18-6-2019

Clasificación del set de datos “Iris”

Comparativa de la precisión de múltiples métodos de clasificación

ELENA GARRIDO JIMÉNEZ
TÉCNICAS DE SIMULACIÓN

Introducción

El set de datos Iris

En este trabajo, se intentará clasificar las tres especies del set de datos conocido como Iris (Fisher, in Becker, Chambers y Wilks, 1988) en función de los cuatro atributos principales que se recogen de las flores de estas plantas (conocidas en español como “lirios”): longitud del pétalo, longitud del sépalo, anchura del pétalo y anchura del sépalo.

Este set de datos está compuesto por un total de 150 casos, que como se presenta a continuación mediante los estadísticos descriptivos y los diagramas de cajas y bigotes, no presenta problemas graves de outliers. Se ha de tener en cuenta que, a pesar de no tener valores perdidos, es un conjunto de datos que tiene relativamente pocos casos y presenta problemas de normalidad, como se ilustra en los diagramas de cajas y bigotes.

El código empleado en este apartado es el siguiente:

```
data("iris")
data <- data.frame(iris)

print(summary(data))
print(boxplot(data[1:4]))
```

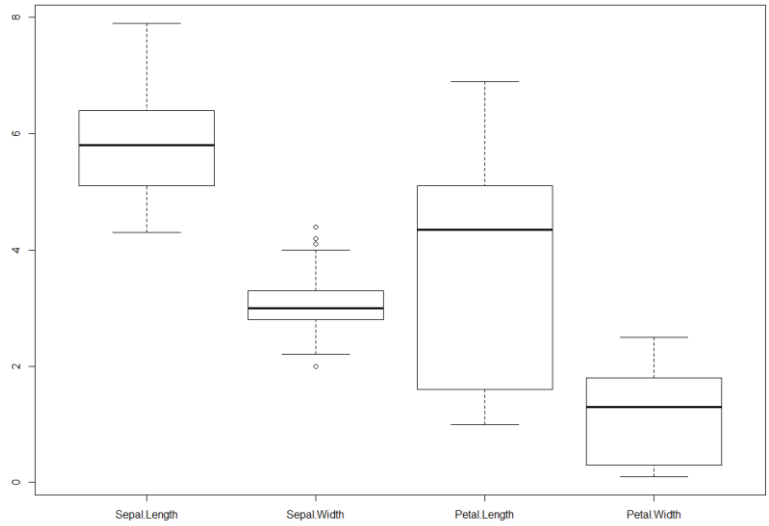


Figura 1. Gráfico de cajas y bigotes del set de datos Iris.

Tabla 1. Estadísticos descriptivos del set de datos Iris.

	Longitud del sépalo	**Anchura del sépalo**	**Anchura del pétalo**	**Longitud del pétalo**
Mínimo	5.100	2.000	1.000	0.100
Cuartil 1	5.100	2.800	1.600	0.300
Mediana	5.800	3.000	4.350	1.300
Media	5.843	3.057	3.758	1.199
Tercer cuartil	6.400	3.300	5.100	1.800
Máximo	7.900	4.400	6.900	2.500

Breve resumen de los métodos de clasificación estudiados

Análisis lineal discriminante

El análisis lineal discriminante es un algoritmo que se basa en el aprendizaje supervisado, y se usa cuando al menos hay dos clases de la variable dependiente. Se basa en algunos conceptos de la Teoría Bayesiana de la Probabilidad (James, Witten, Hastie y Tibshirani, 2013). Siendo π_k la probabilidad a priori de observar cada uno de los tipos que componen en su conjunto a la variable dependiente, se puede definir la función de densidad como $f_k \equiv P(X = x|Y = k)$, la cual variará en cuanto a sus valores de la siguiente forma: a mayor aproximación de X a x, es decir, $X \approx x$, mayor será el valor de la función y viceversa (James et al., 2013).

De este modo, y aplicándolo a este caso, se puede reescribir el Teorema de Bayes como (James et al., 2013):

$$P(Y = k|X = x) = \frac{\pi_k \cdot f_k(x)}{\sum_{l=1}^{K} \pi_l \cdot f_l(x)} \qquad \text{Ecuación 1}$$

Para facilitar el seguimiento de las fórmulas en los siguientes apartados, usemos la notación $p_k(X) = P(Y = k|X)$. Definiendo la probabilidad a posteriori de que una observación $X = x$ pertenezca a la clase k como $p_k(x)$, el método del análisis lineal discriminante lo que hace es intentar encontrar un clasificador que se aproxime al clasificador definido por Bayes (James et al., 2013). Para ello, se asume que $X = (X_1, \dots, X_p)$ se obtiene de una distribución normal multivariante, es decir, que cada una de las distribuciones que componen en su conjunto a $X \sim N(\mu_k, \Sigma)$, donde el primer término es el vector medio de una clase, y sigma representa a una matriz de covarianza, la misma para cada clase k (James et al., 2013).

De este modo, la estimación de la clase se puede definir en base a la siguiente fórmula:

$$\hat{\delta}_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k \log(\pi_k) \qquad \text{Ecuación 2}$$

que representa los conocidos límites de decisión de Bayes, definidos al intentar maximizar el valor del estimador mediante la minimización del error (James et al., 2013).

Al tratarse de un algoritmo de aprendizaje supervisado, necesita de un conjunto de datos para poder realizar el aprendizaje necesario.

Análisis cuadrático discriminante

Esta técnica de clasificación es utilizada cuando no se cumple el supuesto básico del análisis lineal discriminante, que es el supuesto de normalidad multivariante (James et al., 2013). De esta manera, asume que la observación de un dato cualquiera de la clase k sigue una distribución normal de la forma $X_k \sim N(\mu_k, \Sigma_k)$, donde Σ_k es la matriz de covarianza específica para la clase k (James et al., 2013).

De este modo, la estimación de la clase se puede definir como:

$$\hat{\delta}_k(x) = -\frac{1}{2} x^T \Sigma_k^{-1} + x^T \Sigma_k^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma_k^{-1} \mu_k + \log(\pi_k) \qquad \text{Ecuación 3}$$

que representa los conocidos límites de decisión de Bayes, definidos al intentar maximizar el valor del estimador mediante la minimización del error (James et al., 2013). De nuevo, nos hallamos ante un algoritmo de aprendizaje supervisado.

Regresión logística multinomial

Para entender la regresión logística multinomial, en primer lugar hemos de definir cómo se realiza la regresión logística binaria.

Siendo $\pi = P(Y_i = 1|X = x_i)$, donde Y=1 indica la pertenencia a un grupo determinado y X=x_i indica el valor de un conjunto de variables independientes, tenemos que la probabilidad π viene determinada por la ecuación (James et al., 2013):

$$\pi_i = P(Y_i = 1|X = x_i) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_n X_n}}{1+e^{\beta_0+\beta_1 X_1+\cdots+\beta_n X_n}} \qquad \text{Ecuación 4}$$

Es decir, la probabilidad de que un objeto sea o no clasificado en una categoría viene dada por la expresión anterior. El método de la regresión logística binaria trata de estimar los diferentes valores de los parámetros beta para optimizar la clasificación realizada, siendo $x_1, \dots, x_n$ los valores de las variables independientes a considerar (James et al., 2013).

Una regresión logística multinomial se basa en este mismo planteamiento, pero con ciertas modificaciones: si para una variable dependiente definimos cada una de sus probabilidades como $P(Y = k|X)$ donde $k \in 1, \dots K$, siendo K el número total de posibles valores de la variable dependiente, podemos definir entonces que:

$$P(Y = K|X) = 1 - P(Y = 1|X) - \cdots - P(Y = K - 1|X) \qquad \text{Ecuación 5}$$

Donde cada uno de los términos $P(Y = k|X)$ sigue a su vez la forma de la Ecuación 4. Así, podemos determinar la pertenencia a cada uno de los grupos resolviendo cada una de las ecuaciones pertinentes. Al igual que los métodos anteriores, se trata de un algoritmo de aprendizaje supervisado.

K vecinos más cercanos (k nearest neighbours)

Como en la vida real no se disponen en múltiples casos de probabilidades a priori de los distintos fenómenos, existen algunos algoritmos que intentan estimar la distribución de la variable dependiente en base a las variables independientes, como es el caso de los k vecinos más cercanos (James, 2013). En base a una observación inicial x_0 y a un número natural k, se identifican los k puntos más cercanos en base a la distancia euclídea de todas las variables numéricas consideradas a la observación inicial (representados ahora por $\mathcal{N}_0$) (James et al., 2013). Tras realizar lo anterior, se puede estimar la probabilidad condicionada de una clase l como una fracción de los puntos en $\mathcal{N}_0$ cuyos valores igualan a l (James et al., 2013):

$$P(Y = l|X = x_0) = \frac{\sum_{i \in \mathcal{N}_0} I(y_i=l)}{K} \qquad \text{Ecuación 6}$$

Por último, el algoritmo aplica el teorema de Bayes para realizar la clasificación, y clasifica la observación inicial x_0 en la clase con mayor probabilidad de ocurrencia (James et al., 2013). Se trata de un algoritmo de aprendizaje supervisado.

Red neuronal artificial

Una red neuronal es un modelo computacional cuyos cálculos se hacen con un conjunto de sencillas unidades procesadoras que funcionan en paralelo (Sun, 2008). Los seis elementos necesarios para definir una red neuronal (Rumelhart, Hinton, y McClelland, en Sun 2008) son:

1. Unidades de procesamiento (u_i): representan elementos abstractos sobre los que se pueden definir patrones con significado. Pueden ser de 3 tipos: unidades de entrada, salida u ocultas. Las unidades de entrada y salida tienen estados definidos por la tarea a realizar (en este caso, la clasificación del set de datos Iris), mientras que las unidades ocultas son parámetros libres cuyos estados pueden determinarse como sea necesario con un algoritmo de aprendizaje
2. Estado de activación (a) en un momento dado (t): el estado de las unidades se representa mediante un vector de números reales $a(t)$. Éstos pueden ser números binarios o continuos, acotados o no. Normalmente, la activación variará entre los valores 0 y 1

$$a(t) = (a_1(t), \dots, a_n(t)) \qquad \text{Ecuación 7}$$

3. Patrón de conectividad: la fuerza de la conexión entre dos unidades determinará en qué medida el estado activación de una unidad afecta al estado de activación de otra unidad en un momento posterior. La fuerza de las conexiones entre la unidad i y la j puede representarse mediante una matriz W con valores de pesos w_{ij}. Potencialmente, los pesos de la matriz permiten que cada unidad se conecte al resto de las unidades
4. Regla de propagación de los estados de activación: esta regla toma el vector a(t) de la activación de las neuronas en la capa inmediatamente anterior y lo combina con la matriz W para producir una activación en cada unidad receptora:

$$net_i(t) = W \times a(t) \qquad \text{Ecuación 8}$$

5. Regla de activación: indica cómo se combina el input neto de una unidad para producir su nuevo estado de activación. La función F deriva la nueva activación estado:

$$a_i(t+1) = F(net_i(t)) \qquad \text{Ecuación 9}$$

   Por ejemplo, F puede ser una función de tipo umbral, lineal, gaussiana o sigmoideas dependiendo del tipo de red. Es importante que la función de activación sea diferenciable para poder ejecutar el algoritmo de retropropagación, que explicaremos posteriormente
6. Algoritmo para modificar patrones de conectividad: esto permite que la red cometa menos errores a medida que recibe feedback de su ejecución anterior, modificando los valores de la conectividad entre neuronas

Al igual que todos los casos anteriores, se trata de un algoritmo de aprendizaje supervisado.

El problema del tamaño del set de entrenamiento

La información al respecto del mínimo de casos necesarios para entrenar los métodos de aprendizaje supervisados es muy escasa, no pudiendo determinar en sí una respuesta clara

para nuestro estudio, de manera que será uno de los aspectos a determinar. Como ejemplo de esta falta de consenso, un estudio de Beleites, Neugebauer, Bocklitz, Krafft y Popp (2015) encontró que para este tipo de clasificador se necesitan al menos entre 75 y 100 casos que compongan en su conjunto el set de entrenamiento, mientras que Haldar (2015) determina que una forma de determinarlo es tener como set de datos de entrenamiento un total de diez veces el número de parámetros a determinar por el modelo.

Presentación del código

El código general realizado para abordar el estudio fue el siguiente:

```
library(MASS)
library(neuralnet)
library(nnet)
library(class)

normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
data[,1:4] <- lapply(data[,1:4], normalize)

classify <- function(dataset, proporcion, output, vecinos, ocultas){
  accuracy_output <- c()
  index <- sample(nrow(data), nrow(data)*proporcion)
  training_set <- data[index, ]
  test_set <- data[-index, ]
  fit <- switch(output,
                "LDA" = lda(Species ~ Sepal.Width + Sepal.Length + Pet
al.Length + Petal.Width, data=training_set),
                "QDA" = qda(Species ~ Sepal.Width + Sepal.Length + Pet
al.Length + Petal.Width, data=training_set),
                "LMR" = multinom(Species ~ Sepal.Width + Sepal.Length
+ Petal.Length + Petal.Width, data=training_set),
                "KNN" = knn(train = training_set[,1:4], test = test_se
t[,1:4], cl = training_set[,5], k=vecinos, prob = TRUE),
                "NN" = nnet(Species ~ Sepal.Width + Sepal.Length + Pet
al.Length + Petal.Width, data=training_set, size=ocultas, maxit=100)
  )
  if(output == "KNN"){
    table_accuracy <- table(actual=test_set[,5], predicted = fit)
  }else{
    if(output == "LDA" || output == "QDA"){
      predictions <- predict(fit, test_set[,1:4])$class
    }
    else if (output == "NN" || output == "LMR"){
      predictions <- predict(fit,test_set,type="class")
    }
    table_accuracy <- as.data.frame(table(predictions, test_set[,5]))
  }
  accuracy_output <- accuracy(table_accuracy)
  return(accuracy_output)
}
```

Como se puede comprobar, el código consta de dos funciones principales: la primera de ellas determina, a partir de la transformación de la tabla de contingencia que se obtiene mediante la segunda de las funciones en un dataframe, la precisión del modelo en la predicción que realiza; por otra parte, la segunda de las funciones determina, además del método de clasificación a emplear, las predicciones que el método elegido realiza en las distintas repeticiones programadas.

Además de todo lo anterior, es importante tener en cuenta que, antes del análisis, se llevaba siempre a cabo una normalización inicial de los datos, debido a que mejora el procesamiento de los mismos y elimina el conocido problema de la dependencia de la escala, aun cuando ha de hacerse con cierta cautela (Sarle, 2002).

Técnica knn (*k-nearest neighbours)*: estudio del número de vecinos

La técnica de los k vecinos más cercanos se basa, como ya hemos comentado en el primero de los apartados, en establecer grupos definidos en principio por la distancia euclídea entre los datos para las distintas dimensiones de estudio. Sin embargo, ¿cómo se establece el número de grupos idóneo para poder realizar una buena clasificación?

Una primera aproximación nos viene dada de la mano de Tiwari, Srivastava, Srivastava y Tiwari (2016), donde se nos indica que, aun cuando tradicionalmente el valor de k se establece entre 1 y la raíz cuadrada del total de sujetos en el set de entrenamiento, se estudia este valor entre 1 y 30. En nuestro caso, intentaremos estudiar el valor de k para ese mismo rango de valores.

Para ello, el código empleado es el siguiente, además del anterior:

```
#Estudio KNN en funcion de vecinos
resultados <- c()
for (i in c(1:30)){
  resultados <- c(resultados, mean(replicate(10000, classify(data, 0.6
7, "KNN", i, 4))))
}

print(resultados)
plot(resultados, type="b")
axis(1, at = c(1:30), labels = seq(1, 30, by = 1))
```

El código funciona de la siguiente manera: cada vez que se llama a la función, se establece un nuevo conjunto de datos para el set de entrenamiento y para el set de evaluación del modelo, de manera que el algoritmo no emplea los valores realizados en una predicción para comprobar el resultado de la siguiente. Tras realizar un total de diez mil iteraciones, los resultados obtenidos se presentan a continuación:

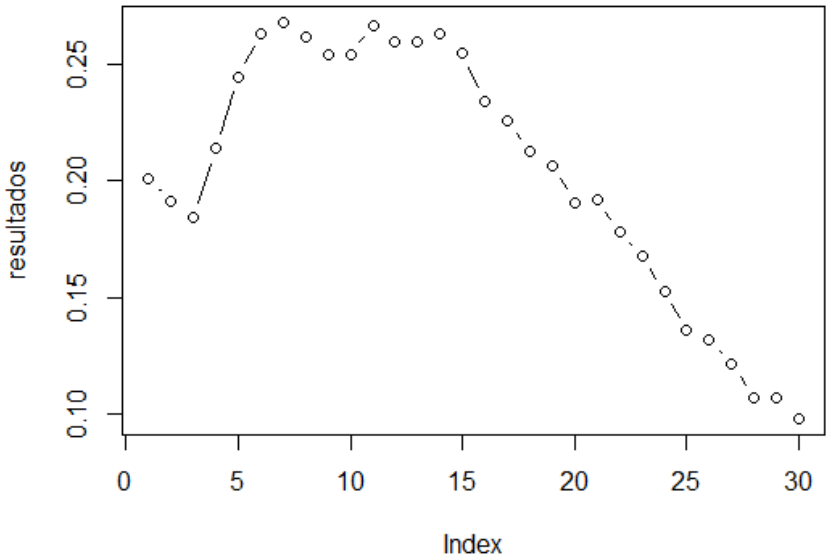


Figura 2. Resultados de la precisión en función del número de vecinos para el set de datos iris.

Tabla 2. Resultados tabulados de la precisión en función del número de vecinos. En amarillo se señala el valor máximo de la precisión y su correspondiente valor de k.

K	PRECISIÓN MEDIA
1	0.20114661
2	0.19097998
3	0.18399553
4	0.21361673
5	0.24423941
6	0.26315720
7	0.26762124
8	0.26163766
9	0.25355002
10	0.25403924
11	0.26651181
12	0.25928892
13	0.25932448
14	0.26263901
15	0.25448866
16	0.23406626
17	0.22541372
18	0.21264536
19	0.20670645
20	0.19084311
21	0.19218442
22	0.17832747
23	0.16791847
24	0.15274424

25	0.13586573
26	0.13217214
27	0.12141683
28	0.10695362
29	0.10721010
30	0.09843894

Tal y como puede comprobarse, el valor máximo de la precisión se alcanza cuando el valor de k es 11, el cual se encuentra próximo al valor de $\sqrt{100} = 10$. De esta forma, k=11 será el valor que se utilizará de cara a comprobar la precisión en el conjunto del set de prueba.

Estudio del número de neuronas en la capa oculta que se utilizará posteriormente

En este apartado, se estudiará el número de neuronas necesarias en la capa oculta para encontrar la mayor precisión. Uno de los artículos encontrados al respecto examina mediante simulación distintos métodos propuestos y determinan que el que menos error comete es uno propuesto por los mismos autores (Sheela y Deepa, 2013). La ecuación se presenta a continuación (ver Ecuación 1):

$$N_H = \frac{4n^2-3}{n^2-8} \qquad \text{Ecuación 10}$$

donde n representa el número de neuronas que conforman el input y N_H representa el número de neuronas de las capas ocultas. El número de neuronas en la capa del input viene dado por el número de variables independientes que se consideran en el modelo (Gad, 2018). De este modo, aplicando la ecuación 10, el número de neuronas que necesitaríamos en nuestra capa oculta sería 7 (truncando un resultado obtenido de 7.625).

Sin embargo, se realizará un estudio de simulación para ver si el resultado obtenido concuerda con el resultado obtenido con la ecuación 1. De manera similar al apartado anterior de los k vecinos más cercanos, se realizará un conjunto de diez mil iteraciones de la función para determinar, la precisión para el número de neuronas en la capa oculta consideradas, que variará entre 1 y 10.

El código empleado es el siguiente, junto con las funciones presentadas en la sección "Presentación del código":

```
#Estudio NN en funcion de neuronas en la capa oculta
resultados <- c()
for (j in c(1:10)){
  resultados <- c(resultados, mean(replicate(10000, classify(data, 0.6
7, "NN", 11, j))))
}
plot(resultados, type="b")
```

Los resultados obtenidos se presentan a continuación:

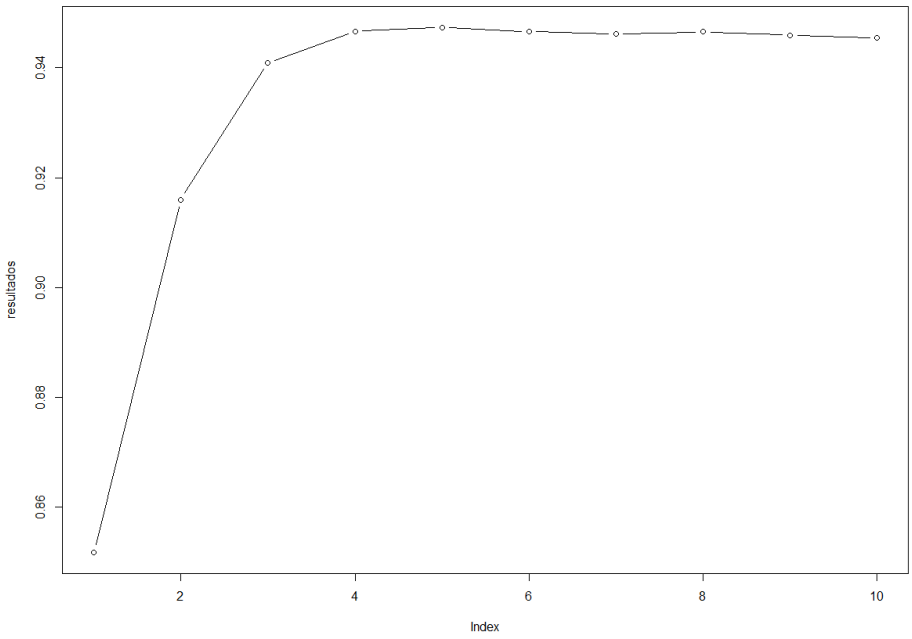


Figura 3. Media de los resultados de la precisión en función del número de neuronas en la capa oculta.

Tabla 3. Resultados tabulados de la precisión en función del número de neuronas en la capa oculta. En amarillo se señala el valor máximo de la precisión y su correspondiente valor del número de neuronas.

NÚMERO DE NEURONAS EN LA CAPA OCULTA	PRECISIÓN ALCANZADA
1	0.851712
2	0.915998
3	0.940934
4	0.946700
5	0.947346
6	0.946610
7	0.946170
8	0.946564
9	0.945942
10	0.945486

Como puede comprobarse, el valor máximo de la precisión se alcanza en diez mil repeticiones del algoritmo con un total de 5 neuronas en la capa oculta, a diferencia de las obtenidas mediante la Ecuación 1. De esta forma, se programarán 5 neuronas en la capa oculta para la siguiente sección.

Estudio de la precisión a lo largo de los distintos sets de entrenamiento

Una vez determinados con la ayuda de los apartados anteriores los parámetros necesarios para dos de los métodos de clasificación considerados en esta práctica, determinemos el método que mejor clasifica el conjunto de datos del test. Para ello, obtendremos los distintos valores medios de la precisión para cada uno tras realizar un conjunto

de diez mil iteraciones. El código específico para esta sección y los resultados obtenidos se presentan a continuación.

```
resultados_lda <- mean(replicate(10000, classify(data, 0.67, "LDA", 11
, 5)))
print(resultados_lda)

resultados_qda <- mean(replicate(10000, classify(data, 0.67, "QDA", 11
, 5)))
print(resultados_qda)

resultados_lmr <- mean(replicate(10000, classify(data, 0.67, "LMR", 11
, 5)))
print(resultados_lmr)

resultados_knn <- mean(replicate(10000, classify(data, 0.67, "KNN", 11
, 5)))
print(resultados_knn)

resultados_nn <- mean(replicate(10000, classify(data, 0.67, "NN", 11,
5)))
print(resultados_nn)
```

Tabla 4. Resultados medios obtenidos para la precisión en función del método de clasificación realizado. En amarillo se señala el valor de la mayor precisión alcanzada

	Método de clasificación utilizado				
	Análisis lineal discriminante	Análisis cuadrático discriminante	Regresión logística multinomial	K vecinos más cercanos (k = 11)	Red neuronal de una capa oculta con 5 neuronas ocultas
Precisión alcanzada	0.977308	0.973596	0.958126	0.2576907	0.946866

Podemos comprobar que la peor de las técnicas ha resultado ser la de los k vecinos más cercanos, aun cuando no necesita del cumplimiento de supuestos como ya hemos comentado anteriormente. La mejor de todas ellas resulta ser el análisis lineal discriminante, aun cuando la diferencia entre esta técnica y el resto excluyendo la de los k vecinos más cercanos resulta ser aproximadamente de 3 centésimas en la precisión. Por tanto, podemos concluir que el resto de los métodos de clasificación resultan ser apropiados para clasificar este set de datos, dado que tienen valores en la precisión muy cercanos a la unidad.

Es importante remarcar en esta sección un hecho importante: normalmente el conjunto del set de datos disponibles se divide en tres secciones en lugar de dos (Ng, n.f.).

La primera de las partes es una de las utilizadas en este trabajo, que se conoce como el set de entrenamiento o *training set*, que compone aproximadamente el 60% de los datos disponibles y que se utiliza para entrenar a los algoritmos de clasificación utilizados que requieran aprendizaje (Ng, n.f.).

La segunda de las partes en las que se divide el conjunto de los datos se conoce como el set de validación cruzada o el *cross-validation set*, que compone aproximadamente el 20% de los datos y que se utiliza para medir la variable dependiente considerada (precisión, índice kappa de concordancia…) para los distintos métodos y sus variantes (como hemos hecho en este mismo apartado) (Ng, n.f.).

La tercera y última de las partes es el conocido propiamente como el set de prueba o el *test set*. Su utilidad es la siguiente: dado que los datos de la sección anterior pueden estar sesgados y favorecer a un tipo de técnica sobre el resto, e incluso dar una precisión que no se cumple en la vida real, se reserva el restante 20% de los datos disponibles para evaluar de nuevo el método con mejores resultados en el set de validación cruzada ante datos distintos. De esa manera, se espera que los resultados concuerden con los obtenidos en el apartado anterior, pudiendo haber graves errores en la metodología de no ser así (Ng, n.f.).

¿Por qué no se ha hecho lo mismo en este caso? Como hemos podido comprobar, se ha dividido la muestra de la siguiente forma: un primer conjunto de datos que componía el 67% de las observaciones para el *training set*, y el porcentaje restante para lo que se ha llamado el *test set*, que se corresponde con una primera valoración de la eficacia de las distintas técnicas consideradas. Esto se debe a que una de las grandes limitaciones del set de datos Iris es que tan solo cuenta con 150 casos, de manera que dividir el conjunto de datos en tres partes podría haber limitado enormemente los resultados obtenidos debido a un pobre set de entrenamiento para aquellos métodos de clasificación que siguen aprendizaje supervisado. De este modo, aun cuando se es consciente de que no se sigue una metodología correcta, podemos asumir que puede ser una primera aproximación al estudio de esta base de datos.

Estudio de la precisión de los distintos algoritmos de clasificación en función del conjunto total de datos disponibles

Anteriormente hemos dispuesto del total de datos que componen el set de datos Iris para poder evaluar la eficacia de los diferentes métodos de clasificación. ¿Pero en qué medida estos métodos se ven afectados por el tamaño del conjunto de datos que componen el set total? En la sección inmediatamente anterior hemos hecho referencia a este problema, tomando incluso la decisión de eliminar el *cross-validation set* con el objetivo de distribuir los datos que lo compondrían entre el set de datos de entrenamiento y el de prueba.

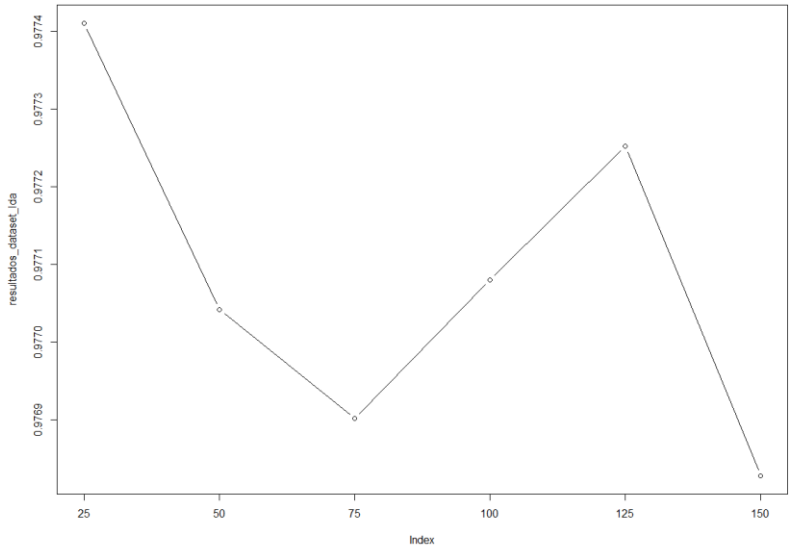
En esta sección estudiaremos el problema planteado, por lo que se irá ampliando progresivamente el conjunto total del set de datos disponibles y se estudiará la precisión obtenida por cada una de las técnicas de clasificación consideradas. Para todos los casos, la proporción de casos que compondrán el conjunto de datos de entrenamiento (67%) y evaluación (33%) no cambiará.

El código programado concretamente para este ejercicio es el que se muestra a continuación. La función cambio_dataset permite seleccionar de manera aleatoria el conjunto de datos, y así obtener un nuevo set de datos que contiene menor número de observaciones que el primero.

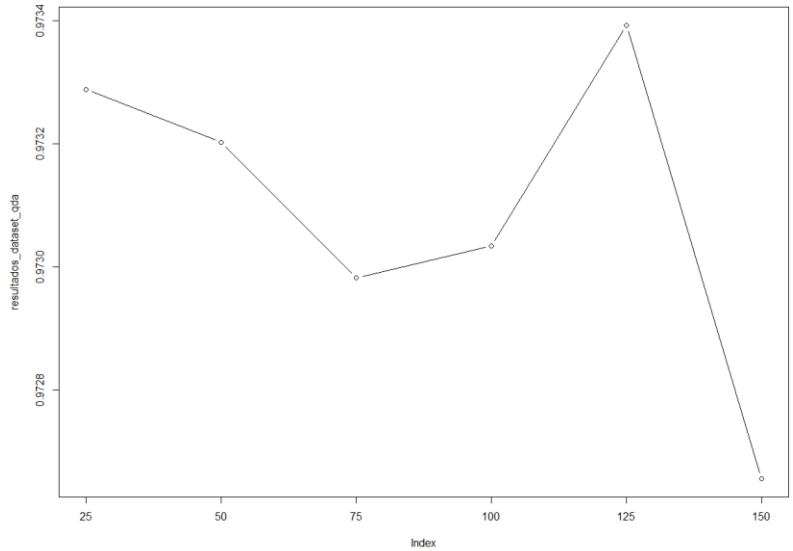
```
cambio_dataset <- function(size, dataset){
  index <- sample(nrow(dataset), size)
  dataset <- dataset[index,]
}
```

```r
# Resultados para LDA
resultados_dataset_lda <- c()
for (i in seq(25, 150, by=25)){
  resultados_dataset_lda <- c(resultados_dataset_lda, mean(replicate(1
0000, classify(cambio_dataset(i,data), 0.67, "LDA", 11, 5))))
}
print(resultados_dataset_lda)
plot(resultados_dataset_lda, xaxt="n", type = "b")
axis(1, at=c(1:6), labels = seq(25, 150, by=25))

# Resultados para QDA
resultados_dataset_qda <- c()
for (i in seq(25, 150, by=25)){
  resultados_dataset_qda <- c(resultados_dataset_qda, mean(replicate(1
0000, classify(cambio_dataset(i,data), 0.67, "QDA", 11, 5))))
}
print(resultados_dataset_qda)
plot(resultados_dataset_qda, xaxt="n", type = "b")
axis(1, at=c(1:6), labels = seq(25, 150, by=25))

# Resultados para LMR
resultados_dataset_lmr <- c()
for (i in seq(25, 150, by=25)){
  resultados_dataset_lmr <- c(resultados_dataset_lmr, mean(replicate(1
0000, classify(cambio_dataset(i,data), 0.67, "LMR", 11, 5))))
}
print(resultados_dataset_lmr)
plot(resultados_dataset_lmr, xaxt="n", type = "b")
axis(1, at=c(1:6), labels = seq(25, 150, by=25))

# Resultados para KNN
resultados_dataset_knn <- c()
for (i in seq(25, 150, by=25)){
  resultados_dataset_knn <- c(resultados_dataset_knn, mean(replicate(1
0000, classify(cambio_dataset(i,data), 0.67, "KNN", 11, 5))))
}
print(resultados_dataset_knn)
plot(resultados_dataset_knn, xaxt="n", type = "b")
axis(1, at=c(1:6), labels = seq(25, 150, by=25))

# Resultados para NN
resultados_dataset_nn <- c()
for (i in seq(25, 150, by=25)){
  resultados_dataset_nn <- c(resultados_dataset_nn, mean(replicate(100
00, classify(cambio_dataset(i,data), 0.67, "NN", 11, 5))))
}
print(resultados_dataset_nn)
plot(resultados_dataset_nn, xaxt="n", type = "b")
axis(1, at=c(1:6), labels = seq(25, 150, by=25))
```

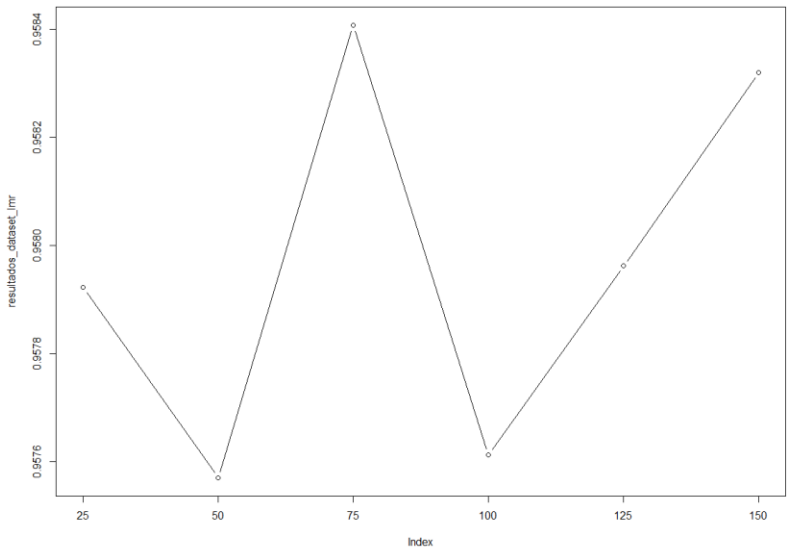
Los resultados medios obtenidos para diez mil iteraciones se presentan en las siguientes gráficas y tablas.


resultados_dataset_lda
0.9774
0.9773
0.9772
0.9771
0.9770
0.9769
25
50
75
100
125
150
Index

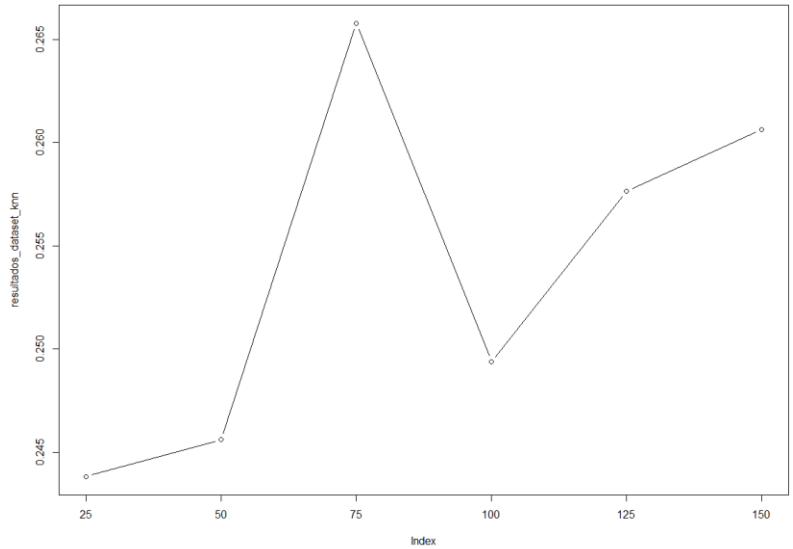
(a)


resultados_dataset_qda
0.9734
0.9732
0.9730
0.9728
25
50
75
100
125
150
Index

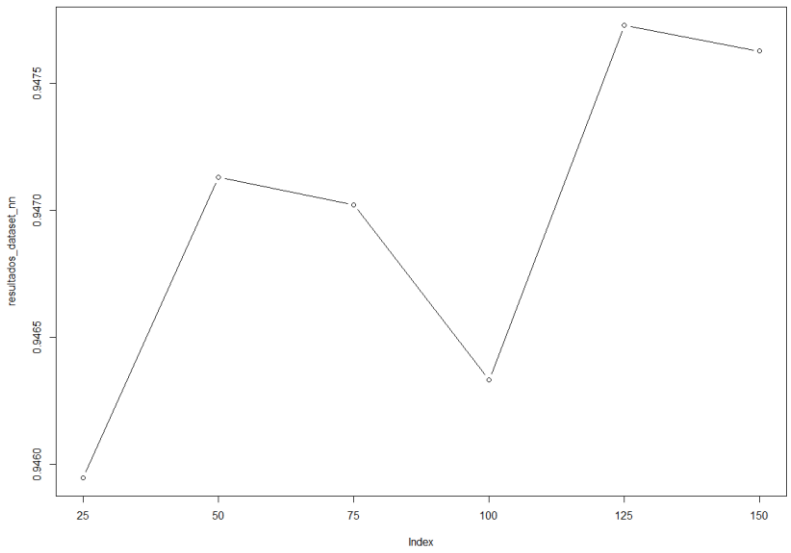
(b)


resultados_dataset_lmr
0.9584
0.9582
0.9580
0.9578
0.9576
25
50
75
100
125
150
Index

(c)


resultados_dataset_knn
0.265
0.260
0.255
0.250
0.245
25
50
75
100
125
150
Index

(d)



(e)

Figura 4. Representación gráfica de la precisión de los distintos métodos de clasificación en función del tamaño del conjunto de datos total disponibles. (a) Gráfica usando el método del análisis lineal discriminante. (b) Gráfica usando el método del análisis cuadrático discriminante. (c) Gráfica usando el método de regresión logística multinomial. (d) Gráfica usando la técnica de los k vecinos más cercanos. (e) Gráfica usando la técnica de la red neuronal.

Tabla 5. Resultados de la precisión obtenida en función del método de clasificación y del tamaño del set de datos.

	Método de clasificación				
N	Análisis lineal discriminante	Análisis cuadrático discriminante	Regresión logística multinomial	K vecinos más cercanos (k = 11)	Red neuronal de una capa oculta con 5 neuronas ocultas
25	0.977410	0.973288	0.957922	0.2438164	0.945948
50	0.977042	0.973202	0.957570	0.2456054	0.947130
75	0.976902	0.972982	0.958408	0.2657817	0.947022
100	0.977080	0.973034	0.957612	0.2493835	0.946334
125	0.977252	0.973392	0.957962	0.2606277	0.947728
150	0.976828	0.972656	0.958320	0.2576407	0.947626

Los resultados obtenidos para el conjunto de la Figura 3 y para la Tabla 4 indican que el número de datos disponible en conjunto no afectan de manera sustancial a la posterior medición de la precisión de cada método de clasificación, pudiendo ser indicativo de que estos métodos no se ven afectados por el tamaño del conjunto total de datos manteniendo constante la proporción de los datos destinados al entrenamiento del algoritmo y a la prueba del mismo.

Estudio de la precisión de los distintos algoritmos de clasificación en función del conjunto de datos disponibles para el *training* y el *test set*

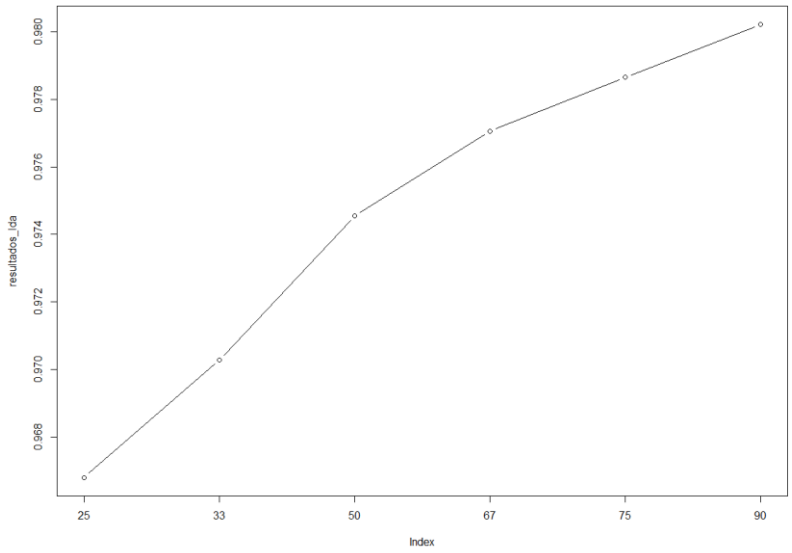
Como hemos podido comprobar anteriormente, las predicciones realizadas apenas se ven afectadas por el tamaño del set de datos en su conjunto, manteniendo una relación constante entre el conjunto de datos dedicados para el entrenamiento y el conjunto de datos dedicados para el *training* y para el *test set*. ¿Pero podemos afirmar que sucede lo mismo si, ante un tamaño constante del conjunto de datos, cambian las proporciones del conjunto de datos utilizados para el entrenamiento y la prueba de los distintos métodos?

Para responder a esta pregunta, utilizaremos el siguiente código programado. En todos los casos, los distintos sets de datos contarán con el total de los mismos (150 casos en total), y se realizarán para cada caso diez mil repeticiones del algoritmo.

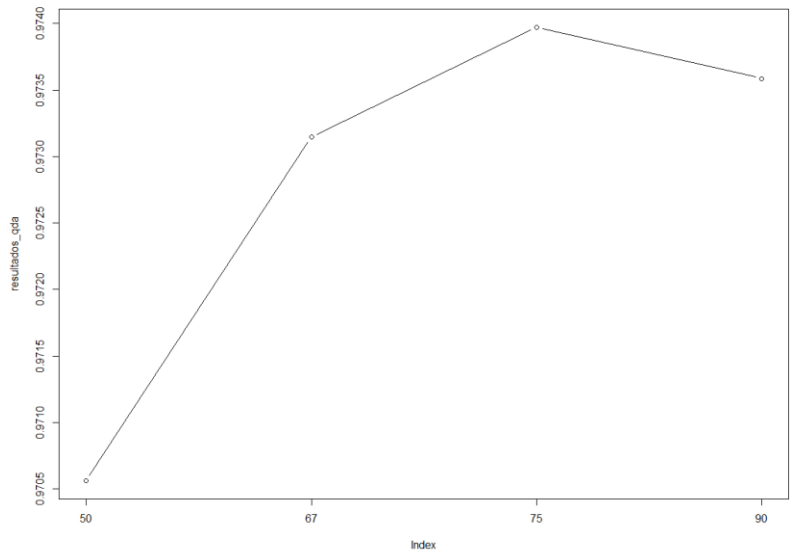
```
#Resultados para LDA
resultados_lda <- c()
for(i in c(0.25, 0.33, 0.5, 0.67, 0.75, 0.9)){
  resultados_lda <- c(resultados_lda, mean(replicate(10000, classify(d
ata, i, "LDA", 11, 5))))
}
print(resultados_lda)
plot(resultados_lda, xaxt="n", type="b")
axis(1, at=c(1:6), labels = c(25, 33, 50, 67, 75, 90))

#Resultados para qDA
resultados_qda <- c()
for(i in c(0.5, 0.67, 0.75, 0.9)){
  resultados_qda <- c(resultados_qda, mean(replicate(10000, classify(d
ata, i, "QDA", 11, 5))))
}
print(resultados_qda)
plot(resultados_qda, xaxt="n", type="b")
axis(1, at=c(1:4), labels = c(50, 67, 75, 90))

#Resultados para LMR
resultados_lmr <- c()
for(i in c(0.25, 0.33, 0.5, 0.67, 0.75, 0.9)){
  resultados_lmr <- c(resultados_lmr, mean(replicate(10000, classify(d
ata, i, "LMR", 11, 5))))
}
print(resultados_lmr)
plot(resultados_lmr, xaxt="n", type="b")
axis(1, at=c(1:6), labels = c(25, 33, 50, 67, 75, 90))

#Resultados para KNN
resultados_knn <- c()
for(i in c(0.25, 0.33, 0.5, 0.67, 0.75, 0.9)){
  resultados_knn <- c(resultados_knn, mean(replicate(10000, classify(d
ata, i, "KNN", 11, 5))))
}
print(resultados_knn)
plot(resultados_knn, xaxt="n", type="b")
axis(1, at=c(1:6), labels = c(25, 33, 50, 67, 75, 90))
```

```r
#Resultados para NN
resultados_nn <- c()
for(i in c(0.25, 0.33, 0.5, 0.67, 0.75, 0.9)){
  resultados_nn <- c(resultados_nn, mean(replicate(10000, classify(dat
a, i, "NN", 11, 5))))
}
print(resultados_nn)
plot(resultados_nn, xaxt="n", type="b")
axis(1, at=c(1:6), labels = c(25, 33, 50, 67, 75, 90))
```

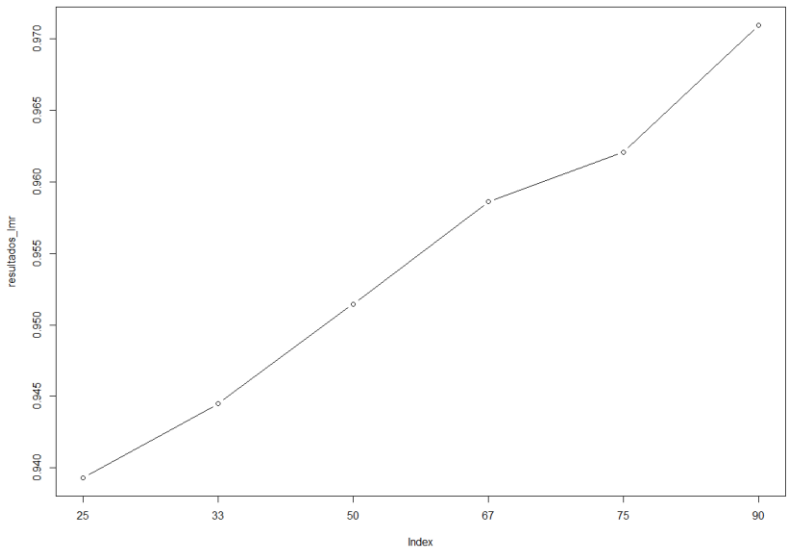
Los resultados obtenidos se presentan a continuación:



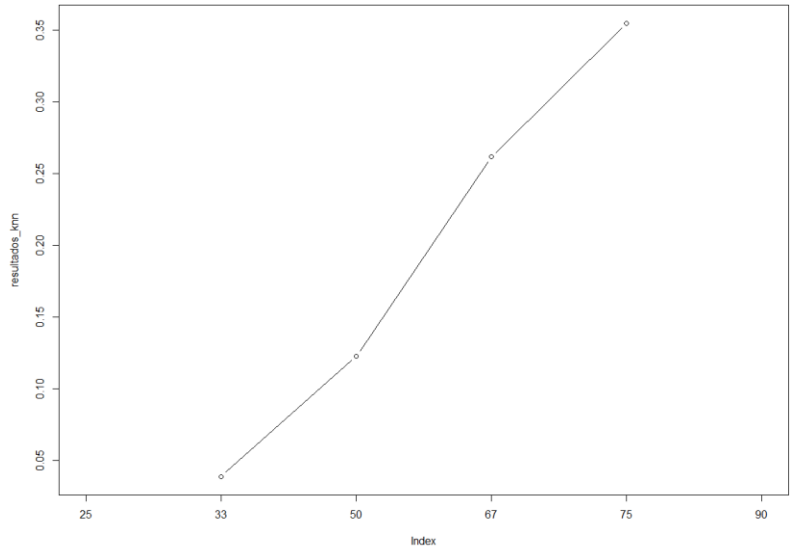
(a)



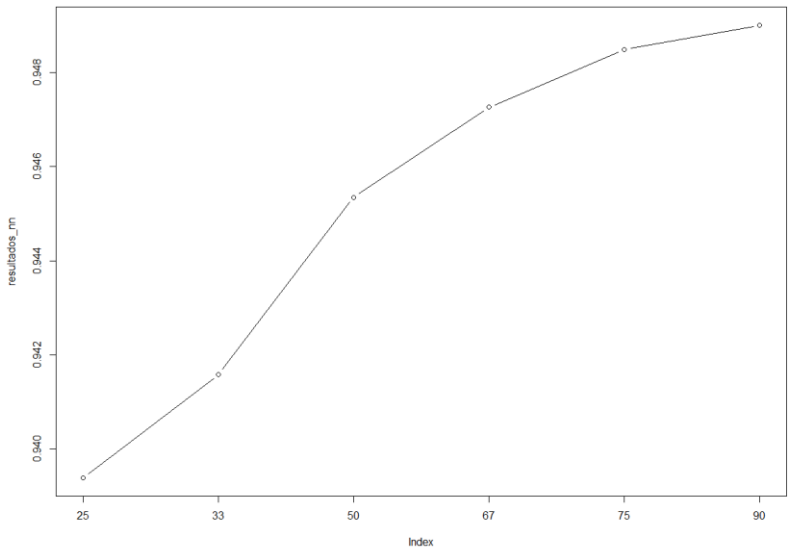
(b)


resultados_lmr
0.940
0.945
0.950
0.955
0.960
0.965
0.970
25
33
50
67
75
90
Index

(c)


resultados_knn
0.05
0.10
0.15
0.20
0.25
0.30
0.35
25
33
50
67
75
90
Index

(d)



(e)

Figura 5. Representación gráfica de la precisión de los distintos métodos de clasificación en función de la proporción de los datos destinadas al training y al test set. (a) Gráfica usando el método del análisis lineal discriminante. (b) Gráfica usando el método del análisis cuadrático discriminante. (c) Gráfica usando el método de regresión logística multinomial. (d) Gráfica usando la técnica de los k vecinos más cercanos. (e) Gráfica usando la técnica de la red neuronal.

Tabla 6. Resultados de la precisión obtenida en función del método de clasificación y de la proporción de datos destinadas al training set. La proporción de datos destinada al test set es complementaria.

	Método de clasificación				
Proporción de datos destinada al *training set* (en %)	Análisis lineal discriminante	Análisis cuadrático discriminante	Regresión logística multinomial	K vecinos más cercanos (k = 11)	Red neuronal de una capa oculta con 5 neuronas ocultas
25	0.9667929	Error: conjunto de datos demasiado pequeño	0.9393053	NaN	0.9393894
33	0.9702752	Error: rango deficiente para la especie "setosa"	0.9444960	0.03897494	0.9415842
50	0.9745400	0.9705640	0.9514667	0.12287147	0.9453520
67	0.9770600	0.9731480	0.9586160	0.26153325	0.9472640
75	0.9786474	0.9739711	0.9620842	0.35450103	0.9484895
90	0.9802200	0.9735867	0.9709667	NaN	0.9490067

Como podemos comprobar, algunos de los métodos de clasificación sí que responden negativamente al cambio en las proporciones entre los sets de datos destinados al

entrenamiento de los diferentes algoritmos o a la evaluación de los mismos. Sin embargo, encontramos que el análisis lineal discriminante, la regresión logística multinomial y la red neuronal apenas varían su rendimiento (en el caso del análisis lineal discriminante y de la red neuronal, la precisión apenas varía aproximadamente una centésima, mientras que en el caso de la regresión logística multinomial varía tan solo unas tres centésimas).

Estos resultados aclaran que, al menos para este set de datos, la precisión se ve mayormente influida que en el estudio anterior para dos de las cinco técnicas estudiadas, mientras que el resto se mantienen "robustas" ante las diferentes condiciones estudiadas.

Discusión y propuestas de mejora

Aun cuando los resultados obtenidos resultan cuanto menos llamativos, en especial los estudiados en los dos últimos apartados del artículo, resulta conveniente indicar que éstos deberían corroborarse con otros sets de datos más complejos y realistas: el set de datos es un set normalmente utilizado para el aprendizaje de métodos de clasificación debido a su sencillez y escasa complejidad. Como hemos visto en la introducción, no presentaba problemas destacables respecto a la presencia de valores atípicos, y aun cuando era notable la falta de normalidad de algunas de las variables, no existían valores perdidos a estudiar.

Lo anteriormente descrito contrasta con la vida real, donde estos problemas de los que carece el set de datos Iris están a la orden del día. Los análisis de datos llevados a cabo por estos profesionales sobre este tipo de datos pueden ser por tanto muchos más ilustrativos de la problemática, tomando este trabajo como introducción al tema.

Finalmente, algunas de las propuestas de mejora son las siguientes: además de estudiar diversos conjuntos de datos para comprobar que se confirman o desmienten los hallazgos obtenidos, sería interesante estudiar otros métodos de clasificación, como la máquina de vectores soporte, y estudiar la precisión obtenida por algoritmos de aprendizaje de tipo supervisado y no supervisado. Por último, sería conveniente estudiar otras propiedades que nos puedan informar acerca de la eficacia de los distintos métodos más allá de la precisión en sí misma, como puedan ser la sensitividad, el estadístico F_1 o el coeficiente de correlación de Matthews.

Referencias

Becker, R. A., Chambers, J. M. y Wilks, A. R. (1988) The New S Language. Wadsworth & Brooks/Cole. Recuperado de: https://rdrr.io/r/datasets/iris.html

Beleites, C., Neugebauer, U., Bocklitz, T., Krafft, C y Popp, J. (2015). Sample Size Planning for Classification Models. *Analytica Chimica Acta*, 760 (Número especial CAC2012), p. 25-33.

Gad, A. (2018). Beginners Ask "How Many Hidden Layers/Neurons to Use in Artificial Neural Networks?". Recuperado de: https://www.linkedin.com/pulse/beginners-ask-how-many-hidden-layersneurons-use-artificial-ahmed-gad/

Haldar, M. (2015). How much training data do you need? Recuperado de: https://medium.com/@malay.haldar/how-much-training-data-do-you-need-da8ec091e956

James, G., Witten, D., Hastie, T. y Tibshirani, R (Eds.). (2013). Classification. En G. James, D. Witten, T. Hastie y R. Tibshirani (Eds.), *An Introduction to Statistical Learning: with Applications in R* (pp. 127-174). New York: Springer Science+Business Media.

James, G., Witten, D., Hastie, T. y Tibshirani, R (Eds.). (2013). Classification. En G. James, D. Witten, T. Hastie y R. Tibshirani (Eds.), *An Introduction to Statistical Learning: with Applications in R* (pp. 127-174). New York: Springer Science+Business Media.

Ng, A. (n.f.). Advice for Applying Machine Learning. Coursera. Recuperado de: https://www.youtube.com/watch?v=sZSKGNbrwus&list=PLLssT5z_DsK-h9vYZkQkYNWcItqhlRJLN&index=58

Sarle, W. S. (2002). comp.ai.neural-nets FAQ, Part 2 of 7: Learning. Recuperado de: http://www.faqs.org/faqs/ai-faq/neural-nets/part2/

Sheela, K. G. y Deepa, S. N. (2013). Review on Methods to Fix Number of Hidden Neurons in Neural Networks. *Mathematical Problems in Engineering*, 2013(6): 11.

Sun, R (Ed). (2008). Connectionist models of cognition. In R. Sun (Ed). The Cambridge Handbook of Computational Psychology (pp 23-58). New York: Cambridge University Press.

Tiwari, A. K., Srivastava, R., Srivastava S. y Tiwari, S. (2016). An efficient approach for the Prediction of G-Protein Coupled Receptors and Their Subfamilies. En A. Nagar, D. P. Mohapatra y N. Chaki (Eds.). Proceedings on 3rd International Conference on Advanced Computing, Networking and informatics, Vol. 2 (pp.). New Delhi: Springer.