Diagnosis de diabetes en pacientes

Alicia Lozoya Colmenar

7/6/2022

Contents

Introducción

En esta PEC se va a realizar un informe para la diagnosis de diabetes en pacientes, a partir de un conjunto de variables usuales en la práctica clínica. Estas son:

· Pregnant: Número de veces embarazada

· Glucose: Concentración de glucosa plasmática a las 2 horas en una prueba de tolerancia oral a la glucosa

· Pressure: Presión arterial diastólica (mm Hg)

· Triceps: Grosor del pliegue cutáneo del tríceps (mm)

· Insulin: Insulina sérica de 2 horas (mu U/ml)

· Mass: Índice de masa corporal (peso en kg/(altura en m)2)

· Pedigree: Función de pedigrí de diabetes

· Age: Edad (años)

· Class: 0 en caso de no tener diabetes y 1 en caso contrario.

Se tiene 8 variables predictoras y una variable binaria como respuesta con valores 0 y 1.

Objetivo

En esta PEC se analizan esto datos mediante la implementación de los diferentes algoritmos estudiados: k-Nearest Neighbour, Naive Bayes, Artificial Neural Network, Support Vector Machine, Arbol de Decisión y Random Forest para diagnosticar si un paciente tiene diabetes.

Preparación del directorio

En este punto voy a organizar mi zona de trabajo que será `workingDir`.

Todos los ficheros con los datos y el código utilizados para generar este informe se pueden encontrar en un repositorio de github. [https://github.com/alocol/Machine-learning.git]

Exploración de los datos

En la siguiente tabla podemos ver las primeras filas de datos con las que vamos a trabajar.

```
  pregnant glucose pressure triceps insulin mass pedigree age class
1        6     148       72      35       0 33.6    0.627  50     1
2        1      85       66      29       0 26.6    0.351  31     0
3        8     183       64       0       0 23.3    0.672  32     1
```

```
4        1      89       66       23       94 28.1    0.167  21      0
5        0     137       40       35      168 43.1    2.288  33      1
6        5     116       74        0        0 25.6    0.201  30      0
```

Siempre que empezamos a manipular cualquier tipo de base de datos, debemos hacer una primera exploración del tipo de datos con los que trabajamos y que estos estén de la forma correcta antes de seguir con el análisis. Una forma rápida de hacer esta primera exploración es con el comando **summary()**. Podemos ver que los siguientes 5 valores despues de *pregnant* tienen un mínimo de 0, esto nos puede dar pistas que hay algo que no esta bien metido en la base de datos, o que se ha nombrado de forma diferente.

```
    pregnant         glucose         pressure         triceps
 Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
 1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00
 Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
 Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54
 3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00
 Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
    insulin           mass          pedigree           age
 Min.   :  0.0   Min.   : 0.00   Min.   :0.0780   Min.   :21.00
 1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437   1st Qu.:24.00
 Median : 30.5   Median :32.00   Median :0.3725   Median :29.00
 Mean   : 79.8   Mean   :31.99   Mean   :0.4719   Mean   :33.24
 3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262   3rd Qu.:41.00
 Max.   :846.0   Max.   :67.10   Max.   :2.4200   Max.   :81.00
     class
 Min.   :0.000
 1st Qu.:0.000
 Median :0.000
 Mean   :0.349
 3rd Qu.:1.000
 Max.   :1.000
```

En el siguiente tabla, podemos ver los 50 primero datos de la glucosa colocados en orden. Podemos observar que 5 de las personas tienen un resultado de 0 y que la siguiente en la lista es de 44, cabe pensar que es un dato que no procede en la tabla y por lo tanto debería aparecer como NA.

```
 [1]  0  0  0  0  0 44 56 57 57 61 62 65 67 68 68 68 71 71 71 71 72 73 73 73 74
[26] 74 74 74 75 75 76 76 77 77 78 78 78 78 79 79 79 80 80 80 80 80 80 81 81 81
```

De la siguiente manera se corriguen los valores 0 por NA.

```
diabetes$glucose[diabetes$glucose == 0] <- NA
diabetes$pressure[diabetes$pressure == 0] <- NA
diabetes$triceps[diabetes$triceps == 0] <- NA
diabetes$insulin[diabetes$insulin == 0] <- NA
diabetes$mass[diabetes$mass == 0] <- NA
```

Como bien se ve en la introducción tenemos una variable categórica o factor. Por lo tanto, vamos a informar al sistema de que lo trate como tal, y no numérica.

```
diabetes$class <- factor(diabetes$class)
levels(diabetes$class) <- c("negativo", "positivo")
```

Me he dado cuenta que el número de valores nulos es muy alto, alrededor de un 50%, como no tenemos la posibilidad de preguntar a los investigadores del estudio que ha sucedido con esos datos, tenemos dos opciones; eliminarlos, lo cual nos reduce mucho la muestra o reemplazarlos por algún valor como la media o mediana de los parametros. En este caso lo voy a reemplazar por la media.

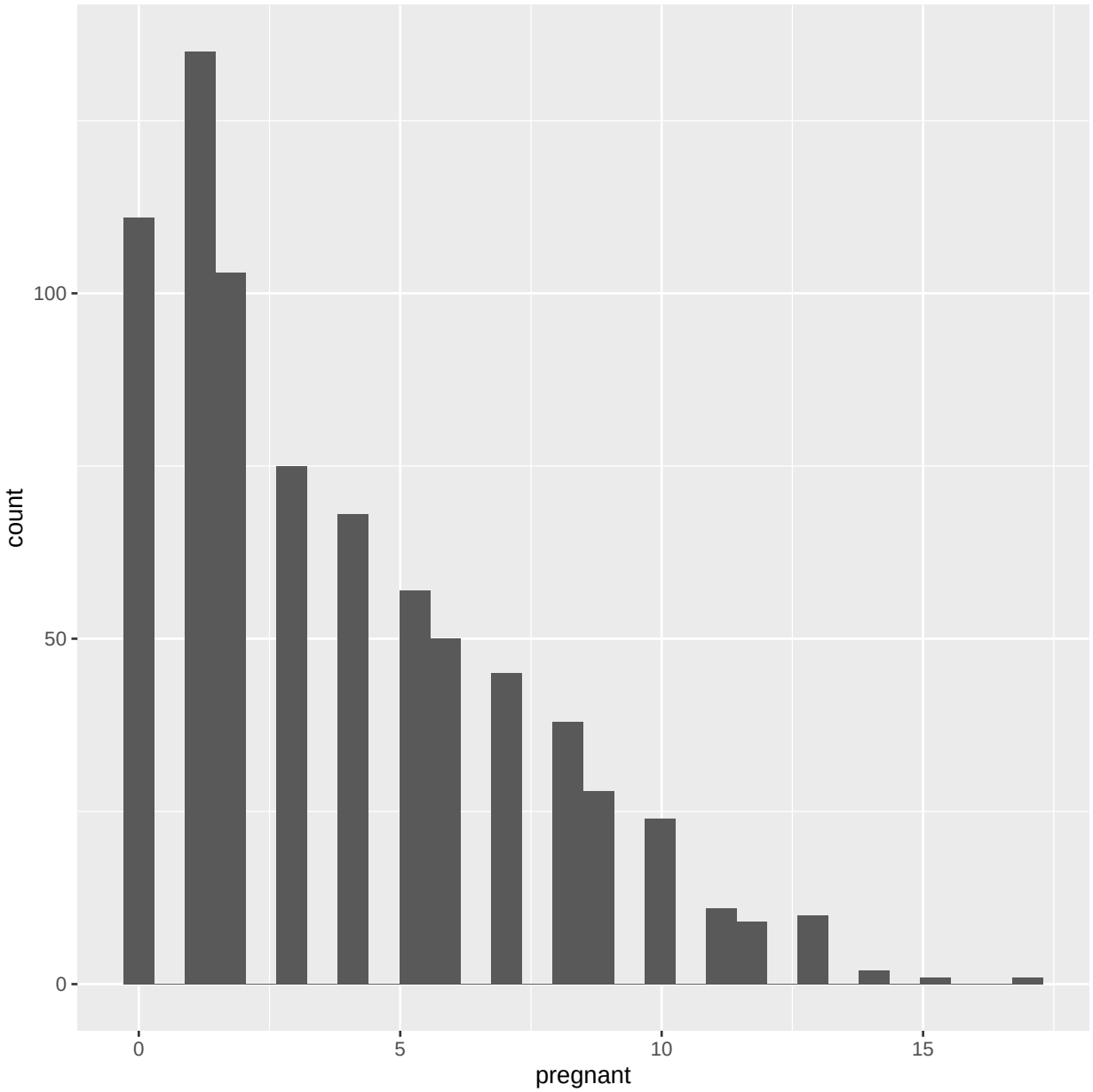
```
   pregnant         glucose          pressure         triceps
 Min.   : 0.000   Min.   : 44.00   Min.   : 24.00   Min.   : 7.00
 1st Qu.: 1.000   1st Qu.: 99.75   1st Qu.: 64.00   1st Qu.:25.00
 Median : 3.000   Median :117.00   Median : 72.20   Median :29.15
 Mean   : 3.845   Mean   :121.69   Mean   : 72.41   Mean   :29.15
 3rd Qu.: 6.000   3rd Qu.:140.25   3rd Qu.: 80.00   3rd Qu.:32.00
 Max.   :17.000   Max.   :199.00   Max.   :122.00   Max.   :99.00
   insulin           mass          pedigree           age
 Min.   : 14.0   Min.   :18.20   Min.   :0.0780   Min.   :21.00
 1st Qu.:121.5   1st Qu.:27.50   1st Qu.:0.2437   1st Qu.:24.00
 Median :155.5   Median :32.40   Median :0.3725   Median :29.00
 Mean   :155.5   Mean   :32.46   Mean   :0.4719   Mean   :33.24
 3rd Qu.:155.5   3rd Qu.:36.60   3rd Qu.:0.6262   3rd Qu.:41.00
 Max.   :846.0   Max.   :67.10   Max.   :2.4200   Max.   :81.00
     class
 negativo:500
 positivo:268
```

Gráficas de los datos

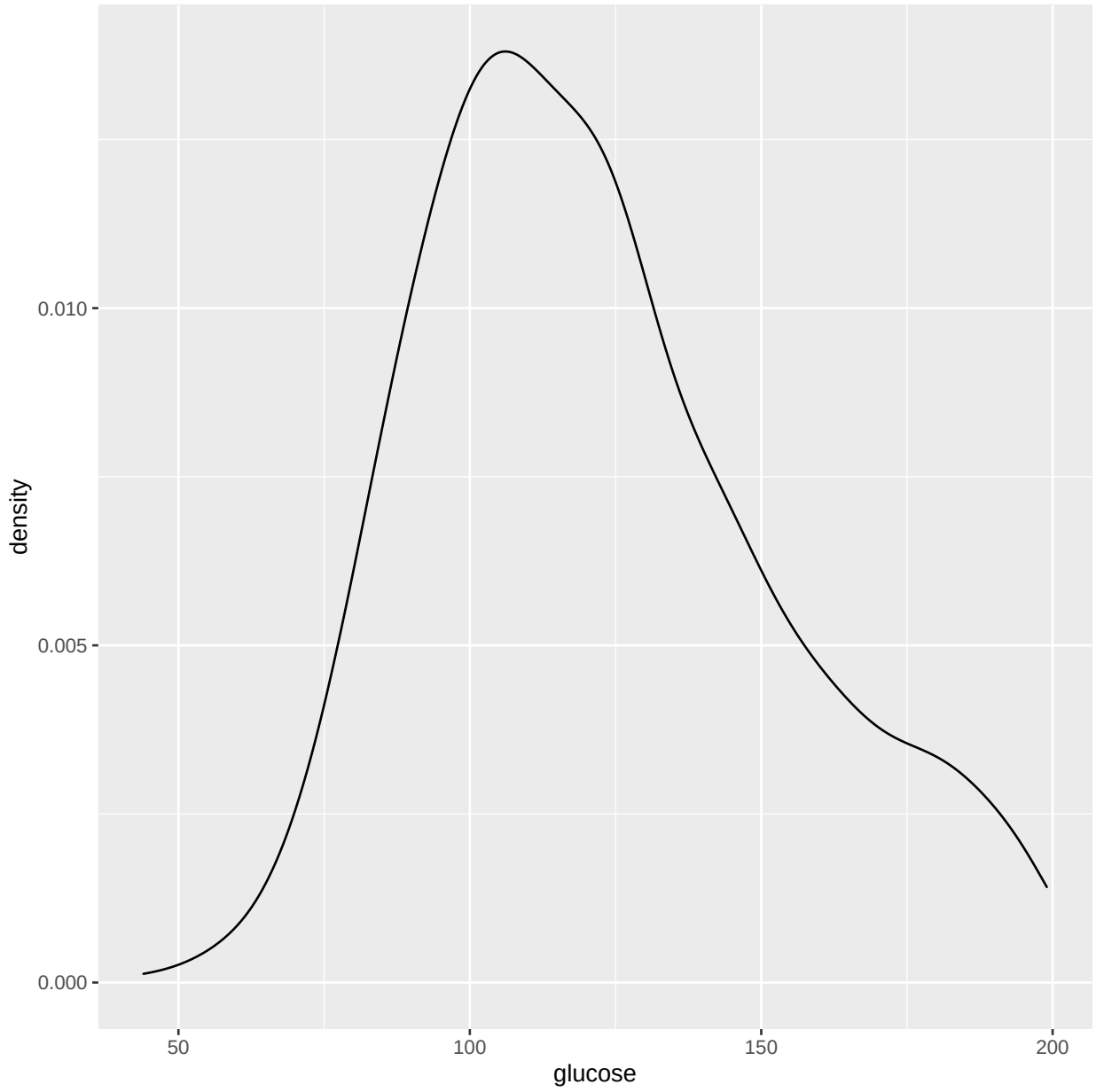
Las gráficas es una forma de ver los resultados que hemos sacado en el resumen de una forma más visual. Como por ejemplo, podemos ver como el número de embarazos se encuentra sesgado a la izquierda, lo que coincide con la media de unos 3 embarazos por persona.

Embarazos:

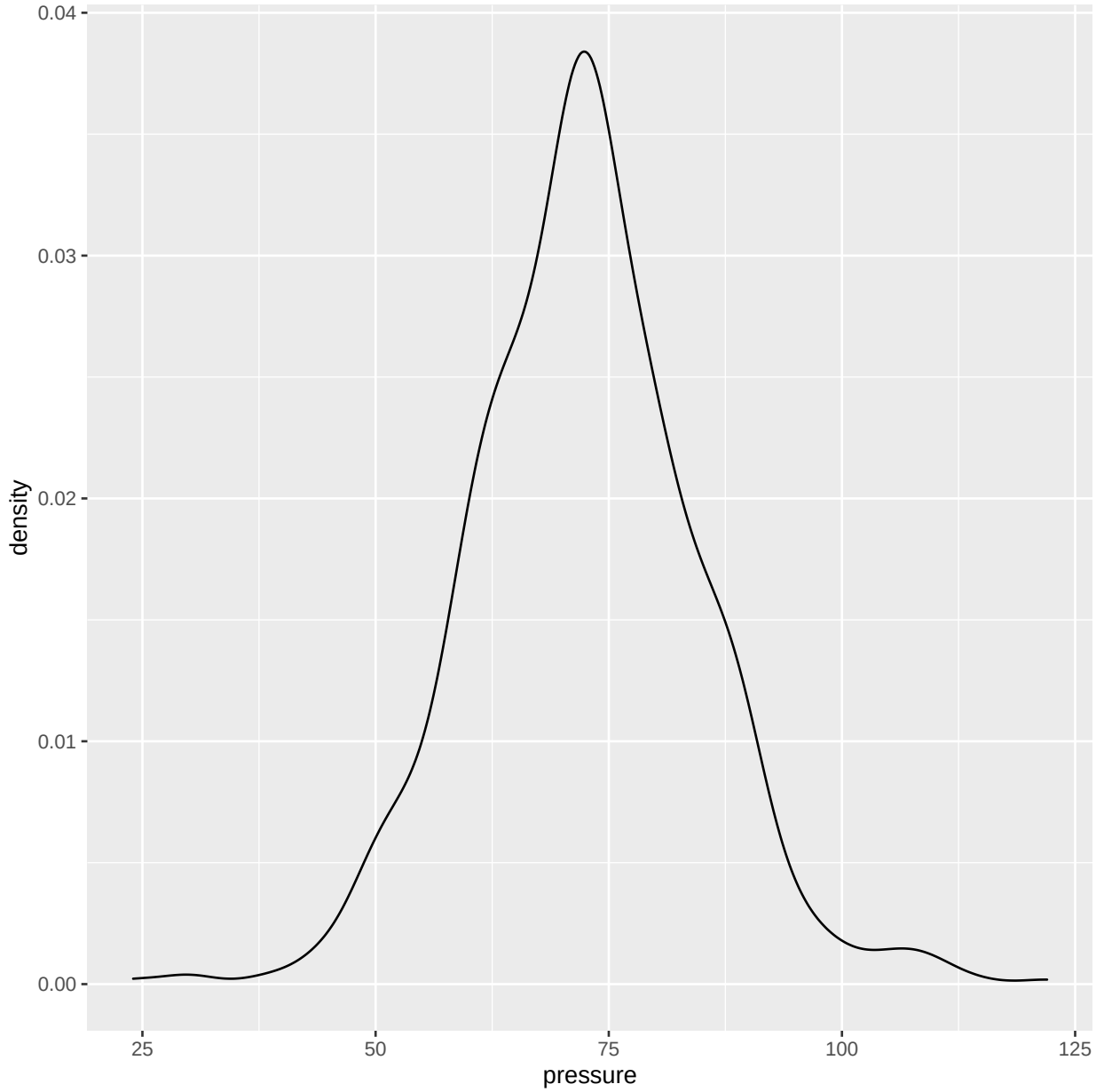
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



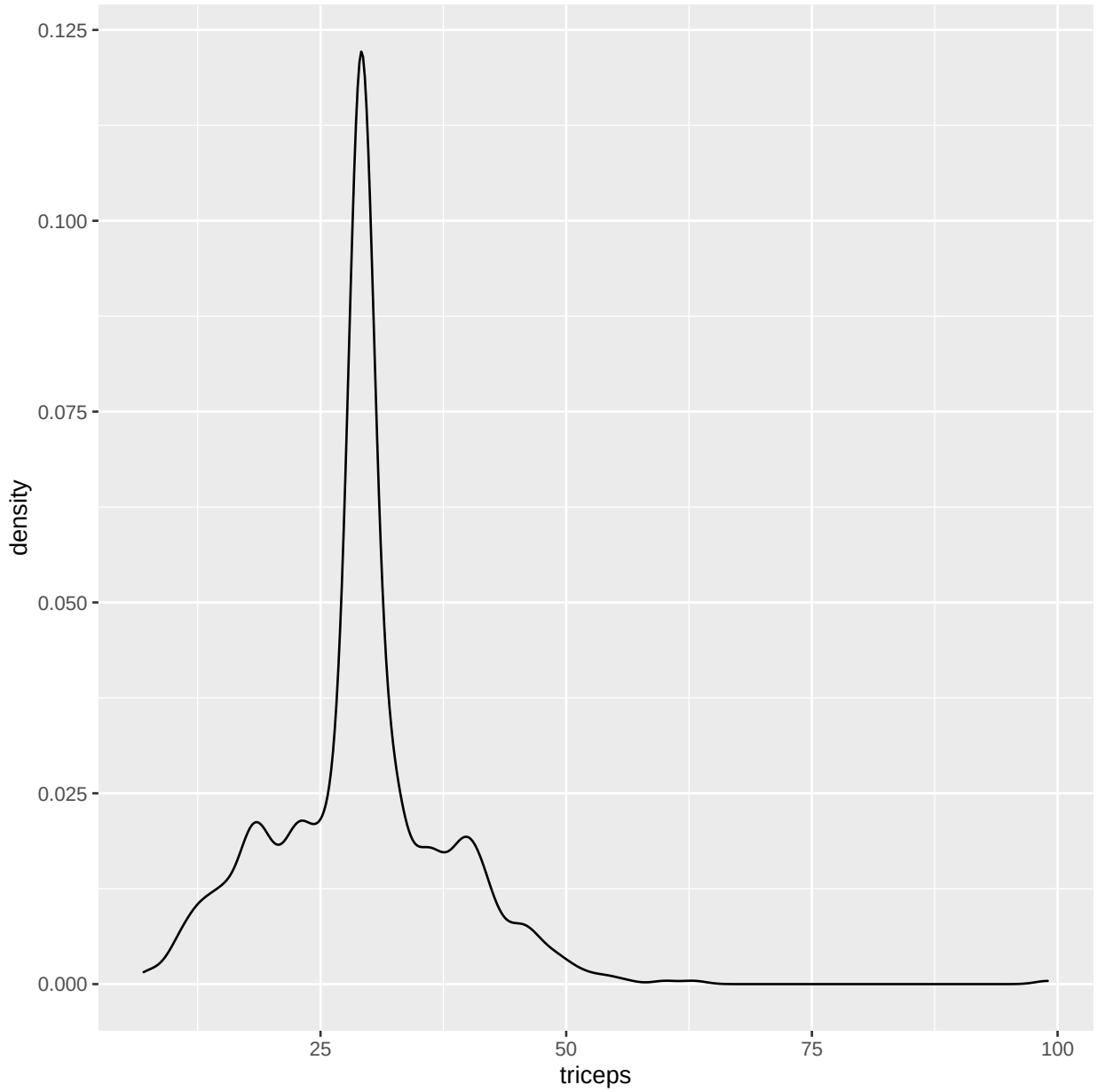
Glucosa:



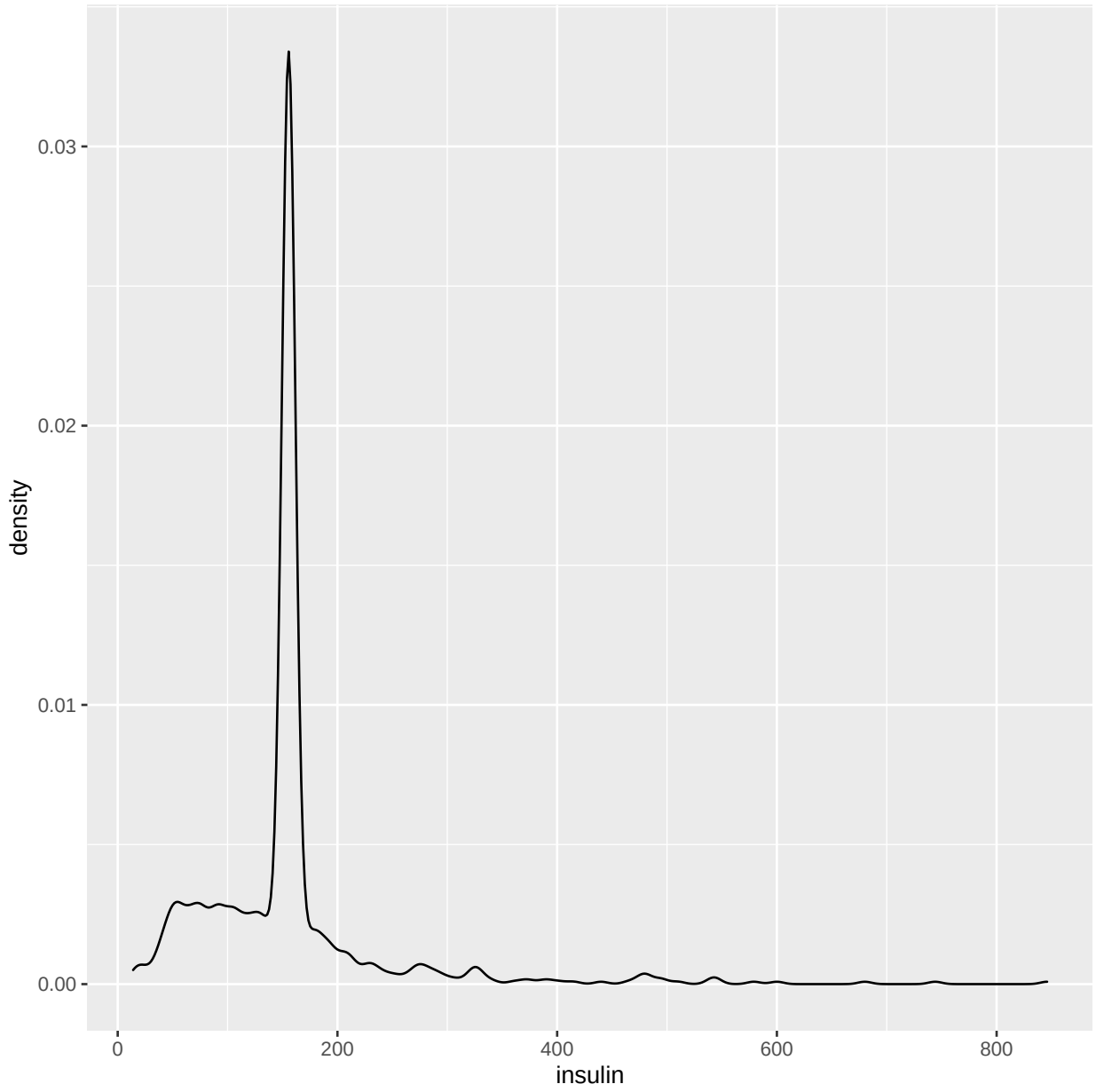
Presión:



Tríceps:

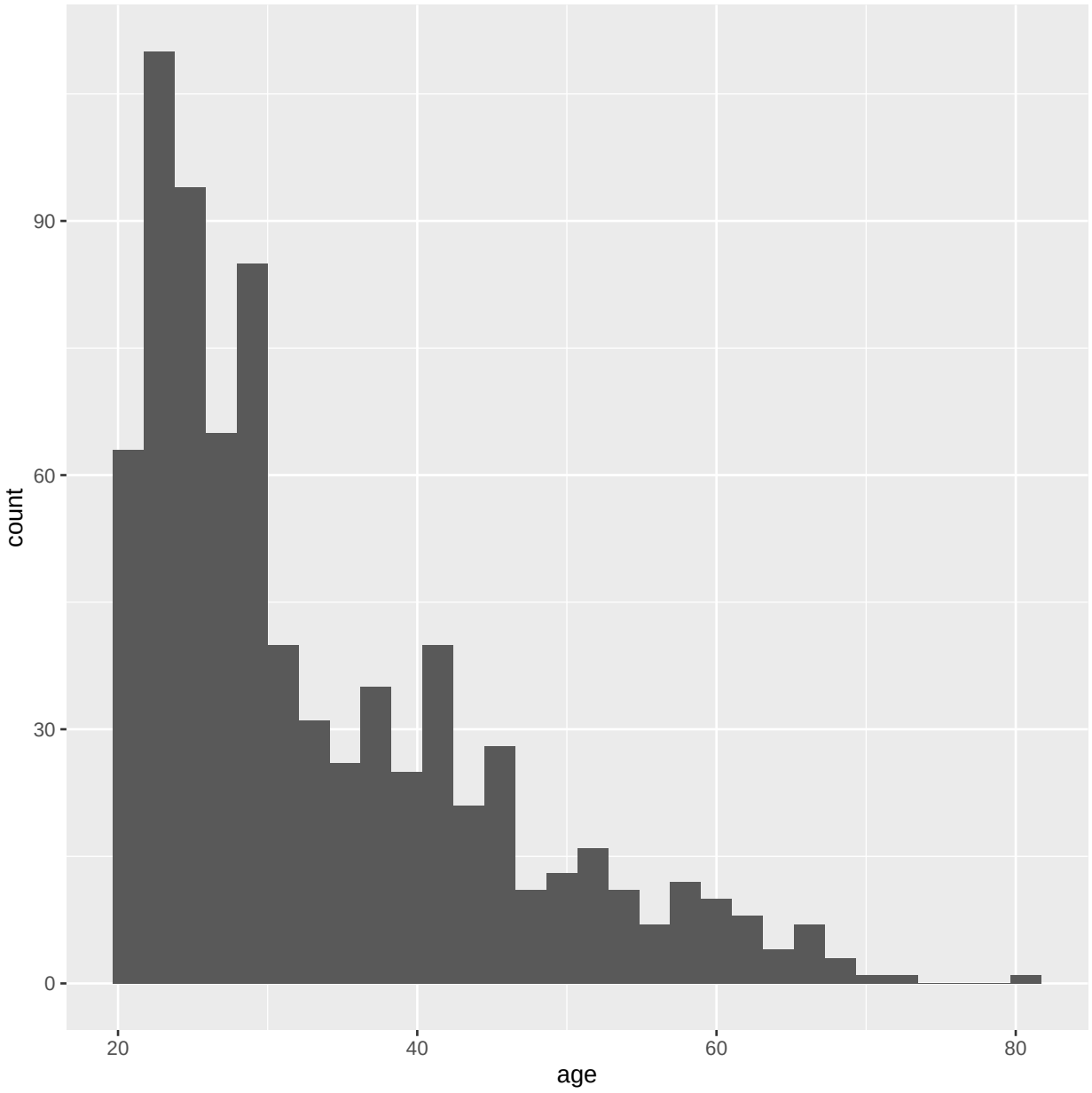


Insuluna:

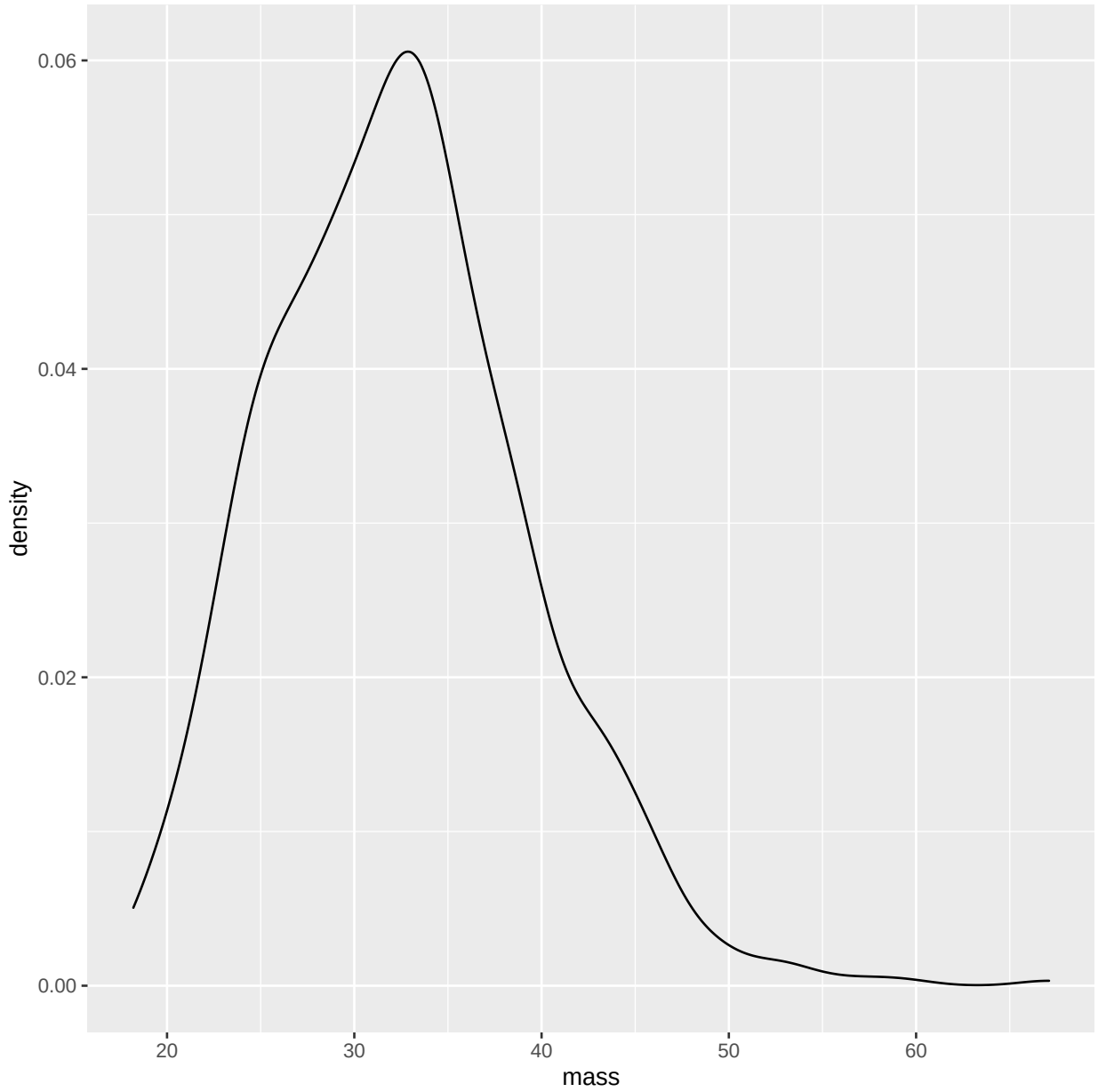


Edad:

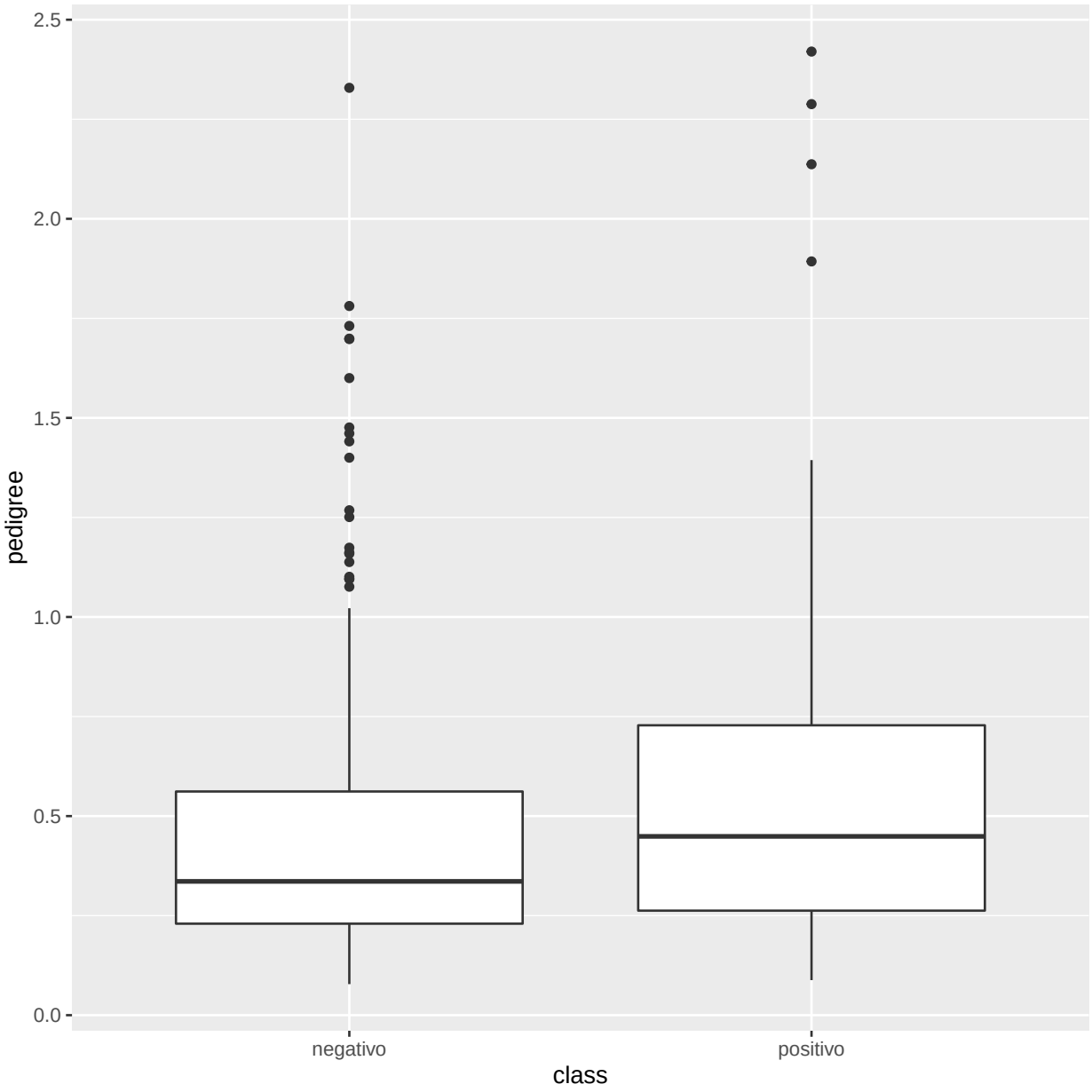
```
‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```



Masa:



Resultados de la diabetes:



Preparación de los datos: train y test

En este punto vamos a dividir los datos en dos, un 67% para el entrenamiento y un 33% para la prueba. Debemos mantener la aleatoriedad de nuestra base de datos por lo tanto utilizaré la función **sample()**. Para garantizar los resultados utilizaremos **set.seed()**.

```
set.seed(12345)
train<-sample(1:nrow(diabetes),round(0.67*nrow(diabetes)))
data.train <- diabetes[train,]
data.test <- diabetes[-train,]
```

k-Nearest Neighbour

Preparación de los datos

Como hemos visto en el libro de la asignatura, Knn depende en gran medida de la escala de medición de las variables, por lo tanto el primer paso será normalizar los datos.

```
    pregnant   glucose  pressure   triceps    insulin      mass   pedigree
1 0.35294118 0.6709677 0.4897959 0.3043478 0.17013008 0.3149284 0.23441503
2 0.05882353 0.2645161 0.4285714 0.2391304 0.17013008 0.1717791 0.11656704
3 0.47058824 0.8967742 0.4081633 0.2407980 0.17013008 0.1042945 0.25362938
4 0.05882353 0.2903226 0.4285714 0.1739130 0.09615385 0.2024540 0.03800171
5 0.00000000 0.6000000 0.1632653 0.3043478 0.18509615 0.5092025 0.94363792
6 0.29411765 0.4645161 0.5102041 0.2407980 0.17013008 0.1513292 0.05251921
        age    class
1 0.4833333 positivo
2 0.1666667 negativo
3 0.1833333 positivo
4 0.0000000 negativo
5 0.2000000 positivo
6 0.1500000 negativo
```

Entrenamiento del modelo

```
set.seed(12345)
train<-sample(1:nrow(diabetes_n),round(0.67*nrow(diabetes_n)))
data.train_n <- diabetes_n[train, -9]
data.test_n <- diabetes_n[-train,-9]
data.train.l <- diabetes[train, 9]
data.test.l <- diabetes[-train, 9]

predknn1 <- knn(train = data.train_n, test = data.test_n, cl = data.train.l, k=1)

predknn3 <- knn(train =data.train_n, test = data.test_n, cl = data.train.l, k=3)

predknn5 <- knn(train =data.train_n, test = data.test_n, cl = data.train.l, k=5)

predknn7 <- knn(train =data.train_n, test = data.test_n, cl = data.train.l, k=7)

predknn11 <- knn(train =data.train_n, test = data.test_n, cl = data.train.l, k=11)
```

Evaluación de los modelos k: 1, 3, 5, 7, 11

K:1

```
Confusion Matrix and Statistics

          data.test.l
predknn1   negativo positivo
  negativo      129       36
```

```
  positivo       40       48

               Accuracy : 0.6996
                 95% CI : (0.639, 0.7554)
    No Information Rate : 0.668
    P-Value [Acc > NIR] : 0.1583

                  Kappa : 0.3308

 Mcnemar's Test P-Value : 0.7308

            Sensitivity : 0.5714
            Specificity : 0.7633
         Pos Pred Value : 0.5455
         Neg Pred Value : 0.7818
             Prevalence : 0.3320
         Detection Rate : 0.1897
   Detection Prevalence : 0.3478
      Balanced Accuracy : 0.6674

       'Positive' Class : positivo
```

K:3

```
Confusion Matrix and Statistics

          data.test.l
predknn3   negativo positivo
  negativo      133       38
  positivo       36       46

               Accuracy : 0.7075
                 95% CI : (0.6473, 0.7628)
    No Information Rate : 0.668
    P-Value [Acc > NIR] : 0.1015

                  Kappa : 0.3366

 Mcnemar's Test P-Value : 0.9075

            Sensitivity : 0.5476
            Specificity : 0.7870
         Pos Pred Value : 0.5610
         Neg Pred Value : 0.7778
             Prevalence : 0.3320
         Detection Rate : 0.1818
   Detection Prevalence : 0.3241
      Balanced Accuracy : 0.6673

       'Positive' Class : positivo
```

K5:

```
Confusion Matrix and Statistics

          data.test.l
predknn5   negativo positivo
  negativo      134       34
  positivo       35       50

               Accuracy : 0.7273
                 95% CI : (0.668, 0.7812)
    No Information Rate : 0.668
    P-Value [Acc > NIR] : 0.02505

                  Kappa : 0.387

 Mcnemar's Test P-Value : 1.00000

            Sensitivity : 0.5952
            Specificity : 0.7929
         Pos Pred Value : 0.5882
         Neg Pred Value : 0.7976
             Prevalence : 0.3320
         Detection Rate : 0.1976
   Detection Prevalence : 0.3360
      Balanced Accuracy : 0.6941

       'Positive' Class : positivo
```

K:7

```
Confusion Matrix and Statistics

          data.test.l
predknn7   negativo positivo
  negativo      137       34
  positivo       32       50

               Accuracy : 0.7391
                 95% CI : (0.6804, 0.7921)
    No Information Rate : 0.668
    P-Value [Acc > NIR] : 0.00877

                  Kappa : 0.4083

 Mcnemar's Test P-Value : 0.90203

            Sensitivity : 0.5952
            Specificity : 0.8107
         Pos Pred Value : 0.6098
         Neg Pred Value : 0.8012
             Prevalence : 0.3320
         Detection Rate : 0.1976
   Detection Prevalence : 0.3241
      Balanced Accuracy : 0.7029
```

```
       'Positive' Class : positivo
```

K:11

```
Confusion Matrix and Statistics

          data.test.l
predknn11  negativo positivo
  negativo      144       36
  positivo       25       48

               Accuracy : 0.7589
                 95% CI : (0.7013, 0.8103)
    No Information Rate : 0.668
    P-Value [Acc > NIR] : 0.001045

                  Kappa : 0.4379

 Mcnemar's Test P-Value : 0.200415

            Sensitivity : 0.5714
            Specificity : 0.8521
         Pos Pred Value : 0.6575
         Neg Pred Value : 0.8000
             Prevalence : 0.3320
         Detection Rate : 0.1897
   Detection Prevalence : 0.2885
      Balanced Accuracy : 0.7117

       'Positive' Class : positivo
```

En la siguiente tabla veremos las 3 métricas para explicar el rendimiento. Como cabe esperar, cuanto más alta es la K, mayor exactitud presenta el modelo en el momento de acertar casos (k11 = 0.7588933). Por otro lado, los modelos con mayor sensibilidad son k5(0.5952381) y k7(0.5952381). En cuanto a la especificidad k11(0.8520710) es el que mayor porcentaje presenta.

```
  KNN  ACCURACY SENSITIVITY SPECIFICITY
1   1 0.6996047   0.5714286   0.7633136
2   3 0.7075099   0.5476190   0.7869822
3   5 0.7272727   0.5952381   0.7928994
4   7 0.7391304   0.5952381   0.8106509
5  11 0.7588933   0.5714286   0.8520710
```

Naive Bayes

Entrenamiento del modelo

Sin la activación de *laplace*.

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = data.train[-9], y = data.train.l)

A-priori probabilities:
data.train.l
 negativo  positivo
0.6427184 0.3572816

Conditional probabilities:
            pregnant
data.train.l     [,1]     [,2]
    negativo 3.093656 2.898326
    positivo 5.239130 3.679420

            glucose
data.train.l     [,1]     [,2]
    negativo 110.4945 25.21397
    positivo 140.8933 28.89031

            pressure
data.train.l     [,1]     [,2]
    negativo 71.08662 12.11688
    positivo 75.78936 11.96242

            triceps
data.train.l     [,1]     [,2]
    negativo 27.97525 8.775574
    positivo 31.65162 8.605705

            insulin
data.train.l     [,1]     [,2]
    negativo 144.7483 83.50940
    positivo 178.9154 94.38902

            mass
data.train.l     [,1]     [,2]
    negativo 31.06401 6.612260
    positivo 35.12834 6.250083

            pedigree
data.train.l      [,1]      [,2]
    negativo 0.4466405 0.3108068
    positivo 0.5448913 0.3581410

            age
data.train.l     [,1]     [,2]
    negativo 30.67069 11.29831
    positivo 38.11957 10.69839
```

Con activación de *laplace* = 1.

```
Naive Bayes Classifier for Discrete Predictors
```

```
Call:
naiveBayes.default(x = data.train[-9], y = data.train.l, laplace = 1)

A-priori probabilities:
data.train.l
 negativo  positivo
0.6427184 0.3572816

Conditional probabilities:
            pregnant
data.train.l     [,1]     [,2]
    negativo 3.093656 2.898326
    positivo 5.239130 3.679420

            glucose
data.train.l     [,1]     [,2]
    negativo 110.4945 25.21397
    positivo 140.8933 28.89031

            pressure
data.train.l     [,1]     [,2]
    negativo 71.08662 12.11688
    positivo 75.78936 11.96242

            triceps
data.train.l     [,1]     [,2]
    negativo 27.97525 8.775574
    positivo 31.65162 8.605705

            insulin
data.train.l     [,1]     [,2]
    negativo 144.7483 83.50940
    positivo 178.9154 94.38902

            mass
data.train.l     [,1]     [,2]
    negativo 31.06401 6.612260
    positivo 35.12834 6.250083

            pedigree
data.train.l      [,1]      [,2]
    negativo 0.4466405 0.3108068
    positivo 0.5448913 0.3581410

            age
data.train.l     [,1]     [,2]
    negativo 30.67069 11.29831
    positivo 38.11957 10.69839
```

Evaluación del modelo

Sin activación de *laplace*.

```
Confusion Matrix and Statistics

          Reference
Prediction negativo positivo
  negativo      137       34
  positivo       32       50

               Accuracy : 0.7391
                 95% CI : (0.6804, 0.7921)
    No Information Rate : 0.668
    P-Value [Acc > NIR] : 0.00877

                  Kappa : 0.4083

 Mcnemar's Test P-Value : 0.90203

            Sensitivity : 0.5952
            Specificity : 0.8107
         Pos Pred Value : 0.6098
         Neg Pred Value : 0.8012
             Prevalence : 0.3320
         Detection Rate : 0.1976
   Detection Prevalence : 0.3241
      Balanced Accuracy : 0.7029

       'Positive' Class : positivo
```

Activación de $laplace = 1$.

```
Confusion Matrix and Statistics

          Reference
Prediction negativo positivo
  negativo      137       34
  positivo       32       50

               Accuracy : 0.7391
                 95% CI : (0.6804, 0.7921)
    No Information Rate : 0.668
    P-Value [Acc > NIR] : 0.00877

                  Kappa : 0.4083

 Mcnemar's Test P-Value : 0.90203

            Sensitivity : 0.5952
            Specificity : 0.8107
         Pos Pred Value : 0.6098
         Neg Pred Value : 0.8012
             Prevalence : 0.3320
         Detection Rate : 0.1976
   Detection Prevalence : 0.3241
      Balanced Accuracy : 0.7029
```

```
       'Positive' Class : positivo
```

Como vemos en la siguiente tabla, los resultados entre la activación de laplace o no, no afecta a los resultados. Es decir, que no hay una mejoría en el modelo como sería la reducción de falsos positivos y falsos negativos.

```
  laplace  ACCURACY SENSITIVITY SPECIFICITY
1       0 0.7391304   0.5952381   0.8106509
2       1 0.7391304   0.5952381   0.8106509
```

Artificial Neural Network

Este apartado lo he realizado en Python, por lo tanto se encuentra en un documento en el mismo repositorio.

Para esta parte, he utilizado los datos normalizados ya generados en el aparatado de Knn.

Support Vector Machine

Entrenamiento del modelo

kernel lineal:

```
modelsvm <- ksvm(class ~ ., data = data.train, kernel = "vanilladot")
```

```
 Setting default kernel parameters
```

```
modelsvm
```

```
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 1

Linear (vanilla) kernel function.

Number of Support Vectors : 270

Objective Function Value : -264.9171
Training error : 0.225243
```

Radial bases "rbfdot":

```
modelsvm1 <- ksvm(class ~ ., data = data.train, kernel = 'rbfdot')
modelsvm1
```

```
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
```

```
 parameter : cost C = 1

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  0.135255739484634

Number of Support Vectors : 310

Objective Function Value : -235.6111
Training error : 0.180583
```

Evaluación del modelo

kernel lineal:

```
Confusion Matrix and Statistics


predsvm    negativo positivo
  negativo      147       35
  positivo       22       49

               Accuracy : 0.7747
                 95% CI : (0.7182, 0.8247)
    No Information Rate : 0.668
    P-Value [Acc > NIR] : 0.0001328

                  Kappa : 0.4715

 Mcnemar's Test P-Value : 0.1119614

            Sensitivity : 0.5833
            Specificity : 0.8698
         Pos Pred Value : 0.6901
         Neg Pred Value : 0.8077
             Prevalence : 0.3320
         Detection Rate : 0.1937
   Detection Prevalence : 0.2806
      Balanced Accuracy : 0.7266

       'Positive' Class : positivo
```

Radial basis "rbfdot":

```
Confusion Matrix and Statistics


predsvm1   negativo positivo
  negativo      149       40
  positivo       20       44

               Accuracy : 0.7628
```

```
               95% CI : (0.7055, 0.8139)
    No Information Rate : 0.668
    P-Value [Acc > NIR] : 0.0006438

                  Kappa : 0.4313

 Mcnemar's Test P-Value : 0.0141714

            Sensitivity : 0.5238
            Specificity : 0.8817
         Pos Pred Value : 0.6875
         Neg Pred Value : 0.7884
             Prevalence : 0.3320
         Detection Rate : 0.1739
   Detection Prevalence : 0.2530
      Balanced Accuracy : 0.7027

       'Positive' Class : positivo
```

Como podemos observar en la tabla el modelo lineal kernel tiene mejor rendimiento que el expresado por rbf. Por lo tanto, en este tipo de base de datos no serviría para mejorar el resultado.

```
  FUNCION  ACCURACY SENSITIVITY SPECIFICITY
1  kernel 0.7747036   0.5833333   0.8698225
2     rbf 0.7628458   0.5238095   0.8816568
```

Arbol de Decisión

Entrenamiento del modelo

Vemos que el tamaño del arbol generado es de 16, lo que indica que tienen 16 decisiones de profundidad.

```
Call:
C5.0.default(x = data.train[-9], y = data.train$class)

Classification Tree
Number of samples: 515
Number of predictors: 8

Tree size: 16

Non-standard options: attempt to group attributes
```

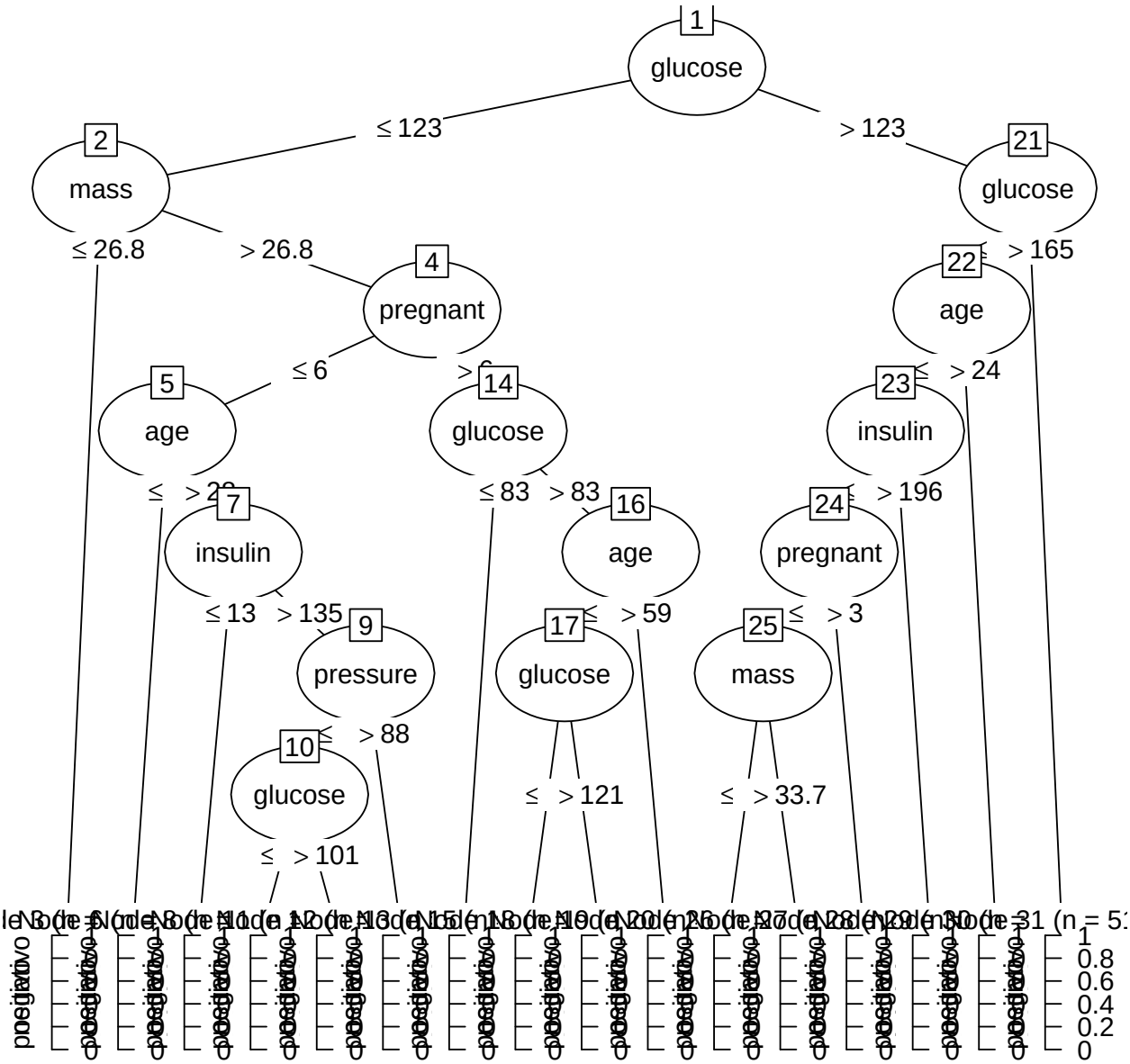
En este paso vamos a ver un resumen del modelo. Como cabe esperar la variable glucose es la que mayor asociación presenta con la diabetes.

Si nos dirigimos a la matriz de confusión del modelo podemos resaltar la tasa de error que es de un 18.3%.

```
Call:
C5.0.default(x = data.train[-9], y = data.train$class)
```

```
C5.0 [Release 2.07 GPL Edition]     Wed Jun 15 20:34:19 2022
-------------------------------

Class specified by attribute ‘outcome’

Read 515 cases (9 attributes) from undefined.data

Decision tree:

glucose > 123:
:...glucose > 165: positivo (51/6)
:   glucose <= 165:
:   :...age > 24: positivo (131/53)
:       age <= 24:
:       :...insulin > 196: negativo (12)
:           insulin <= 196:
:           :...pregnant > 3: positivo (4/1)
:               pregnant <= 3:
:               :...mass <= 33.7: negativo (16/1)
:                   mass > 33.7: positivo (6/2)
glucose <= 123:
:...mass <= 26.8: negativo (79)
    mass > 26.8:
    :...pregnant > 6:
        :...glucose <= 83: negativo (6)
        :   glucose > 83:
        :   :...age > 59: negativo (2)
        :       age <= 59:
        :       :...glucose <= 121: positivo (26/5)
        :           glucose > 121: negativo (3)
        pregnant <= 6:
        :...age <= 28: negativo (115/11)
            age > 28:
            :...insulin <= 135: negativo (13)
                insulin > 135:
                :...pressure > 88: negativo (6)
                    pressure <= 88:
                    :...glucose <= 101: negativo (11/1)
                        glucose > 101: positivo (34/14)


Evaluation on training data (515 cases):

        Decision Tree
      ----------------
      Size      Errors

        16   94(18.3%)   <<


       (a)   (b)    <-classified as
      ----  ----
```

```
       250    81    (a): class negativo
        13   171    (b): class positivo


    Attribute usage:

    100.00% glucose
     73.59% age
     61.55% mass
     46.99% pregnant
     19.81% insulin
      9.90% pressure


Time: 0.0 secs
```



Evaluación del modelo

```
Confusion Matrix and Statistics


adpred     negativo positivo
  negativo      118       18
  positivo       51       66

               Accuracy : 0.7273
                 95% CI : (0.668, 0.7812)
    No Information Rate : 0.668
    P-Value [Acc > NIR] : 0.025047
```

```
                Kappa : 0.4404

 Mcnemar's Test P-Value : 0.000117

            Sensitivity : 0.7857
            Specificity : 0.6982
         Pos Pred Value : 0.5641
         Neg Pred Value : 0.8676
             Prevalence : 0.3320
         Detection Rate : 0.2609
   Detection Prevalence : 0.4625
      Balanced Accuracy : 0.7420

       'Positive' Class : positivo
```

Mejora del modelo

Con esta forma, el programa va generando árboles de decisión y se quedará con el mejor. Vemos que el número de árboles se ha reducido a 8.2.

```
Call:
C5.0.default(x = data.train[-9], y = data.train$class, trials = 10)

Classification Tree
Number of samples: 515
Number of predictors: 8

Number of boosting iterations: 10
Average tree size: 8.2

Non-standard options: attempt to group attributes

Confusion Matrix and Statistics


adpredboost negativo positivo
   negativo      144       35
   positivo       25       49

               Accuracy : 0.7628
                 95% CI : (0.7055, 0.8139)
    No Information Rate : 0.668
    P-Value [Acc > NIR] : 0.0006438

                  Kappa : 0.4488

 Mcnemar's Test P-Value : 0.2452781

            Sensitivity : 0.5833
            Specificity : 0.8521
         Pos Pred Value : 0.6622
```

```
         Neg Pred Value : 0.8045
             Prevalence : 0.3320
         Detection Rate : 0.1937
   Detection Prevalence : 0.2925
      Balanced Accuracy : 0.7177

       'Positive' Class : positivo
```

En la siguiente tabla vemos que hay una mejoría en el rendimiento del modelo tras el boosting, pero hay que resaltar, que la sensibilidad es mejor en el primer modelo, y esta es importante para nuestro estudio.

```
   ACCURACY SENSITIVITY SPECIFICITY
1 0.7272727   0.7857143   0.6982249
2 0.7628458   0.5833333   0.8520710
```

Random Forest

Entrenamiento del modelo

Vamos a crear 2 modelos, el primero con 100 árboles y el segundo con 200.

Podemos ver que con 100 árboles, número de variables utilizadas es de 2 y con un error al clasificar las observaciones de un 25,05%.

```
Call:
 randomForest(x = data.train[-9], y = data.train$class, ntree = 100)
               Type of random forest: classification
                     Number of trees: 100
No. of variables tried at each split: 2

        OOB estimate of  error rate: 24.85%
Confusion matrix:
         negativo positivo class.error
negativo      283       48   0.1450151
positivo       80      104   0.4347826
```

Podemos ver que con 200 árboles, número de variables utilizadas es de 2 y con un error al clasificar las observaciones de un 26.02%.

```
Call:
 randomForest(x = data.train[-9], y = data.train$class, ntree = 200)
               Type of random forest: classification
                     Number of trees: 200
No. of variables tried at each split: 2

        OOB estimate of  error rate: 26.8%
Confusion matrix:
         negativo positivo class.error
negativo      275       56   0.1691843
positivo       82      102   0.4456522
```

Evaluación del modelo

Vamos a ver que tal realiza las predicciones.

```
predrf1 <- predict(modelrf1, data.test, type = "response")
predrf2 <- predict(modelrf2, data.test, type = "response")
```

Modelo de 100 árboles:

```
Confusion Matrix and Statistics


predrf1    negativo positivo
  negativo      141       32
  positivo       28       52

               Accuracy : 0.7628
                 95% CI : (0.7055, 0.8139)
    No Information Rate : 0.668
    P-Value [Acc > NIR] : 0.0006438

                  Kappa : 0.4589

 Mcnemar's Test P-Value : 0.6985354

            Sensitivity : 0.6190
            Specificity : 0.8343
         Pos Pred Value : 0.6500
         Neg Pred Value : 0.8150
             Prevalence : 0.3320
         Detection Rate : 0.2055
   Detection Prevalence : 0.3162
      Balanced Accuracy : 0.7267

       'Positive' Class : positivo


   ACCURACY SENSITIVITY SPECIFICITY
1 0.7628458   0.6190476   0.8343195
```

Modelo de 200 árboles:

```
Confusion Matrix and Statistics


predrf2    negativo positivo
  negativo      141       31
  positivo       28       53

               Accuracy : 0.7668
                 95% CI : (0.7097, 0.8175)
    No Information Rate : 0.668
```

```
    P-Value [Acc > NIR] : 0.0003884

                  Kappa : 0.4695

 Mcnemar's Test P-Value : 0.7945723

            Sensitivity : 0.6310
            Specificity : 0.8343
         Pos Pred Value : 0.6543
         Neg Pred Value : 0.8198
             Prevalence : 0.3320
         Detection Rate : 0.2095
   Detection Prevalence : 0.3202
      Balanced Accuracy : 0.7326

       'Positive' Class : positivo


   ACCURACY SENSITIVITY SPECIFICITY
1 0.7667984   0.6309524   0.8343195
```

Comparando las tablas con las métricas de rendimiento, vemos que no hay una mejora entre el modelo de 100 árboles y el de 200.

Conclusión // Discusión

En todos los algoritmos que hemos estudiado, se han propuesto 3 métricas para evaluar su rendimiento; *accuracy*, *sensitivity* y *specificity*.

Podemos decir en base a los resultados que los algoritmos que mejor funcionan son las redes neuronales, ya que tienen una exactitud del 79%, mientras que el resto esta por debajo de los 77%. En cuanto a la sensibilidad (parametro importante para diagnosticar a los posibles diabéticos) podemos decir que el árbol de decisión funciona bastante bien con un 78%, y por último la especificidad siguen siendo las redes neuronales las que tienen el resultado más alto, entre un 91% y 93%.