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

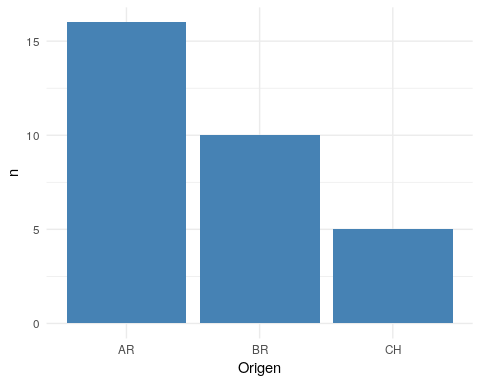
Este dataset tiene 12 variables en total, contando la variable de clase llamada Origen. El dataset consta de 31 datapoints o muestras clasificadas en 3 clases etiquetadas como AR, BR, CH

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.

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))

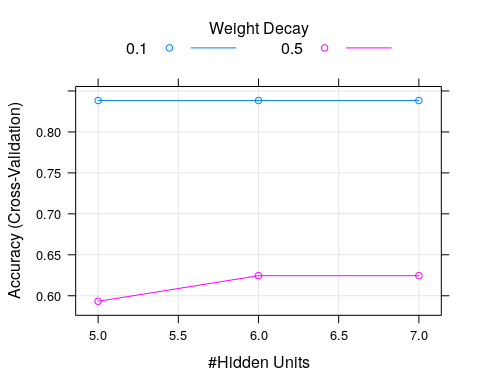
## y  
## AR BR CH  
## AR 15 0 0  
## BR 1 9 1  
## CH 0 1 4

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

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

print(table(p$class,y))

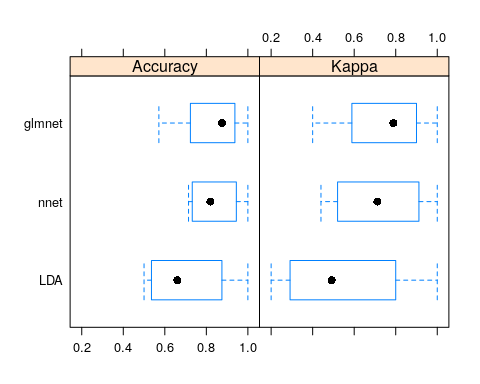
## y  
## AR BR CH  
## AR 15 0 0  
## BR 1 9 1  
## CH 0 1 4

## Comparación modelos

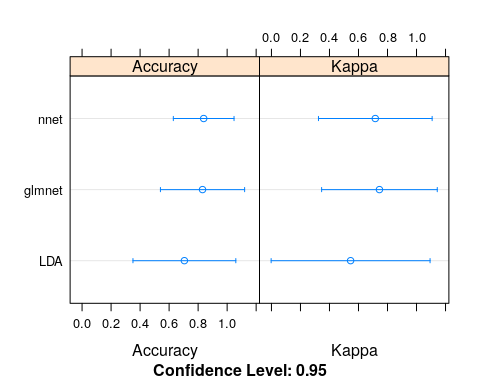
results <- resamples(list(LDA=model.lda,nnet=model.nnet,glmnet=model.glmnet))  
# summarize the distributions  
summary(results)

##   
## 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 0.5000000 0.5535714 0.6607143 0.7053571 0.8125000 1 0  
## nnet 0.7142857 0.7410714 0.8194444 0.8382937 0.9166667 1 0  
## glmnet 0.5714286 0.7991071 0.8750000 0.8303571 0.9062500 1 0  
##   
## Kappa   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## LDA 0.20 0.3367647 0.4911765 0.5455882 0.7000000 1 0  
## nnet 0.44 0.5600000 0.7117647 0.7158824 0.8676471 1 0  
## glmnet 0.40 0.6833333 0.7888889 0.7444444 0.8500000 1 0

# boxplots of results  
bwplot(results)



# dot plots of results  
dotplot(results)

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

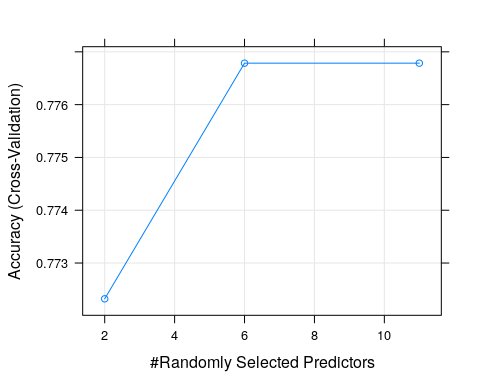
## **TODO agregar algo con bootstrapping ??**

# 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 0.7723214 0.6132285  
## 6 0.7767857 0.6164717  
## 11 0.7767857 0.6029379  
##   
## 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)



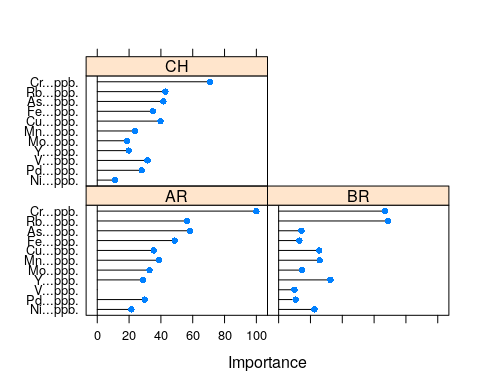
print(model.rf$finalModel)

##   
## 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. 100.00 66.804 70.90  
## Rb...ppb. 56.40 68.820 42.79  
## As...ppb. 58.24 14.312 41.58  
## Fe...ppb. 48.68 13.020 34.97  
## Cu...ppb. 35.52 25.425 39.73  
## Mn...ppb. 38.80 25.854 23.67  
## Mo..ppb. 32.89 14.845 18.71  
## Y...ppb. 28.82 32.532 19.86  
## V...ppb. 0.00 9.923 31.59  
## Pd...ppb. 29.89 10.688 27.99  
## Ni...ppb. 21.46 22.518 11.05

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