clasificacion

Ana Diedrichs

May 22, 2019

```
library(ggplot2)
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
suppressMessages(library(tidyverse))
```

Datos

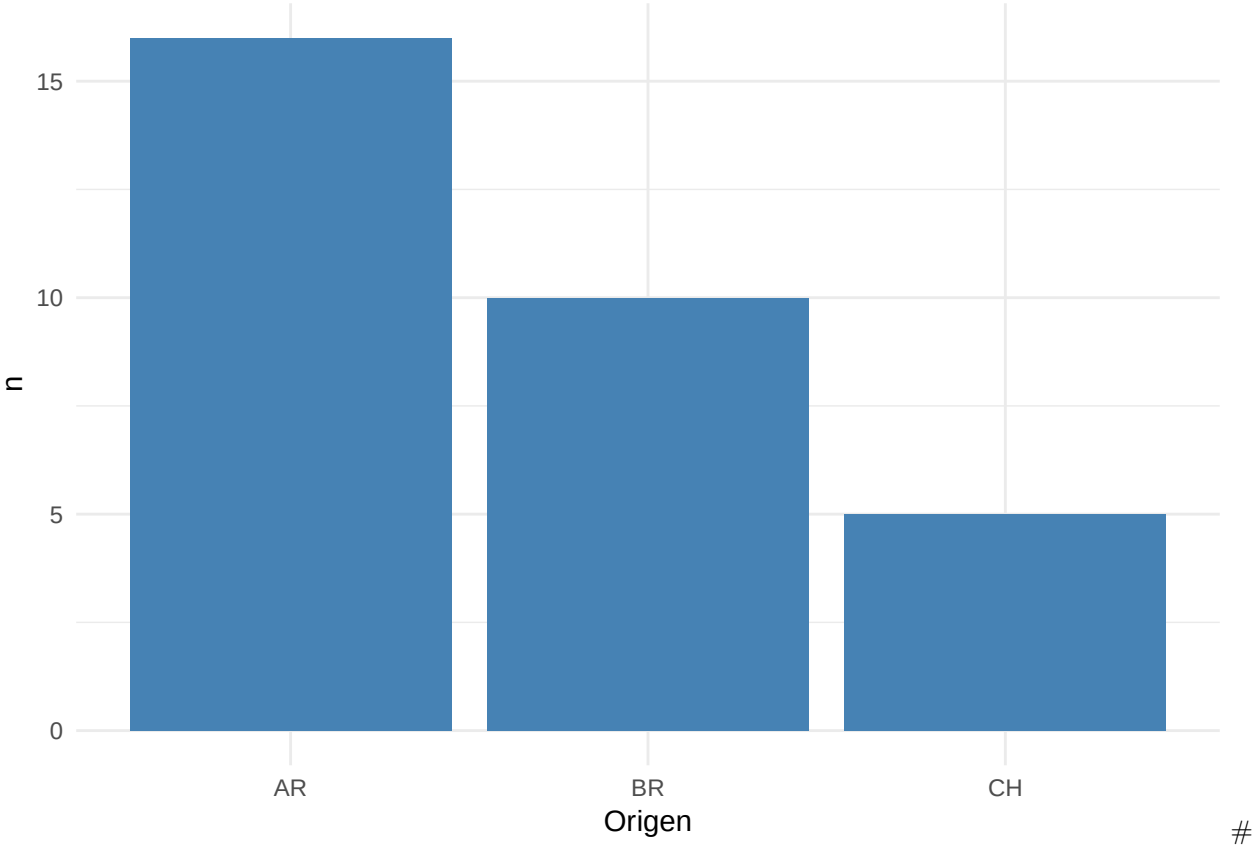
Este dataset tiene 12 variables en total, contando la variable de clase llamada Origen. El dataset consta de 31 datapoints o muestras clasificadas en 3 clases etiquetadas como BR, CH, AR

En el siguiente cuadro y gráfico observamos como se distribuyen las muestras según su origen. Notamos que el dataset está desbalanceado, pues no hay la misma cantidad de datapoints para cada clase.

Table 1: Tabla que muestra distribución de datapoints por clase

Origen	n
AR	16
BR	10
CH	5

```
myplot <- ggplot(data=d, aes(x=Origen, y=n)) +
  geom_bar(stat="identity", fill="steelblue")+
  theme_minimal()

print(myplot)
```



#

Experimentos

Sobre el total del dataset emplearemos k-fold cross validation con k=4 para los modelos:

- LDA linear discriminant analysis
- nnet neural networks

Al final se muestran los resultados de los modelos sobre cross validation, agrupados.

LDA

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
x = data[,-1]
y = data$Origen
#index <- sample(1:nrow(data), round(nrow(data) * 0.7))
#train <- data[index,]
#test <- data[-index,]
SEED <- 1234 # seed semilla para números aleatorios
set.seed(SEED)
```

```
mySeeds <- sapply(simplify = FALSE, 1:11, function(u) sample(10^4, 3))

METRIC <- "Accuracy" #
train_control <- trainControl(method="cv", number=4,seeds = mySeeds,classProbs=TRUE)

set.seed(SEED)
mySeeds <- sapply(simplify = FALSE, 1:11, function(u) sample(10^4, 3))
train_control <- trainControl(method="cv", number=4,seeds = mySeeds,classProbs=TRUE)
model.lda <- train(as.factor(Origen)~., data=data,
                   trControl=train_control, method="lda",metric=METRIC)

p <- predict(model.lda$finalModel,x,type="class")
```

```
print(table(p$class,y))
```

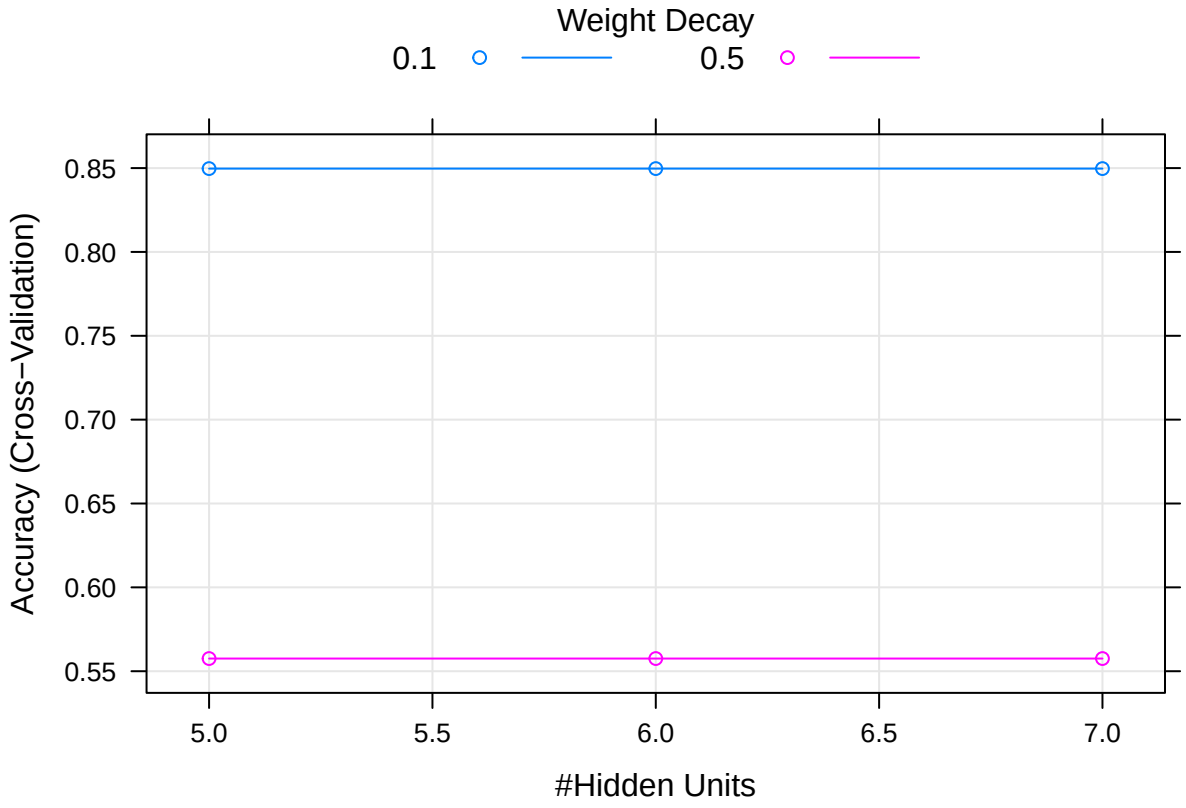
```
##     y
##      AR BR CH
##   AR 15  0  0
##   BR  1  9  1
##   CH  0  1  4
```

Neural network

```
my.grid <- expand.grid(.decay = c(0.5, 0.1), .size = c(5, 6, 7))

set.seed(SEED)
mySeeds <- sapply(simplify = FALSE, 1:11, function(u) sample(10^4, 6))
train_control <- trainControl(method="cv", number=4,seeds = mySeeds,classProbs=TRUE)
model.nnet <- train(as.factor(Origen)~., data=data,
                   trControl=train_control, method="nnet", tuneGrid=my.grid,
                   maxit = 1000, trace = F,metric=METRIC)

p <- predict(model.nnet$finalModel,x,type="class")

print(table(p,y))
```

```
##     y
## p    AR BR CH
##   AR 16  0  0
##   BR  0 10  2
##   CH  0  0  3
```

```
plot(model.nnet)
```



glmnet

```
set.seed(SEED)
mySeeds <- sapply(simplify = FALSE, 1:11, function(u) sample(10^4, 3))

train_control <- trainControl(method="cv", number=4,seeds = mySeeds,classProbs=TRUE)
model.glmnet <- train(as.factor(Origen)~., data=data,
                  trControl=train_control, method="glmnet",metric=METRIC)
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground

## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground

## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground

## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground

## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
```

```
## ground

## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground

## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground

## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground

## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground

## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground

## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground

## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground

## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground

## Warning: from glmnet Fortran code (error code -69); Convergence for 69th
## lambda value not reached after maxit=100000 iterations; solutions for
## larger lambdas returned

## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```r
p <- predict(model.lda$finalModel,x,type="class")
```

```r
print(table(p$class,y))
```

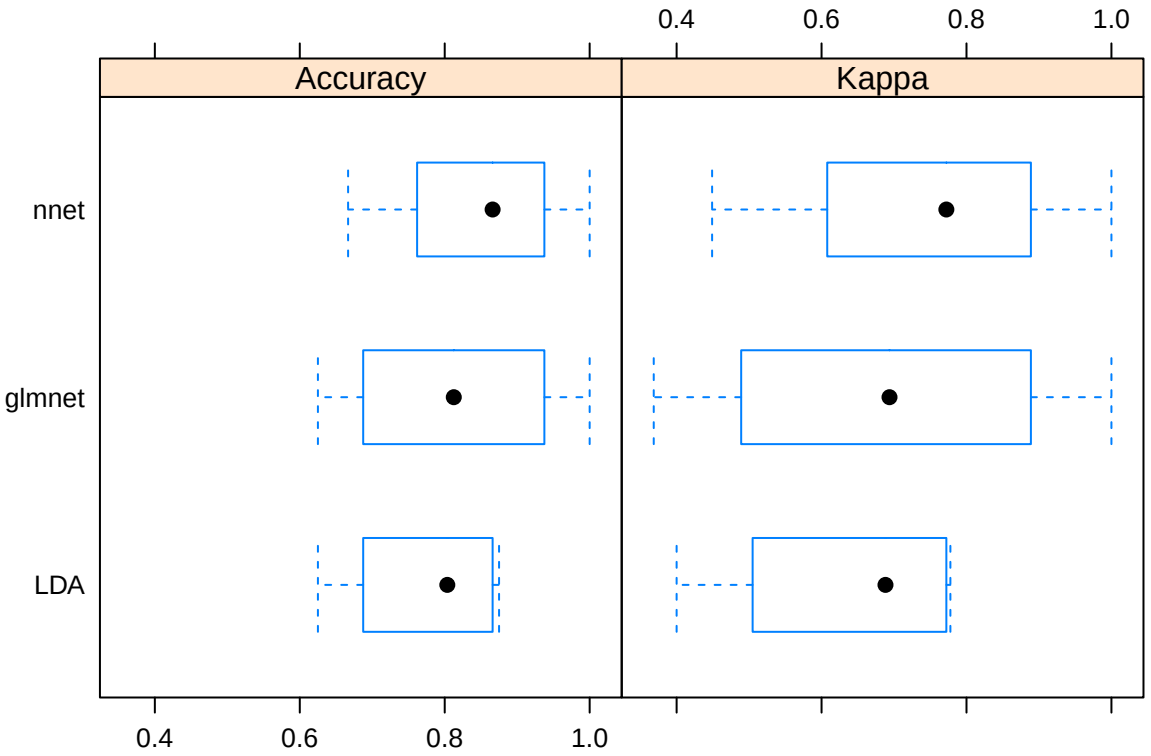
```
##     y
##      AR BR CH
##   AR 15  0  0
##   BR  1  9  1
##   CH  0  1  4
```

Comparación modelos

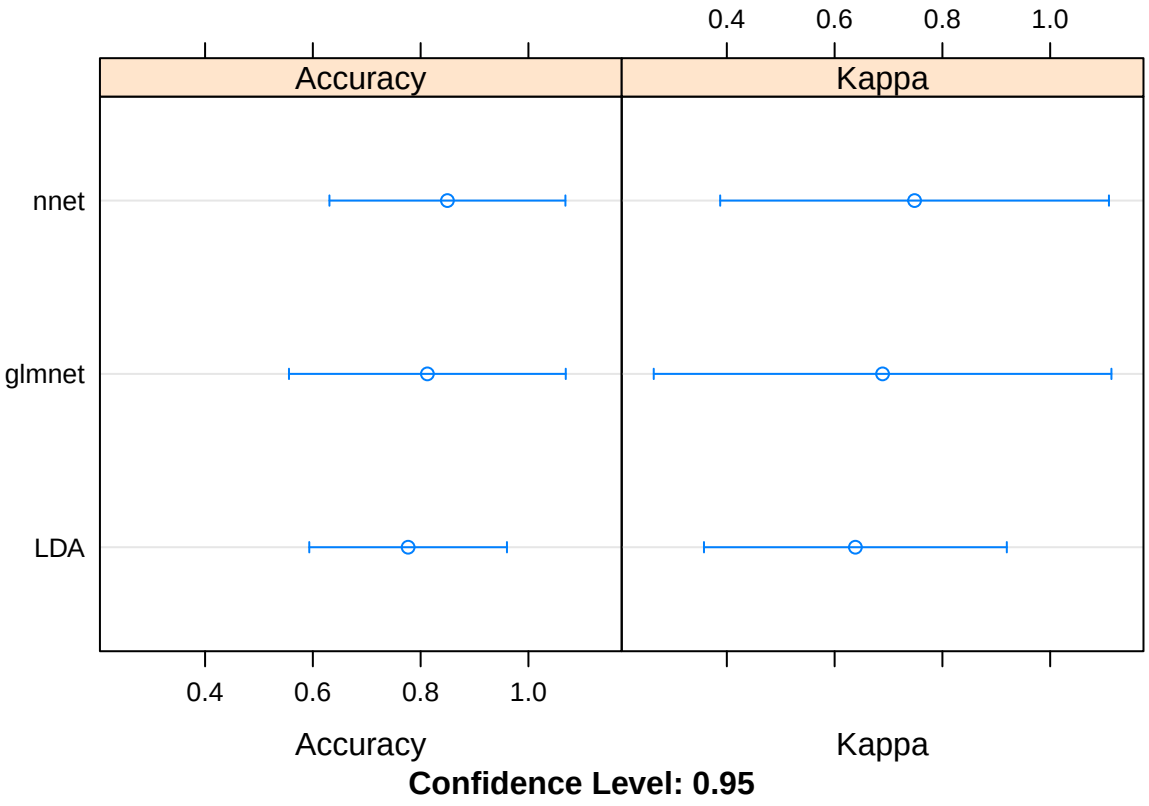
```
results <- resamples(list(LDA=model.lda,nnet=model.nnet,glmnet=model.glmnet))
# summarize the distributions
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: LDA, nnet, glmnet
## Number of resamples: 4
##
## Accuracy
##             Min.   1st Qu.    Median      Mean   3rd Qu.  Max. NA's
## LDA    0.6250000 0.7187500 0.8035714 0.7767857 0.8616071 0.875    0
## nnet   0.6666667 0.8095238 0.8660714 0.8497024 0.9062500 1.000    0
## glmnet 0.6250000 0.7187500 0.8125000 0.8125000 0.9062500 1.000    0
##
## Kappa
##             Min.   1st Qu.    Median      Mean   3rd Qu.       Max. NA's
## LDA    0.4000000 0.5573171 0.6882114 0.6385501 0.7694444 0.7777778    0
## nnet   0.4489796 0.6872449 0.7722222 0.7483560 0.8333333 1.0000000    0
## glmnet 0.3684211 0.5494223 0.6937669 0.6889887 0.8333333 1.0000000    0
```

```
# boxplots of results
bwplot(results)
```




```
# dot plots of results
dotplot(results)
```



Observamos que la red neuronal tuvo un mejor desempeño que LDA.

TODO agregar algo con bootstrapping ??

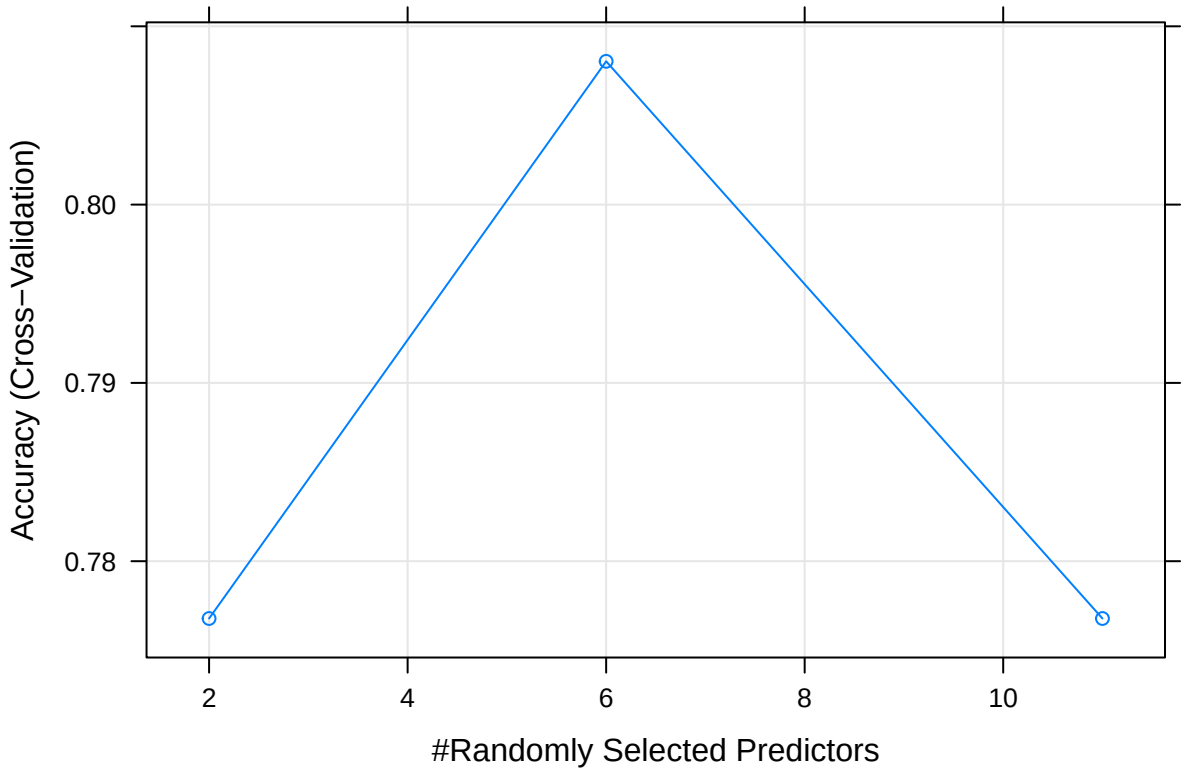
chusmeando que daba random forest

```
#' ## Random Forest
#'
set.seed(SEED)
model.rf <- train(as.factor(Origen)~., data=data,
                  trControl=train_control, method="rf",metric=METRIC, importance=T)

#'  ### Results of random forest model
print(model.rf)
```

```
## Random Forest
##
## 31 samples
## 11 predictors
##  3 classes: 'AR', 'BR', 'CH'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 23, 24, 23, 23
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
```

```
##    2    0.7767857  0.6256523
##    6    0.8080357  0.6659544
##   11    0.7767857  0.6342084
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 6.
```

```
plot(model.rf)
```




```
print(model.rf$finalModel)
```

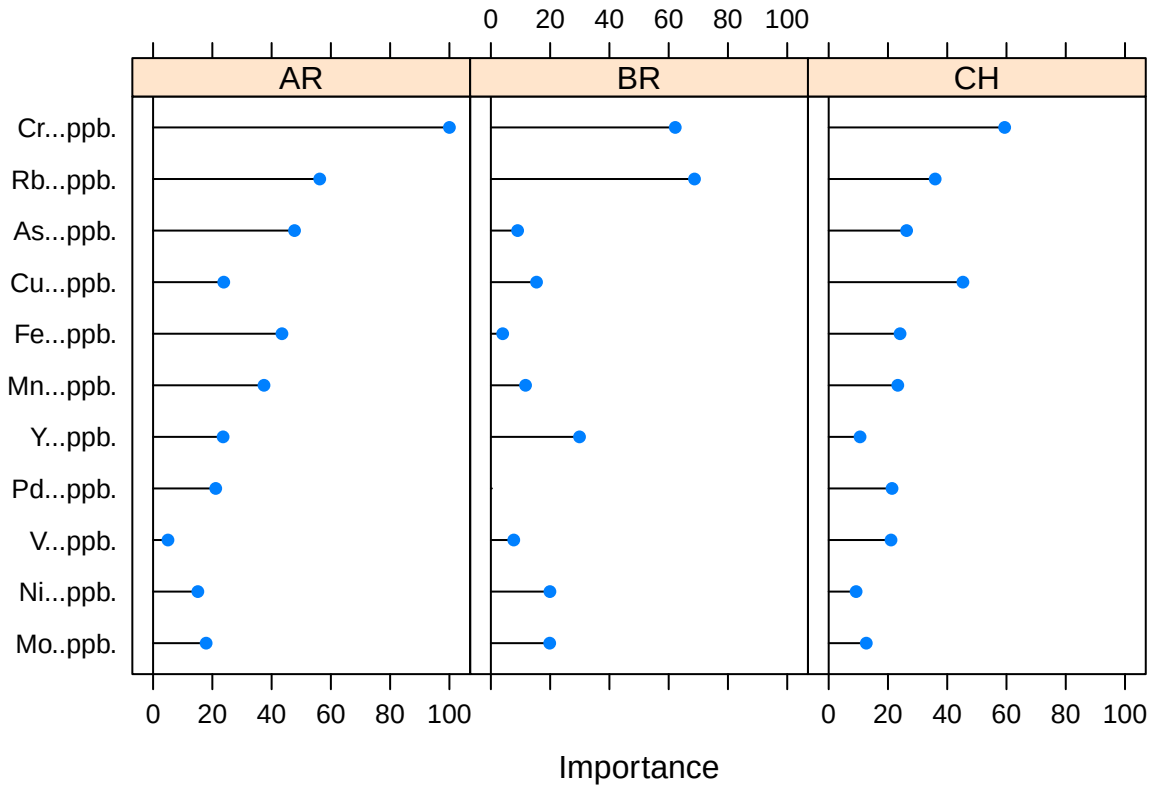
```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, importance = ..1)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 19.35%
## Confusion matrix:
##    AR BR CH class.error
## AR 16  0  0         0.0
## BR  1  7  2         0.3
## CH  1  2  2         0.6
```

```
#' ### Variable importance
varImp(model.rf)
```

```
## rf variable importance
##
##   variables are sorted by maximum importance across the classes
```

```
##                AR     BR     CH
## Cr...ppb. 100.000 62.215 59.384
## Rb...ppb.  56.237 68.715 35.934
## As...ppb.  47.735  9.021 26.298
## Cu...ppb.  23.855 15.400 45.304
## Fe...ppb.  43.483  3.982 24.077
## Mn...ppb.  37.426 11.692 23.311
## Y...ppb.   23.624 29.908 10.577
## Pd...ppb.  21.139  0.000 21.368
## V...ppb.    5.025  7.716 21.027
## Ni...ppb.  15.104 19.922  9.246
## Mo..ppb.   17.900 19.846 12.684
```

```
plot(varImp(model.rf))
```




```
#' Predicción en conjunto de testeo test-set
pred <- predict(model.rf,data)
c <- confusionMatrix(as.factor(pred), as.factor(data$Origen),mode = "prec_recall")
print(c)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction AR BR CH
##         AR 16  0  0
##         BR  0 10  0
##         CH  0  0  5
##
## Overall Statistics
##
```

```
##                Accuracy : 1
##                  95% CI : (0.8878, 1)
##     No Information Rate : 0.5161
##     P-Value [Acc > NIR] : 1.246e-09
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: AR Class: BR Class: CH
## Precision               1.0000    1.0000    1.0000
## Recall                  1.0000    1.0000    1.0000
## F1                      1.0000    1.0000    1.0000
## Prevalence              0.5161    0.3226    0.1613
## Detection Rate          0.5161    0.3226    0.1613
## Detection Prevalence    0.5161    0.3226    0.1613
## Balanced Accuracy       1.0000    1.0000    1.0000
```