Tema 3. : Clasificación. Clasificadores y predictores

3.1. Métodos funcionales
3.2. Métodos bayesianos
3.3. Árboles
3.4. Métodos basados en reglas
3.5. Métodos neuronales
3.6. Predictores
3.7. Aplicaciones empresariales

INTRODUCCIÓN. VALIDACIÓN DE UNA CLASIFICACÓN/ PREDICCIÓN.

En este tema (y también en los dos próximos) empezamos a estudiar algunas de las técnicas más extendidas de la minería de datos. En concreto en éste nos centraremos en técnicas utilizadas para llevar a cabo tareas predictivas: **clasificación** y **predicción** de una variable numérica o regresión (en un sentido amplio).

La distinción básica entre uno y otro tipo de tareas es la naturaleza de la variable respuesta o variable a predecir. Mientras en la clasificación esta variable es de un tipo especial, llamado **clase**, de naturaleza binario o nominal; en el caso de la regresión, la variable a predecir sería numérica (cuantitativa).

Andamos, en definitiva, buscando la obtención de un *modelo* que permita la clasificación de individuos o la predicción del valor que toma un variable para un cierto individuo. Es lo que llamaremos: un *clasificador* o un *predictor*.

El desarrollo del tema será el de ir analizando las técnicas según la naturaleza de sus métodos de obtención y hay que remarcar que, en más de un caso, un mismo método puede usarse para llevar a cabo tanto clasificaciones como predicciones (a veces con alguna pequeña adaptación). Así por ejemplo, la regresión logística (estrictamente un método de regresión) puede usarse para clasificar, hay árboles de clasificación y de regresión, o pueden usarse redes neuronales MLP tanto para clasificar como para predecir valores numéricos.

Teniendo en cuenta esto y que en algunos entornos informáticos (en WEKA, por ejemplo), se muestran juntas las aplicaciones orientadas a una y otra tarea, también lo haremos aquí.

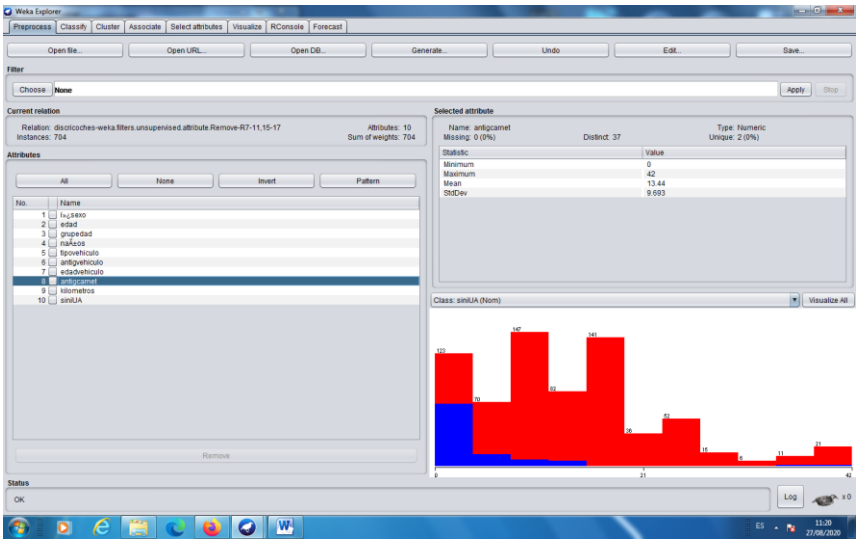
Algunos de los métodos que vamos a estudiar presentan como resultado final un tipo concreto de patrón para la clasificación o la regresión (un clasificador o un predictor) de una naturaleza muy bien definida. Este el caso de los árboles de clasificación (y regresión) de las reglas de clasificación o de las redes neuronales. Junto con estos tres grandes grupos, también consideraremos los métodos bayesianos que no dan como resultado un patrón definido por un tipo de objeto particular como los árboles, las reglas o las redes sino que ofrecen una asignación o distribución de probabilidades que permite luego la clasificación.

El resto de métodos suponen, hasta cierto punto un gran cajón de sastre en el que encontramos la regresión, el análisis discriminante y algunos otros que pueden caracterizarse de alguna forma por el empleo, tras su estimación, de funciones que permiten ejecutar después la tarea de la clasificación o la regresión.

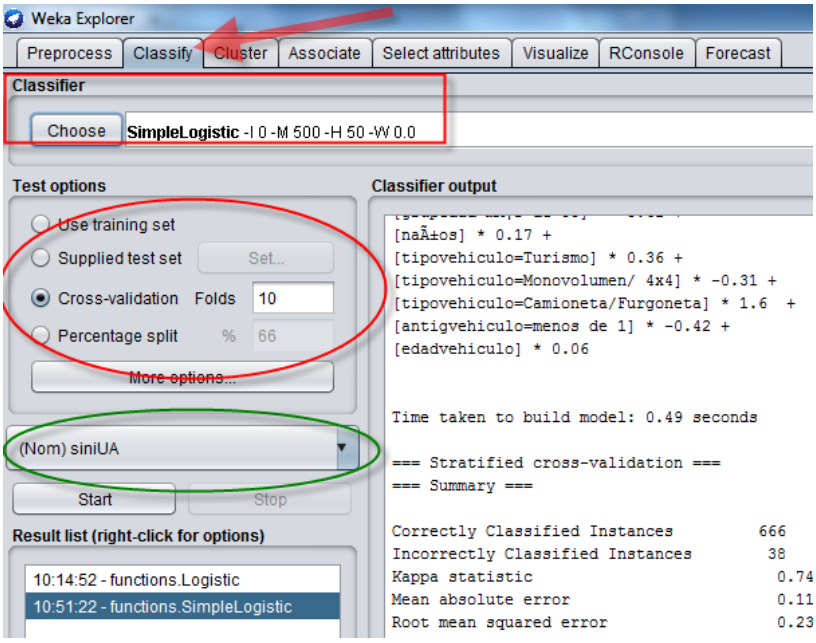
Antes de adentrarnos en el análisis de los distintos métodos parémonos a echar un vistazo a los resultados típicos de un proceso de clasificación y/o regresión y a su

análisis (evaluación). Para ilustrar la cuestión utilizaremos algunas técnicas de clasificación aplicada a un conjunto de datos concreto usando el entorno WEKA.

Vamos a utilizar la base de datos discricochesR10 (ya dispuesta en formato arff en www.uv.es/mlejarza/datamine/discricochesR10.arff). Esta base de datos cuenta con información de 10 atributos de 704 clientes de una aseguradora automovilística. El ultimo atributo (siniUA= siniestros en el último año) WEKA lo considera ,como atributo nominal o "clase" que se quiere "predecir". Para ello contamos con información del sexo, edad, grupo de edad, años en la compañía, tipo de vehículo, antigüedad del vehículo, etc.)



Si utilizamos algún clasificador (a nuestros efectos, de momento, daría igual cuál , pero usaremos una regresión logística simple con las opciones por defecto) , con las opciones de validación por defecto (validación cruzada con 10 capas) para obtener una clasificación según la variable siniUA obtendremos el objeto "clasificador" ,(en este caso una función de regresión logística) y , después una sumarización de resultados donde se muestran los principales indicadores para la evaluación de la calidad de la clasificación.



Veamos éstos con algo de detalle:

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         666               94.6023 %
Incorrectly Classified Instances        38                5.3977 %
Kappa statistic                          0.7413
Mean absolute error                      0.1136
Root mean squared error                  0.2318
Relative absolute error                 48.0629 %
Root relative squared error             67.5285 %
Total Number of Instances              704

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,667    0,010    0,914      0,667    0,771      0,753    0,928     0,790     con siniestros
               0,990    0,333    0,950      0,990    0,969      0,753    0,928     0,985     sin siniestros
Weighted Avg.  0,946    0,289    0,945      0,946    0,942      0,753    0,928     0,959

=== Confusion Matrix ===

   a   b   <-- classified as
  64  32 |   a = con siniestros
   6 602 |   b = sin siniestros
```

Se detalla primero el número y porcentaje de casos correctamente e incorrectamente clasificados (éste y los otros resultados corresponden a la validación cruzada de 10 estratos o capas).

El estadístico Kappa compara esta acuracidad general (94,6% de aciertos) con la que se produciría por puro azar asignando, eso sí, el mismo número de casos clasificados como a=con siniestros (64+32=96) y como b= sin siniestros (602+6=608).

¿Cual sería este porcentaje de aciertos por puro azar?:

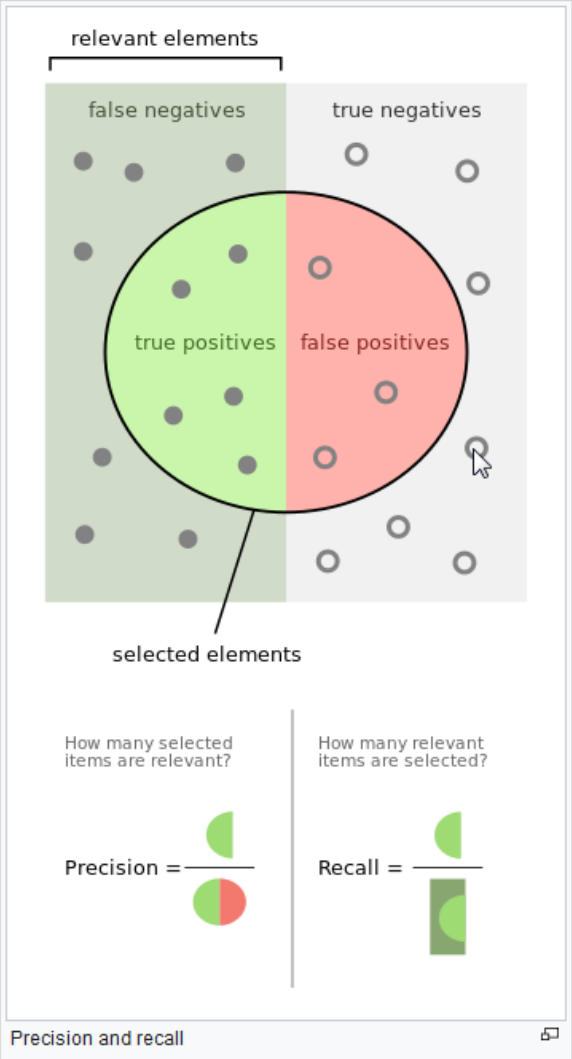
Como en la base de datos hay 70 casos con siniestros y 634 sin siniestros de un total de 704 casos, por puro azar las predicción con siniestro serían correctas en una proporción de 70/704 (es decir en 70×96/704=9.54 casos) y, de la misma forma, las predicciones sin siniestro ocurrirían un total de 608×634/704=547.54 veces. El número de aciertos por azar sería de 9.54+547.54=557.08 (en términos porcentuales 79.13%).

Pues bien, el estadístico kappa sería el cociente entre lo que ha mejorado la acuracidad con el modelo con respecto la máxima mejora posible :

$$\kappa = \frac{aciertos_{\mathrm{mod}} - aciertos_{azar}}{N - aciertos_{azar}} = \frac{666 - 557.08}{704 - 557.08} = 0.7413$$

Aparecen otros indicadores de error de fácil interpretación y una tabla con la acuracidad (acierto) en las asignaciones de cada clase, obtenida a partir de la matriz de confusión que aparece al final.

Así pues, la tabla detalla, para cada clase y también por término medio (ponderado por el tamaño de cada clase) , los siguientes indicadores:



La ratio de verdaderos positivos (TP Rate) : TP/(TP+FN) Correctos positivos sobre el total de los de la clase. (A veces se llama sensibilidad)

La ratio de falsos positivos (FP Rate) : FP/(FP+TN) Erróneos positivos sobre el total de los **auténticos** negativos (A su valor complementario, a veces , se le llama especificidad)

La precisión : el cociente entre los verdaderos positivos y el total de clasificados como positivos

La exhaustividad (recall) : el cociente entre los verdaderos positivos (detectados) y el total de efectivamente positivos , en la realidad

La medida F : media armónica entre la precisión y la exhaustividad

MCC. El coeficiente de correlación de Matthews. Toma valores entre -1 y +1 (+1 quivale a la predicción perfecta, 0 indica una predicción equivalente a la producida por azar y -1 indica una predicción en total desacuerdo (contraria) a la realidad. Su expresión la tenemos debajo

$$\mathrm{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

Las últimas dos columnas muestran indicadores obtenidos del análisis de la curva ROC (Receiver Operating Characteristic) llamada así por sus primeras aplicaciones en el análisis de señales durante la segunda guerra mundial. Y la curva precisón-exhaustividad (recall) .

La curva ROC parte de un representación en ejes coordenados donde :

en el eje de abscisas se representan los FPR (ratio de falsos positivos) o complementario de la especificidad [equivalente al ratio de verdaderos negativos TNR]

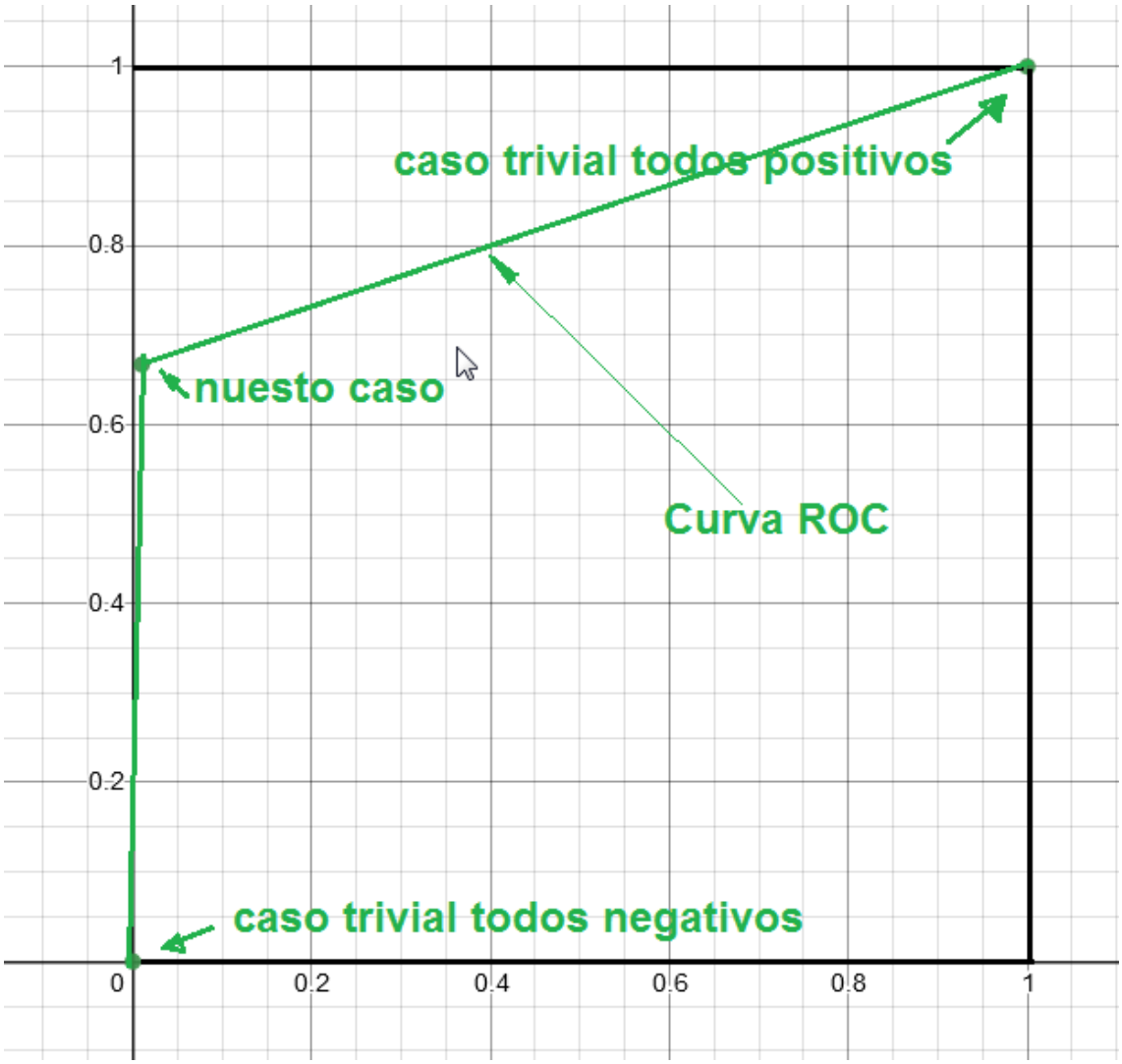
y en el eje de ordenadas los TPR (ratio de verdaderos positivos) o sensibilidad.

Sobre este plano podemos representar la actuación de un clasificador sobre la predicción de pertenencia a una clase, la de varios clasificadores, la de un clasificador sobre varias clases, etc.

En el plano ROC el punto (0,0) representa una clasificación con una ratio de falsos positivos de 0 (es decir una ratio de verdaderos negativos de 1) y una ratio de verdaderos positivos de 0.Es decir, una clasificación consistente en que todo se clasifica como negativo (no pertenece a la clase considerada).

Análogamente, el punto (1,1) Tiene una sensibilidad del 100% (detecta todos los casos positivos) y una especificidad del 0 % “no pilla ni un caso negativo. Porque los asigna todos a la clase considerada (todos los considera positivos.)

Una clasificación con TPR= 0,667 y FPR=0,01 como el resultado de nuestro clasificador al predecir los clientes con siniestros , vendrá dada por el punto (0.01,0.667)



El área baja la curva ROC nos dará una buena medida de la calidad del clasificador o de la clasificación. El ideal sería que detectara todos los positivos adecuadamente y sólo los positivos. En ese caso el punto del clasificador sería el (0,1) y el área bajo la curva sería de 1. El caso extremo pésimo sería un clasificador representado por el punto (1,0) en cuyo caso el área bajo la curva ROC sería 0.

En nuestro ejemplo es 0,928 tanto para la clase con siniestro como para la clase sin siniestro, ya que son complementarias.

Obviamente, lo dicho para el área de la curva ROC puede decirse análogamente para la curva precisión-exhaustividad de la última columna.

3.1. Métodos funcionales

Dentro de los métodos funcionales de clasificación y regresión podríamos incluir todos aquellos que generan una función de clasificación o predicción numérica y que, de alguna forma, no obtienen explícitamente un árbol, un conjunto de reglas o una distribución de probabilidad. De hecho, WEKA, por ejemplo considera como método funcional todos estos métodos incluidas las redes neuronales MLP.

Nosotros consideraremos separadamente los modelos neuronales y tampoco seremos demasiado exhaustivos en el estudio de todos los modelos. Pero sí diremos algo de los modelos de regresión y, especialmente del análisis discriminante.

3.1.1. Regresión. Regresión logística

La regresión es un conocido método estadístico de estimación/ predicción de una variable respuesta en función de otras variables explicativas.

El caso lineal, tanto simple como múltiple, es ya suficientemente conocido y no nos detendremos aquí en él. La inferencia estadística sobre el modelo lineal, tal como se estudia ampliamente en Econometría y otras disciplinas tampoco va a ser objeto de nuestro análisis y, al respecto, remitimos al lector a estas disciplinas.

Tampoco entraremos en el conjunto de métodos generales que caen bajo el epígrafe de “modelos lineales generalizados” (que incluyen la regresión lineal y la logística, entre otros muchos).Pero sí le prestaremos algo de atención a un caso particular de regresión que admite su uso tanto para la predicción cuantitativa de una variable numérica como para la clasificación en sentido estricto. Se trata de la regresión logística.

La regresión logística es uno de los modelos lineales generalizados (MLG) más extendidos en su uso y pretende predecir las probabilidades para variables de respuesta categóricas a partir de información de variables explicativas de diferente índole.

Por esta razón podemos considerar que esta variable (una probabilidad) a estimar/predecir es objeto de una predicción numérica, pero también , en la medida en que la clasificación a una u otra clase puede hacerse en función de la mayor probabilidad de pertenencia a la misma es, también, un método aplicable a la tarea de la clasificación.

Supongamos que disponemos de una variable binaria y que puede tomar los valores {0,1} con probabilidades p, (1-p) . Las observaciones de esta variables y_i podemos considerarlas como una variable aleatoria binomial de parámetros 1 y p_i . A partir de aquí y de las observaciones de un vector de variables explicativas X el modelo propone la estimación de los valores de p_i según un función:

$$p_i = \frac{1}{1+e^{-\beta' X_i}} = \frac{1}{1+e^{-(\beta_0+\beta_1 x_{1i}+\beta_2 x_{2i}+...+\beta_k x_{ki})}}$$

que también puede expresarse como:

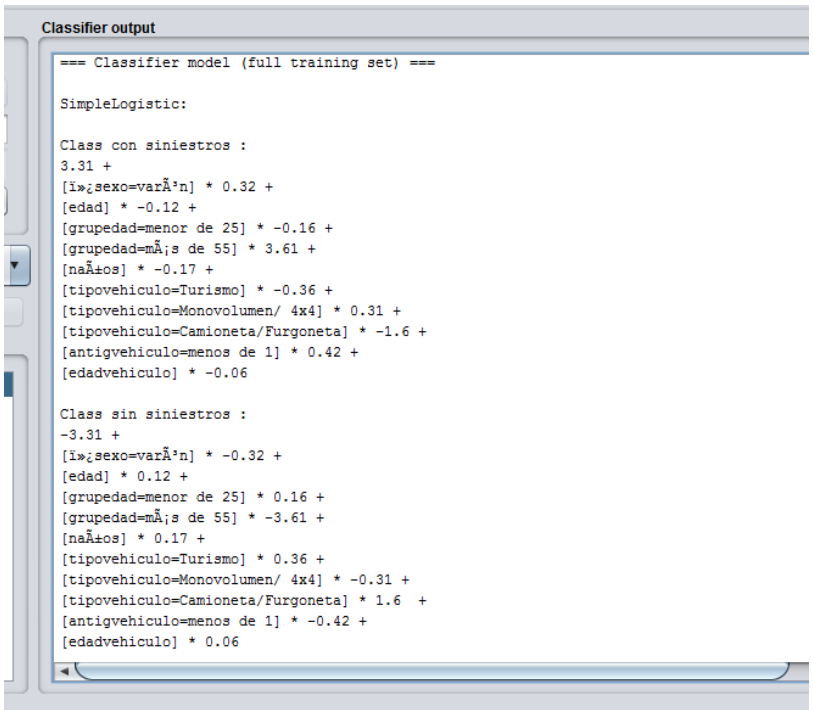
$$\text{logit}_i = \log \frac{p_i}{1-p_i} = \beta' X_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + ... + \beta_k x_{ki}$$

Los cocientes $p_i/(1-p_i)$ son conocidos como odds (las ratios típicas de las apuestas) y sus logaritmos (logit) se convierten en las nuevas variables a estimar en un modelo (ya) lineal (una vez transformado). A partir de aquí la regresión es conceptualmente y metodológicamente semejante al caso lineal : podríamos estimar los coeficientes β , predecir las Odds , y después las probabilidades y la clase de pertenencia y podríamos estudiar el comportamiento de los residuos, la calidad del ajuste, etc. .

A partir de la estimación de logit de la(s) clase(s) considerada para cierto individuo podemos estimar las probabilidad de pertenencia a esa clase como :

$$p_i = \frac{e^{\text{logit}_i}}{1+e^{\text{logit}_i}}$$ con lo cual obtenemos lo que se conoce como una clasificación suave (se llama así a un procedimiento que obtiene la probabilidad de pertenencia) , la clasificación final (fuerte) puede hacerse asignando a la clase en cuestión si su probabilidad de pertenencia es mayor o igual a 0,5 (mayor que la de no pertenencia)

En el ejemplo que vimos en la introducción vimos una aplicación de la regresión logística con WEKA . Analizamos la calidad de la clasificación obtenida. Pero antes de eso, en el análisis aparece la estimación del modelo que ahora mostramos.



La regresión logística es, en cualquier caso,un método estadístico estándar y puede llevarse a cabo con cualquier software estadístico de cierto nivel como SPSS, SAS minitab,R etc.

Reproducimos abajo un script de R (www.uv.es/mlejarza/datamine/logit.R) que llevaría a cabo el análisis sobre los datos del mismo ejemplo. Dos cosas a destacar: la regresión logística se hace como caso particular de modelo lineal generalizado. La evaluación de la clasificación se hace sobre el conjunto original de datos en su totalidad.

```
library(car)# para disponer de la función recode ( recodificación de variables)

logit <- read.table("~/datamining/datos/discricochesR10.csv",   header=TRUE, sep=",", na.strings="NA",
dec=".", strip.white=TRUE)
summary(logit)

# recodificamos siniUA como variable binaria numérica
# se añade la función paste para que se aplique la numerización a los valores  y no a los niveles

logit$siniUA <-as.numeric(paste( recode(logit$siniUA,   '"sin siniestros" = 0; "con siniestros" = 1; ')))

#convertimos el factor sexo en la variable binaria varon
logit$varon<- as.numeric(paste(recode(logit$ï..sexo, '"mujer" = 0;"varÃ³n"= 1')))

#convertimos las variables categóricas de más de dos nivels en variables binarias ( 1 por nivel)
logit$edad.menos25<- as.numeric(paste(recode(logit$grupedad, '"menor de 25" = 1;c("25-40","40-
55","mÃ¡s de 55")= 0')))
logit$edad.25.40<- as.numeric(paste(recode(logit$grupedad, '"25-40" = 1;c("menor de 25","40-55","mÃ¡s
de 55")= 0')))
logit$edad.40.55<- as.numeric(paste(recode(logit$grupedad, '"40-55" = 1;c("menor de 25","25-40","mÃ¡s
de 55")= 0')))
logit$edad.mas55<- as.numeric(paste(recode(logit$grupedad, '"mÃ¡s de 55" = 1;c("menor de 25","40-
55","25-40")= 0')))


logit$ant.vehi.menos1<- as.numeric(paste(recode(logit$antigvehiculo, '"menos de 1" = 1;c("1-4","5-
10","10+")= 0')))
logit$ant.vehi.1.4<- as.numeric(paste(recode(logit$antigvehiculo, '"1-4" = 1;c("menos de 1","5-
10","10+")= 0')))
logit$ant.vehi.5.10<- as.numeric(paste(recode(logit$antigvehiculo, '"5-10" = 1;c("menos de 1","1-
4","10+")= 0')))
logit$ant.vehi.mas.10<- as.numeric(paste(recode(logit$antigvehiculo, '"10+" = 1;c("menos de 1","1-4","5-
10")= 0')))

logit$camioneta<- as.numeric(paste(recode(logit$tipovehiculo, '"Camioneta/Furgoneta" =
1;c("Monovolumen/ 4x4","Motocicleta","Turismo")= 0')))
logit$monovolumen<- as.numeric(paste(recode(logit$tipovehiculo, '"Monovolumen/ 4x4" =
1;c("Camioneta/Furgoneta","Motocicleta","Turismo")= 0')))
logit$motocicleta<- as.numeric(paste(recode(logit$tipovehiculo, '"Motocicleta" =
1;c("Camioneta/Furgoneta","Monovolumen/ 4x4","Turismo")= 0')))
logit$turismo<- as.numeric(paste(recode(logit$tipovehiculo, '"Turismo" =
1;c("Camioneta/Furgoneta","Monovolumen/ 4x4","Motocicleta")= 0')))

summary(logit)
```

```
save (logit,file="C:/Users/ASUS/Documents/datamining/logit.RData")# guardamos los datos ya
procesados

############### logit como glm ( modelo lineal general) ##########
# estimamos un modelo logit como caso particular de glm ( modelo lineal general)
# glm con variable de respuesta binomial, función de enlace logística( por defecto)
modelo_logistico <- glm(siniUA ~ varon + edad + + edad.menos25 + edad.25.40 + edad.40.55 +
                    edad.mas55 + ant.vehi.menos1 + ant.vehi.1.4 +ant.vehi.5.10 +
                    ant.vehi.mas.10 + camioneta + monovolumen + motocicleta + turismo +
                    naÃ.os + edadvehiculo + antigcarnet + kilometros  ,
                  data = logit, family = "binomial")

summary(modelo_logistico)# resultados del modelo obtenido

prediccion <- predict(modelo_logistico, type="response")# obtiene las predicciones
modelo_logistico$fitted.values# también genera las predicciones

error<-prediccion-logit$siniUA# errores de predicción
plot(logit$siniUA,prediccion)
hist(error)
plot(density(error))

clase.predicha = floor (prediccion+0.5)# asignamos cada observación a una de las dos clases
confusion<-table (logit$siniUA,clase.predicha)#obtenemos la matriz de confusión sobre la totalidad de los
datos
confusion
diag(prop.table(confusion, 1))#  % de aciertos en la predicción  de cada clase ( todos los datos)
sum(diag(prop.table(confusion))) # % de aciertos
```

3.1.2. Análisis Discriminante

El análisis discriminante se propone la determinación de un criterio que nos permita decidir a qué grupo pertenece un cierto individuo, a partir de la información disponible sobre él cifrada en términos de los valores que toman ciertas variables consideradas. En esta perspectiva la discriminación se plantea como el problema de identificar (adscribir) un individuo anónimo como perteneciente a una clase o grupo

La situación de nuestro problema de decisión es la siguiente:

Tenemos un individuo, **w**, que puede pertenecer a una de varias posibles poblaciones H_1 ,H_2 ,…,H_r. Pero no sabemos a cuál de ellas pertenece de hecho.

- Disponemos de un conjunto de n variables X_1 ,X_2 ,..X_n.

- Nuestro individuo nos viene caracterizado por los valores que toman para él las n variables. En consecuencia, podemos caracterizarlo por el vector de observaciones:

$$X(w)=\begin{pmatrix} x_1(w) \\ x_2(w) \\ . \\ . \\ x_n(w) \end{pmatrix}$$

- Nos planteamos la obtención de una regla de decisión que nos permita asignar al individuo w a una de las r poblaciones o grupos.

  La ejecución de esta regla de decisión es lo que constituye la identificación o discriminación.

  La solución del problema suele acometerse construyendo ciertas funciones (funciones = f (X)), llamadas discriminantes, con la esperanza de que estas funciones nos definan sobre R^n una partición $\{ R_1,R_2 ..,R_r \}$,de modo que podamos adoptar el criterio de decisión de que si $X(w) \in R_i$, entonces $w \in H_i$.

  Básicamente hay tres posiciones, (que bajo ciertas condiciones, presentan resultados similares) de afrontar el problema de la discriminación y la determinación de las funciones discriminantes :

A. Apoyarse para la construcción de las funciones discriminantes en la distancia entre el individuo w y los distintos grupos.

B. Apoyarse en el criterio de minimizar la probabilidad de error de clasificación (con o sin información inicial) o maximizar la verosimilitud de pertenencia al grupo.

C. Plantear la identificación como un típico problema de decisión cuya solución pasa por la minimización de la función de pérdida asociada a la "mala clasificación " (o misclassification).

   Utilizando un criterio de discriminación geométrico, resulta razonable asignar el individuo anónimo a aquel grupo del que se encuentre más cercano. Lo habitual es considerar la distancia al centro de gravedad del grupo y utilizar una medida de la distancia que no esté sujeta ni a la escala de medida de las variables ni a su covariación o correlación , y, en este sentido, la distancia de Mahalanobis es la más adecuada.

   Así pues si consideramos la distancia entre el individuo w (representado por el vector de observaciones X) y el grupo H_i , como la distancia de Mahalanobis entre X y el centro de gravedad del grupo i-simo M_i :

$$D(X,H_i) = (X - M_i)' V^{-1} (X - M_i)$$

   El criterio de discriminación será asignar w al grupo H_i si:

$$D(X,H_i) = \min_s \{D(X,H_s)\}$$

   Es fácil obtener una función discriminante que discrimine entre los grupos H_i y H_j :

$$W_{ij}(X) = (M_i - M_j)' V^{-1} X - 1/2 (M_i - M_j)' V^{-1} (M_i + M_j)$$

Otro criterio de discriminación basado en la distribución de las variables en los distintos grupos será el que asigne el individuo anónimo a aquel grupo en el que se maximice la función de verosimilitud.

Según este criterio asignaremos el individuo w al grupo H_i si :

$L_i(X)= \max_s\{ L_s(X)\}$ (donde L es la función de verosimilitud).

En este caso la función discriminante para cada par de grupos H_i y H_j sería :

$V_{ij}(X) = \log L_i(X) - \log L_j(X)$

De forma que si para cierto i se cumple que:

$V_{ij}(X) > 0 \quad \forall\, i \neq j$ asignaremos el individuo anónimo al grupo H_i.

Si las poblaciones son normales multivariantes y las matrices de varianzas (poblacionales) coinciden, el criterio máximo verosímil y el geométrico coinciden y el discriminador máximo verosímil asume la misma expresión que el discriminador lineal de Wald-Anderson.

Sin embargo, en el caso en el que cada población tenga una matriz de varianzas propia el discriminador adopta una expresión cuadrática. De forma que para cada dos grupos la función discriminante queda como:

$$Q_{ij}(X) = \frac{1}{2}X'(V_j^{-1} - V_i^{-1})X + X'(V_j^{-1}M_i - V_i^{-1}M_j)X + \frac{1}{2}M_j'V_j^{-1}M_j + \frac{1}{2}M_i'V_i^{-1}M_i + \frac{1}{2}\log|V_j| + \frac{1}{2}\log|V_i|$$

Welch fue el primero en proponer la discriminación siguiendo un criterio de minimizar la probabilidad de error de clasificación o bien, maximizar la verosimilitud de pertenencia a un grupo. Estudió también, la posibilidad de disponer de conocimiento a priori acerca de la pertenencia de un individuo a cada grupo y el modo de utilizarlo en la discriminación a través del uso del Teorema de Bayes.

Con información inicial el discriminador asociado era, entonces:

$B_{ij}(X) = \log L_i(X) - \log L_j(X) + \log(q_i/q_j)$

donde q_i y q_j son las probabilidades iniciales de que w pertenezca a H_i y H_j , respectivamente. (Habitualmente las proporciones de individuos de cada grupo en la muestra)

Otros muchos autores han hecho hincapié en las distintas consecuencias que pueden tener distintos errores de clasificación. En muchas situaciones prácticas distintos errores de clasificación pueden tener asociados costes muy distintos. En ese caso, no es el error de clasificación lo que se debe minimizar sino el coste esperado o decantarse por aquella alternativa que nos minimice el máximo coste u optar por algún otro criterio decisional adecuado.

El análisis discriminantes es muy fácilmente ejecutable en el entorno WEKA dentro del grupo de clasificadores funcionales bajo el epígrafe LDA (linear discriminante

analysis). Hay que hacer notar que para poder utilizarlo es necesario que todos los atributos clasificadores sean numéricos. Esto conllevará que en el ejemplo que estamos siguiendo debamos prescindir de los atributos sexo, grupo de edad, tipo de vehículo, antigüedad de vehículo. Esto puede llevarse a cabo de forma sencilla eliminándolos en el panel de preproceso. (También podrían codificarse numéricamente pensando una adecuada codificación de apropiada interpretación, lo que , a menudo, no es sencillo)

Si optamos por todas las opciones por defecto nos muestra, como resultado del análisis, los centroides de las clases predichas pero no la función lineal discriminante. Y también nos da la evaluación de la clasificación (que por defecto es la de la validación cruzada con 10 estratos)

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         665               94.4602 %
Incorrectly Classified Instances        39                5.5398 %
Kappa statistic                          0.7222
Mean absolute error                      0.145
Root mean squared error                  0.2502
Relative absolute error                 61.3538 %
Root relative squared error             72.9071 %
Total Number of Instances              704

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,615    0,003    0,967      0,615    0,752      0,746    0,882     0,774     con siniestros
                 0,997    0,385    0,942      0,997    0,969      0,746    0,882     0,966     sin siniestros
Weighted Avg.    0,945    0,333    0,946      0,945    0,939      0,746    0,882     0,940

=== Confusion Matrix ===

   a   b   <-- classified as
  59  37 |   a = con siniestros
   2 606 |   b = sin siniestros
```

Ejercicio: ¿cómo sería el porcentaje de acierto si validamos la clasificación sobre la totalidad de la muestra? ¿ y si reservamos un tercio de los datos para el test (precentage Split 66%)?.

También en R (y en otros paquetes estadísticos se puede llevar a cabo un análisis discriminante. Si bien, la validación de la clasificación por defecto en R será la validación jacknife , esto es, “dejando uno fuera”.

El script para el mismo caso será: (www.uv.es/mlejarza/datamine/discriminanteDM.R)

```
discri <- read.table("~/datamining/datos/discricochesR10.csv",   header=TRUE, sep=",", na.strings="NA",
dec=".", strip.white=TRUE)
summary(discri)

# Análisis lineal discriminante con predicción Jacknife
library(MASS)
fit <- lda(siniUA ~ edad + naÃ.os + edadvehiculo + antigcarnet +kilometros, data=discri,
        na.action="na.omit", CV=TRUE)
#la opción CV=TRUE permite obtener las clases predichas y la probabilidades de pertenencia
#con la validación dejando uno fuera
fit # muestra los resultados
```

```
#matriz de confusión
ct <- table(discri$siniUA, fit$class)
ct
diag(prop.table(ct, 1))# porcentajes de  correctamente clasificados
sum(diag(prop.table(ct)))#porcentaje total de aciertos

#para poder representar el comportamiento de la(s) funcion(es) discriminante
# en el conjunto de datos es necesario no hacer la validación jacknife
dis<-lda(siniUA ~ edad + naÃ.os + edadvehiculo + antigcarnet +kilometros, data=discri,
       na.action="na.omit")
#al llamar a "dis" se obtine entre otras  cosas la f.l.discriminante (coeficientes)
# pero  esto no ocurre al llamar a fit
dis
plot(dis) # grafico de la función discriminante en los grupos(clases)
```

3.2. Métodos bayesianos

La modelización probabilística ha sido, desde el principio, uno de los esquemas más utilizados en inteligencia artificial para ser aplicado a problemas de decisión automática; como la ayuda al diagnóstico, por ejemplo. Si lo pensamos bien, la clasificación es una tarea que encaja bien en este esquema y no es extraño que también hayan sido aplicados a ella los métodos probabilísticos (bayesianos).
Utilizar estos métodos en un problema de clasificación supone proceder a una estimación de las probabilidades de pertenencia a una **clase** de un individuo dado, a partir de la información disponible sobre los atributos considerados.
Dos cuestiones se deducen de aquí:
1.- La solución del problema es la obtención de una o varias probabilidades condicionadas: Probabilidad de pertenencia a una clase, condicionada a que el individuo tiene tal y tal y tal y… y tal atributo
2.- La solución al problema es, en principio, una probabilidad y no una clasificación en sentido estricto. Esto es lo que se conoce como clasificación suave. También es este el caso de la regresión logística (y otros métodos). Pero, por un lado, ello no impide que se puede llevar a cabo una clasificación discriminante (asignación a una clase) y también presenta algunas ventajas. Lo primero es tan simple como asignar a la clase de máxima probabilidad a posteriori. Y entre las ventajas está el hecho de contar con una información más precisa que puede permitir tomar decisiones mejores. Veamos un ejemplo de esto:
Supongamos un problema de inversión en el que consideramos dos o más posibles compañías en las que hacerlo. El problema de clasificación puede expresarse en términos de rentabilidad de las compañías; digamos con una variable de clasificación con las clases rentable, no rentable. Resuelto el problema, supongamos que la solución probabilistica es P(A_rentable)=0.8 , P(A_no_rentable)=0.2 , P(B_rentable)=0.55, P(B_no_rentable)=0.45. La clasificación discriminante nos considera ambas compañías rentables, pero contar con la información adicional de las probabilidades de pertenencia nos permite posibilidades de actuación mejores, diversificando la inversión por ejemplo asignando más recursos a la inversión más probablemente rentable y menos a la menos rentable.

El método bayesiano más simple es la aplicación directa del teorema de bayes para calcular la probabilidad de que un individuo pertenezca a la clase C a partir de la información (condicionada) de que tiene los atributos A_1,A_2,A_3,…A_p :

$$P(C \mid A_1, A_2, ...A_P) = \frac{P(A_1, A_2, ...A_P \mid C).P(C)}{P(A_1, A_2, ...A_P)}$$

En la práctica hay que tener en cuenta que para cada atributo habrá distintos valores posibles y que para calcular la probabilidad condicionada del numerador hay que considerar la conjunción en cada caso de los valores que toma cada atributo. Se trata, en realidad, para cada caso individual de la probabilidad condicionada:

$$P(A_1, A_2, ...A_P \mid C) = P\left[(A_1 = a_1 * \cap A_2 = a_2 * \cap ... \cap A_p = a_p *) \mid C\right]$$

Cuya estimación, en general requeriría conocer una gran cantidad de probabilidades condicionadas.

Otro tanto ocurriría en el denominador con las probabilidades "no condicionadas" pero este problema se suele obviar en la metodología bayesiana, ya que el denominador acaba siendo un factor de normalización para que las probabilidades sumen 1 y no es necesario conocerlo pudiéndose trabajar sólo con los numeradores (sabiendo que la probabilidades será todas "proporcionales " a los numeradores)

3.2.1.Naïve Bayes

Una forma de solucionar el problema de la inflación de parámetros a estimar es considerar que todos los atributos se distribuyen independientemente unos de otros. De esta forma la probabilidad (condicionada) de la intersección será (supuesta la independencia) el producto de las probabilidades (condicionadas) de cada atributo:

$$P(A_1, A_2, ...A_P \mid C) = P\left[(A_1 = a_1 * \cap A_2 = a_2 * \cap ... \cap A_p = a_p *) \mid C\right] =$$

$$\text{supuesta la independencia} = P(A_1 = a_1 * \mid C) \cdot P(A_2 = a_2 * \mid C) \cdot ... \cdot P(A_p = a_p * \mid C)$$

De esta forma sólo hay que estimar las probabilidades de cada rasgo de cada atributo condicionadas a la pertenencia a una u otra clase y, partir de ahí la probabilidad de que un individuo pertenezca a una clase sabiendo que tiene esos rasgos de esos atributos será proporcional al producto de esas probabilidades (de cada rasgo condicionada a la clase) por la probabilidad marginal de pertenencia a la clase.

Y la asignación a una u otra clase se hará de acuerdo al criterio de máxima probabilidad a posteriori.(Nótese que sólo maximizamos el numerador de la expresión del T. de Bayes)

$$C_{MAP} = \underset{C \in \Omega_c}{\arg\max} \left\{ P(P(A_1, A_2, ...A_P \mid C).P(C) \right\} = \underset{C \in \Omega_c}{\arg\max} \left\{ P(C) \prod_{i=1}^{p} P(A_i \mid C) \right\}$$

El problema en el método Naïve Bayes se reduce a estimar la probabilidades condicionadas $P(A_i \mid C)$ (la probabilidad marginal de cada clase, P(C) , se estima por la frecuencia relativa marginal)

Si los atributos son discretos (cualitativos, nominales, ordinales o numéricos discretos) la estimación más sencilla es la máximo-verosimil, estimando las probabilidades condicionadas por sus frecuencias relativas en los datos:

$$P(A_i = a_i \mid C = c) = \frac{n(A_i = a_i, C = c)}{n(C = c)}$$: número de casos(a_i, c) dividido por casos c.

Estos estimadores son apropiados si el tamaño de la muestra es grande pero tienden a sobreajustarse a los datos. Para solución el posible sobreajuste suele utilizarse también el estimador basado en la sucesión de Laplace:

$$P(A_i = a_i \mid C = c) = \frac{n(A_i = a_i, C = c) + 1}{n(C = c) + \left|\Omega_{A_i}\right|}$$

siendo $\left|\Omega_{A_i}\right|$ el número de casos posibles del atributo A_i

Este estimador permite conceder alguna oportunidad a aquellos casos que no cuentan con ninguna representación en la base de datos (como puede verse en el ejemplo siguiente)

Por otro lado, si el atributo que consideremos es continuo podemos tomar alguna hipótesis sobre su distribución condicionada como que sigue una normal y que sus parámetros son las EMV correspondientes (u otras distribuciones si son más razonables).

Ejemplo:
Supongamos la siguiente pequeña base de datos formada por 6 observaciones de dos atributos (uno discretos, D, y otro numérico ,N,) y una clase a predecir, C

Individuo	D	N	C
1	a	5	si
2	a	2.2	no
3	a	1.8	no
4	b	4	si
5	b	2	si
6	a	3	no

Para aplicar un clasificador NaïveBayes necesitamos estimar la distribución de probabilidad marginal de C y las distribuciones de D y de N condicionadas a C.
Tenemos 3 **sí** y 3 **no** en la base de datos así que la distribución marginal de C es:

C	P(C
si	0.5
no	0.5

La distribución de D condicionada a cada valor de C será:

D	P(D|C=si)	P(D|C=no)		D	P(D|C=si)	P(D|C=no)
a	1/3	1		a	0,4	0,8
b	2/3	0		b	0,6	0,2
Estimación Máximo verosímil				Estimación sucesión de Laplace		

(Observamos como en el segundo caso las opciones de probabilidad no son tan "extremas")

Y la distribución de N condiconada a C supuesta Normalidad sería:

Distribución N | C=si	N(μ=3.67; σ=1,53)
Distribución N | C=no	N(μ=2.33; σ=0,61)

La clasificación de un individuo con D= b y N=3,5 quedaría como:

P(C| D=b,N=3.5) α P(C) .P(D=b|C).P(N=3,5|C)
(donde α simboliza la proporcionalidad)
Y que para cada posibilidad resultaría:

$P(C=si \mid D=b) \alpha P(C) .P(D=b|C=si).P(N=3,5|C=si)$
$P(C=no \mid D=b) \alpha P(C) .P(D=b|C=no).P(N=3,5|C=no)$

Lo que empleando la EMV nos daría:
$P(C=si \mid D=b) \alpha 0.5\times 2/3 \times f(3.5; N(\mu=3.67; \sigma=1,53)= 0.5\times0.67\times0.26=0.087$
$P(C=no \mid D=b) \alpha 0.5\times 0 \times f(3.5; N(\mu=2.33; \sigma=0,61))= 0.5\times0\times0.1=0$

Una vez normalizadas las probabilidades nos da una clasificación de :
C=sí con probabilidad 1 y C=no con probabilidad 0 (Es imposible clasificar un individuo con D=b como C=no porque no había ninguno en la base de datos, si usamos el EMV)

En cambio, usando la E.S.Laplace:
$P(C=si \mid D=b) \alpha 0.5\times 0.6 \times f(3.5; N(\mu=3.67; \sigma=1,53)= 0.5\times0.6\times0.26=0.078$
$P(C=no \mid D=b) \alpha 0.5\times 0.2 \times f(3.5; N(\mu=2.33; \sigma=0,61))= 0.5\times0.2\times0.1=0.01$
Que una vez normalizado (dividiendo por 0.088=0.078+0.01) nos clasifica:

C=sí con probabilidad 0.89 y C=no con probabilidad 0.11
La discriminación nos conduciría , en los dos casos a la opción C=si

A pesar de que el empleo del discriminador Naïve Bayes supone una gran simplificación al suponer los distintos atributos todos ellos independientes entre sí suele conducir a resultados sorprendentemente acertados en muchos casos de clasificación como discriminación (como clasificación fuerte). La razón es que la asignación a una clase se hace no tanto en función del valor de probabilidad estimado como de la máxima probabilidad a posteriori y, a menudo ocurrirá que aunque la simplificación de la independencia aleje las probabilidades obtenidas de las correctas, el orden entre ellas no suele alterarse demasiado.
Obviamente en situaciones en que los atributos estén muy asociados/correlacionados la redundancia de algunos rasgos podrá distorsionar los resultados de forma más drástica, por supuesto.

Entre los clasificadores disponibles en el entorno WEKA contamos con el clasificador Naïve Bayes .Empleado el clasificador al ejemplo que venimos considerando, (DiscricochesR10) nos arroja un porcentaje de acierto , ciertamente algo peor que otros métodos (logística, por ejemplo) pero no demasiado bajo y obtiene una ratio de verdaderos positivos para la clase “con siniestro” mejor que en la regresión logística. (Para la clase “sin siniestro” el comportamiento es peor, en cambio)

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         600               85.2273 %
Incorrectly Classified Instances       104               14.7727 %
Kappa statistic                          0.5069
Mean absolute error                      0.1539
Root mean squared error                  0.3526
Relative absolute error                 65.1102 %
Root relative squared error            102.7547 %
Total Number of Instances              704

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,781    0,137    0,475      0,781    0,591      0,530    0,903     0,791     con siniestros
                 0,863    0,219    0,962      0,863    0,910      0,530    0,903     0,968     sin siniestros
Weighted Avg.    0,852    0,208    0,895      0,852    0,866      0,530    0,903     0,944

=== Confusion Matrix ===

   a   b   <-- classified as
  75  21 |   a = con siniestros
  83 525 |   b = sin siniestros
```

El paquete de R { e1071} (también el {naivebayes}) permite también ejecutar este clasificador en R con bastante facilidad. Aquí se reproduce el script:

```
naive <- read.table("~/datamining/datos/discricochesR10.csv",   header=TRUE, sep=",", na.strings="NA", dec=".", strip.white=TRUE)
summary(naive)
library(e1071)
library(naivebayes)
nb<- naiveBayes(siniUA ~ ., data =naive)
nb
#predecimos la clase ( usando la totalidad de los datos de entrenamiento)
pred <- predict(nb, naive)
pred
#obtenemos la matriz d econfusión
tab <- table(naive$siniUA, pred, dnn = c("Actual", "Predicha"))
tab
#analizamos la matriz de confusion con detalle necesitamos el paquete {caret}
library(caret)
confusionMatrix(tab)
```

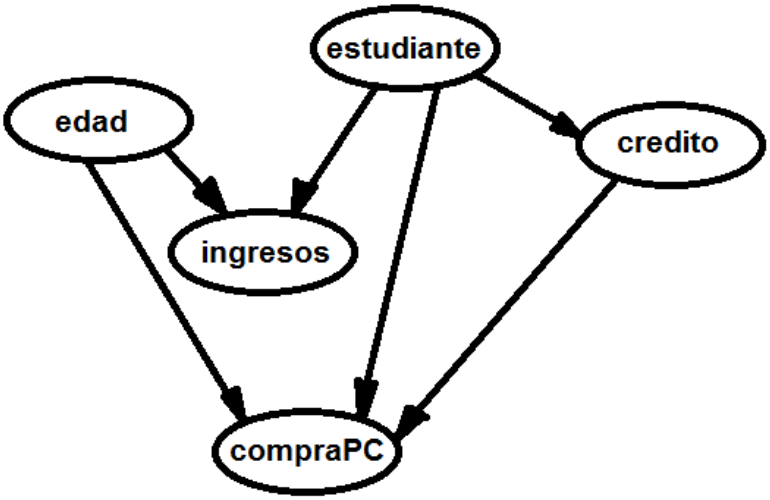
Sobre la totalidad de los datos arrojaría estos resultados:

```
                Predicha
Actual           con siniestros sin siniestros
  con siniestros             75             21
  sin siniestros             83            525

               Accuracy : 0.8523
                 95% CI : (0.8239, 0.8777)
    No Information Rate : 0.7756
    P-Value [Acc > NIR] : 2.12e-07

                  Kappa : 0.5069

 Mcnemar's Test P-Value : 2.21e-09

            Sensitivity : 0.4747
            Specificity : 0.9615
         Pos Pred Value : 0.7813
         Neg Pred Value : 0.8635
```

```
          Prevalence : 0.2244
      Detection Rate : 0.1065
Detection Prevalence : 0.1364
   Balanced Accuracy : 0.7181

    'Positive' Class : con siniestros
```

3.2.2.Redes bayesianas

Las redes bayesianas son un formalismo que ha demostrado una gran potencialidad en la representación del conocimiento (e incluso en la modelización del razonamiento causal) en distintos campos de la Inteligencia Artificial con notable éxito en situaciones de incertidumbre. También en la minería de datos tienen utilidad en tareas de clasificación y otras (clustering y asociación) .
Una red bayesiana en un grafo dirigido (u orientado) y acíclico que cuenta con variables en sus nodos y cuyos arcos representan relación entre las variables.
La dirección de los enlaces hace que entre dos nodos ligados el antecesor sea considerado “padre” del sucesor. Y el hecho de que no haya ciclos descarta la circularidad de las relaciones. Esta red bayesiana representa las relaciones entre las variables edad, estudiante, ingresos crédito y compraPc en un dominio de conocimiento que modeliza situaciones de compra de un PC por estudiantes.



Los arcos nos indican la existencia de cierta dependencia entre las variables: los ingresos, dependen directamente de la edad y del hecho de ser estudiante, la compra de un Pc depende de la edad, del hecho de ser estudiante y del nivel de crédito pero, según esta red, el nivel de crédito es independiente de la edad y la compra del PC no depende del nivel de ingresos.

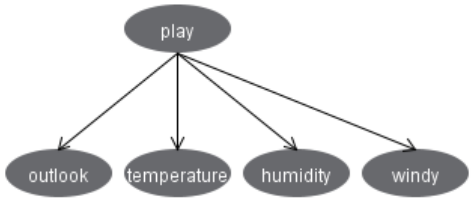
Así especificada la red sólo da cuenta de la relación de dependencia o independencia en términos cualitativos pero es posible y conveniente que la red dé cuenta también del “grado” de esta dependencia esta puede conseguirse introduciendo en los nodos las distribuciones de probabilidad de las variables padres condicionadas a la variable de cada nodo. (El nodo raíz que no tiene padres considera la distribución marginal).

La imagen de abajo muestra la representación de un conjunto de datos muy similar a otro visto en el T.1. Sobre las situaciones meteorológicas y el juego o no de un partido de tenis.

outlook	temperature	humidity	windy	play
sunny	'(79.75-inf)'	'(80.5-88.25]'	FALSE	no
sunny	'(79.75-inf)'	'(88.25-inf)'	TRUE	no
overcast	'(79.75-inf)'	'(80.5-88.25]'	FALSE	yes
rainy	'(69.25-74.5]'	'(88.25-inf)'	FALSE	yes
rainy	'(-inf-69.25]'	'(72.75-80.5]'	FALSE	yes
rainy	'(-inf-69.25]'	'(-inf-72.75]'	TRUE	no
overcast	'(-inf-69.25]'	'(-inf-72.75]'	TRUE	yes
sunny	'(69.25-74.5]'	'(88.25-inf)'	FALSE	no
sunny	'(-inf-69.25]'	'(-inf-72.75]'	FALSE	yes
rainy	'(74.5-79.75]'	'(72.75-80.5]'	FALSE	yes
sunny	'(74.5-79.75]'	'(-inf-72.75]'	TRUE	yes
overcast	'(69.25-74.5]'	'(88.25-inf)'	TRUE	yes
overcast	'(79.75-inf)'	'(72.75-80.5]'	FALSE	yes
rainy	'(69.25-74.5]'	'(88.25-inf)'	TRUE	no

Probability Distribution Table For play

yes	no
0,633	0,367



Probability Distribution Table For windy

play	TRUE	FALSE
yes	0,35	0,65
no	0,583	0,417

Probability Distribution Table For outlook

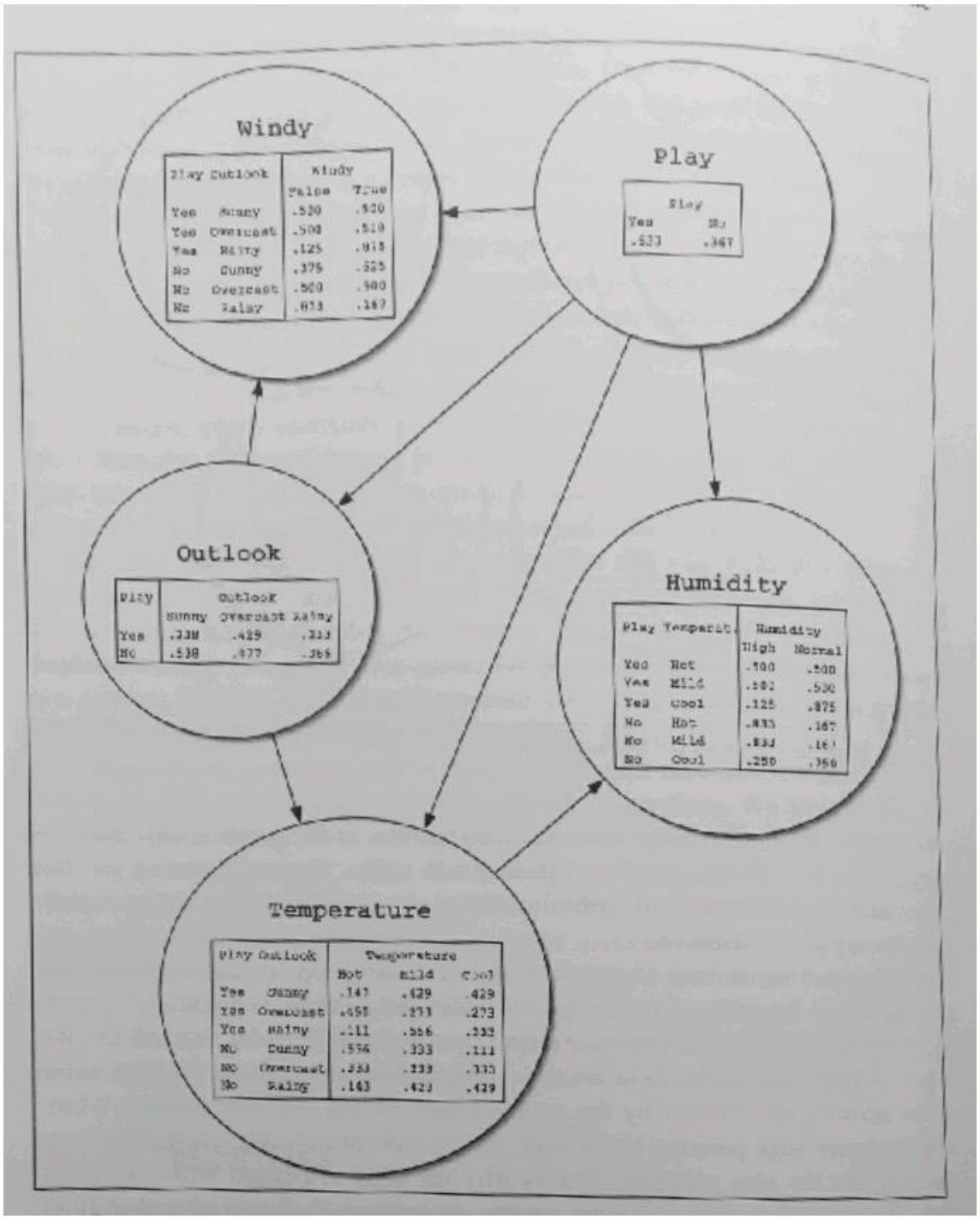
play	sunny	overcast	rainy
yes	0,238	0,429	0,333
no	0,538	0,077	0,385

Probability Distribution Table For humidity

play	'(-inf-72.75]'	'(72.75-80.5]'	'(80.5-88.25]'	'(88.25-inf)'
yes	0,318	0,318	0,136	0,227
no	0,214	0,071	0,214	0,5

Probability Distribution Table For temperature

play	'(-inf-69.25]'	'(69.25-74.5]'	'(74.5-79.75]'	'(79.75-inf)'
yes	0,318	0,227	0,227	0,227
no	0,214	0,357	0,071	0,357



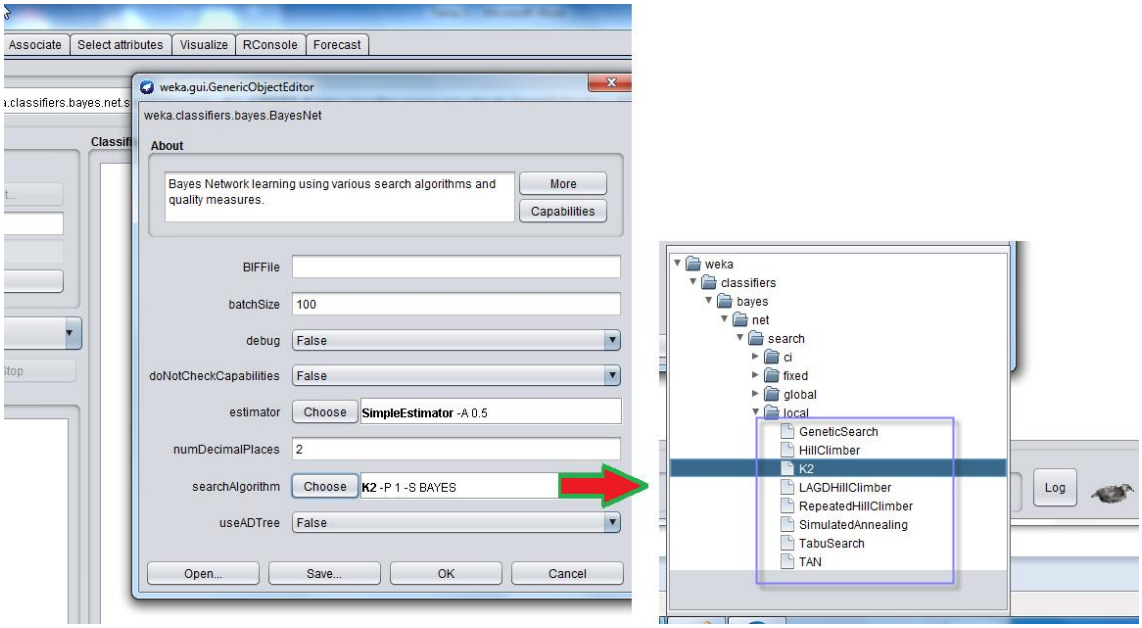
En esta figura tenemos de una red más compleja tres nodos (windy, temperature y humidity) que tienen dos padres . Como consecuencia la distribución condicionada de esos nodos tienen dos condicionantes (dos columnas en la izquierda que recorren todas las posibilidades condicionantes de las dos variables-padres(play×outlook o play×temperatures, según el caso).

Una red bayesiana puede ser un modelo de representación de una realidad compleja que, una vez construido, puede ser muy potente para el razonamiento probabilístico sobre ella. Pero el problema es, en realidad, la construcción de la red. Un experto en el dominio que pretendemos modelizar puede establecer el conjunto de relaciones de

dependencia entre las distintas variables pero en la mayoría de la situaciones tampoco esto va a poder solucionar el problema.

En definitiva dado un conjunto de datos D, necesitamos encontrar un grafo orientado acíclico G que represente lo mejor posible el conjunto de relaciones de dependencia e independencia presentes en los datos. La forma de medir la calidad de esta representación, desde el punto de bayesiano sería calcular las distribuciones a posteriori de la red y confrontarlas con los datos. Explorar todas las posibles redes bayesianas y evaluar cuál es la mejor es una estrategia imposible ya que el número de redes bayesianas posibles crece exponencialmente con las variables a considerar y por ello es habitual optar por estrategias de búsqueda heurística que proceden encontran redes “cada vez” mejores.

Existen varios algoritmos para estas búsquedas heurísticas. Entre los más utilizados están el algoritmo K2 , el algoritmo Hill Climber (HC) o el TAN Tree augmented naïve Bayes (que como su nombre indica parte de la consideración de independencia (naive) entre los atributos predictores y va paulatinamente mejorando la calidad de la estimación. La especificación concreta de estos algoritmos sobrepasa las pretensiones de este curso. Digamos sólo que, éstos y algunos otros están disponibles entre las opciones del clasificador “redes- bayesianas” del entorno WEKA



Utilizando el algoritmo TAN y las opciones por defecto para clasificar nuestros conocidos datos de discricochesR10 obtenemos en la validación cruzada con 10 estratos resultados muy similares a los obtenidos con la clasificación logística:

```
Correctly Classified Instances         668               94.8864 %
Incorrectly Classified Instances        36                5.1136 %
Kappa statistic                          0.762
Mean absolute error                      0.0595
Root mean squared error                  0.2062
Relative absolute error                 25.1701 %
Root relative squared error             60.0974 %
Total Number of Instances              704

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,708    0,013    0,895      0,708    0,791      0,769    0,952     0,874     con siniestros
               0,987    0,292    0,955      0,987    0,971      0,769    0,952     0,987     sin siniestros
Weighted Avg.  0,949    0,254    0,947      0,949    0,946      0,769    0,952     0,972

=== Confusion Matrix ===

   a   b   <-- classified as
  68  28 |   a = con siniestros
   8 600 |   b = sin siniestros
```

3.3. Árboles

Posiblemente los modelos basados en árboles sean los de más fácil utilización y comprensión. Lo que unido a que son capaces de ofrecer buenos resultados en las tareas de clasificación los hacen altamente atractivos. Un árbol (de decisión) es un conjunto de condiciones dispuestas en una estructura jerárquica organizada de tal manera que la decisión final a tomar pueda determinarse siguiendo las condiciones según se van cumpliendo desde la raíz del árbol hasta la hoja final de llegada.
En las tareas de clasificación suele asumirse que las clases (o etiquetas) son disjuntas por ello cada partición del árbol de decisión debe ser también disjunta. Cómo sean están particiones (criterio de tipos de partición) y cómo se vayan seleccionando (criterio de selección) son las dos claves que conforman los distintos algoritmos de generación de árboles.

Para los atributos (predictores) nominales los tipos de partición pueden ser o bien una rama para cada posible valor, o bien sólo admitir particiones binarias ($x_i=v_k$, $x_i \neq v_k$) . En realidad aunque el atributo pueda tomar más de dos posibles valores a través de sucesivas particiones binarias podemos dar cuenta de todas las posibilidades de forma equivalente
Para los atributos numéricos el criterio suele ser la partición en dos intervalos disjuntos ($x_i \leq C$, $x_i>C$). La elección de la constante C suele hacerse de forma que maximice algún criterio de discriminación.

Cómo se vayan seleccionando las distintas particiones, cómo se vaya secuenciando (en qué momento del proceso) y qué punto de corte consideren constituyen lo que serían los criterios de selección que caracterizan cada algoritmo.

La idea subyacente es que cada partición seleccionada en un determinado momento del proceso maximice algún criterio de discriminación: error (de clasificación) esperado , criterio de Gini, criterio Gain (ganacia de entropía) y distintas variantes de estos.

Todos ellos buscan la partición ***s*** con el menor **índice de impureza *I*(*s*)** (un indicador de la impureza de las ramas obtenidas; entendiendo por impureza apartarse de la situación ideal “todas las instancias de la rama son de la misma clase”):

$$I(s) = \sum_{j=1,2...,n} p_j . f(p_j^1 \cdot p_j^2 \cdot ... \cdot p_j^c)$$

donde:

n es el número de nodos hijos de s (ramas de la partición s)

p_j es la probabilidad de caer en el nodo hijo j (cumplir la condición)

$p_j^1, p_j^2, ..., p_j^c$ son las proporciones de elementos de las clases, 1,2,...,c que caen en el nodo j

y $f()$ es una cierta función que mide la impureza que en cada caso difiere segun la tabla:

Criterio	**Función f()**
ERROR ESPERADO	$\min(p_j^1, p_j^2, ..., p_j^c)$
GINI (CART)	$1 - \sum \left(p_j^i \right)^2$
Entropía (gain)	$\sum \left(p_j^i \right) . \log(p_j^i)$
DKM (Kerns y Mansour)	$2\left(\prod \left(p_j^i \right) \right)^{1/2}$

Hay algunas variantes de estos criterios para algunos algoritmos : el gain ratio del algoritmo C4.5, la ortogonalidad o criterios basados en el área bajo la curva ROC. Los últimos estudios experimentales apuntan a que, en general, los criterios Gain ratio (C4.5, o su versión para Java ,J48) y el criterio DKM tienden a dar mejores resultados que el de GINI.

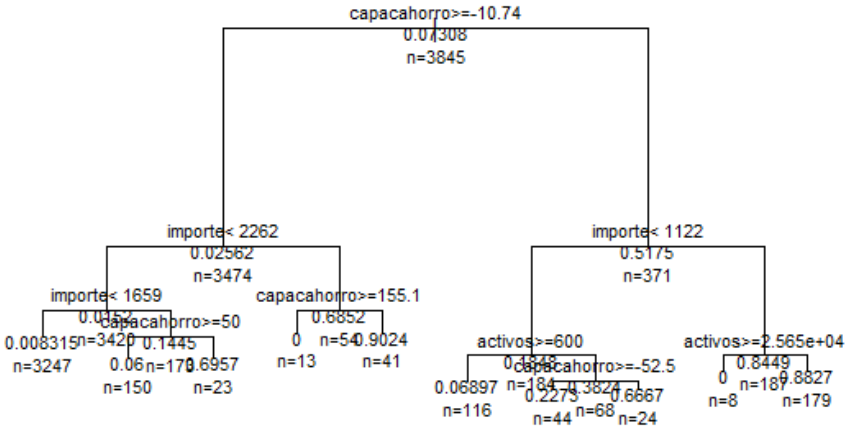
Otra cuestión importante en la construcción de modelos de árboles es la “poda” de ramas para evitar la excesiva **expresividad** de un árbol que puede conllevar un problema de sobreajuste.(Que sea capaz de reproducir a la perfección los datos pero falle en la generalización a nuevos ejemplares) Algunos algoritmos (como el C4.5) utilizan un proceso de prepoda según la cardinalidad de las ramas que si no superan un cierto número de elementos no se abren, otros realizan una pospoda tras la obtención del árbol completo utilizando distintos criterios de parada del proceso.

Entre los algoritmos de construcción de árboles más utilizados y fácilmente disponibles en paquetes de uso corriente nos encontramos con:
En spss: CHAID (Chi-Square Automatic Interaction Detector),QUEST (Quick Unbiased Efficient Statistical Tree) y CART
En R hay varios pero el menos complejo de aplicar es CART (función rpart)
En weka hay muchísimos más algoritmos entre los que destaca el J48 (que es una versión java del C4.5)
Consideramos para la aplicación práctica de algunos de estos modelos, la base de datos en creditscore.sav (https://www.uv.es/mlejarza/actuariales/tam/creditscore.sav) que recoge información sobre clientes de un banco que han solicitado un préstamo y éstos están clasificados entre los buenos clientes que han sido capaces de devolverlo en tiempo y forma y los que no. Obviamente queremos obtener un adecuado

discriminador para ayudar en la decisión de conceder o no un crédito a un futuro candidato.
Utilizamos, en primer lugar un árbol CART (dos en realidad uno de regresión y otro de clasificación) mediante el paquete {rpart} de R.
(www.uv.es/mlejarza/datamine/credittree.R)

```
require(ggplot2)
require(rpart)

library(haven)
creditscore <- read_sav("https://www.uv.es/mlejarza/actuariales/tam/creditscore.sav")
View(creditscore)
names(creditscore)

#Arbol de decisión con todas las variables
# continuaa y categoricas)
# primero actúa como arbol de regresión y predice la variable EstatusPrest considerada como numérica
ct = rpart(EstatusPrest ~ ., data=creditscore)
# resultado
ct
#EXPLICACIÓN
# "node), split, n, deviance, yval "
# node): indica el número de  nodo
# split: indica el criterio de bifurcacion  (split)
# n: indica el  numero de individuos en el grupo
# deviance= devianza
# yval: indica el valor predecido
# (yprob): indica la  probabilidad de pertenecer a cada clase
n= 3845

node), split, n, deviance, yval
      * denotes terminal node

 1) root 3845 260.464000 0.073081920
   2) capacahorro>=-10.74 3474  86.719920 0.025618880
     4) importe< 2262.5 3420  51.209360 0.015204680
       8) importe< 1659 3247  26.775490 0.008315368 *
       9) importe>=1659 173  21.387280 0.144508700
        18) capacahorro>=50 150   8.460000 0.060000000 *
        19) capacahorro< 50 23   4.869565 0.695652200 *
     5) importe>=2262.5 54  11.648150 0.685185200
      10) capacahorro>=155.0825 13   0.000000 0.000000000 *
      11) capacahorro< 155.0825 41   3.609756 0.902439000 *
   3) capacahorro< -10.74 371  92.636120 0.517520200
     6) importe< 1122.5 184  27.717390 0.184782600
      12) activos>=600 116   7.448276 0.068965520 *
      13) activos< 600 68  16.058820 0.382352900
        26) capacahorro>=-52.5 44   7.727273 0.227272700 *
        27) capacahorro< -52.5 24   5.333333 0.666666700 *
     7) importe>=1122.5 187  24.502670 0.844919800
      14) activos>=25650 8   0.000000 0.000000000 *
      15) activos< 25650 179  18.536310 0.882681600 *

# GRAFICO
plot(ct, margin=0.05, compress=TRUE, main="ÁRBOL DE DECISIÓN")
text(ct, use.n=TRUE, pretty=1, all=TRUE, cex=0.7)
```

ÁRBOL DE DECISIÓN



post(ct, filename = "credit.ps") #guarda el arbol en un archivo post-script

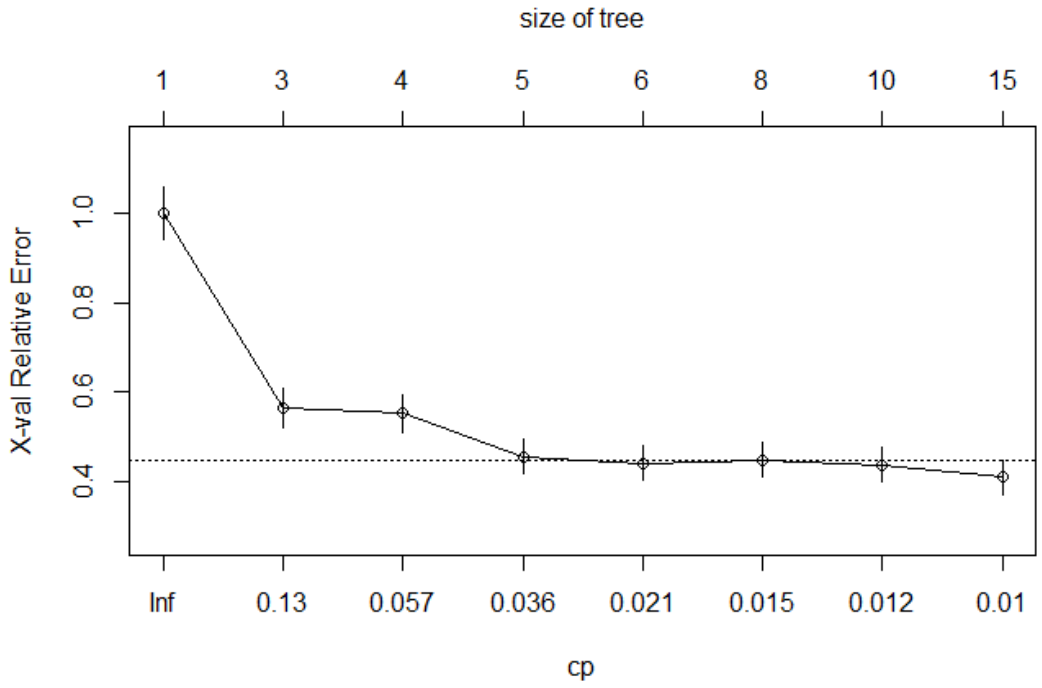
```
# convertimos el árbol de regresión en uno de clasificación
#convirtiendo  la variable EstatusPrest en FACTOR.
creditscore <- within(creditscore, {
  EstatusPrest <- factor(EstatusPrest, labels=c('bueno','malo'))
})

ct = rpart(EstatusPrest ~ ., data=creditscore)
ct
#EXPLICACIÓN
# "node), split, n, loss, yval, (yprob)"
# node): indica el número de  nodo
# split: indica el criterio de bifurcacion  (split)
# n: indica el  numero de individuos en el grupo
# loss: indica el numero de individuoss malclasifcados
# yval: indica el valor predecido
# (yprob): indica la  probabilidad de pertenecer a cada clase
n= 3845

node), split, n, loss, yval, (yprob)
      * denotes terminal node

  1) root 3845 281 bueno (0.926918075 0.073081925)
    2) capacahorro>=-10.74 3474  89 bueno (0.974381117 0.025618883)
      4) importe< 2262.5 3420  52 bueno (0.984795322 0.015204678)
        8) importe< 1659 3247  27 bueno (0.991684632 0.008315368) *
        9) importe>=1659 173  25 bueno (0.855491329 0.144508671)
         18) capacahorro>=50 150   9 bueno (0.940000000 0.060000000) *
         19) capacahorro< 50 23   7 malo (0.304347826 0.695652174) *
      5) importe>=2262.5 54  17 malo (0.314814815 0.685185185)
       10) capacahorro>=155.0825 13   0 bueno (1.000000000 0.000000000
) *
       11) capacahorro< 155.0825 41   4 malo (0.097560976 0.902439024)
*
    3) capacahorro< -10.74 371 179 malo (0.482479784 0.517520216)
      6) importe< 1122.5 184  34 bueno (0.815217391 0.184782609)
       12) activos>=600 116   8 bueno (0.931034483 0.068965517) *
       13) activos< 600 68  26 bueno (0.617647059 0.382352941)
         26) capacahorro>=-52.5 44  10 bueno (0.772727273 0.227272727)
```

```
           52) tipotrabajo< 3.5 37   5 bueno (0.864864865 0.135135135)
*
           53) tipotrabajo>=3.5 7   2 malo (0.285714286 0.714285714) *
         27) capacahorro< -52.5 24   8 malo (0.333333333 0.666666667)
           54) importe< 630 9   3 bueno (0.666666667 0.333333333) *
           55) importe>=630 15   2 malo (0.133333333 0.866666667) *
      7) importe>=1122.5 187  29 malo (0.155080214 0.844919786)
       14) activos>=25650 8   0 bueno (1.000000000 0.000000000) *
       15) activos< 25650 179  21 malo (0.117318436 0.882681564)
         30) importe< 1575 86  19 malo (0.220930233 0.779069767)
           60) activos>=5500 24  11 bueno (0.541666667 0.458333333)
            120) financiado< 86.69681 13   2 bueno (0.846153846 0.1538
46154) *
            121) financiado>=86.69681 11   2 malo (0.181818182 0.81818
1818) *
           61) activos< 5500 62   6 malo (0.096774194 0.903225806) *
         31) importe>=1575 93   2 malo (0.021505376 0.978494624) *
```

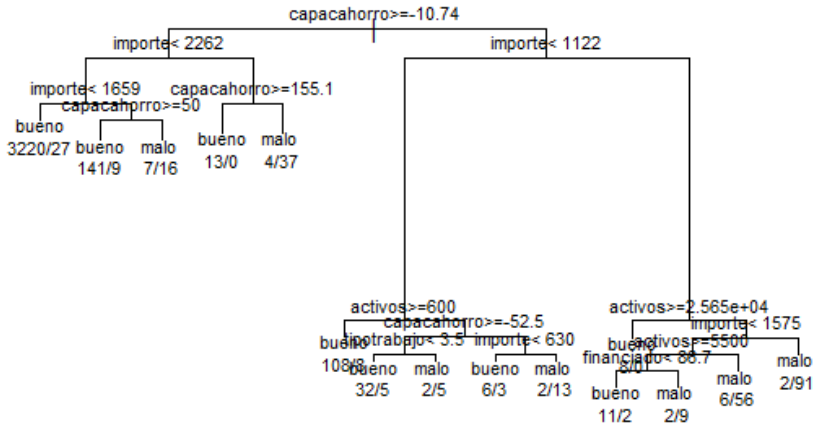
ct$cptable #tabla de reducción de error con el tamaño del árbol de utlidad en la PODA

```
     CP nsplit rel error    xerror       xstd
1 0.22953737      0 1.0000000 1.0000000 0.05743380
2 0.07117438      2 0.5409253 0.5658363 0.04393612
3 0.04626335      3 0.4697509 0.5516014 0.04340348
4 0.02846975      4 0.4234875 0.4555160 0.03958647
5 0.01601423      5 0.3950178 0.4412811 0.03898398
6 0.01423488      7 0.3629893 0.4483986 0.03928655
7 0.01067616      9 0.3345196 0.4377224 0.03883169
8 0.01000000     14 0.2811388 0.4092527 0.03758797
```

plotcp(ct) # gráfico DE UTILIDAD EN LA PODA



# GRAFICO
plot(ct, margin=0.05, compress=TRUE, main="ÁRBOL DE DECISIÓN")
text(ct, use.n=TRUE, pretty=0, all=FALSE, cex=0.7)
post(ct, filename = "credittree.ps") #guarda el arbol en un archivo post-script
summary(ct)




```
summary(ct)
```

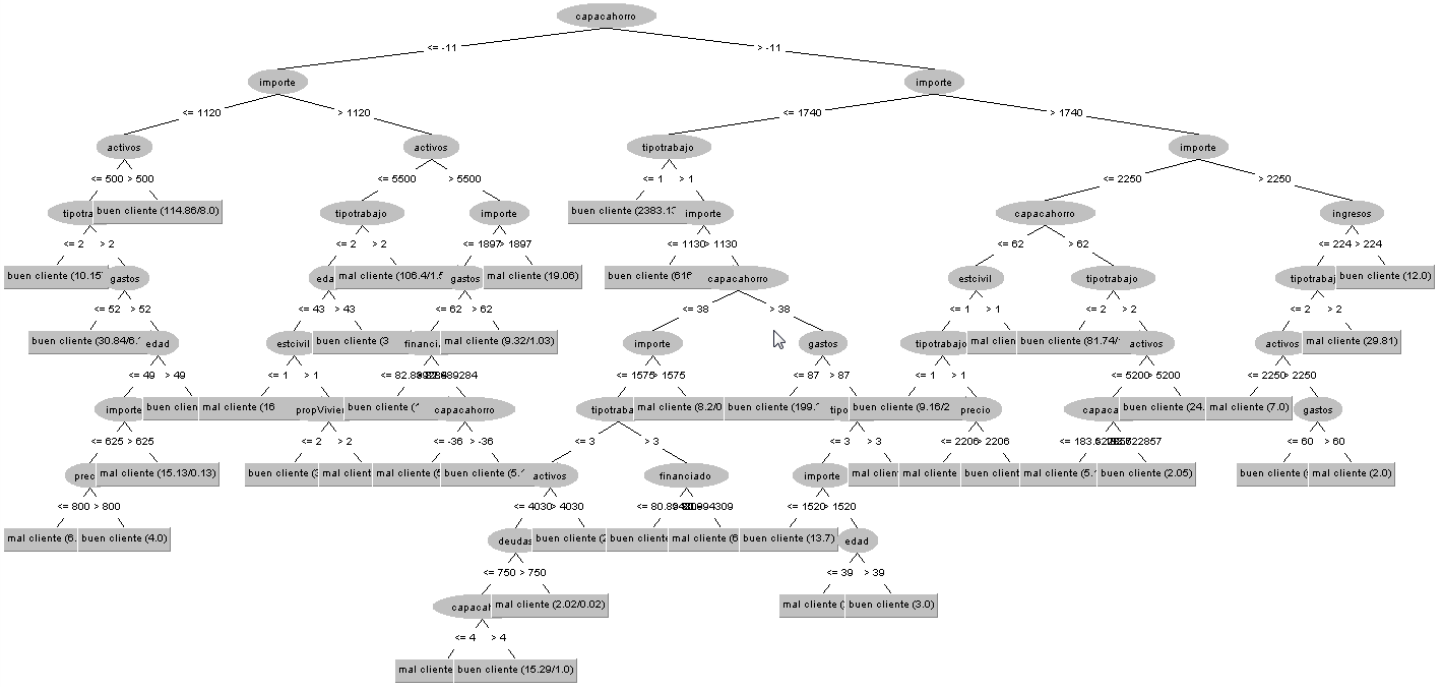
(La salida de la función summary es muy extensa y no la reproducimos aquí)

```
pred=predict(ct,type="class")

mconfusion <-table(creditscore$EstatusPrest,pred)
mconfusion # matriz de confusión
pred
        bueno malo
  bueno  3539   25
  malo     54  227
diag(prop.table(mconfusion, 1)) #  proporción Verdaderos positivos
bueno      malo
0.9929854 0.8078292
# total percent correct
sum(diag(prop.table(mconfusion))) # acuracidad (global)
[1] 0.9794538
```

Veamos, ahora el algoritmo J48 de WEKA sobre los mismos datos aplicado a la clasificación de los clientes entre buenos y malos.

Antes convertimos el archivo en un formato aceptable para WEKA (Excel, por ejemplo) colocamos la variable de clase al final de la base de datos y aplicamos en el prepoceso los filtros NumericToNominal para el último atributo y el filtro RenameNominalValues con las opciones “-R last -N "0: buen cliente, 1: mal cliente"; todo ello para optar con una variable de clase con esos dos valores.

Tras ello, el árbol obtenido (con las opciones por defecto) se muestra en formato texto como lista de bifurcaciones ordenadas y también de forma gráfica tal como aparece abajo:


capacahorro
<= -11
> -11
importe
importe
<= 1120
> 1120
<= 1740
> 1740
activos
activos
tipotrabajo
importe
<= 500
> 500
<= 5500
> 5500
<= 1
> 1
<= 2250
> 2250
buen cliente (114.86/8.0)
tipotrabajo
importe
capacahorro
ingresos
<= 2
> 2
<= 1897
> 1897
<= 1130
> 1130
<= 62
> 62
<= 224
> 224
buen cliente (10.15
gastos
mal cliente (19.06)
capacahorro
estcivil
tipotrabajo
tipotrabaj
buen cliente (12.0)
<= 52
> 52
<= 43
> 43
<= 62
> 62
<= 38
> 38
<= 1
> 1
<= 2
> 2
<= 2
> 2
buen cliente (30.84/6.
edad
estcivil
financi.
mal cliente (9.32/1.03)
importe
gastos
tipotrabajo
activos
activos
mal cliente (29.81)
<= 49
> 49
<= 1575
> 1575
<= 87
> 87
<= 1
> 1
<= 5200
> 5200
<= 2250
> 2250
importe
mal cliente (7.0)
gastos
<= 625
> 625
<= 2
> 2
<= -36
> -36
<= 3
> 3
<= 3
> 3
<= 2206
> 2206
<= 60
> 60
prec
mal cliente (15.13/0.13)
activos
financiado
importe
<= 800
> 800
<= 4030
> 4030
<= 1520
> 1520
mal cliente (6.
buen cliente (4.0)
deudas
edad
<= 750
> 750
<= 39
> 39
capacat
mal cliente (2.02/0.02)
<= 4
> 4
mal cliente
buen cliente (15.29/1.0)
buen cliente (3.0)
mal cliente (2.0)

También se muestran los habituales indicadores de calidad , en este caso, para la validación cruzada de 10 capas:

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        3739               97.2432 %
Incorrectly Classified Instances       106                2.7568 %
Kappa statistic                          0.7924
Mean absolute error                      0.0332
Root mean squared error                  0.1586
Relative absolute error                 24.4403 %
Root relative squared error             60.9477 %
Total Number of Instances             3845

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,987    0,210    0,984      0,987    0,985      0,793    0,912     0,983     buen cliente
               0,790    0,013    0,825      0,790    0,807      0,793    0,912     0,741     mal cliente
Weighted Avg.  0,972    0,196    0,972      0,972    0,972      0,793    0,912     0,965

=== Confusion Matrix ===

    a    b   <-- classified as
 3517   47 |    a = buen cliente
   59  222 |    b = mal cliente
```

Si utilizamos para validar el modelo los mismo datos de entrenamiento (que es lo que hicimos en el árbol CART) podemos ver que el comportamiento del modelo J48 es mejor que el CART :

```
Correctly Classified Instances        3816               99.2458 %
Incorrectly Classified Instances        29                0.7542 %
Kappa statistic                          0.9423
Mean absolute error                      0.0145
Root mean squared error                  0.0829
Relative absolute error                 10.7171 %
Root relative squared error             31.8634 %
Total Number of Instances             3845

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,999    0,089    0,993      0,999    0,996      0,943    0,994     0,999     buen cliente
               0,911    0,001    0,985      0,911    0,946      0,943    0,994     0,965     mal cliente
Weighted Avg.  0,992    0,083    0,992      0,992    0,992      0,943    0,994     0,997

=== Confusion Matrix ===

    a    b   <-- classified as
 3560    4 |    a = buen cliente
   25  256 |    b = mal cliente
```

3.4. Métodos basados en reglas

Obviamente, a partir de un árbol de decisión (de hecho lo hemos visto en alguna de las salidas de resultados anteriores) puede obtenerse un conjunto de reglas en el que cada bifurcación del árbol se corresponde con una regla del tipo “si condición, entonces tal y en otro caso, entonces cuál”. El conjunto de reglas obtenido de un árbol de decisión se basa en bifurcaciones donde cada condición apela también a las condiciones complementarias y va formando una ruta única de aplicación de reglas donde no puede ser aplicada, a la vez, nada más que una.

Sin embargo existen otros procedimientos de generación de reglas que no tienen por qué proceder de esta manera, varias reglas podrían aplicarse en un cierto momento y, tampoco tienen por qué barrer exhaustivamente todas las posibilidades. De forma,

que, aunque los árboles generan reglas, aquí nos referimos a estos otros procedimientos que generan reglas que no cumplen estas particularidades. Son los sistemas de aprendizaje por reglas de cobertura (covering).
En estos sistemas se busca una condición cuyo cumplimiento sea cubierto (de ahí el nombre) por el mayor número posible de instancias de una de las clases (que se toma como patrón) y de una u otra forma se reitera el procedimiento hasta construir una adecuada clasificación.
Hay varios algoritmos distintos de reglas de cobertura, algunos de los cuales pueden utilizarse desde la opción Rules del panel de clasificación de WEKA, que también incluye el procedimiento oneR (aplica sólo una regla sobre la variable más discriminante) y el trivial ZeroR que aplica la regla de clasificar cualquier individuo como de la clase más frecuente (acierta en un porcentaje de veces igual a esta frecuencia).
Reproducimos los resultados obtenidos (en WEKA) con algunos procedimientos basados en reglas aplicados al problema del creditscore.

```
Scheme:       weka.classifiers.rules.JRip -F 3 -N 2.0 -O 2 -S
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

JRIP rules:
===========

*(capacahorro <= -11) and (importe >= 1250) and (activos <= 5000) and
(tipotrabajo >= 2) => EstatusPrest=1 (101.0/0.0)
*(capacahorro <= 2) and (importe >= 1145) and (gastos >= 75) =>
EstatusPrest=1 (48.0/6.0)
*(tipotrabajo >= 3) and (importe >= 2000) and (ingresos <= 220) =>
EstatusPrest=1 (50.0/5.0)
*(ingresos <= 50) and (importe >= 760) and (antiguedad <= 2) and
(activos <= 2800) and (edad >= 33) => EstatusPrest=1 (18.0/1.0)
*(capacahorro <= 24.597213) and (importe >= 1600) => EstatusPrest=1
(24.0/11.0)
*(financiado >= 86.666667) and (capacahorro <= -17.0648) and (gastos
>= 60) => EstatusPrest=1 (16.0/3.0)
*(tipotrabajo >= 4) and (activos <= 2000) and (capacahorro <= 0) =>
EstatusPrest=1 (10.0/3.0)
*(importe >= 1750) and (ingresos <= 157) and (activos <= 2500) and
(financiado <= 87.587588) => EstatusPrest=1 (8.0/0.0)
*(tipotrabajo >= 3) and (importe >= 1220) and (ingresos <= 105) and
(edad <= 32) and (activos <= 4000) => EstatusPrest=1 (8.0/0.0)
*(tipotrabajo >= 2) and (financiado >= 88.372093) and (capacahorro <=
65) and (importe >= 1250) and (ingresos >= 92) => EstatusPrest=1
(7.0/1.0)
 => EstatusPrest=0 (3555.0/21.0)(REGLA POR DEFECTO)

Number of Rules : 11
Time taken to build model: 0.69 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances        3733               97.0871 %
Incorrectly Classified Instances       112                2.9129 %
Kappa statistic                          0.7918
Mean absolute error                      0.0376
Root mean squared error                  0.1594
Relative absolute error                 27.73   %
Root relative squared error             61.252  %
Total Number of Instances             3845
```

```
=== Detailed Accuracy By Class ===
TPRate  FPRate Precision Recall  F-Measure MCC ROCArea  PRCArea  Class
0,981  0,164    0,987    0,981   0,984    0,792  0,910   0,986     0
0,836  0,019    0,781    0,836   0,808    0,792  0,910   0,728     1
0,971  0,153    0,972    0,971   0,971    0,792  0,910   0,967     Avg.

=== Confusion Matrix ===

    a    b   <-- classified as
 3498   66 |    a = 0
   46  235 |    b = 1
```

```
Scheme: weka.classifiers.rules.DecisionTable -X 1 -S
"weka.attributeSelection.BestFirst -D 1 -N 5"
Test mode:    10-fold cross-validation
=== Classifier model (full training set) ===
Decision Table:

Number of training instances: 3845
Number of Rules : 75
Non matches covered by Majority class.
      Best first.
      Start set: no attributes
      Search direction: forward
      Stale search after 5 node expansions
      Total number of subsets evaluated: 92
      Merit of best subset found:   96.853
Evaluation (for feature selection): CV (leave one out)
Feature set: 11,12,14,15

Time taken to build model: 0.48 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        3707               96.4109 %
Incorrectly Classified Instances       138                3.5891 %
Kappa statistic                          0.703
Mean absolute error                      0.059
Root mean squared error                  0.1658
Relative absolute error                 43.4445 %
Root relative squared error             63.706  %
Total Number of Instances             3845

=== Detailed Accuracy By Class ===

TPRate FPRate Precision Recall F-Measure  MCC  ROCArea  PRCArea  Class
0,990  0,363   0,972     0,990  0,981    0,710  0,966   0,996     0
0,637  0,010   0,833     0,637  0,722    0,710  0,966   0,814     1
0,964  0,337   0,962     0,964  0,962    0,710  0,966   0,983     avg.

=== Confusion Matrix ===
    a    b   <-- classified as
 3528   36 |    a = 0
  102  179 |    b = 1
```

```
Scheme:       weka.classifiers.rules.PART -M 2 -C 0.25 -Q 1
Test mode:    10-fold cross-validation
=== Classifier model (full training set) ===
PART decision list
------------------
```

```
capacahorro > -11 AND importe <= 1740 AND tipotrabajo <= 1: 0
(2383.13/1.0)

capacahorro > -11 AND precio <= 2800 AND importe <= 1210 AND
capacahorro > 2: 0 (645.15/1.0)

capacahorro > 66 AND importe <= 2600 AND gastos <= 87 AND
tipotrabajo <= 2: 0 (102.98)

importe > 1900 AND capacahorro <= 161 AND tipotrabajo > 1 AND
capacahorro <= 63: 1 (84.41)

capacahorro > 154.28: 0 (80.97/0.08)

importe <= 1120 AND tipotrabajo > 2 AND edad > 49: 0 (50.26)

capacahorro <= -62 AND activos <= 14000 AND financiado > 42.042042 AND
tiempoPres > 42: 1 (39.77/0.77)

importe <= 750 AND tipotrabajo <= 3 AND capacahorro > -60: 0 (61.12)

importe > 2150 AND edad > 30 AND gastos > 53: 1 (17.52)

activos > 3850 AND gastos <= 86 AND financiado <= 83.202512 AND
estcivil <= 2 AND deudas <= 1172: 0 (97.31/1.0)

capacahorro > 38 AND importe <= 1420: 0 (31.96)

activos > 6200 AND ingresos > 105: 0 (18.62)

tipotrabajo > 2 AND importe > 1210 AND activos <= 5500 AND
capacahorro <= -0.72: 1 (44.39/0.66)

tipotrabajo > 3 AND propVivienda <= 1: 1 (14.15/1.15)

tiempoPres > 42 AND tipotrabajo <= 3 AND edad > 43 AND
deudas <= 1100: 0 (21.42/0.26)

capacahorro <= 89 AND financiado > 86.538462 AND edad > 29 AND
activos <= 4300: 1 (22.19/0.19)

capacahorro <= 89 AND tipotrabajo <= 2 AND importe <= 1575: 0
(23.41/1.0)

capacahorro <= 89 AND antiguedad <= 2 AND gastos <= 82 AND
tiempoPres > 18 AND importe > 950 AND antiguedad <= 0: 1 (11.41/1.41)

capacahorro <= 89 AND capacahorro <= 38 AND ingresos <= 88 AND
importe > 650 AND gastos <= 82 AND deudas <= 96 AND gastos > 55 AND
ingresos <= 51: 1 (13.12/1.12)

capacahorro <= 89 AND edad > 22 AND importe <= 1550 AND gastos <= 82
AND importe <= 950: 0 (20.47)

capacahorro <= 89 AND capacahorro <= 38 AND  ingresos <= 88 AND
antiguedad > 0 AND propVivienda > 1 AND antiguedad > 2 AND
precio > 1179: 0 (10.29)

activos <= 18500 AND propVivienda > 1: 1 (24.2/2.2)
```

```
propVivienda <= 1 AND tiempoPres <= 54 AND importe > 1050: 1
(10.59/0.59)

estcivil <= 1 AND propVivienda <= 1 AND edad <= 40: 0 (6.08)

estcivil > 1: 0 (4.04)

propVivienda <= 1: 1 (4.0/1.0)

: 0 (2.01)

Number of Rules  :      27

Time taken to build model: 0.27 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        3751               97.5553 %
Incorrectly Classified Instances        94                2.4447 %
Kappa statistic                          0.8178
Mean absolute error                      0.0275
Root mean squared error                  0.1517
Relative absolute error                 20.2982 %
Root relative squared error             58.283  %
Total Number of Instances             3845

=== Detailed Accuracy By Class ===

TPRate FPRate Precision  Recall F-Measure  MCC  ROCArea PRCArea  Class
0,988  0,178  0,986      0,988  0,987    0,818  0,930   0,989     0
0,822  0,012  0,840      0,822  0,831    0,818  0,930   0,770     1
0,976  0,166  0,975      0,976  0,975    0,818  0,930   0,973

=== Confusion Matrix ===

    a    b   <-- classified as
 3520   44 |    a = 0
   50  231 |    b = 1
```

Dejamos como ejercicio la aplicación del método oneR y ZeroR así como el método M5 que es de aplicación en la predicción numérica (regresión) y por ello requerirá que la variable `EstatusPrest` vuelva a ser definida como numérica

3.5. Métodos neuronales

La redes neuronales son uno de los paradigmas fundamentales en el ámbito de la inteligencia artificial casi desde sus comienzos y muy pronto demostraron tener un amplio campo de aplicabilidad en el procesos de aprendizaje de patrones a partir de datos, es decir en la minería de datos. Sus dos procedimientos fundamentales de aprendizaje (supervisado y no supervisado) vienen a corresponderse con métodos de aplicación respectiva a dos tareas específicas de la minería de datos:
Aprendizaje supervisado→tareas de clasificación y regresión
Aprendizaje no supervisado→tareas de clustering

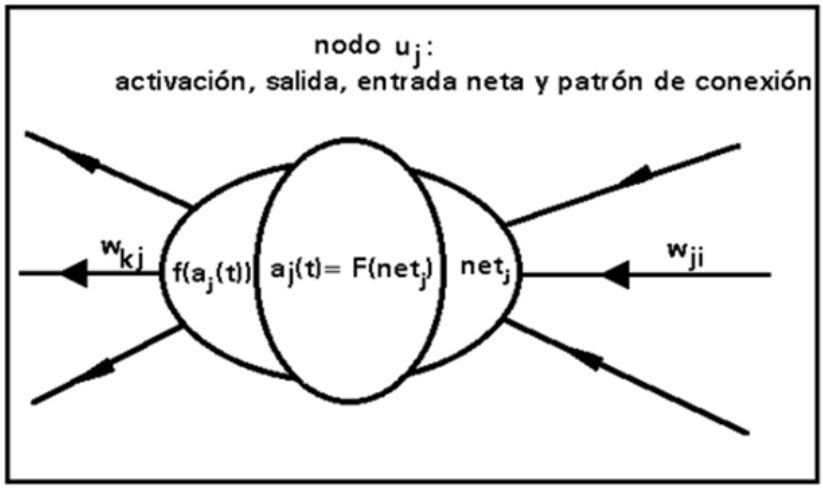
Una red neuronal es un sistema de computación que consta de un gran número de elementos simples, muy interconectados que procesan la información respondiendo

dinámicamente frente a los estímulos externos y son susceptibles de modificar su comportamiento, aprendiendo.
Entre sus principales características están:
1) Emulan el comportamiento biológico de la cognición: proceso, representación y memoria distribuidos
2) Su principal cualidad es su **adaptabilidad dinámica** (aprendizaje, generalización y autoorganización)
3) Su potencialidad es el principio "Sistémico": "El todo es más que la suma de las partes": cada unidad o neurona no procesa más que señales (intensidades eléctricas; números reales, si se quiere), la red es capaz de procesar esquemas complejos y ensayar hipótesis, representar y discriminar patrones, ajustar modelos, etc. de forma análoga a como lo hacen los cerebros biológicos.
4) Paralelismo masivo en el proceso
5) Memoria asociativa distribuida.
6)Tolerancia a fallos : el proceso en paralelo implica cierta redundancia en la representación, lo que permite que el sistema siga funcionando aunque se degrade parcialmente.(Igual ocurre con los cerebros biológicos).
7) Tratamiento simultaneo de grandes cantidades de información.
8) Reconstrucción de datos parciales: clausura de datos de entrada.
9) Capacidad de aprendizaje

Una red neuronal se caracteriza por:

- *Un conjunto de unidades de procesamiento* $W = \{u_1, u_2, u_3, \ldots u_n\}$.
- *Un estado de activación* que en cada instante t , representa el nivel de activación de cada neurona uj a través de un valor de activación real $a_j(t)$.El patrón de activación global será, un vector n-dimensional a(t).
- *Unas salidas de las unidades: función de salida* o *de transferencia* f. $o_j(t) = f(a_j(t))$ (el output es función de la activación de a_j
- *Un patrón de conexión* representado por una matriz de **pesos** o **ponderaciones sinápticas** , W, en la que cada w_{ji} representa la intensidad y sentido con el que la salida de la neurona i-ésima afecta a la activación de la neurona j-ésima.
- *Una regla de propagación*. Forma de combinar las salidas de todas las neuronas para incidir como entrada de las demás: $net_j = S\, w_{ji}\, o_i(t) = S\, w_{ji}\, f(a_i(t))$.
- *La regla o función de activación* será una función ,F, que a partir del estado actual de activación de una neurona y de su entrada neta, determinada por el patrón de conexión y por la regla de propagación, nos evalúe el estado de activación subsiguiente:
  $$a_j(t+1) = F(a_j(t), net_j(t)) = F(a_j(t), S\, w_{ji}\, f(a_i(t)))$$
- *Una regla de aprendizaje :* en función de las salidas de la red y de los errores podrá modificar los pesos
- *Una representación del ambiente*: unidades o capas de entrada y salida

La activación de la neurona j en el momento t+1 depende , a través de la función de activación, F, de su activación en t y de la entrada neta que recibe de las otras neuronas. La entrada neta que recibe depende de las salidas de las demás neuronas $f(a_i)$ y del patrón de conexión (ponderaciones)



La funciones de activación y salida no tienen por qué coincidir pero suelen hacerlo.
Para representar entradas y salidas de rango continuo se usan activaciones continuas.
Función sigmoide (logística)

El aprendizaje de una red neuronal consiste en la modificación de la estructura (conectividad) de la red, a través del reajuste de los pesos de las conexiones w_{ij}. (nuevas conexiones, pérdida de conexiones existentes, modificación de los pesos).

Hay que distinguir entre el entrenamiento y el aprendizaje: el entrenamiento es el **proceso (externo)** a la red por el que se van introduciendo pares de Entradas/ Salidas-deseadas (o sólo las entradas en el aprendizaje no supervisado) para que la red reajuste sus pesos y el aprendizaje es el **proceso interno** que se consigue con el entrenamiento y que consiste en la alteración de la matriz de pesos y su adecuación (el método , en definitiva, por el que se reajustan los pesos según se procesan los datos de entrenamiento)

Como hemos comentado, las redes de aprendizaje supervisado son las que tienen aplicación en las tareas de clasificación y/o predicción numérica. En especial la más utilizada de ellas, que puede emplearse tanto en la clasificación como en la predicción de una variable numérica, es la conocida como Perceptrón Multicapa, MLP, de sus siglas en inglés Multi-Layer Perceptron.

El perceptrón multicapa es un red que, además de contar con una capa de entrada y otra de salida cuenta con una o varias capas intermedias (u ocultas), De ahí su nombre. En la fase de funcionamiento la red se alimenta hacia adelante, desde la capa de entrada que procesa la información relativa a las variables de entrada (variables predictoras/atributos de clasificación o regresión). Cada variable es procesada en cada neurona. Las señales recorren la red, activando las neuronas de las capas ocultas según el patrón de activación “aprendido” y llegan a la capa de salida que ofrece la activación (evalúa el valor predicho) de la(s) variable(s) de salida la(s) variable(s) a predecir.

La clave del funcionamiento correcto de la red está en la fase previa de aprendizaje que es **supervisada**. Se suministran a la red pares de entradas y salidas (deseadas)

procedentes de la base de datos. Y se procede a ir actualizando, modificando los pesos hasta que se obtengan unas salidas con un mínimo error.

El proceso de actualización de los pesos se basa en la aplicación de la llamada **regla delta generalizada** que actualiza los pesos proporcionalmente al error (delta) detectado. La actualización de estos pesos se realiza de delante hacia atrás (a partir del error detectado en la salida) retropropagando (a veces a este tipo de redes se le llama, por eso, Back-propagation networks) el error desde la capa de salida a las internas y corrigiendo los pesos en el proceso, el proceso se va repitiendo hasta alcanzar una mínima cota de error aceptable.

Las redes MLP funcionan bastante bien como clasificadores y predictores si se les suministra un importante conjunto de entrenamiento y son capaces de generalizar relaciones altamente no lineales entre las entradas y las salidas. Pero, entre sus defectos, están las altas necesidades de memoria de cálculo y la muy difícil interpretación del patrón obtenido.

En R, en SPSS, en WEKA y en otras muchas aplicaciones podemos contar con paquetes y rutinas que nos permiten aplicar una red MLP tanto a problemas de clasificación como de predicción.

Utilizaremos el archivo salus.sav, que contiene información de un conjunto de supuestos clientes de un seguro de salud que hacen referencia a características de filiación, de salud, y de uso de los servicios médicos (http://www.uv.es/mlejarza/actuariales/tam/salus.sav)

Usando R : (reproducimos el script)
(http://www.uv.es/mlejarza/actuariales/tam/neuralnet.R)

```
library(haven)
salus <- read_sav("salus.sav")
View(salus)

# LIBRERIAS Y DATOS
# ------------------------------------------------------
library(MASS); library(neuralnet); library(ggplot2)
set.seed(65)
datos    <- salus
n        <- nrow(datos)
muestra  <- sample(n, n * .70)
train    <- datos[muestra, ]
test     <- datos[-muestra, ]

# NORMALIZACION DE VARIABLES
# ------------------------------------------------------
maxs     <- apply(train, 2, max)
mins     <- apply(train, 2, min)
datos_nrm <- as.data.frame(scale(datos, center = mins, scale = maxs - mins))
train_nrm <- datos_nrm[muestra, ]
test_nrm  <- datos_nrm[-muestra, ]

# FORMULA
# ------------------------------------------------------
nms  <- names(train_nrm)
frml <- as.formula("tipoasegurado ~sexo + Edad + talla + peso + imcorporal + glucemia +
                    colesterol + tensMax + tensMin + enfercronica + deporte +
                    tabaco + alcohol" )
```

```
nms;frml

#MODELO
# ------------------------------------------------------
modelo.nn <- neuralnet(frml,
                  data         = train_nrm,
                  hidden       = c(7,5), # dos capas de 7 y 5 neuronas
                  threshold    = 0.05,   # umbralas  :las iteraciones se detendran cuando el "Cambio" del error sea menor a 5% entre una iteracion de optimizacion y otra. Este "Cambio" es calculado como la derivada parcial de la funcion de error respecto a los pesos.
                  algorithm    = "rprop+" ) # retropropagación mejorada
summary(modelo.nn)
# PREDICCION
# ------------------------------------------------------
#primero eliminamos de la base de datos test_las variables que no están en
# el modelo
test_nrm1 <- within(test_nrm, {
  cirugiMay <- NULL
  Cirugimen <- NULL
  coste <- NULL
  IndRiesgsalud <- NULL
  NacAmb <- NULL
  NacESpecial <- NULL
  NacMT <- NULL
  urgencias <- NULL
})
pr.nn   <- compute(modelo.nn,within(test_nrm1,rm(tipoasegurado)))
pr.nn

# se transoforma el valor escalar al valor nominal original
tipoasegurado.predict <- pr.nn$net.result*(max(datos$tipoasegurado)-
min(datos$tipoasegurado))+min(datos$tipoasegurado)
tipoasegurado.real    <- (test_nrm$tipoasegurado)*(max(datos$tipoasegurado)-
min(datos$tipoasegurado))+min(datos$tipoasegurado)
error<-tipoasegurado.real-tipoasegurado.predict

#  ERROR CUADRATICO medio
# ------------------------------------------------------
(se.nn <- sum((tipoasegurado.real - tipoasegurado.predict)^2)/nrow(test_nrm))
summary(error)
plot(error)
hist(error)

#GRAFICOS
# ------------------------------------------------------
#predicción y errores
qplot(x=tipoasegurado.real, y=tipoasegurado.predict, geom=c("point","smooth"), method="lm",
     main=paste("Real Vs Prediccion.  Error Cuadratico medio=", round(se.nn,2)))
# Red
plot(modelo.nn)


#############################################################################################
###########  PREDICCIÓN DEL COSTE SANITARIO   ####################
#FÓRMULA
```

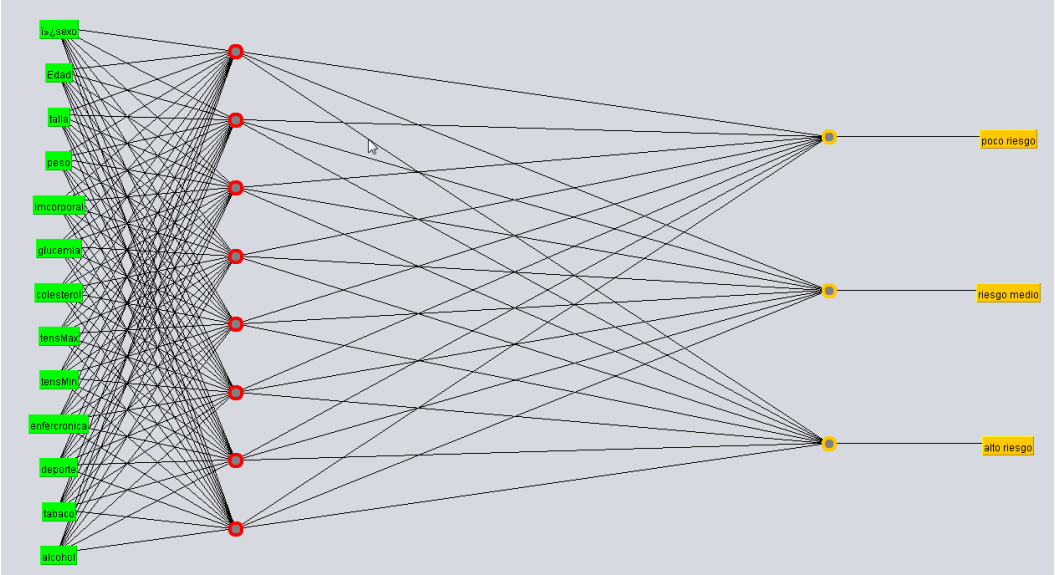
```
frml2 <- as.formula("coste ~sexo + Edad + talla + peso + imcorporal + glucemia +
                 colesterol + tensMax + tensMin + enfercronica + deporte +
                tabaco + alcohol" )
frml2
#MODELO
# ------------------------------------------------------
modelo.nn2 <- neuralnet(frml2,
                  data         = train_nrm,
                  hidden       = c(8,6), # dos capas de 7 y 5 neuronas
                  threshold    = 0.05,   # umbralas  :las iteraciones se detendran cuando el "Cambio" del error
sea menor a 5% entre una iteracion de optimizacion y otra. Este "Cambio" es calculado como la derivada
parcial de la funcion de error respecto a los pesos.
                  algorithm    = "rprop+" ) # retropropagación mejorada

summary(modelo.nn2)
# PREDICCION
# ------------------------------------------------------
#primero eliminamos de la base de datos test_las variables que no están en
# el modelo
test_nrm2 <- within(test_nrm, {
  cirugiMay <- NULL
  Cirugimen <- NULL
  tipoasegurado <- NULL
  IndRiesgsalud <- NULL
  NacAmb <- NULL
  NacESpecial <- NULL
  NacMT <- NULL
  urgencias <- NULL
})

pr.nn2   <- compute(modelo.nn2,within(test_nrm2,rm(coste)))
pr.nn2
# se transforma el valor escalar al valor nominal original
coste.predict <- pr.nn2$net.result*(max(datos$coste)-min(datos$coste))+min(datos$coste)
coste.real    <- (test_nrm$coste)*(max(datos$coste)-min(datos$coste))+min(datos$coste)
error2<-coste.real-coste.predict
# error cuadrático medio y gráficos de errore
(se.nn2 <- sum((error2)^2)/nrow(test_nrm))
summary(error2)
plot(error2)
hist(error2)

#GRAFICOS
# ------------------------------------------------------
#predicción  y errores
qplot(x=coste.real, y=coste.predict, geom=c("point","smooth"), method="lm",
    main=paste("Real Vs Prediccion.  Error Cuadratico medio=", round(se.nn2,2)))
# Red
plot(modelo.nn2)

#########################################################################33
```

Para tratar el archivo salus con WEKA, antes del proceso, eliminamos la variable IndRiesgsalud, de esta forma, además, la variable nominal tipoasegurado se quedará la última y weka la considerará, por defecto, la variable de clasificación.

Eliminamos, igualmente, todas las variables que tienen que ver con los servicios médicos recibidos por cada cliente y la variable coste (ya que estas están directamente vinculadas a la clasificación de los clientes según riesgo).

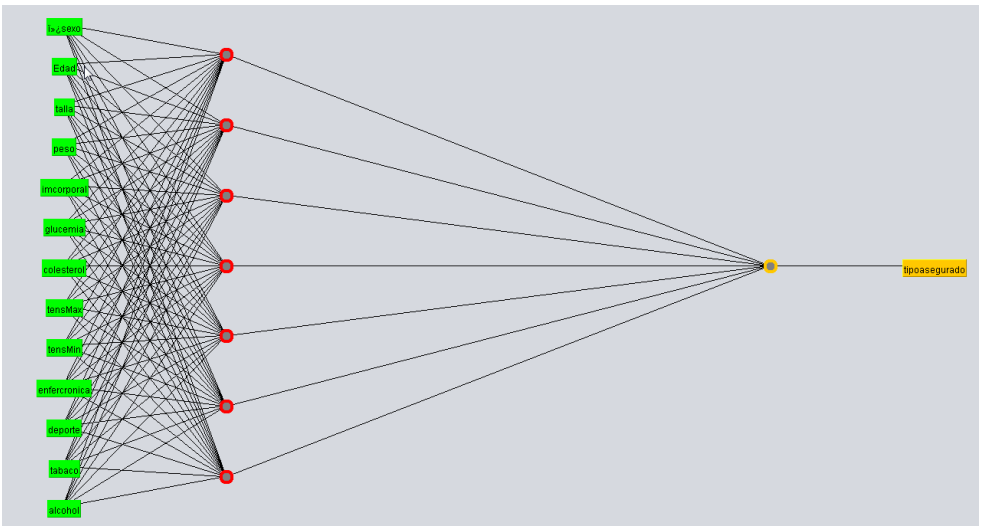
También renombramos los valores con el filtro correspondiente y sus opciones : weka.filters.unsupervised.attribute.RenameNominalValues -R last -N "0:poco riesgo,1:riesgo medio,2:alto riesgo".

Procediendo así, al ser la variable de clasificación de carácter nominal, la aplicación de la red la binariza convirtiendo las tres clases en tres unidades de salida binarias. Esto, no sólo complica la topología de la red, sino que también añade complejidad al proceso de entrenamiento (muchos más parámetros a calibrar). La red tarda más de 10 en minutos en entrenarse.



Si modificamos el carácter nominal de la variable y lo consideramos numérico, sólo habrá una neurona de salida y el proceso se hará más rápido. Esto puede hacerse editando el archivo arff y modificando la línea

@attribute tipoasegurado {0,1,2} por @attribute tipoasegurado numeric



Con todo, el proceso sigue siendo de muy larga ejecución.

3.6. Predictores

A lo largo del tema hemos tratado al alimón las tareas de clasificación y regresión (o predicción) y ello nos ha llevado a considerar que la obtención de un clasificador y la de un predictor son semejantes. Esto no es exactamente así. Hemos visto, aunque quizá de una forma no sistemática que hay algunas herramientas que pueden usarse para ambas tareas porque admiten que la variable de salida sea o bien nominal o bien numérica, también hemos visto que algunas otras herramientas sólo admiten un uso y no los dos, o que algunas herramientas necesitan técnicas ligeramente diferentes para un caso u otro.

Con todo un aspecto importante a considerar es que en la medida en que la predicción es una tarea bien diferente a la clasificación la evaluación de los predictores ha de ser forzosamente distinta. Mientras la matriz de confusión, los diferentes errores de clasificación, y la evaluación de éstos y sus costes asociados son muy relevantes en la evaluación de un buen clasificador, en el caso de un predictor la custión es bien diferente.

Aquí el tratamiento estadístico y visual de los residuos, y el promedio numérico de los errores, el análisis de la varianza (o la devianza) explicada y no explicada y otros aspectos semejantes cobre importancia capital.

Buena parte de estas cuestiones forman parte del desarrollo tradicional de otras disciplinas (Econometría, predicción con datos transversales, predicción con datos temporales) y , en consecuencia, no nos vamos a detener aquí en su explicación exhaustiva.