0.9900000
## 6 0.9929639 0.9964820 0.9947230
## FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## 1 0.9880739 0.9666070
## 2 0.9898239 0.9822588
## 3 0.9900000 0.9879167
## 4 0.9979101 0.9958203
## 5 0.9941667 0.9908333
## 6 0.9964820 0.9929639
```

```
summary(df)
```

```
## ADASYN, FALSE, FALSE FALSE, FALSE, FALSE FALSE, FALSE, TRUE
## Min. :0.9678 Min. :0.9850 Min. :0.1817
## 1st Qu.:0.9885 1st Qu.:0.9909 1st Qu.:0.7893
## Median :0.9916 Median :0.9928 Median :0.9297
## Mean :0.9915 Mean :0.9941 Mean :0.8661
## 3rd Qu.:0.9966 3rd Qu.:0.9979 3rd Qu.:0.9835
## Max. :1.0000 Max. :1.0000 Max. :1.0000
```

```
## NA's :10      NA's :2
## FALSE, TRUE, FALSE SMOTE, FALSE, FALSE
## Min. :0.9867   Min. :0.9617
## 1st Qu.:0.9909   1st Qu.:0.9890
## Median :0.9930   Median :0.9921
## Mean :0.9940     Mean :0.9916
## 3rd Qu.:0.9976   3rd Qu.:0.9973
## Max. :1.0000     Max. :1.0000
## NA's :2         NA's :2
```

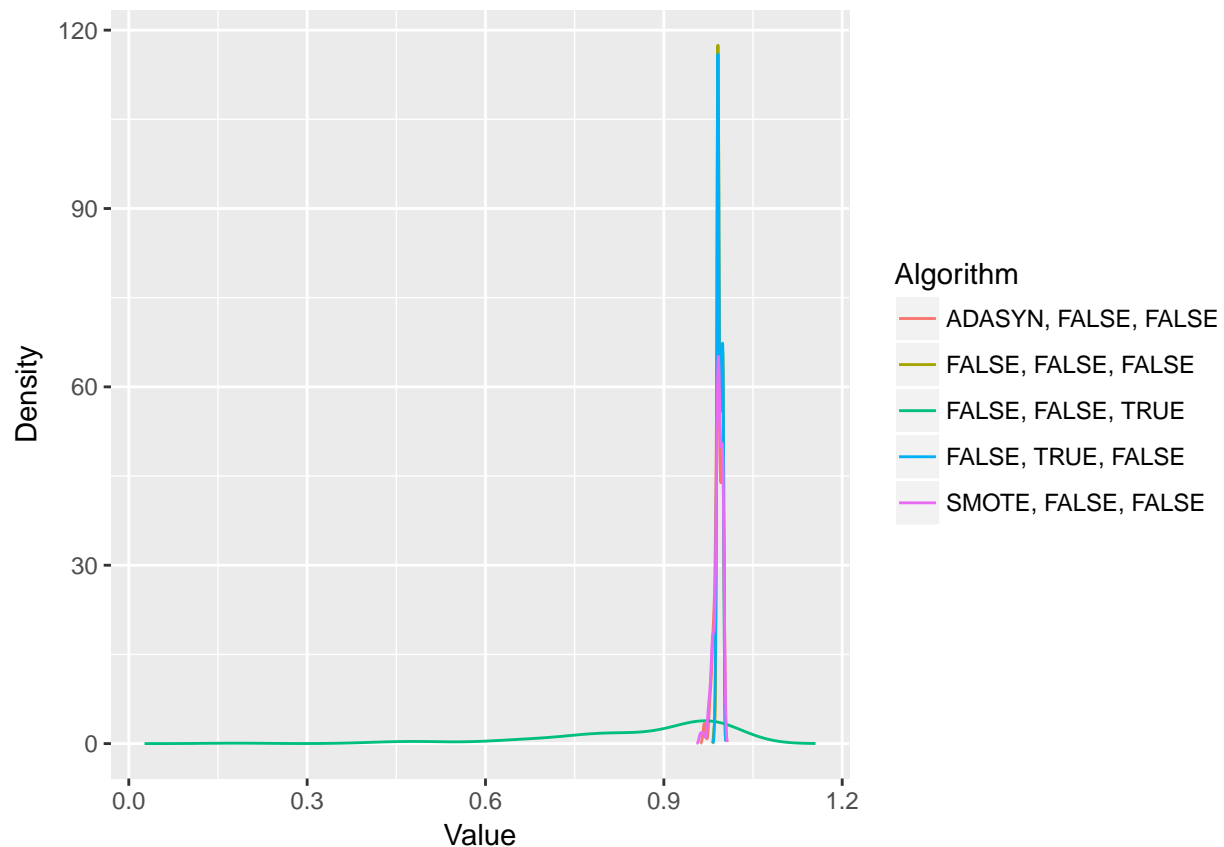
Verificando a média de cada coluna selecionada

```
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 ADASYN, FALSE, FALSE = 0.991515107971523"
## [1] "Media da coluna FALSE, FALSE, FALSE = 0.994079795966482"
## [1] "Media da coluna FALSE, FALSE, TRUE = 0.866115590019536"
## [1] "Media da coluna FALSE, TRUE, FALSE = 0.994041858311255"
## [1] "Media da coluna SMOTE, FALSE, FALSE = 0.991624024564933"
```

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 = 145.62, df = 4, p-value < 2.2e-16
```

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

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

Plotando os ranks

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

```
## ADASYN, FALSE, FALSE  FALSE, FALSE, FALSE  FALSE, FALSE, TRUE
##          3.120833          2.312500          4.400000
##  FALSE, TRUE, FALSE  SMOTE, FALSE, FALSE
##          2.237500          2.929167
```

Plotando grafico de Critical Difference

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

