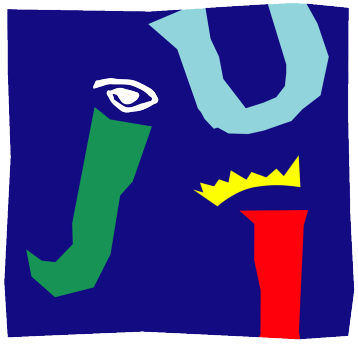

UNIVERSITAT JAUME·I

Máster en Matemática Computacional

Proyecto Final de Máster

Bosques aleatorios supervisados y no supervisados. Aplicación al análisis de desigualdades económicas en el mundo

Autor:
Joan Boters Pitarch

Tutora académica:
Irene Epifanio López

Resumen

El objetivo principal del trabajo consiste en estudiar con detalle el método de aprendizaje estadístico conocido como: bosques aleatorios. Serán considerados tanto como método supervisado como no supervisado. En primer lugar, desarrollaremos y explicaremos la parte teórica del método, comenzando por los árboles de decisión. En segundo lugar, presentamos la implementación en el software `R` de los bosques aleatorios, lo haremos mediante el estudio la librería *randomForest*. Por último, aplicaremos los bosques aleatorios a una base de datos, creada exclusivamente para la realización de dicho trabajo, donde abordaremos un problema de carácter social como son: las desigualdades económicas en los diferentes países del mundo.

Palabras clave

Bosques aleatorios, Árboles de decisión, Aprendizaje estadístico supervisado, Aprendizaje estadístico no supervisado, Desigualdades.

Keywords

Random forests, Decision Trees, Supervised statistical learning, Unsupervised statistical learning, Inequalities.

Índice general

Introducción

El aprendizaje estadístico hace referencia a un amplio conjunto de herramientas, cuya función es comprender las relaciones entre los datos. En otras palabras, el aprendizaje estadístico tiene como finalidad desarrollar un modelo tan preciso como sea posible, y que pueda ser usado para predecir o estimar alguna característica de los datos, que pueda aportar conocimiento y valor sobre los mismos.

Respecto a los métodos de aprendizaje, estos pueden ser clasificados como: supervisados y no supervisados. En el aprendizaje supervisado se construye un modelo estadístico, que intenta predecir o estimar un valor de salida Y (output) basado en valores de entrada (input). Es por eso que, en muchas ocasiones, los problemas supervisados se reducen a la estimación de una función $\hat{f}$ que aporta el posible valor de Y en función de los valores de entrada. A su vez, los métodos con los que pretendemos estimar la función $\hat{f}$ se puede clasificar como: paramétricos y no paramétricos.

Los métodos paramétricos son en los que la obtención de $\hat{f}$ se reduce al cálculo de unos coeficientes. Además, en algunos casos, se delimita la *forma* de la función $\hat{f}$; un ejemplo podría ser el caso de la regresión lineal. En cambio, los métodos no paramétricos ni asumen una forma determinada para $\hat{f}$, ni se reducen al cálculo de una cantidad de coeficientes, simplemente exploran los datos con la finalidad de acercarse a ellos tanto como sea posible; un claro ejemplo pueden ser los árboles de decisión.

Por otra parte, en el aprendizaje no supervisado solo tenemos valores de entrada (inputs), es decir, no hay un valor de salida supervisado que haga de guía al modelo. En ese caso, nuestro objetivo será comprender las relaciones y estructuras que proporcionan los datos.

Objetivos y organización

El objetivo principal del trabajo es llevar a cabo un estudio detallado de los fundamentos teóricos y conceptuales que hay detrás de los métodos de aprendizaje conocidos como **bosques aleatorios.** Para ello, el trabajo deberá abordar el estudio de los **árboles de decisión** y la técnica de remuestreo conocida como **bootstrap**, ya que ambos conceptos son el soporte conceptual que hay detrás de los bosques aleatorios.

Como objetivo secundario del trabajo, intentaremos aplicar esta técnica de aprendizaje a una base de datos creada *ad hoc*, en la que se recogen diferentes indicadores socio-económicos de los diferentes países del mundo, con la finalidad de sacar algunas

conclusiones acerca de las desigualdades económicas existentes. El estudio de este tipo de datos puede ser relevante para proponer acciones que puedan ayudar a determinados países. Esto estaría vinculado con diversos objetivos de desarrollo sostenible en el marco de la Agenda 2030 de las Naciones Unidas, principalmente en el objetivo 10 *«Reducción de las desigualdades»*.

Respecto a la organización del trabajo,se puede diferenciar entre: parte teórica y aplicación. Hay que señalar que en la memoria, a nivel teórico y práctico, introduciremos tanto los bosques aleatorios supervisados, con dos tipos de respuesta (numérica y categórica) y también los bosques aleatorios no supervisados, que no son tan conocidos, y abordaremos diversas problemáticas en la aplicación, como son, por ejemplo, los datos faltantes.

En resumen, en el primer capítulo describiremos con detalle los árboles de decisión y posteriormente en el segundo capítulo abordaremos las técnicas de remuestreo y los bosques aleatorios. Finalmente, en la parte de aplicación dedicaremos el tercer capítulo para explicar la librería de `R` que usaremos en el análisis de los datos, en cuya construcción se implementan los bosques aleatorios y sus funcionalidades, para terminar con la descripción y el análisis de nuestra base de datos mediante el uso de los bosques aleatorios.

Capítulo 1

Árboles de decisión

Los métodos basados en árboles de decisión, se fundamentan en la realización de particiones rectangulares del dominio dimensional donde está comprendida nuestra muestra. En primer lugar, vamos a describir el método más conocido y usual como son los *Clasification and Regression Trees* (*CART*) introducido por [6].

1.1. Árboles de regresión y clasificación

Consideremos en primer lugar, un problema de regresión, donde Y es nuestra variable continua de respuesta. Sean X_1 y X_2 nuestros campos de entrada, donde $X_i \subset [0,1], \quad i \in \{1,2\}$, por tanto tenemos que $Y\colon X_1 \times X_2 \longmapsto \mathbb{R}$, luego $Y = Y(x_1, x_2)$.

Si queremos realizar un estudio mediante árboles de decisión, debemos hacer particiones de nuestro dominio de la variable respuesta. En este caso, particiones sobre $[0,1] \times [0,1]$. Nos restringiremos a **particiones binarias recursivas**, es decir, primero dividiremos el dominio en dos, para ello habrá que elegir una variable y un punto de separación (más conocido como *split-point*). A continuación, repetiremos el proceso considerando cada parte como dos partes independientes.

Veamos dos ejemplos en la Figura 1.1, de cómo realizar las particiones.

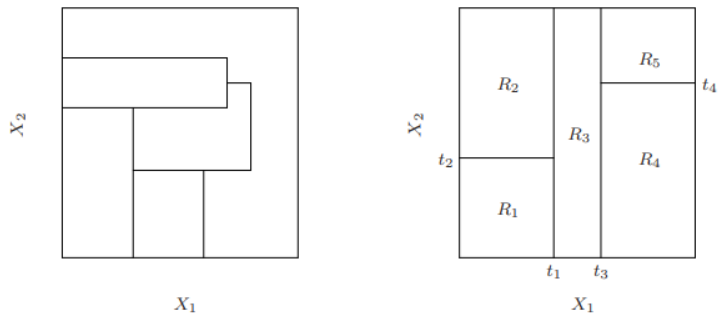

Figura 1.1: No realizada mediante particiones recursivas (izquierda). Realizada mediante particiones recursivas (derecha)

Como se puede observar, la figura de la izquierda no se ha realizado mediante particiones binarias recursivas, ya que no todos los polígonos que aparecen son rectángulos. En cambio, la figura de la derecha, sí se ha obtenido mediante dicho proceso.

Nota 1.1. Como hemos visto en la Figura 1.1, podemos realizar particiones del dominio que quedan fuera de nuestro estudio, debido a que nos estamos restringiendo a un estilo concreto. La motivación de dicha restricción se debe a la dificultad de describir las diferentes regiones cuando no son rectangulares. En cambio, esto no sucede cuando las regiones son siempre rectangulares, luego es por un motivo de **simplificación**.

Ejemplo 1.2. Es muy habitual resumir el proceso de particiones binarias recursivas realizadas en la Figura 1.1 con un diagrama en árbol, como el que aparece en la Figura 1.2:

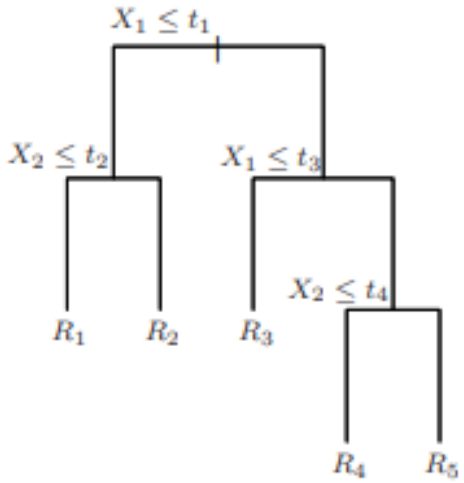

Figura 1.2: Ejemplo de diagrama de árbol.

Así, podemos observar que en este caso nuestro dominio de la variable respuesta Y, se puede describir como unión de 5 regiones R_i (véase Figura 1.1):

$$[0,1] \times [0,1] = \cup_{i=1}^{5} R_i \qquad R_i \cap R_j = \emptyset \quad i \neq j$$

Para terminar con este caso particular, una vez se han determinado las regiones del dominio, se procede a dar un valor constante de la variable respuesta Y en cada región, es decir:

$$Y(x_1, x_2) = \sum_{k=1}^{5} c_k \mathcal{I}\{(x_1, x_2) \in R_k\} \qquad \text{donde } c_k \in \mathbb{R}$$

Más tarde, veremos como se determinan las constantes c_k, ya que para hacerlo se necesita una muestra de nuestra variable Y, junto con sus entradas X_1, X_2 que la originan.

De forma general, podemos considerar que nuestra muestra consiste en p entradas, y una respuesta para cada N observaciones, es decir $X = (x_i, y_i) \quad i = 1, 2, \ldots, N$ donde

$x_i = (x_{i1}, x_{i2}, \ldots, x_{ip}) \in \mathbb{R}^p$. El algoritmo del método necesita decidir automáticamente las variables de separación, el *split-point*, además de la forma del árbol.

Supongamos primero que ya hemos dividido $\mathbb{R}^p$ en M regiones que denominamos $R_1, \ldots, R_M$ y denotamos por $f : \mathbb{R}^p \longmapsto \mathbb{R}$ la estimación del valor de nuestra variable respuesta, dada por el algoritmo. Entonces como ya sabemos, esta función será de la siguiente forma:

$$f(x) = \sum_{m=1}^{M} c_m \mathcal{I}\{x \in R_m\}. \tag{1.1}$$

Asumiendo como criterio de minimización del error, el sumatorio de los residuos al cuadrado, es decir,

$$\epsilon(X) = \sum (y_i - f(x_i))^2. \tag{1.2}$$

Entonces, veamos que el mínimo se alcanza para dicha métrica en cada región R_m, cuando se satisface

$$2\sum (y_i - c_m) = 0 \Longleftrightarrow c_m = ave(y_i \colon x_i \in R_m). \tag{1.3}$$

Ahora bien, determinar la mejor partición para dicho criterio de minimización del error, es computacionalmente inviable. Por tanto, para encontrar la variable de separación y su *split-point*, procederemos mediante el siguiente algoritmo.
Sea la posición de la variable de separación j y sea s el *split-point* de dicha variable, podemos definir la pareja de semiplanos generada mediante esta división como:

$$R_1(j,s) = \{X | \, X_j \leq s\} \text{ y } R_2(j,s) = \{X | \, X_j > s\}. \tag{1.4}$$

Luego, buscamos la posición de la variable y su *split-point* que satisface

$$\min_{j,s} \Big[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \Big]. \tag{1.5}$$

Para cuales sea las elecciones del par (j,s) la minimización del interior viene determinada por

$$\hat{c_1} = ave(y_i \colon x_i \in R_1(j,s)) \text{ y } \hat{c_2} = ave(y_i \colon x_i \in R_2(j,s)) \tag{1.6}$$

Además, si consideramos como fija la posición j, computacionalmente podemos determinar rápidamente cuál es *split-point* **óptimo**. Por tanto, si realizamos esto para cada posición j y posteriormente comparamos las p, parejas obtenidas, el objetivo de encontrar la mejor pareja (j,s), se convierte en **viable**.

Nota 1.3. El argumento anterior, no tiene en cuenta la posibilidad que la elección de la variable en posición j', obtenga en iteraciones posteriores errores más pequeños que la variable en posición j, que ha sido elegida. En otras palabras, el árbol que hemos obtenido escoge **a cada paso** la variable que provoca mayor disminución en el error, sabiendo que el árbol resultante puede que no sea el más preciso posible. Es lo que se denomina un algoritmo **voraz**.

Si nos centramos en segundo lugar, en los problemas de clasificación, entonces nuestra variable Y de respuesta es discreta, luego $Y\colon \mathbb{R}^p \longmapsto \{1, 2, \ldots, K\}$. Para estos problemas, se suele usar como métrica la proporción de observaciones de la clase k para cada nodo terminal m, es decir,

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} \mathcal{I}(y_i = k), \quad \text{donde } N_m = \#\{x_i \in R_m\}. \tag{1.7}$$

Para cada región R_m, clasificaremos sus observaciones por la moda en dicho nodo terminal, por tanto,

$$k(m) = \arg\max_k \hat{p}_{mk}. \tag{1.8}$$

Respecto a las medidas del error, en los problemas de clasificación son habituales:

1. **Proporción de mal clasificados:** $1 - \hat{p}_{mk(m)}$
2. **Índice de Gini:** $\sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$
3. **Entropía :** $-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$

La ventaja de las últimas respecto a la primera, es la suavidad y diferenciabilidad como funciones de probabilidad p, y dado que hay que usar algoritmos de optimización numérica, hacen que estas dos métricas sean más adecuadas para los métodos de clasificación. Además ambas medidas son más sensibles, que la **proporción de mal clasificados**, para cambios en el número de observaciones en los nodos.

Ejemplo 1.4. Supongamos que tenemos una muestra con 2 clases, donde hay $(400, 400)$ para cada una. Supongamos que hacemos la siguiente división, $(300, 100)$ y $(100, 300)$, entonces, nuestra proporción de mal clasificados es del 25 %, la misma que si consideramos la división dada por $(200, 400)$ y $(200, 0)$. En cambio, es preferible elegir la segunda división ya que origina un **nodo puro** y, por tanto, ya no se divide más.
Por otra parte, si consideramos el **índice de Gini** o la **entropía**, para ambas medidas obtenemos mejores resultados para la segunda división.

1.2. Tamaño del árbol

Claramente, un árbol de tamaño muy grande puede originar fenómenos como el *overfitting*. Por otra parte, un árbol pequeño, **no captura** las tendencias más importantes en los datos. Es por eso que, hay que buscar un equilibrio entre el tamaño y el error.

Hay 2 posibles enfoques:

1. Ir dividiendo en regiones siempre que se disminuya el error por debajo de cierto umbral que consideremos *aceptable*. Generalmente, origina árboles más pequeños, pero no tiene en cuenta que una división *no aceptable* puede ir acompañada posteriormente de una *muy buena división*.

2. Podemos crear un árbol T_0 *grande*, el cual siga un algún criterio de parada. A continuación, podemos ir *podando* dicho árbol, con la finalidad de obtener un subárbol que nos dé un mejor equilibrio entre tamaño del árbol y precisión.

Desarrollemos el segundo enfoque para los problemas de regresión. Para ello, sea T_0 nuestro árbol original (*grande*) y definimos $T \subset T_0$, un subárbol de T_0, que puede ser obtenido mediante podas del original. Sea $|T|$ el número de nodos terminales y definimos

$$Q_m(T) := \frac{1}{N_m} \sum_{x_i \in R_m} (y_i - c_m)^2. \tag{1.9}$$

Puesto que evaluar todos los posibles subárboles es muy costoso computacionalmente, aplicaremos métodos que permitan seleccionar un pequeño subconjunto de subárboles **prometedores** para su posterior análisis y selección. Así pues, definamos la siguiente métrica

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) \; + \; \alpha \, |T| \qquad \alpha \geq 0 \tag{1.10}$$

que llamaremos **coste de complejidad**.

Nota 1.5. Veamos que si $\alpha = 0$, no se tiene en cuenta el tamaño del árbol y entonces se busca minimizar el error al máximo. Lo que puede provocar sobreajuste en nuestra muestra de entrenamiento.
Por otra parte, si $\alpha \longmapsto \infty$, entonces obtendremos un árbol que no capta tendencias al devolver incluso, en algunos casos, una salida constante.

Como para cada $\alpha \geq 0$ podemos encontrar un T_α, tal que $C_\alpha(T_\alpha) = \text{mín}_{T \subset T_0} C_\alpha(T)$, nuestro objetivo será encontrar una sucesión creciente de valores de α, que genere a su vez un sucesión decreciente de árboles, ya que si $\alpha_1 < \alpha_2 \Rightarrow T_{\alpha_1} \supseteq T_{\alpha_2}$. Luego queremos determinar,

$$0 = \alpha_0 < \alpha_1 < \alpha_2 \ldots \Longrightarrow T_0 \supseteq T_1 \supseteq T_2 \ldots ,$$

donde los $\{T_i\}_{i \geq 0}$ son subárboles obtenidos mediante la poda del árbol original. Finalmente, una vez tenemos nuestra sucesión de candidatos a árboles *prometedores*, se lleva a cabo un análisis de cada uno de ellos, con la finalidad de poder compararlos y determinar cuál aporta mejores predicciones.

Nota 1.6. Es muy habitual realizar el análisis anterior mediante **validación cruzada**.

Capítulo 2

Bosques aleatorios

En este capítulo, vamos a desarrollar el método estadístico principal que usaremos en este trabajo, como son los *bosques aleatorios*, también conocidos como *Random Forest*. Para poder comprender con totalidad esta técnica de aprendizaje estadístico, es necesario comprender el funcionamiento de la técnica de remuestreo llamada *bootstrap* y analizar el efecto que produce cuando se aplica a los árboles de decisión.

2.1. Bootstrap

El bootstrap es una herramienta estadística muy potente y usada frecuentemente, que tiene como objetivo cuantificar la incertidumbre de cierto estimador involucrado en un método de aprendizaje estadístico. Sin embargo, la fuerza de esta técnica de remuestreo recae en que puede ser fácilmente aplicable en un rango amplio de métodos estadísticos, incluidos en los que la medición de la varianza es difícil de realizar.

En multitud de ocasiones es prácticamente imposible disponer de un tamaño lo suficientemente grande de observaciones, dado que no podemos acudir a la población real para obtener la cantidad de observaciones necesarias con las que podamos llevar a cabo estimaciones precisas. En este momento, es cuando se utilizan las técnicas de remuestreo, por ejemplo el bootstrap.

Definición 2.1. Sea $\mathcal{Z}$ nuestra muestra de datos original, de tamaño n. Diremos que Z_B es una **muestra bootstrap** de $\mathcal{Z}$ si se ha obtenido mediante extracciones con reemplazamiento sobre las observaciones de $\mathcal{Z}$ y tiene tamaño n.

Ejemplo 2.2. Sean X e Y dos variables aleatorias, las cuales hacen referencia al retorno de inversión que obtendremos por haber invertido una suma de dinero en dichas operaciones. Sea $\alpha \in [0,1]$ la proporción de dinero que invertiremos en X y el resto en Y. Entonces queremos obtener,

$$\hat{\alpha} = \min_{\alpha} Var\big(\alpha X + (1-\alpha)Y\big).$$

Sea $g(\alpha) := Var\big(\alpha X + (1-\alpha)Y\big)$, puesto que es conocido que si X e Y son dos

variables aleatorias se tiene que,

$$\begin{aligned} Var\big(X + Y\big) &= Var(X) + Var(Y) + 2\,Cov(X,Y) \\ Var\big(c\,X\big) &= c^2\,Var\big(X\big) \quad c \in \mathbb{R}. \end{aligned} \tag{2.1}$$

Entonces, aplicando la propiedad anterior y asumiendo que $Var(X) = \sigma_X^2, Var(Y) = \sigma_Y^2$ y $Cov(X,Y) = \sigma_{XY}$, obtenemos que

$$g(\alpha) = \alpha^2\,\sigma_X^2 + (1-\alpha)^2\sigma_Y^2 + 2\,\alpha\,(1-\alpha)\sigma_{XY}. \tag{2.2}$$

Si derivamos en 2.2, es fácil ver que tenemos

$$g'(\alpha) = 2\alpha\sigma_X^2 - 2(1-\alpha)\sigma_Y^2 + 2\,(1-2\alpha)\sigma_{XY}. \tag{2.3}$$

Para determinar $\hat{\alpha}$, hacemos $g'(\alpha) = 0$ y si manipulamos la expresión 2.3 se tiene que,

$$\hat{\alpha} = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}. \tag{2.4}$$

Ahora bien, no conocemos las distribuciones de nuestras variables X e Y, por tanto desconocemos su varianza, luego no podemos calcular $\hat{\alpha}$ teórico, pero sí podemos hacer una estimación y **cuantificar** el posible error cometido.

Supongamos que disponemos de una base de datos $\mathcal{Z}$ con resultados de operaciones pasadas de X e Y, donde la base de datos tiene tamaño n (suele ser pequeño). De nuestro conjunto de datos original, extraemos $Z^k, \quad k = 1, \ldots, B$ muestras bootstrap. A continuación, para cada una de ellas, calculamos el valor muestral de nuestro parámetro, denotados por $\hat{\alpha_k}, \quad k = 1, \ldots, B$. Así, una estimación del error estándar será:

$$SE_B(\hat{\alpha}) = \sqrt{\frac{1}{B-1}\sum_{r=1}^{B}(\hat{\alpha_r} - \frac{1}{B}\sum_{k=1}^{B}\hat{\alpha_k})^2}.$$

La estimación de $\hat{\alpha}$ que se escoge, es la obtenido sobre $\mathcal{Z}$. Las muestras bootstrap son utilizadas para aproximar el error estándar cometido en la estimación del parámetro.

Propiedad 2.3. *Sea $X = \{x_i\}_{i=1}^n$ una muestra aleatoria con n observaciones. La probabilidad de que la observación x_j con $1 \leq j \leq n$ no esté contenida en una muestra $Y = \{y_i\}_{i=1}^n$ bootstrap sobre X es*

$$P\big(\{x_j \notin Y\}\big) = \left(1 - \frac{1}{n}\right)^n.$$

Demostración. Puesto que la muestra bootstrap, se origina mediante extracciones con reemplazamiento sobre X, cada una de estas extracciones son independientes entre si. Así, como

$$P\big(\{x_j \neq y_1\}\big) = 1 - \frac{1}{n}.$$

Por la independencia de cada extracción, se llega a

$$P\big(\{x_j \notin Y\}\big) = \left(1 - \frac{1}{n}\right)^n.$$

□

Propiedad 2.4. *En una muestra bootstrap se suele dejar fuera aproximadamente un tercio de las observaciones.*

Demostración. Por la propiedad anterior es conocido que

$$P\big(\{x_j \notin Y\}\big) = (1 - \frac{1}{n})^n \longmapsto_{n\to\infty} \frac{1}{e} \approx \frac{1}{3}.$$

Consideremos

$$Z = \#\{ \text{ observaciones de X que no están contenidas en la muestra bootstrap } \}.$$

Es claro que Z se comporta como una $Bi(n, \frac{1}{3})$, donde n es el tamaño de X Luego, es fácil ver que

$$E[Z] = \frac{n}{3}.$$

□

2.2. Bosques Aleatorios

Como hemos dicho anteriormente, el bootstrap es una técnica de remuestreo muy extendida, que se utiliza en muchos métodos de aprendizaje estadístico, un ejemplo son los bosques aleatorios.

Los árboles de decisión explicados en el Capítulo 1, tienen la desventaja, como método estadístico, de padecer una **alta varianza**. Esto se puede ver reflejado en que si aplicamos los *CART* para cada mitad de nuestra muestra, podemos obtener predicciones totalmente dispares, es decir, hay una alta varianza intrínseca en el método. Por otra parte, tienen la ventaja de que son métodos con poco sesgo, es decir, si hacemos crecer suficiente el árbol, y obtenemos muchos nodos terminales, entonces la diferencia entre nuestra estimación y su valor real tenderá a ser pequeña.

En cambio, un método estadístico con baja varianza podría ser una regresión lineal, ya que si lo aplicamos a cada mitad de la muestra, podemos obtener aproximaciones similares. Aunque la regresión lineal suele generar sesgos elevados en sus predicciones, puesto que la calidad de las aproximaciones en los extremos de la recta puede llegar a ser bastante mala.

2.2.1. Bagging

La técnica llamada *bootstrap aggregation* o *bagging*, tiene como propósito general la reducción de la varianza de nuestro método de aprendizaje estadístico. Recordemos que si tenemos $Z_1, Z_2, \ldots, Z_n$ i.i.d con σ^2 de varianza, entonces

$$Var(\overline{Z}) = \frac{\sigma}{n}.$$

En otras palabras, si hacemos la media sobre un conjunto de predicciones habremos reducido la varianza de nuestra predicción.

Sea $\mathcal{X}$ nuestra muestra original, obtenemos $X_1, \ldots, X_m$ muestras bootstrap sobre $\mathcal{X}$. Aplicamos nuestro método de aprendizaje estadístico a cada una de nuestras muestras, siendo $\hat{f}_1(x), \hat{f}_2(x), \ldots, \hat{f}_m(x)$ nuestras aproximaciones. Así, podemos definir como la predicción general el promedio de todas las estimaciones anteriores, es decir,

$$\hat{f}_{bag}(x) = \frac{1}{m} \sum_{k=1}^{m} \hat{f}_k(x). \tag{2.5}$$

El razonamiento anterior, puede ser aplicado a los árboles de decisión, disminuyendo así su alta varianza, y por tanto mejorando su calidad en las predicciones en muchas ocasiones.

2.2.2. Fundamentos de los bosques aleatorios

Los bosques aleatorios proporcionan una mejora sobre el *bagging* aplicado a los árboles de decisión. Esta mejora se produce mediante algunos retoques que intentan reducir la correlación entre cada uno de los árboles involucrados en el método. Aunque se podría pensar que la independencia en las muestras bootstrap implicaría independencia en nuestros árboles, esto no es así. Supongamos que hay un predictor en nuestra muestra, que es realmente influyente en nuestra variable respuesta (en ocasiones se les llaman predictores *fuertes*), entonces aunque haya remuestreo sobre las observaciones, habría que esperar que en la mayoría de árboles este predictor se encuentre en la separación inicial y, por tanto, dando lugar a particiones similares, que conllevan una correlación entre los valores de salida que proporciona el método. Luego, los árboles de decisión no son independientes entre ellos.

Sean $Z_1, ..., Z_n$ variables aleatorias idénticamente distribuidas y $\rho = Cor(Z_i, Z_j)$, donde $i \neq j$ con $1 \leq i, j \leq n$. Entonces

$$\begin{aligned} Var(\overline{Z}) &= \frac{1}{n^2} \sum_{1 \leq i,j \leq n} Cov(Z_i, Z_j) \\ &= \frac{1}{n^2} \left(n\,\sigma^2 + \sum_{i \neq j} Cov(Z_i, Z_j) \right) \\ &= \frac{\sigma^2}{n} + \frac{2}{n^2} \left(\sum_{i=1}^{n} n - i \right) \rho\,\sigma^2 \\ &= (1 - \rho)\,\frac{\sigma^2}{n} + \rho\sigma^2. \end{aligned} \tag{2.6}$$

Notemos que cuando $n \longmapsto \infty$, el primer factor desaparece. Luego, **la idea fundamental** de los bosques aleatorios es reducir la varianza mediante el promedio de muchos árboles, y realizar algunos retoques para que la correlación entre árboles se reduzca. Este último aspecto, se consigue mediante la **elección aleatoria** de las variables de entrada **disponibles** en cada partición del árbol.

Así, cuando hacemos crecer un árbol sobre una muestra bootstrap, antes de cada separación seleccionamos aleatoriamente m predictores, los cuales son los candidatos para dicha separación.

Nota 2.5. Hay que observar que de esta manera si hay un predictor de los denominados *fuertes*, este no va influir siempre, en las separaciones iniciales. Solo lo hará cuando haya sido seleccionado de manera aleatoria.

En [4] se aconseja que para problemas regresión se tome $m \approx \frac{p}{3}$ y para los de clasificación $m \approx \sqrt{p}$, donde p es el número de predictores de nuestra muestra inicial. En cambio, en [1, pág. 592] recomiendan que el parámetro m debería ser adaptable al problema, ya que los valores anteriores no siempre son los mejores. En este trabajo m será un parámetro adaptable.

Definición 2.6. Sea $\mathcal{B} = \{T_b(x)\}_{b=1}^{B}$ un bosque aleatorio sobre $\mathcal{X}$, nuestra muestra original. La predicción $\hat{f}_{RF}$ de los bosques aleatorios es obtenida por

1. Para los casos de **regresión**:

$$\hat{f}_{RF}(x) = \frac{1}{B}\sum_{k=1}^{B} T_k(x) \tag{2.7}$$

2. Para los casos de **clasificación**, cada árbol *emite su voto* mediante $T_b(x)$ para cada $b = 1, ..., B$ y la salida $\hat{f}_{RF}(x)$ es la clase más votada.

2.2.3. *Out of bag* error

Es conocido por la Propiedad 2.4 que aproximadamente un tercio de las observaciones **no** están contenidas en cada muestra bootstrap.

Definición 2.7. A cada observación de la muestra original X que no esté contenida en una muestra bootstrap sobre X, la denominaremos observaciones *out of bag*. También las denotaremos como observaciones OOB.

Podremos usar este tipo de observaciones para realizar una estimación del error. Como haremos uso de las observaciones *out of bag*, denominaremos a esta métrica como *error OOB*. Consideremos x_i la observación i-ésima, entonces es claro por la Propiedad 2.4 que existe un conjunto de índices $J_i \subset \{1, 2, ..., B\}$ donde dicha observación es OOB y $|J_i| \approx \frac{B}{3} \quad \forall i$. Entonces, podemos definir como la aproximación OOB de x_i

$$\hat{f}_{OOB}(x_i) := \frac{1}{|J_i|}\sum_{k\in J_i} \hat{f}_k(x_i). \tag{2.8}$$

Luego el error *out of bag* producido en nuestro bosque aleatorio $\mathcal{B}$ es:

$$Er_{OOB}\Big(\mathcal{B}\Big) = \sum_{i=1}^{n}\Big(y_i - \hat{f}_{OOB}(x_i)\Big)^2. \tag{2.9}$$

Nota 2.8. En el ejercicio 15.2 de [1] se propone demostrar que el error OOB coincide **asintóticamente** con la validación cruzada. La ventaja del error OOB frente a la validación cruzada, es que es una métrica que se va calculando mientras se ejecuta el algoritmo. En cambio, la validación cruzada se realiza una vez se ha determinado el modelo, con el coste computacional que conlleva.

2.2.4. Importancia de las variables de entrada

Una de las características fundamentales de los bosques aleatorios, es que podemos medir y cuantificar la influencia de las variables de entrada. Hay diferentes maneras de realizar el cálculo pero en este trabajo solo consideraremos las dos siguientes:

1. La importancia de la variable j-ésima, se mide como la disminución total del error por la separación con dicha variable, promediado entre todos los árboles donde aparece. Para problemas de clasificación se usa el *índice de Gini*, para regresión *la suma de los residuos al cuadrado.*

2. Consideremos las observaciones OOB propias de cada árbol $b \in \{1, 2, ..., B\}$ y calculamos su precisión, para regresión usamos (2.9), para clasificación la proporción de malclasificados. A continuación, permutamos aleatoriamente los valores en la variable j-ésima de dichas observaciones, y acto seguido, volvemos a calcular su precisión.

   La diferencia entre las dos se promedia entre todos los árboles y se normaliza entre la desviación estándar de las diferencias.

Nota 2.9. Ambas formas dan lugar a *rankings* de importancia similares para las variables, aunque **no son equivalentes**. La segunda forma suele dar lugar a ordenaciones más uniformes en los valores del ranking, ya que se puede calcular para todo árbol del bosque, en cambio la primera manera solo permite calcular en los árboles donde dicha variable interviene en algunas de las separaciones. En [1, pág. 594], se puede observar un ejemplo de ambos *rankings*.

2.2.5. Proximidades

En muchas ocasiones es interesante (aunque estemos en un problema de regresión) clasificar nuestras observaciones en tanto que son *parecidas* entre ellas. Ahora bien, cuando tenemos una cantidad de predictores numerosa, no hay un criterio claro para decidir que dos observaciones *se parecen.*

Pues bien, dado que los árboles de decisión se fundamentan en la realización de particiones, la construcción de los nodos terminales nos proporcionan una herramienta fundamental para llevar a cabo estudios locales de nuestras observaciones.

Definición 2.10. Sea $\mathcal{T}$ un árbol de decisión sobre la muestra $\mathcal{X}$. Diremos que $x_i, x_j \in X$ son próximos en $\mathcal{T}$ si pertenecen a la misma región del árbol. Lo denotaremos como

$$x_i \, \mathcal{R} \, x_j$$

Definición 2.11. Sea $\mathcal{X}$ una muestra de tamaño n, y sea $\mathcal{B} = \{T_b(x)\}_{b=1}^{B}$ un bosque aleatorio. Definimos la función de **proximidad** entre dos observaciones en b-árbol como $R^b : \{1, 2, ..., n\}^2 \longmapsto \{0, 1\}$ tal que:

$$R^b(i,j) := \begin{cases} 1 \quad si \qquad x_i \mathcal{R} x_j \\ \\ 0 \qquad \text{en otro caso} \end{cases}$$

Definición 2.12. Sea $\mathcal{B} = \{T_b(x)\}_{b=1}^{B}$ un bosque aleatorio. Entonces definimos la matriz de **proximidad** $Prox(\mathcal{B}) = \big(prox(i,j)\big)_{1\leq i,j\leq n}$ asociada al bosque aleatorio $\mathcal{B}$ donde los coeficientes satisfacen:

$$prox(i,j) = \frac{1}{B}\sum_{b=1}^{B} R^b(i,j)$$

Nota 2.13. Hay que notar que $\forall k \in \{1,2,...,n\}$ se tiene que $prox(k,k) = 1$ y que $prox(i,j) = prox(j,i)$. Luego, la matriz de proximidad es una matriz simétrica, donde la diagonal principal siempre tiene 1 como entrada y que está acotada por 1. Además, se puede demostrar que es una matriz definida positiva.

Nota 2.14. Podemos entender la matriz $Prox(\mathcal{B})$, como la matriz que recoge en la entrada $prox(i,j)$ la **proximidad** entre las observaciones i y j.

Propiedad 2.15. *La matriz* $A := \big(1 - prox(i,j)\big)_{1\leq i,j\leq n}$ *es una matriz de distancia.*

Observación 2.16. Dada la matriz de proximidades del bosque aleatorio, obtenemos la matriz de distancia A de la propiedad anterior, cuyas entradas a_{ij} nos determinan la relación de proximidad que tienen las observaciones i y j. Así, podemos aplicar a dicha matriz algoritmos de agrupación de forma que podamos determinar los grupos de observaciones que son similares entre ellos.

2.2.6. Tratamiento de valores faltantes

Es muy habitual en nuestras muestras originales encontrar que alguna observación no tiene todos los campos de entrada completos. A los valores que no aparecen en la muestra se les conoce como *valores faltantes*. Otra de las ventajas de los bosques aleatorios es el tratamiento que les da a los valores faltantes.

A continuación, explicaremos los procedimientos que se tratan en [4], aunque cabe destacar que hoy en día hay procedimientos más avanzados como los que proporciona el *surrogate split* explicado en [1, pág. 311].

Missquick: En primer lugar, si hay un dato faltante $x(m,j)$ para la observación m y la variable j numérica, sustituimos $x(m,j)$ por la mediana muestral de la variable j. En cambio si la variable j es categórica, se sustituye por la **moda** muestral.

Missright: En segundo lugar, se puede hacer uso de la matriz de proximidades para la sustitución. Supongamos que tenemos datos faltantes, por ejemplo $x(m,j)$, hacemos un primer reemplazo rápido (como por ejemplo en el caso anterior) y que no importa si es poco preciso. A continuación, hacemos crecer un bosque aleatorio para la muestra con el reemplazo rápido, por tanto obtendremos una matriz de proximidades, así sustituimos nuestros valores faltantes por:

a) En caso de que sea una variable numérica:

$$\hat{x}(m,j) = \frac{1}{B}\sum_{k=1}^{B} prox(m,k)\, x(k,j)$$

b) Si es una variable categórica ponderamos por su proximidad a la observación m y sustiuimos por $\hat{x}(m, j)$ la categoría más frecuente.

Se repite el proceso entre 4 y 6 iteraciones.

Nota 2.17. Es claro que la segunda forma de sustitución de valores faltantes es mucho más costosa computacionalmente, pero funciona muy bien cuando tenemos una base de datos con muchos valores faltantes. Por otra parte los autores de los bosques aleatorios [4], recomiendan usar la primera forma de sustitución, la más rápida, si hay un 20 % o menos de datos faltantes.

2.2.7. Bosques aleatorios para aprendizaje no supervisado

Como ya sabemos, cuando tratamos métodos de aprendizaje estadístico no supervisado, nos centramos en las relaciones entre las propias observaciones, ya que no tenemos en nuestra muestra original $\mathcal{X}$ ninguna variable de interés que etiquetar como variable respuesta. Así, para poder usar los bosques aleatorios para problemas de aprendizaje no supervisado crearemos una muestra artificial a partir de la original.

En primer lugar, creamos una variable binaria que etiqueta por clase 0 a nuestras observaciones de la muestra original. A continuación, siendo n el tamaño de nuestra muestra original, realizamos para cada variable, n extracciones siguiendo la distribución muestral univariada de dicha variable en la muestra original. Así, hemos obtenido nuestro conjunto de observaciones artificial, las cuales serán etiquetadas por la clase 1 de nuestra variable respuesta binaria.

Si consideramos como $\mathcal{X}_0$ la muestra constituida por la original y la artificial, podemos aplicar los bosques aleatorios a dicha muestra sobre la variable binaria creada artificialmente. Luego, usaremos árboles de clasificación de 2 categorías en los bosques aleatorios.

Para terminar, nuestro estudio se centrará en la matriz de proximidades, donde prescindiremos de las observaciones artificiales. Es decir, nos restringiremos a estudiar las proximidades y las distancias aportadas por el método de aprendizaje, entre las observaciones originales. Con dichas proximidades, podemos emplear técnicas de *clustering* que solo necesiten de distancias como *Partitioning Around Medoids* (PAM), u otras técnicas como el *escalado multidimensional*.

Observación 2.18. En [4] se puntualiza que si el ratio de mal clasificados que aporta el bosque aleatorio, es superior al 40 %,entonces es que no hay una dependencia clara entre las variables y, por tanto, no podemos extraer conclusiones de nuestros resultados.

Capítulo 3

Implementación en R

En este capítulo, nuestro objetivo es abordar la implementación computacional en `R` de las técnicas de aprendizaje explicadas anteriormente.

En primer lugar, en este trabajo usaremos el paquete de `R` `randomForest`, desarrollado en [7] y mantenido por Andy Liaw. Dicho paquete se fundamenta en el código original desarrollado en Fortran por Leo Breiman y Adele Cutler y publicado en [4]. Explicaremos con detalle la función principal del paquete, también denotada por `randomForest()`, aunque también haremos mención y describiremos funciones secundarias que nos permitirán mostrar e ilustrar ciertas métricas de interés para la interpretación de los datos.

3.1. Funciones del paquete `randomForest`

La función principal `randomForest()` implementa el algoritmo de bosques aletorios para regresión y clasificación creado por Leo Breiman en [5].

a) En este trabajo, fijaremos como forma de uso de la función principal:

```
randomForest(x, y=NULL, xtest=NULL, ytest=NULL, ntree=500,...)
```

b) Respecto a **los argumentos principales** que puede recibir la función:

x matriz de predictores de tamaño $n \times p$, siendo n el número de observaciones y p de variables predictoras.

y vector que hace referencia a nuestra variable respuesta. Si es de tipo factor, el algoritmo trabaja con árboles de clasificación, si es numérico usa árboles de regresión. En caso de omitirse, el algoritmo trabaja como aprendizaje no supervisado.

na.action tipo de tratamiento que reciben los datos faltantes. Por defecto, se usa `na.fail`, cuya función es devolver un error en caso de encontrar datos faltantes.

xtest matriz de predictores del conjunto test.

ytest valores de la variable respuesta del conjunto de datos test.

ntree número de árboles del bosque aleatorio. Por defecto se hacen crecer 500 árboles.

mtry valor del parámetro m. Por defecto, se siguen las recomendaciones expuestas en [4].

replace ¿las extracciones se hacen con reemplazamiento? Por defecto, `replace=TRUE`.

nodesize tamaño mínimo de los nodos terminales. Por defecto, en regresión el tamaño mínimo de los nodos es 5, mientras que es 1 para clasificación.

maxnodes máximo número de nodos terminales para los árboles del bosque. Por defecto, `maxnodes=NULL`.

importance ¿se debe calcular la importancia de las variables? Por defecto, `importance=FALSE`.

localImp ¿se debe calcular la importancia local para cada observación? Por defecto, `localImp=FALSE`.

nPerm número de permutaciones que realizamos sobre las observaciones OOB, para calcular la importancia de la variable j-ésima.

proximity ¿se deben calcular las proximidades de nuestras observaciones? Por defecto, `proximity=FALSE`.

oob.prox ¿se calculan solo las proximidades sobre las observaciones OOB?

keep.forest ¿guardamos los árboles involucrados en el bosque?

c) Si nos centramos en el objeto que devuelve la función `randomForest()`, podemos observar que es una lista con los siguientes elementos:

call nos devuelve como hemos llamado a nuestra función

type tipo de problema sobre el que estamos trabajando: regresión, clasificación o no supervisado.

predicted valores de la variable respuesta que aporta el modelo sobre las observaciones OOB.

importance objeto que almacena los vectores que miden la importancia de las variables. Para regresión, la primera columna se calcula mediante permutaciones como se explica en 2.2.4 b) y la segunda como en 2.2.4 a).

oob.times cantidad de veces que la observación i-ésima es del tipo OOB.

proximity en caso de que hayamos pedido calcular las proximidades en los argumentos, este objeto almacena la matriz de proximidades.

mse se calcula el sumatorio de los residuos al cuadrado como en (1.2).

Nos centramos ahora en algunas de las **funciones secundarias** del paquete. Tal y como hemos comentado en la Nota 2.5, en este trabajo se considerará m como un valor adaptable a los datos del problema en cuestión. En el paquete hay una función llamada `tuneRF()` que pretende abordar y automatizar esta problemática y cuya finalidad es encontrar el valor **óptimo** de m. Para ello usará el error OOB como métrica de referencia.

Como argumentos principales de la función `tuneRF()` podemos destacar:

x matriz con los predictores.

y vector de la variable respuesta.

mtryStart valor inicial para el parámetro m.

ntreeTry número de árboles que utilizaremos en cada bosque aleatorio. Por defecto el valor es 50.

stepFactor incremento del parámetro m en cada iteración.

Respecto a la importancia de las variables, en el paquete podemos encontrar las funciones `importance()` y `varImpPlot()`, donde la primera tiene una salida matricial con los valores en los que se cuantifica la importancia de las variables; en cambio, la segunda representa de manera visual los valores anteriores (en escala). Ambas funciones reciben como argumento un objeto de la clase `randomForest`.

Finalmente, entre las funciones del paquete que tratan los valores faltantes, podemos destacar dos: `na.roughfix()` y `rfImpute()`. La primera función, sustituye cada valor faltante de la variable j-ésima por la mediana (caso numérico) o por la moda (caso categórico). En cambio, la segunda función, utiliza las ponderaciones que nos otorga la matriz de proximidades, como hemos explicado en el apartado 2.2.6 b) del trabajo. Esta última función, `rfImpute()`, recibe como argumentos:

x matriz con los predictores.

y vector de datos de la variable respuesta. No se permiten NA.

iter número de iteraciones que se llevan a cabo para imputar.

ntree número de árboles del RF.

Nota 3.1. Más adelante podemos usar otras funciones contenidas en el paquete, pero no las explicaremos ya que su función será de representación o de validación y su descripción no se ajusta a los objetivos del trabajo.

Capítulo 4

Aplicación

En este capítulo, nos centraremos en primer lugar en la base de datos creada *ad hoc* para el trabajo, para ello mencionaremos los indicadores que contiene y la fuente de donde han sido extraídos. En segundo lugar, usaremos los datos mencionados y aplicaremos los bosques aleatorios con la finalidad de extraer conocimiento y valor de los mismos.

Nota 4.1. Puesto que el índice de Gini no es un indicador muy común en el contexto en el que se ubica dicho trabajo, hemos decidido dedicar una sección del capítulo para su explicación. Por otra parte, el resto de indicadores se consideran conocidos para los lectores.

4.1. Base de datos

Para el desarrollo práctico del trabajo, hemos decidido aplicarlo a datos que hagan referencia a las realidades socio-económicas de algunos países. El estudio de este tipo de datos puede ser relevante para proponer acciones que puedan ayudar en determinados países.

La base de datos es inédita, ha sido construida para este trabajo, a partir de datos extraídos de [8]. En dicha organización llamada *WorldBank*, el *Development Data Group* coordina el trabajo estadístico y mantiene una serie de bases de datos macroeconómicas, financieras y sectoriales. Gran parte de los datos provienen de los sistemas estadísticos de los países miembros, y la **calidad de los datos globales depende del desempeño de estos sistemas nacionales**. Además, el Banco Mundial trabaja con los países en el desarrollo y mejora de la capacidad, eficiencia y eficacia de sus propios sistemas estadísticos nacionales.

Observación 4.2. Nos hemos centrado en indicadores del año 2016, para los casos en que este no se hubiera calculado para dicho año, se ha usado el más reciente hasta la fecha, con límite en el año 2000. Si no se ha calculado en todo el siglo XXI, se considera dato faltante `NA`.

Respecto a los indicadores que hemos incluido en el trabajo los podemos clasificar, en función de su tipología:

1. **Económicos:** porcentaje del PIB en impuestos, exportaciones, importaciones, gasto social y formación bruta de capital (**FCB**).

2. **Educativos:** tasa de alfabetización por género.

3. **Sanitarios:** camas de hospital cada 100.000 personas, esperanza de vida por género y porcentaje de muertes causadas por enfermedades infecciosas transmisibles, condiciones maternales, prenatales y en materia de nutrición.

4. **Sociales:** porcentaje de población activa desempleada por género (modelado de la Organización Internacional del Trabajo), renta per cápita (en US$ actuales).

5. **Seguridad:** homicidios intencionales por cada 100.000 habitantes.

6. **Desigualdad:** índice de Gini.

7. **Complementarios:** porcentaje del PIB en ciencia, educación y porcentajes sobre las exportaciones totales de productos de alta tecnología y combustible.

Finalmente, nuestra base de datos consta de 266 observaciones (filas) y 20 variables (columnas).

4.1.1. Curva de Lorenz e índice de Gini

El grado de desigualdad económica existente en una sociedad y su evolución en el tiempo son temas que mantienen el interés permanente de la opinión pública y de los especialistas en el estudio del bienestar colectivo. Por su parte, en la literatura que aborda el análisis de la distribución del excedente de la economía, se han propuesto diferentes medidas que pretenden sintetizar esta variable, con el objeto de efectuar comparaciones intertemporales y entre países, a la vez de permitir asignar un valor absoluto a la desigualdad y derivar conclusiones sobre el nivel de concentración del ingreso en una población determinada. En este trabajo, usaremos la medida conocida como **índice de Gini**.

El índice de Gini es uno de los indicadores sintéticos más utilizados en el análisis estadístico para medir la desigualdad, debido a su facilidad de cálculo e interpretación. Dicho índice aporta un valor numérico $I_G \in [0, 100]$ donde 0 representa la perfecta **igualdad** en la tasa de participación de los ingresos de la población y 100 la perfecta **desigualdad**.

Para poder entender cómo se define el índice de Gini, antes debemos comprender qué es la **curva de Lorenz**. Esta medida fue propuesta en 1905 con el propósito de ilustrar la desigualdad en la distribución de la salud y, desde su aparición, su uso se ha popularizado entre los estudiosos de la desigualdad económica.

Definición 4.3. Sea un conjunto poblacional de tamaño N. Ordenamos los ingresos de cada individuo y_i de manera ascendente, es decir, $\mathcal{Y} = \{y_i\}_{i=1}^{N}$ donde se satisface que

$$y_1 \leq y_2 \leq \ldots \leq y_N.$$

La **curva de Lorenz** es la que representa el porcentaje acumulado de ingreso recibido, de acuerdo a la cuantía de su ingreso. En otras palabras, es una función de distribución del ingreso poblacional.

En un primer momento, el **coeficiente de Gini** se definió como:

$$CG = \frac{1}{2\,\mu}\left(\frac{\sum_{i\neq j}|y_i - y_j|}{N\,(N-1)}\right) \quad \text{donde} \quad \mu = \frac{1}{N}\sum_{i=1}^{N} y_i. \tag{4.1}$$

Posteriormente, se propuso una nueva manera de calcular el indicador, que se demostró que era equivalente a la que se ha presentado previamente.

$$CG = 1 - 2\,A_{CL}, \tag{4.2}$$

donde A_{CL} representa el área bajo la curva de Lorenz.

Observación 4.4. El coeficiente de Gini CG es un valor entre 0 y 1. El índice de Gini se define como $I_G = 100\,CG$.

Ejemplo 4.5. Veamos un ejemplo en la Figura 4.1 de dos curvas de Lorenz. Estas reflejan dos ejemplos ficticios de posibles distribuciones en el ingreso.

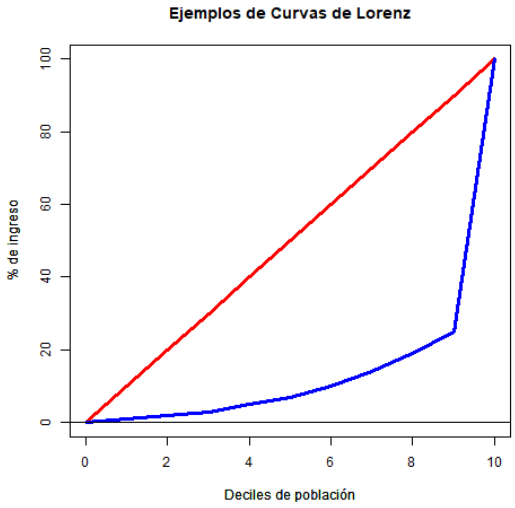

Figura 4.1: Ejemplos de curvas de Lorenz.

La diagonal es una curva de Lorenz que representa la perfecta igualdad. Por su parte, la curva que queda por debajo, muestra una clara desigualdad en el ingreso, ya que solamente un 10 % de la población dispone del 75 % de los ingresos totales de los individuos del país. De hecho, el área bajo la curva azul representa un 13′6 % del total, por tanto si aplicamos la ecuación 4.2, sabemos que

$$CG = 1 - 2\,(0,136) = 0{,}728.$$

4.2. Análisis principal de los datos

En esta sección abordaremos el análisis de nuestra base de datos. En primer lugar, mostremos su estructura:

```
x = read.csv(file = "BBDDTFM_Regiones.csv")
str(x)
```

```
'data.frame': 266 obs. of  23 variables:
$ nombres                    : chr  "Aruba" NA "Afganistan" NA ...
$ codigo_nombres             : chr  "ABW" "AFE" "AFG" "AFW" ...
$ Impuestos                  : num  NA 19.38 9.5 NA 9.73 ...
$ GastoPIB                   : num  NA 25.4 43.9 NA 17.5 ...
$ GastoCiencia               : num  NA 0.646 NA 0.226 NA ...
$ EducacionGasto             : num  5.52 4.22 4.23 3.03 3.42 ...
$ ExpPIB                     : num  71.8 25.5 NA 17.6 28.1 ...
$ ExportaAltaTecnologia      : num  4.04 7.83 NA 3.36 16.23 ...
$ ExportacionesCombustible   : num  0.105 30.969 3.45 67.505 92.701 ...
$ ImpPIB                     : num  70.4 29.1 NA 22.5 25.2 ...
$ CapitalFormation           : num  20.6 21.9 NA 20.7 27.2 ...
$ AlfabetH                   : num  96.9 NA 45.4 NA 80 ...
$ AlfabetM                   : num  96.7 NA 17 NA 53.4 ...
$ CamasHospital              : num  NA 0.912 0.5 NA 0.8 ...
$ CausasdeMuerte             : num  NA 52.9 39.4 61.9 61.6 ...
$ EVH                        : num  73.3 60.6 62.3 55.9 57.2 ...
$ EVM                        : num  78.2 64.9 65.3 58 62.8 ...
$ ParoH                      : num  NA 6.11 10.57 5 6.9 ...
$ ParoM                      : num  NA 7.04 14.33 6.08 7.5 ...
$ HomicidiosIntencionados    : num  1.93 10.3 6.55 8.1 4.85 ...
$ RentaCapita                : num  28452 1401 509 1666 3506 ...
$ GiniInd                    : num  NA NA NA NA 42.7 33.7 NA NA 32.5 42
  ...
$ region                     : int  0 0 0 0 0 0 0 1 0 0 ...
```

4.2.1. Matriz de predictores y tratamiento de los datos faltantes

Como hemos explicado en la Nota A.1, hemos añadido una variable binaria que identifica las regiones. Puesto que nuestro análisis se va a centrar en países, vamos a descartar estas observaciones.

Por otra parte, vamos a considerar como variable respuesta, en la parte supervisada de nuestro estudio, el **índice de Gini**. Ahora bien podemos observar en la estructura de los datos anterior que hay datos faltantes (`NA`) para este indicador. Así, las observaciones con `NA` en dicho indicador no van a intervenir en el entrenamiento, si no que serán usadas posteriormente para predecir su posible valor. Finalmente, puesto que en los bosques aleatorios intervienen elecciones aleatorias fijaremos con una semilla aleatoria, de forma que aseguraremos la reproducibilidad de las soluciones para cada nueva ejecución del script. Así, veamos la estructura de la base de datos con la que vamos entrenar nuestro modelo:

```
set.seed(2021)
# eliminamos las regiones
x = x[(x$region==0), ]
```

```
# detectamos las observaciones con indice de Gini conocido
train = which(is.na(x$GiniInd)==FALSE)
str(x[train, ])
```

```
'data.frame': 160 obs. of  23 variables:
$ nombres                   : chr  "Angola" "Albania" "Emiratos <c1>rabes
    Unidos" "Argentina" ...
$ codigo_nombres            : chr  "AGO" "ALB" "ARE" "ARG" ...
$ Impuestos                 : num  9.7325 17.5902 0.0435 12.0973 21.2774
    ...
$ GastoPIB                  : num  17.51 23.96 4.47 26.18 25.71 ...
$ GastoCiencia              : num  NA 0.154 0.7 0.613 0.241 ...
$ EducacionGasto            : num  3.42 3.96 NA 5.55 2.76 ...
$ ExpPIB                    : num  28.1 29 101 12.5 33.7 ...
$ ExportaAltaTecnologia     : num  16.233 0.784 2.618 9.022 6.138 ...
$ ExportacionesCombustible: num  92.7 11.18 31.56 2.5 3.49 ...
$ ImpPIB                    : num  25.2 45.8 75.7 13.6 42.3 ...
$ CapitalFormation          : num  27.2 25.2 26 17.7 18 ...
$ AlfabetH                  : num  80 98.4 92.6 99.1 99.8 ...
$ AlfabetM                  : num  53.4 96.1 95.1 99.1 99.7 ...
$ CamasHospital             : num  0.8 2.89 1.36 4.95 4.2 3.84 7.42 4.82
    0.79 5.76 ...
$ CausasdeMuerte            : num  61.58 2.96 7.03 15.49 4.81 ...
$ EVH                       : num  57.2 76.4 76.8 72.8 71 ...
$ EVM                       : num  62.8 80.1 78.8 79.6 77.9 ...
$ ParoH                     : num  6.9 16.12 1.12 7.12 17.73 ...
$ ParoM                     : num  7.5 14.45 4.2 9.14 17.48 ...
$ HomicidiosIntencionados : num  4.847 2.737 0.705 6.033 2.963 ...
$ RentaCapita               : num  3506 4124 38142 12790 3592 ...
$ GiniInd                   : num  42.7 33.7 32.5 42 32.5 34.4 30.8 26.6
    38.6 27.6 ...
$ region                    : int  0 0 0 0 0 0 0 0 0 0 ...
```

Como podemos observar, hemos reducido a 160 nuestro número de observaciones, que son todas países con índice de Gini conocido. Además, las variables del tipo `chr` no van a ser usadas para el entrenamiento, tampoco usaremos la variable `region` puesto que su única finalidad era discernir entre países y regiones.

En referencia al tratamiento de los datos faltantes existentes en nuestros datos de entrenamiento, los vamos a tratar mediante el reemplazo por la mediana muestral como hemos explicado en 2.2.6, es decir vamos a usar la función `na.roughfix()`, así:

```
require(randomForest)

# posicion de la variable respuesta
indG = which(names(x)=='GiniInd')

# nos quedamos con las variables numericas excepto region y la de
    respuesta

trainM = na.roughfix(x[train, -c(1,2,indG,23)])
dim_trainM = dim(trainM)
```

```
10
11    # damos nombres a las filas para saber de que pais se trata
12    row.names(trainM) = x[train, 1]
13
14    #guardamos nuestro v.respuesta para nuestro cjto de datos
15    resp_trainM = x[train,indG]
```

Notemos que nuestra matriz de predictores es de tamaño 19 columnas por 160 observaciones, cuyos nombres hacen referencia a los países a los que corresponden los datos. Además, hemos registrado el valor del índice de Gini para las observaciones asociadas a nuestra matriz de predictores.

4.2.2. Parámetros óptimos para los bosques aleatorios

De acuerdo con [5], los resultados de los bosques aleatorios, solo son sensibles a la elección del parámetro `mtry`. Sin embargo, aquí ajustaremos los siguientes parámetros:

1. `ntree`: el número de árboles.
2. `mtry`: número de predictores aleatorios escogidos para cada separación.
3. `nodesize`: número de observaciones mínimas para cada nodo terminal.

Empezamos en primer lugar, por el tamaño del árbol `ntree`. Hay que recordar que en el paquete de `R randomForest`, la función `plot()` proporciona una gráfica donde podemos observar el **error OOB** en función del tamaño del bosque. Además, el argumento `do.trace` en la función principal del paquete permite recoger la evolución de dicho error según el número de árboles.

```
1     # aplicamos los RF sobre nuestros datos
2
3     modelo_inicial = randomForest(x = trainM, y =resp_trainM, do.trace =
          25,
4     proximity = TRUE)
5
6     # guardamos nuestra grafica
7     png("ajuste_ntree")
8     plot(modelo_inicial, col = "blue")
9     dev.off()
```

Como se puede observar en el siguiente resultado aportado por el `R`, es suficiente considerar como tamaño del bosque $N = 100$, ya que a partir de este valor el error OOB se estabiliza.

```
1     |      Out-of-bag    |
2     Tree|      MSE  %Var(y) |
3     25  |    30.71    52.62 |
4     50  |    27.47    47.08 |
5     75  |    28.29    48.48 |
6     100 |    27.69    47.45 |
7     125 |    27.96    47.92 |
```

```
 150 |     27.86     47.74 |
 175 |     27.56     47.23 |
 200 |     27.61     47.31 |
 225 |     27.85     47.73 |
 250 |     27.83     47.70 |
 275 |     27.67     47.42 |
 300 |     27.84     47.71 |
 325 |     27.84     47.70 |
 350 |     27.98     47.94 |
 375 |     27.92     47.84 |
 400 |     27.79     47.63 |
 425 |     27.69     47.45 |
 450 |     27.62     47.33 |
 475 |     27.71     47.48 |
 500 |     27.49     47.11 |
```

Además, si queremos visualizar estos resultados, podemos introducir la gráfica que arroja la función `plot()`, como podemos ver en la Figura 4.2:

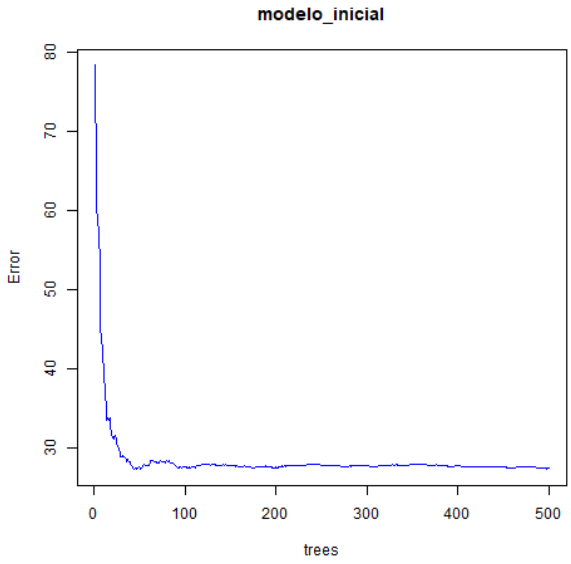

Figura 4.2: Error OOB en función del número de árboles para el modelo inicial.

Así, ya hemos determinado el parámetro para `ntree`.

Observación 4.6. Notemos que por las propiedades de los bosques aleatorios, el parámetro **no es crítico**, puesto que, una vez superado cierto umbral (en este caso los 100 árboles aproximadamente) el incremento en su valor no da lugar a sobreajustes de nuestro modelo.

Para determinar el parámetro `mtry`, vamos a razonar de manera análoga a la del parámetro anterior, ya que en la Figura 4.3 vamos a mostrar el error OOB en función del parámetro m.

```
# los posibles valores de m son

pos_m = 1:dim(trainM)[2]

# para cada valor de m, calculamos el error OOB

oobsMtry = sapply(pos_m, function(x) {
  RF_m = randomForest(x = trainM, y = resp_trainM, ntree = N, mtry = x)
  return(RF_m$mse[RF_m$ntree])
})

plot(x = 1:length(oobsMtry), y = oobsMtry, type = 'l', xlab = "Valor
    para m",
ylab = "Error MSE OOB")
```

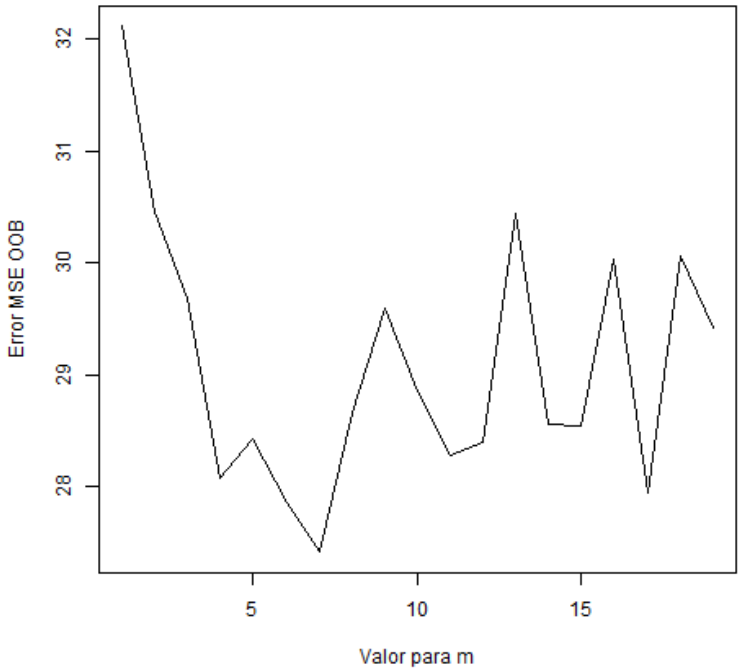

Figura 4.3: Error OOB en función de la cantidad de variables elegidas para cada separación.

Como se puede observar hay mucha variación en el error en función del valor de m. Con la finalidad de estabilizar el error MSE OOB para los valores de m, vamos a usar el comando `replicate`, y realizaremos 10 y 20 iteraciones para cada valor posible de m promediando sus resultados. Posteriormente volveremos a mostrar la gráfica anterior para decidir el valor del parámetro m **óptimo**. Luego, en la Figura 4.4 podemos ver los resutados:

```
    oobsMtry10 = sapply(pos_m, function(x) {
      v = replicate(n = 10, randomForest(x = trainM, y = resp_trainM, ntree
          = N,
      mtry = x)$mse[N])
      return(mean(v))
    })

    oobsMtry20 = sapply(pos_m, function(x) {
      v = replicate(n = 20, randomForest(x = trainM, y = resp_trainM, ntree
          = N,
      mtry = x)$mse[N])
      return(mean(v))
    })


    plot(x = 1:length(oobsMtry10), y = oobsMtry10, type = "l", col = "red",
    axes = FALSE, xlab = "Valor para m", ylab = "Error MSE OOB")
    points(x = pos_m, y = oobsMtry10, pch = 21, bg = "black")
    par(new=TRUE)
    plot(x = 1:length(oobsMtry20), y = oobsMtry20, type = "l", col = "blue"
        ,
    lty = 5, xlab = "Valor para m", ylab = "Error MSE OOB")
    points(x = pos_m, y = oobsMtry20, pch = 21, bg = "black")

    legend(x = "topright",          # Posicion
    legend = c("10 RF", "20 RF"), # Textos de la leyenda
    lty = c(1, 5),           # Tipo de lineas
    col = c("red", "blue"),          # Colores de las lineas
    lwd = 2)                 # Ancho de las lineas
```

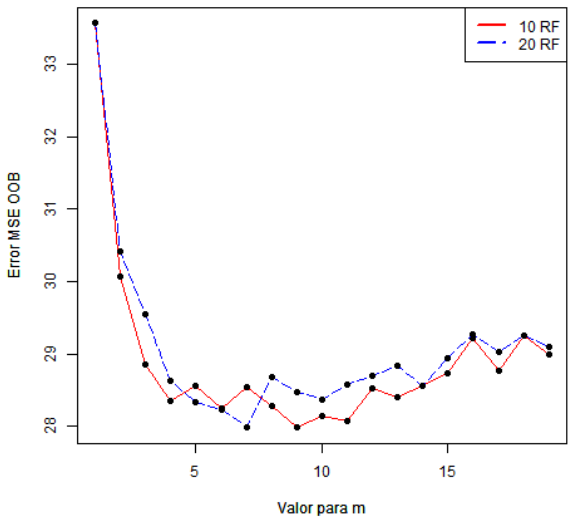

Figura 4.4: Error OOB **estabilizado** en función de la cantidad de variables elegidas para cada separación.

Como podemos observar en la Figura 4.4 cuando realizamos 20 iteraciones el valor $m = 7$ arroja el mejor error OOB, en comparación con los otros posibles valores. Por otra parte, cuando realizamos 10 iteraciones el mínimo se alcanza para $m = 9$. Así, consideraremos de ahora en adelante dos modelos, el primero con $m_1 = 7$ y el segundo con $m_2 = 9$.

Nota 4.7. Hay que recordar que por la Nota 2.5, sabemos que la recomendación en este caso sería que $m = \frac{19}{3} \approx 6$, pero como hemos podido observar en este caso, los valores que minimizan el error OOB no siempre coinciden con la recomendación. Este motivo es el que justifica que consideremos la elección del valor de m, como un parámetro ajustable que dependa de los datos.

Finalmente, razonamos de manera idéntica para el valor del parámetro `nodesize`. Obtenemos las siguientes gráficas que mostramos en la Figura 4.5, dependiendo de m_i $i = 1, 2$.

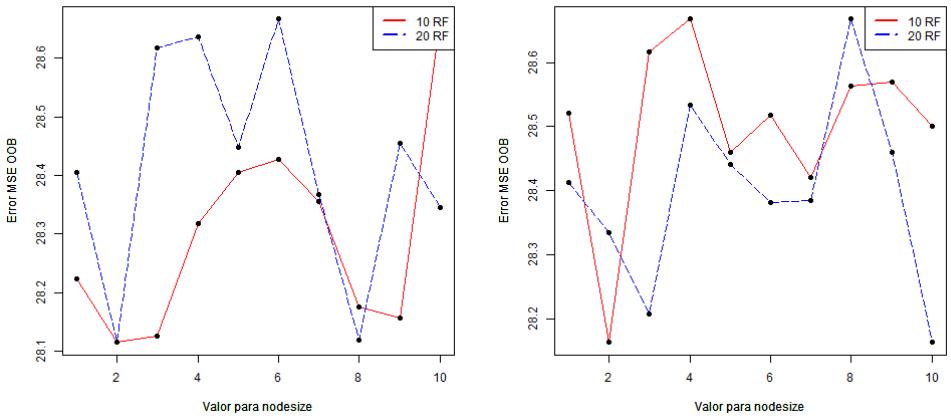

Figura 4.5: Error OOB **estabilizado** en función de la cantidad de observaciones mínimas para cada nodo terminal, para m_1 (izquierda) y m_2 (derecha).

Es fácil comprobar que la elección óptima para el parámetro `nodesize` es 2, ya que es la que mejores registros de error OOB obtiene para m_i con $i = 1, 2$.

Observación 4.8. Hay que destacar que en este caso hemos seguido una lógica *secuencia-lista*, es decir, en primer lugar se óptimiza un parámetro, para en segundo lugar optimizar el siguiente. Este razonamiento, no siempre arroja el modelo **más óptimo**, ya que para ello habría que estudiarlos a la vez, ya que sus posibles valores pueden influir entre ellos y en consecuencia en la precisión del modelo.

4.2.3. Modelo con los parámetros óptimos

Puesto que ya hemos calculado los diferentes valores de los parámetros, de manera que estos minimizan el error OOB, es el momento de aplicar los bosques aleatorios, con la finalidad de extraer conocimiento de nuestra base de datos. Así

```
    # valor de m=7
    modelo_opt = randomForest(x = trainM, y = resp_trainM, ntree = N,
    mtry = m_opt, proximity = TRUE, importance = TRUE, nodesize = 2)

    # valor de m=9
    modelo_opt2 = randomForest(x = trainM, y = resp_trainM, ntree = N,
    mtry = m_opt2, proximity = TRUE, importance = TRUE, nodesize = 2)
```

Notemos que al indicar como `proximity = TRUE` e `importance = TRUE`, nuestro modelo calculará la matriz de proximidades y la importancia de nuestros predictores sobre la variable respuesta.

Una de las medidas más usadas para determinar **la bondad del ajuste** producido por el método, suele ser el **coeficiente de determinación** denotado como RSQ, que se calcula como:

$$RSQ = 1 - \frac{MSE}{Var(Y)}, \tag{4.3}$$

donde $Var(y)$ denota la varianza de la variable respuesta.

Cuando esta medida estadística es negativa o cercana a 0, quiere decir que el modelo es peor que usar la media muestral de nuestra variable respuesta. Por otra parte, cuando esta medida más cerca está de 1, mejor es la fiabilidad de las predicciones de nuestro modelo.

En el caso que nos ocupa, nuestros resultados han sido $RSQ_1 = 0{,}5331318$ y $RSQ_2 = 0{,}4982786$, para m_1 y m_2 respectivamente. Así, por un lado podemos afirmar que nuestros modelos son mejores que la ausencia del mismo, pero por otro lado la bondad del ajuste no es **lo suficientemente alta** como para garantizar que nuestras predicciones vayan a ser adecuadas para todas las observaciones.

Nota 4.9. Un modelo se suele considerar *bueno*, cuando tiene un coeficiente de determinación superior al 0,7

Observación 4.10. Cabe destacar la complejidad del problema, ya que estamos intentando predecir un índice sobre desigualdad en base a una **cantidad limitada** de predictores, los cuales en muchos casos han sido rellenados y además puede que no se correspondan con el año de cálculo de nuestra variable respuesta. Es por eso, que aunque no podamos calificar de *bueno* nuestro modelo, podamos estar satisfechos teniendo en cuenta las limitaciones existentes.

4.2.4. Importancia de las variables

Puesto que el modelo nos realiza un ranking sobre la importancia de las variables, veamos en la Figura 4.6 cuáles han sido las más determinantes para el primer modelo:

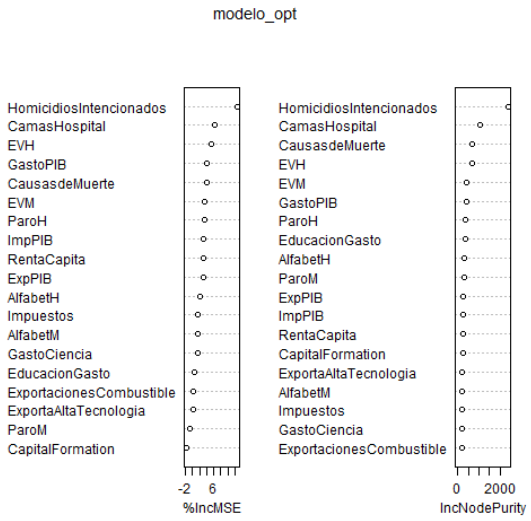

Figura 4.6: Ranking de importancia de los predictores sobre el índice de Gini para el modelo con $m_1 = 7$.

Notemos que el ranking de la izquierda de la Figura 4.6, se calcula en base a lo explicado en el punto 2 de la Sección 2.2.4. En cambio, el ranking de la derecha, usa la disminución del error total que ha producido sobre los árboles del bosque, tal y como explicamos en el punto 1 de la Sección 2.2.4.

En ambos rankings, el predictor **más fuerte** sobre nuestra variable respuesta es `HomicidiosIntencionados`. Si queremos visualizar el efecto marginal de los predictores más fuertes sobre Y, podemos usar la función `partialPlot()`. Dicha función estima el efecto marginal de la variable j-ésima, calculando la media empírica en función de los posibles valores que pueda tomar la variable en la posición j, es decir :

$$f_j(x) := \frac{1}{n} \sum_{i=1}^{n} \hat{h}_{RF}\big(x_i^1, ..., x_i^{j-1}, x, x_i^{j+1}, ..., x_i^p\big). \tag{4.4}$$

Veamos en la Figura 4.7 los efectos marginales de las 6 variables más importantes como son: `HomicidiosIntencionados`, `CamasHospital` ,`EVH` , `EVM` , `CausasdeMuerte` y `GastoPIB`.

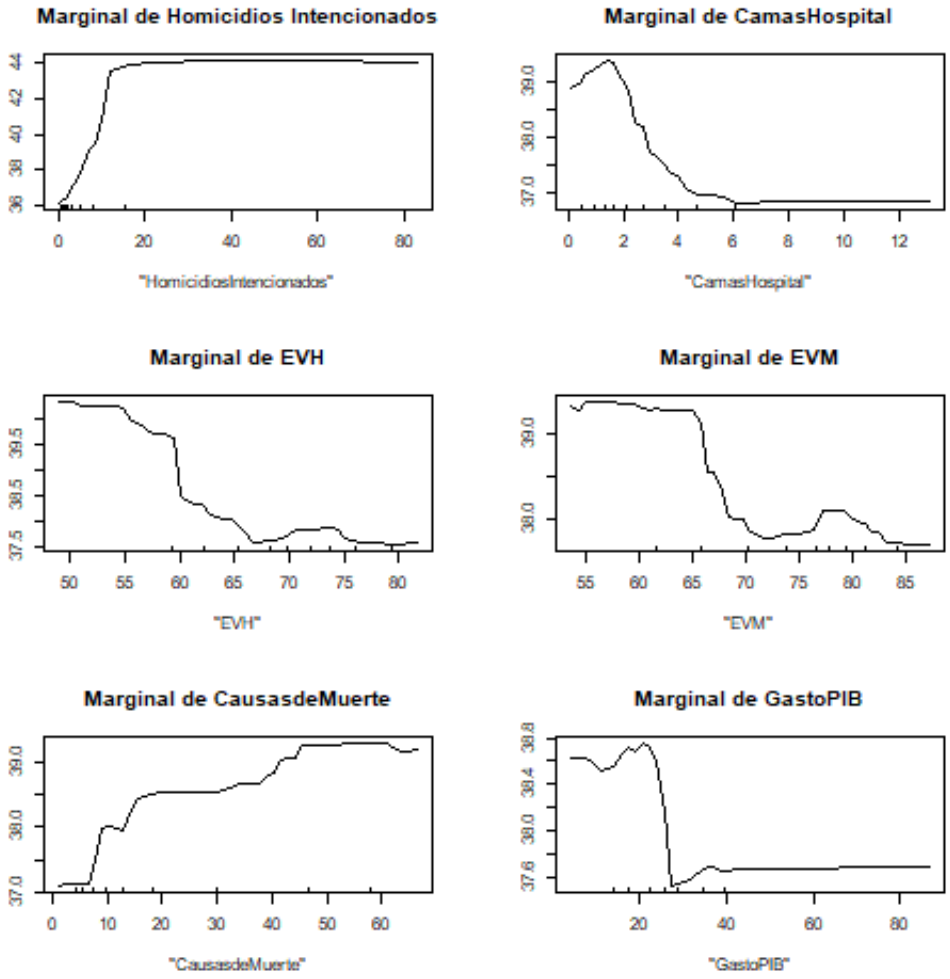

Figura 4.7: Función marginal de los predictores con más influencia sobre el índice de Gini.

De la Figura 4.7 podemos extraer las siguientes conclusiones:

1. Un claro indicador de una sociedad desigual en materia económica, es la **violencia** existente materializada en homicidios intencionados. Como podemos ver, empíricamente, sociedades donde hay una tasa de aproximadamente 15 homicidios intencionales (por cada 100.000 habitantes) tienen una diferencia de hasta 8 puntos, en relación al índice de Gini, con otras donde hay una clara *paz social*.

2. Respecto a las **capacidades sanitarias** del país, podemos observar el efecto positivo que aporta el incremento del número de camas de hospital disponibles, en el reflejo de sociedades con menos desiguales económicas. Ocurre lo mismo, con la esperanza de vida de la población, ya que sociedades en la que los ciudadanos viven más, son indicativas de una mayor distribución en el ingreso de su población. Finalmente, como era esperable, en los países donde hay todavía un gran número de muertes por causas maternales, prenatales o en materia de nutrición, tienden a padecer una desigualdad en el ingreso muy superior.

3. En referencia al **ámbito institucional**, cabe destacar que los países que dedican un 35 % de su PIB o superior en gasto público, empíricamente tienden a tener una diferencia de 1 sobre el índice de Gini con las que no lo hacen.

Nota 4.11. Solo introduciremos los resultamos para el modelo con $m_1 = 7$, ya que aporta un coeficiente de determinación superior al que usa $m_2 = 9$ y porque los resultados que arrojan ambos modelos son muy similares.

4.2.5. Residuos. Observaciones con alto error

Con la finalidad de obtener una visión holística acerca de la bondad del ajuste de nuestro modelo hemos tratado y explicado el coeficiente de determinación. Ahora bien, no es la única manera de hacerlo, puesto que es usual hacer un estudio de los residuos que nos aporta nuestro modelo, para poder diferenciar los países para los cuales el modelo no predice bien. Veamos en el Cuadro 4.1, la descriptiva de nuestros residuos:

Modelo con m_1	**Modelo con m_2**
Min. : 0.02917	Min. : 0.01786
1st Qu.: 1.77716	1st Qu.: 1.85867
Median : 3.32705	Median : 3.96471
Mean : 4.14131	Mean : 4.28000
3rd Qu.: 5.88900	3rd Qu.: 6.03673
Max. :16.53636	Max. :16.96351

Cuadro 4.1: Estadísticos descritivos básicos de los residuos para los modelos con m_1 y m_2 respectivamente.

Como podemos observar, en ambos modelos, para el 75 % de las observaciones se obtiene un error inferior a 6 puntos de diferencia con su valor **real** y de media se comete un error de 4 puntos en ambos. En cambio, hay países para los que los modelos es claro que no predice bien, como podemos ver en la Figura 4.8:

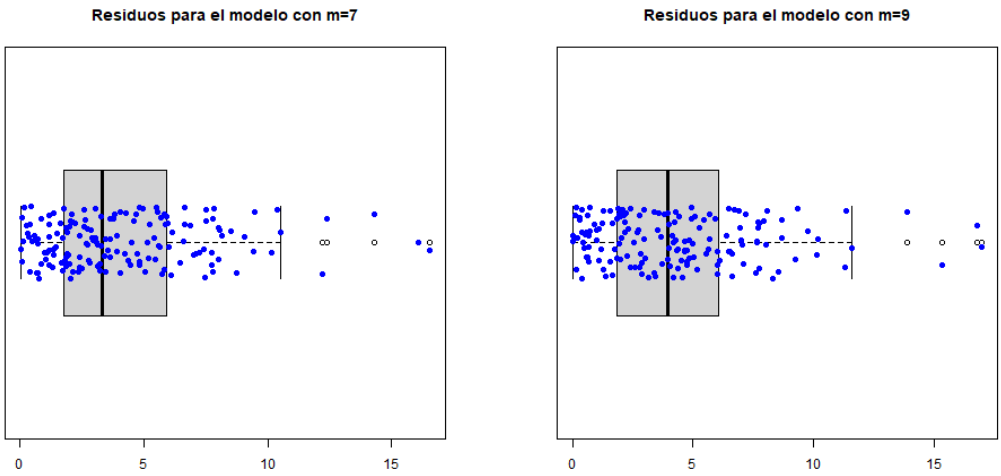

Figura 4.8: Diagrama de caja y bigotes para los residuos de ambos modelos.

Hay que notar que, existen valores de residuos atípicos (*outlier*), es decir, los modelos no consiguen ajustar bien dichas observaciones. Es por eso que, en los Cuadros 4.2 y 4.3 mostramos que países componen los *outliers*.

Nombre	**Predicción**	**Realidad**
Zambia	40.5	57.1
Iraq	45.6	29.5
República Centroafricana	41.9	56.2
Sudáfrica	50.5	63
Mozambique	41.8	54

Cuadro 4.2: Países con residuos atípicos para el modelo con parámetro m_1.

Nombre	**Predicción**	**Realidad**
Sudáfrica	46	63
Iraq	46.2	29.5
Zambia	41.7	57.1
República Centroafricana	42.2	56.2

Cuadro 4.3: Países con residuos atípicos para el modelo con parámetro m_2.

Este hecho se puede deber a diferentes cuestiones: político-culturales del propio país, falsedad en sus propias estadísticas etc. Aunque, me gustaría destacar el caso de *Iraq*; como vemos en todos los países donde el residuo es un valor atípico, la predicción es un valor inferior al real, en cambio en el caso de *Iraq* esto no ocurre, es decir, según nuestro modelo, debido a los predictores asociados *Sudáfrica* y a *Iraq* el índice de Gini estimado debería ser similar, pero como hemos visto la diferencia **es de más** de 30 puntos. Este hecho puede deberse a varios motivos: en primer lugar *Iraq* estuvo inmerso en un conflicto armado entre los años (2003-2011), proporcionando un contexto **muy difícil** para los estadísticos de nuestra muestra (sean durante la guerra o posteriores). Además en segundo lugar, la última vez que se calculó el índice de Gini en dicho país fue en el año 2012, justo el año posterior al fin del conflicto, haciendo que el coste en vidas y el contexto socio-político no fuera el adecuado como para considerarlo fiable.

4.2.6. Proximidades

Vamos a analizar en esta sección la matriz de proximidades, que hemos obtenido para ambos modelos. En primer lugar, en el Cuadro 4.4, podemos ver cuáles han sido los países *más similares* a España según nuestros modelos.

Notemos que el país más **similar** a España según el modelo que define m_1 es Italia, en cambio, para m_2 es Grecia. Hay que destacar también países como: Turquía, Irlanda y Suiza; los cuáles *a priori* no se esperaba que fueran *«más próximos»* que países como Portugal. En cualquier caso, hay que resaltar que el grado de proximidad con dichos países, es bajo, entorno al $0'2$. Recordemos que 1 indica la máxima proximidad y 0 la mínima.

Como hemos visto en la Observación 2.16, una vez determinada la matriz de proximidades, podemos obtener la matriz de distancia y a dicha matriz le podemos aplicar

País	Proximidades para m_1	País	Proximidades para m_2
España	1	España	1
Italia	0.333	Grecia	0.230
Irlanda	0.307	Luxemburgo	0.2
Albania	0.25	Armenia	0.181
Suiza	0.25	Italia	0.166
Turquía	0.25	Turquía	0.166

Cuadro 4.4: Países con mayor *similaridad* a España para m_1 (izquierda) y m_2 (derecha)

algoritmos de agrupación como, por ejemplo, el *Partitioning Around Medoids* (PAM). Así, vamos a proyectar nuestras observaciones en dos dimensiones en la Figura 4.9,para ello usaremos la función `cdmscale()`, y agruparemos mediante $k \in \{2, 3, 4, 5\}$ grupos.

Nota 4.12. Cada color indica la pertenencia a cada uno de los grupos, y los nombres representan los *medoides* de cada grupo.

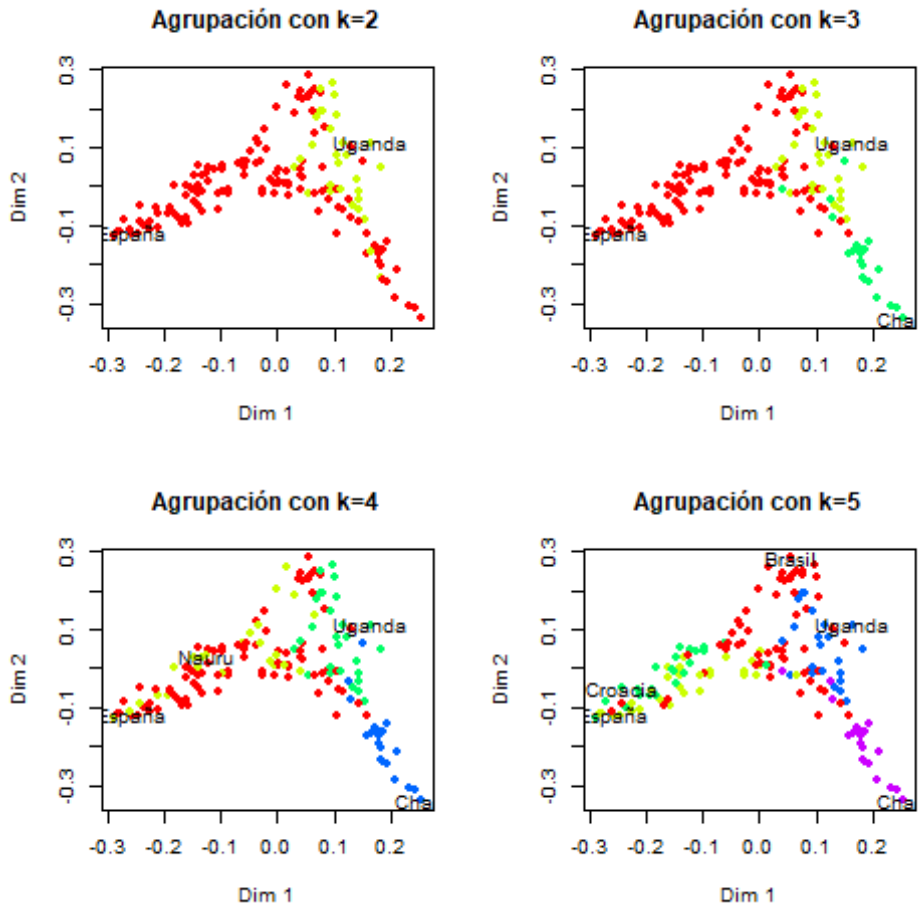

Figura 4.9: Proyección de las observaciones en función del número de agrupaciones.

4.2.7. Predicciones de los países sin índice de Gini conocido

Recordemos que en nuestra matriz de predictores, no se habían incluido las observaciones cuyo índice de Gini era un dato faltante. Puesto que ahora tenemos dos modelos *óptimos* (con m_1 y m_2 respectivamente), veamos en el Cuadro 4.5 qué descriptiva obtenemos sobre nuestras predicciones en dichos países:

Predicciones para m_1	Predicciones para m_2
Min. :32.30	Min. :31.82
1st Qu.:35.49	1st Qu.:35.12
Median :37.60	Median :37.53
Mean :38.88	Mean :38.75
3rd Qu.:42.78	3rd Qu.:42.51
Max. :46.96	Max. :46.53

Cuadro 4.5: Estadísticos descriptivos básicos, para las predicciones del índice de Gini, de los países donde no es conocido, hecha por los modelos m_1 (izquierda) y m_2 (derecha).

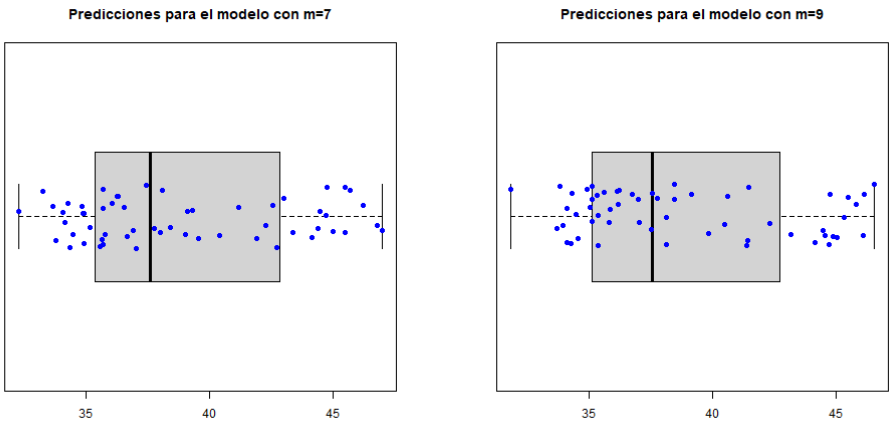

Figura 4.10: Diagrama de caja y bigotes para las estimaciones del índice de Gini, de los países donde no es conocido, hecha por los modelos m_1 (izquierda) y m_2 (derecha).

Como podemos observar en los datos anteriores, las predicciones tienen una media de aproximadamente 38 puntos en el índice de Gini y una mediana de 37 puntos, para ambos modelos. Es decir, la mayoría de países de los cuáles no conocemos su índice de Gini, son países **muy desiguales**, según las predicciones de nuestros modelos.

Nota 4.13. En la muestra de los países para los cuáles desconocemos el índice de Gini, hay aproximadamente un 38 % de datos faltantes. Además, dichos valores han sido sustituidos por su respectiva mediana muestral, es por eso, que nuestras estimaciones van a estar **influenciadas** por dichas **limitaciones**.

Por otra parte, en el Cuadro 4.6 podemos observar qué países son los más **redistributivos** según cada modelo.

	País	**Predicción m_1**	**País**	**Predicción m_2**
1	Corea, Rep. Democrática	32.305	Corea, Rep. Democrática	31.8235
2	Turkmenistán	33.2905	Mónaco	33.6815
3	Brunei Darussalam	33.686	Hong Kong	33.81
4	Hong Kong	33.815	Qatar	33.9485
5	Singapur	34.099	Aruba	34.083

Cuadro 4.6: Países más iguales económicamente según las predicciones de los modelos para m_1 (izquierda) y m_2 (derecha).

Hay que destacar que, según los resultados anteriores, el país más **redistributivo**, de entre los que no reportaban el índice de Gini, es *Corea, República Democrática*. A continuación, respecto a la segunda posición hay divergencias entre cada modelo, pero es cierto que para ambos modelos la región de *Hong Kong* en *China* recibe buenas estimaciones, en comparación a los otros países de la muestra.

Ahora bien,en el Cuadro 4.7 nos centramos en los países con **más desigualdad económica**, de entre los que no reportaban el índice de Gini, según nuestros modelos.

	País	**Predicción m_1**	**País**	**Predicción m_2**
1	Guinea Ecuatorial	46.9615	Guyana	46.5255
2	San Vicente y las Granadinas	46.779	Belice	46.1095
3	Belice	46.1805	Bahamas	46.0685
4	Guyana	45.6735	Puerto Rico	45.7975
5	Saint Kitts y Nevis	45.4835	Bermudas	45.4785

Cuadro 4.7: Países más desiguales económicamente según las predicciones de los modelos para m_1 (izquierda) y m_2 (derecha).

En ambos modelos, podemos observar que los países: *Guyana* y *Belice* reciben unas puntuaciones muy altas. En cambio, países como *Guinea Ecuatorial*, es el país peor puntuado según el modelo con parámetro m_1, en cambio para el modelo m_2 no se encuentra dentro de los 5 países con peor puntuación, pero su índice de Gini predicho con m_2 es 45, similar al obtenido con m_1 .

Dado que hay dos modelos que consideramos *óptimos*, podemos observar en la Figura 4.11 una comparación entre las predicciones de cada uno, con la finalidad de conocer si tienen un comportamiento **similar** como modelos, o si por el contrario **difieren** mucho respecto a sus estimaciones.

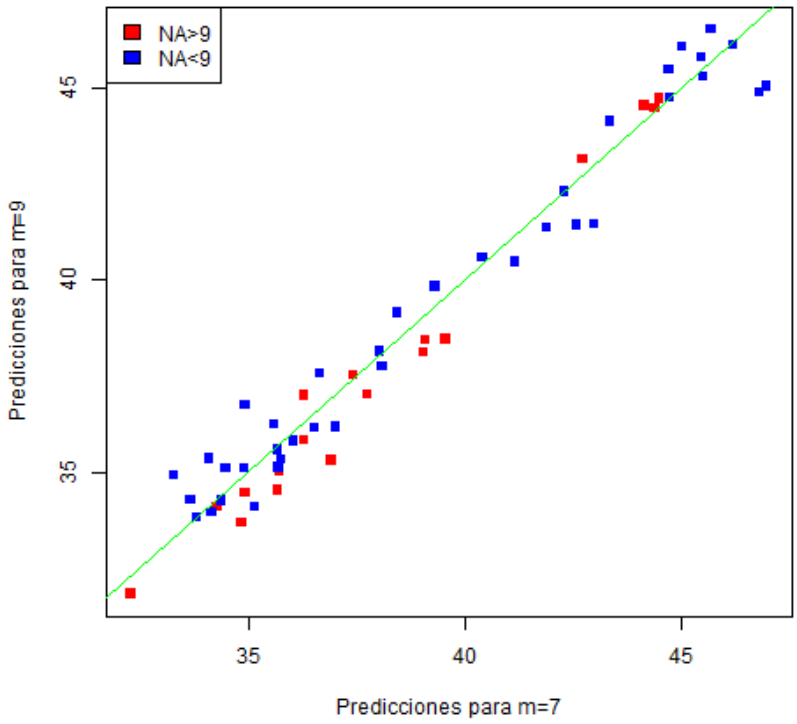

Figura 4.11: Comparación entre las predicciones del índice de Gini, de los países que no lo reportaban, para ambos modelos.

Notemos que los valores, para cualquier caso, están en un entorno *cercano* de la recta diagonal (recta verde en la Figura 4.11) $y = x$. Luego, podemos afirmar que el comportamiento **cualitativo** entre los modelos, no varía.

Observación 4.14. El estudio anterior, rellenando los datos faltantes mediante bosques aleatorios producía peores resultados, es por eso que ha sido omitido en esta sección.

4.3. Análisis complementario de los datos

En esta sección, continuaremos con el análisis de los datos, aunque vamos a llevar a cabo un planteamiento con matices diferentes al anterior, con la intención de sacar más conocimiento de nuestra base de datos.

4.3.1. Segmentación y regresión mediante bosques aleatorios

En el análisis de datos es muy habitual en primer lugar, realizar una segmentación de nuestras observaciones para poder discernir cuáles pertenecen al mismo grupo. En segundo lugar, una vez han sido definidos los grupos, aplicamos nuestros modelos para cada grupo. El razonamiento anterior, pretende extraer la mayor cantidad de conocimiento posible mediante el aumento de la granularidad de nuestra base de datos.

Para el caso que nos ocupa plantearemos el problema de la forma siguiente:

1. Los datos faltantes serán rellenados mediante la función `rfImpute()`, es decir, usaremos los bosques aleatorios.

2. Aplicaremos los bosques aleatorios de forma no supervisada, así, obtendremos una matriz de proximidades con la que poder trabajar.

3. El número de grupos lo determinaremos aplicando PAM con 2 a 5 grupos y calculando la silueta [12] que devuelve PAM, de forma que nos quedaremos con el número de grupos que devuelva mayor silueta.

4. Finalmente, aplicaremos los bosques aleatorios de forma supervisada sobre el índice de Gini, para cada uno de los grupos.

Como hemos comentado anteriormente, para usar los bosques aleatorios para rellenar los datos faltantes usaremos la función `rfImpute`. Así:

```
indG = which(names(x)=='GiniInd')
Gini = x[train, indG]
dataframe = rfImpute(x = x[train, -c(1,2,indG,23)], y = Gini , ntree =
    100, mtry = 9)
```

Puesto que, ya no tenemos datos faltantes en nuestra base de datos llamada `dataframe`, le vamos a aplicar los bosques aleatorios de forma **no supervisada**:

```
# aplicamos RF no supervisados
RFno_sup = randomForest(x = dataframe, proximity = TRUE, importance =
    TRUE, keep.forest = TRUE)
M = RFno_sup$proximity
```

Dado que, hemos calculado nuestra matriz de proximidades M, ya sabemos que podremos calcular nuestra matriz de distancia y aplicar el algoritmo de agrupación *PAM*, pero para poder hacerlo de forma adecuada, primero debemos calcular el **número de grupos**.

```
set.seed(2021)

MDist = as.matrix(1-M)

# que numero k de cluster
k = 2:10

silueta = sapply(k, function(u){
  clust = pam(MDist, k = u, diss = TRUE)
  return(clust$silinfo$avg.width)
})
```

En la Figura 4.12, podemos ver los resultados del promedio de la *silueta* que es la medida que usaremos para definir el número de grupos.

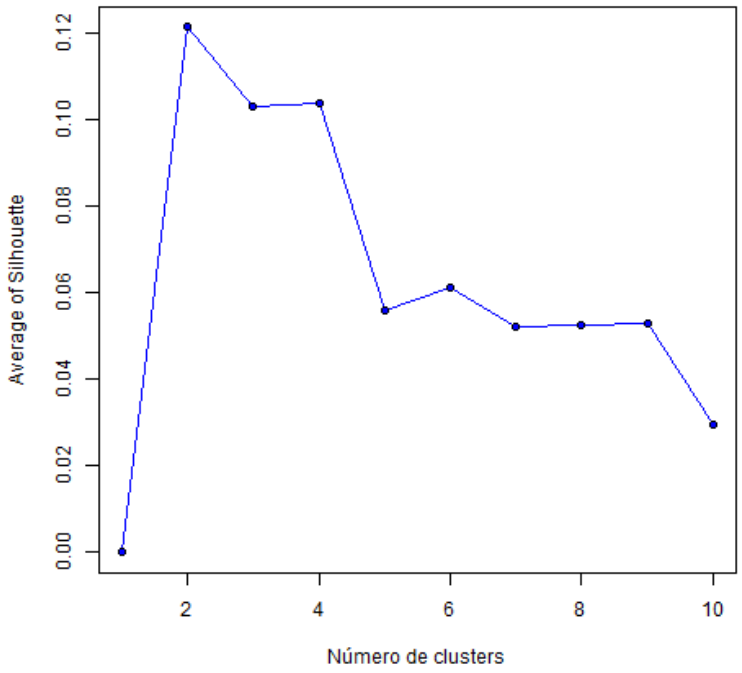

Figura 4.12: Cálculo de la *silhouette* para nuestra matriz de distancia.

Es claro que según la medida anterior, el valor adecuado de grupos es $k = 2$. Veamos en la Figrua 4.13 una proyección de las agrupaciones resultantes de aplicar el *PAM* con $k = 2$. Aunque, el valor de la silueta, nos indica que no hay grupos separados.

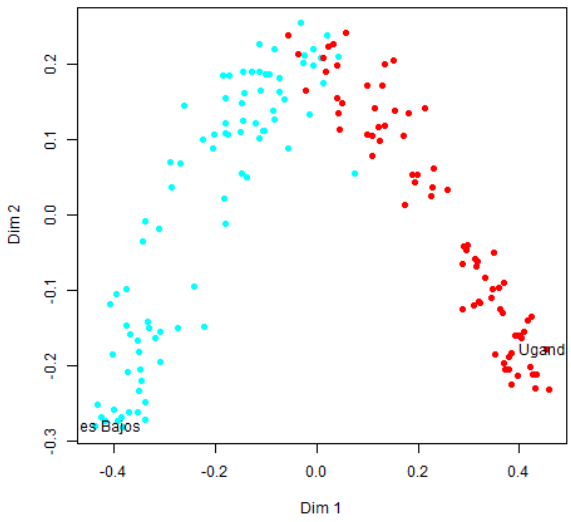

Figura 4.13: Proyección de los segmentos de países del modelo de bosques aleatorios no supervisados.

Aplicamos los bosques aleatorios para cada segmento de observaciones, definido por: *Países Bajos* y *Uganda*

```
#quitamos el indice de Gini

dataframe = dataframe[,-1]

modelo_seg1_m1 = randomForest(x = dataframe[cluster$clustering==1, ], y
    = Gini[cluster$clustering==1], importance = TRUE, ntree = 100, mtry
    = 7)

modelo_seg2_m1 = randomForest(x = dataframe[cluster$clustering==2, ], y
    = Gini[cluster$clustering==2], importance = TRUE, ntree = 100, mtry
    = 7)

modelo_seg1_m2 = randomForest(x = dataframe[cluster$clustering==1, ], y
    = Gini[cluster$clustering==1], importance = TRUE, ntree = 100, mtry
    = 9)

modelo_seg2_m2 = randomForest(x = dataframe[cluster$clustering==2, ], y
    = Gini[cluster$clustering==2], importance = TRUE, ntree = 100, mtry
    = 9)
```

En el Cuadro 4.8 podemos ver los valores para el *RSQ* para cada segmento y valor del parámetro m. Hay que notar que, los valores para el primer segmento, que es el que define *Uganda*, son muy bajos, tanto como para que podamos afirmar que el modelo es **insuficiente** como para proseguir con nuestro análisis. Respecto al segundo segmento, los valores son muy similares a los que hemos obtenido en el análisis principal.

	Modelo m_1	Modelo m_2
Segmento 1	0.1574911	0.2229155
Segmento 2	0.5278506	0.5499826

Cuadro 4.8: Valores de los *RSQ* para los modelos con m_1 y m_2.

En este caso, segmentar nuestra base de datos **no** ha tenido el efecto esperado en cuanto al aumento de **la precisión de las estimaciones** de los modelos y por tanto, **no seguiremos con nuestro análisis**.

4.3.2. Clasificación por nivel de desarrollo humano

En esta sección, aplicaremos los bosques aleatorios sobre una variable respuesta Y binaria, es decir, usaremos los bosques aleatorios en un problema de clasificación, donde sólo hay dos categorías.

Para ello, usaremos un estadístico conocido como ***índice de desarrollo humano*** (IDH), como se cita en [10] : *«Un país obtiene un IDH más alto cuando la esperanza de vida es mayor, el nivel de educación es mayor y el ingreso nacional bruto INB (PPA) per cápita es mayor»*. Por otra parte, en [9, pág. 25] obtenemos una clasificación, en función del IDH de los países, en 4 categorías: desarrollo humano muy alto, desarrollo humano alto, desarrollo humano medio y desarrollo humano bajo. Así, vamos a construir una variable binaria donde le asociaremos el valor 1 a las observaciones que hacen referencia a países con desarrollo humano medio o bajo: en cambio, asociaremos el valor 0 a países con desarrollo humano alto o muy alto.

```
cod_nom = c("SYR","MHL","VNM","PSE","IRQ","MAR","KGZ","GUY","SLV","TJK"
    ,
"CPV","GTM","NIC","IND","NAM","TLS","HND","KIR","BTN","BGD",
"STP","COG","SWZ","LAO","VUT","GHA","ZMB","GNQ","MMR","KHM",
"KEN","NPL","AGO","CMR","ZWE","PAK","SLB","PNG","COM","RWA",
"NGA","TZA","UGA","MRT","MDG","BEN","LSO","SEN","TGO","SDN",
"HTI","AFG","DJI","MWI","ETH","GMB","GIN","LBR","YEM","GNB",
"COD","MOZ","SLE","BFA","ERI","MLI","BDI","SSD","TCD","CAF",
"NER", "CIV")

ind_idh = sapply(cod_nom,FUN = function(u){return(which(x$codigo_
    nombres==u))})
idh[ind_idh]=1
x[,24]=idh
names(x)[24]= "IndDesaHumano"
```

Los datos faltantes de nuestra matriz de predictores, en este caso, los reemplazaremos por la mediana muestral. A continuación, aplicaremos los bosques aleatorios (con los parámetros del análisis principal con parámetro m_1) para una variable que hemos denotado como `idh` del tipo factor.

```
x = x[(x$region==0), ]
x$IndDesaHumano = as.factor(x$IndDesaHumano)

dataframe = x[,-c(1,2,23)]
str(dataframe)

modelo_clas = randomForest(x = na.roughfix(dataframe[,-21]), y =
    dataframe[,21], importance = TRUE, proximity = TRUE, ntree = 100,
    mtry = 7)
```

En el Cuadro 4.9 podemos observar el **nivel de precisión** que obtenemos con dicho modelo.

Real/ Pred	0	1	**Error**
0	136	8	0.06
1	11	61	0.15

Cuadro 4.9: Matriz de confusión resultante de aplicar los bosques aleatorios para el problema de clasificación definido en el texto.

Luego, nuestro modelo clasifica erróneamente el 6 % de los países con desarrollo humano alto o muy alto: en cambio, este error aumenta hasta el 15 % para los países con desarrollo medio o bajo. En los Cuadros 4.10 y 4.11 podemos ver qué países han sido mal clasificados por nuestro modelo.

	País
1	Belice
2	Botswana
3	Egipto, República Árabe de
4	Gabón
5	Somalia
6	Ucrania
7	Uzbekistán

Cuadro 4.10: Países con IDH alto o muy alto, para los que el modelo señala que deberían ser bajo o medio.

	País
1	Cabo Verde
2	Djibouti
3	Guinea Ecuatorial
4	Guyana
5	Iraq
6	Islas Marshall
7	Namibia
8	Ribera Occidental y Gaza
9	El Salvador
10	Tayikistán
11	Vietnam

Cuadro 4.11: Países con IDH bajo o medio, para los que el modelo señala que deberían ser alto o muy alto.

Para terminar, en la Figura 4.14 podemos ver cuál ha sido el **ranking de importancia** de nuestras variables predictoras, sobre nuestra variable respuesta binaria.

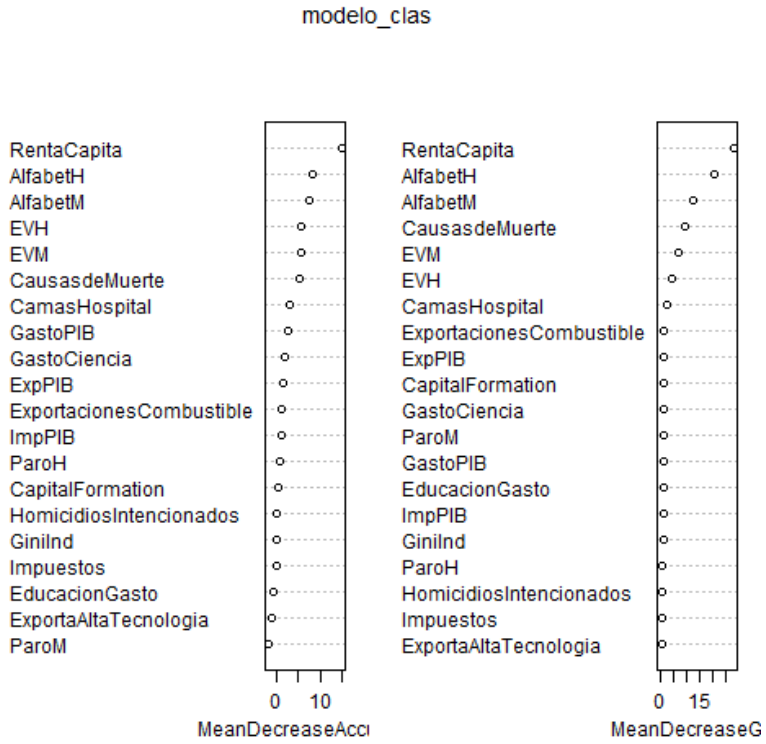

Figura 4.14: Ranking de importancia de las variables predictoras para el problema de clasificación.

Como podemos observar, las variables que anteriormente hemos indicado que influían en el IDH ocupan las primeras posiciones en el ranking. Este hecho da muestras de la **consistencia** del modelo. Por otra parte, es sorprendente la posición **tan baja** que ocupa el índice de Gini en el ranking, lo que indica que el indicador IDH no tiene en cuenta para su medición el grado de desigualdad económica, lo cuál en estudios como el que hemos pretendido en este trabajo presenta ciertas limitaciones.

Nota 4.15. Existen otro tipo de índices como el ***índice de desarrollo ajustado por desigualdad*** (IDHD), que pretender subsanar las limitaciones anteriores. De hecho, este indicador también aparece en [9], pero la clasificación que se lleva a cabo en el estudio y que hemos usado para el problema de clasificación se centra en el IDH.

Capítulo 5

Conclusiones

En cualquier curso elemental de minería de datos se suelen estudiar los árboles de decisión, sobre todo, los llamados *Clasification and Regression Trees* (CART). En cambio, los bosques aleatorios, aunque se basan en los árboles de decisión, son menos conocidos, especialmente en la faceta de aprendizaje no supervisado, que revisamos también en este trabajo.

Como hemos explicado en el trabajo, los bosques aleatorios mediante el promedio de resultados de los diferentes árboles y la aleatoriedad en la elección de variables para determinar cada *split*, permiten disminuir la varianza del método, dando lugar a una mayor estabilidad de nuestras predicciones. Además, este método de aprendizaje permite trabajar con datos de diferentes formatos (categóricos, numéricos...), incluso con datos faltantes, mediante el *rellenado* de datos usando los propios bosques aleatorios o a través del uso de los denominados *surrogate splits*. Otra de las ventajas inherentes en la propia construcción del método, es la existencia de observaciones *out of bag*, cuya función es **esencial** para determinar el error y las predicciones del método. Además, también permiten que prescindamos de métodos de validación cruzada, con la finalidad de estudiar el error. También cabe destacar que los bosques aleatorios, nos proporcionan un ranking de importancia de las variables explicativas sobre la variable respuesta, este nos puede ser de gran ayuda en el análisis de los datos y en la extracción de conocimiento, ya que el método también permite obtener la función marginal empírica de cada variable. Finalmente, respecto a las ventajas del método, destacaría: los pocos parámetros que hay que ajustar para trabajar con él, que permite el aumento en el número de variables explicativas y que el aumento en el número de árboles no produce sobreajuste. Por otro lado, permite ser empleado tanto en aprendizaje supervisado, como en no supervisado.

Si nos centramos en las desventajas, es claro que reducen considerablemente la facilidad de intepretación de los CART; además, la implementación en bases de datos con un gran número de observaciones conlleva un coste computacional y de memoria muy elevado, que no siempre es asumible. Por otra parte, hereda algunas de las características de los árboles de decisión como es: la no detección de relaciones lineales entre las variables.

Respecto a la parte de aplicación del trabajo, hemos aplicado los bosques aleatorios a una base de datos, creada *ad hoc* para el mismo, en la que intentamos estudiar un problema de carácter social como son: las desigualdades económicas en los diferentes países del mundo. Para llevar a cabo el estudio, nos hemos centrado en el índice de Gini como medida que identifica el grado de desigualdad económica existente en cada país.

Posteriormente, en los resultados del estudio, pudimos comprobar que las variales explicativas más influyentes sobre el índice de Gini, según nuestros modelos, son: la **violencia** existente materializada en homicidios intencionados y el nivel de desarrollo **sanitario**, el cual se puede explicar a su vez a través de la esperanza de vida de la población, del número de camas de hospital por habitante o incluso por la cantidad de muertes por causas maternales, prenatales o por desnutrición.

Por último, algunos de los aspectos con los que se podría continuar el trabajo en el futuro podrían ser los siguientes:

1. Dada la **complejidad** del problema y las **limitaciones** que nos hemos encontrado en lo referente a la calidad y la obtención de los datos, podría ser beneficioso aumentar el número de variables explicativas en nuestra base de datos, con la finalidad de obtener una mayor precisión en nuestros modelos.

2. Respecto a la detección de *outliers*, en [11] se explica como podemos obtener la **función de distribución** para la predicción de **cada observación**, es decir, $F\big(Y \leq y | X = x\big)$. De esta manera, el método nos podría proporcionar **intervalos de confianza** para cada observación, con lo que determinando un **nivel de confianza** α, se podrían determinar las observaciones atípicas.

3. En referencia al uso de paquetes de `R`, podría ser beneficioso el uso de la librería `rpart` desarrollada en [15]. En ella se implementan los árboles de decisión junto con los *surrogate splits*, los cuales permiten que no tengamos que rellenar los datos faltantes. En caso de que una observación tenga un dato faltante para la variable de separación, los árboles usarán un criterio de separación auxiliar.

4. Finalmente, en [13] se presenta un método que intenta mejorar la precisión de los bosques aleatorios, mediante la ponderación de los árboles de decisión, favoreciendo a los que presentan una mayor precisión. Por otra parte, una idea estrechamente relacionada con la anterior se presenta en [14], donde se propone una construcción guiada de los árboles de decisión, de modo que cada árbol complemente tanto como sea posible los árboles existentes en el conjunto.

Apéndice A

Construcción de la base de datos

Aquí adjuntamos el código para la construcción de la base de datos

```
require(readr)
### Cargamos todas las tablas

gastoID = read_csv("gastoID.csv")
AlfabeHombres = read_csv("AlfabeHombres.csv")
AlfabeMujeres = read_csv("AlfabeMujeres.csv")
CamasHospi = read_csv("CamasHospi.csv")
CausasMuerte = read_csv("CausasMuerte.csv")
EsperanzaVidaHombres = read_csv("EsperanzaVidaHombres.csv")
EsperanzaVidaMujeres = read_csv("EsperanzaVidaMujeres.csv")
ExpCombustible = read_csv("ExpCombustible.csv")
ExportacionesAltaTec = read_csv("ExportacionesAltaTec.csv")
ExportacionesPIB = read_csv("ExportacionesPIB.csv")
GastoEducacion = read_csv("GastoEducacion.csv")
Importaciones = read_csv("Importaciones.csv")
ParoHombres = read_csv("ParoHombres.csv")
ParoMujeres = read_csv("ParoMujeres.csv")
gastoPIB = read_csv("gastoPIB.csv")
Gini_Index = read_csv("Gini_Index.csv")
GrossCapitalFormation = read_csv("GrossCapitalFormation.csv")
income_x_person = read_csv("income_x_person.csv")
homicidiosintencionales = read_csv("homicidiosintencionales.csv")
Impuestos = read_csv("tax.csv")


names(gastoID)=NULL
M = as.matrix(gastoID[,2:dim(gastoID)[2]])
M[is.na(M)]=0
nombres = M[,1]
codigo_nombres = M[,2]
GastoCiencia = apply(M,MARGIN = 1, function(x) {
  y = as.numeric(x[45:61]);
  ind = which(y!=0);
  if (length(ind)!=0){
    return(y[ind[length(ind)]])
  }
  else {
    return(0)
  }
```

```
})

############
names(AlfabeHombres)=NULL
M = as.matrix(AlfabeHombres[,2:dim(AlfabeHombres)[2]])
M[is.na(M)]=0
AlfabetH = apply(M,MARGIN = 1, function(x) {
  y = as.numeric(x[44:60]);
  ind = which(y!=0);
  if (length(ind)!=0){
    return(y[ind[length(ind)]])
  }
  else {
    return(0)
  }
})

#########
names(AlfabeMujeres)=NULL
M = as.matrix(AlfabeMujeres[,2:dim(AlfabeMujeres)[2]])
M[is.na(M)]=0
AlfabetM = apply(M,MARGIN = 1, function(x) {
  y = as.numeric(x[44:60]);
  ind = which(y!=0);
  if (length(ind)!=0){
    return(y[ind[length(ind)]])
  }
  else {
    return(0)
  }
})

names(CamasHospi)=NULL
M = as.matrix(CamasHospi[,2:dim(CamasHospi)[2]])
M[is.na(M)]=0
CamasHospital = apply(M,MARGIN = 1, function(x) {
  y = as.numeric(x[44:60]);
  ind = which(y!=0);
  if (length(ind)!=0){
    return(y[ind[length(ind)]])
  }
  else {
    return(0)
  }
})

names(CausasMuerte)=NULL
M = as.matrix(CausasMuerte[,2:dim(CausasMuerte)[2]])
M[is.na(M)]=0
CausasdeMuerte = apply(M,MARGIN = 1, function(x) {
  y = as.numeric(x[44:60]);
  ind = which(y!=0);
  if (length(ind)!=0){
    return(y[ind[length(ind)]])
  }
```

```
95        else {
96          return(0)
97        }
98      })
99
100     names(EsperanzaVidaHombres)=NULL
101     M = as.matrix(EsperanzaVidaHombres[,2:dim(EsperanzaVidaHombres)[2]])
102     M[is.na(M)]=0
103     EVH = apply(M,MARGIN = 1, function(x) {
104       y = as.numeric(x[44:60]);
105       ind = which(y!=0);
106       if (length(ind)!=0){
107         return(y[ind[length(ind)]])
108       }
109       else {
110         return(0)
111       }
112     })
113
114     names(EsperanzaVidaMujeres)=NULL
115     M = as.matrix(EsperanzaVidaMujeres[,2:dim(EsperanzaVidaMujeres)[2]])
116     M[is.na(M)]=0
117     EVM = apply(M,MARGIN = 1, function(x) {
118       y = as.numeric(x[44:60]);
119       ind = which(y!=0);
120       if (length(ind)!=0){
121         return(y[ind[length(ind)]])
122       }
123       else {
124         return(0)
125       }
126     })
127
128     names(ExpCombustible)=NULL
129     M = as.matrix(ExpCombustible[,2:dim(ExpCombustible)[2]])
130     M[is.na(M)]=0
131     ExportacionesCombustible = apply(M,MARGIN = 1, function(x) {
132       y = as.numeric(x[44:60]);
133       ind = which(y!=0);
134       if (length(ind)!=0){
135         return(y[ind[length(ind)]])
136       }
137       else {
138         return(0)
139       }
140     })
141
142     names(ExportacionesAltaTec)=NULL
143     M = as.matrix(ExportacionesAltaTec[,2:dim(ExportacionesAltaTec)[2]])
144     M[is.na(M)]=0
145     ExportaAltaTecnologia = apply(M,MARGIN = 1, function(x) {
146       y = as.numeric(x[44:60]);
147       ind = which(y!=0);
148       if (length(ind)!=0){
149         return(y[ind[length(ind)]])
```

```
      }
      else {
        return(0)
      }
    })

    names(ExportacionesPIB)=NULL
    M = as.matrix(ExportacionesPIB[,2:dim(ExportacionesPIB)[2]])
    M[is.na(M)]=0
    ExpPIB = apply(M,MARGIN = 1, function(x) {
      y = as.numeric(x[44:60]);
      ind = which(y!=0);
      if (length(ind)!=0){
        return(y[ind[length(ind)]])
      }
      else {
        return(0)
      }
    })

    names(GastoEducacion)=NULL
    M = as.matrix(GastoEducacion[,2:dim(GastoEducacion)[2]])
    M[is.na(M)]=0
    EducacionGasto = apply(M,MARGIN = 1, function(x) {
      y = as.numeric(x[44:60]);
      ind = which(y!=0);
      if (length(ind)!=0){
        return(y[ind[length(ind)]])
      }
      else {
        return(0)
      }
    })

    names(Importaciones)=NULL
    M = as.matrix(Importaciones[,2:dim(Importaciones)[2]])
    M[is.na(M)]=0
    ImpPIB = apply(M,MARGIN = 1, function(x) {
      y = as.numeric(x[44:60]);
      ind = which(y!=0);
      if (length(ind)!=0){
        return(y[ind[length(ind)]])
      }
      else {
        return(0)
      }
    })

    names(ParoHombres)=NULL
    M = as.matrix(ParoHombres[,2:dim(ParoHombres)[2]])
    M[is.na(M)]=0
    ParoH = apply(M,MARGIN = 1, function(x) {
      y = as.numeric(x[44:60]);
      ind = which(y!=0);
      if (length(ind)!=0){
```

```
205           return(y[ind[length(ind)]])
206         }
207         else {
208           return(0)
209         }
210       })
211
212       names(ParoMujeres)=NULL
213       M = as.matrix(ParoMujeres[,2:dim(ParoMujeres)[2]])
214       M[is.na(M)]=0
215       ParoM = apply(M,MARGIN = 1, function(x) {
216         y = as.numeric(x[44:60]);
217         ind = which(y!=0);
218         if (length(ind)!=0){
219           return(y[ind[length(ind)]])
220         }
221         else {
222           return(0)
223         }
224       })
225
226       names(gastoPIB)=NULL
227       M = as.matrix(gastoPIB[,2:dim(gastoPIB)[2]])
228       M[is.na(M)]=0
229       GastoPIB = apply(M,MARGIN = 1, function(x) {
230         y = as.numeric(x[44:60]);
231         ind = which(y!=0);
232         if (length(ind)!=0){
233           return(y[ind[length(ind)]])
234         }
235         else {
236           return(0)
237         }
238       })
239
240       names( Gini_Index )=NULL
241       M = as.matrix( Gini_Index[,2:dim(Gini_Index )[2]])
242       M[is.na(M)]=0
243       GiniInd = apply(M,MARGIN = 1, function(x) {
244         y = as.numeric(x[44:60]);
245         ind = which(y!=0);
246         if (length(ind)!=0){
247           return(y[ind[length(ind)]])
248         }
249         else {
250           return(0)
251         }
252       })
253
254       names(GrossCapitalFormation )=NULL
255       M = as.matrix( GrossCapitalFormation[,2:dim( GrossCapitalFormation )
            [2]])
256       M[is.na(M)]=0
257       CapitalFormation = apply(M,MARGIN = 1, function(x) {
258         y = as.numeric(x[44:60]);
```

```
259       ind = which(y!=0);
260       if (length(ind)!=0){
261         return(y[ind[length(ind)]])
262       }
263       else {
264         return(0)
265       }
266     })
267
268     names( income_x_person )=NULL
269     M = as.matrix(  income_x_person[,2:dim(income_x_person  )[2]])
270     M[is.na(M)]=0
271     RentaCapita = apply(M,MARGIN = 1, function(x) {
272       y = as.numeric(x[44:60]);
273       ind = which(y!=0);
274       if (length(ind)!=0){
275         return(y[ind[length(ind)]])
276       }
277       else {
278         return(0)
279       }
280     })
281
282     names( homicidiosintencionales )=NULL
283     M = as.matrix(  homicidiosintencionales[,2:dim(homicidiosintencionales
          )[2]])
284     M[is.na(M)]=0
285     HomicidiosIntencionados = apply(M,MARGIN = 1, function(x) {
286       y = as.numeric(x[44:60]);
287       ind = which(y!=0);
288       if (length(ind)!=0){
289         return(y[ind[length(ind)]])
290       }
291       else {
292         return(0)
293       }
294     })
295
296     names( Impuestos )=NULL
297     M = as.matrix(  Impuestos[,2:dim(Impuestos  )[2]])
298     M[is.na(M)]=0
299     Impuestos = apply(M,MARGIN = 1, function(x) {
300       y = as.numeric(x[44:60]);
301       ind = which(y!=0);
302       if (length(ind)!=0){
303         return(y[ind[length(ind)]])
304       }
305       else {
306         return(0)
307       }
308     })
309
310     database_matrix = cbind(nombres, codigo_nombres,Impuestos, GastoPIB ,
          GastoCiencia, EducacionGasto, ExpPIB ,ExportaAltaTecnologia,
          ExportacionesCombustible, ImpPIB,CapitalFormation, AlfabetH,
```

```
        AlfabetM, CamasHospital, CausasdeMuerte, EVH, EVM, ParoH, ParoM,
        HomicidiosIntencionados,RentaCapita, GiniInd)
311 database = as.data.frame(database_matrix)
312 names(database)
313 head(database)
314 str(database)
315
316 for (i in 3:(dim(database)[2])) {
317   database[ ,i] = as.numeric(database[ ,i])
318
319 }
320 dim(database)
321 require(readr)
322
323 newdatabase = database
324 newdatabase[newdatabase==0]=NA
325 head(newdatabase)
326
327 write.csv(newdatabase, file = "BBDDTFM.csv", row.names = F)
```

Nota A.1. Posteriormente, realizamos unas modificaciones ha dicha base de datos, ya que añadimos una columna extra, llamada *Region*, donde identificamos con $Region = 1$ las observaciones que no fueran un país.

Bibliografía

[1] Trevor Hastie, Robert Tibshirani, Jerome Friedman. *The Elements of Statistical Learning: Data mining, inference, and prediction*, Second Edition. Springer Series in Statistics, Springer, New York (2009)

[2] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. *An Introduction to Statistical Learning with Applications in R*. Springer Series in Statistics, Springer, New York (2015).

[3] Robin GenuerJean, Michel Poggi. *Random Forests with R*. Springer (2020).

[4] Leo Breiman. *Using Random Forest 4.0*. Berkeley (2001). www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

[5] Leo Breiman. *Random Forests*. Machine Learning 45, 5–32 (2001).

[6] Leo Breiman, Jerome Friedman, Richard A. Olshen, Charles J. Stone. *Classification and regression trees* . The Wadsworth statistics / probability series, CRC (1984).

[7] Andy Liaw, Matthew Wiener (2002). *Classification and Regression by randomForest*. R News 2(3), 18–22. https://CRAN.R-project.org/doc/Rnews/

[8] *World Bank: the Development Data Group*. data.worldbank.org

[9] Programa de las Naciones Unidas para el Desarrollo. *Informe sobre Desarrollo Humano 2019*. http://hdr.undp.org/sites/default/files/hdr_2019_overview_-_spanish.pdf

[10] Wikipedia. *Índice de desarrollo humano*. https://es.wikipedia.org/wiki/%C3%8Dndice_de_desarrollo_humano

[11] N. Meinshausen. *Quantile regression forests*. Journal of Machine Learning Research, 7:983–999, (2006).

[12] Leonard Kaufman, Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley Series in Probability and Statistics, Wiley, (2005).

[13] S.J. Winham, R.R. Freimuth, J.M. Biernacka. *A weighted random forests approach to improve predictive performance*. Statistical Analysis and Data Mining: The ASA Data Science Journal, 6:496–505, (2013).

[14] S. Bernard, S. Adam, L. Heutte. *Dynamic Random Forests*. Pattern Recognition Letters, 33:1580–1586, (2012).

[15] Terry Therneau and Beth Atkinson (2019). rpart: *Recursive Partitioning and Regression Trees*. R package version 4.1-15. https://CRAN.R-project.org/package=rpart