Facultad de Matemáticas

Departamento de Estadística e
Investigación Operativa

TRABAJO FIN DE GRADO

Modelos de Clasificación con datos no balanceados

Rocío Espinar Lara
Tutor: Rafael Pino Mejías

Sevilla, Junio 2018

Índice general

Resumen

El problema de las distribuciones de datos no balanceados entre clases, ha recibido una atención considerable en disciplinas como el Aprendizaje Automático o Machine Learning, en inglés y Minería de Datos, entre otros.
En el contexto de problemas de clasificación, se dice que un conjunto de datos no está balanceado si una de las clases (mayoritaria) está sensiblemente más representada que el resto de clases. Esta problemática puede conducir, en términos de clasificación, a aprendizajes sesgados en perjuicio de la clase minoritaria, que usualmente, contiene los casos de mayor interés.
En esta memoria se aborda el problema de clases no balanceadas, en la que se estudian diferentes técnicas de clasificación con las que solventar o disminuir dicho problema.
En la introducción, se describe el problema de la distribución desigual de clases.
En el capítulo 2, se presentan diversos conceptos del Aprendizaje Estadístico utilizados en este trabajo.
Posteriormente, se hablará de los árboles de clasificación y los modelos Random Forests, describiendo sus principales características.
En el capítulo 4, se verán algunas de las medidas de rendimiento que serán usadas para medir la eficiencia de un modelo de clasificación.
En el capítulo 5, se proponen técnicas para solventar el problema descrito en este trabajo, como por ejemplo técnicas de remuestreo, para reducir/incrementar el tamaño muestral de la clase mayoritaria/minoritaria y ténicas basadas en costes que penalizan la clasificación errónea, entre otros.
Por último, se realiza una comparación empírica de los métodos descritos sobre un conjuntos de datos (insolvencias). Se ha utilizado para ello el entorno de programación estádistico R [18].

Abstract

The problem of unbalanced data distributions between classes has received a considerable attention in disciplines such as Machine Learning and Data Mining, among others.

In the context of classifications problems, it is said that a data set is unbalanced if one of the classes (majority) is significantly more represented than the rest of the classes. This problem can lead, in terms of classification, to biased learning to the detriment of the minority class, which usually contains the cases of greatest interest.
This report deals with the problem of unbalanced classes, in which different classification techniques are studied to solve or reduce this problem.
In the introduction, the problem of unequal classes distributions is described.
In Chapter 2, several concepts of Statistical Learning used in this report are presented.
Later, the classification trees and the Random Forests models will be discussed, describing their main characteristics.
In Chapter 4, you can find some of the most common performance measures that will be used to measure the efficiency of a classification model.
In Chapter 5, techniques are proposed to solve the problem described in this project, such as resampling methods to reduce/increase the size of the majority/minority class and techniques based on costs that penalize the erroneus classification.
Finally, an empirical comparison of the methods described on a data set (insolvences) is carried out. The statistical programmes environment R has been used for this purpose.

Capítulo 1

Introducción

1.1. Motivación

La información almacenada en una base de datos puede ser agrupada en diversas clases dependiendo de diversas características que presenten los datos pertenecientes a ella.

Cuando se tienen elementos históricos y se quiere aprender la relación que guarda dicha información con la clase de pertenencia, entonces se habla de un problema de Aprendizaje Estadístico. Estos problemas se refieren a la inducción del conocimiento a partir de una información previa. Cuando se trabaja con información que se quiere clasificar, se puede realizar mediante un proceso supervisado que determine la posibilidad de que algún ejemplo nuevo pertenezca a una clase ya determinada.

Para própositos de este trabajo, se consideran aquellos conjuntos de datos que se encuentran no balanceados, es decir, que algunas de sus clases tenga muchas más instancias que las otras. En el marco de la clasificación de datos con estas características, el desequilibrio entre clases provoca que se pierda capacidad de clasificar nuevas instancias entre las distintas clases.

1.2. Descripción del problema

Los problemas de Aprendizaje Automático son aquellos que se generan a partir de la determinación de características de forma intuitiva por un sistema computacional.
En particular, el Aprendizaje Supervisado implica la identificación de patrones o características para determinar la pertenencia de una instancia a una clase determinada. Normalmente el proceso de aprendizaje se lleva a cabo en dos etapas, el entrenamiento del modelo de clasificación, y el proceso final de clasificación. En el primer paso se establece un esquema a partir de datos previos conocidos, y con dicho esquema, en el segundo paso se determina la clase para las instancias desconocidas. Dicho esquema puede ser por ejemplo un árbol de decisión o un perceptrón multicapas, entre otros. Pero en cada caso, el objetivo final es clasificar ejemplos desconocidos a partir de los conocidos, por lo cual el proceso de entrenamiento es muy importante. Cuando en la información almacenada en un conjunto de datos existe una diferencia importante en la cantidad de elementos de cada clase entonces se habla de *clases no balanceadas*.

Este problema de la representación desigual de clases, también conocido como el problema de clases no balanceadas, se presenta cuando en el conjunto de datos de entrenamiento no hay un número (aproximadamente) igual de muestras de cada clase.[4] Este problema resulta de interés en aquellos dominios de aplicación en los que clasificar erróneamente un objeto de la clase minoritaria tiene un costo medio muy elevado.
En general se desea aproximar la regla de decisión óptima, que minimiza el riesgo global de clasificación. Cuando esta búsqueda se hace en base a un conjunto de entrenamiento con clases altamente no balanceadas, la regla de decisión produce fronteras de decisión sesgadas a favor de la clase mayoritaria.

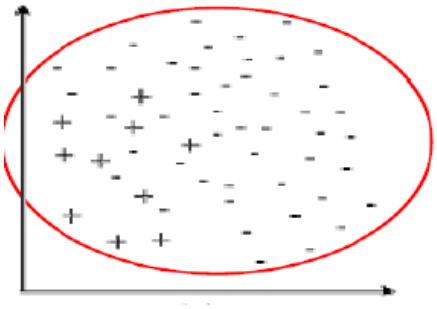

Figura 1.1: Gráfico de una distribución no balanceada

En la figura 1.1 se puede observar de manera gráfica, el problema de las clases no balanceadas: hay un número reducido de (+)(clase minoritaria) en comparación con el número correspondiente de (-)(clase mayoritaria).

En la actualidad, el escenario de una distribución de datos no balanceados se refleja en varias situaciones como por ejemplo la detección de fraudes, la prevención de intrusiones, las finanzas o incluso, en investigaciones médicas. Sin embargo, en muchos de los problemas en los que el desbalanceo de clases está presente, la clase minoritaria suele ser la de mayor interés, de modo que clasificar erróneamente casos pertenecientes a dicha clase supone un alto coste.

La importancia de este problema se puede apreciar en el interés y el número de trabajos publicados recientemente. En esta memoria se presentan las técnicas principales , en particular las basadas en procedimientos de muestreo y en la generación de casos sintéticos. Para ilustrar el efecto de los métodos presentados, se utiliza un modelo que en los últimos años ha alcanzado gran popularidad, Random Forests, si bien puede emplearse cualquier otra técnica sobre los conjuntos de datos resultantes del tratamiento de los datos no balanceados.

Capítulo 2

Aprendizaje Estadístico

En este capítulo se presentan los conceptos fundamentales del Aprendizaje Estadístico donde se inscribe este trabajo.[8] Se utilizará este término en lugar de Aprendizaje Automático para recalcar los aspectos estadísticos propios de estos procesos.

2.1. Introducción

El Aprendizaje Estadístico juega un papel importante en diversas áreas de la ciencia, finanzas y la industria. Algunos ejemplos de problemas de aprendizaje podrían ser:

- Predecir si un paciente, hospitalizado debido a un infarto, tendrá un segundo ataque.
- Estimar la cantidad de glucosa en sangre de una persona diabética.
- Identificar los factores de riesgo para el cáncer de próstata, basado en variables clínicas y demográficas.
- Identificar correos electrónicos "spam".
- Sistema de Reconocimento Óptico de caracteres (OCR).

Figura 2.1: Taxonomía de los métodos de Aprendizaje Automático

En la figura 2.1 se puede observar cómo los métodos de Aprendizaje Estadístico son clasificados taxonómicamente por Kononenko [10] dependiendo de los objetivos y del tipo de información disponible.

El Aprendizaje Estadístico es más comunmente usado para clasificación. La decisión sobre los modelos a considerar depende de si el problema se inscribe en el marco del Aprendizaje Supervisado o en el Aprendizaje No Supervisado.

Aprendizaje Supervisado. El aprendizaje supervisado es una técnica de Aprendizaje Estadístico en base a unos datos de entrenamiento, que generaliza una función a partir de haber considerado un número determinado de instancias históricas. Estos datos consisten en objetos con dos partes: variables de entrada (*inputs*) y variables de salida o clases. El objetivo es usar esos campos de entrada para predecir los valores de las salidas.

Los datos de entrada son a menudo llamados predictores y, más clásicamente variables independientes. Los datos de salida son comúnmente reconocidos como respuestas, o variables dependientes. Las variables respuesta pueden ser tanto variables cuantitativas (regresión) como cualitativas (clasificación), también conocidas como variables categóricas o discretas. El proceso de construcción del modelo tiene en cuenta los errores de predicción para intentar minimizar el error total sobre el llamado conjunto de entrenamiento. Algunas cuestiones importantes que surgen en el marco del Aprendizaje Estadístico son por ejemplo, cómo medir el error de clasificación o de predicción, la definición y la estimación de la capacidad de generalización, comparación de modelos, selección de modelos, y el dilema sesgo-varianza, que tiene su base en que no por aumentar la complejidad del modelo, mejora necesariamente el rendimiento de este.

Aprendizaje No Supervisado. El aprendizaje no supervisado tiene lugar cuando no se dispone de datos " etiquetados " para el entrenamiento. Sólo se conocen las variables de entrada, pero no existen variables de salida.

Por tanto solo se puede describir la **estructura** de los datos para intentar encontrar algún tipo de organización que simplifique el análisis. Dentro de los tipos de algoritmo más habituales en aprendizaje no supervisado se encuentran los algoritmos de agrupamiento o *clustering*, en el que los datos son agrupados tratando de hacer máxima la similaridad entre los elementos de un mismo grupo, y al mismo tiempo, minimizando la similaridad entre los distintos grupos. En otras palabras, se trata de crear grupos homogéneos con máxima heterogeneidad entre ellos.

2.2. Dilema sesgo-varianza

Sea un modelo $Y = f(X)+\varepsilon$, con $E[\varepsilon] = 0$, $V[\varepsilon] = \sigma^2$. Se considera un modelo de predicción $g(X, w)$, que depende de un vector de parámetros w, estimados a partir de una muestra de entrenamiento T_n. Dado x_0, suponiendo el criterio error cuadrático, el error esperado de predicción en x_0 es:

$$\begin{aligned} e(x_0) &= E[(Y - \hat{g}\,(x_0))^2 \,/\, X = x_0] = \\ &= \sigma^2 + [E\{\hat{g}\,(x_0)\} - f(x_0)]^2 + E[\hat{g}\,(x_0) - E\{\hat{g}\,(x_0)\}]^2 = \\ &= \sigma^2 + Sesgo(\hat{g}\,(x_0))^2 + Var(\hat{g}\,(x_0)) \end{aligned}$$

$$\boxed{Error\ irreducible + Sesgo^2 + Varianza}$$

En general, a mayor complejidad del modelo, menor es el sesgo, es decir, más acertado es el valor medio de predicción, pero a su vez la varianza también es mayor, lo que hace que disminuya su fiabilidad.
A menor complejidad, su sesgo será mayor, pero a cambio la varianza se reduce. Por lo tanto, no se puede reducir simultáneamente ambas componentes del error de predicción.

Es muy importante por tanto, realizar cuidadosamente un proceso de configuración de la complejidad del modelo predictivo que se vaya a usar, como por ejemplo, determinar el tamaño de un árbol de clasificación.

2.3. Capacidad de Generalización

Considerando una muestra aleatoria simple $T_n = (x_i, y_i)$, $i = 1, 2, \cdots, n$ de una variable aleatoria bidimensional (X, Y) y suponiendo que $Y = f(X)+\varepsilon$, donde la función f es desconocida y ε es la variable error, cuya media se suele suponer 0. El objetivo que se plantea es construir un modelo de predicción apropiado de modo que a partir del conocimiento de $X = x$ se obtenga una estimación de $E[Y/X = x]$ o bien una aproximación a la distribución $Y/X = x$. En la terminología del Aprendizaje Estadístico, la muestra suele recibir el nombre de **muestra o conjunto de entrenamiento**.

Ejemplo. En la figura 2.2 se visualiza una realización muestral con n=10.

Se han generado los datos de esta simulación suponiendo $f(x) = \sin(2\pi X)$:

$$Y = \sin(2\pi X) + \varepsilon, \quad \varepsilon \sim \mathrm{N}(0, \sigma^2),\ X \sim U(0, 1).$$

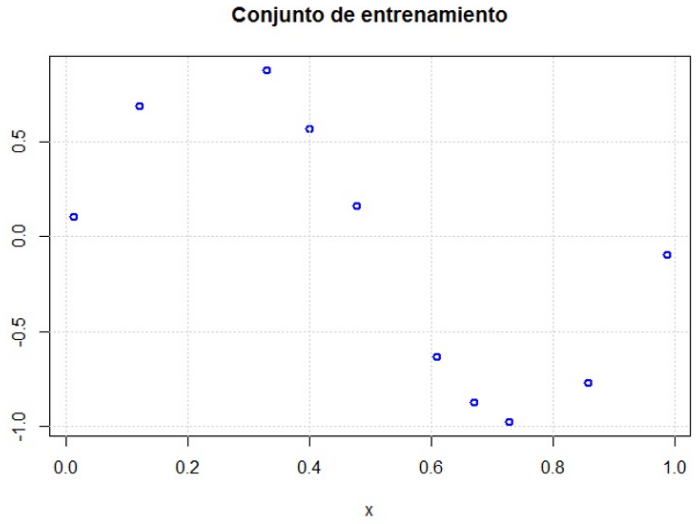

Figura 2.2: Visualización Gráfica n=10

Entre los diversos modelos de predicción que podrían utilizarse en este problema se va a considerar la familia de modelos de regresión polinomial. El modelo de orden p construye la siguiente combinación lineal:

$$g(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_p x^p$$

La cuestión que surge es cuál es el valor más adecuado para el parámetro p. Esta pregunta puede abordarse considerando una familia de modelos cuyo orden varíe en un rango de valores p_{min}, $p_{min} + 1, \cdots, p_{max}$.

Para cada valor de p en ese rango es necesario estimar los coeficientes $\beta_0, \cdots, \beta_p$ y medir adecuadamente el rendimiento que cabe esperar para el modelo asociado. A partir del proceso anterior se podrá realizar una selección razonada del valor más apropiado para p. Cuanto mayor sea p, más parámetros tendrá este modelo, es decir, a mayor valor de dicho parámetro, mayor será la complejidad del modelo.
La estimación de los parámetros requiere definir y minimizar un criterio de error sobre los datos de entrenamiento.

Para modelos de regresión lineal como son los de regresión polinomial es habitual el procedimiento de mínimos cuadrados, donde se desea minimizar la suma total de los cuadrados de los residuos.

Por tanto, fijado p, se considera el siguiente problema de optimización:

$$\hat{\beta} = (\hat{\beta_0}, \hat{\beta_1}, \cdots, \hat{\beta_p}) = \arg\min_{\beta} \sum_{i=1}^{n} = (y_i - \beta_0 - \beta_1 x_i - \cdots, \beta_p x_i^p)^2$$

Dado que se trata de un problema de regresión lineal múltiple, se puede calcular fácilmente la solución de dicho problema. En el caso de las redes neuronales por ejemplo, se necesitan métodos numéricos para afrontar el problema de la estimación de parámetros.

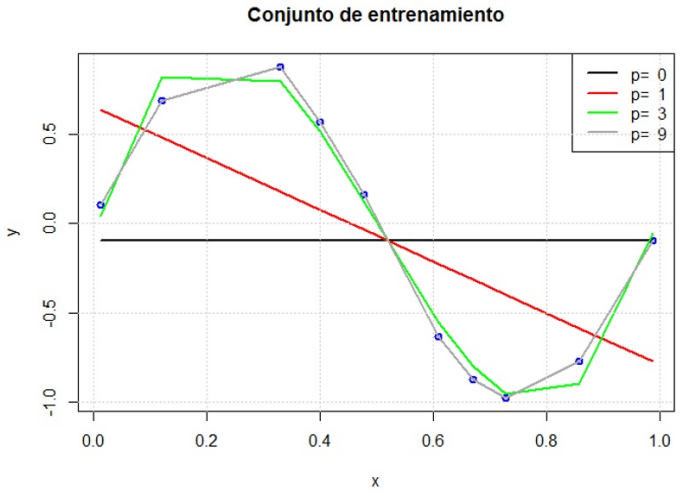

Figura 2.3: Ajuste según el valor de p para n=10

En la figura 2.3 se observa que a mayor valor de p mejor es el ajuste sobre T_n de hecho para $p = 9$ el ajuste es perfecto (algo esperable, puesto que son 10 puntos y 10 coeficientes).

Se supone ahora que se dispone de otra muestra de tamaño 100 extraída de la misma población. Si el modelo ajustado es lo realmente útil, las predicciones deben ser " buenas ", no solo en el conjunto de entrenamiento, sino en cualquier observación de la misma población. Es aquí donde surge el concepto de generalización. Interesa más una buena generalización, que un error muy pequeño en el entrenamiento.

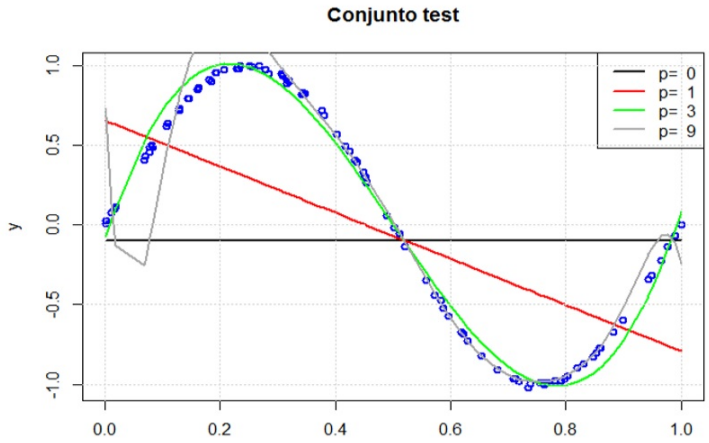

Figura 2.4: Ajuste según el valor de p para n=100

En la figura 2.4 se observa que el modelo de mayor complejidad, $p = 9$ presenta un rendimiento muy pobre sobre los datos nuevos, siendo el modelo $p = 3$ el que presenta un rendimiento más satisfactorio. Los modelos con $p = 0$ y $p = 1$ son demasiado simples.

Por lo tanto, lo anteriormente citado ha permitido poner de manifiesto varios aspectos de interés:

- La estimación del rendimiento de un modelo de predicción no puede efectuarse sobre la muestra de entrenamiento.
- Conviene disponer de otros mecanismos para evaluar el rendimiento de un modelo. Por ejemplo, disponer de otra muestra, llamada **muestra o conjunto test**, donde se evalúe el rendimiento.
- No por aumentar la complejidad del modelo se tendrá un mejor rendimiento esperado, pues modelos de gran complejidad suelen presentar el problema del sobreajuste; donde el rendimiento suele ser muy bueno en la muestra de entrenamiento, y pobre en la muestra test. Esta problemática ha sido citada en la sección 2.2 conocida como el dilema sesgo-varianza.

2.3.1. Medidas de error

Dado un modelo de predicción $g(x)$ para una variable dependiente Y, se considera una medida de predicción $L(Y, g(x))$. Algunas medidas usuales son:

Si Y es cuantitativa (**Regresión**):

Error cuadrático:

$$(Y - g(x))^2$$

Error Absoluto:

$$|Y - g(x)|$$

Si Y es cualitativa (**Clasificación**):

Pérdida 0-1:

$$I(Y \neq g(X))$$

Log.Verosimilitud, Entropía Cruzada o Desviación :

$$-2\sum_{k-1}^{k} I(Y = k) \log \hat{P} [Y = k/X] = -2 \log \hat{P}_Y (X);$$

$$g(X) = \arg \max_{1 \leq k \leq K} \hat{P} [Y = k/X], \text{donde K es el número de clases.}$$

Una vez se ha definido $L(Y, g(x))$, se tienen las siguientes definiciones:

Definición 1 *Dado un modelo de predicción $g_\lambda(x)$, siendo λ un parámetro de complejidad, construido a partir de un conjunto de entrenamiento $T_n = (x_i, y_i) \quad i = 1, 2, \cdots, n$, se define el* ***error empírico o error de entrenamiento*** *como:*

$$v_{T_n} = \frac{1}{n} \sum_{i=1}^{n} L(y_i, \overset{\wedge}{g_\lambda} (x_i))$$

En el caso de modelos de regresión y clasificación, se suele trabajar con el Error Cuadrático medio (ECM), y la tasa de error respectivamente:

$$ECM_{T_n} = \frac{1}{n} \sum_{i=1}^{n} (\overset{\wedge}{g_\lambda} (x_i) - y_i)^2$$

$$TE_{T_n} = \frac{1}{n}\sum_{i=1}^{n} I(\hat{g_\lambda}\,(x_i) \neq y_i)$$

Definición 2 *Dado un modelo de predicción $g_\lambda(x)$, ajustado sobre un conjunto de entrenamiento T_n, se define el* ***error de generalización****:*

$$Err_{T_n} = E_{(X,Y)}[L(Y, \hat{g_\lambda}\,(X))/T_n]$$

Si se toma valor esperado sobre los conjuntos de entrenamiento posibles, se tiene el error esperado.

Definición 3 *Dado un modelo de predicción $g_\lambda(x)$, y dado un tamaño muestral n, se define el* ***error esperado***

$$Err_n = E_{T_n}\{Err_{T_n}\} = E_{T_n}\{E_{(X,Y)}[L(Y, \hat{g_\lambda}\,(X))/T_n]\}$$

El error empírico tiende a ser inferior al de generalización, debido al uso de los mismos datos para ajustar el modelo y para estimar el error. Este optimismo puede formalizarse como:

Definición 4 *Se define el* ***error de generalización interno como:***

$$Errin_{T_n} = \frac{1}{n}\sum_{i=1}^{n} E_Y[L(Y_i, \hat{g_\lambda}\,(X_i))/T_n]$$

En esta definición se fijan los predictores observados en T_n, tomando valor esperado en las variables independientes Y_i, con lo que se ayuda a estudiar el concepto de optimismo.

Definición 5 *Fijados los vectores x_i de T_n, se define el* ***optimismo del error empírico*** *como:*

$$op = Errin_{T_n} - v_{Tn}$$

siendo v_{Tn} el error empírico. Veáse la definición 1.

Definición 6 *Se define el optimismo medio o esperado como:*

$$w = E_Y\{op\}$$

Aquí están fijados los predictores, y se consideran los posibles conjuntos de entrenamiento donde sólo varía la variable dependiente. En general lo que se puede estimar, es el ***optimismo medio****.*

Proposición 1 *Para funciones error usuales se verifica:*

$$w = \frac{2}{n} \sum_{i=1}^{n} \mathrm{Cov}(y_i, \hat{y}_i)$$

Mientras mayor sea el ajuste de los datos, es decir, mientras más fuerte sea el efecto de una observación sobre su predicción, mayor será el sesgo (inferior) del error empírico.

Dado además que en general la covarianza será no negativa, usualmente $w \geq 0$.

Existen criterios basados en el optimismo medio, que proporcionan una estimación del error de generalización interno para aquellos casos en los que el conjunto de datos no es demasiado grande, lo que permite aproximar el error de generalización.

Estas medidas han sido usadas tradicionalmente como criterios de bondad de ajuste para ayudar a seleccionar un modelo concreto dentro de una familia de modelos.

Estos criterios pueden emplearse en la construcción de modelos que dependan de la selección de algún parámetro de ajuste, como en el caso de un modelo de regresión lineal, sería el orden. Así se construiran los distintos modelos para los distintos valores considerados de dicho parámetro y se seleccionaría el que ofrezca un menor valor para el criterio usado.

Se puede considerar la distancia de Kullback-Leibler como el origen de los criterios de información. KL mide la distancia entre una distribución o modelo " verdadero" y un modelo candidato. A partir de esta medida se presentaron otras que explotan la función de log-verosimilitud, destacando el criterio de información de Akaike (AIC) , y el criterio de Información Bayesiano (BIC).

2.4. Métodos de Remuestreo

2.4.1. Introducción

Una alternativa a los criterios de información, suponiendo que se dispone de suficientes datos, es realizar una partición de la muestra disponible en tres partes: un conjunto de entrenamiento, un conjunto de validación y un conjunto test.

El conjunto de entrenamiento es usado para ajustar los modelos, el conjunto de validación para determinar su complejidad , y por último el conjunto test es usado para la estimación del error de generalización del modelo finalmente seleccionado.

Figura 2.5: Partición del conjunto de datos

Hastie et al (2008)[21] sugieren que una partición típica podría ser 50 % para entrenamiento, y 25 % para los conjuntos de validación y test respectivamente. Se puede observar una ilustración de dicha partición en la figura 2.5.

Si por ejemplo se va a construir un árbol de clasificación, el conjunto de validación serviría para ajustar el tamaño del árbol, o en el caso de un problema de regresión polinomial, para estimar el tamaño del polinomio.
Un error sería usar el conjunto test repetidamente para la selección del modelo de menor error test, pues esto suele conducir a una subestimación del verdadero error de generalización.

Si la muestra disponible no es suficientemente grande, la anterior división puede conducir a submuestras excesivamente pequeñas, y una alternativa a esto, podría ser recurrir a métodos de remuestreo.

2.4.2. Validación Cruzada

Probablemente el método más simple y más usado para estimar el error de predicción es validación cruzada [21]. Esta estrategia implica dividir el conjunto de observación en dos partes, una de ellas será usada como conjunto de entrenamiento, y la otra parte será usada como conjunto de validación o también conocido como conjunto *hold-out*.[8]

Validación Cruzada con partición en k-subconjuntos

Este método divide aleatoriamente el conjunto de observación en k subconjuntos de tamaño aproximadamente igual. El primer subconjunto se usa como conjunto de validación, del cual ya se puede calcular el error cuadrático medio MSE_1, y el resto como conjunto de entrenamiento para ajustar el modelo.
Este procedimiento es repetido k veces, cada vez con un subconjunto diferente para el conjunto de validación y se obtiene $MSE_1, \cdots, MSE_k$.
Por lo tanto, se puede calcular una estimación del error como:

$$CV_{(k)} = \frac{1}{k} \sum_{i=1}^{k} MSE_i$$

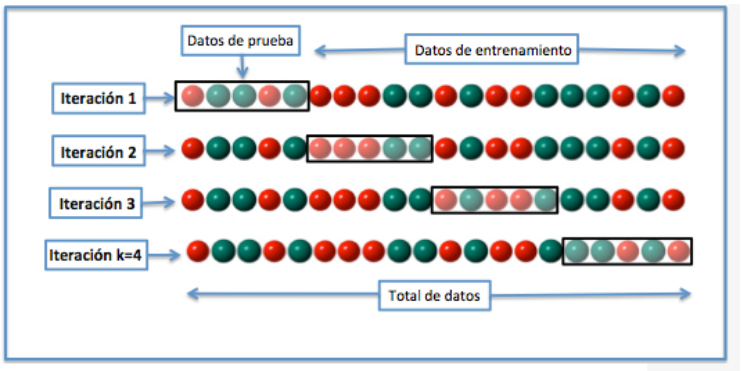

Figura 2.6: Partición en 4 subconjuntos

Este método es conceptualmente simple, y fácil de implementar.
En la imagen 2.6 se puede observar el método descrito anteriormente, para $k = 4$.

Se suele usar $k = 10$. Asintóticamente, la determinación de la complejidad del modelo mediante AIC es equivalente al uso de Validación Cruzada.

Jackknife

El Jackknife, o también conocido como Leave-one-out Cross-Validation, tal y como el método anterior divide el conjunto de observaciones en dos partes. Sin embargo, en lugar de crear dos subconjuntos de tamaño comparable, una única observación (x_1, y_1) se utiliza para el conjunto de validación, y las observaciones restantes $\{(x_2, y_2), \cdots, (x_n, y_n)\}$ componen el conjunto de entrenamiento.

Ya que (x_1, y_1) no es usada para el proceso de ajuste, $MSE_1 = (y_1 - \hat{y}_1)^2$, proporciona una aproximación para el error del conjunto test, aunque esta es una estimación pobre ya que es altamente variable al estar basada sobre una única observación.

Se repite el proceso con (x_2, y_2) obteniendo así $MSE_2 = (y_2 - \hat{y}_2)^2$. Repitiendo dicho procedimiento n veces, se producen n errores cuadrados, $MSE_1, \cdots, MSE_n$.

El estimador *jackknife* para el test MSE, es la mediana de esos n errores estimados, es decir

$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^{n} MSE_i$$

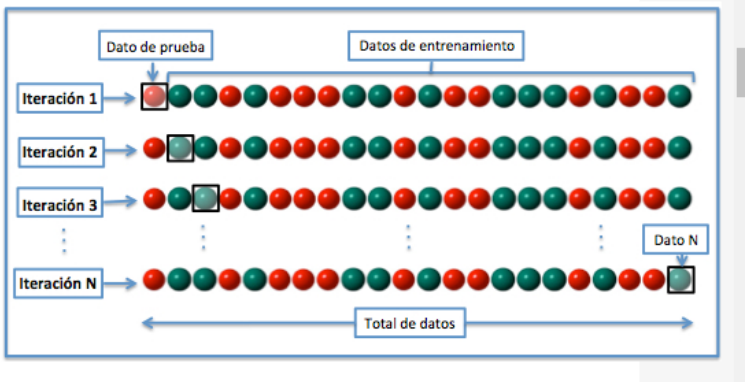

Figura 2.7: Jackknife

En la figura 2.7 puede observarse el procedimiento descrito para la elección de cada uno de los conjuntos (entrenamiento o prueba y validación) en cada iteración. Este método es un caso particular de Validación Cruzada con partición en k-subconjuntos, para $k = n$.

Capítulo 3

Árboles de decisión y Random Forests

3.1. Introducción

Los árboles de decisión son técnicas de predicción basadas en un proceso recursivo de particionamiento del conjunto de datos. Nos ayudan a tomar la decisión más acertada, desde un punto de vista probabilístico, ante un abanico de posibles decisiones. Su principal ventaja es la facilidad de interpretación del modelo resultante que consiste en una colección de reglas de tipo " *if then* ".Veáse la figura 3.1.

```
if Petal.Length< 2.45 Species=setosa
 else
   if Petal.Width< 1.75 Species= versicolor
   else  Species=virginica
```

Figura 3.1: Árbol de clasificación de los datos iris.

Un árbol de decisión es un modelo predictivo basado en un proceso de particionamiento recursivo, el cual a partir de unas entradas produce una salida. Los valores que pueden tomar las salidas pueden ser valores discretos (árboles de clasificación) o continuos (árboles de regresión).

Para definir un algoritmo de particionamiento recursivo son necesarios los siguientes elementos:

1. Un conjunto de cuestiones, generalmente se construyen árboles binarios, para hacer las divisiones.
2. Un criterio para evaluar la bondad de las divisiones y seleccionar la mejor en cada etapa o nodo.
3. Una regla de parada.
4. Una regla para asignar cada nodo terminal a una clase (clasificación) o bien una predicción cuantitativa (regresión).

En este trabajo se considera el algoritmo " CART " (Breiman et al.(1984))[2]. Otros usados en Ciencias de la Computación son C4.5 y C5.0 ambos sugeridos por Quinlan (1993 y 2004 respectivamente) [16], [17].

3.2. Árboles de clasificación

3.2.1. Particiones

Se puede abordar dos casos, uno en el que las variables predictoras son cuantitativas o cualitativas ordinales, y otro que engloba las variables cualitativas nominales.

En el primero de estos, en principio son posibles infinitas divisiones, cada una de ellas asociada a un punto de corte c, que divide el rango de valores de la variable en dos regiones disjuntas:

$$\{x < c\} \quad y \quad \{x \geq c\},\ c \in \mathbb{R},$$

pero dado el tamaño finito del conjunto de datos, solo hay que considerar las divisiones

$$\{x < x_{(j)}\},\ j = 2, \cdots, n \qquad \text{con } x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(n)}$$

Equivalentemente, en el caso cuantitativo se pueden considerar los puntos medios de los intervalos definidos por la muestra ordenada.

En el caso de las variables cualitativas nominales, dado un subconjunto S de categorías de variables, se plantea

$$\{x \in S\} \quad y \quad \{x \notin S\}$$

En un principio se deben considerar $2^{K-1}-1$ subconjuntos para un total de K categorías. En la metodología " CART " *(Classification and Regression Trees)* [2] se contemplan algunos métodos para reducir el número de subconjuntos a explorar, obteniendo así un mayor ahorro computacional.

3.2.2. Evaluación de la bondad de las divisiones

Los criterios para elegir la mejor división posible de un nodo t se basan en las llamadas funciones de impureza o diversidad, la idea es conseguir nuevos nodos donde la mayoría de observaciones pertenezcan a la misma clase. Para ello introducimos la siguiente definición:

Definición 7 *Dado un problema de clasificación con C clases, una función de impureza es una función ϕ definida sobre el conjunto*

$$P = \{(p_1, \cdots, p_c) : \sum_{j=1}^{c} p_j = 1, \quad p_j \geq 0 \quad \forall j\}$$

verificando :

1. *ϕ alcanza su único máximo en $(1/C, \cdots, 1/C)$.*

2. *ϕ alcanza su mínimo exclusivamente en los vectores $(1, 0, \cdots, 0)$, $(0, 1, 0, \cdots, 0), \cdots, (0, \cdots, 1)$.*

3. *ϕ es una función simétrica de sus argumentos.*

Por ejemplo, se supone un problema de clasificación con 4 clases, todas con la misma frecuencia de aparición en el nodo inicial, es decir, la estimación de la función de probabilidad sería $(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$.
Una buena regla de particionamiento del nodo sería aquella que diese lugar a dos nuevos nodos con particiones $(\frac{1}{2}, \frac{1}{2}, 0, 0)$ y $(0, 0, \frac{1}{2}, \frac{1}{2})$.
A su vez, la división ideal de cada uno de estos nodos sería llegar a proporciones $(1, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0)$ y $(0, 0, 0, 1)$.

Para formalizar esta idea se verá algunas funciones de impureza, como son el índice de Gini y la entropía.

Índice De Gini

Dado un problema de clasificación con C clases, se define el índice de diversidad de Gini como:

$$\phi(p_1, \cdots, p_c) = \sum_{i \neq j} p_i p_j = 1 - \sum_{i=1}^{C} p_i^2$$

Una interpretación interesante de este índice es la siguiente: supongamos que en el nodo t cada objeto es asignado aleatoriamente a la clase j con probabilidad p_j. Entonces, la probabilidad de clasificación errónea coincide con el índice de Gini.

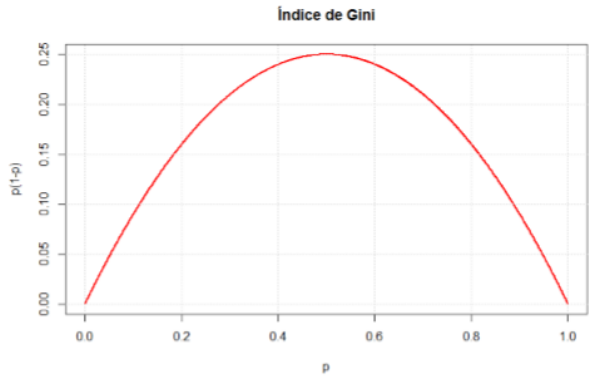

Figura 3.2: Ilustración Índice de Gini para C=2

Este índice es usado por el paquete "*rpart*" [20] de R como criterio de selección de la mejor regla de división.

En cada nodo debe estimarse la función de probabilidad de las C categorías, de modo que se tendrá lo que puede denominarse medida de impureza en el nodo t, dado por la expresión:

$$\phi(t) = \phi(p(Y = 1/t), \cdots, p(Y = C/t))$$

Denotando por Y a la variable respuesta del problema de clasificación.

Además si se divide un nodo t en dos nodos hijos, t_I y t_D se tiene, para funciones de impureza cóncavas que:

$$\Delta\phi = p(t)\phi(t) - p(t_I)\phi(t_I) - p(t_D)\phi(t_D) \geq 0$$

Puede observarse un ejemplo en la figura 3.2 para el caso de $C = 2$.

Por lo tanto de la propiedad anterior, se deduce que la impureza (ponderada) se mantiene o se reduce con cualquier división del nodo. La regla de decisión óptima sería seleccionar el par (variable, división), que maximiza la reducción de la impureza $\Delta\phi$.

Entropía

Otra función de impureza es la entropía, definida por:

$$\phi(p_1, \cdots, p_c) = -\sum_{i=1}^{C} p_i \log(p_i)$$

Este concepto fue introducido por Shannon y Weaver (1949) [19]. En el ámbito de la Teoría de la Información, puede interpretarse como el mínimo número de bits de información necesarios para codificar la clasificación de un objeto arbitario en el nodo t.

De acuerdo a los primeros elementos del algoritmo, la construcción de un árbol de clasificación consiste en:

- Comenzar con el nodo raíz, donde se incluyen todos los casos.
- Determinar el par (variable,división) que conduzca a una mayor reducción de la impureza.
- Aplicar a cada nodo el anterior proceso, hasta que se verifiquen las condiciones de finalización.

3.2.3. Regla de Parada

Determinar una regla de parada depende de la implementación. En principio podría lograrse un árbol con un nodo final por cada observación, pero en tal caso se obtiene un modelo muy complejo, que si bien ofrece un error empírico nulo, a cambio cabe esperar una baja capacidad de generalización.

En la librería "rpart" [20] de R , se consideran tres parámetros de parada en la opción rpart.control:

- **minsplit:** que denota el mínimo número de observaciones que debe haber en cada nodo para intentar dividirlo.

- **minbucket:** denota el mínimo número de observaciones en un nodo terminal.
- **cp:** donde cp denota la mínima cantidad de reducción de impureza que deben producir las divisiones.

3.2.4. Regla de clasificación en cada nodo

La regla de clasificación $d(t)$ para cualquier caso que pertenezca a un nodo viene dada por la categoría de mayor probabilidad estimada:

$$d(t) = \arg \max_j p\left(Y = j/t\right)$$

En general, estas probabilidades son estimadas mediante las frecuencias relativas de las C clases en el nodo. Se puede incorporar una función de probabilidades a priori de las C clases, que se utilizará en la estimación de las probabilidades de cada categoría en cada nodo y en la estimación de la probabilidad de cada nodo $p(t)$.

También es posible considerar costes de errores de clasificación, algo de interés en aquellas situaciones donde no tiene la misma gravedad los errores asociados a clasificaciones incorrectas. Esta técnica también es usada como método para el tratamiento de datos no balanceados.

3.2.5. Recortes de árboles (*Prunning*)

El tamaño del árbol (número de nodos terminales) es un parámetro que controla la complejidad del modelo.
Un árbol muy pequeño puede no capturar la estructura de los datos, y un árbol muy grande puede sobreajustar *(overfitting)*.
La estrategia preferida consiste en construir un árbol grande T_0, y posteriormente recortar dicho árbol, más aún, se genera una selección de subárboles de entre los cuales se elige el mejor según cierto criterio.

Cualquier árbol $T \subset T_0$ que puede obtenerse recortando T recibe el nombre de subárbol.

Una idea sería elegir aquel subárbol cuyo error de VC sea el mínimo. Sin embargo, para tener en cuenta la variabilidad del proceso de remuestreo, se recomienda usar la llamada **Regla 1-ES**. Esta técnica induce a tomar el menor subárbol tal que su error VC sea menor que:

$$\textbf{Error VC mínimo} + 1\,\textbf{desviación típica}.$$

3.3. Árboles de regresión

Se considera ahora una variable cuantitativa que se desea predecir a partir del conocimiento de un conjunto de variables predictoras. En cada nodo terminal, la predicción de la variable respuesta se calcula mediante la media de todos los valores observados de dicha variable evaluada en los casos que pertenecen a dicho nodo.

La diferencia fundamental con los árboles de clasificación se encuentra en los criterios de selección de las divisiones en cada etapa, ya que no tiene sentido el concepto de impureza. En este caso se trabaja con la suma de cuadrados del error (SCE), también llamada desviación. Antes de la primera división, el valor de la SCE es:

$$\text{SCE} = \sum_{i=1}^{n} (y_i - \overline{y})^2$$

Si se realiza una división a partir de una variable, de forma similar a la vista en los problemas de clasificación, se tendrán los nuevos nodos, t_I y t_D, de modo que:

$$\underbrace{\sum_{i \in t_I} (y_i - \overline{y_I})^2 + \sum_{i \in t_D} (y_i - \overline{y_D})^2}_{SCE_W} + \underbrace{n_I(\overline{y_I} - \overline{y})^2 + n_I(\overline{y_I} - \overline{y})^2}_{SCE_B}$$

Por lo tanto el criterio de selección dado un nodo t, consiste en seleccionar el par (variable, división) que produzca el menor valor para SCE_W.
Equivalentemente, se busca el mayor valor de SCE_B.

Es posible aplicar procedmientos similares, de construcción y recorte, a los árboles de clasificación.

3.4. Variables Suplentes (“*surrogates*”)

La metodología CART y la librería “rpart” [20] incluyen variables sustitutas para cada posible partición seleccionada en cada nodo.
En particular, es de interés para clasificar o predecir observaciones donde falte el valor de alguna de las variables “ titulares ”.

Sea p_I y p_D la probabilidad de que la regla primaria envíe un caso al nodo hijo izquierdo o derecho respectivamente, en general estimadas mediante las frecuencias relativas correspondientes.
Si usamos la regla sustituta como predictora de la partición $\{t_I, t_D\}$, la decisión será la definida por la partición de dicha regla, es decir, t_I si la regla sustituta manda el caso a $t_{I'}$, y t_D en caso contrario. Esta regla tiene probabilidad de acierto $P[t_I \cap t_{I'}] + P[t_D \cap t_{D'}]$, que es estimada precisamente, mediante el coeficiente de concordancia. Veáse la definición 8.

Definición 8 ***El coeficiente de concordancia*** *es una estimación de la probabilidad de que una regla de división sea capaz de particionar del mismo modo que la regla “primaria” de particionamiento del árbol. Definido como:*

$$CC = P[t_I \cap t_{I'}] + P[t_D \cap t_{D'}]$$

Se calcula ocomo la proporción de casos coincidentes para ambas particiones.

Definición 9 *Esta medida predictiva de asociación (**adj**) evalúa la reducción relativa de error que ofrece la regla sustituta como predictora de la partición definida por la regla primaria. Esta dada por:*

$$adj = \frac{\text{mín}\{p_I, p_D\} - (1 - CC)}{\text{mín}\{p_I, p_D\}}$$

Las variables son ordenadas según el coeficientes de concordancia. Una regla simple para predecir si un caso de un nodo va a ser asignado al nodo hijo izquierdo o derecho mediante la regla primaria es la definida por máx $\{p_I, p_D\}$ cuyo error de clasificación es por tanto mín $\{p_I, p_D\}$.

Si $adj < 0$, se descarta esa regla. Además se impone que debe enviar al menos dos observaciones a cada nodo hijo.

3.5. Random Forests (Bosques aleatorios)

3.5.1. Introducción

Las técnicas de combinación de modelos consisten en la agregación de cierto número de modelos para obtener una clasificación o predicción a partir de los distintos clasificadores o predictores previamente generados. La construcción de cada uno de los modelos elementales puede depender de aspectos como por ejemplo de qué manera se tome la muestra de entrenamiento (muestras bootstrap[1], reponderación), de la selección de variables y la elección del modelo (por ejemplo árboles de decisión, o una mezcla de modelos de distinta naturaleza). Esta memoria se centra en el Random Forests.

Random Forests [3] construye un número elevado de árboles de decisión. Para clasificar un sujeto, se elige la clase más votada entre los distintos árboles. En problemas de regresión, se calcula la media de todas las predicciones.

3.5.2. Construcción de cada árbol

Dado un conjunto de entrenamiento de tamaño n con p variables predictoras:

1. Seleccionar una muestra con reemplazamiento de tamaño n de la muestra de entrenamiento (*muestra bootstrap*).

2. En cada nodo, se eligen aleatoriamente $m < p$ variables predictoras, y se particiona según la mejor división entre esas m variables.

3. Cada árbol se construye hasta el mayor tamaño posible (no hay recorte).

La tasa de error del modelo final depende de dos elementos:

- La correlación entre dos árboles cualesquiera del bosque. A mayor correlación, menor error.

- La fuerza de cada árbol en el bosque. Un árbol con pequeña tasa de error es un clasificador fuerte. Aumentar la fuerza de los árboles individuales disminuye la tasa de error del bosque.

[1]Una muestra bootstrap es una muestra aleatoria de tamaño n extraída con reemplazamiento a partir de la muestra disponible.

Se ha de notar que reducir m , es decir, reducir el número de las variables predictoras , reduce tanto la correlación como la fuerza, y viceversa. Este es el único parámetro a ajustar respecto al cual Random Forests es sensible, puede ser ajustado con la ayuda de procedimientos de validación cruzada.

3.5.3. Ventajas del Random Forests

Algunas de las ventajas de la construcción del modelo mediante Random Forests son:

- Proporciona muy buenos resultados en los estudios empíricos.
- Se ejecuta de forma eficiente sobre grandes bases de datos.
- Puede manejar miles de variables sin tener que eliminar ninguna.
- Da estimaciones de la importancia de cada variable.
- Genera internamente un estimador insesgado del error de generalización durante el proceso de construcción (OOB). Veáse la subsección 3.5.4
- Dispone de un método efectivo de estimación de valores perdidos.
- Se calculan prototipos que dan información sobre la relación entre las variables y la clasificación.
- Calcula proximidades entre pares de casos que pueden emplearse en el análisis de conglomerados, identificación de "outliers", o escalamiento de los datos para obtener representaciones gráficas.

3.5.4. Estimador OOB

El estimador OOB ("*Out Of the Bag*") del error de predicción o clasificación se basa en el aprovechamiento de las observaciones no incluidas en cada muestra bootstrap. Al igual que los estimadores basados en un conjunto test, proporciona una estimación insesgada del error de generalización.

En problemas de clasificación, donde está inscrito este trabajo, se puede obtener dicho estimador de la siguiente forma:

Sea un conjunto de entrenamiento $\mathbf{D} = \{D_i = (X_i, Y_i),\ i = 1, 2, \cdots, n\}$, donde las clases de cada caso se identifican mediante las clases Y_i.

Para $b = 1$ hasta B:
Generar una muestra bootstrap $\mathbf{D}^*$ del conjunto $\mathbf{D}$.
Sea $\mathbf{D}_b = \{D_i / D_i \notin \mathbf{D}_*\} = \mathbf{D} - \mathbf{D}^*$.

Construir el modelo A_b sobre $\mathbf{D}^*$.
Aplicar A_b a cada elemento de $\mathbf{D}_b$.
Aplicar lo anterior a el siguiente b.

Para cada caso D_i, se tendrán las predicciones para dicho caso proporcionadas por aquellos modelos en cuyo conjunto de entrenamiento no se incluye D_i. De forma similar al procedimiento de Validación Cruzada, se obtiene la predicción agregada para D_i mediante la clase donde más veces es clasificado dicho caso:

$$V_i = \arg \max_{j \in \{1,2,\cdots,K\}} \{ \#[A_b(X_i) = j / D_i \in \mathbf{D}_b] \}$$

A partir de las clasificaciones " *Out of the bag* ", se define el estimador OOB de la tasa de error :

$$\boxed{OOB = \frac{1}{n} \sum_{i=1}^{n} I(Y_i \neq V_i)}$$

Por lo tanto la tasa de error OOB es la proporción de casos cuya clasificación OOB no coincide con su clase real.
La clasificación OOB de un caso es la que se obtiene al recorrer los modelos ajustados sobre muestras bootstrap que no incluyen a ese caso.

3.5.5. Importancia de cada variable y próximidad entre casos

Como se mencionó anteriormente, una de las ventajas del Random Forests es que es posible dar una estimación de la importancia de cada variable de la siguiente forma:

Sean p variables predictoras.
Para $b = 1$ hasta B
Generar una muestra bootstrap $\mathbf{D}^*$ del conjunto $\mathbf{D}$.
Sea $\mathbf{D}_b = \{D_i / D_i \notin \mathbf{D}_*\} = \mathbf{D} - \mathbf{D}^*$.

Construir el modelo A_b sobre $\mathbf{D}^*$.

Aplicar A_b a cada elemento de $\mathbf{D}_b$.
Calcular el número de clasificaciones correctas sobre los elementos de $\mathbf{D}_b$.

$$C_b = \#\{\, [A_b(X_i) = Y_i / D_i \in \mathbf{D}_b] \,\}$$

Para $j = 1, 2, \cdots, p$
Permutar aleatoriamente los valores de la variable j entre los casos del conjunto $\mathbf{D}_b$.
Calcular de nuevo el número de clasificaciones correctas sobre $\mathbf{D}_b$, sea C_{jb}.
Calcular $R_{jb} = C_b - C_{jb}$.
Repetir para el siguiente j.
Repetir para el siguiente b.
Decrecimiento medio de la precisión para cada variable:

$$IM_j = \frac{1}{B} \sum_{b=1}^{B} R_{jb}$$

Cada decrecimiento medio se divide por la desviación típica de las R_{jb}.

Otra medida de importancia se basa en el criterio de Gini. Cada vez que una variable se elige para dividir un nodo, el índice de Gini se mantiene o se reduce, la media de los decrecimientos para cada variable es lo que se conoce como **decrecimiento medio del índice de Gini**.

Por otro lado, la proximidad se calcula para todos los casos, incluso aquellos incluidos en el conjunto test, si se ha definido.

Definición 10 *Se define* ***la proximidad entre dos casos****, y es denotada como $C(i, j)$, a la proporción de árboles en los que ambos casos se incluyen en el mismo nodo terminal.*

Capítulo 4

Medidas de Rendimiento

Para medir el rendimiento, o *performance* en inglés, se necesita de alguna medida que cuantifique la calidad del clasificador, permitiendo además, la comparación y selección.

Existen muchas medidas de *performance*, la mayoría basadas (directa o indirectamente) en la matriz de confusión. Dicha matriz es una herramienta que permite la visualización de los resultados de un algoritmo que se emplea en el aprendizaje supervisado. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa el total de instancias de cada fila en la clase real.

Algunas de estas medidas se centran en aspectos concretos de la clasificación, mientras otras tratan de sintetizar el comportamiento global del clasificador.

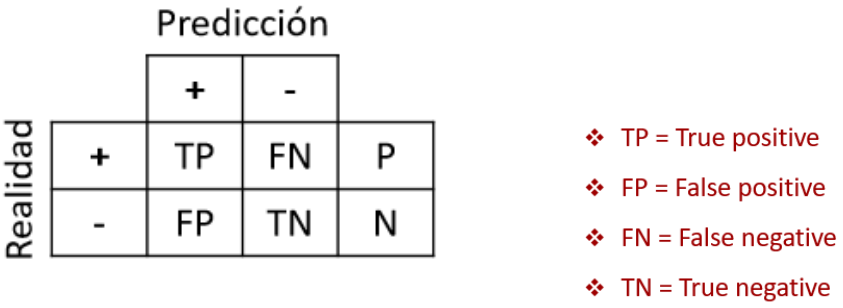

Figura 4.1: Matriz de confusión

Para la definición de las siguientes medidas se usará la notación presente en la figura 4.1. Algunas de las más importantes podrían ser:

Precisión y sensitividad. La precisión y la sensitividad son dos medidas diferentes, pero suelen usarse conjuntamente. La precisión representa el porcentaje de casos positivos predichos correctamente. Por otro lado, la sensitividad representa la capacidad para detectar una condición correctamente, por ejemplo, en el caso de la detección de una enfermedad, se define la sensitividad como la capacidad de nuestro estimador para dar como casos positivos los casos realmente enfermos; proporción de enfermos correctamente identificados.

$$\text{Precisión } = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Sensitividad} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\text{P}}$$

Especificidad. Se refiere a la capacidad del modelo para excluir correctamente una condición. Para el caso de una enfermedad, nos indica la capacidad de nuestro estimador para dar como casos negativos los casos realmente sanos, es decir, proporción de sanos correctamente identificados.

$$\text{Especificidad} = \frac{\text{TN}}{\text{TN} + \text{FP}} = \frac{\text{TN}}{\text{N}}$$

Índice Kappa de Cohen[7]. Medida de concordancia entre las categorías pronosticadas por el clasificador y las categorías observadas, que tiene en cuenta las posibles concordancias debidas al azar. Por ejemplo, si se considera el caso en el que un *individuo A* lanza una moneda a el aire, siendo positivo que dicho resultado sea cara, negativo el caso opuesto y se hace lo mismo con otro *individuo B*. Es previsible encontrar un promedio del orden del 50 % de coincidencias, por lo que cabe considerar en el cálculo de dicho índice el papel que tuvo el azar en la coincidencia de observaciones.

Se supone **D** como la suma de los elementos de la diagonal de la matriz de confusión.

$$\text{Índice Kappa} = \frac{D_{observado} - D_{azar}}{D_{perfecto} - D_{azar}}$$

Figura 4.2: Concordancia

Si se observa en la imagen 4.2 las distintas matrices de confusión asociadas a: las predicciones observadas en primer lugar, las perfectas en segundo, cuya matriz de confusión es una matriz diagonal, es decir, no hay errores en la clasificación, y por último la matriz de confusión debida a el azar.
Se puede calcular dicha medida de concordancia como:

$$K = \frac{6-7}{14-7} = -0.142857$$

La interpretación del índice kappa varía en función del resultado obtenido, es decir:

- Valor 1: Concordancia perfecta.
- Valor 0: Concordancia debida al azar.
- Valor negativo: Concordancia menor que la que cabría esperar por azar.

AUC. Otra de las medidas más utilizadas en problemas de clasificación es calcular el valor del área bajo la curva ROC (*Receiver Operating Characteristic*).

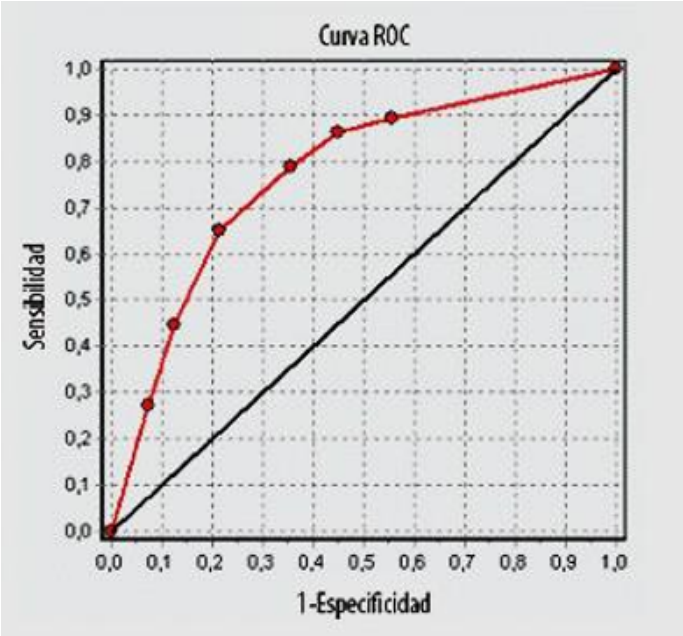

Figura 4.3: Gráfica curva ROC

Usualmente las reglas de clasificación se basan en la estimación de la probabilidad de una clase de interés dado el vector de predictores.

Suponiendo dos clases etiquetadas como " P " (que se corresponde con (+), y " N " (-), y dado un valor punto de corte p_c, la regla se puede expresar como:

$$\hat{\phi} = \begin{cases} P & \text{si } \hat{P}\,[1/x] \geq p_c \\ N & \text{si } \hat{P}\,[1/x] < p_c \end{cases}$$

Por tanto para cada punto de corte se tiene una regla de clasificación distinta, a la que le corresponderá una sensitividad y especificidad.

La representación gráfica de la sensitividad, frente a los valores 1 - Especificidad para cada valor de t recibe el nombre de curva ROC. Un ejemplo es ilustrado en la figura 4.3.

El área bajo la curva o **AUC** (*area under curve*, en inglés) es una medida de rendimiento del modelo base de clasificación, de modo que cuanto mas cercano a 1 sea dicha área, mejor será el modelo.

- Punto (0,0): Clasificador que clasifica correctamente todas las instancias de la clase - , pero falla con todas las de la clase +.
- Punto (1,1): Clasificador que clasifica correctamente todas las instancias de la clase + , pero falla con todas las de la clase -.
- Punto (0,1): Clasificador ideal.

Estas son algunas de las medidas que se usarán en la práctica para la selección de la técnica que más se adecúe con el fin de solventar el problema producido por las clases no balanceadas.

Capítulo 5

Tratamiento de datos no balanceados

En esta sección se estudian algunos de los métodos más relevantes para la mejora de predicciones en clases no balanceadas.

5.1. Cortes Alternativos

Cuando hay dos categorías posibles de resultados, un método para incrementar la precisión de las muestras en la clase minoritaria es determinar puntos de corte que conduzcan a valores de especificidad/sensitividad mayores.
Existen varias técnicas para determinar un nuevo corte:

1. Si hay un objetivo en particular en el que debemos conocer la especificidad y sensitividad, este punto puede ser encontrado en la curva ROC, y su corte correspondiente puede determinarse.

2. Otro método es encontrar el punto más cercano al ideal $(0, 1)$, es decir, el de menor distancia para el modelo óptimo, que esta asociado con el punto de la curva ROC más arriba a la izquierda.

3. Otro método es el índice de Youden‘s J , que mide el rendimiento de las muestras correctamente, es decir, mide la diferencia entre la tasa de verdaderos positivos (TP) y falsos positivos (FP). Un buen test debe tener alta esta diferencia, o de forma equivalente, tener una alta especificidad y sensitividad.

Este índice puede ser calculado para cada corte de la curva ROC. El valor máximo de dicho índice, puede ser usado como criterio para seleccionar el valor de corte óptimo, es decir, el corte asociado con el mayor valor de índice Youden puede mostrar un mayor rendimiento en relación con el valor predeterminado de 0.5.

En nuestro análisis, el punto de corte alternativo para el modelo se obtuvo del conjunto de validación.
Es importante usar un conjunto de datos independiente para obtener el corte, especialmemte para una muestra de tamaño pequeño.
Si se usan las predicciones del conjunto de entrenamiento es probable que haya un sesgo optimista en las probabilidades de clase que conduzca a evaluaciones inexactas de sensitividad y especificidad.
Si se usa el conjunto test, ya no es una fuente imparcial para juzgar el rendimiento del modelo. Vale la pena señalar que la base del modelo no cambia. Se usan los mismos parámetros del modelo.
El principal impacto de los cortes alternativos es tener un equilibrio entre los tipos de errores particulares.

5.2. Métodos de Remuestreo

Cuando hay conocimientos a priori de la clase no balanceada, un método sencillo para reducir su impacto en el conjunto de entrenamiento es seleccionar una muestra de dicho conjunto para tener frecuencias de sucesos aproximadamente iguales durante la colección de datos iniciales. Básicamente en lugar de lidiar con el conjunto de datos no balanceado, se puede intentar equilibrar las frecuencias de las clases. Haciendo esto se elimina el problema del desequilibrio fundamental que afecta a el conjunto de entrenamiento.

Si un método de muestreo a priori no es posible, hay métodos de muestreo post-hoc que pueden atenuar el efecto del desequilibrio en el conjunto de entrenamiento. Dos métodos de muestreo post-hoc son Downsampling y Upsampling. Upsampling es una técnica que simula o atribuye datos adicionales para mejorar el equilibrio de las clases, mientras Downsampling es una técnica que reduce el tamaño de la muestra para mejorar el equilibrio de dichas clases, también puede darse la hibridación de ambas.

5.2.1. Upsampling

El método de remuestreo Upsampling u Oversampling consiste en balancear la distribución de los datos añadiendo ejemplos a la clase minoritaria (veáse la figura 5.1). *Ling and li(1998)*[11] proporcionó un método de *upsampling* en el cual los casos de las clases minoritarias se muestrean con reemplazamiento (*bootstrap*) hasta que cada clase tenga aproximadamente el mismo número, dejando igual la clase mayoritaria.

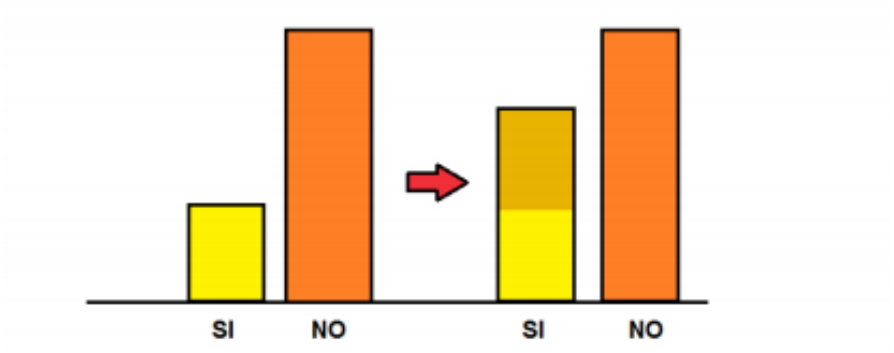

Figura 5.1: Representación gráfica técnica Upsampling

Si hacemos esto, algunas muestras de la clase minoritaria podrán aparecer en el conjunto de entrenamiento con una frecuencia bastante alta mientras cada muestra en la clase mayoritaria tiene una única realización en los datos.

Es decir, el algoritmo es el siguiente:

Se genera un conjunto de datos balanceado de tamaño $2N$ formado por:

1. Los N casos de la clase mayoritaria.
2. Una muestra aleatoria con reemplazamiento (*bootstrap*) de tamaño N extraida del conjunto de los n casos de la clase minoritaria.

Esta técnica, además de aumentar el tiempo de procesado de los datos, el principal problema de este método tiene lugar cuando generamos ejemplos ruidosos que se producen al realizar un elevado número de réplicas de ciertas instancias, modificando en exceso la distribución de la clase minoritaria, y que puede derivar en un incremento desmesurado de las probabilidades de dicha clase.

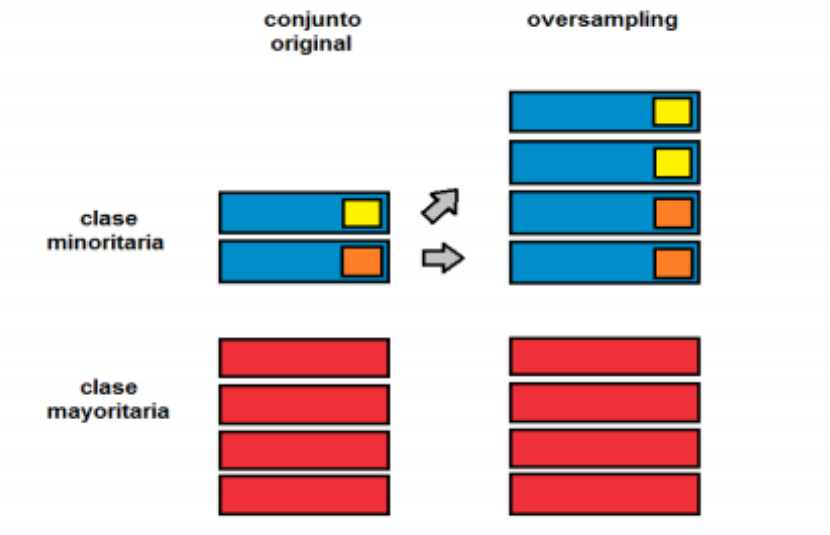

Figura 5.2: Representación gráfica de la técnica Random Oversampling

Se puede observar en la figura 5.2 como las instancias de la clase minoritaria han sido duplicadas para obtener el mismo número de instancias en ambas clases.

5.2.2. Downsampling

Consiste en balancear la distribución de los datos eliminando instancias de la clase mayoritaria. Veáse la figura 5.3.

A pesar de su sencillez y de la reducción del tiempo de procesado de los datos, existe el riesgo de eliminar elementos de la muestra potencialmente importantes en el proceso de clasificación, por eso se han desarrollado métodos capaces de realizar una selección inteligente sobre los elementos del conjunto de datos de la clase mayoritaria.

La idea generalizada en la que se basan estos algoritmos es suponer que las instancias próximas entre si tienen mayor probabilidad de pertenecer a la misma clase.

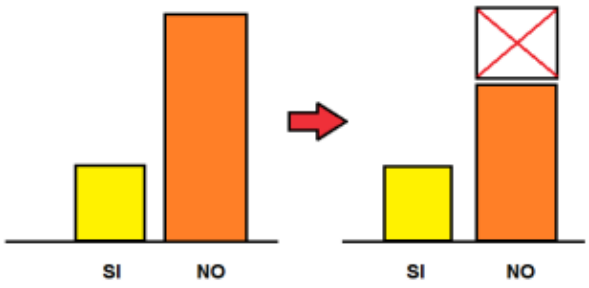

Figura 5.3: Representación gráfica técnica Downsampling

El algoritmo usado es el siguiente:
Se genera un conjunto de datos balanceados de tamaño $2n$ formado por:

1. Los n casos de la clase minoritaria.
2. Una selección aleatoria de n casos entre los N de la clase mayoritaria.

5.3. SMOTE

La técnica SMOTE (Synthetic Minority Oversampling Method), [5] genera nuevas instancias de la clase minoritaria interpolando los valores de las instancias minoritarias más cercanas a una dada. Veáse 5.4.

Descripción del Método

Es un proceso, descrito por Chawla et al (2002) [6], de muestreo de datos en el cual, la clase minoritaria es sobremuestreada (upsampled) para crear muestras sintéticas en lugar de sobremuestrear con remplazamiento.

Este método esta inspirado en una técnica probada por Ha & Bunke (1997). Ellos crearon un conjunto de entrenamiento extra para la interpretación de los datos reales.

Chawla propuso generar muestras sintéticas. La clase minoritaria es sobremuestreada (upsampling), SMOTE sintetiza nuevos casos, un punto es seleccionado aleatoriamente de la clase minoritaria y se determinan sus k vecinos más cercanos *(K-NNs)*.

El nuevo punto de datos sintético es una combinación aleatoria de los predictores del punto seleccionado aleatoriamente y sus vecinos.

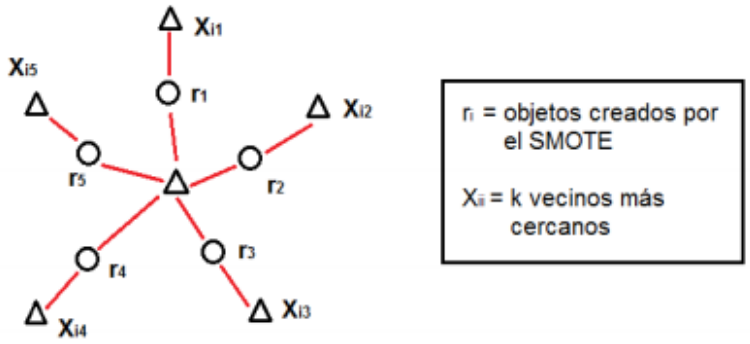

Figura 5.4: Synthetic Minority Oversampling Technique

La implementación de R usa cinco vecinos más cercanos. Las muestras sintéticas crean el clasificador, para obtener regiones de decisión más grandes y menos específicas. El efecto es que los árboles de decisión generalizan mejor.

Algoritmo. Veamos a continuación el algoritmo:

Algoritmo SMOTE (T,F,k)
Entrada: Número de muestras de la clase minoritaria T; cantidad de SMOTE $F\,\%$; Número de vecinos cercanos k
Salida: (F/100) T muestras sintéticas de la clase minoritaria.

1. (Si F es menor que 100 %, aleatorizar las muestras de la clase minoritaria, ya que solo se aplicará SMOTE a un porcentaje aleatorio de dichas muestras)
2. **If** $F < 100$
3. Aleatorizar las T muestras de la clase minoritaria
4. $T = (F/100) * T$
5. $F = 100$
6. **endif**
7. $F = (F/100)$ (*La cantidad de SMOTE es asumida para ser múltiplos enteros de 100)
8. k =Número de vecinos cercanos
9. *numattrs* = Número de atributos
10. *Sample* [][]: conjunto de muestras de la clase minoritaria original
11. *newindex* : mantiene la cuenta del número de muestras sintéticas generadas, inicialmente es 0

12. *Synthetic*[ ][ ]: conjunto de muestras sintéticas
(Computar k vecinos cercanos para cada una de las muestras de la clase minoritaria)
13. **for** $i \leftarrow 1$ **to** T
14. Computar k vecinos cercanos para i, y guardar los índices en el *nnarray*
15. *Populate*$(F, i, nnarray)$
16. **endfor**
Populate$(F, i, nnarray)$ (Función para generar muestras sintéticas)
17. **while** $F \neq 0$
18. Elegir un número aleatorio entre 1 y k, llamado *nn*. Este paso elige uno de los k vecinos más cercanos de i
19. **for** *attr* $\leftarrow 1$ **to** *numattrs*
20. Computar: *di f* = *Sample*[*nnarray*[*nn*]][*attr*]-*Sample*[*i*][*attr*]
21. Computar: *gap* = número aleatorio entre 0 y 1
22. *Synthetic*[*newindex*][*attr*]=*Sample*[*i*][*attr*] + *gap*∗*di f*
23. **endfor**
24. *newindex*
25. $F = F - 1$
26. **endwhile**
27. **return**(*fin del *Populate*)

5.4. Método ROSE

5.4.1. Descripción del método

ROSE (*Menardi y Torelli, 2004*)[9] proporcionó un nuevo método para tratar los problemas de estimación del modelo y la precisión en la evaluación del aprendizaje no balanceado. Esto se hizo generando nuevos ejemplos artificiales para las clases, de acuerdo con un método suavizado de *bootstrap*.

Consideramos un conjunto de entrenamiento Tn, de tamaño n, cuyas filas son de la forma (x_i, y_i), $i = 1, \cdots, n$, donde y_i pertenecen a el conjunto $\{Y_0, Y_1\}$, y x_i se tratan de supuestas realizaciones de un vector aleatorio x definido en $\mathbb{R}^d$, con una función de densidad desconocida $f(x)$.

Sea el número de unidades en la clase Y_j, con $j = 0, 1$, denotado por $n_j < n$. El procedimiento ROSE para generar un nuevo ejemplo artificial consiste en los siguientes pasos:

1. Seleccionar $y^* = Y_j$ con probabilidad π_j.
2. Seleccionar $(x_i, y_i) \in T_n$ tal que $y_i = y^*$.
3. Muestrear x^* de $K_{H_j}(\,.\,, x_i)$, con K_{H_j} una distribución de probabilidad centrada en x_i y matriz de varianzas y covarianzas H_j.

Esencialmente, tomamos una observación del conjunto de entrenamiento, que pertenecerá a una de las dos clases (mayoritaria o minoritaria), y generamos una nueva muestra (x^*, y^*) en su vecino, donde la forma del vecino es determinada por la forma de contorno de K y su ancho es debido a H_j.

$$\begin{aligned}
\hat{f}\,(x|y^* = y_j) &= \sum_{n_j}^{i=1} p_i P_r(x|x_i) \\
&= \sum_{n_j}^{i=1} \frac{1}{n_j} P_r(x|x_i) \\
&= \sum_{n_j}^{i=1} \frac{1}{n_j} K_{H_j}(x - x_i)
\end{aligned}$$

Puede mostrarse fácilmente que, dada una selección de la clase Y_j, la generación de nuevas muestras de Y_j, de acuerdo a el algoritmo ROSE, se corresponde con la generación de datos del estimador de densidad del núcleo de $f(x|Y_j)$, siendo el núcleo K y la matriz de suavización H_j.

Repitiendo los pasos de 1 a 3, m veces, se produce un nuevo conjunto de entrenamiento sintético T_m^*, de tamaño m, donde el nivel de desequilibrio es definido por las probabilidades π_j (si $\pi_j = 1/2$, entonces aproximadamente el mismo número de muestras pertenecen a ambas clases).
El tamaño m puede establecerse en el conjunto de entrenamiento de tamaño n, o ser elegido de cualquier manera.

ROSE combina las ténicas de *downsampling* y *upsampling*, generando una muestra de datos aumentada (especialmente en las muestras pertenecientes a la clase " rara "), lo que ayuda al clasificador a estimar una regla de clasificación más precisa, pues se dirigirá la misma atención a ambas clases.

Además de mejorar el proceso de aprendizaje, la generación sintética de nuevas muestras de un estimador de densidad condicionadas de dos clases, también puede ayudar a la precisión de la estimación.

5.4.2. Selección de la matriz $\mathbf{H}_j$

En principio el problema es bastante crítico, ya que las diferentes elecciones de la matriz conduce a K_{H_j} mayor o menor, un ejemplo podría ser
" conjuntos de vecinos "(*neighborhoods*) más grandes o más pequeños de observaciones a partir de las cuales se generan las muestras sintéticas.

La idea de la base de estos métodos es minimizar un criterio de optimalidad, como por ejemplo el AMISE.

Definición 11 *El AMISE (Asymptotic Mean Integrated Squared Error) viene dado por la expresión:*

$$AMISE(s) = \frac{R(K)}{ns} + \frac{1}{4} m_2(K)^2 s^4 R(f'') \ , \ con\ f\ función\ de\ densidad.$$

$$siendo\ R(g) = \int g(x)^2\, dx \ \ para\ una\ función\ g, \ \ y\ m_2(K) = \int x^2 K(x) dx.$$

Entre las diversas alternativas posibles, se considera una función tipo núcleo gaussiana $K(u; x, H_j) \sim \mathrm{N}_d(0, Id)$, dada por la expresión:

$$K(u; x, H_j) = \frac{1}{2\pi^{d/2} |H_j|^{1/2}}\, e^{-\frac{1}{2}(u-x)^t H_j^{-1}(u-x)}$$

con matrices de suavización diagonales de la forma:

$$H_j = \mathrm{diag}\ (h_1^{(j)}, \cdots, h_d^{(j)})$$

y minimizamos el AMISE bajo la suposición de que las densidades condicionales subyacentes a los datos, siguen una distribución normal, de modo que:

$$h_q^{(j)} = \left(\frac{4}{(d+2)n} \right)^{\frac{1}{d+4}} \hat{\theta}_q\ (j), \qquad \forall q = 1, \cdots, d, \quad j = 1, 2.$$

y donde $\hat{\theta}_q\ (j)$ es la estimación de la desviación estándar de la muestra de la q-dimensión de las observaciones pertenecientes a la clase Y_j.

Aunque esta estrategia pueda parecer ingenua, se han obtenido buenos resultados en la mayoría de los casos en los que el método ha sido probado. De hecho, no estamos interesados en crear una estimación exacta de las densidades condicionales, sino generar nuevos ejemplos en un vencindario razonable de las observaciones.

En resumen, ROSE generaliza la técnica estándar de *upsampling*, reemplazando los ejemplos raros, al permitir la generación de algunos clones de los datos observados, sin producir lazos.

Por otro lado ROSE, incluye *upsampling* con reemplazamiento como caso especial cuando los elementos de H_j tienden a 0.

Capítulo 6

Aplicación en R

6.1. Lectura y preparación de los datos

Se usarán las técnicas descritas en el capítulo anterior con el fin de tratar con un conjunto de datos no balanceado **("Insolvencia.RData")** disponible en el espacio de trabajo. Esta muestra es tomada de un conjunto de pequeñas y medianas empresas del Reino Unido, en la que puede observarse un conjunto de 2877 observaciones y 16 variables. La variable dependiente "*failed_insolvent*" toma los valores "Yes" o "No" según la empresa haya sido declarada o no como insolvente.

```
load("Insolvencia.RData")
attach(datos)
table(failed_insolvent)


## failed_insolvent
##   No  Yes
## 2740  137


datos$failed_insolvent =
           factor(as.character(datos$failed_insolvent),
           levels = rev(levels(datos$failed_insolvent)))
table(datos$failed_insolvent)


##
##  Yes   No
##  137 2740
```

En dicha muestra puede observarse que las frecuencias de clase son muy diferentes para la pertencia a las distintas clases (declararse insolvente o no), con un total de 137 observaciones declaradas como insolventes (clase minoritaria), y el resto de instancias pertenecientes a la clase mayoritaria. La clase minoritaria es la de mayor interés, por ello interesa que aparezca en primer lugar. Hay una gran cantidad de ratios[1] identificados como útiles para predecir los incumplimientos de pagos de las empresas.

Veamos en primer lugar un resumen de estos ratios o variables:

```
summary(datos)
```

```
##       CETL              STLTA              TLCA
##  Min.   :-0.9480   Min.   :0.00000   Min.   :  0.0000
##  1st Qu.: 0.0000   1st Qu.:0.00000   1st Qu.:  0.5454
##  Median : 0.3333   Median :0.00000   Median :  0.9873
##  Mean   : 2.2197   Mean   :0.06748   Mean   :  2.7228
##  3rd Qu.: 1.2353   3rd Qu.:0.00000   3rd Qu.:  1.6046
##  Max.   :80.5000   Max.   :0.80000   Max.   :159.0000
##       NWTA                QACA             NCNW            CRATIO
##  Min.   :-34.50000   Min.   :0.0000   Min.   :-4.700   Min.   : 0.0000
##  1st Qu.: -0.03448   1st Qu.:0.9231   1st Qu.: 0.000   1st Qu.: 0.7143
##  Median :  0.13750   Median :1.0000   Median : 0.750   Median : 1.0814
##  Mean   : -0.03672   Mean   :0.8758   Mean   : 3.318   Mean   : 2.6059
##  3rd Qu.:  0.52863   3rd Qu.:1.0000   3rd Qu.: 2.800   3rd Qu.: 1.9714
##  Max.   :  1.00000   Max.   :1.0000   Max.   :34.000   Max.   :51.0000
##      CASHTA              PRTA                 TCTD
##  Min.   :0.00000   Min.   :-18.250000   Min.   :  0.0000
##  1st Qu.:0.01333   1st Qu.:  0.000000   1st Qu.:  0.9643
##  Median :0.21569   Median :  0.131783   Median :  1.0000
##  Mean   :0.34290   Mean   :  0.008251   Mean   : 12.1243
##  3rd Qu.:0.63636   3rd Qu.:  0.500000   3rd Qu.:  1.0000
##  Max.   :1.00000   Max.   :  1.000000   Max.   :185.0000
##       TCTL              TDTA           ln_assets         CHNW_new
##  Min.   :0.0000   Min.   :0.00000   Min.   : 8.987   Min.   :-7.3000
##  1st Qu.:0.9167   1st Qu.:0.02857   1st Qu.: 9.680   1st Qu.:-0.2857
##  Median :1.0000   Median :0.23256   Median :10.491   Median : 0.0000
##  Mean   :0.8457   Mean   :0.33327   Mean   :10.538   Mean   : 0.1763
##  3rd Qu.:1.0000   3rd Qu.:0.57895   3rd Qu.:11.350   3rd Qu.: 0.3077
##  Max.   :1.0000   Max.   :1.00000   Max.   :12.333   Max.   :18.8000
##    CHNWTA_new       failed_insolvent
##  Min.   :-66.5000   Yes: 137
##  1st Qu.: -0.2889   No :2740
##  Median :  0.0000
##  Mean   : -0.2265
##  3rd Qu.:  0.2533
##  Max.   : 11.9000
```

[1]Matemáticamente, un ratio o razón es la relación entre dos variables. En las empresas los ratios se usan para analizar los estados contables y financieros.

En la figura 6.1 se pueden observar los diferentes ratios financieros de nuestro conjunto de datos.

Ratio	Formulation	Formulación
CTEL	Capital employed / Total liabilities	Capital invertido / Pasivos totales
STLTA	Short-term liabilities / Total assets	Pasivos a corto plazo /Activos Totales
TLCA	Total liabilities / Current assets	Pasivos Totales / Activos Corrientes
NWTA	Net Worth / Total assets	Patrimonio neto / Activos Totales
QACA	Quick assets / Current assets	liquidez a corto plazo / Activos corrientes
NCNW	Cash / Net Worth	Efectivo / Patrimonio neto
CRATIO	Current assets / Current liabilities	Activos corrientes / Pasivos corrientes
CASHTA	Cash / Total assets	Efectivo / Activos totales
PRTA	Retained profit / Total assets	Beneficios retenidos / Activos totales
TCTD	Trade creditors / Trade debtors	Acreedores comerciales / Deudores comerciales
TCTL	Trade creditors / Total liabilities	Acreedores comerciales / Pasivos totales
TDTA	Trade debtors / Total assets	Deudores comerciales / Activos totales
LN_ASSETS	Neperian logarithm of total assets	logaritmo neperiano de Activos totales
CHNW	Percentage change in net worth	Porcentaje de cambio en el patrimonio neto
CHNWTA	Percentage change in net worth / Total assets	Porcentaje de cambio en el patrimonio neto / Activos totales

Figura 6.1: Ratios Financiero

A continuación se hará una partición del conjunto de datos, con el fin de obtener los conjuntos de entrenamiento, test y validación, donde se usará un porcentaje de 60 % , 25 %, y 15 % respectivamente. Veáse la subsección 2.4.1. Instalamos el paquete *"caret"* [14], que incluye una serie de funciones que facilitan el uso de una gran variedad de métodos de clasificación y regresión.

```
#install.packages("caret")
library(caret)


set.seed(156)
```

```
n=nrow(datos)
indices=1:n
ient=sample(indices,floor(n*0.6))
ival=sample(setdiff(indices,ient),floor(n*0.15))
itest=setdiff(indices,union(ient,ival))

training  = datos[ient,]
validation  = datos[ival,]
testing     = datos[itest,]
training_valid=rbind(training,validation)
dim(training)


## [1] 1726   16


dim(validation)
```

```
## [1] 431  16

dim(training_valid)

## [1] 2157   16

dim(testing)

## [1] 720  16

Index= 1:nrow(training)  #SE USARA EN trainControl

predictors = names(training)[names(training) != "failed_insolvent"]
predictors  #Nombres de las variables predictoras

##  [1] "CETL"       "STLTA"      "TLCA"       "NWTA"       "QACA"
##  [6] "NCNW"       "CRATIO"     "CASHTA"     "PRTA"       "TCTD"
## [11] "TCTL"       "TDTA"       "ln_assets"  "CHNW_new"   "CHNWTA_new"

testResults = data.frame(failed_insolvent =
                           testing$failed_insolvent)
validResults = data.frame(failed_insolvent =
                            validation$failed_insolvent)
```

fiveStats devuelve las medidas Accuracy, kappa, AUC, ROC, sensitividad y especificidad, se define ya que será de utilidad más adelante.

```
fiveStats = function(...)
  c(twoClassSummary(...), defaultSummary(...))
```

6.2. Random Forests sobre los datos originales

Para el ajuste, debido a que el conjunto de validación es muy reducido, vamos a usar Validación Cruzada con partición en tres partes, gracias a la opción *number = 3* de la función *trainControl*, incluida en el paquete "*caret*"[14].

```
set.seed(1410)
ctrlcv = trainControl(method = "cv",number=3,
                      classProbs = TRUE,
                      summaryFunction = fiveStats,
                      verboseIter=TRUE)
```

Para el ajuste del modelo Random Forests, se usará la función *train* incluida en el paquete *" caret "*, citado anteriormente. Esta función establece una serie de parámetros para una serie de rutinas en árboles de clasificación y regresión. Con el parámetro *tuneLength* = 3, se prueban 3 valores diferentes para *mtry*, con el fin de encontrar el valor óptimo basándonos en estos 3 valores. Para la elección del modelo óptimo se ha usado la sensitividad (*metric = "Sens"*).

```
rfFit = train(failed_insolvent ~ ., data = training,
              method = "rf",
              trControl = ctrlcv,
              do.trace=TRUE,
              tuneLength=3,
              metric = "Sens")
```

```
rfFit
```

```
## Random Forest
##
## 1726 samples
##   15 predictor
##    2 classes: 'Yes', 'No'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
```

```
## Summary of sample sizes: 1151, 1150, 1151
## Resampling results across tuning parameters:
##
##   mtry  ROC        Sens        Spec       Accuracy   Kappa
##    2    0.7579356  0.00000000  0.9993939  0.9542301  -0.00111828
##    8    0.7492566  0.01282051  0.9987879  0.9542301   0.02006285
##   15    0.7456617  0.01282051  0.9987879  0.9542301   0.02006285
##
## Sens was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 8.
```

```
#probabilidades estimadas de "Yes"
validResults$RF = predict(rfFit, validation,
                          type = "prob")[,1]
testResults$RF = predict(rfFit, testing,
                         type = "prob")[,1]
```

Como puede observarse, según la sensitividad, el modelo óptimo es para mtry=8.

A continuación vamos a calcular las medidas de rendimiento en el conjunto test. Para ello es necesario la instalación del paquete *" pROC "*[22].

```
#istall.packages(pROC)
library(pROC)


## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##    cov, smooth, var


rfTestROC = roc(testResults$failed_insolvent,
                testResults$RF,levels =
                  rev(levels(testResults$failed_insolvent)))

rfTestROC
plot(rfTestROC)
```

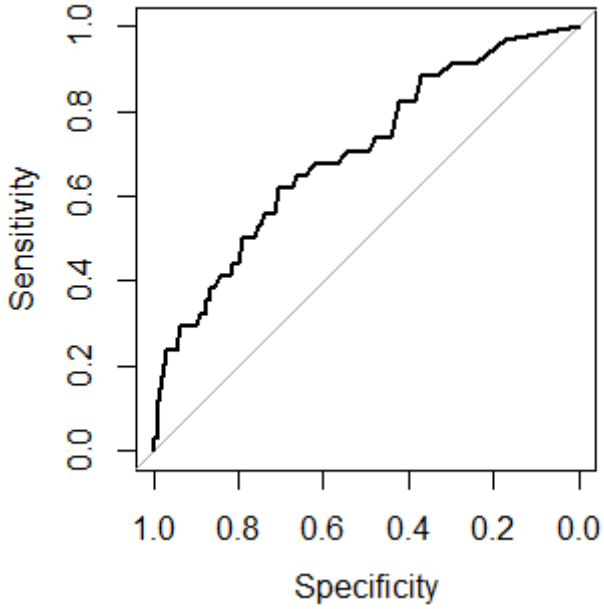

Figura 6.2: Curva ROC modelo Random Forests

```
##
## Call:
## roc.default(response = testResults$failed_insolvent,
## predictor = testResults$RF,
## levels = rev(levels(testResults$failed_insolvent)))
##
## Data: testResults$RF in 686 controls
##(testResults$failed_insolvent No)<
## < 34 cases (testResults$failed_insolvent Yes).
## Area under the curve: 0.6996
```

El área bajo la curva ROC es 0.699, sabemos que el ajuste del modelo será mejor para áreas bajo la curva cercanas a 1, por lo tanto el ajuste del modelo, no es bueno. Se puede ver una representación gráfica de dicha curva en la figura 6.2.

Veamos la matriz de confusión:

```
rfTestCM = confusionMatrix(predict(rfFit, testing),
                           testResults$failed_insolvent)
rfTestCM


## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes  No
##        Yes   0   0
##        No   34 686
##
##                Accuracy : 0.9528
##                  95% CI : (0.9346, 0.9671)
##     No Information Rate : 0.9528
##     P-Value [Acc > NIR] : 0.5455
##
##                   Kappa : 0
##  Mcnemar's Test P-Value : 1.519e-08
##
##             Sensitivity : 0.00000
##             Specificity : 1.00000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.95278
##              Prevalence : 0.04722
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.00000
##       Balanced Accuracy : 0.50000
##
##        'Positive' Class : Yes
##
```

Hasta ahora hemos usado en nuestro análisis el algoritmo Random Forests, que es uno de los ensembles de árboles de decisión más conocidos y usados en clasificación. Este algoritmo combina clasificadores basados en árboles de decisión aleatorios, de forma que la clase a la que pertenece una nueva instancia vendrá determinada por el voto mayoritario de cada uno de estos árboles. Aplicado a nuestro conjunto de datos hemos podido ver que los resultados no son buenos (Sensitividad 0 y especificidad 1) desde el punto de vista estadístico. En las secciones siguientes describiremos nuestra propuesta para hacer frente a estos problemas mediante el uso de las técnicas descritas en el capítulo 5.

6.3. Puntos de corte alternativos

Como se describió anteriormente en la memoria, cuando hay dos categorías posibles de resultados, un método para incrementar la precisión de la clasificación de las muestras en la clase minoritaria, es determinar puntos de corte que conduzcan a valores de sensitividad mayores. Hay varias formas de determinar un nuevo corte, el cual no tiene porque ser necesariamente el 0.5 que se induce por defecto. En este método es necesario usar un conjunto de datos independiente para determinar el nuevo corte, por lo que usaremos el conjunto de validación. Puede verse una representación gráfica de la curva ROC en el conjunto de validación en la figura 6.3.

```
rfvalidROC = roc(validResults$failed_insolvent, validResults$RF,
                 levels = rev(levels(validResults$failed_insolvent)))

rfvalidROC


##
##Call:
##roc.default(response = validResults$failed_insolvent,
##predictor = validResults$RF,levels =
##rev(levels(validResults$failed_insolvent)))
##
##Data: validResults$RF in 406 controls
##(validResults$failed_insolvent No) < 25 cases
##
##(validResults$failed_insolvent Yes).
## Area under the curve: 0.7213
```

Para el cálculo del corte óptimo, usaremos dos métodos:

- Encontrar el punto más cercano al ideal $(0, 1)$ en la curva ROC, es decir, el de menor distancia para el modelo óptimo, que está asociado con el más arriba a la izquierda.

  ```
  rfThresh = coords(rfvalidROC, x = "best", ret="threshold",
                    best.method="closest.topleft")
  rfThresh

  ## [1] 0.045
  ```

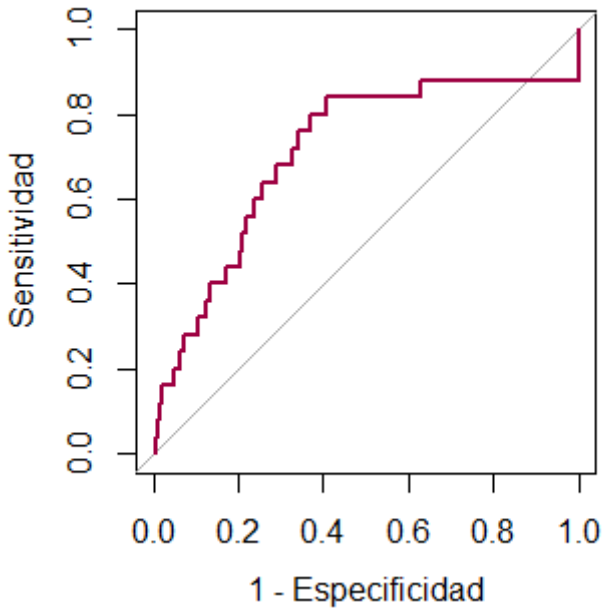

Figura 6.3: Curva ROC conjunto de validación

- Obtener el punto de corte por el método de Youden, que maximiza la distancia a la diagonal.

```
rfThreshY = coords(rfvalidROC, x = "best", ret="threshold",
                   best.method="youden")

rfThreshY

## [1] 0.031
```

Los puntos de corte son distintos, en nuestro caso, para cada método. Por simplificación, se usará para el resto del estudio el corte obtenido a través del método que encuentra el punto más cercano a el ideal $(0, 1)$ de la curva ROC, es decir, 0.045. Veamos la matriz de confusión tras usar como método para clasificar los datos dicho corte para los conjuntos de validación y test respectivamente.

```
validResults$rfAlt = factor(ifelse(validResults$RF > rfThresh,
                                   "Yes", "No"))

testResults$rfAlt = factor(ifelse(testResults$RF > rfThresh,
                                  "Yes", "No" ))
```

```
rfAltvalidCM = confusionMatrix(validResults$rfAlt,
validResults$failed_insolvent)


rfAltvalidCM


## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes  No
##        Yes  19 138
##        No    6 268
##
##                Accuracy : 0.6659
##                  95% CI : (0.6192, 0.7103)
##     No Information Rate : 0.942
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1208
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.76000
##             Specificity : 0.66010
##          Pos Pred Value : 0.12102
##          Neg Pred Value : 0.97810
##              Prevalence : 0.05800
##          Detection Rate : 0.04408
##    Detection Prevalence : 0.36427
##       Balanced Accuracy : 0.71005
##
##        'Positive' Class : Yes
##
```

```
rfAltTestCM = confusionMatrix(testResults$rfAlt,
testResults$failed_insolvent)


rfAltTestCM


## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes  No
##        Yes  22 231
##        No   12 455
##
##                Accuracy : 0.6625
##                  95% CI : (0.6267, 0.697)
##     No Information Rate : 0.9528
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0764
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.64706
##             Specificity : 0.66327
##          Pos Pred Value : 0.08696
##          Neg Pred Value : 0.97430
##              Prevalence : 0.04722
##          Detection Rate : 0.03056
##    Detection Prevalence : 0.35139
##       Balanced Accuracy : 0.65516
##
##        'Positive' Class : Yes
##
```

Puede observarse que se obtuvo unos valores más equilibrados para la especificidad y sensitividad en el caso del corte obtenido por el método descrito anteriormente (sensitividad 0.64706 y especificidad 0.66327), en comparación con el corte al 0.5 (sensitividad 0 y especificidad 1) reflejado a continuación.

```
rfTestCM = confusionMatrix(predict(rfFit, testing),
testResults$failed_insolvent)

rfTestCM


## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes  No
##        Yes   0   0
##        No   34 686
##
##                Accuracy : 0.9528
##                  95% CI : (0.9346, 0.9671)
##     No Information Rate : 0.9528
##     P-Value [Acc > NIR] : 0.5455
##
##                   Kappa : 0
##  Mcnemar's Test P-Value : 1.519e-08
##
##             Sensitivity : 0.00000
##             Specificity : 1.00000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.95278
##              Prevalence : 0.04722
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.00000
##       Balanced Accuracy : 0.50000
##
##        'Positive' Class : Yes
##
```

En la figura 6.4, puede verse situados en la curva ROC distintos rendimientos para distintos umbrales para el conjunto Test.

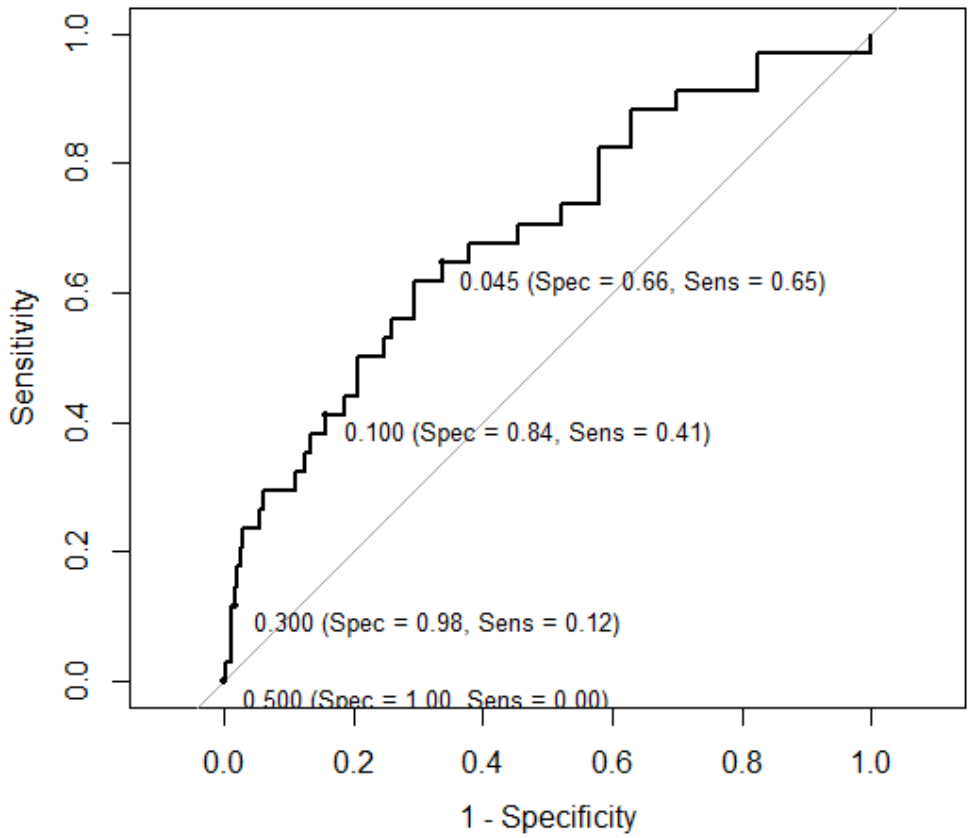

Figura 6.4: Curva ROC conjunto Test con distintos puntos de corte

6.4. Generación de muestras balanceadas

6.4.1. Downsampling

Para este método se mantienen todos los casos de la clase minoritaria, y se elige aleatoriamente una muestra con el mismo número de casos en la clase mayoritaria. Para mantener el uso de entrenamiento/validación lo aplicaremos por separado a esos dos conjuntos.

```
set.seed(1237)
downSampled = downSample(training[, -ncol(training)],
                         training$failed_insolvent)
dim(downSampled)


## [1] 156  16
```

```
table(downSampled$Class)


##
## Yes  No
##  78  78
```

Para el conjunto de entrenamiento, podemos observar que aplicando la técnica de downsampling, tanto la clase mayoritaria como la minoritaria están formadas ahora por 78 instancias en cada una de las clases. Aplicando esto mismo a el conjunto de Validación:

```
downSampled_valid = downSample(validation[, -ncol(validation)],
                          validation$failed_insolvent)
dim(downSampled_valid)


## [1] 50 16


table(downSampled_valid$Class) #25 de cada clase


##
## Yes  No
##  25  25
```

Como es una muestra pequeña, vamos a unir el conjunto de validación y entrenamiento dividido anteriormente, por lo tanto solo tendremos dos conjuntos, training_valid y test.

```
downSampled_train_valid=rbind(downSampled,downSampled_valid )
dim(downSampled_train_valid)


## [1] 206  16


table(downSampled_train_valid$Class)


##
## Yes  No
## 103 103
```

6.4.2. Upsampling

Por otra parte Upsampling, deja todas las instancias de la clase mayoritaria, y aumenta el número de casos de la clase minoritaria muestreando con reemplazamiento.

```
set.seed(1237)
upSampled = upSample(training[, -ncol(training)],
                     training$failed_insolvent)
dim(upSampled)


## [1] 3296   16


table(upSampled$Class)


##
##  Yes   No
## 1648 1648
```

En este caso, para el conjunto de entrenamiento, obtenemos 1648 muestras de cada clase. Veamos ahora en el conjunto de Validación:

```
upSampled_valid = upSample(validation[, -ncol(validation)],
                     validation$failed_insolvent)
dim(upSampled_valid)


## [1] 812  16


table(upSampled_valid$Class) #406 casos de cada clase


##
## Yes  No
## 406 406
```

Y como en el caso de Downsampling, unimos el conjunto de entrenamiento y Validación en uno solo.

```
upSampled_train_valid=rbind(upSampled,upSampled_valid )
dim(upSampled_train_valid)


## [1] 4108   16
```

```
table(upSampled_train_valid$Class) #2054 instancias en cada clase


##
##  Yes   No
## 2054 2054
```

6.4.3. SMOTE

Esta técnica incluye Upsampling y Downsampling, para ello, es necesario la instalación del paquete " DMwR " [12].Un paquete que incluye funciones y datos, de minería de datos con R.

```
#install.packages("DMwR")
library(DMwR)
```

Se aplica la técnica SMOTE.

```
set.seed(1237)
smoted = SMOTE(failed_insolvent ~ ., data = training)
smoted_valid = SMOTE(failed_insolvent ~ ., data = validation)
smoted_train_valid=rbind(smoted,smoted_valid)
table(smoted_train_valid$failed_insolvent)


##
## Yes  No
## 309 412
```

Aplicando la técnica SMOTE, a la unión del conjunto de entrenamiento y validación, obtenemos una muestra con 309 casos para la clase minoritaria "Yes", y 412 de la mayoritaria "No".

6.4.4. ROSE: Un paquete para aprendizajes binarios

Para esta parte de la memoria es necesaria la instalación del paquete " ROSE " [15], dicho paquete es usado para aprendizajes binarios no balanceados.

```
install.packages("ROSE")


library(ROSE)
```

La primera ayuda proporcionada por el paquete involucra la creación de nuevos datos artificiales por remuestreo en ambas clases. La función `ovun.sample` integra algunas técnicas de resampling. Está dotado con el argumento `method`, en el que podemos elegir entre $c(‘‘$`over`$’’, ‘‘$`under`$’’, ‘‘$`both`$’’)$.

La opción ‘‘`over`’’, determina muestras obtenidas mediante Upsampling, también conocido como oversampling o sobremuestreo, con reemplazamiento de la clase minoritaria hasta que o bien el tamaño específico N sea alcanzado o las muestras de dicha clase tengan probabilidad p de ocurrir.

```
roseoverN<-ovun.sample (failed_insolvent ~ .,
                        data = training,
                        method = "over"
                       ,N =3296,seed=1)$data

table(roseoverN$failed_insolvent)


##
##   No  Yes
## 1648 1648


table(training$failed_insolvent)


##
##  Yes   No
##   78 1648


#mismo número de instancias de cada clase
```

Alternativamente, se puede designar otro método por el argumento p, el cual representaría la probabilidad de la clase positiva (clase minoritaria) en la nueva muestra. En este caso, la proporción de casos positivos será aproximadamente igual a la probabilidad especificada p.

```
roseoverp<-ovun.sample (failed_insolvent ~ .,
                         data = training,method = "over"
                        ,p=0.5,seed=1)$data

table(roseoverp$failed_insolvent)


##
##   No  Yes
## 1648 1592


table(training$failed_insolvent)


##
##  Yes   No
##   78 1648


#podemos observar un desbalanceo menor
```

Ahora tendriamos 1648 para la clase "No" y 1592 para "Yes", frente a las 137 instancias de "No" y 2740 de "Yes" del conjunto de datos antes de aplicar la técnica descrita anteriormente, es decir, un conjunto de datos más equilibrado.

Similar a la opción "`over`" , la opción "`under`" determina muestras mediante submuestreo sin remplazamiento de la clase mayoritaria hasta que o bien el número N sea determinado, o bien "p". Cuando usamos `method`= "`under`", se obtiene una muestra de tamaño reducido. Por ejemplo, si usamos $p = 0.5$.

```
roseunderp <-ovun.sample(failed_insolvent ~ .,
                          data = training,method = "under"
                         ,p=0.5,seed=1)$data

table(roseunderp$failed_insolvent)


##
##  No Yes
##  77  78
```

```
table(training$failed_insolvent)


##
##  Yes   No
##   78 1648


#podemos observar un desbalanceo menor
```

Cuando usamos el método `"both"`, la clase minoritaria es sobremuestreada, mientras que la mayoritaria es submuestreada sin reemplazamiento. En este caso, los argumentos N y p sirven para estabilizar la cantidad de sobremuestreo y submuestreo.

```
roseboth<-ovun.sample(failed_insolvent ~ .,
                      data = training,method = "both"
                       ,p=0.5,seed=1)$data

table(roseboth$failed_insolvent)


##
##  No Yes
## 899 827


table(training$failed_insolvent)


##
##  Yes   No
##   78 1648


   #podemos observar un desbalanceo menor
```

La generación de datos de acuerdo con ROSE intenta eludir las trampas de los métodos de remuestreo anteriores dibujando un nuevo conjunto de datos sintético, posiblemente equilibrado, de las dos estimaciones de densidad de kernel condicional de las clases. Involucra los argumentos, `formula` , `data`, `N`, y `p`, veámoslo:

```
rose <- ROSE(failed_insolvent~.,
             data=training, seed=3)$data

table(rose$failed_insolvent)


##
##  No Yes
## 847 879


rose_valid = ROSE(failed_insolvent ~ .,
                  data = validation)$data

rose_train_valid=rbind(rose,rose_valid)

table(rose_train_valid$failed_insolvent)


##
##   No  Yes
## 1068 1089
```

6.5. Construcción de los modelos Random Forests

A continuación se va a construir un modelo Random Forests sobre los conjuntos de datos balanceados tras aplicar cada una de las técnicas anteriores. En primer lugar, se construirá sobre el conjunto de datos obtenido tras aplicar la técnica **Downsampling**.

```
ctrcvdown=ctrlcv
ctrcvdown$index=list(1:nrow(downSampled))
ctrcvdown$indexFinal=1:nrow(downSampled)

set.seed(1410)
rfDown = train(Class ~ ., data = downSampled_train_valid,
               "rf",
               trControl = ctrcvdown,
               ntree = 100,do.trace=TRUE,
               tuneLength = 3,
               metric = "ROC")
```

```
rfDown


## Random Forest
##
## 206 samples, 156 used for final model
##  15 predictor
##   2 classes: 'Yes', 'No'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 156
## Resampling results across tuning parameters:
##
##   mtry  ROC     Sens  Spec  Accuracy  Kappa
##    2    0.6872  0.88  0.52  0.70      0.40
##    8    0.6680  0.72  0.60  0.66      0.32
##   15    0.6992  0.72  0.60  0.66      0.32
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 15.
```

Hemos elegido como métrica para la elección de el mejor modelo, la curva ROC, por ello la elección es mtry = 15. Vamos a guardar ROC de test. En la figura 6.5 se puede observar una ilustración de la curva ROC tras aplicar el método Random Forests a el conjunto de datos obtenido mediante Downsampling.

```
testResults$RFdown = predict(rfDown, testing,
                             type = "prob")[,1]

rfDownROCT = roc(testResults$failed_insolvent, testResults$RFdown,
                 levels = rev(levels(validResults$failed_insolvent)))
rfDownROCT


##
## Call:
##roc.default(response = testResults$failed_insolvent,
##predictor = testResults$RFdown,
##levels = rev(levels(validResults$failed_insolvent))
##
## Data: testResults$RFdown in 686 controls
## (testResults$failed_insolvent No) < 34 cases
## (testResults$failed_insolvent Yes).
## Area under the curve: 0.7113
```

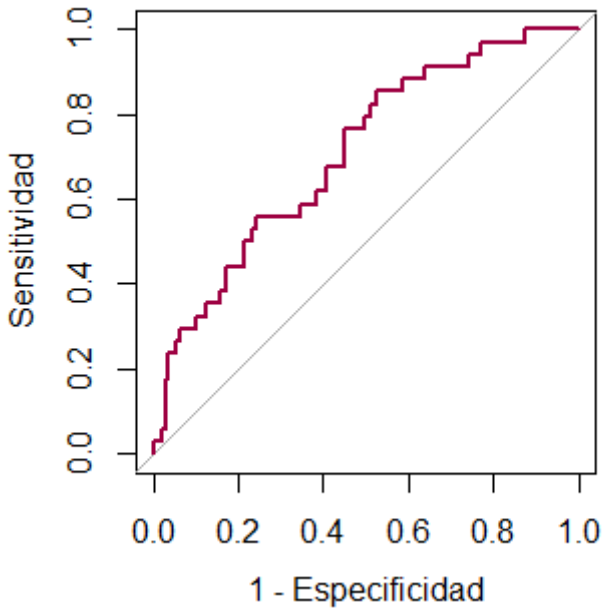

Figura 6.5: Curva ROC RF+Downsampling

Se ha obtenido un AUC (*area under curve*) de 0.7113. Se puede apreciar una mejora en comparación del AUC de 0.699 obtenido de la construcción Random Forests sobre el conjunto de datos original (sin balancear).

Veamos ahora el modelo Random Forests, con los datos obtenidos tras aplicar la técnica de **Upsampling**.

```
set.seed(1410)
ctrvalup=ctrlcv
ctrvalup$index=list(1:nrow(upSampled))
ctrvalup$indexFinal=1:nrow(upSampled)

rfUp = train(Class ~ ., data = upSampled_train_valid,
             "rf",
             trControl = ctrvalup,
             ntree = 100,
             tuneLength = 3, #tuneLength =3 pues tarda demasiado
             do.trace=TRUE,
             metric = "ROC")
```

```
rfUp
```

```
## Random Forest
##
## 4108 samples, 3296 used for final model
##   15 predictor
##    2 classes: 'Yes', 'No'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 3296
## Resampling results across tuning parameters:
##
##   mtry  ROC        Sens        Spec       Accuracy   Kappa
##    2    0.7205647  0.03694581  0.9901478  0.5135468  0.02709360
##    8    0.7041848  0.05418719  0.9926108  0.5233990  0.04679803
##   15    0.6722075  0.11330049  0.9876847  0.5504926  0.10098522
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
validResults$RFup = predict(rfUp, validation,
                            type = "prob")[,1]
testResults$RFup = predict(rfUp, testing,
                           type = "prob")[,1]
rfUpROC = roc(validResults$failed_insolvent, validResults$RFup,
              levels = rev(levels(validResults$failed_insolvent)))
```

```
rfUpROC
```

```
##
##Call:
##roc.default(response = validResults$failed_insolvent,
##predictor = validResults$RFup,
##levels = rev(levels(validResults$failed_insolvent)))
##
## Data: validResults$RFup in 406 controls
## (validResults$failed_insolvent No) < 25 cases
## (validResults$failed_insolvent Yes).
## Area under the curve: 0.7201
```

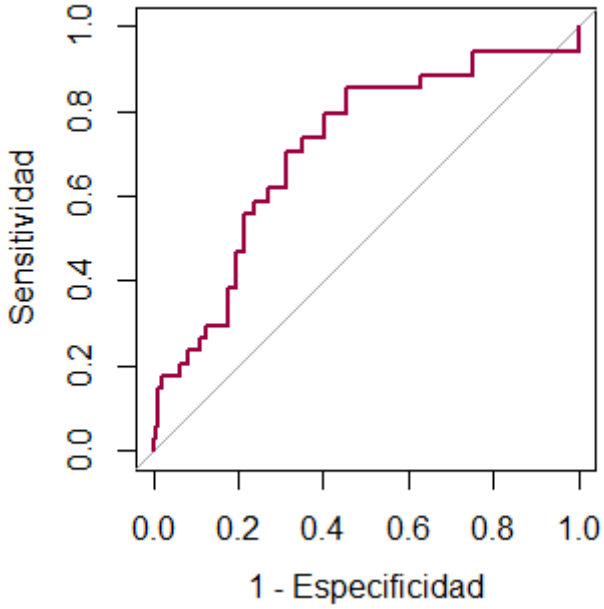

Figura 6.6: Curva ROC RF+Upsampling

En la figura 6.6 puede observarse una ilustración de la curva ROC tras aplicar el método Random Forests a el conjunto de datos obtenido mediante Upsampling.

Vamos a guardar también los datos para el conjunto test.

```
rfUpROCT = roc(testResults$failed_insolvent, testResults$RFup,
               levels = rev(levels(testResults$failed_insolvent)))
rfUpROCT



## Call:
##roc.default(response = testResults$failed_insolvent,
##predictor = testResults$RFup,
##levels = rev(levels(testResults$failed_insolvent)))
##
## Data: testResults$RFup in 686 controls
## (testResults$failed_insolvent No) < 34 cases
## (testResults$failed_insolvent Yes).
## Area under the curve: 0.7325
```

Puede observarse que el área bajo la curva ha mejorado, siendo ahora AUC = 0.7325.
El modelo óptimo Random Forests ha sido construido usando el valor para *mtry* igual a 2.

Ahora vamos a hacer el mismo procedimiento, para los datos balanceados mediante la técnica SMOTE.

```
set.seed(1410)
ctrvalsm=ctrlcv
ctrvalsm$index=list(1:nrow(smoted))
ctrvalsm$indexFinal=1:nrow(smoted)
rfSmote = train(failed_insolvent ~ ., data = smoted_train_valid,
                  "rf",
                  trControl = ctrvalsm,
                  ntree = 100,
                  tuneLength = 3,
                  do.trace=TRUE,
                  metric = "ROC")
```

```
rfSmote


## Random Forest
##
## 721 samples, 546 used for final model
##  15 predictor
##   2 classes: 'Yes', 'No'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 546
## Resampling results across tuning parameters:
##
##   mtry  ROC        Sens       Spec  Accuracy   Kappa
##    2    0.7622667  0.4666667  0.82  0.6685714  0.2975779
##    8    0.7546000  0.4666667  0.79  0.6514286  0.2650602
##   15    0.7554667  0.4933333  0.81  0.6742857  0.3132530
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

Guardamos las predicciones para el conjunto de Validación y test, y calculamos el AUC en el conjunto test.

```
validResults$RFsmote = predict(rfSmote, validation,
                               type = "prob")[,1]
testResults$RFsmote = predict(rfSmote, testing,
                              type = "prob")[,1]
```

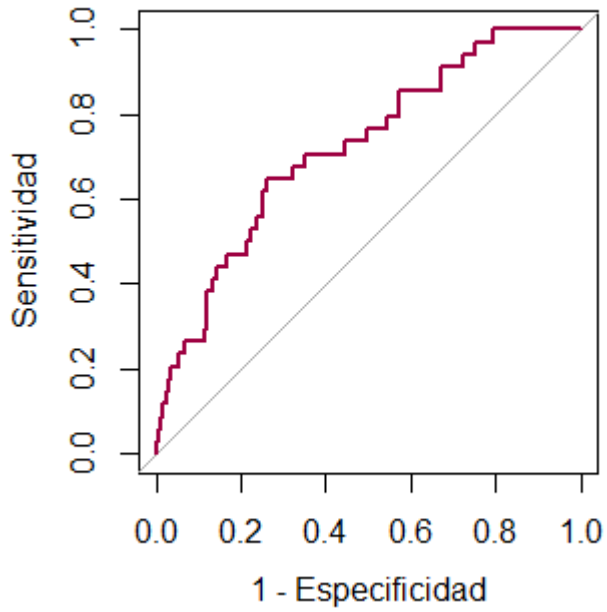

Figura 6.7: Curva ROC RF+Smote

```
rfSmoteROCT = roc(testResults$failed_insolvent, testResults$RFsmote,
                  levels = rev(levels(testResults$failed_insolvent)))

 rfSmoteROCT


##
##Call:
## roc.default(response = testResults$failed_insolvent,
## predictor = testResults$RFsmote,
## levels = rev(levels(testResults$failed_insolvent)))
##
## Data: testResults$RFsmote in 686 controls
## (testResults$failed_insolvent No) < 34 cases
## (testResults$failed_insolvent Yes).
## Area under the curve: 0.7266  #mejora
```

Veamos ahora el modelo random forests, con los datos obtenidos tras aplicar la técnica **ROSE**.

```
set.seed(1410)
ctrvalr=ctrlcv
ctrvalr$index=list(1:nrow(rose))
ctrvalr$indexFinal=1:nrow(rose)
rfrose = train(failed_insolvent ~ ., data = rose_train_valid,
                  "rf",
                  trControl = ctrvalr,
                  ntree = 100,
                  tuneLength = 3,
                  do.trace=TRUE,
                  metric = "ROC")
```

```
rfrose


## Random Forest
##
## 2157 samples, 1726 used for final model
##   15 predictor
##    2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 1726
## Resampling results across tuning parameters:
##
##   mtry  ROC        Sens       Spec       Accuracy   Kappa
##    2    0.8991273  0.8190045  0.8428571  0.8306265  0.6613131
##    8    0.8817173  0.8416290  0.8238095  0.8329466  0.6655962
##   15    0.8765999  0.8099548  0.8047619  0.8074246  0.6146438
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

Guardamos las predicciones para el conjunto de Validación y test, y calculamos el AUC en el conjunto test.

```
validResults$RFrose = predict(rfrose, validation,
                                type = "prob")[,1]
testResults$RFrose = predict(rfrose, testing,
                               type = "prob")[,1]
```

```
rfroseROCT = roc(testResults$failed_insolvent, testResults$RFrose,
                 levels = rev(levels(testResults$failed_insolvent)))
rfroseROCT


##
## Call:
## roc.default(response = testResults$failed_insolvent,
## predictor = testResults$RFrose,
## levels = rev(levels(testResults$failed_insolvent)))
##
## Data: testResults$RFrose in 686 controls
## (testResults$failed_insolvent No) > 34 cases
## (testResults$failed_insolvent Yes).
## Area under the curve: 0.7442
```

En la figura 6.8 puede observarse una ilustración de la curva ROC tras haber aplicado el método Random Forest a el conjunto de datos obtenido con la técnica ROSE.

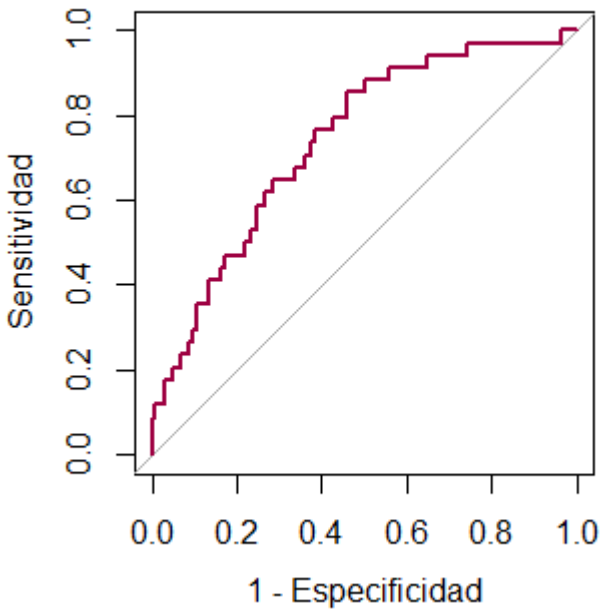

Figura 6.8: Curva ROC RF+ROSE

Se puede observar como todos los valores de AUC de los modelos Random Forests construidos sobre los conjuntos de datos balanceados, han mejorado, en comparación con el AUC del modelo construido sobre el conjunto de datos original.

Con la siguiente función vamos a construir una tabla con un resumen de los distintos modelos construidos. Los parámetros son:

- **x:** Modelo.
- **evl:** Conjunto de Validación.
- **tst:** Conjunto test.

Esta función determina el mejor umbral en la curva ROC en Validación según `best.method = ‘‘closest.topleft’’`, es decir, el corte más cercano a el ideal (0,1) en la curva ROC, que se corresponde con el más arriba a la izquierda.

La salida de dicha función son los valores AUC en los conjuntos de Validación y test (valROC , testROC) , y los valores sensitividad y especifidad para cada uno de estos modelos.

```
samplingSummary = function(x, evl, tst)
  {
    lvl = rev(levels(tst$failed_insolvent))
    evlROC = roc(evl$failed_insolvent,
                 predict(x, evl, type = "prob")[,1],
                 levels = lvl)
    tstROC= roc(tst$failed_insolvent,
                predict(x, tst, type = "prob")[,1],
                levels = lvl)
    rocs = c(auc(evlROC),auc(tstROC))
    cut = coords(evlROC, x = "best", ret="threshold",
                 best.method="closest.topleft")
    bestVals = coords(tstROC, cut, ret=c("sensitivity", "specificity"))
    out = c(rocs, bestVals*100)
    names(out) = c("valROC", "testROC", "testSens", "testSpec")
    out
  }

rfResults = rbind(samplingSummary(rfFit, validation, testing),
                  samplingSummary(rfDown, validation, testing),
                 samplingSummary(rfUp, validation, testing),
                  samplingSummary(rfSmote, validation, testing),
                  samplingSummary(rfrose,validation,testing))
rownames(rfResults) = c("Original", "Down--Sampling", "Up--Sampling",
"SMOTE","ROSE")
```

```
rfResults
```

```
##                    valROC   testROC testSens testSpec
## Original       0.7212808 0.6996227 64.70588 66.32653
## Down--Sampling 0.7410345 0.7112845 55.88235 70.99125
## Up--Sampling   0.7201478 0.7325073 58.82353 76.09329
## SMOTE          0.7090148 0.7266335 67.64706 65.88921
## ROSE           0.7640394 0.7441691 41.17647 86.44315
```

En la tabla anterior quedan recogidos los valores de AUC , especificidad y sensitividad resultantes de aplicar cada uno de los métodos descritos en el capitulo 5.

Para la elección del modelo óptimo, no existe un criterio que indique si es mejor regirse por una alta especificidad y sensitividad, o el área mayor bajo las distintas curvas ROC. En la figura 6.9 puede verse una representación de la curva ROC para las distintas técnicas usadas.

Si nos fijamos en la tabla, se puede observar que el mayor AUC se obtiene para el modelo resultante de aplicar la técnica de Random Forests a el conjunto de datos obtenido mediante el método ROSE, esto mismo ocurre si nos guiamos por la especificidad. Sin embargo, si se usa la medida sensitividad, el modelo elegido sería el obtenido tras balancear los datos mediante la técnica SMOTE.

```
rocCols = c("black", rgb(1, 0, 0, .5), rgb(0, 0, 1, .5),
"green","yellow")


#REPRESENTAR ALGUNAS CURVAS ROC TEST
plot(roc(testResults$failed_insolvent, testResults$RF,
   levels = rev(levels(testResults$failed_insolvent))),
   type = "S", col = rocCols[1], legacy.axes = TRUE)

plot(roc(testResults$failed_insolvent, testResults$RFdown,
   levels = rev(levels(testResults$failed_insolvent))),
   type = "S", col = rocCols[2],add = TRUE,legacy.axes = TRUE)

plot(roc(testResults$failed_insolvent, testResults$RFup,
   levels = rev(levels(testResults$failed_insolvent))),
   type = "S", col = rocCols[3],add = TRUE, legacy.axes = TRUE)

plot(roc(testResults$failed_insolvent, testResults$RFsmote,
   levels = rev(levels(testResults$failed_insolvent))),
   type = "S", col = rocCols[4], add = TRUE, legacy.axes = TRUE)

plot(roc(testResults$failed_insolvent, testResults$RFrose,
   levels = rev(levels(testResults$failed_insolvent))),
   type = "S", col = rocCols[5], add = TRUE, legacy.axes = TRUE)


legend(0.2,0.6,c("Original", "Down-Sampling ","Up-sampling",
"SMOTE","ROSE"),
       lty = rep(1, 5),
       lwd = rep(2, 5),
       cex = .8,
       col = rocCols)
```

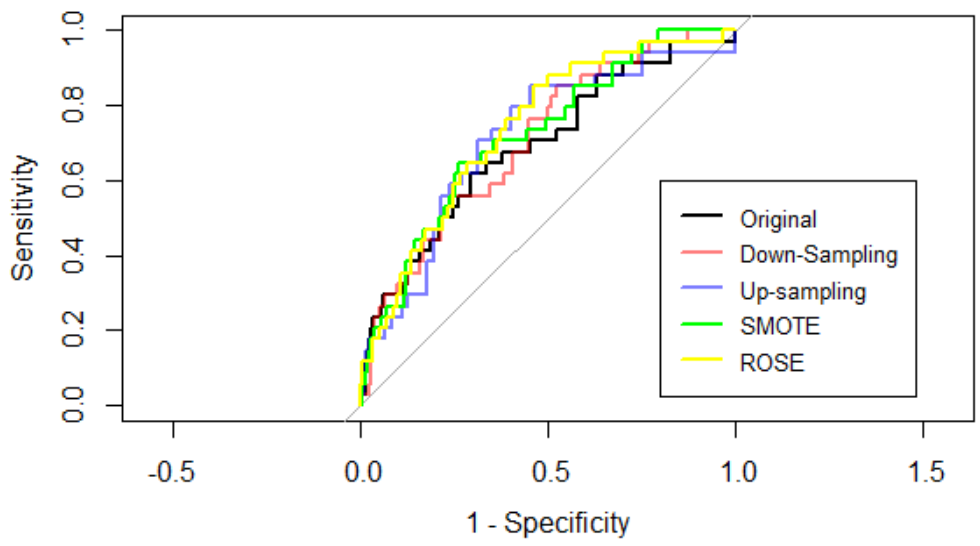

Figura 6.9: Curvas ROC

Bibliografía

[1] Andy Liaw and Matthew Wiener (2002) *Classification and Regression by randomForest.* https://CRAN.R-project.org/doc/Rnews/.

[2] Breiman, Friedman, Olshen and Stone (1984). *Classification and Regression Trees.* Chapman & Hall.

[3] Breiman,L (2001).*Random Forest. Machine Learning*

[4] Carlos Mera, José Miguel Arrieta (2015) Colombia. *Estudio Comparativo de técnicas de Balanceo en el Aprendizaje de múltiples instancias.*

[5] Chawla, Bowyer, Hall, Kegelmeyer. *SMOTE: Synthetic minority over-sampling technique.*Journal of Artificial Intelligence Research 16 (2002) 321–357

[6] Chawla , Bowyer , Hall, Kegelmeyer(2002).*SMOTE: Synthetic Minority Over-sampling Technique.*

[7] Cohen, Jacob (1960). *A coefficient of agreement for nominal scales.* Educational and Psychological Measurement 20 (1)

[8] Gauth James, Daniella Witten, Trevor Hastie, Robert Tibshirani (2013) New York. Springer. *An Introduction of Statistical Learning.*

[9] Giovanna Menardi, Nicola Torelli(2004). *Training and assessing classification rules with imbalanced data*

[10] I.Kononenko (2007) *Machine Learning and Data Mining.* Woodhead (Horwood Publishing)

[11] Ling, C. X., and Li, C (1998). *Data mining for direct marketing: Problems and solutions.*

[12] L. Torgo (2010) *Data Mining with R, learning with case studies.* Chapman and Hall/CRC http://www.dcc.fc.up.pt/ ltorgo/DataMiningWithR

[13] Max Kuhn, Kjell Johnson, *Applied Predictive Modeling.* 2013 (New York)

[14] Max Kuhn. Contributions from Jed Wing and Steve Weston and Andre Williams and Chris Keefer and Allan Engelhardt and Tony Cooper and Zachary Mayer and Brenton Kenkel and the R Core Team and Michael Benesty and Reynald Lescarbeau and Andrew Ziem and Luca Scrucca and Yuan Tang and Can Candan and Tyler Hunt.(2018)
caret: Classification and Regression Training.
https://CRAN.R-project.org/package=caret .

[15] Nicola Lunardon and Giovanna Menardi and Nicola Torelli (2014)
ROSE: a Package for Binary Imbalanced Learning.
Journal: {R} Journal(Vol.6)

[16] Quinlan,R. (1993).C4.5 .*Programs for Machine Learning.* Morgam Kaufman, San Mateo.

[17] Quinlan, R.(2004). C5.0. www.rulequest.com

[18] R Core Team (2018). *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/.

[19] Shannon , Weaber (1949) EEUU. *The mathematical theory of communication*

[20] Terry Therneau and Beth Atkinson (2018). *rpart: Recursive Partitioning and Regression Trees.* https://CRAN.R-project.org/package=rpart

[21] Trevor Hastie, Robert Tibshirani, Jerome Friedman (2008) *The elements of statistical learning. Data mining, inference, and predictions.*

[22] Xavier Robin and Natacha Turck and Alexandre Hainard and Natalia Tiberti and Frédérique Lisacek and Jean-Charles Sanchez and Markus Müller (2011)
pROC: an open-source package for R and S+ to analyze and compare ROC curves. Journal: BMC Bioinformatics (Vol.12).