

Universidade da Coruña

Máster en Técnicas Estadísticas

Asignatura de Análisis Exploratorio de Datos

Primer Cuatrimestre 2009/10

Trabajo:

Árboles de clasificación y regresión

Alumnos:

- **Pilar Cacheiro Martinez**
- **Iván Díaz Costoya**
- **José Benito Pérez López**

Contenido

1.	Introducción	3
1.1.	Contexto y objetivo	3
1.2.	Organización del trabajo	3
2.	Árboles de decisión en Minería de Datos	4
2.1.	Modelización algorítmica en análisis estadístico	4
2.2.	Introducción a la minería de datos	6
2.3.	Tareas predictivas de regresión y clasificación	9
2.4.	Algoritmos supervisados de inducción	11
2.5.	Árboles de decisión	13
3.	Algoritmos TDIDT	17
3.1.	Algoritmos de partición	17
3.2.	Algoritmo ID3	20
3.3.	Algoritmo C4.5	23
3.4.	Aplicación de J48 a los datos de prueba	26
4.	CART: Classification and Regression Trees	52
4.1.	Introducción	52
4.2.	Construcción del árbol máximo	52
4.3.	Poda del árbol	57
4.4.	Selección del árbol óptimo	58
4.5.	Ventajas y desventajas de CART	59
4.6.	Algunas herramientas que implementan CART	61
4.7.	Aplicación de CART a los datos de prueba	62
4.8.	Conclusión	68
5.	NBTree	69
5.1.	Introducción	69
5.2.	Árboles de clasificación	69
5.3.	Clasificadores Naive-Bayes	70
5.4.	Algoritmo NBTree	71
5.5.	Ventajas NBTree	74
5.6.	Aplicación de NBTree a los datos de prueba	74
6.	Decision Stump	81
6.1.	Introducción	81
6.2.	Aplicación de Decision Stump a los datos de prueba	81
6.3.	Conclusión	82
7.	Comparativa y conclusiones	83
8.	Bibliografía	84

1. Introducción

1.1. Contexto y objetivo

La Minería de Datos es una disciplina que ha evolucionado ampliamente en los últimos tiempos y en la que como veremos confluyen diferentes áreas de conocimiento o ciencias. Esto provoca que dentro del marco de trabajo de la Minería de Datos, en ocasiones sea confuso definir los conceptos y técnicas manejadas.

En el caso de los árboles de regresión y clasificación que nos ocupará en este trabajo es especialmente relevante la confluencia entre Minería de Datos (*Data Mining* en inglés) y Aprendizaje Automático (*Machine Learning* en inglés), dado que ambos enfoques resuelven el mismo problema de predicción mediante la utilización de algoritmos supervisados.

En este trabajo se analizará la resolución de tareas de regresión y clasificación mediante técnicas basadas en algoritmos supervisados de inducción para el entrenamiento de árboles de decisión, desde un enfoque de Minería de Datos; aunque como ya hemos dicho de igual forma se podría plantear como parte del Aprendizaje Automático en Inteligencia Artificial.

1.2. Organización del documento

En el siguiente capítulo se introducen los conceptos identificados en el contexto del trabajo, la Minería de Datos, haciendo especial hincapié en la tarea o problema de regresión y clasificación que nos ocupa.

En el siguiente capítulo se detalla de manera general la técnica basada en el entrenamiento de árboles de decisión para resolver el problema de regresión y clasificación.

A continuación se incluirán cuatro capítulos, cada uno referido a un tipo de árbol de regresión y clasificación, en el que se incluirá una descripción detallada y la aplicación de algún algoritmo a los datos de prueba del fichero EncuestaAccidentes.xls.

El siguiente capítulo incluye una comparativa entre los árboles entrenados en cada una de las cuatro aplicaciones mencionadas anteriormente.

Por último se incluye un capítulo de referencias.

2. Árboles de decisión en Minería de Datos

2.1. Modelización algorítmica en análisis estadístico

La Estadística es una ciencia de base matemática que trata fenómenos aleatorios a través de la recolección, análisis e interpretación de datos.

El Análisis Estadístico piensa en los datos como generados por una caja negra, en la que por un lado entra un vector de variables (x), y por el otro salen las variables respuesta (y), en la que en su interior Leo Brieman sitúa la naturaleza [Brieman, 2001b], tal y como se representa en la siguiente imagen:



El análisis en estadística aplicada tiene dos objetivos principales, la descripción y la inferencia de datos, área esta última que incluye la predicción o pronóstico asociado a futuras observaciones y que nos ocupa en este trabajo.

El problema de predicción en la inferencia del análisis estadístico trata de estimar una función f , que operando sobre x nos pronostique y . El problema se llamará regresión cuando la salida pueda ser un valor numérico continuo, y clasificación cuando sea una etiqueta de clase.

Para la resolución del problema, el análisis estadístico utilizará una muestra de la población en estudio que contenga las variables de respuesta " y " para el vector de entrada " x ", con la que se especificará (según los casos también se dirá calibrará, entrenará o aprenderá) un modelo que estime las características de f necesarias para la predicción.

En [Brieman, 2001b], "Statistical Modeling: The two cultures", Leo Breiman habla de dos culturas o enfoques de la modelización en el contexto del análisis estadístico:

- uno se basa en que los datos son generados por un modelo de datos estocástico dado; y llamaremos Modelización de Datos
- el otro utiliza modelos algorítmicos, y trata los mecanismos de los datos como desconocidos; y llamaremos Modelización Algorítmica

En la *modelización de datos* el analista comienza especificando un modelo de datos estocástico en el interior de la caja, por ejemplo el representado por diferentes modelos de la regresión paramétrica o no-paramétrica

(Regresión Lineal, Regresión Logística, Regresión Multinomial o los modelo Cox) o la inferencia bayesiana.

El modelo dice cómo el analista cree que son generados los datos, y contiene una componente determinista y una aleatoria, con lo que podríamos representarlo como la siguiente fórmula:

Variable respuesta=función(variables entrada, aleatoriedad, parámetros)

El modelo no está totalmente especificado, así que el analista utiliza los propios datos para seleccionar el modelo particular para cada problema, mediante el calibrado (especificación de parámetros si es un modelo paramétrico, o ajustando funciones si es no-paramétrico).

Una vez calibrado el modelo, se valida mediante test de bondad de ajuste y análisis de residuos, con lo que podrá ser usado para la predicción.

De esta forma, en este enfoque el interior de la caja negra está relleno, tal y como se representa en la siguiente figura:



Claramente este enfoque ha sido y es efectivo en un amplísimo rango de situaciones. Sin embargo en ocasiones, por el origen o la naturaleza de los datos, insistir en especificar de inicio un modelo puede limitar al estadístico [Faraway, 2006].

Tukey (1977) abogó por el Análisis Exploratorio de Datos (EDA son las siglas en inglés), que son métodos gráficos y descriptivos que pueden ayudar a clarificar la estructura de los datos o al menos sugerir una forma apropiada para el modelo. Pero EDA no es una solución completa, en especial para la predicción. A continuación veremos otro enfoque completo, y apropiado para ciertos problemas del análisis estadístico.

En enfoque de *modelización algorítmica* considera desconocido el interior de la caja negra, y no busca clarificarlo; su aproximación es la un algoritmo o proceso de inducción de conocimiento, que opera sobre el vector x para predecir la respuesta y .

De esta forma la caja negra se representaría así:



La validación se realiza en este enfoque con medidas de exactitud de la predicción.

Este segundo enfoque ha aprovechado avances en otras ramas de la ciencia, como es la Inteligencia Artificial, en la que se conoce como Aprendizaje Automático o Machine Learning la rama que trata la inducción de conocimiento, aunque más centrado en el estudio de la complejidad computacional del problema; y en la que se conoce como Aprendizaje Supervisado el que se basa en algoritmos que utilizan muestras de entrenamiento, como es lo habitual en el caso de problemas de predicción.

Ambos enfoques han desarrollado técnicas eficaces, en el sentido de su capacidad para predecir o explicar algo; y que según el caso puede ser más apropiado el uso unas u otras técnicas, o su combinación, para resolver el problema al que se enfrenta el analista.

En general los modelos generados con el primer enfoque facilitan la interpretación del modelo y la difusión de las conclusiones, y los modelos generados por el segundo descubren relaciones estructurales más complejas pero de más difícil, o en ocasiones imposible según la técnica utilizada, interpretación.

Tiene especial relevancia en la elección del enfoque y la técnica las siguientes encrucijadas, incluidas en [Brieman, 2001b] como lecciones aprendidas:

- La multiplicidad de modelos
- Simplicidad vs. Exactitud
- La maldición de la dimensionalidad

2.2. Introducción a la minería de datos

La Minería de Datos surge en este contexto como un proceso o metodología que busca el descubrimiento de conocimiento procesable en los datos, utilizando las técnicas de modelización estadística. Su enfoque principal es el de aprovechar el conocimiento implícito en las bases de datos, y está influido por los avances en otras disciplinas, como los sistemas de apoyo a las decisiones, el aprendizaje automático, las bases de datos o la recuperación de información.

Se denomina Minería de Datos¹ al conjunto de técnicas y herramientas aplicadas al proceso no trivial de extraer y presentar conocimiento implícito, previamente desconocido, potencialmente útil y humanamente comprensible, a partir de grandes conjuntos de datos, con objeto de predecir de forma automatizada tendencias y comportamientos; y describir de forma automatizada modelos previamente desconocidos².

La Minería de Datos en [Morales, 2003] se presenta como un proceso completo de descubrimiento de conocimiento que involucra varios pasos [Morales, 2003]:

1. Entendimiento del dominio de aplicación, el conocimiento relevante a utilizar y las metas del usuario.
2. Seleccionar un conjunto de datos en donde realizar el proceso de descubrimiento.
3. Limpieza y pre-procesamiento de los datos, diseñando una estrategia adecuada para manejar ruido, valores incompletos, valores fuera de rango, valores inconsistentes, etc.
4. Selección de la tarea de descubrimiento a realizar, por ejemplo, clasificación, agrupamiento o clustering, reglas de asociación, etc.
5. Selección de los algoritmos a utilizar.
6. Transformación de los datos al formato requerido por el algoritmo específico de explotación de datos, hallando los atributos útiles, reduciendo las dimensiones de los datos, etc.
7. Llevar a cabo el proceso de minería de datos para encontrar patrones interesantes.
8. Evaluación de los patrones descubiertos y presentación de los mismos mediante técnicas de visualización. Quizás sea necesario eliminar patrones redundantes o no interesantes, o se necesite repetir algún paso anterior con otros datos, con otros algoritmos, con otras metas o con otras estrategias.
9. Utilización del conocimiento descubierto, ya sea incorporándolo dentro de un sistema o simplemente para almacenarlo y reportarlo a las personas interesadas.

Es este proceso es muy importante la etapa del pre-procesamiento de los datos y su transformación al formato requerido por el algoritmo, ya que dependiendo de cómo se realicen estas tareas, va a depender la calidad final de los patrones descubiertos. Un patrón es interesante si es fácilmente entendible por las personas, potencialmente útil, novedoso o valida alguna hipótesis que el usuario busca confirmar. Un patrón interesante representa conocimiento [Ale, 2005].

¹ Servente y García Martínez, 2002; Perichinsky y García-Martínez, 2000; Perichinsky

et al., 2000; Perichinsky et al., 2001; Perichinsky et al., 2003

² Piatetski-Shapiro et al., 1991; Chen et al., 1996; Mannila, 1997

En [Witten & Frank 2000] se define la minería de datos como el proceso de extraer conocimiento útil y comprensible, previamente desconocido, desde grandes cantidades de datos almacenados en distintos formatos, y el término Minería de Datos Inteligente³ refiere específicamente a la aplicación de métodos de aprendizaje automático⁴, para descubrir y enumerar patrones presentes en los datos, para estos, se desarrollaron un gran número de métodos de análisis de datos basados en la estadística⁵. En la medida en que se incrementaba la cantidad de información almacenada en las bases de datos, estos métodos empezaron a enfrentar problemas de eficiencia y escalabilidad y es aquí donde aparece el concepto de minería de datos. Una de las diferencias entre al análisis de datos tradicional y la minería de datos es que el primero supone que las hipótesis ya están construidas y validadas contra los datos, mientras que el segundo supone que los patrones e hipótesis son automáticamente extraídos de los datos [Hernández Orallo, 2000].

En [Fayyad et al. 1996a] se define el descubrimiento de conocimiento a partir de datos (KDD sus siglas en inglés) como “el proceso no trivial de identificar patrones válidos, novedosos, potencialmente útiles y, en última instancia, comprensibles a partir de dos datos” que consta de diferentes fases, en la que la Minería de Datos propiamente dicha sería una fase que aglutinaría las técnicas de modelización estadística y aprendizaje automático.

El proceso KDD tendría entonces según [Hernández, J., Ramírez, M. J., Ferri, C., 2005] como entrada datos y como salida conocimiento para la toma de decisiones, y constaría de las siguientes fases principales:

1. Integración y recopilación (data warehouse)
2. Selección, limpieza y transformación
3. Minería de Datos
4. Evaluación e interpretación
5. Difusión y uso

En principio la Minería de Datos puede aplicarse a cualquier tipo de datos, siendo las técnicas diferentes para cada una de ellas. Los principales tipos de datos que podrían diferenciarse serían principalmente podrían ser los provenientes de bases de datos relacionales, los provenientes de otros tipos de estructuras de bases de datos (espaciales, temporales, textuales y multimedia) y los datos no estructurados provenientes de la web o de otros tipos de repositorios de documentos [Hernández, J., Ramírez, M. J., Ferri, C., 2005].

³ Evangelos y Han, 1996; Michalski et al., 1998

⁴ Michalski et al., 1983; Holsheimer y Siebes, 1991

⁵ Michalski et al, 1982

Los modelos que definen las relaciones o resúmenes de los datos pueden ser de dos tipos: predictivos y descriptivos [Hernández, J., Ramírez, M. J., Ferri, C., 2005]. Los modelos predictivos pretenden estimar valores futuros o desconocidos de variables de interés a partir de otras, mientras que los descriptivos tratan de identificar patrones que explican o resumen los datos examinados.

Los diferentes tipos de problemas que pueden ser resueltos mediante Minería de datos son llamados tareas, cada una de las cuales tiene sus propios requisitos y tipos de datos que maneja y obtiene.

Las tareas pueden producir modelos predictivos o descriptivos. Las principales tareas predictivas son la clasificación, regresión, categorización, y priorización; mientras que agrupamiento, reglas de asociación, correlación y factorización, así como el descubrimiento de dependencias funcionales e instancias anómalas son las principales descriptivas.

La Minería de Datos es un campo multidisciplinar, que se nutre de otras áreas relacionadas, como son la Estadística, los sistemas de apoyo a toma de decisiones, las bases de datos, la recuperación de información, el aprendizaje automático, la visualización de datos, la computación paralela y distribuida, y otras disciplinas; y que tiene aplicación en multitud de áreas como banca y finanzas, análisis de mercado, seguros, ... [Hernández, J., Ramírez, M. J., Ferri, C., 2005].

Las principales técnicas de la minería de datos, cada una con diferentes algoritmos de implementación, se basan principalmente en los siguientes métodos:

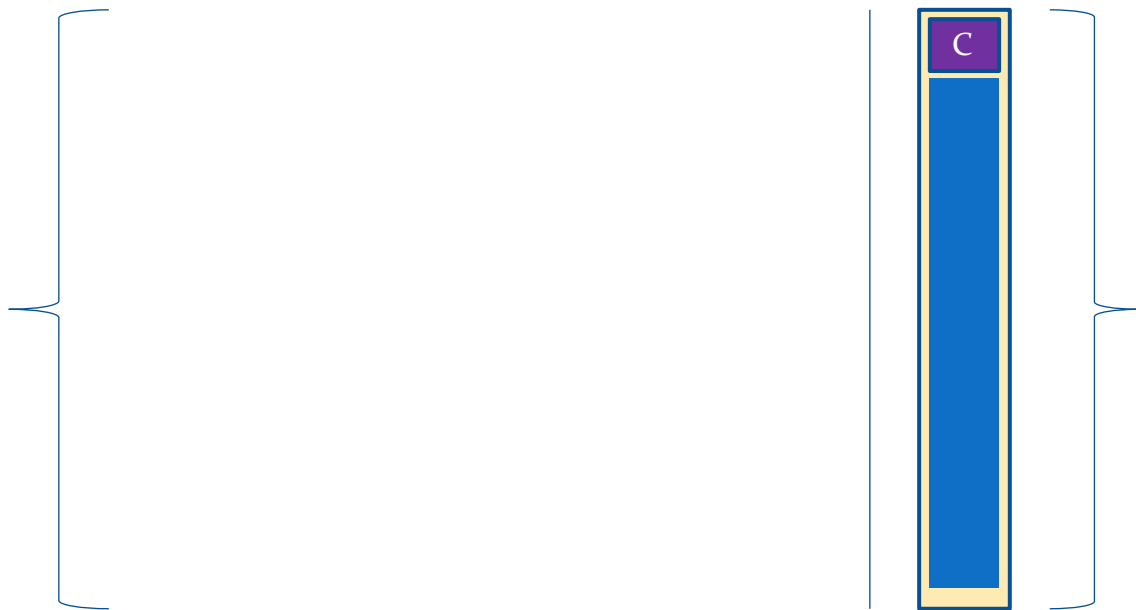
- Algebraicos y estadísticos
- Bayesianos
- Conteos de frecuencias y tablas de contingencia
- Árboles
- Sistemas de aprendizaje de reglas
- Relacionales, estructurales y declarativos
- Redes neuronales artificiales
- Núcleo y máquinas de soporte vectorial
- Estocásticos y difusos
- Distancia o densidad

2.3. Tareas predictivas de regresión y clasificación

Si representamos cada individuo por un vector (traspuesto) de dimensión p , en el que cada ítem corresponde a una característica cuantitativa y/o cualitativa, la salida buscada en el problema sería un valor, cuantitativo o cualitativo según el caso, para ese individuo. Por tanto las tareas de clasificación y regresión en Minería de Datos se utilizan para clasificar o estimar un conjunto de datos basado en los valores de sus atributos.

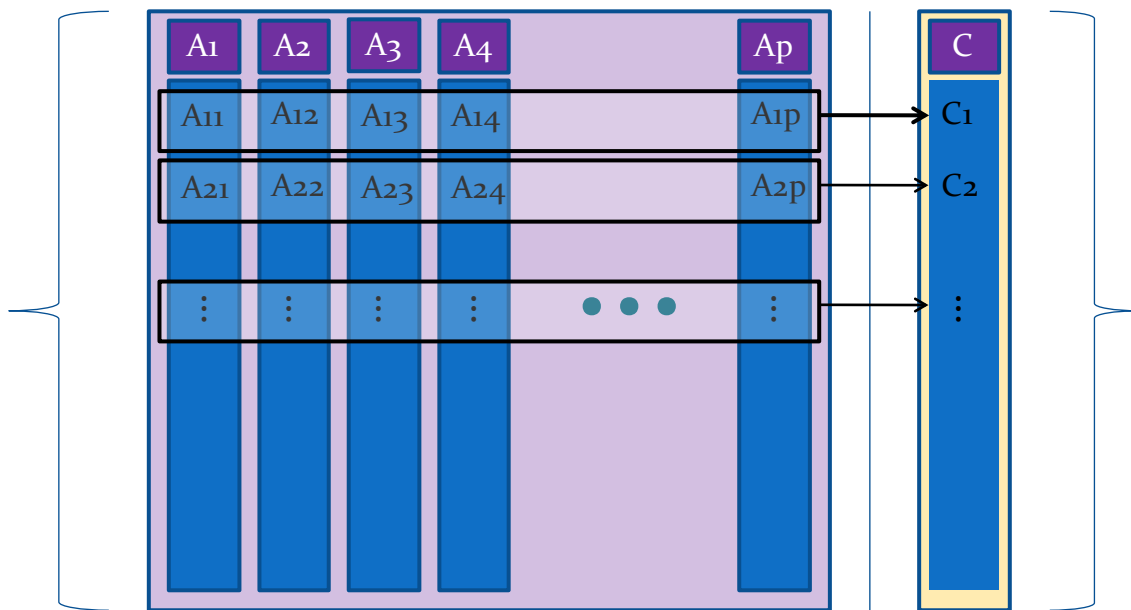
Cuando la variable que se quiere estimar es una clase (categórica o numérica discreta con pocos valores posibles) se llama tarea de clasificación (y a la variable que se quiere clasificar se le llama variable clasificadora); en caso contrario, es decir se quiere estimar una variable numérica continua o discreta con un elevado número de valores, se llama tarea de regresión (y a la variable que se quiere estimar se le llama regresora). Las variables que recogen los atributos a partir de los que se quiere realizar la estimación se llaman explicativas.

La tarea de clasificación comienza con la necesidad de estimar los valores de un atributo poblacional C.



Las técnicas basadas en algoritmos supervisados, utilizadas para la predicción de regresión o clasificación como veremos más adelante para el caso de los árboles de decisión, se basan en la creación de un modelo predictivo, basado en un serie de variables explicativas, que se incorporarán al proceso por su nivel de significación.

En el caso de la clasificación se trata de encontrar las propiedades comunes entre un conjunto de individuos y objetos que permitan su clasificación, es decir, cada técnica trata de identificar los atributos relevantes para la clasificación (variables explicativas) y construir una función (fórmula algebraica o algoritmo) que estime para cada individuo poblacional el valor de su atributo clasificador a partir de sus valores en los atributos explicativos, tal y como representa la imagen siguiente:



Por tanto la tarea consiste en aprender una función $\lambda: E \rightarrow C$, donde E es el conjunto de todos los posibles elementos de entrada, es decir, $E = A_1 \times A_2 \times \dots \times A_n$, llamada clasificador y que se ha representado como:



La clasificación suave supone aprender una función $\theta: E \rightarrow \mathbb{R}$ que significa el grado de certeza de la predicción hecha por la función λ . Esta extensión permite la ordenación (ranking) de predicciones o la selección de los n mejores ejemplos.

Por último la estimación de probabilidad de clasificación es una generalización de la clasificación suave, en la que en lugar de aprender las funciones λ y θ , se trata de aprender m funciones $\theta_j: E \rightarrow \mathbb{R}$, donde m es el número de clases de C . Es decir, cada función θ_j a aprender retorna un valor real p_j , que se denomina probabilidad de la clase C_j , y significa el grado de certeza de que un ejemplo sea de la clase j . Si además se cumple que todos los p_j son no negativos y suman 1, entonces representa la probabilidad de que un ejemplo sea de la clase j . El conjunto de las m funciones aprendidas se llama estimador de probabilidad.

El caso de la regresión se definiría de forma semejante, pero con el caso de que C sea numérico entero o real.

2.4. Algoritmos supervisados de inducción

Los algoritmos supervisados son procesos de inducción que se basan en una muestra de datos que incluye las variables respuesta para estimar un modelo que pronostique las respuestas para futuras observaciones.

De esta forma los algoritmos supervisados se componen de dos fases principales para generar el modelo:

1. Fase de entrenamiento:

En esta fase se analiza un conjunto de datos (llamado de entrenamiento), y mediante un método supervisado, se desarrolla un modelo utilizando las características disponibles en los ellos, y se estiman los parámetros del modelo. El método se conoce como supervisado debido a que, para el conjunto de entrenamiento, se conoce la clase de pertenencia y se le indica al modelo si la clasificación que realiza es correcta o no. La construcción del modelo se realimenta de estas indicaciones del supervisor [Chen et al., 1996].

Es decir esta fase tiene una primera sub-fase, que llamaremos Estimación; donde se elige y desarrolla un modelo y se estiman los parámetros del mismo.

Según la técnica utilizada puede existir una segunda sub-fase, que llamaremos Calibración (de hiper-parámetros) o primera validación; en la que estimar parámetros de optimización del modelo y hacer una primera validación del modelo resultante.

2. Fase de Validación o prueba:

La fase de Validación (o segunda validación si hubo Calibración) del modelo, permitirá validar el modelo y compararlo con otros modelos.

Para la realización de validación existe una serie de técnicas (de elección de datos y de medidas de evaluación) llamadas Técnicas de Validación.

La validación puede realizarse con la misma muestra que el entrenamiento, con una parte de la muestra reservada para ello o con validación cruzada. El primer caso no es recomendable, pues facilitaría el sobreajuste (a los datos) del modelo. Si se dispone de suficientes datos el mejor criterio sería reservar una parte para la validación, pero sino el mejor sería la validación cruzada.

La validación cruzada (con m pliegues) consiste en partir aleatoriamente la muestra en m grupos de datos (generalmente $m=10$), y realizar el entrenamiento m veces, cada uno con la sub-muestra que se obtiene al escoger en cada ocasión $m-1$ grupos de datos distintos, con lo que se calculan m ratios de error independientes. Finalmente, se construye un modelo con todos los datos y se obtienen sus ratios de error y precisión promediando las n ratios de error disponibles.

Otra posible técnica de validación, para estimar el error de un modelo cuando se disponen de pocos datos, es el bootstrapping; que consiste en construir primero un modelo con todos los datos, y luego se crean

numerosos conjuntos de datos (llamados bootstrap samples) haciendo un muestreo de los datos originales con reemplazo. Por último se construye un modelo con cada conjunto y se calcula su ratio de error sobre el conjunto de prueba (que son los datos sobrantes en cada muestreo). El error final estimado para el modelo construido con todos los datos se calcula promediando los errores obtenidos para cada muestra.

Existen diferentes medidas de evaluación de los modelos generados en función de la tarea.

- En el caso de la clasificación lo habitual es evaluar la calidad de los patrones encontrados respecto a su precisión predictiva, que se calcula como el número de instancias del conjunto de prueba clasificadas correctamente dividido por el número de instancias totales en el conjunto de prueba.
- En el caso de la regresión la manera más habitual de evaluar un modelo es mediante el error cuadrático medio del valor predicho respecto al valor que se utiliza como validación, aunque se pueden utilizar otras medidas del error.

2.5. Árboles de decisión

Un árbol consiste en un grafo direccional, con nodos (variables de entrada), ramas (grupos de registros en las variables de entrada) y hojas (valores de la variable de salida). Tiene un único nodo inicial o raíz, y un único nodo origen para el resto de nodos.

Los árboles permiten representar de forma gráfica una serie de reglas o condiciones organizadas de forma jerárquica, de forma que la decisión final que se debe tomar respecto a un vector o individuo de entrada, se puede determinar siguiendo las condiciones que se cumplen desde la raíz del árbol hasta alguna de sus hojas.

Los árboles de decisión han sido utilizados desde hace siglos, y son especialmente apropiados para expresar procedimientos médicos, legales, comerciales, estratégicos, matemáticos, lógicos, etc.

El método de predicción basado árboles de decisión es quizás el más sencillo de utilizar y entender de los basados en algoritmos de supervisión de la Minería de Datos.

En el resto del trabajo se estudiarán metodologías de construcción de algoritmos supervisados para la generación o entrenamiento de árboles de decisión, detallando alguno de ellos y aplicándolo a los datos de estudio.

Los métodos basados en el entrenamiento de árboles de predicción son métodos de aprendizaje supervisado que se basan en algoritmos de inducción que construyen un árbol a partir del conjunto de datos de

entrenamiento, para utilizarlo desde ese momento en la tarea predictiva de clasificación o regresión. Por tanto constan de dos fases: una fase de construcción del modelo (representado como un árbol en este caso) llamada entrenamiento (en la que se puede incluir o no un primer nivel de validación, aunque es recomendable) y una segunda de prueba o segundo nivel de validación.

Un paso importante en la construcción del árbol de decisión es la poda, la cual elimina las ramas no necesarias, resultando en una clasificación más rápida y una mejora en la precisión de la clasificación de datos [Han y Kamber, 2001].

La metodología de árboles ha calado en las ciencias de Estadística y Computación [Faraway, 2006], y un precursor fue el algoritmo CHAID (Chi-squared Automatic Interaction Detection) desarrollado por Morgan and Sonquist (1963) e implementado por Kass (1980), aunque el libro de Breiman, Friedman, Olshen y Stone (1984) introduce las principales ideas para la Estadística. Se desarrollaron tres metodologías concurrentemente en los años 70 [Quinlan, 1993], con lo que se desarrollaron diversos algoritmos de aprendizaje diseñados para la obtención de modelos de árbol de decisión, de los que podemos destacar:

- Algoritmo CHAID (Chi-squared Automatic Interaction Detection), implementado por Kass (1980)
- Algoritmo CART (Classification And Regression Trees) desarrollado por Breiman, Friedman, Losen y Stone (1984)
- Algoritmo ID3 (Iterative Dichotomiser 3) de Quinlan (1986), y sus posteriores evoluciones C4.5 (Quinlan, 1993) y C5.0 (Quinlan, 1997).

A continuación se muestra una tabla comparando algunos de estos algoritmos:

Algoritmos	Vector entrada	Variable salida	Tipo predicción	Ramas por división	Criterio de división
CHAID	Categorico Numérico	Categorica Numérica	Clasificación Regresión	≥ 2	Chi-cuadrado/F
CART	Categorico Numérico	Categorica Numérica	Clasificación Regresión	$= 2$	GINI/ Desviación cuadrática media
C4.5	Categorico Numérico	Categorica	Clasificación	≥ 2	Ganancia proporcional

Una de las ventajas más sobresalientes de los árboles de decisión es su carácter descriptivo, que permite entender e interpretar fácilmente las decisiones tomadas por el modelo, ya que tenemos acceso a las reglas que

se utilizan en la tarea predictiva (aspecto no contemplado en otras técnicas de Minería de Datos). De hecho, es posible derivar fácilmente reglas de decisión (para cada rama terminal) siguiendo las rutas marcadas en la estructura del árbol que llevan a un determinado nodo hoja (la decisión del modelo).

Por tanto, con respecto a la interpretabilidad del modelo generado, los árboles de predicción representa un compromiso entre los dos enfoques presentados en el apartado de contexto de este trabajo.

Por otro lado, las reglas de decisión proporcionadas por un modelo de árbol tienen poder predictivo (no sólo descriptivo) desde el momento en que se evalúa su precisión a partir de unos datos independientes (datos test) a los utilizados en la construcción del modelo (datos de entrenamiento).

Otro rasgo atractivo de los árboles de decisión es que son intrínsecamente robustos a los valores perdidos. No obstante, los árboles de decisión presentan algunas debilidades (Shmueli, Patel y Bruce, 2008): son sensibles a pequeños cambios en los datos y, a diferencia de los modelos que asumen una relación particular entre la respuesta y la predicción (p. ej., una relación lineal como en una regresión lineal), los árboles de decisión son no lineales y no paramétricos. Esto permite una amplia gama de relaciones entre los predictores y la respuesta, pero puede ser una debilidad: dado que las particiones se realizan sobre predictores únicos más que sobre combinaciones de predictores, el árboles de decisión probablemente omite relaciones entre predictores, en particular estructuras lineales como las de los modelos de regresión lineal o logística.

Por tanto de los árboles de decisión se pueden mencionar las siguientes ventajas:

1. Se obtiene conocimiento estructurado en forma de reglas de clasificación o de los valores de una variable de intervalo. Esto facilita interpretar en un lenguaje llano la caracterización de las clases o los valores de una variable de intervalo.
2. Al ser un procedimiento de análisis no paramétrico, no se requiere validar supuestos distribucionales de probabilidad.
3. Permite trabajar con todo tipo de variables predictoras: binarias, nominales, ordinales y de intervalo o razón.
4. Permite valores desconocidos para las variables predictoras en los individuos, tanto en la fase de construcción del árbol como en la de predicción.
5. En el caso de Clasificación se puede establecer probabilidad a priori de las clases.
6. Se puede ponderar las observaciones usando una variable ad-hoc.

El algoritmo de aprendizaje determina los siguientes aspectos:

- Compatibilidad específica con el tipo de variables: naturaleza de las variables de entrada y la variable de salida.
- Procedimiento de evaluación de la distancia entre grupo en cada división: criterio de división.
- Puede imponer restricciones en el número de ramas en que se divide cada nodo.
- Parámetros de poda [prepoda / postpoda]: n° mínimo de registros por nodo o rama, valor crítico del criterio de división, diferencia de rendimiento entre el árbol ampliado y el reducido. La prepoda implica utilizar criterios de parada durante la construcción del árbol, mientras que la post-poda aplica los parámetros de poda al árbol completo.

3. Algoritmos TDIDT

En este capítulo se analizará la familia de métodos de inducción para el entrenamiento de árboles de decisión conocida como TDIDT (Top Down Induction Trees), y en particular en los algoritmos ID3 y C4.5 desarrollados por Quinlan. Además se realizará una prueba de la implementación de este último en la plataforma WEKA, el algoritmo J4.8.

Tanto el ID3 como el C4.5 generan árboles y reglas de decisión a partir de datos preclasificados. Para construir los árboles se utiliza el método de aprendizaje “divide y reinarás” o partición, que particiona el conjunto de ejemplos en subconjuntos a medida que avanza (trabajar sobre cada subconjunto es más sencillo que trabajar sobre el total de los datos).

3.1. Algoritmos de partición

La tarea para la cual los árboles de decisión se adecúan mejor es la clasificación, por su estructura de condición y ramificación. Las clases son disjuntas, y por ello un árbol de clasificación utiliza particiones disjuntas, lo que conduce cada covariable de entrada a una única hoja de salida.

Esta propiedad fue utilizada en los primeros algoritmos para el entrenamiento de árboles de decisión, llamados de *partición*, que se basaron en el principio de “divide y vencerás”: a través de un algoritmo de aprendizaje supervisado se realizan divisiones sucesivas del espacio multi-variable para maximizar la distancia entre grupos en cada división (esto es, realizar particiones que discriminen).

En el proceso recursivo se deben establecer algunos criterios:

1. Cómo son los cortes posibles y un número máximo de cortes determinados por un predictor desde el nodo. Los cortes que se establecen para variables ordinales y de intervalo se realizan por intervalos consecutivos.
2. Una condición de admisibilidad para los cortes posibles.
3. Una medida de contenido de información del árbol respecto al conjunto de individuos o un criterio de optimización de los cortes; es decir, obtener la mejor combinación de cortes admisibles respecto a una variable predictora.
4. Determinar la descripción de la variable objetivo en los nodos del árbol. Para clasificación: El grupo con la mayor representación determina la clase a la que asigna el nodo. En caso de empates se puede elegir cualquiera. Para regresión: En los nodos se estiman las medias muestrales de la variable respuesta condicionadas a los nodos.

En cada iteración los dos puntos más importantes para que el algoritmo funcione bien son:

- Particiones posibles
- Criterio de selección de particiones

Las *particiones posibles* son el conjunto de condiciones exhaustivas y excluyentes que permite el método. Cuando mayor sea más expresivo y preciso será el árbol generado, pero también la complejidad, por lo que la mayoría de los algoritmos permiten sólo un juego muy limitado de particiones nominales o numéricas discretas.

Por muy limitado que sea el juego de particiones posibles, su número puede dispararse; y dado que en los algoritmos de partición una vez elegida la partición se continúa hacia abajo la construcción del árbol, sin volver a plantearse las particiones construidas, entonces la elección de la partición es crítica.

Por ello es crítico en esta técnica la elección de un criterio que ayude a seleccionar la partición más prometedora en cada caso con el menor esfuerzo computacional.

Los *criterios de selección de particiones* o de corte consiste en elegir una medida de la pureza o capacidad de discriminar de la partición, para elegir en cada caso la que optimice la pureza.

Las medidas elegidas se basan principalmente en la frecuencia relativa de las clases en cada uno de los hijos de la partición respecto a las clases en el padre, o en la medida de contenido de información.

El proceso de división finaliza cuando todos los registros de una rama tienen el mismo valor en la variable de salida (nodo hoja puro), dando lugar al modelo completo (máxima especificidad). Cuanto más abajo están las variables de entrada en el árbol, menos importantes son en la clasificación de salida (y menos generalización permiten, debido a la disminución del número de entradas en las ramas descendientes).

Para evitar el sobreajuste del modelo, se puede realizar una poda del árbol que elimine las ramas con pocos registros o poco significativas. En consecuencia, si partimos del modelo completo, tras la poda del árbol éste ganará en capacidad de generalización (evaluada con datos de test), a costa de reducir el grado de pureza de sus hojas (Hernández et al., 2004; Larose, 2005).

La medida de contenido de la información es la suma ponderada de una medida de contenido de la información de las hojas del árbol (y es una función de incertidumbre o entropía aplicada a una distribución de probabilidad), y por tanto de su "pureza".

Al ser ésta una medida aditiva en los nodos, en un paso del algoritmo es suficiente con optimizar el incremento de la medida de contenido de información del árbol en el nodo que se está explorando. En este caso, se obtiene la combinación de cortes que hace máxima la reducción de la incertidumbre en los nodos del árbol.

En el caso de clasificación, sea n_{ik} el número de observaciones de clase k en el nodo terminal i , y sea p_{ik} la proporción observada de clase k en el nodo i . Si llamamos D_i a la medida de información para el nodo i (así que la medida total es $\sum D_i$), hay varias posibilidades de elección para la medida. A continuación mostramos las más comunes:

1. Deviance: $D_i = -2 \sum_k n_{ik} \log p_{ik}$
2. Entropía: $D_i = - \sum_k p_{ik} \log p_{ik}$
3. Índice Gini: $D_i = 1 - \sum_k p_{ik}^2$
4. DKM: $D_i = 2 \left(\prod_k p_{ik} \right)^{1/2}$

La medida elegida diferencia los principales algoritmos:

- Los algoritmos ID3 y C4.5 desarrollados por Quinlan se basan en índices de entropía modificados
- El algoritmo CART se basa en el índice de Gini

Según los experimentos realizados los índices basados en Entropía y DKM se comportan ligeramente mejor que los basados en Gini.

Aunque los árboles de decisión parecen idóneos para clasificación, se han adaptado para otras tareas, como la regresión. Al igual que en el caso de árboles de clasificación, la medida de contenido de la información puede ser una suma de la medida de contenido de información de las hojas o nodos hijos ponderadas por los pesos de los mismos.

En realidad un árbol de regresión se construye de manera similar a un árbol de decisión para clasificación, pero con las siguientes diferencias:

- La función aprendida tiene dominio real y no discreto
- Los nodos hoja del árbol se etiquetan con valores reales

La medida de información más utilizada en los árboles de regresión de tipo partición es la *Suma de Residuos al Cuadrado* (RSS son las siglas en inglés). Se escoge la partición que minimiza la RSS de cada partición posible

Otro criterio de corte considera la estadística F que selecciona como variable de segmentación la que tenga un valor mayor de la estadística F , asociado a

la variable Y condicionada a los nodos hijos dados por los cortes. Este estadístico es una medida de desigualdad de medias, en este caso las medias de Y condicionada a los nodos.

3.2. Algoritmo ID3

El algoritmo ID3 o *Induction Decision Trees*, diseñado en 1993 por J. Ross Quinlan [Quinlan, 1993a], toma objetos de una clase conocida y los describe en términos de una colección fija de propiedades o de variables, produciendo un árbol de decisión sobre estas variables que clasifica correctamente todos los objetos [Quinlan, 1993].

Hay ciertas cualidades que diferencian a este algoritmo de otros sistemas generales de inferencia. La primera se basa en la forma en que el esfuerzo requerido para realizar una tarea de inducción crece con la dificultad de la tarea.

El ID3 fue diseñado específicamente para trabajar con masas de objetos, y el tiempo requerido para procesar los datos crece sólo linealmente con la dificultad, como producto de:

1. La cantidad de objetos presentados como ejemplos
2. la cantidad de variables dadas para describir estos objetos
3. la complejidad del concepto a ser desarrollado (medido por la cantidad de nodos en el árbol de decisión)

Esta linealidad se consigue a costa del poder descriptivo ya que los conceptos desarrollados por el ID3 sólo toman la forma de árboles de decisión basados en las variables dadas, y este lenguaje es mucho más restrictivo que la lógica de primer orden o la lógica multi-valuada, en la cual otros sistemas expresan sus conceptos [Quinlan, 1993].

El ID3 fue presentado como descendiente del CLS y, como contrapartida de su antecesor, es un mecanismo mucho más simple para el descubrimiento de una colección de objetos pertenecientes a dos o más clases. Cada objeto debe estar descrito en términos de un conjunto fijo de variables, cada una de las cuales cuenta con su conjunto de posibles valores. Por ejemplo, la variable humedad puede tener los valores alta, baja y la variable clima, soleado, nublado, lluvioso.

Una regla de clasificación en la forma de un árbol de decisión puede construirse para cualquier conjunto C de variables de esta forma [Quinlan, 1993]:

1. Si C está vacío, entonces se lo asocia arbitrariamente a cualquiera de las clases
2. Si C contiene los representantes de varias clases, se selecciona una variable y se particiona C en conjuntos disjuntos C_1, C_2, \dots, C_n ,

donde C_1 contiene aquellos miembros de C_i que tienen el valor i para la variable seleccionada. Cada una de estos subconjuntos se maneja con la misma estrategia

El resultado es un árbol en el cual cada hoja contiene un nombre de clase y cada nodo interior especifica una variable para ser testeada con una rama correspondiente al valor de la variable.

El objetivo de ID3 es crear una descripción eficiente de un conjunto de datos mediante la utilización de un árbol de decisión. Dados datos consistentes, es decir, sin contradicción entre ellos, el árbol resultante describirá el conjunto de entrada a la perfección. Además, el árbol puede ser utilizado para predecir los valores de nuevos datos, asumiendo siempre que el conjunto de datos sobre el cual se trabaja es representativo de la totalidad de los datos.

Dados:

1. Un conjunto de datos
2. un conjunto de descriptores de cada dato
3. un clasificador/conjunto de clasificadores para cada objeto

Se desea obtener un árbol de decisión simple basándose en la entropía, donde los nodos pueden ser:

1. Nodos intermedios: en donde se encuentran los descriptores escogidos según el criterio de entropía, que determina cuál rama es la que debe tomarse
2. hojas: estos nodos determinan el valor del clasificador

Este procedimiento de formación de reglas funcionará siempre, dado que no existen dos objetos pertenecientes a distintas clases pero con idéntico valor para cada una de sus variables; si este caso llegara a presentarse, las variables son inadecuadas para el proceso de clasificación.

Hay dos conceptos importantes a tener en cuenta en el algoritmo ID3 [B1urock, 1996], la entropía y el árbol de decisión.

- La entropía se utiliza para encontrar el parámetro más significativo en la caracterización de un clasificador
- El árbol de decisión es un medio eficiente e intuitivo para organizar los descriptores que pueden ser utilizados con funciones predictivas.

3.2.1. Limitaciones de ID3

El ID3 puede aplicarse a cualquier conjunto de datos, siempre y cuando las variables sean discretas. Este sistema no cuenta con la facilidad de trabajar con variables continuas ya que analiza la entropía sobre cada uno de los valores de una variable, por lo tanto, tomaría cada valor de una variable continua individualmente en el cálculo de la entropía, lo cual no es útil en

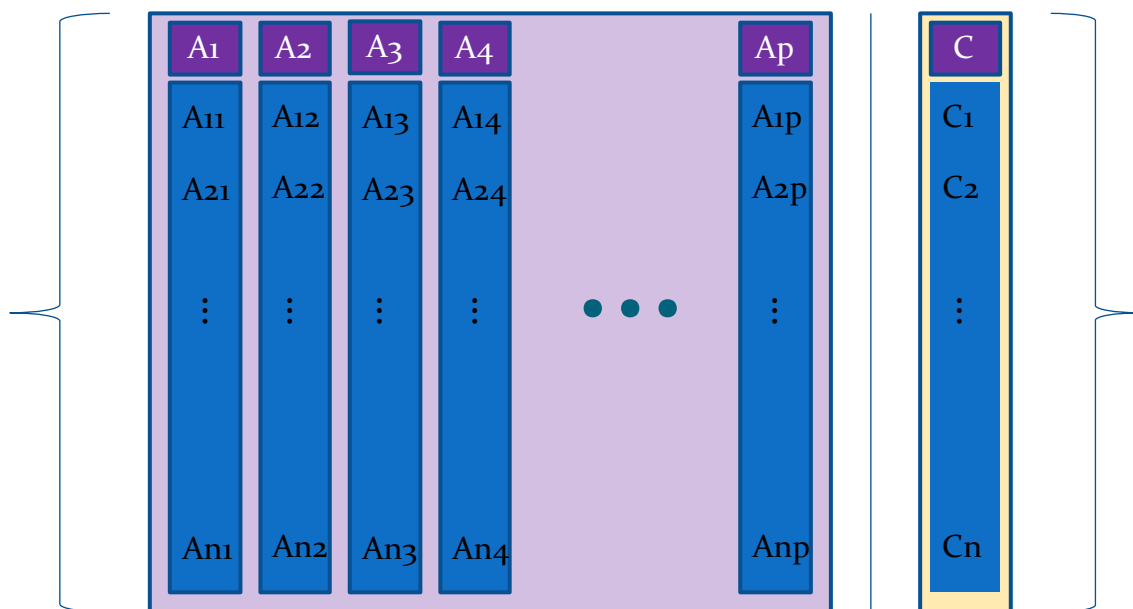
muchos de los dominios. Cuando se trabaja con variables continuas, generalmente se piensa en rangos de valores y no en valores particulares.

Existen varias maneras de solucionar este problema del ID3, como la agrupación de valores presentada en [Gallion et al., 1993] o la discretización de los mismos explicada en [B1urock, 1996; Quinlan, 1993c], que dio lugar al algoritmo C4.5 que se presentará a continuación.

3.2.2. Pseudo-código del algoritmo ID3

A continuación, se presenta el algoritmo del método ID3 para la construcción de árboles de decisión en función de un conjunto de datos previamente clasificados:

Sea $S=\{A|C\}$ un conjunto de entrenamiento de n registros, que tiene un sub-conjunto A de p atributos no clasificadores $A=\{A_j, j=1,...,p\}$, que llamaremos matriz de diseño, y un atributo clasificador C .



Llamamos $A(k)$ a la matriz de diseño resultante tras la iteración $k \in \{0, \dots, p\}$.

Llamamos $\text{Nodo}(k,i)$ el sub-conjunto de registros de $S\{A(k)|C\}$ que tienen el valor $r(k)[i]$ en el atributo $r(k)$, y al que se le elimina el registro

Inicio

- Si $S\{A|C\}$ vacío, devolver Error
- Si todos los registros tienen el mismo valor en C , devolver un único nodo con ese valor.
- Si R vacío, devolver un único nodo con el valor más frecuente de C , avisando de que en este caso habrá registros mal clasificados por este motivo
- Si no, creamos un nodo inicial $N(0,-)$ y llamamos $A(0)=A$

Para cada iteración $k \in \{1, \dots, p\}$.

- En cada nodo $N(k-1, i)$, generado en la iteración $k-1$ (si $k=0$ no hay nodo anterior, y $A(-1)=A$)
 - Calculamos $D(k)=(d_1(k), \dots, d_n(k))^t$, el vector atributo que maximiza la ganancia de información entre los atributos de $S(k)=\{A(k)|C\}$
 - Generamos un nodo que llamamos $N(d(k))$

Si no: $R(k-1)=R$, $S\{R(k-1)|C\}=S\{R|C\}$, $\text{Nodo}(k-1, -)$

Si no:

Sea $R(k) \subset R(k-1)$ el conjunto de $p-k+1$ atributos no clasificadores que quedan tras la iteración $k-1$, donde se seleccionó el atributo $r(k-1)$, con $r(k-1)[1], \dots, r(k-1)[m(k-1)]$ los $m(k-1)$ diferentes valores que puede tomar.

Sea $\text{Nodo}(k-1, i)$ el sub-conjunto de registros de $S\{R(k-1)|C\}$ que tienen el valor $r(k-1)[i]$ en el atributo $r(k-1)$, y al que se le elimina el registro

Para cada nodo $N(k-1, i)$, calcular $r(k)=\arg(j) \max \text{Ganancia}(\text{Nodo}(k-1, i))$

Calcular a partir del nodo actual, el árbol simple con R_k , y que tiene m ramas que van cada una a uno nodo S_i formado por $R' \cap R'$

3.3. Algoritmo C4.5

El C4.5 es una extensión del ID3 que acaba con muchas de sus limitaciones. Por ejemplo, permite trabajar con valores continuos para los atributos, separando los posibles resultados en dos ramas: una para aquellos $A_i \leq N$ y otra para $A_i > N$. Además, los árboles son menos frondosos porque cada hoja no cubre una clase en particular sino una distribución de clases, lo cual los hace menos profundos y menos frondosos. Este algoritmo fue propuesto por Quinlan en 1993.

El C4.5 genera un árbol de decisión a partir de los datos mediante particiones realizadas recursivamente, según la estrategia de profundidad-primero (*depth-first*). Antes de cada partición de datos, el algoritmo considera todas las pruebas posibles que pueden dividir el conjunto de datos y selecciona la prueba que resulta en la mayor ganancia de información o en la mayor proporción de ganancia de información. Para cada atributo discreto, se considera una prueba con n resultados, siendo n el número de valores posibles que puede tomar el atributo. Para cada atributo continuo, se realiza una prueba binaria sobre cada uno de los valores que toma el atributo en los datos.

En cada nodo, el sistema debe decidir cuál prueba escoge para dividir los datos. Los tres tipos de pruebas posibles propuestas por C4.5 son [Quinlan, 1993c]:

1. La prueba "estándar" para las variables discretas, con un resultado y una rama para cada valor posible de la variable
2. una prueba más compleja, basada en una variable discreta, en donde los valores posibles son asignados a un número variable de grupos con un resultado posible para cada grupo, en lugar de para cada valor
3. si una variable A tiene valores numéricos continuos, se realiza una prueba binaria con resultados $A \leq Z$ y $A > Z$, para lo cual debe determinarse el valor límite Z.

Todas estas pruebas se evalúan de la misma manera, mirando el resultado de la proporción de ganancia, o alternativamente, el de la ganancia resultante de la división que producen. Ha sido útil agregar una restricción adicional: para cualquier división, al menos dos de los subconjuntos T_i deben contener un número razonable de casos. Esta restricción, que evita las subdivisiones casi triviales, es tomada en cuenta solamente cuando el conjunto T es pequeño.

Las principales características del algoritmo C4.5 son:

- Permite trabajar con valores continuos para los atributos, separando los posibles resultados en 2 ramas $A_i \leq N$ y $A_i > N$.
- Los árboles son menos frondosos, ya que cada hoja cubre una distribución de clases no una clase en particular.
- Utiliza el método "divide y vencerás" para generar el árbol de decisión inicial a partir de un conjunto de datos de entrenamiento.
- Se basa en la utilización del criterio de proporción de ganancia (gain ratio), definido como $I(X_i, C)/H(X_i)$. De esta manera se consigue evitar que las variables con mayor número de posibles valores salgan beneficiadas en la selección.
- Es Recursivo.

3.3.1. Mejoras de C4.5 respecto a ID3

Evitar sobreajuste (Overfitting)

A medida que se añaden niveles AD, las hipótesis se refinan tanto que describen muy bien los ejemplos utilizados en el aprendizaje, pero el error de clasificación puede aumentar al evaluar los ejemplos. Es decir, clasifica muy bien los datos de entrenamiento pero luego no sabe generalizar al conjunto de prueba. Es debido a que aprende hasta el ruido del conjunto de entrenamiento, adaptándose a las regularidades del conjunto de entrenamiento.

Este efecto es, por supuesto, indeseado. Hay varias causas posibles para que esto ocurra. Las principales son:

- Exceso de ruido (lo que se traduce en nodos adicionales)

- Un conjunto de entrenamiento demasiado pequeño como para ser una muestra representativa de la verdadera función objetivo.

Hay varias estrategias para evitar el sobreajuste en los datos. Pueden ser agrupadas en dos clases:

- Estrategias que frenan el crecimiento del árbol antes de que llegue a clasificar perfectamente los ejemplos del conjunto de entrenamiento.
- Estrategias que permiten que el árbol crezca completamente, y después realizan una poda.

Condicionar la post-poda (post prunnig)

La post-poda es una variante de la poda y es usada por el C4.5. Consiste en una vez generado el árbol completo, plantearse qué es lo que se debe "podar" para mejorar el rendimiento y de paso obtener un árbol más corto.

Pero además el C4.5 convierte el árbol a un conjunto de reglas antes de podarlo. Hay tres razones principales para hacer esto:

- Ayuda a distinguir entre los diferentes contextos en los que se usa un nodo de decisión, debido a que cada camino de la raíz a una hoja se traduce en una regla distinta.
- Deja de existir la distinción entre nodos que están cerca de la raíz y los que están lejos. Así no hay problemas para reorganizar el árbol si se poda un nodo intermedio.
- Mejora la legibilidad. Las reglas suelen ser más fáciles de entender.

Otras mejoras

- Determinar que tan profundo debe crecer el árbol de decisión.
- Reducir errores en la poda.
- Manejar atributos continuos.
- Escoger un rango de medida apropiado.
- Manejo de datos de entrenamiento con valores faltantes.
- Manejar atributos con diferentes valores.
- Mejorar la eficiencia computacional.

3.3.2. Pseudocódigo de C4.5

Sea:

- R: conjunto de atributos no clasificadores,
- C: atributo clasificador,
- S: conjunto de entrenamiento, devuelve un árbol de decisión

Comienzo

- Si S está vacío,
 - o Devolver un único nodo con Valor Falla; 'para formar el nodo raíz

- Si todos los registros de S tienen el mismo valor para el atributo clasificador,
 - o Devolver un único nodo con dicho valor; 'un unico nodo para todos
- Si R está vacío,
 - o Devolver un único nodo con el valor más frecuente del atributo Clasificador en los registros de S [Nota: habrá errores, es decir, registros que no estarán bien clasificados en este caso];
- Si R no está vacío,
- $D \leftarrow$ atributo con mayor Proporción de Ganancia (D,S) entre los atributos de R;
- Sean $\{d_j \mid j=1,2,\dots, m\}$ los valores del atributo D;
- Sean $\{S_j \mid j=1,2,\dots, m\}$ los subconjuntos de S correspondientes a los valores de d_j respectivamente;
- Devolver un árbol con la raíz nombrada como D y con los arcos nombrados d_1, d_2,\dots,d_m , que van respectivamente a los árboles $C4.5(R-\{D\}, C, S_1)$, $C4.5(R-\{D\}, C, S_2)$, $C4.5(R-\{D\}, C, S_m)$;

Fin

3.4. Aplicación de J48 a los datos de prueba

LA Weka (*Gallirallus australis*) es un ave endémica de Nueva Zelanda. Esta Gallinácea en peligro de extinción es famosa por su curiosidad y agresividad. De aspecto pardo y tamaño similar a una gallina las wekas se alimentan fundamentalmente de insectos y frutos.

Este ave da nombre a una extensa colección de algoritmos de Máquinas de conocimiento desarrollados por la universidad de Waikato (Nueva Zelanda) implementados en Java [1, 2]; útiles para ser aplicados sobre datos mediante los interfaces que ofrece o para embeberlos dentro de cualquier aplicación. Además Weka contiene las herramientas necesarias para realizar transformaciones sobre los datos, tareas de clasificación, regresión, clustering, asociación y visualización. Weka está diseñado como una herramienta orientada a la extensibilidad por lo que añadir nuevas funcionalidades es una tarea sencilla.

El clasificador J48 de la plataforma WEKA es una implementación del algoritmo C4.5.

Weka tiene otros algoritmos para la generación de árboles de decisión:

- DecisionStump: árbol con un solo nivel RandomForest y RandomTree: uno o varios árboles construidos con un subconjunto aleatorio de atributos
- RepTree: Algoritmo simplificado con preprda y division de atributos continuos una sola vez

- LMT, NBtree: Árboles de modelos, las hojas se crean mediante otro algoritmo de aprendizaje (regresión logística y Naive Bayes)
- M5P: Árbol de regresión

3.4.1. Fichero de datos

El fichero de datos contiene una encuesta sobre Accidentes en Empresas Mineras. La tabla recoge los resultados de una encuesta en varias empresas mineras, sobre las circunstancias que rodearon la ocurrencia de un suceso (variable S) que puede ser Accidente o Incidente en función de su gravedad. El objetivo del análisis es determinar las condiciones asociadas a uno u otro tipo de suceso con objeto de conocer su casuística y adoptar medidas preventivas en su caso.

		DATOS RELATIVOS AL INCIDENTE					DATOS RELATIVOS AL TRABAJADOR								DATOS RELATIVOS A LA OBRA Y A LA EMPRESA					
ENCUESTA	SUCESO	HORA	DÍA	MES	HORAS DE OPERACIÓN	EDAD	NACIONALIDAD	ANTIGÜEDAD EN MESES	TIPO DE CONTRATO	TIEMPO EN OBRA	PUESTO DE TRABAJO	FORMACIÓN	RECONOCIM. PREVIO DEL PELIGRO	FACTORES PERSONALES	COMUNIDAD AUTÓNOMA	RÉGIMEN	PLAZO DE EJECUCIÓN	EVALUACIÓN DEL RIESGO	CONDICIONES ADECUADAS	DIRECCIÓN Y SUPERVISIÓN
E	S	H	D	M	HO	Ed	Na	An	TC	TO	PT	F	RP	FP	CA	R	PE	ER	CT	DS
M1	A	DU	2	9	10	30	Española	12	OS	>1M	Chófer	G	No	Sí	Castilla y León	Sb	D	Sí	Sí	R
M2	I	DU	5	9	10	55	Española		OS	>1M	Mecánico	G	Sí	Sí	Asturias	CP		Sí	No	R
M3	I	DC	2	10	6	22	Marroquí	1	OS	S/M	Ayudante mecánico	G	No	Sí	Aragón	CP		Sí	Sí	SR
M4	A	PH	2	12	3	28	Búlgara	7	OS	<1S	Chófer	G	Sí	Sí	Comunidad Valenciana	CP	F	No	No	R
M5	A	DC	1	12	10	38	Española	8	OS	>1M		G/E	No	No	Castilla y León	Sb	D	Sí	No	SR
M6	A	PH	1	1	2	37	Española	9	OS	S/M	Peón especialista	G	No	No	Comunidad Valenciana	CP	D	Sí	Sí	R
M7	I	DU	4	1	2	41	Española	48	In	>1M	Jefe de equipo	G/E	No	No	Castilla y León	Sb		Sí	Sí	R
M8	A	DC	1	1	3	22	Búlgara	2	OS	S/M	Chófer	N	No	No	Comunidad Valenciana	CP	D	Sí	Sí	R
M9	A	DC	5	1	1	45	Española	14	In	>1M	Maquinista	G	Sí	No	Asturias	CP	F	No	Sí	R
M10	A	DC	2	1	1	34	Búlgara	2	OS	<1S	Chófer	N	No	Sí	Comunidad Valenciana	CP	D	Sí	No	R
M11	A	DC	1	1	9	29	Colombiana	3	OS	>1M	Chófer	G/E	No	Sí	Castilla y León	Sb		No	No	R
M12	A	PH	1	2	2	41	Libia	3	OS	<1S	Chófer	G	No	Sí	Comunidad Valenciana	CP	D	No	No	R
M13	A	HE	6	2	9,5	44	Española	17	OS	>1M	Oficial de 1ª conductor	G	No	No	Asturias	Sb		Sí	Sí	R
M14	A	DU	5	3	9	42	Polaca	6	OS	>1M	Maquinista de compactador	G/E	No	No	Castilla y León	Sb		Sí	Sí	R
G1	I	DC	5	9	10	39	Española	20	In	>1M	Chófer	G	No	No	Aragón	CP		Sí	Sí	R
G2	I	DU	2	9	10	44	Española	12	OS	>1M	Chófer	G	No	No	Aragón	CP		No	Sí	R
G3	I	DU	4	9	10	32	Española	20	In	>1M	Motonivelista	G	Sí	No	Aragón	CP		Sí	No	R
G4	I	DU	1	9	10	20	Española	5	OS	>1M	Ayudante mecánico	G	Sí	No	Aragón	CP		No	No	R
G5	I	DC	1	10	10	24	Española		OS	>1M	Retrista	G	No	Sí	Aragón	CP		Sí	Sí	R
G6	I	DC	6	10	8	24	Española	5,5	OS	>1M	Perforista	G	No	No	Aragón	CP		Sí	Sí	S/C
G7	I	DC	1	10	10	25	Armenia	14	OS	>1M	Chófer	G	No	No	Aragón	CP		Sí	No	R
G8	I	PH	2	11	8	25	Española	6	T	>1M	Administrativo	G	No	No	Aragón	CP		No	Sí	R
G9	I	DC	4	11	10	45	Española	20	In	>1M	Buldocerista	G	No	No	Aragón	CP		Sí	Sí	R
G10	A	DU	2	11	10	39	Rumana	8	OS	>1M	Chófer	G	No	No	Aragón	CP		Sí	Sí	R
G11	I	Ot	5	11	10	26	Marroquí	8	OS	>1M	Ayudante perforista	G	Sí	No	Aragón	CP		No	No	SR
G12	I	DU	3	11	10	28	Española	10	OS	>1M	Mecánico	G	Sí	No	Aragón	CP		No	Sí	R
G13	I	DC	1	11	10	26	Española	8	OS	>1M	Administrativo	G	No	No	Aragón	CP		Sí	Sí	R
G14	I	PH	1	11	10	20	Rumana	10	OS	>1M	Ayudante perforista	G	Sí	No	Aragón	CP		No	Sí	R
G15	A	HE	4	11	10	42	Española	16	In	>1M	Ayudante mecánico	G	No	No	Aragón	CP		No	Sí	R
G16	I	Ot	3	12	8	34	Española	15	In	>1M	Chófer	G	No	No	Aragón	CP		Sí	Sí	R
G17	I	Ot	4	12	10	29	Marroquí	8	OS	>1M	Ayudante perforista	G	Sí	No	Aragón	CP		No	No	R
G18	I	Ot	3	12	10	34	Libia	10	OS	>1M	Chófer	G	No	No	Aragón	CP		Sí	Sí	R
G19	I	Ot	1	12	10	30	Ecuatoriana	3	OS	>1M	Ayudante perforista	G	No	No	Aragón	CP		Sí	Sí	R
G20	I	Ot	3	12	10	37	Española	15	In	>1M	Buldocerista	G	No	No	Aragón	CP		Sí	Sí	R
G21	I	Ot	1	12	10	29	Ecuatoriana	11	OS	>1M	Chófer	G	No	No	Aragón	CP		Sí	Sí	R
G22	A	DC	5	12	10	54	Española	60	In	>1M	Perforista	G	No	No	Aragón	CP		No	Sí	R
G23	I	DC	2	12	10	39	Española	18	In	>1M	Motonivelista	G	No	No	Aragón	CP		Sí	Sí	R
G24	I	Ot	6	12	8	33	Española	20	In	>1M	Mecánico	G/E	No	No	Aragón	CP		Sí	Sí	R
G25	I	DU	5	1	10	30	Española	12	OS	>1M	Retrista	G/E	No	No	Aragón	CP		Sí	Sí	R
G26	I	DC	2	1	10	46	Española	30	In	>1M	Retrista	G	No	No	Aragón	CP		Sí	Sí	R
G27	I	Ot	5	1	10	33	Española	17	OS	>1M	Chófer	G	No	No	Aragón	CP		Sí	Sí	R
G28	A	DC	6	1	9	38	Española	28	In	>1M	Perforista	G	No	No	Aragón	CP		No	Sí	R
G29	I	DC	2	1	10	33	Española	14	OS	>1M	Perforista	G	Sí	No	Aragón	CP		No	Sí	R
G30	I	Ot	4	1	10	32	Española	18	OS	>1M	Motonivelista	G	No	No	Aragón	CP		No	Sí	R
G31	I	DC	4	1	10	30	Marroquí	8	OS	>1M	Ayudante mecánico	G	No	No	Aragón	CP		Sí	Sí	R
G32	I	Ot	2	1	10	41	Española	30	In	>1M	Encargado	G/E	No	No	Aragón	CP		Sí	Sí	S/C
G33	I	Ot	3	2	10	29	Ecuatoriana	10	OS	>1M	Chófer	G	No	Sí	Aragón	CP		Sí	Sí	R
G34	A	Ot	1	2	10	52	Chilena	48	OS	>1M	Perforista	G	No	Sí	Aragón	CP		No	Sí	R
G35	I	DC	5	2	10	32	Española	8	OS	>1M	Chófer	G	No	No	Aragón	CP		Sí	Sí	R
G36	I	DU	4	2	10	25	Búlgara	1	OS	S/M	Ayudante mecánico	G	Sí	No	Aragón	CP		Sí	Sí	R
G37	I	Ot	1	2	10	28	Española	14	OS	>1M	Perforista	G	Sí	No	Aragón	CP		No	Sí	R
G38	I	DC	2	2	10	40	Rumana	7	OS	>1M	Retrista	G	No	No	Aragón	CP		Sí	Sí	R
G39	I	Ot	3	2	10	58	Española	180	In	>1M	Mecánico	G/E	Sí	No	Aragón	CP		No	Sí	R
G40	I	Ot	5	3	8	25	Española	10	T	>1M	Administrativo	G	No	No	Aragón	CP		Sí	Sí	R
G41	I	PH	1	3	10	25	Armenia	19	OS	>1M	Chófer	G/E	No	No	Aragón	CP		Sí	Sí	R
G42	I	DC	1	3	10	29	Ecuatoriana	19	OS	>1M	Chófer	G	No	Sí	Aragón	CP		Sí	Sí	R
G43	A	Ot	4	3	9	33	Española	23	In	>1M	Mecánico	G/E	Sí	No	Aragón	CP		Sí	Sí	R
G44	I	Ot	5	3	10	45	Española	24	In	>1M	Buldocerista	G/E	No	No	Aragón	CP		Sí	Sí	R
G45	A	Ot	4	9	10	45	Española	20	In	>1M		G	No	No	Asturias	CP		No	Sí	SR
G46	I	Ot	3	3	10	20	Rumana	14	OS	>1M	Ayudante perforista	G/E	No	No	Aragón	CP		Sí	Sí	R
G47	I	Ot	2	3	10	39	Española	23	In	>1M	Motonivelista	G/E	No	No	Aragón	CP		No	Sí	R
G48	I	Ot	1	3	10	25	Española	11	OS	>1M	Perforista	G/E	No	No	Aragón	CP		Sí	Sí	S/C

3.4.2. Pre-procesado

Un vez seleccionado en Weka el fichero de datos se realiza la etapa de pre-procesado con la tabla descrita anteriormente, elegimos la pestaña "Tabla simplificada" siguiente:

E	S	H	D	M	HO	Ed	Na	An	TC	TO	PT	F	RP	FP	CA	R	ER	CT	DS
M1	A	DU	M	S	10	30	Nac	1,00	OS	>1M	PV	G	No	Si	CasLe	Sb	Si	Si	R
M2	I	DU	V	S	10	55	Nac	0,10	OS	>1M	OP	G	Si	Si	Ast	CP	Si	No	R
M3	I	DC	M	O	6	22	Afr	0,08	OS	S/M	OP	G	No	Si	Ar	CP	Si	Si	SR
M4	A	PH	M	D	3	28	EE	0,58	OS	<1S	PV	G	Si	Si	CVal	CP	No	No	R
M5	A	DC	L	D	10	38	Nac	0,67	OS	>1M	SD	G/E	No	No	CasLe	Sb	Si	No	SR
M6	A	PH	L	E	2	37	Nac	0,75	OS	S/M	OP	G	No	No	CVal	CP	Si	Si	R
M7	I	DU	J	E	2	41	Nac	4,00	In	>1M	PSM	G/E	No	No	CasLe	Sb	Si	Si	R
M8	A	DC	L	E	3	22	EE	0,17	OS	S/M	PV	N	No	No	CVal	CP	Si	Si	R
M9	A	DC	V	E	1	45	Nac	1,17	In	>1M	PMG	G	Si	No	Ast	CP	No	Si	R
M10	A	DC	M	E	1	34	EE	0,17	OS	<1S	PV	N	No	Si	CVal	CP	Si	No	R
M11	A	DC	L	E	9	29	LA	0,25	OS	>1M	PV	G/E	No	Si	CasLe	Sb	No	No	R
M12	A	PH	L	F	2	41	Afr	0,25	OS	<1S	PV	G	No	Si	CVal	CP	No	No	R
M13	A	HE	S	F	9,5	44	Nac	1,42	OS	>1M	PV	G	No	No	Ast	Sb	Si	Si	R
M14	A	DU	V	Mz	9	42	EE	0,50	OS	>1M	PMG	G/E	No	No	CasLe	Sb	Si	Si	R
G1	I	DC	V	S	10	39	Nac	1,67	In	>1M	PV	G	No	No	Ar	CP	Si	Si	R
G2	I	DU	M	S	10	44	Nac	1,00	OS	>1M	PV	G	No	No	Ar	CP	No	Si	R
G3	I	DU	J	S	10	32	Nac	1,67	In	>1M	PV	G	Si	No	Ar	CP	Si	No	R
G4	I	DU	L	S	10	20	Nac	0,42	OS	>1M	OP	G	Si	No	Ar	CP	No	No	R
G5	I	DC	L	O	10	24	Nac	0,10	OS	>1M	PMG	G	No	Si	Ar	CP	Si	Si	R
G6	I	DC	S	O	8	24	Nac	0,46	OS	>1M	PMG	G	No	No	Ar	CP	Si	Si	S/C
G7	I	DC	L	O	10	25	EE	1,17	OS	>1M	PV	G	No	No	Ar	CP	Si	No	R
G8	I	PH	M	N	8	25	Nac	0,50	T	>1M	PSM	G	No	No	Ar	CP	No	Si	R
G9	I	DC	J	N	10	45	Nac	1,67	In	>1M	PV	G	No	No	Ar	CP	Si	Si	R
G10	A	DU	M	N	10	39	EE	0,67	OS	>1M	PV	G	No	No	Ar	CP	Si	Si	R
G11	I	Ot	V	N	10	26	Afr	0,67	OS	>1M	PMG	G	Si	No	Ar	CP	No	No	SR
G12	I	DU	Mx	N	10	28	Nac	0,83	OS	>1M	OP	G	Si	No	Ar	CP	No	Si	R
G13	I	DC	L	N	10	26	Nac	0,67	OS	>1M	PSM	G	No	No	Ar	CP	Si	Si	R
G14	I	PH	L	N	10	20	EE	0,83	OS	>1M	PMG	G	Si	No	Ar	CP	No	Si	R
G15	A	HE	J	N	10	42	Nac	1,33	In	>1M	OP	G	No	No	Ar	CP	No	Si	R
G16	I	Ot	Mx	D	8	34	Nac	1,25	In	>1M	PV	G	No	No	Ar	CP	Si	Si	R
G17	I	Ot	J	D	10	29	Afr	0,67	OS	>1M	PMG	G	Si	No	Ar	CP	No	No	R
G18	I	Ot	Mx	D	10	34	Afr	0,83	OS	>1M	PV	G	No	No	Ar	CP	Si	Si	R
G19	I	Ot	L	D	10	30	LA	0,25	OS	>1M	PMG	G	No	No	Ar	CP	Si	Si	R
G20	I	Ot	Mx	D	10	37	Nac	1,25	In	>1M	PV	G	No	No	Ar	CP	Si	Si	R
G21	I	Ot	L	D	10	29	LA	0,92	OS	>1M	PV	G	No	No	Ar	CP	Si	Si	R
G22	A	DC	V	D	10	54	Nac	5,00	In	>1M	PMG	G	No	No	Ar	CP	No	Si	R
G23	I	DC	M	D	10	39	Nac	1,50	In	>1M	PV	G	No	No	Ar	CP	Si	Si	R
G24	I	Ot	S	D	8	33	Nac	1,67	In	>1M	OP	G/E	No	No	Ar	CP	Si	Si	R
G25	I	DU	V	E	10	30	Nac	1,00	OS	>1M	PMG	G/E	No	No	Ar	CP	Si	Si	R
G26	I	DC	M	E	10	46	Nac	2,50	In	>1M	PMG	G	No	No	Ar	CP	Si	Si	R
G27	I	Ot	V	E	10	33	Nac	1,42	OS	>1M	PV	G	No	No	Ar	CP	Si	Si	R
G28	A	DC	S	E	9	38	Nac	2,33	In	>1M	PMG	G	No	No	Ar	CP	No	Si	R
G29	I	DC	M	E	10	33	Nac	1,17	OS	>1M	PMG	G	Si	No	Ar	CP	No	Si	R
G30	I	Ot	J	E	10	32	Nac	1,50	OS	>1M	PV	G	No	No	Ar	CP	No	Si	R
G31	I	DC	J	E	10	30	Afr	0,67	OS	>1M	OP	G	No	No	Ar	CP	Si	Si	R
G32	I	Ot	M	E	10	41	Nac	2,50	In	>1M	PSM	G/E	No	No	Ar	CP	Si	Si	S/C
G33	I	Ot	Mx	F	10	29	LA	0,83	OS	>1M	PV	G	No	Si	Ar	CP	Si	Si	R
G34	A	Ot	L	F	10	52	LA	4,00	OS	>1M	PMG	G	No	Si	Ar	CP	No	Si	R
G35	I	DC	V	F	10	32	Nac	0,67	OS	>1M	PV	G	No	No	Ar	CP	Si	Si	R
G36	I	DU	J	F	10	25	EE	0,08	OS	S/M	OP	G	Si	No	Ar	CP	Si	Si	R
G37	I	Ot	L	F	10	28	Nac	1,17	OS	>1M	PMG	G	Si	No	Ar	CP	No	Si	R
G38	I	DC	M	F	10	40	EE	0,58	OS	>1M	PMG	G	No	No	Ar	CP	Si	Si	R
G39	I	Ot	Mx	F	10	58	Nac	15,00	In	>1M	OP	G/E	Si	No	Ar	CP	No	Si	R
G40	I	Ot	V	Mz	8	25	Nac	0,83	T	>1M	PSM	G	No	No	Ar	CP	Si	Si	R
G41	I	PH	L	Mz	10	25	EE	1,58	OS	>1M	PV	G/E	No	No	Ar	CP	Si	Si	R
G42	I	DC	L	Mz	10	29	LA	1,58	OS	>1M	PV	G	No	Si	Ar	CP	Si	Si	R
G43	A	Ot	J	Mz	9	33	Nac	1,92	In	>1M	OP	G/E	Si	No	Ar	CP	Si	Si	R
G44	I	Ot	V	Mz	10	45	Nac	2,00	In	>1M	PV	G/E	No	No	Ar	CP	Si	Si	R
G45	A	Ot	J	S	10	45	Nac	20,00	In	>1M	SD	G	No	No	Ast	CP	No	Si	SR
G46	I	Ot	Mx	Mz	10	20	EE	1,17	OS	>1M	PMG	G/E	No	No	Ar	CP	Si	Si	R
G47	I	Ot	M	Mz	10	39	Nac	1,92	In	>1M	PV	G/E	No	No	Ar	CP	No	Si	R
G48	I	Ot	L	Mz	10	25	Nac	0,92	OS	>1M	PMG	G/E	No	No	Ar	CP	Si	Si	S/C

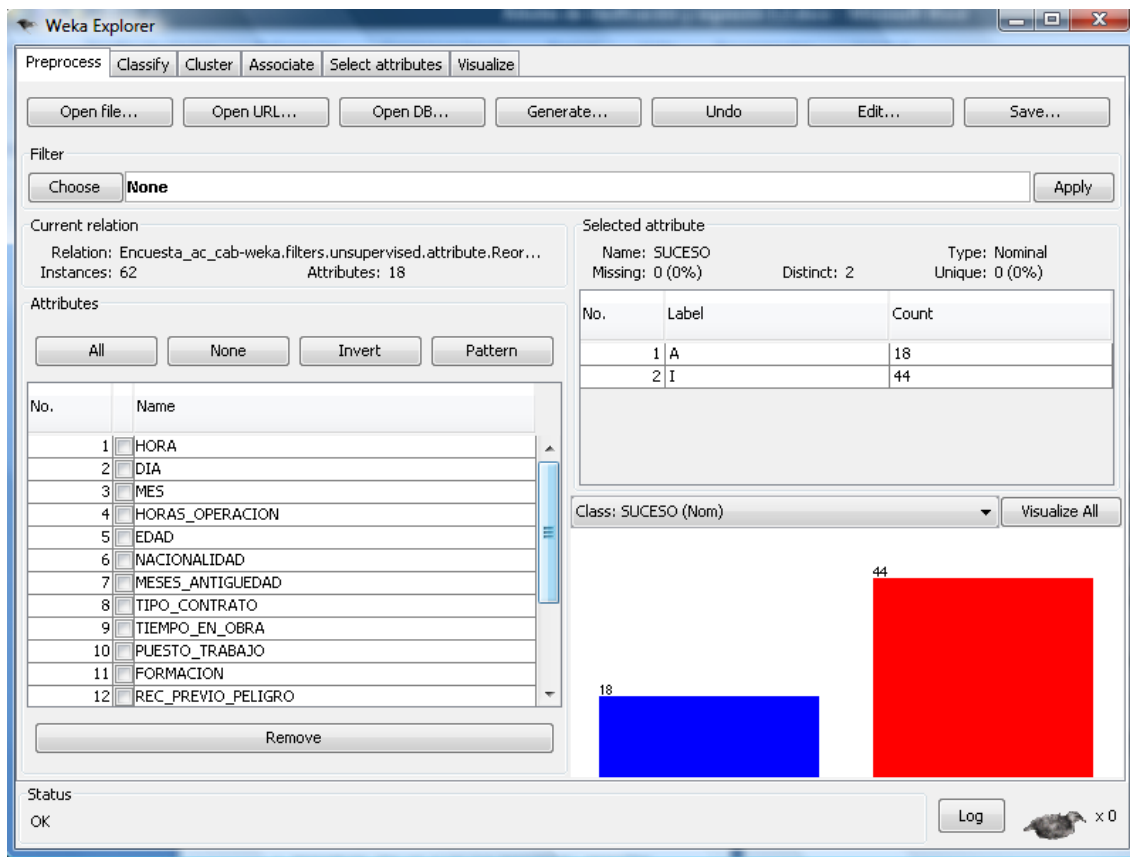
Puede comprobarse que esta tabla ya incluye una serie de tareas de pre-procesado iniciales:

- En esta tabla se han “discretizado” los atributos numéricos, tal y como se muestra en la pestaña de etiquetas, que incluimos a continuación.
- Se ha eliminado el atributo “Plazo de ejecución” por la abundancia de datos ausentes.
- En esta tabla también se han estimado los tres datos faltantes adicionales, dos del atributo “Meses de antigüedad” y dos del atributo “Puesto de trabajo” (que se aparecen en fuente roja), como en la antigüedad del trabajador se asigna el valor 0.1 años si no se dispone del dato real.

SIGNIFICADO DE LAS ETIQUETAS:

SUCESO (S)	A	Accidente		
	I	Incidente		
HORA (H)	PH	Primera hora de la mañana		
	DC	Después de comer		
	DU	Dos últimas horas		
	HE	Horas extras		
	Ot	Otras horas del día		
DÍA (D)	L	Lunes		
	M	Martes		
	Mx	Miércoles		
	J	Jueves		
	V	Viernes		
	S	Sábado		
	D	Domingo		
MES (M)	E	Enero		
	F	Febrero		
	Mz	Marzo		
	Ab	Abril		
	My	Mayo		
	Jn	Junio		
	Jl	Julio		
	Ag	Agosto		
	S	Septiembre		
	O	Octubre		
	N	Noviembre		
	D	Diciembre		
NACIONALIDAD (Na)	Afr	Africanos	Lib	Libia
			Marr	Marroquí
	EE	Europa del Este	Arm	Armenia
			Bul	Búlgara
			Pol	Polaca
			Rum	Rumana
	LA	Latinoamericanos	Chil	Chilena
			Col	Colombiana
	Nac	Nacionales (Españoles)	Ecu	Ecuatoriana
			Esp	Española
TIPO DE CONTRATO (TC)	OS	Obra y servicio		
	T	Temporal		
	In	Indefinido		
TIEMPO EN OBRA (TO)	<1S	Menos de una semana		
	S/M	Entre una semana y un mes		
	>1M	Más de un mes		
PUESTO DE TRABAJO (PT)	PMG	Puestos de trabajo que implican uso de maquinaria de gran	AyPerf	Ayudante perforista
			Maq	Maquinista
			Perf	Perforista
			Retr	Retrista
	PV	Puestos de trabajo en vehículos	Buld	Buldocerista
			Chof	Chófer
			Mot	Motonivelista
	PSM	Puestos de trabajo de oficina o sin uso de maquinaria	Adm	Administrativo
			Enc	Encargado
			JeEq	Jefe de equipo
	OP	Otros puestos de trabajo (electricistas, mecánicos,...)	AyMec	Ayudante mecánico
			Mec	Mecánico
			PeEs	Peón especialista
	SD	Sin determinar		
FORMACIÓN (F)	G/E	Formación genérica y específica		
	G	Formación genérica		
	N	Ninguna		
COMUNIDAD AUTÓNOMA (CA)	CasLe	Castilla y León		
	Ast	Asturias		
	CVal	Comunidad Valenciana		
	Ar	Aragón		
RÉGIMEN (R)	Sb	Subcontrata		
	CP	Contrata principal		
PLAZO DE EJECUCIÓN (PE)	D	Dentro de plazo de ejecución		
	F	Fuera de plazo de ejecución		
DIRECCIÓN Y SUPERVISIÓN (DS)	S/C	Con supervisión y/o control de las condiciones de trabajo		
	R	Sólo recursos preventivos sin acciones de control y/o supervisión		
	SR	Sin recursos preventivos		

A continuación realizaremos en la pestaña correspondiente de Weka, que mostramos a continuación, el resto de tareas de pre-procesado no incluidas en el fichero inicial:

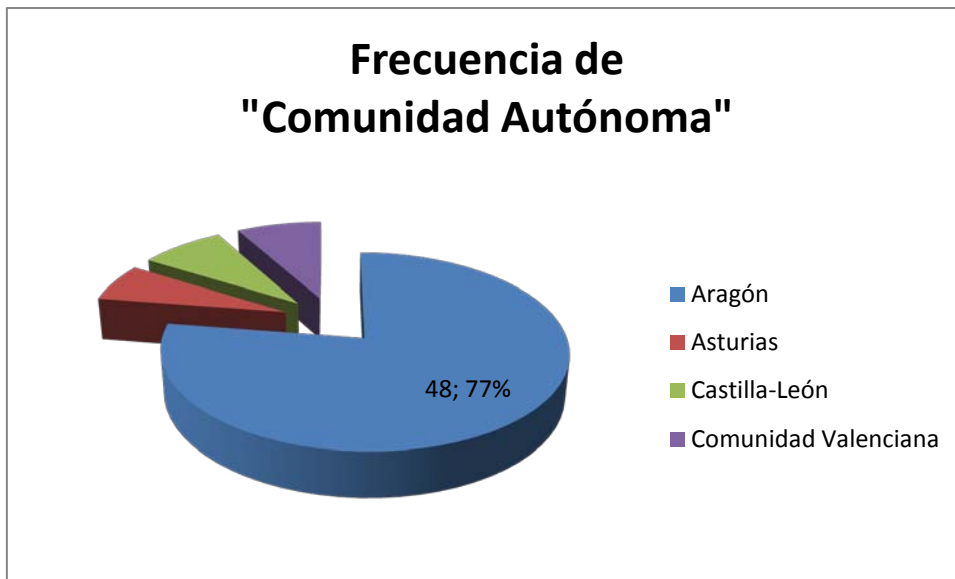


La tareas que se realizan son:

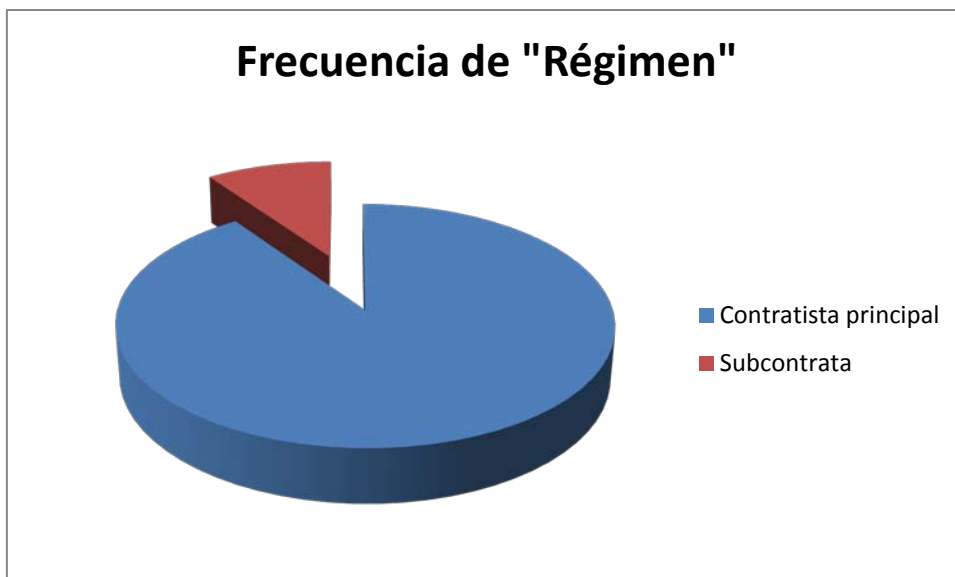
- La eliminación del estudio el atributo "Encuesta", que es un identificador de cada registro y por tanto no tiene ningún valor predictivo
- La selecciona del atributo "Suceso" como clase respuesta.

Además el estudio propone analizar la posible eliminación de dos variables explicativas más en función de los datos iniciales: "Comunidad Autónoma" y "Régimen".

En el caso de "Comunidad Autónoma" la encuesta recoge datos de cuatro Comunidades Autónomas de las 17 posibles, que es un 20% de variabilidad poblacional; que unido a que la moda muestral, "Aragón" concentra el 77% de los registros (48 de 62), tal y como veremos en la siguiente gráfica, hace muy poco representativa la muestra respecto a esta variable, con lo que decidimos eliminarla del estudio.

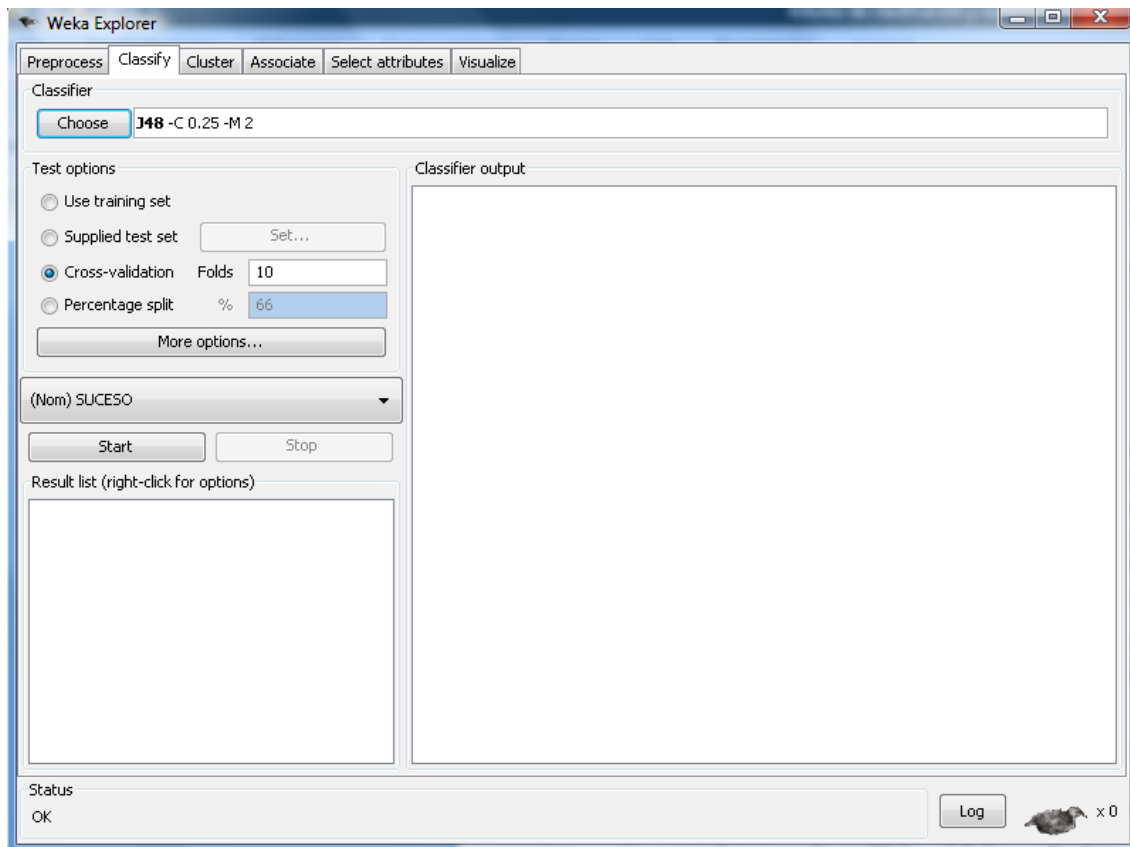


Muy distinto nos parece el caso del término "Régimen", en el que aunque la variabilidad es muy pequeña tal y como se muestra en la gráfica siguiente, la encuesta sí recoge registros de los dos valores posibles, por lo que decidimos mantenerla en el estudio.



3.4.3. Interfaz de usuario y parámetros del algoritmo

Una vez realizada primera fase de pre-procesado se elige el algoritmo de clasificación J48 de la pestaña de Clasificadores de Weka, con los parámetros y opciones de validación por defecto.

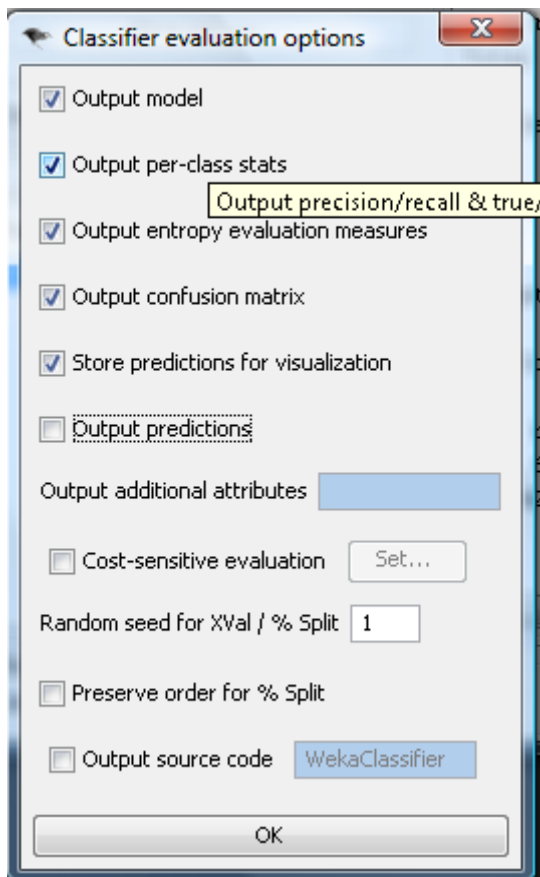


Hay cuatro opciones de validación:

- **Use training set:** Con esta opción Weka entrenará el método con todos los datos disponibles y luego lo aplicará otra vez sobre los mismos.
- **Supplied test set:** Marcando esta opción tendremos la oportunidad de seleccionar, pulsando el botón Set . . . , un fichero de datos con el que se probará el clasificador obtenido con el método de clasificación usado y los datos iniciales.
- **Cross-validation:** Pulsando el botón Cross-validation Weka realizará una validación cruzada estratificada del número de particiones dado (Folds). La validación cruzada consiste en: dado un número n se divide los datos en n partes y, por cada parte, se construye el clasificador con las $n-1$ partes restantes y se prueba con esa. Así por cada una de las n particiones. Una validación-cruzada es estratificada cuando cada una de las partes conserva las propiedades de la muestra original (porcentaje de elementos de cada clase).
- **Percentage Split:** Se define un porcentaje con el que se construirá el clasificador y con la parte restante se probará.

Nosotros en el estudio utilizaremos el método de Validación-Cruzada.

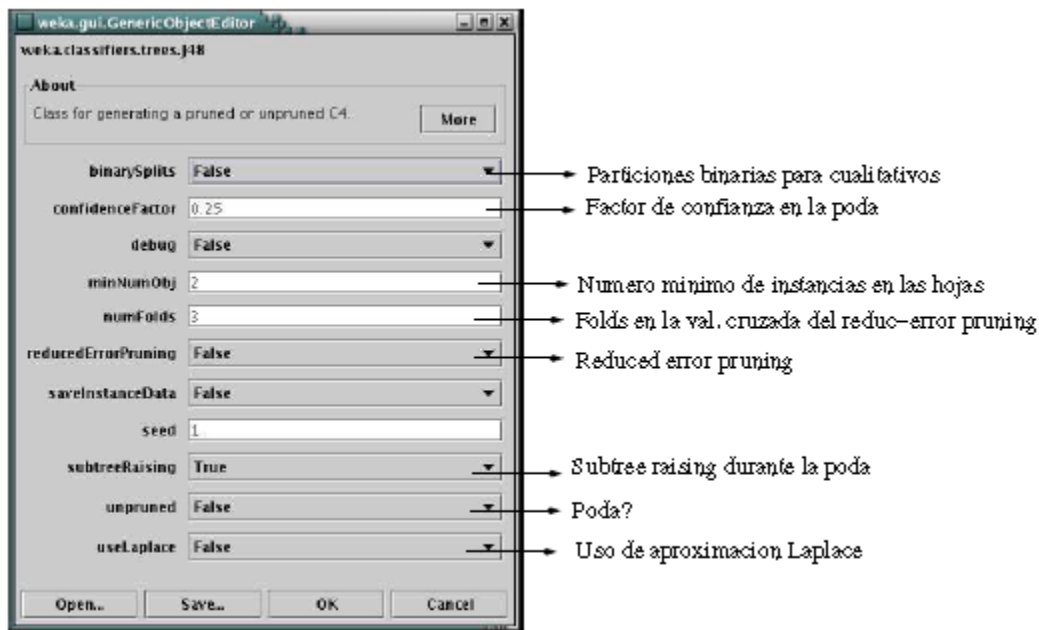
Una vez definido el método de prueba Weka nos permite seleccionar algunas opciones más con el botón More Options:



Las opciones que se nos presentan son:

- Output Model Si la activamos una vez construido y probado el clasificador, nos mostrará en la salida
- del clasificador (parte media derecha de la ventana 9) el modelo que ha construido.
- Output per-class stats Activada muestra estadísticas referentes a cada clase.
- Output entropy evaluation
- Output confusion matrix Muestra la matriz de confusión del clasificador. Esta tabla cuyo número de columnas es el número de atributos muestra la clasificación de las instancias. Da una información muy útil porque no sólo refleja los errores producidos sino también informa del tipo de éstos.

Por último seleccionaremos los parámetros del algoritmo, que explicamos a continuación junto con los parámetros por defecto:



3.4.4. Ejecución y datos de salida del algoritmo

A continuación realizaremos una primera ejecución del algoritmo utilizando los parámetros por defecto, que aprovecharemos para explicar la salida de la ejecución de este algoritmo de Weka.

3.4.4.1. Factor de confianza para la poda = 0.25

=== Run information ===

Scheme: `weka.classifiers.trees.J48graft -C 0.25 -M 2 -A`

Relation: `Encuesta_ac_cab-weka.filters.unsupervised.attribute.Reorder-R2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,1`

Instances: 62

Attributes: 18

HORA

DIA

MES

HORAS_OPERACION

EDAD

NACIONALIDAD

MESES_ANTIGUEDAD

TIPO_CONTRATO
TIEMPO_EN_OBRA
PUESTO_TRABAJO
FORMACION
REC_PREVIO_PELIGRO
FACTORES_PERSONALES
REGIMEN
EVALUACION_RIESGO
CONDICIONES_ADECUADAS
DIRECCION_SUPERVISION
SUCESO

Test mode: 10-fold cross-validation

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

J48graft pruned tree

REGIMEN = Sb: A (6.0/1.0)

REGIMEN = CP

| HORAS_OPERACION <= 3: A (6.0)

| HORAS_OPERACION > 3: I (50.0/7.0)

Number of Leaves : 3

Size of the tree : 5

Time taken to build model: 0 seconds

=== Stratified cross-validation ===

=== Summary ===

<i>Correctly Classified Instances</i>	<i>50</i>	<i>80.6452 %</i>
<i>Incorrectly Classified Instances</i>	<i>12</i>	<i>19.3548 %</i>
<i>Kappa statistic</i>	<i>0.5144</i>	
<i>K&B Relative Info Score</i>	<i>2231.8001 %</i>	
<i>K&B Information Score</i>	<i>19.4768 bits</i>	<i>0.3141 bits/instance</i>
<i>Class complexity order 0</i>	<i>54.0845 bits</i>	<i>0.8723 bits/instance</i>
<i>Class complexity scheme</i>	<i>45.1089 bits</i>	<i>0.7276 bits/instance</i>
<i>Complexity improvement (Sf)</i>	<i>8.9755 bits</i>	<i>0.1448 bits/instance</i>
<i>Mean absolute error</i>	<i>0.2713</i>	
<i>Root mean squared error</i>	<i>0.3939</i>	
<i>Relative absolute error</i>	<i>65.2231 %</i>	
<i>Root relative squared error</i>	<i>86.5802 %</i>	
<i>Total Number of Instances</i>	<i>62</i>	

=== Detailed Accuracy By Class ===

<i>Class</i>	<i>TP Rate</i>	<i>FP Rate</i>	<i>Precision</i>	<i>Recall</i>	<i>F-Measure</i>	<i>ROC Area</i>
<i>A</i>	<i>0.611</i>	<i>0.114</i>	<i>0.688</i>	<i>0.611</i>	<i>0.647</i>	<i>0.726</i>
<i>I</i>	<i>0.886</i>	<i>0.389</i>	<i>0.848</i>	<i>0.886</i>	<i>0.867</i>	<i>0.726</i>
<i>Weighted Avg.</i>	<i>0.806</i>	<i>0.309</i>	<i>0.801</i>	<i>0.806</i>	<i>0.803</i>	<i>0.726</i>

=== Confusion Matrix ===

```
a b <-- classified as
```

```
11 7 | a = A
```

```
5 39 | b = I
```

El primero grupo de información de la salida del algoritmo es la información de ejecución:

```
=== Run information ===
```

La prima línea de este grupo en esta ejecución será:

```
Scheme: weka.classifiers.trees.J48graft -C 0.25 -M 2 -A
```

Muestra el algoritmo utilizado, el clasificador de árbol J48, y los parámetros utilizados, en este caso que el factor de confianza para la poda es 0.25, que usa particiones binarias y que restringe el mínimo número de instancias a hojas de dos.

La segunda línea:

```
Relation: Encuesta_ac_cab-weka.filters.unsupervised.attribute.Reorder-  
R2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,1
```

muestra información acerca de la relación. Las relaciones en Weka son como ficheros de datos, y el nombre de las relaciones contiene el nombre del fichero de datos utilizado para construirlo. También se mostrarían los filtros aplicados en el preprocesado, que en este caso no se ha utilizado ninguno.

A continuación se muestra el número de registros de la muestra y el número y nombre de los términos:

```
Instances: 62
```

```
Attributes: 18
```

```
HORA
```

```
DIA
```

```
MES
```

```
HORAS_OPERACION
```

```
EDAD
```

```
NACIONALIDAD
```

```
MESES_ANTIGUEDAD
```

TIPO_CONTRATO
TIEMPO_EN_OBRA
PUESTO_TRABAJO
FORMACION
REC_PREVIO_PELIGRO
FACTORES_PERSONALES
REGIMEN
EVALUACION_RIESGO
CONDICIONES_ADECUADAS
DIRECCION_SUPERVISION
SUCESO

El último apartado de este grupo:

Test mode: 10-fold cross-validation

muestra que se ha utilizado la Validación Cruzada Estratificada con 10 carpetas.

El siguiente grupo de la salida del algoritmo se muestra el modelo generado indicando que se ha usado toda la muestra como conjunto de entrenamiento y poda:

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

J48graft pruned tree

REGIMEN = Sb: A (6.0/1.0)

REGIMEN = CP

| HORAS_OPERACION <= 3: A (6.0)

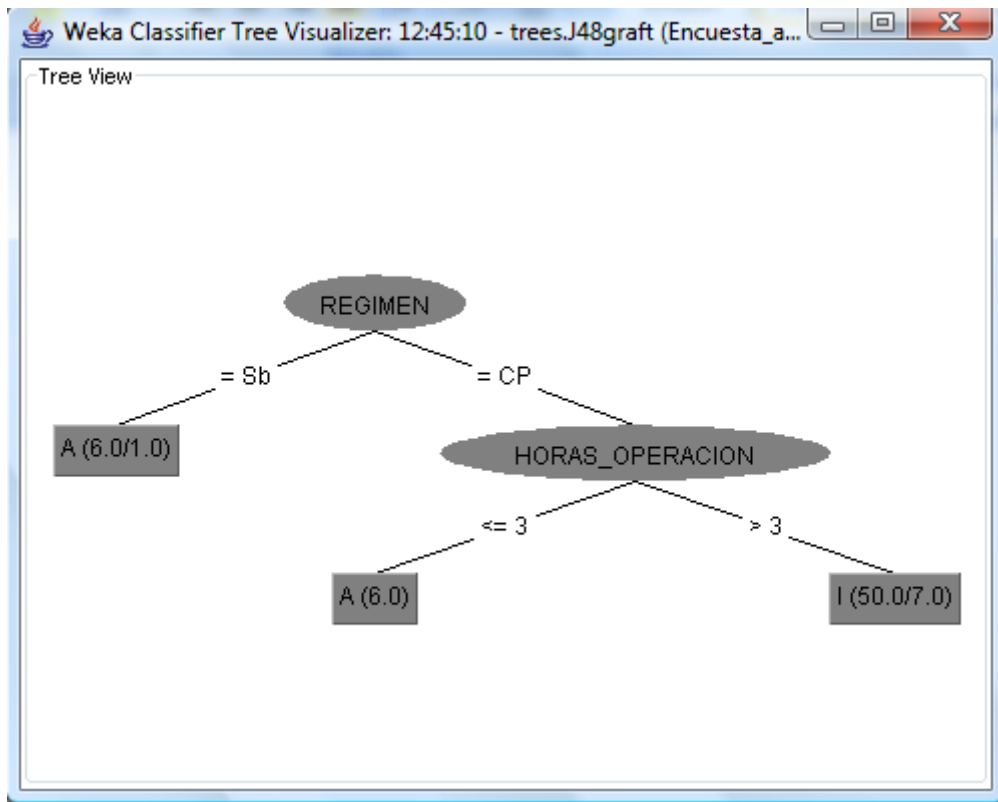
| HORAS_OPERACION > 3: I (50.0/7.0)

Number of Leaves : 3

Size of the tree : 5

Time taken to build model: 0 seconds

A continuación se muestra el modelo de árbol generado en formato texto, cuyo equivalente gráfico sería:



El modelo indica que en primer término se discriminaría por el Régimen, y en caso de "Contratista Principal" se discriminaría también por "Horas de Operación".

Por último se indica que en este caso el árbol tiene un tamaño de 5 nodos (incluyendo el raíz) de los que 3 son hojas. Gráficamente el árbol sería:

En tercer y último grupo muestra información de la fase de validación:

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances

52

83.871 %

Incorrectly Classified Instances	10	16.129 %
Kappa statistic	0.5811	
K&B Relative Info Score	2451.756 %	
K&B Information Score	21.3964 bits	0.3451 bits/instance
Class complexity order 0	54.0845 bits	0.8723 bits/instance
Class complexity scheme	39.9845 bits	0.6449 bits/instance
Complexity improvement (Sf)	14.1 bits	0.2274 bits/instance
Mean absolute error	0.2635	
Root mean squared error	0.3712	
Relative absolute error	63.3575 %	
Root relative squared error	81.5916 %	
Total Number of Instances	62	

=== Detailed Accuracy By Class ===

Class	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area
	0.611	0.068	0.786	0.611	0.688	A
	0.932	0.389	0.854	0.932	0.891	I
Weighted Avg.	0.839	0.296	0.834	0.839	0.832	0.772

=== Confusion Matrix ===

a b <-- classified as

11 7 | a = A

41 | b = I

El primer subgrupo "summary" es un resumen, que incluye dos líneas con el resumen de los registros correcta e incorrectamente clasificados (número y porcentaje) durante la Validación-Cruzada; el estadístico Kappa, que mide

la precisión en la predicción de la clase respuesta; y otros indicadores de esta fase.

En siguiente subgrupo *"Detailed Accuracy By Class"* muestra en las dos primeras columnas el *TP Rate* (porcentaje de aciertos positivos) y el *FP Rate* (porcentaje de falsos positivos).

Las tres columnas siguientes son variables relativas al sistema de recuperación de información, *Recall* es el ratio de registros importantes, y *Precision* es la proporción frente al total. Por último *F-measure* es la combinación de ambos según la fórmula:

$$\frac{2 * recall * precision}{recall + precision}$$

El último subgrupo sería la matriz de confusión, que detalla los registros clasificados correcta e incorrectamente para cada clase.

En esta ejecución inicial la tasa de error es del 80.64% y lo que vamos a hacer es variar el parámetro "Factor de confianza para la poda" para buscar un modelo con mejor tasa de error. El resto de parámetros no nos parece relevante o adecuado su variación.

El valor por defecto de este parámetro es 0.25, si los reducimos haremos una poda más agresiva, lo que debería reducir el tamaño del árbol resultante, y si lo ampliamos haríamos una poda menos agresiva, con lo que obtendríamos un árbol de mayor tamaño. A continuación ejecutaremos el algoritmo para los valores 0.1 y 0.5 del parámetro "Factor de confianza para la poda".

3.4.4.2. Factor de confianza para la poda = 0.1

=== Run information ===

Scheme: *weka.classifiers.trees.J48graft -C 0.1 -M 2 -A*

Relation: *Encuesta_ac_cab-weka.filters.unsupervised.attribute.Reorder-R2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,1*

Instances: *62*

Attributes: *18*

HORA

DIA

MES

HORAS_OPERACION
EDAD
NACIONALIDAD
MESES_ANTIGUEDAD
TIPO_CONTRATO
TIEMPO_EN_OBRA
PUESTO_TRABAJO
FORMACION
REC_PREVIO_PELIGRO
FACTORES_PERSONALES
REGIMEN
EVALUACION_RIESGO
CONDICIONES_ADECUADAS
DIRECCION_SUPERVISION
SUCESO

Test mode: 10-fold cross-validation

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

J48graft pruned tree

REGIMEN = Sb: A (6.0/1.0)

REGIMEN = CP

| HORAS_OPERACION <= 3: A (6.0)

| HORAS_OPERACION > 3: I (50.0/7.0)

Number of Leaves : 3

Árboles de clasificación y regresión v1.2.docx

Size of the tree : 5

Time taken to build model: 0 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	52	83.871 %
Incorrectly Classified Instances	10	16.129 %
Kappa statistic	0.5811	
K&B Relative Info Score	2451.756 %	
K&B Information Score	21.3964 bits	0.3451 bits/instance
Class complexity order 0	54.0845 bits	0.8723 bits/instance
Class complexity scheme	39.9845 bits	0.6449 bits/instance
Complexity improvement (Sf)	14.1 bits	0.2274 bits/instance
Mean absolute error	0.2635	
Root mean squared error	0.3712	
Relative absolute error	63.3575 %	
Root relative squared error	81.5916 %	
Total Number of Instances	62	

=== Detailed Accuracy By Class ===

Class	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area
	0.611	0.068	0.786	0.611	0.688	A

	0.932	0.389	0.854	0.932	0.891	0.772	1
Weighted Avg.	0.839	0.296	0.834	0.839	0.832	0.772	

=== Confusion Matrix ===

a b <-- classified as

11 7 | a = A

3 41 | b = I

En este caso el árbol es el mismo, aunque con una tasa de acierto mejor 83.87%.

3.4.4.3. Factor de confianza para la poda = 0.5

=== Run information ===

Scheme: weka.classifiers.trees.J48graft -C 0.5 -M 2 -A

Relation: Encuesta_ac_cab-weka.filters.unsupervised.attribute.Reorder-R2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,1

Instances: 62

Attributes: 18

HORA

DIA

MES

HORAS_OPERACION

EDAD

NACIONALIDAD

MESES_ANTIGUEDAD

TIPO_CONTRATO

TIEMPO_EN_OBRA

PUESTO_TRABAJO

FORMACION

REC_PREVIO_PELIGRO

FACTORES_PERSONALES

REGIMEN

EVALUACION_RIESGO

CONDICIONES_ADECUADAS

DIRECCION_SUPERVISION

SUCESO

Test mode: 10-fold cross-validation

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

J48graft pruned tree

REGIMEN = Sb: A (6.0/1.0)

REGIMEN = CP

/ HORAS_OPERACION <= 3: A (6.0)

/ HORAS_OPERACION > 3

/ / MESES_ANTIGUEDAD <= 1.67: I (40.0/2.0)

/ / MESES_ANTIGUEDAD > 1.67

/ / / DIA = M: I (3.0)

/ / / DIA = V: A (2.0/1.0)

/ / / DIA = L

/ / / / *EDAD* <= 31.5: *I* (0.0/11.0)

/ / / / *EDAD* > 31.5: *A* (1.0)

/ / / *DIA* = *J*: *A* (2.0)

/ / / *DIA* = *S*: *A* (1.0)

/ / / *DIA* = *Mx*: *I* (1.0)

Number of Leaves : 10

Size of the tree : 15

Time taken to build model: 0.02 seconds

=== *Stratified cross-validation* ===

=== *Summary* ===

<i>Correctly Classified Instances</i>	51	82.2581 %
---------------------------------------	----	-----------

<i>Incorrectly Classified Instances</i>	11	17.7419 %
---	----	-----------

<i>Kappa statistic</i>	0.5623
------------------------	--------

<i>K&B Relative Info Score</i>	2447.9339 %
------------------------------------	-------------

<i>K&B Information Score</i>	21.363 bits	0.3446 bits/instance
----------------------------------	-------------	----------------------

<i>Class complexity order 0</i>	54.0845 bits	0.8723 bits/instance
-----------------------------------	--------------	----------------------

<i>Class complexity scheme</i>	42.5138 bits	0.6857 bits/instance
----------------------------------	--------------	----------------------

<i>Complexity improvement (Sf)</i>	11.5707 bits	0.1866 bits/instance
------------------------------------	--------------	----------------------

<i>Mean absolute error</i>	0.2573
----------------------------	--------

<i>Root mean squared error</i>	<i>0.3814</i>
<i>Relative absolute error</i>	<i>61.8656 %</i>
<i>Root relative squared error</i>	<i>83.8298 %</i>
<i>Total Number of Instances</i>	<i>62</i>

=== Detailed Accuracy By Class ===

<i>TP Rate</i>	<i>FP Rate</i>	<i>Precision</i>	<i>Recall</i>	<i>F-Measure</i>	<i>ROC Area</i>	<i>Class</i>
<i>0.667</i>	<i>0.114</i>	<i>0.706</i>	<i>0.667</i>	<i>0.686</i>	<i>0.764</i>	<i>A</i>
<i>0.886</i>	<i>0.333</i>	<i>0.867</i>	<i>0.886</i>	<i>0.876</i>	<i>0.764</i>	<i>I</i>
<i>Weighted Avg.</i>	<i>0.823</i>	<i>0.27</i>	<i>0.82</i>	<i>0.823</i>	<i>0.821</i>	<i>0.764</i>

=== Confusion Matrix ===

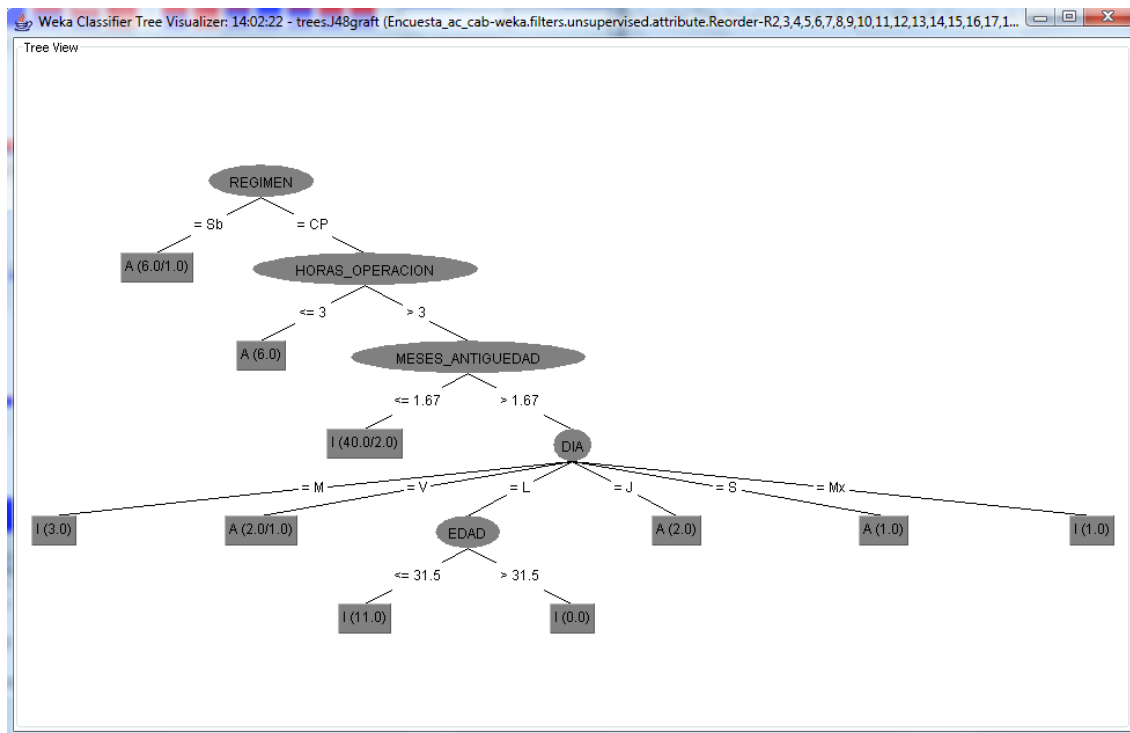
a b <-- classified as

12 6 / a = A

5 39 / b = I

Este árbol tiene una tasa de acierto sensiblemente menor que el anterior y superior al primero, 82.25%.

La poda en este árbol es mucho menos agresiva, con lo que tenemos un árbol de mayor tamaño, 15 nodos de los que 10 son hojas, que gráficamente tendría la siguiente representación:



Nos parece interesante comentar la interpretación de los elementos adicionales que aporta este modelo, ya que a partir de la discriminación por “Régimen” y “Horas de operación” que ya hacían los modelos anteriores, en este modelo se añade una discriminación por “Meses de Antigüedad” para el caso de “más de 3” horas de operación en régimen de Contratista Principal; y para el caso de más de 1.67 meses de antigüedad, aparece como discriminante el “día de la semana”, y en concreto para el día “lunes” sería discriminante la “Edad”.

Por tanto en este modelo miraríamos el Regimen, en caso de Contratista Principal miraríamos las Horas de Operación, en caso de más de 3 miraríamos los Meses de Antigüedad, en caso de más de 1.67 miraríamos el día de la semana, y por último si es Lunes miraríamos la Edad.

3.4.5. Conclusiones

El modelo con factor de confianza para la poda 0.1 nos muestra un modelo con tasa de acierto suficientemente alta de 83.71%, que deja como variables predictoras “Régimen” y “Horas de Operación”, y que dada su simplicidad y tasa de acierto resulta de gran eficiencia.

Sin embargo nos ha parecido también muy interesante para una empresa de seguros de accidente laborales el modelo con una poda menos agresiva, con tasa de confianza para la poda de 0.5, pues aunque tiene una tasa de acierto sensiblemente menor, 82.25% y un tamaño mayor, las variables predictoras adicionales arrojan una información que puede ser de gran interés, pues la conclusión podría ser: “en la población de los Contratistas Principales que ya han pasado un periodo de laboral inicial de 1.67 meses

en la empresa, a partir de las 3 primeras horas de trabajo de los Lunes es significativa su edad en la predicción de Accidente”.

4. CART: Classification and Regression Trees

4.1. Introducción

CART (Classification & Regression Trees) es un método no-paramétrico, de segmentación binaria, desarrollado por Breiman, Freidman, Olshen, Stone en su publicación "Classification and Regression Trees" (1984). El resultado de esta técnica es un árbol de decisión en cuyas divisiones los datos son partidos en dos grupos mutuamente excluyentes. El nodo inicial es llamado nodo raíz o grupo madre y se divide en dos grupos hijos. Luego el procedimiento de partición es aplicado a cada grupo hijo por separado. Las divisiones se seleccionan de modo que la "impureza" de los hijos sea menor que la del grupo madre, formando así grupos cada vez más homogéneos respecto a la variable que se desea discriminar. Las particiones se hacen en forma recursiva hasta que se alcanza un criterio de parada. El objetivo es particionar la respuesta en grupos homogéneos y a la vez mantener el árbol razonablemente pequeño. Este árbol se usa luego para clasificar nuevos datos.

Algunas de sus aplicaciones prácticas son:

- Diagnóstico médico
- Análisis de riesgo en crédito
- Clasificador de objetos para manipulador de robot (Tan 1993)
- Spam

El análisis de árboles de clasificación y regresión (CART) generalmente consiste en tres pasos (Timofeev, 2004):

- Construcción del árbol máximo.
- Poda del árbol máximo
- Selección del árbol óptimo mediante un procedimiento de validación cruzada (*cross-validation*).

4.2. Construcción del árbol máximo

El árbol máximo es construido utilizando un procedimiento de partición binario, comenzando en la raíz del árbol. Este árbol es un modelo que describe el conjunto de entrenamiento (grupo de datos original) y generalmente es sobreajustado, es decir, contiene gran cantidad de niveles y nodos que no producen una mejor clasificación y puede ser demasiado complejo.

En función de la naturaleza de nuestra variable respuesta construiremos un árbol de clasificación (respuesta categórica) o un árbol de regresión (respuesta numérica).

4.2.1. Árboles de clasificación

El algoritmo de CART determina las variables y valores de partición de los datos en función de una medida de la impureza de los nodos. Esta función de impureza será denotada por $i(t)$. Existen varias medidas de impureza (criterios de particionamiento) que nos permiten analizar varios tipos de respuesta, las dos medidas más comunes presentadas por Breiman et al. (1984), para árboles de clasificación son:

1. El índice Gini, que tiene la forma

$$i(t) = \sum_{k \neq l} p(k | t)p(l | t) = \sum_{k=1}^K p(k | t)(1 - p(k | t))$$

siendo,

$p(k | t), p(l | t)$ la probabilidad condicionada de las clases k y l en el nodo t

y

$1, 2, \dots, K$, el índice de clases.

El algoritmo buscará encontrar la partición que maximice $\Delta i(t)$ en

$$\Delta i(t) = - \sum_{k=1}^K [p(k | t_p)]^2 + P_L \sum_{k=1}^K [p(k | t_L)]^2 + P_R \sum_{k=1}^K [p(k | t_R)]^2$$

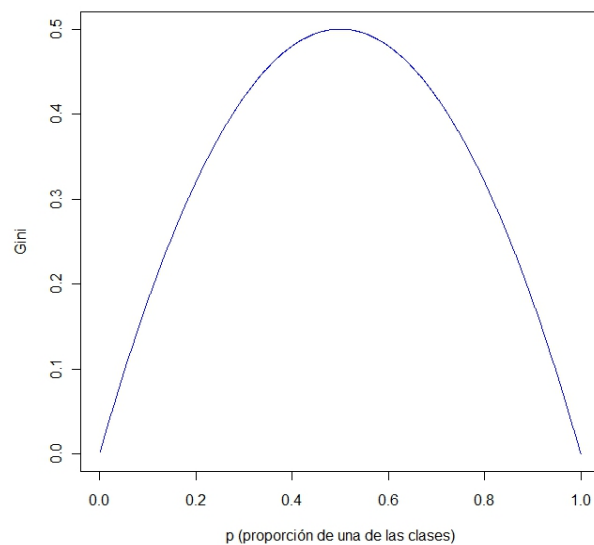
siendo,

t_p , el nodo padre o raíz,

t_L, t_R , los nodos hijos izquierdo y derecho, y

P_L, P_R , la probabilidad de caer en uno de los nodos hijos

Este índice es el más utilizado. En cada división, el índice Gini tiende a separar la categoría más grande en un grupo aparte, mientras que el índice de información tiende a formar grupos con más de una categoría en las primeras decisiones. Como puede verse en la gráfica, para un árbol de dos clases, cuando una de las clases es mayoritaria, el índice Gini es bajo, mientras que sus valores crecen a medida que la proporción de las clases se iguala.



2. El índice "Towing". A diferencia del índice Gini, Towing busca las dos clases que juntas formen más del 50% de los datos, esto define dos "super categorías" en cada división para las cuales la impureza es definida por el índice Gini. Aunque el índice towing produce árboles más balanceados, este algoritmo trabaja más lento que la regla de Gini (Deconinck et al., 2006). Para usar el índice towing se selecciona la partición s que maximiza

$$\frac{p_L p_R}{4} \left[\sum_j |p(j | t_L) - p(j | t_R)| \right]^2$$

donde t_L y t_R representan los nodos hijos izquierdo y derecho respectivamente, p_L y p_R representan la proporción de observaciones en t_p que pasaron a t_L y t_R en cada caso.

4.2.2. Árboles de regresión

Los árboles de regresión no tienen clases. En su lugar, tenemos un vector Y que representa los valores respuesta para cada observación de las variables predictoras. Al no tener clases, no se pueden aplicar los criterios *Gini* o *Towing*. En su lugar se usa el algoritmo de mínimos cuadrados, que escoge la variable y el punto de partición que minimiza la suma de los cuadrados de los residuos de los nodos resultantes. Es decir, para cada variable j y punto de partición s , se pueden definir dos regiones

$$R_1(j, s) = \{X | X_j \leq s\}$$

y

$$R_2(j, s) = \{X \mid X_j > s\}.$$

Entonces, el algoritmo buscará la variable j y punto de partición s que resuelva

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

Y para cada par (j,s) las minimizaciones dentro de los corchetes se resuelven mediante la media de la variable respuesta dentro de cada partición:

$$\hat{c}_1 = \text{ave}(y_i \mid x_i \in R_1(j, s))$$

$$\hat{c}_2 = \text{ave}(y_i \mid x_i \in R_2(j, s))$$

4.2.3. Variables predictoras categóricas

Cuando dividimos una variable predictora categórica que tiene q posibles valores no ordenados, hay $2^q - 1$ posibles particiones de los valores de q en dos grupos, y los cálculos resultan prohibitivos cuando q es grande. Sin embargo, cuando las clases de la variable resultado son de tipo 0-1, estos cálculos se simplifican. Si ordenamos las clases predictoras según la proporción de observaciones que caen en la clase objetivo 1. Y así podemos partir la variable predictora categórica como si fuese ordenada. Esta técnica da lugar a la partición óptimo en términos de índice Gini o mínimos cuadrados.

4.2.4. Tratamiento de valores predictores ausentes

Los datos ausentes son una de las maldiciones del análisis y modelización estadística. La mayoría de las técnicas existentes simplemente pasan por alto las entradas de datos incompletas, haciendo que la cantidad de información disponible se reduzca considerablemente. Una alternativa sería completar los valores ausentes con la media de los valores presentes de ese predictor, pero para los modelos basados en árboles hay dos aproximaciones mejores.

La primera y más general es aplicable a predictores categóricos. Se crea una nueva categoría denominada "ausente" y se trata como tal. Esto podría llevarnos a descubrir un comportamiento distinto en las observaciones con datos ausentes respecto a las no ausentes.

La segunda aproximación más general es la construcción de variables sustitutas (surrogate variables). Cuando buscamos un predictor para una partición consideramos sólo las observaciones para las cuales el predictor está presente. Una vez escogido el mejor predictor (variable primaria) y punto de partición, formamos una lista con todos los posibles predictores sustitutos y puntos de partición. El primer predictor sustituto es aquel que consigue con más fiabilidad la partición de los datos conseguida por la variable primaria. El segundo predictor sustituto es el que consigue la segundo mejor partición de los datos, y así sucesivamente. Cuando se envían observaciones desde el nodo raíz hasta los nodos terminales a través del árbol, tanto en la fase de entrenamiento como durante la predicción, se usan las variables sustitutas en orden.

Las variables sustitutas explotan la correlación entre las distintas variables de un conjunto de datos. A mayor correlación entre el variable a la que pertenece el dato ausente y las otras variables predictoras, menor será la pérdida de información debida a la ausencia del dato.

4.2.5. La matriz de pérdidas (loss matrix)

En algunos árboles de clasificación las consecuencias de una clasificación errónea son más serias en unas clases que en otras. Por ejemplo, es peor predecir que una persona no tendrá un ataque al corazón cuando sí lo va a tener que el caso contrario. Para tener esto en cuenta, se puede definir una matriz de pérdidas L de dimensiones $K \times K$:

$$L = \begin{pmatrix} L_{11} & L_{12} & \cdot & \cdot & \cdot & L_{1K} \\ L_{21} & L_{22} & & & & L_{2K} \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ L_{K1} & L_{K2} & \cdot & \cdot & \cdot & L_{KK} \end{pmatrix}$$

siendo L_{xy} la pérdida producida por clasificar una clase x como clase y . La pérdida por clasificaciones correctas debe ser 0, es decir, $L_{xx} = 0, \forall x$:

$$L = \begin{pmatrix} 0 & L_{12} & \cdot & \cdot & \cdot & L_{1K} \\ L_{21} & 0 & & & & L_{2K} \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ L_{K1} & L_{K2} & \cdot & \cdot & \cdot & 0 \end{pmatrix}$$

Para incorporar las pérdidas al modelo podríamos modificar el índice Gini tal que:

$$i(t) = \sum_{k \neq l} L_{kl} p(k | t) p(l | t)$$

4.3. Poda del árbol

¿Qué tamaño debería tener el árbol? Es obvio que un árbol demasiado grande podría sobreajustar los datos, mientras que un árbol demasiado pequeño podría no reflejar la estructura subyacente de los datos con fiabilidad. A día de hoy parece claro que un algoritmo que incluya un regla de parada no es fiable y pasa por alto rasgos importantes de la estructura de los datos.

La mejor estrategia es construir un árbol grande, deteniendo el crecimiento sólo cuando se alcanza un tamaño de nodo mínimo, por ejemplo 5 observaciones por nodo. En este punto el árbol, que generalmente es sobreajustado, es acortado mediante un proceso denominado "poda por coste - complejidad" (cost-complexity pruning).

Una forma, introducida por Breiman et al. (1984), es, partiendo del árbol máximo, buscar una serie de árboles anidados de tamaños decrecientes (De'ath & Fabricius, 2000), cada uno de los cuales es el mejor de todos los árboles de su tamaño.

Estos árboles pequeños son comparados para determinar el óptimo. Esta comparación esta basada en una función de coste - complejidad, $R_\alpha(T)$.

Para cada árbol T , la función coste - complejidad se define como (Deconinck et al., 2006):

$$R_\alpha(T) = R(T) + \alpha | \tilde{T} |$$

donde $R(T)$ es el promedio de la suma de cuadrados entre los nodos, puede ser la tasa de mala clasificación total o la suma de cuadrados de residuales total dependiendo del tipo de árbol, $|\tilde{T}|$ es la complejidad del árbol, definida como el número total de nodos del sub-árbol y α es el parámetro de complejidad.

El parámetro α es un número real mayor o igual a cero. Cuando $\alpha = 0$ se tiene el árbol más grande y a medida que α se incrementa, se reduce el tamaño del árbol. La función $R_\alpha(T)$ siempre será minimizado por el árbol más grande, por tanto se necesitan mejores estimaciones del error, para esto Breiman et al. (1984) proponen obtener estimadores "honestos" del error por "validación cruzada".

Computacionalmente el procedimiento es exigente pero viable, pues solo es necesario considerar un árbol de cada tamaño, es decir, los árboles de la secuencia anidada.

4.4. Selección del árbol óptimo

De la secuencia de árboles anidados es necesario seleccionar el árbol óptimo y para esto no es efectivo utilizar comparación o penalización de la complejidad (De'ath & Fabricius, 2000), por tanto se requiere estimar con precisión el error de predicción y en general esta estimación se hace utilizando un procedimiento de validación cruzada.

El objetivo es encontrar la proporción óptima entre la tasa de mala clasificación y la complejidad del árbol, siendo la tasa de mala clasificación el cociente entre las observaciones mal clasificadas y el número total de observaciones.

El procedimiento de validación cruzada puede implementarse de dos formas:

Si se cuenta con suficientes datos se parte la muestra, sacando la mitad o menos de los datos y se construye la secuencia de árboles utilizando los datos que permanecen y luego predecir, para cada árbol, la respuesta de los datos que se sacaron al iniciar el proceso; obtener el error de las predicciones y seleccionar el árbol con el menor error de predicción.

Pero en general no se cuenta con suficientes datos como para utilizar el procedimiento anterior, de modo que otra forma alternativa será:

Validación cruzada con partición en V (v-fold cross validation).

La idea básica de la "validación cruzada" es sacar de la muestra de aprendizaje una muestra de prueba, con los datos de la muestra de aprendizaje se calculan los estimadores y el subconjunto sacado es usado para verificar el desempeño de los estimadores obtenidos utilizándolos como "datos nuevos". El desempeño entendido como el error de predicción, es acumulado para obtener el error medio absoluto del conjunto de prueba.

Para la metodología CART generalmente se utiliza la validación cruzada con partición en v (v-fold cross validation), tomando $v = 10$ y el procedimiento es el siguiente:

1. Dividir la muestra en diez grupos mutuamente excluyentes y de aproximadamente igual tamaño.

2. Sacar un conjunto de cada vez y construir el árbol con los datos de los grupos restantes.
3. Se utiliza el árbol para predecir la respuesta del conjunto eliminado.
4. Calcular el error estimado para cada subconjunto.
5. Repetir los puntos dos y tres para cada tamaño de árbol.
6. Seleccionar el árbol con la menor tasa de mala clasificación.
7. Al llegar a este punto se procede a analizar el árbol obtenido.

4.5. Ventajas y desventajas de CART

Antes de tomar la decisión de aplicar CART al análisis de datos reales, es importante identificar las posibles ventajas e inconvenientes y comparar éste con otros métodos de clasificación estadística.

4.5.1. Ventajas de CART

- Acepta variables predictoras continuas y categóricas.
- Los resultados son fáciles de entender e interpretar.
- No tiene problema en trabajar con datos ausentes.
- CART es no-paramétrico. Es decir, no necesita que previamente se especifique un modelo estadístico para los datos.
- CART no necesita que las variables se seleccionen de antemano. El algoritmo identificará por sí mismo las variables más significativas y eliminará las no significativas.
Para comprobar esta propiedad, se puede incluir una variable con datos generados aleatoriamente y construir un nuevo árbol con estos datos y usando los mismos parámetros. Ambos árboles deberían ser idénticos.
- Los resultados de CART son invariantes frente a transformaciones monótonas de sus variables predictoras.
Cambiar una o más variables por su logaritmo o raíz cuadrada no cambia la estructura del árbol. Sólo se modificarán los valores sobre los que se particionan los datos en los diferentes nodos.
- CART es robusto frente a datos atípicos.
Los datos atípicos pueden afectar negativamente a los resultados de muchas técnicas de modelización estadística, como PCA o regresión lineal. Sin embargo CART los maneja con facilidad, aislándolos en nodos independientes. Esta propiedad es muy importante, ya que los datos financieros, a menudo modelizados mediante árboles de clasificación, tienen datos atípicos con mucha frecuencia.
- CART no tiene hipótesis de partida. Muchas otras técnicas son difícilmente aplicables a la vida real, al asumir hipótesis demasiado estrictas o complejas.
- Rápido. Pruebas realizadas con la aplicación *CART* de Salford System's obtuvieron los siguientes resultados: un modelo basado en

300.000 entradas y 1.000 variables puede ser generado en menos de una hora. Si los datos consisten en 100.000 registros y 450 variables, el árbol se genera en aproximadamente 10 minutos, mientras que 100 variables y un millón de registros generarían un árbol en menos de 30 minutos.

- CART es flexible y tiene la habilidad de ajustarse rápidamente a nuevos datos.
A menudo la muestra de entrenamiento es continuamente modificada con nuevas observaciones. Por ejemplo, muchos bancos usan el sistema de *credit scoring Basel II* para clasificar diferentes compañías en niveles de riesgo. Este sistema usa un grupo de coeficientes e indicadores que requieren continuas correcciones para ajustarse a los cambios del mercado.
- Particiones binarias. Otros tipos de árboles consideran particiones múltiples en lugar de binarias en cada nivel. Aunque esto podría ser útil en casos particulares, no es una buena estrategia general. El problema de las particiones múltiples es que fragmentan los datos demasiado pronto, dejando a menudo datos insuficientes en el siguiente nivel. Además, las particiones múltiples pueden conseguirse mediante una serie de particiones binarias consecutivas, por tanto éstas son, en general, preferibles.
- Tiene en cuenta las interacciones que puedan existir entre las distintas variables predictoras.
- Poda. Algunas técnicas de construcción de árboles de decisión utilizan reglas de parada para detener el crecimiento del árbol en lugar de la estrategia de sobreajuste-poda utilizada por CART. Esta última es preferible, pues las reglas de parada se arriesgan a detener el crecimiento demasiado pronto, pasando por alto información relevante.

4.5.2. Desventajas de CART

- Puede generar árboles inestables.
Uno de los mayores problemas con los árboles es su elevada varianza. Modificaciones insignificantes de la muestra de entrenamiento, como eliminar ciertas observaciones, podría generar cambios radicales en el árbol de decisión, tales como aumento o disminución de la complejidad o cambios en la selección de variables y puntos de partición. La principal razón para esta inestabilidad es la naturaleza jerárquica del proceso. El efecto de un error en una partición se propaga hacia abajo, a todos los nodos hijos por debajo de ella. Un criterio de partición más estable podría aliviar esto hasta cierto punto, pero la inestabilidad inherente no es eliminada.
- CART se divide sólo en función de una variable. Esto hace que la superficie de predicción no sea muy suave, ya que es un conjunto de

planos. En otras palabras, todas las particiones de los datos son perpendiculares a los ejes.

- Ausencia de una función global de las variables y como consecuencia pérdida de la representación geométrica.
- Requiere un gran número de datos para asegurarse que el número de observaciones en los nodos terminales sea significativo.
- Dificultad para elegir el árbol óptimo.
- El proceso de selección de variables es sesgado hacia las variables con más valores diferentes.

4.6. Algunas herramientas que implementan CART

4.6.1. Rpart (recursive partitioning)

El lenguaje R dispone de este paquete desarrollado originalmente por Terry M. Therneau y Beth Atkinson en base al libro Classification en Regression Trees de Breiman, Freidman, Olshen, Stone. En su versión original coincidía casi de manera absoluta con la aplicación comercial CART de Salford Systems. La única diferencia reseñable estaba en el tratamiento de los surrogates o predictores sustitutos. La consecuencia es que rpart favorece a las variables con menos datos ausentes.

Durante los últimos años rpart no ha evolucionado demasiado, de ahí que los gráficos y la visualización hayan quedado algo anticuados.

4.6.2. CART

Herramienta gráfica desarrollada y comercializada por Salford Systems, empresa propietaria del algoritmo CART original. También basada en el libro Classification en Regression Trees de Breiman, Friedman, Olshen, Stone y en un código Fortran elaborado por Friedman para contrastar las ideas contenidas en el libro.

Contiene utilidades para el análisis estadístico y la minería de datos orientada a la inferencia de árboles de decisión para tareas de clasificación o regresión.

4.6.3. Weka

Weka es una colección a algoritmos para tareas de data mining. Entre sus herramientas cuenta con árboles de decisión entre los que se encuentra

[weka.classifiers.trees.SimpleCart.](#)

Es muy similar a los algoritmos implementados en CART o *rpart*. El tratamiento de los datos ausentes, sin embargo, se basa en el método denominado "observaciones fraccionadas" (fractional instances) y no en los "surrogate splits".

4.7. Aplicación de CART a los datos de prueba

El análisis de árboles de clasificación y regresión (CART) comienza siempre con la construcción del árbol máximo para luego, mediante un proceso de poda, seleccionar el árbol óptimo en función de la tasa de error del modelo.

El primer paso, tras cargar los datos, es decidir el método de partición, que dependerá de la naturaleza de la variable respuesta. La función *rpart* admite, en su argumento *method*, cuatro posibilidades:

- `method="anova"`: la variable respuesta es continua
- `method="class"`: la variable respuesta categórica
- `method="poisson"`: variable respuesta con distribución de Poisson (2 columnas)
- `method="exp"`: variable respuesta de tipo "supervivencia"

En este caso la variable respuesta es categórica y por tanto el método apropiado es `"class"`. Las primeras líneas de código para efectuar el análisis serán:

```
# Cargar datos
library(rpart)
setwd("C:/Users/Propietario/Desktop/ARBOLES")
datos=read.table("Encuesta Accidentes.csv",header=T,dec=",")

# Se elimina la variable "Encuesta", que no es predictora
datos=datos[-1]
```

Más adelante se analizará la supresión de otras variables, para las cuales quizá no exista suficiente representación para cada tipo de suceso.

```
# Construcción del árbol máximo, mediante minsplitt=2
arbol=rpart
(
  S~.,
  data=datos,
  method="class",
  parms = list(split = "gini"),
  maxsurrogate = 0,
  minsplitt=2,
  xval=10
)
```

)

El argumento `split="gini"` establece el índice GINI como criterio de división de los nodos. `Maxsurrogate` determina el método de actuación frente a valores ausentes dentro de una variable. En este caso los datos ausentes de la variable *Antigüedad en meses* han sido previamente sustituidos por 0.1 y la variable *Plazo de ejecución* tiene demasiados valores ausentes para tenerla en cuenta en la construcción del árbol, así pues, en la práctica, no habrá datos ausentes, y el valor `Maxsurrogate=0` será el adecuado. Por último `minsplit` determina el número mínimo de ejemplos que debe tener un nodo para intentar una nueva división. Al establecer `minsplit=2` se genera el árbol máximo, con nodos terminales de hasta una sola observación.

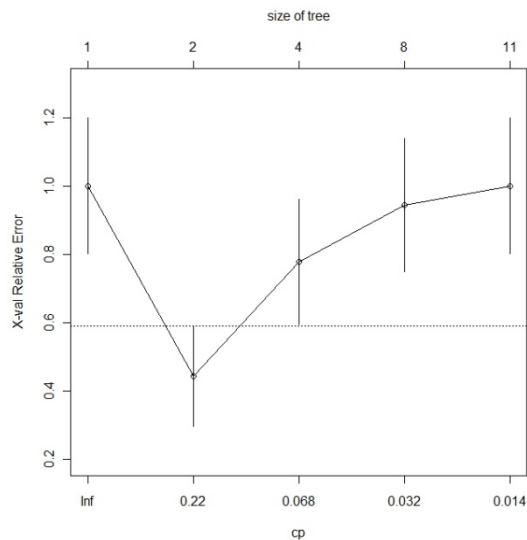
```
# Reglas que determinan el árbol
arbol

n= 62

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

1) root 62 18 I (0.29032258 0.70967742)
  2) CA=Ast,CasLe,CVal 14 2 A (0.85714286 0.14285714)
    4) PT=PMG,PV,SD 11 0 A (1.00000000 0.00000000) *
    5) PT=OP,PSM 3 1 I (0.33333333 0.66666667)
      10) H=PH 1 0 A (1.00000000 0.00000000) *
      11) H=DU 2 0 I (0.00000000 1.00000000) *
  3) CA=Ar 48 6 I (0.12500000 0.87500000)
    6) An>=1.795 9 4 I (0.44444444 0.55555556)
      12) D=J,L,S,V 5 1 A (0.80000000 0.20000000)
        24) PT=OP,PMG 4 0 A (1.00000000 0.00000000) *
        25) PT=PV 1 0 I (0.00000000 1.00000000) *
      13) D=M,Mx 4 0 I (0.00000000 1.00000000) *
  7) An< 1.795 39 2 I (0.05128205 0.94871795)
    14) H=HE 1 0 A (1.00000000 0.00000000) *
    15) H=DC,DU,Ot,PH 38 1 I (0.02631579 0.97368421)
      30) Ed>=38 6 1 I (0.16666667 0.83333333)
        60) H=DU 2 1 A (0.50000000 0.50000000)
          120) M=N 1 0 A (1.00000000 0.00000000) *
          121) M=S 1 0 I (0.00000000 1.00000000) *
        61) H=DC 4 0 I (0.00000000 1.00000000) *
      31) Ed< 38 32 0 I (0.00000000 1.00000000) *
```

Muchos de los nodos terminales están compuestos por pocas o incluso una sola observación, por lo que más adelante deberá realizarse un proceso de poda para evitar un sobreajuste de los datos. Este sobreajuste quedará de

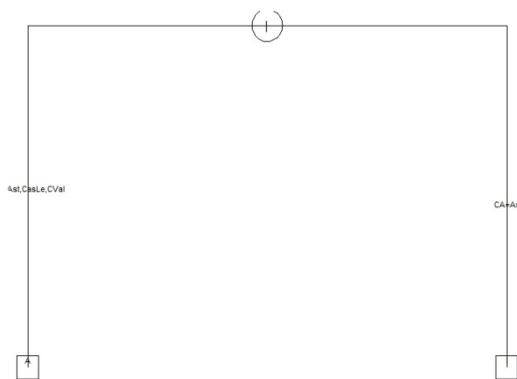


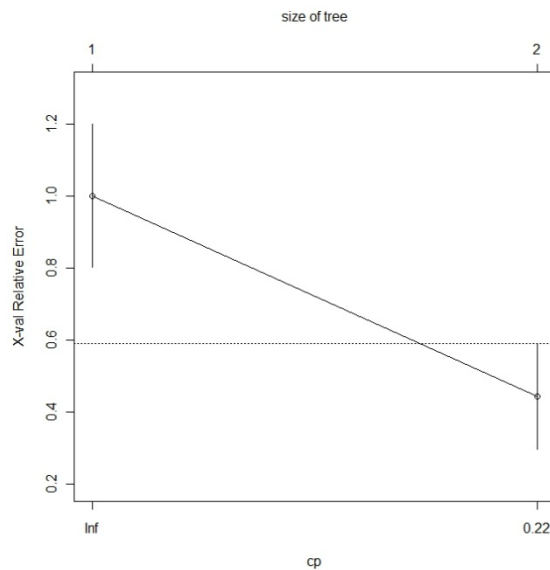
El árbol óptimo será aquel cuyo error en la validación cruzada (**xerror** en la *cptable*) sea mínimo. En este caso será el árbol con una sola partición, que se ve a continuación.

```
# Árbol podado, plot y tasa de error
arbol_p=prune (arbol, cp=arbol$cptable [which.min
(arbol$cptable[ , "xerror"]),"CP"])

windows(); plot(arbol_p,uniform=T)
text(arbol_p,pretty=0,fancy=T,cex=0.7)

windows(); plotcp(arbol_p)
```

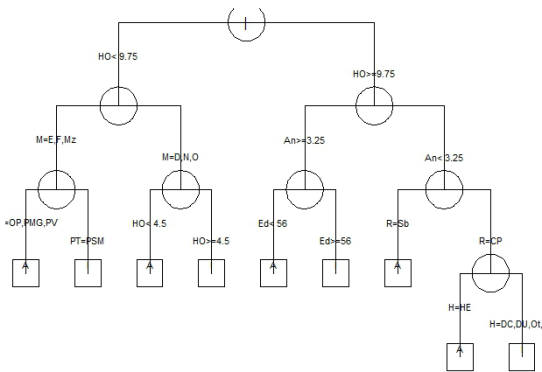
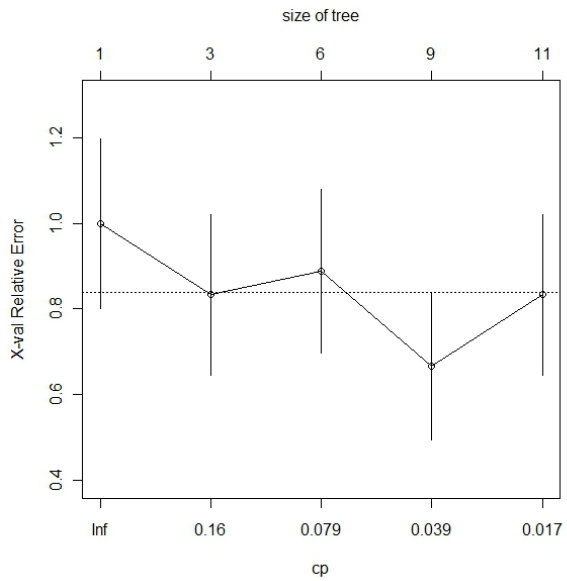




El árbol final consiste en una sola partición que da lugar a dos hojas. Esta partición divide los casos en función de la variable *Comunidad Autónoma*, incluyendo en una hoja los casos ocurridos en *Aragón* y en la otra hoja el resto de sucesos. Pero esta variable, como hemos visto en el apartado de preprocesado de los datos, tiene poca representación en aquellos casos que no son *Aragón*, por lo tanto podemos probar a eliminarla y ver el nuevo árbol que se genera sin ella.

Si repetimos todo el proceso anterior con la variable *CA* eliminada, la *complexity parameter table*, la gráfica de error para la validación cruzada y el árbol definitivo, ya podado, serán los siguientes:

	CP	nsplit	rel error	xerror	xstd
1	0.22222222	0	1.00000000	1.00000000	0.1985611
2	0.11111111	2	0.55555556	0.83333333	0.1873381
3	0.05555556	5	0.22222222	0.88888889	0.1914126
4	0.02777778	8	0.05555556	0.66666667	0.1728253
5	0.01000000	10	0.00000000	0.83333333	0.1873381



La tasa de error de validación cruzada ha aumentado de 0.444 a 0.667 al prescindir de la variable “Comunidad Autónoma”. Para comprobar el número de instancias correctamente clasificadas del nuevo modelo utilizaremos la siguiente función:

```

cci<-function(arbol_f)
{
  z=xpred.rpart(arbol_f,cp=arbol_f$cptable[which.min(arbol_f$cptable[, "xerror"]), "CP"])
  y=datos$S
  w=0
  for (a in 1:length(z))
  {
    if (z[a]=="2" & y[a]=="I" | z[a]=="1" & y[a]=="A")
      {w=w+1}
  }
  return(w/length(z)*100)
}

```

```
    }  
cci(arbol)
```

```
[1] 77.41935
```

Esta función utiliza `xpred.rpart()` para predecir los resultados de la validación cruzada y luego los compara con los resultados reales de los datos, dando un porcentaje de aciertos. Este dato es equivalente al [Correctly Classified Instances](#) de Weka, y por tanto lo usaremos más adelante para realizar una comparativa de resultados.

4.8. Conclusión

El árbol obtenido muestra un porcentaje de aciertos del 77.41935% y unos resultados que parecen bastante coherentes. Datos como que se produzcan más accidentes durante las horas extras o que los conductores de vehículos y maquinaria pesada sufran más accidentes que el personal de oficina parecen bastante intuitivos. Además, dado el reducido tamaño de nuestra base de datos, un árbol con particiones binarias y univariantes como el que produce `rpart` parece lo más adecuado.

5. NBTree

5.1. Introducción

NBTree es un algoritmo híbrido entre un árbol de clasificación y el clasificador Naive Bayes. Ya se ha explicado con anterioridad los árboles de clasificación (capítulo 2.5), simplemente resumiremos aquí algunas de sus características destacando sus ventajas. Describiremos los clasificadores Naive Bayes para poder comprender como funciona el algoritmo NBTree.

5.2. Árboles de clasificación

Un árbol de clasificación está formado por nodos, ramas y hojas. Cada nodo representa un test univariado o decisión sobre los valores de un atributo concreto.

El primer nodo del árbol es conocido como el nodo raíz. Finalmente están los nodos terminales u hojas en los que se toma una decisión acerca de la clase a asignar. Así, a la hora de clasificar un nuevo caso, tendrán que compararse los valores de los atributos con las decisiones o tests que se toman en los nodos, siguiendo la rama que coincida con dichos valores en cada test. Finalmente se llega a un nodo terminal u hoja que predice la clase para el caso tratado. Un árbol de decisión también se puede ver como un conjunto de reglas si-entonces, si bien la diferencia más obvia entre los dos paradigmas es que las reglas de decisión son independientes entre sí, mientras que las reglas extraídas del árbol de decisión no lo son.

Los árboles de clasificación son uno de los métodos de aprendizaje inductivo supervisado no paramétrico más utilizado. Como forma de representación del conocimiento, los árboles de clasificación destacan por su sencillez.

Ventajas de los árboles de clasificación:

- Son rápidos
- Segmentan los datos

Inconvenientes:

- Fragmentación al aumentar el número de particiones
- La interpretabilidad empeora al aumentar el número de particiones

5.3. Clasificadores Naive-Bayes

Es un método probabilístico para clasificación. Este método se basa en una aplicación del teorema de Bayes, pero con unas restricciones y suposiciones de partida.

Naive-Bayes se basa en un modelo de independencia condicional de los atributos predictores dada la clase, a pesar de lo cual garantiza una clasificación óptima si se cumplen un conjunto de suposiciones explícitas.

En la siguiente figura podemos observar una representación gráfica de este modelo:

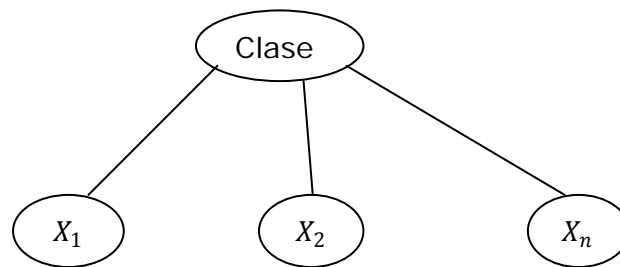


Fig. El clasificador Naive-Bayes

En el momento de clasificar una nueva instancia, se basa en el teorema de Bayes para calcular la probabilidad a posteriori con la que la instancia puede pertenecer a cada una de las clases del problema,

$$P(c_i | X_1 = x_1, \dots, X_n = x_n) = \frac{P(c_i)P(X_1 = x_1, \dots, X_n = x_n | c_i)}{P(X_1 = x_1, \dots, X_n = x_n)}$$

donde $P(c_i)$ es la probabilidad a priori de la clase a en el conjunto de entrenamiento.

De todas formas, sabiendo que la instancia I es una conjunción de los n valores de sus atributos descriptivos $X_1 = x_1, \dots, X_n = x_n$ y que $P(I)$ es igual para todas las clases, se puede restringir la regla anteriormente expuesta, quedando de la siguiente forma,

$$P(C = c_i | X_1 = x_1, \dots, X_n = x_n) \propto P(C = c_i)P(X_1 = x_1, \dots, X_n = x_n | C = c_i)$$

donde $P(X_1 = x_1, \dots, X_n = x_n | C = c_i)$ supone la probabilidad de que ocurra la instancia I cuando la clase es c_i . Estas probabilidades se pueden estimar mediante las frecuencias del conjunto de entrenamiento. Sin embargo, esta expresión no es operacional, ya que el término $P(X_1 = x_1, \dots, X_n = x_n | C = c_i)$ será

habitualmente cero, debido al número de parámetros tan elevado del orden de 2^n para variables predictoras dicotómicas- que se debe estimar para cada valor de la variable C . Para hacer operativa la expresión anterior se debe tener en cuenta la suposición de que los atributos que definen cada instancia son independientes entre sí dada la clase del problema. De esta forma se puede utilizar la siguiente expresión,

$$P(X_1 = x_1, \dots, X_n = x_n | C = c_i) = \prod_{k=1}^n P(X_k = x_k | C = c_i)$$

donde las probabilidades a priori y condicionales del valor de cada atributo X_k dada la clase c_i se estiman a través de las frecuencias del conjunto de entrenamiento. Utilizando la última igualdad se calculará la probabilidad a posteriori de cada clase dada la instancia, clasificándose la misma con la clase que computa la mayor de estas probabilidades.

Ventajas de los clasificadores Naive Bayes:

- Son rápidos
- Son fáciles de interpretar
- Son robustos para atributos irrelevantes
- Utilizan evidencia de muchos atributos, una propiedad muy útil en casos donde no hay un efecto principal

Inconvenientes:

- Asume independencia de los atributos
- Baja efectividad en grandes bases de datos

5.4. Algoritmo NBTree

El algoritmo NBTree es un algoritmo híbrido entre los árboles de clasificación y el clasificador Naive-Bayes. Se puede definir NBTree como un árbol de clasificación cuyas hojas son clasificadores Naive-Bayes. Cada hoja del NBTree contiene un clasificador Naive-Bayes local que no considera las variables que se encuentran involucradas en los test univariados que están en el camino que lleva hasta la hoja.

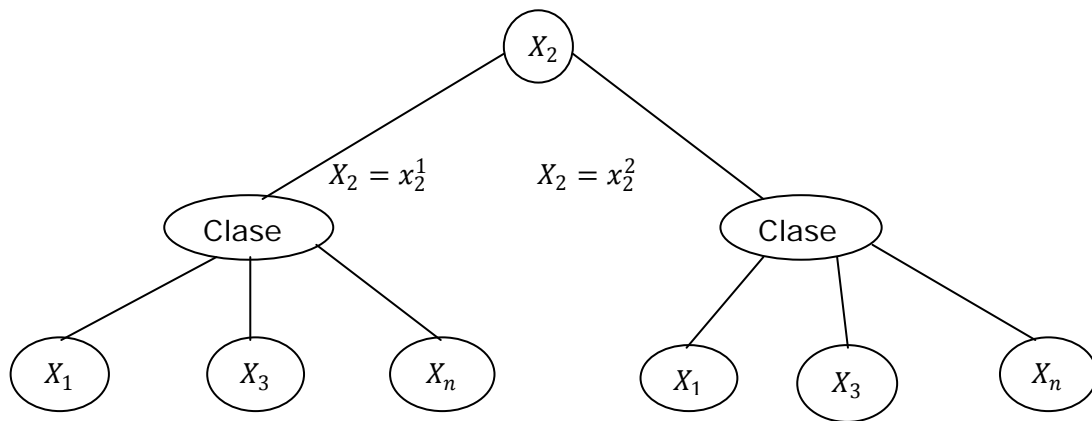


Fig: Ejemplo de estructura NBTREE con un nodo de decisión en el atributo X_2 y dos clasificadores Naive Bayes como hojas.

Propiedades de los árboles NBTREE:

- i) Cada nodo interno representa una variable
- ii) Cada nodo interno tiene tantas ramas salientes como estados tiene la variable representada en dicho nodo
- iii) Todas las hojas están al mismo nivel
- iv) En cualquier camino que se recorra desde la raíz hasta las hojas no existen variables repetidas.

La condición iii) se impone para simplificar el modelo, pero puede no ser tenida en cuenta en la práctica.

Este algoritmo intenta utilizar las ventajas de los árboles de clasificación (segmentación) y de los clasificadores Naive-Bayes (acumulación de la evidencia de múltiples atributos).

Este clasificador es tan fácil de interpretar como los árboles de clasificación y los clasificadores NB. El árbol de decisión segmenta los datos (una tarea que es fundamental en el proceso de minería de datos con conjuntos de datos de gran tamaño). Cada partición de los datos, representada por una hoja, es descrita por medio de un clasificador NB.

NBTREE es un clasificador híbrido nativo, es decir, no está construido a través de un método de ensamble.

5.4.1 El algoritmo híbrido

El algoritmo es similar a los clásicos esquemas de partición recursiva, excepto que las hojas creadas son categorizadores Naive-Bayes en lugar de nodos que predicen la pertenencia a una clase.

El umbral para atributos continuos se elige utilizando la técnica de minimización de la entropía, del mismo modo que en los árboles de decisión.

La utilidad de un nodo se calcula discretizando los datos y calculando la exactitud de aplicar validación cruzada con 5 grupos utilizando Naive-Bayes en el nodo.

La utilidad de una partición es la suma ponderada de la utilidad de los nodos, donde el peso dado a cada nodo es proporcional al número de instancias que caen en ese nodo.

Intuitivamente, se trata de aproximar si la exactitud para un clasificador Naive-Bayes en cada hoja es mayor que un sólo clasificador Naive-Bayes en el nodo actual. Para evitar particiones con escaso valor, se define una partición como significativa si la reducción relativa del error es mayor del 5% y hay al menos 30 instancias en el nodo.

El uso directo de la validación cruzada para seleccionar atributos no se ha utilizado de manera habitual debido al gran coste que conlleva usarlo de manera general. Sin embargo, si los datos son discretizados, Naive-Bayes puede ser validado de este modo en un tiempo que es lineal con el número de instancias, el número de atributos y el número de valores etiquetados. La razón es que se pueden quitar las instancias, actualizar los contadores, clasificarlos y repetir para un diferente conjunto de instancias.

Dadas m instancias, n atributos y l valores etiquetados, la complejidad de la fase de selección del atributo para atributos discretos es $O(m \cdot n^2 \cdot l)$. Si el número de atributos es menor que $O(\log m)$, lo que ocurre habitualmente, y el número de etiquetas es pequeño, entonces el tiempo empleado en seleccionar el atributo mediante validación cruzada es menor que el tiempo empleado clasificando las instancias por cada atributo.

Algoritmo NBTree

Entrada: un conjunto T de ejemplos etiquetados

Salida: un árbol de decisión con clasificadores Naive.Bayes en las hojas

1. Para cada atributo X_i , evalúa la utilidad, $u(X_i)$, de una partición en el atributo X_i . Para atributos continuos, se encuentra un umbral en esta etapa
2. Sea $j = \arg \max_i (u_i)$, i.e., el atributo con la máxima utilidad
3. Si u_j no es significativamente mejor que la utilidad del nodo actual, crea un clasificador Naive-Bayes para ese nodo
4. Particiona T según el test sobre X_j . Si X_j es continuo, se usa una partición umbral; si X_j es discreto, se realiza una partición múltiple para todos los valores posibles
5. Para cada hijo, se llama al algoritmo recursivamente en la parte de T que coincide con el test que conduce a la hoja

5.5. Ventajas NBTree

Bajo ciertas condiciones los NBTree son buenos clasificadores:

- Muchos atributos son relevantes para la clasificación
- Los atributos no son necesariamente independientes
- La base de datos es grande
- La interpretabilidad del clasificador es importante, ya que tanto la segmentación como los clasificadores Naive-Bayes de las hojas son fáciles de entender

5.6. Aplicación de NBTree a los datos de prueba

Empleamos el algoritmo NBTree de Weka con los datos de prueba.

Cargamos los datos y aplicamos el algoritmo NBTree. No presenta opciones este algoritmo, tan solo debug (True or False), que depura la información que se muestra.

Seleccionamos la variable clase, en nuestro caso S (suceso) y como test aplicamos validación cruzada con 10 grupos

```

=== Run information ===
Scheme: weka.classifiers.trees.NBTree
Relation: Encuesta Accidentes
Instances: 62
Attributes: 20
E
.
DS
Test mode: 10-fold cross-validation
=== Classifier model (full training set) ===
NBTree
-----
TC = OS
| R = Sb: NB 2
| R = CP
|| H = DU: NB 4
|| H = DC: NB 5
|| H = PH: NB 6
|| H = HE: NB 7
|| H = Ot: NB 8
TC = In: NB 9
TC = T: NB 10

```

Obtenemos un árbol con 8 hojas, con sus correspondientes clasificadores Naive Bayes. A continuación describe cada uno de estos clasificadores, indicando la probabilidad de que el suceso sea Accidente o Incidente si el ejemplo cae en esa hoja.

Leaf number: 2 Naive Bayes Classifier

Class

Attribute	A (0.86)	I (0.14)
-----------	-------------	-------------

=====

E	Ed	PT	ER
[total] 67.0 62.0	[total] 6.0 1.0	[total] 10.0 5.0	[total] 7.0 2.0
H	Na	F	CT
[total] 10.0 5.0	[total] 9.0 4.0	[total] 8.0 3.0	[total] 7.0 2.0
D	An	RP	DS
[total] 11.0 6.0	[total] 6.0 1.0	[total] 7.0 2.0	[total] 8.0 3.0
M	TC	FP	
[total] 12.0 7.0	[total] 8.0 3.0	[total] 7.0 2.0	
HO	TO	R	
[total] 6.0 1.0	[total] 8.0 3.0	[total] 7.0 2.0	

Leaf number: 4 Naive Bayes Classifier

Class

Attribute	A (0.22)	I (0.78)
=====		
E		Ed
[total] 63.0 68.0		[total] 2.0 7.0
H		Na
[total] 6.0 11.0		[total] 5.0 10.0
D		An
[total] 7.0 12.0		[total] 2.0 7.0
M		TC
[total] 8.0 13.0		[total] 4.0 9.0
HO		TO
[total] 2.0 7.0		[total] 4.0 9.0

PT
[total] 6.0 11.0
F
[total] 4.0 9.0
RP
[total] 3.0 8.0
FP
[total] 3.0 8.0
CA
[total] 5.0 10.0

R
[total] 3.0 8.0
ER
[total] 3.0 8.0
CT
[total] 3.0 8.0
DS
[total] 4.0 9.0

Leaf number: 5 Naive Bayes Classifier

Class

Attribute	A (0.21)	I (0.79)
=====		
E		Ed
[total] 64.0 72.0		[total] 3.0 11.0
H		Na
[total] 7.0 15.0		[total] 6.0 14.0
D		An
[total] 8.0 16.0		[total] 3.0 11.0
M		TC
[total] 9.0 17.0		[total] 5.0 13.0
HO		TO
[total] 4.0 12.0		[total] 5.0 13.0

PT
[total] 7.0 15.0
F
[total] 5.0 13.0
RP
[total] 4.0 12.0
FP
[total] 4.0 12.0
CA
[total] 6.0 14.0

R
[total] 4.0 12.0
ER
[total] 4.0 12.0
CT
[total] 4.0 12.0
DS
[total] 5.0 13.0

Leaf number: 6 Naive Bayes Classifier

Class

Attribute	A (0.57)	I (0.43)
=====		
E		Ed
[total] 65.0 64.0		[total] 5.0 4.0
H		Na
[total] 8.0 7.0		[total] 7.0 6.0
D		An
[total] 9.0 8.0		[total] 5.0 4.0
M		TC
[total] 10.0 9.0		[total] 6.0 5.0
HO		TO
[total] 5.0 4.0		[total] 6.0 5.0

PT
[total] 8.0 7.0
F
[total] 6.0 5.0
RP
[total] 5.0 4.0
FP
[total] 5.0 4.0
CA
[total] 7.0 6.0

R
[total] 5.0 4.0
ER
[total] 5.0 4.0
CT
[total] 5.0 4.0
DS
[total] 6.0 5.0

Leaf number: 7 Naive Bayes Classifier

Class

Attribute	A (0.5)	I (0.5)
=====		
E		Ed
[total] 62.0 62.0		[total] 1.0 1.0
H		Na
[total] 5.0 5.0		[total] 4.0 4.0
D		An
[total] 6.0 6.0		[total] 1.0 1.0
M		TC
[total] 7.0 7.0		[total] 3.0 3.0
HO		TO
[total] 1.0 1.0		[total] 3.0 3.0

PT	R
[total] 5.0 5.0	[total] 2.0 2.0
F	ER
[total] 3.0 3.0	[total] 2.0 2.0
RP	CT
[total] 2.0 2.0	[total] 2.0 2.0
FP	DS
[total] 2.0 2.0	[total] 3.0 3.0
CA	
[total] 4.0 4.0	

Leaf number: 8 Naive Bayes Classifier

Class

Attribute	A (0.14)	I (0.86)
=====		
E		Ed
[total] 63.0 73.0		[total] 2.0 12.0
H		Na
[total] 6.0 16.0		[total] 5.0 15.0
D		An
[total] 7.0 17.0		[total] 2.0 12.0
M		TC
[total] 8.0 18.0		[total] 4.0 14.0
HO		TO
[total] 2.0 12.0		[total] 4.0 14.0

PT	R
[total] 6.0 16.0	[total] 3.0 13.0
F	ER
[total] 4.0 14.0	[total] 3.0 13.0
RP	CT
[total] 3.0 13.0	[total] 3.0 13.0
FP	DS
[total] 3.0 13.0	[total] 4.0 14.
CA	
[total] 5.0 15.0	

Leaf number: 9 Naive Bayes Classifier

Class

Attribute	A (0.33)	I (0.67)
=====		
E		Ed
[total] 68.0 75.0		[total] 7.0 14.0
H		Na
[total] 11.0 18.0		[total] 10.0 17.0
D		An
[total] 12.0 19.0		[total] 7.0 14.0
M		TC
[total] 13.0 20.0		[total] 9.0 16.0
HO		TO
[total] 7.0 14.0		[total] 9.0 16.0

PT	R
[total] 11.0 18.0	[total] 8.0 15.0
F	ER
[total] 9.0 16.0	[total] 8.0 15.0
RP	CT
[total] 8.0 15.0	[total] 8.0 15.0
FP	DS
[total] 8.0 15.0	[total] 9.0 16.0
CA	
[total] 10.0 17.0	

Leaf number: 10 Naive Bayes Classifier

Class

Attribute A I
 (0.25) (0.75)

=====

E		Ed		PT		R
[total] 62.0 64.0		[total] 1.0 3.0		[total] 5.0 7.0		[total] 2.0 4.0
H		Na		F		ER
[total] 5.0 7.0		[total] 4.0 6.0		[total] 3.0 5.0		[total] 2.0 4.0
D		An		RP		CT
[total] 6.0 8.0		[total] 1.0 3.0		[total] 2.0 4.0		[total] 2.0 4.0
M		TC		FP		DS
[total] 7.0 9.0		[total] 3.0 5.0		[total] 2.0 4.0		[total] 3.0 5.0
HO		TO		CA		
[total] 1.0 3.0		[total] 3.0 5.0		[total] 4.0 6.0		

Y por último obtenemos un resumen de los resultados del test:

Number of Leaves : 8

Size of the tree : 11

Time taken to build model: 1.22 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances 53 85.4839 %

Incorrectly Classified Instances 9 14.5161 %

Kappa statistic 0.6295

Mean absolute error 0.1819

Root mean squared error 0.3417

Relative absolute error 43.7289 %

Root relative squared error 75.1018 %

Total Number of Instances 62

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure ROC Area Class

0.667 0.068 0.8 0.667 0.727 0.824 A

0.932 0.333 0.872 0.932 0.901 0.824 I

Weighted Avg. 0.855 0.256 0.851 0.855 0.851 0.824

=== Confusion Matrix ===

a b <-- classified as

12 6 | a = A

3 41 | b = I

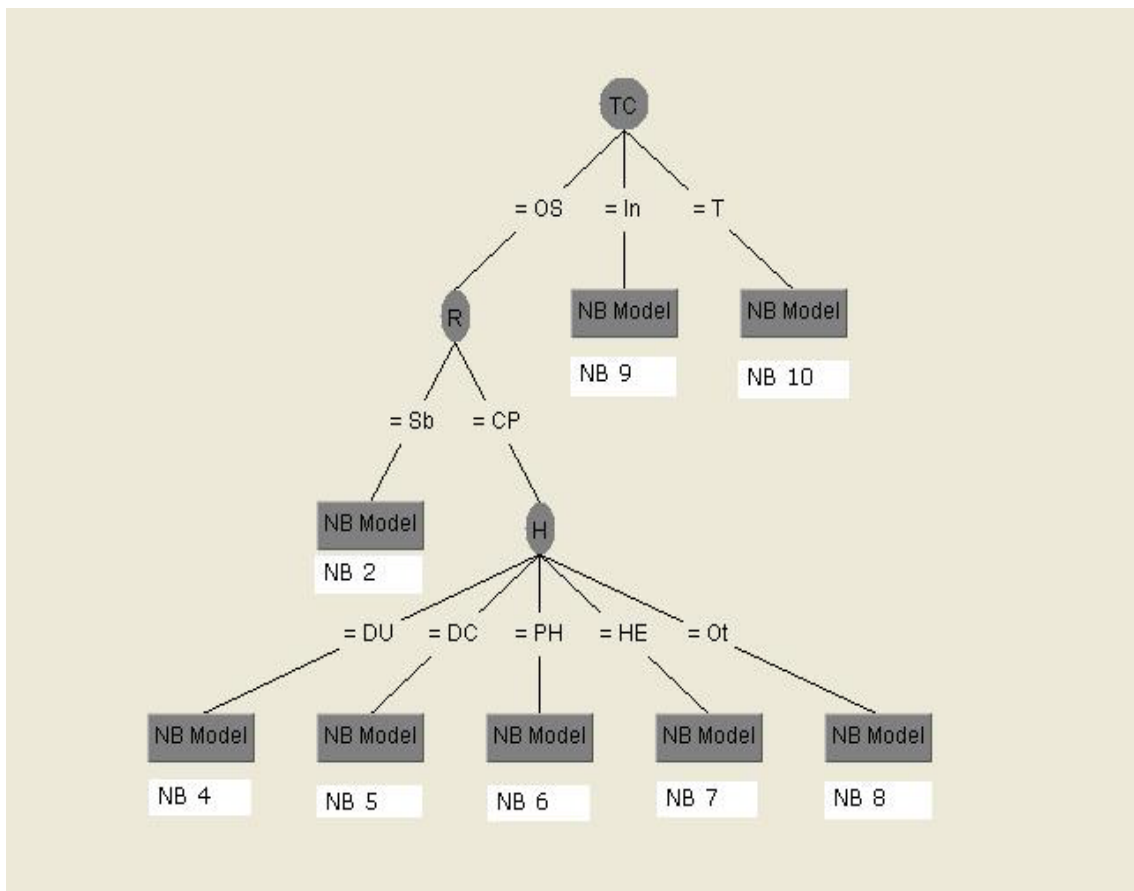


Fig. Árbol NBTree

Empleamos validación cruzada con 10 grupos, ya que es el test que nos proporciona mejores resultados. Obtenemos un mayor porcentaje de instancias correctamente clasificadas y mayor precisión por clase.

Con anterioridad (capítulo 3.4) se ha explicado lo referente al preprocesado de los datos. En este caso, hemos ejecutando el algoritmo eliminando los atributos sugeridos CA, R y DS. En el caso de CA, decidimos mantenerlo en este tipo de árbol ya que los resultado obtenidos al eliminarlo disminuyen el porcentaje de instancias correctamente clasificadas al 69.3548 % y además este algoritmo no utiliza esta variable como uno de los discriminantes principales.

Respecto al atributo R, en el preprocesado se había decidido mantenerlo , y en nuestro caso esta variable constituye uno de los nodos del árbol.

Observamos que en la hoja que corresponde con el NB 7, la probabilidad de pertenencia a cada una de las clases es la misma 0.5, por lo tanto, y dado

que las instancias se clasifican según la clase con mayor probabilidad, surge la duda de que ocurre con las instancias que caen en esa hoja, ya que la pertenencia a una de las clases es aleatoria.

6. Decision Stump

6.1. Introducción

Decision Stump es un modelo de aprendizaje automático consistente en un árbol de decisión de una sola partición. Acepta clases numéricas o categóricas. Las instancias que considera inclasificables las cuelga en nuevas ramas que se unen al nodo raíz. No se usa con mucha frecuencia, ya que muy pocos problemas pueden ser modelizados con precisión usando simplemente una variable. Por tanto, este método cometerá errores importantes.

Sin embargo, algoritmos débiles como éste a menudo son usados como componentes en técnicas más complejas como *Bagging* o *Boosting*. También es frecuente el uso de múltiples particiones "decision stump" binarias dentro de técnicas conjuntas que usan las predicciones de métodos que se comportan de manera ligeramente distinta.

6.2. Aplicación de Decision Stump a los datos de prueba

Si tratamos de modelizar los datos del fichero "Encuesta sobre Accidentes de Empresas Mineras" mediante Decision Stump obtendremos los siguientes resultados:

<pre>==== Run information ==== Scheme: weka.classifiers.trees.DecisionStump Relation: Encuesta Accidentes Weka Instances: 62 Attributes: 20 . . . Test mode: 10-fold cross-validation ==== Classifier model (full training set) ==== Decision Stump Classifications CA = Ar : I CA != Ar : A CA is missing : I Class distributions CA = Ar A I</pre>	<pre>0.125 0.875 CA != Ar A I 0.8571428571428571 0.14285714285714285 CA is missing A I 0.2903225806451613 0.7096774193548387 Time taken to build model: 0 seconds ==== Stratified cross-validation ==== ==== Summary ==== Correctly Classified Instances 54 87.0968 % Incorrectly Classified Instances 8 12.9032 % . . .</pre>
--	--

Como ya se viera anteriormente, la variable "Comunidad Autónoma" discrimina muy bien los datos, obteniéndose un 87.1% de instancias correctamente clasificadas en la validación cruzada. Pero si, en función de lo visto en el apartado de preprocesado de datos (3.4), decidiésemos eliminar esta variable por falta de representación, los resultados serían los siguientes:

<pre> ==== Run information ==== Scheme: weka.classifiers.trees.DecisionStump Relation: Encuesta Accidentes Weka weka.filters.unsupervised.attribute.Remove- R16 Instances: 62 Attributes: 18 . . . Test mode: 10-fold cross-validation ==== Classifier model (full training set) ==== Decision Stump Classifications HO <= 9.75 : A HO > 9.75 : I HO is missing : I Class distributions HO <= 9.75 A I </pre>	<pre> 0.6111111111111112 0.3888888888888889 HO > 9.75 A I 0.1590909090909091 0.8409090909090909 HO is missing A I 0.2903225806451613 0.7096774193548387 Time taken to build model: 0 seconds ==== Stratified cross-validation ==== ==== Summary ==== Correctly Classified Instances 46 74.1935 % Incorrectly Classified Instances 16 25.8065 % . . . </pre>
--	--

El porcentaje de casos bien clasificados cae hasta el 74.2%

6.3. Conclusión

El porcentaje de aciertos, 74.1935%, es bastante alto, probablemente debido a la escasa cantidad de variables y ejemplos que componen nuestros datos. Además hay que tener en cuenta que variables como CA o HO discriminan los datos con mucha precisión. Sin embargo, es de esperar que para conjuntos de datos más complejos que éste, el algoritmo Decision Stump, usado aisladamente, muestre una escasa efectividad.

7. Comparativa y conclusiones

Una vez obtenidos los resultados de los distintos algoritmos, el criterio más lógico de selección parece ser el porcentaje de casos correctamente clasificados en una validación cruzada, cuyos valores, junto con el tamaño del árbol (número de nodos más número de hojas) son los siguientes:

ALGORITMO	PORCENTAJE DE ACIERTO	TAMAÑO
J48	83.71	5
J48	82.25	15
CART	77.41	17
NBTree	85.48	11
Decision Stump	74.19	3

Teniendo en cuenta estos dos criterios el árbol que obtiene mejores resultados sería el construido con el algoritmo NBTree, ya que es el que proporciona un mayor porcentaje de aciertos y un tamaño reducido, lo cual le confiere una mayor interpretabilidad.

Otro de los criterios a menudo utilizados para comparar la efectividad de un árbol de clasificación es el tiempo de ejecución. En nuestro caso no lo tenemos en cuenta, debido al reducido tamaño de la base de datos, ya que las diferencias no serán significativas en ningún caso.

También puede ser en ocasiones de interés generar modelos menos eficientes en el sentido de menor capacidad de predicción y/o mayor tamaño, como es el segundo caso de la ejecución del algoritmo J48, pero que, como ya se explicó en el apartado correspondiente, el modelo generado pueda ofrecer información de interés para la toma de decisiones.

Como trabajo adicional para intentar mejorar estos resultados se puede intentar una selección de variables predictoras más eficaz, eliminando, a criterio del investigador, aquella o aquellas variables que parezcan menos eficaces y observando los nuevos resultados.

8. Bibliografía

Breiman, L. (2001a). Random forest. *Machine Learning* 45, 5-32.

Breiman, L. (2001b). Statistical modeling: The two cultures. *Statistical Science* 16, 199-231.

Hastie, T., Tibshirani, R., Friedman, J., 2001. *The Elements of Statistical Learning. Data Mining, Inference, and Prediction*

Hernández, J., Ramírez, M. J., Ferri, C., 2005. Introducción a la Minería de Datos.

<http://analisisydecision.es/sobre-la-historia-de-cart-y-rpart/>

<http://cran.r-project.org/web/packages/rpart/rpart.pdf>

<http://math.uprm.edu/~edgar/trees1.pdf>

<http://redalyc.uaemex.mx/redalyc/pdf/402/40211215.pdf>

<http://www.java2s.com/Open-Source/Java-Document/Science/weka/weka.classifiers.trees.htm>

Hernández Orallo, J., Ramírez Quintana, M. J., Ferri Ramírez, C., 2004. *Introducción a la Minería de Datos*.

Morrison, Darin. 2007.

<http://web.cecs.pdx.edu/~mm/MachineLearningSpring2007/DarinMorrison.pdf>

Portugal, Roberto and Carrasco, M.: Ensamble de Algoritmos Bayesianos con Árboles de decisión: una alternativa de clasificación. XVII Congreso Chileno de Control Automático – ACCA'07. 2007.

Robles Forcada, Luis :Clasificación Supervisada basada en Reyes Bayesianas. Aplicación en Biología Computacional. Tesis Doctoral. 2003.

Ron Kohavi: Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid. In: Second International Conference on Knowledge Discovery and Data Mining, 202-207. 1996.

Serna Pineda, S. C., 2009. *Comparación de Árboles de Regresión y Clasificación y regresión logística*.

Timofeev, R., 2004. *Classification and Regression Trees (CART). Theory and*

Árboles de clasificación y regresión v1.2.docx

Applications

Yongheng Zhao and Yanxia Zhang. Comparison of decision tree methods for finding active objects: 36th COSPAR Scientific Assembly. 2006.