Universidad de Valladolid

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERÍAS INDUSTRIALES

Grado en Ingeniería en Organización Industrial

MÉTODOS DE REGRESIÓN Y CLASIFICACIÓN BASADOS EN ÁRBOLES

Autor:

Fernández Villafañez, Sergio

Tutor:

García Escudero, Luis Ángel

Departamento de Estadística e Investigación Operativa

Valladolid, Junio de 2022

RESUMEN

Las empresas cuentan en la actualidad con grandes cantidades de datos, que son adquiridos de forma muy automatizada. Es de suma importancia para estas empresas dar un uso adecuado a estos datos y contar con procedimientos, basados en dichos datos, que sirvan de ayuda a la toma de decisiones y que permitan generar ventajas competitivas respecto a sus competidores. El Aprendizaje Automático proporciona técnicas eficientes para realizar predicciones basándose en la experiencia adquirida con los datos disponibles mediante potentes métodos numéricos y estadísticos.
En esta memoria se presentan algunas de estas técnicas de Aprendizaje Automático basadas en árboles de decisión, que sirven para la predicción de variables numéricas y cualitativas. A pesar de su aparente simplicidad, los árboles de regresión y clasificación proporcionan técnicas predictivas efectivas y su extensión a los "bosques aleatorios" es una de las técnicas recientes más recomendables en Aprendizaje Automático. Esta memoria también trata su aplicación práctica mediante software estadístico.

ABSTRACT

Companies today have vast amounts of data, which is acquired in a highly automated manner. For these companies, it is of great importance to make proper use of this data and to apply data-based procedures that help them make decisions and that allow them to generate competitive advantages over their competitors. Machine Learning provides efficient techniques to learn from data using powerful numerical and statistical methods.
This work reviews some machine learning tools, which are used for the prediction of numerical and qualitative variables based on trees. Despite its apparent simplicity, regression and classification trees provide effective predictive techniques and their extension to "random forests" is one of the most recommended recent techniques in Machine Learning. This work also deals with its practical application through statistical software.

ÍNDICE:

ÍNDICE DE FIGURAS:

ÍNDICE DE TABLAS:

1. INTRODUCCIÓN

El Trabajo Fin de Grado que se presenta en esta memoria presente se fundamenta en una extensión de los conocimientos adquiridos en la asignatura de "Estadística Empresarial" dentro del Grado en Ingeniería en Organización Industrial en la Universidad de Valladolid.

El principal objetivo del trabajo consiste en entender el funcionamiento de los árboles de decisión, como modelos predictivos muy utilizados en Aprendizaje Automático (Machine Learning) supervisado. Estos modelos, junto a su extensión a los denominados "bosques aleatorios", pensamos pueden ser de gran utilidad en diferentes áreas de la industria en los que sea necesario realizar predicciones para interesantes variables respuesta que no son observadas directamente. En este trabajo se pretende profundizar en su aplicación tratando, como ejemplos, su aplicación en diferentes conjuntos de datos prácticos.

El Aprendizaje Automático supervisado trata de crear estos modelos predictivos "aprendiendo" de la información en un conjunto patrón o de entrenamiento donde se conocen los valores de las variables que se usan en esta predicción junto a los valores de la variable respuesta a predecir. Será clave evitar modelos predictivos que "sobreajusten" este conjunto patrón o de entrenamiento.

Los árboles de decisión que se tratarán en esta memoria se han aplicado a la predicción de variables continuas (árboles de decisión de regresión) y a variables categóricas (árboles de decisión de clasificación).

Veremos también como los "bosques aleatorios" sirven para aumentar su capacidad predictiva y su aplicabilidad respecto a la metodología básica de árboles de decisión.

Este trabajo se ha realizado utilizando el lenguaje de programación R. Esto ha implicado un esfuerzo de aprendizaje de este lenguaje ya que el paquete estadístico utilizado en la titulación fue el `statgraphics`.

2. CLASIFICACIÓN SUPERVISADA

2.1 INTRODUCCIÓN

Para comprender de qué tratan los problemas de clasificación supervisada, es necesario entender previamente en qué consiste el machine learning.

Entendemos el concepto de Machine learning o aprendizaje automático como sistemas que aprenden de forma continua y son capaces de predecir los comportamientos futuros a partir de un conjunto de datos. Entendemos por "aprendizaje" cuando realizamos cada vez mejor una acción a partir de la experiencia, es decir, a medida que vamos contando con más datos e información.

De esta forma, a medida que los modelos desarrollados mejoran con la experiencia, estos serán capaces de generalizar comportamientos y podremos aplicarlos en otros conjuntos de datos.

Añadir que el aprendizaje automático se relaciona con el reconocimiento de patrones y la automatización de métodos científicos a través de modelos estadísticos y matemáticos.

De forma general, podemos distinguir tres ramas de aprendizaje:

- Aprendizaje Supervisado
- Aprendizaje No Supervisado
- Aprendizaje Reforzado

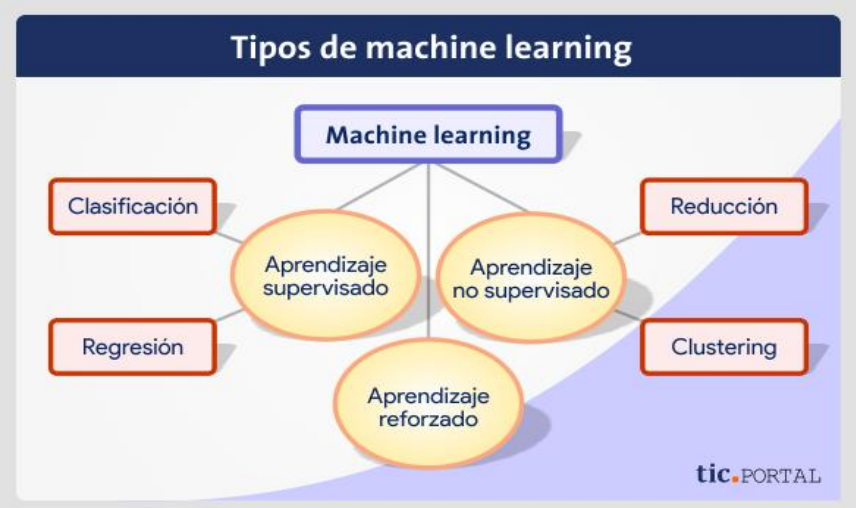


Figura 1: Clasificación general de los problemas de Machine Learning. (Fuente: ticportal,2021)

El aprendizaje automático supervisado tiene como objetivo predecir la variable denominada respuesta o dependiente en función del resto de las variables denominadas variables de entrada, predictoras o independientes. El modelo aprende del conjunto de datos denominado conjunto de entrenamiento.

Un ejemplo de la variable respuesta puede ser si una persona padece o no una enfermedad partiendo de diferentes variables de entrada las cuales representan diferentes características del individuo a analizar. Este caso lo estudiaremos más delante de forma práctica.

Por otro lado, estaría el aprendizaje no supervisado, donde el proceso de entrenamiento se realiza sobre un conjunto de datos que no está etiquetado, es decir, no conocemos la variable respuesta u objetivo. Por lo general, se utiliza en procesos donde nuestro objetivo es encontrar grupos similares en un conjunto. Un ejemplo de este aprendizaje podría ser cuando en las entidades financieras desarrollan modelos para agrupar a sus clientes en función de unas determinadas características.

Finalmente, los últimos años ha aparecido una nueva rama de machine learning conocida como *Reinforcement Learning* o aprendizaje reforzado y que su principal finalidad es, a diferencia de los modelos anteriores donde el objetivo principal es disminuir el error cometido, maximizar la recompensa. Para ello, aprenderá en función de un sistema de recompensas y castigos.

Para el desarrollo de este proyecto, nos vamos a centrar en el aprendizaje supervisado.

Dentro de las metodologías de predicción de aprendizaje supervisado existen problemas de clasificación o de regresión siendo la principal diferencia el resultado que queremos obtener. Por un lado, en los problemas de clasificación la variable de estudio (objetivo) es una variable cualitativa mientras que en los problemas de regresión la variable de estudio es una variable cuantitativa. Un ejemplo clásico de los modelos de clasificación es aquellos que determinan si un correo es *spam* o no mientras que un ejemplo de modelos de regresión sería predecir el precio de un coche de segunda mano en función de sus características.

En cuanto a terminología aplicada en aprendizaje automático es frecuente denominar a las variables de entrada como características y a las variables de salida como atributos.

Para realizar las predicciones haremos uso de un conjunto de datos que lo dividiremos en dos partes. La primera se utilizará como muestra de entrenamiento y que será utilizado para determinar los parámetros del modelo. La muestra del conjunto de datos restante se denomina test y se utiliza para comprobar el comportamiento del modelo y sus resultados.

Para la resolución de este tipo de problemas suponemos que los valores de las p variables explicativas proceden de un vector aleatorio, es decir, $X = (X_1, \dots, X_p)$, a su vez, la variable respuesta procede de una variable aleatoria Y. A partir de estos X e Y el aprendizaje automático supervisado tiene como objetivo encontrar una función $f(X)$ lo más cercana posible a Y.

A nivel teórico, para medir esta proximidad entre la función $f(X)$ y la variable respuesta Y se debe utilizar la función de perdida esperada siendo nuestro objetivo minimizar al máximo dicha perdida.

Tomando la distribución conjunta del vector aleatorio X y la variable aleatoria Y:

$$E_{\{X,Y\}}(L(f(X), Y))$$

Además, se puede añadir un vector aleatorio $Z = (x_i, y_i)$ para intentar disminuir $E_Z(L(f(X), Y))$. El aprendizaje automático supervisado comienza a través de una muestra aleatoria de tamaño n del vector aleatorio Z.

$$Z = \{(x_i, y_i)\}_{i=1}^{n}$$

Este conjunto de datos Z será el conjunto de datos de entrenamiento. Una vez se ha fijado Z, se puede definir la función de perdida empírica como:

$$\frac{1}{n}\sum_{i=1}^{n} L(f(x_i), y_i)$$

Se pueden realizar predicciones para una variable respuesta Y numérica, aunque también haremos predicciones para variables Y que tomen niveles categóricos.

Los valores de las k variables explicativas provienen de un vector aleatorio $X = (X_1, \dots, X_K)'$. Como hemos visto anteriormente, se toma una muestra aleatoria de estas variables en $\{(x_i, y_i)\}_{i=1}^{n}$, siendo y_i los valores de la variable respuesta y x_i un vector con los valores de las k variables explicativas en los n individuos.

En problemas de regresión, con variable respuesta numérica, la función de perdida típicamente aplicada es:

$$L(u, v) = (u - v)^2$$

En los problemas de clasificación en los que buscamos clasificar a un individuo, se utiliza la función de perdida 0-1:

$$L(u, v) = \begin{cases} 0 \text{ si } u = v \text{ (coinciden)} \\ 1 \text{ si } u \neq v \text{ (no coinciden)} \end{cases}$$

Estos problemas buscan una función f que minimice $E(L(f(X), Y))$ para la función perdida 0-1 definida anteriormente. La esperanza se obtiene a partir del vector aleatorio X y de la variable aleatoria Y.

Como $L(f(X), Y)$ solo toma valores 0-1, para minimizar esta variable aleatoria se busca minimizar la probabilidad de obtener una mala clasificación, es decir, que $f(X)$, predicción de la variable Y, no coincida con Y.

De esta manera estamos asumiendo que todos los errores producidos al clasificar erróneamente un individuo tienen el mismo coste, lo cual no es cierto.

Para los problemas de clasificación con dos clases $q = 2$, la variable respuesta tendrá dos valores posibles $Y = 0$, clase 1 o $Y = 1$, clase 2. De esta manera la regla podría consistir en asignar un individuo a la clase 1:

$$\hat{f}(x) = \hat{P}(Y = 1 | X = x) > a$$

Siendo a una constante entre 0-1.

Denotaremos por un valor "+" cuando la variable respuesta tome el valor $Y = 1$. En nuestra base de datos Diabetes indicaría que sí que tiene la enfermedad y daremos el valor "-" en caso contrario. El valor de la constante a condiciona la predicción del valor para una misma estimación.

Para este tipo de problemas podemos analizar cómo cambian los errores al clasificar mal una observación al mover la constante a:

		Constante a		
		+	-	← Predicho
Realidad	Casos positivos	VP(a)	FN(a)	P
	Casos negativos	FP(a)	VN(a)	N

Tabla 1: Errores mala clasificación

Teniendo en cuenta la tabla anterior, podemos obtener métricas de interés:

La tasa de verdaderos positivos (TPR) también denominada:

$$\text{Sensibilidad} = \frac{VP(a)}{P}$$

La cual es la probabilidad existente de clasificar de manera correcta a un individuo positivo.

Y la tasa de falsos positivos (TFR) también denominada:

$$1 - \text{Especificidad} = 1 - \frac{VN(a)}{N} = \frac{FP(a)}{N}$$

La cual es la probabilidad existente de clasificar de manera correcta a un individuo negativo.

La sensibilidad y la especificidad están en contraposición, por lo tanto, no es posible incrementar ambos valores al mismo tiempo, cambiando el valor de a.

Las denominadas curvas ROC son una representación de la sensibilidad frente a 1-especificidad las cuales nos permiten elegir el mejor valor de la constante a para nuestro objetivo.

En este gráfico se representa cada resultado de la predicción. El mejor predictor se representaría en la esquina superior izquierda, representando este un 100% de sensibilidad, es decir, ningún falso negativo y un 100% de especificidad, es decir, ningún falso positivo. El área bajo la curva ROC es uno de los criterios utilizados más importantes para comparar diferentes reglas de clasificación, una regla es buena cuando su área bajo la curva es cercana a uno.

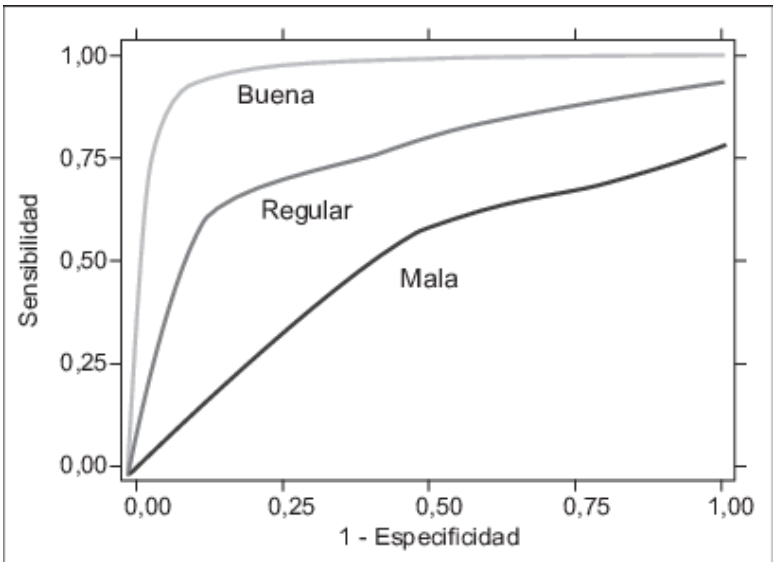


Figura 2: Representación de una Curva ROC genérica. (Fuente: Carlos Manterola,2010)

Muchos de los métodos que veremos se tratan de métodos lineales, donde la predicción de la respuesta se obtiene a través de las combinaciones lineales de las variables respuesta de la forma:

$$\beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k = \beta_0 + \beta' x$$

Por lo que contaremos con $p = k + 1$ parámetros libres $\beta_0, \beta_1, \dots, \beta_k$ a estimar. Este tipo de problemas son más sencillos de entender gracias al efecto lineal de las variables. Además, muchas veces, este modelo admite expresiones cerradas para los estimadores $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_k$ de los parámetros.

A continuación, presentaremos los métodos clásicos de aprendizaje supervisado más destacados.

2.2 REGRESIÓN LINEAL

Comenzaremos presentando la *regresión lineal simple*. La regresión lineal simple se basa en la creación de un modelo lineal, el cual relaciona una variable dependiente o variable respuesta y en función de una variable independiente o variable predictor x. Matemáticamente esta relación lineal se representa:

$$y \approx \beta_0 + \beta_1 x$$

Como se aprecia en la fórmula matemática, estamos realizando una aproximación. Por ejemplo, podemos hacer una regresión de la variable y *(Calificación en un examen)* sobre la variable x *(Tiempo de estudio)*.

$$\text{Calificación} \approx \beta_0 + \beta_1 \text{Tiempo}$$

β_0, β_1 son dos constantes que representan la ordenada en el origen y la pendiente, conocidos como parámetros del modelo. Cuando se hallan realizado las pruebas de entrenamiento del modelo necesarias se podrá calcular la calificación en base a un valor del tiempo proporcionado. Calculando:

$$\hat{y} \approx \hat{\beta}_0 + \hat{\beta}_1 x$$

Donde $\hat{y}$ representa una predicción de Y en función de $X = x$, en esta fórmula se utiliza el sombrero para denotar el valor estimado de un parámetro desconocido.

Para calcular los valores constantes β_0, β_1 los cuales son desconocidos necesitamos n pares de observaciones:

$$\{(x_i, y_i)\}_{i=1}^{n}$$

Los cuales representan una medición de X y otra de Y. El objetivo es encontrar un valor β_0 y una pendiente β_1 tales que la línea resultante se acerque el máximo posible al conjunto n de observaciones. Para ello se utilizará el método de los mínimos cuadrados.

Este método se basa en minimizar la suma de los cuadrados de los errores:

$$\sum_{i=1}^{n} {e_i}^2 = \sum_{i=1}^{n} (y_i - \hat{y_i})^2$$

Es decir, la suma de los cuadrados de la diferencia entre los valores reales observados (y_i) y los valores estimados $(\hat{y}_i)$. Realizando algunos cálculos se puede calcular β_0, β_1 como:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

Donde $\bar{x}$ e $\bar{y}$ son las medias de las muestras:

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \quad \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Las cuales definen las estimaciones de coeficientes por mínimos cuadrados para la regresión lineal simple.

En este caso de la regresión lineal múltiple se parte de n observaciones $\{(x_i, y_i)\}_{i=1}^{n}$ con $x_i = (x_{il}, \dots, x_{ik})'$, entonces:

$$y_i = \beta_0 + \beta_1 x_{1l} + \cdots + \beta_k x_{ik} = \varepsilon_i$$

De la ecuación anterior, tenemos que $\beta_0, \dots, \beta_k$ son parámetros desconocidos. Además, para el error aleatorio ε_i se suelen exigir las siguientes hipótesis:

- $E(\varepsilon_i) = 0$
- $Var(\varepsilon_i) = \sigma^2$
- ε_i sigue una distribución normal
- ε_i son independientes

Representando los vectores matricialmente tenemos:

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, X = \begin{pmatrix} 1 & x_{11} & \dots & x_{1k} \\ 1 & x_{21} & \dots & x_{2k} \\ \vdots & \vdots & \dots & \vdots \\ 1 & x_{n1} & \dots & x_{nk} \end{pmatrix}, \beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_k \end{pmatrix}, \varepsilon = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

La matriz X se conoce como matriz de diseño. Para estimar el vector β de parámetros desconocidos se debe resolver el problema:

$$\hat{\beta} = \arg mín \|Y - X\beta\|^2$$

Realizando la derivación matricial e igualando a 0 la ecuación obtenemos:

$$0 = \frac{\partial}{\partial \beta}\|Y - X\beta\|^2 = -2X'(Y - X\beta)$$

Lo que nos lleva a la siguiente ecuación:

$$\hat{\beta} = (X'X)^{-1}X'Y$$

2.3 DISCRIMINANTE LINEAL

El Análisis Discriminante Lineal (LDA) es un método de clasificación supervisado en el que la variable respuesta Y toma valores $\{1, \dots, q\}$ que codifican los posibles niveles para una variable categórica.

Para poder predecir los valores de esta variable Y partiremos de los valores de p variables discriminantes en el vector $X = (X_1, \dots, X_p)'$ y con el espacio de posibles valores que denominaremos χ. Este espacio vendrá dado por $\chi = R_p$

El Análisis Discriminante trata de encontrar reglas discriminantes donde este espacio Ж se divide en q regiones $R_1, \dots, R_q$. De esta manera podemos predecir que la variable Y pertenecerá a un nivel K cuando se cumpla que $x \in R_K$.

Para realizar esta división de la forma más optima posible, se necesitaría:

$$R_K = \left\{ x \in \chi : P(Y = K|X = x) = \max_{K=1,\dots,q} P(Y = K|X = x) \right\}$$

Lo que implica esta ecuación es que se le asigna una clase K a x para obtener una variable respuesta Y perteneciente a K, la cual sea la más probable en función de las variables predictivas $X = x$.

Así la regla de discriminación óptima a nivel teórico, parte del Teorema de Bayes, y nos diría que

$$P(Y = K|X = x) = \frac{P(Y = K)P(X = x|G = K)}{P(Y = k)P(X = x\,|G = k)} = \frac{\pi_K f_K(x)}{\sum_{K=1}^{q} \pi_k f_k(x)}$$

para

$$P(X = x|G = K) = f_K(x)$$

siendo $f_K(x)$ la densidad de X condicionada a que $G = K$.

Si conociésemos π_k y $f_k(x)$ para $k = 1, \dots, q$ la expresión nos permite, como hemos comentado anteriormente, obtener la forma óptima de definir la regla discriminante que divide χ de la mejor forma posible.

Entonces, esta regla de Bayes la podemos definir como:

$$R_{\text{Bayes}}(K) = \left\{ x \in \chi : \pi_K f_K(x) = \max_{K=1,\dots,q} \pi_k f_k(x) \right\}$$

La cual sería una división de del espacio χ y se puede definir la función de bayes f_{Bayes} como $f_{\text{Bayes}}(x) = K$ para $x \in R_{\text{Bayes}}(K)$.

La función de f_{Bayes} alcanza el mínimo de:

$$f \to E[L(f(X), Y)]$$

Cuando se utiliza la función de "perdida 0-1":

$$L(K,k) = \begin{cases} 0 & \text{si } K = k \text{ (acierto)} \\ 1 & \text{si } K \neq k \text{ (error)'} \end{cases}$$

donde la esperanza $E[L(f(X),Y)]$ se halla gracias a la distribución conjunta de X e Y se tiene que

$$E[L(f(X),Y)] = P(f(X) \neq Y).$$

Por lo tanto, podemos afirmar que esta regla sería la mejor discriminante siempre y cuando nuestra finalidad sea reducir la perdida esperada o la probabilidad total de clasificación errónea.

No obstante, hay que comentar que solamente sería posible minimizar la probabilidad o reducir la perdida esperada de forma óptima en el caso de conocer con exactitud el proceso generativo de nuestros datos. En la realidad, las distribuciones f_k son desconocidas, así como también suelen serlo las probabilidades.

La aproximación que solemos realizar es obtener aproximaciones razonables de las probabilidades, así como de las distribuciones. De esta forma esperaremos que la regla resultante proporciona también una aproximación lo suficiente razonable a la regla optima de Bayes.

2.4 DISCRIMINANTE CUADRÁTICO

El Análisis Discriminante Cuadrático (QDA) se parece en gran medida al Análisis Discriminante Lineal (LDA) visto anteriormente, pero, en este caso para distribuciones con dispersiones con grandes diferencias entre sí, se asume:

$$X|Y = k \sim N_p(\mu_k, \sum_k)$$

La regla de Bayes para $q = 2$ depende ahora de la siguiente función:

$$Q(x) = \beta_0 + \beta' x + x' B x$$

Con $\beta_0 \in R, \beta \in R^p$ y B una matriz (pxp) simétrica. En este caso, la mejor separación que se puede hacer a través de Bayes ya no es un simple hiperplano, si no que depende de la función vista anteriormente $Q(x)$ la cual es una función cuadrática en x, y por ello este procedimiento recibe el nombre de discriminación cuadrática.

Una vez vistos los dos modelos, la preferencia entre uno u otro está en la compensación entre el sesgo y la varianza. Hay que tener en cuenta que para p predictores la estimación de la matriz de covarianza requiere el cálculo de $\frac{p(p+1)}{2}$ parámetros, en cambio el QDA estima una matriz de covarianza distinta para cada clase con lo que obtenemos $\frac{kp(p+1)}{2}$ parámetros, lo que genera una cantidad muy grande de parámetros.

Como el LDA asume una matriz de covarianza común para las q clases, el LDA es mucho menos flexible que el QDA y tendrá una varianza menor lo que genera una mejor predicción, en contra de esto cabe destacar que si la predicción de que las q clases

comparten una matriz de covarianza común es errónea entonces LDA sufrirá un alto sesgo.

Por lo tanto, se puede afirmar que el LDA será preferible al QDA cuando hay pocas observaciones de entrenamiento y la reducción de la varianza será el valor más importante. Por lo contrario, el QDA será preferible al LDA cuando el número de observaciones es muy elevado. También es preferible este método si la suposición de una matriz de covarianza común es discutible.

2.5 DISCRIMINANTE LOGÍSTICO

El Análisis Discriminante Logístico es similar a un modelo de regresión lineal, pero en este caso se encarga de realizar la estimación de la probabilidad de ocurrencia de un evento binario partiendo de unas variables predictoras. Este modelo se encuadra en los denominados modelos lineales generalizados y aunque se pueda clasificar a partir de este modelo, su función principal es estimar la probabilidad de pertenencia al grupo. El resultado final de la clasificación se realiza de acuerdo con las probabilidades predichas.

Este modelo de regresión logística parte de una serie de restricciones. En primer lugar, debe existir una interdependencia de las observaciones, así como la existencia de una relación lineal entre el algoritmo natural de probabilidades (*odds*) y la variable continua. Por otro lado, la regresión logística no necesita de una distribución normal de la variable continua independiente.

Partiendo de p variables explicativas en un vector aleatorio $X = (X_1, \dots, X_K)'$ y una variable respuesta categórica con $q = 2$ niveles, que constan de los valores 0 o 1 en la variable Y, la discriminación logística asume:

$$\mathrm{P(Y = 1|X_1 = x_1, \dots, X_n = x_n)} = \frac{\exp{(\beta_0 + \beta' x)}}{1 + \exp{(\beta_0 + \beta' x)}}$$

Con $x = (x_1, \dots, x_K)' \in R^p, \beta = \left(\beta_1, \dots, \beta_p\right)' \in R^p y\, \beta_0 \in R$. Con lo que se puede obtener:

$$P(Y = 0|X_1 = x_1, \dots, X_n = x_n) = \frac{1}{1 + \exp{(\beta_0 + \beta' x)}}$$

A partir de una muestra de entrenamiento $\{(x_i, y_i)\}_{i=1}^n$, los parámetros $\beta_0\, y\, \beta_1$ son estimados a través de $\hat{\beta}_0$ y $\hat{\beta}_1 = \left(\hat{\beta}_1, \dots, \hat{\beta}_p\right)'$, hallados mediante algoritmos iterativos.

Si la estimación de $P(Y = 1|X_1 = x_1, \dots, X_n = x_n)$ es mayor que una constante $a \in (0,1)$, entonces se predice la clase $Y = 1$:

$$\frac{\exp{(\hat{\beta}_0 + \hat{\beta}_1 x + \cdots + \hat{\beta}_p x_p)}}{1 + \exp{(\hat{\beta}_0 + \hat{\beta}_1 x + \cdots + \hat{\beta}_p x_p)}} \geq a$$

Por lo que a la regla de clasificación se le asignará la clase $Y = 1$ si:

$$\sigma\left(\hat{\beta}_0 + \hat{\beta}' x\right) = \sigma\left(\hat{\beta}_0 + \hat{\beta}_1 x + \cdots + \hat{\beta}_p x_p\right) \geq a$$

Con la siguiente función logística:

$$\sigma(t) = \frac{e^t}{1 + e^t} = \frac{1}{1 + e^{-t}}$$

2.6 ÁRBOLES DE REGRESIÓN Y CLASIFICACIÓN

Los árboles de regresión y clasificación CART (*Classification and Regression Trees*) son utilizados por primera vez en el año 1964 por los científicos Sonquist y Morgan en la Universidad de Michigan.

Método por el cual, se pueden realizar decisiones de forma secuencial a través de los resultados y probabilidades obtenidas y se utilizan para producir búsquedas binarias, sistemas de expertos y árboles de juegos.

Se debe diferenciar por un lado los árboles de clasificación los cuales sirven para predecir la pertenencia de los objetos a un grupo a partir de las variables explicativas cuantitativas y cualitativas. Por otro lado, los árboles de regresión se utilizan para obtener un modelo explicativo a partir de las variables explicativas cuantitativas y cualitativas. Esta técnica ofrece muchas ventajas entre ellas:

- Facilidad de interpretación visual de la decisión tomada.
- Las variables independientes disminuyen.
- Se obtiene un nivel elevado de comprensión que será utilizado en la toma de decisiones.
- Excelente técnica para el control de la gestión empresarial.

Profundizaremos en el siguiente capítulo sobre este punto ya que es el tema principal de nuestro trabajo.

2.7 SESGO Y VARIANZA

En primer lugar, definimos brevemente estos dos conceptos:

- El sesgo es el valor que representa lo lejos que nuestro modelo define el valor estimado respecto al valor real.
- La varianza es una medida de dispersión que nos permite conocer qué tan dispersos están los datos del valor medio.

El error cuadrático de un estimador puntual θ al estimar un parámetro desconocido θ, se puede descomponer en la suma entre el sesgo de estimación al cuadrado y la varianza del estimador.

Por lo que algunos estimadores sesgados tendrá menor error medio cuadrático que otros estimadores insesgados si los estimadores sesgados logran tener una varianza bastante menor que la varianza del insesgado y se puede compensar el termino de sesgo al cuadrado.

Supongamos que tenemos el siguiente modelo de regresión

$$Y = f(X) + \varepsilon$$

con $E[\varepsilon] = 0$ y $Var(\varepsilon) = \sigma^2$ y se cuenta con la función estimadora f. (Teniendo en cuenta que esta función f depende del azar ya que es dependiente de la muestra de entrenamiento aleatoria), podemos obtener el error cuadrático medio al predecir la respuesta por $\hat{f}(x_0)$ cuando $X = x_0$ con $x_0 = (x_{01}, \dots, x_{0p})'$:

$$E\left[\left(Y - \hat{f}(x_0)\right)^2 \middle| X = x_0\right] = \sigma^2 + \left(E\hat{f}(x_0) - f(x_0)\right)^2 + Var\left(\hat{f}(x_0)\right)$$
$$= \text{"variabilidad irreducible"} + (\text{"sesgo"})^2 + \text{"Varianza"}$$

La variabilidad irreducible se refiere al termino de σ^2, el cual depende de la variabilidad del término del error ε, que no puede ser en ningún caso eliminada. Los otros dos términos vistos también anteriormente, dependen de la función $\hat{f}$.

Podemos ver estos dos conceptos representados gráficamente:

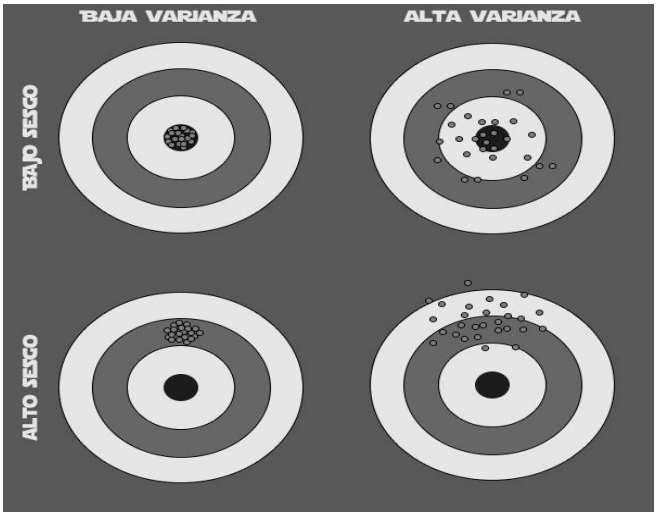


Figura 3: Sesgo-Varianza. (Fuente: Koldo Pina - Data Science,2020)

Contamos con cuatro situaciones:

- Sesgo pequeño y variabilidad alta
- Sesgo pequeño y variabilidad baja
- Sesgo alto y variabilidad alta
- Sesgo alto y variabilidad baja

Uno de los principales inconvenientes es el equilibrio entre el sesgo y la varianza del modelo que desarrollemos y que veremos cómo resolverlo en el siguiente apartado.

Nuestro principal objetivo será obtener un modelo que presente dicho equilibrio ya que si alguna de las dos variables pesa más que la otra puede dar lugar a modelos sobreajustados (*overfitting*) o infraajustados (*underfitting*).

2.8 ERROR APARENTE Y ERROR DE GENERALIZACIÓN (VALIDACIÓN CRUZADA)

Para evitar el problema que surge al intentar lograr una buena relación entre varianza y sesgo (sobreajuste), es importante distinguir entre los dos tipos de errores que existen en el aprendizaje automático: Error aparente y Error de generalización.

La función optima f serviría idealmente para reducir la perdida esperada:

$$E(L(f(X),Y))$$

donde esta esperanza se calcula con la distribución conjunta de (X,Y).

En el caso de clasificación se toma $L(u,v) = 1_{\{u \neq v\}}$ y en el caso de regresión $L(u,v) = (u-v)^2$.

Partiendo de una muestra aleatoria simple $X_1, \dots, X_n$, tenemos que la media muestral

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

permite aproximar la media poblacional $\mu = EX_i$.

Con nuestra muestra de entrenamiento $\mathcal{X} = \{(x_i, y_i)\}_{i=1}^n$ la cual es una muestra aleatoria simple de $\boldsymbol{\mathcal{X}} = \{(X_i, Y_i)\}_{i=1}^n$ podríamos pensar, al estimar la función f optima, tener en cuenta que $E(L(f(X),Y))$ es estimable por:

$$\frac{1}{n}\sum_{i=1}^{n} L(f(x_i), y_i)$$

Si llamamos $\hat{f}$ al estimador de la función de predicción que se obtiene desde nuestro conjunto de datos de entrenamiento $\mathcal{X}$. El error aparente se define mediante:

$$\overline{err} = \frac{1}{n}\sum_{i=1}^{n} L(\hat{f}(x_i), y_i)$$

En el caso de clasificación, el error aparente coincide con la proporción de observaciones clasificadas de forma errónea $y_i \neq \hat{f}(x_i)$ y para el caso de regresión coincide con la suma de errores $(y_i - \hat{f}(x_i))^2$

Es común referirse a este error aparente como error de entrenamiento ya que es el error que aparece probando $\hat{f}$ en el conjunto de datos de entrenamiento.

El *error de generalización* se define:

$$Err_{\mathcal{Z}} = E_{\{X,Y\}}\left[L\left(\hat{f}(X), Y\right) | \boldsymbol{\mathcal{X}} = \mathcal{X}\right]$$

Es común referirse a este error de generalización como error en el conjunto de datos de test o error test. Es decir:

$$Err = E[Err_{\mathcal{Z}}]$$

Este es el error que realmente nos interesa estimar ya que el error aparente resulta bastante optimista puesto que usamos las variables respuestas ya conocidas para evaluar el buen funcionamiento de $\hat{f}$.

La *validación cruzada*, la cual nos va a permitir estimar el error de generalización, es una técnica que se utiliza para evaluar los resultados de predicción de un modelo y su principal objetivo es garantizar la independencia de dichos resultados del posible sobreajuste al conjunto específico de entre entrenamiento/test.

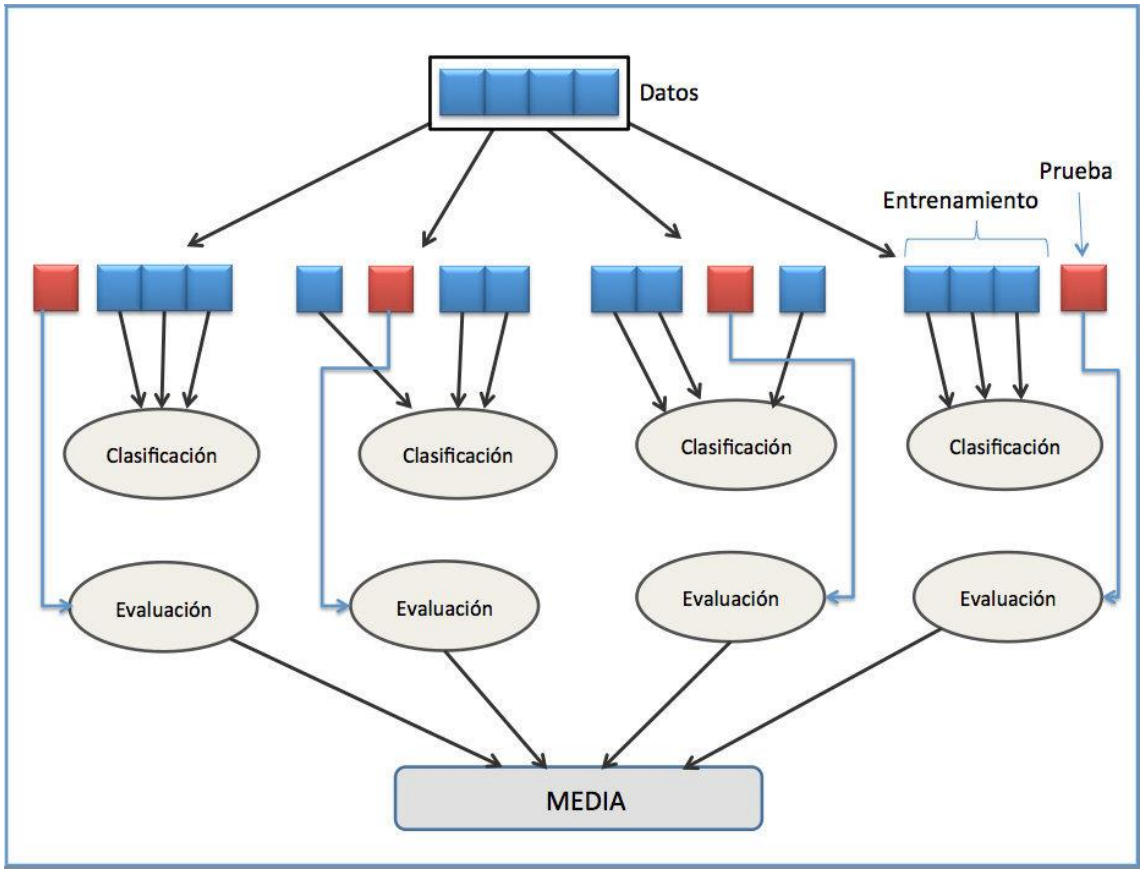


Figura 4: Validación cruzada k-fold. (Fuente: Joan.domenech91 - Trabajo propio,2020)

En la validación cruzada el conjunto de datos se divide en:

$$\boldsymbol{\mathcal{X}} = \{\mathcal{X}_1; \mathcal{X}_2\}$$

Donde usamos solo $\mathcal{X}_1$ como muestra de entrenamiento y $\mathcal{X}_2$ como muestra de prueba y con esta situación se desarrolla y evalúa el modelo.

Debido a que los valores de Z_2 no se emplean a la hora de entrenar el modelo que si realizo solo con los valores de Z_1, de esta manera evitamos el sobreajuste. Así, tendremos una idea más fiable de como funcionará el modelo predictivo para nuevas observaciones.

En la parte práctica de nuestros modelos, se verá cómo hemos dividido el modelo en dos partes un 60% destinado a $\mathcal{X}_1$ es decir, el conjunto de entrenamiento y un 40% a $\mathcal{X}_2$ es decir, el conjunto de validación o test.

Para realizar la validación cruzada existen varios procedimientos:

- Validación cruzada dejando uno fuera (*Leave One Out cross validation*)
- Validación cruzada de K-fold (*k-fold cross validation*)

La validación cruzada *Leave-one-out cross validation* se caracteriza por escoger como datos de test una muestra o registro y el resto serían los datos de entrenamiento. El error cometido con esta validación es muy bajo, sin embargo, el coste computacional es elevado.

Para obtener el error de generalización usando *Leave-one-out* usamos:

$$LOO = \frac{1}{n}\sum_{i=1}^{n} L(\hat{f}_{-i}(x_i), y_i)$$

Siendo $\hat{f}_{-i}(x_i)$ la predicción de y_i para $X = x_i$. Esto se cumple cuando la función $\hat{f}_{-i}$ se ha predicho al eliminar la observación i-ésima de la muestra patrón y utilizando el método predictivo a analizar.

En la validación cruzada k-fold el conjunto de datos se divide en k subconjuntos.

$$\boldsymbol{\mathcal{X}} = \{\mathcal{X}_1; \dots; \mathcal{X}_K\}$$

Donde uno de los subconjuntos se utiliza como muestra de entrenamiento $(K-1)$ y el resto se emplea en evaluar el modelo. Este proceso es repetido K veces y se obtiene como resultado de la evaluación del modelo la media aritmética de cada una de las iteraciones. La principal desventaja es el coste computacional que supone para un número de iteraciones elevado.

3. ÁRBOLES DE DECISIÓN

3.1 INTRODUCCIÓN

Un árbol de decisión es un modelo predictivo que separa el conjunto de predictores y agrupa observaciones con valores próximos para la variable respuesta o la variable dependiente. El árbol de decisión es un algoritmo supervisado de aprendizaje automático ya que es necesario una variable dependiente en el conjunto de entrenamiento para que el modelo aprenda.

Un árbol se forma a través de una secuencia de preguntas, las cuales generan dos únicas respuestas si o no. Si la variable es numérica entonces estaremos hablando de árboles de regresión, y hablaremos de árboles de clasificación si la variable es categórica.

Esta secuencia de preguntas realiza una partición en el espacio de valores posibles para las variables predictoras $X = (X_1, \dots, X_p)'$. En el caso de querer realizar predicciones para una nueva observación $x = (x, \dots, x_p)'$ se usa la clase con mayor frecuencia para los árboles de clasificación y en los árboles de regresión se usa la media.

A continuación, se puede apreciar un árbol sencillo:

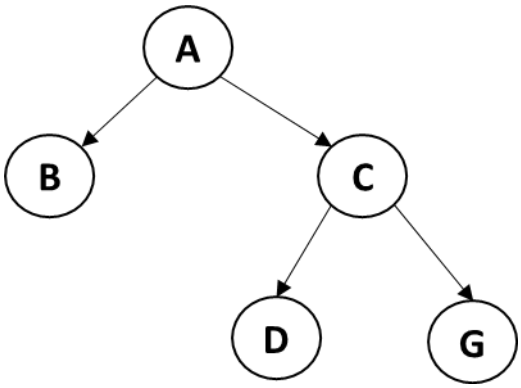


Figura 5: Ejemplo de árbol de decisión

El árbol de la imagen se compone de las partes que explico a continuación:

- Un nodo es la materia sobre la que se forma el árbol.
- Un nodo raíz (A), ya que este no tiene antecesores.
- Un nodo padre (A,C) si tiene algún nodo por debajo de él.
- Un nodo hijo (B,C,D,G), son los que tienen por encima un nodo padre.
- Un nodo hermano (B,C;D,G) es aquel que tiene el mismo padre.
- Un nodo terminal (B,D,G) es aquel que no tiene más nodos por debajo.
- Una Rama es la unión entre dos nodos.
- El grado de un nodo es la cantidad de nodos descendientes directos que tiene.

3.2 ÁRBOLES DE CLASIFICACIÓN

Para explicar la metodología de los árboles de clasificación, debemos saber que la variable respuesta en este caso será categórica con q niveles. Las variables clasificadoras $X = (X_1, \dots, X_p)'$ pueden ser numéricas o categóricas. Y partimos de nuestra muestra de entrenamiento con observaciones ya clasificadas:

$$\{(x_i, y_i)\}_{i=1}^{n} \; con \; y_i \in \{1, \dots, q\}$$

Para realizar las particiones del árbol debemos seleccionar como se van a llevar a cabo dichas particiones en los nodos existentes. Para ello llamaremos partición s (split), asumiendo que dicha partición ocurre cuando se cumple o no una condición.

Para denotar dichas afirmaciones, utilizaremos:

$$p_{mk} = \text{"proporcion de observaciones en el grupo k en el nodo m"}$$

$$p_{mk} = \frac{\sum_{i \in R_m} I_{\{g_i = k\}}}{n(m)}$$

Para $k = 1, \dots, q$, donde I_A vale 1 si la condición A es cierta y vale 0 si no es cierta.

A continuación, a través de la regla de resubstitucion podemos asignar las nuevas observaciones a las clases correspondientes. Por esta regla, cuando llega una nueva observación $x = (x, \dots, x_p)'$ esta realiza la toma de decisiones necesaria hasta que llega a un nodo terminal m y se asigna a un grupo $k(m) \in 1, \dots, q$ si se cumple que:

$$p_{mk(m)} = \max_{1 \le k \le q} p_{mk}$$

Algunas de las formas para medir la impureza $Q_m(T)$ en el nodo m del árbol T son las siguientes:

- Entropía:

$$Q_m(T) = -\sum_{k=1}^{q} p_{mk} \log(p_{mk})$$

- Índice de Gini:

$$Q_m(T) = \sum_{k=1}^{q} p_{mk}(1 - p_{mk})$$

- Error de mala clasificación:

$$Q_m(T) = 1 - p_{mk(m)}$$

La medida de la impureza $Q_m(T)$ obtendrán su valor más alto cuando $p_{mq} = \frac{1}{q}$ y obtendrá su valor más pequeño 0, si $p_{mq} = 1$ para algún k.

Tras la definición de estas funciones de impureza, se puede llevar a cabo la partición de los nodos. La cual se realiza seleccionando la partición s con la que se obtiene una mayor reducción de la impureza.

Cuando se realiza dicha partición esta genera dos nuevos nodos:

- El nodo m_L si se satisface dicha condición
- El nodo $m_R$si no se satisface la condición

Para realizar la división con la condición s en el nodo m, tenemos que poner atención en el decrecimiento $\Delta(s,m)$ de la impureza al generar un nuevo árbol T' con las dos divisiones vistas anteriormente. Matemáticamente:

$$\Delta(s,m) = Q_m(T) = \frac{n(m_L)}{n(m)} Q_{m_L}(T') + \frac{n(m_R)}{n(m)} Q_{m_R}(T')$$

A esta manera de actuar se la denomina como "avariciosa" ya que el procedimiento solo se centra en la reducción de la impureza en el paso actual sin valorar sus futuras implicaciones.

Por último, también se puede hallar la impureza total del árbol a través de la siguiente expresión:

$$Q(T) = \sum_{m \in T} \frac{n(m)}{n} Q_m(T)$$

3.3 ÁRBOLES DE REGRESIÓN

Por otro lado, existen los árboles de regresión, el procedimiento es parecido al visto anteriormente para los árboles de clasificación, pero en este caso la variable de salida Y es cuantitativa.

En este caso el procedimiento también requiere realizar una división $R_1, \dots, R_M$, dependientes de los valores de las variables explicativas $X_1, \dots, X_p$ para obtener:

$$f(x) = E(Y|X_1 = x_1, \dots, X_p = x_p)$$

Gracias a funciones con la siguiente forma:

$$\hat{f}(x) = \sum_{m=1}^{M} \hat{c}_m I_{\{x \in R_m\}}$$

Con esto queremos expresar que la función regresora tendrá un valor constante para todo x asignado a R_m.

A través del conjunto de entrenamiento $\{(x_i, y_i)\}_{i=1}^n$ con $y_i \in R$, la división $R_1, \dots, R_M$ es conocida y realizamos un criterio de mínimos cuadrados $\sum_{i=1}^n (y_i - \hat{f}(x_i))^2$, se puede apreciar que el mejor valor que puede tomar la constante $\hat{c}_m$ es la media de las respuestas y_i para todas las observaciones que están en R_m.

$$\hat{c}_m = \frac{1}{n(m)} \sum_{i \in R_m} y_i$$

En este caso, no se utilizarán las mismas ecuaciones de impureza mostradas para los árboles de clasificación. Por el contrario, se empleará el error promedio del árbol T en el nodo m:

$$Q_m(T) = \frac{1}{n(m)} \sum_{i \in R_m} (y_i - \hat{c}_m)^2$$

Para hallar la impureza total del árbol, se haría a través de la siguiente fórmula:

$$Q(T) = \frac{1}{n} \sum_{m=1}^{M} \sum_{i \in R_m} (y_i - \hat{c}_m)^2$$

Análogamente a lo que hemos visto en los árboles de clasificación, en este caso utilizamos el siguiente criterio de penalización:

$$CP_\alpha(T) = Q(T) + \alpha|T|$$

Donde $|T|$ es el número de nodos terminales tras realizar la poda del árbol T_{max}.

3.4 PODA DE ÁRBOLES

La poda de árboles es necesaria cuando se obtiene un árbol muy grande y complejo, ya que cuando un árbol tiene demasiadas ramas o divisiones estas desembocan en muchos nodos terminales, los cuales tendrán una alta pureza. Esto implica que no tendremos un error aparente muy elevado, en cambio sí pudriéramos obtener un gran sobreajuste y por lo tanto un error de generalización.

Para mostrar con un ejemplo práctico la poda de árboles voy a utilizar la base de datos “Diabetes”, en la que más adelante se entrará en detalle.

Como se puede apreciar en la siguiente imagen, el árbol es muy complejo, con demasiadas divisiones lo que crea un sobreajuste, pero solo con unas pocas muestras mal clasificadas. Estos árboles tan desarrollados los vamos a denotar a través de T_{max}.

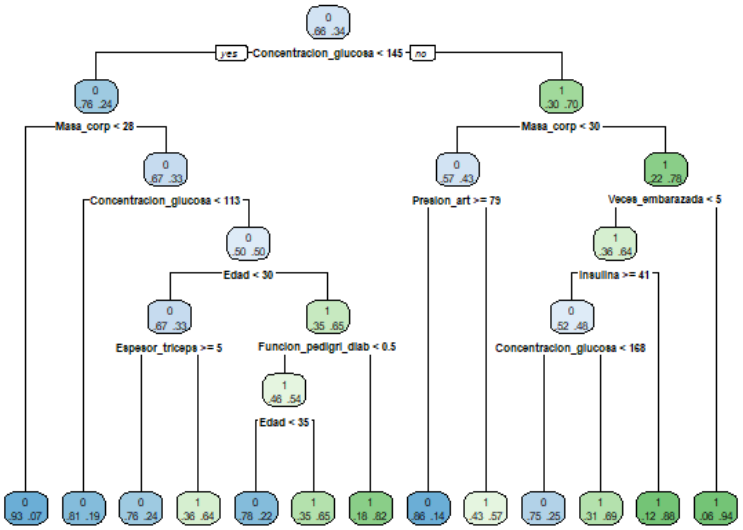


Figura 6: Ejemplo de árbol de clasificación

Cuando se realiza la poda de un árbol se cortan algunas ramas y todas las que dependían de estas se deben eliminan. Denominaremos $T_1 > T_2$ para expresar que T_2 es el resultado de podar el árbol T_1. Y denotaremos $|T|$ al tamaño del árbol, el cual se define gracias a la cantidad de nodos terminales.

El interés principal será reducir el criterio penalizado para una constante α definida:

$$CP_\alpha(T) = \sum_{m=1}^{|T|} \frac{n(m)}{n} Q_m(T) + \alpha|T|$$

En el caso de tener un valor α grande entonces se penalizará en mayor parte a los subárboles de gran tamaño, aunque estos cuenten con una impureza $Q(T)$ pequeña. Por lo que lo más correcto sería aplicar el error de clasificación:

$$CP_\alpha(T) = \text{Error aparente} + \alpha|T|$$

Para un valor constante de α, existe un árbol podado de forma que se reduzca al máximo la cantidad de subárboles $T(\alpha) < T_{max}$ y este minimiza el criterio CP_α.

Además, existe una forma secuencial de búsqueda de las ramas con menor importancia del árbol complejo T_{max} y eliminado estas divisiones se obtiene la siguiente secuencia:

$$T_{max} = T_0 > T_1 > \cdots > T_L = \{t_0\}$$

Los cuales están asociados con los siguientes parámetros:

$$0 = \alpha_0 < \alpha_1 < \cdots < \alpha_L$$

De tal forma que cada T_l tiene un CP_α óptimo para $\alpha \in [\alpha_l, \alpha_{l+1})$.

Conseguimos así que $T_{max} = T_0$, es decir, que coincida con el árbol primitivo. Además de esta manera el árbol tendrá todas las observaciones en el nodo raíz $\{t_0\}$.

A continuación, debemos elegir cual de todos estos árboles $L + 1$ es el más apropiado, para ello usaremos la validación cruzada de tipo k-fold. Usaremos entonces la regla "1-se", la cual se basa en un gráfico como el mostrado en la siguiente imagen:

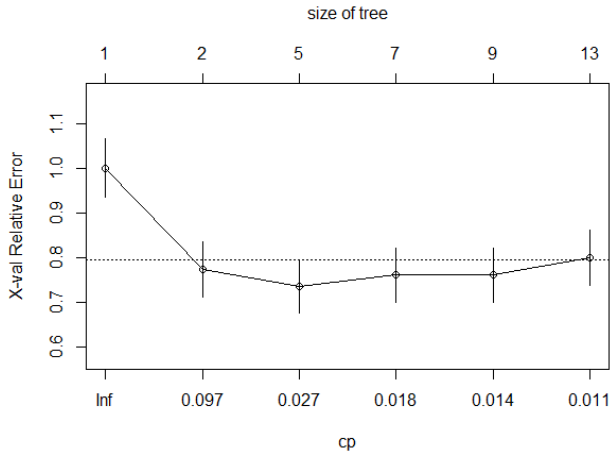


Figura 7: Ejemplo gráfico del error

Como se aprecia en el gráfico anterior, en la parte inferior se muestra CP_α para los valores de α vistos anteriormente. El caso de $\alpha = inf$ representa la situación en la que se toma un valor muy elevado. En la parte superior se muestra el tamaño de los árboles asociados a sus CP_α. Por último, el grafico refleja también el error de generalización para cada uno de los árboles-CP_α a través de un círculo. Y las líneas verticales representan el error estándar de las estimaciones del error de generalización.

La mejor elección será el árbol más simple siempre y cuando sea equivalente al error estimado de generalización. En nuestro caso, el árbol con menor error estimado de generalización sería para un tamaño 5, pero elegimos el árbol de tamaño 2 porque, teniendo en cuenta la variabilidad en esa estimación del error (las barras representan una unidad del "error standard" o estimador de la desviación típica), vemos que su error estimado de generalización no es muy diferente del de tamaño 5. En este caso, elegimos el árbol de tamaño 2 porque es "equivalente" pero de menor tamaño que el de tamaño 5. Se consideran equivalentes (o no de funcionamiento inferior) todos aquellos árboles cuyo estimador del error de generalización quede por debajo de la línea horizontal discontinua mostrada en la Figura 7, en lo que se conoce como la "regla 1-se".

3.5 HERRAMIENTAS PRÁCTICAS

Para la realización de los casos prácticos se ha utilizado el lenguaje de programación R, el cual, fue desarrollado por los neozelandeses Ross Ihaka y Robert Gentleman, en 1996. Es de software libre y es uno de los más utilizados para realizar análisis estadísticos y gráficos. Cabe destacar que R forma parte del sistema GNU y se distribuye bajo la licencia GNU GPL. La consola de R se ve como muestro en la siguiente imagen:

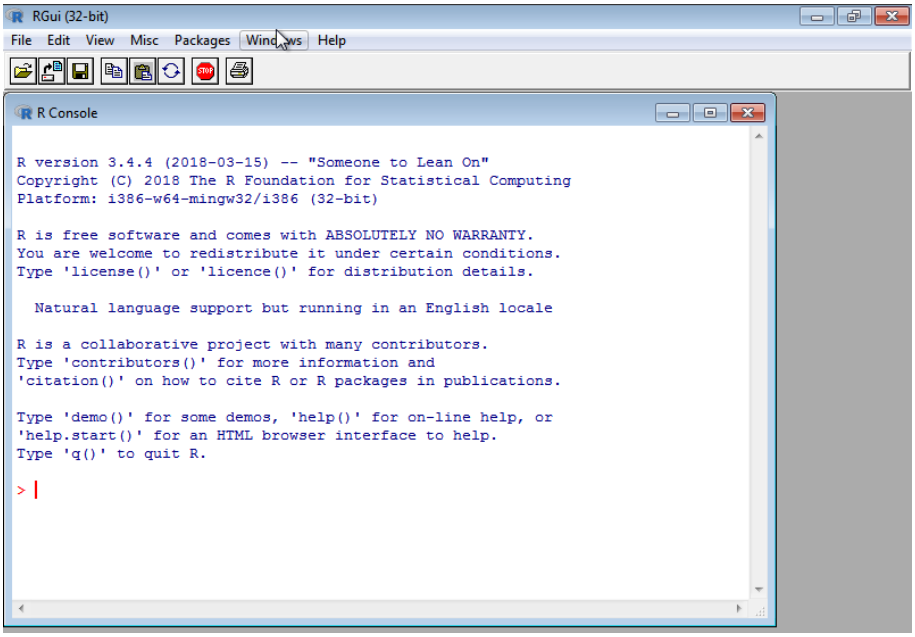


Figura 8: Consola R

En cambio, nosotros vamos a utilizar el programa `Rstudio` que es más completo que la propia consola. `Rstudio` es un IDE (Entorno de Desarrollo Integrado) para R, en el cual se puede encontrar una consola, un editor de sintaxis para apoyar la ejecución del código, así como, herramientas para el trazado y la depuración, entre otras.

El programa Rstudio se puede visualizar en la siguiente imagen:

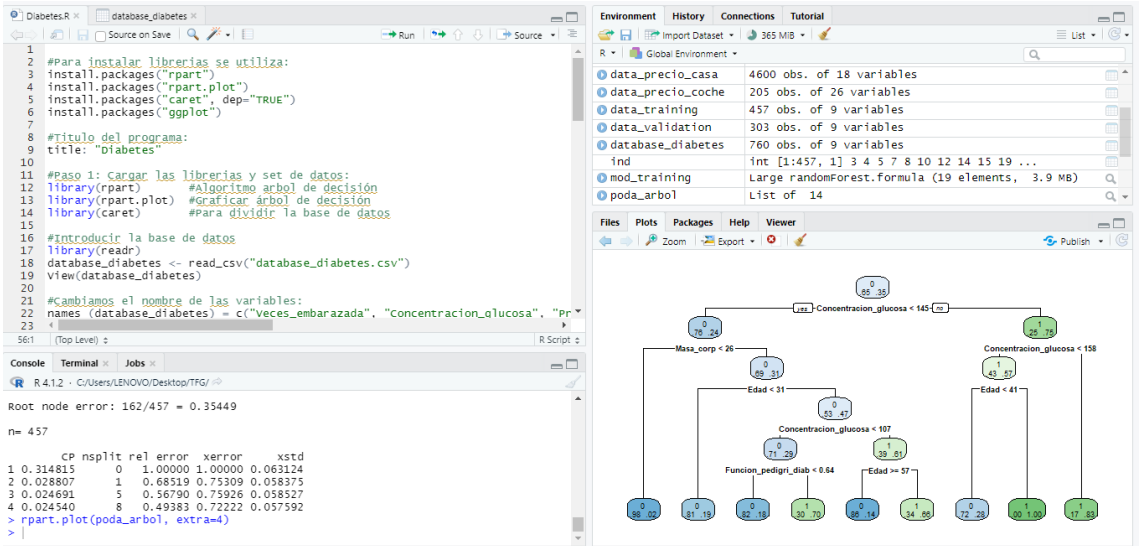


Figura 9: Programa Rstudio

3.6 EJEMPLO ÁRBOL DE REGRESIÓN

En primer lugar, vamos a realizar un ejemplo para el método de árboles de regresión, para ello hemos utilizado un conjunto de datos que busca predecir el precio de un vehículo a través de distintas variables que representan las características del vehículo. Se ha utilizado el lenguaje de programación R.

En el programa de R, se deben instalar los paquetes necesarios para la realización del árbol de regresión:

```
#Para instalar librerías se utiliza:

install.packages("rpart")

install.packages("rpart.plot")

install.packages("ggplot")
```

A continuación, se deben cargar las librerías:

```
#Paso 1: Cargar las librerías y set de datos:

library(rpart) #Algoritmo árbol de decisión

library(rpart.plot) #Graficar árbol de decisión
```

El conjunto de datos original se encuentra en formato CSV como vemos en la imagen siguiente:

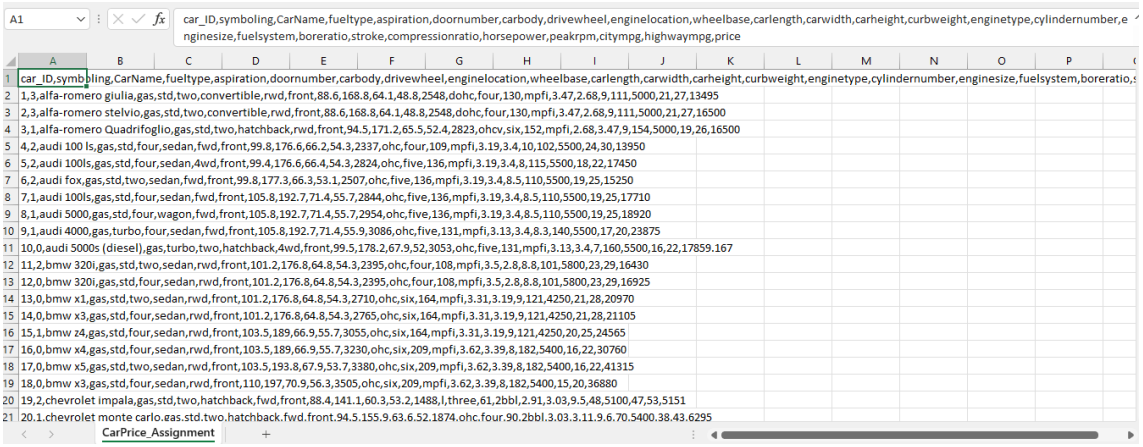
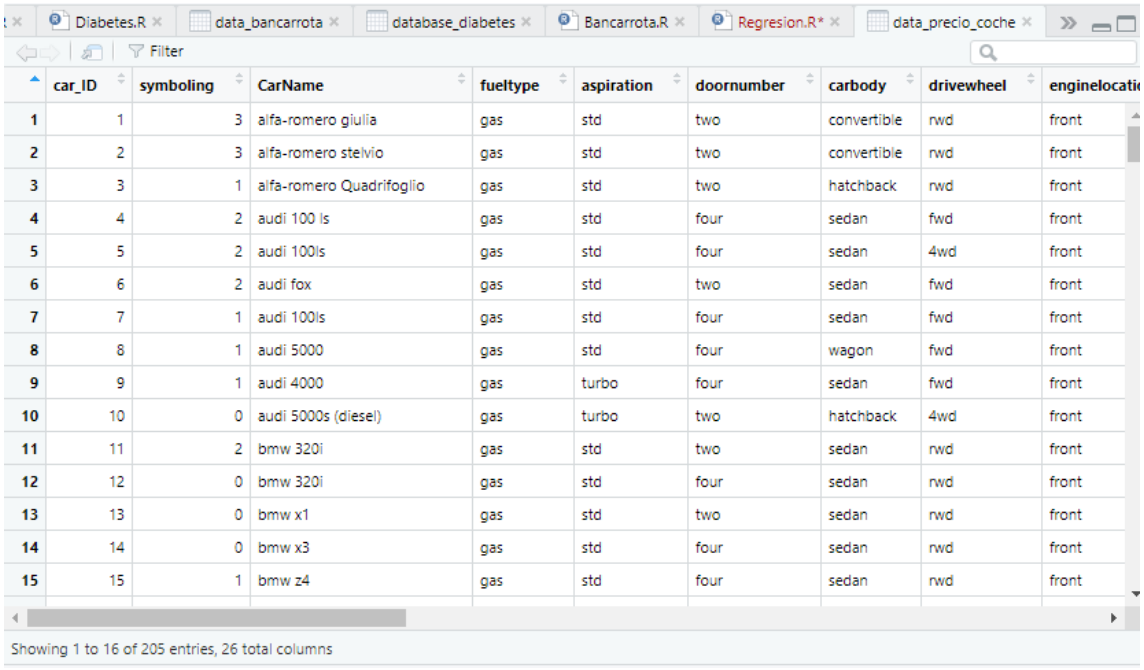


Figura 10: Conjunto de datos CSV (Precio_coche)

Y así, al importarlo a Rstudio:



Diabetes.R | data_bancarrota | database_diabetes | Bancarrota.R | Regresion.R* | data_precio_coche

Filter

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocati
1	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front
2	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front
3	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front
4	4	2	audi 100 ls	gas	std	four	sedan	fwd	front
5	5	2	audi 100ls	gas	std	four	sedan	4wd	front
6	6	2	audi fox	gas	std	two	sedan	fwd	front
7	7	1	audi 100ls	gas	std	four	sedan	fwd	front
8	8	1	audi 5000	gas	std	four	wagon	fwd	front
9	9	1	audi 4000	gas	turbo	four	sedan	fwd	front
10	10	0	audi 5000s (diesel)	gas	turbo	two	hatchback	4wd	front
11	11	2	bmw 320i	gas	std	two	sedan	rwd	front
12	12	0	bmw 320i	gas	std	four	sedan	rwd	front
13	13	0	bmw x1	gas	std	two	sedan	rwd	front
14	14	0	bmw x3	gas	std	four	sedan	rwd	front
15	15	1	bmw z4	gas	std	four	sedan	rwd	front

Showing 1 to 16 of 205 entries, 26 total columns

Figura 11: Conjunto de datos Rstudio (Precio_coche)

La variable que vamos a predecir es la que se denomina Price (Precio) que es la que determinara el precio del vehículo. El resto de las variables quedan explicadas en la siguiente tabla:

Explicación de las variables	
Car_ID	ID del coche
Symboling	Factor de riesgo
CarName	Nombre del coche
Fueltype	Tipo de combustible
Aspiration	Aspiración
Doornumber	Número de puertas
Carbody	Carrocería
Drivewheel	Rueda motriz
Enginelocation	Ubicación del motor
Wheelbase	Distancia entre ejes
Carlength	Longitud del coche
Carwidth	Anchura del coche
Carheight	Altura de la cabina
Curbweight	Peso en vacío
Enginetype	Tipo de motor
Cylindernumber	Número de cilindros
Enginesize	Tamaño del motor
Fuelsystem	Sistema de combustible
Boreratio	Relación de agujeros
Stroke	Carrera

Compressionratio	Relación de compresión
Horsepower	Caballos de vapor
Peakrpm	Pico de revoluciones
Citympg	Consumo en ciudad
Highwaympg	Consumo en carretera
Price	Precio

Tabla 2: Explicación de las variables (Precio_coche)

Para poder crear un modelo necesitamos seleccionar las variables de interés. Para ello, en este caso vamos a escoger únicamente las variables cuantitativas. Para realizar esta selección de variables debemos ejecutar el siguiente código:

```
# Seleccionamos las variables de interés:

data<-
data_precio_coche[,c("price","highwaympg","citympg","peakrpm","horsepower",
"compressionratio", "stroke", "boreratio", "enginesize", "curbweight", "carheight",
"carwidth", "carlength", "wheelbase")]
```

Una vez dividido el conjunto de datos podemos realizar el árbol de regresión, para ello utilizamos la función `rpart`.

En primer lugar, introducimos la variable dependiente de nuestro modelo, el precio (price) la cual vamos a predecir con una selección del resto de variables.

Después indicamos que nuestra variable dependiente es cuantitativa e introducimos nuestro conjunto de datos:

```
#Creamos el árbol de decisión:

arbol <- rpart(formula = price ~ .,method= "anova", data= data)
```

Una vez realizado nuestro árbol, imprimimos su información a través de `print(arbol)` y obtenemos lo siguiente:

```
> #Imprimimos el resultado
> print(arbol)
n= 125

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

 1) root 125 7490232000 13206.370
   2) enginesize< 182 116 2596206000 11517.890
     4) curbweight< 2544 70  312392900  8411.743
       8) horsepower< 89 50   62718040  7493.100 *
       9) horsepower>=89 20  101991600 10708.350 *
     5) curbweight>=2544 46  580705700 16244.630
      10) wheelbase< 100.8 20  143774400 14013.060 *
      11) wheelbase>=100.8 26  260718400 17961.230
        22) carheight>=56.1 11   56018770 15880.450 *
        23) carheight< 56.1 15  122148000 19487.130 *
   3) enginesize>=182 9  300804900 34969.000 *
```

Figura 12: División numérica árbol (Precio_coche)

Para imprimir el grafico del árbol:

```
#Para imprimir el gráfico:

rpart.plot(arbol)
```

Que nos genera el siguiente gráfico:

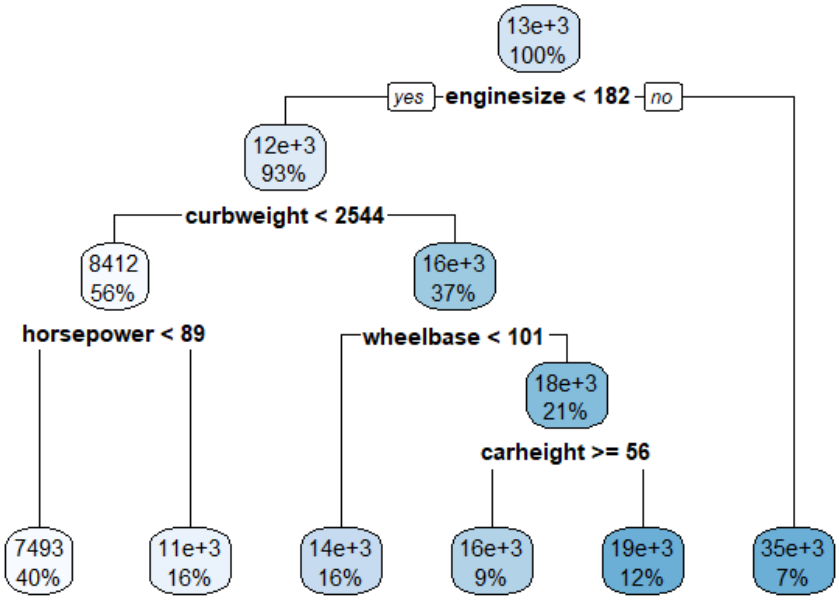


Figura 13: Gráfico del árbol (Precio_coche)

A continuación, explicamos el resultado obtenido en la imagen anterior:

La primera variable que toma el modelo para decidir es el tamaño del motor (Enginesize) la cual solo es mayor o igual a 182cc en el 7% de los casos y, en este caso, el valor del coche queda ya fijado al llegar el árbol a un nodo final en 34.969 USD.

Por otro lado, si el tamaño del motor es menor de 182cc, la siguiente variable que toma el modelo en consideración es el peso en vacío (curbweight), En este caso, la variable se analiza viendo si es menor de 2.544 libras. En caso afirmativo, en el lado izquierdo del árbol, tenemos una nueva división con la variable caballos de vapor (horsepower), la cual si es menor de 89 hp se fija un precio para el coche de 7.493,1 USD. Por otro lado, si los caballos de vapor del coche son mayores o iguales a 89 hp entonces el precio del coche lo estima en 10.708,35 USD.

Si en cambio vamos por el lado derecho de la primera división de la variable peso en vacío (curbweight), podemos ver que la siguiente variable que toma para dividir el modelo es la distancia entre ejes (wheelbase). Si esta es menor a 100,8 pulgadas, entonces tenemos un nodo final que nos indica que el precio del coche es de 14.013,06 USD. En cambio, si la condición de la variable de que sea menor de 100,8 pulgadas no se

cumple, entonces llegamos a una nueva división a través de la variable que indica la altura de la cabina (carheight).

En esta última división, el modelo valora si el coche tiene una altura mayor o igual a 56,1 pulgadas. En caso afirmativo el valor del coche lo estima en 15.880,45 USD. En caso contrario, el precio será de 19.487,13 USD.

Para sacar las estadísticas de los resultados del árbol creado utilizamos la siguiente transacción:

```
> printcp(arbol) #Estadisticas de resultados

Regression tree:
rpart(formula = price ~ ., data = data_training, method = "anova")

Variables actually used in tree construction:
[1] carheight  curbweight enginesize horsepower wheelbase

Root node error: 7490231753/125 = 59921854

n= 125

        CP nsplit rel error  xerror     xstd
1 0.613228      0   1.00000 1.00984 0.218271
2 0.227377      1   0.38677 0.47025 0.074197
3 0.023526      2   0.15939 0.24866 0.067870
4 0.019717      3   0.13587 0.24378 0.067165
5 0.011021      4   0.11615 0.22275 0.066465
6 0.010000      5   0.10513 0.21725 0.066085
```

Figura 14: Estadísticas error (Precio_coche)

Muestro el gráfico de la evolución del error a medida que se incrementan los nodos (para ello he utilizado la función `plotcp(arbol)`):

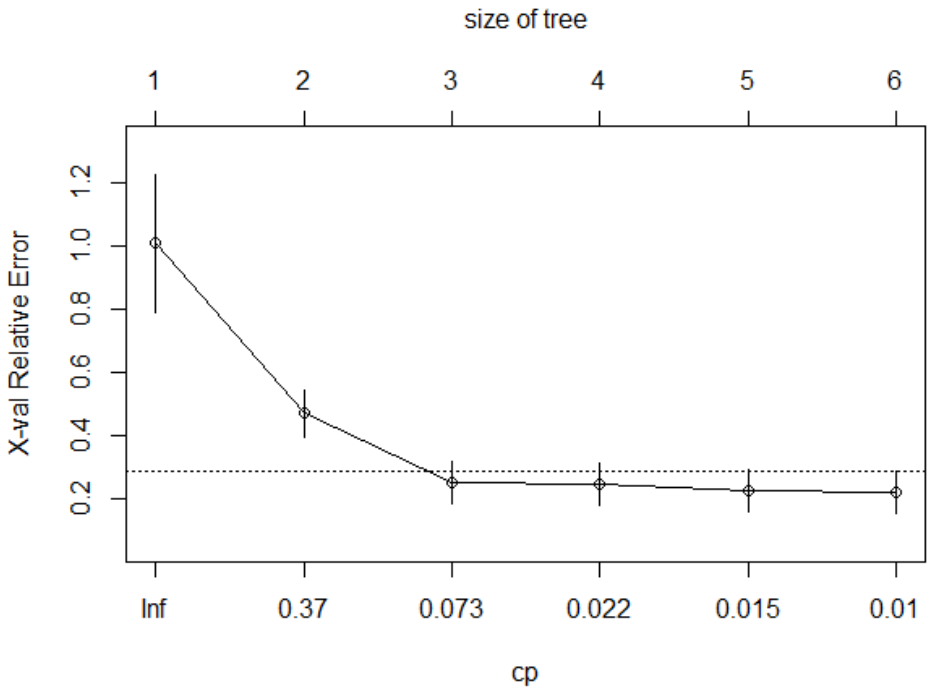


Figura 15: Gráfico del error (Precio_coche)

La composición del gráfico anterior se explicó en el apartado 4.4 poda de árboles.

La mejor elección será el árbol más simple siempre y cuando sea equivalente al error estimado de generalización. En nuestro caso, el árbol con menor error estimado de generalización sería para un tamaño 6, pero elegimos el de tamaño 3 con un $CP = 0{,}073$, porque es de menor tamaño y es equivalente al de tamaño 6, ya su error queda también por debajo de la línea que nos marca las mejores reglas a emplear.

Para ello utilizamos la función `prune` e indicamos el CP donde queremos que se realice el corte del árbol:

```
#Para podar el árbol:

poda_arbol <- prune(arbol, cp=0.073)
```

A continuación, muestro la información que nos aporta el nuevo árbol obtenido tras la poda:

```
> #Mostramos los resultados con el arbol podado:
> print(poda_arbol)
n= 125

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

1) root 125 7490232000 13206.370
  2) enginesize< 182 116 2596206000 11517.890
    4) curbweight< 2544 70  312392900  8411.743 *
    5) curbweight>=2544 46  580705700 16244.630 *
  3) enginesize>=182 9  300804900 34969.000 *
```

Figura 16: División numérica árbol podado (Precio_coche)

Muestro el árbol de regresión resultante:

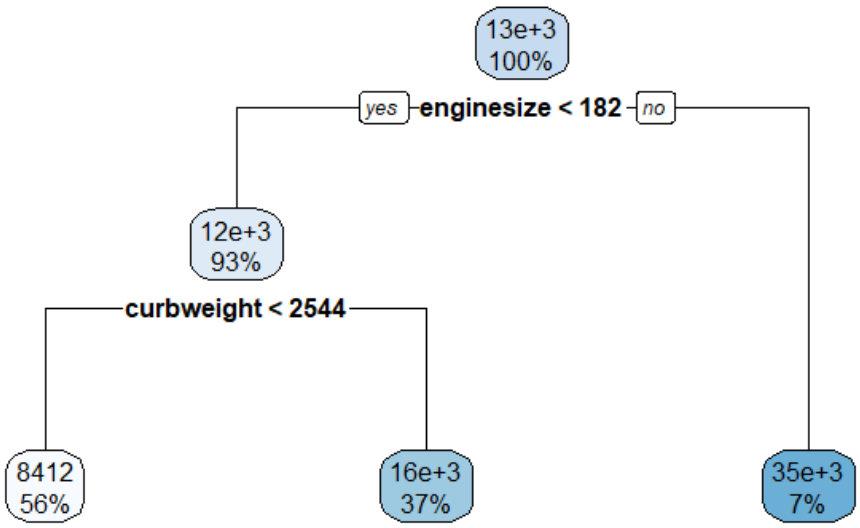


Figura 17: Gráfico del árbol tras poda (Precio_coche)

Como podemos observar, con tan solo dos variables podemos obtener una buena aproximación del precio de un coche reduciendo la complejidad del modelo, pero manteniendo su capacidad de predicción.

Por lo tanto, podemos confirmar que, este modelo podría aplicarse en la industria automovilística en diversos ámbitos. Desde mi punto de vista sería interesante utilizar este modelo para estudiar si los precios de la competencia están por encima o por debajo de la media en función de las características del vehículo

3.7 EJEMPLO ÁRBOL DE CLASIFICACIÓN: DATASET - DIABETES

Para realizar este ejemplo, hemos utilizado un conjunto de datos que busca predecir si un ser humano tendrá o no diabetes. Para ello se ha utilizado el lenguaje de programación R.

En primer lugar, se instalan los paquetes necesarios para la realización del árbol de clasificación:

```
#Para instalar librerías se utiliza:

install.packages("rpart")

install.packages("rpart.plot")

install.packages("ggplot")
```

A continuación, se deben cargar las librerías:

```
#Paso 1: Cargar las librerías y set de datos:

library(rpart) #Algoritmo árbol de decisión

library(rpart.plot) #Graficar árbol de decisión
```

El conjunto de datos original se encuentra en formato CSV como vemos en la imagen siguiente:

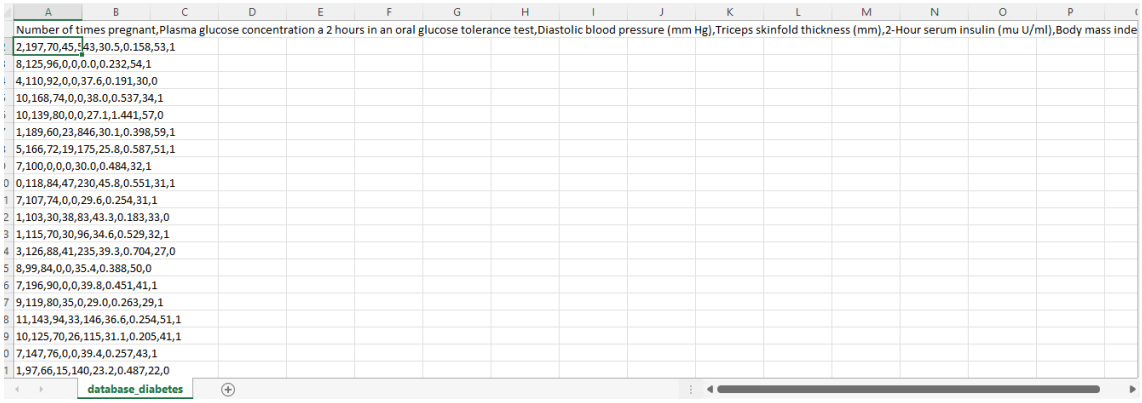
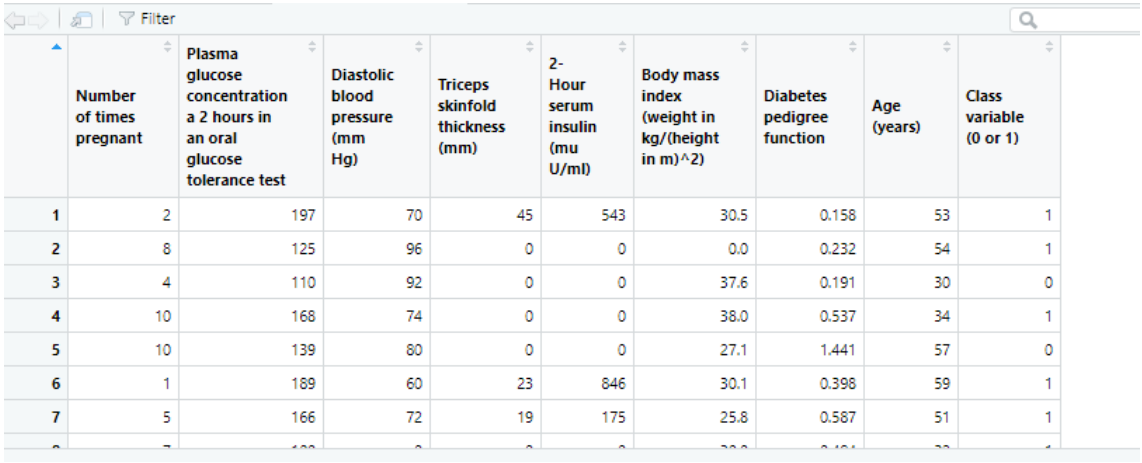


Figura 18: Conjunto de datos CSV (Diabetes)

Y al importarlo a Rstudio:



	Number of times pregnant	Plasma glucose concentration a 2 hours in an oral glucose tolerance test	Diastolic blood pressure (mm Hg)	Triceps skinfold thickness (mm)	2-Hour serum insulin (mu U/ml)	Body mass index (weight in kg/(height in m)^2)	Diabetes pedigree function	Age (years)	Class variable (0 or 1)
1	2	197	70	45	543	30.5	0.158	53	1
2	8	125	96	0	0	0.0	0.232	54	1
3	4	110	92	0	0	37.6	0.191	30	0
4	10	168	74	0	0	38.0	0.537	34	1
5	10	139	80	0	0	27.1	1.441	57	0
6	1	189	60	23	846	30.1	0.398	59	1
7	5	166	72	19	175	25.8	0.587	51	1

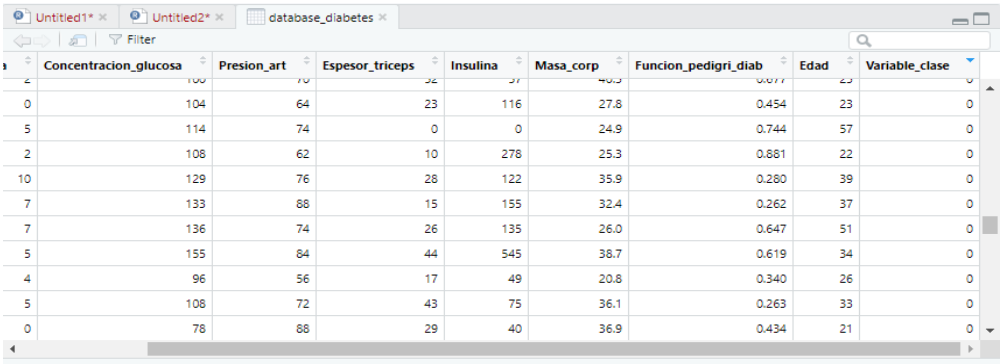
Figura 19: Conjunto de datos Rstudio (Diabetes)

Tras importar el conjunto de datos, lo primero que haremos para la legibilidad del conjunto de datos es traducir el nombre de las variables a español y de esta manera reducir su longitud:

```
#Cambiamos el nombre de las variables:

names(database_diabetes)=c("Veces_embarazada","Concentracion_glucosa",
"Presion_art","Espesor_triceps","Insulina","Masa_corp","Funcion_pedigri_diab","Edad
","Variable_clase")
```

Tras este cambio se muestra así el conjunto de datos en Rstudio:



	Concentracion_glucosa	Presion_art	Espesor_triceps	Insulina	Masa_corp	Funcion_pedigri_diab	Edad	Variable_clase
0	104	64	23	116	27.8	0.454	23	0
5	114	74	0	0	24.9	0.744	57	0
2	108	62	10	278	25.3	0.881	22	0
10	129	76	28	122	35.9	0.280	39	0
7	133	88	15	155	32.4	0.262	37	0
7	136	74	26	135	26.0	0.647	51	0
5	155	84	44	545	38.7	0.619	34	0
4	96	56	17	49	20.8	0.340	26	0
5	108	72	43	75	36.1	0.263	33	0
0	78	88	29	40	36.9	0.434	21	0

Figura 20: Conjunto de datos Rstudio varias en español (Diabetes)

La variable que vamos a predecir es la que hemos denominado Variable_clase (Class variable (0 o 1)) que es la que determinara si una persona tiene (1) o no diabetes (0). El resto de las variables quedan explicadas en la siguiente tabla:

Explicación de las variables	
Number of times pregnant	Número de veces que ha estado embarazada
Plasma glucose concentration 2 hours in an oral glucose tolerance test	Concentración de glucosa en plasma a las 2 horas en una prueba de tolerancia a la glucosa oral
Diastolic blood pressure (mm Hg)	Presión arterial diastólica (mm Hg)
Triceps skinfold thickness (mm)	Espesor del pliegue cutáneo del tríceps (mm)
2-Hour serum insulin (mu U/ml)	Insulina sérica a 2 horas (mu U/ml)
Body mass index (weight in kg/(height in m)^2)	Índice de masa corporal (peso en kg/(altura en m)^2)
Diabetes pedigree function	Probabilidad de tener diabetes según antecedentes familiares
Age (years)	Edad (años)
Class variable (0 or 1)	Variable de clase (0 o 1)

Tabla 3: Explicación de las variables (Diabetes)

El siguiente paso será dividir la base de datos en dos partes: un conjunto de entrenamiento y un conjunto de prueba o validación. El 60% de los datos estarán dedicados al entrenamiento y el otro 40% a la validación. Para ello hemos utilizado la función `createDataPartition`, la cual genera una lista aleatoria de identificadores y a partir de ellos poder obtener el conjunto de entrenamiento y de validación.

```
#Dividimos la base de datos en dos, un 60% para entrenamiento y un 40% para validación:

ind <- createDataPartition(data$`Age (years)`, p=0.6,list=F)

data_training <- data[ind,]

data_validation <- data[-ind,]
```

Una vez dividido el conjunto de datos podemos realizar el árbol de clasificación, para ello utilizamos la función `rpart`.

En primer lugar, introducimos la variable dependiente de nuestro modelo (Variable_clase) la cual vamos a predecir con una selección del resto de variables.

Después indicamos que nuestra variable dependiente es cualitativa e introducimos nuestro conjunto de datos:

```
#Creamos el árbol de decisión:

arbol <- rpart(formula = Variable_clase ~ ., method= "class", data= data_training)
```

Una vez realizado nuestro árbol, imprimimos su información a través de `print(arbol)` y obtenemos lo siguiente:

```
> #Imprimimos el resultado
> print(arbol)
n= 457

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

 1) root 457 155 0 (0.66083151 0.33916849)
   2) Concentracion_glucosa< 144.5 359  86 0 (0.76044568 0.23955432)
     4) Masa_corp< 27.85 122   8 0 (0.93442623 0.06557377) *
     5) Masa_corp>=27.85 237  78 0 (0.67088608 0.32911392)
      10) Concentracion_glucosa< 112.5 128  24 0 (0.81250000 0.18750000) *
      11) Concentracion_glucosa>=112.5 109  54 0 (0.50458716 0.49541284)
        22) Edad< 29.5 52  17 0 (0.67307692 0.32692308)
          44) Espesor_triceps>=5 41  10 0 (0.75609756 0.24390244) *
          45) Espesor_triceps< 5 11   4 1 (0.36363636 0.63636364) *
        23) Edad>=29.5 57  20 1 (0.35087719 0.64912281)
          46) Funcion_pedigri_diab< 0.496 35  16 1 (0.45714286 0.54285714)
            92) Edad< 34.5 9   2 0 (0.77777778 0.22222222) *
            93) Edad>=34.5 26   9 1 (0.34615385 0.65384615) *
          47) Funcion_pedigri_diab>=0.496 22   4 1 (0.18181818 0.81818182) *
   3) Concentracion_glucosa>=144.5 98  29 1 (0.29591837 0.70408163)
     6) Masa_corp< 30 21   9 0 (0.57142857 0.42857143)
      12) Presion_art>=79 7   1 0 (0.85714286 0.14285714) *
      13) Presion_art< 79 14   6 1 (0.42857143 0.57142857) *
     7) Masa_corp>=30 77  17 1 (0.22077922 0.77922078)
      14) Veces_embarazada< 4.5 42  15 1 (0.35714286 0.64285714)
        28) Insulina>=40.5 25  12 0 (0.52000000 0.48000000)
          56) Concentracion_glucosa< 167.5 12   3 0 (0.75000000 0.25000000) *
          57) Concentracion_glucosa>=167.5 13   4 1 (0.30769231 0.69230769) *
        29) Insulina< 40.5 17   2 1 (0.11764706 0.88235294) *
      15) Veces_embarazada>=4.5 35   2 1 (0.05714286 0.94285714) *
```

Figura 21: División numérica árbol (Diabetes)

Como se aprecia en la imagen, de las 457 observaciones se obtienen 155 que representan el 33,9% las cual tendrán diabetes, por el contrario, el 63,5% no tendrán diabetes.

Para imprimir el grafico del árbol:

```
#Para imprimir el gráfico:

rpart.plot(arbol, extra=4) #Con extra=4 saca las probabilidades por clase
```

Que nos genera el siguiente gráfico:

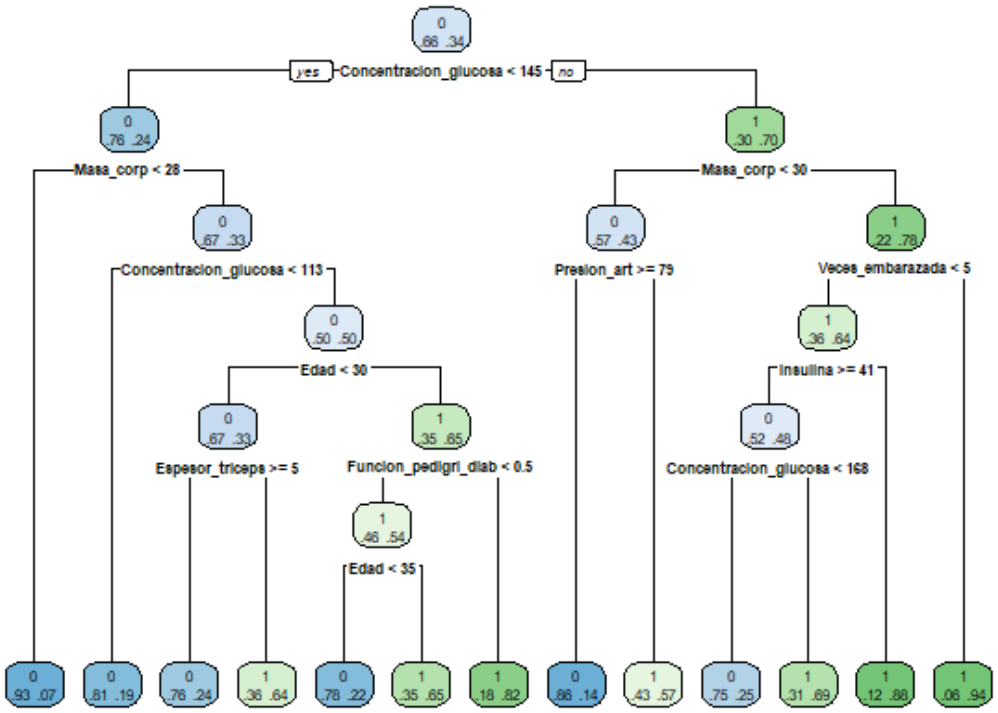


Figura 22: Gráfico del árbol (Diabetes)

A continuación, explicamos el árbol:

La primera variable que toma el modelo para decidir es Concentración_glucosa. Si esta es menor de 144,5 mg/dL entonces el 76% de las personas no tendrán diabetes. Por el contrario, un 70% si tendrán diabetes a partir de una concentración superior a 144,5 mg/dL en sangre. Si continuamos por el lado izquierdo de la división, la siguiente variable que entra en juego es la masa corporal. Si esta es menor de $27{,}85\ Kg^2/m$ llegamos a un nodo final que nos dice que el 93% de las personas no tendrán diabetes. En cambio, si son mayores o iguales a los $27{,}85\ Kg^2/m$, el porcentaje disminuye al 67%.

Para el grupo de personas con una masa corporal mayor o igual a los $27{,}85\ Kg^2/m$, la variable Concentración_glucosa vuelve a entrar en juego para un valor de 112,5 mg/dL. Si la variable es menor de 112,5 mg/dL llegamos a un nodo final en el que obtenemos que el 81% de las personas no tendrán diabetes.

Por el otro lado donde la variable debe ser mayor o igual a 112,5 mg/dL y menor de 144,5 mg/dL, la división del árbol continua con la variable edad. Si esta es menor de 30 años entonces obtenemos que el 67% de las personas no tendrán diabetes. Por el contrario, si es mayor o igual a 30 años el 65% de las personas si tendrán diabetes. En el lado izquierdo de esta última variable edad, obtenemos una nueva división gracias a la variable espesor_triceps la cual si es mayor o igual de 5mm entonces el 76% no tendrán

la enfermedad. Por el otro lado si la condición no se cumple podemos afirmar que el 64% si tendrán la enfermedad.

Volviendo a la división de la variable edad y tomando el camino de la derecha tenemos una nueva partición a través de la variable Función_pedigri_diab. En el caso de que sea mayor o igual de 0,496 entonces llegamos a un nodo final que nos indica que el 82% de las personas tendrán la enfermedad. Si en cambio la condición de que sea menor de 0,496 se cumple nos encontramos con una nueva división por la variable edad, pero en este caso evaluando si la persona es menor de 35 años. Si se cumple la condición esto implicaría que la persona tuviera una edad mayor o igual a los 30 años ya que partimos de esa división anterior y además una edad inferior a los 35 años. Para esta franja de edad podemos afirmar que en un 78% de los casos las personas no padecen la enfermedad. En el caso de que tengan una edad mayor o igual a los 35 años entonces el 65% tendrán diabetes.

Si pasamos a evaluar la primera división que se realizó a través de la variable Concentración_glucosa, el lado derecho del árbol en el que la condición de que sea menor de 144,5 mg/dL no se cumple entonces como dijimos anteriormente en este caso el 70% de las personas tendrán diabetes. A continuación, el modelo realiza una nueva división evaluando de nuevo la masa corporal. Si esta es menor de $30\ Kg^2/m$ entonces el 57% de las personas no tendrán diabetes. En el lado donde se cumple la anterior condición tenemos una nueva variable decisora que es la presión arterial. Si la persona tiene una presión arterial mayor o igual a 79 mmHg entonces podemos afirmar que el 86% de las personas no tendrán diabetes. En cambio, si es menor a 79 mmHg entonces el 57% si la padecerá.

Volviendo a la división de la masa corporal si tomamos el camino en el que esta masa era mayor o igual a los $30\ Kg^2/m$ podemos afirmar que el 78% de las personas tendrán diabetes. Si seguimos por este camino llegamos a una nueva división a través de la variable veces_embarazada si esta es mayor o igual a 5 veces entonces en este caso las mujeres tendrán una probabilidad del 94% de padecer diabetes. Por el contrario, si no se cumple la anterior condición el porcentaje se reduce hasta el 64% de personas que padecerían la enfermedad. En este lado de la decisión el árbol continuó a través de la variable insulina, la cual si es menor a los 40,5 mg/dL entonces el 88% de las personas tendrán diabetes. Por el otro lado, si es mayor o igual a los 40,5 mg/dL , esto nos lleva a una última división utilizando de nuevo la variable Concentración_glucosa. Si esta es menor de 167,5 mg/dL y a la vez es mayor o igual de 144,5 mg/dL entonces podemos afirmar que el 75% de las personas no tendrán la enfermedad. Por el contrario, si la concentración en glucosa es mayor de los 167,5 mg/dL entonces el 69% de las personas si tendrán diabetes.

Para sacar las estadísticas de los resultados del árbol creado utilizamos la siguiente transacción:

```
> printcp(arbol) #Estadisticas de resultados

Classification tree:
rpart(formula = Variable_clase ~ ., data = data_training, method = "class")

Variables actually used in tree construction:
[1] Concentracion_glucosa Edad                  Espesor_triceps
[4] Funcion_pedigri_diab  Insulina              Masa_corp
[7] Presion_art           Veces_embarazada

Root node error: 155/457 = 0.33917

n= 457

        CP nsplit rel error  xerror     xstd
1 0.258065      0   1.00000 1.00000 0.065295
2 0.036559      1   0.74194 0.77419 0.060690
3 0.019355      4   0.63226 0.73548 0.059677
4 0.016129      6   0.59355 0.76129 0.060360
5 0.012903      8   0.56129 0.76129 0.060360
6 0.010000     12   0.50968 0.80000 0.061326
```

Figura 23: Estadísticas error (Diabetes)

Muestro el gráfico de la evolución del error a medida que se incrementan los nodos (para ello he utilizado la transacción `plotcp(arbol)`):

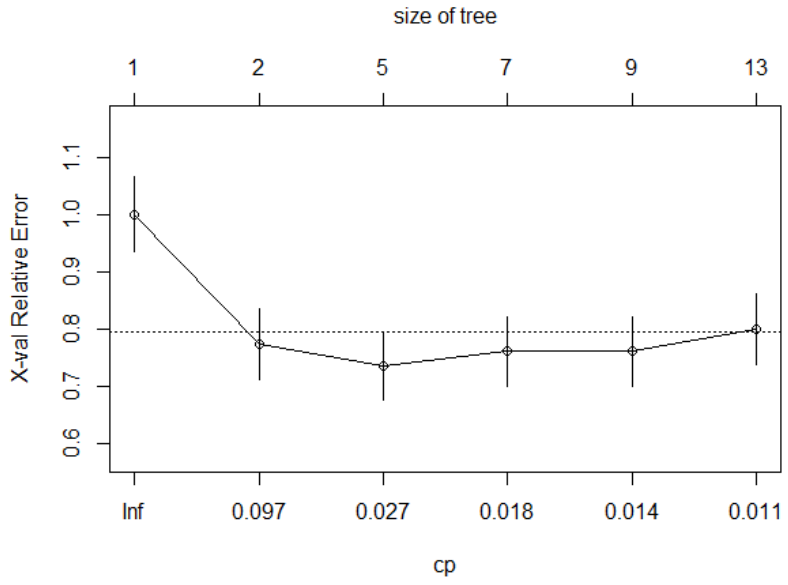


Figura 24: Gráfico del error (Diabetes)

La composición del gráfico anterior se explicó en el apartado 4.4 poda de árboles.

La mejor elección será el árbol más simple siempre y cuando sea equivalente al error estimado de generalización. En nuestro caso, el árbol con menor error estimado de generalización sería para un tamaño 2 con un $CP = 0{,}097$, elegimos este porque es de menor tamaño, aun siendo equivalente al de tamaño 9 ya su error queda también por debajo de la línea que nos marca la regla empleada.

Para ello utilizamos la función `prune` e indicamos el CP donde queremos que se realice el corte del árbol:

```
#Para podar el árbol:

poda_arbol <- prune(arbol, cp= 0.097)
```

Muestro el árbol de clasificación resultante:

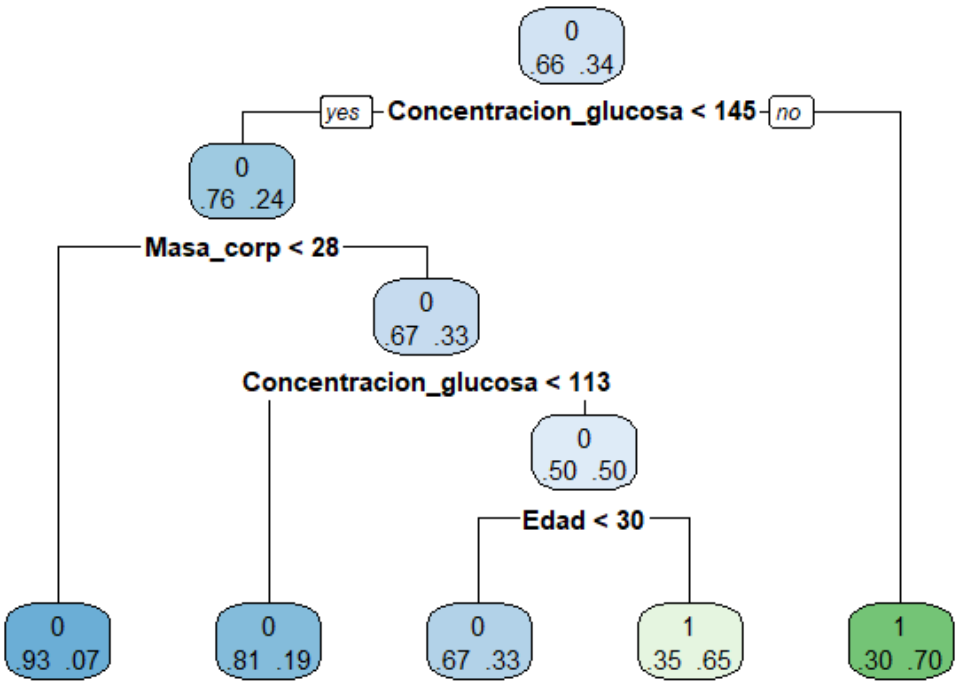


Figura 25: Gráfico del árbol tras poda (Diabetes)

A través de la poda se puede apreciar que se ha eliminado el camino para las personas con una concentración en glucosa mayor o igual a 144,5 mg/dL, siendo este ahora un nodo final y determinando que el 70% de las personas no tendrán diabetes si se cumple esta condición. El camino para las personas con una concentración menor de 144,5 mg/dL es igual que lo visto antes de la poda, pero elimina también las decisiones posteriores a la variable edad.

Seguidamente mostramos la capacidad de predicción de nuestro modelo a través del siguiente código:

```
#Para ver como predice nuestro modelo:

testpredArbol <- predict(arbol, newdata = data_validation, type = "class")
```

Y lo mostramos a través de la siguiente tabla:

```
> #Visualizamos la matriz de confusion:
> table(testpredArbol,data_validation$variable_clase)

testpredArbol   0   1
            0 160  40
            1  34  69
```

Figura 26: Matriz de confusión (Diabetes)

La cual indica que predice correctamente que 160 casos no tenían diabetes y en decir que sí en 69. En cambio, el modelo no predice correctamente 40 casos ya que indica que los sujetos no padecen la enfermedad cuando en realidad sí lo hacen y predice lo contrario para 34 sujetos.

Por último, vamos a evaluar la efectividad del modelo. Para ello sumo todos los aciertos y lo dividimos entre el número de predicciones:

```
> #Calculamos la efectividad del modelo:
> sum(testpredArbol == data_validation$variable_clase) / length(data_validation$variable_clase)*100
[1] 75.57756
```

Figura 27: Porcentaje de efectividad (Diabetes)

Obteniendo una efectividad del 75,58%.

Por último, para comprobar la validez del modelo vamos a utilizar el método de las Curvas ROC explicadas anteriormente. Para ello necesitamos instalar y ejecutar la siguiente librería:

```
#Para realizar la curva ROC

install.packages("pROC")

library(pROC)
```

A continuación, para poder obtener el gráfico de la curva de ROC necesitamos realizar una predicción, pero en este caso, dicha predicción se realiza a través de la probabilidad:

```
#Realizamos una nueva predicción pero en este caso sobre la probabilidad

testpredArbol_prob <- predict(arbol, newdata = data_validation, type = "prob")
```

Posteriormente, podemos a través de la función `roc` realizar el gráfico y mostrarlo gracias a la función `plot`:

```
#Realizo la curva ROC

roc.arbol <- roc(data_validation$Variable_clase, testpredArbol_prob[,2])

plot(roc.arbol)
```

El gráfico generado se muestra a continuación:

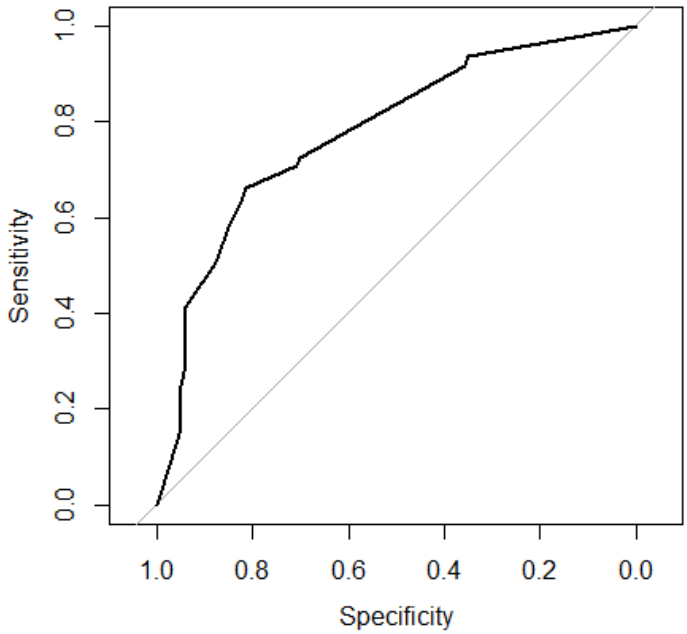


Figura 28: Curva ROC clasificación (Diabetes)

Para poder ver el área bajo la curva necesitamos ejecutar el siguiente código:

```
#Muestro el área bajo la curva el cual cuanto más cercano a uno es mejor:

roc.arbol$auc
```

Y obtenemos lo siguiente:

```
> roc.arbol$auc
Area under the curve: 0.7771
```

Figura 29: Área bajo la curva clasificación (Diabetes)

La curva ROC nos ayuda a determinar la capacidad de clasificación de nuestro modelo entre dos clases. En este caso si un paciente tiene o no diabetes.

Concretamente, el resultado anterior significa que nuestro modelo tiene un 78% de probabilidad de distinguir entre pacientes con diabetes o sin ella.

Como podemos observar, este modelo nos ayuda a determinar en función de una serie de variables sencillas de obtener de un paciente si este tendrá o no diabetes.

3.8 EJEMPLO ÁRBOL DE CLASIFICACIÓN: DATASET - BANCARROTA

Para realizar este ejemplo, hemos utilizado un conjunto de datos el cual busca predecir si una compañía entrara en bancarrota o no. Para ello se ha utilizado el mismo programa que en el caso anterior `Rstudio` y el lenguaje de programación R.

En primer lugar, se instalan los paquetes necesarios para la realización del árbol de clasificación:

```
#Para instalar librerías se utiliza:

install.packages("rpart")

install.packages("rpart.plot")

install.packages("ggplot")
```

A continuación, se deben cargar las librerías que vamos a utilizar:

```
#Paso 1: Cargar las librerías y set de datos:

library(rpart) #Algoritmo árbol de decisión

library(rpart.plot) #Graficar árbol de decisión
```

El conjunto de datos tiene el siguiente aspecto en CSV:

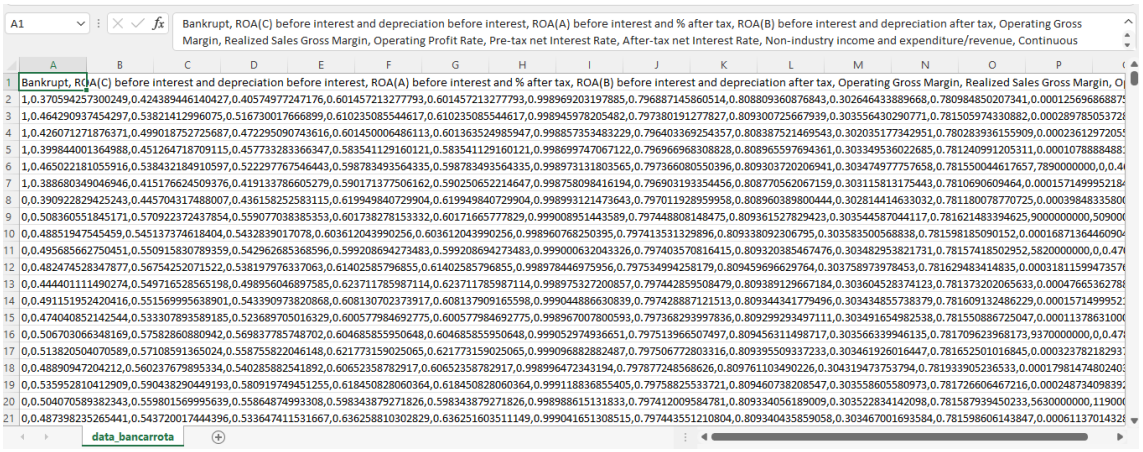
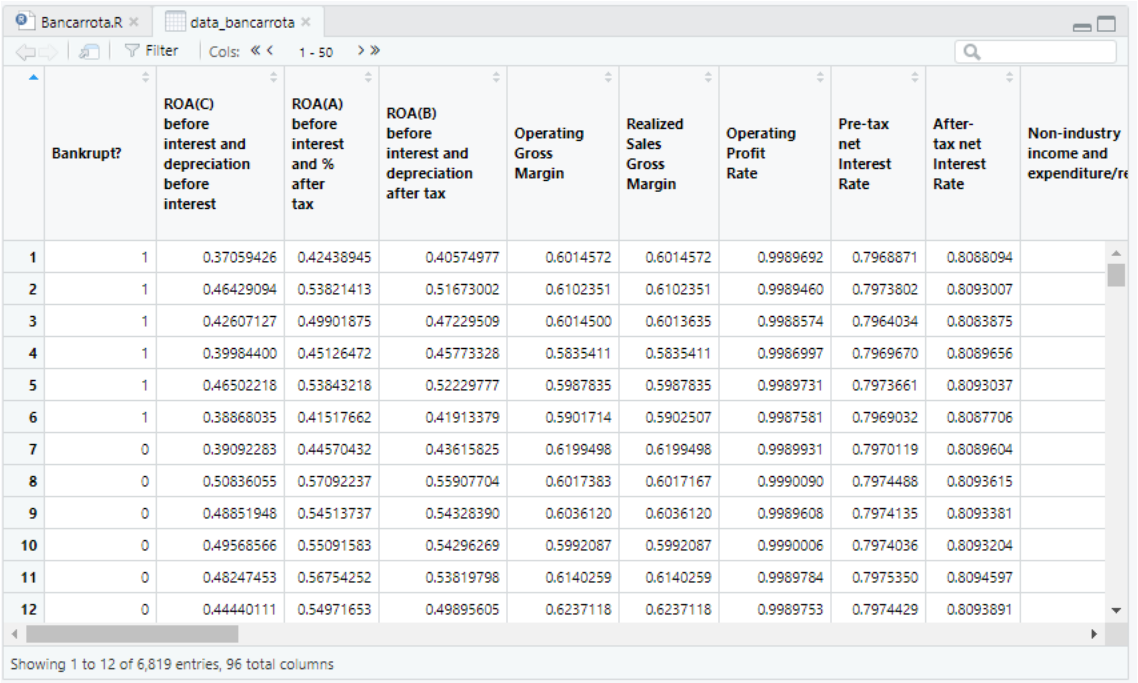


Figura 30: Conjunto de datos CSV (Bancarrota)

Si lo importamos a Rstudio:



Bancarrota.R | data_bancarrota

Filter | Cols: « < 1 - 50 > »

	Bankrupt?	ROA(C) before interest and depreciation before interest	ROA(A) before interest and % after tax	ROA(B) before interest and depreciation after tax	Operating Gross Margin	Realized Sales Gross Margin	Operating Profit Rate	Pre-tax net Interest Rate	After-tax net Interest Rate	Non-industry income and expenditure/re
1	1	0.37059426	0.42438945	0.40574977	0.6014572	0.6014572	0.9989692	0.7968871	0.8088094	
2	1	0.46429094	0.53821413	0.51673002	0.6102351	0.6102351	0.9989460	0.7973802	0.8093007	
3	1	0.42607127	0.49901875	0.47229509	0.6014500	0.6013635	0.9988574	0.7964034	0.8083875	
4	1	0.39984400	0.45126472	0.45773328	0.5835411	0.5835411	0.9986997	0.7969670	0.8089656	
5	1	0.46502218	0.53843218	0.52229777	0.5987835	0.5987835	0.9989731	0.7973661	0.8093037	
6	1	0.38868035	0.41517662	0.41913379	0.5901714	0.5902507	0.9987581	0.7969032	0.8087706	
7	0	0.39092283	0.44570432	0.43615825	0.6199498	0.6199498	0.9989931	0.7970119	0.8089604	
8	0	0.50836055	0.57092237	0.55907704	0.6017383	0.6017167	0.9990090	0.7974488	0.8093615	
9	0	0.48851948	0.54513737	0.54328390	0.6036120	0.6036120	0.9989608	0.7974135	0.8093381	
10	0	0.49568566	0.55091583	0.54296269	0.5992087	0.5992087	0.9990006	0.7974036	0.8093204	
11	0	0.48247453	0.56754252	0.53819798	0.6140259	0.6140259	0.9989784	0.7975350	0.8094597	
12	0	0.44440111	0.54971653	0.49895605	0.6237118	0.6237118	0.9989753	0.7974429	0.8093891	

Showing 1 to 12 of 6,819 entries, 96 total columns

Figura 31: Conjunto de datos Rstudio (Bancarrota)

A continuación, explicamos las variables del conjunto de datos. La variable binaria "Bankrupt" (Bancarrota) determinará si la empresa entra o no en bancarrota. El resto de las variables vienen explicadas en la siguiente tabla:

Explicación de las variables.	
Bankrupt	Bancarrota
ROA(C) before interest and depreciation before interest	ROA(C) antes de intereses y depreciación antes de intereses
ROA(A) before interest and % after tax	ROA(A) antes de intereses y % después de impuestos
ROA(B) before interest and depreciation after tax	ROA(B) antes de intereses y amortizaciones después de impuestos
Operating Gross Margin	Margen bruto de explotación
Realized Sales Gross Margin	Margen bruto de ventas realizadas
Operating Profit Rate	Tasa de beneficio de explotación
Pre-tax net Interest Rate	Tipo de interés neto antes de impuestos
After-tax net Interest Rate	Tipo de interés neto después de impuestos
Non-industry income and expenditure/revenue	Ingresos y gastos no industriales/ingresos
Continuous interest rate (after tax)	Tasa de interés continua (después de impuestos)
Operating Expense Rate	Tasa de gastos de explotación
Research and development expense rate	Tasa de gastos de investigación y desarrollo
Cash flow rate	Tipo de flujo de caja

Interest-bearing debt interest rate	Tipo de interés de la deuda con intereses
Tax rate (A)	Tipo impositivo (A)
Net Value Per Share (B)	Valor neto por acción (B)
Net Value Per Share (A)	Valor neto por acción (A)
Net Value Per Share (C)	Valor neto por acción (C)
Persistent EPS in the Last Four Seasons	BPA persistente en las últimas cuatro temporadas
Cash Flow Per Share	Flujo de caja por acción
Revenue Per Share (Yuan ¥)	Ingresos por acción (Yuan ¥)
Operating Profit Per Share (Yuan ¥)	Beneficio de explotación por acción (Yuan ¥)
Per Share Net profit before tax (Yuan¥)	Beneficio neto por acción antes de impuestos (Yuan ¥)
Realized Sales Gross Profit Growth Rate	Tasa de crecimiento del beneficio bruto de las ventas realizadas
Operating Profit Growth Rate	Tasa de crecimiento de los beneficios de explotación
After-tax Net Profit Growth Rate	Tasa de crecimiento del beneficio neto después de impuestos
Regular Net Profit Growth Rate	Tasa de crecimiento del beneficio neto regular
Continuous Net Profit Growth Rate	Tasa de crecimiento del beneficio neto continuo
Total Asset Growth Rate	Tasa de crecimiento del activo total
Net Value Growth Rate	Tasa de crecimiento del valor neto
Total Asset Return Growth Rate Ratio	Tasa de crecimiento del rendimiento total de los activos
Cash Reinvestment %	Reinversión de efectivo %
Current Ratio	Ratio de corriente
Quick Ratio	Ratio de rapidez
Interest Expense Ratio	Ratio de Gastos de Interés
Total debt/Total net worth	Deuda total/Patrimonio neto total
Debt ratio %	Ratio de endeudamiento
Net worth/Assets	Patrimonio neto/Activo
Long-term fund suitability ratio (A)	Ratio de adecuación de fondos a largo plazo (A)
Borrowing dependency	Dependencia del endeudamiento
Contingent liabilities/Net worth	Pasivo contingente/Patrimonio neto
Operating profit/Paid-in capital	Beneficio de explotación/Capital desembolsado
Net profit before tax/Paid-in capital	Beneficio neto antes de impuestos/Capital desembolsado
Inventory and accounts receivable/Net value	Inventario y cuentas por cobrar/Valor neto
Total Asset Turnover	Volumen de negocios del activo total
Accounts Receivable Turnover	Volumen de negocios de las cuentas por cobrar

Average Collection Days	Días medios de cobro
Inventory Turnover Rate (times)	Índice de rotación de existencias (veces)
Fixed Assets Turnover Frequency	Frecuencia de rotación del activo fijo
Net Worth Turnover Rate (times)	Índice de rotación del patrimonio neto (veces)
Revenue per person	Ingresos por persona
Operating profit per person	Beneficio de explotación por persona
Allocation rate per person	Tasa de asignación por persona
Working Capital to Total Assets	Capital circulante/activo total
Quick Assets/Total Assets	Activo Rápido/Activo Total
Current Assets/Total Assets	Activo corriente/Activo total
Cash/Total Assets	Efectivo/Activo total
Quick Assets/Current Liability	Activo Rápido/Pasivo Corriente
Cash/Current Liability	Efectivo/Pasivo Corriente
Current Liability to Assets	Pasivo corriente/activo
Operating Funds to Liability	Fondos de explotación a pasivo
Inventory/Working Capital	Inventario/Capital de trabajo
Inventory/Current Liability	Inventario/Pasivo corriente
Current Liabilities/Liability	Pasivo corriente/Pasivo
Working Capital/Equity	Capital circulante/patrimonio neto
Current Liabilities/Equity	Pasivo corriente/Patrimonio neto
Long-term Liability to Current Assets	Pasivo a largo plazo/activo corriente
Retained Earnings to Total Assets	Ganancias retenidas a activos totales
Total income/Total expense	Ingresos totales/Gastos totales
Total expense/Assets	Gasto total/Activo
Current Asset Turnover Rate	Tasa de rotación del activo corriente
Quick Asset Turnover Rate	Tasa de rotación del activo rápido
Working capitcal Turnover Rate	Índice de rotación del capital circulante
Cash Turnover Rate	Tasa de rotación del efectivo
Cash Flow to Sales	Flujo de caja a ventas
Fixed Assets to Assets	Activos fijos a activos
Current Liability to Liability	Pasivo Corriente a Pasivo
Current Liability to Equity	Pasivo corriente a patrimonio neto
Equity to Long-term Liability	Patrimonio neto a pasivo a largo plazo
Cash Flow to Total Assets	Flujo de caja a activos totales
Cash Flow to Liability	Flujo de caja a pasivo
CFO to Assets	CFO a Activos
Cash Flow to Equity	Flujo de caja a patrimonio neto
Current Liability to Current Assets	Pasivo Corriente a Activo Corriente
Liability-Assets Flag	Indicador de Pasivo-Activo
Net Income to Total Assets	Ingresos netos a activos totales
Total assets to GNP price	Precio de los activos totales al PNB
No-credit Interval	Intervalo sin crédito
Gross Profit to Sales	Beneficio bruto a ventas
Net Income to Stockholder's Equity	Ingresos netos a fondos propios

Liability to Equity	Pasivo a Fondos Propios
Degree of Financial Leverage (DFL)	Grado de apalancamiento financiero (DFL)
Interest Coverage Ratio (Interest expense to EBIT)	Ratio de Cobertura de Intereses (Gastos de intereses a EBIT)
Net Income Flag	Indicador de ingresos netos
Equity to Liability	Patrimonio neto en relación con el pasivo

Tabla 4: Explicación de las variables (Bancarrota)

El siguiente paso será dividir la base de datos en dos partes, un 60% de los datos estarán dedicados al entrenamiento y el otro 40% a la validación. Para ello hemos utilizado la función `createDataPartition`, la cual genera una lista aleatoria de identificadores y a partir de ellos se puede obtener el conjunto de entrenamiento y de validación.

```
#Dividimos la base de datos en dos, un 60% para entrenamiento y un 40% para validación:

ind <- createDataPartition(data$`Bankrupt`, p=0.6,list=F)

data_training <- data[ind,]

data_validation <- data[-ind,]
```

Tras realizar la división del conjunto de datos podemos realizar el árbol de clasificación, para ello utilizamos la función `rpart`.

En primer lugar, introducimos la variable dependiente de nuestro modelo (Bankrupt) la cual vamos a predecir con una selección del resto de variables.

Después indicamos que nuestra variable dependiente es cualitativa e introducimos nuestro conjunto de datos:

```
#Creamos el árbol de decisión:

arbol <- rpart(formula = Bankrupt ~ ., method= "class", data= data_training)
```

Una vez realizado nuestro árbol, imprimimos su información a través de print(árbol) y obtenemos lo siguiente:

```
> print(arbol)
n= 4092

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

 1) root 4092 132 0 (0.967741935 0.032258065)
   2) Net Value Growth Rate>=0.0003670415 3984  85 0 (0.978664659 0.021335341)
     4) Borrowing dependency< 0.3794911 3520  32 0 (0.990909091 0.009090909) *
     5) Borrowing dependency>=0.3794911 464  53 0 (0.885775862 0.114224138)
      10) ROA(C) before interest and depreciation before interest>=0.4707259 324  14 0 (0.956790123 0.043209877) *
      11) ROA(C) before interest and depreciation before interest< 0.4707259 140  39 0 (0.721428571 0.278571429)
        22) Interest-bearing debt interest rate< 0.0004480448 81   8 0 (0.901234568 0.098765432) *
        23) Interest-bearing debt interest rate>=0.0004480448 59  28 1 (0.474576271 0.525423729)
          46) Fixed Assets Turnover Frequency< 0.0002163584 18   4 0 (0.777777778 0.222222222) *
          47) Fixed Assets Turnover Frequency>=0.0002163584 41  14 1 (0.341463415 0.658536585)
            94) Operating Profit Growth Rate< 0.847758 8   1 0 (0.875000000 0.125000000) *
            95) Operating Profit Growth Rate>=0.847758 33   7 1 (0.212121212 0.787878788) *
   3) Net Value Growth Rate< 0.0003670415 108  47 0 (0.564814815 0.435185185)
     6) Cash/Total Assets>=0.01304279 89  29 0 (0.674157303 0.325842697)
      12) Net profit before tax/Paid-in capital>=0.1063268 77  17 0 (0.779220779 0.220779221)
        24) Operating profit per person>=0.374624 52   4 0 (0.923076923 0.076923077) *
        25) Operating profit per person< 0.374624 25  12 1 (0.480000000 0.520000000)
          50) Research and development expense rate< 4.135e+08 15   4 0 (0.733333333 0.266666667) *
          51) Research and development expense rate>=4.135e+08 10   1 1 (0.100000000 0.900000000) *
      13) Net profit before tax/Paid-in capital< 0.1063268 12   0 1 (0.000000000 1.000000000) *
     7) Cash/Total Assets< 0.01304279 19   1 1 (0.052631579 0.947368421) *
```

Figura 32: División numérica árbol (Bancarrota)

Como se aprecia en la imagen, de las 4.092 observaciones se obtienen 132 con un valor 1, es decir, las que sí entraran en bancarrota representan el 3,22%. Por el contrario, el 96,78% no entrarán en bancarrota. Además, podemos apreciar que el algoritmo ha determinado que el mejor corte que se podía realizar es a través de la variable Tasa de crecimiento del valor neto (Net Value Growth Rate) evaluando si esta es mayor o igual a 0,00036 o no. Si es mayor o igual entonces implica que el 98% de las compañías no entran en bancarrota. Esta división divide al árbol en dos grandes partes que explicaremos a continuación.

Para imprimir el gráfico del árbol, utilizamos el siguiente código:

```
#Para imprimir el gráfico:

rpart.plot(arbol, extra=4) #Con extra=4 saca las probabilidades por clase
```

Que nos genera el siguiente gráfico:

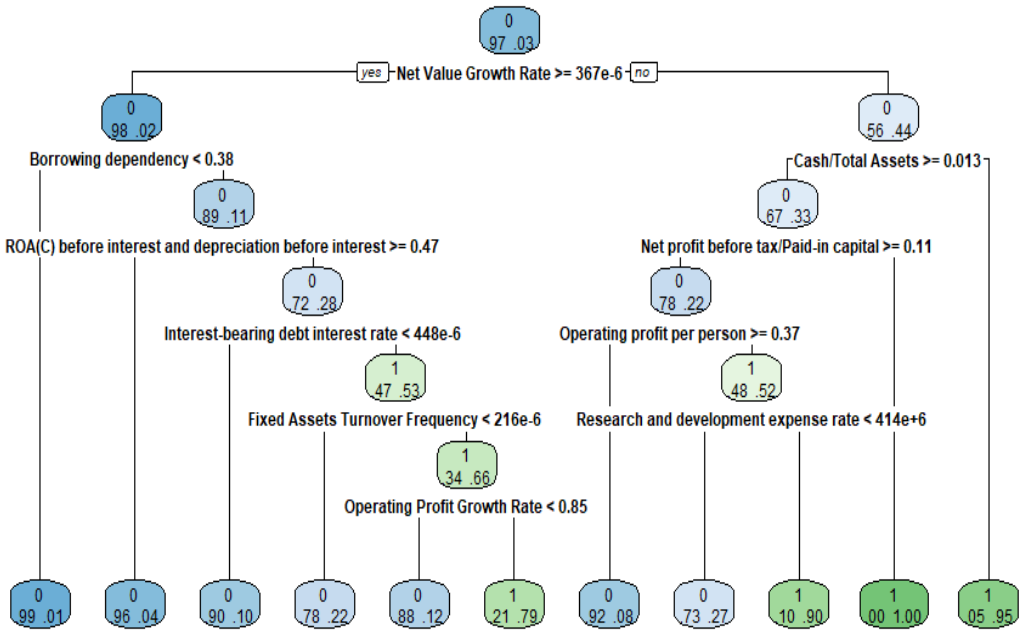


Figura 33: Árbol de clasificación (Bancarrota)

Cabe destacar que este árbol está formado por solo 10 de las 96 variables que tenía nuestra base de datos, eso se debe a que la función `rpart` realiza un proceso de optimización para encontrar el árbol óptimo posible basándose en las variables disponibles.

A continuación, procedemos con la explicación del árbol. Como hemos mencionado anteriormente, la primera división se realiza con la variable Tasa de crecimiento del valor neto (Net Value Growth Rate), la cual nos divide al árbol en dos grandes partes. Si continuamos con el árbol por el lado donde se cumple la condición (mayor o igual que 0,00036), la siguiente división se obtiene mediante la variable Dependencia del endeudamiento (Borrowing dependency), la cual evalúa si es menor de 0,38. Si esto se cumple nos lleva a un nodo final en el que el 99% de las empresas no entran en bancarrota, por el contrario, si esto no se cumple solo podríamos afirmar que el 89% de las empresas no entrarían en bancarrota. Se sigue con la división del árbol si no se cumple la anterior condición (Borrowing dependency) a través de la variable ROA(C) antes de intereses y depreciación antes de intereses (ROA(C) before interest and depreciation before interest) la cual evalúa los valores mayores o iguales a 0,47. Si esto se cumple, el árbol finaliza indicando que el 96% de compañías no entran en bancarrota, si esto no se cumple entonces serán solo el 72% de las empresas las que no entren en bancarrota.

La siguiente variable decisora es la deuda financiera neta (Interest-bearing debt interest rate) la cual si es menor de 0,0004 termina el árbol indicando que el 90% de empresas

no entrarán en bancarrota. Si esta afirmación no se cumple entonces solo podemos afirmar que el 47% de las empresas no entran en bancarrota.

La siguiente división se obtiene al no cumplirse la anterior condición, y la nueva división se hace a través de la variable Frecuencia de rotación del activo fijo (Fixed Assets Turnover Frequency), la cual si es menor de 0,0002 entonces indica que el 78% de las compañías no entran en bancarrota. Por el otro lado, si la condición no se cumple entonces el gráfico muestra que el 66% de las empresas si entran en bancarrota. Si la condición anterior se cumple entonces tenemos una nueva división a través de la variable Tasa de crecimiento de los beneficios de explotación (Operating Profit Growth Rate) la cual nos genera dos nodos finales. Si esta variable es menor de 0,847 entonces podemos afirmar que el 88% de las empresas no entraran en bancarrota en cambio si no se cumple la condición entonces podemos afirmar que el 79% de las empresas sí que entran en bancarrota.

A continuación, vamos a evaluar qué es lo que ocurre cuando vamos por el lado derecho del árbol, es decir, cuando no se cumple la condición de la variable Tasa de crecimiento del valor neto (Net Value Growth Rate). En este caso nos muestra que el 56% de las empresas no entran en bancarrota. Después de esta variable, la siguiente partición la realiza con Efectivo/Activo total (Cash/Total Assets) en la cual si es menor a 0,013 genera un nodo final en el que se puede comprobar que el 95% de las empresas entraran en bancarrota. Si procedemos por la rama de la izquierda, continuamos con la variable Beneficio neto antes de impuestos/Capital desembolsado (Net profit before tax/Paid-in capital) si esta no cumple la condición de ser mayor o igual a 0. 106 genera un nodo final en el que se puede comprobar que el 100% de las empresas entrarán en bancarrota. Por contra si es menor de 0,106 obtenemos que el 78% no entraran en bancarrota. Además, esta última decisión genera una nueva rama a través de la variable Beneficio de explotación por persona (Operating profit per person). Si esta variable es mayor o igual a 0,37 entonces el 92% de las empresas no entraran en bancarrota. En cambio, si esta condición no se cumple el porcentaje de empresas que no entra en bancarrota disminuye hasta el 48%. En este lado se forma una última división a través de la variable Tasa de gastos de investigación y desarrollo (Research and development expense rate) la cual, si es menor 4,135e+08 entonces el modelo predice que el 73% de las empresas no entraran en bancarrota, pero si esto no se cumple entraran en bancarrota el 90%.

Para sacar las estadísticas de los resultados del árbol creado utilizamos la siguiente función (`printcp(arbol)`):

```
> printcp(arbol) #Estadisticas de resultados

Classification tree:
rpart(formula = Bankrupt ~ ., data = data_training, method = "class")

Variables actually used in tree construction:
 [1] Borrowing dependency
 [2] Cash/Total Assets
 [3] Fixed Assets Turnover Frequency
 [4] Interest-bearing debt interest rate
 [5] Net profit before tax/Paid-in capital
 [6] Net Value Growth Rate
 [7] Operating Profit Growth Rate
 [8] Operating profit per person
 [9] Research and development expense rate
[10] ROA(C) before interest and depreciation before interest

Root node error: 132/4092 = 0.032258

n= 4092

        CP nsplit rel error  xerror     xstd
1 0.064394      0   1.00000 1.00000 0.085623
2 0.030303      3   0.78030 0.91667 0.082092
3 0.024621      5   0.71970 0.96970 0.084359
4 0.010000     10   0.57576 0.96212 0.084039
```

Figura 34: Estadísticas error (Bancarrota)

Muestro el gráfico de la evolución del error a medida que se incrementan los nodos (para ello he utilizado la función `plotcp(árbol)`):

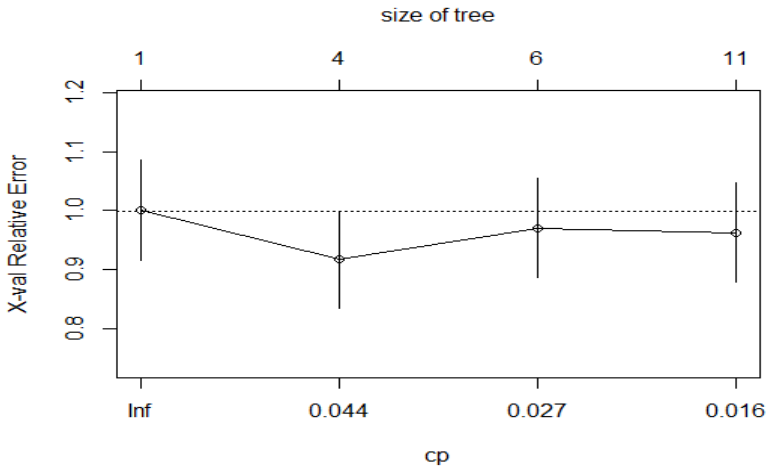


Figura 35: Gráfico del error (Bancarrota)

La composición del gráfico anterior se explicó en el apartado 4.4 poda de árboles.

La mejor elección será el árbol más simple siempre y cuando sea equivalente al error estimado de generalización. En nuestro caso, el árbol con menor error estimado de generalización sería para un tamaño 4 con un $CP = 0{,}044$, elegimos este porque es de menor tamaño, aun siendo equivalente al de tamaño 11 ya que su error queda también por debajo de la línea que nos marca la regla empleada.

A continuación, vamos a realizar la poda del árbol a partir del punto donde comenzó a aumentar el error. Para ello utilizamos la función `prune` e indicamos el CP donde queremos que se realice el corte del árbol:

```
#Para podar el árbol:

poda_arbol <- prune(arbol, cp= 0.044)
```

Ahora muestro los cortes que ha realizado el nuevo árbol podado:

```
> print(poda_arbol)
n= 4092

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

 1) root 4092 132 0 (0.96774194 0.03225806)
   2) Net Value Growth Rate>=0.0003670415 3984  85 0 (0.97866466 0.02133534) *
   3) Net Value Growth Rate< 0.0003670415 108  47 0 (0.56481481 0.43518519)
     6) Cash/Total Assets>=0.01304279 89  29 0 (0.67415730 0.32584270)
      12) Net profit before tax/Paid-in capital>=0.1063268 77  17 0 (0.77922078 0.22077922)
        24) Operating profit per person>=0.374624 52   4 0 (0.92307692 0.07692308) *
        25) Operating profit per person< 0.374624 25  12 1 (0.48000000 0.52000000)
          50) Research and development expense rate< 4.135e+08 15   4 0 (0.73333333 0.26666667) *
          51) Research and development expense rate>=4.135e+08 10   1 1 (0.10000000 0.90000000) *
      13) Net profit before tax/Paid-in capital< 0.1063268 12   0 1 (0.00000000 1.00000000) *
     7) Cash/Total Assets< 0.01304279 19   1 1 (0.05263158 0.94736842) *
```

Figura 36: División numérica tras poda (Bancarrota)

Se puede apreciar que para este caso se obtiene solamente la parte derecha del árbol primitivo.

Lo que nos genera un nuevo árbol de clasificación:

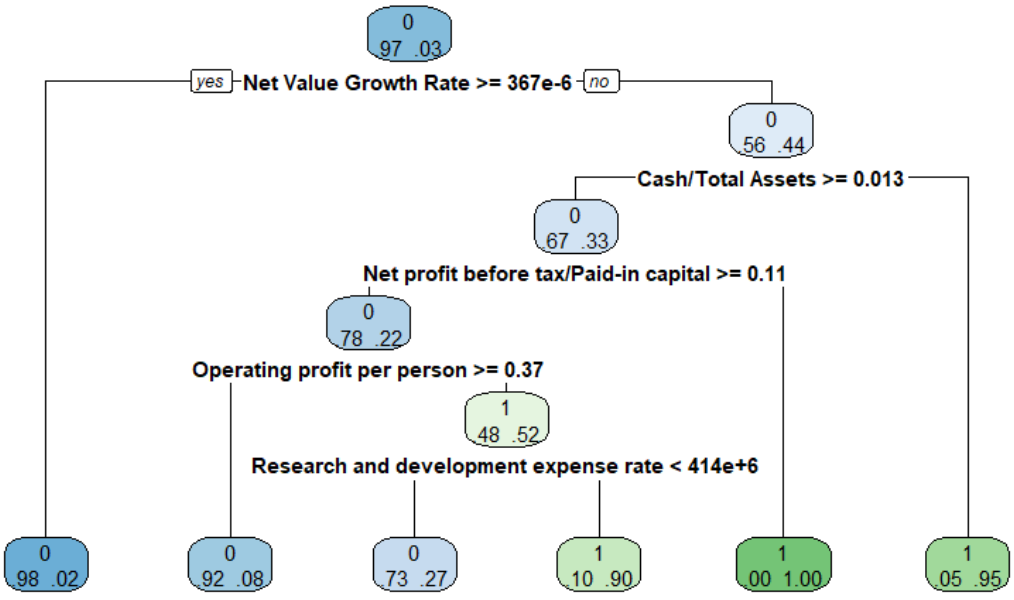


Figura 37: Gráfico del árbol tras poda (Bancarrota)

Se puede apreciar lo explicado anteriormente.

Una vez el modelo ha sido entrenado, ahora puede ser usado para predecir nuevas instancias en el conjunto de datos de validación (data_validation). Para ello utilizo la siguiente función:

```
#Para ver como predice nuestro modelo:

testpredArbol <- predict(arbol, newdata = data_validation, type = "class")
```

Y lo mostramos a través de la siguiente tabla:

```
> #Visualizamos la matriz de confusion:
> table(testpredArbol,data_validation$Bankrupt)

testpredArbol    0    1
            0 2613   64
            1   26   24
```

Figura 38: Matriz de confusión (Bancarrota)

La cual indica que ha predicho correctamente en 2.613 observaciones al predecir que dichas empresas no entrarían en bancarrota y no han entrado. También ha predicho correctamente que 24 empresas si entraran en bancarrota. Por otro lado, se aprecia que se equivoca en 64 empresas al predecir que dichas empresas no entran en bancarrota cuando en verdad si lo hacen. Y en 24 ocasiones se ha equivocado prediciendo que una empresa entra en bancarrota cuando en realidad no lo hace.

Por último, vamos a evaluar la efectividad del modelo para ello sumo todos los aciertos y lo dividimos entre el número de predicciones:

```
> #Calculamos la efectividad del modelo:
> sum(testpredArbol == data_validation$Bankrupt) / length(data_validation$Bankrupt)*100
[1] 96.69967
```

Figura 39: Porcentaje de efectividad (Bancarrota)

Obteniendo una efectividad del 96,69%, podemos afirmar que hemos obtenido un modelo bastante fiable, con una desviación menor del 5% de las observaciones.

Por último, para comprobar la validez del modelo vamos a utilizar el método de las Curvas ROC explicadas anteriormente, para ello necesitamos instalar y llamar a la siguiente librería:

```
#Para realizar la curva ROC

install.packages("pROC")

library(pROC)
```

A continuación, para poder hallar el gráfico de la curva de ROC necesitamos realizar una predicción, pero en este caso, dicha predicción se realiza a través de la probabilidad:

```
#Realizamos una nueva predicción pero en este caso sobre la probabilidad
testpredArbol_prob <- predict(arbol, newdata = data_validation, type = "prob")
```

Posteriormente, podemos a través de la función `roc` realizar el gráfico y mostrarlo gracias a la función `plot`:

```
#Realizo la curva ROC
roc.arbol <- roc(data_validation$Bankrupt, testpredArbol_prob[,2])
plot(roc.arbol)
```

El gráfico generado se muestra a continuación:

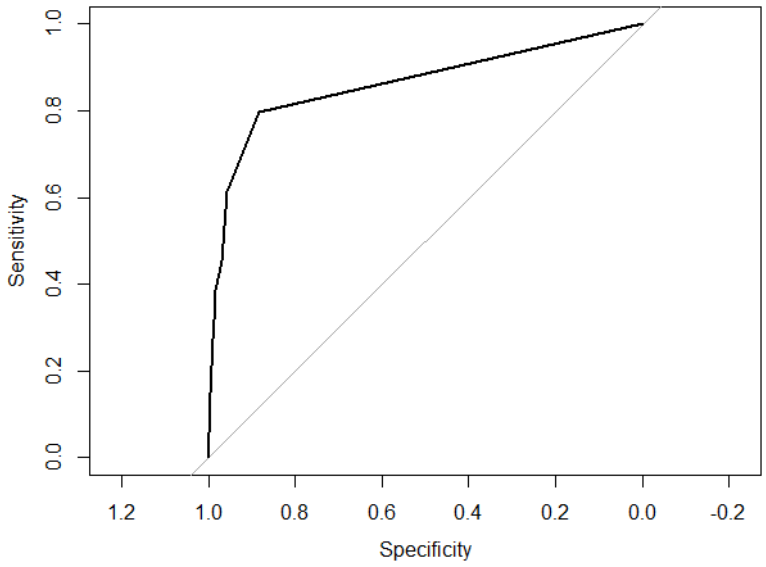


Figura 40: Curva ROC clasificación (Bancarrota)

Para poder ver el área bajo la curva necesitamos ejecutar el siguiente código:

```
#Muestro el área bajo la curva el cual cuanto más cercano a uno es mejor:
roc.arbol$auc
```

Y obtenemos lo siguiente:

```
> roc.arbol$auc
Area under the curve: 0.8613
```

Figura 41: Área bajo la curva clasificación (Bancarrota)

En este caso concreto, y considerando el área bajo la curva, podemos afirmar que nuestro modelo tiene un 86% de probabilidad de distinguir entre empresas que están en bancarrota y las que no.

4. RANDOM FOREST

Random Forest, también conocido como bosque aleatorio, es una técnica de aprendizaje automático basada en árboles de decisión, introducida en Breiman 2001.

4.1 BAGGING

Previamente a explicar en detalle el concepto de Random Forest, considero conveniente hablar del Bagging el cual nace en Breiman 1996 y se utiliza principalmente cuando se emplean árboles muy complejos que cuentan con una alta variabilidad.

El objetivo principal de esta técnica es reducir dicha variabilidad sin empeorar el sesgo. Los métodos Bagging son métodos donde utilizamos algoritmos simples en paralelo.

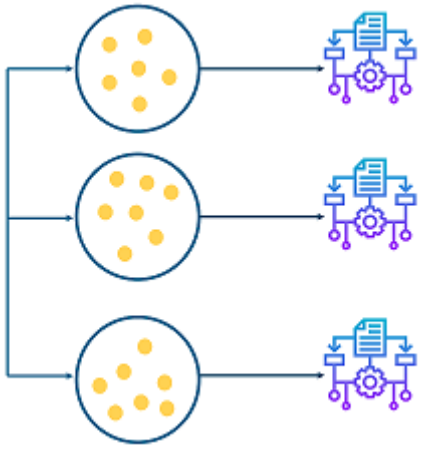


Figura 42: Bagging. (Fuente: machinelearningparatodos,2020)

De esta forma intentamos aprovechar la independencia de cada algoritmo reduciendo el error al promediar las salidas de cada uno de los algoritmos simples.

Contando con los predictores $\hat{f}_1(x), \dots, \hat{f}_B(x)$ en un problema de regresión con B observaciones independientes, tenemos:

$$\hat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}_b(x)$$

Esta predicción es una estimación de la función de regresión $f(x)$ pero con una variabilidad menor.

En cambio, este predictor no se utiliza de forma práctica por no ser habitual contar con B conjuntos de entrenamiento independientes, cuando lo habitual es tener un único conjunto de entrenamiento. Por lo tanto, se emplean B muestras "Bootstrap" del conjunto de entrenamiento. A partir de la muestra patrón vista anteriormente $Z = \{(x_i, y_i)\}_{i=1}^{n}$ se pueden obtener muestras patrón "Bootstrap" $Z = \{(x_i^*, y_i^*)\}_{i=1}^{n}$ en las que $x_i^* = x_{i1},\ y_i^* = y_{i1}, \dots, x_n^* = x_{in}$ e $y_n^* = y_{in}$ para $\{i_1, \dots, i_n\}$. Son muestras aleatorias con un reemplazamiento con tamaño n para $\{1, \dots, n\}$.

El estimador "Bagging" para un problema de regresión en el que se pueden obtener B muestras de Bootstrap del conjunto de entrenamiento $\{Z^{*b}\}_{b=1}^{B}$ para ajustar los predictores $\left\{\hat{f}_{Z^{*b}}(x)\right\}_{b=1}^{B}$, se define:

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}_{Z^{*b}}(x)$$

Gracias a este estimador podemos reducir la variabilidad en la estimación de $f(x)$ y realizar una estimación de la incertidumbre.

Para los problemas de clasificación en lo que se cuenta con una variable categórica con q niveles, se realiza de forma parecida a la vista para los problemas de regresión. Se pueden obtener B muestras de Bootstrap del conjunto de entrenamiento $\{Z^{*b}\}_{b=1}^{B}$ para ajustar los predictores $\left\{\hat{f}_{Z^{*b}}(x)\right\}_{b=1}^{B}$. Para una nueva observación x tenemos:

$$\widehat{p_{k,bag}}(x) = \frac{\#\left\{b : \widehat{\hat{f}_{Z^{*b}}(x)} = k\right\}}{B} \; para\; k = 1, \dots, q$$

Para ello tenemos que realizar la siguiente asignación:

$$\hat{f}_{bag}(x) = K$$

Siempre y cuando ocurra:

$$\widehat{p_{K,bag}}(x) = \max_{k=1,\dots,q} \widehat{p_{k,bag}}(x)$$

Este es el método conocido como "votación por mayoría", el cual aparte de asignar las observaciones a las clases, muestra cuando la observación x esta mas cerca de la clase K si $\widehat{p_{K,bag}}(x)$ tiene un tamaño mas grande que el resto de $\widehat{p_{k,bag}}(x)$ para $k \neq K$.

4.2 RANDOM FOREST

En este caso buscamos mejorar la técnica de "bagging", ya que cuando se trabaja con dicha técnica, trabajamos con árboles muy correlados.

La técnica de los árboles aleatorios parte de las B muestras Bootstrap, entonces al aumentar el tamaño del árbol, en cada división de los nodos se podrán utilizar solo un conjunto de m variables elegidas aleatoriamente para $m \ll p$ tratando de promover la independencia.

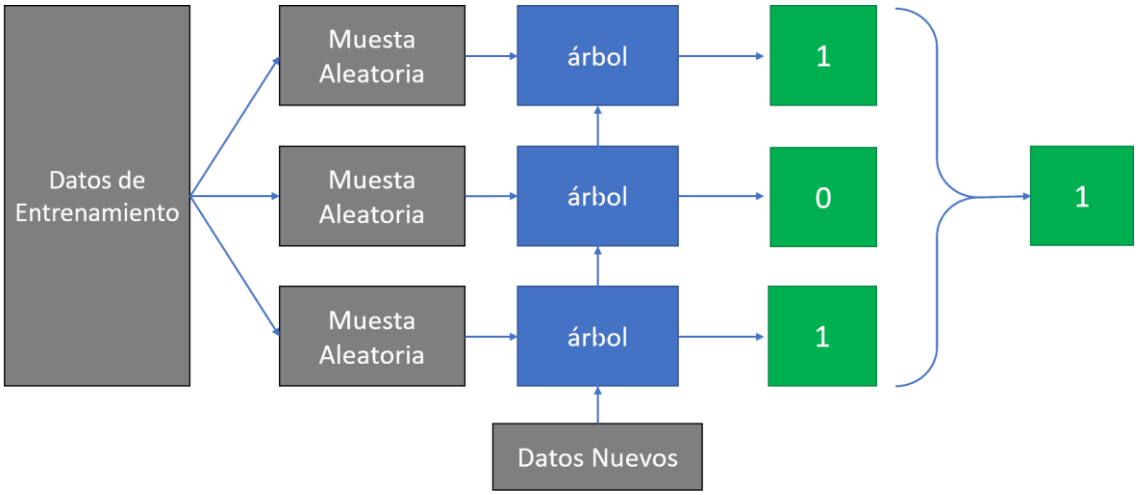


Figura 43: Bosques Aleatorios. (Fuente: iartificial,2020)

Para los problemas de clasificación se suele utilizar

$$m = \sqrt{p}$$

(redondeando si fuera necesario). En cambio, para los problemas de regresión se suele usar

$$m = {}^{p}/_{3}.$$

Este procedimiento se realiza consecutivamente hasta obtener un árbol desarrollado, a las que se les puede añadir predicciones a través del voto por mayoría para la clasificación o a través del promedio en el caso de regresión.

Gracias a la introducción de la aleatoriedad, se puede reducir la correlación entre los árboles. Ya que, sin esta aleatoriedad, cuando se encuentra una variable más útil que otras esta será utilizada la mayor parte de veces en las primeras divisiones y por lo tanto generar árboles parecidos.

4.3 EJEMPLO RANDOM FOREST: DATASET – DIABETES

Para mostrar los casos prácticos de los bosques aleatorios o Random Forest, vamos a utilizar los mismos conjuntos de datos que hemos utilizado para realizar los árboles de clasificación y así poder comparar los resultados obtenidos.

En primer lugar, comenzaremos con la base de datos que buscaba predecir si una persona padece o no la enfermedad de diabetes. Para ello empleamos la variable de clase (Class variable (0 or 1)).

Para utilizar esta técnica, necesitaremos instalar un nuevo paquete "`randomForest`", para ello utilizamos el siguiente código:

```
#Para instalar el paquete necesario para realizar Bosques Aleatorios

install.packages("randomForest")
```

Tras ejecutar este código, podemos llamar a la librería "`randomForest`".

```
#Ejecutamos las librerías

library(randomForest)
```

Lo siguiente que necesitamos es cargar la base de datos (database_diabetes). Se puede ver la explicación de sus 9 variables en el apartado 3.7.

```
#Para leer la base de datos

database_diabetes <- read_csv("01. datasets/database_diabetes.csv")
```

Antes de continuar, debemos transformar el nombre de las variables y quitar los caracteres que pueden provocar errores en el código. Por esta razón, hemos decidido juntar todas las palabras de cada variable, quitando los espacios, barras o cualquier símbolo que pudiera llevar a confusión por parte del programa. Lo hemos realizado a través del siguiente código:

```
#Cambio los nombres de la base de datos

names (database_diabetes) = c("Veces_embarazada", "Concentracion_glucosa", "Presion_art", "Espesor_triceps", "Insulina", "Masa_corp", "Funcion_pedigri_diab","Edad","Variable_clase")
```

Para la técnica de los bosques aleatorios es necesario convertir la variable de clase (Variable_clase) en factor, es decir, si tiene valor 0 podemos afirmar que la persona no es diabética en cambio si la variable muestra un valor 1 si es diabética. Para ello utilizo la función factor sobre esta variable:

```
#Convertimos la variable diabetes en factor

database_diabetes$Variable_clase <- factor(database_diabetes$Variable_clase)
```

El siguiente paso será dividir la base de datos en dos partes como hemos realizado anteriormente, un 60% de los datos estarán dedicados al entrenamiento y el otro 40% a la validación. Para ello hemos utilizado la función `createDataPartition`, la cual genera una lista aleatoria de identificadores y a partir de ellos se puede obtener el conjunto de entrenamiento y de validación.

```
#Dividimos la base de datos en dos, un 60% para entrenamiento y un 40% para validación:

ind <- createDataPartition(database_diabetes$'Variable_clase', p=0.6,list=F)

data_training <- data[ind,]

data_validation <- data[-ind,]
```

Tras realizar la división del conjunto de datos podemos ejecutar la función `randomForest`.

En primer lugar, introducimos la variable de clase de nuestro modelo Variable_clase (Class variable (0 or 1)) la cual vamos a predecir con una selección del resto de variables e introducimos la base de datos.

```
#Generamos un modelo a través de la función randomForest

mod_training <- randomForest(Variable_clase ~ ., data = data_training)
```

A continuación, podemos realizar la predicción a partir del modelo generado anteriormente con la función de los bosques aleatorios.

Pero ahora vamos a utilizar el conjunto de datos, separado anteriormente para hacer la validación. Para ello debemos ejecutar el siguiente código:

```
#Hacemos una predicción

pred_mod <- predict(mod_training, newdata = data_validation)
```

Una vez ejecutado el código mostrado, imprimimos los resultados con una tabla, que nos mostrará los aciertos del sí y del no de la variable de clase.

```
#Tabla para mostrar los resultados de la predicción

table(pred_mod,data_validation$Variable_clase)
```

Los resultados de la matriz de confusión se pueden ver en la siguiente imagen:

```
> table(pred_mod,data_validation$variable_clase)

pred_mod   0   1
       0 172  37
       1  26  68
```

Figura 44: Matriz de confusión RandomForest (Diabetes)

Como se puede apreciar, de las 303 observaciones que tiene el conjunto de datos de validación, ha predicho correctamente 172 observaciones al predecir que dichas personas no tendrían diabetes. También ha predicho correctamente que 68 personas tienen la enfermedad. Por otro lado, se aprecia que se equivoca en 37 personas al predecir que estas no tendrían la enfermedad y en realidad si la tienen. Y también se equivoca al predecir que 26 personas si padecen la enfermedad y en realidad no la padecen.

Por último, vamos a evaluar la efectividad del modelo. Para ello sumo todos los aciertos y lo dividimos entre el número de predicciones:

```
> sum(pred_mod == data_validation$variable_clase) / length(data_validation$variable_clase)*100
[1] 79.20792
```

Figura 45: Porcentaje de efectividad RandomForest (Diabetes)

Obteniendo una efectividad del 79,21%, podemos afirmar que hemos tenido una mejoría importante respecto al modelo realizado anteriormente a través de los árboles de clasificación que nos daba una efectividad del 75,25%.

Por último, para comprobar la validez del modelo vamos a utilizar el método de las Curvas ROC explicadas anteriormente, para ello necesitamos instalar y llamar a la siguiente librería:

```
#Para realizar la curva ROC

install.packages("pROC")

library(pROC)
```

A continuación, para poder hallar el gráfico de la curva de ROC necesitamos realizar una predicción, pero en este caso, dicha predicción se realiza a través de la probabilidad:

```
#Realizamos una nueva predicción pero en este caso sobre la probabilidad
testpredArbol_prob <- predict(mod_training, newdata = data_validation, type = "prob")
```

Posteriormente, podemos a través de la función `roc` realizar el gráfico y mostrarlo gracias a la función `plot`:

```
#Realizo la curva ROC
roc.arbol <- roc(data_validation$Variable_clase, testpredArbol_prob[,2])
plot(roc.arbol)
```

El gráfico generado se muestra a continuación:

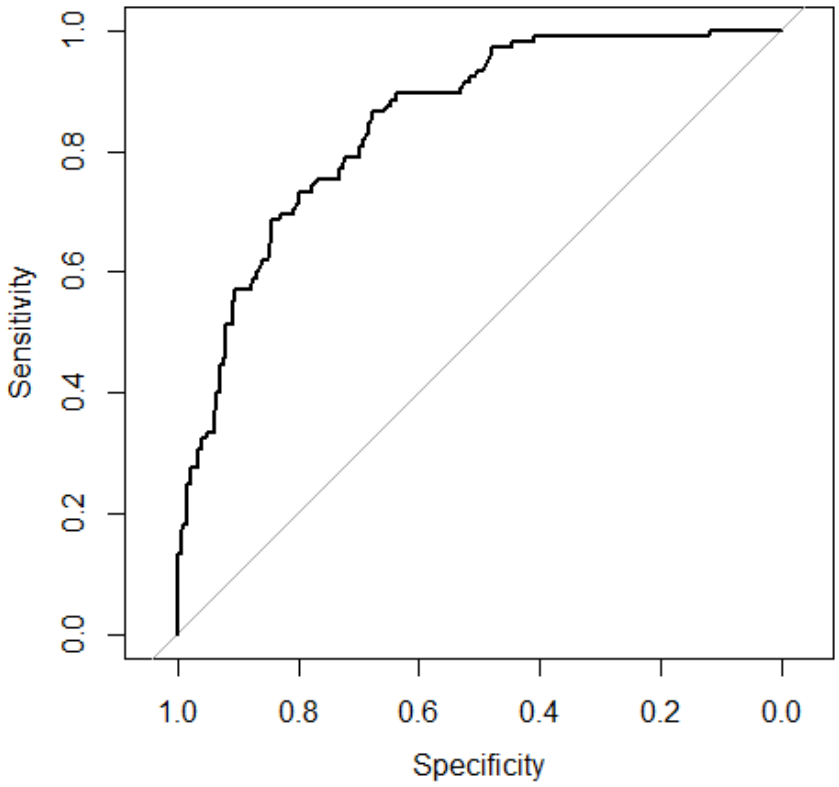


Figura 46: Curva ROC Random Forest (Diabetes)

Para poder ver el área bajo la curva necesitamos ejecutar el siguiente código:

```
#Muestro el área bajo la curva el cual cuanto más cercano a uno es mejor:
roc.arbol$auc
```

Y obtenemos lo siguiente:

```
> roc.arbol$auc
Area under the curve: 0.8465
```

Figura 47: Área bajo la curva Random Forest (Diabetes)

Como se puede apreciar tenemos una mejora considerable respecto al modelo realizado con el método de clasificación ya que con Random Forest nuestro modelo posee una probabilidad del 85% de distinguir entre pacientes con diabetes o sin ella.

4.4 EJEMPLO RANDOM FOREST: DATASET - BANCARROTA

A continuación, realizaremos el estudio para base de datos que busca predecir qué empresa entrará o no en bancarrota. Para ello utilizábamos la variable de clase Bancarrota (Bankrupt).

Para utilizar esta técnica, necesitaremos instalar un nuevo paquete "`randomForest`", para ello utilizamos el siguiente código:

```
#Para instalar el paquete necesario para realizar Bosques Aleatorios

install.packages("randomForest")
```

Tras ejecutar este código, podemos llamar a la librería "`randomForest`".

```
#Ejecutamos las librerías

library(caret)

library(randomForest)
```

Lo siguiente que necesitamos es cargar la base de datos (data_bancarrota). Se puede ver la explicación de sus 96 variables en el apartado 3.8.

```
#Para leer la base de datos

data_bancarrota <- read_csv("01. datasets/data_bancarrota.csv")
```

Antes de continuar, debemos transformar el nombre de las variables y quitar los caracteres que pueden provocar errores en el código. Por esta razón, hemos decidido juntar todas las palabras de cada variable, quitando los espacios, barras o cualquier símbolo que pudiera llevar a confusión por parte del programa. Lo hemos realizado a través del siguiente código:

```
#Cambio los nombres de la base de datos

names(data_bancarrota)=c('Bancarrota','ROAC','ROAA','ROAB','Margenbrutodeexplotación' ,'Margenbrutodeventasrealizadas' ,'Tasadebeneficiodeexplotación' …)
```

Para la técnica de los bosques aleatorios es necesario convertir la variable de clase bancarrota en factor, es decir, si tiene valor 0 la empresa no entraría en bancarrota en cambio sí tiene valor 1 si lo haría. Para ello utilizo la función factor sobre esta variable:

```
#Convertimos la variable bancarrota en factor

data_bancarrota$Bankrupt <- factor(data_bancarrota$Bankrupt)
```

El siguiente paso será dividir la base de datos en dos partes como hemos realizado anteriormente, un 60% de los datos estarán dedicados al entrenamiento y el otro 40% a la validación. Para ello hemos utilizado la función `createDataPartition`, la cual genera una lista aleatoria de identificadores y a partir de ellos se puede obtener el conjunto de entrenamiento y de validación.

```
#Dividimos la base de datos en dos, un 60% para entrenamiento y un 40% para validación:

ind <- createDataPartition(data$`Bankrupt`, p=0.6,list=F)

data_training <- data[ind,]

data_validation <- data[-ind,]
```

Tras realizar la división del conjunto de datos podemos ejecutar la función `randomForest`.

En primer lugar, introducimos la variable de clase de nuestro modelo Bancarrota (Bankrupt) la cual vamos a predecir con una selección del resto de variables e introducimos la base de datos.

```
#Generamos un modelo a través de la función randomForest

mod_training <- randomForest(Bancarrota ~ .,data=data_training)
```

A continuación, podemos realizar la predicción a partir del modelo generado anteriormente con la función de los bosques aleatorios.

Pero ahora vamos a utilizar el conjunto de datos, separado anteriormente para hacer la validación. Para ello debemos ejecutar el siguiente código:

```
#Hacemos una predicción del modelo

pred_mod <- predict(mod_training, newdata = data_validation, type='class')
```

Una vez ejecutado el código mostrado, imprimimos los resultados con una matriz de confusión, que nos mostrará los aciertos del sí y del no de la variable bancarrota.

```
#Tabla para mostrar los resultados de la predicción

table(pred_mod,data_validation$Bancarrota)
```

Los resultados de ejecutar este código se pueden ver en la siguiente imagen:

```
> table(pred_mod,data_validation$Bancarrota)

pred_mod    0    1
       0 2632   75
       1    7   13
```

Figura 48: Matriz de confusión RandomForest (Bancarrota)

Como se puede apreciar, de las 2727 observaciones que tiene el conjunto de datos de validación, ha predicho correctamente 2632 observaciones al predecir que dichas empresas no entrarían en bancarrota y no han entrado. También ha predicho correctamente que 13 empresas si entraran en bancarrota. Por otro lado, se aprecia que se equivoca en 75 empresas al predecir que dichas empresas no entran en bancarrota cuando en verdad si lo hacen. Y en solo 7 ocasiones se ha equivocado prediciendo que una empresa entra en bancarrota cuando en realidad no lo hace.

Por último, vamos a evaluar la efectividad del modelo para ello sumo todos los aciertos y lo dividimos entre el número de predicciones:

```
> sum(pred_mod == data_validation$Bancarrota) / length(data_validation$Bancarrota)*100
[1] 96.99303
```

Figura 49: Porcentaje de efectividad RandomForest (Bancarrota)

Obteniendo una efectividad del 96,99%, podemos afirmar que hemos mejorado el modelo ligeramente realizado anteriormente a través de los árboles de clasificación que nos daba una efectividad del 96,88%.

Por último, para comprobar la validez del modelo vamos a utilizar el método de las Curvas ROC explicadas anteriormente, para ello necesitamos instalar y llamar a la siguiente librería:

```
#Para realizar la curva ROC

install.packages("pROC")

library(pROC)
```

A continuación, para poder hallar el gráfico de la curva de ROC necesitamos realizar una predicción, pero en este caso, dicha predicción se realiza a través de la probabilidad:

```
#Realizamos una nueva predicción pero en este caso sobre la probabilidad

testpredArbol_prob <- predict(mod_training, newdata = data_validation, type = "prob")
```

Posteriormente, podemos a través de la función `roc` realizar el gráfico y mostrarlo gracias a la función `plot`:

```
#Realizo la curva ROC
roc.arbol <- roc(data_validation$Bancarrota, testpredArbol_prob[,2])
plot(roc.arbol)
```

El gráfico generado se muestra a continuación:

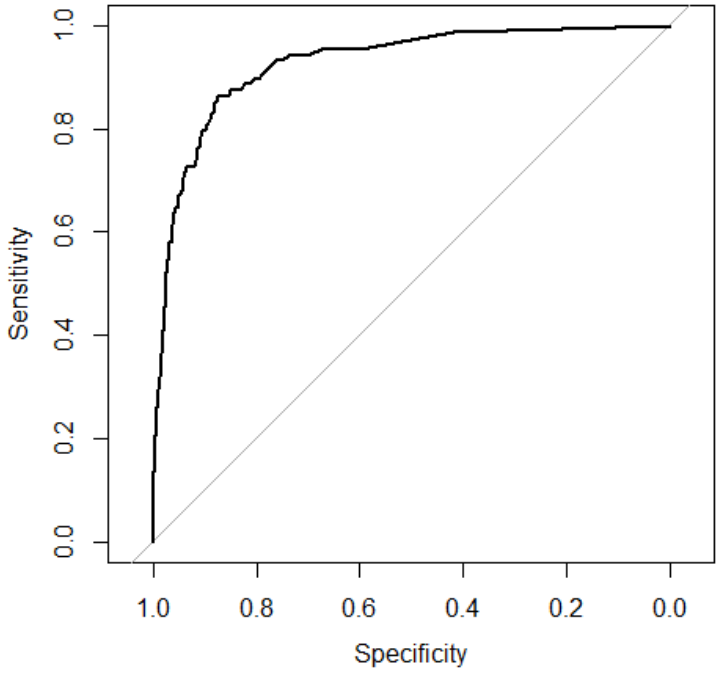


Figura 50: Curva ROC Random Forest (Bancarrota)

Para poder ver el área bajo la curva necesitamos ejecutar el siguiente código:

```
#Muestro el área bajo la curva el cual cuanto más cercano a uno es mejor:
roc.arbol$auc
```

Y obtenemos lo siguiente:

```
> roc.arbol$auc
Area under the curve: 0.9266
```

Figura 51: Área bajo la curva Random Forest (Bancarrota)

Como se puede apreciar tenemos una mejora considerable respecto al modelo realizado con el método de clasificación donde obteníamos un valor del 85% por lo que se ha incrementado en 8 puntos porcentuales el valor AUC.

En este caso, el modelo presenta una probabilidad del 93% de predecir correctamente si una empresa entra en bancarrota.

5. CONCLUSIÓN

En este trabajo se ha estudiado y presentado la metodología de árboles de decisión y su extensión en los bosques aleatorios. Esta metodología se ha presentado mediante varios ejemplos prácticos y usando lenguaje de programación en el entorno R. Esta metodología, de árboles y de bosques aleatorios, ha implicado profundizar en técnicas predictivas que no habían sido tratadas en las asignaturas relacionadas con la Estadística cursadas en la titulación, donde se había presentado solo una breve introducción a las técnicas de regresión y a la discriminación lineal. Se ha comprobado que estas metodologías de árboles de decisión pueden ser de gran utilidad como herramienta para realizar predicciones en Ingeniería, en particular en la práctica profesional de un Ingeniero en Organización Industrial.

Las principales conclusiones que podemos obtener del estudio son las siguientes. En relación con las ventajas que presenta esta metodología podemos afirmar que las principales ventajas de los árboles de decisión:

- Son modelos sencillos de comprender y fáciles de explicar, incluso a otros profesionales sin conocimientos en Estadística.
- Esta facilidad en su interpretación hace que no puedan ser vistos como "cajas negras" para realizar predicciones, como sucede con otros enfoques también utilizados en Aprendizaje Automático supervisado.
- No está demasiado afectado por observaciones muy extremas o atípicas al contrario de lo que ocurre con otros métodos basados en estimadores poco robustos como pueden ser las matrices de varianzas-covarianzas. Permiten tratar bastante bien las observaciones faltantes o perdidas (missing) porque se sustituyen las variables no observadas por otras variables (subrogadas) que sí son observadas.
- No se precisan rígidas restricciones e hipótesis sobre las distribuciones de las variables implicadas en la predicción. Así, por ejemplo, no es necesaria la hipótesis de normalidad en la variable respuesta, en el caso de la regresión, y se puede tratar de forma análoga con variables explicativas categóricas (ordinales y cualitativas).

No obstante, presentan desventajas como:

- En ocasiones se tiene capacidades predictivas bajas ya que los procedimientos predictivos son, por definición, más simples que otros procedimientos que permiten definir reglas predictivas más complejas.
- También, en ocasiones, las reglas obtenidas con los árboles de clasificación pueden tener una variabilidad más alta de lo deseado ya que cambios memores en los datos pueden cambiar algunas ramas de forma notable.

Estas dos desventajas se ven notablemente solventadas por el método de bosques aleatorios (sin perder todas las ventajas anteriormente comentadas). Esto ha sido corroborado en los conjuntos de datos utilizados viendo como los resultados mejoran ligeramente respecto a los modelos originales basados en simples árboles de decisión.

Como línea de investigación futura sugerimos analizar la aplicabilidad de otra extensión de los árboles de clasificación, que son también muy utilizadas en la actualidad, y que son conocidas como técnicas “boosting”. Esta metodología también trata de evitar las dos desventajas anteriormente comentadas.

6. BIBLIOGRAFÍA

European Knowledge Center for Information Technology (Ed.). (2021, 23 abril). Machine learning: ¿cómo aprenden las máquinas a trabajar por su cuenta?, TIC Portal. https://www.ticportal.es/glosario-tic/machine-learning

Heras, J. M. (2020, 18 septiembre). *Random Forest (Bosque Aleatorio): combinando árboles*. IArtificial.net. https://www.iartificial.net/random-forest-bosque-aleatorio/#%C2%BFQue_es_un_Random_Forest

James, G., Witten, D., Hastie, T., y Tibshirani, R. (2021). *An Introduction to Statistical Learning: With Applications in R* (2nd 2021 ed.). Springer.

Kuhn, M., & Johnson, K. (2013). *Applied predictive modeling*. New York: Springer.

Parra, F. (s. f.). *6 Métodos de clasificación | Estadística y Machine Learning con R*. https://bookdown.org/content/2274/metodos-de-clasificacion.html

Peña, D. (2002) *Análisis de datos multivariantes.* McGraw-hill.

Rueda, J. F. V. (2019, 9 agosto). *Aprendizaje supervisado y no supervisado*. healthdataminer.com. https://healthdataminer.com/data-mining/aprendizaje-supervisado-y-no-supervisado/#:%7E:text=El%20aprendizaje%20supervisado%20supone%20que,de%20datos%20no%20etiquetados%20previamente