Arana, Carlos

Working Paper

Modelos de aprendizaje automoático mediante árboles de decisión

Serie Documentos de Trabajo, No. 778

Provided in Cooperation with:
University of CEMA, Buenos Aires

Suggested Citation: Arana, Carlos (2021) : Modelos de aprendizaje automoático mediante árboles de decisión, Serie Documentos de Trabajo, No. 778, Universidad del Centro de Estudios Macroeconómicos de Argentina (UCEMA), Buenos Aires

This Version is available at:
https://hdl.handle.net/10419/238403

UNIVERSIDAD DEL CEMA
Buenos Aires
Argentina

Serie
DOCUMENTOS DE TRABAJO

Área: Negocios

MODELOS DE APRENDIZAJE AUTOMÁTICO MEDIANTE ÁRBOLES DE DECISIÓN

Carlos Arana

Febrero 2021
Nro. 778

www.cema.edu.ar/publicaciones/doc_trabajo.html
UCEMA: Av. Córdoba 374, C1054AAP Buenos Aires, Argentina
ISSN 1668-4575 (impreso), ISSN 1668-4583 (en línea)
Editor: Jorge M. Streb; asistente editorial: Valeria Dowding jae@cema.edu.ar

Modelos de Aprendizaje Automático Mediante Árboles de Decisión

Carlos Arana *

Febrero 2021

Resumen

Los modelos de aprendizaje automatico (*machine learning*) supervisados de clasificación mediante particiones binarias recursivas, tambien llamados "árboles de decisión"se encuentran entre los más utilizados en la ciencia de datos, no sólo por su interpretabilidad y su performance sino tambien por ser la base de los modelos más potentes utilizados en la actualidad: los ensambles de árboles de decisión. Al seguir siendo considerados los modelos de clasificación por excelencia, es que en este trabajo presentaré sus fundamentos, sus elementos constitutivos y los procedimientos involucrados para su implementación, puesta en marcha y medición de performance predictiva.

Palabras Claves : Inteligencia Artificial, Aprendizaje Automático, Ciencia de Datos, Clasificadores, Árboles de Decisión.

1. Introducción

Los clasificadores mediante árboles de decisión son modelos no paramétricos de aprendizaje supervisado, cuyo objetivo es poder predecir a qué clase pertenece un caso del que

*Ingeniero Industrial (UBA) y Doctor en Ingeniería Industrial (cand). Es profesor de las materias 'Métodos Cuantitativos' y 'Ciencia de Datos en Negocios' en la Maestría en Dirección de Empresas (MADE) de UCEMA. Es director del posgrado en 'Ciencia de Datos y Big Data' de la Facultad de Ingeniería de la Universidad de Buenos Aires. A su vez se desempeña como consultor especializado en Inteligencia Artificial y sus aplicaciones en negocios e industria

conocemos uno o más atributos o mediciones. El algoritmo que se utiliza para armar los árboles se denomina "partición binaria recursiva", pues en cada paso del entrenamiento del modelo el algoritmo realiza sucesivas divisiones o particiones de un subconjunto de datos a partir la aplicación de una decisión asociada a una de las variables, por ende separando a esos datos en dos nuevos subconjuntos (de allí lo de partición binaria). Siguiendo con este proceso en forma recursiva hasta un cierto punto previamente estipulado en el que el proceso de bifurcación se detiene, obtendremos finalmente el clasificador por árbol de decisión. Cada nuevo dato, del que conoceremos el valor de sus atributos, recorrerá las sucesivas ramificaciones del árbol, a partir de las reglas y decisiones generadas mediante el proceso recién descripto.

Aunque mediante este tipo de algoritmos de árbol de decisión podemos generar modelos predictivos tanto para variables objetivo de tipo cuantitativa (regresión) como de tipo cualitativa o categórica (clasificación), en este trabajo nos concentraremos en el estudio y desarrollo de estos últimos, pues a partir de su comprensión es muy simple encarar el estudio de los primeros, llamados “árboles de regresión”.

Estos tipos de modelos de aprendizaje supervisado basados en árboles de decisión se encuentran entre los modelos más utilizados y de mejor performance en la actualidad, por lo que justifica que desarrollemos en detalle los procesos involucrados en su diseño metodológico, armado, entrenamiento y evaluación (Figura 1).

Los primeros desarrollos sobre estos clasificadores fueron encarados por el Profesor e investigador de UC Berkeley, Leo Breiman. Él junto al reconocido estadístico Jerome Friedman son a su vez los desarrolladores y creadores de los ensambles de arboles de decisión, modelos compuestos que mejoran la performance de los arboles de decisión.

2. Clasificadores a partir de particiones

Comencemos definiendo lo que es un “clasificador”. Definiremos “clasificador” al algoritmo que en forma sistemática nos permite saber a qué clase asignar cada uno de los casos que se presenten. Sea $(x_1, x_2, \ldots, x_N)$ el vector de mediciones correspondiente a un caso, perteneciendo a un espacio de mediciones X, que contiene a todas los posibles mediciones.

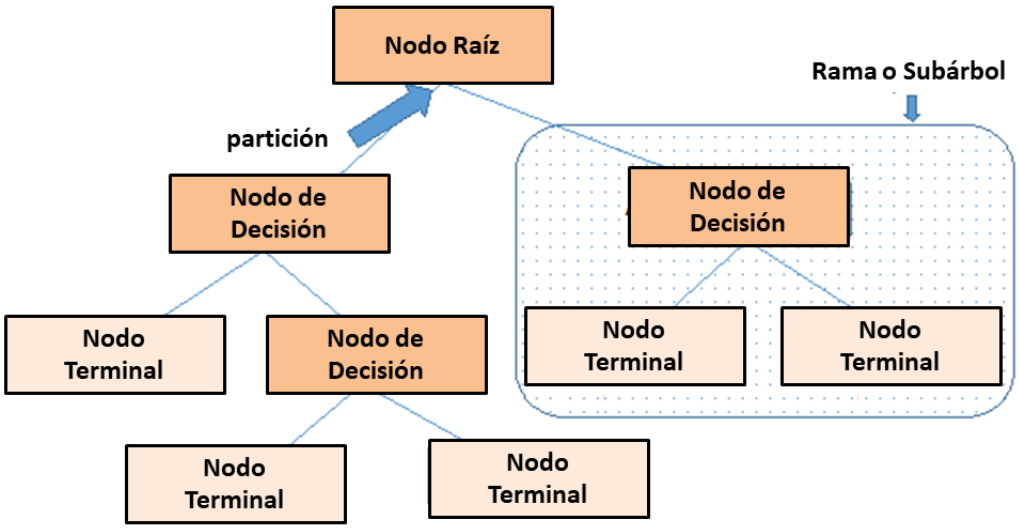

Figura 1: Estructura de un árbol de decisión

A su vez llamaremos C al conjunto de las J clases posibles, siendo la salida del modelo, que llamamos clase, $y \in \mathcal{C} = \{1, 2, \ldots, J\}$. Nuestra regla de clasificación, o clasificador, deberá asignar cada caso a una clase del espacio C en función de sus mediciones. Para desarrollar nuestro clasificador deberemos contar con una muestra, para que a partir de ella, y mediante un método inductivo, podamos generar las reglas que serán la base del modelo. Esta muestra consistirá de N casos con sus correspondientes atributos o mediciones, tanto de las X como de la clase j a la que corresponda. La denominaremos "muestra.º "set de aprendizaje", y la representaremos con la letra $\mathcal{L}$.

2.1. Estimando la exactitud del clasificador

Dado un clasificador $h(x)$ definido en X y sus correspondientes valores en C, llamaremos $R^*(h)$ a su tasa de error de clasificación verdadera. Esta tasa se calcularía a partir de evaluar la cantidad de valores de X a los que el clasificador le asigna una clase incorrecta, sobre el total de valores de X. Pero ¿cómo podríamos calcular esta tasa de error "Verdadera", si por definición los casos con lo que contamos son muestras del conjunto X, el universo, cuyo tamaño es mucho mayor? La forma natural de poder estimar este error es que conozcamos el valor de la clase de la totalidad de los casos del universo, o si no, de una muestra de un tamaño similar a este. Es bien sabido que en la vida real no contaremos con todos los datos del universo, si no con una muestra, a la que hemos llamado $\mathcal{L}$. Es con esa muestra original con la que deberemos estimar el valor de $R^*(h)$, llamándose a este tipo de estimaciones

muestrales “estimadores internos”.

$$R^*(h) = \Pr[h(x) \neq Y] \tag{1}$$

El primer caso de estimación interno que analizaremos es el “estimador de sustitución” o “error de entrenamiento”. Para su determinación lo que haremos es correr todos los casos de $\mathcal{L}$ en el clasificador. La proporción de casos en los que el clasificador arrojó una clase distinta a la real es el *estimador de sustitución o entrenamiento.*

La fórmula para calcular este estimador del error es:

$$R(h) = \frac{1}{N} \sum_{n=1}^{N} 1[h(x_n) \neq j_n]. \tag{2}$$

dónde j_n es la clase del caso n.

Aquí se nos presenta un serio problema: para calcular el estimador de sustitución o error de entrenamiento, hemos utilizado los mismos datos con los cuales se construyó el clasificador, o sea $\mathcal{L}$. Al proceso de construcción del algoritmo también se lo denomina “entrenamiento”, de allí el nombre de “error de entrenamiento”. Es lógico que esta estimación no nos esté dando un panorama muy alentador respecto al error que comete el clasificador, pues el proceso de construcción/entrenamiento del clasificador se basa en minimizar este mismo error.

El método más adecuado para estimar el verdadero error de clasificación es el de estimación por ’conjunto de prueba’. Para determinar este estimador, llamado “error de prueba”, se divide al set de aprendizaje $\mathcal{L}$ en dos subsets, $\mathcal{L}_1$ y $\mathcal{L}_2$. El clasificador se entrena utilizando sólo los datos de $\mathcal{L}_1$, para luego verificar la proporción de casos mal clasificados por el clasificador ahora sobre los casos del conjunto $\mathcal{L}_2$. Los N_2 casos incluidos en el subset $\mathcal{L}_2$ no han sido utilizados para la construcción del clasificador, por lo que nunca fueron evaluados para determinar ningún error durante este proceso de entrenamiento. La fórmula para calcular el error de prueba (ep) es :

$$R^{(ep)}(h) = \frac{1}{N_n} \sum_{(x_n, j_n) \in \mathcal{L}_2} 1[h(x_n) \neq j_n]. \tag{3}$$

Lo que debemos asegurarnos es que $\mathcal{L}_1$ y $\mathcal{L}_2$ sean *IID*, o sea independientes e idénticamente distribuidos. ¿Qué significad esto? Que la selección de los casos especificados que

componen una de los subconjuntos no influye en ningún sentido sobre la selección de los casos seleccionados en el otro subconjunto. A su vez se exige que ambas provengan de la misma distribución de probabilidad, o sea que hayan sido seleccionados del universo de casos utilizando el mismo procedimiento estadístico.

Al ser $\mathcal{L}_1$ y $\mathcal{L}_2$ dos subconjuntos que se forman fraccionando $\mathcal{L}$, la suma de los casos que los componen debe ser igual a la cantidad de casos en $\mathcal{L}$. Las proporciones para fraccionar a $\mathcal{L}$ más utilizadas para conformar $\mathcal{L}_1$ son $^2/_3$, $^3/_4$, $^8/_{10}$ o $^9/_{10}$. Al ser siempre $\mathcal{L}_2$ el complemento de $\mathcal{L}_1$ evaluado sobre $\mathcal{L}$, tendremos que $\mathcal{L}_2 = \mathcal{L} - \mathcal{L}_1$. Por ejemplo, si $\mathcal{L}_1$ se conforma como $9/10\mathcal{L}_1$ entonces $\mathcal{L}_2$ será $1/10\mathcal{L}$.

El problema al determinar los estimadores del error de clasificación, en este caso el error de prueba, es que se pueden llegar a estar utilizando pocos casos, por lo que la estimación reduce su eficiencia (en términos estadísticos implica que es un estimador de mayor varianza).

Para sortear este último inconveniente relacionado con la eficiencia del estimador del error de clasificación mediante "error de prueba" es que se utiliza un tercer método e estimación llamada "Validación Cruzada de k Dobleces"(*k-fold Cross Validation*). Para su determinación primero se divide a $\mathcal{L}_2$ en k subconjuntos de un tamaño similar. Llamaremos a estos subconjuntos $\mathcal{L}_1$, $\mathcal{L}_2$, ..., $\mathcal{L}_k$. Se construye/entrena un clasificador, utilizando el mismo mecanismo utilizado anteriormente con todos los datos $\mathcal{L}$ y con los datos de $\mathcal{L}_1$ en del método de conjunto de prueba, para cada uno de los k conjuntos $\mathcal{L}_1$ a $\mathcal{L}_k$. A cada uno de estos clasificadores los denominados $h^{(k)}(x)$.

Como no hemos utilizado ninguno de los subconjuntos $\mathcal{L}_k$ para la construcción de los respectivos clasificadores podremos ahora utilizarlos para calcular, en cada clasificador $h^{(k)}(x)$, el estimador por subconjunto de prueba:

$$R^{(ep)}(h^{(k)}) = \frac{1}{N_n} \sum_{k=1}^{K} 1(h^{(k)}(x_n) \neq j_n). \tag{4}$$

Contando con las k estimaciones por subconjuntos de prueba para cada clasificador $d^v(x)$ podremos ahora calcular el tercer estimador interno de la tasa de error verdadero del clasificador, llamado "*error por validación cruzada de k dobleces*", de la siguiente manera:

$$R^{(CV)}(h) = \frac{1}{K}\sum_{1}^{K} R^{(ep)}(h^{(k)}). \tag{5}$$

que no es otra cosa que el promedio de los errores de prueba de cada uno de los clasificadores construidos a partir de los k conjuntos de entrenamiento de tamaño $(\mathcal{L} - \mathcal{L}_k)$ evaluados sobre los k conjuntos de prueba $(\mathcal{L}_k)$ respectivos.

Un caso especial de validación cruzada de k dobleces es la validación cruzada de N dobleces, donde cada subconjunto de prueba tiene un tamaño $N - 1$ y se evalúa el error del clasficador construido a partir de esos casos en un solo caso aislado (o sea en un set de pruaba de tamaño unitario). A este tipo de estimación interno cuyo subconjunto de prueba es de tamaño unitario se lo denomina estimador de "Validación Cruzada Dejando Uno Afuera" o LOOCV, por sus siglas en inglés de *Leave One Out Cross Validation*.

2.2. Clasificadores con estructura de árbol binario

Los clasificadores con estructura de árbol binario se construyen a partir de una sucesión recursiva de divisiones de subconjuntos en dos nuevos subconjuntos descendentes, comenzando con el conjunto inicial, que denominaremos X.

Como podemos observar en la figura 2 los subconjuntos X_2 y X_3 son disjuntos, o sea que $X = X_2 \cup X_3$. Análogamente también son disjuntos los conjuntos X_4 y X_5, pues $X_4 \cup X_5 = X_2$. Los subconjuntos que no se vuelven a dividir son llamados conjuntos terminales. A los subconjuntos terminales los representamos con un rectángulo y a los no terminales (o de decisión) con un círculo.

Los subconjuntos terminales forman una partición de X.Las subconjuntos no terminales son formadas por divisiones que responden a condiciones en alguna de las coordenadas $x = (x_1, x_2, \ldots, x_p)$. pPor ejemplo la división 1, conforma los subconjuntos disjuntos $X2$ y $X3$ a partir de X fruto d la regla/decisión asociada a esta división.

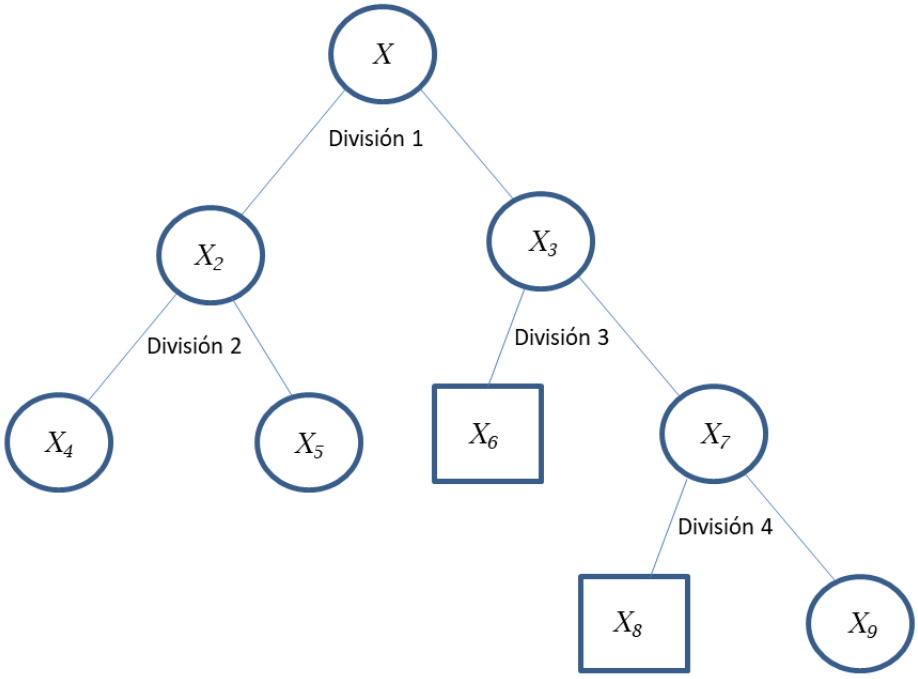

Figura 2: Clasificador por Árbol de Decisión de seis divisiones

El clasificador por árbol binario predice la clase para un caso cuyo vector de medidas es x de la siguiente manera: de las condiciones de la primera división se determina si x pasa a formar parte de los subconjuntos X_2 o X_3. Si x fuera al subconjunto X_3, x se sometería a las condiciones de la división 3 y se determinaría si ahora formará parte del subconjunto X_6 o del X_7.

Si x siguiera el recorrido por las sucesivas divisiones, y sus condiciones asociadas, indefectiblemente terminaría su travesía formando parte de un subconjunto terminal.

El proceso completo de construcción en clasificador de árbol binario se compone de tres elementos:

1. La selección de las condiciones asociado a cada división a partir de un conjunto de condiciones posibles
2. La decisión de cuando seguir dividiendo a cada nodo o de que pase a ser un nodo terminal

3. La asignación de una clase específica a cada uno de los nodos terminales.

El problema crucial se presenta a la hora de determinar las condiciones asociadas a las divisiones y de declarar que un nodo pasa a ser un nodo terminal a partir de los casos que contiene el set de entrenamiento $\mathcal{L}$. El tercer punto, el de la asignación de una clase específica, es mucho más sencillo e intuitivo: se evaluarán las clases de los casos que compone cada uno de los subconjuntos o nodos terminales y se tomará la clase mayoritaria como asignación de clase para todos los casos que terminen su recorrido por el árbol en ese nodo.

2.3. Construcción del clasificador de árbol binario

Como habíamos mencionado en el punto anterior los mayores desafíos a la hora de construir un clasificador por árbol binario es la selección de los puntos en las x en los que se localizarán las divisiones, que como sabemos responderá a una condición sobre una variable también a determinar, y que serán las responsables de la generación de los sucesivos subconjuntos que van a ir conformando el clasificador. La idea fundamental asociada a este proceso tan crucial es que la "pureza" de los dos subconjuntos generados a partir de una división sea mayor a la del conjunto padre, del cuál ambos se generaron.

Por ejemplo, si tuviéramos dos clases y la proporción en un nodo t_1 fuera $(p_1, p_2) = (1/2, 1/2)$, una buena división sería aquella que generara dos nodos hijos tal que todos los casos que fueran de la clase 1 pasaran a formar parte del nodo hijo 1 y todos los de la clase 2 fueran parte del nodo hijo 2.

Desarrollemos ahora un poco este concepto de "pureza" de nodos y conjunto de nodos, y veamos como asociarlo al proceso de determinación de las divisiones y los correspondientes valores del atributo que las generan. Los pasos de este proceso son:

1. Definir a $p(j/t)$ como la proporción de casos del nodo t cuya clase es j. Siendo $(1, 2, \ldots.., J)$ los valores posibles que puede tomar la clase, entonces

$$p(1/t) + p(2/t) + \ldots. + p(J/t) = 1 \tag{6}$$

2. Definir a $i(t)$ como una medida de la impureza del nodo t, que tome valores no negativos, siendo esta una función de los $p(j/t)$ y cuyo valor 0 (su mínimo) se dará cuando la todos los casos sean de una misma clase y su valor será 1 (su máximo) cuando la proporción de casos de las clases sean iguales entre sí. En nuestro ejemplo nos quedaría:

$$i(t) = \begin{cases} 0 & si \quad p(j=1/t) = 0 \; y \; p(j=2/t) = 1 \;\; o \;\; p(j=1/t) = 1 \; y \; p(j=2/t) = 0 \\ 1 & si \qquad p(j=1/t) = 1/2 \; y \; p(j=2/t) = 1/2 \\ (0,1) & si \qquad cualquier \; otro \; valor \; de \; p(j=1/t) = p(j=2/t) = 1/2 \end{cases} \tag{7}$$

A partir de lo anterior podemos afirmar que la impureza de un nodo será alta cuando las clases de los casos que contenga estén distribuidas de forma homogénea, será baja cuando haya preponderancia de alguna clase, y será nula (o sea que será "puro") cuando el nodo contenga todos casos de sólo una clase.

Tomemos cualquier nodo t y supongamos que tenemos un valor de un atributo que consideramos que podría ser candidato a formar una división a partir de una condición lógica que se aplique sobre él. Si aplicamos la condición lógica a todos los casos que componen el nodo generaremos dos nuevos nodos t_{der} y t_{izq}, siendo la cantidad de casos que contiene cada uno de los nodos una fracción de la cantidad de casos que poseía su nodo padre t. Llamaremos a estas fracciones p_{der} y p_{izq} respectivamente. La figura 3 muestra cómo quedaría la estructura de esta división binaria "División":

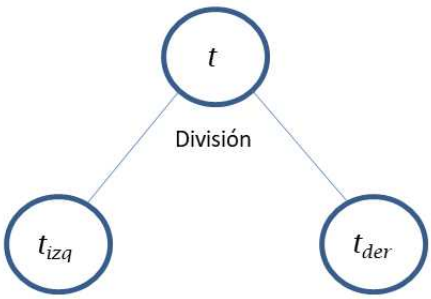

Figura 3: División de nodo padre en hijo a izquierda y a derecha

La "bondad de la división" es una medida de la performance de esta división y se evaluará mediante la impureza de los nuevos nodos hijos izquierdo y derecho (t_{izq} y t_{der}) generados a partir de ella. Como venimos planteando anteriormente lo que nos interesa es que la división haya generado dos nodos hijos que en su conjunto tengan una menor impureza que su nodo padre. La formula de este decrecimiento de la impureza es:

$$\Delta i(s,t) = i(t) - p_{izq} i(t_{izq}) - p_{der} i(t_{der}) \tag{8}$$

Aquí vemos que lo que hacemos es restar a la impureza del nodo padre las impurezas de los nodos hijos, ponderados por la proporción de casos que contiene cada uno de ellos. Esta variación es el decrecimiento de la impureza asociado a la división s en el nodo t.

La medida de impureza elegida cumple con las condiciones y las características vimos en el punto 2, y su expresión es la suguiente:

$$i(t) = \sum_{j=1}^{J} -p(j/t) log[p(j/t)] \tag{9}$$

3. Definir un set S de divisiones posibles s sobre un atributo o medición x, para luego en cada nodo y subnodo del árbol encontrar la división s^* que maximice el decrecimiento de la impureza.

$$\Delta i(s^*,t) = max \Delta i(s^*,t) \tag{10}$$

Ahora nos resta saber cuando dejar de dividir los subnodos generados a partir del proceso de maximización de la variación e impureza recién descripto. Este punto de corte se determina a partir de una regla práctica que ha probado dar buenos resultados, también llamada "heurística", que nos indica que cuando se haya llegado a un nodo cuyo variación

de la impureza (s^*, t) sea menor o igual a un cierto valor β el proceso de división binaria se detendrá y a ese nodo pasará a ser un nodo terminal.

$$\Delta i(s^*, t) < \beta \tag{11}$$

2.4. Criterio para evaluar la perfomance de una divisón

Como hemos visto el el criterio de división más común es el que evalúa la variación de la impueza. A esta medida de impureza en la teoría de árboles de decisión (y en general en la teoria de la información) se la denomina *entropía*. La entropía es una medida de desorden que puede ser aplicada a un conjunto de datos. El desorden corresponde a cuán mezclado (impuro) está el subconjunto de datos con respecto a la clase. Así que, por ejemplo, un subconjunto mixto en el que esten presentes casos de diferentes clases tendría alta entropía.

Como ya hemos visto en (9) , más técnicamente la entropía se define como:

$$i(t) = \sum_{j=1}^{J} -p(j/t)log[p(j/t)] \tag{12}$$

Cada $p(j/t)$ es la probabilidad (el porcentaje relativo) de la clase j dentro del subconjunto o nodo t, que varía de $p(j/t) = 1$ cuando todos los miembros del conjunto pertenen a la clase j, a $p(j/t) = 0$ cuando no hay miembros del conjunto perteneciente a la clase j.

Dado que la ecuación de entropía podría no prestarse a una comprensión intuitiva, la figuira 5 muestra la función entropía de un conjunto que contiene 10 instancias de dos clases, + y -. AllÍ se observa que la entropía mide el desorden general del conjunto, que va desde cero, el mínimo desorden (el conjunto tiene miembros todos con la misma, única clase) a uno, el máximo desorden (las clases están igualmente mezcladas). Ya que sólo hay dos clases, $p(+) = 1 - p(-)$, empezando con todas las instancias negativas en la parte inferior izquierda, $p(+) = 0$, el conjunto tiene un mínimo (es pura) y la entropía es cero. Si empezamos a cambiar las etiquetas de clase de los elementos del conjunto de - a +, la entropía aumenta. La entropía se maximiza en 1 cuando la instancia las clases están equilibradas (cinco de cada una), y $p(+) = p(-) = 0,5$. A medida que se cambian más etiquetas de clase,

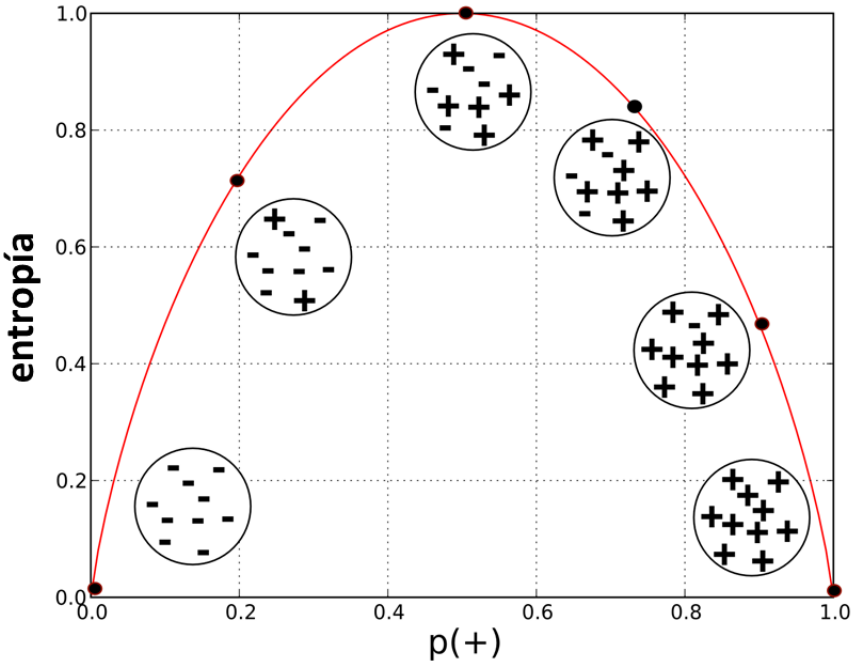

Figura 4: Curva de entropía. (extraído con permiso: *Data Science for Business* - Foster Provost and Tom Fawcett -O'Reilly. Copyright 2013)

la clase + comienza a predominar y la entropía baja de nuevo. Cuando todas las instancias son positivo, p(+) = 1 la entropía es mínima de nuevo.

La entropía es sólo una parte de la historia. Nos gustaría conocer cuanto se reduce la impureza al haber elegido ese valor de la variable para realizar la division y generar los nuevos subnodos. Un atributo segmenta un conjunto de casos en dos subconjuntos. La entropía sólo mide cuán impuro es un subconjunto individual. Afortunadamente, como la entropía nos revela cuán desordenado es cualquier subconjunto, utilizaremos el concepto de variacion de la impureza para medir cuanto disminuye la entropía de la división, o sea de los dos subconjuntos generados a partir de ella.

Veamos un ejemplo:

El conjunto padres tiene 30 instancias que consisten en 16 puntos y 14 estrellas . Este es un problema de dos clases ($\circ$ y $\star$)

$i(padre) = -p(\circ)log[p(\circ)] + p(\star)log[p(\star)]$= -0.53×(-0.9) + 0.47× (-1.1) = 0,99

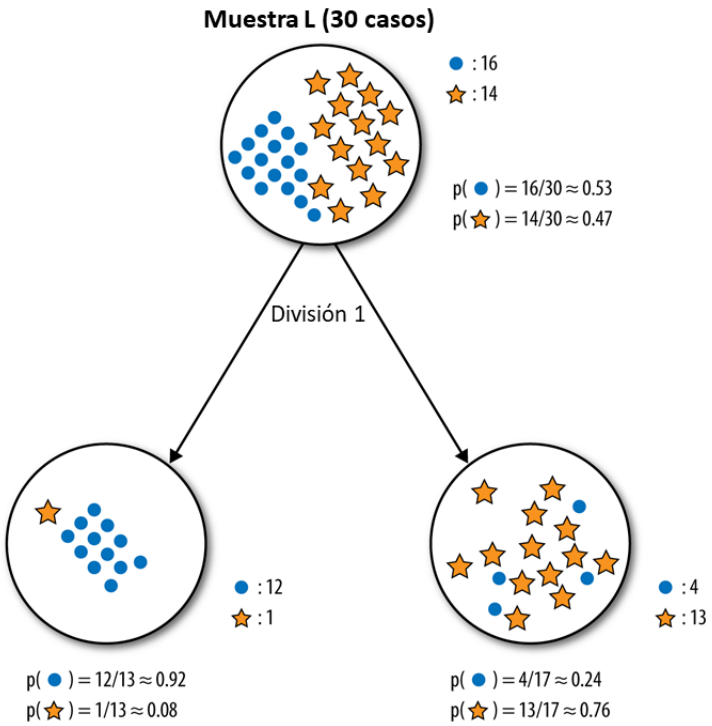

Figura 5: División del nodo padre del ejemplo en nodo hijo izquierdo y nodo hijo derecho(extraído con permiso: *Data Science for Business* - Foster Provost and Tom Fawcett -O'Reilly. Copyright 2013)

La entropía del nodo hijo de la izquierda es:

$i(nodo_{hijoizq}) = -p(\circ)log[p(\circ)] + p(\star)log[p(\star)]$= -0.92×(-0.12) + 0.08×(-3.7) = 0,39

La entropía del nodo hijo de la derecha:

$i(nodo_{hijoder}) = -p(\circ)log[p(\circ)] + p(\star)log[p(\star)]$= - 0.24×(- 2.1) + 0.76×(- 0.39) = 0.79

2.5. Predicción de la clase de un nuevo caso utilizando el clasificador

Si tuviéramos un nuevo caso, con su correspondiente vector de medidas X, y quisiéramos saber la predicción de clase que nos arrojaría la aplicación del clasificador, lo que haríamos

sería hacer que el caso recorra los diferentes nodos del árbol siguiendo el camino trazado por las sucesivas condiciones asociadas a cada división binaria, hasta llegar a un nodo terminal. Como cada nodo terminal tendrá una clase asociada, será esta la clase que prediga el clasificador para este nuevo caso.

Veamos otro ejemplo: a partir de una muestra de cientos de personas a las que en el pasado se les ofreció una póliza de seguro de vida (de las que se conocía su salario y su edad resultado de la operación o sea si lo compraron la póliza o no lo hicieron) se entrenó un árbol que clasificará a un nuevo cliente de una seguradora como comprador o no del seguro. Veamos qué pasaría con nuevo caso que tenga asociado un vector de atributos = (Ingreso=60, Edad=35). El árbol que se construyó a partir de lo casos del conjunto de entrenamiento es el siguiente:

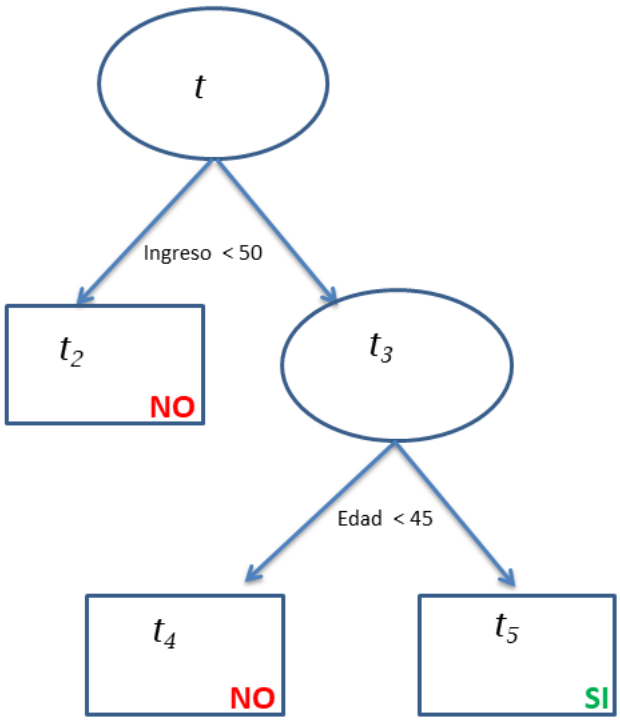

Figura 6: Árbol del ejemplo de candidato a adquirir una póliza de seguro de vida

El recorrido del nuevo candidato se iniciará evaluando la división s_1: $Ingreso < 50$, Al ser negativa la respuesta pasaremos al nodo hijo t_3, dónde someternos al vector de mediciones del nuevo caso a una nueva decisión $Edad < 45$. Esta condición genera una nueva división, y al haber sido afirmativa la respuesta a la condición seguiremos al subnodo hijo que se genera a la izquierda de la división, o sea el nodo t_4. Al ser este nodo un nodo terminal cuya clase

mayoritaria es "NO", se clasificará a esta nueva observación como de clase “NO”.

Otra forma de ver la forma en que un árbol clasifica a los casos a partir de su vector de atributos o mediciones es mediante las divisiones binarias en un espacio de características o espacio de datos. El especio de características o datos es una representación bidimensional o tridimensional donde los ejes representan a los atributos, y un símbolo (puede ser también una letra o un nro.) representan a la clase. En nuestro ejemplo las divisiones, y los correspondientes nodos terminales obtenidos partir del proceso de divisiones binarias recursivas recién descriptos, es:

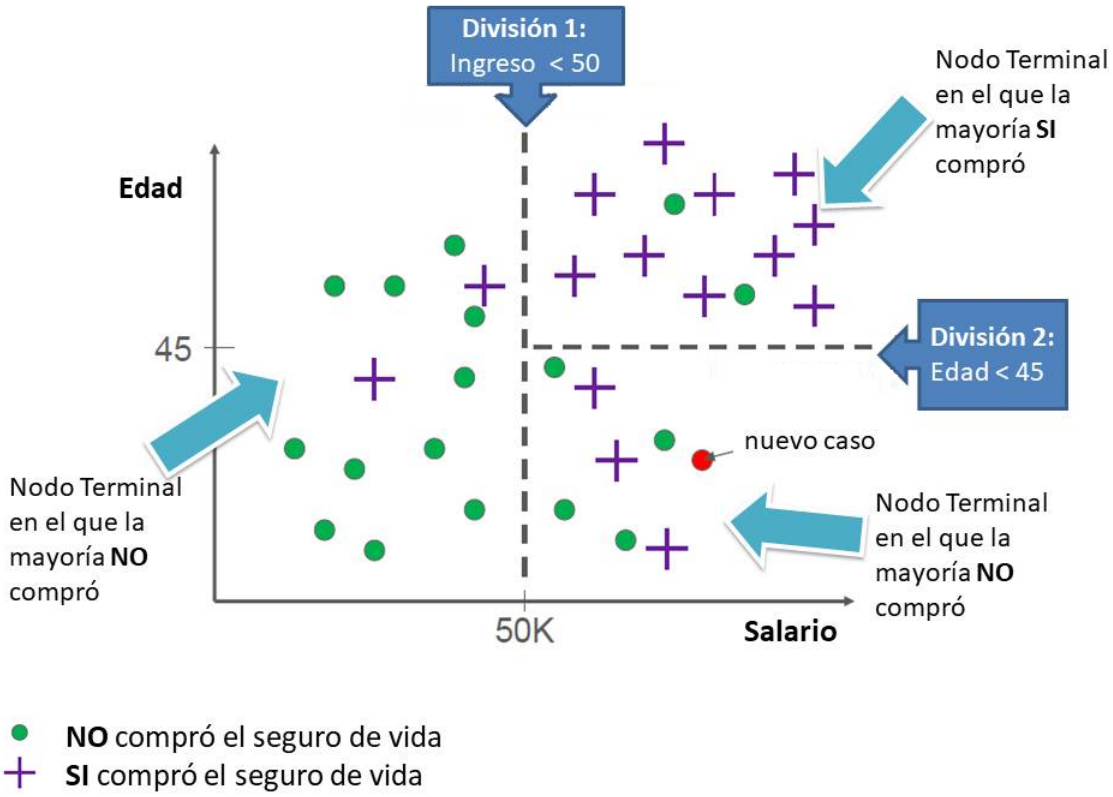

Figura 7: Divisiónes binarias recursivas del árbol del de decisión del ejemplo de las pólizas de seguro de vida

Aquí podemos observar claramente como a cada nodo terminal se le asigna la clase que se encuentra en la mayoría de los casos del set de entrenamiento que se encuentran confinados entre sus límites. A su vez podemos notar también que por la definición de división binaria todas las los nodos en el espacio de características terminan siendo rectángulos, siendo esta

la única forma geométrica que se pude generar a partir de este procedimiento de partición binaria recursiva.

2.6. Regla de asignación de clase en nodos terminales y estimadores del error de sustitución

Hemos visto que en un clasificador la asignación de clase de un nodo terminal se realiza mediante la regla de la mayoría: la clase asociada al nodo terminal t será la clase que tenga la mayor proporción de casos, o sea la $j(t)$ que tenga la mayor $p(j/t)$. A esta clase j, mayoritaria en el nodo t, la llamaremos $j^*(t)$. Formalmente:

si $p(j/t) = max\, p(j/t)$ entonces $j^{(*)}(t) = j_t$

Una estimación de la probabilidad de clasificar correctamente como "j" a un caso del nodo terminal t será $p(j/t)$. Por ende -para este ejemplo en el que sólo tenemos dos clases, i y j- la probabilidad de clasificar en forma errada como j a un caso que cuya clase real es "j" sería $1 - p(j/t)$.

A este último estimador lo tomaremos como el estimador de sustitución(t) del error de clasificación dado que el caso cayó en un nodo terminal t, que podemos escribir de esta manera:

$$p(j/t) = 1 - max\, p(j/t) \tag{13}$$

SI llamamos $R(t)$ a $R(t) = r(t)p(t)$ entonces el estimador de sustitución del error global de clasificación $R^*(T)$ del clasificador de árbol binario T, siendo $\widetilde{T}$ el conjunto de sus nodos terminales, es:

$$R(T) = \sum_{t \in \widetilde{T}} R(t) \tag{14}$$

Una propiedad muy importante y relevante de $R(t)$ es que cuantas más bifurcaciones posea el árbol menor será su valor. O sea que si el árbol pasa de tener tamaño T a un tamaño mayor T', entonces

$$R(T') \leq R(T) \tag{15}$$

Y ahora expresado en términos de bifurcación y nodos hijos, tendremos que:

$$R(t') \geq R(t_{izq}) + R(t_{der}) \tag{16}$$

2.7. Arboles del tamaño adecuado

El crecimiento de los arboles y la determinación del tamaña adecuado es una de las tareas más delicadas a la hora de desarrollar un árbol de decisión. Diversos métodos se han probado para encontrar un punto de detención para el proceso recursivo de bifurcación y partición, pero ninguno de ellos permitía encontrar un punto en el que se utilice toda la información contenida en el set de entrenamiento $\mathcal{L}$ y su vez el árbol no se sobreajustara a los datos de entrenamiento

Recordemos que la regla de detención de las bifurcaciones consiste en determinar un umbral β y decidir por no seguir dividiendo un nodo si el máximo decrecimiento de la impureza es menor al umbral $max\ \Delta i(s,t) < \beta$.

Los problemas para determinar el tamaño óptimo del árbol surgen al evaluar diferentes valores de β. Si β es muy bajo lógicamente llegaremos a tener nodos de tamaño muy pequeño, mientras que si β es muy grande el proceso se detendrá en un punto en el que no estaremos seguros si los nodos subsiguientes a esa división podrían haber aportado una importante reducción de la impureza, y al haber cortado el proceso con un β grande nos estamos perdiendo

la capacidad de este árbol de mayor tamaño de mejorar la estimación del error de clasificación

El procedimiento que se terminó implementando en *Breiman et.al* fue a la inversa: permitir que el árbol creciera hasta que los nodos terminales fueran muy pequeños(considerando las divisiones hasta llegar a nodos muy pequeños) , y a partir de allí empezar un proceso de "poda" o recombinaciónen el se vayan generando, en forma secuencial, árboles cada vez más pequeños

El punto de detención de este nuevo proceso se realizaría mediante la evaluación del error en un set de prueba o mediante validación cruzada.

3. Proceso de Poda

Recordemos que el estimador del error de clasificación $R^*(T)$ es:

$$R(T) = \sum_{t \in \widetilde{T}} r(t)p(t) = \sum_{t \in \widetilde{T}} R(t) \tag{17}$$

El primer paso de este proceso es dejar crecer al árbol hasta que sus nodos terminales sean de tamaño muy pequeño o hasta que sean completamente homogéneos (puros o de idéntico valor). El estándar para dejar crecer el árbol es determinar un $Nmin$ tal que $N(t) \leq N_{min}$

El proceso irá generando subárboles a partir de las podas sucesivas de nodos, que eventualmente llegará a un árbol t_1, que será el nodo raíz. Para empezar a definir el proceso de poda, llamaremos t' un descendiente de un nodo t, si existe una trayectoria que recorra hacia abajo el árbol, que conecte a t' con t.

Definiremos a la poda de una rama a la eliminación de todos los nodos descendientes de un nodo T_t. El árbol podado resultante será $T - T_t$,

Al haber muchísimas formas de podar un árbol a partir de esta definición de poda es que se torna necesario idear un mecanismo selectivo para implementar este proceso, que

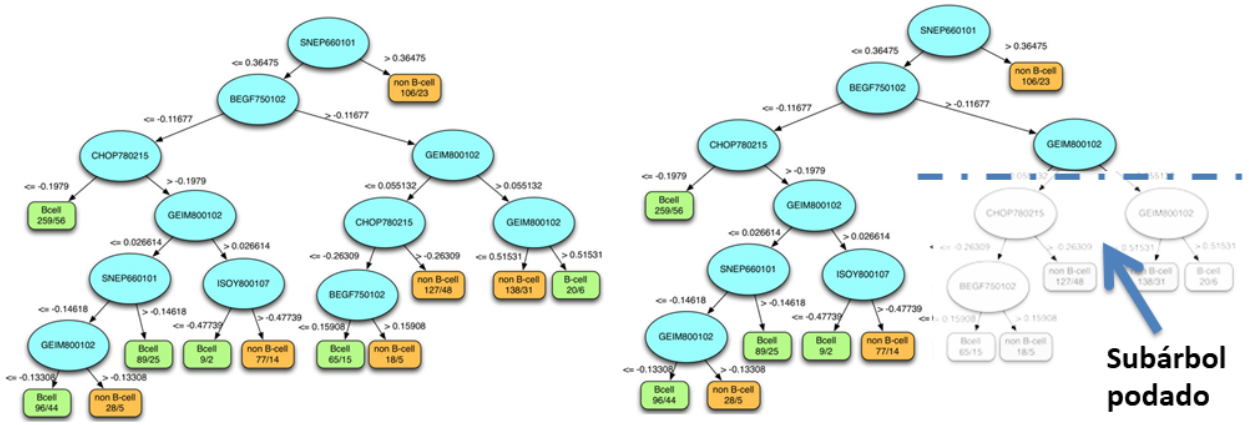

Figura 8: Subárbol podado

seleccione el mejor subárbol a podar. La determinación del mejor subárbol a podar se lleva a cabo a partir de la reducción recursiva del estimadores del error $R^*(T)$.

El proceso se inicia en el árbol completo T_{max}, computándose el error $R(t)$ de cada uno de sus nodos, y podándose en forma progresiva en cada paso al nodo que hace que el $R(T)$ del nuevo árbol sea los más pequeño posible.

Para cada H, siendo $1 \leq H \leq L$, llamaremos T_H al conjunto de todos los subárboles de T_{max} de tamaño H que se pueden generar a partir de la poda progresiva de nodos de T_{max}. Seleccionaremos el subárbol T para el que

$$R(T_H) = min\ R(T) \tag{18}$$

Aquí nos encontramos nuevamente con el problema de tener que determinar cual es el punto apropiado para detener este proceso.

3.1. Poda de Costo-complejidad mínimo

La idea detrás de este proceso es: para cada subárbol $T < T_{max}$ se define su complejidad como el numero de nodos terminales del árbol $|\widetilde{T}|$. Sea $\alpha \geq 0$ llamado parámetro de com-

plejidad que define el costo-complejidad asociado

$$R_\alpha(T) = R(T) + \alpha|\widetilde{T}| \tag{19}$$

El segundo término puede ser interpretado como una penalización asociada a la cantidad de nodos terminales.

Ahora para cada valor de α deberemos seleccionar mediante el proceso de poda el subárbol de Tmax que minimice el costo complejidad

$$R_\alpha(T_\alpha) = min\ R_\alpha(T) \tag{20}$$

4. Conclusiones

Los modelos de clasificación y regresión basados en la metodología *ÇART: Classification and Regression tress"* implementada por Breiman y otros es la base de la inmensa mayoría de los desarrollos de machine learning aplicados en la actualidad. La modelización de sistemas de la vida real, en particular en lo que contamos con información estructurada (que en la vida profesional representan la mayoría de los casos), exige el desarrollo de algoritmos de alta complejidad como son los ensambles de arboles de decisión. Estos sofisticados modelos, que son los que presentan la mejor performance, están completamente basados en árboles de decisión de partición binaria recursiva. Es por eso que resulta de de fundamental importancia el aprendizaje y la comprensión acabada de sus componentes fundamentales, sus mecanismos y los procesos que los componen, tanto computaciones como lógicos y estadísticos, pues es sobre ellos y su dominio que el profesional o el practicante de la ciencia de datos podrán desarrollar modelos predictivos de clasificación robustos y confiables.